Athénée Royal Verdi
Un regard sur la vie

3ème degré

Notes de cours
SCIENCES INFORMATIQUE

K. PINAN BARRANCO
2022-2023

Table des matières

Module 0 : Fondamentaux de l'informatique

Petite histoire de l'informatique

A l'échelle de l'humanité, l'ordinateur et le réseau mondial qu'est internet sont deux révolutions encore jeunes pour l'Homme...
"Informatik"

En janvier 2018, 81% des français possédaient un ordinateur, 88% avaient un accès à Internet et 54% de la population mondiale était connectée.
Mais comment ces petites bestioles (les ordinateurs) qui parfois nous énervent parce qu'elles ne "font pas toujours ce que l'on veut" sont arrivées dans nos vies, aussi bien dans le travail que dans notre vie personnelle ?

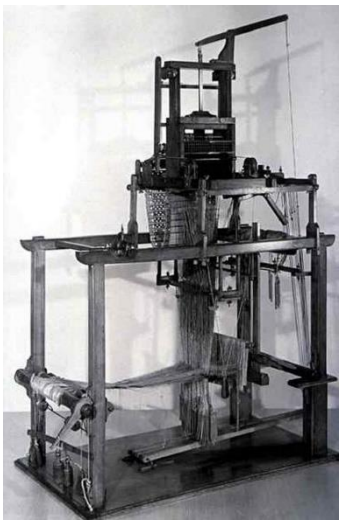
Les origines

Avant de pouvoir définir ne serait-ce qu'une période ou une date exacte sur les origines, il faut d'abord définir le mot « informatique ». En effet, on associe souvent ce terme avec une idée de gestion automatisée, de programmation, de systèmes d'exploitation et donc d'ordinateur. Le mot est utilisé dès 1957 en Allemagne pour parler d'un traitement automatique de l'information : « *informatik* » est la parole alors utilisée pour désigner ces fonctions. Pourtant, de nos jours, c'est un terme qui inclut aussi bien le domaine industriel lié à l'ordinateur que la science du traitement de l'information.

De l'invention de la table de multiplication de Pythagore au VIe siècle avant J.C., jusqu'aux ordinateurs bientôt réellement quantiques (Le D-Wave utilisé et perfectionné par Google et la NASA), l'Homme n'a eu de cesse de chercher à perfectionner sa façon de calculer en inventant toujours des outils de calcul de plus en plus poussés : *La Pascaline* (1642) de **Blaise Pascal** (1623-1662) est une première ébauche qui se rapproche des appareils automatisés et programmables. Néanmoins, afin d'avoir réellement un outil qui met en œuvre un enchaînement d'opérations basiques, il faut attendre le XIXe siècle. En effet, les théories de programmation naissent avec l'invention du *métier à tisser* de **Jospeh Marie Jacquard** en 1801.

En 1834, un calculateur mécanique programmable, basé sur un système de cartes perforées reliées à un organe de commande, *La machine analytique* de **Charles Babbage** (1791-1871), étoffe ces théories. Ce dernier s'est inspiré de la machine mécanisée sélectionnant les fils de chaîne à l'aide d'un programme inscrit sur des cartes perforées (que l'on doit à **Basile Bouchon**)...

Les débuts de l'informatique



Au début du XXème siècle nait **Kurt Gödel** (1906-1978), Autrichien naturalisé Américain qui était un éminent logicien et mathématicien. Ce dernier est connu pour son théorème d'incomplétude "affirmant que n'importe quel système logique suffisamment puissant pour décrire l'arithmétique des entiers admet des propositions sur les nombres entiers ne pouvant être ni infirmées ni confirmées à partir des axiomes de la théorie". [4]
Pourtant, en 1936, **Alan Turing** infirme cette théorie en présentant une expérience de pensée nommée "Machine de Turing" et ce afin de résoudre le problème fondamental de la décidabilité en arithmétique. Cette "Machine" est considéré comme le premier ordinateur à ce jour.

L'aire des "mastodontes"

Avec les avancées considérables de l'électronique, de gros ordinateurs voient le jour et sont de plus en plus perfectionnés. Le premier, en 1946 créé par l'**ENIAC** (Electronic Numerical Integrator and Computer) par **P. Eckert** et J. **Mauchly**, est composé de 19000 tubes, pèse 30 tonnes, occupe une surface de 72m² et consomme 140 kilowatts. Nous sommes donc loin de nos ordinateurs portables qui ne pèsent que 3 kg environ... C'est aussi le cas entre 1949 et 1951 avec le **Whirlwind** crée au **MIT** par **Jay Forrester**, **Ken Olsen** et leur équipe. Cet ordinateur fut le prototype des ordinateurs utilisés pour le réseau informatique de défense américain **SAGE**, ou encore en 1955, lorsque **IBM** lance l'IBM 704, machine considérée comme le premier super ordinateur qui est dédié au calcul scientifique et où sera développé le langage **FORTRAN** (premier langage de programmation universel - FORmula TRANslator).
La course à l'invention révèle de nombreux projets plus innovants les uns que les autres, jusqu'aux inventions d'**ARPANET** (1969 - Ancêtre d'Internet), en novembre 1971, la mise en vente du premier microprocesseur par **Marcian Hoff** de **Intel** ou encore, toujours chez le même constructeur, l'invention du **disque dur** de type **Winchester**, la carte à puce par le journaliste Roland Moreno ... cette progression d'ingéniosité en tout genre va réduire le format des ordinateurs et faire en sorte qu'ils soient accessibles au grand public. Dans un article de presse au sujet du Micral, le mot microcomputer (micro-ordinateur) apparaît pour la première fois au Etats Unis en 1973.

La micro-informatique

1977 est une date importante dans l'histoire de l'informatique... En effet, c'est cette année que Steeve Jobs décide de quitter son garage pour installer sa société **Apple Computer** dans des bureaux en Californie. C'est aussi en 1977 que l'entreprise **Commodore Business Machines Inc.** présente son ordinateur PET, qu'Apple Computer présente son ordinateur Apple, que **Texas Instruments** présente une nouvelle gamme de calculatrices de poche programmables et qu'**Atari** présente sa console de jeux Video Computer System (plus connue sous le nom de Atari 2600). La course à la miniature est lancée. Les machines vont être de plus en plus perfectionnées, abordables pour le grand public et vouées à perfectionner le travail de l'être humain. Par exemple, en 1983 **Microsoft** sort la version 1.0 du logiciel **Word** qui fonctionne sous **MS/DOS**. Toujours en 83, est inventé le langage **C++** dérivé du langage **C**. Ce langage est une extension orientée objet créée par **Bjarn Stroustrup**.

Les années 80 sont une période de l'histoire de l'informatique où les ordinateurs vont être démocratisés, notamment grâce au perfectionnement des résolutions d'écrans et à l'amélioration croissante des cartes graphiques qui vont permettre à une population de plus en plus large de profiter de l'informatique à la maison. La sortie sur le marché d'ordinateurs comme l'**Atari ST** (le 1040STf / 10000 F en 1986) ou l'**Amiga 1000** de Commodore (18000 F en 1985) remporte un franc succès et le grand public en Europe, en Asie et sur le continent nord-américain commence à faire l'acquisition d'ordinateurs dits de maison.

La suite, nous la connaissons tous, avec la sortie en 1983 d'une interface graphique (GUI) développée par **Microsoft** pour son système d'exploitation MS-DOS, interface graphique qui prendra le nom de **Windows**. Puis cet OS va se décliner en version 1.0x, 2.x, 3.0, etc, jusqu'à arriver à la version de **Windows 95**.
De son côté, la société Apple Computer s'est efforcée depuis 1984 de créer un système d'exploitation qui a toujours été vu comme un concurrent du système d'exploitation Windows. Du système 0.0 : System 0.97 de 1984, en passant par le système de Mac OS 9 en 1999, puis Guépard en 2001 (MAC OS X 10.0) jusqu'à El Capitan en 2015 (MAC OS X 10.11) et mac OS 10.15 (Catalina), la société Apple a toujours été la concurrente directe de Microsoft... ce qui a permis une évolution informatique jusqu'à lors sans précédent.

Aujourd'hui, les ordinateurs se tiennent dans une main (smartphone) et l'évolution ne va de cesse que l'invention est toujours en constante croissance. Nous stockons tous nos données grâce au *cloud computing* (serveurs distants utilisant beaucoup d'énergie) et nous sommes tous connectés 24h/24h. Les technologies comme la Réalité Virtuelle ou la Réalité Augmentée commencent à se démocratiser... pourtant, le numérique et l'informatique, outils utiles à l'Homme, ont aussi un impact environnemental. Nous verrons prochainement quel est cet impact et de quelle façon nous pouvons faire cohabiter numérique et environnement.

Module 1 : Logiciel de bureautique

❖ Histoire

Le mot « Bureautique » a été proposé pour la première fois en 1976 à Grenoble par un groupe de chercheurs dont Louis Naugès était un des animateurs. Le problème était de traduire les mots anglais « office automation ». On avait hésité entre « bureaumatique » qui avait été jugé trop proche de « bureaucratique ». Ce mot regroupait alors tout ce qui concerne l'information en rapport avec le travail de bureau**.**

❖ Logiciel de bureautique

Les logiciels de bureautique les plus connus sont ceux de la suite Microsoft Office. Les plus utilisés sont Excel (tableur), Word (traitement de texte), PowerPoint (pour faire des présentations), et bien d'autres.

Avec ces différents logiciels, vous allez pouvoir rédiger et éditer des courriers administratifs, des courriers professionnels ou personnels, etc. Word vous permet de faire une mise en forme du texte, pour des documents clairs.

A l'aide du tableur, vous allez pouvoir calculer, faire des graphiques, ou tableaux. Bien que l'on pense que ce logiciel est utile uniquement aux professionnels, les particuliers l'utilisent souvent pour faire leurs comptes.

Enfin, le logiciel PowerPoint vous accompagne dans vos présentations. Ce sont souvent les étudiants et les professionnels qui l'utilisent, de façon à présenter des projets. En effet, il est possible d'y inclure des images, graphiques, ou encore vidéos, pour un contenu dynamique.

Module 2 : Gestion pérenne des données

1. La cybersécurité

Introduction

Le mot **cybersécurité** est un néologisme désignant le rôle de l'ensemble des lois, politiques, outils, dispositifs, concepts et mécanismes de sécurité, méthodes de gestion des risques, actions, formations, bonnes pratiques et technologies qui peuvent être utilisés pour protéger les personnes et les actifs informatiques matériels et immatériels (connectés directement ou indirectement à un réseau) des états et des organisations.
Le préfixe cyber a donné avec Internet et la généralisation du numérique un grand nombre de mots tels que cyberespace, cybersécurité, cybercafé, cyberculture, cyberdémocratie, cybermarché, cyber-réputation.
Le terme cybersécurité est construit à partir du préfixe Cyber d'origine grecque, réapparu au milieu du XX^{e} siècle avec cybernétique, ce dernier concernant l'étude des processus de contrôle et de communication chez l'être vivant et la machine.
C'est par réaction contre les risques liés à l'omniprésence des technologies de l'information et de la communication et leur capacité d'interconnexion et d'échange de données que se constitue progressivement la cybersécurité en tant que nouvelle discipline (spécialité) pleine et entière.

Visionnage d'un documentaire[1]

Après la diffusion de ce dernier, qu'ai-je appris ? Désappris ? Quels sont les informations que je souhaite retenir ?

……
……
……
……
……
……
……
……
……
……
……
……

[1] https://www.youtube.com/watch?v=jxVJVvfNav4 (Complément d'enquête - Hackers les nouveaux braqueurs - France 2 - 28/01/2021)

……
……
……
……
……
……
……
……

Que penses-tu du hacking ? Explique et justifie ton choix.
……
……
……
……
……
……
……
……
……

Exercices :

A l'aide d'internet, cite 5 types d'attaques différents qu'un utilisateur ou une société privée peut être victime. Explique brièvement chaque type.
……
……
……
……
……
……
……
……
……
……
……

2. Assurer la pérennité des données

Sauvegarde de données sur plusieurs médias

Sauvegarde (backup)

Afin d'assurer la pérennité des données il est important d'effectuer des sauvegardes (backup) de ses fichiers vers des supports de bonnes qualités et notamment pour les supports virtuels comme Internet. En effet il se peut que les sauvegardes s'effectuent en utilisant des sites Internet. Il est donc indispensable de s'assurer de la valeur de ce stockage dans le temps et respect culturel.

Une bonne structure et logique d'arborescence des dossiers de sauvegarde est aussi importante, afin de retrouver facilement son travail même longtemps après sa création. Pour cela une documentation bien à jour facilite grandement les recherches de fichiers sur le disque du PC, du serveur, des disques amovibles, CDs, DVD …

En informatique une grandeur rigueur de travail est indispensable notamment dans le cadre de sauvegarde de données. Il est aussi fondamental d'effectuer souvent des sauvegardes de données afin de récupérer des versions récentes pour les fichiers de données défectueux ou bien rendus inadaptés par les dernières modifications.

En ce qui concerne les progiciels, les DVD d'installation doivent être rangés en des endroits connus et sûrs, dès lors les logiciels ne sont donc pas concernés par les backups
Unités de stockage : disque dur, serveur, disque amovible, clé USB, Internet (attention !), bandes magnétiques …

Il est judicieux de sauvegarder ses données sur plusieurs médias et en des endroits différents, sécurisés et à l'abri d'incendie éventuel ou de vol.

2.1 Adopter un comportement rationnel face à la présence d'un virus

Un virus informatique est un automate auto réplicatif à la base non malveillant, mais aujourd'hui souvent additionné de code malveillant (donc classifié comme logiciel malveillant), conçu pour se propager à d'autres ordinateurs en s'insérant dans des logiciels légitimes, appelés « hôtes ».

- Il peut perturber plus ou moins gravement le fonctionnement de l'ordinateur infecté.
- Il peut se répandre à travers tout moyen d'échange de données numériques comme les réseaux informatiques et les cédéroms, les clefs USB, etc.

2.2 Logiciel Anti-virus

Il est fondamental d'installer un progiciel anti-virus dès réception du PC et ce avant toute utilisation et en particuliers Internet. Il est aussi judicieux d'effectuer un contrôle anti-virus à chaque utilisation d'un média extérieur (clé USB, disque amovible…)
Ne pas ouvrir les courriels (e-mail) douteux, avec des fichiers attachés dont la source est inconnue ou non vérifiée.

La base de données de l'antivirus doit être mise à jour régulièrement. L'idéal est d'avoir un logiciel qui effectue cette mise à jour automatiquement dès la parution d'une nouvelle version. En effet l'anti-virus est un logiciel qui peut détecter uniquement les virus connus dans sa base de données.

2.3 Installation

En informatique, l'installation est la procédure permettant l'intégration d'un logiciel sur un ordinateur. Cette procédure est plus ou moins complexe selon l'ordinateur, le système d'exploitation et selon qu'il existe un installeur (petit logiciel qui se charge de l'installation) ou pas. Une rigueur procédurale est de toute façon de mise, car de l'installation du logiciel dépendent sa fiabilité et sa performance. Toutes les documentations officielles - guide d'installation - doivent être lues avant installation, car plusieurs critères entrent en ligne de compte dont le système d'exploitation, le processeur 32 bits ou 64 bits… Les processus d'installation peuvent être différents voire incompatibles dans certains (à détecter avant déballage des DVD d'installation pour un échange éventuel)

2.4 Logiciel espion (spyware)

Un logiciel espion (aussi appelé mouchard ou espiogiciel ; en anglais spyware) est un logiciel malveillant qui s'installe dans un ordinateur dans le but de collecter et transférer des informations sur l'environnement dans lequel il s'est installé, très souvent sans que l'utilisateur en ait connaissance. L'essor de ce type de logiciel est associé à celui d'Internet qui lui sert de moyen de transmission de données.
Un logiciel espion est composé de trois mécanismes distincts :

Le mécanisme d'infection, qui installe le logiciel. Ce mécanisme est identique à celui utilisé par les virus, les vers ou les chevaux de Troie. Par exemple, l'espiogiciel Cydoor utilise le logiciel grand public Kazaa comme vecteur d'infection ;
Le mécanisme assurant la collecte d'information. Pour l'espiogiciel Cydoor, la collecte consiste à enregistrer tout ce que l'utilisateur recherche et télécharge via le logiciel Kazaa.

Le mécanisme assurant la transmission à un tiers. Ce mécanisme est généralement assuré via le réseau Internet. Le tiers peut être le concepteur du programme ou une entreprise.

2.5 Cheval de Troie

Un cheval de Troie (Trojan Horse en anglais) est un logiciel d'apparence légitime, conçu pour exécuter des actions à l'insu de l'utilisateur. En général, il utilise les droits appartenant à son environnement pour détourner, diffuser ou détruire des informations, ou encore pour ouvrir une porte dérobée (fonctionnalité inconnue de l'utilisateur légitime, qui donne un accès secret au logiciel qui permet à un pirate informatique de prendre, à distance, le contrôle de l'ordinateur). Les chevaux de Troie informatiques sont programmés pour être installés de manière invisible, notamment pour corrompre l'ordinateur hôte.

La principale différence entre les virus, les vers et les chevaux de Troie est que ces derniers ne se répliquent pas. Ils sont divisés en plusieurs sous-classes comprenant entre autres les portes dérobées, les logiciels espions, les injecteurs, etc. On peut en trouver sur des sites malveillants ou autres. Cela dépend de ce que l'utilisateur télécharge.

2.6 Prévenir un accès direct à des données sensibles

Notions de protection des données sensibles

Un nom d'identifiant couplé à un mot de passe sont indispensables pour protéger l'accès aux données d'un ordinateur personnel ou bien mis en réseau. Dès que l'ordinateur est allumé l'identifiant et le mot de passe doivent être encodés. Une protection des progiciels est également très souvent utilisée. En général le mot de passe comporte au minimum 8 positions en mélangeant les lettres, les chiffres, des caractères spéciaux. Il est important de les mémoriser ou de les placer dans un endroit sûr et sécurisé (ne pas les écrire sur un papier collé sur son écran, ne pas les mettre non plus dans le tiroir de son bureau…)

Un mot de passe (en anglais : password) est un moyen d'authentification pour utiliser une ressource ou un service dont l'accès est limité et protégé.

Le mot de passe doit être tenu secret pour éviter qu'un tiers non autorisé puisse accéder à la ressource ou au service. C'est une méthode parmi d'autres pour effectuer une authentification, c'est-à-dire de vérifier qu'une personne correspond bien à l'identité (identifiant) déclarée. Il s'agit d'une preuve que l'on possède et que l'on transmet au service chargé d'autoriser l'accès

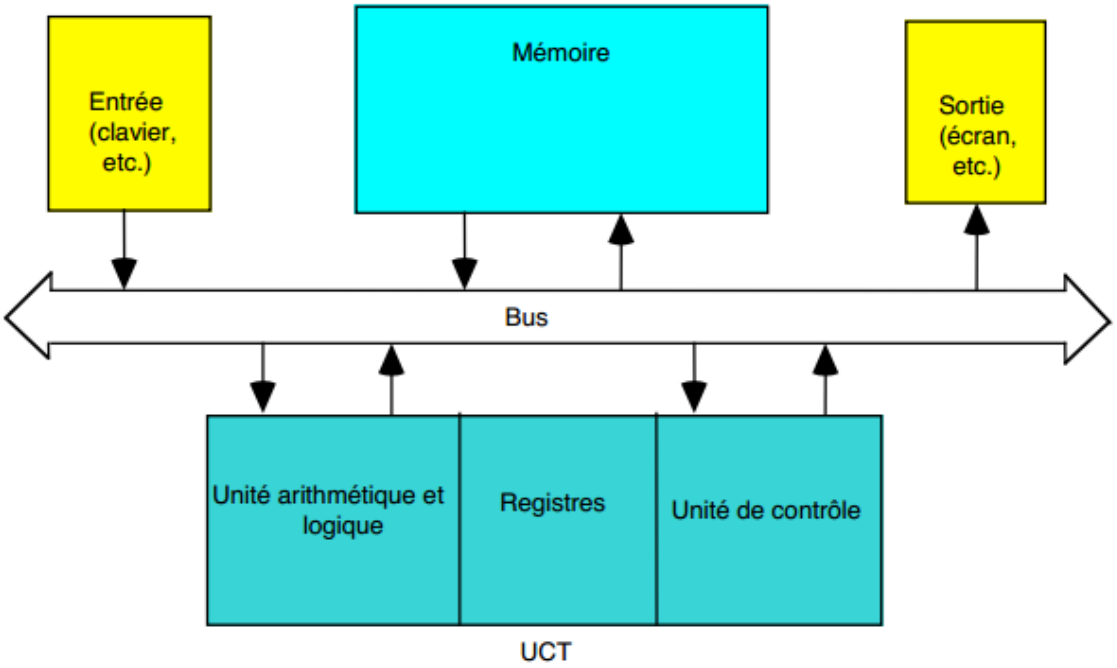
Module 3 : Hardware

3.1 Choix et exploitation d'un système informatique complet.

Choisir, parmi un ensemble d'offres, celle qui répond le mieux à un besoin précis

Apprécier l'adéquation des caractéristiques d'un composant en fonction de son utilisation prévue. En d'autres mots : bien se renseigner quant aux caractéristiques d'un composant (en lisant sa description) pour qu'il soit bien adapter à ce qu'on veut lui faire faire.

Architecture générale d'un ordinateur : unité centrale et périphériques



[2]

UCT étant l'unité centrale de traitement
La mémoire se décomposant en :
RAM (mémoire vive → où tout document ou fragment de programme se stocke pour le traitement). Rappelons que la RAM se vide lors de l'arrêt de l'ordinateur
ROM où se stockent les fonctionnalités et instructions primaires de l'ordinateur
De masse où se stockent les logiciels et les données. Structurée en dossiers

[2] **http://info.uqam.ca/~privat/INF2170/notes_de_cours/ndc02-structure_fonctionnement.pdf**

3.1 Architecture détaillée

Mémoire vive et morte, mémoire de masse, processeur, carte-mère, bus interface, périphériques, connectique

Mémoire vive

RAM (Read Active Memory aussi Random Active Memory) La **mémoire vive**, ou **mémoire système** aussi appelée **RAM** de l'anglais *Random Access Memory* (que l'on traduit en français par **mémoire à accès direct**[1]), est la mémoire informatique dans laquelle un ordinateur place les données lors de leur traitement.

Les caractéristiques de cette mémoire sont sa rapidité d'accès, essentielle pour fournir rapidement les données au processeur, et sa volatilité qui implique une perte totale de toutes les données mémoire dès que l'ordinateur cesse d'être alimenté en électricité. Cette caractéristique a tendance à disparaitre avec les dernières évolutions technologiques conduisant à des types de mémoire RAM non-volatile, comme les MRAM

Mémoire ROM

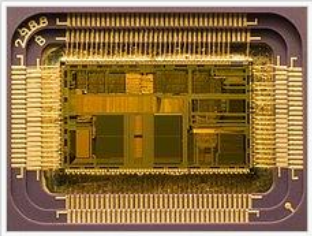
Originalement, l'expression **mémoire morte** (en anglais, *Read-Only Memory* : ***ROM***) désignait une mémoire informatique non volatile (c'est-à-dire une mémoire qui ne s'efface pas lorsque l'appareil qui la contient n'est plus alimenté en électricité) dont le contenu était fixé lors de sa programmation, et pouvait être lue plusieurs fois par l'utilisateur, mais ne pouvait plus être modifiée.

Mémoire de masse

Ce sont les disques du PC ainsi que les disques périphériques. Nous commençons à découvrir des mémoires sur internet (mais attention aux fuites de données et confidentialité). Nous pouvons aussi désigner qu'en informatique, une mémoire de masse est une mémoire de grande capacité, non volatile et qui peut être lue et écrite par un ordinateur.

Processeur

Le processeur (ou CPU de l'anglais Central Processing Unit, « Unité centrale de traitement ») est le composant de l'ordinateur qui exécute les instructions machine des programmes informatiques.



La puce d'un microprocesseur Intel 80486DX2 dans son boîtier (taille réelle : 12 × 6,75 mm).

Avec la mémoire notamment, c'est l'un des composants qui existent depuis les premiers ordinateurs et qui sont présents dans tous les ordinateurs. Un processeur construit en un seul circuit intégré est un microprocesseur.

Carte-mère

La carte mère est un matériel informatique (composé de circuits imprimés et de ports de connexions) servant à interconnecter tous les composants d'un micro-ordinateur. Elle est l'élément essentiel d'un micro-ordinateur.

La carte mère (en anglais « mainboard » ou « motherboard ») est la carte permettant la connexion de l'ensemble des éléments essentiels (comme la mémoire) et optionnels (comme les périphériques externes) du micro-ordinateur.

Au cours de son évolution, la carte mère a intégré certaines fonctions qui, avant, étaient des accessoires (internes ou externes). Dans un premier temps, la mémoire cache du processeur fut intégrée à celle-ci. Puis vinrent les cartes contrôleuses de disques durs, la carte son, la carte USB, la carte réseau et même parfois la carte graphique qui font maintenant partie intégrante du chipset.
Au démarrage, la carte mère doit reconnaître tous les éléments qui lui sont connectés (mémoire, périphériques…) et les configurer.

Périphériques

Un périphérique **informatique** est un dispositif connecté à un système informatique (ordinateur ou console de jeux[1]) qui ajoute à ce dernier des fonctionnalités.
On peut classer généralement les périphériques en deux types : les périphériques d'entrée et les périphériques de sortie. Les périphériques d'entrée servent à fournir des informations (ou données) au système informatique : clavier (frappe de texte), souris (pointage), scanneur (numérisation de documents papier), micro, webcam, etc. Les périphériques de sortie servent à faire sortir des informations du système informatique : écran, imprimante, haut-parleur, etc. On peut également rencontrer des périphériques d'entrée-sortie qui opèrent dans les deux sens : un lecteur de CD-ROM ou une clé USB, par exemple, permettent de stocker des données (sortie).

Connectique

La connectique regroupe toutes les techniques liées aux connexions physiques des liaisons électriques ainsi que des transmissions de données, c'est-à-dire les *connecteurs et prises*.
La connectique est omniprésente dans nos vies que ce soit pour relier nos appareils électriques à des prises d'alimentation électrique ou pour relier les différents éléments de nos systèmes informatiques, nos systèmes audios ou vidéo.
Le défi des manufacturiers et des ingénieurs est de maximiser la standardisation des connexions tout en conservant la fonctionnalité de ces connexions. La maximisation de la standardisation réduit les coûts et facilite la vie des consommateurs mais cette standardisation ne doit pas se faire au détriment de la puissance et de la fonctionnalité des appareils.

3.2 Exploitation d’un nouveau périphérique par l’ordinateur

Installation de pilote

Notion de pilote

Un pilote informatique, souvent abrégé en pilote, est un programme informatique destiné à permettre à un autre programme (souvent un système d'exploitation) d'interagir avec un périphérique. En général, chaque périphérique a son propre pilote. De manière simpliste, un pilote d'imprimante est un logiciel qui explique à l'ordinateur comment utiliser l'imprimante. Sans pilote, l'imprimante ou la carte graphique par exemple ne pourraient pas être utilisées
Lors de l'installation d'un nouveau composant ou d'un nouveau périphérique, il est nécessaire d'installer le pilote, généralement fourni par le constructeur. Sur de nombreux systèmes d'exploitation, la procédure *Plug-and-Play* détecte automatiquement le nouveau composant ou le nouveau périphérique. Il est cependant parfois nécessaire de lui fournir le support du pilote (CD-ROM, DVD ou emplacement), puis de procéder à l'installation et au paramétrage. Sous Linux, les pilotes n'ont pas besoins d'être téléchargés car ils sont fournis avec la distribution la plupart du temps (logiciel libre)

Comparaison entre un système d’exploitation et application

Système d’exploitation

En informatique, un **système d'exploitation** (souvent appelé **OS** pour *Operating System*, le terme anglophone) est un ensemble de programmes qui dirige l'utilisation des capacités d'un ordinateur par des logiciels applicatifs. Il reçoit de la part des logiciels applicatifs des demandes d'utilisation des capacités de l'ordinateur — capacité de stockage des mémoires et des disques durs, capacité de calcul du processeur.
Le système d'exploitation est le premier programme exécuté lors de la mise en marche de l'ordinateur, après l’amorçage. Il offre une suite de services généraux qui facilitent la création de logiciels applicatifs et sert d'intermédiaire entre ces logiciels et le matériel informatique[1].

Un système d'exploitation apporte commodité, efficacité et capacité d'évolution, permettant d'introduire de nouvelles fonctions et du nouveau matériel sans remettre en cause les logiciels
Il y a plusieurs systèmes d’exploitation : Windows, Linux…

Application

En français : progiciel, en anglais : Software
En informatique, un **logiciel** est un ensemble composé d'un ou plusieurs programmes, ainsi que les fichiers nécessaires pour les rendre opérationnels. Le logiciel détermine les tâches qu'un appareil informatique peut effectuer et donne à l'ordinateur sa valeur ajoutée.

Notion d'interfaçage homme / machine

- Manipulation du clavier et de la souris, click, double-click, combinaison de touches
- Manipulation des objets à travers une ligne de commande. Exemple : >cmd
- Manipulation des objets de l'interface graphique

3.4 Exercice choix du matériel informatique

Tâche :

Choisir un ordinateur et ses périphériques sur base des critères imposés dans le scénario créé par le professeur.

Contexte :

Les élèves travaillent par groupes de 2.

Le temps imparti est de 2 périodes de 50 minutes.

Les élèves disposent :
- d'un ordinateur pourvu de l'équipement nécessaire ;
- d'une connexion Internet ;
- d'un scénario

Consigne :

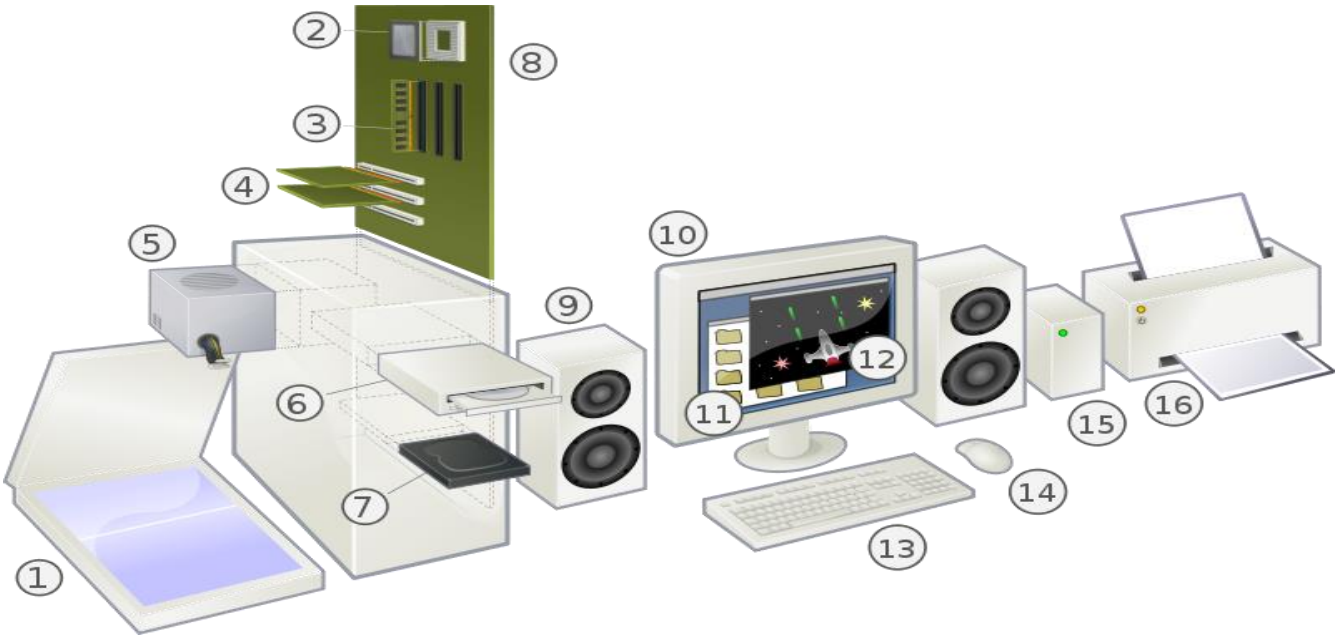
Choisir un ordinateur et ses périphériques sur base des critères imposés dans le scénario créé par le professeur.

Le travail en classe doit se dérouler dans le calme et dans le respect de ses pairs.

Une présentation orale de +- 10 minutes sera demandée à chaque groupe.

3.5 Composition d’un ordinateur

Numériseur de document (scanner) - CPU (Microprocesseur) - Mémoire vive (RAM) - Cartes de périphériques – Alimentation - Lecteur de disques (CD) - Disque dur - Carte mère – Enceintes – Écran - Logiciel système - Logiciel d'application – Clavier – Souris - Disque dur externe - Imprimante



1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

12.

13.

14.

15.

16.

3.6 R.a.i.d

Introduction

RAID est l'acronyme anglais de Redundant Array of Inexpensive Disks. Il s'agit d'un ensemble de standards destinés à regrouper des disques peu chers pour en faire une capacité de stockage plus importante et plus performante à un prix moindre comparé à de gros disques performants.

Au fil des ans, l'acronyme a vu se remplacer le " I " de Inexpensive par le " I " de Independant. L'objectif est d'essentiellement assurer la redondance des informations, donc fournir des fonctionnalités de sécurité permettant d'assurer l'Intégrité et la Disponibilité des données.

Il existe aujourd'hui plusieurs standards de " niveaux de RAID " ainsi que quelques autres solutions hybrides.

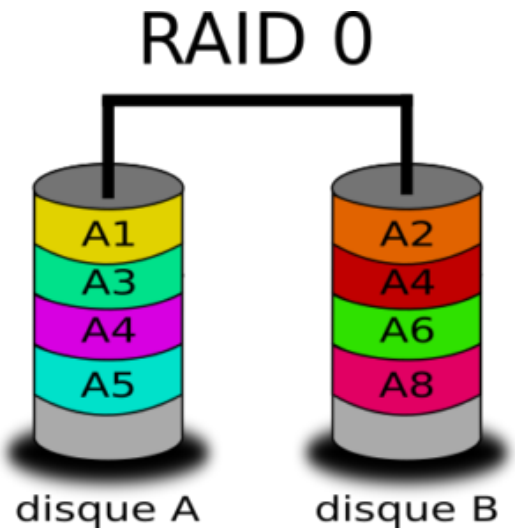
RAID 0	GRAPPE DE DISQUES AVEC STRIPING
RAID 1	MIROIR DE DISQUE
RAID 2	GRAPPE DE DISQUES EN PARALLELE AVEC DONNEES DE CONTROLE SEPAREES
RAID 3	GRAPPE DE DISQUES EN PARALLELE AVEC DISQUE DE PARITE
RAID 4	STRIPING AVEC PARITE
RAID 5	STRIPING ET PARITE ALTERNEE
RAID 6	GRAPPE DE DISQUES INDEPENDANTS AVEC DEUX SCHEMAS DE PARITE
RAID 7	PROPRIETAIRE
RAID 10	HAUTE DISPONIBILITE ET HAUTE PERFORMANCE
RAID 50	HAUTE PERFORMANCE
RAID 53	HAUTE PERFORMANCE
RAID 0+1	HAUTE PERFORMANCE

RAID 0 : grappe de disques avec striping

Il s'agit d'une solution permettant d'améliorer grandement les performances en lecture/écriture. Les données sont stockées en parallèle, sous forme de blocks (en anglais : bandes - strips). Par contre, cette solution n'offre aucune mesure de sécurité.

Les données sont réparties sur tous les disques de la grappe et accédées en parallèle.

Le RAID 0 peut s'effectuer à partir de 2 disques.



Avantages

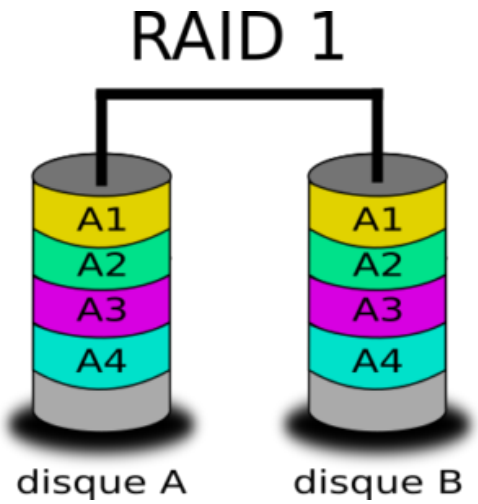
* Rapidité
* Facilité d'implémentation

Inconvénients

* Aucune sécurité en Disponibilité et en Intégrité.
* Les risques de perte de données sont multipliés par le nombre de disques en place : si l'un d'eux est en faute, toutes les données sont perdues.

RAID 1 : miroir de disque

Dans ce mode de fonctionnement, deux disques dupliquent les mêmes informations. L'accès est plus rapide en lecture (il est possible de lire des informations depuis les deux disques en même temps), cependant, l'écriture n'est pas améliorée.



Avantages

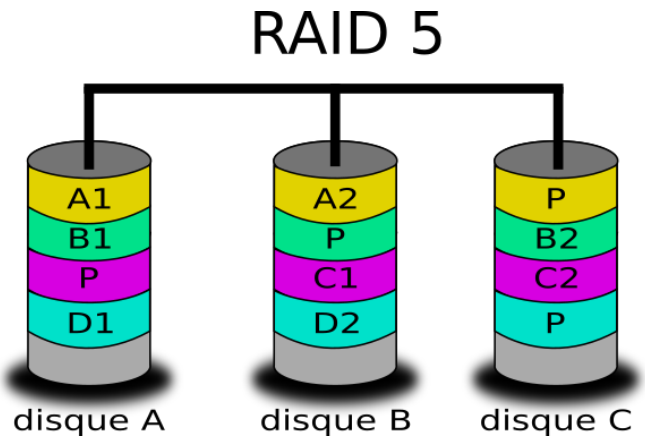
* Accélération de la lecture
* Disponibilité des données en cas de faille d'un disque

Inconvénients

* Nécessite un 2e disque, pour dupliquer les données, donc pas très efficace

RAID 5 : architecture qui bénéficie des avantages de RAID 0 et de RAID 1

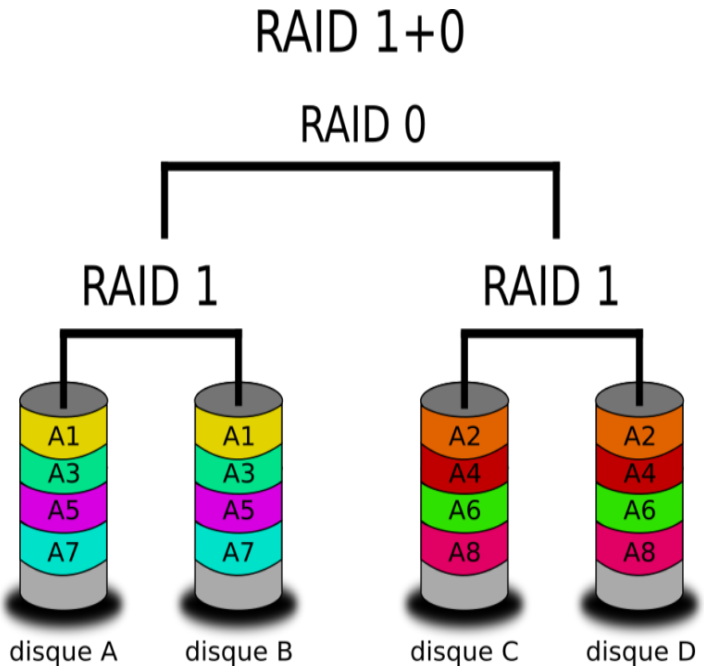
Les données sont en effet écrites par bande constituées de blocs de données (RAID 0) et d'un bloc de parité (RAID 1). Ainsi, si l'un des disques durs connaît une défaillance, il est possible de reconstituer les données sur un disque de remplacement grâce au bloc de parité.



Le système va écrire les premières données dans des bandes des disques A et B. La parité, calculée en fonction des données de A et B, est écrite sur le disque C. Les données suivantes sont écrites ensuite sur B et C, le bloc de parité est écrit sur A.
Si le disque A défaille, les données peuvent être reconstituées en effectuant un calcul mettant en jeu les données des disques B et C et le bloc de parité. L'opération OU exclusif (xor) est utilisée pour créer le bloc de parité ou retrouver les données perdues.

RAID 10 : **Haute disponibilité et haute performance**

Le RAID 10, qui se met en place à partir de 4 disques, est la superposition de deux technologies : le RAID 1 et le RAID 0. Les données sont découpées en blocs (RAID 0) écrits sur des disques distincts qui sont eux-mêmes dupliqués sur des disques miroirs (RAID 1).



Avantages

* Haute disponibilité (grâce aux miroirs)
* Bonne performance (grâce aux découpages en bandes)

Inconvénients

* Cher, car il faut commencer avec 4 disques

Grandeur des unités informatique :

1 Ko = 1 Kilo-octet = 1 000 octets
1 Mo = 1 Méga-octet = 1 000 000 octets
1 Go = 1 Giga- octets = 1 000 000 000 octets
1 To = 1 Tera- octets = 1 000 000 000 000 octets

Exercices :

Complète les affirmations ci-dessous par < , > ou =

2500 octets		0.5 Ko
10Go		10 000 000 000 octets
45 Mo		45 000 000 octets
77 000 000 octets		77 Go
20 To		22 000 000 000 000 octets

Module 4 : Multimédia (HTML)

4.1. Traiter des séquences sonores optimisées pour des besoins spécifiques

Lorsque je vous ai présenté les images et la balise `<img />` , j'ai commencé par un petit tour d'horizon des différents formats d'images (JPEG, PNG, GIF, etc.). Pour la vidéo et l'audio, je vais faire pareil... mais c'est plus compliqué.

En fait, le fonctionnement des vidéos est même tellement complexe qu'on pourrait faire un cours entier à ce sujet ! Étant donné qu'on parle ici de HTML, nous n'allons pas passer toutes nos prochaines nuits à étudier les subtilités de l'encodage vidéo. Je vais donc simplifier les choses et vous expliquer juste ce que vous avez besoin de savoir.

Les formats audio

Pour diffuser de la musique ou n'importe quel son, il existe de nombreux formats. La plupart d'entre eux sont compressés (comme le sont les images JPEG, PNG et GIF), ce qui permet de réduire leur poids :

- **MP3** : vous ne pouvez *pas* ne pas en avoir entendu parler ! C'est l'un des plus vieux, mais aussi l'un des plus compatibles (tous les appareils savent lire des MP3), ce qui fait qu'il est toujours très utilisé aujourd'hui ;
- **AAC** : utilisé majoritairement par Apple sur iTunes, c'est un format de bonne qualité. Les iPod, iPhone et autres iPad savent les lire sans problème ;
- **Ogg** : le format Ogg Vorbis est très répandu dans le monde du logiciel libre, notamment sous Linux. Ce format a l'avantage d'être libre, c'est-à-dire qu'il n'est protégé par aucun brevet ;
- **WAV** (**format non compressé**) : évitez autant que possible de l'utiliser car le fichier est très volumineux avec ce format. C'est un peu l'équivalent du Bitmap (BMP) pour l'audio.

La compatibilité dépend des navigateurs, mais elle évolue dans le bon sens au fil du temps. Pensez à consulter caniuse.com pour connaître la compatibilité actuelle des MP3, AAC, Ogg, WAV...

Insertion d'un élément audio

En théorie, il suffit d'une simple balise pour jouer un son sur notre page :

```html
<audio src="musique.mp3"></audio>
```

En pratique, c'est un peu plus compliqué que cela.
Si vous testez ce code... vous ne verrez rien ! En effet, le navigateur va seulement télécharger les informations générales sur le fichier (on parle de ***métadonnées***), mais il ne se passera rien de particulier.

Vous pouvez compléter la balise des attributs suivants :

- `controls` : pour ajouter les boutons « Lecture », « Pause » et la barre de défilement. Cela peut sembler indispensable, et vous vous demandez peut-être pourquoi cela n'y figure pas par défaut, mais certains sites web préfèrent créer eux-mêmes leurs propres boutons et commander la lecture avec du JavaScript ;
- `width` : pour modifier la largeur de l'outil de lecture audio ;
- `loop` : la musique sera jouée en boucle ;
- `autoplay` : la musique sera jouée dès le chargement de la page (évitez d'en abuser, c'est en général irritant d'arriver sur un site qui joue de la musique tout seul !) ;
- `preload` : indique si la musique peut être préchargée dès le chargement de la page ou non. Cet attribut peut prendre les valeurs :
  - `auto` (par défaut) : le navigateur décide s'il doit précharger toute la musique, uniquement les métadonnées ou rien du tout,
  - `metadata` : charge uniquement les métadonnées (durée, etc.),
  - `none` : pas de préchargement. Utile si vous ne voulez pas gaspiller de bande passante sur votre site.

On ne peut pas forcer le préchargement de la musique, c'est toujours le navigateur qui décide.
Les navigateurs mobiles, par exemple, ne préchargent jamais la musique, pour économiser la bande passante (le temps de chargement étant long sur un portable).

Ajoutons les contrôles et ce sera déjà mieux !

```html
<audio src="hype_home.mp3" controls></audio>
```

L'apparence du lecteur audio change en fonction du navigateur. La figure suivante représente par exemple le lecteur audio dans Google Chrome.

Le lecteur audio dans Google Chrome

Pourquoi ouvrir la balise pour la refermer immédiatement après ?

Cela vous permet d'afficher un message ou de proposer une solution de secours pour les navigateurs qui ne gèrent pas cette nouvelle balise. Par exemple :

```html
<audio src="hype_home.mp3" controls>Veuillez mettre à jour votre navigateur !</audio>
```

Ceux qui ont un navigateur récent ne verront pas le message. Les anciens navigateurs, qui ne comprennent pas la balise, afficheront en revanche le texte qui se trouve à l'intérieur.

Et si le navigateur ne gère pas le MP3, comment faire ?

Il faut proposer plusieurs versions du fichier audio. Dans ce cas, on va construire notre balise comme ceci :

```html
<audio controls>
    <source src="hype_home.mp3">
    <source src="hype_home.ogg">
</audio>
```

Le navigateur prendra automatiquement le format qu'il reconnaît.

4.2. Traiter des séquences animées (vidéos, animation intégrée…)

Les formats vidéo

Le stockage de la vidéo est autrement plus complexe. On a besoin de trois éléments :

- **un format conteneur** : c'est un peu comme une boîte qui va servir à contenir les deux éléments ci-dessous. On reconnaît en général le type de conteneur à l'extension du fichier : AVI, MP4, MKV... ;
- **un codec audio** : c'est le format du son de la vidéo, généralement compressé. Nous venons de les voir, on utilise les mêmes : MP3, AAC, OGG... ;
- **un codec vidéo** : c'est le format qui va compresser les images. C'est là que les choses se corsent, car ces formats sont complexes et on ne peut pas toujours les utiliser gratuitement. Les principaux à connaître pour le Web sont :
  - **H.264** : l'un des plus puissants et des plus utilisés aujourd'hui... mais il n'est pas 100 % gratuit. En fait, on peut l'utiliser gratuitement dans certains cas (comme la diffusion de vidéos sur un site web personnel), mais il y a un flou juridique qui fait qu'il est risqué de l'utiliser à tout va,
  - **Ogg Theora** : un codec gratuit et libre de droits, mais moins puissant que H.264. Il est bien reconnu sous Linux mais, sous Windows, il faut installer des programmes pour pouvoir le lire,
  - **WebM** : un autre codec gratuit et libre de droits, plus récent. Proposé par Google, c'est le concurrent le plus sérieux de H.264 à l'heure actuelle.

Là encore, surveillez bien la compatibilité sur caniuse.com : H.264, Ogg Theora, WebM... Le format H.264 semble sortir du lot. Il est quand même conseillé de proposer si possible chaque vidéo dans plusieurs formats, pour qu'elle soit lisible sur un maximum de navigateurs.

Pour convertir une vidéo dans ces différents formats, je vous conseille l'excellent logiciel gratuit Miro Video Converter.

Il vous suffit de glisser-déposer votre vidéo dans la fenêtre du programme et de sélectionner le format de sortie souhaité. Cela vous permettra de créer plusieurs versions de votre vidéo !

Les proportions de la vidéo sont toujours conservées. Si vous définissez une largeur et une hauteur, le navigateur fera en sorte de ne pas dépasser les dimensions indiquées, mais il conservera les proportions.

Insertion d'une vidéo

Il suffit d'une simple balise `<video>` pour insérer une vidéo dans la page :

```html
<video src="sintel.webm"></video>
```

Mais, là encore, vous risquez d'être déçu si vous utilisez seulement ce code. Aucun contrôle ne permet de lancer la vidéo !

Rajoutons quelques attributs (la plupart sont les mêmes que pour la balise `<audio>`) :

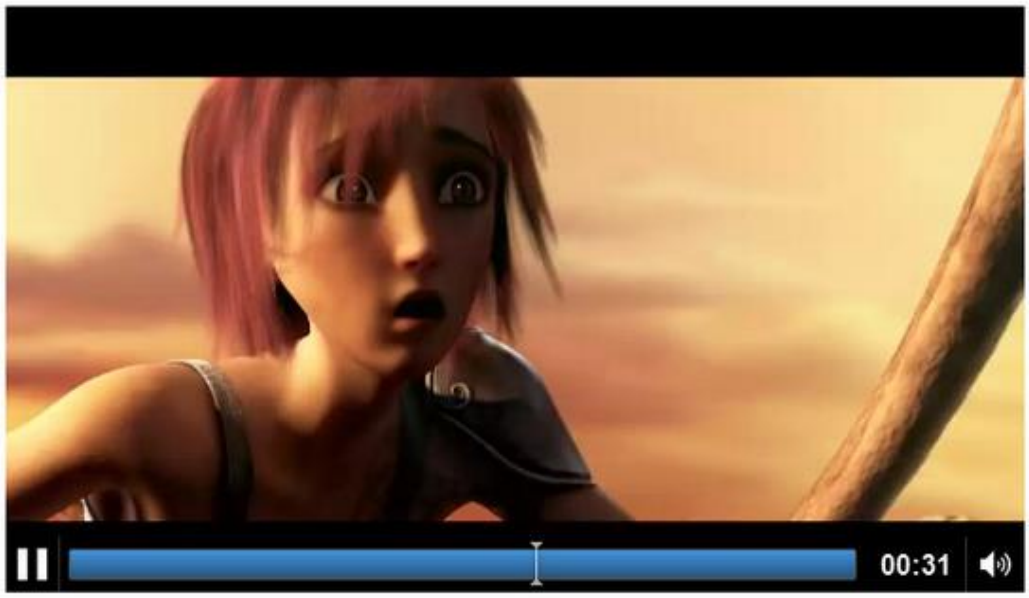
- `poster` : image à afficher à la place de la vidéo tant que celle-ci n'est pas lancée. Par défaut, le navigateur prend la première image de la vidéo mais, comme il s'agit souvent d'une image noire ou d'une image peu représentative de la vidéo, je vous conseille d'en créer une ! Vous pouvez tout simplement faire une capture d'écran d'un moment de la vidéo.
- `controls` : pour ajouter les boutons « Lecture », « Pause » et la barre de défilement. Cela peut sembler indispensable, mais certains sites web préfèrent créer eux-mêmes leurs propres boutons et commander la lecture avec du JavaScript. En ce qui nous concerne, ce sera largement suffisant !
- `width` : pour modifier la largeur de la vidéo.
- `height` : pour modifier la hauteur de la vidéo.
- `loop` : la vidéo sera jouée en boucle.
- `autoplay` : la vidéo sera jouée dès le chargement de la page. Là encore, évitez d'en abuser, c'est en général irritant d'arriver sur un site qui lance quelque chose tout seul !
- `preload` : indique si la vidéo peut être préchargée dès le chargement de la page ou non. Cet attribut peut prendre les valeurs :
  - `auto` (par défaut) : le navigateur décide s'il doit précharger toute la vidéo, uniquement les métadonnées ou rien du tout ;
  - `metadata` : charge uniquement les métadonnées (durée, dimensions, etc.) ;
  - `none` : pas de préchargement. Utile si vous souhaitez éviter le gaspillage de bande passante sur votre site.

On ne peut pas forcer le préchargement de la vidéo, c'est toujours le navigateur qui décide.

Voici un code un peu plus complet :

```html
<video src="sintel.webm" controls poster="sintel.jpg" width="600"></video>
```

Et le résultat :



Une vidéo avec les options de lecture et une taille définie

Pourquoi ouvrir et refermer immédiatement après la balise ?

La réponse est la même que pour la balise `<audio>` . Cela vous permet d'afficher un message ou d'utiliser une technique de secours (en Flash) si le navigateur ne reconnaît pas la balise :

```html
<video src="sintel.webm" controls poster="sintel.jpg" width="600">
    Il est temps de mettre à jour votre navigateur !
</video>
```

Comment contenter tous les navigateurs, puisque chacun reconnaît des formats vidéo différents ?

Vous utiliserez la balise `<source>` à l'intérieur de la balise `<video>` pour proposer différents formats. Le navigateur prendra celui qu'il reconnaît :

```html
<video controls poster="sintel.jpg" width="600">
    <source src="sintel.mp4">
    <source src="sintel.webm">
    <source src="sintel.ogv">
</video>
```

Les iPhone, iPad et iPod ne reconnaissent à l'heure actuelle que le format H.264 (fichier `.mp4`)... et uniquement si celui-ci apparaît en premier dans la liste ! Je vous recommande donc d'indiquer le format H.264 en premier pour assurer une compatibilité maximale.

Comment protéger ma vidéo ? Je ne veux pas qu'on puisse la copier facilement !

Ce n'est pas possible. Les balises n'ont pas été conçues pour limiter ou empêcher le téléchargement. C'est assez logique quand on y pense : pour que le visiteur puisse voir la vidéo, il faut bien de toute façon qu'il la télécharge d'une manière ou d'une autre !

N'espérez donc pas empêcher le téléchargement de votre vidéo avec cette technique.

Les lecteurs vidéo Flash permettent de « protéger » le contenu des vidéos mais, là encore, des solutions de contournement existent. De nombreux plugins permettent de télécharger les vidéos, de YouTube par exemple.

En résumé

- Insérer de la musique ou de la vidéo n'était pas possible autrefois en HTML. Il fallait recourir à un plugin comme Flash.
- Depuis HTML5, les balises `<audio>` et `<video>` ont été introduites et permettent de jouer de la musique et des vidéos sans plugin..
- Il existe plusieurs formats audio et vidéo. Il faut notamment connaître :
  - pour l'audio : MP3 et Ogg Vorbis ;
  - pour la vidéo : H.264, Ogg Theora et WebM.
- Aucun format n'est reconnu par l'ensemble des navigateurs : il faut proposer différentes versions de sa musique ou de sa vidéo pour satisfaire tous les navigateurs.
- Il faut ajouter l'attribut `controls` aux balises `<audio>` et `<video>` pour permettre au visiteur de lancer ou d'arrêter le média.

Module 5 : Création de site WEB

5.1. Concevoir et réaliser un site WEB à plusieurs documents liés, dans le respect des normes

5.1.1 Notion d'URL, d'hypertexte, de site, de butineur (browser), de serveur

Notions d'URL

Le sigle URL (de l'anglais Uniform Resource Locator, littéralement « localisateur uniforme de ressource »), auquel se substitue informellement le terme adresse web, désigne une chaîne de caractères utilisée pour adresser les ressources du World Wide Web : document HTML, image, son, forum Usenet, boîte aux lettres électronique, entre autres.

Une URL est une chaîne de caractères combinant les informations nécessaires pour indiquer à un logiciel comment accéder à une ressource Internet. Ces informations peuvent notamment comprendre le protocole de communication, un nom d'utilisateur, un mot de passe, une adresse IP ou un nom de domaine, un numéro de port TCP/IP, un chemin d'accès, une requête.

Les informations nécessaires varient selon la ressource et le contexte d'utilisation de l'URL. En outre un identificateur de fragment peut être ajouté à la fin d'une URL pour identifier un élément à l'intérieur de la ressource.

URL absolue

Une URL absolue permet d'indiquer comment accéder à une ressource indépendamment de tout contexte où elle peut être précisée ou transmise. Elle commence par l'indication d'un schéme de représentation (spécifique au protocole de communication utilisé pour accéder à cette ressource), suivi de l'ensemble des paramètres permettant de localiser sur le réseau le service hébergeant la ressource, puis permet de préciser à ce service le nom d'une ressource à traiter, transmettre des données de traitement, acheminer et récupérer les résultats, puis de préciser éventuellement quelle partie de ce résultat sera utilisée.

- Http – protocole de communication pour accéder au serveur web - schéme
- : - caractère de séparation (obligatoire si schéme précisé)
- // - chaîne de caractères pour les protocoles dont la requête comprend un chemin d'accès, permettant de préciser et localiser le service avant ce chemin.
- Nom de domaine (pourrait être remplacé par l'adresse IP)
- Chemin absolu contenant le nom de la page web

URL relative

Les protocoles utilisant un chemin hiérarchique permettent l'utilisation d'URL relatives. Une URL relative ne contient ni protocole ni nom de domaine. Ceux-ci sont déduits à partir de l'URL de la ressource contenant l'URL relative.

Les URL relatives sont souvent utilisées pour les hyperliens à l'intérieur d'un même site web. Si le document d'URL http://fr.wikipedia.org/wiki/Web contient l'URL relative Navigateur, cela correspond à http://fr.wikipedia.org/wiki/Navigateur. Les URL relatives sont directement inspirées de la syntaxe des systèmes de fichiers Unix. L'usage d'URL relatives permet de ne pas avoir à reprendre l'ensemble des liens lors du changement d'adresse d'un site.

Hypertexte

Un hyperlien ou lien hypertexte, est une référence système hypertexte permettant de passer automatiquement d'un document consulté à un document lié. Les hyperliens sont notamment utilisés dans le World Wide Web pour permettre d'accéder à une page web.

Un lien hypertexte ou hyperlien permet en cliquant dessus d'atteindre un autre endroit de la page, une autre page ou un autre site évalué comme pertinent par l'auteur.

Site

Un site ou site web (de l'anglais web site, qui se traduit littéralement en français par site de la toile) est un ensemble de pages web hyper liées entre elles et accessible à une adresse web. On dit aussi site internet par métonymie, le World Wide Web reposant sur l'Internet.
Butineur (browser)

Un navigateur Web est un logiciel conçu pour consulter le World Wide Web. Techniquement, c'est au minimum un client HTTP.
Il existe de nombreux navigateurs Web, pour toutes sortes de matériels (ordinateur personnel, tablette tactile, téléphones mobiles, etc.) et pour différents systèmes d'exploitation (Linux, Windows, Mac OS, iOS et Android). Les plus utilisés à l'heure actuelle sont, Google Chrome, Mozilla Firefox, Internet Explorer, Safari, Opera.

Serveur

On appelle serveur Web aussi bien le matériel informatique que le logiciel, qui joue le rôle de serveur informatique sur un réseau local ou sur le World Wide Web.

- En tant que matériel, un serveur Web est un ordinateur comme un autre. Comme tout serveur, il est relié à un réseau informatique et fait fonctionner un logiciel serveur.
- En tant que logiciel, un serveur Web est plus précisément un serveur HTTP, HTTP étant le principal protocole de communication employé par le World Wide Web.

Un serveur web héberge plusieurs sites web

5.2. Fonctionnement des sites web

Comment fonctionnent les sites web ?

Non, n'ayez pas peur de poser des questions, même si vous pensez qu'elles sont « bêtes ». Il est très important que nous en parlions un peu avant de nous lancer à fond dans la création de sites !
Je suis certain que vous consultez des sites web tous les jours. Pour cela, vous lancez un programme appelé le *navigateur web*, en cliquant sur l'une des icônes représentées à la figure suivante.

Les icônes des navigateurs web les plus répandus
Avec le navigateur, vous pouvez consulter n'importe quel site web. Voici par exemple un navigateur affichant le célèbre site web Wikipédia :

Le site web Wikipédia

Je suis sûr que vous avez l'habitude d'utiliser un navigateur web ! Aujourd'hui, tout le monde sait aller sur le Web… mais qui sait vraiment comment le Web fonctionne ? Comment créer des sites web comme celui-ci ?

J'ai entendu parler de HTML, de CSS, est-ce que cela a un rapport avec le fonctionnement des sites web ?

Tout à fait !

Il s'agit de **langages informatiques** qui permettent de créer des sites web. Tous les sites web sont basés sur ces langages, ils sont incontournables et universels aujourd'hui. Ils sont la base même du Web. Le langage HTML a été inventé par un certain Tim Berners-Lee en 1991…
Tim Berners-Lee suit encore aujourd'hui avec attention l'évolution du Web. Il a créé le *World Wide Web Consortium* (W3C), qui définit les nouvelles versions des langages liés au Web. Il a par ailleurs créé plus récemment la *World Wide Web Foundation*, qui analyse et suit l'évolution du Web.
Les langages HTML et CSS sont à la base du fonctionnement de tous les sites web. Quand vous consultez un site avec votre navigateur, il faut savoir que, en coulisses, des rouages s'activent pour permettre au site web de s'afficher. L'ordinateur se base sur ce qu'on lui a expliqué en HTML et CSS pour savoir ce qu'il doit afficher, comme le montre la figure suivante.

HTML et CSS sont deux « langues » qu'il faut savoir parler pour créer des sites web. C'est le navigateur web qui fera la traduction entre ces langages informatiques et ce que vous verrez s'afficher à l'écran.

HTML et CSS : deux langages pour créer un site web
Pour créer un site web, on doit donner des instructions à l'ordinateur. Il ne suffit pas de taper le texte qui devra figurer dans le site (comme on le ferait dans un traitement de texte Word, par exemple), il faut aussi indiquer où placer ce texte, insérer des images, faire des liens entre les pages, etc.

Les rôles de HTML et CSS

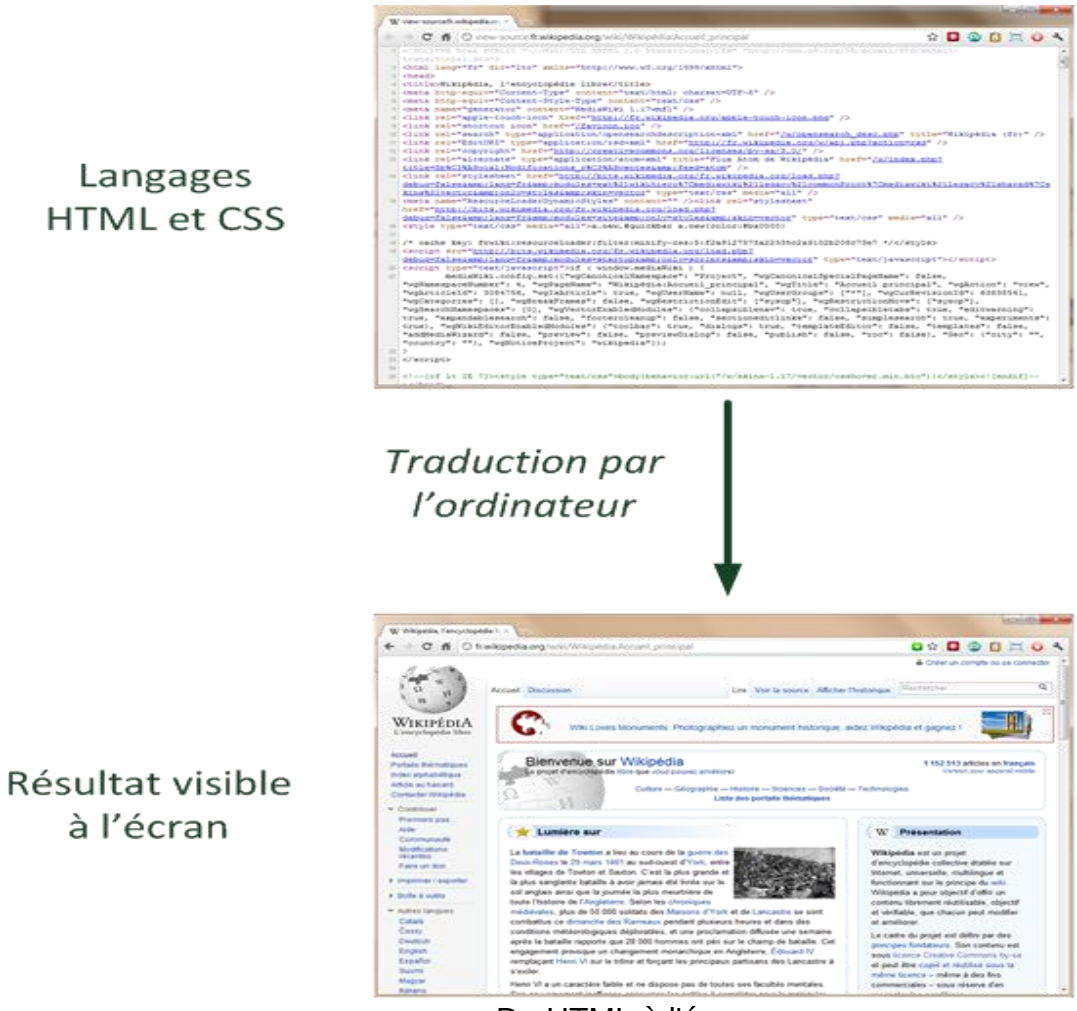
Pour expliquer à l'ordinateur ce que vous voulez faire, il va falloir utiliser un langage qu'il comprend. Et c'est là que les choses se corsent, parce qu'*il va falloir apprendre deux langages* !

Pourquoi avoir créé deux langages ? Un seul aurait suffi, non ?

Vous devez vous dire que manipuler deux langages va être deux fois plus complexe et deux fois plus long à apprendre… mais ce n'est pas le cas ! Je vous rassure, s'il y a deux langages c'est, au contraire, pour faciliter les choses. Nous allons avoir affaire à deux langages qui *se complètent* car ils ont des rôles différents :

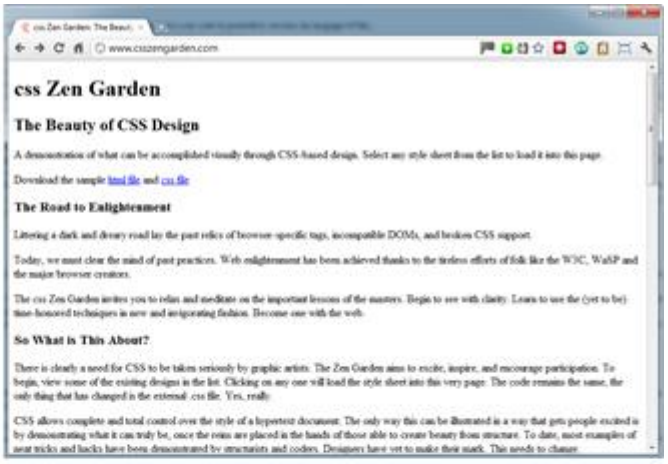
HTML (*HyperText Markup Language*) : il a fait son apparition dès 1991 lors du lancement du Web. Son rôle est de gérer et organiser le contenu. C'est donc en HTML que vous écrirez ce qui doit être affiché sur la page : du texte, des liens, des images… Vous direz par exemple : « Ceci est mon titre, ceci est mon menu, voici le texte principal de la page, voici une image à afficher, etc. » ;

CSS (*Cascading Style Sheets*, aussi appelées *feuilles de style*) : le rôle du CSS est de gérer l'apparence de la page web (agencement, positionnement, décoration, couleurs, taille du texte…). Ce langage est venu compléter le HTML en 1996.
ous pouvez très bien créer un site web uniquement en HTML, mais celui-ci ne sera pas très beau : l'information apparaîtra « brute ». C'est pour cela que le langage CSS vient toujours le compléter.
Pour vous donner une idée, la figure suivante montre ce que donne la même page sans CSS, puis avec le CSS.



Du HTML à l'écran

HTML
(pas de CSS)



HTML + CSS

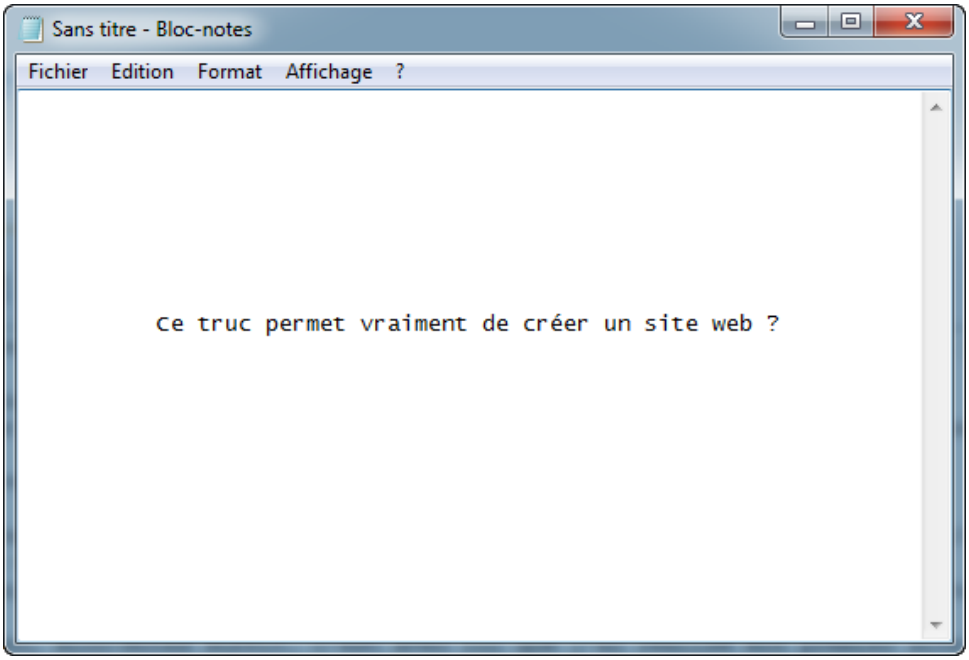
Avec et sans CSS

Le HTML définit le contenu (comme vous pouvez le voir, c'est brut de décoffrage !). Le CSS permet, lui, d'arranger le contenu et de définir la présentation : couleurs, image de fond, marges, taille du texte…

Comme vous vous en doutez, le CSS a besoin d'une page HTML pour fonctionner. C'est pour cela que nous allons d'abord apprendre les bases du HTML avant de nous occuper de la décoration en CSS.
Vos premières pages ne seront donc pas les plus esthétiques, mais qu'importe ! Cela ne durera pas longtemps.

L'éditeur de texte

De quel logiciel vais-je avoir besoin pour créer mon site web ?
Vais-je devoir casser ma tirelire pour acheter un logiciel très complexe que je vais mettre des mois à comprendre ?
Il existe effectivement de nombreux logiciels dédiés à la création de sites web. Mais, je vous rassure, vous n'aurez pas à débourser un seul centime. Pourquoi aller chercher un logiciel payant et compliqué, alors que vous avez déjà tout ce qu'il faut chez vous ?
Eh oui, accrochez-vous bien parce qu'il suffit de… Bloc-Notes !



Le logiciel Bloc-Notes de Windows

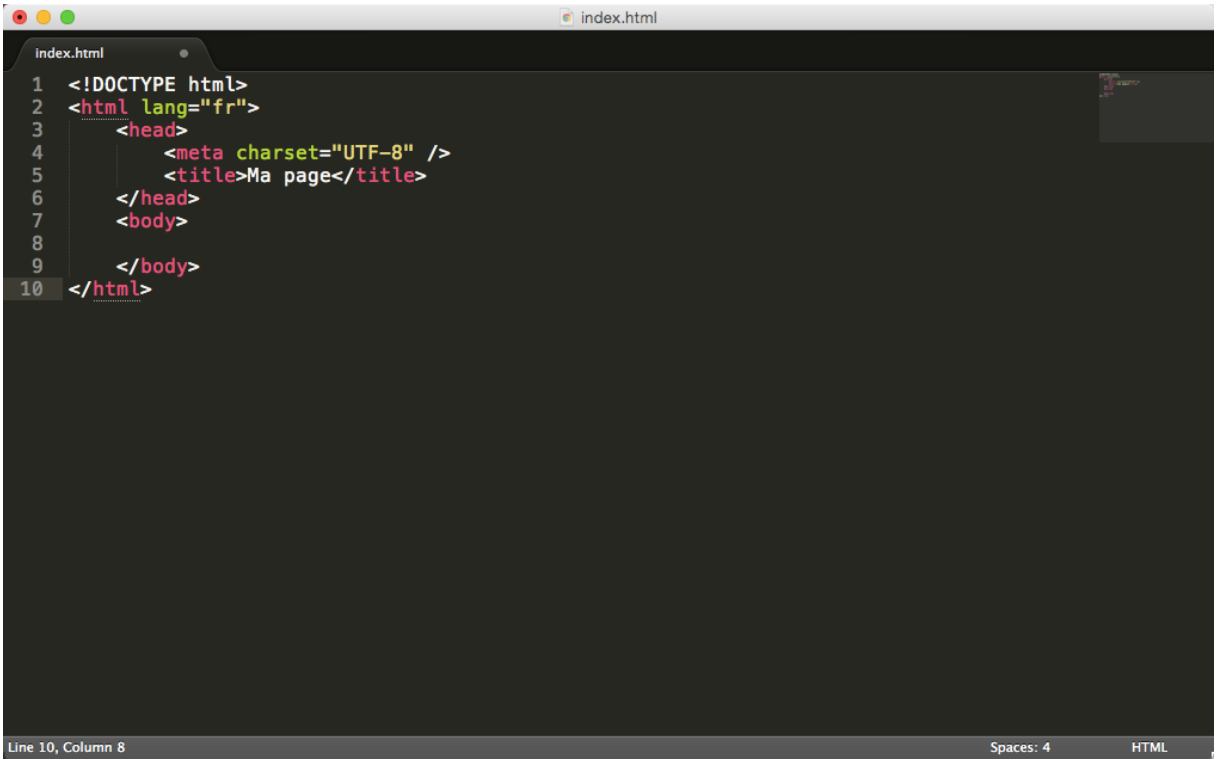
Incroyable mais vrai : on peut tout à fait créer un site web uniquement avec Bloc-Notes, le logiciel d'édition de texte intégré par défaut à Windows.
Sublime Text : mon éditeur
Sublime Text est un éditeur de texte devenu très populaire parmi les développeurs. On l'utilise aussi bien pour développer en HTML et CSS que dans d'autres langages (Python, Ruby, etc.).

Site web de Sublime Text

Il a l'avantage d'être simple, épuré et facile à lire dès le départ. Pas de centaines de boutons dont on ne comprend pas à quoi ils servent.

Malgré les apparences, il ne faut pas croire qu'il est limité. Au contraire : il est possible de l'étendre avec tout un système de plugins. Ça devient un peu plus compliqué et on ne rentrera pas là-dedans, mais il faut savoir que certains personnalisent énormément leur Sublime Text pour gagner du temps !

Sublime Text peut tout à fait être utilisé gratuitement, mais son auteur demande à payer au bout d'un certain temps d'usage. Vous pourrez toujours continuer à l'utiliser gratuitement. Personnellement, je considère qu'il en vaut vraiment le coup et je l'ai acheté. Je vous laisse choisir et vous faire votre propre idée à ce sujet.



L'éditeur Sublime Text : c'est beau, c'est propre, c'est pur

En somme, **Sublime Text est à la fois simple et puissant**. Même pour l'usage basique que nous allons avoir, il s'avèrera très pratique.

Voici quelques logiciels que vous pouvez essayer sous Windows si vous voulez en tester plusieurs :
Sublime Text (j'insiste ;o) ;
Notepad++ ;
Brackets ;

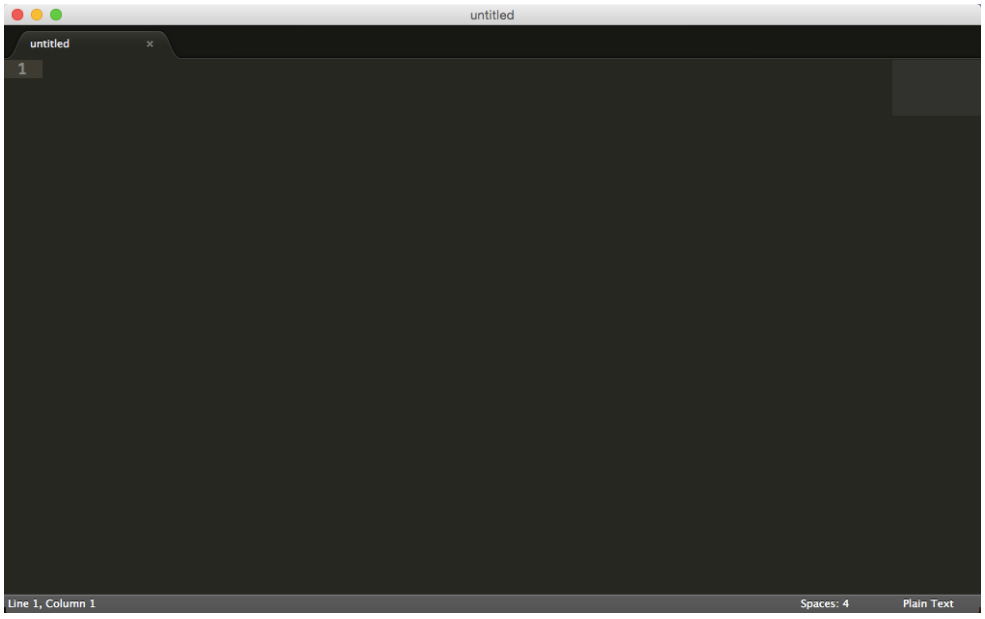
… et bien d'autres si vous recherchez « Éditeur de texte » sur le Web.

5.3 Créez votre première page web en HTML

Ça y est, vous avez installé tous les logiciels ? Vous devriez maintenant avoir un éditeur de texte pour créer votre site (comme Sublime Text) et plusieurs navigateurs pour le tester (Mozilla Firefox, Google Chrome…).
Dans ce chapitre, nous allons commencer à pratiquer ! Nous allons découvrir les bases du langage HTML et enregistrer notre toute première page web ! Alors oui, bien sûr, ne vous attendez pas encore à réaliser une page web exceptionnelle dès ce deuxième chapitre, mais patience… cela viendra !

Allez, mettons-nous en situation ! Comme je vous l'ai dit, nous allons créer notre site dans un éditeur de texte. Vous avez dû en installer un, à la suite de mes conseils dans le premier chapitre : qu'il s'appelle Sublime Text, Notepad++, Brackets, jEdit, vim, TextWrangler… peu importe. Ces logiciels ont un but très simple : vous permettre d'écrire du texte !

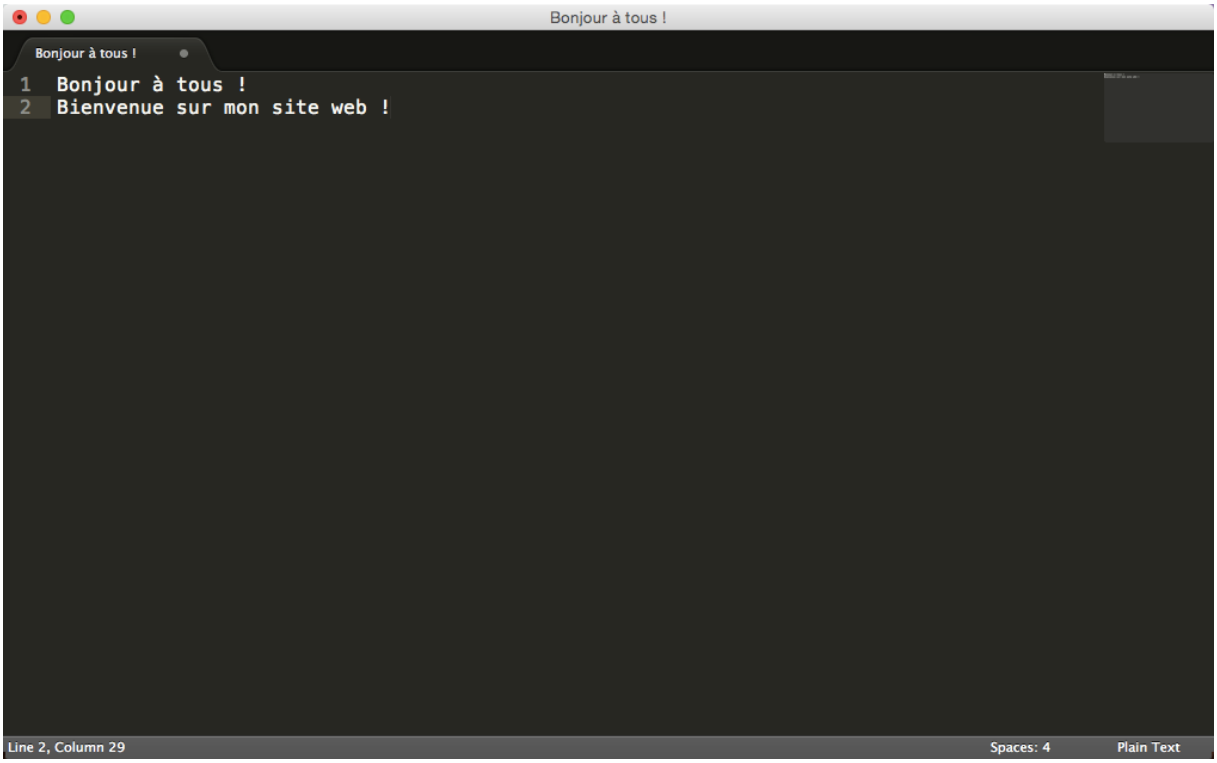
Dans la suite de ce cours, je travaillerai avec Sublime Text. Je vais donc l'ouvrir :



Ouverture de Sublime Text

Bon, qu'est-ce qu'on fait maintenant ? Qu'est-ce qu'on écrit sur cette feuille blanche (euh... noire) ?

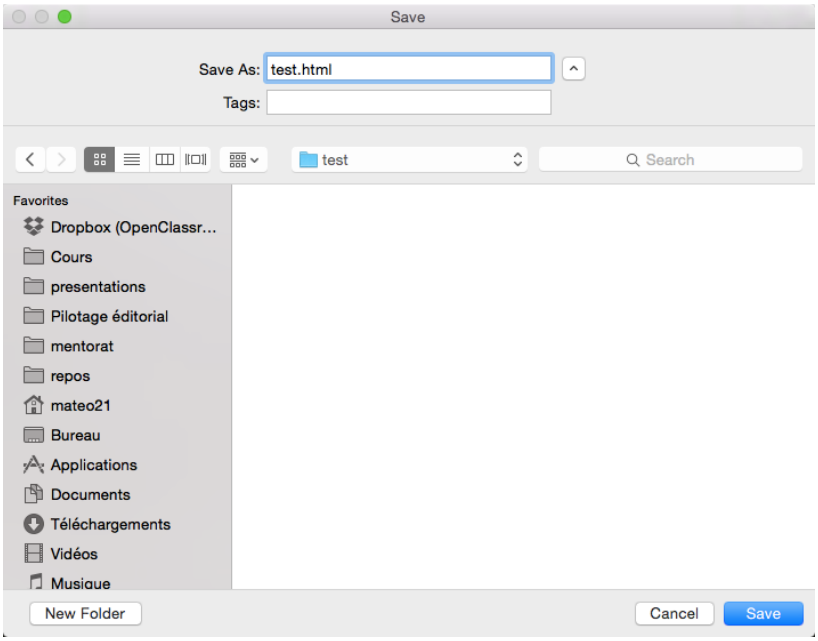
On va faire un petit essai. Je vous invite à écrire ce qui vous passe par la tête, comme moi à la figure suivante.



Du texte dans Sublime Text

Vous pouvez écrire les mêmes phrases que moi ou ce que vous voulez ; le but est d'écrire quelque chose.

Maintenant, enregistrons ce fichier. Pour cela, c'est très simple : comme dans tous les programmes, vous avez un menu Fichier > Enregistrer (ou File > Save en anglais). Une boîte de dialogue vous demande où enregistrer le fichier et sous quel nom. Enregistrez-le où vous voulez. Donnez au fichier le nom que vous voulez, en terminant par .html ; par exemple test.html , comme indiqué à la figure suivante.

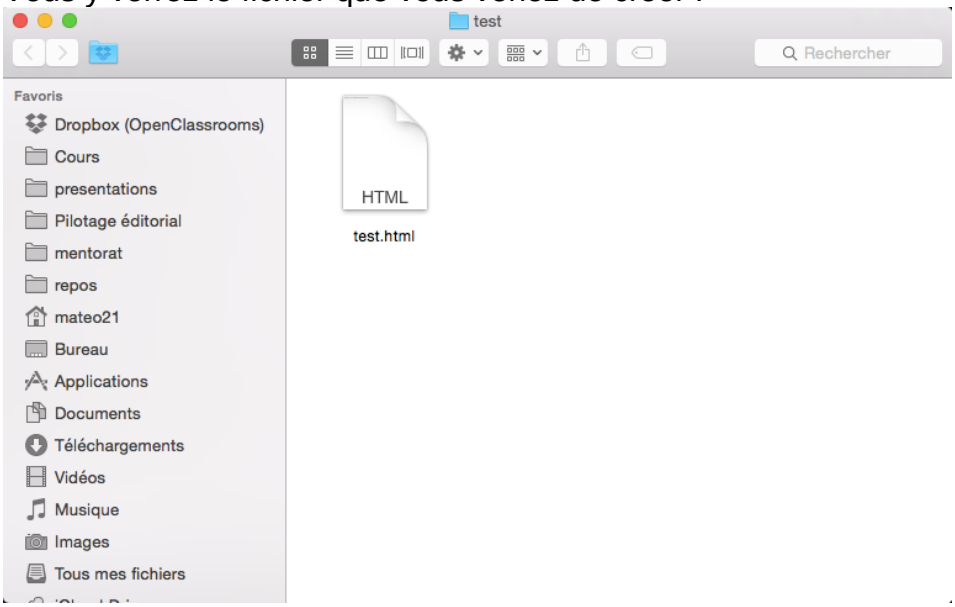


Enregistrement d'un fichier sous Sublime Text

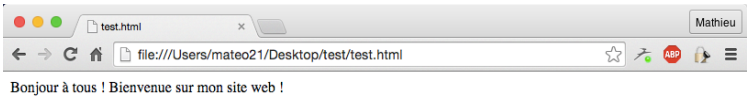
Je vous recommande de créer un nouveau dossier dans vos documents, qui contiendra les fichiers de votre site. Pour ma part, j'ai créé un dossier test dans lequel j'ai mis mon fichier test.html

Ouvrez maintenant l'explorateur de fichiers dans le dossier où vous avez enregistré votre page (l'Explorateur sous Windows ou le Finder sous Mac).

Vous y verrez le fichier que vous venez de créer :



Le fichier dans l'explorateur



L'icône qui représente le fichier dépend de votre navigateur web par défaut, en général. Vous pouvez y voir une icône de Firefox, de Chrome... ou un aperçu, comme ici.
N'y prêtez pas attention. Faites simplement un double-clic sur ce fichier et… votre navigateur s'ouvre et affiche le texte que vous avez écrit.

Cela ne marche pas bien, on dirait ! Tout le texte s'affiche sur la même ligne alors qu'on avait écrit deux lignes de texte différentes !?

Le texte s'affiche sur la même ligne alors qu'on avait demandé à l'écrire sur deux lignes différentes. Que se passe-t-il ?
En fait, pour créer une page web, il ne suffit pas de taper simplement du texte comme on vient de le faire. En plus de ce texte, il faut aussi écrire ce qu'on appelle des balises, qui vont donner des instructions à l'ordinateur comme « aller à la ligne », « afficher une image », etc.

5.4 Les balises et leurs attributs

Bon, tout cela était trop facile. Évidemment, il a fallu que ces satanés informaticiens s'en mêlent et compliquent les choses. Il ne suffit pas d'écrire « simplement » du texte dans l'éditeur, il faut aussi donner des instructions à l'ordinateur. En HTML, on utilise pour cela des **balises**.

Les balises

Les pages HTML sont remplies de ce qu'on appelle des *balises*. Celles-ci sont invisibles à l'écran pour vos visiteurs, mais elles permettent à l'ordinateur de comprendre ce qu'il doit afficher.
Les balises se repèrent facilement. Elles sont entourées de « chevrons », c'est-à-dire des symboles < et > , comme ceci : <balise> .

À quoi est-ce qu'elles servent ? Elles indiquent la nature du texte qu'elles encadrent. Elles veulent dire par exemple : « Ceci est le titre de la page », « Ceci est une image », « Ceci est un paragraphe de texte », etc.

On distingue deux types de balises : les balises en paires et les balises orphelines.

Les balises en paires

Elles s'ouvrent, contiennent du texte, et se ferment plus loin. Elles ressemblent à ça:

```
1 <titre>Ceci est un titre</titre>
```

On distingue une balise ouvrante (<titre>) et une balise fermante (</titre>) qui indique que le titre se termine. Cela signifie pour l'ordinateur que tout ce qui n'est *pas* entre ces deux balises… n'est pas un titre.

```
1 Ceci n'est pas un titre <titre>Ceci est un titre</titre> Ceci n'est pas un titre
```

Les balises orphelines

Ce sont des balises qui servent le plus souvent à insérer un élément à un endroit précis (par exemple une image). Il n'est pas nécessaire de délimiter le début et la fin de l'image, on veut juste dire à l'ordinateur « Insère une image ici ».

Une balise orpheline s'écrit comme ceci :

```
<image />
```

Notez que le / de fin n'est pas obligatoire. On pourrait écrire seulement `<image>` . Néanmoins, pour ne pas les confondre avec le premier type de balise, les webmasters recommandent de rajouter ce `/` (*slash*) à la fin des balises orphelines. Vous me verrez donc mettre un `/` aux balises orphelines et je vous recommande de faire de même, c'est une bonne pratique.

Vous me verrez donc mettre un / aux balises orphelines et je vous recommande de faire de même, c'est une bonne pratique.

Les attributs

Les attributs sont un peu les options des balises. Ils viennent les compléter pour donner des informations supplémentaires. L'attribut se place après le nom de la balise ouvrante et a le plus souvent une valeur, comme ceci :

```
<balise attribut="valeur">
```

À quoi cela sert-il ? Prenons la balise <image /> que nous venons de voir. Seule, elle ne sert pas à grand-chose. On pourrait rajouter un attribut qui indique le nom de l'image à afficher :

```
<image nom="photo.jpg" />
```

L'ordinateur comprend alors qu'il doit afficher l'image contenue dans le fichier photo.jpg .

Dans le cas d'une balise fonctionnant « par paire », on ne met les attributs que dans la balise ouvrante et pas dans la balise fermante. Par exemple, ce code indique que la citation est de Neil Armstrong et qu'elle date du 21 juillet 1969 :

```
<citation auteur="Neil Armstrong" date="21/07/1969">
C'est un petit pas pour l'homme, mais un bond de géant pour l'humanité.
</citation>
```

Toutes les balises que nous venons de voir sont fictives. Les vraies balises ont des noms en anglais (eh oui !) ; nous allons les découvrir dans la suite de ce cours.

Structure de base d'une page HTML5

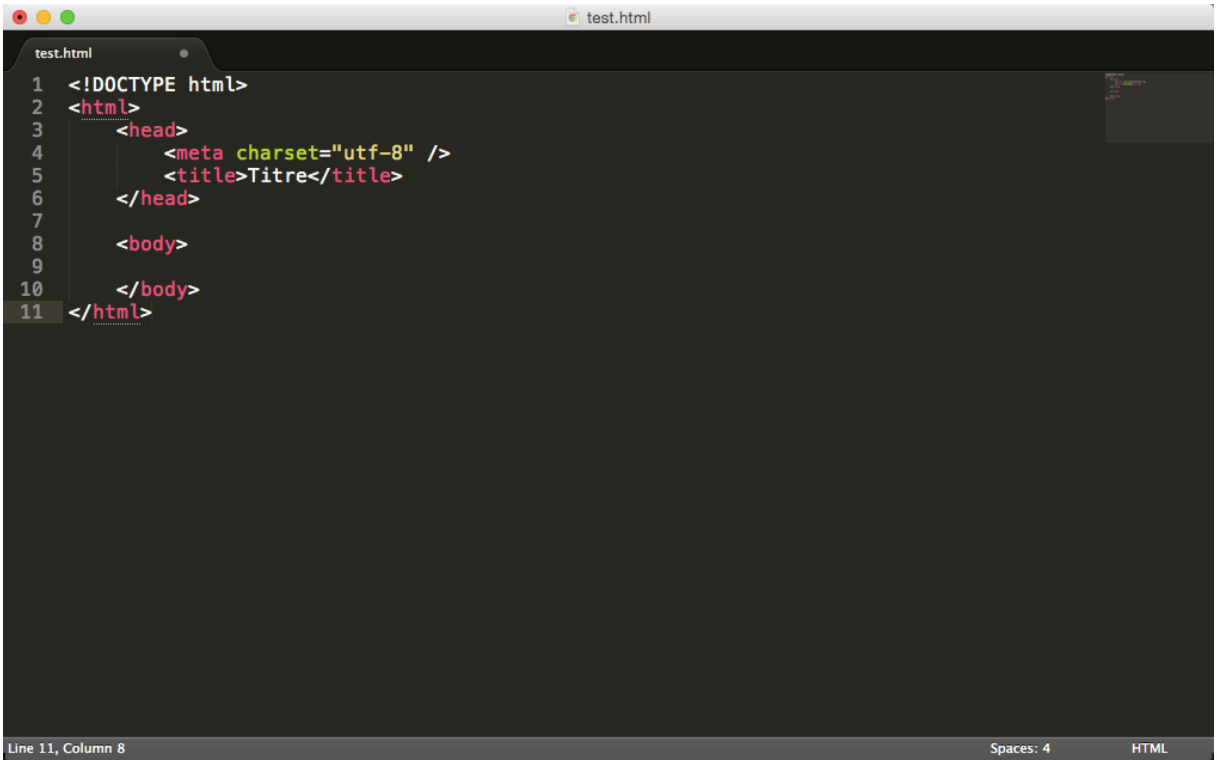
Reprenons notre éditeur de texte (dans mon cas, Sublime Text). Je vous invite à écrire ou à copier-coller le code source ci-dessous dans votre éditeur de texte. Ce code correspond à la base d'une page web en HTML5 :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Titre</title>
    </head>

    <body>

    </body>
</html>
```

J'ai mis des espaces au début de certaines lignes pour « décaler » les balises.

Ce n'est pas obligatoire et cela n'a aucun impact sur l'affichage de la page, mais cela rend le code source plus lisible. On appelle cela l'***indentation***. Dans votre éditeur, il suffit d'appuyer sur la touche Tab pour avoir le même résultat.

Copié dans Sublime Text, vous devriez voir:



Code HTML5 minimal dans Sublime Text

Vous noterez que les balises s'ouvrent et se ferment dans un ordre précis. Par exemple, la balise <html> est la première que l'on ouvre et c'est aussi la dernière que l'on ferme (tout à la fin du code, avec </html>). *Les balises doivent être fermées dans le sens inverse de leur ouverture.* Un exemple :

- <html><body></body></html> : **correct**. Une balise qui est ouverte à l'intérieur d'une autre doit aussi être fermée à l'intérieur.
- <html><body></html></body> : **incorrect**, les balises s'entremêlent.

Euh, on pourrait avoir des explications sur toutes les balises que l'on vient de copier dans l'éditeur, m'sieur ?

Bien sûr, c'est demandé si gentiment. 
Ne prenez pas peur en voyant toutes ces balises d'un coup, je vais vous expliquer leur rôle !

Le doctype

```
1 <!DOCTYPE html>
```

La toute première ligne s'appelle le ***doctype***. Elle est indispensable car c'est elle qui indique qu'il s'agit bien d'une page web HTML.
Ce n'est pas vraiment une balise comme les autres (elle commence par un point d'exclamation). Vous pouvez considérer que c'est un peu l'exception qui confirme la règle.

Cette ligne du doctype était autrefois incroyablement complexe. Il était impossible de la retenir de tête. Pour XHTML 1.0, il fallait écrire :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Dans le cadre de HTML5, il a été décidé de la simplifier, pour le plus grand bonheur des webmasters. Quand vous voyez une balise doctype courte (`<!DOCTYPE html>`), cela signifie que la page est écrite en HTML5.

La balise </html>

```
1 <html>
2
3 </html>
```

C'est la balise principale du code. Elle englobe tout le contenu de votre page. Comme vous pouvez le voir, la balise fermante </html> se trouve tout à la fin du code !

L'en-tête <head> et le corps <body>

Une page web est constituée de deux parties :

- L'en-tête <head> : cette section donne quelques informations générales sur la page, comme son titre, l'encodage (pour la gestion des caractères spéciaux), etc. Cette section est généralement assez courte. Les informations que contient l'en-tête ne sont pas affichées sur la page, ce sont simplement des informations générales à destination de l'ordinateur. Elles sont cependant très importantes !
- Le corps <body> : c'est là que se trouve la partie principale de la page. Tout ce que nous écrirons ici sera affiché à l'écran. C'est à l'intérieur du corps que nous écrirons la majeure partie de notre code.

Pour le moment, le corps est vide (nous y reviendrons plus loin). Intéressons-nous par contre aux deux balises contenues dans l'en-tête…

L'encodage (charset)

```
1 <meta charset="utf-8" />
```

Cette balise indique l'encodage utilisé dans votre fichier .html .

Sans rentrer dans les détails, car cela pourrait vite devenir compliqué, l'encodage indique la façon dont le fichier est enregistré. C'est lui qui détermine comment les caractères spéciaux vont s'afficher (accents, idéogrammes chinois et japonais, caractères arabes, etc.).

Il y a plusieurs techniques d'encodage, portant des noms bizarres, et utilisées en fonction des langues : ISO-8859-1, OEM 775, Windows-1253… Une seule cependant devrait être utilisée aujourd'hui autant que possible : UTF-8. Cette méthode d'encodage permet d'afficher sans aucun problème pratiquement tous les symboles de toutes les langues.

Il ne suffit pas de *dire* que votre fichier est en UTF-8. Il faut aussi que votre fichier soit bien *enregistré* en UTF-8. C'est heureusement le cas désormais par défaut dans la plupart des éditeurs de texte.

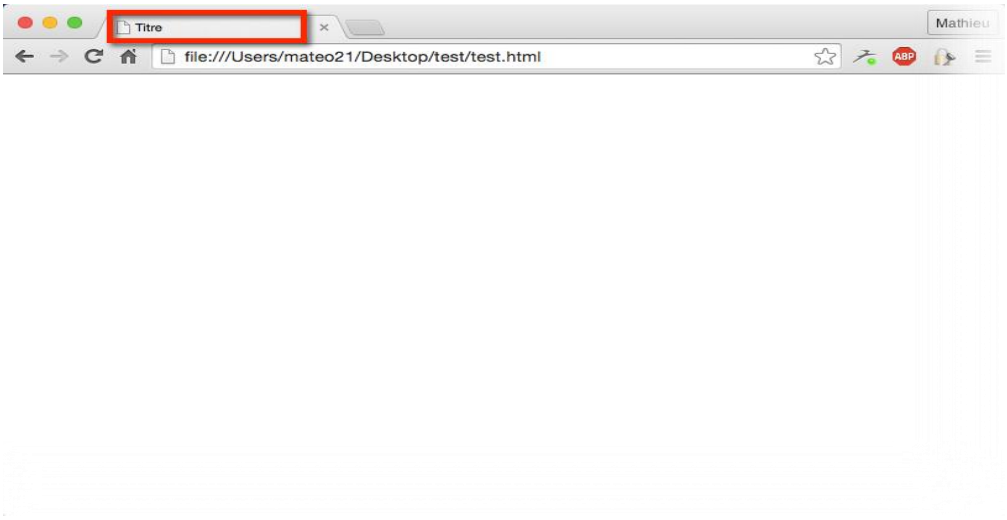
Si les accents s'affichent mal par la suite, c'est qu'il y a un problème avec l'encodage. Vérifiez que la balise meta indique bien UTF-8, et que votre fichier est enregistré en UTF-8 (sous Sublime Text, allez dans le menu File > Save with Encoding > UTF-8 pour vous assurer que votre fichier est enregistré en UTF-8.).

Le titre principal de la page

```
1 <title>
```

C'est le titre de votre page, probablement l'élément le plus important ! Toute page doit avoir un titre qui décrit ce qu'elle contient.
Il est conseillé de garder le titre assez court (moins de 100 caractères, en général).

Le titre ne s'affiche pas dans votre page mais en haut de celle-ci (souvent dans l'onglet du navigateur). Enregistrez votre page web et ouvrez-la dans votre navigateur. Vous verrez que le titre s'affiche dans l'onglet, comme sur la figure suivante.



Le titre de la page apparaît en haut du navigateur

Il faut savoir que le titre apparaît aussi dans les résultats de recherche, comme sur Google (figure suivante).

Les commentaires

Un **commentaire** en HTML est un texte qui sert simplement de mémo. Il n'est pas affiché, il n'est pas lu par l'ordinateur, cela ne change rien à l'affichage de la page.

Bref, cela ne sert à rien ?

Eh bien si !
Cela sert à *vous* et aux personnes qui liront le code source de votre page. Vous pouvez utiliser les commentaires pour laisser des indications sur le fonctionnement de votre page.

Quel intérêt ? Cela vous permettra de vous rappeler comment fonctionne votre page si vous revenez sur votre code source après un long moment d'absence. Ne rigolez pas, cela arrive à tous les webmasters.

Insérer un commentaire

Un commentaire est une balise HTML avec une forme bien spéciale :

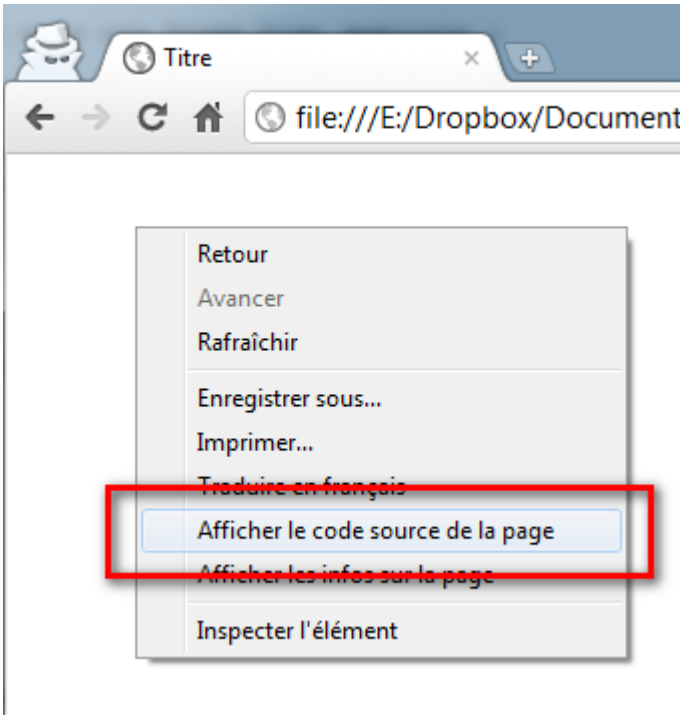
<!-- Ceci est un commentaire -->

Vous pouvez le mettre où vous voulez au sein de votre code source : il n'a aucun impact sur votre page, mais vous pouvez vous en servir pour vous aider à vous repérer dans votre code source (surtout s'il est long).

```
<!DOCTYPE html>
<html>
    <head>
        <!-- En-tête de la page -->
        <meta charset="utf-8" />
        <title>Titre</title>
    </head>

    <body>
        <!-- Corps de la page -->
    </body>
</html>
```

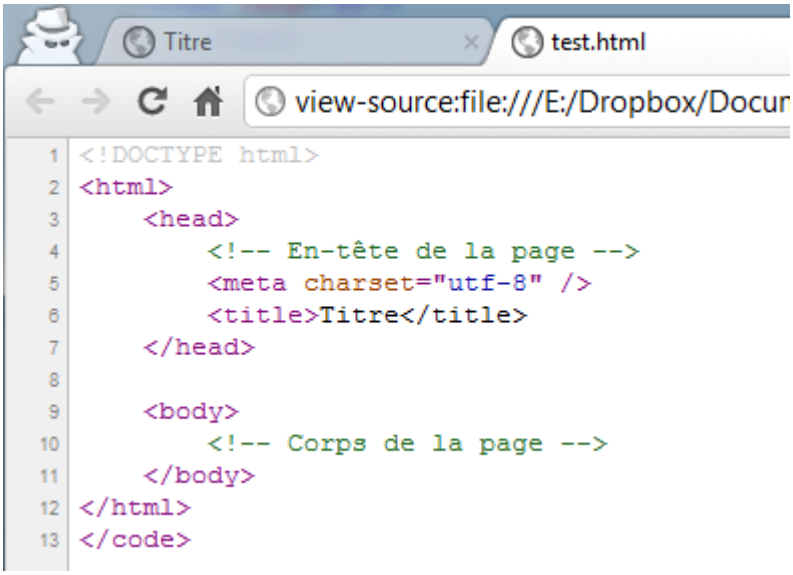
Tout le monde peut voir vos commentaires… et tout votre code HTML !

Terminons par une remarque importante : *tout le monde peut voir le code HTML de votre page* une fois celle-ci mise en ligne sur le Web. Il suffit de faire un clic droit sur la page et de sélectionner « Afficher le code source de la page » (l'intitulé peut changer selon votre navigateur), comme le montre la figure suivante.



Menu Afficher le code source

Le code source s'affiche alors (figure suivante).



Affichage du code source

Vous pouvez tester cette manipulation sur n'importe quel site web, cela marche ! Garanti à 100 %. Cela s'explique assez facilement : le navigateur *doit* obtenir le code HTML pour savoir ce qu'il faut afficher. Le code HTML de tous les sites est donc public.

La morale de l'histoire ? Tout le monde pourra voir votre code HTML et vous ne pouvez pas l'empêcher. Par conséquent, ne mettez pas d'informations sensibles comme des mots de passe dans les commentaires… et soignez votre code source, car je pourrai venir vérifier si vous avez bien suivi mon cours à la lettre ! 

Lorsque vous regarderez le code de certains sites web, ne prenez pas peur s'il vous paraît long ou semble ne pas respecter les mêmes règles que celles que je vous présente dans ce cours. Tous les sites ne sont pas écrits en HTML5 (loin de là) et, parfois, certains webmasters rédigent très mal leur code, ce ne sont pas toujours des exemples à suivre !

En résumé

- On utilise l'éditeur de texte (Sublime Text, Notepad++, jEdit, vim…) pour créer un fichier ayant l'extension .html (par exemple : test.html). Ce sera notre page web.
- Ce fichier peut être ouvert dans le navigateur web simplement en faisant un double-clic dessus.
- À l'intérieur du fichier, nous écrirons le contenu de notre page, accompagné de balises HTML.
- Les balises peuvent avoir plusieurs formes :
    - <balise> </balise> : elles s'ouvrent et se ferment pour délimiter le contenu (début et fin d'un titre, par exemple) ;
    - <balise /> : balises orphelines (on ne les insère qu'en un seul exemplaire), elles permettent d'insérer un élément à un endroit précis (par exemple une image).
- Les balises sont parfois accompagnées d'attributs pour donner des indications supplémentaires (exemple : <image nom="photo.jpg" />).
- Une page web est constituée de deux sections principales : un en-tête (<head>) et un corps (<body>).
- On peut afficher le code source de n'importe quelle page web en faisant un clic droit puis en sélectionnant Afficher le code source de la page .

5.5 Organisez votre texte

❖ Les paragraphes

La plupart du temps, lorsqu'on écrit du texte dans une page web, on le fait à l'intérieur de paragraphes. Le langage HTML propose justement la balise `<p>` pour délimiter les paragraphes.

```
<p>Bonjour et bienvenue sur mon site !</p>
```

- `<p>` signifie « Début du paragraphe » ;
- `</p>` signifie « Fin du paragraphe ».

Comme je vous l'ai dit au chapitre précédent, on écrit le contenu du site web entre les balises `<body></body>` . Il nous suffit donc de mettre notre paragraphe entre ces deux balises et nous aurons enfin notre première vraie page web avec du texte !

Je reprends donc exactement le même code qu'au chapitre précédent et j'y ajoute mon paragraphe :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Paragraphes</title>
    </head>

    <body>
        <p>Bonjour et bienvenue sur mon site !</p>
    </body>
</html>
```

Sauter une ligne

En HTML, si vous appuyez sur la touche `Entrée` , cela ne crée pas une nouvelle ligne comme vous en avez l'habitude. Essayez donc ce code :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Essais de sauts de ligne</title>
    </head>

    <body>
        <p>Bonjour et bienvenue sur mon site !
            Ceci est mon premier test alors soyez indulgents s'il vous plaît, j'apprends petit à petit comment cela marche.
            Pour l'instant c'est un peu vide, mais revenez dans 2-3 jours quand j'aurai appris un peu plus de choses, je vous assure que vous allez être surpris !</p>
    </body>
</html>
```

Tout le texte s'affiche sur la même ligne alors qu'on est bien allé à la ligne dans le code ! Taper frénétiquement sur la touche `Entrée` dans l'éditeur de texte ne sert donc strictement à rien.

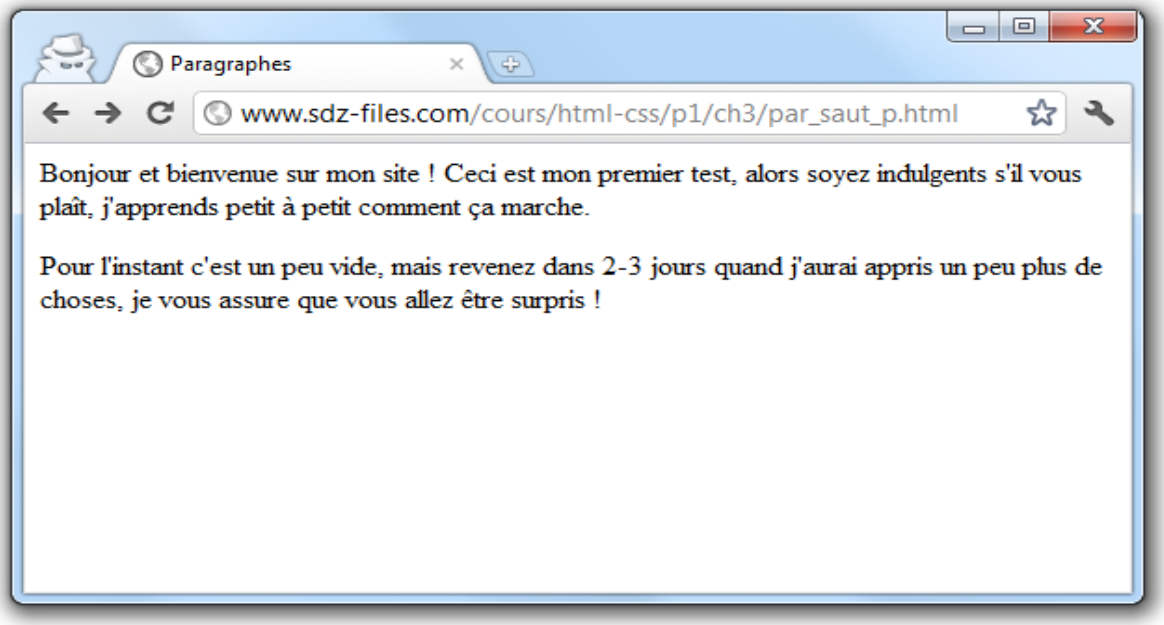
Comme vous devez vous en douter, il y a pourtant bien un moyen de faire des sauts de ligne en HTML.

En fait, si vous voulez écrire un deuxième paragraphe, il vous suffit d'utiliser une deuxième balise `<p>` .
Votre code HTML devrait donc être au final rempli de balises de paragraphe !

Un exemple :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Paragraphes</title>
    </head>

    <body>
        <p>Bonjour et bienvenue sur mon site !
        Ceci est mon premier test alors soyez indulgents s'il vous plaît, j'apprends petit à petit comment cela marche.</p>

        <p>Pour l'instant c'est un peu vide, mais revenez dans 2-3 jours quand j'aurai appris un peu plus de choses, je vous assure que vous allez être surpris !</p>
    </body>
</html>
```

Le résultat se trouve à la figure suivante.



Deux paragraphes avec 2 balises <p>

Oui, mais si je veux juste aller à la ligne dans un paragraphe et non pas sauter une ligne ?

Eh bien devinez quoi : il existe une balise « Aller à la ligne » !
C'est une balise **orpheline** qui sert juste à indiquer qu'on doit aller à la ligne : `<br />` . Vous devez obligatoirement la mettre *à l'intérieur d'un paragraphe*.

Voici comment l'utiliser dans un code :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Sauts de ligne</title>
    </head>

    <body>
        <p>Bonjour et bienvenue sur mon site !<br />
        Ceci est mon premier test alors soyez indulgents s'il vous plaît, j'apprends petit à petit comment cela marche.</p>

        <p>Pour l'instant c'est un peu vide, mais revenez dans 2-3 jours quand j'aurai appris un peu plus de choses, je vous assure que vous allez être surpris !</p>
    </body>
</html>
```

Vous pouvez théoriquement mettre plusieurs balises `<br />` d'affilée pour faire plusieurs sauts de lignes, mais on considère que c'est une mauvaise pratique qui rend le code délicat à maintenir. Pour décaler un texte avec plus de précision, on utilisera le CSS, ce langage qui vient compléter le HTML et dont je vous parlerai un peu plus loin.

Donc c'est compris ?

- `<p> </p>` : pour organiser son texte en paragraphes ;
- `<br />` : pour aller à la ligne.

Maintenant qu'on sait écrire des paragraphes, voyons voir comment on crée des titres.

❖ Les titres

Lorsque le contenu de votre page va s'étoffer avec de nombreux paragraphes, il va devenir difficile pour vos visiteurs de se repérer. C'est là que les titres deviennent utiles.

En HTML, on est verni, on a le droit d'utiliser six niveaux de titres différents. Je veux dire par là qu'on peut dire « Ceci est un titre très important », « Ceci est un titre un peu moins important », « Ceci est un titre encore moins important », etc.

On a donc six balises de titres différentes :

- `<h1> </h1>` : signifie « titre très important ». En général, on s'en sert pour afficher le titre de la page au début de celle-ci ;
- `<h2> </h2>` : signifie « titre important » ;
- `<h3> </h3>` : pareil, c'est un titre un peu moins important (on peut dire un « sous-titre », si vous voulez) ;
- `<h4> </h4>` : titre encore moins important ;
- `<h5> </h5>` : titre pas important ;
- `<h6> </h6>` : titre vraiment, mais alors là vraiment pas important du tout.

Attention : ne confondez pas avec la balise `<title>` ! La balise `<title>` affiche le titre de la page dans la barre de titre du navigateur, comme nous l'avons vu. Les titres `<h1>` et compagnie, eux, servent à créer des titres qui seront affichés *dans* la page web.

Ne vous laissez pas impressionner par toutes ces balises. En fait, six niveaux de titres, c'est beaucoup. Dans la pratique, personnellement, je n'utilise que les balises `<h1>` , `<h2>` et `<h3>` , et très rarement les autres (je n'ai pas souvent besoin de six niveaux de titres différents). Votre navigateur affiche le titre très important en très gros, le titre un peu moins important en un peu moins gros, etc.

Ne choisissez pas votre balise de titre en fonction de la taille qu'elle applique au texte ! Il faut impérativement bien structurer sa page en commençant par un titre de niveau 1 (`<h1>`), puis un titre de niveau 2 (`<h2>`), etc. Il ne devrait pas y avoir de sous-titre sans titre principal !
Si vous voulez modifier la taille du texte, sachez que nous apprendrons à faire cela en CSS un peu plus tard.

Essayez de créer une page web avec des titres pour voir ce que cela donne :

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Niveaux de titres</title>
    </head>

    <body>
        <h1>Titre super important</h1>
        <h2>Titre important</h2>
        <h3>Titre un peu moins important (sous-titre)</h3>

        <h4>Titre pas trop important</h4>
        <h5>Titre pas important</h5>
        <h6>Titre vraiment pas important du tout</h6>
    </body>
</html>
```

Allez, je vous donne un exemple d'utilisation des titres dans une page web (vous allez voir que je ne me sers pas de toutes les balises) :

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Présentation d'OpenClassrooms</title>
</head>

<body>
    <h1>Bienvenue sur OpenClassrooms !</h1>

    <p>Bonjour et bienvenue sur mon site : OpenClassrooms.<br />
       OpenClassrooms, qu'est-ce que c'est ?</p>

    <h2>Des cours pour débutants</h2>

    <p>OpenClassrooms vous propose des cours (tutoriels) destinés aux débutants : aucune connaissance n'est requise pour lire ces cours !</p>

    <p>Vous pourrez ainsi apprendre, sans rien y connaître auparavant, à créer un site web, à programmer, à construire des mondes en 3D !</p>

    <h2>Une communauté active</h2>

    <p>Vous avez un problème, un élément du cours que vous ne comprenez pas ? Vous avez besoin d'aide pour créer votre site ?<br />
    Rendez-vous sur les forums ! Vous y découvrirez que vous n'êtes pas le seul dans ce cas et vous trouverez très certainement quelqu'un qui vous aidera aimablement à résoudre votre problème.</p>
</body>
</html>
```

Voilà une page web qui prend forme !

Oui, mais moi je veux centrer mon titre, l'écrire en rouge et le souligner !

Nous ferons tout cela lorsque nous apprendrons le CSS (dès la deuxième partie du cours). Il faut savoir que `<h1>` ne signifie pas « Times New Roman, taille 16 pt », mais « *Titre important* ».

Grâce au langage CSS, vous pourrez dire « Je veux que mes titres importants soient centrés, rouges et soulignés ». Pour le moment, en HTML, nous ne faisons que structurer notre page. *Nous rédigeons le contenu avant de nous amuser à le mettre en forme.*

❖ La mise en valeur

Pour mettre *un peu* en valeur votre texte, vous devez utiliser la balise `<em>` `</em>` . Son utilisation est très simple : encadrez les mots à mettre en valeur avec ces balises et c'est bon ! Je reprends un peu l'exemple de tout à l'heure et j'y mets quelques mots en évidence :

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Emphase</title>
</head>

<body>
    <p>Bonjour et bienvenue sur mon site !<br />
    Ceci est mon premier test alors <em>soyez indulgents</em> s'il vous plaît, j'apprends petit à petit comment cela marche.</p>
</body>
</html>
```

Vous voyez sûrement le texte s'afficher en gras. Là encore, le gras n'est qu'une *conséquence*. Le navigateur a choisi d'afficher en gras les mots importants pour les faire ressortir davantage.

La balise `<strong>` ne signifie pas « mettre en gras », mais « important ». On pourra décider plus tard, en CSS, d'afficher les mots « importants » d'une autre façon que le gras si on le souhaite.

❖ Marquer le texte

La balise `<mark>` permet de faire ressortir visuellement une portion de texte. L'extrait n'est pas forcément considéré comme important, mais on veut qu'il se distingue bien du reste du texte. Cela peut être utile pour faire ressortir un texte pertinent après une recherche sur votre site, par exemple.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Marquage du texte</title>
</head>

<body>
    <p>Bonjour et bienvenue sur mon site !<br />
    Ceci est mon premier test alors <mark>soyez indulgents</mark> s'il vous plaît, j'apprends petit à petit comment cela marche.</p>
</body>
</html>
```

Par défaut, `<mark>` a pour effet de surligner le texte. On pourra changer l'affichage en CSS (décider de surligner dans une autre couleur, d'encadrer le texte, etc.). C'est le même principe que ce que je vous disais pour les balises précédentes : elles indiquent le *sens* des mots et non pas comment ceux-ci doivent s'afficher.

N'oubliez pas : HTML pour le fond, CSS pour la forme

Je vais peut-être vous sembler un peu lourd, mais il est très important qu'on se comprenne bien car les débutants font souvent la même grosse erreur à ce stade. Ils ont vu les balises `<em>` ,`<strong>` , `<mark>` … et ils se disent : « Chouette, j'ai découvert comment mettre en italique, en gras et comment surligner du texte en HTML ! ».

Et pourtant… ce n'est pas à cela que servent ces balises ! Je sais, je sais, vous allez me dire « Oui mais quand j'utilise `<strong>` , le texte apparaît en gras, donc c'est pour mettre en gras. » et pourtant, c'est une erreur de croire que cette balise sert à cela.

Le rôle des balises est d'indiquer le *sens* du texte. Ainsi, `<strong>` indique à l'ordinateur « Ce texte est important ». C'est tout.
Et pour *montrer* que le texte est important, l'ordinateur décide de le mettre en gras (mais il pourrait aussi bien l'écrire en rouge !). La plupart des navigateurs affichent les textes importants en gras, mais rien ne les y oblige.

Je ne comprends pas. À quoi cela sert-il que l'ordinateur sache qu'un texte est important ?
Il n'est pas assez intelligent pour comprendre !

Détrompez-vous ! De nombreux programmes analysent le code source des pages web, à commencer par les robots de moteurs de recherche. Ces robots parcourent le Web en lisant le code HTML de tous les sites. C'est le cas des robots de Google et de Bing, par exemple.

Les mots-clés « importants » ont tendance à avoir plus de valeur à leurs yeux, donc si quelqu'un fait une recherche sur ces mots, il a plus de chances de tomber sur votre site. Bien entendu, c'est une explication grossière et il ne faut pas croire qu'utiliser la balise à tout-va améliorera votre référencement. Il faut simplement faire confiance aux ordinateurs : ils comprennent ce qu'un texte « important » veut dire et peuvent se servir de cette information.

Mais alors, comment fait-on pour mettre spécifiquement en gras, pour écrire en rouge, et tout et tout ?

Tout cela se fait en CSS. Souvenez-vous :

- le HTML définit le fond (contenu, logique des éléments) ;
- le CSS définit la forme (apparence).

Nous verrons le CSS plus loin ; pour l'instant nous nous concentrons sur le HTML et ses balises, qui ont chacune un sens particulier.

❖ Les listes

Les listes nous permettent souvent de mieux structurer notre texte et d'ordonner nos informations.
Nous allons découvrir ici deux types de listes :

- les listes non ordonnées ou *listes à puces* ;
- les listes ordonnées ou *listes numérotées*, ou encore *énumérations*.

Liste non ordonnée

Une liste non ordonnée ressemble à ceci :

- Fraises
- Framboises
- Cerises

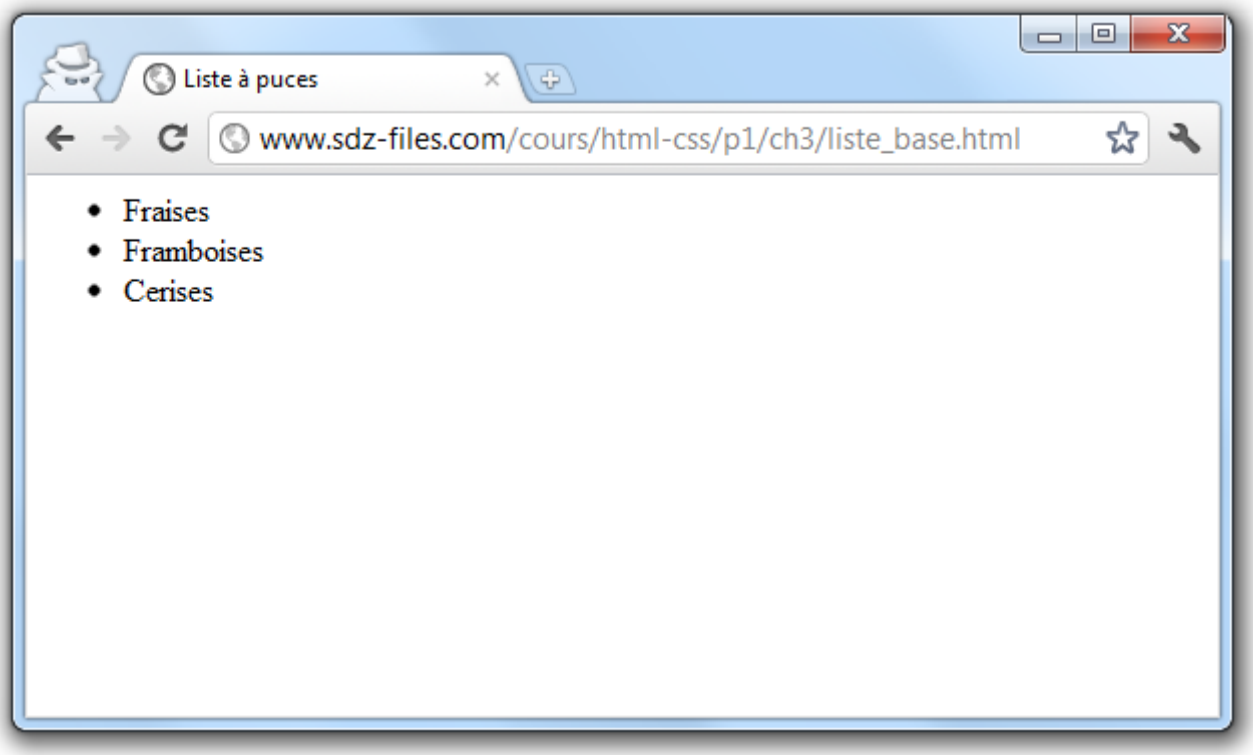
C'est un système qui nous permet de créer une liste d'éléments sans notion d'ordre (il n'y a pas de « premier » ni de « dernier »). Créer une liste non ordonnée est très simple. Il suffit d'utiliser la balise `<ul>` que l'on referme un peu plus loin avec `</ul>` .
Commencez donc à taper ceci :

```
1 <ul></ul>
```

Et maintenant, voilà ce qu'on va faire : on va écrire chacun des éléments de la liste entre deux balises `<li></li>` . Chacune de ces balises doit se trouver entre `<ul>` et `</ul>` . Vous allez comprendre de suite avec cet exemple :

```html
<ul>
    <li>Fraises</li>
    <li>Framboises</li>
    <li>Cerises</li>
</ul>
```

Le résultat se trouve à la figure suivante.



Une liste non ordonnée

Notez que la liste doit être placée à l'intérieur de `<body></body>`. À partir de maintenant, je ne mets pas tout le code de la page pour rester lisible.

Retenez donc ces deux balises :

- `<ul></ul>` délimite toute la liste ;
- `<li></li>` délimite un élément de la liste (une puce).

Vous pouvez mettre autant d'éléments que vous voulez dans la liste à puces, vous n'êtes pas limité à trois éléments.

Et voilà, vous savez créer une liste à puces ! Pas si dur une fois qu'on a compris comment imbriquer les balises.

Pour ceux qui ont besoin de faire des listes complexes, sachez que vous pouvez *imbriquer* des listes à puces (créer une liste à puces **dans** une liste à puces). Si vous voulez faire ça, ouvrez une seconde balise `<ul>` **à l'intérieur** d'un élément `<li></li>` .
Si vous fermez les balises dans le bon ordre, vous n'aurez pas de problème.
Attention néanmoins, cette technique est un peu compliquée à maîtriser.

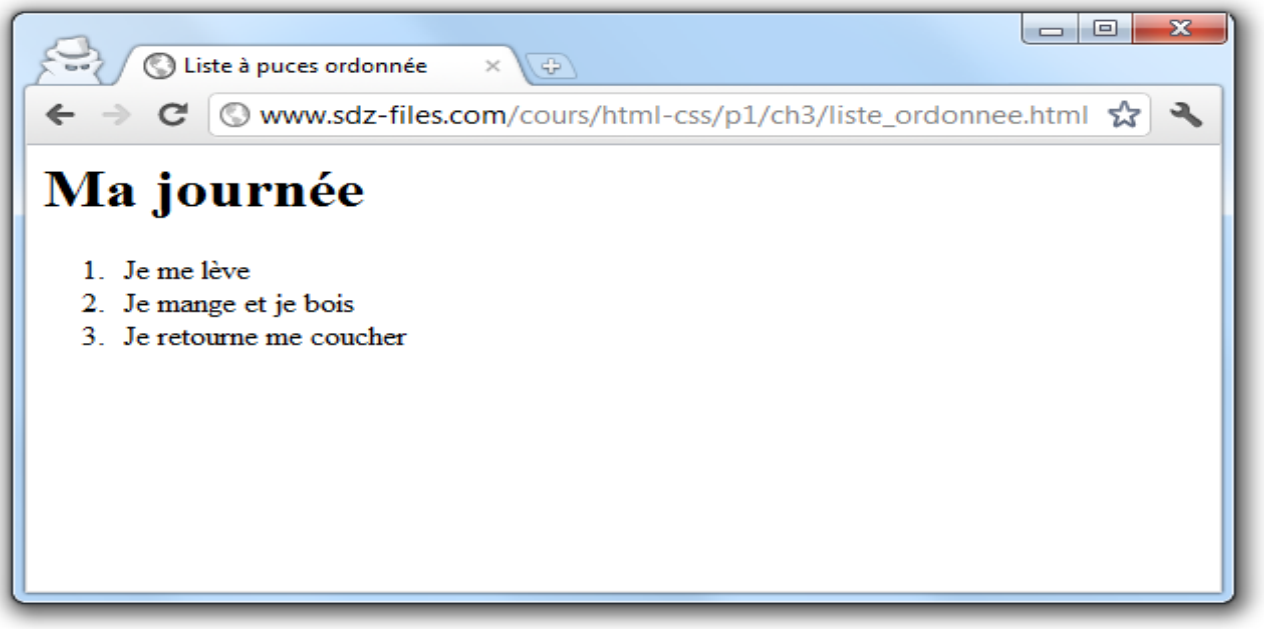
Liste ordonnée

Une liste ordonnée fonctionne de la même façon, seule une balise change : il faut remplacer `<ul></ul>` par `<ol></ol>` .
À l'intérieur de la liste, on ne change rien : on utilise toujours des balises `<li></li>` pour délimiter les éléments.

L'ordre dans lequel vous placez les éléments de la liste est important. Le premier `<li></li>` sera l'élément n° 1, le second sera le n° 2 ,etc...

Comme c'est particulièrement intuitif, je vous laisse admirer la simplicité de cet exemple (résultat à la figure suivante) :

```
<h1>Ma journée</h1>

<ol>
    <li>Je me lève.</li>
    <li>Je mange et je bois.</li>
    <li>Je retourne me coucher.</li>
</ol>
```



Une liste ordonnée

Par rapport à l'exemple précédent, tout ce qu'on a eu à changer est donc la balise `<ol>`

Pour information, il existe un troisième type de liste, beaucoup plus rare : la liste de définitions. Elle fait intervenir les balises `<dl>` (pour délimiter la liste), `<dt>` (pour délimiter un terme) et `<dd>` (pour délimiter la définition de ce terme).

En résumé

- Le HTML comporte de nombreuses balises qui nous permettent d'organiser le texte de notre page. Ces balises donnent des indications comme « Ceci est un paragraphe », « Ceci est un titre », etc.
- Les paragraphes sont définis par la balise `<p> </p>` , et les sauts de ligne par la balise `<br />` .
- Il existe six niveaux de titre, de `<h1> </h1>` à `<h6> </h6>` , à utiliser selon l'importance du titre.
- On peut mettre en valeur certains mots avec les balises `<strong>` , `<em>` et `<mark>` .
- Pour créer des listes, on doit utiliser la balise `<ul>` (liste à puces, non ordonnée) ou `<ol>` (liste ordonnée). À l'intérieur, on insère les éléments avec une balise `<li>` pour chaque item.

5.6 Créez des liens

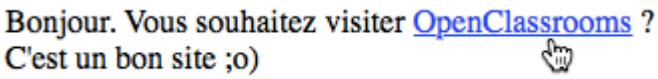
Comme vous le savez, un site web est composé de plusieurs pages. Comment faire pour aller d'une page vers une autre ? À l'aide d'un lien, pardi ! Dans ce chapitre, nous allons justement apprendre à créer des liens entre nos pages.

Je suppose que chacun d'entre vous sait ce qu'est un lien : il s'agit d'un texte sur lequel on peut cliquer pour se rendre sur une autre page.
On peut faire un lien d'une page a.html vers une page b.html , mais on peut aussi faire un lien vers un autre site (par exemple, http://www.siteduzero.com). Dans les deux cas, nous allons voir que le fonctionnement est le même.

Un lien vers un autre site

Il est facile de reconnaître les liens sur une page : ils sont écrits d'une façon différente (par défaut, en bleu et soulignés), et un curseur en forme de main apparaît lorsqu'on pointe dessus.

Je vous propose d'essayer de coder le lien qui amène vers OpenClassrooms, comme à la figure suivante.



Lien vers OpenClassrooms

Pour faire un lien, la balise que nous allons utiliser est très simple à retenir : `<a>` . Il faut cependant lui ajouter un attribut, `href` , pour indiquer vers quelle page le lien doit conduire.

Par exemple, le code ci-dessous est un lien qui amène vers OpenClassrooms, situé à l'adresse `https://openclassrooms.com` :

```
<a href="https://openclassrooms.com">OpenClassrooms</a>
```

Nous allons placer ce lien au sein d'un paragraphe. Voici donc comment reproduire l'exemple de la figure précédente :

```html
<p>Bonjour. Vous souhaitez visiter <a href="https://openclassrooms.com">OpenClassrooms</a> ?<br />
C'est un bon site ;o)</p>
```

Par défaut, le lien s'affiche en bleu souligné. Si vous avez déjà ouvert la page, le lien s'affiche en violet.
Nous verrons comment changer cette apparence lorsque nous étudierons le CSS.

Si vous voulez faire un lien vers un autre site, il suffit donc de copier son adresse (on parle d'*URL*) en `http://` . Notez que certains liens commencent parfois par `https://` (sites sécurisés, comme OpenClassrooms) ou d'autres préfixes (`ftp://` ...).

Si vous faites un lien vers un site qui comporte une adresse un peu bizarre avec des &, comme `http://www.site.com/?data=15&name=mateo21` , vous devrez remplacer tous les « & » par « & » dans votre lien, comme ceci : `http://www.site.com/?data=15&name=mateo21` .
Vous ne verrez pas la différence, mais cela est nécessaire pour avoir une page web correctement construite en HTML5.

Les liens que nous venons de voir sont appelés ***liens absolus***, car on indique l'adresse complète. Nous allons maintenant voir que l'on peut écrire les liens d'une façon un peu différente, ce qui va nous être utile pour faire des liens entre les pages de notre site.

Un lien vers une autre page de son site

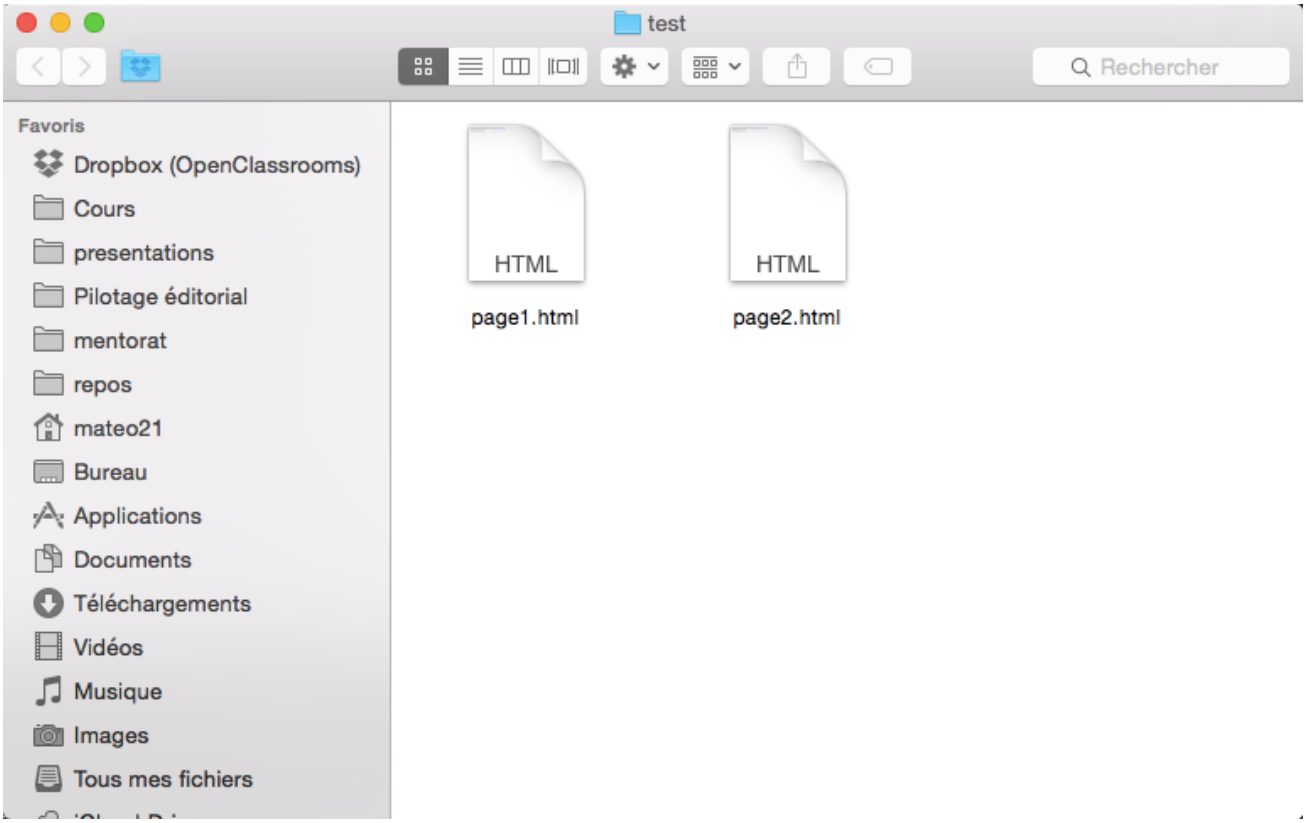
Nous venons d'apprendre à créer des liens vers des sites existants. Mais je suis sûr que vous aimeriez faire des liens entre les différentes pages de votre site, non ?

Oui, justement, comment je fais pour faire un lien vers une autre page de mon site ? Je ne connais pas son adresse en `http://...` , je commence à peine à créer mon site, là ! Je n'ai pas d'adresse.

En effet, pour le moment, vous êtes en train de créer votre site sur votre ordinateur. Vous êtes le seul à pouvoir le voir, et il n'a pas encore « d'adresse web » qui commence en `http://` comme la plupart des sites. Heureusement, cela ne va pas nous empêcher de travailler.

Deux pages situées dans un même dossier

Pour commencer, nous allons créer deux fichiers correspondant à deux pages HTML différentes. Comme je suis très inspiré, je vous propose de les appeler `page1.html` et `page2.html` . Nous aurons donc ces deux fichiers sur notre disque *dans le même dossier* (figure suivante).



Plusieurs fichiers HTML dans un même dossier

Comment faire un lien de la page 1 vers la page 2, sans avoir d'adresse en `http://` ? En fait, c'est facile : si les deux fichiers sont situés dans le même dossier, il suffit d'écrire comme cible du lien le nom du fichier vers lequel on veut amener. Par exemple : `<a href="page2.html">` . On dit que c'est un ***lien relatif***.

Voici le code que nous allons utiliser dans nos fichiers `page1.html` et `page2.html` .

`page1.html`

```html
<p>Bonjour. Souhaitez-vous consulter <a href="page2.html">la page 2</a> ?</p>
```

`page2.html`

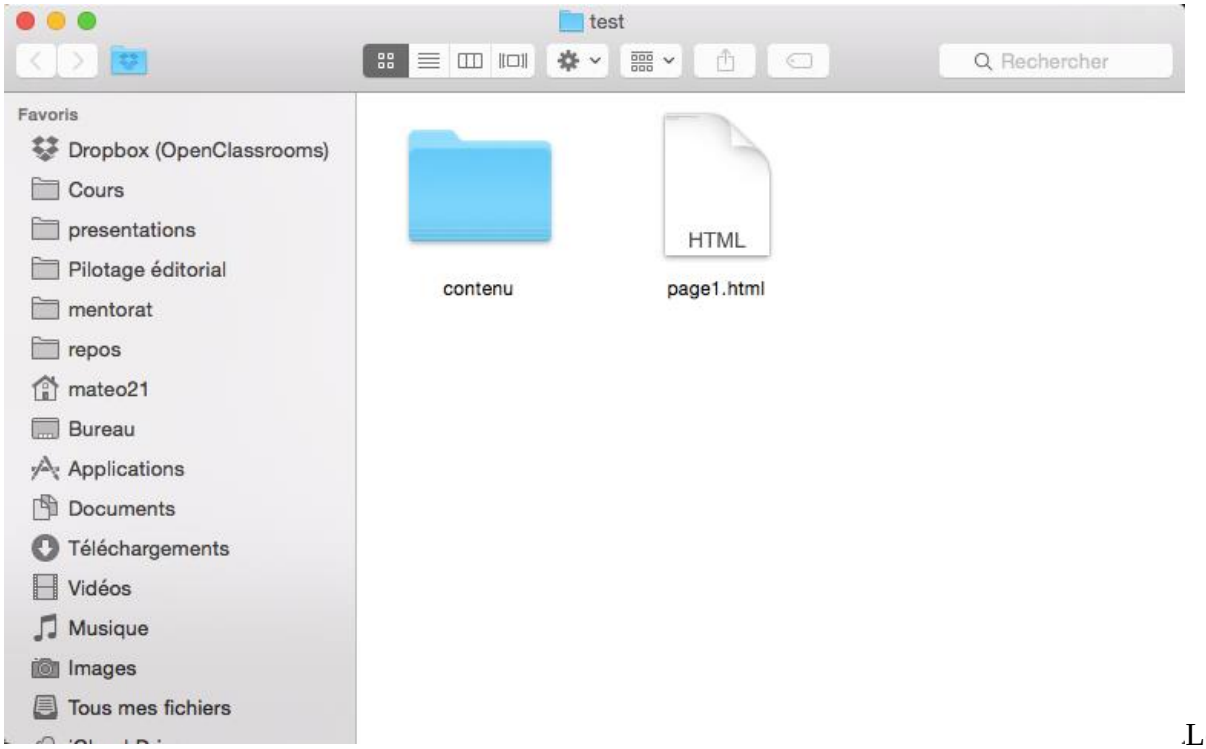
La page 2 (page d'arrivée) affichera simplement un message pour indiquer que l'on est bien arrivé sur la page 2 :

```html
<h1>Bienvenue sur la page 2 !</h1>
```

Deux pages situées dans des dossiers différents

Les choses se corsent un petit peu si les pages sont situées dans des dossiers différents. Idéalement, elles ne devraient pas être trop loin l'une de l'autre (dans un sous-dossier, par exemple).

Imaginons que `page2.html` se trouve dans un sous-dossier appelé `contenu` , comme à la figure suivante.



Le fichier page2.html se trouve à l'intérieur du dossier Contenu

Dans ce cas de figure, le lien doit être rédigé comme ceci :

```
1 <a href="contenu/page2.html">
```

S'il y avait plusieurs sous-dossiers, on écrirait ceci :

html

```
1 <a href="contenu/autredossier/page2.html">
```

Et si le fichier ne se trouve pas dans un sous-dossier, mais dans un dossier « parent », on fait comment ?

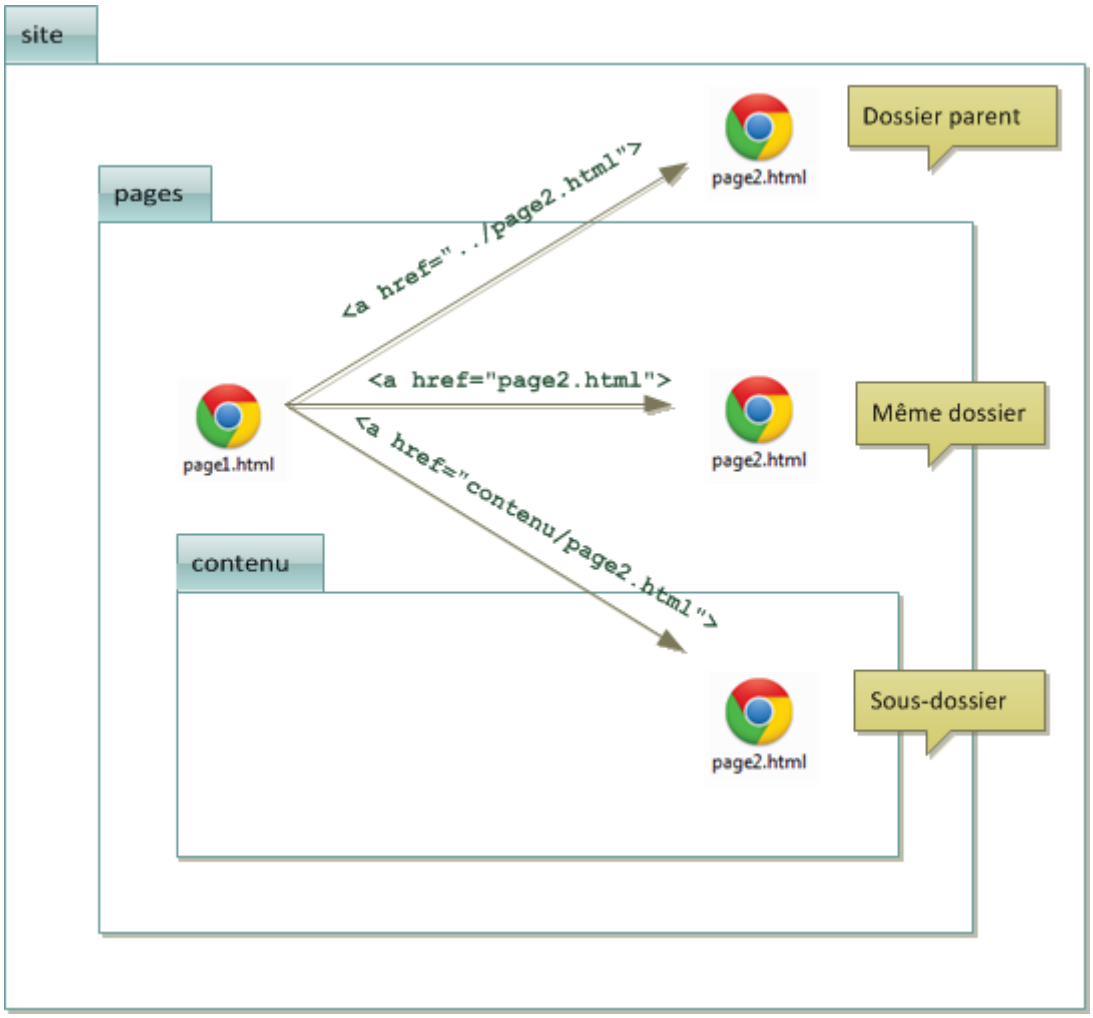
Si votre fichier cible est placé dans un dossier qui se trouve « plus haut » dans l'arborescence, il faut écrire deux points, comme ceci :

html

```
1 <a href="../page2.html">
```

Résumé en images

Les liens relatifs ne sont pas bien compliqués à utiliser une fois qu'on a compris le principe. Il suffit de regarder dans quel « niveau de dossier » se trouve votre fichier cible, pour savoir comment écrire votre lien. La figure suivante fait la synthèse des différents liens relatifs possibles.



Les différents liens relatifs

Un lien vers une ancre

Une **ancre** est une sorte de point de repère que vous pouvez mettre dans vos pages HTML lorsqu'elles sont très longues.
En effet, il peut alors être utile de faire un lien amenant plus bas dans la même page, pour que le visiteur puisse sauter directement à la partie qui l'intéresse.

Pour créer une ancre, il suffit de rajouter l'attribut `id` à une balise qui va alors servir de repère. Ce peut être n'importe quelle balise, un titre, par exemple.
Utilisez l'attribut `id` pour donner un nom à l'ancre. Cela nous servira ensuite pour faire un lien vers cette ancre. Par exemple :

html

```html
<a href="#mon_ancre">Aller vers l'ancre</a>
```

Normalement, si vous cliquez sur le lien, cela vous amènera plus bas dans la même page (à condition que la page comporte suffisamment de texte pour que les barres de défilement se déplacent automatiquement).
Voici un exemple de page comportant beaucoup de texte et utilisant les ancres (j'ai mis n'importe quoi dans le texte pour remplir) :

html

```html
<h1>Ma grande page</h1>

<p>
    Aller directement à la partie traitant de :<br />
    <a href="#cuisine">La cuisine</a><br />
    <a href="#rollers">Les rollers</a><br />
    <a href="#arc">Le tir à l'arc</a><br />
</p>
<h2 id="cuisine">La cuisine</h2>

<p>... (beaucoup de texte) ...</p>

<h2 id="rollers">Les rollers</h2>

<p>... (beaucoup de texte) ...</p>

<h2 id="arc">Le tir à l'arc</h2>

<p>... (beaucoup de texte) ...</p>
```

S'il ne se passe rien quand vous cliquez sur les liens, c'est qu'il n'y a pas assez de texte. Dans ce cas, vous pouvez soit rajouter du blabla dans la page pour qu'il y ait (encore) plus de texte, soit réduire la taille de la fenêtre de votre navigateur pour faire apparaître les barres de défilement sur le côté.

L'attribut `id` sert à donner un nom « unique » à une balise, pour s'en servir de repère. Et, croyez-moi, vous n'avez pas fini d'entendre parler de cet attribut. Ici, on s'en sert pour faire un lien vers une ancre mais, en CSS, il nous sera très utile pour « repérer » une balise précise, vous verrez.
Évitez cependant de créer des id avec des espaces ou des caractères spéciaux ; utilisez simplement, dans la mesure du possible, des lettres et des chiffres pour que la valeur soit reconnue par tous les navigateurs.

Lien vers une ancre située dans une autre page

Alors là, je vous préviens, on va faire le Mégamix !

L'idée, c'est de faire un lien qui ouvre une autre page ET qui amène directement à une ancre située plus bas sur cette page.
En pratique, c'est assez simple à faire : il suffit de taper le nom de la page, suivi d'un dièse (`#`), suivi du nom de l'ancre.

Par exemple : `<a href="ancres.html#rollers">`

... vous amènera sur la page `ancres.html` , directement au niveau de l'ancre appelée `rollers` .

Voici une page qui contient trois liens, chacun amenant vers une des ancres de la page de l'exemple précédent :

html

```html
<h1>Le Mégamix</h1>
<p>
    Rendez-vous quelque part sur une autre page :<br />
    <a href="ancres.html#cuisine">La cuisine</a><br />
    <a href="ancres.html#rollers">Les rollers</a><br />
    <a href="ancres.html#arc">Le tir à l'arc</a><br />
</p>
```

Un lien qui ouvre une nouvelle fenêtre

Il est possible de « forcer » l'ouverture d'un lien dans une nouvelle fenêtre. Pour cela, on rajoutera `target="_blank"` à la balise `<a>` :

html

```html
<p>Bonjour. Souhaitez-vous visiter <a href="https://openclassrooms.com" title="Vous ne le regretterez pas !" target="_blank">OpenClassrooms</a> ?</p>
```

Selon la configuration du navigateur, la page s'affichera dans une nouvelle fenêtre ou un nouvel onglet. Vous ne pouvez pas choisir entre l'ouverture d'une nouvelle fenêtre ou d'un nouvel onglet.

Notez cependant qu'il est déconseillé d'abuser de cette technique, car elle perturbe la navigation. Le visiteur lui-même peut décider s'il veut ouvrir le lien dans une nouvelle fenêtre. Il fera `Maj` + Clic sur le lien pour ouvrir dans une nouvelle fenêtre ou `Ctrl` + Clic pour ouvrir dans un nouvel onglet.

Un lien pour envoyer un e-mail

Si vous voulez que vos visiteurs puissent vous envoyer un e-mail, vous pouvez utiliser des liens de type `mailto` . Rien ne change au niveau de la balise, vous devez simplement modifier la valeur de l'attribut `href` comme ceci :

```html
<p><a href="mailto:votrenom@bidule.com">Envoyez-moi un e-mail !</a></p>
```

Il suffit donc de faire commencer le lien par `mailto:` et d'écrire l'adresse e-mail où on peut vous contacter. Si vous cliquez sur le lien, un nouveau message vide s'ouvre, prêt à être envoyé à votre adresse e-mail.

Un lien pour télécharger un fichier

Beaucoup d'entre vous se demandent comment cela se passe pour le téléchargement d'un fichier... En fait, il faut procéder exactement comme si vous faisiez un lien vers une page web, mais en indiquant cette fois le nom du fichier à télécharger.

Par exemple, supposez que vous vouliez faire télécharger `monfichier.zip` . Placez simplement ce fichier dans le même dossier que votre page web (ou dans un sous-dossier) et faites un lien vers ce fichier :

```html
<p><a href="monfichier.zip">Télécharger le fichier</a></p>
```

C'est tout ! Le navigateur, voyant qu'il ne s'agit pas d'une page web à afficher, va lancer la procédure de téléchargement lorsqu'on cliquera sur le lien.

En résumé

- Les liens permettent de changer de page et sont, par défaut, écrits en bleu et soulignés.
- Pour insérer un lien, on utilise la balise `<a>` avec l'attribut `href` pour indiquer l'adresse de la page cible. Exemple : `<a href="https://openclassrooms.com">` .
- On peut faire un lien vers une autre page de son site, simplement en écrivant le nom du fichier : `<a href="page2.html">` .
- Les liens permettent aussi d'amener vers d'autres endroits sur la même page. Il faut créer une ancre avec l'attribut `id` pour « marquer » un endroit dans la page, puis faire un lien vers l'ancre comme ceci : `<a href="#ancre">` .

5.7 Insérez des images

Insérer une image dans une page web ? Vous allez voir, c'est d'une facilité déconcertante... Enfin presque. Il existe différents **formats** d'images que l'on peut utiliser sur des sites web, et on ne doit pas les choisir au hasard. En effet, les images sont parfois volumineuses à télécharger, ce qui ralentit le temps de chargement de la page (beaucoup plus que le texte !).

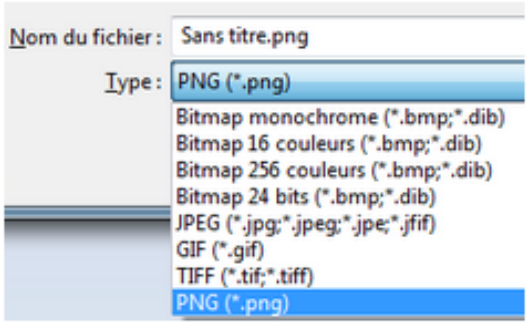
Pour faire en sorte que vos pages restent lisibles et rapides à télécharger, suivez donc activement mes conseils !

❖ Les différents formats d'images

Savez-vous ce qu'est un format d'image ?

Quand vous avez une image « entre les mains », vous avez la possibilité de l'enregistrer dans plusieurs « formats » différents. Le poids (en Ko, voire en Mo) de l'image sera plus ou moins élevé selon le format choisi, et la qualité de l'image va changer.

Par exemple, le logiciel de dessin Paint (même si c'est loin d'être le meilleur) vous propose de choisir entre plusieurs formats lorsque vous enregistrez une image (figure suivante).



Différents formats d'image proposés par Paint

Certains formats sont plus adaptés que d'autres selon l'image (photo, dessin, image animée...). Notre but ici est de faire le tour des différents formats utilisés sur le Web, pour que vous les connaissiez et sachiez choisir celui qui convient le mieux à votre image. Rassurez-vous, il n'y a pas beaucoup de formats différents, cela ne sera donc pas bien long.

Toutes les images diffusées sur Internet ont un point commun : elles sont **compressées**. Cela veut dire que l'ordinateur fait des calculs pour qu'elles soient moins lourdes et donc plus rapides à charger.

Le JPEG

Les images au format JPEG (*Joint Photographic Expert Group*) sont très répandues sur le Web. Ce format est conçu pour réduire le poids des photos (c'est-à-dire la taille du fichier associé), qui peuvent comporter plus de 16 millions de couleurs différentes. La figure suivante est une photo enregistrée au format JPEG.

Une photo de montagne en JPEG

Les images JPEG sont enregistrées avec l'extension `.jpg` ou `.jpeg` .

Notez que le JPEG détériore un peu la qualité de l'image, d'une façon généralement imperceptible. C'est ce qui le rend si efficace pour réduire le poids des photos.
Quand il s'agit d'une photo, on ne peut généralement pas détecter la perte de qualité. Par contre, si ce n'est pas une photo, vous risquez de voir l'image un peu « baver ». Dans ce cas, il vaut mieux utiliser le format PNG.

Le PNG

Le format PNG (*Portable Network Graphics*) est le plus récent de tous. Ce format est adapté à la plupart des graphiques (je serais tenté de dire « à tout ce qui n'est pas une photo »). Le PNG a deux gros avantages : il peut être rendu transparent et il n'altère pas la qualité de l'image.

Le PNG a été inventé pour concurrencer un autre format, le GIF, à l'époque où il fallait payer des royalties pour pouvoir utiliser des GIF. Depuis, le PNG a bien évolué et c'est devenu le format le plus puissant pour enregistrer la plupart des images.

Le PNG existe en deux versions, en fonction du nombre de couleurs que doit comporter l'image :

- PNG 8 bits : 256 couleurs ;
- PNG 24 bits : 16 millions de couleurs (autant qu'une image JPEG).

La figure suivante est une image PNG en 24 bits, représentant Zozor, qui sera notre mascotte tout au long de ce cours.

Zozor en PNG

Au fait, si le PNG 24 bits peut afficher autant de couleurs qu'une image JPEG, et qu'en plus il peut être rendu transparent sans modifier la qualité de l'image... quel est l'intérêt du JPEG ?

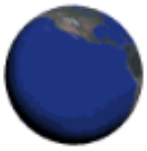
La compression du JPEG est plus puissante sur les photos. Une photo enregistrée en JPEG se chargera toujours beaucoup plus vite que si elle était enregistrée en PNG. Je vous conseille donc toujours de réserver le format JPEG aux photos.

Le GIF

C'est un format assez vieux, qui a été néanmoins très utilisé (et qui reste très utilisé par habitude). Aujourd'hui, le PNG est globalement bien meilleur que le GIF : les images sont généralement plus légères et la transparence est de meilleure qualité. Je vous recommande donc d'utiliser le PNG autant que possible.

Le format GIF est limité à 256 couleurs (alors que le PNG peut aller jusqu'à plusieurs millions de couleurs).

Néanmoins, le GIF conserve un certain avantage que le PNG n'a pas : il peut être animé. D'où l'explosion ces dernières années des GIF animés sur le web (aussi appelés "*reaction gifs*").



Un GIF animé

Il existe un format adapté à chaque image

Si on résume, voici quel format adopter en fonction de l'image que vous avez :

- **Une photo** : utilisez un JPEG.
- **N'importe quel graphique avec peu de couleurs** (moins de 256) : utilisez un PNG 8 bits, ou éventuellement un GIF.
- **N'importe quel graphique avec beaucoup de couleurs** : utilisez un PNG 24 bits.
- **Une image animée** : utilisez un GIF animé.

Les erreurs à éviter

Bannissez les autres formats

Les autres formats non cités ici, comme le format BITMAP (`*.bmp`) sont à bannir car bien souvent ils ne sont pas compressés, donc trop gros. Ils ne sont pas du tout adaptés au Web. On peut en mettre sur son site, mais le chargement sera vraiment *extrêmement* long !

Choisissez bien le nom de votre image

Si vous voulez éviter des problèmes, prenez l'habitude d'enregistrer vos fichiers avec des noms en minuscules, sans espace ni accent ; par exemple : `mon_image.png` .

Vous pouvez remplacer les espaces par le caractère *underscore* (« _ »), comme je l'ai fait ici.

5.8 Insérer une image

Revenons maintenant au code HTML pour découvrir comment placer des images dans nos pages web !

Insertion d'une image

Quelle est la fameuse balise qui va nous permettre d'insérer une image ? Il s'agit de... `<img />` !

C'est une balise **orpheline** (comme `<br />`). Cela veut dire qu'on n'a pas besoin de l'écrire en deux exemplaires, comme la plupart des autres balises que nous avons vues jusqu'ici. En effet, nous n'avons pas besoin de délimiter une portion de texte, nous voulons juste insérer une image à un endroit précis.

La balise doit être accompagnée de deux attributs obligatoires :

- `src` : il permet d'indiquer où se trouve l'image que l'on veut insérer. Vous pouvez soit mettre un chemin absolu (ex. : `http://www.site.com/fleur.png`), soit mettre le chemin en relatif (ce qu'on fait le plus souvent). Ainsi, si votre image est dans un sous-dossier `images`, vous devrez taper : `src="images/fleur.png"`
- `alt` : cela signifie « texte alternatif ». On doit *toujours* indiquer un texte alternatif à l'image, c'est-à-dire un court texte qui décrit ce que contient l'image. Ce texte sera affiché à la place de l'image si celle-ci ne peut pas être téléchargée (cela arrive), ou dans les navigateurs de personnes handicapées (non-voyants) qui ne peuvent malheureusement pas « voir » l'image. Cela aide aussi les robots des moteurs de recherche pour les recherches d'images. Pour la fleur, on mettrait par exemple : `alt="Une fleur"`

Les images doivent se trouver obligatoirement à l'intérieur d'un paragraphe (`<p></p>`). Voici un exemple d'insertion d'image :

html

```
<p>
    Voici une photo que j'ai prise lors de mes dernières vacances à la montagne :<br />
    <img src="images/montagne.jpg" alt="Photo de montagne" />
</p>
```

Bref, l'insertion d'image est quelque chose de très facile pour peu qu'on sache indiquer où se trouve l'image, comme on avait appris à le faire avec les liens.
La plus grosse « difficulté » (si on peut appeler cela une difficulté) consiste à choisir le bon format d'image. Ici, c'est une photo, donc c'est évidemment le format JPEG qu'on utilise.

Je le répète : évitez à tout prix les accents, majuscules et espaces dans vos noms de fichiers et de dossiers. Voici un chemin qui va poser problème : `"Images du site/Image toute bête.jpg"` .
Il faudrait supprimer les espaces (ou les remplacer par le symbole « _ »), supprimer les accents et tout mettre en minuscules, comme ceci : `"images_du_site/image_toute_bete.jpg"` .
Sachez donc que, si votre image ne s'affiche pas, c'est très certainement parce que le chemin est incorrect ! Simplifiez au maximum vos noms de fichiers et de dossiers, et tout ira bien.

Ajouter une infobulle

L'attribut permettant d'afficher une bulle d'aide est le même que pour les liens : il s'agit de `title` . Cet attribut est facultatif (contrairement à `alt`).

Voici ce que cela peut donner :

```html
<p>
    Voici une photo que j'ai prise lors de mes dernières vacances à la montagne :<br />
    <img src="images/montagne.jpg" alt="Photo de montagne" title="C'est beau les Alpes quand même !" />
</p>
```

Survolez la photo avec la souris pour voir l'infobulle apparaître.

Miniature cliquable

Si votre image est très grosse, il est conseillé d'en afficher la miniature sur votre site. Ajoutez ensuite un lien sur cette miniature pour que vos visiteurs puissent afficher l'image en taille originale.

De nombreux sites permettent de redimensionner des images, comme ResizeImage.net, par exemple. Je vais ainsi disposer de deux versions de ma photo (comme à la figure suivante) : la miniature et l'image d'origine.

La miniature et son image d'origine

Je les place toutes les deux dans un dossier appelé par exemple `img`. J'affiche la version `montagne_mini.jpg` sur ma page et je fais un lien vers `montagne.jpg`, pour que l'image agrandie s'affiche lorsqu'on clique sur la miniature.

Voici le code HTML que je vais utiliser pour cela :

```html
<p>
    Vous souhaitez voir l'image dans sa taille d'origine ? Cliquez dessus !<br />
    <a href="img/montagne.jpg"><img src="img/montagne_mini.jpg" alt="Photo de montagne" title="Cliquez pour agrandir" /></a>
</p>
```

Parfois, certains navigateurs choisissent d'afficher un cadre bleu (ou violet) pas très esthétique autour de votre image cliquable.
Heureusement, nous pourrons retirer ce cadre dans peu de temps grâce au CSS.

❖ Les figures

Au cours de la lecture de ce cours, vous avez déjà rencontré plusieurs fois des **figures**. Ce sont des éléments qui viennent enrichir le texte pour compléter les informations de la page.

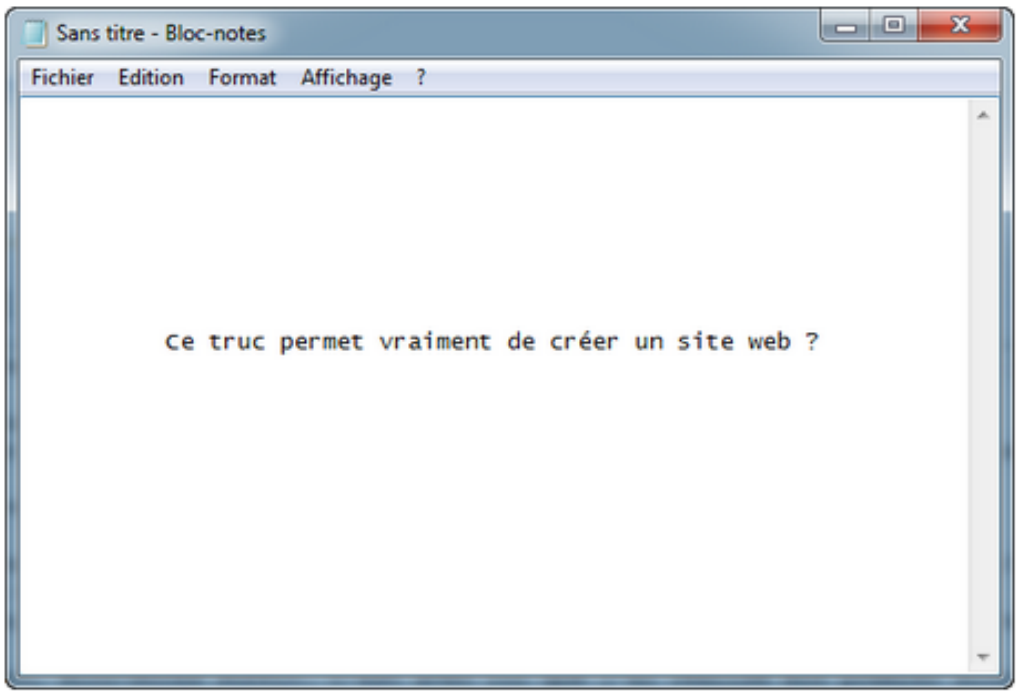
Les figures peuvent être de différents types :

- images ;
- codes source ;
- citations ;
- etc.

Bref, tout ce qui vient *illustrer* le texte est une figure. Nous allons ici nous intéresser aux images mais, contrairement à ce qu'on pourrait croire, les figures ne sont pas *forcément* des images : un code source aussi illustre le texte.

❖ Création d'une figure

Reprenons par exemple cette capture d'écran du premier chapitre, représentée à la figure suivante.



Le logiciel Bloc-Notes

En HTML5, on dispose de la balise `<figure>` . Voici comment on pourrait l'utiliser :

```html
<figure>
    <img src="images/blocnotes.png" alt="Bloc-Notes" />
</figure>
```

Une figure est le plus souvent accompagnée d'une légende. Pour ajouter une légende, utilisez la balise `<figcaption>` à l'intérieur de la balise `<figure>` , comme ceci :

```html
<figure>
    <img src="images/blocnotes.png" alt="Bloc-Notes" />
    <figcaption>Le logiciel Bloc-Notes</figcaption>
</figure>
```

En HTML5, on dispose de la balise `<figure>` . Voici comment on pourrait l'utiliser :

```html
<figure>
    <img src="images/blocnotes.png" alt="Bloc-Notes" />
</figure>
```

Si vous faites de votre image une figure, l'image peut être située en-dehors d'un paragraphe.

```html
<p>Connaissez-vous le logiciel Bloc-Notes ? On peut faire des sites web avec !</p>

<figure>
    <img src="images/blocnotes.png" alt="Bloc-Notes" />
    <figcaption>Le logiciel Bloc-Notes</figcaption>
</figure>
```

Je ne vois pas vraiment de changement. Quand dois-je placer mon image dans un paragraphe et quand dois-je la placer dans une figure ?

Bonne question ! Tout dépend de ce que votre image apporte au texte :

- Si elle n'apporte aucune information (c'est juste une illustration pour décorer) : placez l'image dans un paragraphe.
- Si elle apporte une information : placez l'image dans une figure.

La balise `<figure>` a un rôle avant tout **sémantique**. Cela veut dire qu'elle indique à l'ordinateur que l'image a du sens et qu'elle est importante pour la bonne compréhension du texte. Cela peut permettre à un programme de récupérer toutes les figures du texte et de les référencer dans une table des figures, par exemple.

Enfin, sachez qu'une figure peut très bien comporter plusieurs images. Voici un cas où cela se justifie :

```html
<figure>
    <img src="images/internetexplorer.png" alt="Logo Internet Explorer" />
    <img src="images/firefox.png" alt="Logo Mozilla Firefox" />
    <img src="images/chrome.png" alt="Logo Google Chrome" />
    <figcaption>Logos des différents navigateurs</figcaption>
</figure>
```

En résumé

- Il existe plusieurs formats d'images adaptées au Web :
  - JPEG : pour les photos ;
  - PNG : pour toutes les autres illustrations ;
  - GIF : similaire au PNG, plus limité en nombre de couleurs mais qui peut être animé.
- On insère une image avec la balise `<img />`. Elle doit obligatoirement comporter au moins ces deux attributs : `src` (nom de l'image) et `alt` (courte description de l'image).
- Si une image illustre le texte (et n'est pas seulement décorative), il est conseillé de la placer au sein d'une balise `<figure>`. La balise `<figcaption>` permet d'écrire la légende de l'image.

5.9 Le CSS

Après avoir passé toute une première partie du cours à ne travailler que sur le HTML, nous allons maintenant découvrir le CSS que j'avais volontairement mis à l'écart. Le CSS n'est pas plus compliqué que le HTML. Il vient le compléter pour vous aider à mettre en forme votre page web.

Dans ce premier chapitre sur le CSS, nous allons voir la théorie sur le CSS : qu'est-ce que c'est ? À quoi cela ressemble-t-il ? Où est-ce qu'on écrit du code CSS ? Ces aspects théoriques ne sont pas bien compliqués, mais vous devez obligatoirement les connaître car c'est la base du CSS. C'est d'ailleurs la seule chose que je vous demanderai de retenir par cœur en CSS, vous pourrez retrouver le reste dans le mémo en annexe.

Je vous avais averti dès le début de ce cours : nous allons apprendre deux langages. Nous avons déjà bien entamé notre découverte du HTML, même s'il reste encore de nombreuses choses à apprendre (nous y reviendrons dans quelques chapitres). En revanche, il est temps maintenant de nous intéresser au CSS.

CSS (*Cascading Style Sheets*), c'est cet autre langage qui vient compléter le HTML.
Vous vous souvenez de son rôle ? *Gérer la mise en forme de votre site.*

Petit rappel : à quoi sert CSS ?

CSS ? C'est lui qui vous permet de choisir la couleur de votre texte.
Lui qui vous permet de sélectionner la police utilisée sur votre site.
Lui encore qui permet de définir la taille du texte, les bordures, le fond...

Et aussi, c'est lui qui permet de faire la mise en page de votre site. Vous pourrez dire : je veux que mon menu soit à gauche et occupe telle largeur, que l'en-tête de mon site soit calé en haut et qu'il soit toujours visible, etc.

Où écrit-on le CSS ?

Vous avez le choix car on peut écrire du code en langage CSS à trois endroits différents :

- dans un fichier `.css` (*méthode la plus recommandée*) ;
- dans l'en-tête `<head>` du fichier HTML ;
- directement dans les balises du fichier HTML *via* un attribut `style` (*méthode la moins recommandée*).

Dans un fichier `.css` (recommandé)

Comme je viens de vous le dire, on écrit le plus souvent le code CSS dans un fichier spécial ayant l'extension `.css` (contrairement aux fichiers HTML qui ont l'extension `.html`). C'est la méthode la plus pratique et la plus souple. Cela nous évite de tout mélanger dans un même fichier. J'utiliserai cette technique dans toute la suite de ce cours.

Commençons à pratiquer dès maintenant ! Nous allons partir du fichier HTML suivant :

html

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" />
        <title>Premiers tests du CSS</title>
    </head>

    <body>
        <h1>Mon super site</h1>

        <p>Bonjour et bienvenue sur mon site !</p>
        <p>Pour le moment, mon site est un peu <em>vide</em>. Patientez encore un peu !</p>
    </body>
</html>
```

Vous noterez le contenu de la ligne 5, `<link rel="stylesheet" href="style.css" />` : c'est elle qui indique que ce fichier HTML est associé à un fichier appelé `style.css` et chargé de la mise en forme.

Enregistrez ce fichier sous le nom que vous voulez (par exemple, `page.html`). Pour le moment, rien d'extraordinaire à part la nouvelle balise que nous avons ajoutée.

Maintenant, créez un *nouveau* fichier vide dans votre éditeur de texte (par exemple Sublime Text) et copiez-y ce bout de code CSS (rassurez-vous, je vous expliquerai tout à l'heure ce qu'il veut dire) :

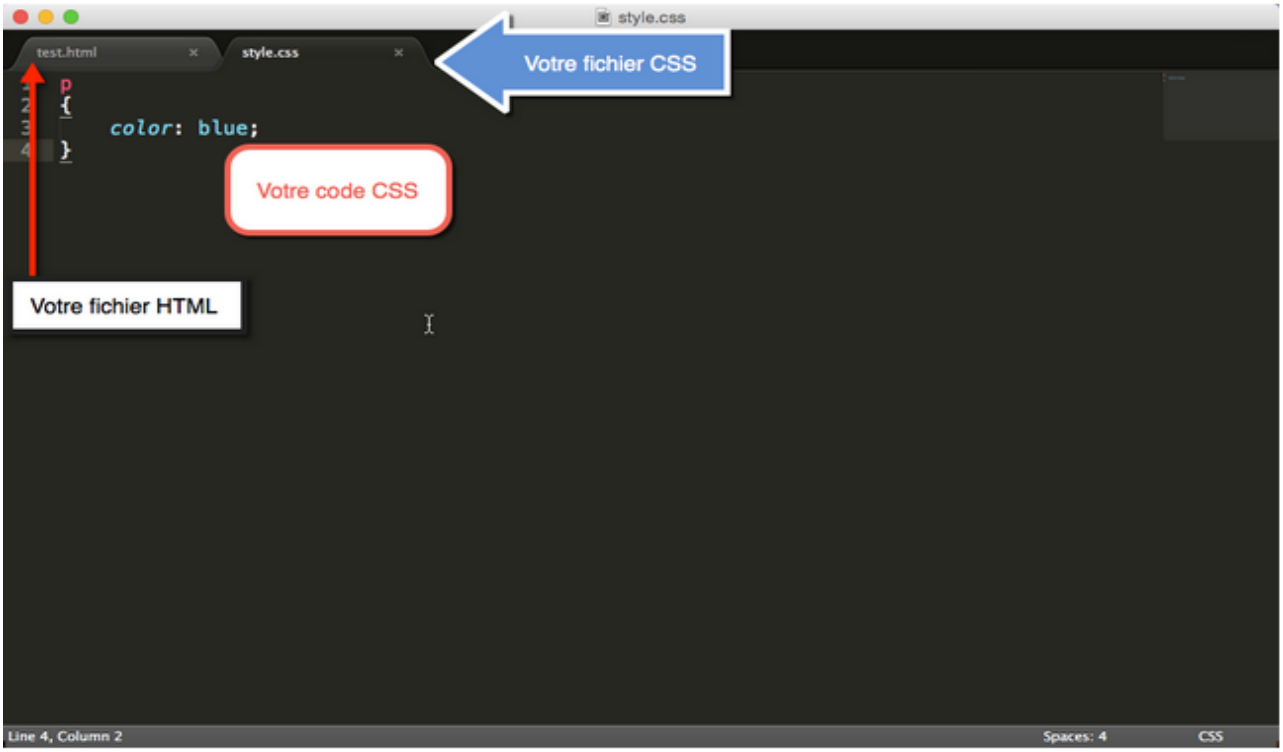
css

```css
p
{
    color: blue;
}
```

Pour obtenir la coloration du code dans Sublime Text, enregistrez bien votre fichier avec l'extension `.css` d'abord.

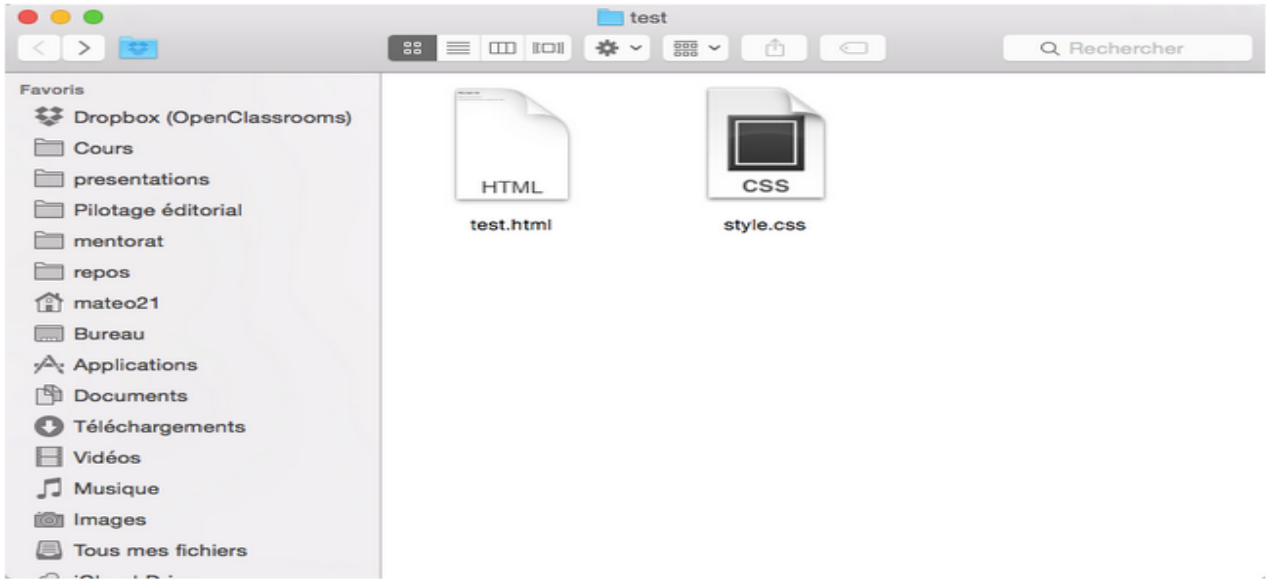
Enregistrez le fichier en lui donnant un nom qui se termine par `.css` , comme `style.css` . Placez ce fichier `.css` dans le même dossier que votre fichier `.html` .

Dans Sublime Text, vous devriez observer quelque chose de similaire à la figure suivante.



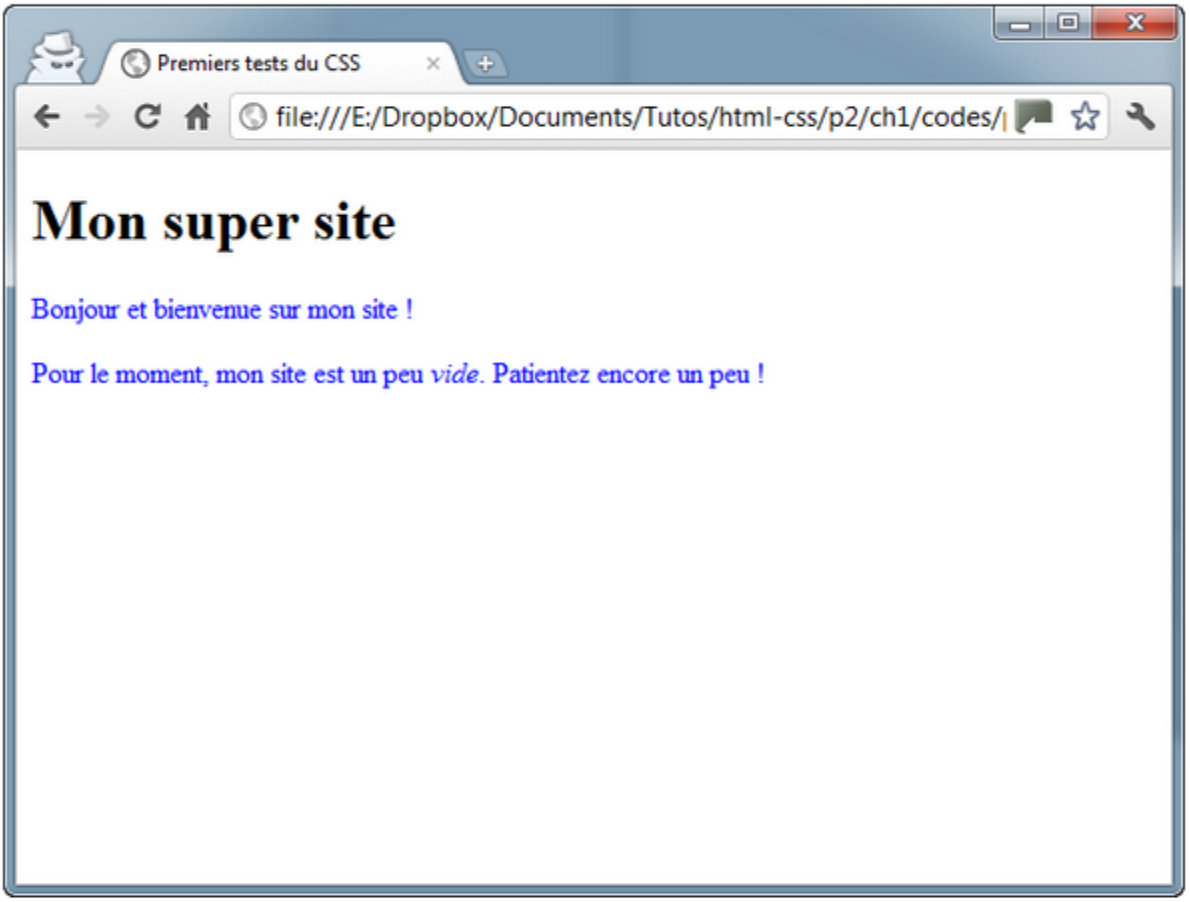
Fichiers HTML et CSS dans Sublime Text

Dans votre explorateur de fichiers, vous devriez les voir apparaître côte à côte. D'un côté le `.html` , de l'autre le `.css` , comme à la figure suivante.



Fichiers HTML et CSS dans l'explorateur de fichiers

Ouvrez maintenant votre fichier `page.html` dans votre navigateur pour le tester, comme vous le faites d'habitude. Regardez, c'est magique : vos paragraphes sont écrits en bleu, comme dans la figure suivante !



Le texte est écrit en bleu

Il est inutile d'ouvrir directement le fichier `style.css` dans le navigateur. Il faut ouvrir le fichier `page.html` (il fera automatiquement appel au fichier `style.css`).

Dans l'en-tête `<head>` du fichier HTML

Il existe une autre méthode pour utiliser du CSS dans ses fichiers HTML : cela consiste à insérer le code CSS directement dans une balise `<style>` à l'intérieur de l'en-tête `<head>` .

Voici comment on peut obtenir exactement le même résultat avec un seul fichier `.html` qui contient le code CSS (lignes 5 à 10) :

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <style>
            p
            {
                color: blue;
            }
        </style>
        <title>Premiers tests du CSS</title>
    </head>

    <body>
        <h1>Mon super site</h1>

        <p>Bonjour et bienvenue sur mon site !</p>
        <p>Pour le moment, mon site est un peu <em>vide</em>. Patientez encore un peu !</p>
    </body>
</html>
```

Testez, vous verrez que le résultat est le même.

❖ Appliquer un style : sélectionner une balise

Maintenant que nous savons où placer le code CSS, intéressons-nous de plus près à ce code. Je vous ai donné, sans vous l'expliquer, un premier bout de code CSS :

```css
p
{
    color: blue;
}
```

Dans un code CSS comme celui-ci, on trouve trois éléments différents :

- **des noms de balises** : on écrit les noms des balises dont on veut modifier l'apparence. Par exemple, si je veux modifier l'apparence de tous les paragraphes `<p>` , je dois écrire `p` ;
- **des propriétés CSS** : les « effets de style » de la page sont rangés dans des propriétés. Il y a par exemple la propriété `color` qui permet d'indiquer la couleur du texte, `font-size` qui permet d'indiquer la taille du texte, etc. Il y a beaucoup de propriétés CSS et, comme je vous l'ai dit, je ne vous obligerai pas à les connaître toutes par cœur ;
- **les valeurs** : pour chaque propriété CSS, on doit indiquer une valeur. Par exemple, pour la propriété `color` , il faut indiquer le nom de la couleur. Pour `font-size` , il faut indiquer quelle taille on veut, etc.

Schématiquement, une feuille de style CSS ressemble donc à cela :

```css
balise1
{
    propriete1: valeur1;
    propriete2: valeur2;
    propriete3: valeur3;
}

balise2
{
    propriete1: valeur1;
    propriete2: valeur2;
    propriete3: valeur3;
    propriete4: valeur4;
}

balise3
{
    propriete1: valeur1;
}
```

Vous repérez dans cet extrait de code les balises, propriétés et valeurs dont je viens de vous parler.

Essayez de changer le nom de la balise affectée par le code CSS. Par exemple, si j'écris `h1` , c'est le titre qui sera écrit en bleu. Modifiez votre fichier `style.css` comme ceci :

```css
h1
{
    color: blue;
}
```

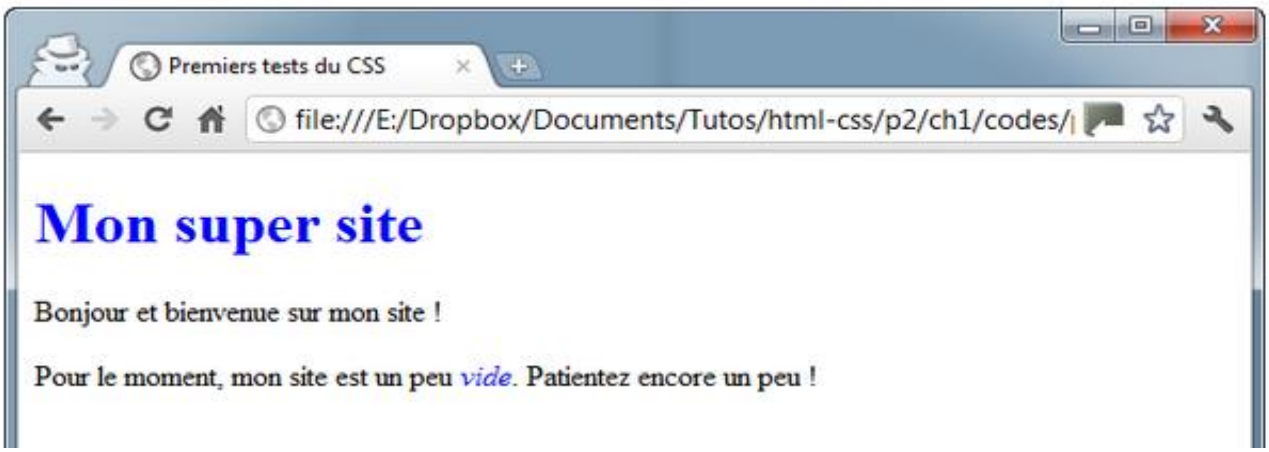
Maintenant, ouvrez à nouveau votre page HTML (souvenez-vous, c'est la page HTML qu'on ouvre dans le navigateur, pas le fichier CSS !) : vous devriez voir son titre s'afficher en bleu

```
h1
{
    color: blue;
}

em
{
    color: blue;
}
```

Il signifie que nos titres `<h1>` et nos textes importants `<em>` doivent s'afficher en bleu. Par contre, c'est un peu répétitif, vous ne trouvez pas ?

Heureusement, il existe un moyen en CSS d'aller plus vite si les deux balises doivent avoir la même présentation. Il suffit de combiner la déclaration en séparant les noms des balises par une virgule, comme ceci :

```css
h1, em
{
    color: blue;
}
```



Cela signifie : « *Je veux que le texte de mes* `<h1>` *et* `<em>` *soit écrit en bleu* ».

Vous pouvez indiquer autant de balises à la suite que vous le désirez.

❖ Appliquer un style : class et id

Ce que je vous ai montré jusqu'ici a quand même un défaut : cela implique par exemple que TOUS les paragraphes possèdent la même présentation (ici, ils seront donc tous écrits en bleu).
Comment faire pour que certains paragraphes seulement soient écrits d'une manière différente ? On pourrait placer le code CSS dans un attribut `style` sur la balise que l'on vise (c'est la technique que je vous ai présentée un peu plus tôt) mais, comme je vous l'ai dit, ce n'est pas recommandé (il vaut mieux utiliser un fichier CSS externe).

Pour résoudre le problème, on peut utiliser ces attributs spéciaux *qui fonctionnent sur toutes les balises* :

- l'attribut `class` ;
- l'attribut `id` .

Que les choses soient claires dès le début : les attributs `class` et `id` sont quasiment identiques. Il y a seulement une petite différence que je vous dévoilerai plus bas.
Pour le moment, et pour faire simple, on ne va s'intéresser qu'à l'attribut `class` .

Comme je viens de vous le dire, c'est un attribut que l'on peut mettre sur n'importe quelle balise, aussi bien titre que paragraphe, image, etc.

```html
<h1 class=""> </h1>
<p class=""> </p>
<img class="" />
```

Oui, mais que met-on comme valeur à l'attribut `class` ?

En fait, vous devez écrire un nom qui sert à identifier la balise. Ce que vous voulez, du moment que le nom commence par une lettre.

Par exemple, je vais associer la classe `introduction` à mon premier paragraphe (ligne 12) :

html

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" />
        <title>Premiers tests du CSS</title>
    </head>

    <body>
        <h1>Mon super site</h1>

        <p class="introduction">Bonjour et bienvenue sur mon site !</p>
        <p>Pour le moment, mon site est un peu <em>vide</em>. Patientez encore un peu !</p>
    </body>
</html>
```

Maintenant que c'est fait, votre paragraphe est identifié. Il a un nom : `introduction` . Vous allez pouvoir réutiliser ce nom dans le fichier CSS pour dire : « *Je veux que seules les balises qui ont comme nom 'introduction' soient affichées en bleu* ».

Pour faire cela en CSS, indiquez le nom de votre classe en commençant par un point, comme ci-dessous :

css

```css
.introduction
{
    color: blue;
}
```

Testez le résultat : seul votre paragraphe appelé `introduction` va s'afficher en bleu (figure suivante) !

Et l'attribut `id` , alors ?

Lui, il fonctionne exactement de la même manière que `class` , à un détail près : il ne peut être utilisé *qu'une fois* dans le code.

Quel intérêt ? Il y en a assez peu pour tout vous dire ; cela vous sera utile si vous faites du JavaScript plus tard, pour reconnaître certaines balises. D'ailleurs, nous avons déjà vu l'attribut `id` dans le chapitre sur les liens (pour réaliser des ancres). En pratique, nous ne mettrons des `id` que sur des éléments qui sont uniques dans la page, comme par exemple le logo :

html

```
<img src="images/logo.png" alt="Logo du site" id="logo">
```

Si vous utilisez des `id` , lorsque vous définirez leurs propriétés dans le fichier CSS, il faudra faire précéder le nom de l' `id` par un dièse (#) :

css

```
#logo
{
    /* Indiquez les propriétés CSS ici */
}
```

Je ne vous propose pas de le tester, cela fonctionne exactement comme `class` .

Si vous vous emmêlez les pinceaux entre `class` et `id` , retenez que deux balises peuvent avoir le même nom avec l'attribut `class` . Un nom d' `id` doit en revanche être unique dans la page HTML.

❖ Les balises universelles

Il arrivera parfois que vous ayez besoin d'appliquer une `class` (ou un `id`) à certains mots qui, à l'origine, ne sont pas entourés par des balises.

En effet, le problème de `class` , c'est qu'il s'agit d'un attribut. Vous ne pouvez donc en mettre que sur une balise. Si, par exemple, je veux modifier uniquement « bienvenue » dans le paragraphe suivant :

html

```
<p>Bonjour et bienvenue sur mon site !</p>
```

Cela serait facile à faire s'il y avait une balise autour de « bienvenue » , mais malheureusement, il n'y en a pas. Par chance, on a inventé… la balise-qui-ne-sert-à-rien.

En fait, on a inventé deux balises dites ***universelles***, qui n'ont aucune signification particulière (elles n'indiquent pas que le mot est important, par exemple). Il y a une différence minime (mais significative !) entre ces deux balises :

- `<span> </span>` : c'est une balise de type **inline**, c'est-à-dire une balise que l'on place au sein d'un paragraphe de texte pour sélectionner certains mots uniquement. Les balises `<strong>` et `<em>` sont de la même famille. Cette balise s'utilise donc au milieu d'un paragraphe et c'est celle dont nous allons nous servir pour colorer « bienvenue » ;
- `<div> </div>` : c'est une balise de type **block**, qui entoure un bloc de texte. Les balises `<p>` , `<h1>` , etc., sont de la même famille. Ces balises ont quelque chose en commun : elles créent un nouveau « bloc » dans la page et provoquent donc obligatoirement un retour à la ligne. `<div>` est une balise fréquemment utilisée dans la construction d'un design, comme nous le verrons plus tard.

Pour le moment donc, nous allons utiliser plutôt la balise `<span>` . On la met autour de « bienvenue », on lui ajoute une classe (du nom qu'on veut), on crée le CSS et c'est gagné !

html

```html
<p>Bonjour et <span class="salutations">bienvenue</span> sur mon site !</p>
```

css

```css
.salutations
{
    color: blue;
}
```

Vous pouvez voir le résultat à la figure suivante.

En résumé

- CSS est un autre langage qui vient compléter le HTML. Son rôle est de mettre en forme votre page web.
- Il faut être vigilant sur la compatibilité des navigateurs avec certaines fonctionnalités récentes de CSS3. Quand un navigateur ne connaît pas une instruction de mise en forme, il l'ignore simplement.
- On peut écrire le code CSS à plusieurs endroits différents, le plus conseillé étant de créer un fichier séparé portant l'extension `.css` (exemple : `style.css`).
- En CSS, on sélectionne quelles portions de la page HTML on veut modifier, et on change leur présentation avec des propriétés CSS :

```css
balise1
{
    propriete1: valeur1;
    propriete2: valeur2;
}
```

- Il existe de nombreuses façons de sélectionner la portion de la page que l'on veut mettre en forme. Par exemple, on peut viser :
  - toutes les balises d'un même type, en écrivant simplement leur nom (`h1` , par exemple) ;
  - certaines balises spécifiques, auxquelles on a donné des noms à l'aide des attributs `class` ou `id` (`.nomclasse` ou `#nomid`) ;
  - uniquement les balises qui se trouvent à l'intérieur d'autres balises (`h3 em`).
  - etc.

❖ La taille

Pour modifier la taille du texte, on utilise la propriété CSS `font-size` . Mais comment indiquer la taille du texte ? C'est là que les choses se corsent, car plusieurs techniques vous sont proposées :

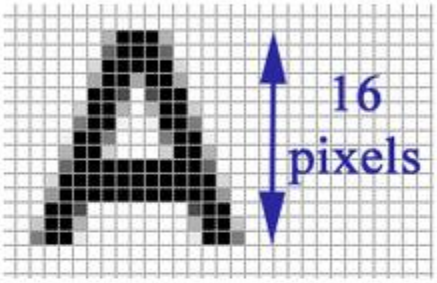
- indiquer une **taille absolue** : en pixels, en centimètres ou millimètres. Cette méthode est très précise, mais il est conseillé de ne l'utiliser que si c'est absolument nécessaire, car on risque d'indiquer une taille trop petite pour certains lecteurs ;
- indiquer une **taille relative** : en pourcentage, « em » ou « ex », cette technique a l'avantage d'être plus souple. Elle s'adapte plus facilement aux préférences de taille des visiteurs.

Une taille absolue

Pour indiquer une taille absolue, on utilise généralement les pixels. Pour avoir un texte de 16 pixels de hauteur, vous devez donc écrire :

```css
font-size: 16px;
```

Les lettres auront une taille de 16 pixels, comme le montre la figure suivante.



Une lettre de 16 pixels de hauteur

Voici un exemple d'utilisation (placez ce code dans votre fichier `.css`) :

```css
p
{
    font-size: 14px; /* Paragraphes de 14 pixels */
}
h1
{
    font-size: 40px; /* Titres de 40 pixels */
}
```

Et le résultat est visible à la figure suivante.

Différentes tailles de texte

Si vous le souhaitez, vous pouvez également définir des tailles en centimètres ou millimètres. Remplacez « px » par « cm » ou « mm ». Ces unités sont cependant moins bien adaptées aux écrans.

Une valeur relative

C'est la méthode recommandée, car le texte s'adapte alors plus facilement aux préférences de tous les visiteurs.

Il y a plusieurs moyens d'indiquer une valeur relative. Vous pouvez par exemple écrire la taille avec des mots en anglais comme ceux-ci :

- `xx-small` : minuscule ;
- `x-small` : très petit ;
- `small` : petit ;
- `medium` : moyen ;
- `large` : grand ;
- `x-large` : très grand ;
- `xx-large` : euh... gigantesque.

Vous pouvez tester l'utilisation de ces valeurs dans votre code CSS :

```css
p
{
    font-size: small;
}
h1
{
    font-size: large;
}
```

Bon, cette technique a un défaut : il n'y a que sept tailles disponibles (car il n'y a que sept noms). Heureusement, il existe d'autres moyens. Ma technique préférée consiste à indiquer la taille en « em ».

- Si vous écrivez `1em`, le texte a une taille normale.
- Si vous voulez grossir le texte, vous pouvez inscrire une valeur supérieure à 1, comme `1.3em`.
- Si vous voulez réduire le texte, inscrivez une valeur inférieure à 1, comme `0.8em`.

Faites attention : pour les nombres décimaux, il faut mettre un point et non une virgule. Vous devez donc écrire « 1.4em » et non pas « 1,4em » !

Exemple :

```css
p
{
    font-size: 0.8em;
}
h1
{
    font-size: 1.3em;
}
```

D'autres unités sont disponibles. Vous pouvez essayer le « ex » (qui fonctionne sur le même principe que le em, mais qui est plus petit de base) et le pourcentage (80 %, 130 %...).

La police

Ah... la police... On touche un point sensible.

En effet, il se pose un problème : pour qu'une police s'affiche correctement, il faut que tous les internautes l'aient. Si un internaute n'a pas la même police que vous, son navigateur prendra une police par défaut (une police standard) qui n'aura peut-être rien à voir avec ce à quoi vous vous attendiez.

La bonne nouvelle, c'est que depuis CSS 3, il est possible de faire télécharger automatiquement une police au navigateur. Je vous expliquerai dans un second temps comment faire cela.

Modifier la police utilisée

La propriété CSS qui permet d'indiquer la police à utiliser est `font-family` . Vous devez écrire le nom de la police comme ceci :

```css
balise
{
    font-family: police;
}
```

La propriété CSS qui permet d'indiquer la police à utiliser est `font-family` . Vous devez écrire le nom de la police comme ceci :

```css
balise
{
    font-family: police;
}
```

Seulement, pour éviter les problèmes si l'internaute n'a pas la même police que vous, on précise en général *plusieurs* noms de police, séparés par des virgules :

```css
balise
{
    font-family: police1, police2, police3, police4;
}
```

Le navigateur essaiera d'abord d'utiliser la `police1` . S'il ne l'a pas, il essaiera la `police2` . S'il ne l'a pas, il passera à la `police3` , et ainsi de suite.
En général, on indique en tout dernier `serif` , ce qui correspond à une police par défaut (qui ne s'applique que si aucune autre police n'a été trouvée).

Il existe aussi une autre police par défaut appelée `sans-serif` . La différence entre les deux est la présence de petites pattes de liaison en bas des lettres, que la police `sans-serif` n'a pas. Oui, c'est subtil.

Oui, mais quelles sont les polices les plus courantes qu'on a le « droit » d'utiliser, me direz-vous ?
Voici une liste de polices qui fonctionnent bien sur la plupart des navigateurs :

- Arial ;
- Arial Black ;
- Comic Sans MS ;
- Courier New ;
- Georgia ;
- Impact ;
- Times New Roman ;
- Trebuchet MS ;
- Verdana.

Ainsi, si j'écris :

```css
p
{
    font-family: Impact, "Arial Black", Arial, Verdana, sans-serif;
}
```

... cela signifie : « *Mets la police Impact ou, si elle n'y est pas, Arial Black, ou sinon Arial, ou sinon Verdana, ou si rien n'a marché, mets une police standard (Sans Serif)* ».

En général, il est bien d'indiquer un choix de trois ou quatre polices (+ Serif ou Sans Serif), afin de s'assurer qu'au moins l'une d'entre elles aura été trouvée sur l'ordinateur du visiteur.

Si le nom de la police comporte des espaces, je conseille de l'entourer de guillemets, comme je l'ai fait pour « Arial Black ».

❖ Italique, gras, souligné, …

Il existe en CSS une série de propriétés classiques de mise en forme du texte. Nous allons découvrir ici comment afficher le texte en gras, italique, souligné... et au passage nous verrons qu'il est même possible d'aller jusqu'à le faire clignoter !

Mettre en italique

Attends un peu là ! Je croyais que la balise `<em>` permettait de mettre un texte en italique ?!

Je n'ai jamais dit cela.
Retournez voir les chapitres précédents si vous avez des doutes, mais je n'ai *jamais* dit que la balise `<em>` était faite pour mettre le texte en italique (de même que je n'ai jamais dit que `<strong>` était faite pour mettre en gras).

`<em>` , mettez-vous bien cela dans la tête, est faite pour insister sur des mots. Cela veut dire que les mots qu'elle entoure sont assez importants.
Pour représenter cette importance, la plupart des navigateurs choisissent d'afficher le texte en italique, mais ce n'est pas une obligation.

Le CSS lui, permet de dire réellement : « Je veux que ce texte soit en italique ». Rien ne vous empêche, par exemple, de décider que tous vos titres seront en italique.

Concrètement, en CSS, pour mettre en italique, on utilise `font-style` qui peut prendre trois valeurs :

- `italic` : le texte sera mis en italique ;
- `oblique` : le texte sera passé en oblique (les lettres sont penchées, le résultat est légèrement différent de l'italique proprement dit) ;
- `normal` : le texte sera normal (par défaut). Cela vous permet d'annuler une mise en italique. Par exemple, si vous voulez que les textes entre `<em>` ne soient plus en italique, vous devrez écrire :

```css
em
{
    font-style: normal;
}
```

Ainsi, dans l'exemple suivant, je me sers de `font-style` pour mettre en italique tous mes titres `<h2>` :

```css
h2
{
    font-style: italic;
}
```

Mettre en gras

Et si nous passions à la mise en gras ?
Alors, là encore, n'oubliez pas que ce n'est pas `<strong>` qui permet de mettre en gras (son rôle est d'indiquer que le texte est important, *donc* le navigateur l'affiche généralement en gras). La mise en gras en CSS peut par exemple s'appliquer aux titres, à certains paragraphes entiers, etc. C'est à vous de voir.

La propriété CSS pour mettre en gras est `font-weight` et prend les valeurs suivantes :

- `bold` : le texte sera en gras ;
- `normal` : le texte sera écrit normalement (par défaut).

Voici par exemple comment écrire les titres en gras :

```css
h1
{
    font-weight: bold;
}
```

Soulignement et autres décorations

La propriété CSS associée porte bien son nom : `text-decoration` . Elle permet, entre autres, de souligner le texte, mais pas seulement. Voici les différentes valeurs qu'elle peut prendre :

- `underline` : souligné ;
- `line-through` : barré ;
- `overline` : ligne au-dessus ;
- `none` : normal (par défaut).

Ce CSS va vous permettre de tester les effets de `text-decoration` :

```css
h1
{
    text-decoration: blink;
}
.souligne
{
    text-decoration: underline;
}
.barre
{
    text-decoration: line-through;
}
.ligne_dessus
{
    text-decoration: overline;
}
```

Et le résultat est visible à la figure suivante.

❖ L’alignement

Le langage CSS nous permet de faire tous les alignements connus : à gauche, centré, à droite et justifié.

C'est tout simple. On utilise la propriété `text-align` et on indique l'alignement désiré :

- `left` : le texte sera aligné à gauche (c'est le réglage par défaut) ;
- `center` : le texte sera centré ;
- `right` : le texte sera aligné à droite ;
- `justify` : le texte sera « justifié ». Justifier le texte permet de faire en sorte qu'il prenne toute la largeur possible sans laisser d'espace blanc à la fin des lignes. Les textes des journaux, par exemple, sont toujours justifiés.

Regardez les différents alignements sur cet exemple :

```css
h1
{
    text-align: center;
}

p
{
    text-align: justify;
}

.signature
{
    text-align: right;
}
```

Le résultat est visible sur la figure suivante.

Vous ne pouvez pas modifier l'alignement du texte d'une balise *inline* (comme `<span>` , `<a>` , `<em>` , `<strong>` ...). L'alignement ne fonctionne que sur des balises de type *block* (`<p>` , `<div>` , `<h1>` , `<h2>` ...) et c'est un peu logique, quand on y pense : on ne peut pas modifier l'alignement de quelques mots au milieu d'un paragraphe !
C'est donc en général le paragraphe entier qu'il vous faudra aligner.

En résumé

- On modifie la taille du texte avec la propriété CSS `font-size` . On peut indiquer la taille en pixels (16px), en « em » (1.3em), en pourcentage (110 %), etc.
- On change la police du texte avec `font-family` . Attention, seules quelques polices sont connues par tous les ordinateurs. Vous pouvez cependant utiliser une police personnalisée avec la directive `@font-face` : cela forcera les navigateurs à télécharger la police de votre choix.
- De nombreuses propriétés de mise en forme du texte existent : `font-style` pour l'italique, `font-weight` pour la mise en gras, `text-decoration` pour le soulignement, etc.
- Le texte peut être aligné avec `text-align` .

❖ Couleur et fond

Couleur du texte

Vous connaissez déjà la propriété qui permet de modifier la couleur du texte : il s'agit de `color` . Nous allons nous intéresser aux différentes façons d'indiquer la couleur, car il y en a plusieurs.

Indiquer le nom de la couleur

La méthode la plus simple et la plus pratique pour choisir une couleur consiste à taper son nom (*in english, of course*).
Le seul défaut de cette méthode est qu'il n'existe que seize couleurs dites « standard ». D'autres couleurs officieuses existent mais, comme elles ne fonctionneront pas forcément de la même manière sur tous les navigateurs, je vais éviter de vous les montrer.

Pour passer tous les titres en bordeaux, on peut donc écrire :

```css
h1
{
    color: maroon;
}
```

Vous trouverez le résultat à la figure suivante.

Seize couleurs, c'est quand même un peu limite quand on sait que la plupart des écrans peuvent en afficher seize *millions*.
D'un autre côté, remarquez, s'il avait fallu donner un nom à chacune des seize millions de couleurs...

Heureusement, il existe en CSS plusieurs façons de choisir une couleur parmi toutes celles qui existent. La première que je vais vous montrer est la notation hexadécimale. Elle est couramment utilisée sur le Web, mais il existe aussi une autre méthode que nous verrons plus loin.

Un nom de couleur en hexadécimal, cela ressemble à : #FF5A28. Pour faire simple, c'est une combinaison de lettres et de chiffres qui indiquent une couleur.
On doit toujours commencer par écrire un dièse (#), suivi de six lettres ou chiffres allant de 0 à 9 et de A à F.

Ainsi, #000000 correspond à la couleur noire et #FFFFFF à la couleur blanche. Mais maintenant, ne me demandez pas quelle est la combinaison qui produit de l'orange couleur « coucher de soleil », je n'en sais strictement rien.

Certains logiciels de dessin, comme Photoshop, **Gimp** et **Paint.NET**, vous indiquent les couleurs en hexadécimal. Il vous est alors facile de copier-coller le code hexadécimal d'une couleur dans votre fichier CSS.

La méthode RGB

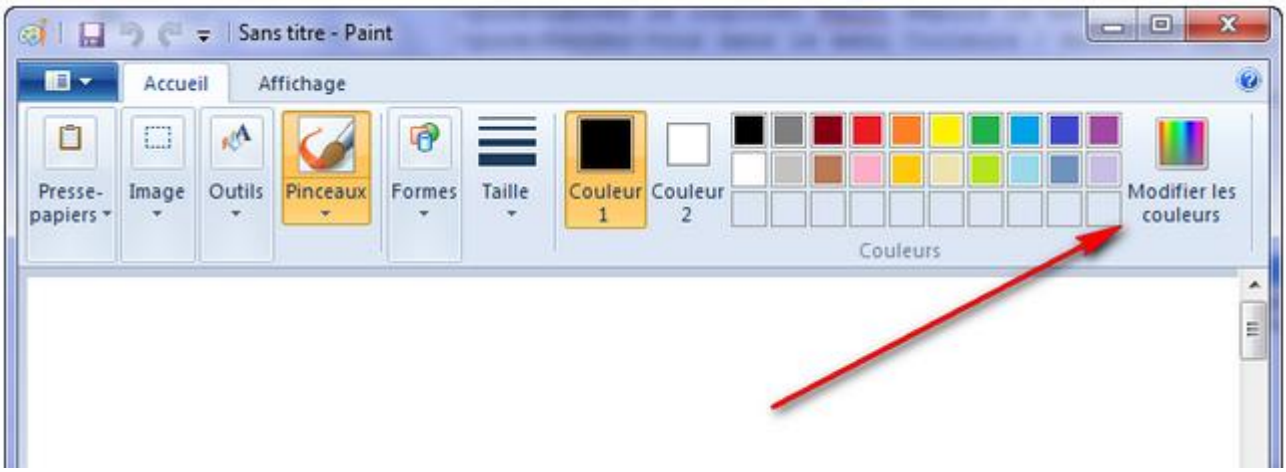
Que signifie RGB ? En anglais, Rouge-Vert-Bleu s'écrit *Red-Green-Blue*, ce qui s'abrège en « RGB ». Comme avec la notation hexadécimale, pour choisir une couleur, on doit définir une quantité de rouge, de vert et de bleu.

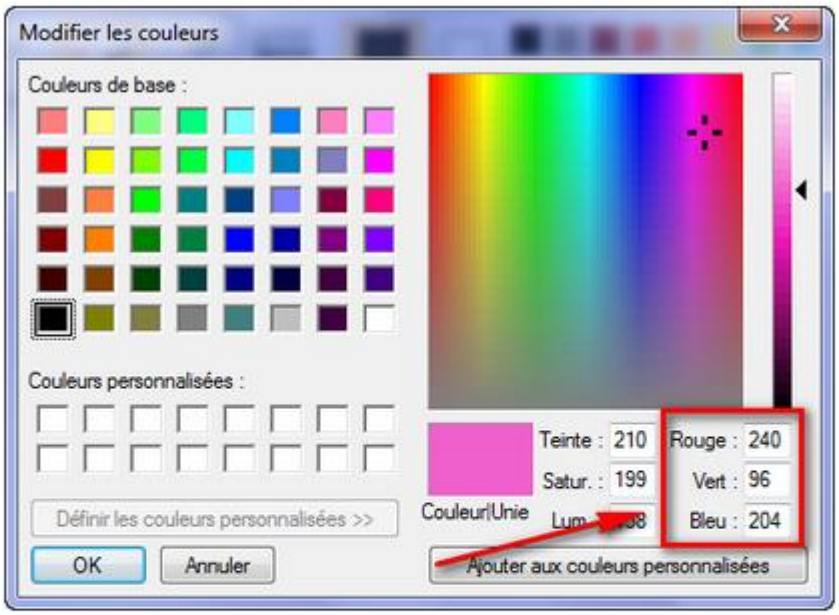
> Encore cette histoire tordue de quantités de rouge-vert-bleu ?

Oui mais là, vous allez voir que c'est beaucoup plus pratique et qu'avec un logiciel de dessin tout simple comme Paint, vous pouvez trouver la couleur que vous désirez. Voici la marche à suivre :

1. Lancez le logiciel Paint depuis le menu `Démarrer` .
2. Rendez-vous dans la section `Modifier les couleurs` , comme indiqué à la figure suivante.
3. Une fenêtre s'ouvre. Dans la zone qui apparaît à droite, faites bouger les curseurs pour sélectionner la couleur qui vous intéresse. Supposons que vous soyez pris d'une envie folle d'écrire vos titres `<h1>` en rose bonbon (supposons seulement). Sélectionnez la couleur dans la fenêtre, comme à la figure suivante.
4. Relevez les quantités de Rouge-Vert-Bleu correspondantes, indiquées en bas à droite de la fenêtre (ici 240-96-204). Recopiez ces valeurs dans cet ordre dans le fichier CSS, comme dans le code ci-dessous.

```css
p
{
    color: rgb(240,96,204);
}
```





Sélection d'une couleur dans Paint

Comme vous avez pu le constater dans l'exemple, pour utiliser la méthode RGB, il faut taper `rgb(Rouge, Vert, Bleu)` en remplaçant « Rouge, Vert, Bleu » par les nombres correspondants. Pour information, ces quantités sont toujours comprises entre 0 et 255.

Et en Bonus Track...

Je vous conseille de taper "Color Picker" sur Google, et vous trouverez plusieurs outils et sites qui vous aident à choisir une couleur.

Par exemple, http://www.colorpicker.com permet de trouver la valeur hexadécimale d'une couleur très facilement :

Couleur de fond

Pour indiquer une couleur de fond, on utilise la propriété CSS `background-color` . Elle s'utilise de la même manière que la propriété `color` , c'est-à-dire que vous pouvez taper le nom d'une couleur, l'écrire en notation hexadécimale ou encore utiliser la méthode RGB.

Pour indiquer la couleur de fond de la page web, il faut travailler sur la balise `<body>` . Eh oui, `<body>` correspond à l'ensemble de la page web ; c'est donc en modifiant sa couleur de fond que l'on changera la couleur d'arrière-plan de la page.

Regardez très attentivement ce fichier CSS :

css

```css
/* On travaille sur la balise body, donc sur TOUTE la page */
body
{
    background-color: black; /* Le fond de la page sera noir */
    color: white; /* Le texte de la page sera blanc */
}
```

Voici le rendu de ce code :

Texte en blanc sur fond noir

Eh, mais tu as demandé à ce que le texte de la balise `<body>` soit écrit en blanc, et tous les paragraphes `<p>` et titres `<h1>` ont pris cette couleur. Comment cela se fait-il ?

Je voulais justement profiter de l'occasion pour vous en parler. Ce phénomène s'appelle **l'*héritage***. Je vous rassure tout de suite, personne n'est mort.

Le CSS et l'héritage

En CSS, si vous appliquez un style à une balise, toutes les balises qui se trouvent à l'intérieur prendront le même style.

C'est en fait simple à comprendre et intuitif. La balise `<body>`, vous le savez, contient entre autres les balises de paragraphe `<p>` et de titre `<h1>`.

Si j'applique une couleur de fond noire et une couleur de texte blanche à la balise `<body>`, tous mes titres et paragraphes auront eux aussi un arrière-plan de couleur noire et un texte de couleur blanche... C'est ce phénomène qu'on appelle *l'héritage* : on dit que les balises qui se trouvent à l'intérieur d'une autre balise « héritent » de ses propriétés.

C'est d'ailleurs de là que vient le nom « CSS », qui signifie « *Cascading Style Sheets* », c'est-à-dire « Feuilles de style en cascade ». Les propriétés CSS sont héritées en cascade : si vous donnez un style à un élément, tous les sous-éléments auront le même style.

Cela veut dire que TOUT le texte de ma page web sera forcément écrit en blanc ?

Non, pas obligatoirement. Si vous dites par la suite que vous voulez vos titres en rouge, ce style aura la priorité et vos titres seront donc en rouge. En revanche, si vous n'indiquez rien de particulier (comme on l'a fait tout à l'heure), alors vos titres hériteront de la couleur blanche.
Cela ne fonctionne pas uniquement pour la couleur, entendons-nous bien. Toutes les propriétés CSS seront héritées : vous pouvez par exemple demander une mise en gras dans la balise `<body>`, et tous vos titres et paragraphes seront en gras.

Exemple d'héritage avec la balise `<mark>`

On a tendance à croire qu'on ne peut modifier que la couleur de fond de la page. C'est faux : vous pouvez changer le fond de n'importe quel élément : vos titres, vos paragraphes, certains mots... Dans ce cas, ils apparaîtront surlignés (comme si on avait mis un coup de marqueur dessus).

Vous vous souvenez par exemple de la balise `<mark>` qui permet de mettre en valeur certains mots ? Utilisons-la à nouveau ici :

html

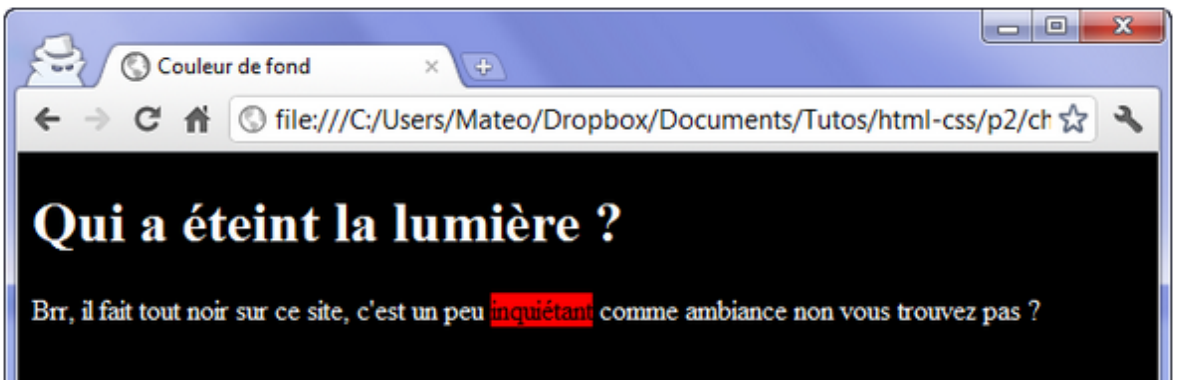
```html
<h1>Qui a éteint la lumière ?</h1>

<p>Brr, il fait tout noir sur ce site, c'est un peu <mark>inquiétant</mark> comme ambiance non vous trouvez pas ?</p>
```

Par défaut, le texte s'affiche sur un fond jaune. Vous pouvez changer ce comportement en CSS :

css

```css
body
{
    background-color: black;
    color: white;
}

mark
{
    /* La couleur de fond prend le pas sur celle de toute la page */
    background-color: red;
    color: black;
}
```

Sur le texte de la balise `<mark>` , c'est la couleur de fond rouge qui s'applique. En effet, même si le fond de la page est noir, c'est la propriété CSS de l'élément le plus précis qui a la priorité (figure suivante).



Le même principe vaut pour toutes les balises HTML et toutes les propriétés CSS ! Si vous dites :

- mes paragraphes ont une taille de 1.2 em ;
- mes textes importants (`<strong>`) ont une taille de 1.4 em ;

... on pourrait penser qu'il y a un conflit. Le texte important fait partie d'un paragraphe, quelle taille lui donner ? 1.2 em ou 1.4 em ? Le CSS décide que c'est la déclaration la plus précise qui l'emporte : comme `<strong>` correspond à un élément plus précis que les paragraphes, le texte sera écrit en 1.4 em.

Images de fond

Dans les exemples qui suivent, je vais modifier l'image de fond de la page. Cependant, tout comme pour la couleur de fond, n'oubliez pas que l'image de fond ne s'applique pas forcément à la page entière. On peut aussi mettre une image de fond derrière les titres, paragraphes, etc.

Appliquer une image de fond

La propriété permettant d'indiquer une image de fond est `background-image` . Comme valeur, on doit renseigner `url("nom_de_l_image.png")` . Par exemple :

```css
body
{
    background-image: url("neige.png");
}
```

Ce qui nous donne la figure suivante.

Bien entendu, votre fond n'est pas forcément en PNG, il peut aussi être en JPEG ou en GIF.
L'adresse indiquant où se trouve l'image de fond peut être écrite en absolu (`http://…`) ou en relatif (`fond.png`).

> Attention lorsque vous écrivez une adresse en relatif dans le fichier CSS ! L'adresse de l'image doit être indiquée *par rapport au fichier* `.css` et non pas par rapport au fichier `.html` . Pour simplifier les choses, je vous conseille de placer l'image de fond dans le même dossier que le fichier `.css` (ou dans un sous-dossier).

Options disponibles pour l'image de fond

On peut compléter la propriété `background-image` que nous venons de voir par plusieurs autres propriétés qui permettent de changer le comportement de l'image de fond.

`background-attachment` : fixer le fond

La propriété CSS `background-attachment` permet de « fixer » le fond. L'effet obtenu est intéressant, car on voit alors le texte « glisser » par-dessus le fond. Deux valeurs sont disponibles :

- `fixed` : l'image de fond reste fixe ;
- `scroll` : l'image de fond défile avec le texte (par défaut).

```css
body
{
    background-image: url("neige.png");
    background-attachment: fixed; /* Le fond restera fixe */
}
```

`background-repeat` : répétition du fond

Par défaut, l'image de fond est répétée en mosaïque. Vous pouvez changer cela avec la propriété `background-repeat` :

- `no-repeat` : le fond ne sera pas répété. L'image sera donc unique sur la page ;
- `repeat-x` : le fond sera répété uniquement sur la première ligne, horizontalement ;
- `repeat-y` : le fond sera répété uniquement sur la première colonne, verticalement ;
- `repeat` : le fond sera répété en mosaïque (par défaut).

Exemple d'utilisation :

```css
body
{
    background-image: url("soleil.png");
    background-repeat: no-repeat;
}
```

`background-position` : position du fond

On peut indiquer où doit se trouver l'image de fond avec `background-position` . Cette propriété n'est intéressante que si elle est combinée avec `background-repeat: no-repeat;` (un fond qui ne se répète pas).

Vous devez donner à `background-position` deux valeurs en pixels pour indiquer la position du fond par rapport au coin supérieur gauche de la page (ou du paragraphe, si vous appliquez le fond à un paragraphe). Ainsi, si vous tapez :

```css
background-position: 30px 50px;
```

... votre fond sera placé à 30 pixels de la gauche et à 50 pixels du haut. Il est aussi possible d'utiliser ces valeurs en anglais :

- `top` : en haut ;
- `bottom` : en bas ;
- `left` : à gauche ;
- `center` : centré ;
- `right` : à droite.

Il est possible de combiner ces mots. Par exemple, pour aligner une image en haut à droite, vous taperez :

```css
background-position: top right;
```

Ainsi, si je veux afficher un soleil en image de fond (figure suivante), en un unique exemplaire (`no-repeat`), toujours visible (`fixed`) et positionné en haut à droite (`top right`), je vais écrire ceci :

```css
body
{
    background-image: url("soleil.png");
    background-attachment: fixed; /* Le fond restera fixe */
    background-repeat: no-repeat; /* Le fond ne sera pas répété */
    background-position: top right; /* Le fond sera placé en haut à droite */
}
```

Un soleil placé en image de fond en haut à droite

Combiner les propriétés

Si vous utilisez beaucoup de propriétés en rapport avec le fond (comme c'est le cas sur ce dernier exemple), vous pouvez utiliser une sorte de « super-propriété » appelée `background` , dont la valeur peut combiner plusieurs des propriétés vues précédemment : `background-image` , `background-repeat` , `background-attachment` et `background-position` .

On peut donc tout simplement écrire :

```css
body
{
    background: url("soleil.png") fixed no-repeat top right;
}
```

C'est la première « super-propriété » que je vous montre, il y en aura d'autres. Il faut savoir que :

- l'ordre des valeurs n'a pas d'importance. Vous pouvez combiner les valeurs dans n'importe quel ordre ;
- vous n'êtes pas obligé de mettre toutes les valeurs. Ainsi, si vous ne voulez pas écrire `fixed` , vous pouvez l'enlever sans problème.

Plusieurs images de fond

Depuis CSS3, il est possible de donner plusieurs images de fond à un élément. Pour cela, il suffit de séparer les déclarations par une virgule, comme ceci :

```css
body
{
    background: url("soleil.png") fixed no-repeat top right, url("neige.png") fixed;
}
```

La première image de cette liste sera placée par-dessus les autres (figure suivante). Attention donc, l'ordre de déclaration des images a son importance : si vous inversez le soleil et la neige dans le code CSS précédent, vous ne verrez plus le soleil !

À noter que les images de fond multiples fonctionnent sur tous les navigateurs sauf sur les anciennes versions d'Internet Explorer, qui ne reconnaît cette fonctionnalité qu'à partir de la version 9 (IE9).

Une dernière chose avant d'en terminer avec les images de fond : dans tous ces exemples, j'ai appliqué un fond à la page entière (`body`). Mais cela ne doit pas vous faire oublier qu'on peut appliquer un fond à n'importe quel élément (un titre, un paragraphe, certains mots d'un paragraphe, etc.).
Je vous conseille donc, pour vous entraîner, d'essayer d'appliquer un fond à vos titres ou paragraphes. Si vous avez un peu de goût (contrairement à moi !) vous arriverez certainement à donner une très belle allure à votre page web.

La transparence

Le CSS nous permet de jouer très facilement avec les niveaux de transparence des éléments ! Pour cela, nous allons utiliser des fonctionnalités de CSS3 : la propriété `opacity` et la notation RGBa.

La propriété `opacity`

La propriété `opacity` , très simple, permet d'indiquer le niveau d'opacité (c'est l'inverse de la transparence).

- Avec une valeur de 1, l'élément sera totalement opaque : c'est le comportement par défaut.
- Avec une valeur de 0, l'élément sera totalement transparent.

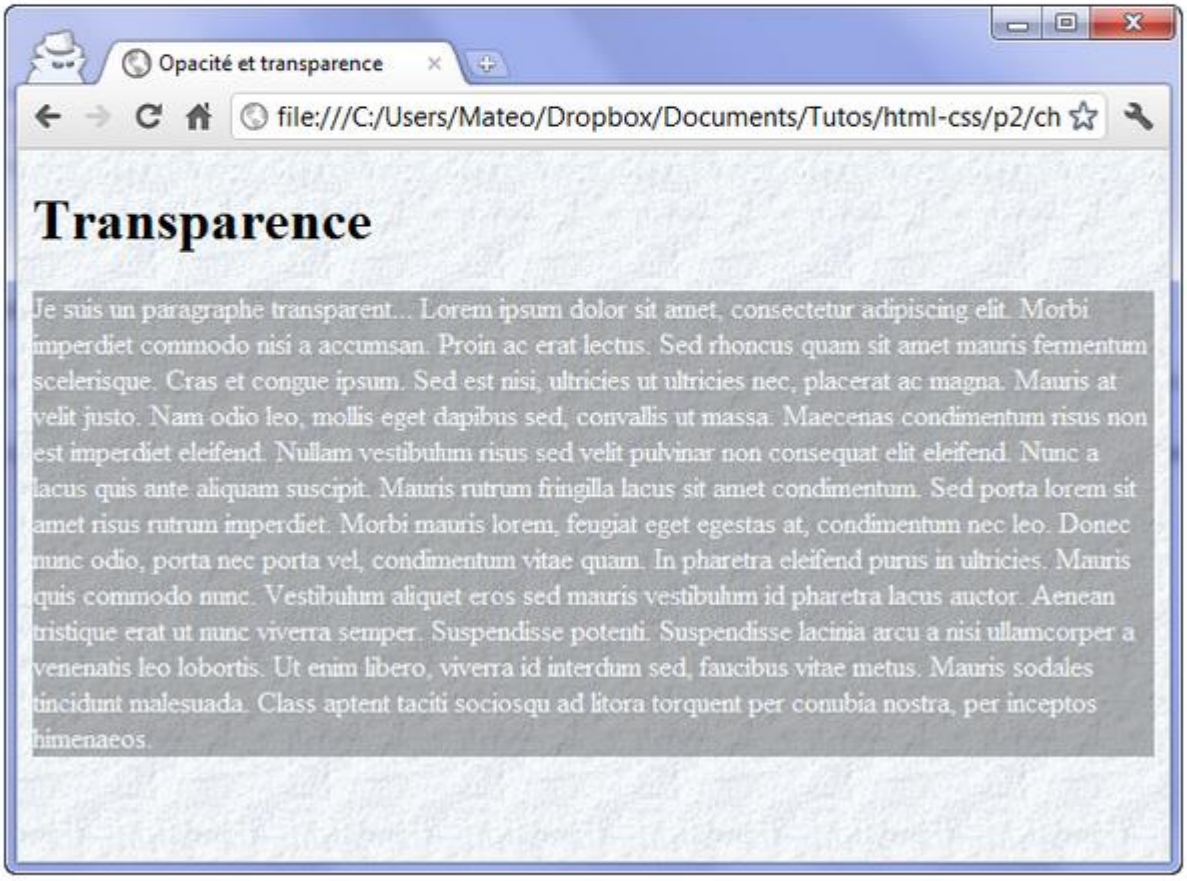
Il faut donc choisir une valeur comprise entre 0 et 1. Ainsi, avec une valeur de 0.6, votre élément sera opaque à 60 %... et on verra donc à travers !

Voici comment on peut l'utiliser :

```css
p
{
    opacity: 0.6;
}
```

Voici un exemple qui va nous permettre d'apprécier la transparence. Vous en trouverez le rendu à la figure suivante.

```css
body
{
    background: url('neige.png');
}

p
{
    background-color: black;
    color: white;
    opacity: 0.3;
}
```



Un paragraphe transparent

Si vous appliquez la propriété `opacity` à un élément de la page, *tout* le contenu de cet élément sera rendu transparent (même les images, les autres blocs à l'intérieur, etc.). Si vous voulez juste rendre la couleur de fond transparente, utilisez plutôt la notation RGBa que nous allons découvrir.

La notation RGBa

CSS3 nous propose une autre façon de jouer avec la transparence : la notation RGBa. Il s'agit en fait de la notation RGB que nous avons vue précédemment, mais avec un quatrième paramètre : le niveau de transparence (appelé « canal alpha »). De la même façon que précédemment, avec une valeur de 1, le fond est complètement opaque. Avec une valeur inférieure à 1, il est transparent.

```css
p
{
    background-color: rgba(255, 0, 0, 0.5); /* Fond rouge à moitié transparent */
}
```

C'est aussi simple que cela. Vous pouvez obtenir exactement le même effet qu'avec `opacity` juste en jouant avec la notation RGBa ; essayez !

Cette notation est connue de tous les navigateurs récents, y compris Internet Explorer (à partir de IE9). Pour les navigateurs plus anciens, il est recommandé d'indiquer la notation RGB classique en plus de RGBa.
Pour ces navigateurs, le fond ne sera alors pas transparent mais, au moins, il y aura bien une couleur d'arrière-plan.

```css
p
{
    background-color: rgb(255,0,0); /* Pour les navigateurs anciens */
    background-color: rgba(255,0,0,0.5); /* Pour les navigateurs plus récents */
}
```

En résumé

- On change la couleur du texte avec la propriété `color` , la couleur de fond avec `background-color` .
- On peut indiquer une couleur en écrivant son nom en anglais (`black` , par exemple), sous forme hexadécimale (`#FFC8D3`) ou en notation RGB (`rgb(250,25,118)`).
- On peut ajouter une image de fond avec `background-image` . On peut choisir de fixer l'image de fond, de l'afficher en mosaïque ou non, et même de la positionner où on veut sur la page.
- On peut rendre une portion de la page transparente avec la propriété `opacity` ou avec la notation RGBa (identique à la notation RGB, avec une quatrième valeur indiquant le niveau de transparence).

❖ Les balises structurantes de HTML5

Je vais vous présenter ici les nouvelles balises introduites par HTML5 pour structurer nos pages. Vous allez voir, cela ne va pas beaucoup changer l'apparence de notre site pour le moment, mais il sera bien construit et prêt à être mis en forme ensuite !

`<header>` : l'en-tête

La plupart des sites web possèdent en général un en-tête, appelé *header* en anglais. On y trouve le plus souvent un logo, une bannière, le slogan de votre site...

Vous devrez placer ces informations à l'intérieur de la balise `<header>` :

```html
<header>
    <!-- Placez ici le contenu de l'en-tête de votre page -->
</header>
```

La figure suivante, par exemple, représente le site du W3C (qui se charge des nouvelles versions de HTML et CSS, notamment). La partie encadrée en rouge correspondrait à l'en-tête :

L'en-tête du site du W3C

L'en-tête peut contenir tout ce que vous voulez : images, liens, textes...

Il peut y avoir plusieurs en-têtes dans votre page. Si celle-ci est découpée en plusieurs sections, chaque section peut en effet avoir son propre `<header>`.

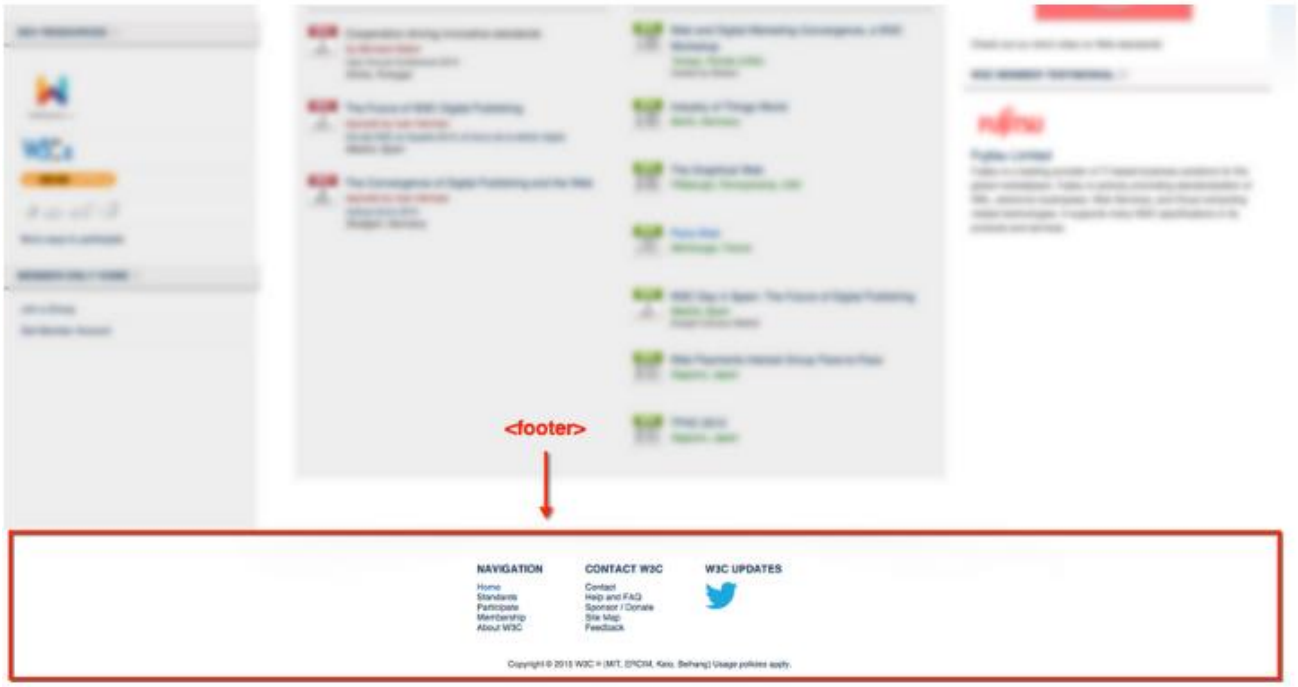
`<footer>` : le pied de page

À l'inverse de l'en-tête, le pied de page se trouve en général tout en bas du document. On y trouve des informations comme des liens de contact, le nom de l'auteur, les mentions légales, etc.

html

```html
<footer>
    <!-- Placez ici le contenu du pied de page -->
</footer>
```

La figure suivante vous montre à quoi ressemble le pied de page du W3C.



Pied de page du W3C

`<nav>` : principaux liens de navigation

La balise `<nav>` doit regrouper tous les principaux liens de navigation du site. Vous y placerez par exemple le menu principal de votre site.

Généralement, le menu est réalisé sous forme de liste à puces à l'intérieur de la balise `<nav>` :

html

```html
<nav>
    <ul>
        <li><a href="index.html">Accueil</a></li>
        <li><a href="forum.html">Forum</a></li>
        <li><a href="contact.html">Contact</a></li>
    </ul>
</nav>
```

Voici le menu sur le site du W3C : `<nav>` .

Le menu de navigation du W3C

`<section>` : une section de page

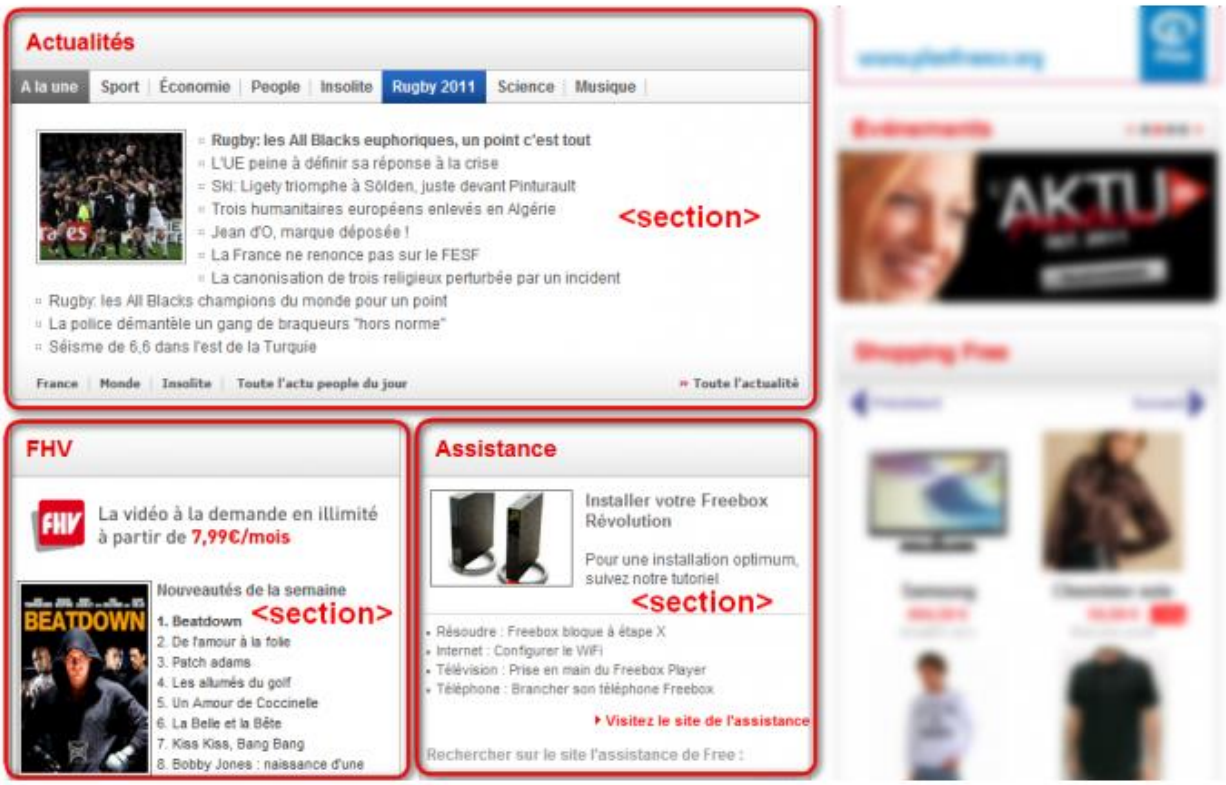
La balise `<section>` sert à regrouper des contenus en fonction de leur thématique. Elle englobe généralement une portion du contenu au centre de la page.

html

```html
<section>
    <h1>Ma section de page</h1>
    <p>Bla bla bla bla</p>
</section>
```

Chaque section peut avoir son titre de niveau 1 (`<h1>`), de même que l'en-tête peut contenir un titre `<h1>` lui aussi. Chacun de ces blocs étant indépendant des autres, il n'est pas illogique de retrouver plusieurs titres `<h1>` dans le code de la page web. On a ainsi « le titre `<h1>` du `<header>` », « le titre `<h1>` de cette `<section>` », etc.

Sur la page d'accueil du portail Free.fr, on trouve plusieurs blocs qui pourraient être considérés comme des sections de page (figure suivante).



Des sections de page sur le portail de Free

`<aside>` : informations complémentaires

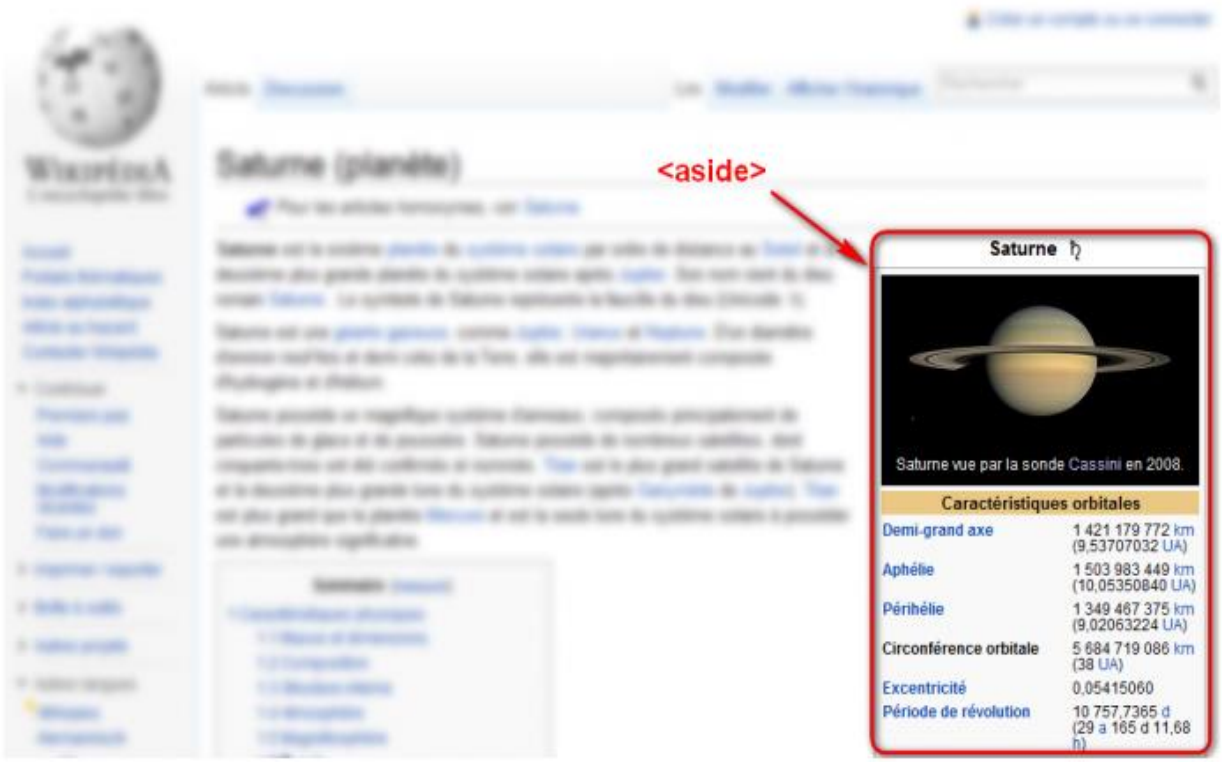
La balise `<aside>` est conçue pour contenir des informations complémentaires au document que l'on visualise. Ces informations sont généralement placées sur le côté (bien que ce ne soit pas une obligation).

```html
<aside>
    <!-- Placez ici des informations complémentaires -->
</aside>
```

Il peut y avoir plusieurs blocs `<aside>` dans la page.

Sur Wikipédia, par exemple, il est courant de voir à droite un bloc d'informations complémentaires à l'article que l'on visualise. Ainsi, sur la page présentant la planète Saturne (figure suivante), on trouve dans ce bloc les caractéristiques de la planète (dimensions, masse, etc.).

Sur Wikipédia, par exemple, il est courant de voir à droite un bloc d'informations complémentaires à l'article que l'on visualise. Ainsi, sur la page présentant la planète Saturne (figure suivante), on trouve dans ce bloc les caractéristiques de la planète (dimensions, masse, etc.).



Bloc d'informations complémentaires sur Wikipédia

`<article>` : un article indépendant

La balise `<article>` sert à englober une portion généralement autonome de la page. C'est une partie de la page qui pourrait ainsi être reprise sur un autre site. C'est le cas par exemple des actualités (articles de journaux ou de blogs).

```html
<article>
    <h1>Mon article</h1>
    <p>Bla bla bla bla</p>
</article>
```

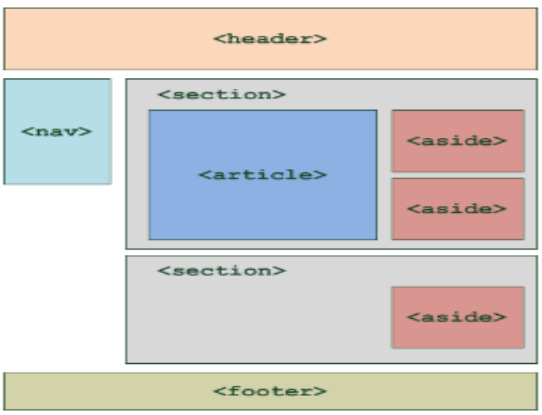
Par exemple, voici un article sur *Le Monde* :

Un article publié sur Le Monde

Résumé

Ouf, cela fait beaucoup de nouvelles balises à retenir.

Heureusement, je vous ai fait un petit schéma (figure suivante) pour vous aider à retenir leur rôle !



Sections de la page identifiées par les balises

Ne vous y trompez pas : ce schéma propose un *exemple* d'organisation de la page. Rien ne vous empêche de décider que votre menu de navigation soit à droite, ou tout en haut, que vos balises `<aside>` soient au-dessus, etc.

On peut même imaginer une seconde balise `<header>`, placée cette fois à l'intérieur d'une `<section>`. Dans ce cas-là, elle sera considérée comme étant l'en-tête de la section.

Enfin, une section ne doit pas forcément contenir un `<article>` et des `<aside>`. Utilisez ces balises uniquement si vous en avez besoin. Rien ne vous interdit de créer des sections contenant seulement des paragraphes, par exemple.

Exemple concret d'utilisation des balises

Essayons d'utiliser les balises que nous venons de découvrir pour structurer notre page web. Le code ci-dessous reprend toutes les balises que nous venons de voir au sein d'une page web complète :

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Zozor - Le Site Web</title>
    </head>

    <body>
        <header>
            <h1>Zozor</h1>
            <h2>Carnets de voyage</h2>
        </header>

        <nav>
            <ul>
                <li><a href="#">Accueil</a></li>
                <li><a href="#">Blog</a></li>
                <li><a href="#">CV</a></li>
            </ul>
        </nav>

        <section>
            <aside>
                <h1>À propos de l'auteur</h1>
                <p>C'est moi, Zozor ! Je suis né un 23 novembre 2005.</p>
            </aside>
            <article>
                <h1>Je suis un grand voyageur</h1>
                <p>Bla bla bla bla (texte de l'article)</p>
            </article>
        </section>

        <footer>
            <p>Copyright Zozor - Tous droits réservés<br />
            <a href="#">Me contacter !</a></p>
        </footer>

    </body>
</html>
```

Ce code peut vous aider à comprendre comment les balises doivent être agencées. Vous y reconnaissez un en-tête, un menu de navigation, un pied de page... et, au centre, une section avec un article et un bloc `<aside>` donnant des informations sur l'auteur de l'article.

À quoi ressemble la page que nous venons de créer ?

À rien !
Si vous testez le résultat, vous verrez juste du texte noir sur fond blanc (figure suivante). C'est normal, il n'y a pas de CSS ! Par contre, la page est bien structurée, ce qui va nous être utile pour la suite.

Une page bien structurée mais sans CSS

Les liens sont volontairement factices (d'où la présence d'un simple `#`), ils n'amènent donc nulle part (eh, c'est juste une page de démo) !

Je ne comprends pas l'intérêt de ces balises. On peut très bien obtenir le même résultat sans les utiliser !

C'est vrai. En fait, ces balises sont seulement là pour expliquer à l'ordinateur « Ceci est l'en-tête », « Ceci est mon pied de page », etc. Elles n'indiquent pas, contrairement à ce qu'on pourrait penser, où doit être placé le contenu. C'est le rôle du CSS, comme nous le verrons dans peu de temps maintenant.

À l'heure actuelle, pour tout vous dire, ces balises ont encore assez peu d'utilité. On pourrait très bien utiliser des balises génériques `<div>` à la place pour englober les différentes portions de notre contenu. D'ailleurs, c'est comme cela qu'on faisait avant l'arrivée de ces nouvelles balises HTML5.

Néanmoins, il est assez probable que, dans un futur proche, les ordinateurs commenceront à tirer parti intelligemment de ces nouvelles balises. On peut imaginer par exemple un navigateur qui choisisse d'afficher les liens de navigation `<nav>` de manière toujours visible ! Quand l'ordinateur « comprend » la structure de la page, tout devient possible.

En résumé

- Plusieurs balises ont été introduites avec HTML5 pour délimiter les différentes zones qui constituent la page web :
  - `<header>` : en-tête ;
  - `<footer>` : pied de page ;
  - `<nav>` : liens principaux de navigation ;
  - `<section>` : section de page ;
  - `<aside>` : informations complémentaires ;
  - `<article>` : article indépendant.
- Ces balises peuvent être imbriquées les unes dans les autres. Ainsi, une section peut avoir son propre en-tête.
- Ces balises ne s'occupent pas de la mise en page. Elles servent seulement à indiquer à l'ordinateur le sens du texte qu'elles contiennent. On pourrait très bien placer l'en-tête en bas de la page, si on le souhaitait.

❖ Découvrez le modèle des boîtes

Une page web peut être vue comme une succession et un empilement de boîtes, qu'on appelle « blocs ». La plupart des éléments vus au chapitre précédent sont des blocs : `<header>` , `<article>` , `<nav>` ... Mais nous connaissions déjà d'autres blocs : les paragraphes `<p>` , les titres `<h1>` ...

Dans ce chapitre, nous allons apprendre à manipuler ces blocs comme de véritables boîtes. Nous allons leur donner des dimensions, les agencer en jouant sur leurs marges, mais aussi apprendre à gérer leur contenu... pour éviter que le texte ne dépasse de ces blocs !

❖ Les balises de type block et inline

En HTML, la plupart des balises peuvent se ranger dans l'une ou l'autre de deux catégories :

- les balises **inline** : c'est le cas par exemple des liens `<a></a>` .
- les balises **block** : c'est le cas par exemple des paragraphes `<p></p>` .

Il existe en fait plusieurs autres catégories très spécifiques, par exemple pour les cellules de tableau (type `table-cell`) ou les puces (type `list-item`). Nous n'allons pas nous y intéresser pour le moment, car ces balises sont minoritaires.

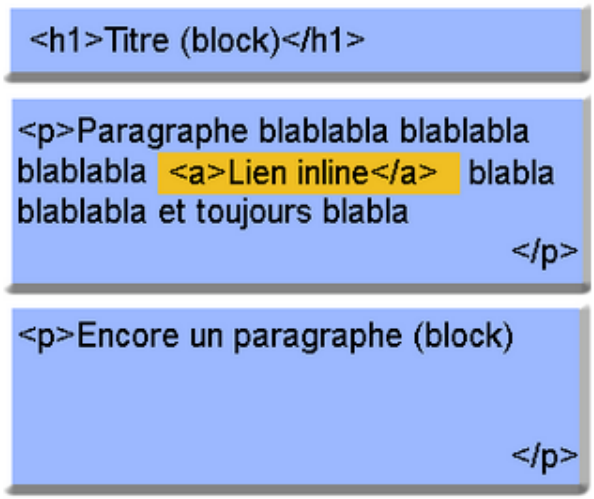
Mais comment je reconnais une balise inline d'une balise block ?

C'est en fait assez facile :

- **block** : une balise de type block sur votre page web crée automatiquement un retour à la ligne avant et après. Il suffit d'imaginer tout simplement un bloc. Votre page web sera en fait constituée d'une série de blocs les uns à la suite des autres. Mais vous verrez qu'en plus, il est possible de mettre un bloc à l'intérieur d'un autre, ce qui va augmenter considérablement nos possibilités pour créer le design de notre site !
- **inline** : une balise de type inline se trouve obligatoirement à l'intérieur d'une balise block. Une balise inline ne crée pas de retour à la ligne, le texte qui se trouve à l'intérieur s'écrit donc à la suite du texte précédent, sur la même ligne (c'est pour cela que l'on parle de balise « en ligne »).

Depuis HTML5, la catégorisation des différents éléments est un peu plus complexe que cela. Cependant, cette petite simplification va vous permettre de bien comprendre la différence entre le concept de "bloc" et le concept de "en ligne".

Pour bien visualiser le concept, voici en figure suivante un petit schéma que je vous ai concocté.



Différence entre une balise inline et une balise block

- Sur fond bleu, vous avez tout ce qui est de type block.
- Sur fond jaune, vous avez tout ce qui est de type inline.

Comme vous pouvez le voir, les blocs sont les uns en dessous des autres. On peut aussi les imbriquer les uns à l'intérieur des autres (souvenez-vous, nos blocs `<section>` contiennent par exemple des blocs `<aside>` !).
La balise inline `<a></a>` , elle, se trouve à l'intérieur d'une balise block et le texte vient s'insérer sur la même ligne.

Les balises universelles

Vous les connaissez déjà car je vous les ai présentées il y a quelques chapitres. Ce sont des balises qui n'ont aucun sens particulier (contrairement à `<p>` qui veut dire « paragraphe » , `<strong>` , « important », etc.).
Le principal intérêt de ces balises est que l'on peut leur appliquer une `class` (ou un `id`) pour le CSS quand aucune autre balise ne convient.

Il existe deux balises génériques et, comme par hasard, la seule différence entre les deux est que l'une d'elle est inline et l'autre est block :

- `<span></span>` (**inline**) ;
- `<div></div>` (**block**).

Respectez la sémantique

Les balises universelles sont « pratiques » dans certains cas, certes, mais attention à ne pas en abuser. Je tiens à vous avertir de suite : beaucoup de webmasters mettent des `<div>` et des `<span>` trop souvent et oublient que d'autres balises plus adaptées existent.

Voici deux exemples :

- **Exemple d'un span inutile** : `<span class="important">` . Je ne devrais jamais voir ceci dans un de vos codes, alors qu'il existe la balise `<strong>` qui sert à indiquer l'importance !
- **Exemple d'un div inutile** : `<div class="titre">` . Ceci est complètement absurde puisqu'il existe des balises faites spécialement pour les titres (`<h1>` , `<h2>` ...).

Oui, vous allez me dire qu'au final le résultat (visuel) est le même. Je suis tout à fait d'accord. Mais les balises génériques n'apportent aucun sens à la page et ne peuvent pas être comprises par l'ordinateur. Utilisez toujours d'autres balises plus adaptées quand c'est possible. Google lui-même le conseille pour vous aider à améliorer la position de vos pages au sein de ses résultats de recherche !

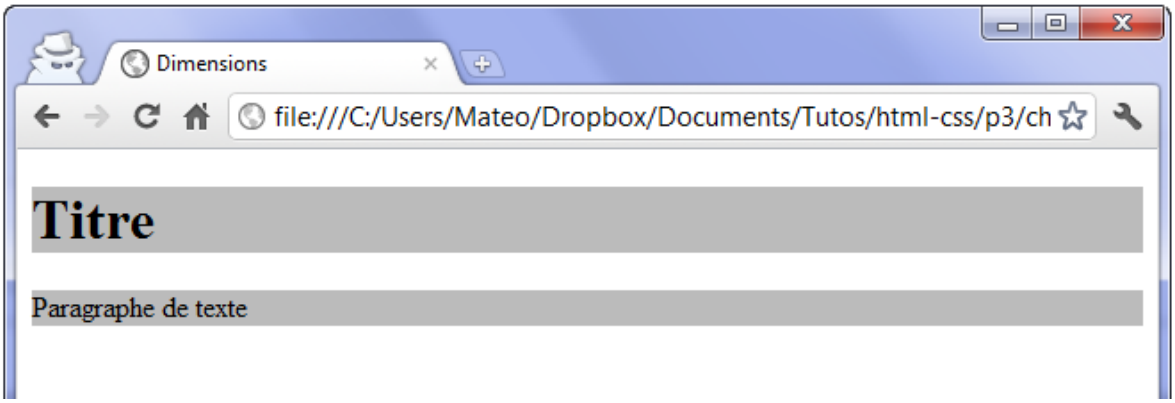
❖ Les dimensions

Pour commencer, intéressons-nous à la taille des blocs. Contrairement à un inline, un bloc a des dimensions précises. Il possède une largeur et une hauteur. Ce qui fait, ô surprise, qu'on dispose de deux propriétés CSS :

- `width` : c'est la largeur du bloc. À exprimer en pixels (px) ou en pourcentage (%) ;
- `height` : c'est la hauteur du bloc. Là encore, on l'exprime soit en pixels (px), soit en pourcentage (%).

Pour être exact, `width` et `height` représentent la largeur et la hauteur du contenu des blocs. Si le bloc a des marges (on va découvrir ce principe un peu plus loin), celles-ci s'ajouteront à la largeur et à la hauteur.

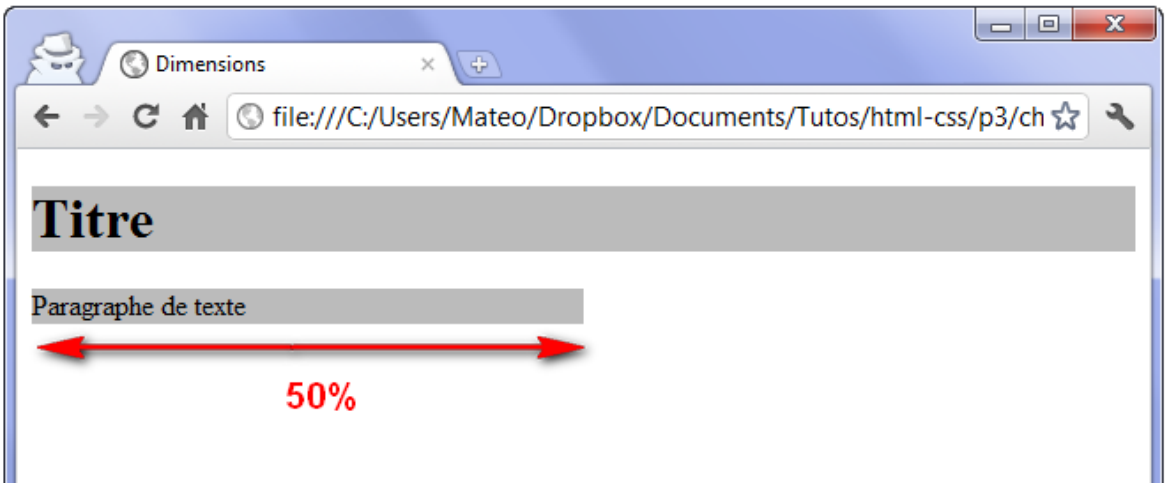
Par défaut, un bloc prend 100 % de la largeur disponible. On peut le vérifier en appliquant à nos blocs des bordures ou une couleur de fond (figure suivante).



Maintenant, rajoutons un peu de CSS afin de modifier la largeur des paragraphes. Le CSS suivant dit : « Je veux que tous mes paragraphes aient une largeur de 50 % ».

```css
p
{
    width: 50%;
}
```

Le résultat est visible à la figure suivante.



Un paragraphe de 50 % de largeur

Les pourcentages seront utiles pour créer un design qui s'adapte automatiquement à la résolution d'écran du visiteur.

Toutefois, il se peut que vous ayez besoin de créer des blocs ayant une dimension précise en pixels :

```css
p
{
   width: 250px;
}
```

Minimum et maximum

On peut demander à ce qu'un bloc ait des dimensions minimales et maximales. C'est très pratique, car cela nous permet de définir des dimensions « limites » pour que notre site s'adapte aux différentes résolutions d'écran de nos visiteurs :

- `min-width` : largeur minimale ;
- `min-height` : hauteur minimale ;
- `max-width` : largeur maximale ;
- `max-height` : hauteur maximale.

Par exemple, on peut demander à ce que les paragraphes occupent 50 % de la largeur *et* exiger qu'il fassent au moins 400 pixels de large dans tous les cas :

```css
p
{
    width: 50%;
    min-width: 400px;
}
```

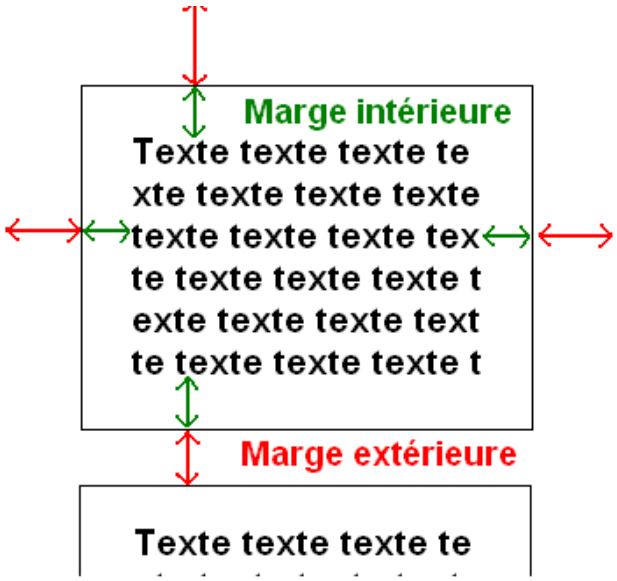
Observez le résultat en modifiant la largeur de la fenêtre de votre navigateur. Vous allez voir que, si celle-ci est trop petite, le paragraphe se force à occuper au moins 400 pixels de largeur.

❖ Les Marges

Il faut savoir que tous les blocs possèdent des marges. Il existe *deux types de marges* :

- les marges intérieures ;
- les marges extérieures.

Regardez bien le schéma qui se trouve à la figure suivante.



Marges extérieure et intérieure

Sur ce bloc, j'ai mis une bordure pour qu'on repère mieux ses frontières.

- L'espace entre le texte et la bordure est la marge intérieure (en vert).
- L'espace entre la bordure et le bloc suivant est la marge extérieure (en rouge).

En CSS, on peut modifier la taille des marges avec les deux propriétés suivantes :

- `padding` : indique la taille de la marge intérieure. À exprimer en général en pixels (px) ;
- `margin` : indique la taille de la marge extérieure. Là encore, on utilise le plus souvent des pixels.

Les balises de type inline possèdent également des marges. Vous pouvez donc aussi essayer ces manipulations sur ce type de balises.

Pour bien voir les marges, prenons deux paragraphes auxquels j'applique simplement une petite bordure (figure suivante) :

CSS

```css
p
{
    width: 350px;
    border: 1px solid black;
    text-align: justify;
}
```

Marges par défaut sur les paragraphes

Comme vous pouvez le constater, il n'y a par défaut pas de marge intérieure (`padding`). En revanche, il y a une marge extérieure (`margin`). C'est cette marge qui fait que deux paragraphes ne sont pas collés et qu'on a l'impression de « sauter une ligne ».

Les marges par défaut ne sont pas les mêmes pour toutes les balises de type block. Essayez d'appliquer ce CSS à des balises `<div>` qui contiennent du texte, par exemple : vous verrez que, dans ce cas, il n'y a par défaut ni marge intérieure, ni marge extérieure !

Supposons que je veuille rajouter une marge intérieure de 12 px aux paragraphes (figure suivante) :

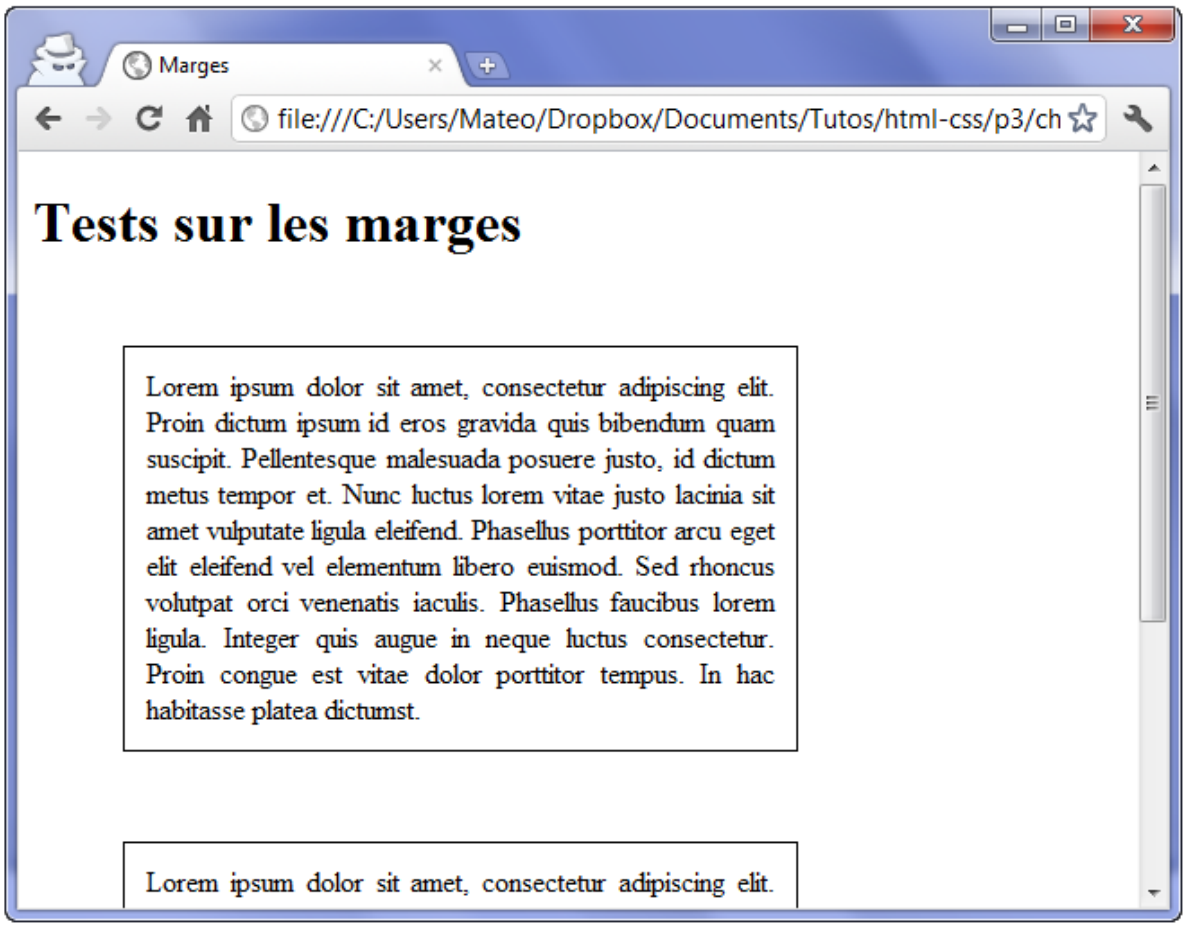
CSS

```css
p
{
    width: 350px;
    border: 1px solid black;
    text-align: justify;
    padding: 12px; /* Marge intérieure de 12px */
}
```

Une marge intérieure ajoutée aux paragraphes

Maintenant, je veux que mes paragraphes soient plus espacés entre eux. Je rajoute la propriété `margin` pour demander à ce qu'il y ait 50 px de marge entre deux paragraphes (figure suivante) :

```css
p
{
   width: 350px;
   border: 1px solid black;
   text-align: justify;
   padding: 12px;
   margin: 50px; /* Marge extérieure de 50px */
}
```



Une marge extérieure ajoutée aux paragraphes

Mais ??? Une marge s'est rajoutée à gauche aussi !

Eh oui, `margin` (comme `padding`, d'ailleurs) s'applique aux quatre côtés du bloc.

Si vous voulez spécifier des marges différentes en haut, en bas, à gauche et à droite, il va falloir utiliser des propriétés plus précises... Le principe est le même que pour la propriété `border`, vous allez voir !

En haut, à droite, en bas, à gauche... et on recommence !

L'idéal serait que vous reteniez les termes suivants en anglais :

- *top* : haut ;
- *bottom* : bas ;
- *left* : gauche ;
- *right* : droite.

Ainsi, vous pouvez retrouver toutes les propriétés de tête.
Je vais quand même vous faire la liste des propriétés pour `margin` et `padding`, histoire que vous soyez sûr d'avoir compris le principe.

Voici la liste pour `margin` :

- `margin-top` : marge extérieure en haut ;
- `margin-bottom` : marge extérieure en bas ;
- `margin-left` : marge extérieure à gauche ;
- `margin-right` : marge extérieure à droite.

Et la liste pour `padding` :

- `padding-top` : marge intérieure en haut ;
- `padding-bottom` : marge intérieure en bas ;
- `padding-left` : marge intérieure à gauche ;
- `padding-right` : marge intérieure à droite.

Il y a d'autres façons de spécifier les marges avec les propriétés `margin` et `padding`. Par exemple :
`margin: 2px 0 3px 1px;` signifie « 2 px de marge en haut, 0 px à droite (le px est facultatif dans ce cas), 3 px en bas, 1 px à gauche ».
Autre notation raccourcie : `margin: 2px 1px;` signifie « 2 px de marge en haut et en bas, 1 px de marge à gauche et à droite ».

❖ Centrer des blocs

Il est tout à fait possible de centrer des blocs. C'est même très pratique pour réaliser un design centré quand on ne connaît pas la résolution du visiteur.

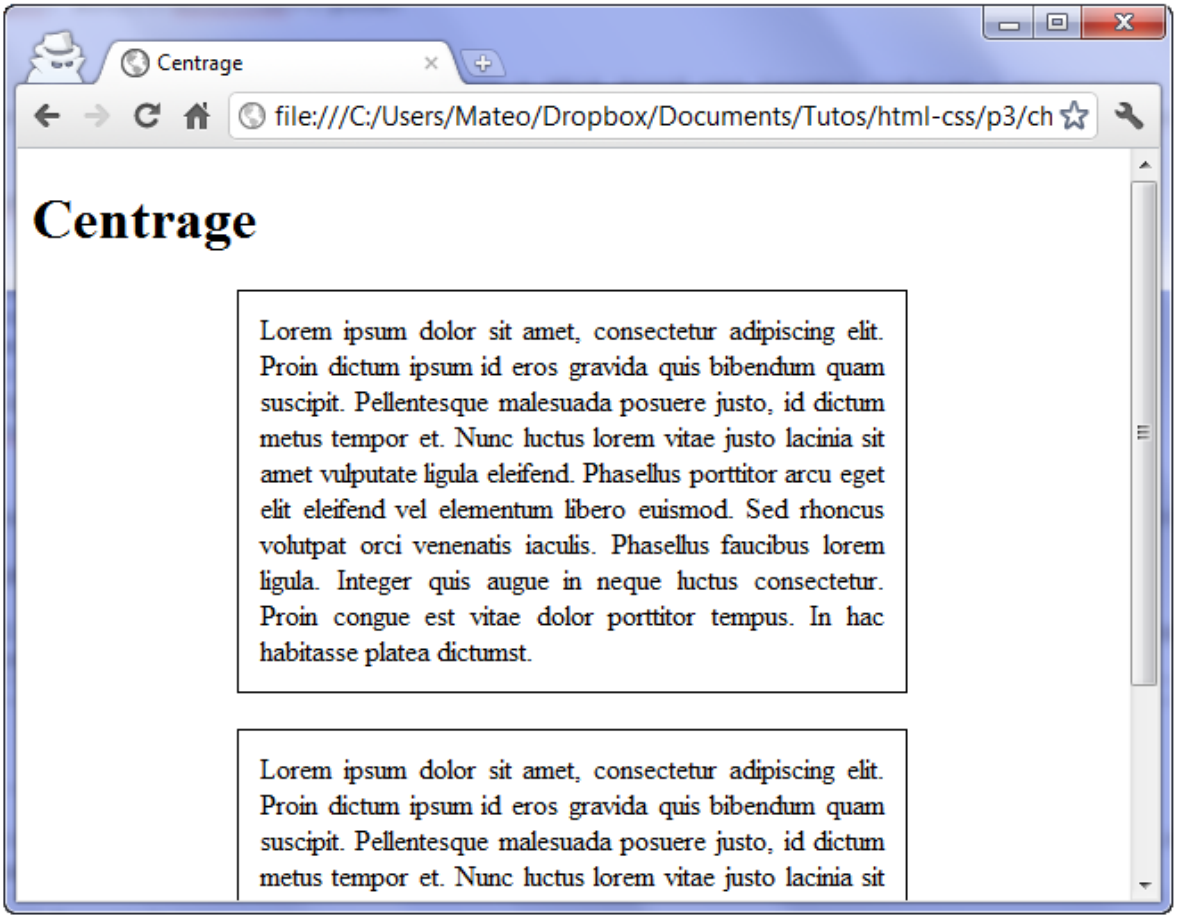
Pour centrer, il faut respecter les règles suivantes :

- donnez une largeur au bloc (avec la propriété `width`) ;
- indiquez que vous voulez des marges extérieures automatiques, comme ceci : `margin: auto;`.

Essayons cette technique sur nos petits paragraphes (lignes 3 et 4) :

```css
p
{
    width: 350px; /* On a indiqué une largeur (obligatoire) */
    margin: auto; /* On peut donc demander à ce que le bloc soit centré avec auto */
    border: 1px solid black;
    text-align: justify;
    padding: 12px;
    margin-bottom: 20px;
}
```

Et voici le résultat à la figure suivante.



Centrage des paragraphes

Ainsi, le navigateur centre automatiquement nos paragraphes !

Il n'est cependant pas possible de centrer verticalement un bloc avec cette technique. Seul le centrage horizontal est permis.

Quand ça dépasse :

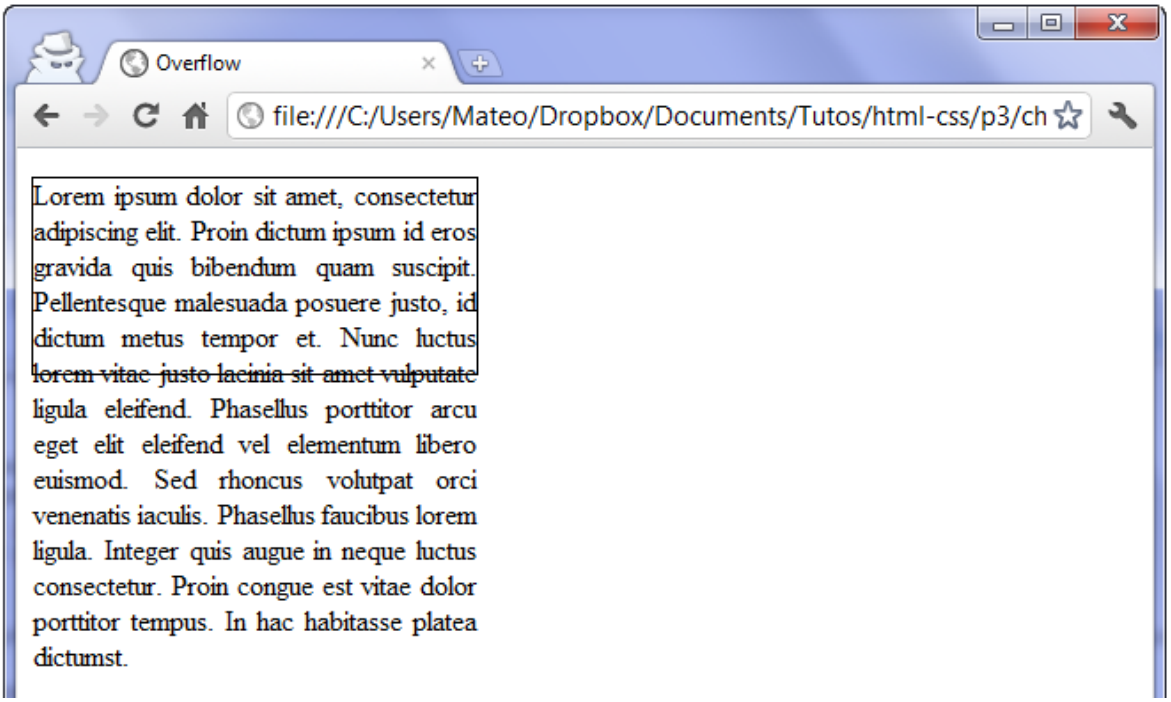
Lorsqu'on commence à définir des dimensions précises pour nos blocs, comme on vient de le faire, il arrive qu'ils deviennent trop petits pour le texte qu'ils contiennent.

Les propriétés CSS que nous allons voir ici ont justement été créées pour contrôler les dépassements... et décider quoi faire si jamais cela devait arriver.

`overflow` : couper un bloc

Supposons que vous ayez un long paragraphe et que vous vouliez (pour une raison qui ne regarde que vous) qu'il fasse 250 px de large et 110 px de haut. Ajoutons-lui une bordure et remplissons-le de texte... à ras-bord (figure suivante) :

```css
p
{
    width: 250px;
    height: 110px;
    text-align: justify;
    border: 1px solid black;
}
```



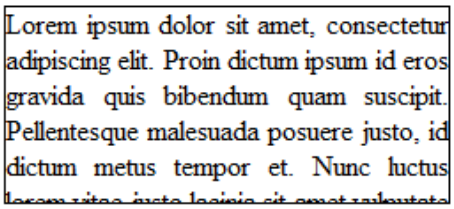
Horreur ! Le texte dépasse des limites du paragraphe !

Eh oui ! Vous avez demandé des dimensions précises, vous les avez eues ! Mais... le texte ne tient pas à l'intérieur d'un si petit bloc.

Si vous voulez que le texte ne dépasse pas des limites du paragraphe, il va falloir utiliser la propriété `overflow` . Voici les valeurs qu'elle peut accepter :

- `visible` (par défaut) : si le texte dépasse les limites de taille, il reste visible et sort volontairement du bloc ;
- `hidden` : si le texte dépasse les limites, il sera tout simplement coupé. On ne pourra pas voir tout le texte ;
- `scroll` : là encore, le texte sera coupé s'il dépasse les limites. Sauf que cette fois, le navigateur mettra en place des barres de défilement pour qu'on puisse lire l'ensemble du texte. C'est un peu comme un cadre à l'intérieur de la page.
- `auto` : c'est le mode « pilote automatique ». En gros, c'est le navigateur qui décide de mettre ou non des barres de défilement (il n'en mettra que si c'est nécessaire). C'est la valeur que je conseille d'utiliser le plus souvent.

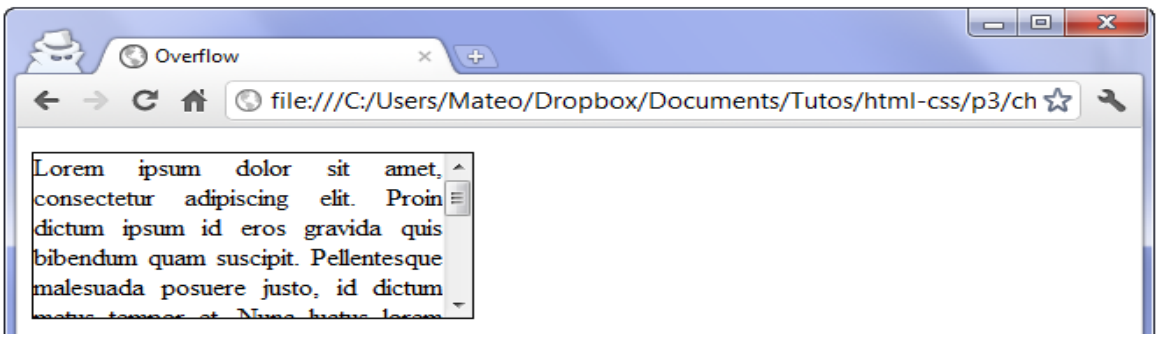
Avec `overflow: hidden;` , le texte est donc coupé (on ne peut pas voir la suite), comme sur la figure suivante.



Le texte est coupé aux limites du paragraphe

Essayons maintenant `overflow: auto;` avec le code CSS suivant (résultat à la figure suivante) :

```css
p
{
    width: 250px;
    height: 110px;
    text-align: justify;
    border: 1px solid black;
    overflow: auto;
}
```



Eurêka ! Des barres de défilement nous permettent maintenant de consulter le contenu qui n'était pas visible.

> Il existe une ancienne balise HTML, `<iframe>` , qui donne à peu près le même résultat. Cependant, l'usage de cette balise est déconseillé aujourd'hui. Elle permet de charger tout le contenu d'une autre page HTML au sein de votre page.

`word-wrap` : couper les textes trop larges

Si vous devez placer un mot très long dans un bloc, qui ne tient pas dans la largeur, vous allez adorer `word-wrap` . Cette propriété permet de forcer la césure des très longs mots (généralement des adresses un peu longues).

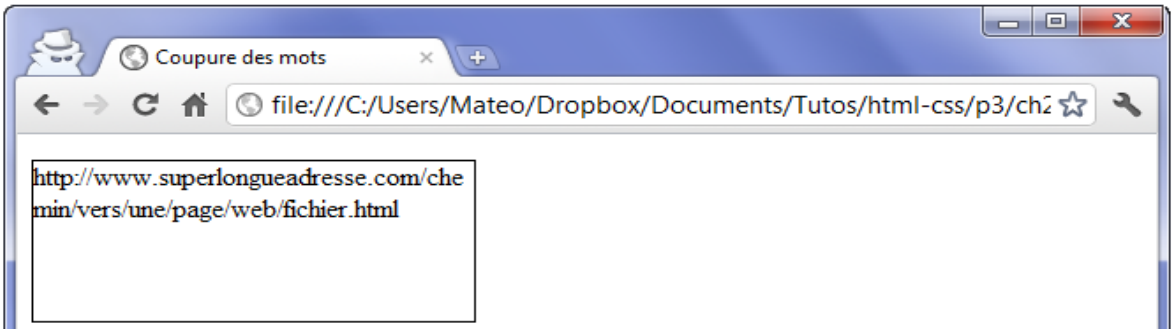
La figure suivante représente ce que l'on peut avoir quand on écrit une URL un peu longue dans un bloc.

Le texte déborde en largeur

L'ordinateur ne sait pas « couper » l'adresse car il n'y a ni espace, ni tiret. Il ne sait pas faire la césure.

Avec le code suivant, la césure sera forcée dès que le texte risque de dépasser (figure suivante).

```
1 p
2 {
3     word-wrap: break-word;
4 }
```



Je conseille d'utiliser cette fonctionnalité dès qu'un bloc est susceptible de contenir du texte saisi par des utilisateurs (par exemple sur les forums de votre futur site). Sans cette astuce, on peut « casser » facilement le design d'un site (en écrivant par exemple une longue suite de « aaaaaaaaaaa »).

En résumé

- On distingue deux principaux types de balises en HTML :
  - le type block (`<p>` , `<h1>` ...) : ces balises créent un retour à la ligne et occupent par défaut toute la largeur disponible. Elles se suivent de haut en bas ;
  - le type inline (`<a>` , `<strong>` ...) : ces balises délimitent du texte au milieu d'une ligne. Elles se suivent de gauche à droite.
- On peut modifier la taille d'une balise de type block avec les propriétés CSS `width` (largeur) et `height` (hauteur).
- On peut définir des minima et maxima autorisés pour la largeur et la hauteur : `min-width` , `max-width` , `min-height` , `max-height` .
- Les éléments de la page disposent chacun de marges intérieures (`padding`) et extérieures (`margin`).
- S'il y a trop de texte à l'intérieur d'un bloc de dimensions fixes, il y a un risque de débordement. Dans ce cas, il peut être judicieux de rajouter des barres de défilement avec la propriété `overflow` , ou de forcer la césure avec `word-wrap` .

❖ Faites votre mise en page avec Flexbox

Allez, il est temps d'apprendre à mettre en page notre site. Vous savez ? Placer un en-tête, des menus sur le côté, choisir où apparaît une information, etc. C'est la pièce manquante du puzzle pour que nous puissions enfin créer notre site !

Il y a plusieurs façons de mettre en page un site. Au fil du temps, plusieurs techniques ont existé :

1. Au début, les webmasters utilisaient des tableaux HTML pour faire la mise en page (berk).
2. Puis, CSS est apparu et on a commencé à faire une mise en page à l'aide de la propriété `float` (bof).
3. Cette technique avait des inconvénients. Une autre, plus pratique, a consisté à créer des éléments de type `inline-block` sur la page (mouais).
4. Aujourd'hui, une bien meilleure technique encore existe : **Flexbox** ! Elle permet toutes les folies (ou presque), et c'est celle que je vous recommande d'utiliser si vous en avez la possibilité, lorsque vous créez un nouveau site. Flexbox est désormais reconnu par tous les navigateurs récents !

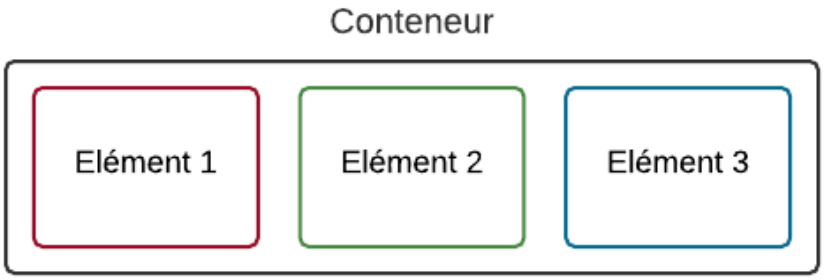
Nous découvrirons donc le fonctionnement de Flexbox dans ce chapitre. Pour ceux qui ont besoin d'informations sur les autres techniques de mise en page plus anciennes, je vous invite à consulter le chapitre "Quelques autres techniques de mise en page". Cela peut toujours vous être utile.

❖ Un conteneur, des éléments

Le principe de la mise en page avec Flexbox est simple : vous définissez un conteneur, et à l'intérieur vous placez plusieurs éléments. Imaginez un carton dans lequel vous rangez plusieurs objets : c'est le principe !

Sur une même page web, vous pouvez sans problème avoir plusieurs conteneurs (plusieurs cartons, si vous préférez). Ce sera à vous d'en créer autant que nécessaire pour obtenir la mise en page que vous voulez.

Commençons par étudier le fonctionnement d'un carton (euh pardon, d'un conteneur).



Un conteneur et ses éléments

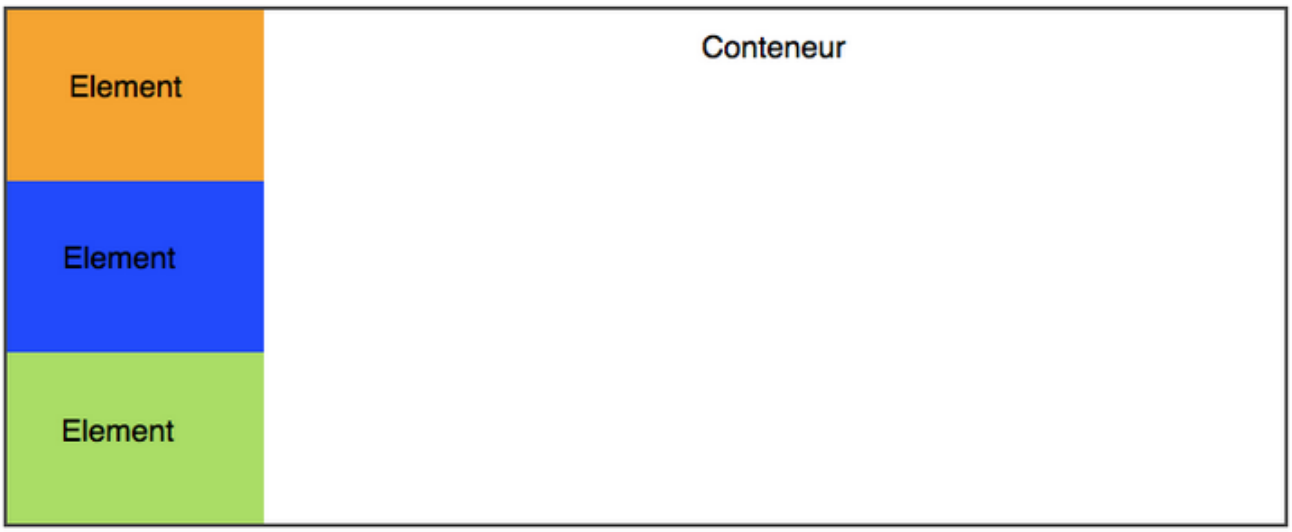
Le conteneur est une balise HTML, et les éléments sont d'autres balises HTML à l'intérieur :

```html
<div id="conteneur">
    <div class="element 1">Element </div>
    <div class="element 2">Element </div>
    <div class="element 3">Element </div>
</div>
```

OK, jusque-là, vous devriez suivre.

> Mais si je fais ça, par défaut, mes éléments vont se mettre les uns en dessous des autres, non ? Ce sont des blocs, après tout !

Oui, tout à fait. Si je mets une bordure au conteneur, une taille et une couleur de fond aux éléments, on va vite voir comment ils s'organisent :



Par défaut, les blocs se placent les uns en dessous des autres

Découvrons maintenant Flexbox. Si je mets une (une seule !) propriété CSS, tout change. Cette propriété, c'est `flex` , et je l'applique au conteneur :

```css
#conteneur
{
    display: flex;
}
```

... alors les blocs se placent par défaut côte à côte. Magique !

Un coup de flex, et les blocs se positionnent côte à côte !

La direction

Flexbox nous permet d'agencer ces éléments dans le sens que l'on veut. Avec `flex-direction` , on peut les positionner verticalement ou encore les inverser. Il peut prendre les valeurs suivantes :

- row : organisés sur une ligne (par défaut) ;
- column : organisés sur une colonne ;
- row-reverse : organisés sur une ligne, mais en ordre inversé ;
- column-reverse : organisés sur une colonne, mais en ordre inversé.

Exemple :

```css
#conteneur
{
    display: flex;
    flex-direction: column;
}
```

Mais mais... c'est pareil qu'au début, non ? On avait ce résultat sans Flexbox, après tout !

C'est vrai. Mais maintenant que nos éléments sont *flex*, ils ont tout un tas d'autres propriétés utiles que nous allons voir juste après, on va y revenir.

Essayez aussi de tester l'ordre inversé, pour voir :

```css
#conteneur
{
    display: flex;
    flex-direction: column-reverse;
}

```

Les éléments sont en colonne... dans l'ordre inverse !

Regardez bien la différence : les blocs sont maintenant dans l'ordre inverse ! Je n'ai pas du tout changé le code HTML, qui reste le même depuis le début.

Le retour à la ligne

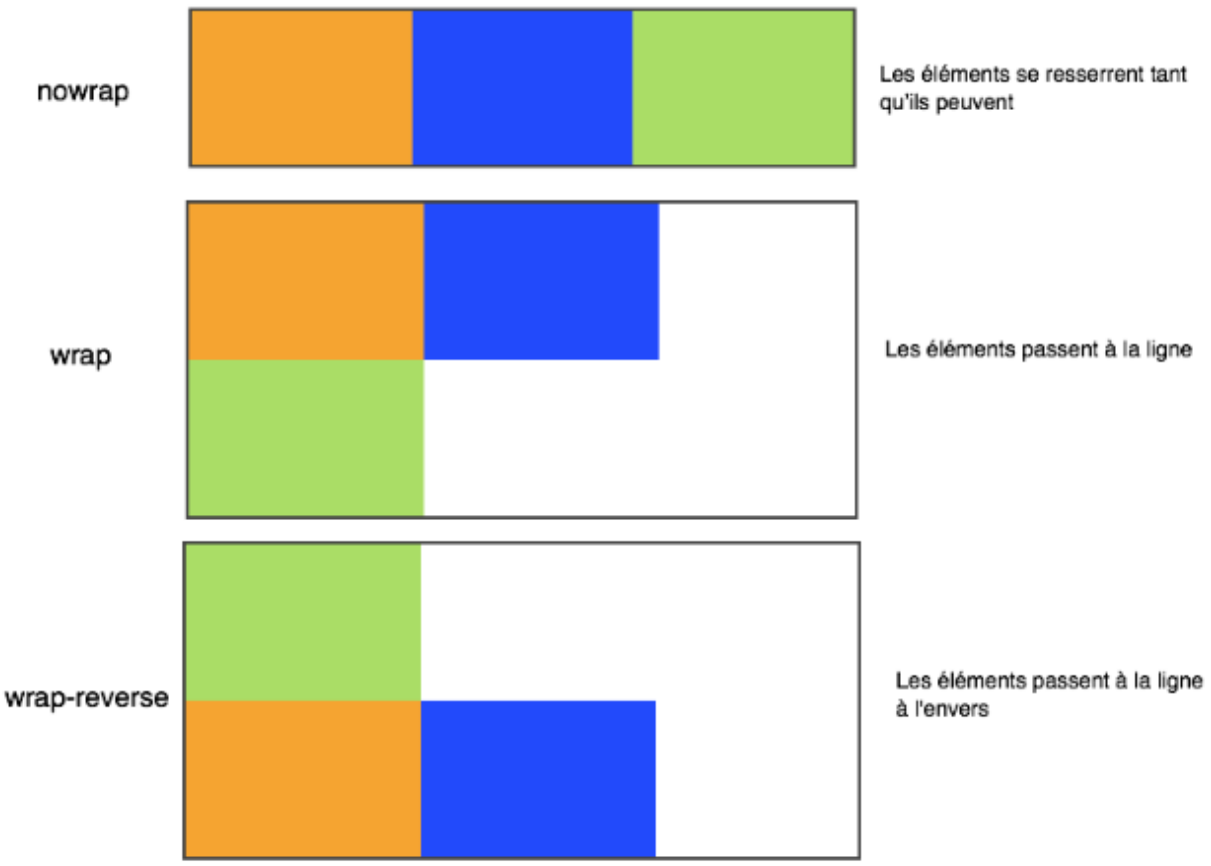
Par défaut, les blocs essaient de rester sur la même ligne s'ils n'ont pas la place (ce qui peut provoquer des bugs de design, parfois). Si vous voulez, vous pouvez demander à ce que les blocs aillent à la ligne lorsqu'ils n'ont plus la place, avec `flex-wrap` qui peut prendre ces valeurs :

- nowrap : pas de retour à la ligne (par défaut) ;
- wrap : les éléments vont à la ligne lorsqu'il n'y a plus la place ;
- wrap-reverse : les éléments vont à la ligne, lorsqu'il n'y a plus la place, en sens inverse.

Exemple :

```css
#conteneur
{
    display: flex;
    flex-wrap: wrap;
}

```

Voici l'effet que prennent les différentes valeurs sur une même illustration :



Gestion du retour à la ligne avec flex-wrap

Alignez-les !

Reprenons. Les éléments sont organisés soit horizontalement (par défaut), soit verticalement. Cela définit ce qu'on appelle **l'*axe principal***. Il y a aussi un axe secondaire (*cross axis*) :

- si vos éléments sont organisés horizontalement, l'axe secondaire est l'axe vertical ;
- si vos éléments sont organisés verticalement, l'axe secondaire est l'axe horizontal.

Pourquoi je vous raconte ça ? Parce que nous allons découvrir comment aligner nos éléments sur l'axe principal *et* sur l'axe secondaire.

Aligner sur l'axe principal

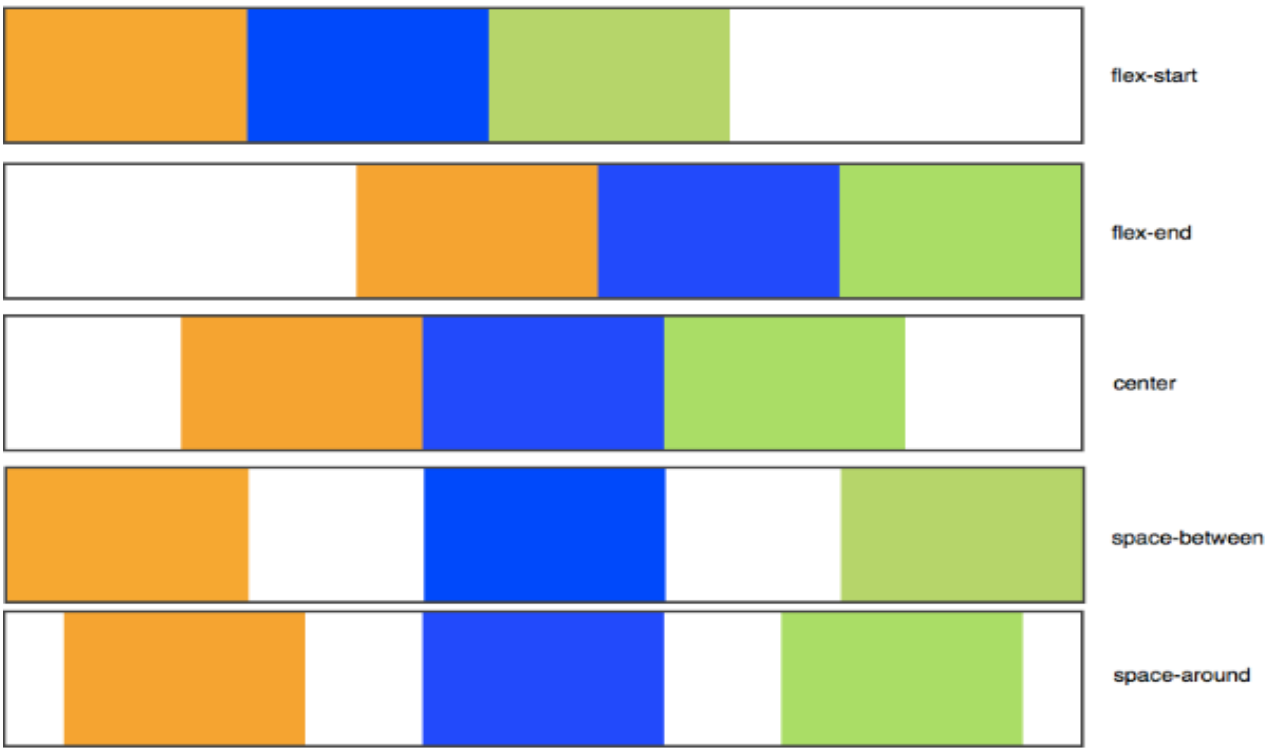
Pour faire simple, partons sur des éléments organisés horizontalement (c'est le cas par défaut).

Pour changer leur alignement, on va utiliser `justify-content` , qui peut prendre ces valeurs :

- flex-start : alignés au début (par défaut) ;
- flex-end : alignés à la fin ;
- center : alignés au centre ;
- space-between : les éléments sont étirés sur tout l'axe (il y a de l'espace entre eux) ;
- space-around : idem, les éléments sont étirés sur tout l'axe, mais ils laissent aussi de l'espace sur les extrémités.

Par exemple :

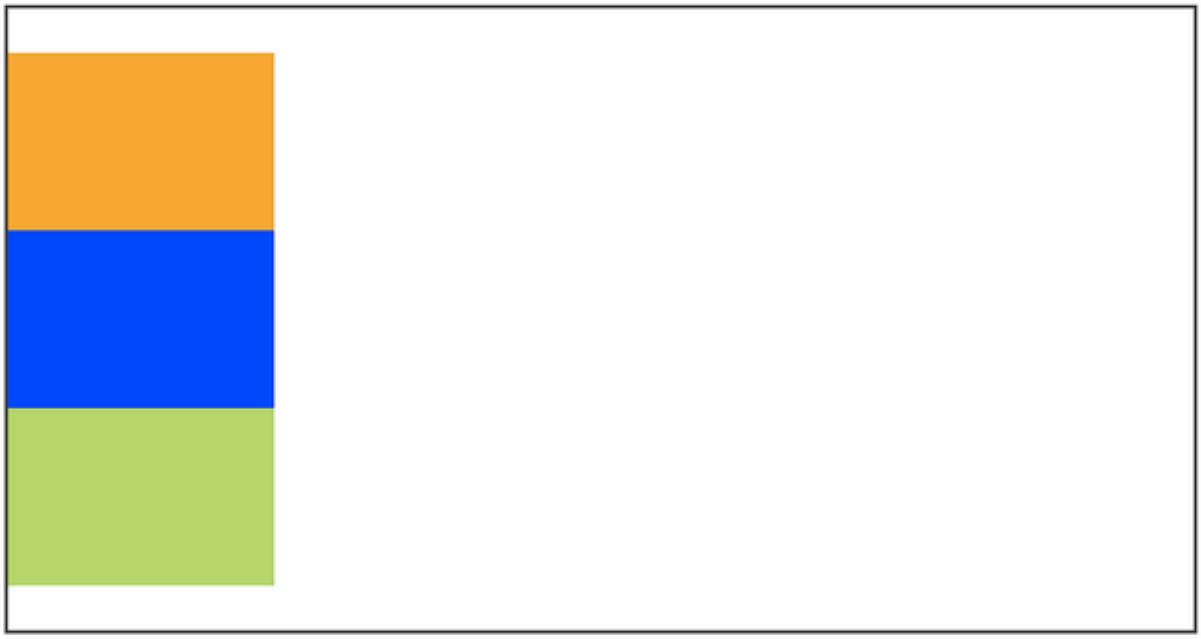
```css
#conteneur
{
    display: flex;
    justify-content: space-around;
}
```



Vous voyez comment les éléments s'alignent différemment selon les cas ? Avec une simple propriété, on peut intelligemment agencer nos éléments comme on veut !

Maintenant, voici ce qu'il faut bien comprendre : **ça marche aussi si vos éléments sont dans une direction verticale**. Dans ce cas, l'axe vertical devient l'axe principal, et `justify-content` s'applique aussi :

```css
#conteneur
{
    display: flex;
    flex-direction: column;
    justify-content: center;
    height: 350px; /* Un peu de hauteur pour que les éléments aient la place de bouger */
}
```



Avec une direction verticale (column), le centrage fonctionne de la même façon, cette fois en hauteur !

Essayez, à vous de jouer !

Aligner sur l'axe secondaire

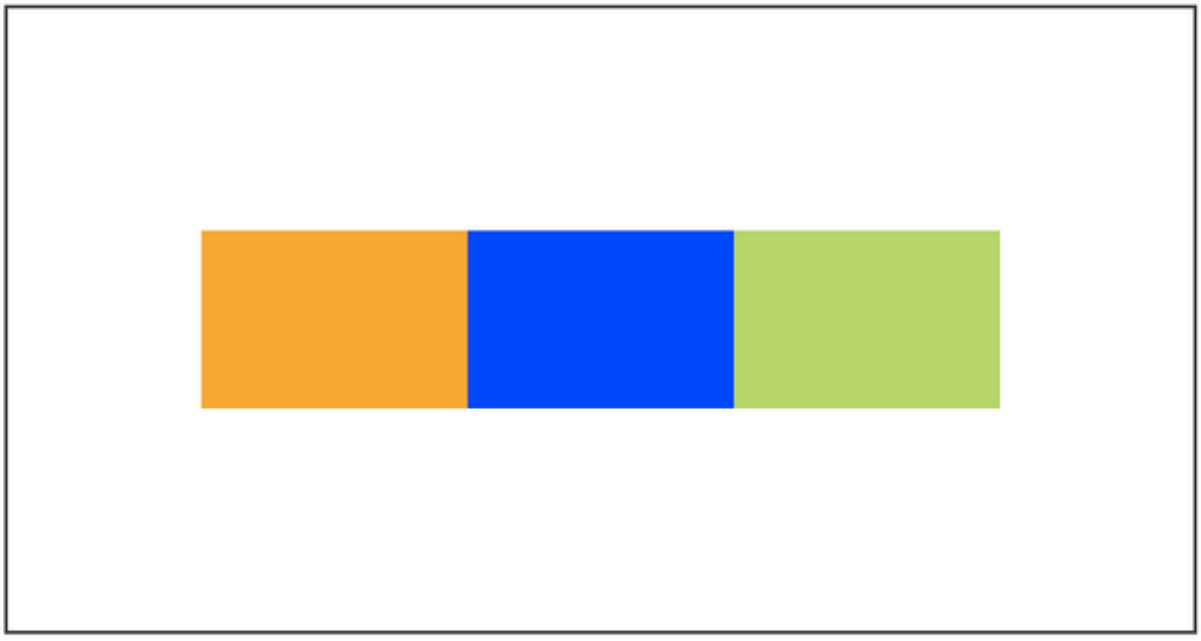
Comme je vous le disais, si nos éléments sont placés dans une direction horizontale (ligne), l'axe secondaire est... vertical. Et inversement : si nos éléments sont dans une direction verticale (colonne), l'axe secondaire est horizontal.

Avec `align-items` , nous pouvons changer leur alignement sur l'axe secondaire. Il peut prendre ces valeurs :

- stretch : les éléments sont étirés sur tout l'axe (valeur par défaut) ;
- flex-start : alignés au début ;
- flex-end : alignés à la fin ;
- center : alignés au centre ;
- baseline : alignés sur la ligne de base (semblable à flex-start).

Pour ces exemples, nous allons partir du principe que nos éléments sont dans une direction horizontale (mais n'hésitez pas à tester aussi dans la direction verticale !).

```css
#conteneur
{
    display: flex;
    justify-content: center;
    align-items: center;
}
```



Un alignement sur l'axe secondaire avec align-items nous permet de centrer

Saint Graal du développeur web, le centrage vertical et horizontal peut d'ailleurs être obtenu encore plus facilement. Dites que votre conteneur est une flexbox et établissez des marges automatiques sur les éléments à l'intérieur. C'est tout ! Essayez !

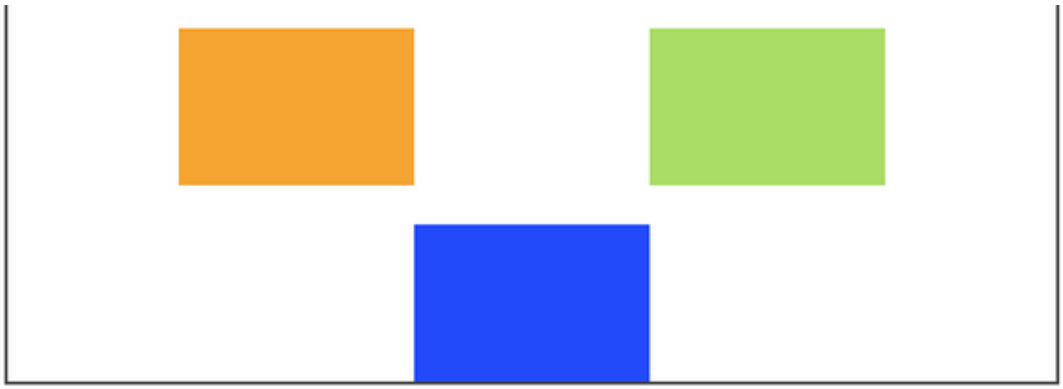
```css
#conteneur
{
    display: flex;
}

.element
{
    margin: auto;
}
```

Aligner un seul élément

Il est possible de faire une exception pour un seul des éléments sur l'axe secondaire avec `align-self` :

```css
#conteneur
{
    display: flex;
    flex-direction: row;
    justify-content: center;
    align-items: center;
}

.element:nth-child(2) /* On prend le deuxième bloc élément */
{
    background-color: blue;
    align-self: flex-end; /* Seul ce bloc sera aligné à la fin */
}

/* ... */
```

Résultats :



Répartir plusieurs lignes

Si vous avez plusieurs lignes dans votre Flexbox, vous pouvez choisir comment celles-ci seront réparties avec `align-content` .

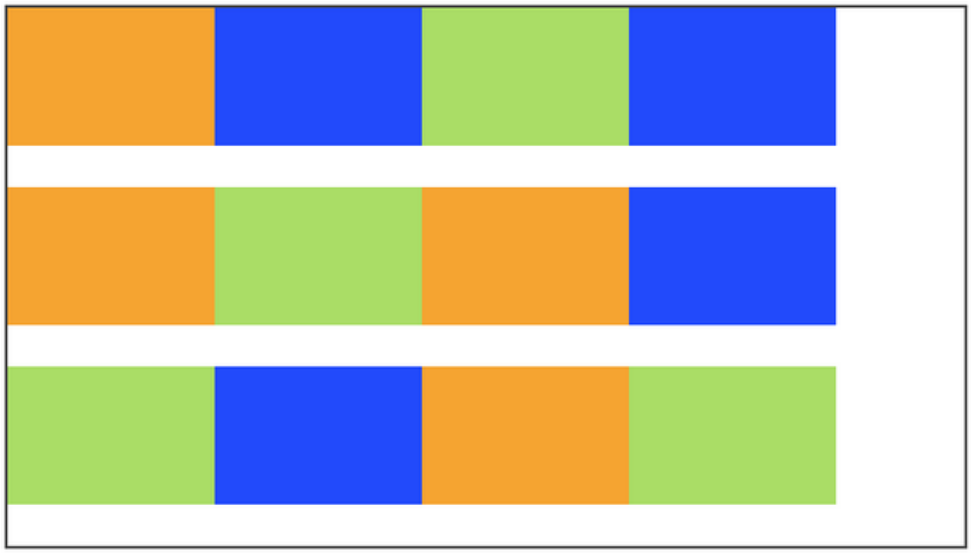
> Cette propriété n'a aucun effet s'il n'y a qu'une seule ligne dans la Flexbox.

Prenons donc un cas de figure où nous avons plusieurs lignes. Je vais rajouter des éléments :

```html
<div id="conteneur">
    <div class="element"></div>
    <div class="element"></div>
    <div class="element"></div>
    <div class="element"></div>
    <div class="element"></div>
    <div class="element"></div>
    <div class="element"></div>
    <div class="element"></div>
    <div class="element"></div>
    <div class="element"></div>
    <div class="element"></div>
    <div class="element"></div>
</div>
```

J'autorise mes éléments à aller à la ligne avec `flex-wrap` :

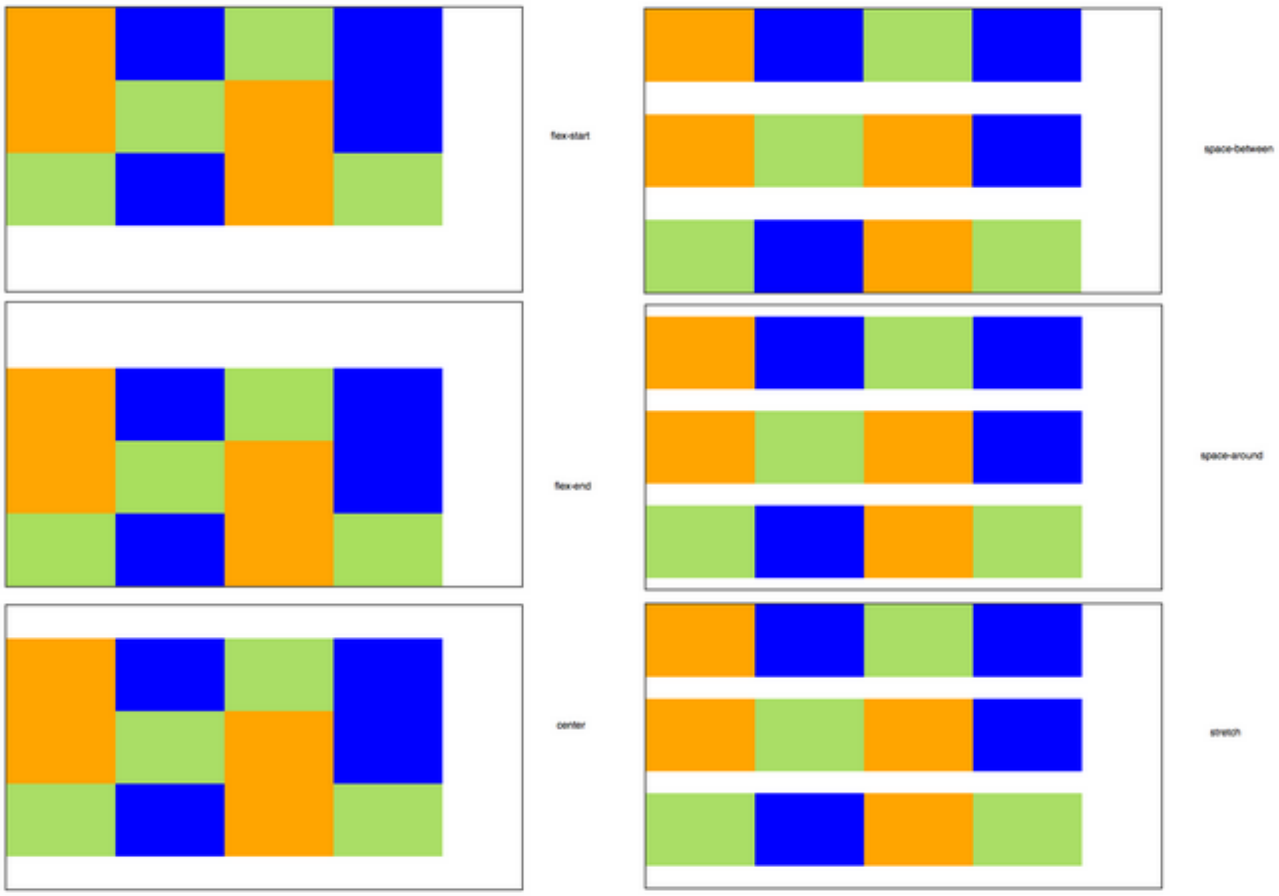
```css
#conteneur
{
    display: flex;
    flex-wrap: wrap;
}
```



Jusque là, rien de vraiment nouveau. Voyons voir comment les lignes se répartissent différemment avec la nouvelle propriété `align-content` que je voulais vous présenter. Elle peut prendre ces valeurs :

- flex-start : les éléments sont placés au début ;
- flex-end : les éléments sont placés à la fin ;
- center : les éléments sont placés au centre ;
- space-between : les éléments sont séparés avec de l'espace entre eux ;
- space-around : idem, mais il y a aussi de l'espace au début et à la fin ;
- stretch (par défaut) : les éléments s'étirent pour occuper tout l'espace.

Voici ce que donnent les différentes valeurs :



Les lignes sont placées différemment avec align-content

Rappel à l'ordre

Sans changer le code HTML, nous pouvons modifier l'ordre des éléments en CSS grâce à la propriété `order` . Indiquez simplement un nombre, et les éléments seront triés du plus petit au plus grand nombre.

Reprenons une simple ligne de 3 éléments :

```css
#conteneur
{
    display: flex;
}
```

Une ligne de 3 éléments

Si je dis que le premier élément sera placé en 3e position, le second en 1re position et le troisième en 2de position, l'ordre à l'écran change !

```css
.element:nth-child(1)
{
    order: 3;
}
.element:nth-child(2)
{
    order: 1;
}
.element:nth-child(3)
{
    order: 2;
}
```

Avec order, nous pouvons réordonner les éléments en CSS

La propriété `flex` est en fait une super-propriété qui combine `flex-grow` (capacité à grossir), `flex-shrink` (capacité à maigrir) et `flex-basis` (taille par défaut). J'utilise simplement flex comme je vous l'ai montré ici, mais si vous voulez en savoir plus, je vous invite à vous renseigner sur ces autres propriétés.

Rendez-vous au prochain chapitre 😊

Encore plus flex : faire grossir ou maigrir les éléments

Allez, encore une dernière technique, après on passe à la pratique.

Avec la propriété `flex` , nous pouvons permettre à un élément de grossir pour occuper tout l'espace restant.

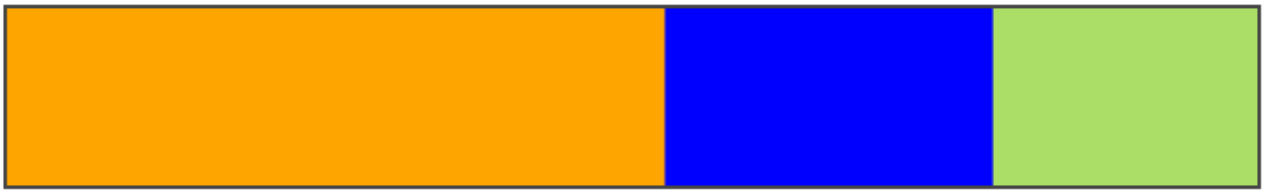
```css
.element:nth-child(2)
{
    flex: 1;
}
```

Le deuxième élément s'étire pour prendre tout l'espace

Le nombre que vous indiquez à la propriété flex indique dans quelle mesure il peut grossir par rapport aux autres.

```css
.element:nth-child(1)
{
    flex: 2;
}
.element:nth-child(2)
{
    flex: 1;
}
```

Ici, le premier élément peut grossir 2 fois plus que le deuxième élément :



Le premier élément peut grossir deux fois plus que le deuxième élément

En résumé

- Il existe plusieurs techniques pour positionner les blocs sur la page. Flexbox est la technique la plus récente et de loin la plus puissante, que je vous recommande d'utiliser.
- Le principe de Flexbox est d'avoir un conteneur, avec plusieurs éléments à l'intérieur. Avec `display: flex;` sur le conteneur, les éléments à l'intérieur sont agencés en mode Flexbox (horizontalement, par défaut).
- Flexbox peut gérer toutes les directions. Avec `flex-direction` , on peut indiquer si les éléments sont agencés horizontalement (par défaut) ou verticalement. Cela définit ce qu'on appelle l'*axe principal*.
- L'alignement des éléments se fait sur l'axe principal avec `justify-content` , et sur l'axe secondaire avec `align-items` .
- Avec `flex-wrap` , on peut autoriser les éléments à revenir à la ligne s'ils n'ont plus d'espace.
- S'il y a plusieurs lignes, on peut indiquer comment les lignes doivent se répartir entre elles avec `align-content` .
- Chaque élément peut être réagencé en CSS avec `order` (pas besoin de toucher au code HTML !).
- Avec la super-propriété `flex` , on peut autoriser nos éléments à occuper plus ou moins d'espace restant.
- Flexbox, c'est cool.

❖ Mémento des balises HTML

Cette page est une liste *non exhaustive* des balises HTML qui existent. Vous trouverez ici un grand nombre de balises HTML. Nous en avons déjà vu certaines dans le cours, mais il y en a d'autres que nous n'avons pas eu l'occasion d'étudier. Généralement, les balises que nous n'avons pas étudiées sont des balises un peu plus rarement utilisées. Peut-être trouverez-vous votre bonheur dans ce lot de nouvelles balises.

Vous pouvez vous servir de cette annexe comme d'un aide-mémoire lorsque vous développez votre site web.

Attention, j'insiste : *cette annexe n'est pas complète et c'est volontaire*. Je préfère mettre *moins* de balises et garder seulement celles qui me semblent les plus utiles dans la pratique.

Balises de premier niveau

Les balises de premier niveau sont les principales balises qui structurent une page HTML. Elles sont indispensables pour réaliser le « code minimal » d'une page web.

Balise	Description
`<html>`	Balise principale
`<head>`	En-tête de la page
`<body>`	Corps de la page

Code minimal d'une page HTML :

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Titre</title>
    </head>

    <body>

    </body>
</html>
```

Balises d'en-tête

Ces balises sont toutes situées dans l'en-tête de la page web, c'est-à-dire entre `<head>` et `</head>` :

Balise	Description
`<link />`	Liaison avec une feuille de style
`<meta />`	Métadonnées de la page web (charset, mots-clés, etc.)
`<script>`	Code JavaScript
`<style>`	Code CSS
`<title>`	Titre de la page

Balises de structuration du texte

Balise	Description
`<abbr>`	Abréviation
`<blockquote>`	Citation (longue)
`<cite>`	Citation du titre d'une œuvre ou d'un évènement
`<q>`	Citation (courte)
`<sup>`	Exposant
`<sub>`	Indice
`<strong>`	Mise en valeur forte
`<em>`	Mise en valeur normale
`<mark>`	Mise en valeur visuelle
`<h1>`	Titre de niveau 1
`<h2>`	Titre de niveau 2
`<h3>`	Titre de niveau 3
`<h4>`	Titre de niveau 4
`<h5>`	Titre de niveau 5
`<h6>`	Titre de niveau 6
`<img />`	Image
`<figure>`	Figure (image, code, etc.)
`<figcaption>`	Description de la figure

`<audio>`	Son
`<video>`	Vidéo
`<source>`	Format source pour les balises `<audio>` et `<video>`
`<a>`	Lien hypertexte
` `	Retour à la ligne
`<p>`	Paragraphe
`<hr />`	Ligne de séparation horizontale
`<address>`	Adresse de contact
`<del>`	Texte supprimé
`<ins>`	Texte inséré
`<dfn>`	Définition
`<kbd>`	Saisie clavier
`<pre>`	Affichage formaté (pour les codes sources)
`<progress>`	Barre de progression
`<time>`	Date ou heure

Balises de listes

Cette section énumère toutes les balises HTML permettant de créer des listes (listes à puces, listes numérotées, listes de définitions...)

Balise	Description
`<ul>`	Liste à puces, non numérotée
`<ol>`	Liste numérotée
`<li>`	Élément de la liste à puces
`<dl>`	Liste de définitions
`<dt>`	Terme à définir
`<dd>`	Définition du terme

Balises sectionnantes

Ces balises permettent de construire le squelette de notre site web.

Balise	Description
`<header>`	En-tête
`<nav>`	Liens principaux de navigation
`<footer>`	Pied de page
`<section>`	Section de page
`<article>`	Article (contenu autonome)
`<aside>`	Informations complémentaires

Balises génériques

Les balises génériques sont des balises qui n'ont pas de sens sémantique.

En effet, toutes les autres balises HTML ont un *sens* : `<p>` signifie « paragraphe », `<h2>` signifie « sous-titre », etc.
Parfois, on a besoin d'utiliser des balises génériques (aussi appelées ***balises universelles***) car aucune des autres balises ne convient. On utilise le plus souvent des balises génériques pour construire son design.

Il y a deux balises génériques : l'une est inline, l'autre est block.

Balise	Description
`<span>`	Balise générique de type inline
`<div>`	Balise générique de type block

Ces balises ont un intérêt uniquement si vous leur associez un attribut `class` , `id` ou `style` :

- `class` : indique le nom de la classe CSS à utiliser.
- `id` : donne un nom à la balise. Ce nom doit être unique sur toute la page car il permet d'identifier la balise. Vous pouvez vous servir de l'ID pour de nombreuses choses, par exemple pour créer un lien vers une ancre, pour un style CSS de type ID, pour des manipulations en JavaScript, etc.
- `style` : cet attribut vous permet d'indiquer directement le code CSS à appliquer. Vous n'êtes donc pas obligé d'avoir une feuille de style à part, vous pouvez mettre directement les attributs CSS. Notez qu'il est préférable de ne pas utiliser cet attribut et de passer à la place par une feuille de style externe, car cela rend votre site plus facile à mettre à jour par la suite.

Ces trois attributs ne sont pas réservés aux balises génériques : vous pouvez aussi les utiliser sans aucun problème dans la plupart des autres balises.

❖ Mémento des propriétés CSS

Propriété	Valeurs (exemples)	Description
`font-family`	*police1, police2, police3*, serif, sans-serif, monospace	Nom de police
`@font-face`	*Nom et source de la police*	Police personnalisée
`font-size`	1.3em, 16px, 120%...	Taille du texte
`font-weight`	bold, normal	Gras
`font-style`	italic, oblique, normal	Italique
`text-decoration`	underline, overline, line-through, blink, none	Soulignement, ligne au-dessus, barré ou clignotant
`font-variant`	small-caps, normal	Petites capitales
`text-transform`	capitalize, lowercase, uppercase	Capitales
`font`	-	Super propriété de police. Combine : `font-weight` , `font-style` , `font-size` , `font-variant` , `font-family` .
`text-align`	left, center, right, justify	Alignement horizontal
`vertical-align`	baseline, middle, sub, super, top, bottom	Alignement vertical (cellules de tableau ou éléments `inline-block` uniquement)
`line-height`	18px, 120%, normal...	Hauteur de ligne

`word-wrap`	break-word, normal	Césure forcée
`text-shadow`	5px 5px 2px blue *(horizontale, verticale, fondu, couleur)*	Ombre de texte

Propriétés de couleur et de fond

Propriété	Valeurs (exemples)	Description
`color`	*nom*, rgb(rouge,vert,bleu), rgba(rouge,vert,bleu,transparence), #CF1A20...	Couleur du texte
`background-color`	*Identique à color*	Couleur de fond
`background-image`	url('image.png')	Image de fond
`background-attachment`	fixed, scroll	Fond fixe
`background-repeat`	repeat-x, repeat-y, no-repeat, repeat	Répétition du fond
`background-position`	*(x y)*, top, center, bottom, left, right	Position du fond
`background`	-	Super propriété du fond. Combine : `background-image` , `background-repeat` , `background-attachment` , `background-position`

Propriétés des boîtes

Propriété	Valeurs (exemples)	Description
`width`	150px, 80%...	Largeur
`height`	150px, 80%...	Hauteur
`min-width`	150px, 80%...	Largeur minimale
`max-width`	150px, 80%...	Largeur maximale
`min-height`	150px, 80%...	Hauteur minimale
`max-height`	150px, 80%...	Hauteur maximale
`margin-top`	23px	Marge en haut
`margin-left`	23px	Marge à gauche
`margin-right`	23px	Marge à droite
`margin-bottom`	23px	Marge en bas
`margin`	23px 5px 23px 5px *(haut, droite, bas, gauche)*	Super-propriété de marge. Combine : `margin-top` , `margin-right` , `margin-bottom` , `margin-left` .
`padding-left`	23px	Marge intérieure à gauche

`padding-right`	23px	Marge intérieure à droite
`padding-bottom`	23px	Marge intérieure en bas
`padding-top`	23px	Marge intérieure en haut
`padding`	23px 5px 23px 5px *(haut, droite, bas, gauche)*	Super-propriété de marge intérieure. Combine : `padding-top` , `padding-right` , `padding-bottom` , `padding-left` .
`border-width`	3px	Épaisseur de bordure
`border-color`	*nom*, rgb(rouge,vert,bleu), rgba(rouge,vert,bleu,transparence), #CF1A20...	Couleur de bordure
`border-style`	solid, dotted, dashed, double, groove, ridge, inset, outset	Type de bordure
`border`	3px solid black	Super-propriété de bordure. Combine , `border-width` , `border-color` , `border-style` . Existe aussi en version `border-top` , `border-right` , `border-bottom` , `border-left` .
`border-radius`	5px	Bordure arrondie
`box-shadow`	6px 6px 0px black *(horizontale, verticale, fondu, couleur)*	Ombre de boîte

Propriétés de positionnement et d'affichage

Propriété	Valeurs (exemples)	Description
`display`	block, inline, inline-block, table, table-cell, none...	Type d'élément (`block` , `inline` , `inline-block` , `none` ...)
`visibility`	visible, hidden	Visibilité
`clip`	rect (0px, 60px, 30px, 0px) *rect (haut, droite, bas, gauche)*	Affichage d'une partie de l'élément
`overflow`	auto, scroll, visible, hidden	Comportement en cas de dépassement
`float`	left, right, none	Flottant
`clear`	left, right, both, none	Arrêt d'un flottant
`position`	relative, absolute, static	Positionnement
`top`	20px	Position par rapport au haut
`bottom`	20px	Position par rapport au bas
`left`	20px	Position par rapport à la gauche
`right`	20px	Position par rapport à la droite
`z-index`	10	Ordre d'affichage en cas de superposition. La plus grande valeur est affichée par-dessus les autres.

Propriétés des listes

Propriété	Valeurs (exemples)	Description
`list-style-type`	disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, none	Type de liste
`list-style-position`	inside, outside	Position en retrait
`list-style-image`	url('puce.png')	Puce personnalisée
`list-style`	-	Super-propriété de liste. Combine `list-style-type` ,, `list-style-position` , `list-style-image` .

Propriétés des tableaux

Propriété	Valeurs (exemples)	Description
`border-collapse`	collapse, separate	Fusion des bordures
`empty-cells`	hide, show	Affichage des cellules vides
`caption-side`	bottom, top	Position du titre du tableau

Autres propriétés

Propriété	Valeurs (exemple)	Description
`cursor`	crosshair, default, help, move, pointer, progress, text, wait, e-resize, ne-resize, auto...	Curseur de souris

Module 6 : Transmissions et réseaux

6.1 Conception et réalisation d'une mise en réseau

Notion de réseau, de réseaux informatiques

Réseau au sens large

En premier ressort, un réseau désigne au sens concret « un ensemble de lignes entrelacées » et, au figuré « un ensemble de relations »
Par extension, il désigne un ensemble interconnecté — fait de composants et de leurs interrelations — autorisant la circulation en mode continu ou discontinu de flux (eau, air, huile...) ou d'éléments finis (marchandises, informations, personnes...).
Le Réseau n être « matériel » (comme le réseau sanguin), « semi-matériel » (comme le réseau lymphatique), « immatériel » (comme le réseau social), « abstrait, symbolique ou normalisé » (comme le Réseau Pert).

Réseaux informatiques

Un réseau informatique est un ensemble d'équipements reliés entre eux pour échanger des informations), on appelle nœud (Node (networking) (en)) l'extrémité d'une connexion, qui peut être une intersection de plusieurs connexions ou équipements (un ordinateur, un routeur, concentrateur, un commutateur).
Indépendamment de la technologie sous-jacente, on porte généralement une vue matricielle sur ce qu'est un réseau. De façon horizontale, un réseau est une strate de trois couches : les infrastructures, les fonctions de contrôle et de commande, les services rendus à l'utilisateur. De façon verticale, on utilise souvent un découpage géographique : réseau local, réseau d'accès et réseau

6.2 Notions de client/serveur et de réseau peer-to-peer

L'environnement client-serveur désigne un mode de communication à travers un réseau entre plusieurs programmes ou logiciels : l'un, qualifié de client, envoie des requêtes ; l'autre ou les autres, qualifiés de serveurs, attendent les requêtes des clients et y répondent.

Par extension, le client désigne également l'ordinateur sur lequel est exécuté le logiciel client, et le serveur, l'ordinateur sur lequel est exécuté le logiciel serveur.
En général, les serveurs sont des ordinateurs dédiés au logiciel serveur qu'ils abritent, et dotés de capacités supérieures à celles des ordinateurs personnels en termes de puissance de calcul, d'entrées-sorties et de connexions réseau. Les clients sont souvent des ordinateurs personnels ou des appareils individuels (téléphone, tablette), mais pas systématiquement. Un serveur peut répondre aux requêtes d'un grand nombre de clients.

Il existe une grande variété de logiciels serveurs et de logiciels clients en fonction des besoins à servir : un serveur web publie des pages web demandées par des navigateurs web ; un serveur de messagerie électronique envoie des mails à des clients de messagerie ; un serveur de fichiers permet de stocker et consulter des fichiers sur le réseau ; un serveur de données à communiquer des données stockées dans une base de données, etc.

Réseau pair à pair (peer to peer – P2P)

Le pair à pair ou pair-à-pair (traduction de l'anglicisme peer-to-peer, souvent abrégé « P2P ») est un modèle de réseau informatique proche du modèle client-serveur mais où chaque client est aussi un serveur.
Le pair à pair peut être centralisé (les connexions passant par un serveur central intermédiaire) ou décentralisé (les connexions se faisant directement). Il peut servir au partage de fichiers en pair à pair, au calcul distribué ou à la communication.
Un partage de fichiers en pair-à-pair est un réseau qui permet de partager des fichiers entre plusieurs ordinateurs connectés entre eux par Internet, chaque internaute pouvant être serveur et receveur d'un autre internaute. Ils forment ainsi des « pairs ».

Serveur

Caractéristiques d'un processus serveur :

1. il attend une connexion entrante sur un ou plusieurs ports réseaux ;
2. à la connexion d'un client sur le port en écoute, il ouvre un socket local au système d'exploitation;
3. suite à la connexion, le processus serveur communique avec le client suivant le protocole prévu par la couche application du modèle OSI.

Client

Caractéristiques d'un processus client :

- il établit la connexion au serveur à destination d'un ou plusieurs ports réseaux ;
- lorsque la connexion est acceptée par le serveur, il communique comme le prévoit la couche applicative du modèle OSI.

Le client et le serveur doivent bien sûr utiliser le même protocole de communication au niveau de la couche transport du modèle OSI. Un serveur est généralement capable de servir plusieurs clients simultanément. On parle souvent d'un service pour désigner la fonctionnalité offerte par un processus serveur. On définit aussi comme serveur, un ordinateur spécialisé ou une machine virtuelle ayant pour unique tâche l'exécution d'un ou plusieurs processus serveur.
Protocole

Un protocole réseau est un protocole de communication mis en œuvre sur un réseau informatique.

Dans les réseaux informatiques et les télécommunications, un protocole de communication est une spécification de plusieurs règles pour un type de communication particulier.

Initialement, on nommait protocole ce qui est utilisé pour communiquer sur une même couche d'abstraction entre deux machines différentes.
Par extension de langage, on utilise parfois ce mot aussi aujourd'hui pour désigner les règles de communication entre deux couches sur une même machine.

Les protocoles de communication les plus utilisés sont les protocoles réseau.
TCP/IP est l'ensemble des protocoles utilisés pour le transfert des données sur Internet. Elle est souvent appelée TCP/IP, d'après le nom de deux de ses protocoles : TCP (Transmission Control Protocol) et IP (Internet Protocol), qui ont été les premiers à être définis.

6.3 Modèle de réseau OSI

Le modèle OSI (de l'anglais Open Systems Interconnection) est un standard de communication, en réseau, de tous les systèmes informatiques. C'est un modèle de communications entre ordinateurs proposé par l'ISO qui décrit les fonctionnalités nécessaires à la communication et l'organisation de ces fonctions.

La norme complète, de référence ISO 7498 est globalement intitulée « Modèle basique de référence pour l'interconnexion des systèmes ouverts (OSI) » et est composée de 4 parties :

- Le modèle de base
- Architecture de sécurité
- Dénomination et adressage
- Cadre général de gestion

La version de cet article ainsi que les articles consacrés à chacune des couches du modèle se concentrent sur la partie 1, révision de 1994. L'UIT-T en a approuvé le texte à l'identique sous le numéro de recommandation X.200 en 1994.
Le texte de la norme proprement dite est très abstrait car il se veut applicable à de nombreux types de réseaux. Pour la rendre plus compréhensible, en plus de présenter la norme, cet article fait des liens avec les réalisations concrètes telles qu'on les trouve dans un ordinateur, c’est-à-dire des piles protocolaires concrètes (un « système réel » au sens de la section 4). De plus, la norme n'indique pas de mécanismes propres à assurer les fonctions définies alors que cet article le fait. Les exemples de services et surtout de protocoles sont pris dans le monde IP (probablement le plus connu mais aussi le plus éloigné de l'esprit de la norme), le monde RNIS (y compris la seconde génération, plus connue sous le nom ATM) et parfois le monde OSI (qui ne fait pas que des modèles).

Les combinaisons offertes par le modèle sont beaucoup plus nombreuses que celles réalisées dans des piles de protocoles existantes, on ne peut donc pas donner d'exemple réel pour toutes les fonctions.

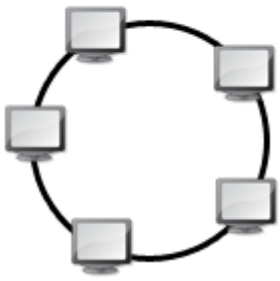
Le modèle est essentiellement une architecture en couches définies et délimitées avec les notions de service, de protocole et d'interface.
Un service est une description abstraite de fonctionnalités à l'aide de primitives telles que demande de connexion ou réception de données.
Un protocole est un ensemble de messages et de règles d'échange réalisant un service.
Une interface (« point d'accès au service » dans la norme) est le moyen concret d'utiliser le service. Dans un programme, c'est typiquement un ensemble de fonctions de bibliothèque ou d'appels systèmes. Dans une réalisation matérielle, c'est par exemple un jeu de registres à l'entrée d'un circuit.
Les détails d'un service varient bien sûr d'une architecture de réseau à l'autre.

6.4 Notions de topologie

Une topologie de réseau est en informatique une définition de l'architecture d'un réseau. Définissant les connexions entre ces postes et une hiérarchie éventuelle entre eux, elle peut avoir des implications sur la disposition géographique des différents postes informatiques du réseau. Ainsi Ethernet peut avoir comme support un simple plafond blanc visible de tous les postes (voir LiFi), alors que cela sera par construction impossible en token ring, bien que possible en token bus

Reconnaître une topologie

Le réseau en anneau



Un réseau a une topologie en anneau quand toutes ses stations sont connectées en chaine les unes aux autres par une liaison bipoint et la dernière à la première. Chaque station joue le rôle de station intermédiaire. Chaque station qui reçoit une trame, l'interprète et la réémet à la station suivante de la boucle si c'est nécessaire. La défaillance d'un hôte rompt la structure d'un réseau en anneau si la communication est unidirectionnelle ; en pratique un réseau en anneau est souvent composé de 2 anneaux contrarotatifs.

Note : les ordinateurs d'un réseau en anneau ne sont pas systématiquement reliés en boucle, mais peuvent être connectés à un répartiteur appelé « MAU », (pour Multi station Access Unit) qui va gérer la communication entre les ordinateurs reliés en allouant à chacun d'eux un « temps de parole ».

En cas de collision de deux messages, les deux seraient perdus, mais les règles d'accès à l'anneau (par exemple, la détention d'un jeton) sont censées éviter ce cas de figure.

Le réseau hiérarchique (réseau en arbre)

Aussi connu sous le nom de réseau en arbre, il est divisé en niveaux. Le sommet, de haut niveau, est connectée à plusieurs nœuds de niveau inférieur, dans la hiérarchie. Ces nœuds peuvent être eux-mêmes connectés à plusieurs nœuds de niveau inférieur. Le tout dessine alors un arbre, ou une arborescence. Le point faible de ce type de topologie réside dans l'ordinateur "père" de la hiérarchie qui, s'il tombe en panne, paralyse alors la moitié du réseau.

Le réseau en bus

Cette topologie est représentée par un câblage unique des unités réseaux. Il a également un faible coût de déploiement et la défaillance d'un nœud (ordinateur) ne scinde pas le réseau en deux sous-réseaux. Ces unités sont reliées de façon passive par dérivation électrique ou optique. Les caractéristiques de cette topologie sont les suivantes :
Lorsqu'une station est défectueuse et ne transmet plus sur le réseau, elle ne perturbe pas le réseau.
Lorsque le support est en panne, c'est l'ensemble du réseau qui ne fonctionne plus.
Le signal émis par une station se propage dans un seul sens ou dans les deux sens.
Si la transmission est bidirectionnelle : toutes les stations connectées reçoivent les signaux émis sur le bus en même temps (au délai de propagation près).

Topologie de réseau en étoile

C'est la topologie la plus courante actuellement. Omniprésente, elle est aussi très souple en matière de gestion et dépannage de réseau : la panne d'un nœud ne perturbe pas le fonctionnement global du réseau. En revanche, l'équipement central (un concentrateur (hub) et plus souvent sur les réseaux modernes, un commutateur (switch)) qui relie tous les nœuds constitue un point unique de défaillance : une panne à ce niveau rend le réseau totalement inutilisable. Le réseau Ethernet est un exemple de topologie en étoile. L'inconvénient principal de cette topologie réside dans la longueur des câbles utilisés

Le réseau linéaire
Il a pour avantage son faible coût de déploiement, mais la défaillance d'un nœud (ordinateur) peut scinder le réseau en deux sous-réseaux.

Le réseau maillé

Une topologie maillée correspond à plusieurs liaisons point à point. (Une unité réseau peut avoir (1,N) connexions point à point vers plusieurs autres unités.) Chaque terminal est relié à tous les autres. L'inconvénient est le nombre de liaisons nécessaires qui devient très élevé lorsque le nombre de terminaux l'est : s'il y a N terminaux, le nombre de liaisons nécessaires est de (n*(n-1))/2 fonction qui croît comme n2
Cette topologie se rencontre dans les grands réseaux de distribution (Exemple : Internet).

L'information peut parcourir le réseau suivant des itinéraires divers, sous le contrôle de puissants superviseurs de réseau, ou grâce à des méthodes de routage réparties.
Elle existe aussi dans le cas de couverture Wi-Fi.

Les autres topologies

Il existe d'autres types de topologies, mais elles ne sont utilisées que dans des réseaux conçus pour des tâches particulières, souvent scientifiques, ou pour effectuer des calculs distribués :

- le réseau en grille
- le réseau en hypercube

Cette dernière topologie serait en théorie la meilleure qui soit (chaque nœud étant relié à chaque autre par un lien direct), mais dans la pratique elle serait aussi la plus chère et de très loin (coût en N² avec le nombre de nœuds !) ; elle n'est utilisée que dans des dispositifs particuliers, en général *à l'intérieur* d'un même châssis.

6.5 Réseaux Ethernet


Connecteur RJ45 pour Ethernet

Le nom Ethernet est un protocole de réseau local à commutation de paquets. Bien qu'il implémente la couche physique (PHY) et la sous-couche Media Access Control (MAC) du modèle IEEE 802.3, le protocole Ethernet est classé dans les couche de liaison de données et physique, puisque la couche LLC (Logical Link Control) 802.2 fait la charnière entre les couches supérieures et la sous-couche MAC (Media Access Control) qui fait partie intégrante du processus 802.3 avec la couche physique, car les formats de trames que le standard définit sont normalisés et peuvent être encapsulés dans des protocoles autres que ses propres couches physiques MAC et PHY. Ces couches physiques font l'objet de normes séparées en fonction des débits, du support de transmission, de la longueur des liaisons et des conditions environnementales

6.6 Le Wifi

Le Wi-Fi est un ensemble de protocoles de communication sans fil régi par les normes du groupe IEEE 802.11 (ISO/CEI 8802-11). Un réseau Wi-Fi permet de relier sans fil plusieurs appareils informatiques (ordinateur, routeur, décodeur Internet, etc.) au sein d'un réseau informatique afin de permettre la transmission de données entre eux.

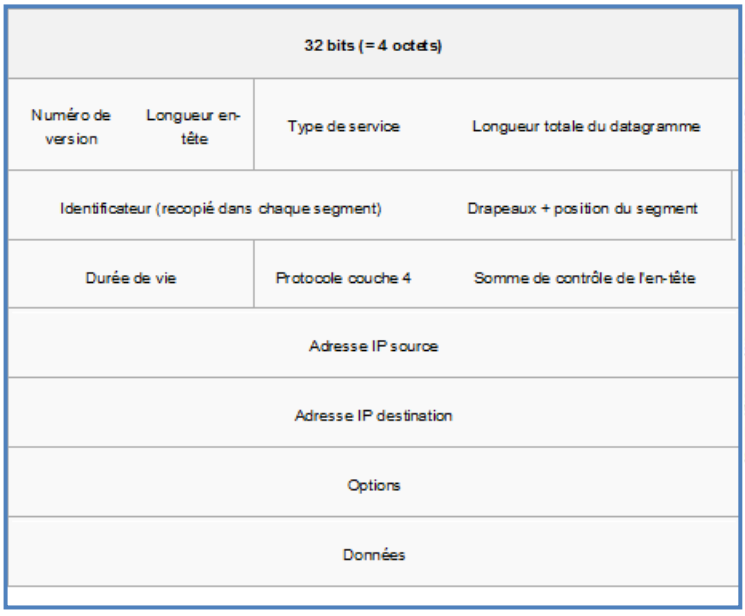
Internet et les réseaux en général

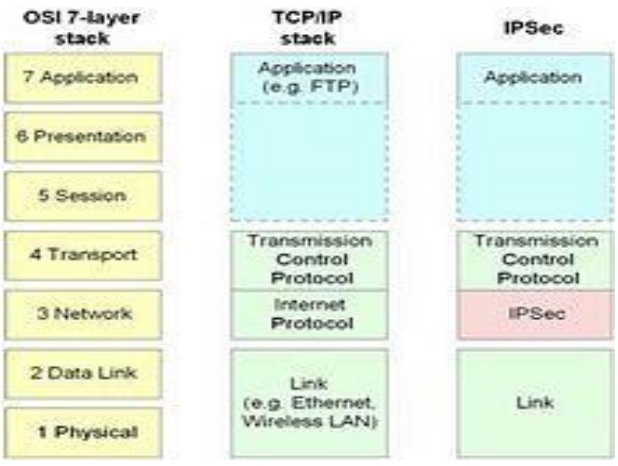
Internet est le nom donné à l'interconnexion de plusieurs réseaux, potentiellement de topologies différentes, l'unification n'en étant faite qu'au niveau du seul adressage IP (v4 ou v6) et d'un grand nombre de protocoles et règles définis par l'IETF. De ce fait, aucun des cas particuliers de topologies citées ci-dessus ne s'applique ; comme pour la plupart des grands réseaux, on dit d'Internet que sa topologie est quelconque, et de toute façon indépendante du plan d'adressage qui y est défini.

6.7 Notions de trame et de datagramme

Le datagramme est un paquet de données dans un réseau informatique utilisé par des protocoles orientés non connectés tel que : IPX ou UDP. Le datagramme est le terme généralement utilisé pour désigner la transmission d'un paquet via un service non « fiable » : il n'y a aucun moyen de s'assurer que le paquet est arrivé à sa destination (un peu comme une lettre sans accusé de réception).
En informatique, ce concept est également utilisé de manière plus générale pour décrire des blocs de données. Il est notamment utilisé pour décrire un bloc de donnée transféré à une fonction, dans l'esprit des réseaux, mais appliqué entre processus internes d'un ordinateur.

La pile TCP/IP





La pile TCP/IP et le modèle OSI
Le datagramme IP (version 4)

Lorsque deux machines communiquent en utilisant le protocole IP, elles s'échangent des datagrammes IP qui ont le format ci-dessous :

Version *(4 bits)* :
Le champ version indique la version utilisée du protocole IP. Début 2006, la version de IP la plus fréquemment utilisée est la version 4. La version 6 commence à apparaitre : il n'y aura pas de version 5. Les 4 bits de ce champ sont donc 0100 (codage en binaire de la valeur décimale 4).

IHL = IP Header Length - longueur de l'en-tête IP *(4 bits)* :
Ce champ indique la longueur de l'entête IP. L'unité est le nombre de mots de 32 bits. Pour la version 4 la longueur de cette entête est de 20 octets soit 5 fois 32 bits : ce champ vaut donc 0101.

Type of service *(8 bits)* :
Ce champ permet d'indiquer que certains datagrammes IP ont une priorité supérieure à d'autres.

Total length *(16 bits)* :
Ce champ indique le nombre d'octets du datagramme, en-tête IP comprise. La longueur maximale du datagramme en octets est 2^{16}-1 = 65 535 octets = 64 ko -1 octet

ID *(16 bits)* :
Ce champ est un identifiant du datagramme IP (le numéro du datagramme).

F = Flags - les drapeaux *(3 bits)* :
le premier bit est inutilisé
le deuxième bit DF (don't fragment) permet d'interdire ou d'autoriser la fragmentation. Positionné à 1, il est interdit de fragmenter ce datagramme IP.
le troisième bit MF (more fragment) est utilisé lors de la fragmentation : il indique si le fragment est le dernier fragment du datagramme (MF=0) ou non (MF=1).

TTL = Time to live - temps restant à vivre *(8 bits)* :
Il s'agit d'une valeur initialisée par l'émetteur et qui est décrémentée de 1 à chaque fois que le datagramme traverse un routeur. Si le TTL arrive à la valeur 0, le datagramme est détruit : ce mécanisme assure la destruction des datagrammes qui se perdent sur le réseau.

Protocole *(8 bits)* :
Ce champ indique la nature des données transportées par ce datagramme IP. 3 protocoles sont principalement utilisés au-dessus de IP : ICMP (code 1), TCP (code 6) et UDP (code 17).

Header Checksum - somme de contrôle de l'en-tête *(16 bits)* :
Il s'agit d'un code détecteur d'erreurs qui ne porte que sur l'entête : la somme des octets de l'entête regroupé par paquets de 16 bits (header checksum compris) doit valoir 2^{16}-1 modulo 2^{16}. En cas d'erreur sur l'entête le datagramme est détruit. IP n'est pas un protocole fiable puisqu'on ne garantit pas que les données arrivent, ni de leur fiabilité.

IP source *(32 bits)* :
Adresse IP de l'expéditeur.

IP destination *(32 bits)* :
Adresse IP du destinataire.

6.8 **Le protocole TCP/IP**

Un système de communication doit pouvoir permettre à n'importe quel hôte de se mettre en relation avec n'importe quel autre. Afin qu'il n'y ait pas d'ambiguïté pour la reconnaissance des hôtes possibles, il est absolument nécessaire d'admettre un principe général d'identification :
1. Le nom de la machine distante,
2. son adresse,
3. la route à suivre pour y parvenir.
Le nom dit « qui » est l'hôte distant, l'adresse nous dit « où » il se trouve et la route « comment » on y parvient.
Les adresses IP (version 4) sont standardisées sous forme d'un nombre de 32 bits qui permet à la fois l'identification de chaque hôte et du réseau auquel il appartient. Chaque adresse IP contient donc deux informations basiques, une adresse de réseau et une adresse d'hôte. La combinaison des deux désigne de manière unique une machine et une seule sur l'Internet.

6.8.1 **Délivrance des adresses IPv4**

On distingue deux types d'adresses IP. Les adresses privées que tout administrateur de réseau peut s'attribuer librement pourvu qu'elles ne soient pas routées sur l'Internet, et les adresses publiques, délivrées par une structure mondiale qui en assure l'unicité.
C'est L'ICANN (Internet Corporation for Assigned Names and Numbers) qui est chargé au niveau mondial de la gestion de l'espace d'adressage IP. Il définit les procédures d'attribution et de résolution de conflits dans l'attribution des adresses, mais délègue le détail de la gestion de ces ressources à des instances régionales puis locales, dans chaque pays, appelées « Regional Internet Registries » ou RIR.
Il y a actuellement trois « Regional Internet Registries » opérationnels :
• l'APNIC pour la région Asie-Pacifique
• l'ARIN pour l'Amérique
• le RIPE NCC pour l'Europe
• l'AfriNIC pour l'Afrique
• le LACNIC pour l'Amérique Latine

6.8.2 **Anatomie d'une adresse IP**

À l'origine, plusieurs groupes d'adresses ont été définis dans le but d'optimiser le cheminement (ou le routage) des paquets entre les différents réseaux. Ces groupes ont été baptisés classes d'adresses IP. Ces classes correspondent à des regroupements en réseaux de même taille. Les réseaux de la même classe ont le même nombre d'hôtes maximum.

À l'origine, plusieurs groupes d'adresses ont été définis dans le but d'optimiser le cheminement (ou le routage) des paquets entre les différents réseaux. Ces groupes ont été baptisés classes d'adresses IP. Ces classes correspondent à des regroupements en réseaux de même taille. Les réseaux de la même classe ont le même nombre d'hôtes maximum.

Une adresse IP est un nombre de 32 bits que l'on a coutume de représenter sous forme de quatre entiers de huit bits, séparés par des points.
La partie réseau de l'adresse IP vient toujours en tête, la partie hôte est donc toujours en queue.

L'intérêt de cette représentation est immédiat quand on sait que la partie réseau et donc la partie hôte sont presque toujours codées sur un nombre entier d'octets. Ainsi, on a principalement les trois formes suivantes :

• Classe A : Un octet réseau, trois octets d'hôtes.
• Classe B : Deux octets réseau, deux octets d'hôtes.
• Classe C : Trois octets réseau, un octet d'hôte.

Deux formats permettent de faire référence à une adresse IP le format binaire et la notation décimale pointée.

Conversion d'adresses IP du format binaire en format décimal

Le bit de poids faible représente la valeur décimale 1, le bit de poids fort la valeur décimale 128. La valeur décimale la plus élevée d'un octet est 255, tous les bits sont mis à 1.

Le tableau suivant indique comment les bits d'un octet sont convertis d'un code binaire en une valeur décimale.

Code binaire	Valeurs binaires	Valeur décimale
00000000	**0**	**0**
00000001	**1**	**1**
00000011	**1+2**	**3**
00000111	**1+2+4**	**7**
00001111	**1+2+4+8**	**15**
00011111	**1+2+4+8+16**	**31**
00111111	**1+2+4+8+16+32**	**63**
01111111	**1+2+4+8+16+32+64**	**127**
11111111	**1+2+4+8+16+32+64+128**	**255**

Conversion décimal – binaire

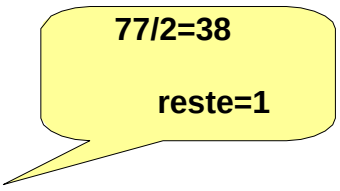
Convertissons 01001101 en décimal à l'aide du schéma ci-dessous :

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	0	1	1	0	1

Le nombre en base 10 est $2^6 + 2^3 + 2^2 + 2^0 = 64 + 8 + 4 + 1 = 77$.

Allons maintenant dans l'autre sens et écrivons 77 en base 2. Il s'agit de faire une suite de divisions euclidiennes par 2. Le résultat sera la juxtaposition des restes.
La méthode : On met le nombre à droite, on le divise par 2 et on note le résultat à sa droite et le reste juste en-dessous

nombre_a_gauche/2	**0**	**1**	**2**	**4**	**9**	**19**	**38**	**77**
reste	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**



6.8.3 **Décomposition en classes**

Classe	Masque de sous réseau par défaut	Adresse réseau	Nombre de réseaux	Nombre d'hôtes
A	255.0.0.0	1.0.0.0 à 126.0.0.0	126 (= 2^7 - 2)	16 777 214
B	255.255.0.0	128.0.0.0 à 191.255.0.0	16 384 (=2^{14})	65 534
C	255.255.255.0	192.0.0.0 à 223.255.255.0	2 097 152 (=2^{21})	254
D	Non défini	224.0.0.0 à 239.255.255.0		
E	Non défini	240.0.0.0 à 255.255.255.0		

Pour distinguer les classes A, B, C, D et E il faut examiner les bits de poids fort de l'octet de poids fort. Ce premier octet désigne le numéro de réseau (NetID) et les 3 autres correspondent à l'adresse de l'hôte (HostID).

Si le premier bit est 0, l'adresse est de classe A. On dispose de 7 bits pour identifier le réseau et de 24 bits pour identifier l'hôte. On a donc les réseaux de 1 à 127 et 224 hôtes possibles, c'est à dire 16 777 216 machines différentes (de 0 à 16 777 215).

Remarque : l'adresse réseau 0.0.0.0 n'existe pas et le NetID « 127 » est réservée pour les communications en boucle locale (loopback), ce qui réduit de deux unités le nombre des machines nommables. Il reste donc seulement 16 777 214 machines adressables dans une classe A !

Si les deux premiers bits sont 10, l'adresse est de classe B. Il reste 14 bits pour identifier le réseau et 16 bits pour identifier la machine. Ce qui fait 214= 16 384 réseaux (128.0 à 191.255) et 65 534 (65 536 - 2) machines.

Si les trois premiers bits sont 110, l'adresse est de classe C. Il reste 21 bits pour identifier le réseau et 8 bits pour identifier la machine. Ce qui fait 221=2 097 152 réseaux (de 192.0.0 à 223.255.255) et 254 (256 - 2) machines.

Adresses particulières

Il existe un certain nombre d'adresses IP réservées :

hostid = 0 désigne le réseau lui même
L'hostid égal à 0 ne sera jamais affecté à un hôte mais il désigne le réseau lui-même.
Exemple : 192.145.56.0 est un réseau de classe C dont l'hostid est à 0 donc cette adresse désigne le réseau lui-même.

0.0.0.0 désigne l'hôte lui-même

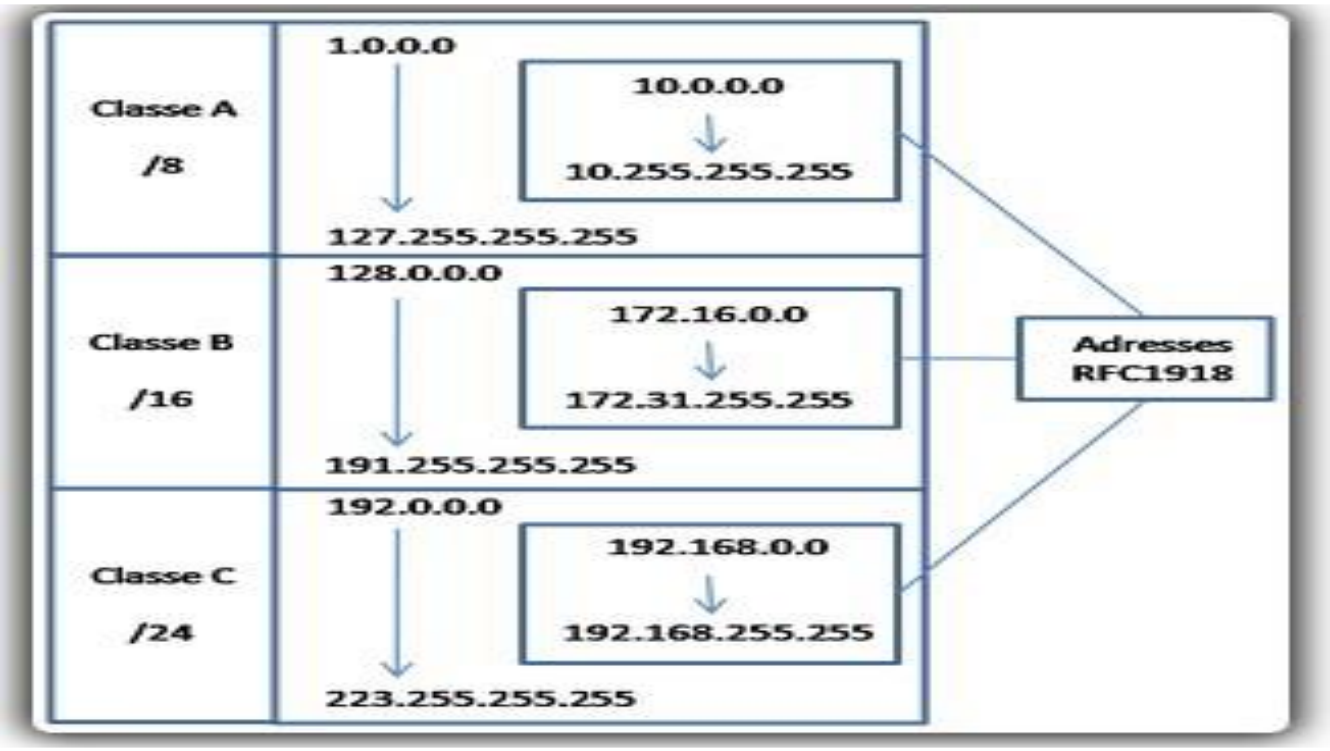
Lorsque tous les bits d'une adresse IP sont à 0, cela signifie "cet hôte-ci sur ce réseau". Cette adresse spéciale est utilisée par un hôte afin d'obtenir une adresse IP de manière dynamique dans le cas du protocole BOOTP.

255.255.255.255 = diffusion limitée
Une diffusion limitée (limited broadcast) est un message qui est envoyé à tous les hôtes du réseau dont fait partie l'expéditeur. La diffusion limitée est représentée par l'adresse spéciale 255.255.255.255.

Exemple : L'adresse de destination 255.255.255.255 indique que le message doit être envoyé à tous les hôtes du réseau dont fait partie l'expéditeur.

Le netid 127.0.0.0 qui aurait du normalement faire partie de la classe A est en fait utilisé pour désigner l'adresse de bouclage (loopback), peu importe le hostid utilisé. Un paquet envoyé à cette adresse ne passe pas par les interfaces réseau mais est déposé directement sur le tampon de réception de la machine elle-même. Cette adresse de bouclage permet de vérifier la configuration de la couche logicielle TCP/IP d'une machine.
Exemple : 127.0.0.1 désigne l'adresse de bouclage sur la machine elle-même.

Au sein de chacune des classes, il existe un sous-espace d'adresses appelées adresses RFC1918. Ces adresses ne sont pas routées sur l'Internet, on les appelle également adresses IP privées.



6.8.4 Exercices

Exercice 1

Convertissez les adresses IP suivantes en binaire :

145.32.59.24
200.42.129.16
14.82.19.54

Trouvez la classe des adresses IP suivantes :

10000000. 00001010. 11011000. 00100111
11101101. 10000011. 00001110. 01011111
01001010. 00011011. 10001111. 00010010
11001001. 11011110. 01000011. 01110101
10000011. 00011101. 00000000. 00000111

Pour chaque adresse, surligner la partie demandée :

PARTIE RESEAU : 1.102.45.177
PARTIE HOTE : 196.22.177.13
PARTIE RESEAU : 133.156.55.102
PARTIE HOTE : 221.252.77.10
PARTIE RESEAU : 123.12.45.77
PARTIE HOTE : 126.252.77.103
PARTIE RESEAU : 13.1.255.102

Exercice 2

Afin de disposer de sous réseaux on utilise le masque de 255.255.240.0 avec une adresse de réseau de classe B
Combien d'hôtes pourra-t-il y avoir par sous réseau ?
Quel est le nombre de sous réseaux disponibles ?

Exercice *3*

Une entreprise veut utiliser l'adresse réseau 192.168.90.0 pour 4 sous réseaux.
Le nombre maximum d'hôtes par sous réseau étant de 25, quel masque de sous réseau utiliseriez-vous pour résoudre ce problème ?

Exercice 4

Quelles sont les adresse IP couvertes par l'adresse CIDR 192.168.10.0/20 ?

Exercice 5

Indiquez en regard de chaque plage d'adresses le réseau en notation standard et CIDR

Plage d'adresses	**Notation CIDR**
Ex : 10.0.0.1. -- 10.255.255.254	10.0.0.0 / 8
172.16.80.1 -- 172.16.87.254	
192.168.15.117 -- 192.168.15.118	
172.16.0.1 -- 172.31.255.254	
10.1.64.1 – 10.1.127.254	
210.44.8.81 -- 210.44.8.94	

Exercice 6

Une machine est configurée avec l'adresse IP 192.168.1.1 et un masque de réseau 255.255.255.0.
Donnez l'adresse du réseau et l'adresse de diffusion sur ce réseau.
Même question avec l'adresse IP 172.26.17.100 et le masque de réseau 255.255.240.0.
Même question avec l'adresse IP 193.48.57.163 et le masque de réseau 255.255.255.224.

Exercice 7

Le réseau 192.168.130.0 utilise le masque de sous réseau 255.255.255.224.
A quels sous réseaux appartiennent les adresses suivantes :
192.168.130.10
192.168.130.93
192.168.130.199
192.168.130.222
192.168.130.250

Exercice 8

Une société possède 73 machines qu'elle souhaite répartir entre 3 sous-réseaux.

- sous-réseaux 1 : 21 machines
- sous-réseaux 2 : 29 machines
- sous-réseaux : 23 machines

Elle souhaite travailler avec des adresses IP privées. On vous demande :

1. De sélectionner la classe des adresses IP
2. De calculer le nombre de bits nécessaires à la configuration des sous-réseaux
3. De calculer le masque de sous-réseau
4. De calculer le nombre de machines configurables dans chaque sous-réseau
5. De calculer les adresses des premières et dernières machines réellement installées dans chaque département.

Exercice 9

1. Pour configurer l'interface d'un hôte qui doit se connecter à un réseau existant, on nous donne l'adresse 172.16.19.40/21.

Quel est le masque réseau de cette adresse ?

Combien de bits ont été réservés pour les sous-réseaux privés ?

Combien de sous-réseaux privés sont disponibles ?
Quelle est l'adresse du sous-réseau de l'exemple ?

Quelle est l'adresse de diffusion du sous-réseau de l'exemple ?

2. Considérons le réseau 40.0.0.0.

Donner le plan d'adressage pour le diviser en 20 sous-réseaux.

3. Considérons le réseau 158.37.0.0.

Donner le plan d'adressage pour avoir 1800 hôtes par sous-réseau.

4. Considérons le sous-réseau 192.168.100.0/24. On souhaite une segmentation par fonctions :
   - Un sous-réseau de 50 hôtes, uniquement pour les secrétaires de l'entreprise.
   - Deux sous-réseaux de 12 hôtes chacun, pour les techniciens et les comptables.
   - Un sous-réseau de 27 hôtes pour les développeurs d'applications.

6.9 Les masques de sous-réseaux

Nous avons déjà vu plusieurs aspects importants des masques qu'il faudra toujours essayer de garder à l'esprit :

- Codés sur 4 octets, soit 32 bits,
- Ils permettent de faire la séparation entre la partie réseau et la partie machine de l'adresse IP,
- La partie réseau est représentée par des bits à 1, et la partie machine par des bits à 0,
- Le masque ne représente rien sans l'adresse IP à laquelle il est associé.

Comment représente-t-on un masque de sous-réseau ?

Comme le masque de sous-réseau est codé sur 32 bits, voici un exemple possible de masque :

```
  _____Réseau_____ _Machine
    |    |    |    |
 11111111.11111111.11111111.00000000
```

Ce qui s'écrit en décimal : 255.255.255.0

Maintenant, plusieurs questions peuvent se poser. Jusqu'ici je comprends, mais comment je peux associer ce masque à une adresse IP, et quel sera le résultat ? Pourquoi les bits à 1 sont séparés de ceux à 0 ?

Comment le masque de sous-réseau et l'adresse IP sont-ils associés ?

Prenons par exemple une machine qui a pour adresse IP 192.168.25.147. Il nous faut lui associer un masque de sous-réseau pour savoir quelle partie de cette adresse représente le réseau. Associons-lui le masque précédent 255.255.255.0. On remarque que les bits des trois premiers octets sont à 1, ils représentent donc la partie réseau de l'adresse, soit 192.168.25, le 147 permettant d'identifier la machine au sein de ce réseau. Dans cet exemple, on remarque qu'un octet a été réservé pour l'adresse machine, ce qui nous donne 2^8 = 256 adresses disponibles pour les machines sur le réseau 192.168.25. Les adresses disponibles pour les machines seront donc :

192.168.25.0 (réservée pour le réseau, voir 5.4)
192.168.25.1
...
192.168.25.254
192.168.25.255 (réservée pour le broadcast, voir 5.4)

On observe donc que c'est le masque qui détermine le nombre de machines d'un réseau. Ainsi, on verra par la suite qu'on choisira le masque en fonction du nombre de machines que l'on veut installer.

6.9.1 Adresses spécifiques (réseau, broadcast)

Il existe des adresses spécifiques au sein d'un réseau. La première adresse d'une plage ainsi que la dernière ont un rôle particulier. La première adresse d'une plage représente l'adresse du réseau. Celle-ci est très importante car c'est grâce à elle qu'on peut identifier les réseaux et router les informations d'un réseau à un autre. La dernière adresse d'une plage représente ce que l'on appelle l'adresse de broadcast.

Cette adresse est celle qui permet de faire de la diffusion à toutes les machines du réseau. Ainsi, quand on veut envoyer une information à toutes les machines, on utilise cette adresse.

Dans notre exemple, l'adresse de réseau sera donc 192.168.25.0, et l'adresse de broadcast 192.168.25.255. On remarque donc qu'il ne nous reste plus que 254 adresses pour identifier nos machines. Ainsi, à chaque fois que l'on choisira un masque en fonction du nombre de machines que l'on veut adresser, il faudra tenir compte de ces deux adresses…

Les bits à 1 et à 0 doivent-ils être contigus ?

Dans l'exemple de masque que nous avons choisi, nous avons vu que les bits à 0 et à 1 étaient regroupés. Cela n'est pas une obligation, mais cela facilite énormément l'exploitation du réseau. En conservant la contiguïté des bits, les adresses de nos machines au sein du réseau se suivent. Ce ne serait pas le cas si l'on avait choisi un masque avec des bits non contigus.

Exemple, si on choisit le masque suivant :
11111111.11111111.11111110.00000001

Ici, on a comme précédemment 8 bits qui représentent la partie machine, cependant, ils ne sont plus à la même place. Cela se traduit en décimal par le masque suivant 255.255.254.1. On voit donc que les adresses dont le dernier bit est a 1 ne seront pas dans le même réseau que celles dont le dernier bit est a 0. Ce qui veut dire que les adresses dont le dernier octet est impair ne seront pas dans le même réseau que les adresses paires.
Dans cet exemple, cela reste encore facile de différencier les adresses paires et impaires, mais lorsque l'on fait des mélanges plus compliqués entre les bits significatifs, cela devient très vite inextricable. On conservera donc toujours la contiguïté des bits significatifs !!!

6.9.2 Quelles adresses pour les masques de sous-réseau ?

Etant donné que l'on conserve la contiguïté des bits, on va toujours rencontrer les mêmes nombres pour les octets du masque. Ce sont les suivants :

11111111
11111110
11111100
...
10000000
00000000

Soit en décimal :
255, 254, 252, 248, 240, 224, 192, 128, et 0.

Ainsi, on peut tout de suite dire si un masque semble valide au premier coup d'œil. Un masque en 255.255.224.0 sera correct alors qu'un masque en 255.255.232.0 ne le sera pas (à moins de ne pas vouloir respecter la contiguïté des bits)
Vous pouvez aller voir tous les masques possibles dans la RFC suivante : RFC 1878

Faire fi de l'écriture par octets

L'écriture de l'adresse IP selon 4 octets séparés par un point est facile à utiliser. Mais quand on se penche sur le problème d'un peu plus près, on se rend compte qu'elle n'est pas très adaptée… Elle a deux défauts principaux :

- Ecriture en décimal alors que l'on raisonne en binaire
- Séparation des octets par des points

Ainsi, lorsqu'on utilise des masques où la séparation réseau/machine se fait sur un octet (tous les bits des octets sont soit à 1, soit à 0) cela est simple. Prenons par exemple le réseau 192.168.25.0/255.255.255.0. Toutes les machines commençant par 192.168.25 appartiendront à ce réseau. Si l'on prend le réseau 192.168.25.32/255.255.255.248 et je vous demande si la machine 192.168.25.47 appartient à ce réseau ? ça devient plus compliqué…

Pour bien comprendre, il faut alors revenir en binaire. Etant donné que les trois premiers octets du masque ont tous leurs bits à 1, c'est sur le quatrième que va se faire la différentiation. Il s'écrit 248, soit 11111000 en binaire. Donc les 5 premiers bits de cet octet représenteront la partie réseau.

Pour notre réseau, le dernier octet vaut 32, soit 00100000, pour notre machine, il vaut 47, soit 00101111. On voit que les 5 premiers bits de ces deux octets ne sont pas identiques (00100 != 00101) et donc que ces deux adresses n'appartiennent pas au même réseau. Cela peut sembler très compliqué, mais on verra par la suite des méthodes simples pour déterminer rapidement l'appartenance à un réseau.

Quelle est cette notation avec un /, comme /24 ?

Une autre notation est souvent utilisée pour représenter les masques. On la rencontre souvent car elle est plus rapide à écrire. Dans celle-ci, on note directement le nombre de bits significatifs en décimal, en considérant que la contiguïté est respectée. Ainsi, pour notre exemple 192.168.25.0/255.255.255.0, on peut aussi écrire 192.168.25.0/24, car 24 bits sont significatifs de la partie réseau de l'adresse.

De même, les écritures suivantes sont équivalentes :

10.0.0.0/255.0.0.0 = 10.0.0.0/8
192.168.25.32/255.255.255.248 = 192.168.25.32/29

Partir de l'existant

La plupart du temps, le choix de l'adressage se fait en fonction des besoins exprimés, et des limites de ce que l'on a le droit de faire. Une certaine plage vous est allouée par votre fournisseur d'accès. Vous pourrez alors découper cette plage en différents réseaux, mais ne surtout pas dépasser celle-ci. Ainsi, si vous possédez une plage de 128 adresses et que vous voulez adresser 500 machines, vous aurez quelques petits problèmes…

En fonction du nombre de machines

Etant donné que le masque détermine le nombre de machines qu'il pourra y avoir sur un réseau, c'est souvent de cette information que l'on part pour choisir le masque. Etant donné que l'on travaille en binaire, le nombre de machines possible au sein d'un réseau sera une puissance de 2.

Pour un nombre donné de machines, il faudra donc choisir la puissance de 2 immédiatement supérieure pour pouvoir adresser les machines. De plus, il faudra prévoir un certain nombre d'adresses supplémentaires pour accueillir de nouvelles machines.

Ainsi, disons que l'on possède le réseau 193.225.34.0/255.255.255.0 et que l'on veut faire un réseau de 60 machines au sein de celui-ci. On veut 60 machines, il faut ajouter deux adresses pour le réseau et le broadcast, ce qui fait 62 adresses au total. La puissance de 2 supérieure à 62 est 64, mais cela ne nous laisserait que 2 adresses pour évoluer, ce qui est un peu juste. On préfèrera donc un réseau de 128 adresses. Pour identifier 128 adresses, il nous faut 7 bits (128 = 2^7) Donc dans notre masque, 7 bits seront à 0 pour identifier la partie machine, et les 25 bits restants seront à 1. Ce qui donne :

11111111.11111111.11111111.10000000

et en décimal :

255.255.255.128

6.9.3 Comment déterminer la plage d'adresses à partir du masque et d'une adresse ?

Nous avons vu précédemment que le masque devait être associé à une adresse IP pour avoir une valeur. Le choix de la plage d'adresses sur laquelle il s'applique est donc tout aussi important !! Nous avons choisi un masque qui nous permettra d'identifier 128 machines. Mais nous possédons une plage de 256 adresses. Où faut-il placer nos 128 adresses dans cette plage ? Peut-on les placer n'importe où ?

La réponse est bien sûr non. Nous n'avons que deux possibilités pour choisir notre plage, les adresses de 0 à 127, et les adresses de 128 à 255. Choisir une plage de 32 à 160 serait une erreur, et le réseau ne fonctionnerait pas.

Voici l'explication :

La différenciation du réseau va se faire sur le premier bit du dernier octet (vu que nos trois premiers octets sont fixés à 193.225.34) Si ce bit est à 0, cela correspond aux adresses de 0 à 127. S'il est à 1, cela correspond aux adresses de 128 à 255. Ainsi, si l'on choisit une plage d'adresses de 32 à 160, les adresses de 32 à 127 auront le premier bit de leur dernier octet à 0, alors que les adresses de 128 à 160 auront ce même bit à 1, elles seront alors considérées comme étant dans deux réseaux différents !!!

Ainsi, quel que soit le nombre de machines à placer dans une plage, on ne peut pas choisir l'adressage n'importe comment.

Une méthode simple pour trouver les adresses de réseau possible

Il n'est pas toujours évident de savoir si une adresse correspond bien à celle d'un réseau selon le masque que l'on a choisi. Avec la méthode suivante, vous devriez pouvoir vous en sortir. Il faut avant tout que vous ayez déterminé le masque de sous-réseau selon le nombre de machines dont vous avez besoin.

Ensuite, selon l'octet significatif (qui n'est pas à 0 ou 255) faites 256-cet_octet=X. L'adresse de réseau devra alors être un multiple de X. Un petit exemple pour être un peu plus clair. On veut par exemple 50 machines, on choisit donc un masque en 255.255.255.192. C'est le dernier octet qui est significatif, on fait donc 256-192=64. Il faut donc que le dernier octet de l'adresse de réseau soit un multiple de 64. Si on prend la plage 10.0.0.0/255.255.255.0, on pourra choisir les adresses de réseau suivantes :

10.0.0.0,
10.0.0.64,
10.0.0.128,
10.0.0.192.

6.9.4 Comment découper une plage réseau quelconque comme somme de plusieurs plages ?

Nous avons vu qu'une plage réseau ne pouvait pas être choisie n'importe comment. Etant donné que les masques et les adresses IP se basent sur un codage binaire, les chiffres utilisés dans les adresses résultantes ne pourront être que des multiples de puissances de 2 en accord avec le masque. Ainsi, une plage 70.0.0.0 ne pourra pas avoir un masque qui définisse plus de deux chiffres sur le premier octet, car 70 est un multiple de 2, mais pas de 4.

Ce n'est pas clair ? un exemple devrait vous aider à mieux comprendre. Disons que l'on veut décrire la plage d'adresses allant de 69.0.0.0 à 79.255.255.255.

La question est de savoir quel masque de sous-réseau associé à quelle adresse de réseau nous permettra de définir cette plage.

Le premier octet varie de 69 à 79. Il prend donc 11 valeurs. 11 n'étant pas une puissance de 2, on sait d'ores et déjà que l'on ne pourra pas définir cette plage avec un seul réseau, mais qu'il va falloir la découper en une somme de plusieurs réseaux. Le but est cependant d'optimiser cette somme de réseaux pour en avoir le moins possible. On pourrait simplement utiliser 11 réseaux avec des masques 255.0.0.0, mais on doit sûrement pouvoir faire plus propre et regrouper plusieurs de ces réseaux en un seul.

La première puissance de 2 inférieure à 11 est 8. Il faut maintenant savoir si l'on peut placer un réseau, dont le premier octet décrira 8 valeurs, dans cette plage. Le seul multiple de 8 de cette plage est 72. On décrirait alors un réseau dont le premier octet varierait de 72 à 79, ce qui est bien compris dans notre plage d'origine. Le réseau 72.0.0.0/248.0.0.0 est donc bien adapté pour décrire notre plage, mais il reste encore à décrire les adresses de 69.0.0.0 à 71.255.255.255.

On effectue le même raisonnement. (Ici le premier octet prend 3 valeurs, la puissance de 2 inférieure à 3 est 2, et le multiple de 2 de cette plage est 70)

On trouve donc le réseau 70.0.0.0/254.0.0.0

Il ne nous reste plus qu'à décrire la plage 69.0.0.0 à 69.255.255.255 qui peut être définie par 69.0.0.0/255.0.0.0.

Et voilà !! Nous avons découpé notre plage d'origine qui allait de 69.0.0.0 à 79.255.255.255 en trois sous-réseaux :

69.0.0.0/255.0.0.0
70.0.0.0/254.0.0.0
72.0.0.0/248.0.0.0

6.9.5 Comment déterminer qu'une machine appartient à mon réseau ?

C'est très simple. Pour cela, il va falloir déterminer si l'adresse de la machine appartient à la plage d'adresses définie par mon adresse et mon masque. Pour cela, je fais un ET logique entre mon adresse et mon masque réseau, j'en déduis donc l'adresse de mon réseau.

Je fais pareil avec l'adresse de l'autre machine et MON masque réseau, et j'obtiens une adresse de réseau. Si les deux adresses de réseau sont les mêmes, ça veut dire que la machine appartient bien au même réseau.

Disons par exemple que ma machine ait pour adresse 192.168.0.140/255.255.255.128 et je veux savoir si les machines A et B ayant pour adresses 192.168.0.20(A) et 192.168.0.185(B) sont sur le même réseau ? Je fais

```
      192.168.0.140
ET  255.255.255.128
  -------------------
=     192.168.0.128
```

de même avec les deux autres adresses Pour A

```
       192.168.0.20
ET   255.255.255.128
   -------------------
=      192.168.0.0
```

et pour B

```
       192.168.0.185
ET   255.255.255.128
   -------------------
=      192.168.0.128
```

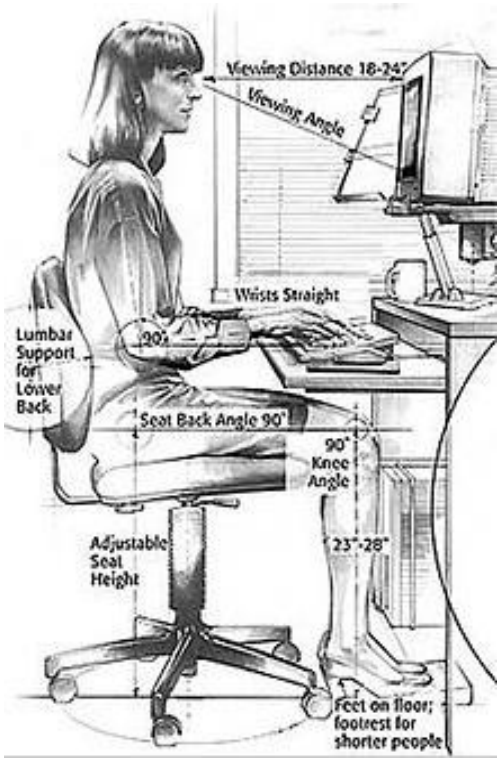
On voit ainsi que les nombres obtenus sont les mêmes pour ma machine et B. On en déduit donc que B est sur le même réseau, et que A est sur un réseau différent.

Module 7 : Gestion responsable de l'outil informatique

7.1 Organisation de son poste de travail

Notion d'ergonomie

En organisation du travail, un poste de travail est le lieu dans lequel un employé dispose des ressources matérielles lui permettant d'effectuer son travail. Dans un contexte informatique, ce terme correspond à l'ensemble des moyens techniques mis à la disposition d'un utilisateur (écran, clavier, imprimante), et par extension l'interface du système d'exploitation



Dans un contexte informatique « Poste de travail » représente principalement le point d'accès à toutes les fonctionnalités d'une application informatique ainsi système d'exploitation, en particulier aux ressources informatiques (messagerie, bureautique, mais aussi imprimante, numériseur de document, ...).

L'origine de ce terme n'est pas forcément connue, mais elle remonte sûrement aux débuts de l'informatique répartie (UNIX avec les terminaux X11, puis Windows avec le client-serveur). Ce terme est utilisé dans Windows pour désigner le micro-ordinateur en lui-même, et c'est de là que l'on accède à tous les disques. Il est maintenant utilisé pour divers systèmes d'exploitation.

Windows étant le système d'exploitation le plus utilisé, le terme « Poste de travail » désigne maintenant dans la plupart des esprits utilisateurs, l'ordinateur et aussi l'environnement (physique) dans lequel il se trouve.
Il est parfois confondu avec le « Bureau », qui dans Windows désigne l'espace de travail des fenêtres, mais aussi la base de toute l'arborescence.
À partir d'un « Poste de travail » on peut ainsi accéder aux disques durs, aux supports amovibles (lecteur de disquette, lecteur de CD-ROM, ...), à divers périphériques, comme les imprimantes, les périphériques d'acquisition d'images (appareil photo numérique, numériseur de document), et utiliser des applications informatiques à distance.
A partir du système Windows Vista le terme « Poste de travail » change en « Ordinateur », mais le reste de l'arborescence reste identique.

Le poste de travail peut aussi être une station de travail Unix ou de plus en plus souvent PC ou portable
Linux.
L'utilisation des écrans a suscité des controverses dans le domaine de la santé. Les problématiques considérées sont essentiellement la posture face à l'écran, et ses effets sur le dos, ainsi que les effets éventuels des rayonnements sur la vue. En conséquence, certains organismes de santé au travail recommandent d'installer un écran suffisamment haut pour qu'il soit en face des yeux lorsque l'on est assis en position droite.

7.2 Notions de sécurité concernant les risques électriques

L'électricité est une énergie dangereuse : invisible, inodore et d'une puissance redoutable. Elle l'est d'autant plus lorsqu'elle est incontrôlée.

Un principe de base et de couper le courant lors de toute manipulation du matériel informatique (en anglais: hardware). Se protéger aussi contre tout courant résiduel sur les composants. Lors du raccordement électrique une prise de terre est indispensable. Utiliser des prises sécurisées raccordées à un battery backup (dispositif de batterie qui maintient le courant pendant un certain temps) pour se prémunir des coupures du réseau électrique. Au niveau sécurité des personnes, employer les câbles avec bon isolant, bonnes fiches, des prises murales placées par des professionnels. Ne pas enlever les fiches des prises en tirant sur le câble, car à la longue il risque de se dénuder. Ne pas brancher trop d'appareils sur la même prise.

Les risques physiques peuvent être : brulures, asphyxie (par blocage de la respiration par paralysie du diaphragme et des muscles respiratoires), syncope (le cœur reçoit une décharge électrique)

7.3 Sécurité élémentaire de manipulation des objets informatiques

Manipulation des objets informatiques en vue d'éviter leur détérioration

Comme dit, avant toute manipulation il est important de couper le courant électrique. Les prises, les fiches doivent être sécurisées pour les risques de foudre et de surtension

Manipulation judicieuse des consommables d'imprimantes

Les cartouches d'encre usagées font partie des DIS (Déchets Industriels Spéciaux) car elles contiennent des matériaux toxiques pour l'environnement (plastiques non biodégradables, métaux lourds, résidus d'encre…) A ce titre elles doivent être traitées par un organisme agréé.

Jet d'encre

Ce procédé se divise en 2 catégories : le jet continu et la goutte à la demande. L'encre est un mélange composé de solvant, de matière colorante, d'un liant et d'additifs :

Le rôle du solvant est de transporter l'encre entre la cartouche et le papier et contribue au séchage ; on utilise généralement de la méthyléthylcétone, des acétates, de l'éther glycol et des alcools ; certains solvants sont toxiques quand ils sont respirés, avalés ou par contact avec la peau et ont un impact sur l'environnement

Le colorant a d'abord été constitué de composants très solubles ayant une bonne tenue à la lumière et sans métaux lourds ; aujourd'hui, on s'oriente vers l'utilisation de pigments très fins (<1 µm) qui sont très toxiques (métaux lourds en général)
Le liant assure la cohésion de l'encre et contrôle sa viscosité ; il permet l'adhésion de la matière colorante au support ; les résines phénoliques ont aujourd'hui tendance à être remplacées par des copolymères

Les additifs sont présents dans l'encre dans une proportion inférieure à 1% ; ils ont pour objet d'améliorer la fluidité, l'adhésion, la rhéologie du liant ou la conductivité de l'encre ; ils peuvent être de puissants allergènes

Laser

Chaque année, 50 millions de cartouches laser sont achetées en Europe. Ces cartouches contiennent 25 000 tonnes de toner et représentent 50 000 tonnes de plastiques. Chaque cartouche en fin d'utilisation contient encore 15% du toner initial. (Source Innotec)

L'impression laser est un procédé d'impression photo-électrique qui utilise de l'encre en poudre (toner). Cette poudre est constituée en majeure partie de fines particules de matière plastique, de résine et de pigments magnétiques. Les polymères (copolymères, résines polyesters) utilisés varient selon le fabricant. Initialement, la taille moyenne des particules se situaient entre 14 et 16 µm ou plus. Quand on passe à une résolution de 600 ppp, la taille des particules peut passer de 8 à 10 µm.

Comme dans le cas de l'impression jet d'encre, le toner n'est pas sans conséquences sur l'environnement et la santé. Une enquête menée sur une soixantaine d'imprimantes laser a en effet révélé que presque un tiers des modèles testés rejette dans l'air des résidus d'encre potentiellement dangereux.

7.4 Adopter un comportement propre à économiser les ressources énergétiques et à réduire au maximum les nuisances sur l'environnement

7.4.1 Implications environnementales de l'utilisation de l'informatique

Comme décrit au point précédent, les intacts des encres d'imprimantes sur la santé et l'environnement sont loin d'être négligeables.

Mise en veille du système

Un dispositif électronique ou informatique est en veille si la plupart de ses fonctions ont été arrêtées temporairement dans le but de diminuer sa consommation électrique.
Au niveau d'un ordinateur, on distingue plusieurs niveaux de veille :
L'état du système d'exploitation dit inactif, pour lequel certains logiciels comme l'économiseur d'écran se mettent en œuvre.
La veille simple où certains composants matériels (écran, disque dur, etc.) sont arrêtés par des signaux spécifiques et paramétrables à partir du panneau de configuration.
La veille prolongée, aussi appelée hibernation (en référence au phénomène animal) où le matériel est éteint tandis que toute la mémoire vive est copiée sur le disque dur, ce qui conserve tout l'état logiciel courant du système d'exploitation et des logiciels éventuellement en cours d'exécution, etc. Cela permet de ne plus consommer d'énergie.

Tri des déchets

Habituellement le principe d'un bon recyclage consiste à créer de nouvelles machines à partir de plusieurs machines récupérées. Avec de la chance, ou peut atteindre un rendement de 80%, soit 8 machines créées à partir des pièces de 10 machines récupérées en panne.

Le tri des déchets et la collecte sélective sont des actions consistant à séparer et récupérer les déchets selon leur nature, à la source, pour éviter les contacts et les souillures. Ceci permet de leur donner une « seconde vie », le plus souvent par le réemploi et le recyclage évitant ainsi leur simple destruction par incinération ou abandon en décharge et, par conséquent, de réduire l'empreinte écologique.

Encres

Les cartouches d'encre usagées font partie des DIS (Déchets Industriels Spéciaux) car elles contiennent des matériaux toxiques pour l'environnement (plastiques non biodégradables, métaux lourds, résidus d'encre…)
A ce titre elles doivent être traitées par un organisme agréé.

Ecrans plasma :

Ils utilisent de microscopiques cellules de plasma gazeux qui sont activées par un courant électrique afin de générer une image. Leur recyclage est coûteux et complexe car ils contiennent des poudres électroluminescentes pouvant être composées de cadmium qui doit être retiré. Opération difficile car les poudres sont encapsulées dans le verre

Ecrans LCD

Les écrans LCD (Liquid Cristal Display – Affichage à cristaux liquides) sont constitués de cristaux liquides entre deux dalles de verre. Ces cristaux constituant un filtre destiné à ne laisser passer que les couleurs formant l'image. Ces écrans sont très plats.

Les composants sont très toxiques car ils contiennent du mercure. Des recherches travaillent sur le recycles de l'oxyde d'indium – sans quoi, ressource qui pourrait être épuisée dans une dizaine d'années

Composants électroniques :

- **Le boîtier**

Aucun problème pour récupérer un ancien boîtier AT ou ATX.

- **La carte mère**

Si elle est encore AT, il faudra le boîtier qui va avec. Toute carte mère est fonctionnelle tant qu'un processeur d'origine y est fixé. Si le processeur a disparu, il va être compliqué d'en trouver un qui soit compatible.

- **La RAM**

On trouvera souvent des cartes mères avec de la RAM. Ajouter de la RAM est généralement facile, la principale difficulté étant de trouver la bonne interface : souvent la SD-RAM, il est rare de trouver de la EDO.

Toute RAM est bonne à prendre (il existe des systèmes de test pour vérifier leur bon état).

- **Processeur**

Sans une carte mère prévue pour son fonctionnement, il n'est pas certain qu'on puisse le réutiliser.

- **L'alimentation**

Beaucoup de PC sont jetés parce qu'on les pense grillés, alors que seul le bloc d'alimentation l'était. Toute alimentation en état de fonctionner est susceptible de pouvoir resservir.

- **Disque dur**

Toujours récupérable tant qu'il fonctionne. Certains disques durs anciens sont toutefois de capacité vraiment faible (quelques mégas) ou prêts à rendre l'âme.

- **Lecteur/Graveur CD**

Toujours récupérable tant qu'il fonctionne. Attention toutefois aux vitesses de lecture ou d'écriture : la gravure en 2x est vraiment lente.

- **Lecteur disquette**

Récupérable tant qu'il est en état de fonctionner

- **Clavier et souris**

Ils risquent d'être en PS/1, si c'est le cas : il faut une carte mère PS/1, ce qui se fait très vieux. Si c'est du PS/2, alors ils sont parfaitement réutilisables. Toutefois, si le clavier est aussi fonctionnel qu'un clavier récent, une souris ancienne peut être fonctionnelle mais peu pratique (souris à boule encrassée et difficilement nettoyable par exemple).
Carte son

- **Carte réseau**

Récupérable ; l'Ethernet est toujours le standard en vigueur.

- **Carte graphique**

Récupérable. Utile pour faire fonctionner les jeux vidéo de l'époque où elle était utilisée...

Module 8 : Gestion des bases de données

Mettez en place une base de données avec phpMyAdmin

Pour communiquer avec MySQL, nous pouvons utiliser différents logiciels : de l'invite de commandes jusqu'à des logiciels accessibles par le navigateur. Ici, vous allez utiliser **phpMyAdmin**, l'un des outils les plus connus permettant de manipuler une base de données MySQL.

phpMyAdmin est livré avec MAMP et XAMPP ; vous allez donc pouvoir vous en servir tout de suite !

Les types de champs MySQL

Alors que PHP ne propose que quelques types de données que l'on connaît bien maintenant (`int` , `string` , `bool` ...), MySQL propose une quantité très importante de types de données.

Mais dans la pratique, vous n'aurez besoin de jongler qu'entre les quatre types de données suivants :

1. `INT` : nombre entier ;
2. `VARCHAR` : texte court (entre 1 et 255 caractères) ;
3. `TEXT` : long texte (on peut y stocker un roman sans problème) ;
4. `DATE` : date (jour, mois, année).

Cela couvrira 99 % de vos besoins, et avec l'expérience vous apprendrez à optimiser vos bases de données, et l'**intérêt des autres types de données** de MySQL.

Les clés primaires

Toute table doit posséder un champ qui joue le rôle de *clé primaire*. La clé primaire permet d'identifier de manière unique une entrée dans la table. En général, on utilise le champ `id` comme clé primaire, comme on vient de le faire.

Prenez le réflexe de créer à chaque fois ce champ « id » en lui donnant l'index `PRIMARY` , ce qui aura pour effet d'en faire une clé primaire.

Vous en profiterez en général pour cocher la case `AUTO_INCREMENT` pour que ce champ gère lui-même les nouvelles valeurs automatiquement (1, 2, 3, 4...).

Connectez-vous à la base de données en PHP

Pour pouvoir travailler avec la base de données en PHP, il faut d'abord s'y connecter.

Il va donc falloir que PHP s'authentifie : on dit qu'il établit une connexion avec MySQL.

Une fois que la connexion sera établie, vous pourrez faire toutes les opérations que vous voudrez sur votre base de données !

Pour se connecter à une base de données MySQL, vous allez devoir utiliser une extension PHP appelée **PDO** ("PHP Data Objects"). Cette extension est fournie avec PHP (en français, "les fonctions PDO sont à votre disposition"), mais parfois il vous faudra activer l'extension.

Vérifiez que PDO est bien activé

Je vous recommande de passer la ligne `display_errors` sur "On" pour que les erreurs s'affichent ; ça va grandement nous aider :

```
display_errors = On
```

Maintenant que nous sommes certains que PDO est activé, nous pouvons nous connecter à MySQL. Nous allons avoir besoin de quatre renseignements :

- **Le nom de l'hôte** : c'est l'adresse IP de l'ordinateur où MySQL est installé. Le plus souvent, MySQL est installé sur le même ordinateur que PHP : dans ce cas, mettez la valeur `localhost` .
- **La base** : c'est le nom de la base de données à laquelle vous voulez vous connecter. Dans notre cas, la base s'appelle `my_recipes` . Vous l'avez créée avec phpMyAdmin dans le chapitre précédent.
- **L'identifiant et le mot de passe** : ils permettent de vous identifier. Renseignez-vous auprès de votre hébergeur pour les connaître.

Voici donc l'instruction PDO pour vous connecter à votre base `my_recipes` :

```php
<?php
// Souvent on identifie cet objet par la variable $conn ou $db
$mysqlConnection = new PDO(
    'mysql:host=localhost;dbname=my_recipes;charset=utf8',
    'root',
    'root'
);
?>
```

Je ne comprends rien à ce code, c'est normal ?

Oui, il faut reconnaître qu'il contient quelques nouveautés.

En effet, PDO est ce qu'on appelle une extension orientée objet. C'est une façon de programmer un peu différente des fonctions classiques que l'on a appris à utiliser jusqu'ici.

Pour l'instant, je vous invite à réutiliser les codes que je vous propose en suivant mes exemples. Vous comprendrez les détails de leur mode de fonctionnement un peu plus tard.

La ligne de code qu'on vient de voir crée une connexion à la base de données.

En fait, on crée la connexion en indiquant dans l'ordre dans les paramètres :

- le nom d'hôte : `localhost` ;
- la base de données : `my_recipes` ;
- l'identifiant (login) : `root` ;
- le mot de passe (rappel : sous MAMP, le mot de passe est `root`).

Lorsque votre site sera en ligne, vous aurez sûrement un nom d'hôte différent, ainsi qu'un identifiant et un mot de passe, comme ceci :

```php
<?php
$db = new PDO('mysql:host=sql.hebergeur.com;dbname=mabase;charset=utf8', 'pierre.durand', 's3cr3t');
?>
```

Il faudra donc penser à changer cette ligne pour l'adapter à votre hébergeur en modifiant les informations en conséquence, lorsque vous enverrez votre site sur le Web.

Effectuez une première requête SQL

Voici la première requête SQL que nous allons utiliser :

```mysql
SELECT * FROM recipes
```

Cela peut se traduire par :

> « Prendre tout ce qu'il y a dans la table `recipes` ».

Analysons chaque terme de cette requête.

- **SELECT** : en langage SQL, le premier mot indique quel type d'opération doit effectuer MySQL. Dans ce chapitre, nous ne verrons que `SELECT` . Ce mot-clé demande à MySQL d'afficher ce que contient une table.
- ***** : après le `SELECT` , on doit indiquer quels champs MySQL doit récupérer dans la table. Si on n'est intéressé que par les champs « nom » et « possesseur », il faudra taper : `SELECT nom, possesseur FROM recipes`
  Si vous voulez prendre tous les champs, tapez `*` . Cette petite étoile peut se traduire par « tout » : « Prendre tout ce qu'il y a... ».
- **FROM** : c'est un mot de liaison qui se traduit par « dans ». `FROM` fait la liaison entre le nom des champs et le nom de la table.
- **recipes** : c'est le nom de la table dans laquelle il faut aller piocher.

Effectuons la requête à l'aide de l'objet PDO :

```php
<?php
$recipesStatement = $db->prepare('SELECT * FROM recipes');
?>
```

Affichez le résultat d'une requête SQL

Le problème, c'est que `$recipesStatement` contient quelque chose d'inexploitable directement : un objet PDOStatement. Cet objet va contenir la requête SQL que nous devons exécuter, et par la suite, les informations récupérées en base de données.

Pour récupérer les données, demandez à cet objet d'exécuter la requête SQL et de récupérer toutes les données dans un format "exploitable" pour vous, c'est-à-dire sous forme d'un tableau PHP.

```php
<?php
$recipesStatement->execute();
$recipes = $recipesStatement->fetchAll();
?>
```

"Fetch" en anglais signifie « va chercher ».

Une fois qu'on a récupéré les données sous forme d'un tableau PHP, on en revient à ce que vous connaissez déjà ! Je vous propose de résumer tout ce que vous venez d'apprendre, de la connexion via PDO à l'affichage du résultat de la requête :

php

```php
<?php
try
{
    // On se connecte à MySQL
    $mysqlClient = new PDO('mysql:host=localhost;dbname=my_recipes;charset=utf8', 'root', 'root');
}
catch(Exception $e)
{
    // En cas d'erreur, on affiche un message et on arrête tout
        die('Erreur : '.$e->getMessage());
}

// Si tout va bien, on peut continuer

// On récupère tout le contenu de la table recipes
$sqlQuery = 'SELECT * FROM recipes';
$recipesStatement = $mysqlClient->prepare($sqlQuery);
$recipesStatement->execute();
$recipes = $recipesStatement->fetchAll();

// On affiche chaque recette une à une
foreach ($recipes as $recipe) {
?>
    <p><?php echo $recipe['author']; ?></p>
<?php
}
?>
```

Affichez seulement le contenu de quelques champs

Avec ce que je vous ai appris, vous devriez être capable d'afficher ce que vous voulez. Personne ne vous oblige à afficher tous les champs !

Par exemple, si j'avais juste voulu lister les noms des recettes et le nom de l'auteur, j'aurais utilisé la requête SQL suivante :

```php
<?php

$sqlQuery = 'SELECT title, author FROM recipes';
```

Filtrez vos données

Rappelez-vous votre objectif : on souhaite lister les recettes qui sont activées, c'est-à-dire celles dont le champ **is_enabled** vaut `TRUE` .

Ça paraît compliqué à faire, non ?

Pas en SQL !

Vous allez voir qu'en modifiant vos requêtes SQL, il est possible de filtrer et trier très facilement vos données. Vous allez ici découvrir les mots-clés suivants du langage SQL :

- `WHERE` ;
- `ORDER BY` ;
- `LIMIT` .

Grâce au mot-clé `WHERE` , vous allez pouvoir trier vos données.

Puisque l'on souhaite récupérer uniquement les recettes avec le champ **is_enabled** à `TRUE` , alors la requête au début sera la même qu'avant, mais vous rajouterez à la fin `WHERE is_enabled = TRUE` .

Cela vous donne la requête :

php

```php
<?php
$sqlQuery = 'SELECT * FROM recipes WHERE is_enabled = TRUE';
```

Plus vous aurez de conditions et plus la requête devient complexe. Avant d'écrire le code PHP, vous pouvez déjà vérifier que la requête SQL est correcte en la testant dans l'onglet "SQL" de phpMyAdmin, comme je viens de vous le montrer.

Il faut utiliser les mots-clés dans l'ordre que j'ai donné : `WHERE` puis `ORDER BY` puis `LIMIT` , sinon MySQL ne comprendra pas votre requête.

Construisez des requêtes en fonction de variables

En guise d'exemple complet, voici la requête pour retrouver les recettes valides de Mathieu, écrite dans le respect des meilleures pratiques :

```php
<?php
$sqlQuery = 'SELECT * FROM recipes WHERE author = :author AND is_enabled = :is_enabled';

$recipesStatement = $db->prepare($sqlQuery);
$recipesStatement->execute([
'author' => 'mathieu.nebra@exemple.com',
'is_enabled' => true,
]);
$recipes = $recipesStatement->fetchAll();
]);
```

> ✗ On ne concatène **JAMAIS** une requête SQL pour passer des variables, au risque de créer des injections SQL ! Le sujet des injections SQL est un peu trop complexe pour être détaillé ici. Si vous souhaitez en apprendre plus à ce sujet, je vous invite à consulter **Wikipedia**.

Ajoutez, modifiez et supprimez !

Vous avez appris dans le chapitre comment ajouter de nouvelles entrées en base sur phpMyAdmin, mais si vous voulez que des utilisateurs contribuent à votre site, il faudra leur proposer un formulaire de création et de modification !

Pour cela, vous allez aborder de nouvelles requêtes SQL fondamentales et les appliquer avec PHP : `INSERT` , `UPDATE` et `DELETE` .

Implémentez l'ajout de recettes

Pour cela, vous aurez besoin de trois choses :

1. Ajouter un formulaire PHP de création de recettes.
2. Vérifier les données soumises en PHP.
3. À l'aide de PDO, exécuter l'insertion de la nouvelle recette en base de données.

Ajoutez avec l'instruction `INSERT INTO`

Pour ajouter une entrée, vous aurez besoin de connaître la requête SQL. En voici une par exemple qui ajoute une recette :

```php
<?php
$sqlQuery = 'INSERT INTO recipes(title, recipe, author, is_enabled) VALUES (:title, :recipe, :author, :is_enabled)';

```

Étudions un peu cette requête.

- D'abord, vous devez commencer par les mots-clés `INSERT INTO` qui indiquent que vous voulez insérer une entrée.
- Vous précisez ensuite le nom de la table (ici `recipes`), puis listez entre parenthèses les noms des champs dans lesquels vous souhaitez placer des informations.
- Enfin – et c'est là qu'il ne faut pas se tromper – vous inscrivez `VALUES` suivi des valeurs à insérer **dans le même ordre que les champs que vous avez indiqués**.

Utilisez cette requête SQL au sein d'un script PHP :

```php
<?php
try
{
    $db = new PDO('mysql:host=localhost;dbname=my_recipes;charset=utf8', 'root', 'root');
}
catch (Exception $e)
{
        die('Erreur : ' . $e->getMessage());
}

// Ecriture de la requête
$sqlQuery = 'INSERT INTO recipes(title, recipe, author, is_enabled) VALUES (:title, :recipe, :author, :is_enabled)';

// Préparation
$insertRecipe = $db->prepare($sqlQuery);

// Exécution ! La recette est maintenant en base de données
$insertRecipe->execute([
    'title' => 'Cassoulet',
    'recipe' => 'Etape 1 : Des flageolets ! Etape 2 : Euh ...',
    'author' => 'contributeur@exemple.com',
    'is_enabled' => 1, // 1 = true, 0 = false
]);
```

Vous remarquerez que je n'ai pas passé de valeur pour le champ `recipe_id` .

C'est normal ! Le champ a la propriété `AUTO_INCREMENT` , MySQL lui attribuera une valeur automatiquement.

Et si j'ai passé une valeur booléenne pour le champ `is_enabled` , c'est parce que j'ai fait le choix de stocker cette information sous forme d'entier :

- 0 pour FAUX ;
- et 1 pour VRAI.

Si vous voulez passer les valeurs PHP `true` ou `false` , alors il faudra déclarer le champ `is_enabled` avec le type MySQL `BOOL` .

Éditez avec l'instruction `UPDATE`

Vos utilisateurs souhaitent maintenant pouvoir modifier leurs recettes.

Pour cela, vous aurez besoin de deux nouveaux mots-clés : `UPDATE` et `SET` .

En imaginant qu'on fournit un formulaire d'édition et que l'on autorise les utilisateurs à éditer les champs `title` et `recipe` , voici la requête SQL correspondante :

mysql

```
1 UPDATE recipes SET title = :title, recipe = :recipe WHERE recipe_id = :id
```

Comment ça marche ?

- Tout d'abord, le mot-clé `UPDATE` permet de dire qu'on va modifier une entrée.
- Ensuite, le nom de la table (`recipes`).
- Le mot-clé `SET` sépare le nom de la table de la liste des champs à modifier.
- Viennent ensuite les champs qu'il faut modifier, séparés par des virgules. Ici, on modifie le champ `title` , puis on fait de même pour le champ `recipe` . Les autres champs ne seront pas modifiés.
- Enfin, le mot-clé `WHERE` est tout simplement indispensable. Il nous permet de dire à MySQL quelle entrée il doit modifier (sinon, toutes les entrées seraient affectées !). On se base très souvent sur le champ `recipe_id` pour indiquer **quelle entrée** doit être modifiée.

Et si je veux désactiver toutes les recettes d'un utilisateur, il va falloir faire les requêtes de mise à jour une à une ?

Non ! Il n'est pas question de passer des heures à modifier chaque entrée une à une pour ça ! En réfléchissant environ 0,5 seconde, vous allez trouver tout seul la requête SQL qui permet de faire ce qu'on souhaite.

C'est bon, vous avez trouvé ? Allez, je vous donne la réponse dans le mille :

mysql
```
1 UPDATE recipes SET is_enabled = 0 WHERE author = 'mickael.andrieu@exemple.com'
```

Supprimez des données avec `DELETE`

Enfin, voilà une dernière requête qui pourra se révéler utile : `DELETE` .

Rapide et simple à utiliser, elle est quand même un poil dangereuse : après suppression, il n'y a aucun moyen de récupérer les données, alors faites bien attention !

Voici comment on supprime par exemple une recette à partir de son identifiant :

mysql
```
1 DELETE FROM recipes WHERE recipe_id=:id
```

Il n'y a rien de plus facile :

- `DELETE FROM` : pour dire « supprimer dans » ;
- `recipes` : le nom de la table ;
- `WHERE` : indispensable pour indiquer quelle(s) entrée(s) doi(ven)t être supprimée(s).

Si vous oubliez le `WHERE` , toutes les entrées seront supprimées. Cela équivaut à vider la table.

Documentation générale PHP

1. Isset

Fonction :

isset — Détermine si une variable est déclarée et est différente de **null**

Description :

Détermine si une variable est considérée définie, ceci signifie qu'elle est déclarée et est différente de **nul**.

Si une variable a été détruite avec la fonction unset(), elle n'est plus considéré comme définie. **isset()** renverra **false** lors de la vérification d'une variable de valeur **null**.

Si plusieurs paramètres sont fournis, alors **isset()** retournera **true** seulement si tous les paramètres sont définis. L'évaluation s'effectue de gauche à droite et s'arrête dès qu'une variable non définie est rencontrée.

2. $_GET

Fonction :

$_GET — Variables HTTP GET

Description :

Un tableau associatif des valeurs passées au script courant via les paramètres d'URL (aussi connue sous le nom de "query string"). Notez que ce tableau n'est pas seulement rempli pour les requêtes GET, mais plutôt pour toutes les requêtes avec un query string.

3. Include

Fonction :

include — L'expression de langage `include` inclut et exécute le fichier spécifié en argument.

Description :

Les fichiers sont inclus suivant le chemin du fichier fourni ; si aucun n'est fourni, l'include_path sera vérifié. Si le fichier n'est pas trouvé dans l' include_path, include vérifiera dans le dossier du script appelant et dans le dossier de travail courant avant d'échouer.

4. Catch & Try

Fonction :

PHP a une gestion des exceptions similaire à ce qu'offrent les autres langages de programmation. Une exception peut être attrapée ("catch") dans PHP. Le code devra être entouré d'un bloc « try » pour faciliter la saisie d'une exception potentielle. Chaque « try » doit avoir au moins un bloc catch correspondant.

Description :

Un bloc catch définit comment réagir à une exception qui a été lancée. Un bloc catch définit un ou plusieurs types d'exceptions ou erreurs qu'il peut gérer, et optionnellement une variable dans laquelle assigner l'exception.

5. While

Fonction :

La boucle while est le moyen le plus simple d'implémenter une boucle en PHP

Description :

La signification d'une boucle while est très simple. PHP exécute l'instruction tant que l'expression de la boucle while est évaluée comme true. La valeur de l'expression est vérifiée à chaque début de boucle, et, si la valeur change durant l'exécution de l'instruction, l'exécution ne s'arrêtera qu'à la fin de l'itération (chaque fois que PHP exécute l'instruction, on appelle cela une itération). Si l'expression du while est false avant la première itération, l'instruction ne sera jamais exécutée.

6. $i ou $a

Fonction :

Les opérateurs d'incrémentation affectent les nombres et les chaînes de caractères et donnera comme résultat 1.

Description :

++$a	Pre-incrémente	Incrémente *$a* de 1, puis retourne *$a*.
$a++	Post-incrémente	Retourne *$a*, puis incrémente *$a* de 1.
--$a	Pré-décrémente	Décrémente *$a* de 1, puis retourne *$a*.
$a--	Post-décrémente	Retourne *$a*, puis décrémente *$a* de 1.

Langage PHP

Quelques différences :

PHP	HTML
Langage de script utilisé pour la création des pages web, il permet la génération des codes HTML et donc, de pages HTML.	Langage de balisage standardisé en vue de la création des pages web ou applications web
Dynamique, interactif	Statique
Peut charger des pages différentes pour chaque visiteur ou selon certaines contraintes. Chaque page pourra être adaptée selon le visiteur Exécuté côté SERVEUR	Ne possède pas la capacité de s'adapter au visiteur Chaque page va être exactement la même, quel que soit le visiteur Exécuté côté CLIENT

Il y a 3 instructions permettant d'afficher du contenu dans le browser (générer des codes HTML).

- echo (expression) ; *Nous utiliserons celle-ci dans le cadre du cours.
- print (expression) ;
- printf (expression) ;

Il existe 3 styles permettant d'intégrer du code PHP dans une page HTML :

Style	Exemple
Style **court**	<? Code PHP ?>
Style **XML**	<?php Code PHP ?>
Style **Script**	<script language="php"> Code PHP </script>

Ex : style XML

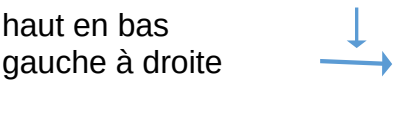
```
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <title>HelloWorld PHP</title>
5   </head>
6   <body>
7       <?php
8           echo "Hello World PHP!!!";
9       ?>
10  </body>
11  </html>
```

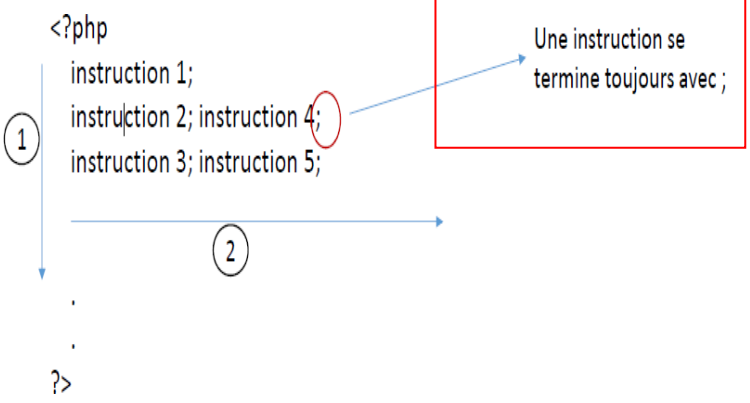
Code de PHP

La structure du code PHP

Dans un bloc de code PHP (<?php .. ?>), les instructions sont exécutées l'une après l'autre:

De haut en bas
De gauche à droite





IMPORTANT !!

Une instruction se termine toujours avec ;

L'ordre d'exécution des instructions ci-contre est :

1) Instruction1,
2) Instruction2,
3) Instruction4,
4) Instruction3,

Et enfin 5) instruction5.

Les variables

- Concept important en programmation même vocation qu'en mathématiques.
- Il s'agit d'un nom qui se réfère à une **valeur** ou une **chose**.
- La valeur d'une variable peut-être modifiée au cours du temps.

Un **nom de variable** en PHP commence par **$** et suivi par **([a-zA-Z_][a-zA-Z0-9_])**

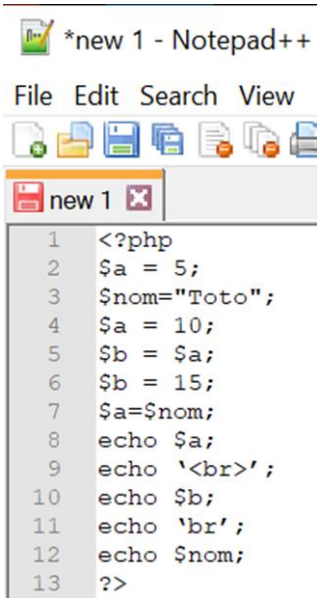
Exemple : $nom, $prenom, $date_de_naissance, ….

ATTENTION : le nom de variable est sensible à la casse : $nom et $Nom serontconsidérées comme deux variables différentes.

Une variable a un certain type, déterminé par son contenu.

- En PHP il n'est pas nécessaire de déclarer son type avant d'utiliser une variable.
- De plus, la variable peut changer de type au cours de l'exécution du programme.

Affectation et récupération de la valeur d'une variable



Variable	Ligne #	Valeur
$a	1	/
	2	5
	3	5
	4	10
	5	10
	6	10
	7	Toto
	8	Toto
	9	Toto
	10	Toto
	11	Toto
	12	Toto
	13	Toto
Valeur finale de $a : Toto		

Ligne 2 : On assigne la valeur **5** à la variable **$a**

Ligne 3 : Pas de changement pour **$a**

Ligne 4 : On "écrase" la valeur **5** pour donner la valeur **10** à **$a**

Ligne 5 : Pas de changement pour **$a**

Ligne 6 : Pas de changement pour **$a**

Ligne 7 : On "*écrase*" la valeur **10** pour donner la valeur de **$nom** (qui est "**Toto**" à ce moment-là du code) à **$a**

Lignes 8, 9, 10, 11, 12, 13 :
Pas de changement pour **$a**

Mise en application des variables :

Valeur1 = 4
Valeur2 = 2
Valeur3 = PINAN

```php
<body>
<?php
    $valeur1 = 4;
    $valeur2 = 2;
    $valeur3 = "PINAN";
    echo "$valeur1 + 6";
    echo "<br>"; echo"\n";
    echo "$valeur1 + $valeur2 <br>";echo"\n";
    echo 'Valeur1 = '.$valeur1. '<br>' ; echo"\n";
    $valeur1 /=$valeur2;
    echo 'valeur1 = '.$valeur1. '<br>' ; echo"\n";
    echo 'valeur2 = '.$valeur2. '<br>' ; echo"\n";
    $valeur3 = $valeur3." kevin";
    echo $valeur3.'<br>' ; echo"\n";
    $valeur1 = $valeur3;
    $valeur3=strlen($valeur1);
    $valeur2=(int)$valeur2;
    echo 'valeur1 = '.$valeur1. '<br>'; echo"\n";
    echo 'valeur2 = '.$valeur2. '<br>'; echo"\n";
    echo 'valeur3 = '.$valeur3. '<br>'; echo"\n";
?>
</body>
```

Rendu dans le navigateur :

4 + 6
4 + 2
Valeur1 = 4
valeur1 = 2
valeur2 = 2
PINAN kevin
valeur1 = PINAN kevin
valeur2 = 2
valeur3 = 11

Variables après modifications :

Valeur1 = PINAN KEVIN
Valeur2 = 2
Valeur3 = 9

Les constantes

Tout comme les variables, elles permettent de représenter une valeur mais celle-ci est fixe : une fois la constante définie, sa valeur ne peut plus changer.

- ✓ Un message d'erreur apparaîtra si l 'on change sa valeur !

Les constantes ne portent pas le symbole en début d'identificateur.

Syntaxe: define("*nom_constante*", VALEUR);

Exemple: define("PI", 3.14);
Echo PI; → 3.14
PI = 3.141592653; → Erreur

Les opérateurs

Opérateurs -Integer et Double

Considérons que nous avons initialisé les variables suivantes : **$a**= **5**; **$b**= **10**; **$c**= **0**;

Opérateurs arithmétiques	**Opérateurs d'assignement**
+: addition ($c = $a + $b; →c = 15)	**=**: affectation une valeur ($c = 50)
-: soustraction ($c = $b -$a; →c = 5)	**+=**: addition et affectation ($a+=5; →a = 10)
***:** multiplication ($c = $a * $b; →c = 50)	**-=**: soustraction et affectation ($b-=5; →$b=5)
/: division ($c = $b/$a; →c = 2)	***=**: multiplication et affectation ($a*=2; →a = 10)
++: incrémentation par 1 ($a++;→a = 6)	***/=**: division et affectation ($a/= 2; →a = 2.5)

Opérateurs – Chaîne de caractères

Une chaîne de caractères n'a *pas de limite en nombre de caractères*.
Elle est toujours délimitée par des **simples** ou **doubles guillemets.**
NB: Une chaine de caractères avec des **doubles guillemet** permet l'évaluation des variables et caractères spéciaux dans la chaîne.
E.g., $nom="Toto";
Echo "Nom : $nom"; //affiche Nom: Toto
Echo 'Nom : $nom'; //affiche Nom: $nom

Opération de concaténation : ' . '

E.g., $nom="PINAN"
Echo 'Nom : '.$nom; → Nom : PINAN
$nom = "Kevin ".$nom;
Echo $nom. → Kevin PINAN

Opérateurs - Boolean

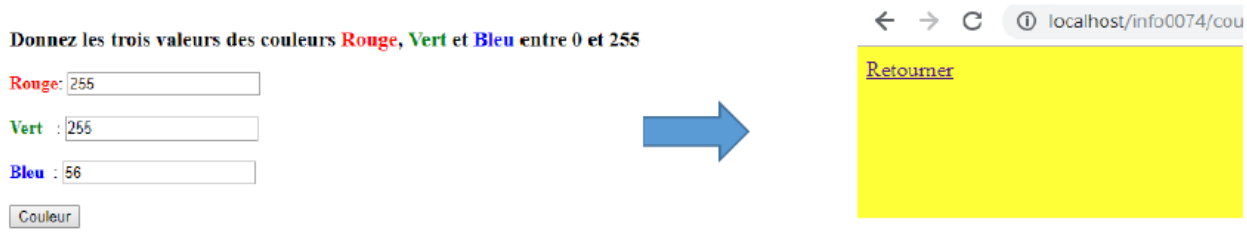
Une variable de type **booléen** peut prendre :
- soit la valeur **TRUE** (vrai),
- soit la valeur **FALSE** (faux).

Les formulaires

Création d'un formulaire avec $POST ou $GET

Nous allons réaliser une application qui permet à l'utilisateur d'encoder des valeurs pour 3 couleurs de base, c'est-à-dire, Rouge, Vert et Bleu. Ces valeurs peuvent aller de 0 à 255.

Ensuite, avec un bouton de type **submit**, le formulaire est envoyé à une page PHP qui récupère ces 3 valeurs et crée une page dont la couleur de fond correspond à la couleur formée à partir des valeurs encodées pour les 3 couleurs de base.




```html
<!DOCTYPE html>
<html>
<head>
    <title>Couleur_POST</title>
</head>
<body>
    <form action="couleur_post.php" method="post">
        <h3>Donnez les valeurs des trois couleurs:
        <b style="color:red">Rouge</b>,
        <b style="color:green">Vert</b> et
        <b style="color:blue">Bleu</b> entre 0 et 255</h3>


        <b style="color:red">Rouge</b>:
        <input type="number" name="rouge"> <br><br>

        <b style="color:green">Vert   </b>:
        <input type="number" name="vert"> <br><br>

        <b style="color:blue">Bleu  </b>:
        <input type="number" name="bleu"> <br><br>

        <input type="submit" name="couleur" value="Couleur">
    </form>
</body>
</html>
```

Couleur_post.html

Pour se faire, nous allons créer deux pages: *couleur.html* et *couleur.php*. Dans la page **couleur_post.html** *on définit le formulaire* et on *renvoie à la page* **couleur_post.php**

```php
<!DOCTYPE html>
<html>
<head>
    <title>Couleur</title>
</head>
<?php
    $couleur_r=$_POST["rouge"];
    $couleur_v=$_POST["vert"];
    $couleur_b=$_POST["bleu"];

?>
<body <?php echo "style=\"background-color:rgb($couleur_r,$couleur_v,$couleur_b);\"";?> >
    <a href="couleur.html">Retourner</a>
</body>
</html>
```

couleur_post.php

Chaque élément du formulaire doit avoir un nom. Il permet au programme php de récupérer la valeur encodée dans cet élément.

En PHP, on utilise:

$_POST["*nom_elemement_formulaire*"] pour récupérer la valeur (*méthode POST*)
$_GET["*nom_elemement_formulaire*"] pour récupérer la valeur (*méthode GET*)

Exemple :

<form name="nom_du_formulaire" action="URL_cible"method="post|get">
élément du formulaire
</form>

Action = URL cible (l'URL du programme qui va recevoir les données du formulaire.)

Method = post / get

Post -> Les données sont envoyées HTTP post transaction.
Get -> Les données sont envoyées en concaténant avec l'URL (max 2048 caractères)

Les conditions

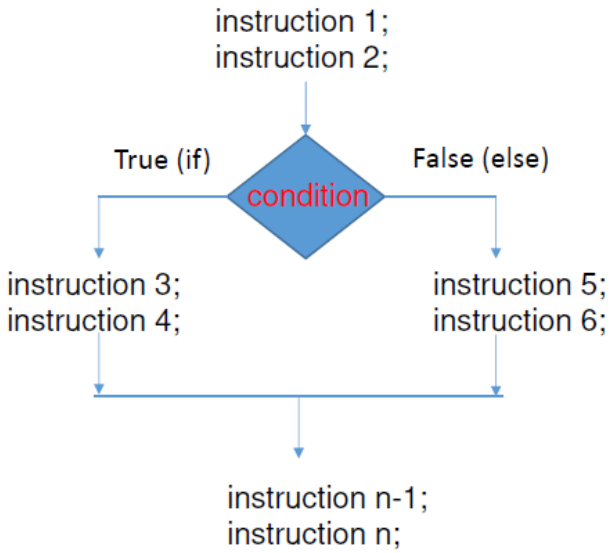
• Le programme est exécuté de manière **linéaire**, c'est-à-dire, *de haut en bas et de gauche à droite*. **Une instruction à la fois**.

• Les **instructions conditionnelles** *permettent de déterminer si certaines instructions doivent être exécutées ou non en fonction de l'évaluation d'une expression booléenne.*

Les **instructions conditionnelles en PHP** sont:
If (*condition***)**….
If (*condition***)**… **else**…
If (*condition 1***)**… **elseif (**condition 2**)**…**else**
Switch… « au cas où »

Symbole	Utilité	Action
==	Tester s'il est ***égale ou non***	Retourne **True** quand est ***égale***
!=	Tester s'il est ***différent ou non***	Retourne **True** quand est ***différent***
<	Tester s'il est ***inférieur ou non***	Retourne **True** quand est ***inférieur***
>	Tester s'il est ***supérieur ou non***	Retourne **True** quand est ***supérieur***
<=	Tester s'il est ***inférieur ou égale***	Retourne **True** quand est ***inférieur ou Égale***
>=	Tester s'il est ***supérieur ou égale***	Retourne **True** quand est ***supérieur ou Égale***

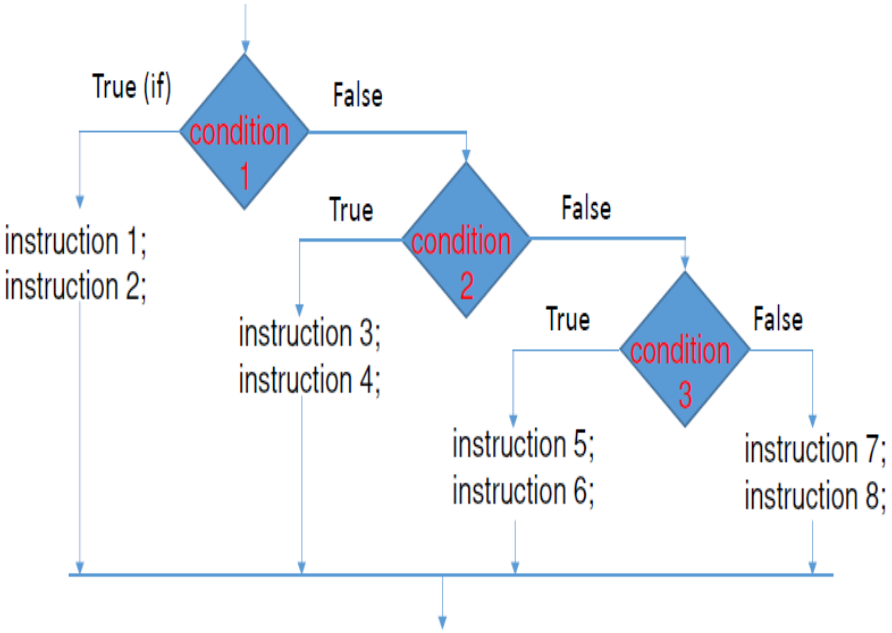
Instruction conditionnelle if …. Else ….




```
instruction 1;
instruction 2;
.
if (condition){
instruction 3;
instruction 4;
.
}else{
instruction 5;
instruction 6;
.
}
instruction n-1;
instruction n;
```

Instruction conditionnelle if …. Elseif …. else

```
if (condition 1){
instruction 1;
instruction 2;
.
}elseif(condition 2) {
instruction 3;
instruction 4;
.
}elseif(condition 3) {
instruction 5;
instruction 6;
.
}else{
instruction 7;
instruction 8;
.
}
```



Instruction Switch

Switch est similaire à if. Il permet de sélectionner une option parmi les blocs de code à être exécuté.

```
switch($var){
case val1:
instruction1;
break;
case val2:
instruction2;
break;
case val3:
instruction3;
break;
default:
instruction5;
}
```

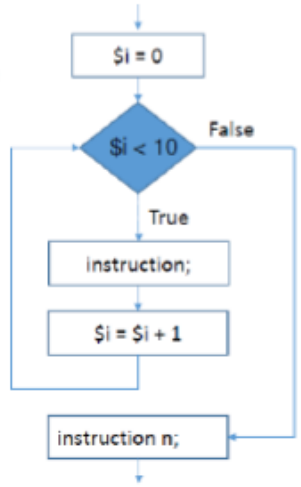
Les boucles

Dans le cas d'une boucle, comme pour les conditions, le programme est exécuté de manière linéaire, c'est-à-dire, de haut en bas et de gauche à droite. → Une instruction à la fois.

Les boucles permettent à certaines instructions d'être exécutées plusieurs fois en fonction d'une condition.

Boucle de type *for*

Permet d'exécuter un bloc de code un nombre de fois bien défini.



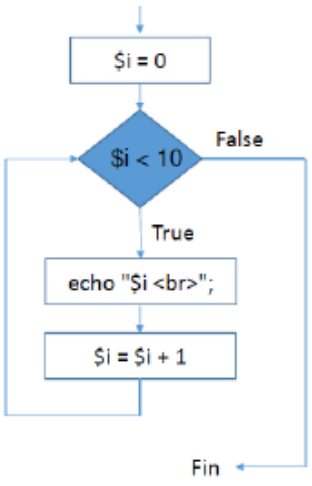
for ($i = 0; $i < 10; $i++) {
instruction;
}
instruction n;

Explication :

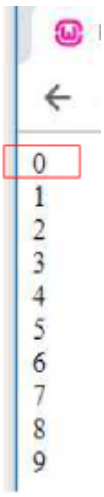
$i = 0 -> Initialisation du compteur
$i < 10 -> Condition
$i++ -> Incrémentation compteur

Illustration du fonctionnement d'une boucle de type ***for***

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Boucle For</title>
5  </head>
6  <body>
7      <?php
8          for ($i = 0; $i<10; $i++){
9              echo "$i <br>";
10         }
11     ?>
12 </body>
13 </html>
```



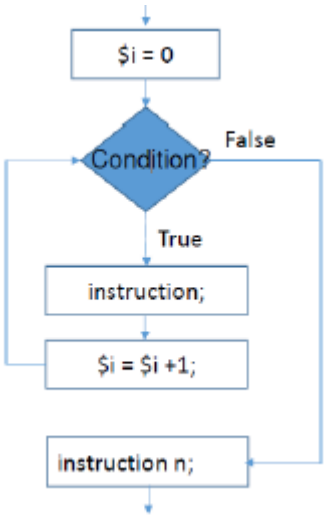
$i = 10



Boucle de type *while*

Permet d'exécuter un bloc de code quand la condition retourne « True »

```
$i = 0;
while($i < 10) {
instruction;
$i = $i+1;
}
instruction n;
```



Explication:

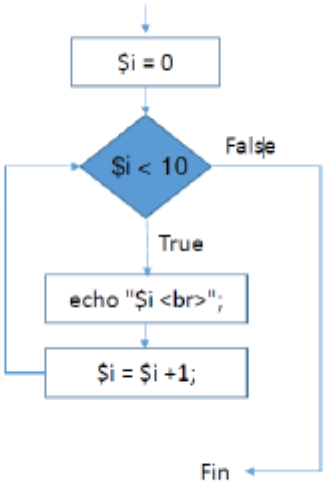
$i = 0 -> variable qui a une certaine valeur

$i < 10 -> condition

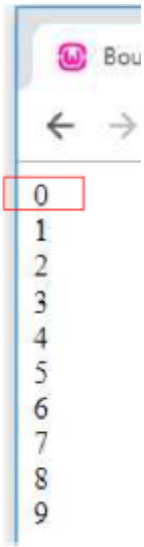
$i+1 -> Incrementation compteur

Illustration du fonctionnement d'une boucle de type ***while***

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Boucle While</title>
5  </head>
6  <body>
7      <?php
8          $i = 0;
9          while ($i < 10){
10             echo "$i <br>";
11             $i = $i + 1;
12         }
13     ?>
14 </body>
15 </html>
```



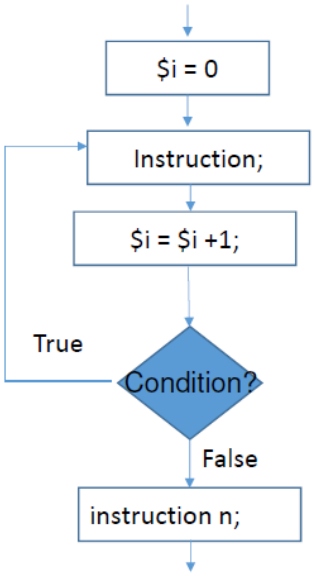
$i = 10



Boucle de type *do*

Il s'agit du même principe que pour la boucle ***while***: elle permet d'exécuter un bloc de code tant que la condition retourne « True ».

```
$i = 0;
do{
instruction;
$i = $i+1;
}while($i < 10);
instruction n;
```

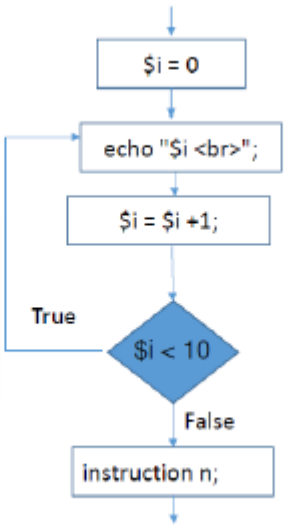


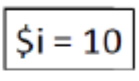
Explication :

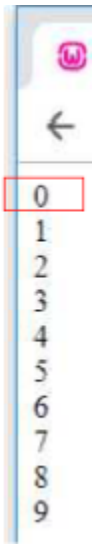
$i = 0 -> variable qui a une certaine valeur
$i = $i+1 -> Incrementation compteur
$i < 10 -> Condition

Illustration du fonctionnement d'une boucle de type ***do***

```
<!DOCTYPE html>
<html>
<head>
    <title>Boucle Do</title>
</head>
<body>
    <?php
        $i = 0;
        do{
            echo "$i <br>";
            $i = $i + 1;
        }while ($i < 10);
    ?>
</body>
</html>
```

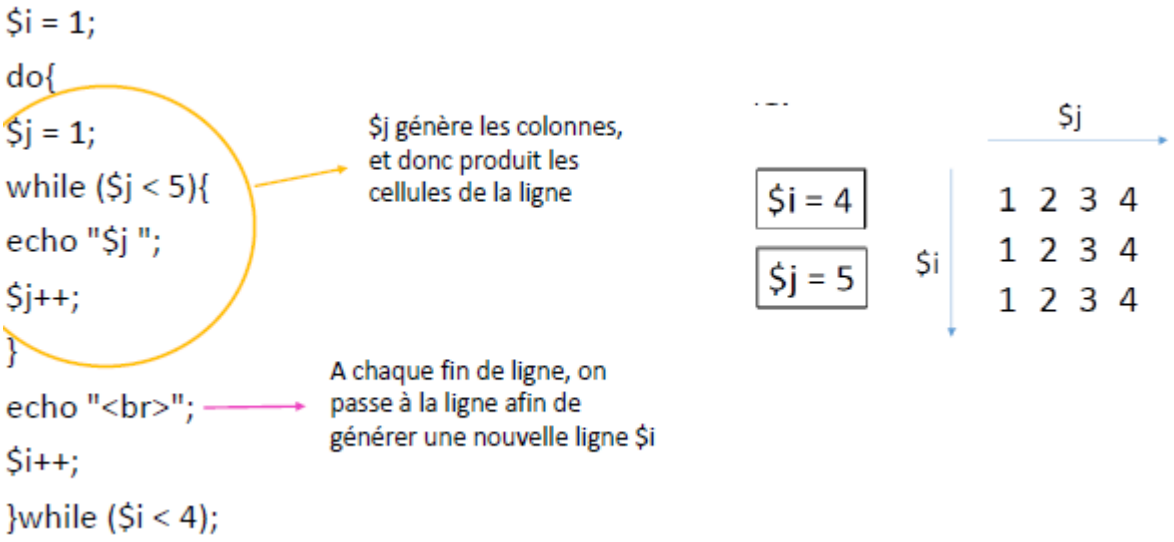






Boucles imbriquées

Il s'agit du même principe que les conditions imbriquées : on peut avoir une boucle qui se trouve dans une autre boucle:



Boucle foreach

Le PHP fournit une boucle spéciale: *foreach.* Elle permet de parcourir les éléments du tableau:

```
Foreach ($liste as$value){
…
}
Foreach ($listeas$key => $value){
…
}
```

Les tableaux

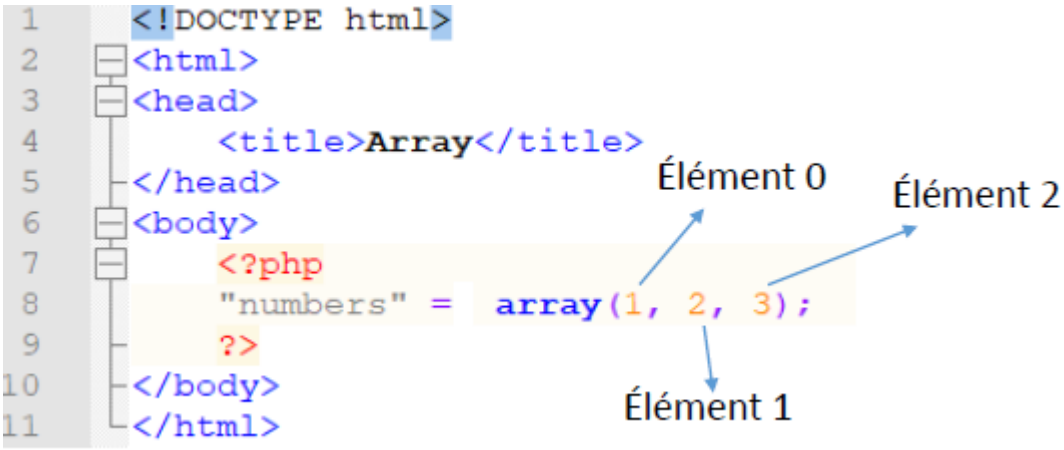
En PHP, les tableaux sont définis par une variable de type ***Array()***.

Le tableau peut stocker plusieurs valeurs.

En PHP, les éléments d'un tableau peuvent être de types différents et sont séparés par une virgule.

Chaque élément du tableau est numéroté par un indice.

La numérotation des indices d'un tableau commence toujours par 0.



<u>Il existe plusieurs manières de créer un tableau:</u>

- $liste = array(); //créer un tableau vide
- $liste = []; //créer un tableau vide
- $liste = array(1,2,3); //créer un tableau qui contient 3 éléments
- $liste = [1,2,3]; //créer un tableau qui contient 3 éléments

<u>Ajouter un élément dans le tableau:</u>

- $liste[] = 4; //l'élément ajouté se trouve en dernière place/indice du tableau

<u>Changer la valeur d'un élément dans le tableau:</u>

- $liste = [1,2,3];
- $liste[1] = 5; //l'élément d'indice 1 a désormais la valeur 5

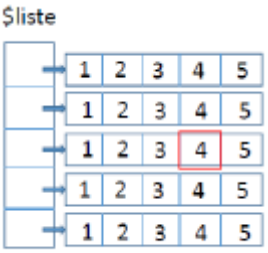
→ Le tableau est devenu [1,5,3]

Les tableaux multidimensionnels

Le tableau multidimensionnel est une variable de type tableau dont chaque élément contient une valeur de type tableau.

→Création d'un tableau de deux dimension (**matrice**):

```
$liste = array();
$liste[] = array(1,2,3,4,5);
$liste[] = array(1,2,3,4,5);
$liste[] = array(1,2,3,4,5);
$liste[] = array(1,2,3,4,5);
$liste[] = array(1,2,3,4,5);
```



Pour accéder à un élément du tableau, on doit indiquer l'indice du premier tableau puis l'indice du deuxième tableau.

Exemple: echo $liste [2] [3];

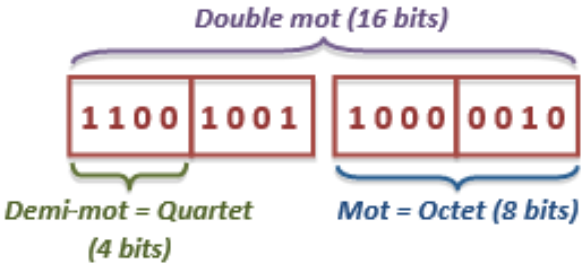
Module 9 : Base et arithmétique

Informations

Un système numérique doit, avant de pouvoir manipuler les nombres, les textes, les images ou les sons, de les représenter comme des suites de 0 et de 1. La valeur 0 ou 1 est appelée booléen, chiffres binaires ou encore bit (binary digit).

Une suite de chiffres binaires est appelée mot binaire. Suivant le processeur, la taille du mot sera différente. Les mots les plus courants ont une taille de 8, 16, 32 ou 64 bits.

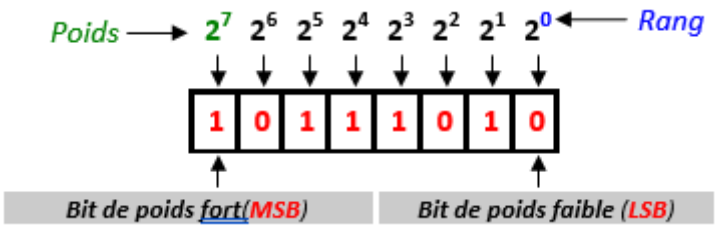
On rencontre aussi les notions de demi-mots ou de double-mot.



Pour représenter un nombre, il est possible d'utiliser différentes bases parmi lesquelles on peut citer : la base 10 (décimale) qui est la base universelle ; la base 2 (binaire) qui est la base utilisée en informatique et dans l'algèbre de Boole ; la base 16 (base hexadécimale) qui est une base utilisée en informatique.

Le système binaire

Dans un système binaire ou base 2, il n'y a que deux chiffres possibles : 0 et 1.
Le système de numération binaire est indiqué par l'**indice 2**.



Le tableau suivant indique comment les bits d'un octet sont convertis d'un code binaire en une valeur décimale.

Code binaire	Valeurs binaires	Valeur décimale
00000000	**0**	**0**
00000001	**1**	**1**
00000011	**1+2**	**3**
00000111	**1+2+4**	**7**
00001111	**1+2+4+8**	**15**
00011111	**1+2+4+8+16**	**31**
00111111	**1+2+4+8+16+32**	**63**
01111111	**1+2+4+8+16+32+64**	**127**
11111111	**1+2+4+8+16+32+64+128**	**255**

Conversion décimal – binaire

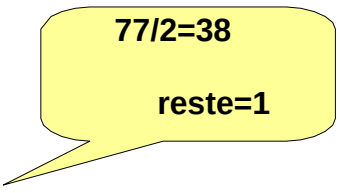
Convertissons 01001101 en décimal à l'aide du schéma ci-dessous :

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	0	1	1	0	1

Le nombre en base 10 est $2^6 + 2^3 + 2^2 + 2^0 = 64 + 8 + 4 + 1 = 77$.

Allons maintenant dans l'autre sens et écrivons 77 en base 2. Il s'agit de faire une suite de divisions euclidiennes par 2. Le résultat sera la juxtaposition des restes.
La méthode : On met le nombre à droite, on le divise par 2 et on note le résultat à sa droite et le reste juste en-dessous

nombre_a_gauche/2	**0**	**1**	**2**	**4**	**9**	**19**	**38**	**77**
reste	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**



Addition binaire

L'addition en binaire se fait avec les mêmes règles qu'en décimale : On commence à additionner les bits de poids faible (les bits de droite) puis on a des retenues lorsque la somme de deux bits de même poids dépasse la valeur de l'unité la plus grande (dans le cas du binaire : 1), cette retenue est reportée sur le bit de poids plus fort suivant...

Par exemple :

```
  0 1 1 0 1
+ 0 1 1 1 0
- - - - - -
  1 1 0 1 1
```

table d'addition

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10 (on pose 0 et on retient 1)

1+1+1 = 11 (on pose 1 et on retient 1)

Exercices :

1. 1011 + 0110 =

2. 10101+ 1101=

3. 110111 + 111011=

4. 1010111 + 10010 =

5. 11011001 + 110110 =

6. 101010001 + 1011100 =

7. 10110110101 + 1001110 =

8. 1101010011 + 101110 =

9. 100110100 + 10101110 =

10. 1010101010 + 10011111 =

Module 10 : Base de la programmation

CMD

Introduction

Les fichiers batch sont des scripts de commandes qui s'exécutent dans l'interpréteur de commande Windows. Alors qu'elle pourrait être l'utilité d'écrire ces lignes de commande dans un script ? En fait, il y a plusieurs raisons :

- éviter d'avoir à réécrire sans cesse les mêmes commandes lors de tâches répétitives ;
- possibilité de créer de vrais petits « programmes » facilitant les tâches qui doivent être réalisées via l'interpréteur de commande.

Nous allons étudier les méthodes permettant de concevoir ces programmes. Dans la suite de ce document, nous utiliserons la convention suivante :

- les termes « interpréteur de commande » ou « interpréteur » désignent l'exécutable « cmd.exe » ;
- dans les syntaxes de commande, les parties encadrées avec les caractères [et] sont optionnelles, les parties encadrées avec les caractères < et > sont à remplacer par différentes informations à fournir (celles-ci seront précisées avec la syntaxe) et les parties encadrées avec les caractères { et } sont des parties à choix multiple où chaque choix est séparé des autres avec le caractère |.

La convention syntaxique est la même que celle pour l'aide en ligne de commande, ainsi il vous sera plus facile de la comprendre.

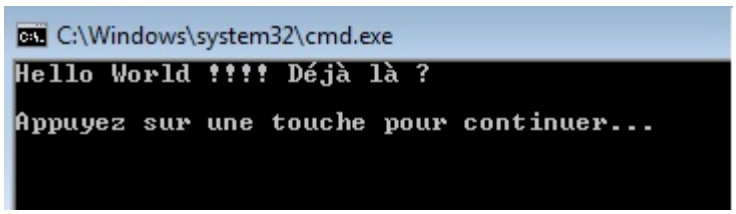
Hello World

Nous allons commencer par le traditionnel « hello world », dont voici le code (Script 1). Copiez le code dans Notepad++, puis enregistrez-le avec l'encodage OEM-850 et l'extension « .bat » ou « .cmd » (seulement sur les systèmes Vista et supérieur).

Script 1

```
@echo off
cls
rem Ceci est une remarque.
:: Ceci est un commentaire.

echo Hello World !!!! Déjà là ?
echo.
pause
```

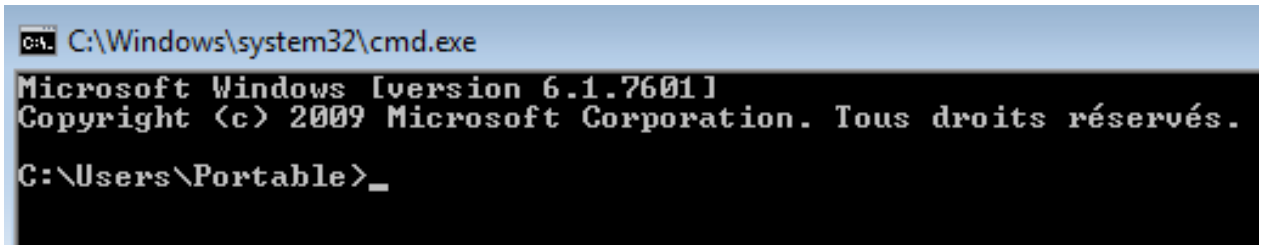
Lorsqu'on exécute ce script en cliquant dessus, on obtient l'affichage suivant.



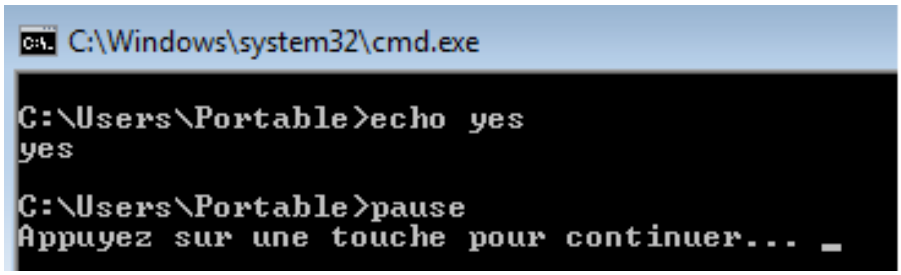
Étudions la composition du script 1. La première ligne, @echo off, est déjà intéressante, elle est composée :

- du préfixe @ qui sert à inverser l'état de l'affichage standard ;
- de la commande echo qui sert à gérer l'affichage en ligne de commande ;
- et du paramètre off qui sert à désactiver l'affichage standard.

L'affichage standard définit ce que l'interpréteur de commande affiche par défaut. Par exemple lors du lancement de l'interpréteur de commande ci-dessous ; l'affichage standard renvoie le chemin d'accès du répertoire courant, soit C:\Users\Portable>.



Lors de l'exécution d'un script, l'affichage standard renvoie, par défaut, le chemin d'accès du répertoire courant suivi de la commande en cours d'exécution comme dans l'exemple ci-dessous.



Le préfixe @, quand il est placé en début de ligne, sert à inverser l'état de l'affichage standard (activé ou désactivé) pour l'exécution de la commande qui le suit (pas seulement pour la commande echo). Ce comportement prend fin une fois la commande exécutée. Ainsi la commande @cd /d "C:\Users\Portable\" ne sera affichée que si l'affichage standard est désactivé. La syntaxe du préfixe @ est la suivante (où <commande> est la commande à exécuter) :

@<commande>

La commande echo gère l'affichage dans l'interpréteur, elle peut :

- modifier l'état de l'affichage standard ;
- afficher l'état de l'affichage standard ;
- afficher un message ou une ligne vide dans l'interpréteur.

Désactiver l'affichage standard peut être fait via la syntaxe suivante (seuls les erreurs et les messages de la commande echo sont affichés).

```
echo off
```

Activer l'affichage standard peut être fait via la syntaxe suivante (tout est affiché).

```
echo on
```

Utilisée sans paramètre, la commande echo renvoie l'état de l'affichage standard en cours.

```
echo
```

Si l'on reprend le script 1, la ligne @echo off permet de désactiver l'affichage standard sans que la commande soit affichée. Sur la deuxième ligne du script 1, la commande cls est utilisée pour vider la fenêtre de l'interpréteur de son contenu, cette commande ne prend aucun paramètre. Sa syntaxe est donc la suivante.

```
cls
```

La ligne suivante du script 1 est vide, elle ne sera donc pas prise en compte lors de l'exécution du script permettant ainsi de le rendre plus lisible. La quatrième ligne est composée de la commande rem et d'une chaîne de caractères, cette commande permet d'insérer des remarques dans son script. Si et seulement si l'affichage standard est activé, la commande rem sera affichée. La syntaxe de la commande rem est la suivante (où <remarque> est la chaîne de caractères à insérer en remarque).

```
rem <remarque>
```

La cinquième ligne du script 1, :: Ceci est un commentaire., est composée du préfixe :: et d'une chaîne de caractères. Le préfixe :: définit la chaîne de caractères qui le suit comme étant un commentaire ; ce comportement prend fin au premier retour à la ligne. Quel que soit l'état de l'affichage standard, la chaîne de caractères préfixée par :: ne sera pas affichée. La syntaxe est la suivante (où <commentaire> est le commentaire à insérer).

```
::<commentaire>
```

Suit une autre ligne vide puis la commande echo Hello World !!!! Déjà là ? qui affiche Hello World !!!! Déjà là ? dans la fenêtre de l'interpréteur. La syntaxe suivante permet donc d'afficher un message même si l'affichage standard est désactivé (où <message> est le message à afficher).

echo <message>

Vient ensuite la commande echo. qui permet d'afficher la ligne vide que l'on voit dans l'affichage obtenu. Si un point suit directement la commande echo et qu'après le point il y a un retour à la ligne, celle-ci affiche une ligne vide.

echo.

Sur la ligne suivante se trouve la commande pause qui met en pause l'exécution du script jusqu'à ce que l'utilisateur appuie sur une touche quelconque du clavier, elle affiche le message Appuyez sur une touche pour continuer… (quel que soit l'état en cours de l'affichage standard) et ne prend aucun paramètre. Dans le script 1, cette commande permet de visualiser le contenu de la fenêtre de l'interpréteur avant que celle-ci ne se ferme.

pause

Différence entre la commande « REM » et le préfixe « : : »

```
cls
rem Remarque 1
:: Commentaire 1
@echo off
rem Remarque 2
:: Commentaire 2
Pause
```

Comme vous le voyez dans l'affichage du script 2, la commande rem Remarque 1, est présente à l'écran ; l'affichage standard étant activé, les commandes exécutées sont toutes affichées. La chaîne :: Commentaire 1 n'est pas affichée, c'est dû au fait que le préfixe :: n'est pas une commande et donc n'est pas renvoyé par l'affichage standard. Vient ensuite la commande @echo off qui désactive l'affichage standard sans que la commande ne soit affichée, suivie de la commande rem Remarque 2 qui n'est pas affichée (l'affichage standard étant à présent désactivé), pas plus que la chaîne :: Commentaire 2 qui est de toute façon exclue par l'affichage standard.

Les variables

Les variables sont gérées via la commande set, il en existe deux types dont voici la liste :

- les variables de type chaîne de caractères ;
- les variables de type nombre entier signé.

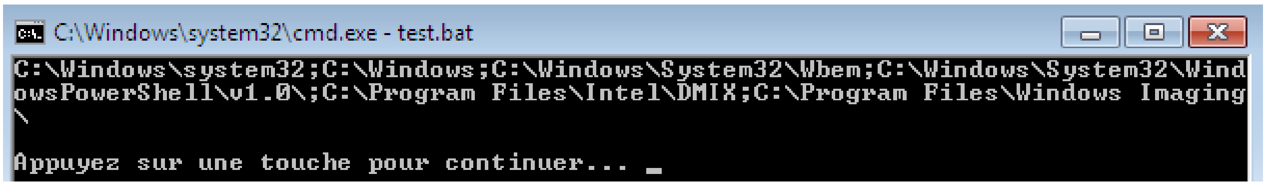
Chaque type de variable est déclaré et traité différemment ; il est possible de les transtyper, c'est-à-dire de les passer d'un type à l'autre, à condition de respecter les règles suivantes :

- une chaîne de caractères ne pouvant être transtypée en nombre que si elle ne contient que des chiffres ;
- un nombre pourra toujours être transtypé en chaîne de caractères (qui ne contiendra que des chiffres).

Les variables sont accessibles via un identifiant qui leur est propre. Ainsi chaque fois qu'il sera fait mention de cet identifiant, l'interpréteur sera capable de fournir la valeur associée. Cela peut être fait en utilisant le caractère % de part et d'autre de l'identifiant, on parle alors d'expansion de la variable. Par exemple avec la variable PATH (qui est une variable d'environnement : fournie par le système), pour obtenir sa valeur ; il faudrait utiliser la syntaxe %PATH% comme dans le script 8.

```
@echo off
echo %PATH%
pause
```

Ce qui produirait un affichage similaire à celui ci-dessous.



Les identifiants de variable ne sont pas sensibles à la casse, c'est-à-dire que l'interpréteur ne fait pas la différence entre majuscule et minuscule lorsqu'il expanse, crée ou modifie une variable. Ainsi les identifiants PATH, Path et path désignent une seule et même variable.

Listes des variables fréquentes :

%COMPUTERNAME% -> Le nom de l'ordinateur

%HOMEDRIVE%->Indique où se situe le lecteur principal

%HOMEPATH%->Indique où se trouve le dossier personnel de l'utilisateur en cours

%LOGONSERVER%->Donne l'adresse réseau de l'ordinateur

%OS%->Donne le système d'exploitation installé

%PATH%->Donne les adresses PATH

%PROCESSOR_ARCHITECTURE%->Donne l'architecture du processeur

%PROCESSOR_IDENTIFIER%Donne l'identification du processeur

%PROGRAMFILES%->Donne l'adresse du répertoire des programmes

%SYSTEMDRIVE%->Donne le lecteur principal

%SYSTEMROOT%->Donne le répertoire où se situe le système

%TEMP%->Donne l'adresse du répertoire temporaire

%USERNAME%->Donne le nom de l'utilisateur

%USERPROFILE%->Donne le répertoire personnel de l'utilisateur en cours

%WINDIR%->Donne le répertoire où est installé Windows

%DATE%->Date de l'ordinateur

%TIME%->L'heure de l'ordinateur

La commande « SET »

La commande set gère les variables dans l'interpréteur, elle permet :

- de créer une variable ;
- d'attribuer une valeur à une variable ;
- de modifier le contenu d'une variable ;
- de supprimer le contenu d'une variable ;
- d'effectuer des opérations mathématiques ou logiques entre des nombres ;
- de récupérer la saisie d'un utilisateur pour la placer dans une variable ;
- et de transtyper le contenu d'une variable.

Voici la syntaxe de la commande set.

Attribue à la variable une valeur sous forme de chaîne de caractères (où <variable> est son identifiant et <chaîne> est une chaîne de caractères).

set ["]<variable>=<chaîne>["]

Attribue à la variable une valeur sous forme d'un nombre entier signé (où <expression> est une expression numérique à évaluer : détaillé par la suite).

set /a ["]<expression>["]

Attribue à la variable une valeur, saisie par l'utilisateur, sous forme de chaîne de caractères après l'affichage de la chaîne d'invite si elle est spécifiée (où <variable> est son identifiant et où <chaîne_invite> est une chaîne de caractères affichée à l'utilisateur afin de l'inviter à saisir une chaîne au clavier).

set /p ["]<variable>=[<chaîne_invite>]["]

Supprime la valeur de la variable de la mémoire, son identifiant reste indexé par l'interpréteur, mais sa valeur est indéfinie.

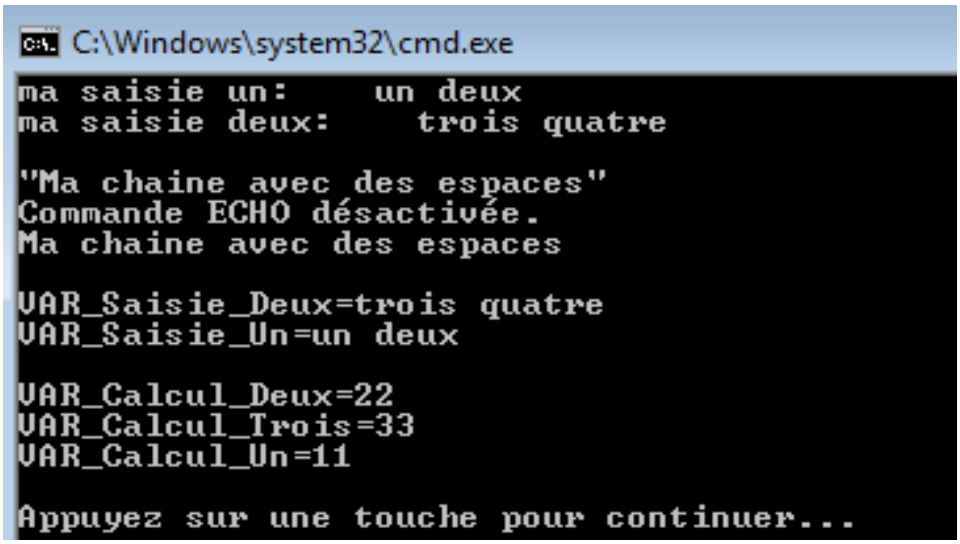
set ["]<variable>=["]

Voici les règles usuelles de la commande set :

- Si la commande set est employée sans paramètre, elle affiche les variables définies dans le contexte courant (détaillé par la suite).
- Si elle est utilisée avec comme paramètre une chaîne (ou un nom de variable), sans valeur ni signe égal ; alors elle affiche la variable dont le nom correspond à la chaîne donnée en paramètre et/ou les variables ayant un nom commençant par la chaîne donnée en paramètre.
- Si elle est utilisée avec un nom de variable et un signe égal sans valeur alors le contenu de la variable est vidé de la mémoire, il est possible de tester si une variable est définie, mais nous aborderons ce point au prochain chapitre.
- Toute chaîne non numérique dans l'expression à évaluer est traitée comme un identifiant de variable et est convertie en nombre avant d'être utilisée (dans son utilisation avec le paramètre /a), si la variable n'existe pas ou qu'elle est indéfinie, elle prend comme valeur 0.
- Une expression numérique à évaluer devrait toujours être placée entre guillemets afin de permettre l'utilisation des opérateurs logiques et des opérateurs de groupement.

Voyons plus en détail le fonctionnement de la commande set.

```
@echo off
set VAR_Espace_Un="Ma chaîne avec des espaces"
set VAR_Espace_Deux"=Ma chaîne avec des espaces"
set "VAR_Espace_Trois=Ma chaîne avec des espaces"
::il y a quatre espaces à la fin de la ligne suivante
set /p VAR_Saisie_Un=    ma saisie un:
set /p "VAR_Saisie_Deux=    ma saisie deux:    "
set /a VAR_Calcul_Un=1+10
set /a VAR_Calcul_Deux"=2+20"
set /a "VAR_Calcul_Trois=3+30"
echo.
echo %VAR_Espace_Un%
echo %VAR_Espace_Deux%
echo %VAR_Espace_Trois%
echo.
set VAR_Saisie
echo.
set VAR_Calcul
echo.
Pause
```



Comme vous pouvez le constater, les guillemets occupent une place prépondérante dans ce script ; observez bien où ils sont placés. De la ligne 2 à la ligne 4, les valeurs des variables sont des chaînes de caractères avec des espaces. La déclaration de la variable VAR_Espace_Un se fait avec des guillemets placés de part et d'autre de la chaîne, la déclaration de la variable VAR_Espace_Deux se fait avec des guillemets : un placé avant le signe égal et l'autre à la fin de la chaîne ; la déclaration de la troisième variable VAR_Espace_Trois se fait avec des guillemets : un placé avant le nom de la variable et l'autre à la fin de la ligne.

Si on se reporte à l'affichage obtenu, on remarque que la première et la troisième variable affichent une sortie correcte ; la deuxième variable quant à elle ne fonctionne pas ; on déduit donc facilement que si les guillemets s'ouvrent avant l'identifiant de la variable et se ferment après la valeur de la variable, ils ne font pas partie de la variable ; et si les guillemets s'ouvrent avant la valeur de la variable et se ferment après la valeur de la variable, ils font partie de la variable.

Les deux saisies de l'utilisateur donnent aussi un résultat intéressant, la ligne de commande qui déclare VAR_Saisie_Un n'utilise pas de guillemets ; cependant la ligne de commande qui déclare VAR_Saisie_Deux, elle, en utilise.

Lors de l'affichage des invites de saisie utilisateur, les espaces se trouvant avant les chaînes d'invites ont disparu, et ceux se trouvant après sont affichés. On en déduit que les espaces en début de ligne sont ignorés et que ceux de la fin sont considérés comme faisant partie de la chaîne d'invite.

En ce qui concerne les variables VAR_Calcul_XXX, elles donnent toutes un résultat valide cependant il est toujours préférable de respecter les règles de syntaxe ; c'est-à-dire toujours mettre l'expression numérique entre guillemets. Cela sera utile dans la suite du chapitre.

PowerShell

Utilisez le PowerShell

PowerShell, ou *Windows PowerShell*, anciennement ***Microsoft Command Shell*** **(MSH)**, est une suite logicielle développée par **Microsoft** qui intègre une **interface en ligne de commande**, un langage de **script** nommé PowerShell ainsi qu'un kit de développement. Il est inclus depuis Windows 7 et s'appuie sur le framework Microsoft.NET.

Les commandes PowerShell sont constituées d'un **verbe** ou **préfixe** et d'un nom, séparés par un tiret. Elles peuvent être suivies de paramètres, on les appelle des ***commandlets*** ou ***cmdlets*** (***command applets*** en anglais ou ***phrases*** en français).

Le préfixe de **cmdlet** est appelé ***verbe*** car il détermine l'action à effectuer sur les entités désignées dans la phrase. Voyons-en quelques-uns des plus utiles :

- Add permet d'**ajouter** des données ou informations sur le nom qui le suit ;
- Get permet d'**obtenir** des données ou informations sur le nom qui le suit ;
- Read permet de **lire** des données ou informations sur le nom qui le suit ;
- Clear permet de **réinitialiser** l'affichage de l'interface ;
- **I**mport et Export permettent d'**importer/exporter** des fichiers de commande ou des **Alias** ;
- New permet de **créer** de nouveaux objets ou variables ;
- Set permet de **définir** des données ou informations sur le nom qui le suit ;
- Write permet de d'**écrire** des données ou informations sur le nom qui le suit et peut agir comme le compte rendu d'une commande.

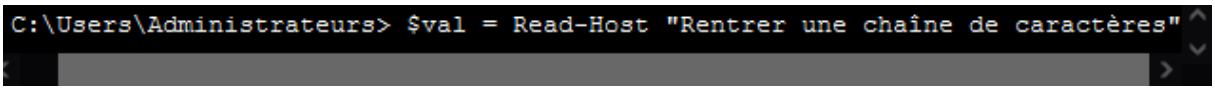
PowerShell est plus proche des langages de **script** comme **Perl** que de langages Shell comme Bash, il n'y a aucune ressemblance entre le langage PowerShell et le **Batch** des fichiers **.BAT** de **DOS/Windows**. Le but de **Microsoft** est de faire un langage de **script** complètement intégré, avec autant de fonctionnalités que celles qui existent sous Unix (ou Linux), et avec le même niveau de sécurité.

Récupérez des chaînes de caractères

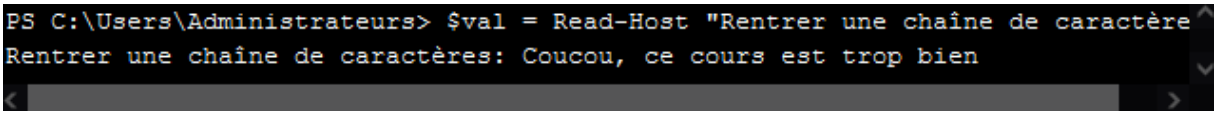
Je lance ma console PowerShell, je trouve la console bleue jolie, j'utiliserai donc cette console pour la suite de ce chapitre.

Nous allons voir une fonction très intéressante : Read-Host. C'est une fonction qui permet de saisir une chaîne de caractères et de l'enregistrer dans une variable ; on utilisera cette fonction souvent dans la suite du cours.

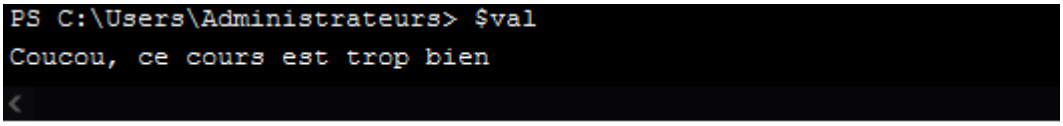
On y va, tapez la commande suivante :



La console vous demande donc de saisir une chaîne de caractères. Vous pouvez entrer la chaîne de caractères de votre choix. Pour ma part, j'écrirai "Coucou, ce cours est trop bien" :



Pour afficher votre chaîne de caractères, tapez :



Et voilà, la variable **$val** vous retourne votre chaîne de caractères "Coucou, ce cours est trop bien" !
Bon, vous allez me demander à quoi ça sert ? Eh ben vous le verrez très rapidement. En effet, dans les scripts que nous allons écrire ensemble, on a besoin que l'utilisateur puisse saisir des informations comme des noms d'utilisateurs, des noms de groupes, des noms d'OU (*unité d'organisation*), des noms de partage, etc.

Gérez vos fichiers et dossiers simplement avec PowerShell

Lancez PowerShell, vous devriez vous trouver dans le répertoire par défaut utilisateur : **C:\Users\Administrateurs**. Bien sûr que si vous vous connectez avec un autre utilisateur, vous serez par défaut dans le répertoire utilisateur.

Par savoir où l'on se situe, on va utiliser la commande **Get-Location :**

```
PS C:\Users\Administrateur> Get-Location

Path
----
C:\Users\Administrateur
```

Cool, nous voyons que nous sommes dans notre bon répertoire. C'est la même commande que *pwd* sous **Linux** (ou **Unix**).

Pour afficher le contenu d'un dossier, utilisez la commande Get-ChildItem :

```
PS C:\Users\Administrateur> Get-ChildItem


    Répertoire : C:\Users\Administrateur


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-r---        28/06/2019 11:59                3D Objects
d-r---        28/06/2019 11:59                Contacts
d-r---        07/07/2019 17:59                Desktop
d-r---        28/06/2019 11:59                Documents
d-r---        28/06/2019 11:59                Downloads
d-r---        28/06/2019 11:59                Favorites
d-r---        28/06/2019 11:59                Links
d-r---        28/06/2019 11:59                Music
d-r---        28/06/2019 11:59                Pictures
d-r---        28/06/2019 11:59                Saved Games
d-r---        28/06/2019 11:59                Searches
d-r---        28/06/2019 11:59                Videos
```

Allez, nous allons créer un répertoire de sauvegarde **SAVE** dans notre dossier utilisateur ; utilisons la commande **New-Item** :

```
PS C:\Users\Administrateur> New-Item -Name "SAVE" -ItemType Directory


 Répertoire : C:\Users\Administrateur


Mode           LastWriteTime    Length Name
----           -------------    ------ ----
d-----      17/09/2019 10:38           SAVE
```

Bon, si on visualise le contenu de notre répertoire courant, nous verrons bien que le dossier **SAVE** a bien été créé.

On peut également utiliser la commande **New-Item** pour créer un fichier texte en tapant :

```
PS C:\Users\Administrateur> New-Item -Name "utilisateurs.txt" -ItemType File
```

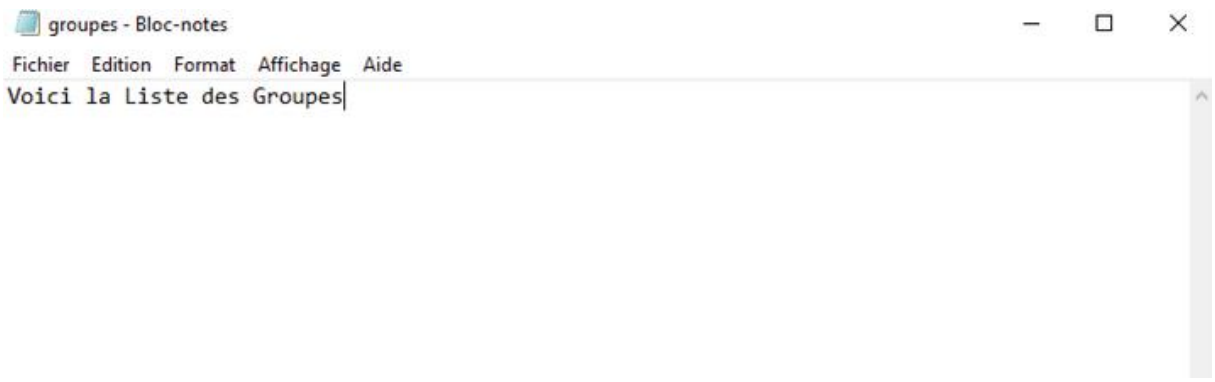
Je peux utiliser aussi cette commande pour créer un fichier texte **groupes.txt** et y insérer du texte en tapant :

PS C:\Users\Administrateur> New-Item -Name "groupes.txt" -ItemType File -Value "Voici la Liste des Groupes"

```
PS C:\Users\Administrateur> New-Item -Name "groupes.txt" -ItemType File -Value "Voici l
```

Pour visualiser le contenu du fichier texte, tapez la commande suivante **:**

```
PS C:\Users\Administrateur> notepad.exe groupes.txt
```



Manipulez vos fichiers avec PowerShell

Nous allons copier notre fichier utilisateurs.txt dans notre répertoire **SAVE** en tapant la commande suivante :

```
PS C:\Users\Administrateur> Copy-Item -Path utilisateurs.txt -Destination SAVE
```

Si nous allons voir le contenu de **SAVE**, nous verrons que notre fichier **utilisateurs.txt** a bien été copié.

```
PS C:\Users\Administrateur> Get-ChildItem SAVE


 Répertoire : C:\Users\Administrateur\SAVE


Mode           LastWriteTime Length Name
----           ------------- ------ ----
-a----      06/07/2019 15:35      0 utilisateurs.txt
```

Vous remarquerez que pour visualiser le contenu de **SAVE**, j'ai utilisé la commande **Get-ChildItem** avec un attribut (**SAVE**) sans m'être déplacé dans mon répertoire **SAVE** ; pensez à utiliser les attributs avec vos commandes **PowerShell**, elles vous permettent d'aller souvent plus vite.

Bon c'est cool, et si je veux déplacer maintenant le fichier **groupes.txt**, j'utiliserai la commande suivante :

PS C:\Users\Administrateur> Move-Item -Path C:\Users\Administrateur\groupes.txt -Destination C:\Users\Administrateur\SAVE

```
PS C:\Users\Administrateur> Move-Item -Path C:\Users\Administrateur\groupes.txt -Destin
```

Vérifions le contenu de nos 2 répertoires, notre fichier **groupes.txt** a bien été déplacé.

```
PS C:\Users\Administrateur> Get-ChildItem


    Répertoire : C:\Users\Administrateur


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-r---        28/06/2019 11:59                3D Objects
d-r---        28/06/2019 11:59                Contacts
d-r---        07/07/2019 17:59                Desktop
d-r---        28/06/2019 11:59                Documents
d-r---        28/06/2019 11:59                Downloads
d-r---        28/06/2019 11:59                Favorites
d-r---        28/06/2019 11:59                Links
d-r---        28/06/2019 11:59                Music
d-r---        28/06/2019 11:59                Pictures
d-r---        28/06/2019 11:59                Saved Games
d-r---        28/06/2019 11:59                Searches
d-r---        28/06/2019 11:59                Videos
-a----        06/07/2019 15:35              0 utilisateurs.txt

PS C:\Users\Administrateur> Get-ChildItem SAVE


    Répertoire : C:\Users\Administrateur\SAVE


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        06/07/2019 15:37             26 groupes.txt
-a----        06/07/2019 15:35              0 utilisateurs.txt
```

Bon, si vous regardez de plus près, je viens de faire le déplacement du fichier **groupes.txt** différemment de la copie du fichier **utilisateurs.txt**. Eh oui, on a utilisé ce que l'on appelle les ***chemins relatifs*** pour la copie du fichier **utilisateurs.txt**, c'est-à-dire que l'on n'a pas utilisé d'**anti-slash**. Si j'utilise les **chemins absolus**, je dois taper le lien complet du fichier et du dossier de destination, et c'est ce que je viens de faire pour le déplacement du fichier **groupes.txt**.

On peut renommer également notre fichier **utilisateurs.txt** dans notre répertoire **SAVE** par **utilisateurs_save.txt** en tapant la commande suivante :

```
PS C:\Users\Administrateur> Rename-Item -Path C:\Users\Administrateur\SAVE\utilisateurs.txt -NewName
C:\Users\Administrateur\SAVE\utilisateurs_save.txt
```

```
PS C:\Users\Administrateur> Rename-Item -Path C:\Users\Administrateur\SAVE\utilisateurs
```

On peut voir que notre fichier **utilisateurs.txt** a bien été renommé en **utilisateurs_save.txt** :

```
PS C:\Users\Administrateur> Get-ChildItem SAVE


    Répertoire : C:\Users\Administrateur\SAVE


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        06/07/2019 15:37             26 groupes.txt
-a----        06/07/2019 15:35              0 utilisateurs_save.txt
```

Tiens, je souhaite créer un sous-dossier **OU** dans **SAVE**, comment faire ? Je peux utiliser la commande **New-Item** comme ceci :

```
PS C:\Users\Administrateur> New-Item -Path C:\Users\Administrateur\SAVE\OU -ItemType Directory
```

```
PS C:\Users\Administrateur> New-Item -Path C:\Users\Administrateur\SAVE\OU -It


 Répertoire : C:\Users\Administrateur\SAVE


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        17/09/2019 10:45                OU

PS C:\Users\Administrateur> Get-ChildItem C:\Users\Administrateur\SAVE


 Répertoire : C:\Users\Administrateur\SAVE


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        17/09/2019 10:45                OU
-a----        06/07/2019 15:37             26 groupes.txt
-a----        06/07/2019 15:35              0 utilisateurs_save.txt
```

Maintenant, nous allons copier le contenu de notre dossier personnel **SAVE** dans un autre dossier **SAUVEGARDE**, mais qui sera à la racine du disque dur.

Allons-y, créons d'abord un dossier SAUVEGARDE **à la racine du disque dur en tapant :**

```
PS C:\Users\Administrateur> New-Item -Path "C:\SAUVEGARDE" -ItemType Directory


 Répertoire : C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        17/09/2019 10:46                SAUVEGARDE
```

Et voilà :

```
PS C:\Users\Administrateur> Get-ChildItem C:\


 Répertoire : C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        15/09/2018 09:19                PerfLogs
d-r---        06/07/2019 19:27                Program Files
d-----        15/09/2018 18:38                Program Files (x86)
d-----        17/09/2019 10:46                SAUVEGARDE
d-r---        17/09/2019 10:35                Users
d-----        17/09/2019 10:30                Windows
```

Maintenant, nous allons effectuer quelques recherches sur le web afin de découvrir de nombreuses autres fonctionnalités.

Installer des logiciels avec PowerShell

Pour lister les commandes disponibles au sein de ce module, rien de plus simple on saisit dans une console PowerShell :

```
Get-Command -Module PackageManagement
```

Voici les commandes disponibles :

- **Find-Package** : Rechercher un paquet au sein des sources de paquets disponibles
- **Get-Package** : Retourne la liste des tous les paquets installés
- **Get-PackageProvider** : Retourne la liste des fournisseurs de paquets (dépôts) utilisés sur la machine
- **Get-PackageSource** : Retourne la liste des sources disponibles pour récupérer les paquets
- **Install-Package** : Installer un paquet (logiciel) sur la machine
- **Register-PackageSource** : Ajouter une source dans un provider
- **Save-Package** : Sauvegarder un paquet en local sans l'installer
- **Set-PackageSource** : Définir un provider comme source pour les paquets
- **Uninstall-Package** : Désinstaller un paquet (logiciel)
- **Unregister-PackageSource** : Retirer l'utilisation un provider des sources pour les paquets

Pour lister tous les providers disponibles, nous allons utiliser la commande suivante :

```
Find-PackageProvider
```

Ceci nous donne toutes les sources utilisables pour récupérer des paquets en ligne, il est essentiel de commencer par cette étape. On va ajouter « chocolatey » qui contient à ce jour plus de 4 300 paquets.

Pour installer un nouveau provider, utiliser la commande suivante :

```
Install-PackageProvider chocolatey
```

Indiquez "*O*" pour "*Oui*" afin de valider l'installation. Ce qui donnera :

```
PS C:\WINDOWS\system32> Install-PackageProvider chocolatey

Le ou les packages proviennent d'une source de package qui n'est pas marquée comme étant approuvée.
Voulez-vous vraiment installer le logiciel à partir de
'https://oneget.org/ChocolateyPrototype-2.8.5.130.package.swidtag' ?
[O] Oui  [T] Oui pour tout  [N] Non  [U] Non pour tout  [S] Suspendre  [?] Aide (la valeur par défaut est « N ») : o

Name                           Version          Source           Summary
----                           -------          ------           -------
chocolatey                     2.8.5.130        https://onege... ChocolateyPrototype provider for the OneGet meta-pa...


PS C:\WINDOWS\system32> _
```

On va en profiter pour définir Chocolatey comme une source de confiance pour le téléchargement des paquets, ce qui évitera d'avoir une confirmation à l'installation de chaque application :

```
PS C:\WINDOWS\system32> Set-PackageSource -Name chocolatey -Trusted

Name                           ProviderName     IsTrusted  Location
----                           ------------     ---------  --------
chocolatey                     Chocolatey       True       http://chocolatey.org/api/v2/
```

Dès à présent Chocolatey fait partie des sources pour l'installation de logiciels en PowerShell :

```
PS C:\WINDOWS\system32> Get-PackageSource

Name                           ProviderName     IsTrusted  Location
----                           ------------     ---------  --------
PSGallery                      PowerShellGet    False
https://www.powershellgallery.com/api/v2/
chocolatey                     Chocolatey       False      http://chocolatey.org/api/v2/
```

Installer un paquet depuis les dépôts « Chocolatey » »

Tout d'abord, il est intéressant de recherche le paquet au sein de nos dépôts afin de trouver son nom exact. Dans ce tutoriel, je prends pour exemple l'installation d'Adobe Reader ça me donnera l'occasion de l'installer sur ma machine...

Lien utile : https://community.chocolatey.org/packages

On peut directement vérifier l'existence d'un paquet avec la commande "***Find-Package***" mais si vous ne connaissez pas le nom exact, il est préférable d'encadrer votre mot clé par des astérisques * qui permet de rechercher ce terme. Voici une copie d'écran avec des exemples et voici une commande correcte :

```
Find-Package -Name *Adobe* -Source Chocolatey
```

Ce n'est pas obligatoire de préciser la source pour recherche le package, mais ça peut être utile si vous avez plusieurs sources définies sur votre poste et que vous souhaitez en prioriser une.

```
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. Tous droits réservés.

PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32> Find-Package -name AdobeReader

Name                           Version          Source           Summary
----                           -------          ------           -------
adobereader                    2015.007.20033   http://chocol... Adobe Reader - View and interact with PDF files


PS C:\WINDOWS\system32> Find-Package -name Adobe
Find-Package : No match was found for the specified search criteria and package name 'Adobe'.
Au caractère Ligne:1 : 1
+ Find-Package -name Adobe
+ ~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (Microsoft.Power...ets.FindPackage:FindPackage) [Find-Package], Exceptio
   n
    + FullyQualifiedErrorId : NoMatchFoundForCriteria,Microsoft.PowerShell.PackageManagement.Cmdlets.FindPackage

PS C:\WINDOWS\system32> Find-Package -name *Adobe*

Name                           Version          Source           Summary
----                           -------          ------           -------
AdobeAIR                       15.0.0.249       http://chocol... Adobe AIR runtime is necessary for AIR based applic...
adobe-creative-cloud           1.0              http://chocol... Adobe Creative Cloud Client Installer for installin...
adobedigitaleditions           4.0.2            http://chocol... Optimize your reading experience with this free rea...
simnetsa-adobereader-fr        11.0.7           http://chocol... Le logiciel Adobe Reader est la norme international...
adobereader-de                 11.0.10          http://chocol... Adobe Reader - View and interact with PDF files
adobereader                    2015.007.20033   http://chocol... Adobe Reader - View and interact with PDF files
adobeshockwaveplayer           12.1.8.159       http://chocol... Displays Web content that has been created by Adobe...
BR.AdobeReaderFR               11.0.09          http://chocol... BR.AdobeReaderFR


PS C:\WINDOWS\system32> _
```

Pour installer le paquet en s'appuyant sur le fournisseur « Chocolatey » qui héberge le paquet que l'on veut, on utilisera cette commande **:**

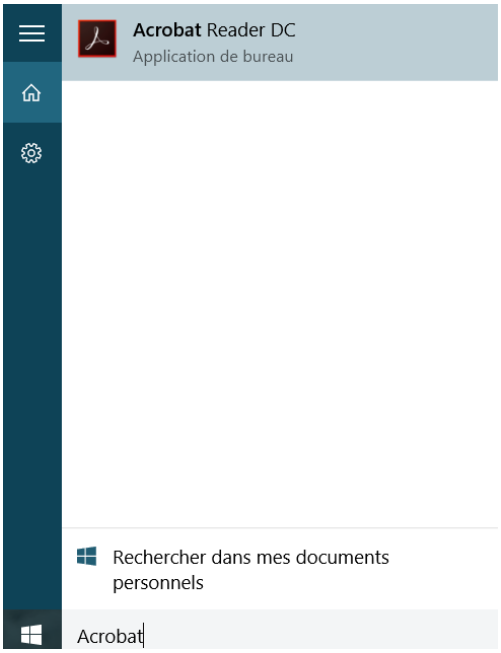
```
Install-Package -Name AdobeReader -ProviderName Chocolatey
```

Si nous n'avions pas approuvé « Chocolatey » comme source sûre, ceci aurait été proposé lors de la première installation. Voyez par vous-même :

```
PS C:\WINDOWS\system32> Install-Package -Name AdobeReader -ProviderName Chocolatey

The provider 'chocolatey v2.8.5.130' is not installed.
chocolatey may be manually downloaded from https://oneget.org/ChocolateyPrototype-2.8.5.130.exe and installed.
Would you like PackageManagement to automatically download and install 'chocolatey' now?
[O] Oui  [N] Non  [S] Suspendre  [?] Aide (la valeur par défaut est « O ») : o

The package(s) come from a package source that is not marked as trusted.
Are you sure you want to install software from 'chocolatey'?
[O] Oui  [T] Oui pour tout  [N] Non  [U] Non pour tout  [S] Suspendre  [?] Aide (la valeur par défaut est « N ») : o

Name                           Version          Source           Summary
----                           -------          ------           -------
adobereader                    2015.007.20033   chocolatey       Adobe Reader - View and interact with PDF files


PS C:\WINDOWS\system32>
```

Dans le haut de votre fenêtre PowerShell, l'avancement quant au téléchargement d'Adobe Reader est indiqué, puis l'application s'installera toute seule en tout transparence.

```
Downloading 'http://ardownload.adobe.com/pub/adobe/reader/win/AcrobatDC/1500720033/AcroRdrDC1500720033_MUI.exe'
   To C:\Users\Florian\AppData\Local\Temp\chocolatey\adobereader\adobereaderinstall.EXE
   [ooooo                                                                                                    ]
```



Après quelques minutes de patience, Acrobat Reader DC est bien installé sur votre machine !

Vous n'avez plus qu'à installer vos logiciels en ligne de commande, maintenant vous n'aurez plus d'excuse 😉