**HTML5**

## Découvrez le fonctionnement des sites web :

* Le Web a été inventé par Tim Berners-Lee au début des années 1990.
* Pour créer des sites web, on utilise deux langages informatiques :
  + HTML : permet d'écrire et organiser le contenu de la page (paragraphes, titres…) ;
  + CSS : permet de mettre en forme la page (couleur, taille…).
* Il y a eu plusieurs versions des langages HTML et CSS. Les dernières versions sont HTML5 et CSS3.
* Le navigateur web est un programme qui permet d'afficher des sites web. Il lit les langages HTML et CSS pour savoir ce qu'il doit afficher.
* Il existe de nombreux navigateurs web différents : Google Chrome, Firefox, Internet Explorer, Safari, Opera… Chacun affiche un site web de manière légèrement différente des autres navigateurs.
* On utilise l'éditeur de texte (Visual-Studio-Code, 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 src="photo.jpg" />).
* Une page web est constituée de deux sections principales :

un en-tête (<head>) et un corps (<body>)

**Comment organisez votre texte « Balises »:**

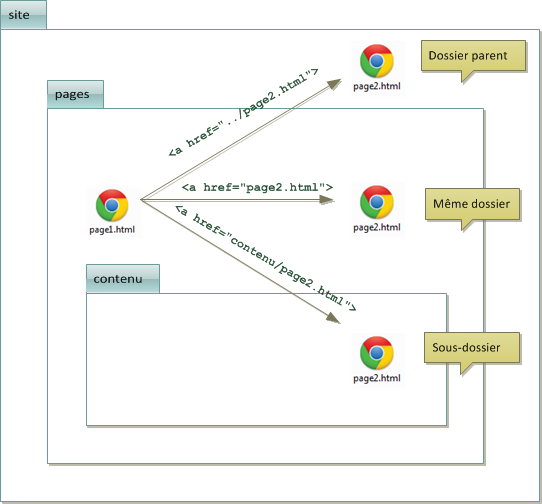
* Le HTML comporte de nombreuses balises qui nous permettent d'organiser le texte de notre page. Ces balises donnent des indications comme « Ceci est un paragraphe », « Ceci est un titre », etc.
* Les paragraphes sont définis par la balise **<p> </p>**  , et les sauts de ligne par la balise **<br />**  .
* 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.

 <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

## Créez et gérer des liens :

* 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">



#### Exmples de Liens :

#### \* Un lien pour envoyer un e-mail :

<p>

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

</p>

#### \* Un lien pour télécharger un fichier :

<p>

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

</p>

#### \* Un lien qui affiche une Infobulle au Survol de la Souris :

<p>Bonjour. Souhaitez-vous visiter

<a href="https://lepoles.org" title="Vous ne le regretterez pas !">Le PôleS</a> ?

</p>

### \* Un lien vers une ancre

<p>

<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>

#### \*Lien vers une ancre située dans une autre page

<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>

## Insérez des images

* 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.
* 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 :

<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>

## CSS.3

## Mettez en place le CSS

* 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 :

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  )

balise1

{

propriete1: valeur1;

propriete2: valeur2;

}

.class

{

propriete1: valeur1;

propriete2: valeur2;

}

#id

{

propriete1: valeur1;

propriete2: valeur2;

}

## Ajoutez de la couleur et un fond

* 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).

## Formatez du texte

* 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  .
* On peut faire en sorte qu'une image soit habillée (« entourée ») de texte avec float  .

#### Utiliser une police personnalisée avec @font-face :

**@font-face { /\* Définition d'une nouvelle police nommée CAC Champagne \*/**

**font-family: 'cac\_champagneregular';**

**src: url('cac\_champagne-webfont.eot');**

**src: url('cac\_champagne-webfont.eot?#iefix') format('embedded-opentype'),**

**url('cac\_champagne-webfont.woff') format('woff'),**

**url('cac\_champagne-webfont.ttf') format('truetype'),**

**url('cac\_champagne-webfont.svg#cac\_champagneregular') format('svg');**

**}**

**h1 /\* Utilisation de la police qu'on vient de définir sur les titres \*/**

**{**

**font-family: 'cac\_champagneregular', Arial, serif;**

**}**

Voici les différents formats de fichiers de polices qui existent et qu'il faut connaître :

* .ttf  : *TrueType Font*. Fonctionne sur IE9 et tous les autres navigateurs ;
* .eot  : *Embedded OpenType*. Fonctionne sur Internet Explorer uniquement, toutes versions. Ce format est propriétaire, produit par Microsoft ;
* .otf  : *OpenType Font*. Ne fonctionne pas sur Internet Explorer ;
* .svg  : *SVG Font*. Le seul format reconnu sur les iPhone et iPad pour le moment ;
* .woff  : *Web Open Font Format*. Nouveau format conçu pour le Web, qui fonctionne sur IE9 et tous les autres navigateurs.

## Ajoutez de la couleur et un fond

* 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).

## Créez des bordures et des ombres

 On peut appliquer une bordure à un élément avec la propriété border  . Il faut indiquer la largeur de la bordure, sa couleur et son type (trait continu, pointillés…).

On peut arrondir les bordures avec border-radius  .

On peut ajouter une ombre aux blocs de texte avec box-shadow  . On doit indiquer le décalage vertical et horizontal de l'ombre, son niveau d'adoucissement et sa couleur.

Le texte peut lui aussi avoir une ombre avec text-shadow

## Créez des apparences dynamiques

* En CSS, on peut modifier l'apparence de certaines sections dynamiquement, après le chargement de la page, lorsque certains évènements se produisent. On utilise pour cela les pseudo-formats.
* Le pseudo-format :hover  permet de changer l'apparence au survol (par exemple : a:hover  pour modifier l'apparence des liens lorsque la souris pointe dessus).
* Le pseudo-format :active  modifie l'apparence des liens au moment du clic, :visited  lorsqu'un lien a déjà été visité.
* Le pseudo-format :focus  permet de modifier l'apparence d'un élément sélectionné.

## le modèle des boîtes

 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

## La mise en page avec Flexbox

* 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.

**Effet Grossir et Magrir avec la propriété FLEX**

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.

**GIT & GITHUB**

## GitHub

## 1 Présentation

### 1.1 GitHub

Plateforme collaborative pour dev, plus grand espace de stockage de travaux collaborative dans le monde! GitHub en lui-meme m'est plus rien de plus q'un reseau social comme fb. Vous construisez un profil, vous y deposez des projets à partager et vous vous connectez avec d'autre utilisateurs en suivant leurs comptes. Meme si la plupart des utilisateurs y deposent des projets de programmes ou de code, rien ne vous empeche d'y placer des textes ou tout type de fichier à présenter dans vos repertoires de projets.

### 1.2 Git

Git est un logiciel de controle de version, ce qui signifie qu'il gere les modifications d'un projet sans écraser n'importe quelle partie du projet.

## 2 Lexique de de Git et GitHub

### 2.1 Vocabulaire

**ligne de commande**: En gros: le programme de l'ordinateur que nous utilisons pour entrer des commandes Git. Dans le monde UNIX, on dit qu’on travaille en “ligne de commande” pour désigner le fait d’interagir avec un système informatique en entrant des lignes d’instructions textuelles dans un terminal, et non à l’aide d’une interface graphique. Les commandes tapées dans le terminal sont interprétées par un Shell. [Doc: Journal du Net](https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1445276-commande-informatique-definition-precise-et-exmples/)

**Depot**: Un repertoire ou de l'espace de stockage ou vos projets peuvent vivre.Les utilisateurs de GitHub raccourcissent ca en "repo" pour "repository". Il peut etre un espace de stockage sur GitHub ou un autre hebergeur en ligne. A l'interieur d'un depot, vous pouvez conserver des fichiers de codes, des fichiers txtdes images.

**Controle de version**: Fondamentalement, l'objectif pour lequel Git a ete concu. Quand vous avez un fichier Word, vous l'écrasez à chaque fois que vous sauvgardez plusieurs versions. Avec Git, vous n'est plus oblige de faire ca. Git conserve des "instantanes" de chaque point dans l'historique d'un projet, de sorte que vous ne pouvez jamais le perdre ou l'ecraser.

**Commit**: C'est la commande qui donne à git toute puissance. Quand vous committez, vous prenez un instantané, une photo de votre depot à ce stade vous donnant un point de controle que vous pouvez ensuite réévaluer ou tout simplement restaurer votre projet à cette version. Il est nécessaire de rappeler que le nom de vos commits doivent-être parlant afin de savoir à quel stade du projet celui-ci a été fait.

**Branche**: Comment plusieurs personnes travaillant sur un projet en meme temps sans que Git ne s'embrouille? Habituellement, elle se debranchent du projet principal avec leur propres versions complètes, des modifications qu'elles ont chacune produites de leur cote. Apres avoir termine, il est temps de fusionner cette branche pour la ramener vers la branche master, le repertoire principal du projet.

### 2.2 Commandes specifiques Git

**git init**: Initialise un nouveau depot git jusqu'a ce que vous executiez les commandes Git qui suivent.

**git config**: Raccourci de configuration, ceci est tout particulièrement utile quand vous paramétrez Git pour la premiere fois indentifiant et mot d'utilisateur.

**git help**: Oublie une commande? pour afficher les 21 commande de git.

**git status**: Verifie le statut de votre repo. Voir les fichiers et quelle sont les modifications à commiter et sur quelle branche ou repo vous etes en train de travailler.

**git add**: Ceci n'ajoute pas des fichiers dans votre repo, Au lieu de cela, cela porte de nouveaux fichiers à l'attestation de Git. Apres avoir ajoute des fichiers, ils dont inclus dans les instantanes du depot Git.

**git commit**: La commande la plus importante de Git. Apres avoir effectue toute sorte de modification vous entrez ca afin de prendreun instantanes du depot. le mettre en memoire. On ecrit ca sous la forme de git commit -m 'votre message'. Le -m indique que la section suivante de la commande devrait etre lu comme in message.

**git branche**: Vous travaillez avec plusieurs collaborateurs et vous voulez produire des modification de votre cote? cette commandevous permet de construire une nouvelle branche, ou une chronologie des commits, des modifications et des ajouts de fichiers qui sont completement les votres. Votre titre va apres la commande. Si vous vouliez creer une nouvelle branche appellee 'chats', vous saisiriez git branche chats.

**git checkout**: C'est une commande de navigation qui vous permet de vous deplacer vers le repertoire que vous voulez verifier. Vouspouvez utiliser cette commande sous la forme de git checkout master pour regarder la branche master, ou git checkout chats pour regarder une autre branche.

**git merge**: Lorsque vous avez fini de travailler sur une branche, vous pouvez fusionner vos modifications vers la branche master,qui est visible pour tous les collaborateur. git merge chats prendrait toutes les modifications que vous avez apportées à la branche chats et les ajoutera à la branche master.

**git push**: Si vous travaillez sur votre ordinateur en local et voud voulez que vos commits soient visibles aussi en ligne surGithub, vous pushez les modifications vers GitHub avec cette commande.

**git pull**: Si vous travaillez en local et que vous voulez la version la plus a jour de votre repo pour travailler dessus, vous pullez les modifications provenant de GitHub avec cette commande.

**git clone**: Permet de dupliquer, cloer un repo existant sur GitHub pour l'avoir en local. La commande git clone doit etre suivit de url de repo correspondant qui copie depuis GitHub direct.

## 3 Fonctionnement GIT

### 3.1 branch master

Cette branche est la branche principale du projet, c'est sur cette branche que les différentes parties vont être merge. Avant de merge sur celle-ci **vérifiez que votre code fonctionne** et **commentez** on ne le dira jamais trop mais dans le travail collaboratif il faut communiquer. Si il y a un conflit c'est que l'assimilation du code de la branch master et l'autre (ou les autres) ne s'est pas effectuée correctement. En cas de conflit il faut vérifier que le code rajouté à bien été pris en compte et que d'autres n'ont pas étées supprimées.

### 3.2 branch dev / test ...

Une branch commune est souvent ajoutée pour les mutualisations de code afin de protége la branch master. On préfèrera pull les modifications des autres branchs.

### 3.3 branch nomDeBranch

Sur cette branch vous metterez à jour vos avancées dans la programmation. Git add . et son commit sont différents de la sauvegarde sur l'ordinateur, pas besoin de commit à chaque sauvegarde en local. **Il ne faut pas confondre ou assimiler le commit et le push!** On doit commit pour push mais chaque commit n'implique pas un push.

### 3.4 Commit

Avant de Commit tu dois entrer git add . , cette commande récupèrera les changements depuis le dernier commit et seront utilisés pour le commit qui suit. git commit -m "tonblabla" sert à sauvegarder tes changements pour ton **versionning**, ainsi en plus d'avoir une sauvegarde dans ton explorateur de fichier (ne gardant que la dernière version), tu auras également une trace de tes changements et te permettera de faire machine arrière si quelque chose se passe mal.

### 3.5 Push

Push sur ta branch quand tu as ajouté une fonctionnalité qui marche, afin qu'on puisse récupérer le travaille fait et voir si c'est compatible. Ce n'est que sur la fin du projet ou que quelqu'un à finit de travailler sur le projet qu'il fera un merge / push sur la branch dev / test. Avant de **push sur une autre branch que la vôtre**, vérifiez que vous ètes à jour sur la destination du push à l'aide de git log et si besoin pullez, attention aux conflits.

### 3.6 Pull

Avant de taper git pull origin nomDeBanch, il faut **obligatoirement** que l'autre branch ai push pour pouvoir récupérer les changements. Il est **recommandé** de push sur notre branch avant de pull le contenu d'une autre pour réduire les risques de **conflits**, mais il est **encore plus fortement conseillé** de faire au moins un commit pour avoir une sauvegarde en cas de conflit.

### 3.7 Conflits

Les conflits sont à Git ce que les bugs sont aux développeurs, fréquents... Pour les éviter au maximum il faut éviter de modifier le même fichier sur deux branch différente. Essayez de pull toutes les modifications apportées à ce même fichier de toute les branchs où il aurait été modifier **avant** de commencer à le modifier de votre côté. [Doc: Atlassian](https://www.atlassian.com/fr/git/tutorials/using-branches/merge-conflicts)

## Documentation

[Blog: Synbioz](https://www.synbioz.com/blog/tech/git-adopter-un-modele-de-versionnement-efficace)  
[Forum: Openclassrooms](https://openclassrooms.com/forum/sujet/git-pull-sans-commit-49970#message-7109921)