HTML5
CSS3

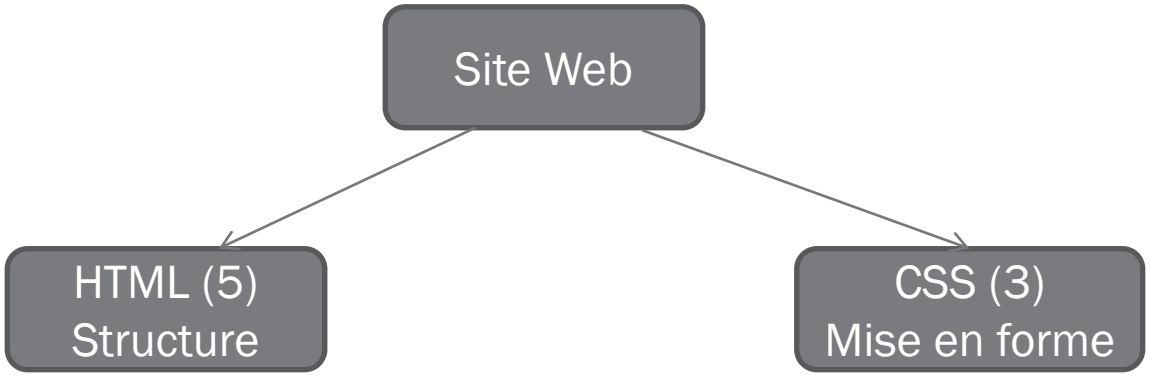
1ÈRE BAC EN INFORMATIQUE

THEORIE

THÉORIE - INTRODUCTION

Site Web Statique	Site Web Dynamique
HTML(5) – CSS (3)	HTML(5) – CSS(3) – PHP – Javascript – SGBD SQL
Coté Client	Coté Serveur
Ajout d'informations dans le site nécessite de modifier le code	Ajout d'informations dans le site ne nécessite pas de modifier le code
Pas très complexe -> vous devriez y arriver seul	Plus complexe -> on paye des développeurs pour cela
Les formulaires HTML feront le lien entre le statique et dynamique	

THÉORIE - INTRODUCTION



Séparation entre le contenu et présentation. Pq ?

THÉORIE - LOGICIELS

- **Logiciels utilisés**
  - Un éditeur (Notepad++ par ex.)
  - Un navigateur (Firefox par ex.)

Rmq : nous utiliserons l'encodage UTF-8, utilisez un éditeur où vous pouvez facilement contrôler l'encodage que vous voulez utiliser

THÉORIE – HTML5

```
<!DOCTYPE html>
<html lang="fr" >
<head>
        <meta charset="UTF-8">
        <title>Le titre de ma page </title>
</head>
<body>
        J'écris ici le contenu de ma page...
</body>
</html>
```

Document HTML5 (2 lignes)

Méta-informations Non affichées

Encodage UTF-8

Langage balisé. Ces balises sont appelées des éléments HTML

THÉORIE – ÉLÉMENTS SÉMANTIQUES

- **Ont une valeur qui leur est associée. En HTML5, les éléments sémantiques ont été renforcés.**
- **La règle en HTML5 est d'utiliser le plus possible ces éléments sémantiques quand cela est adéquat.**
  - Ex : <header>

Quel est l'intérêt d'avoir des éléments sémantiques ?

- Code plus facile à lire, à maintenir
- Meilleur référencement
- Accessibilité (personnes handicapées)

THÉORIE – PRINCIPAUX ÉLÉMENTS HTML

- **En-tête / pied de page**
  - <header> … </header>
  - <footer> … </footer>
- **Navigation**
  - <nav> …. </nav>
- **Titres**
  - <h1> …. </h1>
  - <h2> … </h2>
  - ….
- **Divisions de texte**
  - <section> …. </section>
  - <article> …. </article>
  - …. ---------- > non sémantique
  - …

THÉORIE – PRINCIPAUX ÉLÉMENTS HTML

- **Liens**
  - …
- **Images**
  - <img src=« … » alt=« … »> …. </img>
- **Tableaux**
  - <table> <tr> <td> …. </td> </tr></table>
- **Formulaires**
  - *<form action* = "URL/script" *method* = "get">
    <*input* type = "checkbox" name = "acceptation"/>
    <*input* type = "submit" value = "soumettre" />
    </form>

THÉORIE – FORMULAIRES

- Internet est une architecture Client / Serveur.
- On parle de Serveur Web et de client HTTP (navigateur)
- Protocole utilisé pour les échanges : HTTP
- Méthodes proposées par ce protocole : GET / POST
- Un formulaire doit définir quelle méthode il emploie
  - GET : les variables sont placées dans l'URL et donc visibles
    - Ex : index.php?param1=1&param2=«oli»
  - POST : les variables sont placées dans la requête et sont donc invisibles
    - Ex : index.php

THÉORIE – GET /POST

- GET
  - Obtenir de l'information
  - Idempotent
  - Paramètres visibles (URL)
- POST
  - Poster de l'information (enregistrer des informations côté serveur)
  - Pas idempotent
  - Paramètres invisibles (sécurité)

THÉORIE – GET /POST

- **Quand utiliser GET ? Un exemple concret ?**
  - Consultation d'un catalogue de produits
  - Possibilité d'utiliser des marques-pages / retrouver facilement une page
- **Quand utiliser POST ? Un exemple concret ?**
  - Formulaire avec mot de passe
  - Upload fichiers

THÉORIE – VALIDATION HTML !!!

- **W3C**
  - Consortium
  - Recommandations - Norme
- **Validation HTML**
  - https://validator.w3.org

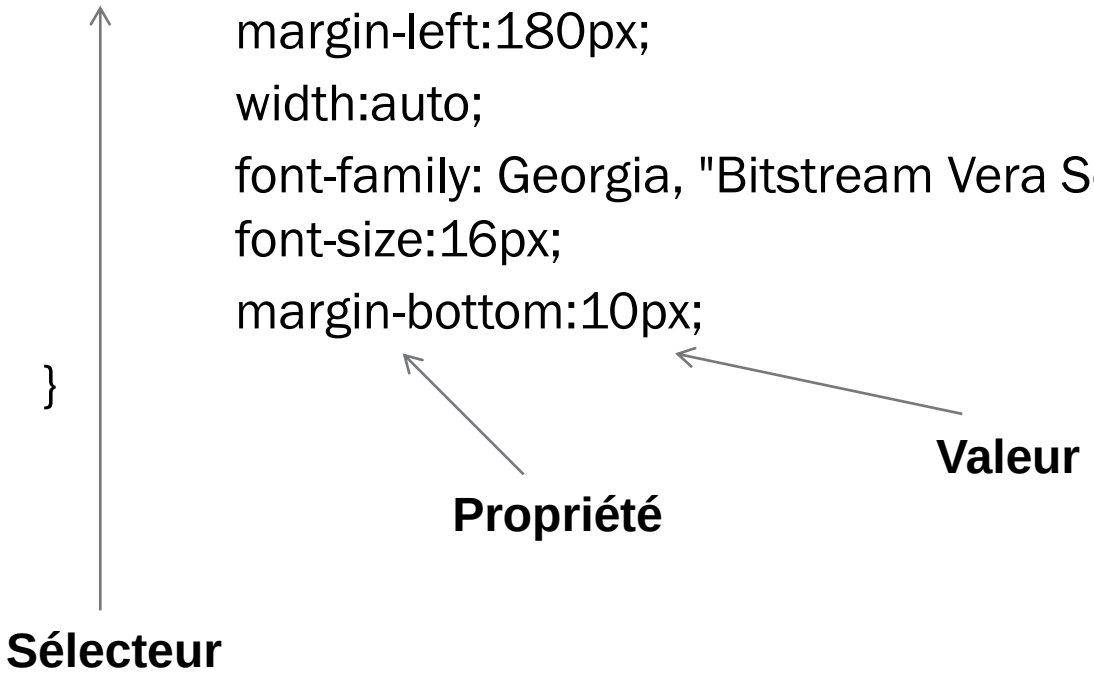
Rmq : n'oubliez pas de valider vos pages HTML !

THÉORIE – CSS

- Création d'un répertoire styles
- Utilisation d'une feuille externe (Ex: style.css)
- Lié la page HTML à une feuille de style (CSS)
  - à placer à l'intérieur de la balise <head> </head>
  - <link rel="stylesheet" type="text/css" href="styles/style.css" />

THÉORIE – CSS EXEMPLE

```
article {
        margin-left:180px;
        width:auto;
        font-family: Georgia, "Bitstream Vera Serif", Norasi, serif;
        font-size:16px;
        margin-bottom:10px;
}
```



THÉORIE – CSS SÉLECTEURS

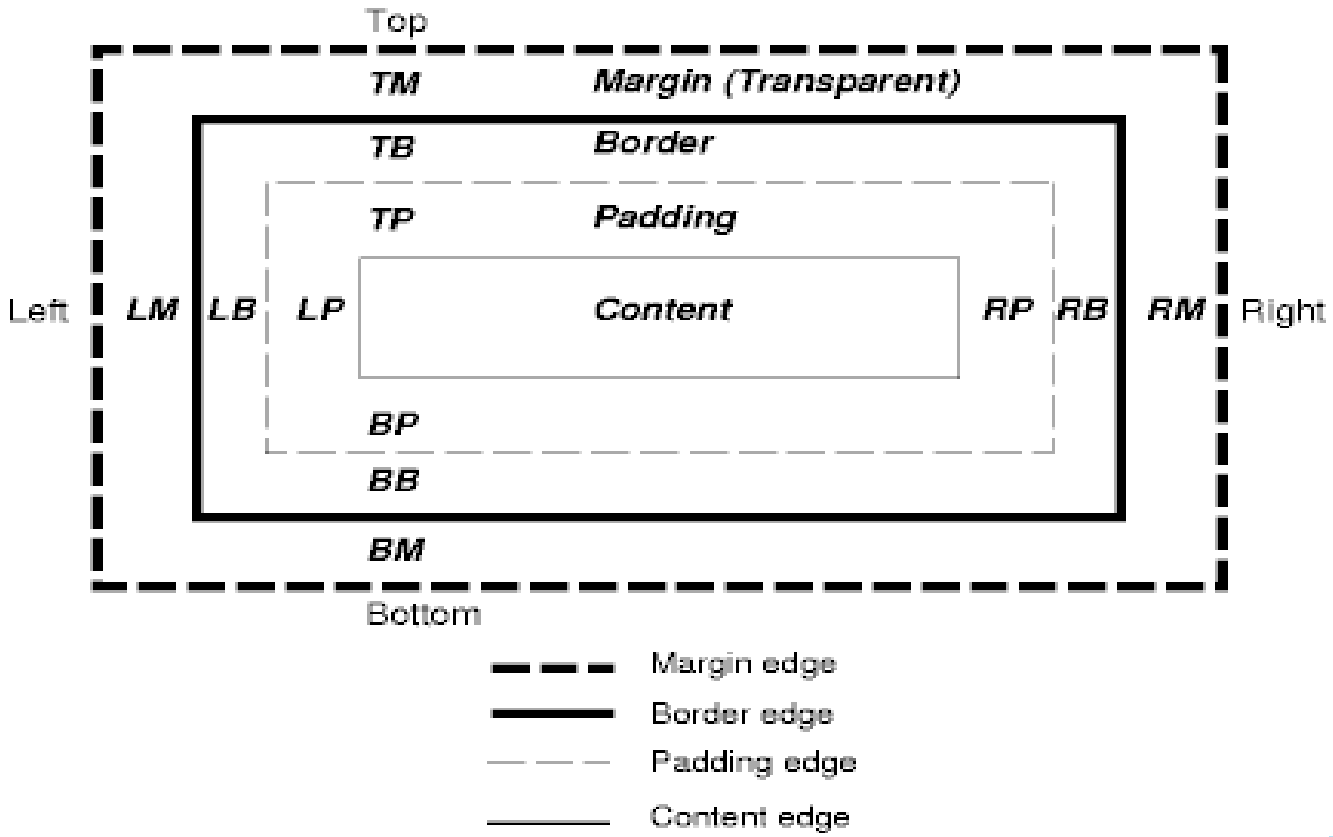
- **Sélecteur simple**
  - Ex: p , section, nav , li
  - Ex: div p (tous les elts p à l'intérieur des div)
- **Sélection universelle**
  - Ex : *
- **Sélection par id**
  - Ex : #logo
  - <header>
- **Sélection par classe**
  - Ex : .souligne
  - super paragraphe

Quand utiliser un id et quand utiliser une classe ?

THÉORIE – CSS FORMATAGE

- **Texte**
  - Ex: p {
    color:#DCDCDC;
    font-size : 12px;
    font-family: "Bitstream Vera Mono« ;
    }
- **Liens**
  - Ex: a:hover { color :orange; }

THÉORIE – CSS FORMATAGE BOÎTES



THÉORIE – CSS POSITIONNEMENT

1. **Positionnement normal - Flux**
   - Les éléments sont placés par le navigateur suivant leur ordre d'apparition et suivant les propriétés CSS définies
2. **En ligne - En bloc**
   - Chaque élément a un rendu par défaut en ligne ou en bloc
   - Ce comportement par défaut peut être changé via les feuilles CSS
   - En ligne : les éléments se placent les uns à coté des autres
   - En bloc : les éléments se placent les uns en dessous des autres

THÉORIE – CSS POSITIONNEMENT

3. **Positionnement inline-block**
   - Positionner les éléments les uns à côtés des autres (inline) mais avec des dimensions (block)

4. **Positionnement flex**
   - Le plus simple
   - Flex- OpenClassroom
   - RECOMMANDE !!!!

THÉORIE – CSS POSITIONNEMENT

- Positionnement relatif (par rapport aux autres éléments)
  - Permet de décaler le positionnement d'un élément de son positionnement normal.
  - Ex: décaler un texte
- Positionnement absolu (par rapport aux côtés du document, aux limites du conteneur)
  - Permet de placer un élément n'importe où sur la page
  - Ex: mise en page 3 colonnes
- Positionnement fixe (par rapport au coin supérieur gauche du navigateur)
  - Idem que positionnement absolu mais reste visible
  - Ex : menu toujours visible
- Positionnement flottant
  - Permet de faire « flotter » un élément
  - Ex: Image entourée de texte
- Positionnement INLINE-BLOCK / FLEX
  - Permet de réaliser assez facilement un design de site (bannière, menu, corps de page).

THÉORIE – CSS CASCADE ET HÉRITAGE

- **Imbrication des éléments HTML**
- **Les éléments enfants héritent des propriétés CSS définies au niveau de leurs parents.**
  - Attention toutes les propriétés ne sont pas héritées !!! (Ex: dimensions, marges, …)
- **Possibilité de changer ce comportement en ajoutant une règle spécifique**
- **Plusieurs feuilles de styles appliquées en cascade**
  - Navigateur -> auteur -> utilisateur
- **Debugging**
  - Permet de voir les différentes propriétés CSS appliquées sur un élément HTML