

Introduction HTML-CSS

FORMATEUR : OLIVIER POUSSEL

A solid blue horizontal bar spanning the width of the slide, located at the bottom.

SOMMAIRE

1^{ère} PARTIE - LE WEB

- Le Web.

2^{ème} PARTIE - HTML & CSS

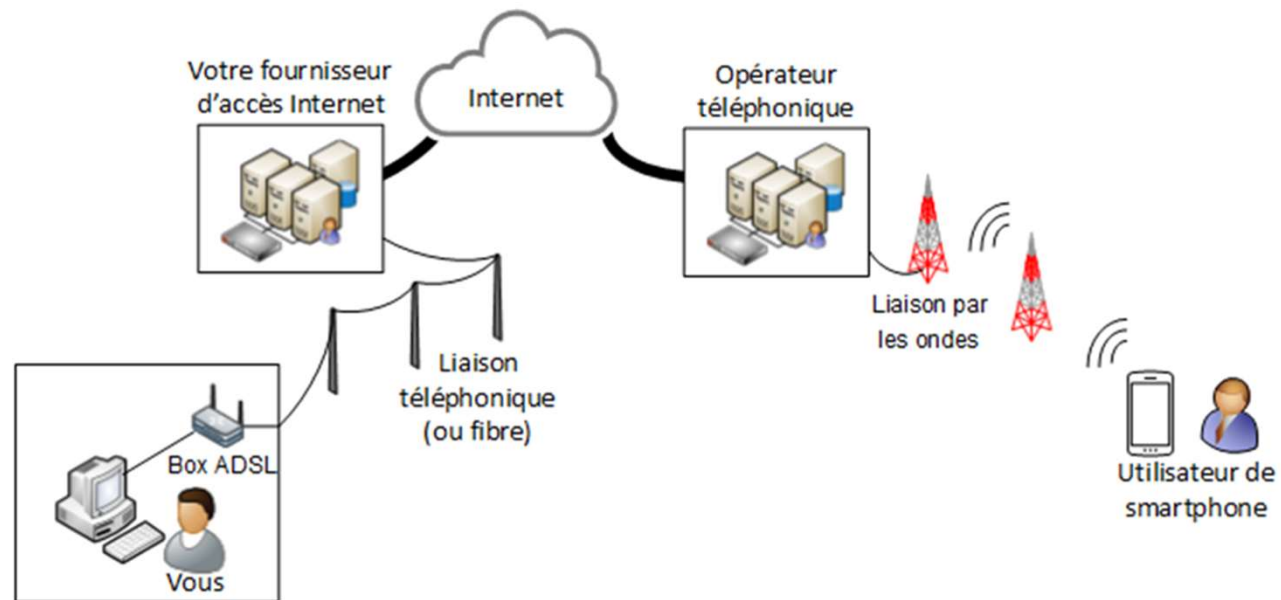
- Présentation et différence entre HTML & CSS ;
- Exemple d'un site sans et avec du CSS ;
- Synthèse des différences entre HTML & CSS ;
- Présentation du HTML ;
- HTML - Balises et attributs ;
- HTML - Structure d'une page ;
- HTML - EXERCICE 1 ;
- Focus sur quelques notions HTML - 1 ;
- Présentation du CSS ;
- OOCSS VS Interface-Oriented ;
- Liaison entre HTML et CSS et syntaxe du CSS ;
- CSS - EXERCICE 2 ;
- Focus sur quelques notions HTML - 2 ;
- Les propriétés width et height ;
- CSS - La notion de « boîtes » ;
- CSS - La notion de « boîtes » padding, border, margin ;
- CSS - La notion de « boîtes » display, position, float ;
- CSS - Unité de mesure ;
- CSS - Police d'écriture ;
- HTML & CSS - EXERCICE 3 ;
- HTML & CSS - Formulaire ;
- HTML & CSS - EXERCICE 4 - TP Formulaire ;
- HTML & CSS – Div & span ;
- HTML & CSS – Balises sémantiques ;
- HTML & CSS – Flexbox ;
- HTML & CSS - EXERCICE 5 - TP Liste d'articles de blog ;
- HTML & CSS – Media queries ;
- HTML & CSS - EXERCICE 6 - TP Media queries ;
- HTML & CSS – Css Grid ;
- HTML & CSS – Tableaux HTML ;
- HTML & CSS – Bootstrap ;
- HTML & CSS – TP Bootstrap ;
- HTML & CSS – lib & framework.

1^{ÈRE} PARTIE

LE WEB

Le Web

Comment le Web fonctionne-t-il ?



Le Web

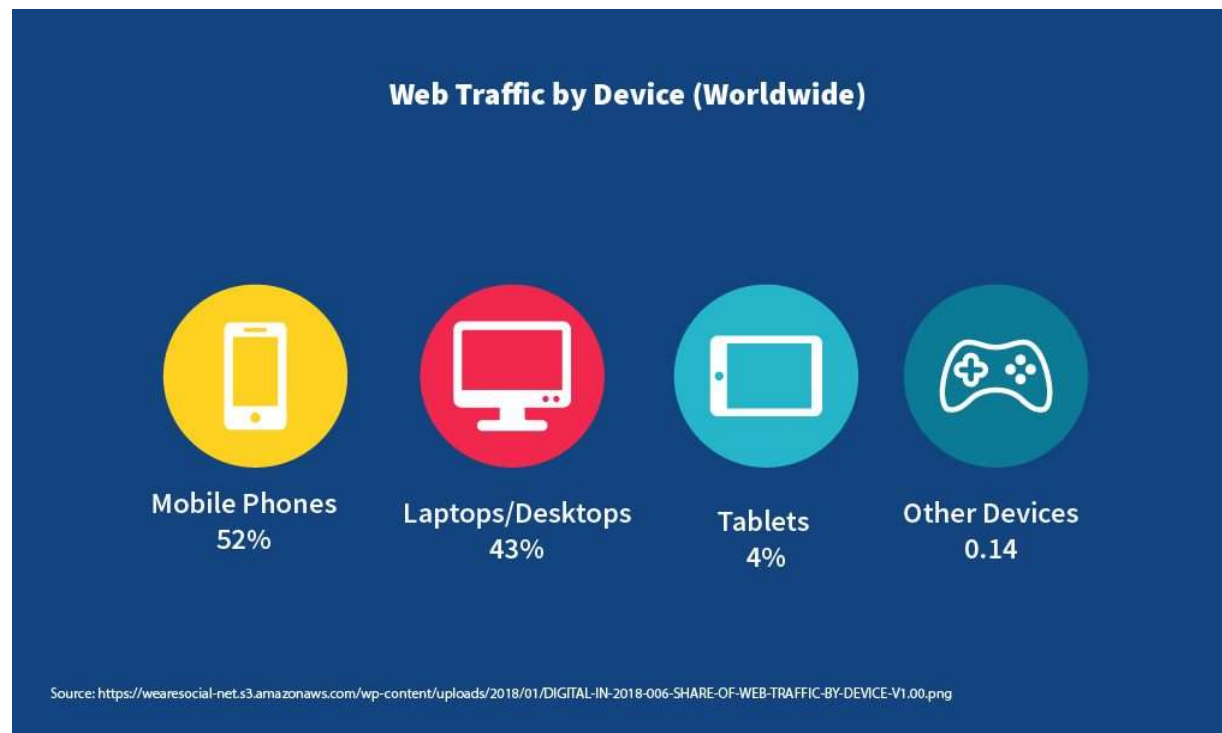
Les notions de Client-Serveur :



Langages côté Client :
HTML, CSS, JavaScript

Langages côté Serveur :
PHP, C# .net, Java JEE, etc.
Gestion base de données :
SQL, NoSQL, etc.

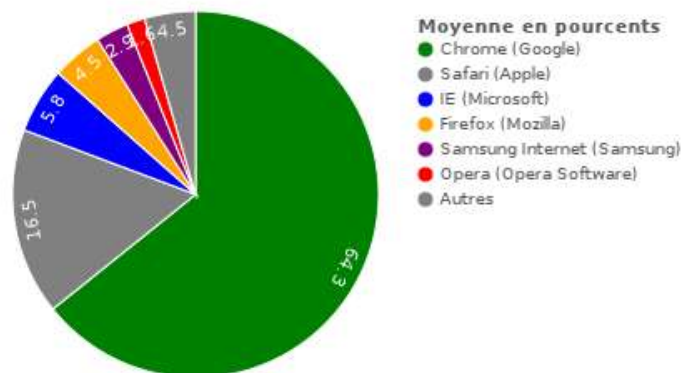
Le Web



Le Web

Les parts de marché des navigateurs Web dans le monde, toutes plateformes confondues
(juin 2020 - mettre à jour)

Source ↕	Chrome (Google) ↕	Safari (Apple) ↕	Firefox (Mozilla) ↕	IE + Edge (Microsoft) ↕	Opera (Opera Software) ↕	Samsung Internet (Samsung) ↕	UC Browser (UCWeb) ↕	Autres ↕
StatCounter ¹	65,5 %	17,0 %	4,3 %	4,2 %	1,9 %	3,3 %	1,8 %	2,1 %
NetMarketshare ²	64,4 %	18,3 %	4,2 %	5,1 %	1,0 %	2,8 %	0,6 %	3,7 %
W3Counter ³	63,0 %	14,4 %	5,1 %	8,0 %	1,8 %	2,5 %	NC	5,2 %
Moyenne	64,3 %	16,5 %	4,5 %	5,8 %	1,6 %	2,9 %	0,8 %	3,7 %



2^{ÈME} PARTIE

HTML & CSS

Présentation et différence entre HTML & CSS

HTML (HyperText Markup Language) est un **langage de description/balisage uniquement** avec une sémantique forte :

- Balise de description de texte ;
- Balise de média ;
- Balise de méta data.

CSS (Cascading Style Sheets) est un langage de **mise en forme uniquement** :

- Sélecteurs ;
- Propriétés ;
- Couleurs.

Exemple d'un site sans et avec du CSS (1/2)

Le site Amazon sans CSS :

Choisir vos préférences en matière de cookies

Nous utilisons des cookies et des outils similaires pour faciliter vos achats, fournir nos services, pour comprendre comment les clients utilisent nos services afin de pouvoir apporter des améliorations, et pour présenter des annonces. Des tiers approuvés ont également recours à ces outils dans le cadre de notre affichage d'annonces.

Désolé, un problème s'est produit lors de l'enregistrement de vos préférences en matière de cookies. Veuillez réessayer.

☐ all | Accepter les cookies [Personnaliser les cookies](#)

Passer au contenu principal

[fr](#)

[Bonjour](#), [Identifiez-vous](#)

[Compte et listes](#) [Compte Retours et Commandes](#)

[Testez](#)

[Prime](#) [Panier 0](#)

Toutes nos catégories

Toutes nos catégories

Go

Rechercher

[Identifiez-vous](#)

Nouveau client ? [Commencer ici](#)

Vos listes d'envies

[Créer une liste](#) [Trouver une liste](#) [Liste d'envies universelle](#) [Liste de mariage](#) [Liste de naissance](#) [Liste de voeux enfants](#) [Découvrez votre style](#) [Explorez Showroom](#)

Votre compte

[Votre compte](#) [Vos commandes](#) [Votre liste d'envies](#) [Vos recommandations](#) [Vos animaux de compagnie](#) [Gérer vos abonnements](#) [Adhésions et abonnements](#) [Votre compte Amazon Prime](#) [Devenez client Amazon Business](#) [Gérer votre contenu et vos appareils](#) [Votre Abonnement Kindle](#) [Votre](#)

[bibliothèque musicale](#) [Votre Prime Video](#) [Votre Amazon Drive](#) [Votre Bibliothèque de jeux et logiciels](#) [Vos applis et vos appareils](#)

Bonjour Entrez votre adresse

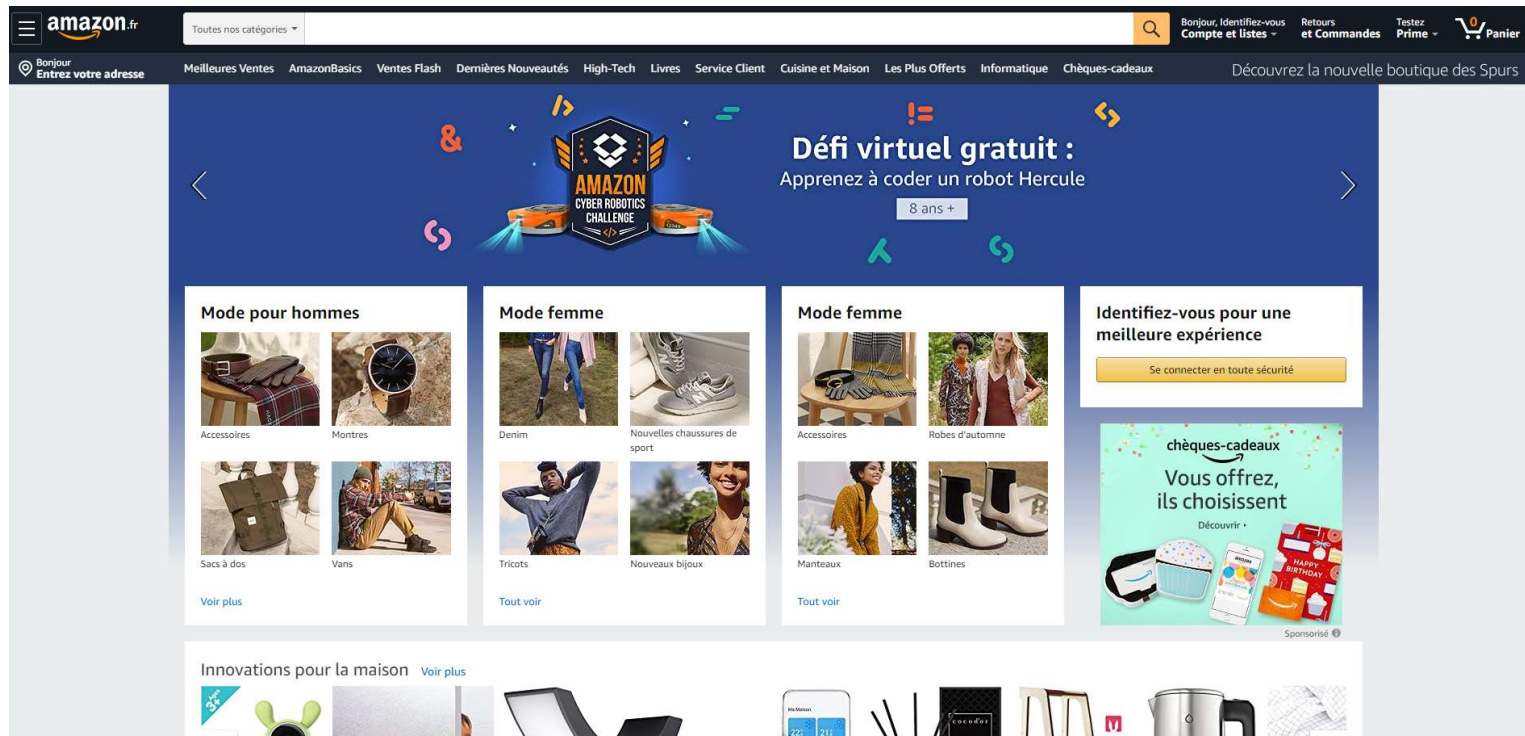
Asus : jusqu'à -27%

[Meilleures Ventes](#) [AmazonBasics](#) [Ventes Flash](#) [Dernières Nouveautés](#) [High-Tech](#) [Livres](#) [Service Client](#) [Cuisine et Maison](#) [Les Plus Offerts](#) [Informatique](#) [Chèques-cadeaux](#) [Vendre](#) [Guide de l'acheteur](#) [Livraison Gratuite](#) [Coupons](#) [Support Client](#) [Accessibilité](#)

[Previous page](#)

Exemple d'un site sans et avec du CSS (2/2)

Le site Amazon avec du CSS :



Synthèse des différences entre HTML & CSS

En résumé :

- HTML et CSS sont différents mais complémentaires ;
- Le **HTML** sert à structurer (quoi et où) les documents avec des **balises** (exemples : mettre une image au-dessus d'un texte, mettre le menu de navigation au-dessus du contenu du site, etc.). On peut comparer cela à la fondation/les murs/la charpente/le toit ;
- Le **CSS** sert à embellir, à mettre en forme les documents en **agissant directement sur les balises HTML** (exemple : faire un menu en couleur, changer police/taille/couleur du texte, mettre en forme un formulaire, créer des jolis boutons, etc.). On peut comparer cela à la peinture/la décoration.

Présentation du HTML (1/2)

HTML (HyperText Markup Language) n'est pas un langage de programmation : c'est un **langage de balisage** qui sert à indiquer au navigateur comment structurer les pages Web.

Langage de description/balisage	Langage de programmation
Langage qui utilise des balises pour définir des éléments dans un document. Ce langage est conçu pour créer une structure, identifier des données ou présenter des données.	Langage formel contenant un ensemble de commandes et une syntaxe permettant de créer des programmes logiciels. Ces programmes peuvent effectuer une tâche spécifique.
Interprété par le navigateur.	Compilé par un compilateur ou interprété par un interpréteur.
HTML, XML, etc.	Java, C#, PHP, Python, JavaScript, etc.
L'ordinateur comprend ou affiche les informations.	L'ordinateur exécute des instructions.

Présentation du HTML (2/2)

- Texte/données :

Mon chat est très grincheux

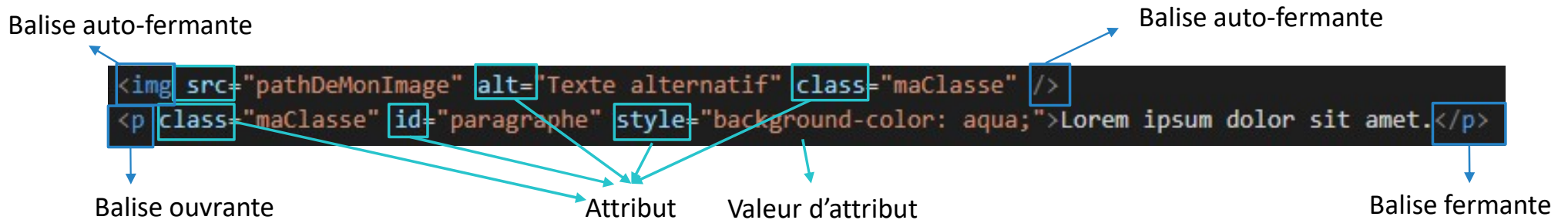
- Texte/données + balise `<p>` :

`<p>`Mon chat est très grincheux`</p>`



HTML - Balises et attributs

- Il existe 2 types de syntaxe de balises :
 - `<balise> </balise>` : balise ouvrante et balise fermante - **contenu entre les 2 balises** ;
 - `<balise/>` : balise auto-fermante - **pas de contenu**.
- On peut ajouter aux balises des **attributs** qui sont comme des options aux balises. Les attributs viennent ajouter des comportements/caractéristiques supplémentaires aux balises. À l'intérieur de ces balises, on affecte une **valeur d'attribut** :



HTML - Structure d'une page

- `<!DOCTYPE html>` : indique au navigateur de quel type de document il s'agit pour bien le lire comme avec xml ;
- `<html></html>` : c'est la racine du document qu'on appelle le DOM (Document Object Model) ;
- `<head></head>` : il s'agit de la tête du document contenant des informations non visibles dans la fenêtre mais utiles pour décrire le document comme le titre (différent du titre dans la balise `<body>`).
- `<body></body>` : il s'agit du corps accueillant le contenu visible de la page (textes, images, liens, vidéos, etc.).

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Titre Document</title>
</head>
<body>
  <h1>Titre de la page</h1>
  <p>paragraphe</p>
  <p>Lorem ipsum dolor sit <em>amet</em> consectetur adipisicing elit.
    <strong>Magni</strong> veritatis atque facere consectetur modi eligendi,
    delectus repellendus. Explicabo, sunt!
  </p>
</body>
</html>
```


HTML - EXERCICE 1

Créer sa première page web :

- Créer un fichier `index.html` ;
- Dans ce fichier, ajouter les éléments suivants :
 - Le `<!DOCTYPE html>` ;
 - Dans la balise `<body>`, ajouter :
 - Un titre `<h1>` ;
 - Une image avec `` ;
 - Un paragraphe avec `<p>` qui doit contenir :
 - ❖ Du texte ;
 - ❖ Un lien `<a>`.

Focus sur quelques notions HTML - 1

- Les titres ;
- Le lien `<a>` ;
- L'imbrication ;
- Générateur de contenu (lorem ipsum) ;
- Indentation et commentaires.

LES TITRES :

Il existe **6 hiérarchies de titre** : `<h1>` étant la plus importante et `<h6>` la moins importante. La taille des titres diffère aussi (corrélée avec l'importance du titre). L'utilisation de ces balises de titre doit se faire dans une logique de hiérarchisation du contenu de la page. Par exemple, on ne peut pas utiliser un `<h3>` avant un `<h1>`, et surtout on ne doit pas utiliser une balise style en fonction de sa taille. **Attention**, `<h1>` ne doit pas être confondue avec `<title>` qui lui se trouve dans le `<head>`.

LE LIEN `<a>` :

Le lien se fait avec la balise et l'attribut suivants :

```
<a href="#">Lien cliquable</a>
```

L'attribut `href` reçoit la cible du lien sous la forme d'une URL. Le `#` peut être utilisé en référence d'un champ vide pour créer un retour vers le haut de la page.

L'IMBRICATION :

Il est possible d'imbriquer des balises entre elles. Par exemple, il est possible de rendre cliquable une image :

```
<a href="#">
  
</a>
```

GÉNÉRATEUR DE CONTENU :

Pour créer du texte/contenu/paragraphe, il est possible d'utiliser un générateur de contenu de *lorem ipsum*. Il existe sur internet plusieurs générateurs de *lorem ipsum*.

INDENTATION ET COMMENTAIRES :

L'éditeur de code (IDE) ne prend pas en compte les espaces. Cependant, dans les bonnes pratiques il faut indenter (retrait en début de ligne) son code pour le rendre lisible et propre et de repérer les différents éléments. Il est possible aussi de faire des commentaires dans son code :

- HTML :

```
<!-- Mon commentaire en HTML -->
```

```
<a href="#">
```

- CSS :

```
/* Mon commentaire en CSS */
```

```
✓ #main {
  border: 1px solid black;
```

Présentation du CSS

Le **CSS** (Cascading Style Sheets) permet de **styler/mettre en forme les pages web** (contenu HTML). On traduit CSS par **feuilles de styles** en français. On dit que ces feuilles de styles sont en cascade car le style qui va être appliqué à un bloc/élément HTML (`<p>`, `<a>`, `<div>`, `<header>`, etc.) va se répercuter en cascade à tous les éléments HTML associés. Par exemple, si on applique un style sur la balise `<p>`, tous les balises `<p>` présentes dans le contenu HTML (fichier **.html**) vont avoir les mêmes propriétés de style.

Le CSS est donc un langage de mise en forme qui peut être utilisé de 2 manières :

- Soit directement dans le fichier **.html** avec la balise `<style>` ;
- Soit en créant un fichier **.css** qui va contenir l'ensemble des styles.

En bonnes pratiques, il est **faut créer un fichier .css** pour séparer la forme (CSS) de la structure HTML, et cela pour 2 raisons :

- Cela évite de réécrire le style à plusieurs endroits avec la balise `<style>`. Avec le fichier **.css**, il suffit de cibler quel élément HTML on souhaite appliquer un style et l'ensemble des éléments HTML associés se voient affecter le même style ;
- Le fichier **.css** peut être utilisé pour plusieurs pages HTML.

Liaison entre HTML et CSS et syntaxe du CSS

LIAISON HTML CSS

Pour affecter du style sur du contenu HTML, il va falloir lier le fichier CSS au fichier HTML afin qu'ils puissent communiquer.

Pour cela on ajoute la balise auto-fermante `<link />` avec les attributs suivants :

```
<link rel="stylesheet" type="text/css" href="style.css" media="screen" />
```

nomDuFichier.css. Dans l'attribut `href`, on indique l'adresse du fichier `.css` (où le fichier se trouve par rapport à `index.html`);

Pour en savoir plus sur les attributs et leurs significations : <https://developer.mozilla.org/fr/docs/Web/HTML/Element/link>

SYNTAXE CSS

La syntaxe CSS suit la logique suivante :

```
sélecteur {  
  propriété: valeur;  
}
```

Les sélecteurs sont de **3 types** :

	HTML	CSS
balise	<pre><p>Lorem, ipsum dolor.</p></pre>	<pre>p { color: blue; font-size: 16px; }</pre>
class .	<pre><p class="text">Lorem, ipsum dolor.</p></pre> <p>Peut apparaître plusieurs fois</p>	<pre>.text { color: green; background-color: aquamarine; }</pre>
id #	<pre><p id="main">Lorem, ipsum dolor.</p></pre> <p>N'apparaît qu'1 fois par page HTML.</p>	<pre>#main { border: 1px solid black; }</pre>

CSS – EXERCICE 2

- **Entrainement au sélecteur :**
 - Aller sur le site CSS diner : <https://flukeout.github.io/> ;
 - Faire les 10 premiers exercices.

- **Reprendre l'exercice 1 pour effectuer les actions suivantes :**
 - Créer un fichier **styles.css** et le relier au fichier **index.html** ;
 - Aller chercher sur internet une image au choix et l'enregistrer dans le répertoire à la racine ;
 - Imbriquer cette image dans le lien **<a>** déjà présent dans le fichier **index.html** -> Trouver le moyen de faire afficher l'image (aide à la recherche internet : type de chemin html). Le lien doit cibler vers un site existant au choix.
 - Appliquer les styles suivants - à chercher dans la documentation (internet) :
 - **<h1>** : taille de police : 50px / couleur de fond : noir / couleur texte : blanc ;
 - **** : bordure noir en trait plein de 2px ;
 - **<p>** : couleur du texte : bleu / taille de police : 32px / famille de police : sans-serif ;
 - L'image imbriquée dans le lien **<a>** : la taille de l'image doit valoir 20%.

Focus sur quelques notions HTML - 2

- Les chemins absolu et relatif ;
- La bordure ;
- Les familles de police en CSS.

LES CHEMINS ABSOLU ET RELATIF :

Pour afficher des images ou des pages il existe 2 types de chemins à connaître :

- **Le chemin relatif** : il fait référence à l'emplacement relatif au répertoire courant (on code le chemin à partir de là où on est). Ce chemin s'exprime de 2 manières :
 - `./` : Le point (.) correspond au répertoire courant ;
 - `../` : Les 2 points (..) correspondent au répertoire parent (on sort du dossier pour aller dans le répertoire parent).
- **Le chemin absolu** : c'est le chemin propre du lecteur (disque). Exemple `D:\developpement\cours`

LA BORDURE

Il est possible de mettre une bordure autour d'un élément avec la propriété **border**. On peut déterminer sa taille, sa couleur et le type de trait (solid, dashed, etc.). Exemple :

```
border: 1px solid black;
```

LES FAMILLES DE POLICE EN CSS

Il est bien sûr possible d'utiliser des polices comme Arial, Calibri, Verdana, Garamond, etc. avec la propriété **font-family** (cf. plus loin dans le cours). Cependant, il faut savoir que le CSS a des polices intégrées par défaut et qu'elles sont issues de **5 familles/types en général** :

- **serif** : les extrémités des lettres ont des extensions ;
- **sans-serif** : police sans extensions ;
- **cursive** : police ressemblant à une écriture manuscrite ;
- **monospace** : les caractères ont tous la même taille et sont espacés équitablement ;
- **fantasy** : Les caractères sont généralement difformes ou très stylisés, au contraire de la famille générique monospace.



Les propriétés **width** et **height**

Tout bloc/élément HTML prendra de la place/espace dans une page. Cet espace pris dans la page peut être déterminé soit par :

- Le type de l'élément HTML :
 - **block** : certains blocs HTML (ex : `<p>`, `<h1>` (et les autres titres), `<div>`, `<table>`, etc.) sont par défaut définis en block - prend toute la largeur par défaut ;
 - **inline** : certains blocs HTML sont par défaut définis en inline (`<a>`, ``, `
`, `<input>`, ``, ``, etc.). Cela signifie que le bloc prendra uniquement la taille en fonction du texte qu'il contient.
- Par les propriétés **width** (largeur) et/ou **height** (hauteur) :

HTML	CSS	RESULTAT
<pre><div class="bloc_div"></div></pre>	<pre>.bloc_div { width: 150px; height: 50px; background-color: blue; }</pre>	

Même si cela est possible, il est déconseillé de mélanger plusieurs unités de mesure (px et % par exemple) pour définir la largeur et/ou la hauteur des différents blocs HTML d'une page, pouvant entraîner des problèmes de taille par rapport aux autres éléments HTML.

CSS - La notion de « boîtes »

Pour concevoir une page web ou une application mobile, il est important de comprendre le concept de « boîtes » en CSS. Tout est relatif à l'imbrication d'éléments HTML les uns par rapport aux autres. En fonction des propriétés/valeurs utilisées, cela va affecter le **flux** de la page courante (notion importante). Le flux est le **comportement par défaut d'affichage des blocs HTML**, c'est-à-dire qu'il va **afficher les éléments en fonction de l'ordre de déclaration** des éléments HTML dans le fichier `.html`.

Il y a :

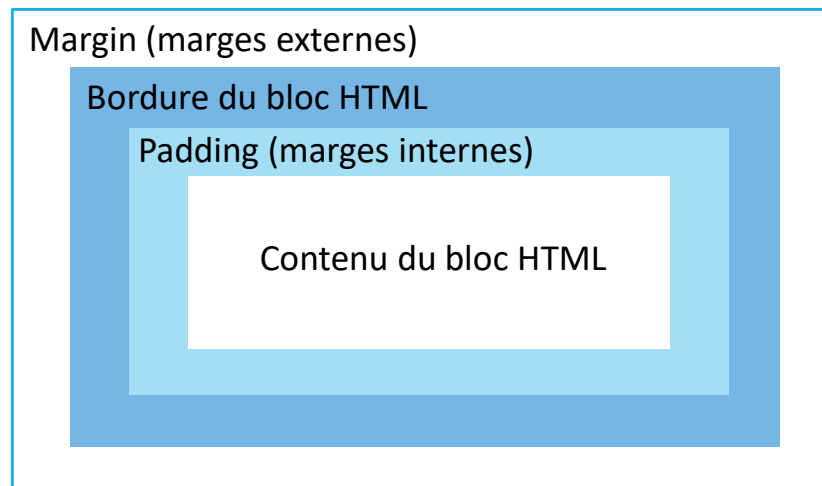
- La notion de « boîtes » propre à un bloc/élément HTML (le **Box Model** ou modèle de boîte) qui va permettre de déterminer sa taille avec les propriétés suivantes : `padding`, `border`, `margin` => **permet de rester dans le flux courant.**
- La notion de « boîtes » pour positionner les différents blocs/éléments HTML les uns par rapport aux autres avec les propriétés suivantes : `display`, `position (absolute, float)` => **permet de sortir du flux courant et se positionne « n'importe où » (pas tous) => les éléments en-dessous prennent sa place dans le flux courant.**

CSS - La notion de « boîtes »

padding, border, margin (1/2)

Chaque bloc/élément HTML possède ses propres « boîtes » qui vont déterminer sa taille. Cette notion s'appuie sur le **Box Model**, qui est la manière de calcul des dimensions de chacune des boîtes propre au bloc HTML pour permettre au navigateur de les afficher correctement. Ces « boîtes » sont les propriétés **padding**, **border** et **margin** qui vont permettre de déterminer la taille du bloc HTML en fonction des marges internes/externes et bordure prédéfinies.

Ci-dessous la représentation d'un bloc HTML et ses « boîtes » :



- la 1^{ère} boîte est le contenu ;
- La 2^{ème} boîte est le contenu + le padding ;
- La 3^{ème} boîte est le contenu + le padding + la bordure ;
- La 4^{ème} boîte est le contenu + le padding + la bordure + la marge.

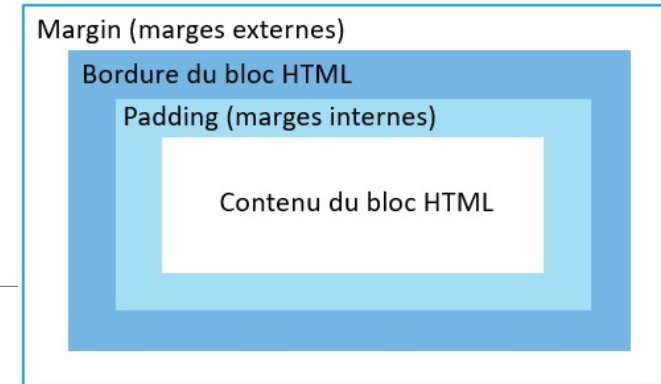
La taille du bloc HTML est calculée de la manière suivante :

- Largeur du bloc (width) : $\text{width} + \text{padding-left} + \text{padding-right} + \text{border-left} + \text{border-right}$;
- Hauteur du bloc (height) : $\text{height} + \text{padding-top} + \text{padding-bottom} + \text{border-top} + \text{border-bottom}$.

CSS - La notion de « boîtes » padding, border, margin (2/2)

SYNTAXE POUR LES PROPRIÉTÉS CSS PADDING, BORDER, MARGIN :

Le mélange de plusieurs unités de mesure est possible :



	PADDING	BORDER	MARGIN
HAUT	padding-top: <i>largeur</i>	border-top: <i>largeur trait couleur</i>	margin-top: <i>largeur</i>
DROITE	padding-right: <i>largeur</i>	border-right: <i>largeur trait couleur</i>	margin-right: <i>largeur</i>
BAS	padding-bottom: <i>largeur</i>	border-bottom: <i>largeur trait couleur</i>	Margin-bottom: <i>largeur</i>
GAUCHE	padding-left: <i>largeur</i>	border-left: <i>largeur trait couleur</i>	Margin-left: <i>largeur</i>
RACCOURCIS	padding: <i>haut, droite, bas, gauche</i> Exemple : 0 2px 1em 4rem OU 5px (toutes les marges seront de 5px)	border: <i>largeur trait couleur (pour toute la bordure)</i> Exemple : 5px solid blue;	margin: <i>haut, droite, bas, gauche</i> Exemple : 0 2px 1em 4rem OU 5px (toutes les marges seront de 5px)
	padding: <i>haut/bas, droite/gauche</i> Exemple : 5px 8px		margin: <i>haut/bas, droite/gauche</i> Exemple : 5px 8px
	Padding: <i>haut, droite/gauche, bas</i> Exemple : 2rem 10em 10px		margin: <i>haut, droite/gauche, bas</i> Exemple : 2rem 10em 10px

CSS - La notion de « boîtes »

display, position, float (1/4)

Les boîtes ne sont pas obligatoirement « insérées » les unes dans les autres. Il est possible de modifier le comportement d'une boîte et de la positionner ailleurs que l'affichage par défaut. Les propriétés `display`, `position`, `float` vont aider dans ce sens.

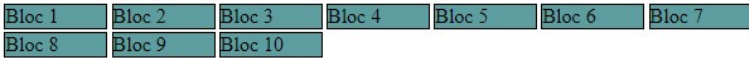
- La propriété `display` permet de modifier le comportement d'un bloc HTML. Cette propriété peut être associée aux valeurs suivantes (liste non exhaustive) :
 - block** : certains blocs HTML (ex : `<p>`, `<h1>` (et les autres titres), `<div>`, `<table>`, etc.) sont par défaut définis en **block**. Cela signifie que le bloc HTML **prend toute la largeur disponible et se positionne en-dessous de l'élément précédent** (comme un rocher qui tombe) :

HTML	CSS	RESULTAT
<pre><p class="premier_bloc">Je suis le premier bloc HTML</p> <p class="deuxieme_bloc">Je suis le deuxième bloc en-dessous du premier bloc</p></pre>	<pre>.premier_bloc { font-size: 16px; background-color: aquamarine; font-family: sans-serif; } .deuxieme_bloc { font-size: 16px; background-color: aqua; font-family: sans-serif; }</pre>	<p>Il n'est pas nécessaire de préciser <i>display: block</i>; car par défaut les éléments HTML <code><p></code> sont en block.</p> <div><div>Je suis le premier bloc HTML</div><div>Je suis le deuxième bloc en-dessous du premier bloc</div></div>


CSS - La notion de « boîtes »

display, position, float (2/4)

- inline-block** : lorsqu'on définit les blocs HTML en **inline-block**, cela signifie qu'on garde les mêmes caractéristiques que la valeur **block** mais que les éléments HTML sont disposés sur **une même ligne** (et non les uns en-dessous des autres). Les éléments HTML passeront en dessous lorsqu'ils n'auront plus de place vis-à-vis du viewport (fenêtre) :

HTML	CSS	RÉSULTAT
<pre><div class="bloc">Bloc 1</div> <div class="bloc">Bloc 2</div> <div class="bloc">Bloc 3</div> <div class="bloc">Bloc 4</div> <div class="bloc">Bloc 5</div> <div class="bloc">Bloc 6</div> <div class="bloc">Bloc 7</div> <div class="bloc">Bloc 8</div> <div class="bloc">Bloc 9</div> <div class="bloc">Bloc 10</div></pre>	<pre>.bloc { display: inline-block; width: 100px; font-size: 20px; background-color: cadetblue; border: 1px solid black; margin-top: 2px; }</pre>	

- inline** : certains blocs HTML sont par défaut définis en **inline** (`<a>`, ``, `
`, `<input>`, ``, ``, etc.). Cela signifie que le bloc prendra uniquement la taille en fonction du texte qu'il contient. S'il n'y a pas de contenu, la largeur du bloc est réduite à 0. Il n'est pas possible de leur appliquer les propriétés width et height, ni de marges haute et basse (margin-top et margin-bottom). Cependant, le bloc ne pourra avoir des marges internes (padding) :

HTML	CSS	RÉSULTAT
<pre><div class="bloc">Bloc 1</div> <div class="bloc">Bloc 2</div> <div class="bloc">Bloc 3</div> <div class="bloc">Bloc</div> <div class="bloc">Bloc</div> <div class="bloc">Bl</div> <div class="bloc"></div></pre>	<pre>.bloc { display: inline; font-size: 20px; background-color: rgb(0, 255, 136); border: 1px solid black; }</pre>	

CSS - La notion de « boîtes »

display, position, float (3/4)

- **none** : lorsqu'on définit les éléments HTML en **none**, cela signifie qu'on fait disparaître les blocs et contenus associés de la page :

HTML	CSS	RÉSULTAT
<pre><div class="bloc">Bloc 1</div> <div class="bloc">Bloc 2</div> <div class="bloc">Bloc 3</div> <div class="bloc">Bloc</div> <div class="bloc">Bloc</div> <div class="bloc">Bl</div> <div class="bloc"></div></pre>	<pre>.bloc { display: none; font-size: 20px; background-color: rgb(0, 255, 136); border: 1px solid black; }</pre>	

- La propriété **position** permet de modifier la position d'un bloc HTML dans une page. Cette propriété est associée aux valeurs suivantes :
 - **static** : **n'affecte pas le flux**. Valeur par défaut d'un bloc HTML. Doit être utilisé si un bloc HTML a été mis hors du flux et qu'on souhaite le remettre dans le flux courant de la page.
 - **relative** : **n'affecte pas le flux**. Permet de modifier la position d'un bloc HTML par rapport à sa position initiale grâce aux propriétés **top**, **left**, **bottom**, **right**.
 - **absolute** : **affecte le flux**. Permet de se positionner par rapport à son parent le plus proche. Si aucun parent n'est trouvé alors l'élément sera positionné par rapport à l'élément racine, soit la page en elle-même. On utilisera les propriétés **top**, **left**, **bottom**, **right** pour modifier sa position.
 - **fixed** : **affecte le flux**. Presque similaire à **absolute**. Permet de se positionner par rapport au **viewport** (à moins que l'un de ses parents ne possède une propriété **transform**, **filter** ou **perspective** dont la valeur est différente de **none**). L'élément sera donc toujours à la même place (de manière fixe) et sa position sera calculé par rapport à la fenêtre avec les propriétés **top**, **left**, **bottom**, **right** ;
 - **sticky** : **n'affecte pas le flux**. La position d'un élément sticky va être calculée par rapport à son parent possédant un mécanisme de défilement (scrolling) le plus proche.

CSS - La notion de « boîtes »

display, position, float (4/4)

La propriété **float** permet de faire « flotter » un élément HTML dans le conteneur (la boîte) dans lequel il est. L'autre élément HTML en-dessous de l'élément flottant viendra l'englober : les éléments HTML de type **inline** vont se positionner autour, et les éléments HTML de type block vont se placer sur la même ligne et prendre tout l'espace disponible. Le **float retire l'élément du flux courant**. Il est associé aux valeurs suivantes :

- **right** ;
- **left** ;
- **inline-start** - récent - attention à la compatibilité des navigateurs - à vérifier sur : <https://caniuse.com/?search=inline-start> ;
- **inline-end** - récent - attention à la compatibilité des navigateurs - à vérifier sur : <https://caniuse.com/?search=inline-end> ;
- **none**.

Pour empêcher un élément HTML de se positionner à côté d'un élément HTML flottant on utilisera la propriété **clear**. Selon la situation, elle peut être associée aux valeurs suivantes :

- **left** : empêche un élément de se positionner à côté d'éléments possédant un *float: left* ;
- **right** : empêche un élément de se positionner à côté d'éléments possédant un *float: right* ;
- **both** : empêche un élément de se positionner à côté d'éléments possédant un *float: left* ou un *float: right* ;
- **inline-start** : empêche un élément de se positionner à côté d'éléments possédant un *float: inline-start* ;
- **inline-end** : empêche un élément de se positionner à côté d'éléments possédant un *float: inline-end* ;
- **none** : valeur par défaut. Laisse les éléments se positionner à côté d'éléments flottants.

CSS - Unité de mesure (1/3)

Il existe en CSS plusieurs unités de mesures pour ajuster la taille de la police et éléments/blocs. Chaque unité de mesure est spécifique et son utilisation dépendra de l’affichage voulu. On distingue :

- **Les unités absolues** : il s’agit d’unités qui ne sont pas « influencées » / qui ne s’ajustent pas en fonction d’un autre élément. Ainsi, l’élément sur lequel sera affecté une unité absolue aura la même taille, quelle que soit la taille de l’écran. Par exemple, 1px correspond à 1 px de l’écran, et ainsi un bloc de 150x150px gardera sa taille quelle que soit la taille de l’écran.
- **Les unités relatives** : il s’agit d’unités qui varient / s’ajustent en fonction de la taille de la police ou de l’élément parent.

Les unités de mesure à éviter pour un affichage sur ordinateur :

Il existe des unités qui ne sont pas conseillées pour un affichage sur un écran, comme le point (pt), centimètre(cm), le millimètre (mm), le pouce (in) et le pica (pc). Ces unités sont davantage conseillées pour des supports haute-résolution ou des supports d’impression.

CSS - Unité de mesure (2/3)

- **px** (pixel) - **unité absolue** : le pixel est donc une unité absolue et ne peut donc pas varier. 1px correspond à 1 px à l'écran. Si des éléments ou textes sont définis en px, il faudra alors les redéfinir manuellement pour chaque résolution d'écran ;
- **%** (pourcentage) - **unité relative** : permet de définir la taille d'un élément/texte par rapport à la taille de l'élément parent. Un élément avec une taille de 50% et qui a un élément parent avec une taille de 50px aura alors une taille de 25px ;
- **em** - **unité relative** : permet de définir la taille d'un élément/texte par rapport à la taille de l'élément parent. Elle correspond à la taille de l'élément en cours et si cette taille n'est pas redéfinie alors cette taille correspondra à la taille de l'élément parent. Par exemple, si dans le body la taille de la police est définie à 16px alors sur l'élément enfant la taille de la police de 2em correspondra à 32 px ($2 \times 16 = 32\text{px}$). Le em étant toujours relatif au parent il suffit de modifier la taille de la police du parent pour voir l'élément changer de taille. Les effets souhaités ne sont toujours pas toujours ceux attendus.

```
.div-parent {  
  font-size: 16px;  
}  
  
.div-enfant {  
  width: 2em /* équivaut à 32px car 2x16=32px */  
}
```


CSS - Unité de mesure (3/3)

- **rem** (root em) - **unité relative** : reprend le principe de l'unité **em**, c'est-à-dire correspondre à la taille d'un élément et cet élément est l'élément racine du document (root). Cette unité agit comme le em mais résout le précédent problème d'héritage. Il ne se base pas sur l'élément parent pour obtenir sa taille mais sur l'élément racine. Ainsi **1rem** prendra sa valeur de la font-size de votre document (body ou html). Ce qui permet d'avoir un comportement beaucoup plus prévisible que le em.

```
.body {  
  font-size: 10px;  
}  
  
.div {  
  width: 2rem /* équivaut à 20px car 2x10=20px */  
}
```

OU

```
html {  
  font-size: 62.5%; /*équivaut à 10px*/  
  /* par exemple : 1,5rem = 15px, 3rem = 30px */  
}
```

- **vw** et **vh** (viewport width - viewport height) - **unité relative** : ces 2 unités correspondent au viewport de la fenêtre :
 - **vw** correspond à la largeur du viewport, soit 1/100^e ou 1% de la largeur de la fenêtre ;
 - **vh** correspond à la hauteur du viewport, soit 1/100^e ou 1% de la hauteur de la fenêtre.
- **vmin** et **vmax** (viewport min - viewport max) - **unité relative** : ces 2 unités permettent de prendre en compte la largeur/hauteur minimale et la largeur/hauteur maximale du viewport (soit 1/100^e ou 1% de la largeur (minimum ou maximum) et 1/100^e ou 1% de la hauteur (minimum ou maximum) de la fenêtre). Par exemple, pour un viewport ayant les dimensions de 2000px de large et 1000px de hauteur, une valeur de **1vmin** sera égale à **10px** et une valeur de **1vmax** équivaudra à **20px**.

CSS - Police d'écriture (1/3)

Au sein du système d'exploitation, sont déjà pré-installées des polices d'écriture qu'on appelle des « Web Safe Fonts » dits polices systèmes (exemples : Arial, Georgia, Verdana, etc.). Ces polices systèmes sont des polices d'écriture qui sont lues de manière universelle par tous les navigateurs :

- Avantage : les polices systèmes n'ont pas besoin d'être téléchargées par le navigateur.
- Inconvénient : limité uniquement à la collection des polices systèmes installées sur le système d'exploitation.

Ainsi, pour ne pas être limité uniquement à l'utilisation des Web Safe Fonts et diversifier les polices d'écriture, il existe 2 solutions :

- **Google fonts** ;
- **@font-face**.

RAPPEL - il existe 5 familles de police :

- **serif** : les extrémités des lettres ont des extensions ;
- **sans-serif** : police sans extensions ;
- **cursive** : police ressemblant à une écriture manuscrite ;
- **monospace** : les caractères ont tous la même taille et sont espacés équitablement ;
- **fantasy** : Les caractères sont généralement difformes ou très stylisés, au contraire de la famille générique monospace.

CSS - Police d'écriture

Google Fonts (2/3)

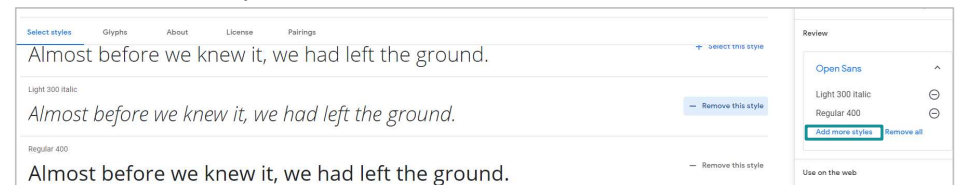
Google propose une collection de polices qu'il est possible d'utiliser gratuitement à travers son service d'hébergement Google Fonts. Ces polices sont libres de droit et peuvent être utilisées autant de fois souhaitée.

Google Fonts permet de pré-charger le jeu complet de caractères de la police via Google dans le navigateur d'un utilisateur lorsque celui-ci tente d'accéder à une page du site. Ainsi, les risques de non comptabilité d'une police sont réduits.

Google Fonts ne se contente pas de fournir les fichiers de polices au navigateur, elle effectue une vérification intelligente pour voir comment elle peut livrer les fichiers dans le format le plus optimisé. Lorsque le navigateur fait une demande à l'API Google Fonts, Google vérifie d'abord quels types de fichiers sont pris en charge par le navigateur. Par exemple :

- Sur la dernière version de Chrome, comme la plupart des navigateurs supporte WOFF2, la police est servie dans ce format hautement compressé ;
- IE11, la police servie est au format WOFF ;
- IE8, la police servie est au format EOT (Embedded OpenType).

- Site de Google Fonts : <https://fonts.google.com/>
- Il est également possible de sélectionner plusieurs polices en même temps :



- Pour utiliser la police, copier-coller le lien fourni par Google Fonts dans la feuille de style :



Mettre ce lien dans le **<head>** avant le **<link>** de la feuille de style

Pour utiliser la police choisie, dans la feuille de style, faites appel la propriété font-family avec les valeurs associées.

CSS - Police d'écriture

@font-face (3/3)

Il est possible également de ne pas dépendre d'un service d'hébergement et d'importer des fichiers de police d'écriture qu'on souhaiterait utiliser pour notre site et de la personnaliser (ou de les installer directement sur l'ordinateur). Il s'agit alors d'utiliser le sélecteur **@font-face** dans la feuille de style. Ainsi, comme pour une image, la police d'écriture sera chargée au moment de l'appel de la feuille de style. Cependant, si vous décidez d'utiliser un **@font-face**, il faudra faire attention au poids des fichiers et des droits d'auteurs sur la police souhaitée.

Pour trouver des polices à télécharger il existe plusieurs sites dont les suivants :

- <https://www.fontsquirrel.com/>
- <https://www.dafont.com/fr/>

Pour utiliser le **@font-face**, il faut respecter les déclarations suivantes :

```
@font-face {  
  font-family: "Bitstream Vera Serif Bold";  
  src: url("/static/styles/libs/font-awesome/fonts/fontawesome-webfont.woff");  
}  
  
body {  
  font-family: "Bitstream Vera Serif Bold", serif;  
}
```

→ Importer la police d'écriture

→ Nommer le nom de la police d'écriture

→ Détailler le chemin d'accès du fichier de la police d'écriture

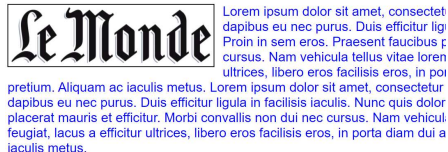
→ Appeler la police d'écriture pour l'utiliser

Pour plus de détails : <https://developer.mozilla.org/fr/docs/Web/CSS/@font-face>

HTML & CSS - EXERCICE 3

■ Reprendre l'exercice 2 pour effectuer les actions suivantes :

- Rajouter une balise `<main>` pour séparer l'entête et la corps principal de la page ;
- Sur l'image imbriquée dans le lien `<a>` (2^{ème} image), modifier la largeur et la hauteur de l'image en mettant comme unité de mesure **px** (mettez la taille que vous souhaitez) ;
- En-dessous de l'image, rajouter un paragraphe de texte :
- **Trouvez le moyen** pour mettre le texte à côté de l'image comme ci-dessous :



- Espacez l'image et le texte de 20px (utilisez la propriété que vous souhaitez) ;

■ Créer un menu :

- À la place du « Hello World », créez un menu avec les balises suivantes : `<header>`, `<nav>`, ``, ``. Faites le menu suivant : **Accueil, Articles, Contact, Mon compte** ;

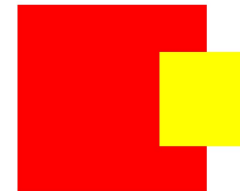
- Mettez la 1^{ère} image dans le menu, à gauche pour qu'il devienne un « logo » cliquable qui renvoie vers le menu. **Trouvez le moyen d'aligner le logo et le menu verticalement** ;
- Changez la police d'écriture avec la Google Fonts **Open Sans** en Regular (400) ;
- Tout en haut de la feuille de style, ajoutez la propriété suivante pour avoir de base une taille de police de 10px :

```
html {  
    font-size: 62.5%;  
}
```

- Changez la couleur et la taille de police du menu et des paragraphes pour harmoniser visuellement le site (ce que vous souhaitez).

■ Entraînement sur les positions :

- Créez 1 bloc avec la balise `<div>` et imbriquez dans celle-ci 1 autre bloc ;
- Ajouter à ces 2 blocs des couleurs de fond (les couleurs que vous souhaitez) ;
- Essayez de reproduire les positions suivantes :



HTML & CSS - Formulaire

Balise `<form>` :

- Action ;
- Méthode.

Balise `<input>` :

- Text ;
- Email ;
- Password ;
- Submit.

Balise `<label>`.

Balise `<textarea>`.

HTML & CSS - EXERCICE 4

TP Formulaire

- **Créer une nouvelle page connexion et la lier à la page d'accueil.** Elle doit contenir :
 - Un titre ;
 - Un formulaire contenant :
 - Un champ email ;
 - Un champ mot de passe ;
 - Un bouton connexion.
 - Le bloc contenant le titre et le formulaire doit être centré au milieu de l'écran.

HTML & CSS – Div & span

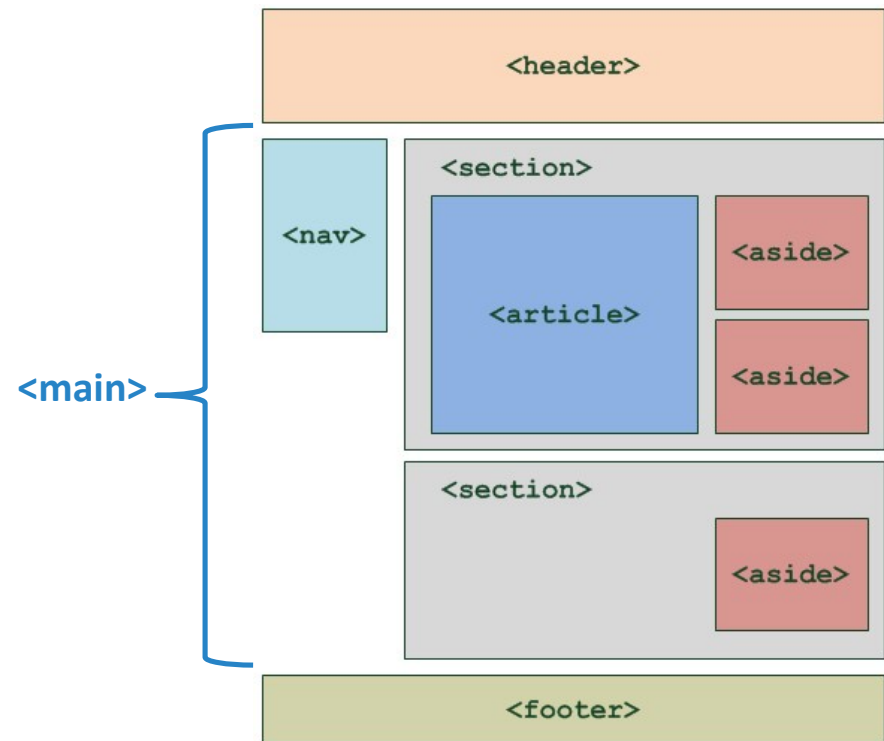
Les balises `<div>` et `` sont des balises neutres. Elles ne servent qu'à grouper d'autres balises sans dénaturer la sémantique de la page.

La différence :

- `<div>` est une balise de type display `block` ;
- `` est une balise de type display `inline`.

HTML & CSS – Balises sémantiques

- `<main>` : contenu principal de la page ;
- `<section>` : une section de la page ;
- `<article>` : un contenu de la page ;
- `<aside>` : un contenu qui a un rapport indirect avec le contenu principal
- Citations :
 - `<q>` : citation en incise (citation au milieu d'un texte) ;
 - `<blockquote>` : bloc de citation ;
 - `<cite>` : contient le titre de l'œuvre citée ou le nom de l'auteur si pas de titre.



Crédit : www.openclassrooms.com

HTML & CSS – Flexbox

Flexbox ou « Modèle de boîte flexible » permet de ne plus raisonner en boîtes classiques en termes de « block » ou « inline », ni en « float ». Ces conteneurs sont dits « flexibles » car leurs enfants directs vont être des éléments flexibles qui vont pouvoir se réarranger (se redimensionner, se réaligner, etc.) automatiquement dans leur conteneur lorsque celui-ci change de dimension. C'est donc une méthode considérable qui permet de placer des blocs plus facilement. En fonction de ce que l'on souhaite faire, on a le choix d'agir soit :

- Sur les blocs parents (flex container) ;
- Sur les blocs enfants (flex items).

Pour activer le flex :

- Mettre la propriété et la valeur suivantes au sélecteur voulu : **display: flex;**. Par défaut, l'orientation du flex sera en **row** (voir ci-dessous) ;
- Possibilité de choisir l'orientation avec **flex-direction** qui prend la valeur **row** ou **row-reverse** ou **column** ou **column-reverse** ;
 - **row** : les enfants se mettent tous sur la même ligne comme une display inline ;
 - **column** : les enfants se mettent tous les uns en dessous des autres comme une liste.
- Possibilité d'espacer les blocs enfants avec **justify-content** qui prend la valeur **flex-start** ou **flex-end** ou **center** ou **space-between** ou **space-around** ou **space-evenly** ;
- Et beaucoup d'autres possibilités !

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

<https://www.alsacreations.com/tuto/lire/1493-css3-flexbox-layout-module.html>

HTML & CSS - EXERCICE 5

TP Liste d'articles de blog

Reprendre les fichiers des exercices 3 et 4 :

➤ Dans l'`index.html` :

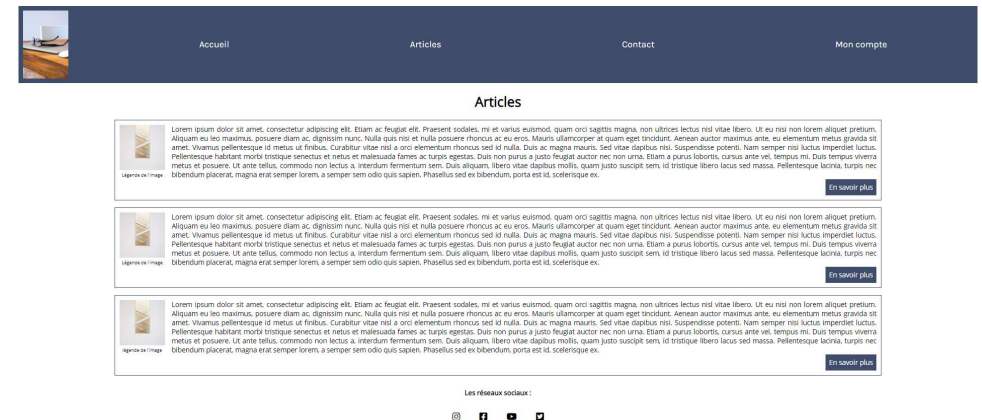
- Effacer le contenu de la balise `<main>` ;
- Utiliser 2 polices d'écriture différentes (1 pour le menu et 1 pour le reste de la page) ;
- Sur le `<main>`, rajouter un width de 80% et trouvez le moyen de centrer le `<main>` ;
- Rajouter un titre `<h1>Articles</h1>` ;
- Modifier la page d'accueil du site pour créer 3 blocs de résumés d'articles avec :
 - 1 image avec 1 légende ;
 - Du texte qui entoure l'image ;
 - 1 bouton « En savoir plus » qui renverra vers la page d'accueil ;
- Rajouter un footer avec les icônes des réseaux sociaux qui doivent renvoyer vers la page d'accueil.

HTML : utilisez les balises `<section>`, `<article>`, `<figure>`, `<footer>`.

CSS : utilisez le `flex` et le `float` pour le positionnement de certains blocs.

➤ Pour le fichier `connexion.html`, liez le formulaire de connexion au lien « Mon compte » du fichier `index.html`.

Reproduire la maquette suivante :



HTML & CSS – Media queries

Les media queries permettent de modifier l'apparence d'un site ou d'une application en fonction du type d'appareil ;

On utilise le sélecteur `@media` suivi d'un attribut et d'une condition

- `/* Sur les écrans, quand la largeur de la fenêtre fait au maximum 1280px */`
`@media screen and (max-width: 1280px)`
- `/* Sur tous types d'écran, quand la largeur de la fenêtre est comprise entre 1024px et 1280px */`
`@media all and (min-width: 1024px) and (max-width: 1280px)`
`@media tv`
`@media all and (orientation: portrait)`

Les bonnes pratiques consistent à développer d'abord :

1. pour le **mobile**, qu'on appelle communément « mobile first » ;
2. ensuite pour la **tablette** ;
3. ensuite pour le **desktop**.

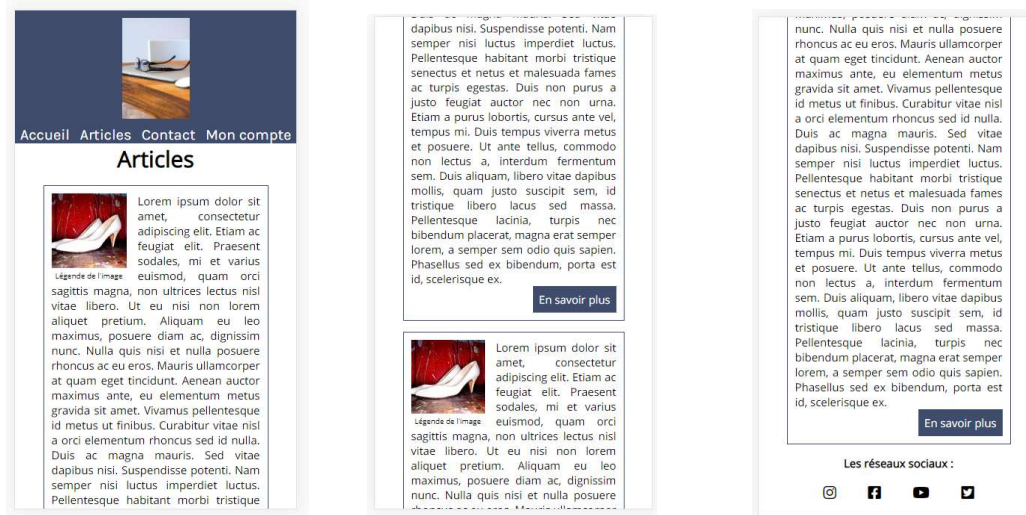
HTML & CSS - EXERCICE 6

TP Media queries

Reprendre le site d'articles et l'adapter pour :

- Le mobile ;
- La tablette.

Modifier le site pour qu'il ressemble à la maquette suivante en mobile :



HTML & CSS – Cssh Grid

Permet de définir une grille en CSS pour placer les éléments HTML selon une ligne et une colonne de cette grille.

On active la grille avec la valeur grid de la propriété display :

```
.wrapper { display: grid; }
```

Puis on peut définir les colonnes et les lignes :

```
.container { grid-template-columns: ... | ...; grid-template-rows: ... | ...; }
```

Puis on peut placer les éléments dans la grille :

```
nav { grid-column: 1; grid-row: 1; }
```

On peut nommer des zones de notre grille et les affecter directement :

```
#inGrid { display: grid; grid-template-areas: "h h" "n c" "f f"; } nav { grid-area: n; /* placement de <nav> dans l'emplacement "n" */ }
```

HTML & CSS – Tableaux HTML

<code><table></code>	→	Racine du tableau / Initialisation du tableau	
<code><thead></code>	→	En-tête du tableau	
<code><tr></code>	→	Ligne	
<code><th colspan="2">The table header</th></code>	→	Cellule d'en-tête	
<code></tr></code>			
<code></thead></code>			
<code><tbody></code>	→	Corps du tableau / Contenu principal du tableau	
<code><tr></code>	→	Ligne	
<code><td>The table body</td></code>	→	Cellule du corps du tableau	
<code><td>with two columns</td></code>			
<code></tr></code>			
<code></tbody></code>			
<code><tfoot></code>	→	Pied-de-page du tableau	
<code><tr></code>	→	Ligne	
<code><th scope="row">Totals</th></code>	→	Cellule	
<code><td>21,000</td></code>	→	Cellule	
<code></tr></code>			
<code></tfoot></code>			
<code></table></code>			

OOCSS VS Interface-Oriented

Interface-oriented :

- Sélection des éléments, le nom de l'élément et les enfants de ces éléments ;
- Ne pas surcharger le HTML ;
- Contrainte : le CSS est spécifique à ce site ou cette page.

Object Oriented CSS (OOCSS) :

- Utilisation des classes CSS pour mettre en forme le contenu. Le but est la réutilisation ;
- Le principe de l'OOCSS :
 - Séparation de la structure et de l'apparence ;
 - Séparation du conteneur et du contenu.

HTML & CSS – Bootstrap

- Bootstrap est un framework (boîte à outils) CSS qui met à disposition des classes CSS pour mettre en forme la page HTML facilement ;
- Grid system ;
- Media queries auto (rwd) ;
- Components.

HTML & CSS – TP Bootstrap

Reprendre la précédente maquette en supprimant le CSS et la refaire avec bootstrap.

HTML & CSS – lib & framework

- Bootstrap ;
- Fontawesome ;
- Google fonts.
- Best Practice