



# Formation HTML 5, CSS 3, Responsive

M2I - Formation - 2023

---

Donjon Audrey



m2iformation.fr

# Module HTML – CSS et responsive



# Introduction

Points principaux qui seront abordés :

- HTML (HyperText Markup Language) : Structure ma page Web.
- CSS (Cascading StyleSheets) : Met en forme ma page Web.
- Les medias queries : Rend ma page adaptable à différents formats d'écrans.
- Bootstrap : Framework Frontend.
- Les fondamentaux de l'accessibilité numérique.

Outils :

- VSC (Visual Studio Code) : Éditeur de texte
- Navigateurs : Firefox, Chrome, Edge...
- Recherche internet / Support de cours

# Préparer l'environnement



1. Installer VSCode :

<https://code.visualstudio.com/Download>

2. Installer Google chrome ou Mozilla firefox :

[https://www.google.com/intl/fr\\_fr/chrome/](https://www.google.com/intl/fr_fr/chrome/)

<https://www.mozilla.org/fr/firefox/new/>



3. Télécharger L'extension chrome ou Firefox Web Developer :

[web-developer Chrome](#)

[web-developer Firefox](#)

4. Installer les extensions vs code suivantes :

a) French Language

b) Trailing spaces

c) Auto Rename tag



# Le HTML



- [1 - Introduction](#)
- [2 - Les bases](#)
- [3 - Les balises](#)
- [4 - Les attributs](#)
- [5 - Les tableaux](#)
- [6 - Les formulaires](#)
- [7 - La sémantique de structure](#)
- [8 - Les vidéos et audios](#)

# Introduction HTML

- ▶ **1969** : L'ancêtre d'Internet, Arpanet, est créé par l'armée américaine pendant la guerre froide. C'est le premier **réseau décentralisé** en forme de « toile », pensé pour éviter qu'une attaque nucléaire ciblée puisse le détruire.
- ▶ **1983** : Arpanet devient Internet.
- ▶ **1990** : Création du Web, du premier **navigateur Web** et du **langage HTML** (langage de balisage d'hypertexte : est le langage de balisage conçu pour représenter les pages web.) par Tim Berners-Lee, aussi appelé le « **père du web** ».
- ▶ **1995-1996** : HTML 2.0 => Amélioration balises, Tableaux, concept de formulaire etc.
- ▶ **1997** : HTML3.2 => Balises HTML améliorés, CSS dans le fichier HTML , formulaire amélior etc.
- ▶ **1999** : HTML4.01 => fichier CSS externe etc.
- ▶ **2014** : HTML5 (*Version actuelle*) => Nouvelles balises HTML, nouveaux éléments de formulaire.

# Les bases du HTML

- ▶ HTML est un langage de **balise**.
- ▶ Une balise permet d'indiquer la **nature** du texte.
- ▶ La syntaxe pour utiliser une balise est la suivante :

```
<balise_double></balise_double> ou <balise_orpheline/>
```

- ▶ La syntaxe pour déclarer une page html est divisée en 4 balises :

```
<!DOCTYPE html> : Doit se situer tout en haut de mon code avant les balises <html>  
<html></html> : Doit englober les balises suivantes  
<head></head> : Doit se situer entre les balises <html> et tout en haut  
<body></body> : Doit se situer entre les balises <html> et après les balises <head>
```

- ▶ L'ajout de commentaire en HTML se fait de la manière suivante :

```
<!-- Commentaire HTML -->
```

# Les balises

- On structure son code à l'aide des **balises**, en voici quelques sunes :

|                                     |  |
|-------------------------------------|--|
| Paragraphe                          | <pre>&lt;p&gt;Je suis un paragraphe&lt;/p&gt;</pre>  |
| Titres (6 niveaux)                  | <pre>&lt;h1&gt;Je suis un Titre niveau 1&lt;/h1&gt; &lt;h2&gt;Je suis un Titre niveau 2&lt;/h2&gt; &lt;h3&gt;Je suis un Titre niveau 3&lt;/h3&gt; ..... &lt;h6&gt;Je suis un Titre niveau 6&lt;/h6&gt;</pre>   |
| Saut de ligne                       | <pre>&lt;br&gt;</pre>  |
| Listes (ordonnées et non ordonnées) | <pre>&lt;ul&gt;     &lt;li&gt;je suis la première ligne de la liste à puce&lt;/li&gt;     &lt;li&gt;je suis la 2ème ligne de la liste à puce&lt;/li&gt;     &lt;li&gt;je suis la 3ème ligne de la liste à puce&lt;/li&gt;     &lt;li&gt;je suis la 4ème ligne de la liste à puce&lt;/li&gt; &lt;/ul&gt; &lt;ol&gt;     &lt;li&gt;je suis la première ligne de la liste numérotée&lt;/li&gt;     &lt;li&gt;je suis la 2ème ligne de la liste numérotée&lt;/li&gt;     &lt;li&gt;je suis la 3ème ligne de la liste numérotée&lt;/li&gt;     &lt;li&gt;je suis la 4ème ligne de la liste numérotée&lt;/li&gt; &lt;/ol&gt;</pre> |

# Les balises

- On structure son code à l'aide des **balises**, en voici quelques sunes :

|                              |   |
|------------------------------|---|
| Image                        | <code>&lt;img src="chemin_vers_mon_image.png" alt="texte_de_image"&gt;</code>   |
| Lien (relatif ou absolu)     | <code>&lt;a href="url(exemple : https://www.google.com/)"&gt;lien absolu : construit avec un protocole et nom de domaine&lt;/a&gt;</code><br><code>&lt;a href="chemin(exemple : ./page_contact.html)"&gt;lien relatif : construit par rapport à la page actuelle&lt;/a&gt;</code> |
| Balises sémantiques de texte | <code>&lt;em&gt;Accentuation du texte&lt;/em&gt;</code><br><code>&lt;strong&gt;Texte important&lt;/strong&gt;</code><br><code>&lt;mark&gt;Texte pertinent dans son contexte&lt;/mark&gt;</code>   |

Documentation en + : <https://developer.mozilla.org/fr/docs/Web/HTML/Element>

# Les attributs

- ▶ Un attribut est une **information additionnelle/spécification** liée à une balise.
- ▶ Les attributs se déclarent dans **les balises ouvrantes** et sont **cumulables**.
- ▶ La syntaxe d'un attribut est la suivante :

```
> <balise_html attribut1="valeur_attribut" attribut2="valeur_attribut"> </balise_html>
```

- ▶ Quelques exemples :

```
> <h1 color="red" align="center"></h1>
> 
> <a href="https://developer.mozilla.org/fr/docs/Web">Lien vers developer.mozilla.org</a>
```

# Exercice 01

Créer une page (home.html) qui permettra d'afficher à l'écran les informations suivantes :

- ▶ Un titre d'onglet “Base clients”
- ▶ Une icône dans l'onglet
- ▶ Un titre de niveau 1: “Ma base client”
- ▶ Un titre de niveau 2: “Description”
- ▶ Une description: “Cette application permet de faire de la gestion de commande et de client.”
- ▶ Les mots “commande” et “client” doivent ressortir en important



## Exercice 02

Reprendre la page et ajouter :

- ▶ Une petite image
- ▶ Deux titres de niveau 3: “Clients” et “Commandes”
- ▶ Une liste de client: “Jean Dupont”, “Sarah Abdram” et “Rachel Taeck”
- ▶ Une liste de commande: “CMD01”, “CMD02”, “CMD03” et “CMD04”
- ▶ Un lien de redirection vers le site de Freepik



# Les spécificités du tableau

- ▶ Principales balises pour afficher un tableau :

- ▶ **Structure** (englobe le tableau) :
- ▶ **En-tête de tableau** (header) :
- ▶ **Contenu de tableau** (body) :
- ▶ **Ajout d'une ligne** :
- ▶ **Ajout de cellule dans une ligne** :

```
> <table> </table>
> <thead></thead>
> <tbody></tbody>
> <tr></tr>
> <th></th> > <td></td>
```

- ▶ Exemple d'attribut :

- ▶ **Bordures** (border=0 par défaut !) :
- ▶ **Fusionner les cellules** (horizontalement) :
- ▶ **Fusionner les cellules** (verticalement) :
- ▶ **Gérer l'alignement**:
- ▶ **Fixer la dimension de la cellule**:

```
> <table border="1">
> <td colspan="2">Cellule qui prend 2 colonnes</td>
> <td rowspan="2">Cellule qui prend 2 lignes</td>
> <td align="right">Cellule</td>
> <td width="50%">Cellule</td>
```

# Exercice 03

Reprendre la **page home.html** en remplaçant les listes à puces des clients et des commandes par **2 tableaux, centré sur la page** et affiché l'un en dessous de l'autre.

Les tableaux doivent avoir le même visuel que ci-après :

**Clients**

| Nom    | Prénom   |
|--------|----------|
| Dupont | Jean     |
| Abdram | Sarah    |
| Taeck  | Rachelle |

**Commandes**

| Numéro de commande | État                   |
|--------------------|------------------------|
| CMD01              | Payé                   |
| CMD02              | Payé                   |
| CMD03              | Non payé               |
| CMD04              | En attente de paiement |

# Exercice 04

Reprendre la page `home.html` et reproduire le rendu suivant avec les tableaux HTML.

| SPA de Chataigneraie |   | SPA de Chambord |   |
|----------------------|---|-----------------|---|
| Chat                 | 4 | Chat            | 2 |
| Chien                | 2 | Chien           | 5 |
| Chaton               | 3 | Chaton          | 6 |
| Chirot               | 5 | Chirot          | 2 |



# Les spécificités du formulaire

- ▶ Principales balises pour afficher un formulaire :
  - ▶ Structure : `<form></form>`
  - ▶ Attribut de Nom unique : `name="form_name"`
  - ▶ Attribut de méthode: `method="GET"`
- ▶ Balises de formulaires :
  - ▶ Étiquette de champ : `<label> </label>`
  - ▶ Champs de saisie : `<input>` (text, number, date, checkbox, password ...)
  - ▶ Liste déroulante:  
`<select><option></option></select>`
  - ▶ Bouton envoyer: `<input type="submit">`
  - ▶ Champs de saisie spécial:  
`<textarea></textarea>`

```
<form action="cible.php" method="GET">
    <!--Champ(input) Prénom(de type text) avec son étiquette(label)-->
    <div>
        <label for="form_id">Prénom :</label><!--Attribut for(label) et id(input) pour identifier le champ et son étiquette-->
        <input id="form_id" type="text" name="form_name"><!--Attribut name : envoie des données avec le nom(name) du champ-->
    </div>
    <!--Champ(input) Nom (de type text) avec son étiquette(label)-->
    <div>
        <label for="form_id">Nom :</label>
        <input id="form_id" type="text" name="form_name">
    </div>
    <!--Liste déroulante-->
    <div>
        <label for="color">Quelle est votre couleur préférée ?</label>
        <select name="color" id="color">
            <option value="rouge">Rouge</option>
            <option value="vert">Vert</option>
            <option value="bleu">Bleu</option>
            <option value="violet">Violet</option>
            <option value="aucune">Je n'ai pas de couleur préférée</option>
        </select>
    </div>
    <!--Champ de texte multi-ligne-->
    <div>
        <label for="message">Votre message :</label>
        <!-- rows = hauteur en nombre de lignes de texte, cols = largeur en nombre de caractères -->
        <textarea name="message" id="message" cols="30" rows="10"></textarea>
    </div>
    <!--Bouton envoyer-->
    <div>
        <input type="submit" value="Envoyer">
    </div>
</form>
```

# Les formulaires et Web Forms 2.0

Web Forms 2.0 est une **extension des fonctionnalités de formulaire** depuis HTML4. On trouvera alors avec HTML5 de nouveaux éléments et attributs pour les formulaires.

Ce qui a été ajouté avec Web Forms 2 en HTML5:

- Attribut input **placeholder**
- Attribut input **autofocus**
- Attribut input **required**

```
<form action="cible.php" method="GET">
<!--
Champ(input)Nom(de type text)avec son étiquette(label).
placeholder : attribut du champ qui permet de fournir une brève indication sur l'information attendue, affiché dans le
champ avant que l'utilisateur entre une valeur.
autofocus : attribut du champ qui sert à indiquer quel élément de formulaire devrait automatiquement obtenir le focus
lorsque la page se charge.
required : attribut de validation du champ qui permet d'indiquer que le champ doit être rempli avant que l'utilisateur
puisse soumettre le formulaire
-->
    <label for="name">Nom :</label>
    <input id="name" type="text" name="name" placeholder="Durand" autofocus required>
</form>
```

À noter : il y a également eu de nouveaux types de champ tel que :

- **datetime**
- **week**
- **email**
- **datetime-local**
- **time**
- **url**
- **date**
- **number**
- **month**
- **range**

# Exercice 05

Réaliser une **fiche d'inscription** des animaux sur un site de garde d'animaux.

**Critères:**

- Nom / Prénom du propriétaire
- Date de naissance du propriétaire
- Type d'animal domestique que je souhaite faire garder avec plusieurs choix possible (chien, chat, rongeur, poisson, autre)
- Si autre animal que la liste préciser ici
- Adresse email du propriétaire
- Numéro de téléphone du propriétaire
- Dernière page de vaccin de l'animal



## Inscription de l'animal à faire garder

Nom du propriétaire:   
Prénom du propriétaire :   
Date de naissance du propriétaire :

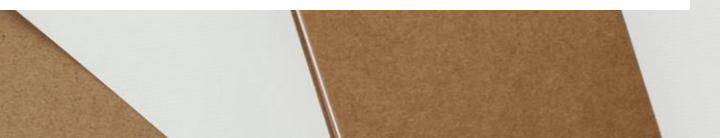
Type d'animal à faire garder

Si vous avez choisi "autre" précédemment, veuillez préciser de quel animal il s'agit :

Adresse mail du propriétaire :

Numéro de téléphone du propriétaire :

Dernière page de vaccin de l'animal :  Aucun fichier sélectionné.



# Les balises sémantiques de structure

|                     |  |
|---------------------|--|
| <header></header>   | Représente l'en-tête de la page(Logo, navigation,etc.)   |
| <nav></nav>         | Menu de navigation   |
| <main></main>       | Contenu principal de la page   |
| <section></section> | Sert à regrouper des contenus en fonction de leur thématique   |
| <article></article> | La balise <article> sert à englober une portion généralement autonome de la page pouvant être repris dans d'autres sites(blog, magazine) |
| <aside></aside>     | Informations complémentaires au document   |
| <footer></footer>   | Pied de page situé en bas du document(contact,mentions légales,etc.)   |

# Les balises audio et video

## Principaux attributs de la balise audio et video :

**controls** : Affiche les contrôles de lecture, pause, volume, etc.

**autoplay** : Le média démarre automatiquement dès que la page est chargée. À noter que certains navigateurs et certains réglages des utilisateurs peuvent empêcher l'autoplay de fonctionner.

**loop** : Le média se répète en boucle.

**muted** : Le média est en sourdine par défaut.

**preload** : Indique comment le navigateur doit charger le média ("none", "metadata", "auto").

Il peut prendre trois valeurs :

- "none" (ne charge pas le média)
- "metadata" (charge seulement les métadonnées comme la longueur du média)
- ou "auto" (charge le média complet).

**src** : URL du média à lire.

## Attributs spécifiques pour video :

**width** : Définit la largeur de la zone de lecture vidéo.

**height** : Définit la hauteur de la zone de lecture vidéo.

**poster** : Détermine l'image à afficher pendant que la vidéo se charge ou jusqu'à ce que l'utilisateur appuie sur le bouton de lecture.

```
<audio controls><!-- controls est un attribut qui indique que des contrôles de lecture doivent être affichés -->
    <source src="audio.mp3" type="audio/mpeg">
        <!--Pointe vers la source de mon fichier audio, il peut être utilisé directement sur la balise audio ou video, ou sur un élément source à l'intérieur de ceux-ci.-->
        Votre navigateur ne supporte pas l'élément audio.
        <!--le texte "Votre navigateur ne supporte pas l'élément audio/vidéo" s'affiche si le navigateur de l'utilisateur ne prend pas en charge les balises audio et video.-->
    </audio>

<video width="320" height="240" controls><!-- controls est un attribut qui indique que des contrôles de lecture doivent être affiché. -->
    <!--Il est toujours recommandé de fournir plusieurs formats pour les médias audio et vidéo pour assurer la compatibilité entre les différents navigateurs.-->
    <source src="movie.mp4" type="video/mp4">
    <source src="movie.ogg" type="video/ogg">
    Votre navigateur ne supporte pas l'élément vidéo.
    <!--le texte "Votre navigateur ne supporte pas l'élément audio/vidéo" s'affiche si le navigateur de l'utilisateur ne prend pas en charge les balises audio et video.-->
</video>
```

# Le CSS



- [1 - Introduction](#)
- [2 - La syntaxe](#)
- [3 - Les sélecteurs](#)
- [4 - Les combinateurs](#)
- [5 - Les propriétés principales](#)
- [6 - Les pseudos classes](#)
- [7 - Structure des éléments](#)
- [8 - Flux et positionnement](#)
- [9 - Les éléments flottants](#)
- [10 - Les flexbox](#)

# Introduction au CSS

- ▶ Naissance du **CSS** pour accompagner le HTML **fin 1996**.
- ▶ Le **CSS** (**Cascading Style Sheets** : feuille de style en cascade) est un **langage de mise en forme**.
- ▶ **HTML** affiche et structure les données, **CSS** les met en forme (**positionnement, couleurs, polices, etc...**).
- ▶ Il est possible d'utiliser le CSS de **trois manières** différentes :
  - ▶ Dans un fichier externe avec l'extension **.css** avec un lien dans le **<head>** du fichier **.html** (méthode la plus courante):

```
> <link type="text/css" rel="stylesheet" href="styles.css">
```

- ▶ Directement **dans le code HTML (partie <head>)** avec la balise **<style> </style>** (méthode à éviter autant que possible).
- ▶ Directement **dans le code HTML sur l'élément à styliser** (méthode à éviter autant que possible). :

```
> <h1 color="red">Titre de niveau 1</h1>
```

# La Syntaxe

```
selecteur{  
    propriété : valeur ;  
}
```

Ici, le **sélecteur**, c'est l'élément HTML ciblé, exemple :

```
css      p{/*Sélecteur*/  
        color:red;  
        /*Propriété : valeur;*/  
    }
```

# Quelques sélecteurs CSS

(sert à appliquer des propriétés CSS sur les éléments HTML que l'on sélectionne)

Sélecteur d'élément ou balise HTML

Ici : Touchera toutes les balises <p></p> de la page web

CSS

```
p{ color:red; }
```

Sélecteur de class

Ici : Touchera la/les balises ayant la class = "text-red"

CSS

```
.text_red{ color:red; }
```

Sélecteur d'ID

Ici : Touchera la balise ayant l'id = "color\_main\_title"

CSS

```
#color_main_title{ color:brown; }
```

Sélecteurs universels

Ici : Touchera tous les éléments HTML de la page

CSS

```
*{ color:red; }
```

Sélecteur d'attribut

Ici : Touchera toutes les balises <a></a> ayant comme attribut :  
href = "https://google.com"

CSS

```
a[href="https://google.com"]{ color:green; }
```

# Quelques Combinateurs

(sert à combiner plusieurs sélecteurs)

Sélection en liste

Ici : Touchera toutes les balises <p></p> et la/les balises ayant la class = "text-red".

CSS

```
p, .text_red {  
    color: red;  
}
```

Sélection des enfants directs

Ici : Touchera toutes les balises <p></p> qui sont enfants directs d'une balise <div></div>.

CSS

```
div > p{  
    color: green;  
}
```

Sélection des enfants à n'importe quel niveau

Ici : Touchera toutes les balises <p></p> qui sont descendants d'une balise <div></div>..

CSS

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

Selection du voisin direct

Ici : Touchera toutes les balises <p></p> qui sont voisines d'une balise <div></div>..

CSS

```
div + p{  
    color: brown;  
}
```

Autres...

<https://www.w3.org/TR/selectors-3/#combinators>

# Principales propriété CSS

| Propriété                  | Syntaxe                       | Exemples   |
|----------------------------|-------------------------------|--|
| Couleur<br>couleur de fond | color<br>background-color     | color: red;   color: rgb(255, 99, 71);   color: #F10606;<br>background-color: gray;   background-color: #808080;<br>background-color: rgba(255, 99, 71, 0.80); |
| Police                     | font-family                   | font-family: Times;   font-family: Impact, Verdana, "Arial Black";   |
| Taille texte               | font-size                     | font-size: 15px;   font-size: small;   font-size: 1,5em;   |
| Styles de police           | font-style<br>font-weight ... | font-style: italic;   font-weight: bold;<br>text-decoration: underline;  |
| Alignment                  | text-align                    | text-align: center;   text-align: right;<br>text-align: justify;   |
| Marges                     | margin<br>padding             | margin: 5px 5px 5px 5px;   margin-top: 10px;   margin-bottom: 30px;  <br>padding: 10px;   padding-left: 15px;  |

# Principales propriété CSS

| Propriété                     | Syntaxe          | Exemples  |
|-------------------------------|------------------|---|
| Bordure                       | border           | <code>border: 1px solid black;<br/>/*épaisseur + type de trait + couleur du trait*/</code>  |
| Bordures arrondies            | border-radius    | <code>Border-radius: 10px;</code>   |
| Soulignements et autres décos | text-decoration  | <code>text-decoration:underline; ou line-through;<br/>ou overline; ou none;</code>  |
| Ombres block                  | box-shadow       | <code>box-shadow: 6px 6px 0px black;<br/>/*Générateur d'ombre sur le block*/</code>   |
| Hauteur, largeur              | height, width    | <code>height:400px;   width:300px;</code>   |
| Image de fond                 | background-image | <code>/* Élément qui aura une image de fond */<br/>.element{<br/>    /* Hauteur pour être sûr de voir l'image de fond */<br/>    height: 200px;<br/>    background-image: url('../images/chat.jpg');<br/>}</code> |

# Principales pseudos classes

| Syntaxe                                    | Exemples  |
|--|---|
| :hover                                     | <pre>/* La pseudo classe "hover" permet d'appliquer des styles uniquement si l'élément est survolé par la souris */ p:hover{     color: red; }</pre>                                  |
| :active                                    | <pre>/* La pseudo classe "active" permet d'appliquer des styles uniquement si l'élément est en train d'être cliqué */ p:active{     color: blue; }</pre>                              |
| :visited                                   | <pre>/* La pseudo classe ":visited" permet de sélectionner les liens ayant déjà été visités */ a:visited{     color: grey; }</pre>  |
| :link                                      | <pre>/* La pseudo classe ":link" permet de sélectionner les liens pas encore visités */ a:link{     color: gold; }</pre>  |
| :first-child / :last-child / :nth-child(x) | <pre>/* Sélectionne uniquement le premier enfant de l'élément parent (dans cet exemple le premier li de chaque ul) */ ul &gt; li:first-child{     color: pink; }</pre>                |
| :not                                       | <pre>/* Sélectionne tous les éléments ne correspondant pas au test (dans cet exemple, sélectionne tous les h2 n'ayant pas la classe "red") */ h2:not(.red){     color: green; }</pre> |

# Exercice 01

Reprendre l'exercice 02 html, en faire une copie pour le transformer en ajoutant du css afin de styliser la page.

## Ma base client

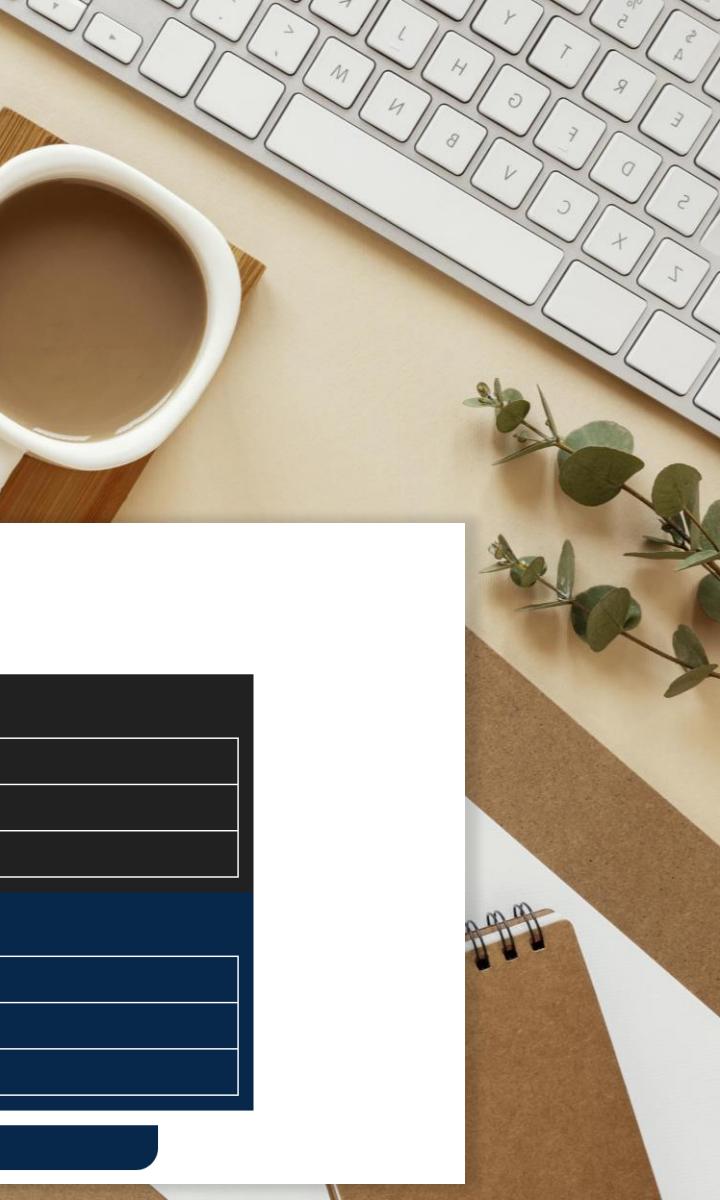
### Description

Cette application permet de faire de la gestion de **commande** et de **client**.

| Clients        |
|----------------|
| Jean Dupont    |
| Sarah Abdram   |
| Rachelle Taeck |

| Commandes |
|-----------|
| CMD01     |
| CMD02     |
| CMD03     |

Lien vers le site de freepik



# Structure des éléments

Modèle de boîte : tout élément est inclus dans une boîte, aide à comprendre la mise en page CSS et le positionnement des éléments d'une page HTML.

## ► Élément HTML de type Block

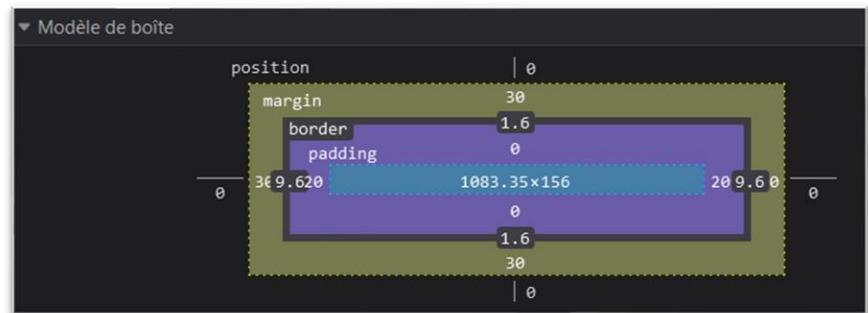
- Occupe toute la largeur disponible de son parent
- L'élément suivant ou précédent l'élément block se positionnera au dessus ou en dessous de celui-ci (retour à la ligne)
- Width & Height modifiables

Exemple de Type block générique: `<h1>...<h6>` / `<p>` / `<div>`

## ► Élément HTML de type Inline

- Occupe la place de leurs contenu
- L'élément inline suivant se positionne à la suite
- Width & Height non modifiables
- et pas de padding et margin (top et bottom)

Exemple de Type inline générique: `<a>` / `<span>`



## ► Élément HTML de type Inline-block

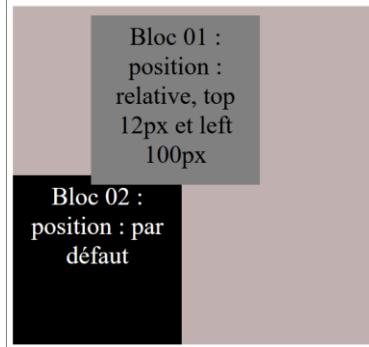
- Se comporte en partie comme un inline :  
Occupe la place de leurs contenu / L'élément inline suivant se positionne à la suite
- Se comporte en partie comme un block :  
Width & Height modifiables et padding et margin (top et bottom) possible

Exemple de Type inline-block générique: `<button>`

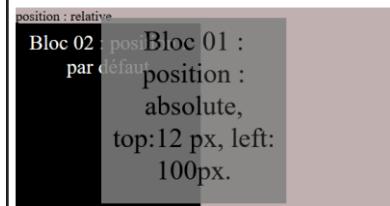
# Flux et Positionnement

- ▶ Le **flux HTML** correspond à l'écoulement des informations (ou données) que le navigateur va interpréter et afficher. **Un navigateur commence par le haut de la page**, place les éléments HTML qu'il rencontre les unes à la suite des autres, **de gauche à droite puis de haut en bas**, à ceci près que cela dépend si ce sont des balises bloc ou en-ligne.
- ▶ La propriété CSS **position** permet de **gérer le positionnement** des éléments.
- ▶ On distingue **4 types principaux** de positionnement :
  - ▶ **(static)**: Valeur par défaut - Positionnement **statique** des éléments **dans le flux**
  - ▶ **relative** : Positionnement ajustée par rapport à la **position initiale** de l'élément **dans le flux**
  - ▶ **absolute** : Positionnement ajustée par rapport au **parent relative** le plus proche **hors flux** (chevauchement d'éléments alors possible)
  - ▶ **fixed** : **semblable a absolute**, mais en s'appuyant sur la **fenêtre visible** du navigateur et **hors flux**.
- ▶ Positionnement ajustée avec les propriétés **top**, **left**, **bottom** et **right**. Les valeurs négatives sont possibles.
- ▶ La propriété **z-index** permet de prioriser l'élément à afficher (en cas de chevauchement)

## Positionnement relatif



## Positionnement absolue



# Les éléments flottants

La propriété CSS **float** permet de sortir un élément du flux normal du HTML pour le placer à gauche ou à droite de son conteneur. Les éléments inlines présents dans le même conteneur (comme les textes) entoureront cet élément.

## DEV FRONT END & DEV BACK END

### Le développeur Front-End

Lorsque l'on parle de «Front-End», il s'agit finalement des éléments du site que l'on voit à l'écran et avec lesquels on peut interagir. Ces éléments sont composés de HTML, CSS et de Javascript contrôlés par le navigateur web de l'utilisateur. Les champs de compétence du Front-End peuvent être séparé en deux : - Le design - Le développement HTML, CSS, Javascript. Le design est traditionnellement réalisé par un web designer qui produit des maquettes graphiques à l'aide de Photoshop ou Fireworks. Cependant, de plus en plus de web designers ont franchi la barrière et savent coder en HTML et CSS. Dans certains cas ils sont aussi capables de produire du Javascript.



### Le développeur Back-End



heures sur 24, sur lequel les pages du site web sont enregistrées.

Le Back-End, c'est un peu comme la partie immergée de l'iceberg. Elle est invisible pour les visiteurs, mais représente une grande partie du développement d'un projet web. Sans elle, le site web reste une coquille vide. On peut décomposer le Back-End en trois parties essentielles : - Un serveur (ou hébergement web) - Une application (en l'occurrence le site web) - Une base de données (ou l'on stocke les données de l'application) Le serveur est comme un disque dur accessible 24

## Exercice 02

Créer 2 pages web :

- Page contact
- Page new-contact

Voici les codes de couleur :

- Noir : #222222
- Vert : #578957
- Vert foncé : #406040
- Jaune : #c7b878

Aidez vous des 2 maquettes des 2 pages à reproduire qui vous seront fournit



# Les FlexBox

Flexbox est un modèle de boîte flexible permettant de gérer le positionnement des éléments dans un conteneur d'une manière plus fluide qu'avec les modèles de boîte classique.  
Pour qu'un conteneur devienne un conteneur flex, il faut changer son type.

Il y a **4 directions possibles** pour **distribuer les enfants** d'un élément flex :

- row** : horizontale, c'est la valeur par défaut
- column** : verticale
- row-reverse** : horizontale inversé
- column-reverse** : verticale inversé

Il est possible de gérer l'alignement sur **l'axe principal** avec la propriété "**justify-content**" et sur **l'axe secondaire** avec la propriété "**align-items**" :

L'axe **principal** et **secondaire** varie en fonction du **paramétrage** de la **direction** de votre flexbox :

Si "flex-direction" vaut "**row**" ou "**row-reverse**", alors l'axe **principal** sera **l'axe horizontal** et l'axe **secondaire** sera **l'axe vertical**.

Si "flex-direction" vaut "**column**" ou "**column-reverse**", alors l'axe **principal** sera **l'axe vertical** et l'axe **secondaire** sera **l'axe horizontal**.

Les valeurs possibles pour les **alignements** sont les suivantes :

**flex-start** : éléments positionnés au début de l'axe, c'est la valeur par défaut.

**flex-end** : éléments positionnés à la fin de l'axe.

**center** : éléments centrés au milieu de l'axe.

**space-between** : les éléments sont espacés à égale distance sur tout l'axe (et sont collés aux "bords" de l'axe).

**space-around** : les éléments sont espacés à égale distance sur tout l'axe (et ne sont pas collés aux "bords" de l'axe).

```
css      div{ display: flex; }
```

```
Css      div{ display: flex; /* Tous les éléments enfants seront disposés en ligne côté à côté */ flex-direction: row; }
```

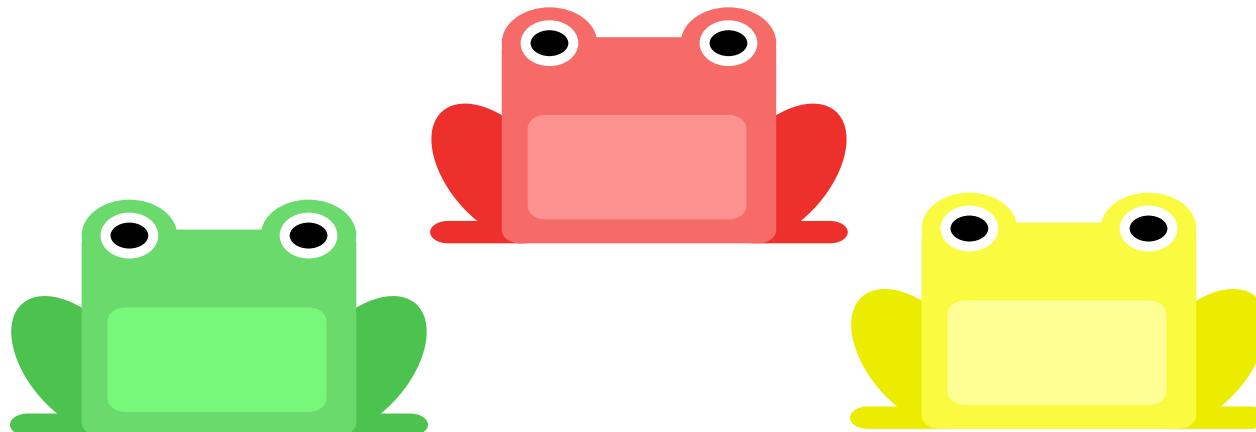
```
Css      div{ display: flex; /* Tous les éléments enfants seront disposés en ligne côté à côté */ flex-direction: row; /* Si il y a un manque de place, les éléments passeront à la ligne du dessous */ flex-wrap: wrap; /* Les éléments seront centrés sur l'axe principal (sur l'axe horizontal dans cet exemple) */ justify-content: center; /* Les éléments seront centrés sur l'axe secondaire (sur l'axe vertical dans cet exemple) */ align-items: center; }
```

# Entraînement Flexbox : Jeux Flexbox Froggy

Outil pour tester les propriétés css qui existent :

<https://codepen.io/enxaneta/full/adLPwv/>

Lien vers le jeu : <https://flexboxfroggy.com/#fr>



# Le responsive design



- [1 - Adapter un site web](#)
- [2 - Le responsive web design](#)
- [3 - Les media queries](#)

# Comment adapter un site web à la taille de l'écran

Pour commencer une page web sur mobile apparaîtra toute petite à cause de la très petite taille de l'écran d'un smartphone. Pour remettre le tout à l'échelle, il faut intégrer cette balise dans le head de la page web :

```
html <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

**Responsive Web Design** désigne le fait qu'un site web est **construit de manière à le rendre compatible, lisible et utilisable sur différentes tailles d'écrans.**

Il existe plusieurs techniques permettant de faire du responsive. Une technique qui se faisait beaucoup était de créer deux site distincts : un pour desktop et un pour mobile. Cette technique a été très utilisée dans le début des années 2010.

Exemple avec Wikipédia :

Version Desktop : <https://fr.wikipedia.org/wiki/France>

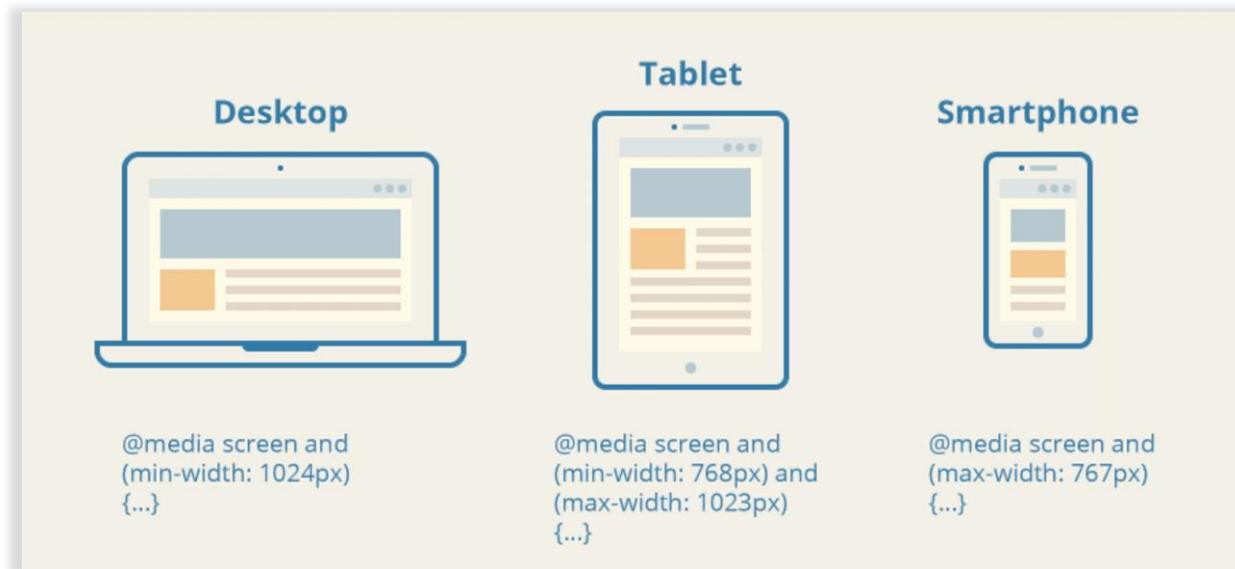
Version mobile : <https://fr.m.wikipedia.org/wiki/France>

# Responsive web design

La **responsive web design** est une approche qui permet de créer des pages web qui sont toujours bien rendu quelque soit la taille du périphérique d'affichage.

En CSS nous allons mettre en place des sites responsives grâce aux **média queries**.

Les **media queries** sont des conditions CSS permettant d'appliquer des styles CSS seulement si ces conditions sont remplies. Ces conditions portent toujours sur les caractéristiques de l'agent utilisateur utilisé pour rendre la page web (navigateur, imprimante, lecteur d'écran, etc...)



# Mise en place des media queries

CSS

```
@media screen and (min-width:480px) {  
    /* Mon code css ici avec Sélecteurs, propriétés et valeurs */  
}  
  
@media screen and (min-width:768px) {  
    /* Mon code css ici avec Sélecteurs, propriétés et valeurs */  
}  
  
@media screen and (min-width:992px) {  
    /* Mon code css ici avec Sélecteurs, propriétés et valeurs */  
}  
  
@media screen and (min-width:1200px) {  
    /* Mon code css ici avec Sélecteurs, propriétés et valeurs */  
}
```

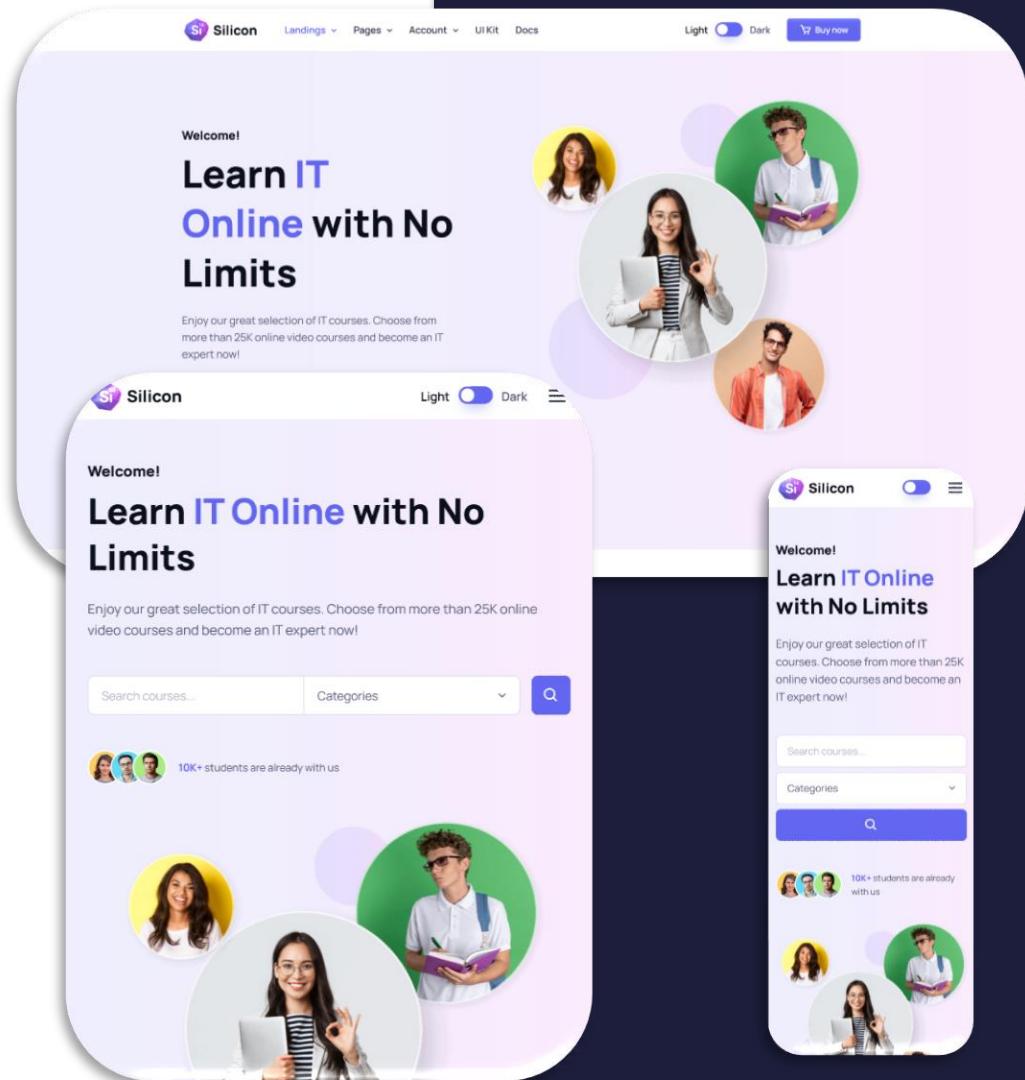
# Exercice 01

À partir du TP02 réalisé précédemment pour la page de la liste des contacts, adapter le pour avoir le même aperçu que les maquettes qui vous seront donné en utilisant les media queries



# { B } Bootstrap

- [1 - Bootstrap](#)
- [2 - Installation](#)
- [3 - Le système de grille](#)
- [4 - La grille responsive](#)
- [5 - Les breakpoints](#)
- [6 - Les outils](#)



# Bootstrap

Bootstrap, qu'est-ce que c'est ?

Bootstrap est un **Framework Front-end**. Ce dernier fournit des outils déjà préparés pour concevoir rapidement des pages web (des boutons, fenêtre modales, tableaux, etc...).

Ce framework est un des projets les plus populaires sur Github. Il a **été développé en 2010 par Twitter** et est actuellement en **version 5.2**

Permet :

- mise en page responsive
- compatibilité des propriétés avec tous les navigateurs
- utiliser des composants Bootstrap (modal, carousel, etc.)
- icônes

# Installation

## Comment l'installer ?

Pour inclure Bootstrap, soit on le fait **en CDN** (à distance), soit en téléchargeant **les fichiers en local**.

En **CDN** :

html

```
<!-- Le fichier CSS doit être inclus avant les autres fichiers CSS -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet"
      integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSm07SnpJef0486qhLnuZ2cdeRh002iuK6FUUVM" crossorigin="anonymous">

<!-- Le fichier JS doit toujours être inclus avant la fermeture de la balise </body> -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGd1KrxdiJJigb/j/68SIy3Te4Bkz" crossorigin="anonymous"></script>
```

En **téléchargeant les fichiers en local**, il y a 2 fichiers à inclure dans votre dossier css et js:

- **bootstrap.min.css** : partie CSS du framework
- **bootstrap.bundle.min.js** : partie JS du framework (la version bundle inclut aussi popper.js qui est une bibliothèque dont Bootstrap a besoin. Si vous ne mettez pas la version bundle, vous devrez aussi inclure cette bibliothèque manuellement.)
- Lien vers installation de Bootstrap : <https://getbootstrap.com/docs/5.3/getting-started/download/>

# Le système de grille de Bootstrap

Ce qui a beaucoup participé à faire connaître Bootstrap, c'est son **système de grille en 12 colonnes** qui permet de facilement pouvoir agencer une page web. Quand on crée un bloc, sa **taille** est définie via un **nombre de colonne** sur un **total de 12** (6 colonnes = 50% du parent par exemple.).

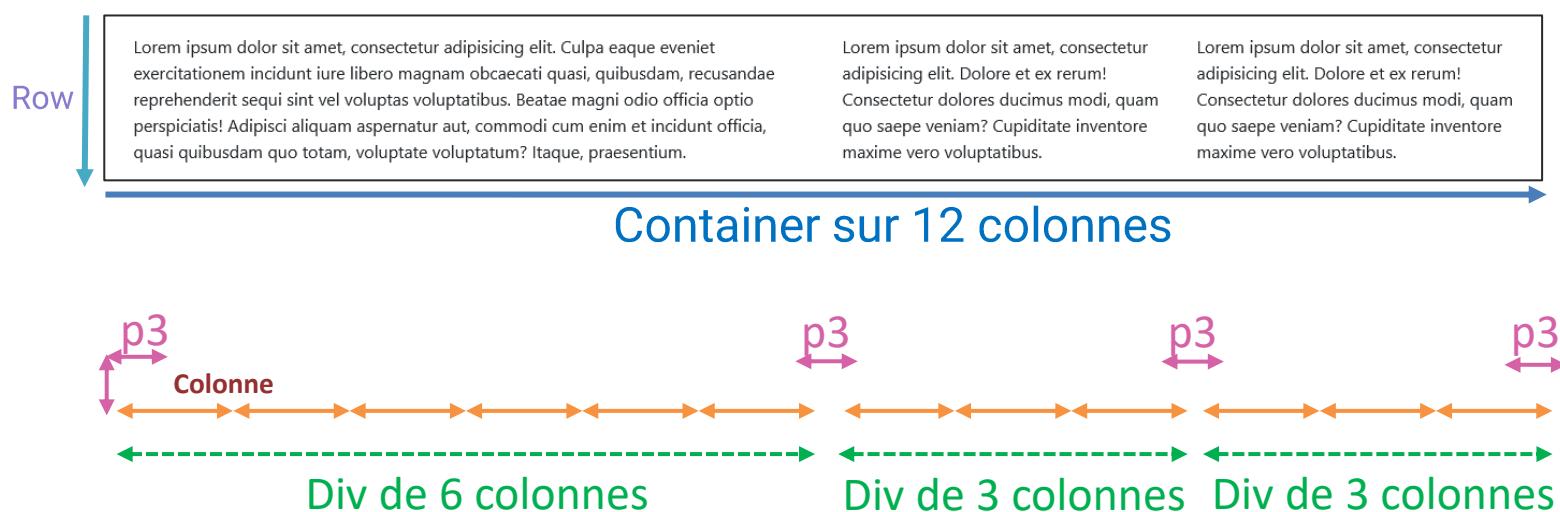
Exemple :

html

```
<!-- Le conteneur générique de bootstrap (les éléments enfants auront la classe .row) -->
<div class="container">
<!-- Crée une ligne (row) d'éléments sur 12 colonnes (ne peut avoir que des éléments ayant une classe col-x en enfants directs !) -->
    <div class="row">
        <!-- Ce bloc fera une taille de 50% (col-6) de son parent, avec une marge intérieur de 3 (p-3) -->
        <div class="col-6 p-3">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Culpa eaque eveniet exercitationem
incident iure libero magnam obcaecati quasi, quibusdam, recusandae reprehenderit sequi sint vel voluptas voluptatibus. <div>
            <!-- Ce bloc fera une taille de 25% (col-3) de son parent, avec une marge intérieur de 3 (p-3) -->
            <div class="col-3 p-3">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Dolore et ex rerum! Consectetur
dolores ducimus modi.</div>
            <!-- Ce bloc fera une taille de 25% (col-3) de son parent, avec une marge intérieur de 3 (p-3) -->
            <div class="col-3 p-3">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Dolore et ex rerum! Consectetur
dolores ducimus modi, quam quo saepe veniam? Cupiditate inventore maxime vero voluptatibus.</div>
        </div>
    </div>
```

# Le système de grille de Bootstrap

Résultat visuel du code précédent :



# La grille responsive de Bootstrap

Encore plus poussé, il est possible de définir la taille des éléments en **nombre de colonnes ET en fonction de la taille de l'écran !**

Exemple :

html

```
<div class="container-fluid">
    <div class="row">
        <!-- Voir explication, en dessous --&gt;
        &lt;div class="col-12 col-sm-8 offset-sm-2 col-xl-6 offset-xl-3"&gt;Lorem ipsum dolor sit amet, consectetur
adipisicing elit. Animi consequatur dignissimos dolor dolorum eveniet facilis, ipsam itaque laboriosam, magni
minima nam necessitatibus non obcaecati quidem saepe, soluta temporibus tenetur vero? A assumenda delectus
dignissimos ea eaque eligendi eos esse facilis fugiat harum inventore ipsam iste libero maiores, minus modi
natus nobis non, optio perspiciatis provident quibusdam quis quisquam quo ratione rem repellat similique sit
suscipit temporibus totam ut veritatis voluptatem! Ab atque consequatur dolore doloribus eius eos error eveniet
fugiat hic, itaque iusto libero natus nisi non placeat quasi qui quia reiciendis rerum saepe sint vero
voluptates! Placeat, quisquam velit?&lt;/div&gt;
    &lt;/div&gt;
&lt;/div&gt;</pre>
```

Dans l'exemple ci-dessus, le bloc est défini en **3 tailles différentes en fonction de 3 scénarios possibles :**

**col-12** : signifie que si l'écran fait **entre 0px et l'infinie**, le bloc fera **12 colonnes** de large (100% de son parent)

**col-sm-8** : signifie que si l'écran fait **entre 576px et l'infinie**, le bloc fera **8 colonnes** de large (66% de son parent)

**col-xl-8** : signifie que si l'écran fait **entre 1200px et l'infinie**, le bloc fera **8 colonnes** de large (66% de son parent)

Les offsets sont des décalages en nombre de blocs vers la droite :

**offset-sm-2** : signifie que si l'écran fait **entre 576px et l'infinie**, le bloc **se décalera de 2 colonnes** vers la droite

**offset-xl-3** : signifie que si l'écran fait **entre 1200px et l'infinie**, le bloc **se décalera de 3 colonnes** vers la droite

Note : si le nombre rattaché au col n'est pas préciser, les éléments auront la même largeur.

# Les breakpoints de Bootstrap

Il existe **6 breakpoints** dans Bootstrap qui s'activeront si la largeur de la fenêtre est comprise entre :

- 0px à l'infinie : **sans nom** (`col-12` et `offset-2` par exemple)
- 576px à l'infinie : **sm** (`col-sm-8` et `offset-sm-2` par exemple)
- 768px à l'infinie : **md** (`col-md-10` et `offset-md-1` par exemple)
- 992px à l'infinie : **lg** (`col-lg-12` par exemple)
- 1200px à l'infinie : **xl** (`col-xl-12` par exemple)
- 1400px à l'infinie : **xxl** (`col-xxl-12` par exemple)

Si vous avez bien regardé, les intervalles de pixels se superposent. La règle est simple, le **breakpoint** le plus élevé prend le dessus sur les plus petit si l'écran est plus large que lui !

Ainsi si par exemple un bloc possède la classe `col-8` et la classe `col-xl-6`, si l'écran fait **500 pixels**, l'élément fera bien **8 colonnes**. par contre si l'écran fait **1920 pixels** de larges, dans ce cas le bloc fera **6 colonnes** de large.

Doc sur les **breakpoints** : <https://getbootstrap.com/docs/5.3/layout/breakpoints/>

# Les outils de Bootstrap

Bootstrap possède beaucoup d'outils intégrés très pratiques, et ils sont tous référencés avec des exemples et avec le code de ces exemples sur la documentation officielle ici : <https://getbootstrap.com/docs/5.3/getting-started/introduction/>

Liste non exhaustive d'outils pratiques :

- Les alertes : <https://getbootstrap.com/docs/5.3/components/alerts/>
- Les navbars : <https://getbootstrap.com/docs/5.3/components/navbar/>
- Les tableaux : <https://getbootstrap.com/docs/5.3/content/tables/>
- Les fenêtres modales : <https://getbootstrap.com/docs/5.3/components/modal/>
- Les formulaires : <https://getbootstrap.com/docs/5.3/forms/overview/>
- La typographie : <https://getbootstrap.com/docs/5.3/content/typography/>
- Les images : <https://getbootstrap.com/docs/5.3/content/images/>
- Les accordéons : <https://getbootstrap.com/docs/5.3/components/accordion/>
- Les boutons : <https://getbootstrap.com/docs/5.3/components/buttons/>
- Les cards : <https://getbootstrap.com/docs/5.3/components/card/>

# Les outils de Bootstrap

Suite liste non exhaustive d'outils pratiques :

- Les carrousels : <https://getbootstrap.com/docs/5.3/components/carousel/>
- Les listes : <https://getbootstrap.com/docs/5.3/components/list-group/>
- Les paginations : <https://getbootstrap.com/docs/5.3/components/pagination/>
- Les popovers : <https://getbootstrap.com/docs/5.3/components/popovers/>
- Les icônes de chargement : <https://getbootstrap.com/docs/5.3/components/spinners/>
- Les textes colorés : <https://getbootstrap.com/docs/5.3/utilities/colors/>
- Les fonds colorés : <https://getbootstrap.com/docs/5.3/utilities/background/>
- Les bordures : <https://getbootstrap.com/docs/5.3/utilities/borders/>
- Les alignements de textes : <https://getbootstrap.com/docs/5.3/utilities/text/>

# Exercice 02

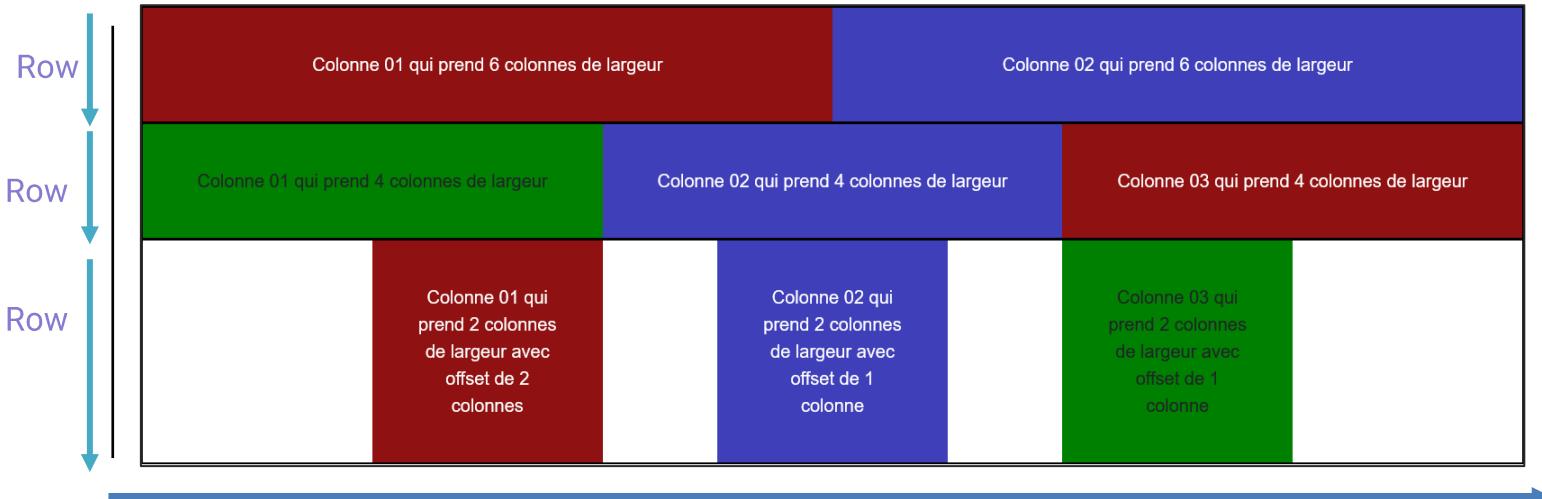
Créer une page (index.html) avec un fichier styles.css qui se trouvera dans un dossier css :

- Un titre d'onglet “Exercice 02”
- L'intégration de Bootstrap à votre page
- Le même visuel que sur le screen ci-après



# Exercice 02

index.html



Page avec un Container-fluid sur 12 colonnes

# Exercice 03

Créer une page (index.html) avec un fichier styles.css qui se trouvera dans un dossier css, on y trouvera aussi :

- Un titre d'onglet “Exercice 03”
- L'intégration de Bootstrap à votre page
- Le même visuel sur les pages qui suivent
- Attention, les visuels représentent une seule même page sur lesquels on applique les class css de Bootstrap pour rendre la page responsive



# Exercice 03

Visuel à partir de 0px

The image shows four identical article cards arranged vertically. Each card has a dark blue header with the word "Article" in white. Below the header is a block of placeholder text: "Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quos, placeat! At asperiores ex doloremque minima, ipsa odit. Nam accusantium ut neque. Vitae ducimus aliquid eum quos quae sequi maiores maxime." The cards are separated by thin white spaces.

Visuel de la page à partir de 800px

The image shows four identical article cards arranged horizontally in a single row. Each card has a dark blue header with the word "Article" in white. Below the header is a block of placeholder text: "Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quos, placeat! At asperiores ex doloremque minima, ipsa odit. Nam accusantium ut neque. Vitae ducimus aliquid eum quos quae sequi maiores maxime." The cards are separated by thin white spaces.

# Exercice 03

Visuel de la page à partir de 1250px

## Article

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quos, placeat! At asperiores ex doloremque minima, ipsa odit. Nam accusantium ut neque. Vitae ducimus aliquid eum quos quae sequi maiores maxime.

## Article

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quos, placeat! At asperiores ex doloremque minima, ipsa odit. Nam accusantium ut neque. Vitae ducimus aliquid eum quos quae sequi maiores maxime.

## Article

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quos, placeat! At asperiores ex doloremque minima, ipsa odit. Nam accusantium ut neque. Vitae ducimus aliquid eum quos quae sequi maiores maxime.

## Article

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quos, placeat! At asperiores ex doloremque minima, ipsa odit. Nam accusantium ut neque. Vitae ducimus aliquid eum quos quae sequi maiores maxime.

# Les fondamentaux de l'accessibilité numérique

1 - Le RGAA

2 - Les bonnes pratiques

3 - La sémantique et les attributs

# Le RGAA (Référentiel Général d'Accessibilité pour les Administrations )

Le Référentiel Général d'Accessibilité pour les Administrations (**RGAA**) est un cadre de référence qui définit les règles pour rendre les services numériques accessibles. Il a été mis en place par le **gouvernement français** en application de l'**article 47 de la loi du 11 février 2005** pour **l'égalité des droits et des chances, la participation et la citoyenneté des personnes handicapées**.

L'**objectif** du RGAA est de **rendre les sites web et les applications mobiles accessibles** à tous, y compris aux personnes ayant un handicap. Cela comprend, par exemple, les personnes malvoyantes ou aveugles, les personnes malentendantes ou sourdes, les personnes ayant des troubles cognitifs ou les personnes ayant des difficultés motrices.

Le RGAA est basé sur les **normes internationales d'accessibilité web WCAG** (Web Content Accessibility Guidelines) élaborées par le W3C (World Wide Web Consortium). Il contient une série de critères et de tests spécifiques qui doivent être respectés pour assurer l'accessibilité.

Il est **obligatoire** pour les services en ligne des administrations de l'État, des collectivités territoriales et des établissements publics qui en dépendent. Il est également recommandé pour toutes les autres organisations qui souhaitent améliorer l'accessibilité de leurs services numériques.

Par exemple, il exige que toutes les **images aient un texte alternatif** (alt) pour que les personnes qui utilisent un lecteur d'écran puissent comprendre ce que l'image représente :

html

```

```

Le **RGAA** couvre aussi d'autres aspects, comme l'accessibilité du contenu vidéo et audio (par exemple, fournir des sous-titres pour les vidéos et des transcriptions pour l'audio), la navigation au clavier (pour les personnes qui ne peuvent pas utiliser une souris), la **structure de la page** (utiliser correctement les titres et les en-têtes), le contraste des couleurs, etc.

# Les bonnes pratiques dans mes pages html

Les bonnes pratiques de construction de pages HTML comprennent un certain nombre de principes clés pour assurer la lisibilité, l'accessibilité, et la performance. En voici quelques-unes :

- **Structurer votre HTML** : Utilisez les balises HTML appropriées pour indiquer la structure de votre contenu. Par exemple, utilisez les éléments `<header>`, `<footer>`, `<nav>`, `<section>`, `<article>`, et `<aside>` pour structurer votre page, et utilisez `<h1>`, `<h2>`, etc., pour indiquer les titres et les sous-titres.
- **Utilisez les balises sémantiques** : Les balises sémantiques fournissent des informations sur le type de contenu qu'elles contiennent. Par exemple, utilisez `<strong>` au lieu de `<b>` pour indiquer un texte important, et utilisez `<em>` au lieu de `<i>` pour indiquer un texte accentué.
- **Fournir des textes alternatifs pour les images** : Utilisez l'attribut alt dans vos balises `<img>` pour fournir une description textuelle des images. Cela est essentiel pour l'accessibilité et peut aider à améliorer le référencement de votre site.
- **Utilisez les attributs de balise appropriés** : Les attributs de balise fournissent des informations supplémentaires sur les éléments HTML. Par exemple, l'attribut lang dans la balise `<html>` indique la langue principale de la page.
- **Eviter les balises obsolètes** : Certaines balises HTML, comme `<font>` et `<center>`, sont obsolètes et ne doivent pas être utilisées. Utilisez plutôt le CSS pour le style et la mise en page.
- **Valider votre HTML** : Utilisez un validateur HTML, comme le validateur HTML du W3C, pour vérifier votre code et vous assurer qu'il est bien formé et sans erreurs.
- **Gardez votre code lisible et organisé** : Cela comprend l'indentation de votre code, l'utilisation de commentaires pour expliquer les sections de code complexes, et le maintien d'une structure cohérente tout au long de votre projet.
- **Responsive Design** : Assurez-vous que votre site est responsive, c'est-à-dire qu'il est conçu pour s'adapter à toutes les tailles d'écran et tous les appareils. Cela implique souvent l'utilisation de CSS pour ajuster la mise en page en fonction de la taille de l'écran.

Ces pratiques sont essentielles pour créer des sites web de haute qualité qui sont accessibles, performants, et faciles à maintenir. m2formation.fr

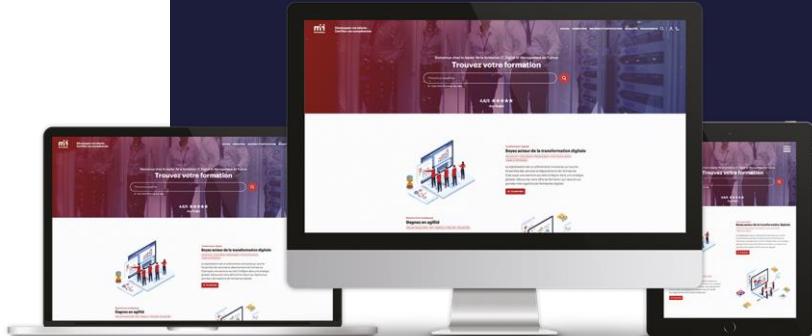
# La sémantique et les attributs essentiels à l'accessibilité

La **sémantique HTML** se réfère à l'utilisation d'éléments HTML pour **donner du sens et du contexte à votre contenu**. Chaque élément HTML a une signification inhérente qui transmet une certaine information à la fois aux navigateurs web et aux technologies d'assistance telles que les lecteurs d'écran.

Par exemple, l'**élément <h1>** est utilisé pour le **titre principal d'une page**, tandis que **<p>** est utilisé pour **le texte standard du paragraphe**. Utiliser ces éléments de manière appropriée permet aux utilisateurs et aux technologies d'assistance de comprendre la structure et le contenu de la page.

En ce qui concerne **l'accessibilité**, plusieurs **attributs HTML** sont essentiels pour rendre le contenu accessible à tous les utilisateurs, y compris ceux qui utilisent des technologies d'assistance :

- **Alt** : L'attribut alt est utilisé avec la balise `<img>` pour fournir une description textuelle des images, ce qui est crucial pour les utilisateurs malvoyants qui utilisent des lecteurs d'écran.
- **Lang** : L'attribut lang dans la balise `<html>` indique la langue principale de la page, ce qui aide les technologies d'assistance à lire le contenu avec la bonne prononciation.
- **Label** : L'attribut label est utilisé avec les éléments de formulaire pour fournir une description textuelle de leur fonction, ce qui est important pour tous les utilisateurs, mais en particulier pour ceux qui utilisent des lecteurs d'écran.
- **Tabindex** : L'attribut tabindex permet de contrôler l'ordre de navigation au clavier sur une page web, ce qui est crucial pour les utilisateurs qui ne peuvent pas utiliser une souris.
- **Role** : L'attribut role définit le rôle d'un élément HTML, aidant les technologies d'assistance à comprendre comment interagir avec lui.
- **Aria-** : Les attributs ARIA (Accessible Rich Internet Applications) sont utilisés pour améliorer l'accessibilité des applications web interactives et des éléments complexes.



# Fin du module

---



m2iformation.fr