

CSS

Cascading Style Sheets

I. Présentation du CSS

- A. Création du CSS
- B. Comment fonctionne le CSS ?
- C. La syntaxe
- D. Les outils

II. Les sélecteurs

- A. Héritage et style par défaut
- B. Class et id
- C. Les priorités
- D. Cibler toujours plus ...
- E. Pseudo-classes / pseudo-éléments

III. Le style

- A. Les unités de mesure
- B. Les polices
- C. Les couleurs
- D. Les backgrounds

IV. Le système de boîtes

- A. Block ou inline
- B. Margin et padding
- C. Les bordures
- D. Les débordements

V. Positionnement des éléments

- A. Sortir du flux
- B. Float - clear
- C. Position
- D. Empilement et z-index

VI. Responsive Design

- A. Display
- B. Flexbox
- C. Grid Layout
- D. Media queries

VII. Bootstrap

- A. Présentation générale
- B. La grille Bootstrap
- C. Style et mise en page
- D. Les composants

Présentation du CSS

Création du CSS



- ❏ Le langage **CSS** est créé en 1996 par le **W3C** (World Wide Web Consortium)
- ❏ La **version 3** est développée depuis 1999
- ❏ Le **CSS** est utilisé pour travailler le style et la mise en forme d'une page web

Comment fonctionne le CSS ?

- ❑ Le **CSS** consiste à **déclarer** un ensemble de **règles de style** ou de **mise en page** à appliquer à des **éléments HTML** seuls ou à des **groupes** d'éléments
- ❑ Ces éléments **HTML** sont **ciblés**, individuellement ou en groupe, à l'aide de **sélecteurs**
- ❑ C'est le **navigateur** qui va **interpréter** et **appliquer** l'ensemble de ces **règles** dans la **page web**
- ❑ Tout comme en **HTML**, il est possible de **commenter** son **code** en **CSS** pour **renseigner** ou **désactiver** une partie du **code** :

`/* commentaire non lu par le navigateur */`

La syntaxe du CSS

- ❏ Une règle **CSS** est composée d'un **sélecteur** suivi d'une déclaration de **style**
`sélecteur { déclaration de style; }`
- ❏ Le **sélecteur** cible un ou plusieurs éléments **HTML** en le(s) nommant
`p { déclaration de style; }`
- ❏ La déclaration de style contient une **propriété** suivie de sa **valeur**
`p {
 color: red;
}`

Comment fonctionne le CSS ?

- ❏ Le **CSS** peut être écrit :
 - directement à l'intérieur de la balise d'un élément **HTML**, avec l'attribut `style` 😡
 - dans le fichier **HTML**, à l'intérieur d'une balise `<style>` dans le `<head>` 😞
 - dans un fichier séparé, avec l'extension `.css` 😊
- ❏ Un fichier **CSS** est appelé depuis le fichier **HTML** grâce à une balise `<link>` dans le `<head>`

```
<link rel="stylesheet" href="monFichier.css" />
```
- ❏ Le même fichier **CSS** peut être appliqué à différentes pages **HTML**

La syntaxe du CSS

- ❏ Comment déclarer le **CSS** directement dans la **balise ouvrante** d'un élément **HTML** :

```
<h1 style="color: red"> ... </h1>
```

- ❏ Comment déclarer le **CSS** dans une balise **<style>** dans le **<head>** du fichier **HTML** :

```
<style>  
  h1 {  
    color: red;  
  }  
</style>
```

- ❏ Comment déclarer le **CSS** à l'intérieur d'une **feuille de style** séparée (fichier **.css**) :

```
h1 { color: red; }
```

La syntaxe du CSS

- ❑ La même **déclaration** peut être appliquée à **plusieurs** éléments :

```
h1, h2 {  
    color : red;  
}
```

- ❑ **Plusieurs** déclarations peuvent être appliquées à un **même élément** :

```
h1 {  
    color : red ;  
    font-size : 20px ;  
}
```

Les outils

- ❏ L'**inspecteur** fourni dans les Outils de développement du **navigateur** (via la touche **f12**)
- ❏ Le **validateur** de syntaxe **CSS** : <https://jigsaw.w3.org/css-validator/>
- ❏ La mise à jour des **compatibilités** des propriétés **CSS** selon les navigateurs :
 - <https://caniuse.com/>
 - <https://developer.mozilla.org/fr/color#compatibilite-navigateurs/>
- ❏ Des **mémos** détaillés des principaux **concepts** en **CSS** : <https://css-tricks.com/guides/>

Les sélecteurs

Héritage et style par défaut

- ❏ Les éléments **HTML** contenant d'autres éléments **HTML** sont appelés des éléments **parents**
- ❏ Les **éléments** qu'ils contiennent sont alors appelés des éléments **enfants**, puis petits-enfants...
- ❏ Un système d'**héritage** de certaines règles **CSS** est établi entre les éléments et des **propriétés CSS** sont ainsi **transmises** des parents aux enfants puis aux petits-enfants...
- ❏ <https://www.alsacreations.com/tuto/545-Comprendre-l-heritage-et-la-parente-des-styles-CSS.html>

Héritage et style par défaut

- ❑ Cette première règle sera appliquée au contenu des balises `<main>`, puis `<h2>`, puis ``, puis `` :

```
body {  
    color : red ;  
}
```

- ❑ Mais cette deuxième règle va "écraser" la première pour le dernier élément ``. Savez-vous pourquoi ?

```
li {  
    color : blue ;  
}
```



```
<body>  
  <main>  
    <h2>titre h2</h2>  
    <ul>  
      <li>list item</li>  
      <li>list item</li>  
    </ul>  
  </main>  
</body>
```

Héritage et style par défaut

- ❑ Certaines **propriétés** ne peuvent pas être héritées, comme les propriétés **background-image** ou **border** par exemple
- ❑ La plupart des éléments possèdent également des **styles par défaut**, comme les propriétés **margin**, **font-size**, etc ...
- ❑ Ces **styles** peuvent bien sûr être **changés** avec une déclaration de règles **CSS** ciblant ces éléments
- ❑ Il est aussi courant d'ajouter, en plus du fichier **CSS**, un fichier **reset.css** qui viendra **réinitialiser** ces déclarations de style **par défaut**
- ❑ <https://www.alsacreations.com/astuce/lire/36-la-technique-du-reset-css.html>

Les attributs **class** et **id**

- ❑ On donne généralement un même attribut **class** à plusieurs **éléments** auxquels on souhaite appliquer le même **style**

```
<p class="nom-classe" > ... </p>
```

- ❑ Pour cibler cette **classe** en **CSS**, on la fait précéder d'un **point** :

```
.nom-classe { color : blue }
```

- ❑ Pour cibler **un seul** élément auquel appliquer un style **unique**, on lui donne un attribut **id** :

```
<p id="nom-id" > ... </p>
```

- ❑ Pour cibler cet **id** en **CSS**, on le fait précéder d'un **#** :

```
#nom-id { color : blue }
```

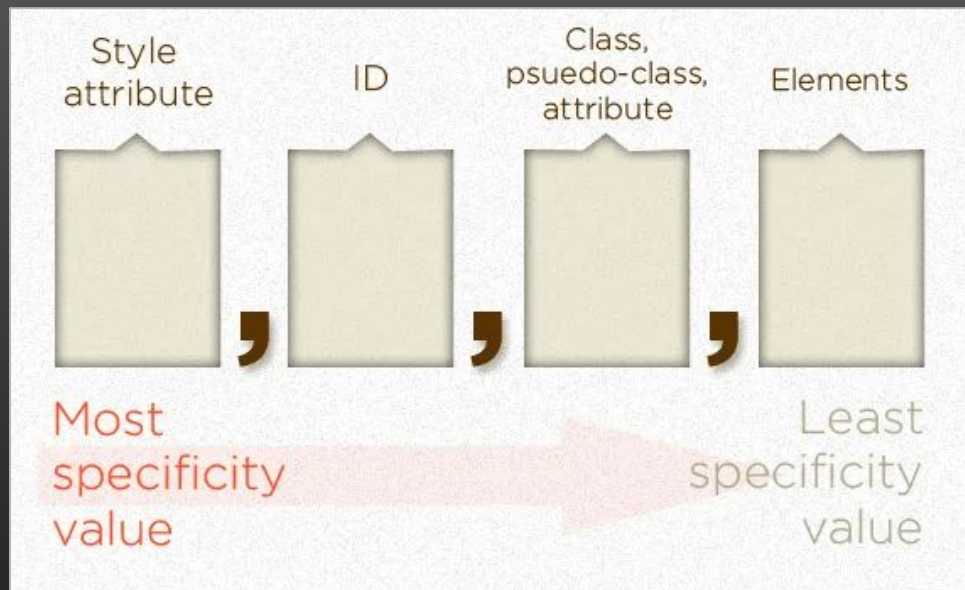

Les attributs **class** et **id**

- ❑ Le nom donné aux **class** et **id** doit être le plus **précis** et **significatif** possible
- ❑ Ce **nom** doit commencer par une **lettre**, être écrit en **minuscule** et ne doit contenir ni accent, espace, ou caractère spécial (excepté le tiret)
- ❑ On peut cibler un seul **élément** parmi ceux possédant la même **classe** :
`h2.nom-classe { color: green }`

Les règles de priorité

- ❏ Les règles **CSS** sont lues et appliquées dans l'**ordre d'écriture** du code (du haut vers le bas), la **dernière** règle déclarée sera donc **prioritaire**, sauf si une règle est plus **précise**
- ❏ Mais il existe néanmoins certaines règles de **priorité** :
 - La règle **CSS** la plus **proche** de sa cible sera prioritaire
 - Le sélecteur le plus **précis** sera prioritaire
- ❏ `<h1 style="color : red"> ... </h1>` **VS** `h1 { color: blue }`

Les règles de priorité



Les règles de priorité

- ❏ Une règle **CSS** peut être **prioritaire** sur toutes les autres **règles** en ajoutant **!important** à la fin de la déclaration

`color: red !important;`

- ❏ Considérée comme une **mauvaise** pratique, elle ne doit être utilisée qu'en **dernier recours**

- ❏ Un article rappelant les règles de **spécificité** : <https://css-tricks.com/specifics-on-css-specificity/>

Cibler toujours plus ...

❏ Les combineurs :

- `div a {`
 `color: red;`
 `}` → Tous les `<a>` à l'intérieur d'une `<div>`
- `div > a {`
 `color: red;`
 `}` → Les `<a>` enfants directs d'une `<div>`
- `div + a {`
 `color: red;`
 `}` → Les `<a>` placés juste après une `<div>`

Cibler toujours plus ...

❏ Les sélecteurs d'attribut :

- `a[title] {`
 `color: red;`
 `}` → Les éléments `<a>` avec un attribut `title`
- `a[href="https://site.com"] {`
 `color: red;`
 `}` → Les éléments `<a>` avec ce href précis

❏ https://developer.mozilla.org/fr/docs/Web/CSS/Attribute_selectors

Cibler toujours plus ...

- ❑ On peut combiner autant de **sélecteurs** que l'on souhaite :

```
div.nom-classe > a[title] { color: gray }
```

- ❑ Le **sélecteur universel** `*` permet de cibler **tous** les éléments à la fois :

```
div * { color: gray }
```

- ❑ On peut parfaitement cibler les éléments **body** ou **html** :

```
body { background-color: gray }
```

- ❑ <https://stackoverflow.com/applying-css-to-html-body-and-the-universal-selector>

Pseudo-classes / pseudo-éléments

❏ Une **pseudo-classe** est un mot-clé ajouté à un **sélecteur** et apportant une **précision** sur l'élément ciblé

- **h1: hover** → lorsque <h1> est survolé par l'utilisateur
- **p: last-child** → le dernier <p> de son élément parent
- **li: nth-child(3)** → le troisième de son élément parent
- **a: first-of-type** → le premier <a> de son élément parent

❏ Parmi les **pseudo-classes** les plus utiles :

:active **:invalid** **:focus** **:visited** **:checked** **:not()** **:first-child ...**

❏ <https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-classes>

Pseudo-classes / pseudo-éléments

- ❏ Un **pseudo-élément** est un mot-clé ajouté à un **sélecteur** et permettant d'agir sur une partie de celui-ci. Un seul **pseudo-élément** peut être utilisé par **sélecteur**

- **p::first-line** → agit sur la première ligne du <p>

- ❏ Les **pseudo-éléments** sont encore peu nombreux, et peuvent nécessiter l'ajout d'un **préfixe** pour assurer la compatibilité avec certains **navigateurs**

- **p::-moz-selection** → agit sur la partie du <p> sélectionnée par l'utilisateur

- ❏ <https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-elements>

Le style

Les unités de mesure

- ❑ En **CSS** les unités de **mesure** peuvent être exprimées avec des valeurs **absolues** ou **relatives**
- ❑ Une valeur **absolue** reste **fixe** quelque soit son **environnement** et est généralement déclarée en **px** (pixels)
- ❑ Une valeur **relative** est dynamique, et **varie** selon son environnement et peut être déclarée en **em** / **rem** / **%**
- ❑ Le **em** et le **%** sont relatifs à la **taille** déclarée du **parent**, et le cas échéant à celle du **navigateur**
- ❑ Le **rem** est relatif à la **taille** de l'élément **racine** de la page web : le **<html>**. Cet élément **racine** hérite par défaut de la taille de **police** du **navigateur** de l'utilisateur
- ❑ <https://medium.com/codshake/unit%C3%A9s-de-mesures-em-vs-rem-eac03dbcb9c7>

Les unités de mesure



CSS Units Cheat Sheet

px

Absolute pixel value

%

A percentage of the parent element.
100% is the width of the parent element

em

Relative to the font size of the element

vh

Relative to 1% of the viewport's height

rem

Relative to the font size of
the root element

vw

Relative to 1% of the viewport's width

Les polices

- ❏ La **taille** de la **police** est déclarée avec la propriété **font-size** et les différentes **valeurs** cités plus haut

font-size: 16px;

- ❏ L'**épaisseur** de la police est déclarée avec **font-weight** et les valeurs **normal** / **bold** / **lighter** / **bolder** ou bien des valeurs entre **100** et **900**

font-weight: normal; / font-weight: 400;

- ❏ Le **style** de la police est déclaré avec **font-style** avec les valeurs **normal** / **italic** / **oblique**

font-style: italic;

Les polices

- ❏ Le **type** de police est déclarée avec **font-family**. On déclare généralement plusieurs polices **alternatives** pour les différents **navigateurs**

font-family: Garamond, "Times New Roman", serif;

- ❏ Des **polices** peuvent aussi être **téléchargées** via des sites comme [GoogleFonts](#) ou [fontSquirrel](#) puis **déclarées** dans le **fichier CSS**

```
@font-face {  
  font-family: "myFont";  
  src: url("myFont.ttf");  
}
```

```
p {  
  font-family: "myFont";  
}
```

Les polices

- Les **polices** peuvent aussi simplement être **intégrées** avec une balise `<link>` fournie par [GoogleFonts](#)

To embed a font, copy the code into the `<head>` of your html

☒ `<link>` ☐ `@import`

```
<link rel="preconnect" href="https://fonts.gstatic.com">
<link href="https://fonts.googleapis.com/css2?family=Lobster&display=swap" rel="stylesheet">
```

CSS rules to specify families

```
font-family: 'Lobster', cursive;
```

Les polices

- ❏ Différents types d'**espacements** dans les polices sont possibles grâce aux **propriétés** suivantes :
line-height / **letter-spacing** / **word-spacing** / **text-align**
- ❏ Certaines **valeurs** peuvent être exprimées avec un facteur **multiplicateur**
line-height : 3; → 3 fois la taille de police
- ❏ Les **valeurs** déclarées peuvent très bien être des valeurs **négatives**
letter-spacing: -3px; → l'écriture se brouille ou s'inverse
- ❏ La propriété **text-align** ne concerne que les éléments de type **inline** et peut prendre les valeurs :
left / **right** / **center** / **justify**

Les polices

- ❏ La propriété **text-decoration** est une propriété raccourcie pour gérer les **valeurs** des propriétés suivantes :

text-decoration-line / **text-decoration-color** / **text-decoration-style**

text-decoration: underline dotted red; → **underline dotted red**

- ❏ Une quatrième **valeur** peut être ajoutée pour également préciser l'**épaisseur** de la décoration choisie

text-decoration: underline dotted red 2px;

- ❏ La **mise en forme** du texte est gérée avec la propriété **text-transform** et peut prendre les valeurs :

capitalize / **uppercase** / **lowercase**

Les polices

- ❏ La propriété **text-shadow** permet d'ajouter une ou plusieurs **ombres** au texte et prend quatre **valeurs** : les **distances** horizontales et verticales, le rayon du **flou** et la **couleur**.

```
text-shadow: 1px 1px 2px red, 0 0 1em blue, 0 0 0.2em blue;
```



text-shadow

- ❏ Les **puces** ou la **numérotation** des listes sont gérées avec les propriétés **list-style-type** et **list-style-image**

```
list-style-type: square; / list-style-type: upper-roman;
```

```
list-style-image: url("emoticon.png");
```





list-style-image



list-style-image

Les couleurs

- ❑ Les **couleurs** peuvent être déclarées avec un **mot-clé** désignant la couleur
`color: burlywood;` → 
- ❑ Les **couleurs** peuvent être déclarées avec une valeur **hexadécimale**
`#000000 [black]` → `#ffffff [white]`
- ❑ Les **couleurs** peuvent être déclarées avec une valeur **rgb** ou **rgba**
`rgb(0, 0, 0) [black]` → `rgb(255, 255, 255) [white]`
`rgba(0, 0, 0, .0)` → `rgba(0, 0, 0, 1) [avec opacité]`
- ❑ Les **couleurs** peuvent être déclarées avec une valeur **hsl** (**hue**, **saturation**, **lightness**)
`hsl(0, 100, 50)` → 

Les couleurs

- ❏ Des **nuanciers** existent en ligne avec beaucoup de **palettes de couleurs** proposées :
→ <https://colours.neilorangepeel.com/> | <https://coolors.co/>
- ❏ La **liste** complète des différents **mots-clés** avec leurs valeurs **hexadécimales** :
→ https://developer.mozilla.org/fr/docs/Web/CSS/color_value
- ❏ Cet **outil** propose des **conversions** entre les différents **codes couleur** selon les besoins :
→ <https://serennu.com/colour/hsltorgb.php>
- ❏ Une extension **Chrome** permet de "**piquer**" une **couleur** sur n'importe quelle **page web** :
→ <https://chrome.google.com/Color-Picker>

Les backgrounds

- ❑ La **propriété** raccourcie **background** est utilisée pour déclarer tout ce qui concerne l'**arrière-plan** d'un élément

`background-color: #b2e4db;`

- ❑ Plusieurs **images** peuvent être **empilées**. La **première** déclarée sera **au-dessus** des suivantes

`background-image: url("images/fire-background.jpg");`

- ❑ L'**image** utilisée peut-être **répétée** ou non, **centrée** ou non...

`background-repeat: repeat-x;` → répétée sur l'axe horizontal
`background-position: 25% 75%;` → axe horizontal / vertical

Les backgrounds

- ❏ L'image peut aussi être **redimensionnée** pour s'adapter à son **conteneur**
`background-size: cover;` `background-size: 75%;`
- ❏ La propriété **background-attachment** permet de **fixer** ou non l'**arrière-plan** de l'élément
`background-attachment: fixed;`
- ❏ Toutes ces **propriétés** peuvent être écrites dans une seule et même **déclaration**
`background: center ("images/fire-background.jpg") fixed;`
- ❏ <https://developer.mozilla.org/fr/docs/Web/CSS/background>

Les backgrounds

- ❏ La valeur de **background-image** peut aussi représenter un **dégradé de couleurs** horizontal ou vertical
`background: linear-gradient (red, yellow, green);`
- ❏ Différents **générateurs** de gradients sont disponibles : <https://uigradients.com/> | <https://cssgradient.io/>
- ❏ Les précisions de **MDN** sur le sujet : <https://developer.mozilla.org/fr/docs/Web/CSS/gradient>



Le système de boîtes

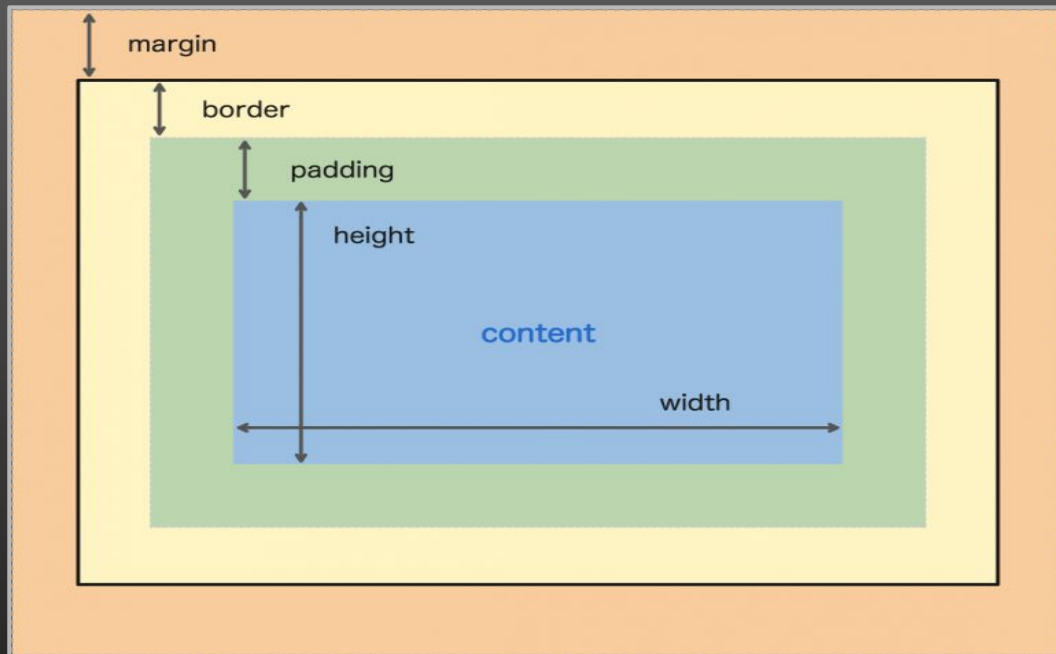
Block ou inline

- ❏ Les éléments **HTML** sont tous contenus dans des **boîtes** qui sont de type **block** ou **inline**
- ❏ Les boîtes de type **block** occupent la totalité de la **largeur** disponible et imposent un **retour à la ligne** à l'**élément** suivant
- ❏ Les boîtes de type **inline** occupent uniquement la **largeur** de leur contenu et plusieurs éléments **inline** peuvent donc se trouver **côte à côte**
- ❏ Une boîte peut également être de type **inline-block** : elle prend des caractéristiques des éléments **block** et également des éléments **inline**

Block ou inline

- ❏ La **largeur** et la **hauteur** d'un élément **block** sont définies avec les propriétés **width** et **height**. Celles d'un élément **inline**, étant celles de son contenu, ne peuvent être redéfinies
- ❏ Un élément **inline-block** occupe la **largeur** de son contenu comme un élément **inline**, mais ses dimensions peuvent être redéfinies comme pour un élément **block**
- ❏ Le **type** d'un élément peut être changé avec la propriété **display** : `p { display: inline };`
- ❏ Cette **propriété** peut donc prendre les valeurs **block** / **inline** / **inline-block** ou bien encore **none** (l'élément n'est alors plus visible)

Margin et padding



- Un modèle de boîte -

Margin et padding

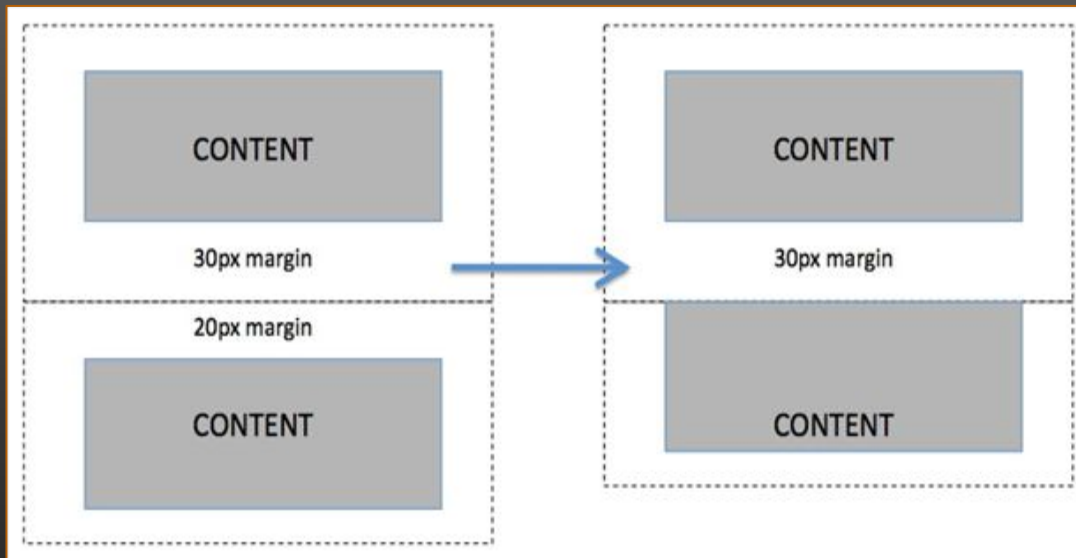
- ❑ Chaque **boîte** possède une marge intérieure **padding** et une marge extérieure **margin**
- ❑ Seuls les éléments de type **inline** ne possèdent pas de marge extérieure **verticale**
- ❑ Les **marges** de chaque côté de l'élément peuvent être déclarées individuellement avec les propriétés **margin-top** / **margin-bottom** / **margin-left** / **margin-right**
- ❑ Ces différentes **valeurs** peuvent également être écrites en une seule **déclaration** :

margin: 10px;	→	tous les côtés
margin: 10px 5px;	→	haut-bas / gauche-droite
margin: 10px 5px 15px;	→	haut / gauche-droite / bas
margin: 10px 5px 15px 8px;	→	haut / droit / bas / gauche

Margin et padding

- ❏ Lorsque deux éléments de type **block** se superposent, leurs **marges** extérieures **fusionnent** pour ne garder qu'**une** des deux (la plus **grande** si elles sont différentes)
- ❏ <https://www.joshwcomeau.com/css/rules-of-margin-collapse/>
- ❏ Un élément de type **block** peut facilement être **centré horizontalement** en utilisant les valeurs de **margin**
`div { margin: 0 auto };`

Margin et padding



- Exemple de margin-collapse -

Les bordures

- ❑ Chaque **boîte** possède également une bordure définie par la propriété raccourcie **border**. Elle permet de déclarer les valeurs de **border-style** / **border-width** / **border-color**
- ❑ La propriété **border-style** possède de nombreuses valeurs telles que **dotted** / **dashed** / **solid** / **groove** ...

`border: solid 2px red;`

- ❑ La **bordure** peut aussi être arrondie avec la propriété **border-radius** et chaque angle peut être déclaré **individuellement** de la même manière qu'avec les propriétés **margin** et **padding**

`border-radius : 10px;`

`border-radius: 10px 5px;`

`border-top-left-radius: 5px;`



tous les angles



haut-bas / gauche-droite



angle haut-gauche ...

Les bordures

- ❑ La propriété `box-shadow` permet de donner une **ombre** à la boîte contenant l'élément
- ❑ Comme `text-shadow`, elle prend **4 valeurs** : distances horizontales et verticales / rayon du flou / couleur

`box-shadow: 10px 5px 5px red;`



Gérer les débordements

- ❑ Par défaut, les **padding** et **border** donnés à un élément viennent **s'ajouter** au **width** de cet élément et agrandir ainsi sa **largeur** totale

```
p {  
    width: 200px;  
    border: 5px;  
    padding: 20px;  
}
```

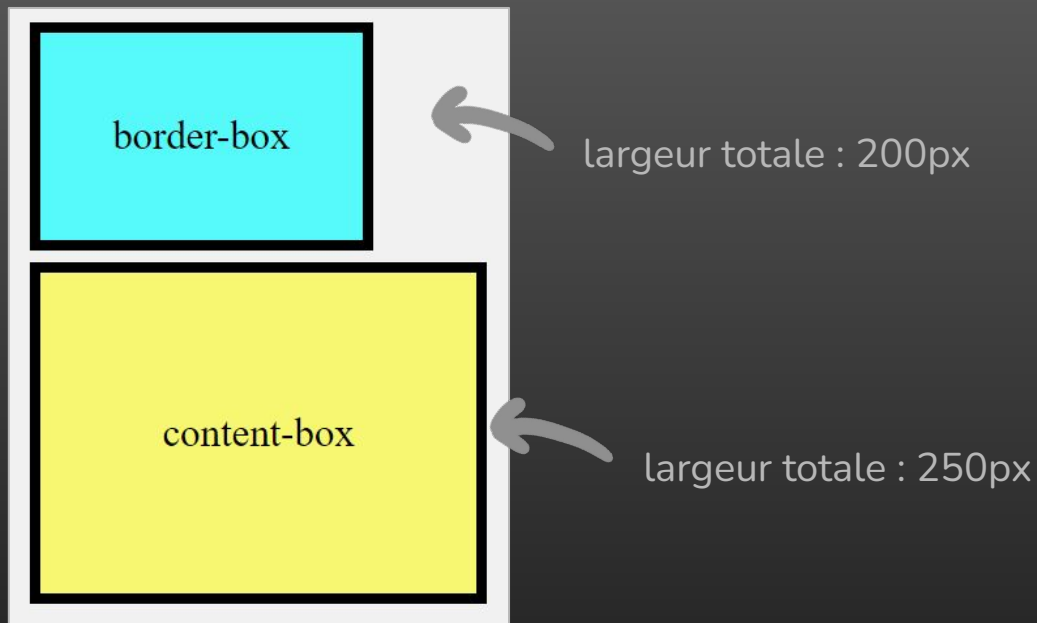
→ largeur totale : 250px

- ❑ La propriété **box-sizing** peut contraindre l'élément à garder sa **largeur** initiale même si **padding** et **border** lui sont ajoutés en ré-ajustant la taille du **width**

```
box-sizing: border-box; → largeur totale : 200px
```

Gérer les débordements

```
p {  
  width: 200px;  
  border: 5px;  
  padding: 20px;  
}
```



Gérer les débordements

- ❏ Lorsque le contenu d'un boîte est plus grand que celle-ci, la propriété **overflow** permet de masquer ou non ce contenu, avec les valeurs **visible** / **hidden** / **scroll** / **auto**

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Voluptatem, natus? Aliquid, amet veniam. Ab aliquid alias cumque dolore quasi dolores vero eligendi, ipsum labore ipsa libero sequi veniam laboriosam. Libero, vel eaque, rerum magnam minima assumenda corrupti fugiat molestias repellat omnis possimus repudiandae nulla voluptatibus alias corporis cupiditate reiciendis fuga unde dignissimos! Tenetur sequi explicabo, qui illo unde aliquam deserunt enim cum velit necessitatibus asperiores nobis inventore eos, deleniti minima nihil totam eveniet quia, cupiditate nemo. Reprehenderit quaerat rerum placeat, fugiat odit sunt? Voluptates dolores nulla autem cumque culpa error odit rem alias modi unde esse placeat quasi eum qui, reiciendis magnam a impedit hic quod adipisci nemo corporis aliquid quas? Odit consequuntur ullam, delectus tempora illum vero adipisci minus, placeat debitis atiquid soluta deserunt. Harum repellat molestias voluptatem id repudiandae libero quos nesciunt doloribus ducimus dolorum reprehenderit itaque dignissimos ipsum consectetur molestiae, ipsam provident voluptates voluptate blanditiis. Voluptates, deleniti.

overflow: visible;

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Voluptatem, natus? Aliquid, amet veniam. Ab aliquid alias cumque dolore quasi dolores vero eligendi, ipsum labore ipsa libero sequi veniam laboriosam. Libero, vel eaque, rerum magnam minima assumenda corrupti fugiat molestias repellat omnis possimus repudiandae nulla voluptatibus alias corporis cupiditate reiciendis fuga unde dignissimos! Tenetur sequi explicabo, qui illo unde aliquam deserunt enim cum velit necessitatibus asperiores nobis inventore eos, deleniti minima nihil totam eveniet quia, cupiditate nemo. Reprehenderit quaerat rerum placeat, fugiat odit sunt? Voluptates dolores nulla autem cumque culpa error odit rem alias modi unde esse placeat quasi eum qui, reiciendis magnam a impedit hic quod adipisci nemo corporis aliquid quas? Odit consequuntur ullam, delectus tempora illum vero adipisci minus, placeat

overflow: hidden;

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Voluptatem, natus? Aliquid, amet veniam. Ab aliquid alias cumque dolore quasi dolores vero eligendi, ipsum labore ipsa libero sequi veniam laboriosam. Libero, vel eaque, rerum magnam minima assumenda corrupti fugiat molestias repellat omnis possimus repudiandae nulla voluptatibus alias corporis cupiditate reiciendis fuga unde dignissimos! Tenetur sequi explicabo, qui illo unde aliquam deserunt enim cum velit necessitatibus asperiores nobis inventore eos, deleniti minima nihil totam eveniet quia, cupiditate nemo. Reprehenderit quaerat rerum placeat, fugiat odit sunt? Voluptates dolores nulla autem cumque culpa error odit rem alias modi unde esse placeat quasi eum qui, reiciendis magnam a impedit hic quod adipisci nemo corporis aliquid

overflow: scroll;

Le positionnement

Sortir du flux

- ❑ En **HTML**, on appelle **flux** la façon dont les **éléments** vont naturellement **se positionner** et s'afficher dans une page web, sans intervention du **CSS**
- ❑ Le **navigateur** qui analyse le code **HTML** va afficher les éléments selon **l'ordre** dans lesquels il va les lire : de gauche à droite et de haut en bas
- ❑ Le **CSS** permet, à l'aide de certaines **propriétés**, de sortir un élément de ce **flux** habituel pour le positionner **autrement**

float - clear

- ❑ la propriété **float** permet de **sortir** un élément du **flux** normal de la page pour le **placer** sur la **gauche** ou sur la **droite**
- ❑ Les éléments **le suivant** dans le code **HTML** viennent alors se positionner **autour** de lui
`float: left; float: right;`
- ❑ La propriété **clear** permet de **sortir** un élément de l'emprise d'un **élément flottant** en remplaçant ce dernier dans le **flux normal**
- ❑ Elle prendra les valeurs **left** / **right** / **both** selon le **côté** où se trouve l'élément **flottant**
`clear: left; clear: right; clear: both;`

float - clear

cing elit. Voluptatem,
umque dolore quasi dolores
niam laboriosam. Libero,
orrupti fugiat molestias
ptatibus alias corporis
aetur sequi explicabo, qui
ssitatibus asperiores nobis

float

iet quia, cupiditate nemo. Reprehenderit quaerat rerum
la autem cumque culpa error odit rem alias modi unde
a impedit hic quod adipisci nemo corporis aliquid quas?
n vero adipisci minus, placeat debitis aliquid soluta
id repudiandae libero quos nesciunt doloribus ducimus

float: right;

float

Lorem ipsum, dolor sit amet con
natus? Aliquid, amet veniam. Al
vero eligendi, ipsum labore ipsa
vel eaque, rerum magnam minir
repellat omnis possimus repudia
cupiditate reiciendis fuga unde c
illo unde aliquam deserunt enim
inventore eos, deleniti minima nihil totam eveniet quia, cupi
placeat, fugiat odit sunt? Voluptates dolores nulla autem cum
esse placeat quasi eum qui, reiciendis magnam a impedit hic
Odit consequuntur ullam, delectus tempora illum vero adipis
deserunt. Harum repellat molestias voluptatem id repudiand
dolorum,

float: left;

Position

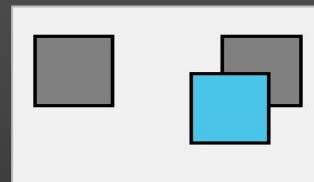
- ❏ La propriété **position** permet également de sortir un élément du **flux** normal de la page afin de le **positionner** de façon spécifique
- ❏ Les différentes **valeurs** de cette propriété sont : **relative** / **absolute** / **fixed** / **sticky**. La valeur par **défaut** d'un élément **positionné** naturellement dans le **flux** est : **static**
- ❏ Un élément de type **block** dont la position est **fixed** ou **absolute** se comportera comme un élément de type **inline-block**
- ❏ Les propriétés **top** / **bottom** / **left** / **right** sont utilisées pour **positionner** l'élément sorti du **flux** de la page :

bottom: 40px; top: 100px

Position

- ❏ Un élément **positionné** de façon **relative** est sorti du **flux** normal de la page, tout en laissant **indisponible** l'**espace** qu'il y occupait

position: relative;



- ❏ La nouvelle **position** de l'élément est alors **calculée** par rapport à son **emplacement** précédent

top: 50px;



va s'éloigner de 50px du haut

left: 20px;

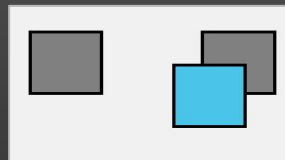


va s'éloigner de la gauche de 20px

Position

- ❑ Comme avec **relative**, un élément positionné avec **sticky** est sorti du **flux** en laissant l'**espace** qu'il occupait **indisponible**, et sa nouvelle **position** est calculée par rapport à son **emplacement** précédent

`position: sticky;`

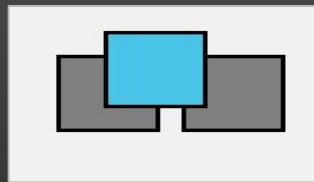


- ❑ Lorsque l'utilisateur **parcourt** la page, dès que l'**élément** se retrouve dans le **viewport** il reste **fixé** dans sa nouvelle **position** pendant que le reste de la page **défile** derrière lui
- ❑ <https://A-propos-de-la-position-sticky-alsacreations.html>

Position

- ❑ Un élément positionné avec **fixed** est sorti du **flux** en laissant l'**espace** qu'il occupait **disponible**, et sa nouvelle position est **calculée** par rapport à la fenêtre du **navigateur**

position: fixed;

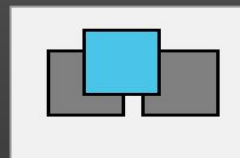


- ❑ Lorsque l'utilisateur **parcourt** la page, l'élément reste **fixe** pendant que le reste de la page **défile** derrière lui

Position

- ❏ Un élément positionné en **absolute** est sorti du **flux** normal et l'**espace** qu'il occupait est de nouveau **disponible**

`position: absolute;`



- ❏ La nouvelle **position** de l'élément est alors **calculée** par rapport à son élément **parent** (qui doit avoir une position **relative**) ou bien par rapport à l'élément **body**

`top: 50px;`



va s'éloigner de 50px du haut

`left: 20px;`

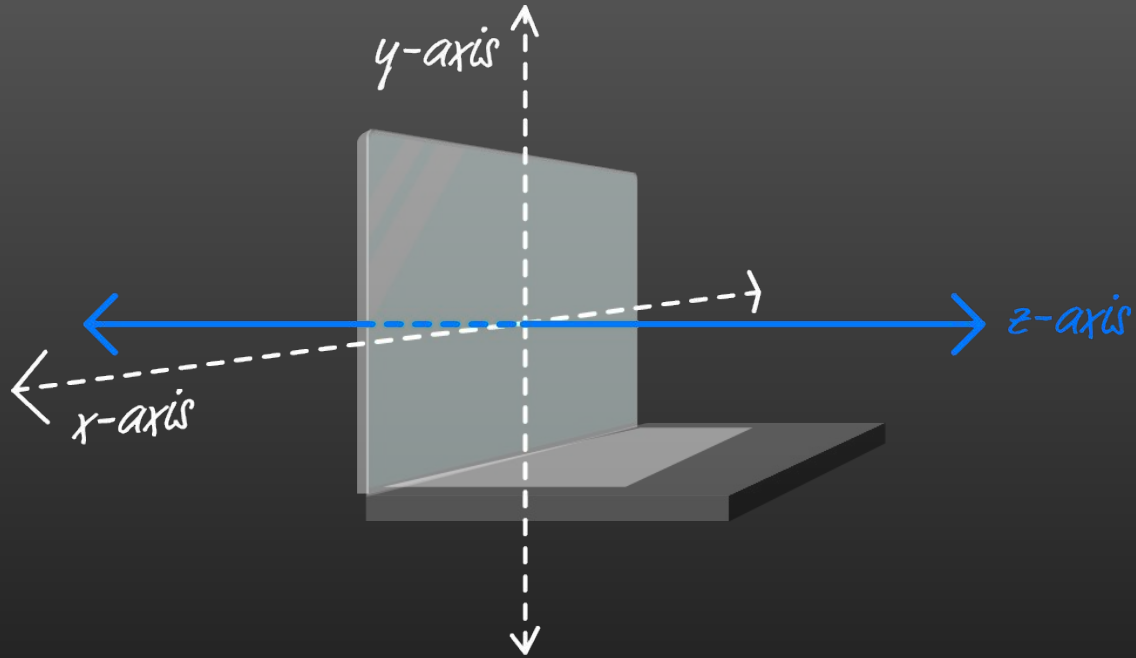


va s'éloigner de 20px de la gauche

Empilement et z-index

- ❑ Les **éléments** d'une page **HTML** sont **empilés** les uns au-dessus des autres, en des **couches** successives
- ❑ Les éléments placés **le plus haut** dans la pile se retrouvent sur le **devant** de l'écran et les éléments **en bas** de la pile se retrouvent à l'**arrière-plan**
- ❑ Par **défaut**, les éléments sont empilés selon leur **ordre d'apparition** dans le code **HTML**
- ❑ La propriété **z-index** permet de **modifier** l'ordre d'**empilement** par défaut en répartissant les éléments le long d'un **axe Z**

Empilement et z-index



Empilement et z-index

- ❑ La valeur donnée à **z-index** indique la **position** de l'élément sur l'axe **Z**. Cette valeur est de **0** par défaut
- ❑ Plus la valeur est **élevée**, plus l'élément est sur le **dessus** de la pile, et donc sur le **devant** de l'écran
- ❑ Si plusieurs éléments ont la **même valeur**, ils sont empilés selon leur **ordre** d'apparition dans le code **HTML**
- ❑ La propriété **z-index** a un effet sur les éléments **uniquement** s'ils ont une position autre que **static**



Responsive Design

Display

- ❏ La propriété **display** offre la possibilité de changer le **type** d'un élément **HTML**, c'est à dire **block** , **inline** ou **inline-block**
- ❏ Un élément **block** occupe la totalité de la **largeur disponible** de l'élément **parent** dans lequel il est contenu. Ses **dimensions** (largeur et hauteur) ainsi que ses **marges** peuvent être **modifiées**
- ❏ Un élément **inline** n'occupe que la **largeur** de son **contenu** et ses **dimensions** ne peuvent donc pas être **modifiées**. Un élément **inline** ne possède pas de **marges verticales**
- ❏ Un élément **inline-block** n'occupe que la **largeur** de son **contenu** et ses **dimensions** (largeur et hauteur) ainsi que ses **marges** peuvent être **modifiées**

Display

- ❏ La propriété **display** peut aussi permettre de choisir un **modèle** de représentation et d'agencement des **boîtes** dans une page : **flexbox** ou **grid**
- ❏ Dans le modèle appelé **Flexbox**, chaque élément est représenté par une **boîte flexible** placée dans un **conteneur** qui est lui-même **flexible**

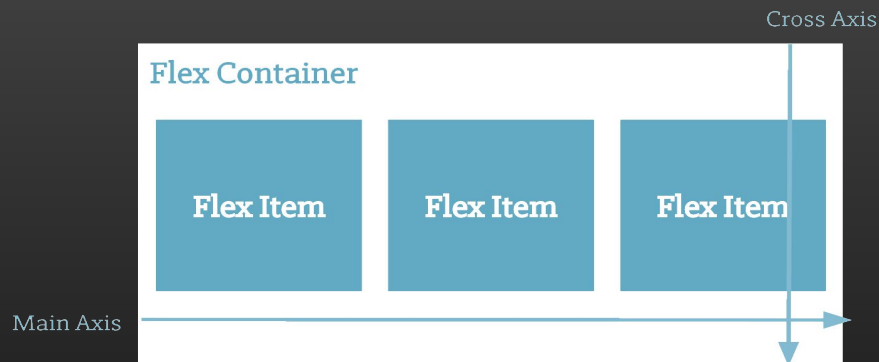
`display: flex;`

- ❏ Dans le modèle **Grid layout**, les éléments sont disposés au sein d'une **grille** composée de **lignes** et de **colonnes**

`display: grid;`

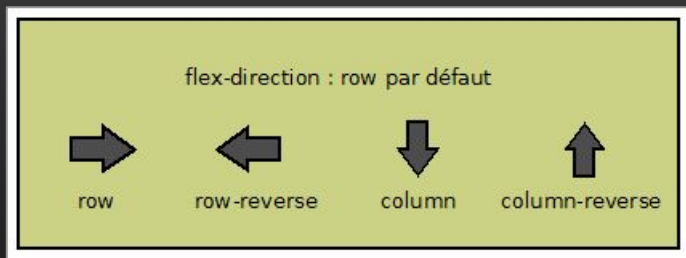
Flexbox

- ❑ La répartition des **boîtes** dans le modèle **flexbox** se fait selon deux axes : l'axe principal (**main axis**) et l'axe secondaire (**cross axis**), perpendiculaire au premier
- ❑ Les **éléments** de la page sont tous contenus dans des **conteneurs flexibles** répartis le long de ces deux **axes**



Flexbox

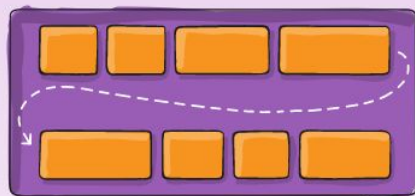
- ❑ Chaque **élément** d'un conteneur est appelé **flex item**, il est lui-même un **conteneur** pouvant contenir à son tour d'autres **conteneurs**, chacun étant réparti sur l'un des deux **axes**
- ❑ La propriété **flex-direction** permet de changer le sens du **main axis** qui par défaut est horizontal (**row**). Par opposition au **cross axis** qui est lui vertical (**column**)
- ❑ Les valeurs **row-reverse** et **column-reverse** permettent d'**inverser** l'ordre de **répartition** des **éléments** à l'intérieur de ces deux **axes**



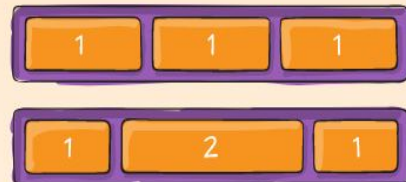
Flexbox

- ❑ Par défaut, les éléments au sein d'un conteneur vont se compresser pour tenir sur une seule ligne ou colonne
- ❑ La propriété **flex-wrap** va permettre aux éléments de se répartir sur plusieurs lignes ou colonnes et garder ainsi leurs tailles respectives
- ❑ Les propriétés **flex-grow** et **flex-shrink** permettent elles de compresser ou étirer un ou des éléments en particulier au sein de leur conteneur

flex-wrap

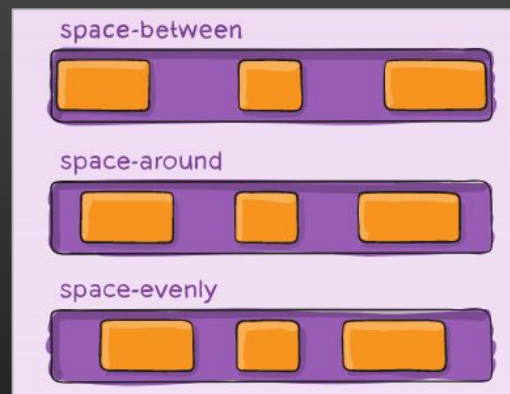
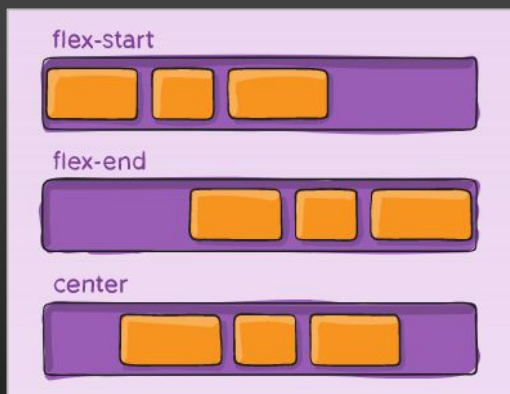


flex-grow



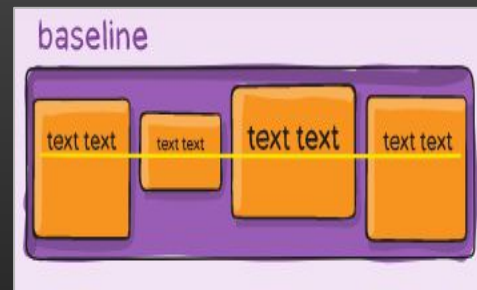
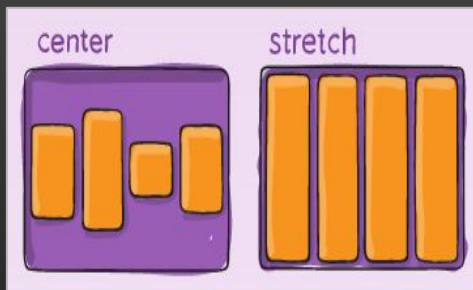
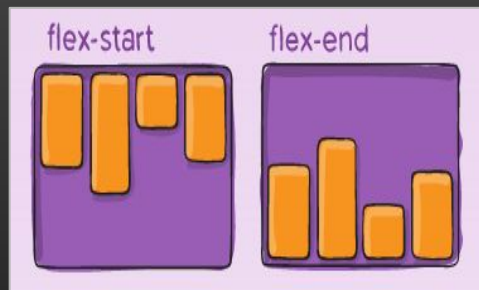
Flexbox

- ❑ La **répartition** des éléments le long de l'**axe principal** est gérée avec la propriété **justify-content**
- ❑ Ses différentes valeurs sont : **flex-start** / **flex-end** / **center** / **space-between** / **space-around** / **space-evenly**



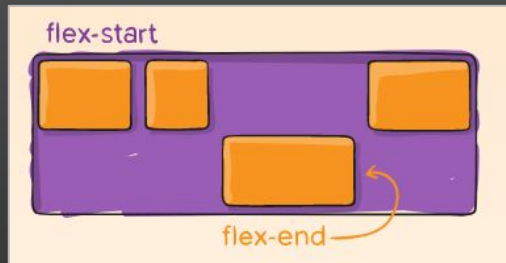
Flexbox

- ❏ La **répartition** des éléments le long de l'**axe secondaire** est gérée avec la propriété **align-items**
- ❏ Elle peut prendre les **valeurs** suivantes : **flex-start** / **flex-end** / **center** / **stretch** / **baseline**



Flexbox

- ❑ La propriété **align-self** permet de donner un **alignement** spécifique à un **élément** en particulier



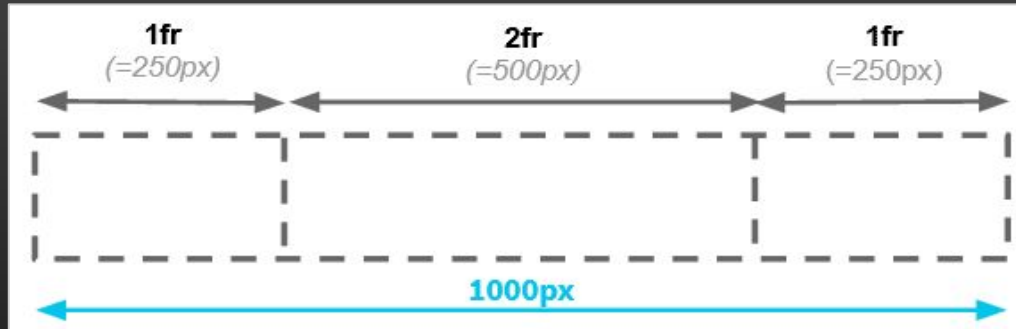
- ❑ Un **mémento** illustré du modèle **flexbox** : <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- ❑ Un petit **tuto** sympa : <https://www.alsacreations.com/tuto/lire/1493-css3-flexbox-layout-module.html>

Grid Layout

- ❑ Le modèle **grid** crée un conteneur composé de **rangées** et de **colonnes** entrecroisées dans lesquelles sont répartis les **éléments**
- ❑ Les propriétés **grid-template-columns** et **grid-template-rows** permettent d'agencer le **nombre** et la **taille** des colonnes et des rangées
- ❑ L'**espacement** entre les rangées ou entre les colonnes se fait **globalement** avec la propriété **grid-gap**, ou bien **individuellement** avec les propriétés **grid-column-gap** et **grid-row-gap**
- ❑ Sur la majorité des **navigateurs**, le préfixe **grid-** n'est aujourd'hui plus **nécessaire**

Grid Layout

- ❑ Les **tailles** des **colonnes** et **rangées** peuvent être exprimées avec l'ensemble des **unités de mesures** connues :
px / em / rem / % / auto
- ❑ L'**unité de mesure** spécifique **fr** est également disponible et représente une **fraction** de l'espace **disponible**



Grid Layout

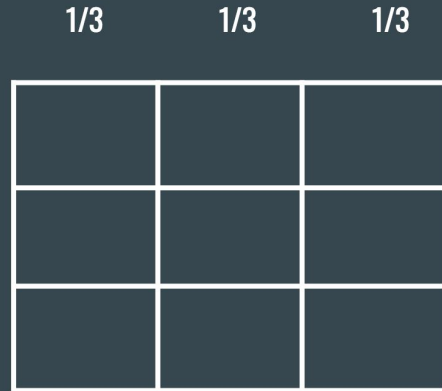
```
.container {  
  grid-template-columns: 1fr 1fr 1fr 1fr;  
  grid-template-rows: 200px 200px;  
  grid-gap: 1.5em;  
};
```



Grid Layout

- La notation `repeat()` permet de déclarer en même temps plusieurs colonnes ou rangées de même taille :
`grid-template-rows: repeat(2, 400px);`

```
.container {  
  /* ---- Defining your grid */  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr  
  grid-template-rows: repeat(3, 1fr);  
}
```



Grid Layout

- ❏ La propriété `grid-template-areas` permet de définir les différentes **cellules** d'une grille pour les **attribuer** aux éléments **HTML** choisis

```
grid-template-areas: "header header header"
                    "nav    cv    aside"
                    "footer footer footer";
```

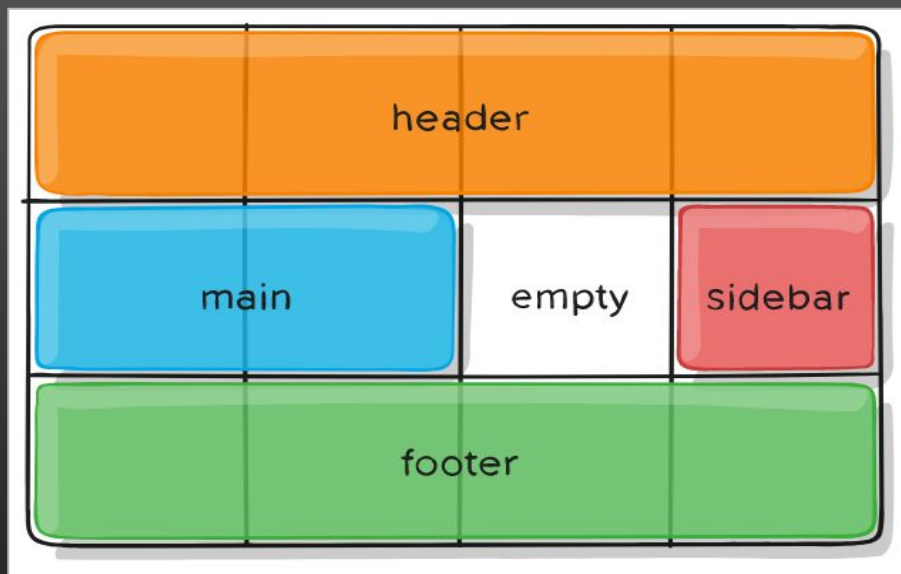
- ❏ La propriété `grid-area` permet ensuite de **lier** un **élément** avec la **cellule** de la grille qu'il **occupera**

```
#cv {
    grid-area: cv ;
};
```

- ❏ Si on désire laisser **vide** un espace de la **grille**, il sera indiqué par un **point** dans `grid-template-areas`

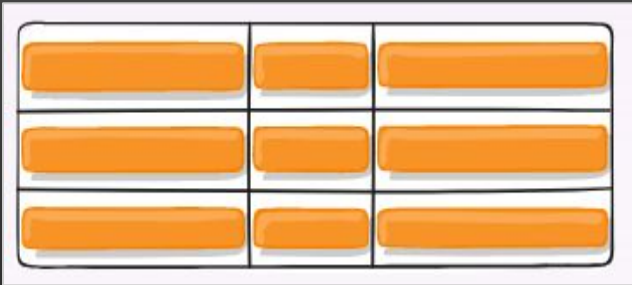
Grid Layout

```
.item-a {  
  grid-area: header;  
}  
.item-b {  
  grid-area: main;  
}  
.item-c {  
  grid-area: sidebar;  
}  
.item-d {  
  grid-area: footer;  
}  
  
.container {  
  display: grid;  
  grid-template-columns: 50px 50px 50px 50px;  
  grid-template-rows: auto;  
  grid-template-areas:  
    "header header header header"  
    "main main . sidebar"  
    "footer footer footer footer";  
}
```

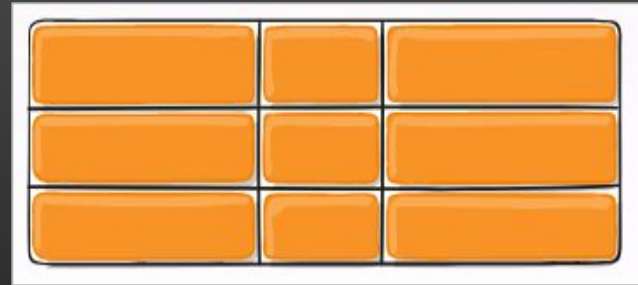


Grid Layout

- ❏ La propriété **align-items** permet de gérer la **disposition** de l'ensemble des **éléments** au sein d'une **rangée** avec les valeurs **start** / **end** / **center** / **stretch**



align-items: center ;

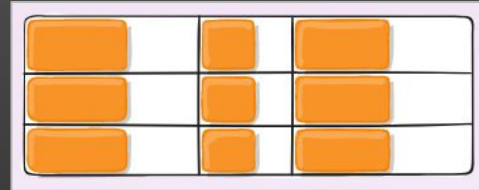


align-items: stretch ;

Grid Layout

- ❑ La propriété **justify-items** permet de gérer la **disposition** de l'ensemble des **éléments** au sein d'une **colonne** avec les mêmes valeurs **start** / **end** / **center** / **stretch**

`justify-items: start;`



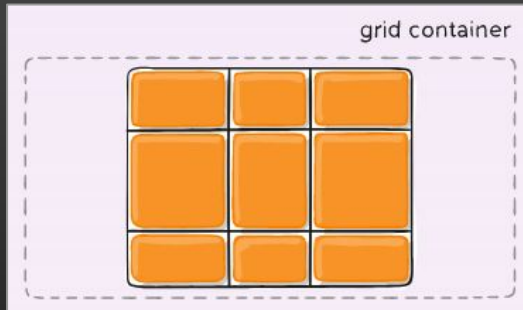
- ❑ La propriété raccourcie **place-items** permet de déclarer simultanément les valeurs de **align-items** et **justify-items**

`place-items: center;`

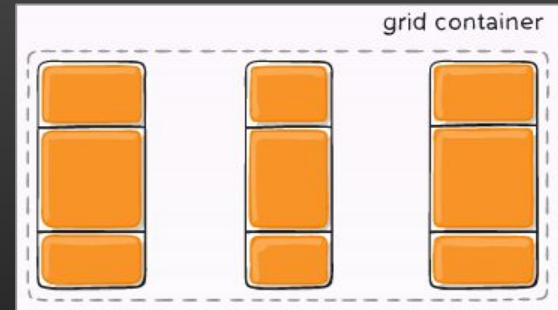


Grid Layout

- ❑ La propriété **justify-content** permet de gérer **horizontalement** la disposition globale des **éléments** dans l'espace restant **disponible** à l'intérieur du **conteneur**
- ❑ Elle prend les **valeurs** suivantes : **start** / **end** / **center** / **stretch** / **space-around** / **space-between** / **space-evenly**



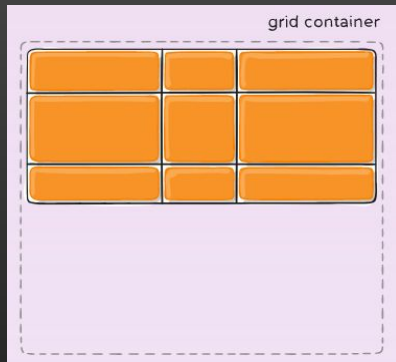
justify-content: center;



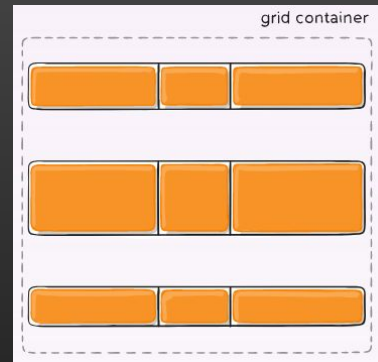
justify-content: space-between;

Grid Layout

- ❑ La propriété **align-content** permet de gérer **verticalement** la disposition des **éléments** dans un conteneur **grid** dans l'espace restant **disponible** à l'intérieur du **conteneur**
- ❑ Elle prend les mêmes valeurs : **start** / **end** / **center** / **stretch** / **space-around** / **space-between** / **space-evenly**



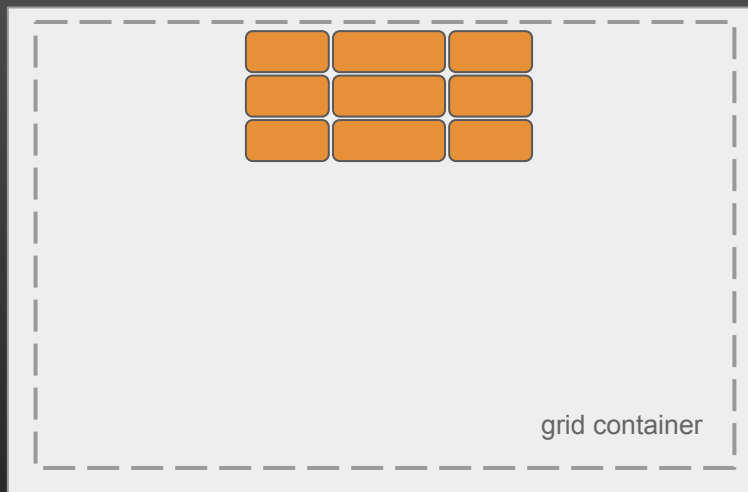
align-content: start;



align-content: space-around;

Grid Layout

- ❏ La propriété raccourcie **place-content** permet de déclarer à la fois les valeurs de **align-content** et **justify-content**

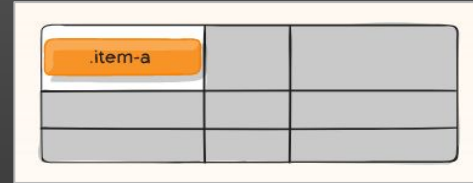


place-content: start center;

Grid Layout

- ❏ La propriété **align-self** permet de gérer la **disposition** d'un élément **unique** au sein d'une **rangée** avec les valeurs **start** / **end** / **center** / **stretch**

align-self: center;



- ❏ La propriété **justify-self** offre les mêmes **possibilités** pour d'un élément **unique** au sein d'une **colonne**

justify-self: end;



Grid Layout

- ❏ La propriété raccourcie **place-self** permet de déclarer **à la fois** les valeurs de **align-self** et **justify-self**

place-self: center;



place-self: center stretch;



Grid Layout

- ❏ Une autre manière de **positionner** les éléments au sein d'une **grille** se fait en se référant aux **numéros** des **lignes** et des **colonnes**
- ❏ Les **propriétés** utilisées sont alors : **grid-column-start** / **grid-column-end** / **grid-row-start** / **grid-row-end**
- ❏ Elles permettent d'**indiquer** précisément les **limites** horizontales et verticales de chaque **élément** en donnant les **numéros** des **colonnes** et des **rangées** où l'élément **commence** et **fin**

grid-column-start: 1;
grid-column-end: 2;

grid-row-start: 1;
grid-row-end: 4;

Grid Layout

```
.box1 {  
  grid-column-start: 1;  
  grid-column-end: 2;  
  grid-row-start: 1;  
  grid-row-end: 4;  
}
```

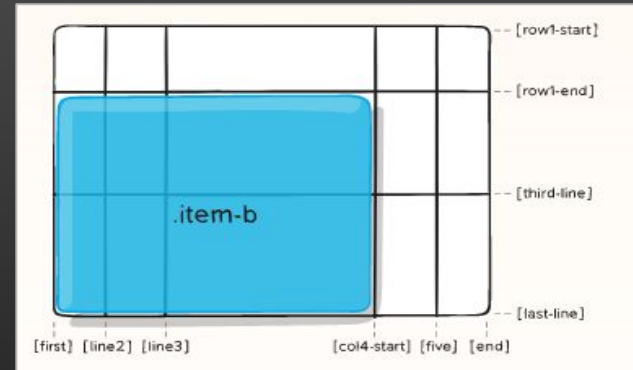
```
.box4 {  
  grid-column-start: 2;  
  grid-column-end: 4;  
  grid-row-start: 3;  
  grid-row-end: 4;  
}
```



Grid Layout

- ❏ La valeur **span** permet de préciser l'**intervalle** occupé par l'**élément** (combien de lignes ou de colonnes)

```
.item-b {  
  grid-column-start: 1;  
  grid-column-end: span 3;  
  grid-row-start: 2;  
  grid-row-end: span 2;  
}
```



Grid Layout

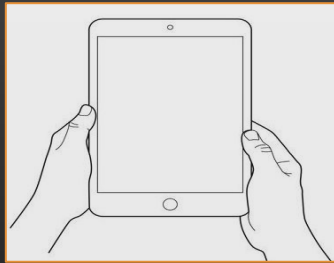
- ❏ Un **élément** contenu dans un modèle **grid** peut parfaitement **contenir** lui-même des **éléments** agencés selon le modèle **flexbox**
- ❏ Un **élément** contenu dans un modèle **flexbox** peut parfaitement **contenir** lui-même des **éléments** agencés selon le modèle **grid**
- ❏ Des **précisions** sur la propriété **grid-area** : <https://developer.mozilla.org/fr/docs/Web/CSS/grid-area>
- ❏ Un **mémento** illustré du modèle **grid** : <https://css-tricks.com/snippets/css/complete-guide-grid/>

Media queries

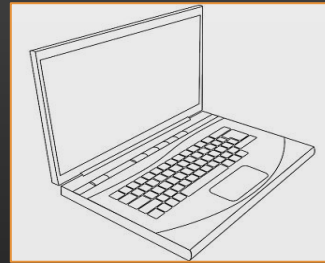
- ❑ Le **responsive design** est la façon de concevoir des **pages web** en adaptant leur **visuel** selon le **support** sur lequel celles-ci seront **consultées**
- ❑ Ces **supports** peuvent être un ordinateur **desktop**, une **tablette** ou bien un **téléphone mobile**, la principale **différence** étant donc la **taille de l'écran**



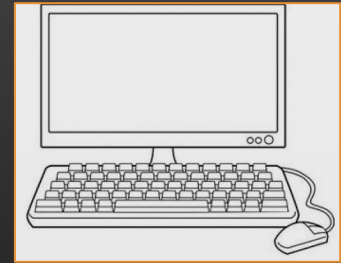
< 768 px



768 px - 992 px



992 px - 1200 px



> 1200 px

Media queries

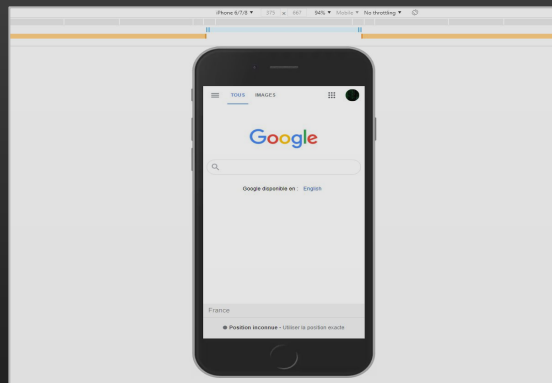
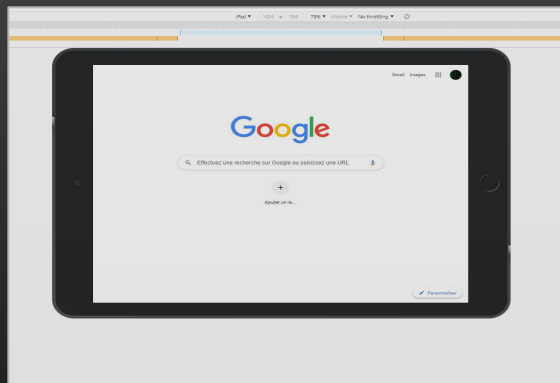
- ❑ Des **règles de style** différentes sont alors écrites selon la **taille de l'écran** utilisé. Ces types de déclaration sont appelés **media queries**
- ❑ Les **media queries** précisent donc une **taille d'écran** qui est alors une **condition** pour que les **règles de style** énoncées soient **appliquées** :

```
@media screen and (max-width: 600px) {  
    h2 {  
        color: grey;  
    }  
}
```

- ❑ Une présentation plus détaillée → <https://openclassrooms.com/fr/responsive-design-avec-les-media-queries>

Media Queries

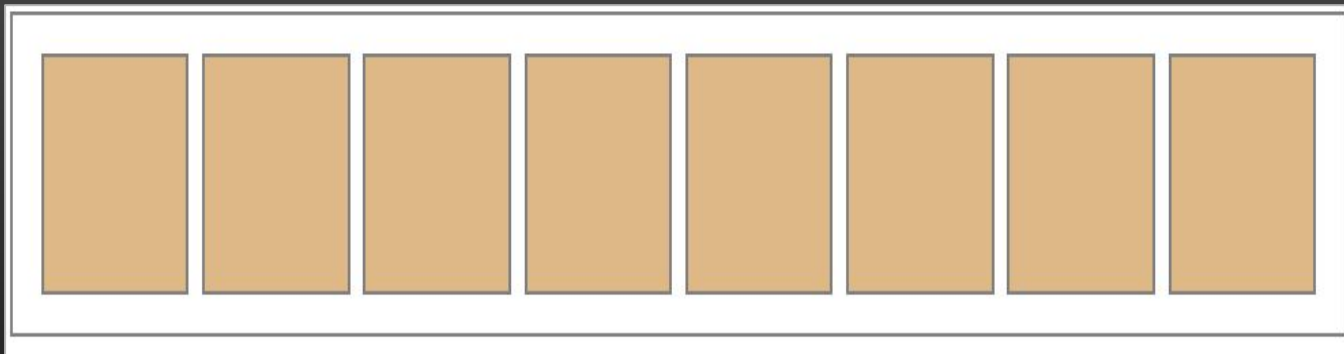
- ❏ Agrandir ou rétrécir la fenêtre du navigateur est un moyen de vérifier que les différentes règles de style s'appliquent bien lorsque la taille de l'écran change
- ❏ Un moyen beaucoup plus fiable est d'utiliser le simulateur dont dispose l'inspecteur du navigateur et dans lequel on peut décider des dimensions de l'écran, ou bien même sélectionner un type de mobile ou tablette



Media Queries

- ❏ Le modèle **flexbox** peut être utilisé pour faire du **responsive** sans avoir à déclarer de **media queries**

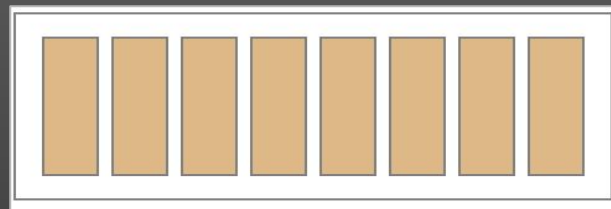
```
.container {  
  display: flex;  
}
```



- rendu obtenu sur écran large -

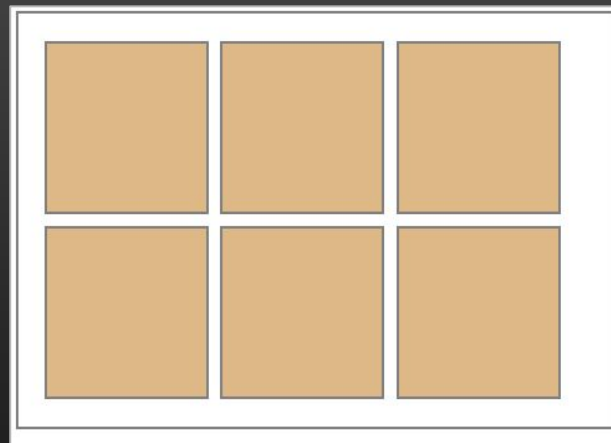
Media Queries

```
.container {  
  display: flex;  
}
```



- rendu obtenu sur petit écran -

```
.container {  
  display: flex;  
  flex-wrap: wrap;  
}
```



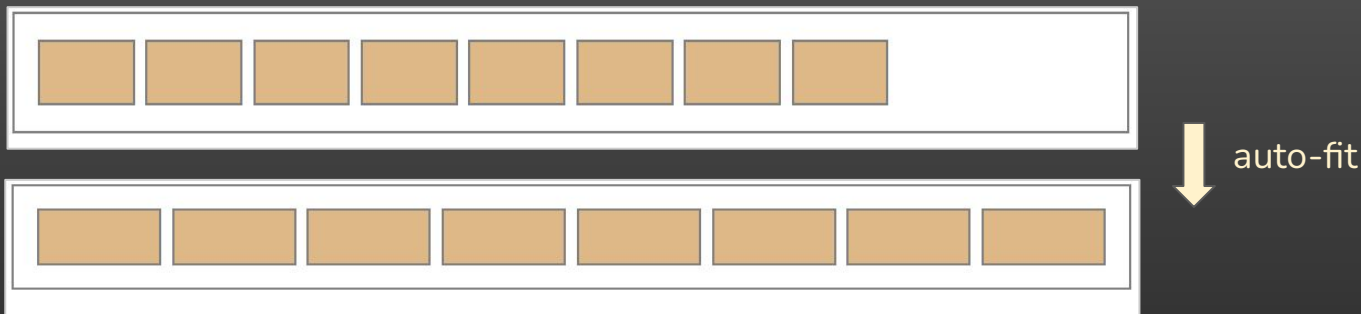
Media Queries

- ❏ Le modèle **grid** peut lui aussi être utilisé pour faire du **responsive** sans avoir à déclarer de **media queries**
- ❏ Certains **mots-clés** supplémentaires sont ainsi **disponibles** lors de l'utilisation de la fonction **repeat()** :
minmax() / auto-fill / auto-fit
- ❏ La fonction **minmax()** permet de préciser des **tailles** minimum et maximum. Toutes les **unités de mesure** sont utilisables, de même que la valeur **auto**

`grid-template-columns: repeat(6, minmax(100px, 200px))`

Media Queries

- ❑ Les mots-clés **auto-fill** et **auto-fit** permettent de combler l'**espace** disponible quand la **taille** de l'écran **grandit**
`grid-template-columns: repeat(auto-fit, minmax(100px, 1fr))`

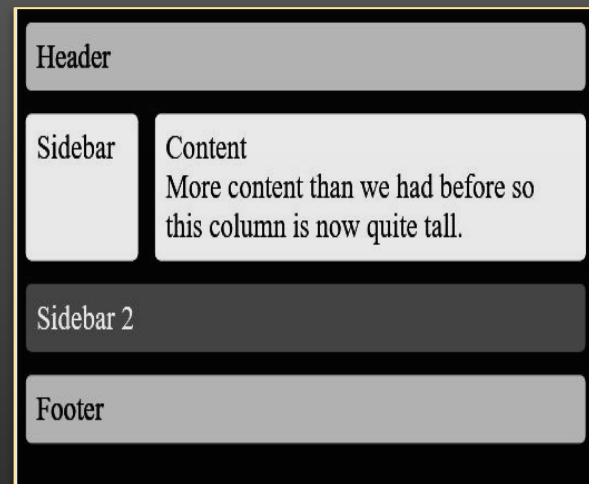
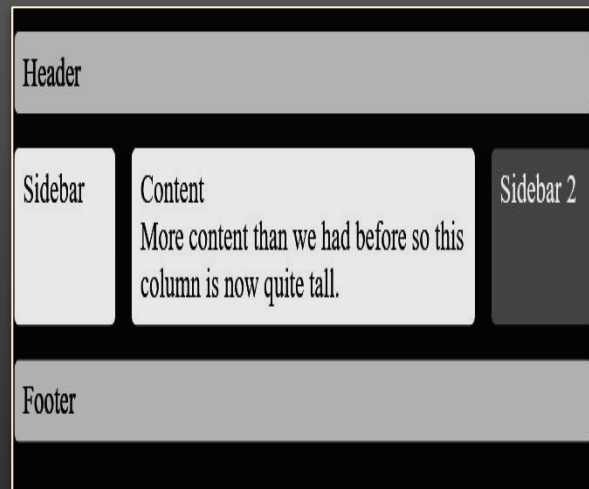
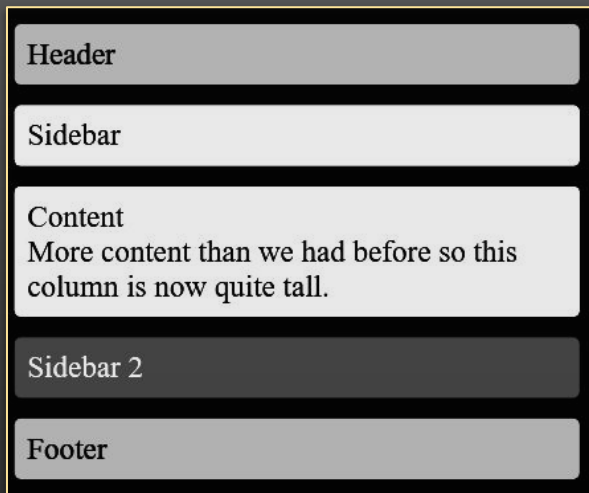


- ❑ Des **précisions** sur la fonction **minmax** → <https://la-cascade.io/css-grid-comment-fonctionne-minmax/>
- ❑ Les mots-clés **auto-fill** et **auto-fit** → <https://css-tricks.com/auto-sizing-columns-css-grid-auto-fill-vs-auto-fit/>

Media Queries

- ❏ Le **CSS** est aujourd'hui **principalement** pensé selon le modèle **mobile first**, c'est à dire que les premières **règles de styles** qui sont écrites sont celles pour les **petits écrans**
- ❏ Viennent ensuite les **media queries** qui contiennent les **règles de style** écrites pour les **tailles** d'écran plus **grandes**
- ❏ Pour rappel, le code **CSS** est lu de haut en bas par le **navigateur** et les **règles de style** seront donc **appliquées** dans leur **ordre** de déclaration
- ❏ Les premières **règles**, celles écrites pour les **mobiles**, sont alors **appliquées**. Puis les **règles** suivantes sont **lues** mais ne seront **appliquées** que si les **tailles** d'écran correspondent aux **media queries**

Media Queries



no media queries (pour mobiles)

@media screen and (min-width: 500px)

@media screen and (min-width: 600px)



Media Queries


- ❑ Des **changements** autres que **mise en page** et **disposition** des éléments peuvent être **déclarés** dans les **media queries**
- ❑ Les **polices** peut être par exemple être **différentes** (leur taille, leur famille, leur couleur...) selon les **tailles d'écran**

```
@media screen and (max-width: 500px) {  
    p {  
        color: grey;  
        font-family: Oswald;  
    }  
}
```


- ❑ Des **éléments** ou des **groupes** d'éléments peuvent également être **ajoutés** ou **enlevés** d'une page selon les **tailles d'écran**

`display: none;`

Media Queries


**Zoror**
Carnets de voyage

ACCUEIL BLOG CV CONTACT



Retour sur mes vacances aux États-Unis...

Voir l'article ▶


**JE SUIS UN GRAND VOYAGEUR**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris metus massa, luctus in tincidunt a, porttitor ac leo. Maecenas at mi feugiat turpis elementum ornare. Sed tempor rutrum lorem, in vestibulum felis elementum ac. Fusce purus orci, scelerisque ut tincidunt in, dignissim vel orci. Nulla lacus ultrices sagittis. Nulla vitae neque dignissim enim tempor suscipit et quis tellus. In hac habitasse platea dictumst. Aenean elit elit, pellentesque nec venenatis ut, convallis eu sem. Mauris eu leo nec arcu volutpat euismod nec eu dolor. Morbi aliquet, mauris quis porttitor dapibus, odio enim viverra eros, quis interdum massa urna at velit. Integer tempor facilis libero non accumsan. Aliquam etiam felis, dictum sed conimentum quis, molestie vel odio, laoreet eget ante massa, a sagittis quam. Cras posuere magna ac urna molestie vitae luctus lacus lobortis. Quisque leo neque, vulputate at semper non, varius porta enim.

Præsent sit amet lectus eros, ac pellentesque nisl. Donec consequat magna sed libero conimentum vitae aliquet elit ornare. Nunc at nulla purus. Aliquam sit amet sapien sit amet nisl aliquet rutrum vel nec mi. Mauris ultricies felis egetas sit varius molestie molestie sapien tristique. Cras lacus lacus, rutrum id sagittis sit amet, malesuada nec orci. Nulla consequat lobortis libero, ac convallis massa consectetur in. Nam facilisis posuere sagittis, sed a ligula id dui vulputate congue quis at tortor. Nunc pellentesque faucibus felis, eu venenatis massa luctum in. Donec venenatis lacus id tortor vestibulum id accumsan est lobortis. Morbi turpis quam, tincidunt in accumsan quis, ullamcorper quis orci. Quisque risi magna, egetas eget consectetur non, mollis ac ante. Donec elit felis, blandit at egestas in, lacus et dolor.


Ut blandit, diam id aliquam volutpat, quam libero euismod neque, ut volutpat nunc ipsum a magna. Donec hendrerit sem in dolor egestas lobortis. Etiam bibendum lobortis interdum. Titam ac felis vitae neque sodales sodales. Nunc tempus dignissim dapibus. Duis sit amet tellus vitae elit suscipit convallis. Sed et tincidunt velit. Donec congue elementum ante eu consectetur. Morbi lectus mauris, sodales a euismod id, dapibus sollicitudin urna. Sed sagittis sagittis placerat. Etiam at lorem nisl. Quisque imperdiet egestas tortor nec viverra. Tempus dignissim dapibus. Duis sit amet tellus vitae elit suscipit convallis. Sed et tincidunt velit. Donec congue elementum ante eu consectetur. Morbi lectus mauris, sodales a euismod id, dapibus sollicitudin urna. Sed sagittis sagittis placerat.

À PROPOS DE L'AUTEUR



Laissez-moi le temps de me présenter : je m'appelle Zoror, j'ai suis né le 23 Novembre 2006.


Bien maigre, n'est-ce pas ? C'est pourquoi, aujourd'hui, j'ai décidé d'écrire ma biographie ou zBiographie, comme vous voulez. Il afin que les zéros sachent enfin qui je suis réellement.



MON DERNIER TWEET

Hil haaaaaaan !
le 12 mai à 23h12

MES PHOTOS



MES AMIS

- ▶ Pupi le lapin
- ▶ Mr Baobab
- ▶ Kawaii
- ▶ Perceval.eu
- ▶ Belette
- ▶ Le comcombre masqué
- ▶ Ptit prince
- ▶ Mr Fan

**JE SUIS UN GRAND VOYAGEUR**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam nec sagittis massa. Nulla facilis. Cras id arcu lorem, et semper purus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Duis vel enim mi, in lobortis sem. Vestibulum luctus elit eu libero ultrices id fermentum sem sagittis. Nulla imperdiet mauris sed sapien dignissim id aliquam est aliquam. Maecenas non odio ipsum, a elementum nisl. Mauris non erat eu erat placerat convallis. Mauris in pretium urna. Cras laoreet molestie odio, consequat consequat velit commodo eu. Integer vitae lectus ac nunc posuere pellentesque non at eros. Suspendisse non lectus lorem.

Vivamus sed libero nec mauris pulvinar facilisis ut non sem. Quisque mollis ullamcorper diam vel faucibus. Vestibulum sollicitudin facilisis feugiat. Nulla euismod sodales hendrerit. Donec quis orci arcu. Vivamus fermentum magna a erat ullamcorper dignissim pretium nunc aliquam. Aenean pulvinar conimentum enim a dignissim. Vivamus sit amet lectus at ante adipiscing adipiscing eget vitae felis. In at fringilla est. Cras id velit ut magna rutrum commodo. Etiam ut scelerisque purus. Duis risus elit, venenatis vel rutrum in, imperdiet in quam. Sed vestibulum, libero ut bibendum consectetur, eros ipsum ultrices nisl, in rutrum diam auge non tortor. Fusce nec massa et risus dapibus aliquam vitae nec diam.

Phasellus ligula massa, congue ac vulputate non, dignissim at auge. Sed auctor fringilla quam quis porttitor. Praesent vitae dignissim magna. Pellentesque quis sem purus, vel elementum mi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Maecenas consectetur euismod urna. In hac habitasse platea dictumst. Quisque tincidunt porttitor vestibulum. Ut aculis, lacus at molestie lacinia, ipsum mi adipiscing ligula, vel mollis sem risus eu lectus. Nunc elit quam, rutrum ut dignissim sit amet, egestas at sem.

À PROPOS DE L'AUTEUR

Laissez-moi le temps de me présenter : je m'appelle Zoror, j'ai suis né le 23 Novembre 2006.

Bien maigre, n'est-ce pas ? C'est pourquoi, aujourd'hui, j'ai décidé d'écrire ma biographie (ou zBiographie, comme vous voulez !) afin que les zéros sachent qui je suis réellement.



MON DERNIER TWEET

Hil haaaaaaan !
le 12 mai à 23h12

MES PHOTOS



MES AMIS

- ▶ Pupi le lapin
- ▶ Mr Baobab
- ▶ Kawaii
- ▶ Perceval.eu
- ▶ Belette

Bootstrap

Présentation



- ❑ **Bootstrap** est un framework **CSS** développé en **2011** par Marc Otto et Jacob Thornton
- ❑ L'objectif est de **faciliter** la création de pages au **design** abouti et au comportement parfaitement **responsive**
- ❑ Ce **framework** est majoritairement composé de **fichiers CSS** qui contiennent un large ensemble de **classes** pour lesquelles sont définies de nombreuses **règles de style**
- ❑ **Bootstrap** propose toute une palette d'**éléments** préconçus appelés **composants**. Il suffit à l'utilisateur de choisir les **classes** qui l'intéressent pour appliquer le **style défini** à ses éléments **HTML**.

Présentation

- ❑ L'intégration de **Bootstrap** dans un projet peut se faire en **téléchargeant** les fichiers nécessaires ou en utilisant un **CDN** depuis une balise **<link>** de notre fichier **HTML**
- ❑ Certains **composants** Bootstrap vont également nécessiter l'utilisation de **JavaScript** pour leur **fonctionnement**, il faudra alors également charger certains **fichiers supplémentaires**
- ❑ La **version 5** de Bootstrap, maintenant disponible depuis **juin 2020**, comporte quelques **changements majeurs**
- ❑ <https://getbootstrap.com/> | <https://getbootstrap.com/docs/4.3/getting-started/introduction/>
- ❑ <https://t-php.fr/78-bootstrap-5-nouveautes.html>

Présentation

- ❑ Bootstrap étant **responsive**, il permet de définir les **comportements** des éléments en fonction des **tailles d'écran**
- ❑ Une série de **préfixes** est donc proposée, correspondant chacune à une **taille d'écran** minimum ou **breakpoint**
- ❑ **Bootstrap** est écrit selon le modèle **mobile first**, le préfixe généralement utilisé **par défaut** est donc **.xs**

Breakpoint	Class infix	Dimensions
X-Small	<i>None</i>	<576px
Small	sm	≥576px
Medium	md	≥768px
Large	lg	≥992px
Extra large	xl	≥1200px
Extra extra large	xxl	≥1400px

La grille Bootstrap

- ❏ La **disposition** générale des **éléments** se fait selon un système de **grille** à **12 colonnes** fonctionnant sur le modèle des boîtes flexibles **flexbox**
- ❏ Deux types de **conteneur**, représentés par les classes **.container** et **.container-fluid**, sont disponibles pour cette **grille**
- ❏ Un conteneur défini avec la classe **.container** occupe toute la **largeur** de l'écran sur un **mobile**, puis à partir de **576px** une **marge horizontale** vient s'ajouter
- ❏ Un conteneur défini avec **.container-fluid** occupera toujours toute la **largeur** disponible quelque soit la **taille de l'écran**

La grille Bootstrap

```
.container {  
  width: 100%;  
  padding-right: 15px;  
  padding-left: 15px;  
  margin-right: auto;  
  margin-left: auto;  
}  
  
@media (min-width: 576px) {  
  .container {  
    max-width: 540px;  
  }  
}  
  
@media (min-width: 768px) {  
  .container {  
    max-width: 720px;  
  }  
}  
  
@media (min-width: 992px) {  
  .container {  
    max-width: 960px;  
  }  
}  
  
@media (min-width: 1200px) {  
  .container {  
    max-width: 1140px;  
  }  
}
```

exemple de règles prédéfinies
pour la classe `.container`

La grille Bootstrap

- ❏ Pour créer une **grille**, il suffit de donc définir un élément **conteneur** en utilisant l'une des classes **.container** ou **.container-fluid**
- ❏ A l'intérieur de ce **container**, la classe **.row** nous permettra de créer les différentes **lignes** de notre **grille**
- ❏ Puis la classe **.col** sera utilisée pour chacun des **éléments** contenus à l'intérieur de chaque **ligne** de la grille pour **répartir** entre eux les **12 colonnes** disponibles

<code><div class="col-2"> </div></code>	→	2 colonnes sur 12
<code><div class="col-6"> </div></code>	→	6 colonnes sur 12
<code><div class="col-4"> </div></code>	→	4 colonnes sur 12

La grille Bootstrap

- ❏ L'ajout d'un des préfixes **sm** / **md** / **lg** / **xl** dans le nom de la **classe** va permettre de **répartir** le nombre de **colonnes** selon les **tailles d'écrans**

`<div class="col-md-2"> </div>` → 2 colonnes à partir de 768px

- ❏ **Par défaut**, un élément aura le **préfixe xs** (celui prévu pour les écrans **mobiles**) et utilisera alors les **12 colonnes** soit toute la **largeur** disponible
- ❏ Les **lignes** d'une grille **Bootstrap** possèdent la propriété **flex : wrap** par défaut. Si les **éléments** qu'elles contiennent sont trop **nombreux** ou trop **larges**, ils vont par défaut se placer dans une **nouvelle ligne** en-dessous

Style et mise en page

- ❑ Bootstrap dispose de différentes catégories de classes permettant d'agir sur style des textes de nos éléments. L'ajout d'un des préfixes **sm / md / lg / xl** à l'une de ces classes permettra de le faire de manière responsive

- ❑ L'épaisseur des polices peut être modifiée en utilisant les différentes classes commençant par **.fw-**

```
<p class="fw-lighter"> </p>
```

- ❑ La casse des polices peut aussi être modifiée avec l'utilisation de plusieurs classes commençant par **.text-**

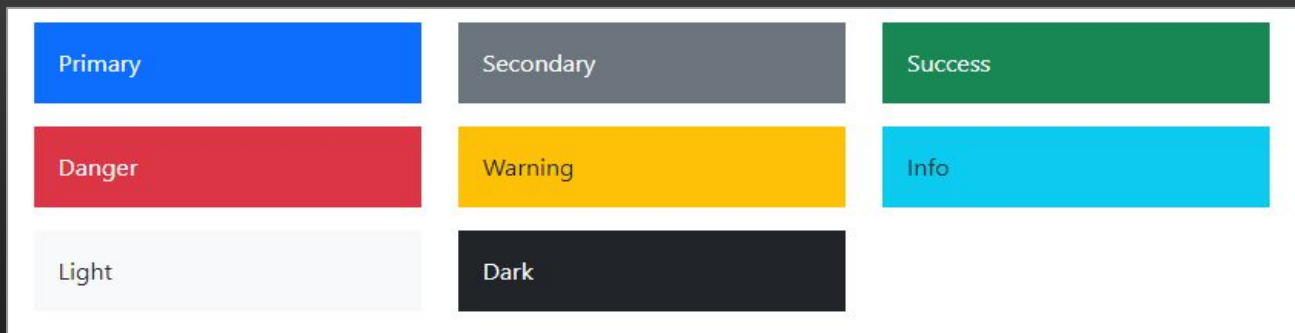
```
<p class="text-lowercase"> </p>
```

- ❑ D'autres classes commençant également par **.text-** vont, elles, permettre de gérer l'alignement d'un texte

```
<p class="text-center"> </p>
```

Style et mise en page

- ❑ **Bootstrap** dispose d'une **palette de couleurs** spécifiques, **accessibles** à l'aide de différentes **classes**
- ❑ L'ajout du préfixe **.text-** permet d'utiliser ces **couleurs** pour les **textes** et l'ajout du préfixe **.bg-** permet de les utiliser pour les **couleurs de fond** de nos éléments
- ❑ Quelques **classes** supplémentaires sont disponibles pour la **couleur** des **textes** comme **.text-black-50** (texte noir semi transparent) ou **.text-white-50** (texte blanc semi transparent)



Style et mise en page

- ❏ **Bootstrap** propose également des groupes de **classes** pour agir sur les **dimensions** de nos **éléments** avec cette même possibilité d'**ajouter** l'un des préfixes **sm / md / lg / xl** pour le faire de manière **responsive**
- ❏ Le groupe de **classes .w-** permet de définir la **largeur** d'un élément par rapport à la **taille** de son **parent**
`<div class="w-75"> </div>` → 75% de la largeur de son parent
- ❏ Le groupe de **classes .h-** permet de définir la **hauteur** d'un élément par rapport à la **taille** de son **parent**
`<div class="h-25"> </div>` → 25% de la hauteur de son parent
- ❏ Les classes **.vw-100** et **.vh-100** permet de donner à un élément la **largeur** et la **hauteur** du **viewport**
`<div class="vw-100"> </div>` → 100% de la largeur du viewport

Style et mise en page

- ❏ **Bootstrap** propose de nombreuses **classes** permettant de gérer les **bordures**, il suffit alors d'**ajouter** les différents noms des **classes** que l'on souhaite utiliser

```
<p class="border border-warning border-3 rounded" > </p>
```



Lorem ipsum dolor, sit amet consectetur adipisicing elit. Sed sit necessitatibus iure nihil. In ad rem eos aperiam odit ipsa ut, ab hic quisquam culpa sit tempore atque sapiente sequi aut aspernatur laboriosam eius quibusdam impedit iusto nesciunt dolor id odio. Tenetur aliquid et error ea omnis dolorum quas rem?

- ❏ Toutes ces **classes** sont consultables **ici** : <https://getbootstrap.com/docs/5.0/utilities/borders/>

Style et mise en page

- ❑ **Bootstrap** propose également de nombreuses **classes** permettant de gérer les propriétés **margin** et **padding** de nos différents **éléments**
- ❑ les classes **p-** et **m-** concernent les **padding** et **margin**, puis l'ajout des préfixes **t-** / **b-** / **l-** / **r-** permet de **cibler** précisément quel **côté** de la marge
- ❑ Les préfixes **-x-** et **-y-** ciblent l'axe **horizontal** ou **vertical**, puis il suffit d'ajouter un **préfixe** entre **0** et **5** ou bien le préfixe **auto** pour préciser la **taille** de l'espacement voulu

`<p class="pt-3 m-x-auto" > </p>` → padding-top = 1rem et margin horizontal = auto

- ❑ Gérer tous ces **préfixes** de manière **responsive** se fait selon le modèle : **{property}{sides}-{breakpoint}-{size}**

`<p class="pt-3-md m-x-md-auto" > </p>` → à partir de 768px

Style et mise en page

- ❑ L'affichage des éléments géré en CSS via la propriété `display` est géré dans Bootstrap avec le préfixe `d-` qui accepte les mêmes valeurs : `none` / `inline` / `block` / `inline-block`...
- ❑ Tout élément peut donc également être transformé en un conteneur flexbox avec la classe `d-flex` et cela peut être organisé de façon responsive avec l'ajout des préfixes `sm` / `md` / `lg` / `xl`
- ❑ L'ensemble des propriétés disponibles en CSS pour gérer le système flexbox le sont également avec Bootstrap
- ❑ Excellent tuto à ce sujet : <https://www.pierre-giraud.com/bootstrap-apprendre-cours/affichage-display-flex/>
- ❑ La documentation officielle est aussi très précise : <https://getbootstrap.com/docs/5.0/utilities/flex/>

Style et mise en page

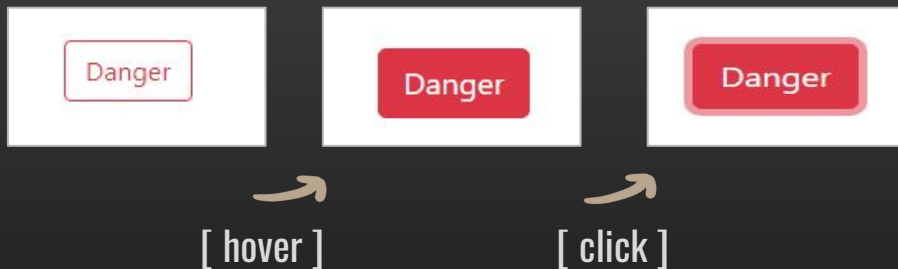
- ❑ Le système de **positionnement** des éléments, géré en **CSS** via la propriété **position**, est entièrement disponible en utilisant les différentes **classes** ayant le préfixe **position-**
- ❑ L'affichage des **images** de manière totalement **responsive** est possible avec les **classes** ayant le préfixe **img-**
- ❑ Quant aux **débordements** ils peuvent être gérés en utilisant différentes **classes** ayant le préfixe **overflow-**
- ❑ La propriété CSS **float** est également accessible en utilisant différentes **classes** ayant le préfixe **float-**

Les composants Bootstrap

- ❑ Bootstrap propose également un large choix d'éléments de formulaires ayant tous un comportement responsive
- ❑ Des visuels sont aussi proposés pour accompagner les systèmes de validation utilisés dans les champs de saisie des formulaires
- ❑ <https://getbootstrap.com/docs/5.0/forms/overview/> | <https://getbootstrap.com/docs/5.0/forms/validation/>

Les composants Bootstrap

- ❑ Bootstrap propose aussi un ensemble de **boutons** répondant toujours aux mêmes codes **couleur** et qui possèdent tous un **effet** lors du **hover** et lors du **click**
- ❑ Des **spinners**, des **fenêtres modales**, des **popovers** ou encore des modèles de **badges** sont également disponibles



Les composants Bootstrap

- ❑ Des composants **interactifs** tels que des **accordéons**, des **carrousels** ou encore des **navbars** avec des **menus déroulants** sont disponibles
- ❑ Pour faire fonctionner des **composants** nécessitant une **animation**, l'utilisation de **JavaScript** est nécessaire
- ❑ Une balise **<script>** fournie par **Bootstrap** doit alors être placée juste **avant** la balise fermante **</body>**

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-gtEjrD/SeCtmISkJKNUaaKMoLD0//ElJ19smozuHV6z3Iehds+3U1b9Bn9Plx0x4" crossorigin="anonymous"></script>

</body>
```