

Programmation Web Front-end

HTML, CSS, JavaScript, Référencement Naturel SEO

Leçon 3 : CSS 3

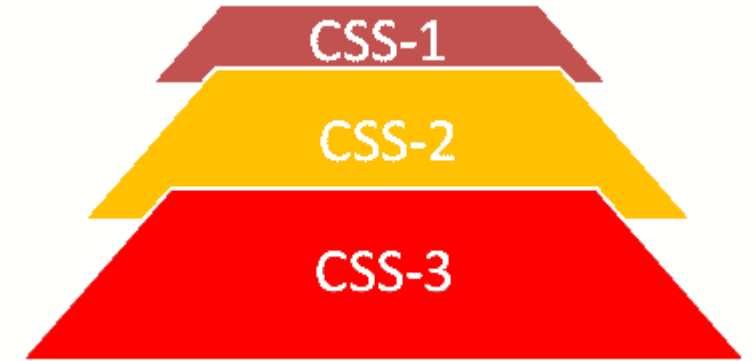
Les bases du CSS

- L'évolution du CSS
- La syntaxe CSS
- Les sélecteurs et les propriétés CSS
- Le responsive design

CSS

- **CSS : Cascading Style Sheets**, aussi appelées **Feuilles de style en Cascade**
- Son rôle est de mettre en forme une page web et gérer l'apparence des éléments de cette page web (agencement des éléments (flexible,...), positionnement, décoration, couleurs, taille du texte, animation...).
- Ce langage est venu compléter le HTML en 1996.
- L'extension de fichier «**.css**», tirant son nom de l'abréviation de l'anglicisme «Cascading Style Sheet», est un format de fichier contenant un code CSS.

L'évolution du CSS



Les différentes versions de CSS

- **CSS 1** : dès 1996, on dispose de la première version du CSS. Elle pose les bases de ce langage qui permet de présenter sa page web, comme les couleurs, les marges, les polices de caractères, etc.
- **CSS 2** : apparue en 1999 puis complétée par CSS 2.1, cette nouvelle version de CSS rajoute de nombreuses options. On peut désormais utiliser des techniques de positionnement très précises, qui nous permettent d'afficher des éléments où on le souhaite sur la page.
- **CSS 3** : c'est la dernière version, qui apporte des fonctionnalités particulièrement attendues comme les bordures arrondies, les dégradés, les ombres, etc.

Nouveautés CSS 3

- Ombres de texte et de boîte
 - définir facilement la couleur (shadow color), l'angle et le niveau de flou de l'ombre (blur level).
- Opacité
 - définir le niveau d'opacité d'un élément pour le rendre totalement opaque (full opacity), transparent ou see-through.
- Style de bordure et angles arrondis
 - inclut de nouvelles fonctionnalités de style de bordure, telles que le rayon de la bordure (border radius), la tranche d'image (image slice) et la source de l'image (image source), ainsi que des valeurs pour l'étirement de la largeur (width stretch).
 - Éléments aux angles arrondis (rounded corners)
 - utiliser la propriété border-radius pour arrondir les coins d'un élément et également définir le rayon du coin (corner radius).

Nouveautés CSS 3

- Transitions et Animations avancées
 - il existe plusieurs propriétés d'animation simples et complexes que vous pouvez utiliser pour définir comment et quand un élément de page Web doit s'animer.
- Transformation 3D
 - appliquer des effets et la mise à l'échelle 3D, rotations ...
- Propriétés de style d'arrière-plan (Background Style Properties)
 - plusieurs nouvelles fonctionnalités CSS3 ont été ajoutées, notamment les propriétés de découpage d'arrière-plan (background clipping), de style, de taille (size) et d'origine (origin).
- Media queries
 - intégrer des requêtes multimédia qui permettent aux développeurs de définir un bloc contenant des propriétés CSS à appliquer uniquement si une condition spécifique devient vraie.
- Combinateur frère
 - (à voir après dans ce cours)

Effet du Cascading

Pourquoi le mot 'Cascading'?

- Il est possible que deux sélecteurs désignent le même élément HTML, et que ces deux sélections appliquent des règles différentes qui rentrent en conflit.
- L'objectif est de contrôler la manière dont le CSS est appliqué au HTML et la manière dont ces conflits sont résolus

Comment?

- En fonction de l'ordre d'apparition des règles dans le CSS
- Quand deux ou plusieurs règles avec la même spécificité (meme sélection) sont définies : **la dernière règle apparue dans le fichier CSS sera appliquée**

Effet du Cascading

Pourquoi le mot 'Cascading'?

L'effet du cascading nous permet d'arbitrer entre plusieurs sources concurrentes de mise en forme d'un même élément.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>CSS</title>
5   <link rel="stylesheet" type="text/css" href="">
6 </head>
7 <body>
8   <header>
9     <nav>
10      <ul>
11        <li>Home</li>
12        <li><a href="about.html">About</a></li>
13        <li><a href="login.html">Login</a></li>
14      </ul>
15    </nav>
16  </header>
17  <section>
18    <h2>Home</h2>
19    <p>Lorem ipsum.</p>
20  </section>
21 </body>
22 </html>
```

• Home
• [About](#)
• [Login](#)

Home

Lorem ipsum.

Le paragraphe est
affiché en vert et non
pas en rose

```
1 /*Selector {
2   Property: value;
3 }*/
4
5 h2 {
6   color: red;
7 }
8
9 p {
10  color: pink;
11 }
12
13 p {
14  color: green;
15 }
```


Syntaxe CSS

Selector {

Property : value;

}

- Un sélecteur cible un élément HTML.
- Un couple propriété:valeur est appelé une déclaration.
- Chaque déclaration doit se terminer par un point-virgule.
- Une règle peut contenir une ou plusieurs déclarations, appelée aussi un Style.
- Une feuille de style CSS contient plusieurs blocks de ce type

Ceci est une règle CSS

```
h1 { font-weight : bold; font-size: 2em; }
```

Ceci est un bloc de déclaration CSS

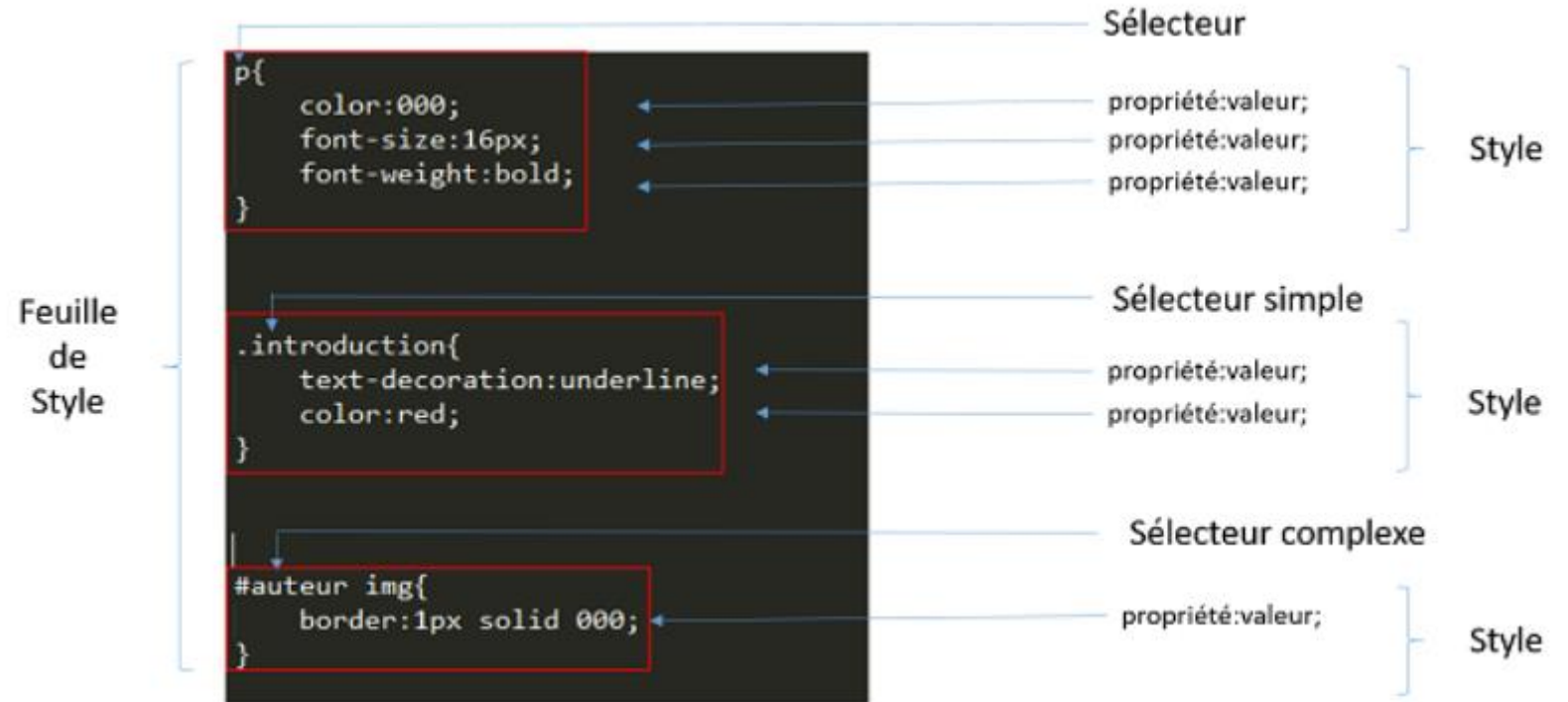
```
h1 { font-weight : bold; font-size: 2em; }
```

```
h1 { font-weight : bold; font-size: 2em; }
```

sélecteur propriété valeur

Syntaxe CSS

- Un sélecteur simple utilise le méta mot de l'élément HTML
- Un sélecteur complexe contient une combinaison de méta mot d'éléments HTML



Syntaxe CSS

- Quelques exemples :

Text Color

```
body {  
    color: blue;  
}  
  
h1 {  
    color: #00ff00;  
}  
  
h2 {  
    color: rgb(255,0,0);  
}
```

Text Alignment

```
h1 {  
    text-align: center;  
}  
  
p.date {  
    text-align: right;  
}  
  
p.main {  
    text-align: justify;  
}
```

Background Color

```
body {  
    background-color: lightblue;  
}
```

Text Font

```
p {  
    font-family: "Times New Roman", Times, serif;  
}
```

Text Size

```
h1 {  
    font-size: 40px;  
}  
  
h2 {  
    font-size: 30px;  
}  
  
p {  
    font-size: 14px;  
}
```

Trois façons pour ajouter du code CSS

Styles Inline

```
<p style = "font-size: 20pt; color: deepskyblue;">  
This text has the <em>font-size</em> and  
<em>color</em> styles applied to it, making it  
20pt and deep sky blue.</p>
```

Styles Incorporés

```
<head>  
  <meta charset = "utf-8">  
  <title>Embedded Style Sheet</title>  
  
  <!-- this begins the style sheet section -->  
  <style type = "text/css">  
    em    { font-weight: bold;  
            color: black; }  
    h1    { font-family: tahoma, helvetica, sans-serif; }  
    p     { font-size: 12pt;  
            font-family: arial, sans-serif; }  
    .special { color: purple; }  
  </style>  
</head>
```

Styles Externes associés

```
<head>  
  <meta charset = "utf-8">  
  <title>Linking External Style Sheets</title>  
  <link rel = "stylesheet" type = "text/css"  
        href = "styles.css">  
</head>
```

Sélecteurs

- Sélecteurs de balises

HTML

<h2> Web Design </h2>

CSS

```
<style>
h2 {
  color: white;
}
</style>
```

Sélecteurs

- Sélecteurs universel (*)

```
* {  
  font-family: "Times New Roman";  
}
```

Sélecteurs

- Sélecteurs de classe

HTML

class:
<h2 class="title"> Web Design </h2>

CSS

```
<style>
.title {
  color: white;
}
</style>
```

- Sélecteurs d'ID

HTML

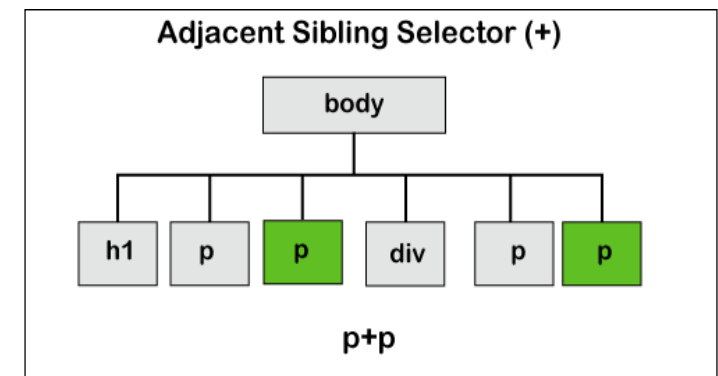
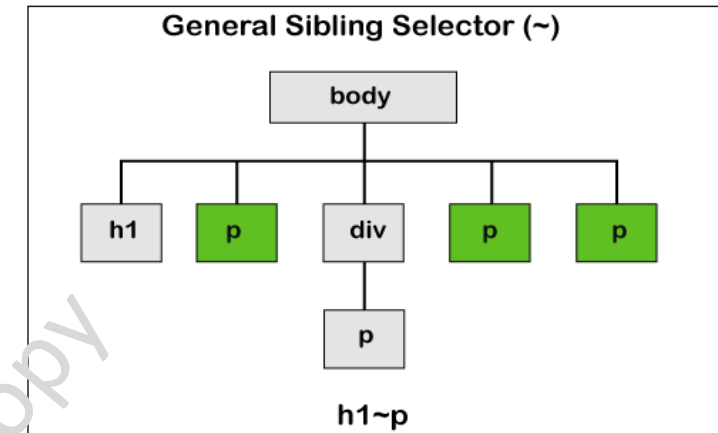
id:
<h2 id="title"> Web Design </h2>

CSS

```
<style>
#title {
  color: white;
}
</style>
```

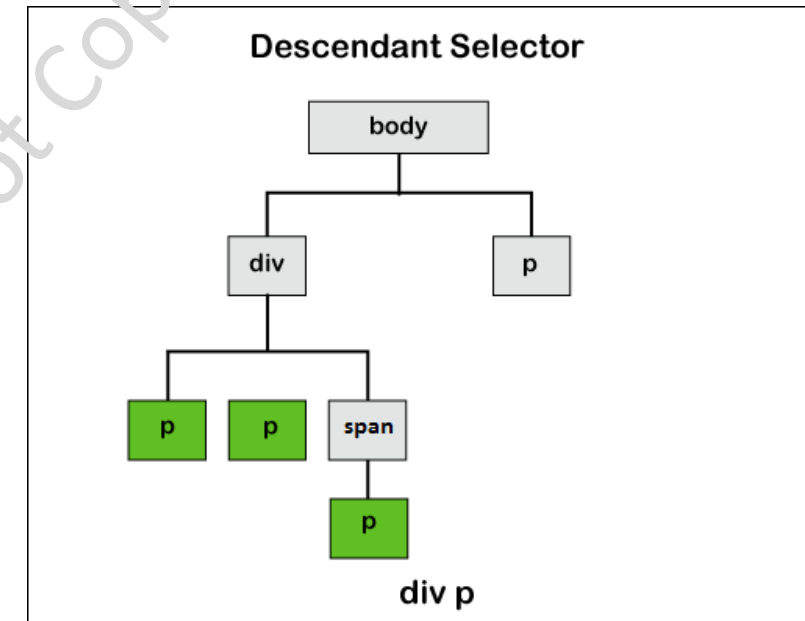
Sélecteurs avec Combinateur frère

- Combinateur frère (general sibling combinator ~):
 - le combineur général frère (via le combineur tilde (~)) sépare deux sélecteurs et correspond à toutes les itérations du deuxième élément, qui suivent le premier élément (mais pas nécessairement immédiatement), et sont des enfants du même élément parent.
- Combinateur adjacent (adjacent sibling combinator +):
 - le combineur frère adjacent (+) sépare deux sélecteurs et correspond au deuxième élément uniquement qui suit immédiatement le premier élément, et les deux sont des enfants du même élément parent.



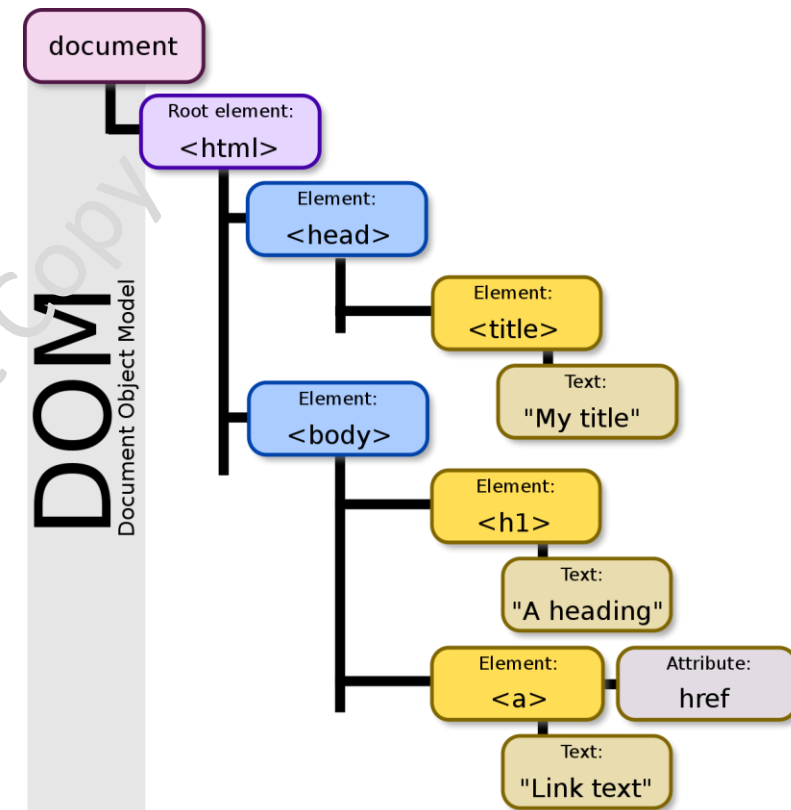
Sélecteur descendant

- Sélecteur descendant CSS:
 - utilisé pour faire correspondre les éléments descendants d'un élément particulier et le représenter à l'aide d'un espace unique. Le mot descendant indique imbriqué n'importe où dans l'arborescence DOM (DOM tree).



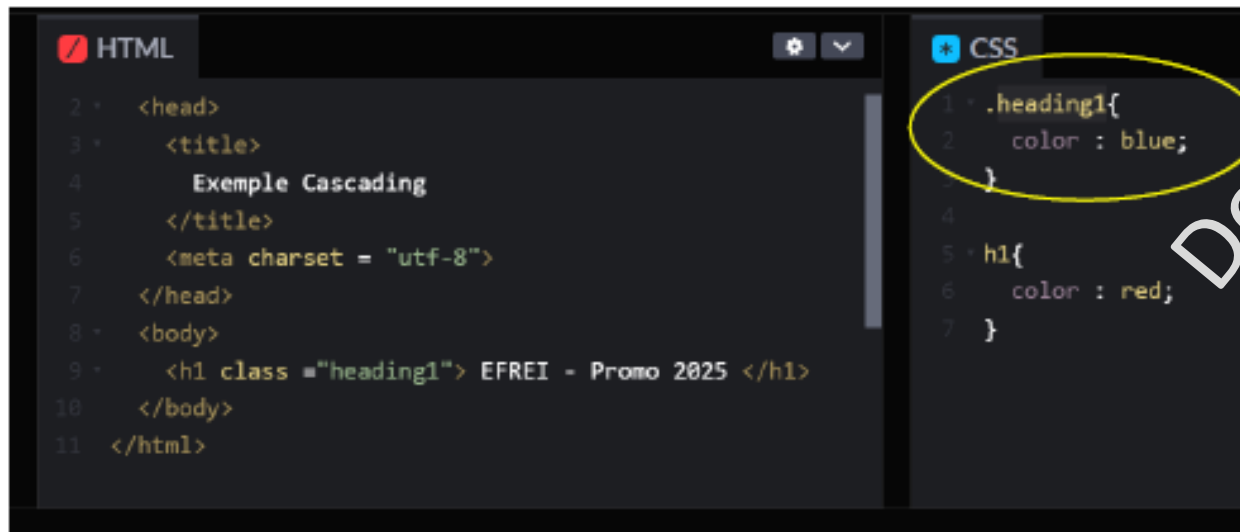
Arborescence DOM

- Le modèle d'objet de document (Document Object Model)
- est une interface indépendante du langage qui traite un document HTML (ou XML) comme une **structure arborescente** dans laquelle **chaque nœud est un objet représentant une partie du document**.
- Le DOM représente un document avec une arborescence logique.



Spécificité des Sélecteurs

- La spécificité mesure combien la sélection est précise
 - Sélecteur **peu spécifique** : cible tous les éléments d'un type donné score faible
 - Sélecteur **plus spécifique** : ne cible que les éléments dont l'attribut class a la valeur choisie score important



```
HTML
1 <head>
2   <title>
3     Exemple Cascading
4   </title>
5   <meta charset = "utf-8">
6 </head>
7 <body>
8   <h1 class ="heading1"> EFREI - Promo 2025 </h1>
9 </body>
10 </html>

CSS
1 .heading1{
2   color : blue;
3 }
4
5 h1{
6   color : red;
7 }
```

→ EFREI - Promo 2025

Héritage CSS

- Certaines valeurs pour une propriété CSS se transmettent des éléments parents vers leurs enfants, d'autres non.
- **Attention:** Les propriétés telles que largeur, marges, remplissage, et bordures ne se transmettent pas par héritage.



```
HTML
<head>
  <title>
    Exemple Cascading
  </title>
  <meta charset = "utf-8">
</head>
<body>
  <h1> EFREI - <strong> Promo 2025</strong> </h1>
</body>
</html>

CSS
1 strong{
2   color : blue;
3 }
4
5 h1{
6   color : red;
7 }
```

→ EFREI - Promo 2025

Héritage CSS

- **Inherit** : héritage activé
- **Initial** : valeur par défaut
- **Unset** : redéfinit la propriété à sa valeur naturelle

```
<body>
  <h1> Efrei - <strong> Promo 2025 </strong></h1>
  <ul>
    <li>Web Programming - <strong>WEB</strong></li>
    <li class="inherit">Databases - <strong>DB</strong></li>
    <li class="reset">Algorithmics & Data structures - <strong>SDD</strong></li>
    <li class="unset">C Prog. - <strong>C</strong></li>
  </ul>
</body>
```

```
h1{color: red;}
body{color: green;}
strong{color: blue;}
.inherit strong{color:inherit;}
.reset strong{color:initial;}
.unset strong{color:unset;}
```

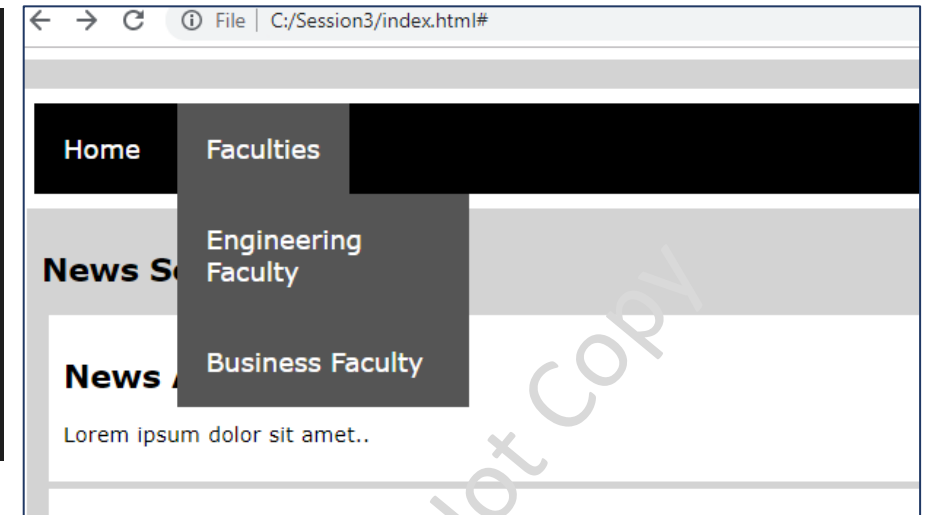
Efrei - Promo 2025

- Web Programming - **WEB**
- Databases - **DB**
- Algorithmics & Data structures - **SDD**
- C Prog. - **C**

Quelques exemples de sélecteurs complexes

Code HTML du menu

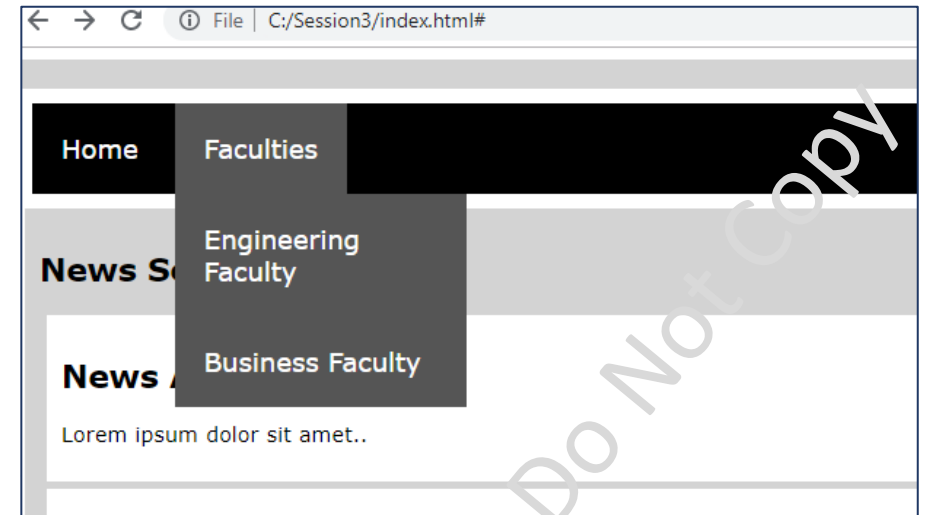
```
<nav>
  <ul>
    <li><a href="index.html">Home</a></li>
    <li>
      <a href="#">Faculties</a>
      <ul>
        <li><a href="https://abc.edu.fr/engineering.html">Engineering Faculty</a></li>
        <li><a href="https://abc.edu.fr/business.html">Business Faculty</a></li>
      </ul>
    </li>
  </ul>
</nav>
```



Quelques exemples de sélecteurs complexes

Code HTML du menu

```
<nav>
  <ul>
    <li><a href="index.html">Home</a></li>
    <li>
      <a href="#">Faculties</a>
      <ul>
        <li><a href="https://abc.edu.fr/engineering.html">Engineering Faculty</a></li>
        <li><a href="https://abc.edu.fr/business.html">Business Faculty</a></li>
      </ul>
    </li>
  </ul>
</nav>
```



```
nav {
  margin: 10px;
  background-color: black;
}
ul {
  display: inline;
  margin: 0;
  padding: 0;
}
```

```
ul li {
  display: inline-block;
}
ul li: hover {
  background: #555;
}
ul li: hover ul {
  display: block;
}
```

```
ul li ul {
  position: absolute;
  width: 200px;
  display: none;
}
ul li ul li {
  background: #555;
  display: block;
}
```

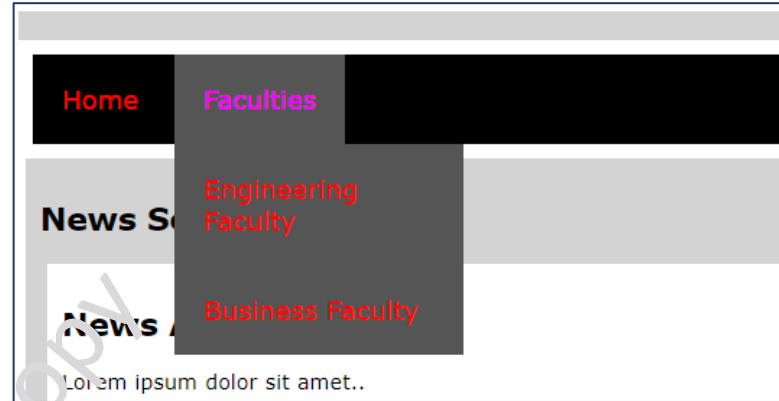
```
ul li ul li a {
  display: block;
}
ul li ul li: hover {
  background: #666;
}
```

```
nav a {
  text-decoration: none;
  color: white;
  font-size: 18px;
  padding: 20px;
  display: inline-block;
}
```

Sélecteur pseudo-classe

- Une pseudo-classe est un mot-clé qui peut être ajouté à un sélecteur afin d'indiquer l'état spécifique dans lequel l'élément doit être pour être ciblé par la déclaration.
- E.g. La pseudo-classe **:hover**, permettra d'appliquer une mise en forme spécifique lorsque l'utilisateur survole l'élément ciblé par le sélecteur (changer la couleur d'un bouton par exemple).

Sélecteur pseudo-classe



- Attention: l'ordre des pseudo-classes est important afin de les rendre efficaces.
 - a:hover doit être ajouté après a:link et a:visited
 - a:active doit être ajouté après a:hover
- Les noms des pseudo-classes ne sont pas case-sensitive c.à.d pas sensibles à la casse (aux majuscules et minuscules).

unvisited link

```
a:link {  
  color: #FF0000;  
}
```

#ff0000
Red

visited link

```
a:visited {  
  color: #00FF00;  
}
```

#00ff00
Lime

mouse over link

```
a:hover {  
  color: #FF00FF;  
}
```

#ff00ff
Magenta

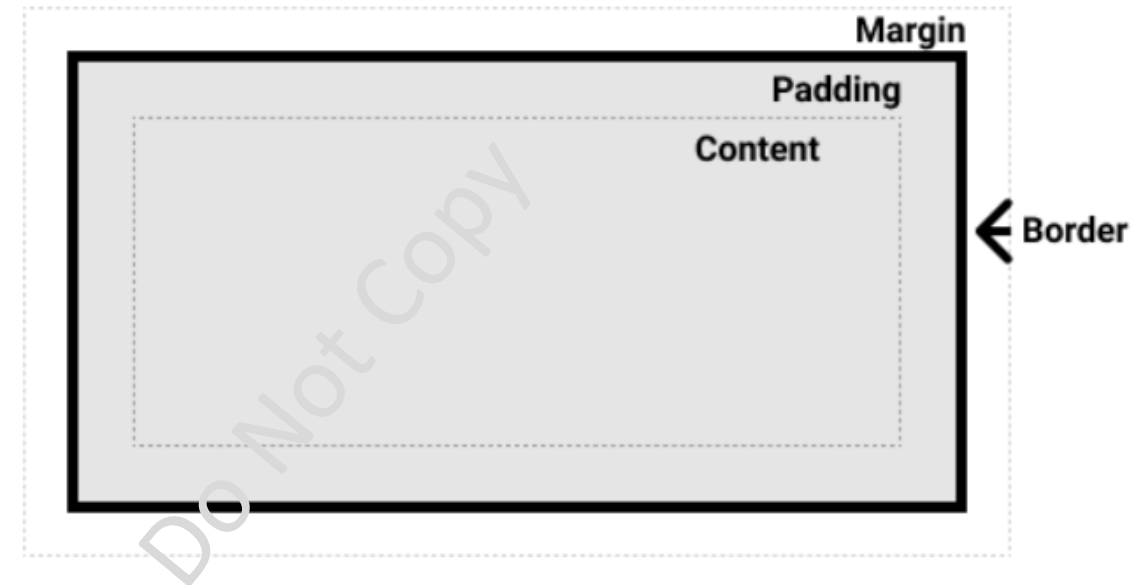
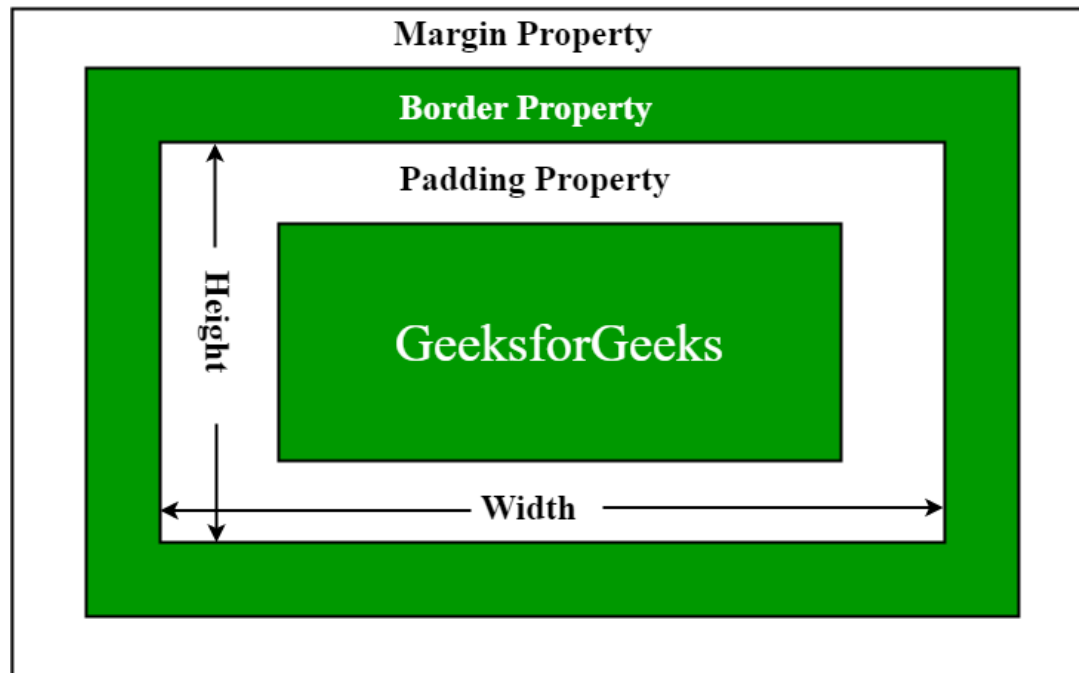
selected link

```
a:active {  
  color: #0000FF;  
}
```

#0000ff
Blue

Propriétés d'une Boite CSS

CSS Box-Model Property



https://developer.mozilla.org/en-US/docs/Web/CSS/Containing_block

Div vs Container vs Section

Div

- est un élément HTML sans restrictions dont le but est justement de donner du sens aux contenus. On peut créer des Div imbriqués.
- en pratique, on utilisera souvent l'élément div à des fins de mise en forme en CSS.
- Ne pas confondre avec la balise **** : l'élément div est un conteneur de niveau bloc tandis que l'élément span est un conteneur en ligne (inline) et va donc plutôt être utilisé pour contenir des bouts de texte.
- L'élément div ne possède pas d'attribut particulier. Il supporte les attributs universels comme l'ensemble des éléments HTML. Quelques exemples d'attributs universels communs qui peuvent être ajoutés aux éléments HTML : class, id, style, title...

Container

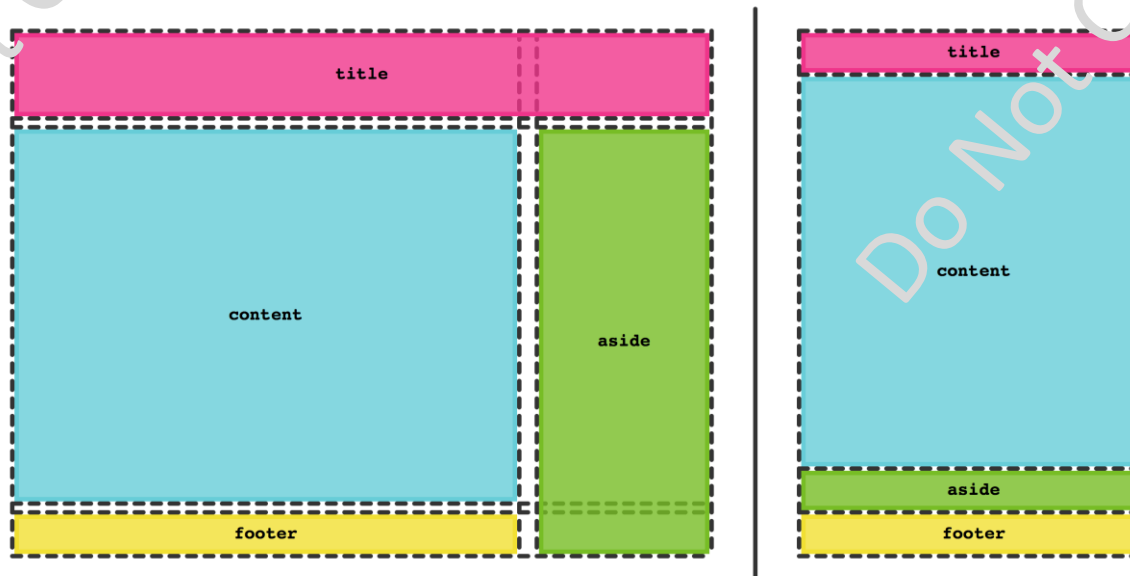
- ressemble à un Div mais verrouillé à 940 pixels de largeur et placé automatiquement au centre de la page.
- Container est utilisé beaucoup en Bootstrap.

A ne pas confondre avec **Section**

- Un élément HTML ayant une largeur complète de la fenêtre d'affichage (viewport)
- on ne peut pas avoir de sections imbriquées

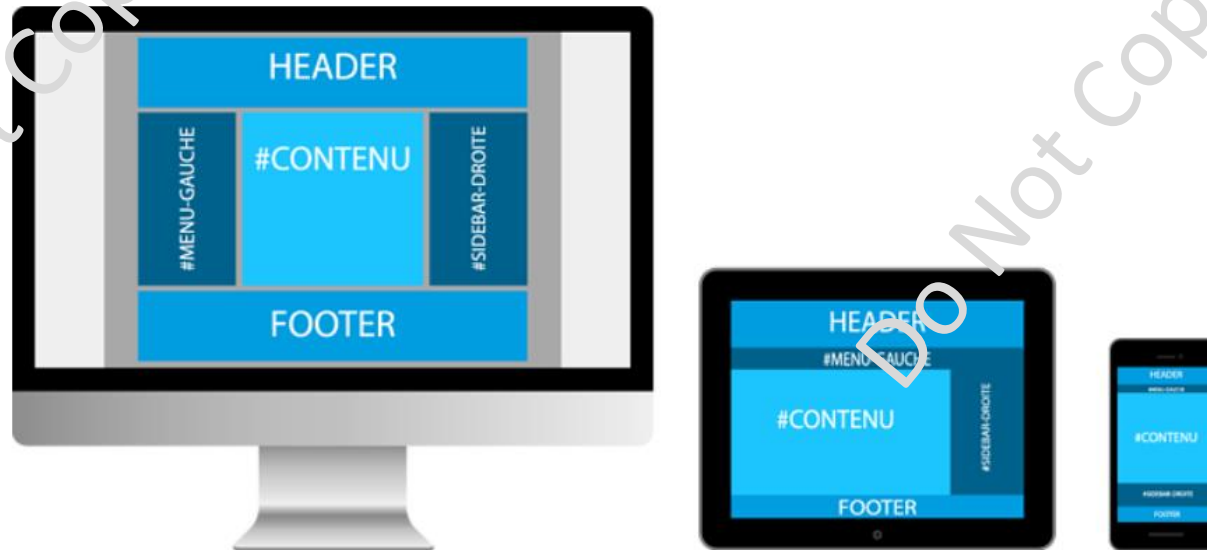
Responsive design

- Le Responsive Design ou plus précisément le Responsive Web Design (RWD) est une technique de conception d'interface digitale qui fait en sorte que **l'affichage d'une quelconque page d'un site s'adapte de façon automatique à la taille de l'écran du terminal qui le lit.**



Responsive design

- Un ensemble des propriétés (de taille et de forme) et des façons de faire qui peuvent être changées pour rendre le site réactif (responsive) pour qu'il s'adapte à tout type d'appareils.



A voir :

<https://www.easybear.fr/blog/reussir-son-responsive-design>

Responsive design vs. Adaptive design

- **Un design Statique** : Un design statique (ou fixe) se réfère à des dimensions figées (par exemple 960px) quelle que soit la surface de l'écran. La grande majorité des sites web était construite sur cette base avant l'arrivée du Responsive Web Design dans les années 2010.
- **Un design Fluide** : Un site web Fluide (ou "liquid") est un site web dont toutes les largeurs de colonnes sont exprimées en unités variables (pourcentages, em, vw, etc.) et qui s'adapte généralement automatiquement à la taille de fenêtre, jusqu'à une certaine mesure.
- **Un design Adaptive** : Un design Adaptatif est une amélioration du design statique : les unités de largeur sont fixes, mais différentes selon la taille de l'écran, qui est détectée via CSS3 Media Queries.
Un tel design tient uniquement compte des principaux points de rupture (320px, 480px, 768px, 1024px, etc.) et adapte le gabarit en conséquence. Au final, on se retrouve avec autant de gabarits fixes que de points de ruptures.
- **Un design Responsive** : Un site web Responsive est une amélioration du design liquide associé à des méthodes CSS3 Media Queries permettant de modifier les styles (ré-organisation de la page par exemple) selon certains critères, pour s'adapter complètement à la taille d'écran, quel que soit le point de rupture.



La responsivité : les étapes clés

1

Concevoir sa mise en page en prenant en compte les contraintes d'un écran de smartphone

2

Mettre en place des directive **viewport** dans l'en-tête de toutes les pages.

3

Définir toutes les dimensions horizontales (largeurs, marges droite et gauche...) en unités relatives (ex : pourcentage).

4

Mettre en place des limites minimale et éventuellement maximale.

5

Définir une directive **media-query** en disposant les éléments au mieux pour un écran plus grand comme celui d'une tablette

6

Définir une autre directive **media-query** pour disposer les éléments pour un écran de bureau.

La responsivité : les étapes clés

- **Déclarer Viewport**

- la fenêtre d'affichage est la zone visible par l'utilisateur d'une page Web.

- La déclaration **viewport** doit être placée dans **TOUTES** les pages HTML du site Web, dans la section **head**.

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

On définit ici que la largeur de l'écran virtuel doit être identique à celle du device, donc de l'écran physique, et que le zoom doit être de 1 (l'internaute pouvant cependant zoomer ou dézoomer s'il le souhaite).

La responsivité : les étapes clés

- Définir les dimensions et les limites

Définir les dimensions en unités relatives : Pour les marges, dimensions des éléments, etc.
L'unité la plus couramment utilisée est le pourcentage

→ Ainsi toutes les dimensions sont proportionnelles à celles de l'écran

Fixer des limites : 4 propriétés permettent de fixer les limites **min-width, max-width, min-height** et **max-height**

→ Contrairement aux autres dimensions, ces valeurs seront définies **en unités absolues** (en pixels par exemple)

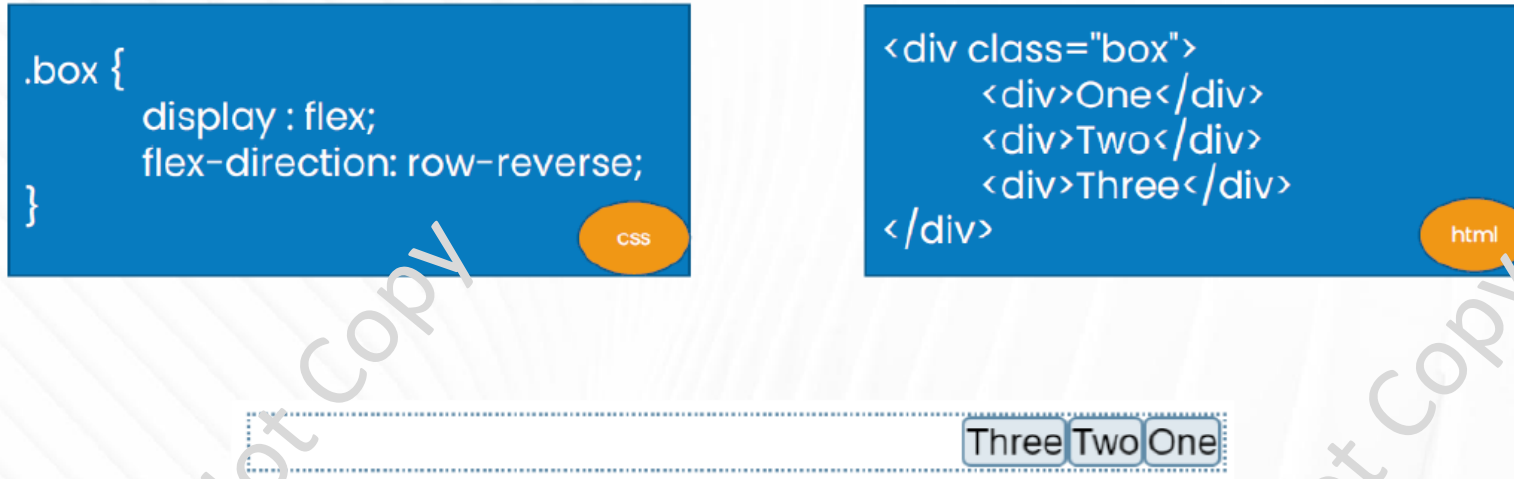
La responsivité : les étapes clés

- Définir des media-queries
- Les **media-queries** permettent, tout en restant sur un terminal du type écran, défiltrer les styles en fonction de la taille de cet écran.
- **@media** associe des règles CSS à un périphérique particulier. En utilisant cette directive, il est donc possible de définir des mises en page spécifiques à chaque périphérique : écran, imprimante, smartphone, etc.

```
/* Règles à appliquer si la page est visionnée sur un écran */
@media screen{
  h1{
    font-size:24pt;
    font-weight:normal;
  }
  nav{
    display:inline-block;
  }
}

/* Règles à appliquer si la page est imprimée */
@media print{
  h1{
    font-size:18pt;
    font-weight:bold;
  }
  nav{
    display:none;
  }
}
```

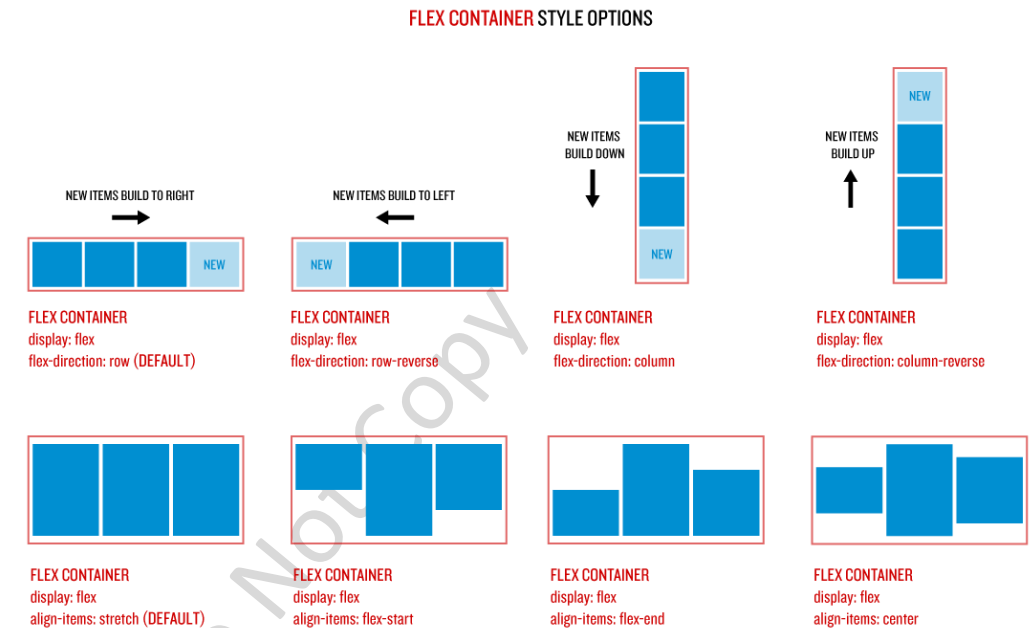
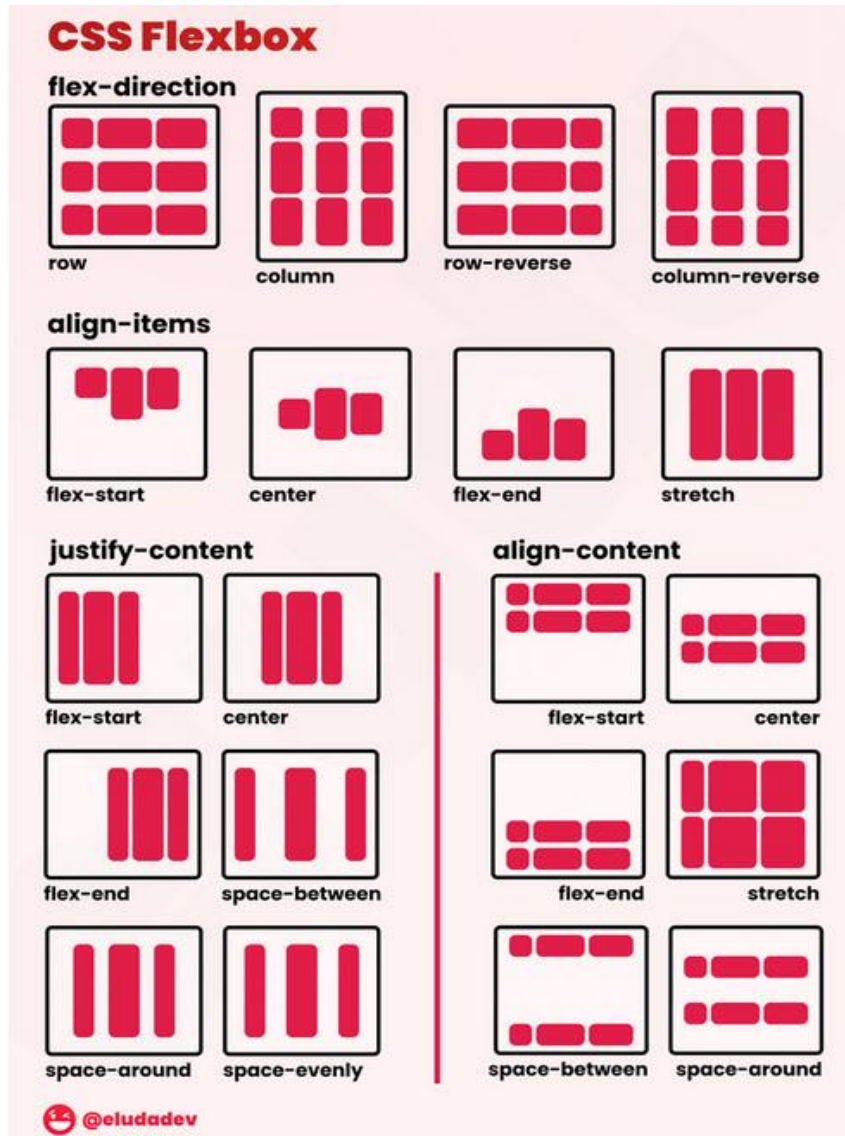
La responsivité : les étapes clés



- **Utiliser des boites flexible**

- Les flex box ou les boites flexibles, est une nouvelle propriété introduite en CSS 3
- Elles rendent la conception adaptative bien plus facile
- Les éléments html sont facilement agencés grâce au "display flex"

Propriétés de Flexbox




www.learnenough.com

<https://modernways.be/myap/it/page/programming/css/CSS%20lay-out%20met%20flexbox.html>

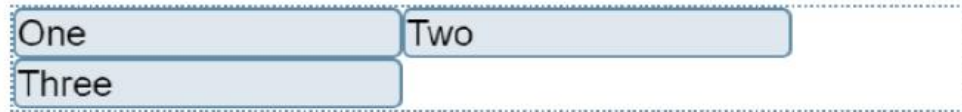

Propriétés de Flexbox

- Exemple: flex-flow : propriété raccourcie de flex direction and flex wrap


```
.box {  
  display : flex;  
  flex-flow: wrap;  
}
```



```
<div class="box">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
</div>
```

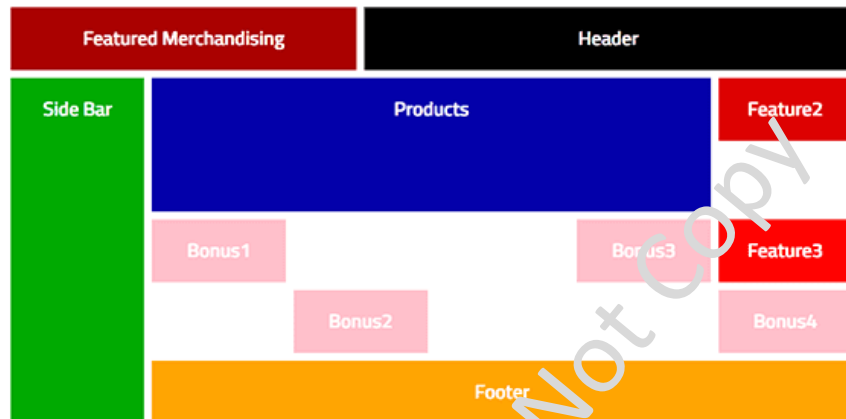


CSS Flexbox vs CSS Grid

CSS Flexbox	CSS Grid
CSS Flexbox is a one-dimensional layout model.	CSS Grid is a two-dimensional model.
A Flexbox container can either facilitate laying out things in a row, or lay them out in a column.	Grid can facilitate laying out items across and down at once.
Flexbox cannot intentionally overlap elements or items in a layout.	CSS Grid helps you create layouts with overlapping elements.
Flexbox is basically content based and it listens to the content and adjusts to it.	Grid operates more on the layout level and it is container based.
Flexbox can be used for scaling, one-sided aligning, and organizing elements within a container.	Grid is useful when you want to define a large-scale layout with more complex and subtle designs.
	

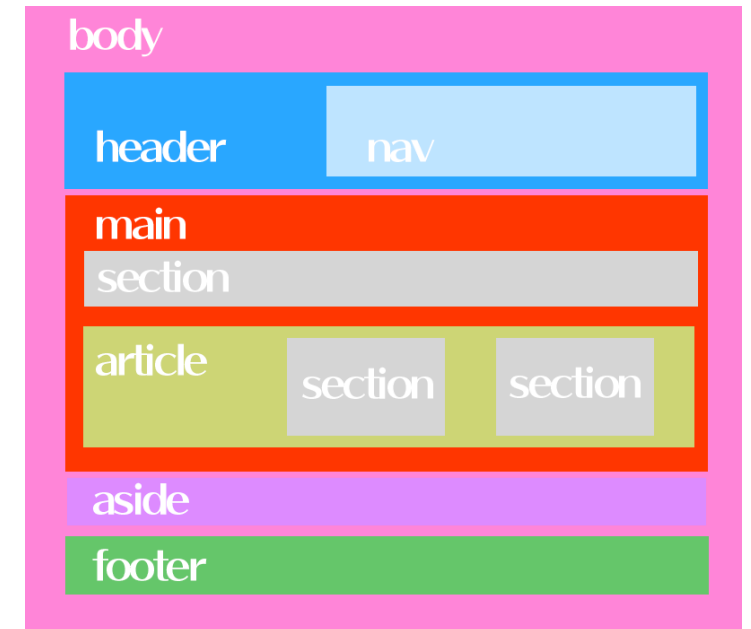
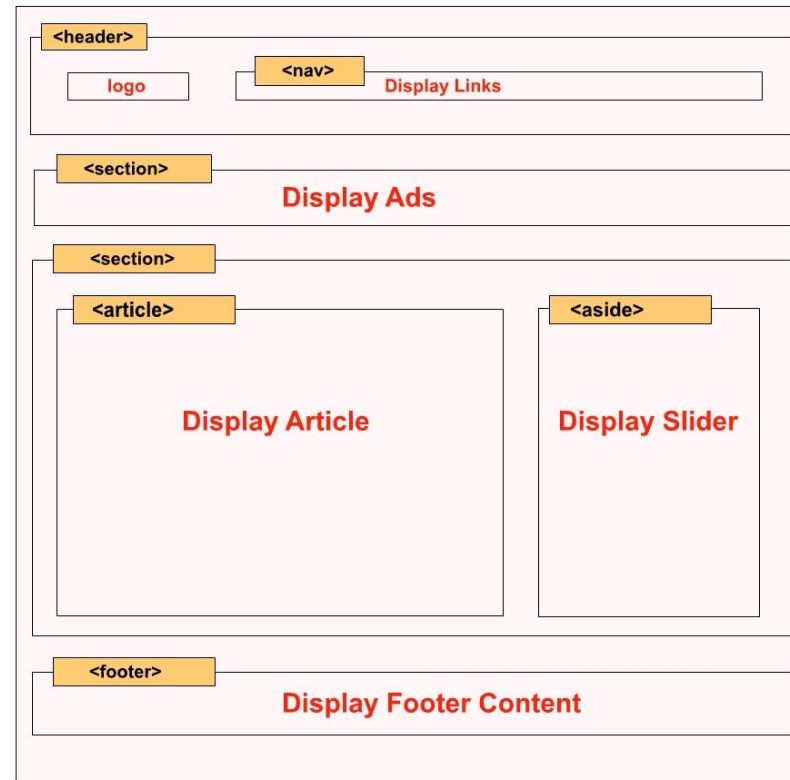
CSS Grid Layout / Mise en page en Grille

- Exemples :



A voir:

<https://materializecss.com/grid.html>



Travail en autonomie

- Continuer l'activité en laboratoire informatique

Do Not Copy

Do Not Copy