

# Les feuilles de style (CSS3)



# Définition de CSS

- Le W3C définit le langage CSS comme le suivant :

« Le **CSS** (Cascading Style Sheets – feuilles de style en cascade) est un langage utilisé par les concepteurs de sites web pour contrôler l'apparence visuelle des documents structurés, tels que les fichiers HTML.

En séparant le contenu de la mise en forme, CSS rend la création et la gestion des pages web plus simples, en s'appuyant sur le principe de distinction entre le contenu et sa présentation visuelle. »

- Grâce à ce langage, nous allons pouvoir créer rapidement et simplement la mise en page du site.
  - **HTML pour écrire le contenu de nos pages web.**
  - **CSS pour présenter ce contenu.**

→**Ces 2 langages sont donc complémentaire**



# Historique

- Conçu initialement pour HTML, la première version dite CSS niveau 1 (CSS-1) est publiée en 1996,
- Les feuilles de styles sont apparues en 1997 avec le browser Internet Explorer 3.0,
- 1998, le W3C publie une nouvelle version CSS-2. Elle introduit la notion de type de média,
- Elles se sont généralisées avec les versions 4.0 d'Internet Explorer et de Netscape Navigator.
- IE 5 gère la quasi totalité des spécifications CSS-2,
- Le site du W3C propose une validation en ligne permettant de vérifier la conformité d'un style à la norme CSS-2.



# Principe de CSS

- Feuilles de style CSS se composent de blocs contenant un ensemble de déclarations, qui sont appliquées à une ou plusieurs balises HTML pour définir l'apparence visuelle de la page web. Ces balises sont sélectionnées à l'aide d'un **sélecteur** CSS placé avant le bloc de déclarations.
- Lors du chargement d'une page HTML, le navigateur crée une structure arborescente représentant l'ensemble des balises de la page, appelée DOM (Document Object Model). Il charge ensuite la ou les feuilles de style CSS et applique progressivement les différentes règles aux éléments du DOM au fur et à mesure qu'il interprète les styles.



# Principe de CSS

- Le problème, lorsqu'on n'utilise pas de feuilles de style, c'est qu'il faut reprendre toutes les pages une à une pour modifier une police de caractère ou une couleur de fond.
- Avec les "Cascading Style Sheets" ([CSS](#)), ce lourd handicap est résolu.
- C'est dans la feuille de style que l'on va déclarer toute la mise en forme des pages : la couleur de fond, les polices de caractère, leurs couleurs, etc. Celle-ci sera liée à chaque page html.
- Ainsi, lorsqu'on en modifiera un élément, les changements apportés à cet élément seront automatiquement appliqués à toutes les pages web qui utilisent ce fichier



# Utilité et avantage

- Séparation du contenu et du style.
- Permet de modifier l'apparence d'une page ou d'un site sans altérer son contenu.
- Réduction du temps de chargement des pages.
- Compense certaines limitations du langage HTML, comme le contrôle des polices, des interlignes, des marges et des indentations, etc.



# Sélecteur, Propriété, Valeur

Une déclaration de style comporte plusieurs parties :

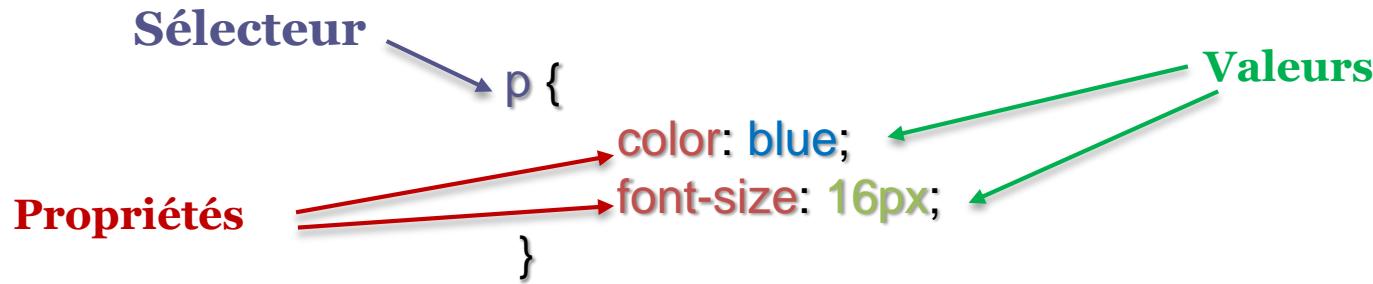
- Un sélecteur indique à quel élément le style doit être appliqué.
- Une propriété spécifie le style à appliquer à cet élément.
- Une valeur définit le comportement associé à une propriété.

**la structure :**

```
sélecteur {  
    propriété1: valeur1;  
    propriété2: valeur2;  
    /* ... autres propriétés ... */  
}
```

- Pour faciliter la lisibilité et la maintenance, il est conseillé d'ajouter des commentaires dans le code CSS avec la syntaxe /\* Commentaire \*/.

# Exemple :



- Tous les paragraphes auront une couleur bleue et une taille de 16 pixels

# Propriétés

- CSS définit plusieurs propriétés permettant de définir la présentation des pages Web :

Propriété	Rôle
color	Couleur du texte
font	Déclaration groupé
font-weight	Graisse
font-size	Taille de police
font-family	Famille de police
font-variant	Variante
font-stretch	Étirement du texte
fext-decoration	Décoration du texte

Tableau 1 : Quelques propriétés pour les textes

Propriété	Rôle
Background	Déclaration groupé
Background-color	Couleur du fond
Background-image	Image du fond
Background-repeat	Répétition de l'image de fond
Background-position	Position de l'image de fond
Background-attachment	Attache de l'image de fond par rapport à la page.
Background-origin	Postion du fond par rapport à la boîte de l'élément
Background-size	Taille de l'image de font par rapport aux dimension de l'élément

Tableau 2 : Quelques propriétés pour les fonds



# Localisation des styles

- La déclaration de règles de styles peut être mise dans 3 places soit:
  1. Dans l'élément entête «**head** »du document HTML.  
`<style> <!--Mon style--> </style>`
  2. Dans le corps , la balise ouvrante d'un élément.  
`< h1 style= "color: blue;" >`
  3. Dans un **fichier CSS** séparé.
- Il est recommandé d'utiliser la dernière méthode
- Pour lier un fichier HTML avec un fichier CSS, on écrit dans la balise **<head>**:

```
<link rel="stylesheet" href="css/fichiercss.css" />
```



# Feuille de style externe

- Une feuille de styles externe est un fichier texte portant habituellement l'extension **.css**
- Le lien entre le document HTML et le fichier CSS s'effectue dans la section **<HEAD>** d'un document HTML

## Exemple :

```
<HEAD>
  <TITLE>Histoire des feuilles de styles</TITLE>
  <link REL="StyleSheet" TYPE="text/css"  HREF="../styles.css">
</HEAD>
```

1. La balise **<LINK>** avertit le navigateur qu'il doit établir un lien
2. **rel=stylesheet** précise qu'il s'agit d'une feuille externe
3. **type="text/css"** avertit qu'il s'agit de feuilles de style en cascade
4. **href=" ... "** définit l'emplacement de la feuille de style

# Feuille de style interne

- Une feuille de styles interne est insérée en en-tête du fichier HTML à l'aide de l'élément <STYLE>,
- Ces styles seront définies uniquement pour le document courant.

## Exemple :

```
<HEAD>
    <TITLE>Nom du document</TITLE>
    <style>
        A { COLOR: red ;}
        .toto {
            COLOR: navy;
            FONT-SIZE: 12px;
            FONT-FAMILY: sans-serif
        }
    </style>
</HEAD>
```

- Application d'un style sur un élément  
`<p CLASS="toto">texte du paragraphe...`



# Feuille de style intra-ligne

- La description d'un style intra-ligne est insérée directement dans la balise d'ouverture d'un élément HTML.
- Le style est appliqué à l'aide de l'attribut HTML style inséré dans la balise d'ouverture de l'élément.

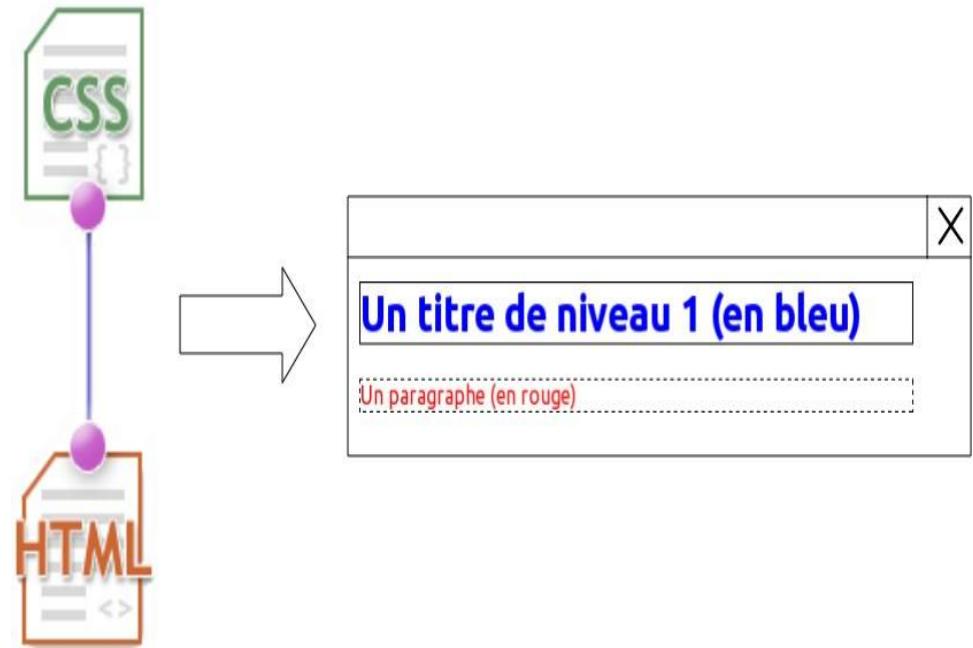
**Exemple :**

```
<p style="color: navy; font-size: 12px; font-family: sans-serif">Ceci est un paragraphe avec un style intra-ligne.</p>
```

# Présentation dans une feuille de style CSS

```
h1 {  
    color: blue;  
    border: solid 1px black; }  
p {  
    color: red;  
    border: dashed 1px black; }
```

```
<body>  
    <h1>Un titre de niveau 1 (en  
        bleu)</h1>  
    <p>Un paragraphe (en  
        rouge)</p>  
</body>
```



# Sélecteur multiple :

- Nous pouvons très facilement appliquer le même style à plusieurs éléments différents en les énumérant et en les séparant par une virgule dans le sélecteur.

```
h1,h2,p {  
    color : yellow;  
    font-family : Arial, sans-serif;  
}
```

- Dans cette exemple, la couleur du texte et la police définies s'appliquent simultanément aux balises `<h1>`, `<h2>` et `<p>`. Cela évite d'écrire trois blocs de styles séparés pour chaque balise.



# Sélecteur universel :

- Le sélecteur universel en CSS est représenté par un astérisque (\*) et sélectionne tous les éléments d'une page web. Il permet d'appliquer des styles globalement à tous les éléments sans avoir à les spécifier individuellement..

```
* {  
|   border : 1px solid red;  
|}  
|
```

→ Chaque élément HTML de la page aura une bordure de 1px

# Les classes de style :

- les classes de style permettent de définir des styles réutilisables que l'on peut appliquer à plusieurs éléments HTML.
- Une classe est définie dans la feuille de style CSS et appliquée à un élément HTML en utilisant l'attribut `class`.
- Une classe est définie avec un point (.) suivi du nom de la classe.

Définie par .



```
.highlight {  
    background-color: yellow;  
    font-weight: bold;  
}
```

→ Dans cette exemple, la classe **highlight** définit un style avec un arrière-plan jaune et du texte en gras



# Les classes de style:

## Applications d'une classe dans le HTML :

- Pour appliquer une classe à un élément HTML, utilisez l'attribut **class**.
  - Une classe est définie avec un point ( . ) suivi du nom de la classe.

```
<p class = "highlight">Ceci est un paragraphe mis en évidence</p>
<p>Ceci est un paragraphe normal.</p>
```

# La classe

# La classe

## **Applications de plusieurs classes au même élément :**

- Pour utiliser plusieurs classes dans le même élément HTML, il faut donner à son attribut **class** la liste des noms des classes en les séparant par un espace.

```
<p class="highlight text-center">Ceci est un paragraphe centré et mis en évidence.</p>
```

**Applique l'arrière-plan jaune et le texte en gras**

## Applique un alignement centré au texte



# Sélecteur d'identifiant (id):

- Le sélecteur d'identifiant (ou sélecteur ID) permet de cibler un élément HTML spécifique à l'aide de son attribut ‘**id**’
  - Un identifiant doit être unique sur la page.
  - Le sélecteur d'identifiant est défini en CSS avec ( **#** ) suivi de l'ID.
- Dans cet exemple, le style sera appliqué à l'élément HTML ayant l'ID **header**

```
#header {  
    background-color: blue;  
    color: white;  
    padding: 20px;  
}
```

## Application dans le HTML :

- L'attribut ‘**id**’ est utilisé pour assigner un ID à un élément HTML.

```
<div id="header">Ceci est l'en-tête.</div>
```

- Lorsque nous avons un élément **<div>** avec l'identifiant **header**. Cela crée une en-tête bleu avec le texte ‘ Ceci est l'en-tête.’ en blanc, centré et avec un espacement autour du texte

# Sélecteur d'attributs (1):

- Il est également possible d'appliquer un style à un élément déterminé dès qu'il possède un attribut donné, quelle que soit la valeur de cet attribut.
- Pour appliquer ce sélecteur, le nom de l'élément doit être suivi du nom de l'attribut placé entre crochets [ et ].

```
abbr[title]{  
    color : red;  
    background-color : gray;  
}
```

→ Tous les éléments `<abbr>` qui possèdent un attribut `title`, quelle que soit sa valeur, ont un contenu affiché en rouge sur fond gris



# Sélecteur d'attributs (2):

## Sélecteur d'attribut exact [attribut="valeur"] :

- Permet de sélectionner tous les éléments qui possèdent un attribut dont la valeur est exactement égale à une valeur donnée.

```
a[href="https://example.com"] {  
|   color: blue;  
}  
|
```

→ Applique une couleur bleue aux liens ayant l'attribut **href** est exactement égal à « **https:// example.com** »

## Sélecteur d'attribut (commence par) [attribut^="valeur"] :

- Permet de sélectionner tous les éléments dont la valeur de l'attribut commence par une chaîne de caractères spécifique.

```
img[src^="https"] {  
|   border: 2px solid green;  
}  
|
```

→ Applique une bordure verte aux images dont l'URL commence par « **https** ».

# Sélecteur d'attributs (3):

## Sélecteur d'attribut (finit par) [attribut\$="valeur"] :

- Permet de sélectionner tous les éléments dont la valeur de l'attribut se termine par une chaîne de caractère spécifique.

```
a[href$=".pdf"] {  
    font-weight: bold;  
}
```

→ Applique un texte en gras à tous les liens ayant un attribut **href** qui se termine par « .pdf ».

## Sélecteur d'attribut (contient) [attribut\*=“valeur”] :

- Permet de sélectionner tous les éléments dont la valeur de l'attribut contient une chaîne de caractères spécifique

```
div[class*="container"] {  
    padding: 10px;  
}
```

→ Applique un espacement interne (padding) à tous les div dont la classe contient "container".

# Sélecteur d'attributs (4):

## Sélecteur Universel avec attributs \*[attribut] :

- Nous pouvons créer un style qui s'applique à tous les éléments possédant un attribut spécifique en utilisant le sélecteur universel \* placé devant les crochets contenant le nom de l'attribut.

```
*[title] {  
    background-color: yellow;  
}
```

→ Applique un fond jaune à tous les éléments qui possèdent un attribut donné, quelle que soit la valeur de cet attribut title.

## Sélection de plusieurs attributs élément[attribut1][attribut2] :

- Il est également possible de sélectionner plusieurs attributs pour un élément en spécifiant plusieurs attributs entre crochets après le nom de l'élément.

```
h2[title][id] {  
    background-color: yellow;  
}
```

→ Applique un fond jaune à tous les éléments h2 qui ont à la fois les attributs title et id.



# Sélecteurs de valeur d'attribut:

- Le sélecteur de valeur d'attributs permet de cibler des éléments en fonction de la valeur d'un ou plusieurs de leurs attributs.
- Nous pouvons créer un style qui s'applique à tous les éléments possédant un attribut spécifique avec une valeur exacte en utilisant le sélecteur **[attribut="valeur"]**. Ce type de sélecteur est utile pour appliquer des styles uniquement aux éléments ayant des attributs spécifiques avec des valeurs définies.

```
input[type="text"] {  
    border: 1px solid blue;  
}
```

→ Le style s'appliquera à tous les éléments **<input>** ayant l'attribut **type** avec la valeur "**text**". Ainsi tous les champs de texte auront une bordure bleue

- Il est aussi possible de particulariser davantage l'application du style en sélectionnant plusieurs attributs et leurs valeurs en utilisant la syntaxe :

**élément[attribut1="valeur1"][attribut2="valeur2"]{  
 définition du style;  
}**

# Sélecteurs de valeur d'attribut(1):

## Autres possibilités :

- Le sélecteur **[attribut~="valeur"]**, est un sélecteur d'attribut "espace séparé" (ou sélecteur d'attribut "mot complet"). Il permet de sélectionner des éléments dont la valeur d'un attribut contient une valeur spécifique parmi des mots séparés par des espaces.

```
[fruit ~="pomme"] {  
    color: red;  
    font-weight: bold;  
}
```

→ Valide (style appliqué) :

- Si un élément a **fruit = "orange pomme banane "**
  - Le style sera appliqué, car pomme est présent parmi les valeurs séparés par des espaces.
- Si un élément a **fruit= "pomme"**
  - Le style sera appliqué, car pomme est exactement la seule valeur.

→ Invalide (style non appliqué) :

- Si un élément a **fruit= "pomme\_fruit"**
  - Le style ne sera pas appliqué car pomme\_fruit n'est pas exactement pomme
- Si un élément a **fruit= "bananepomme "** :
  - Le style ne sera pas appliqué car pomme n'est pas une valeur séparé



# Sélecteurs de valeur d'attribut(2):

## Autres possibilités :

- **élément[attribut\*="valeur"]{ Définition des styles}** : ce sélecteur s'applique lorsque l'attribut spécifié contient la sous-chaîne « val » dans sa valeur.

## Exemple :

**p[data-type\*="active"] { color: blue; }**: Ici, seuls les éléments <p> avec data-type contenant "active" auront leur texte en bleu, tandis que d'autres éléments avec data-type contenant "active" (comme <div>, <span>, etc.) ne seront pas affectés.

- **élément[attribut^="valeur"]{ Définition des styles}** : ce sélecteur s'applique à l'élément dont la valeur de l'attribut commence par la chaîne « valeur »

## Exemple :

**div[data-role^="admin"] { color: green; }** :seuls les éléments <div> ayant un attribut data-role qui commence par "admin" auront leur texte en vert



# Sélecteurs de valeur d'attribut(3):

## Autres possibilités :

- **élément[attribut |= "valeur"]{ Définition des styles}** : ce sélecteur s'applique à l'élément dont la valeur de l'attribut commence par la chaîne « valeur », suivie d'un trait d'union. En d'autres termes, il cible les éléments dont l'attribut est égal à « valeur » ou « valeur-suffixe »

### Exemple :

`[lang |= "fr"]` : s'applique si la valeur de l'attribut commence par « **fr** » suivi d'un trait d'union.

- **élément[attribut \$= "valeur"]{ Définition des styles}** : ce sélecteur s'applique à l'élément lorsque la valeur de l'attribut se termine par la chaîne spécifiée « **valeur** »



# Sélecteurs contextuels parents-enfants:

- Les **sélecteurs contextuels** permettent d'appliquer des styles à un élément seulement lorsque cet élément se trouve à l'intérieur d'un autre élément. Cela permet de cibler des éléments de manière précise en fonction de leur hiérarchie dans le document HTML.

**Syntaxe :** élément\_parent élément\_enfant{  
définition des styles;}

**Exemple:**

```
p {  
|   color: blue;  
}  
→
```

Le style général (couleur bleue) s'applique à tous les éléments <p> de la page

```
div p {  
|   color: red;  
}  
→
```

Le style contextuel (couleur rouge) s'applique uniquement aux éléments <p> qui sont imbriqués à l'intérieur d'un <div>

# Sélecteurs parents-enfants directs:

- Les **sélecteurs parent-enfant direct (>)** cible uniquement les enfants immédiats d'un parent donné, et non les descendants plus éloignés. Par exemple, si un élément A est le parent direct d'un élément B, alors tu peux cibler B uniquement si tu utilises le sélecteur direct.

## Syntaxe :

```
parent > enfant {
    /*Définition des styles */}
```

```
.parent > .enfant {
    color: blue;
    font-weight: bold;
}
```

```
<div class="parent">
    <p class="enfant">Je suis un enfant direct.</p>
    <div class="autre">
        <p class="enfant">Je ne suis pas un enfant direct.</p>
    </div>
</div>
```

- Le sélecteur `.parent > .enfant` s'applique uniquement au premier paragraphe qui est un enfant direct du `div` ayant la classe `parent`.
- Le second paragraphe, qui est à l'intérieur d'un `div` avec la classe `autre`, ne sera pas affecté par ce style, car il n'est pas un enfant direct de `.parent`.



# Sélecteurs d'éléments adjacents:

- Les **sélecteurs d'éléments adjacents en CSS** permettent de cibler un élément qui suit immédiatement un autre élément au même niveau de la hiérarchie. Ce sélecteur est utile pour appliquer des styles à des éléments qui se trouvent directement après un autre, ce qui permet de gérer des mises en page plus précises.

**Syntaxe :**

**élément1 + élément2{  
/\*Définition des styles \*/;}**

```
h1 + p {  
    color: green;  
    font-style: italic;  
}
```

```
<h1>Titre principal</h1>  
<p>Je suis un paragraphe qui suit immédiatement le titre.</p>  
<p>Je suis un autre paragraphe qui ne sera pas affecté par le sélecteur adjacent.</p>
```

- Le sélecteur **h1 + p** s'applique uniquement au premier paragraphe qui suit immédiatement le titre **<h1>**.
- Le seconde paragraphe, qui vient après, ne sera pas affecté par ce style, car il ne suit pas immédiatement le **<h1>**.



# Pseudo-classes:

- Les **pseudo-classes** en CSS sont des sélecteurs qui permettent d'appliquer des styles spécifiques à des éléments en fonction de leur état ou de leur position dans le document. Contrairement aux sélecteurs classiques, qui ciblent des éléments en fonction de leur type, classe ou identifiant, les pseudo-classes permettent de cibler des éléments dans des conditions particulières, offrant ainsi une flexibilité et une interactivité accrues lors de la conception de pages web.

→**Les Pseudo-Classes pour les liens.**



# Pseudo-classes applicables aux liens:

- Il existe quatre pseudo-classes spécifiques aux liens en CSS, qui permettent de gérer différents états d'interaction.

## ■ : link

- Cette pseudo-classe permet d'attribuer un style à un lien qui pointe vers un document non encore vu. C'est l'état normal de tous les liens à l'ouverture de la page.

Exemple :

```
a:link {  
    color: blue; /* Le lien non visité sera affiché en bleu */  
}
```

## ■ : visited

- Cette pseudo-classe permet d'attribuer un style à un lien qui pointe vers un document déjà visité, après un retour sur la page d'origine. Elle aide à différencier les liens que l'utilisateur a déjà explorés.

Exemple :

```
a:visited {  
    color: red; /* Le lien visité sera affiché en rouge */  
}
```

# Pseudo-classes applicables aux liens:

## ■ : hover

- Cette pseudo-classe permet d'appliquer un style à un lien lorsque l'utilisateur passe la souris dessus. Elle offre un retour visuel indiquant que le lien est interactif et prêt à être cliqué.

Exemple :

```
a:hover {  
    |   text-decoration: underline; /* Le lien sera souligné au survol */  
}
```

## ■ : active

- Cette pseudo-classe permet d'attribuer un style à un lien au moment où l'utilisateur clique dessus. Elle indique que le lien est en cours d'activation, offrant une réponse immédiate à l'action de l'utilisateur.

Exemple :

```
a:active {  
    |   color: green; /* Le lien sera affiché en vert lors du clic */  
}
```

# Pseudo-éléments:

- Les **pseudo-éléments** permettent de cibler des parties spécifiques d'un élément dans le document HTML. Ils permettent de modifier le style de ces parties sans avoir besoin d'ajouter des balises supplémentaires dans le HTML.

- **::before**

→ Permet d'insérer du contenu avant le contenu d'un élément.

Exemple :

```
p::before {  
    content: "Note: "; /* Texte ajouté avant chaque paragraphe */  
    font-weight: bold; /* Met le texte en gras */  
}
```

```
<body>  
    <h1>Exemple de Pseudo-Élément ::before</h1>  
    <p>Ceci est un exemple de texte qui utilise le pseudo-élément ::before.</p>  
</body>  
</html>
```

## Exemple de Pseudo-Élément ::before

**Note:** Ceci est un exemple de texte qui utilise le pseudo-élément ::before.

# Pseudo-éléments :

## ■ ::after

→ Permet d'insérer du contenu après le contenu d'un élément.

Exemple :

```
p::after {  
    content: " (fin)"; /* Texte ajouté après chaque paragraphe */  
    font-style: italic; /* Met le texte en italique */  
}
```

```
<body>  
    <h1>Exemple de Pseudo-Élément ::after</h1>  
    <p>Ceci est un exemple de texte qui utilise le pseudo-élément ::after.</p>  
</body>  
</html>
```

## Exemple de Pseudo-Élément ::after

Ceci est un exemple de texte qui utilise le pseudo-élément ::after. *(fin)*

# Pseudo-éléments :

## ■ **::first-line**

- Cible la première ligne d'un bloc de texte, permettant de styliser cette ligne spécifiquement.

Exemple :

```
p::first-line {  
    font-weight: bold; /* Met la première ligne en gras */  
    color: blue; /* Change la couleur en bleu */  
}
```

```
<body>  
    <h1>Exemple de Pseudo-Élément ::first-line</h1>  
    <p>Ceci est un exemple de texte qui utilise le pseudo-élément ::first-line  
        pour styliser la première ligne du paragraphe.</p>  
</body>  
</html>
```

## Exemple de Pseudo-Élément ::first-line

Ceci est un exemple de texte qui utilise le pseudo-élément ::first-line pour styliser la première ligne du paragraphe.

# Pseudo-éléments :

## ■ **::first-letter**

→ Cible la première lettre d'un bloc de texte, permettant d'appliquer un style distinct à cette lettre.

Exemple :

```
p::first-letter {  
    font-size: 2em; /* Augmente la taille de la première lettre */  
    color: red; /* Change la couleur en rouge */  
}
```

```
<body>  
    <h1>Exemple de Pseudo-Élément ::first-letter</h1>  
    <p>Ceci est un exemple de texte qui utilise le pseudo-élément  
        ::first-letter pour styliser la première lettre du paragraphe.</p>  
</body>  
</html>
```

## Exemple de Pseudo-Élément ::first-letter

Ceci est un exemple de texte qui utilise le pseudo-élément ::first-letter pour styliser la première lettre du paragraphe.



# Unités de taille :

- Les **unités de taille en CSS** permettent de définir les dimensions des éléments, telles que la largeur, la hauteur, ou la taille de police. Elles se divisent en deux grandes catégories :

→ **Les unités absolues .**

→ **Le unités relatives.**

# Unités de taille : Unités absolues

- Les unités absolues définissent des tailles fixes, indépendantes de l'environnement(écran,conteneur,etc.).
- **px (pixel)** : Unité courante, correspondant à un pixel sur l'écran.
- **cm (centimètre)** : unité basée sur des dimensions physique.
- **mm (millimètres)** : unité pour des distances très petites
- **in (pouce)** : 1 inch (pouce) équivaut à 2,54.
- **pt (point)** : un point est égal à 1/72 de pouces.
- **pc (pica)** : un pica vaut 12 points



# Unités de taille :Unités relatives

- Les unités relatives définissent des tailles en fonction d'autres éléments ou propriétés, ce qui les rend plus flexibles et adaptées aux mises en page responsives.
- **em** : Correspond à la taille de la police de l'élément parent. Par exemple, **1em** est égal à la taille actuelle du texte. Si la taille du texte parent est de **16px**, alors **1em = 16px**.
- **rem** : Identique à em, mais basée sur la taille de la police racine (généralement la taille du texte définie dans le <html>).
- **%** : Pourcentage, basé sur la taille de l'élément parent. Si vous définissez la largeur d'un élément à 50%, cela prendra la moitié de la largeur du parent.
- **vh** : Viewport Height, basé sur 1% de la hauteur de la fenêtre d'affichage.
- **vw** : Viewport Width, basé sur 1% de la largeur de la fenêtre d'affichage.
- **ex** : Hauteur de la lettre "x" en minuscule dans la police utilisée.
- **ch** : largeur du caractère "o" dans la police utilisée.

# Exemple: index.html

```
<body>
    <h1>Exemple des différentes unités de taille</h1>

    <div class="absolute-units">
        <h2>Unités Absolues</h2>
        <p class="pt">Taille en points (pt)</p>
        <p class="cm">Taille en centimètres (cm)</p>
        <p class="mm">Taille en millimètres (mm)</p>
        <p class="in">Taille en pouces (in)</p>
    </div>

    <div class="relative-units">
        <h2>Unités Relatives</h2>
        <p class="px">Taille en pixels (px)</p>
        <p class="em">Taille en em</p>
        <p class="ex">Taille en ex</p>
        <p class="rem">Taille en rem</p>
        <p class="percent">Taille en pourcentage (%)</p>
        <p class="ch">Taille en ch (basée sur la largeur du "0")</p>
    </div>
</body>
```



# Exemple:styles.css

```
/* Styles pour les unités absolues */
.absolute-units p {
    margin: 10px 0;}
/* Unités absolues */
.pt {
    font-size: 12pt;
    background-color: lightcoral;}
.cm {
    font-size: 2cm;
    background-color: lightblue;}
.mm {
    font-size: 10mm;
    background-color: lightgreen;}
.in {
    font-size: 1in;
    background-color: lightyellow;}
```

```
/* Styles pour les unités relatives */
.relative-units p {
    margin: 10px 0; }

/* Unités relatives */
.px {
    font-size: 16px;
    background-color: lightpink;}
.em {
    font-size: 2em;
    background-color: lightgray;}
.ex {
    font-size: 2ex;
    background-color: lightgoldenrodyellow;}
.rem {
    font-size: 1.5rem;
    background-color: lightcyan;}
.percent {
    font-size: 150%;
    background-color: lightsteelblue;}
.ch {
    font-size: 20ch;
    background-color: lightseagreen;}
```

# Exemple

## Exemple des différentes unités de taille

### Unités Absolues

Taille en points (pt)

Taille en centimètres (cm)

Taille en millimètres (mm)

Taille en pouces (in)

### Unités Relatives

Taille en pixels (px)

Taille en em

Taille en ex

Taille en rem

Taille en pourcentage (%)

Taille en ch (basée



# Cascade, Conflits et Héritage en CSS :

- En CSS, la cascade, les conflits de styles et l'héritage sont des concepts essentiels pour comprendre comment les styles sont appliqués aux éléments HTML et comment prioriser certaines règles.
- L'héritage en CSS est fondé sur le modèle Parent-Enfant(s) : **chaque élément Enfant reçoit en héritage tous les styles de son élément Parent.**
- Cet héritage est très pratique et permet d'éviter beaucoup de répétitions inutiles.

## Exemple:

- La balise <html> est parent de <body>
- La balise <table> est parent de <tr> qui lui-même est parent de <td>.
- En attribuant une propriété à un parent (par exemple *font-size: 1.5em*), elle sera transmise à tous ses enfants, mais aussi aux enfants de ces enfants, etc...



# La cascade et les conflits de styles :

## La cascade :

- La **cascade** en CSS est un principe fondamental qui détermine comment les styles sont appliqués aux éléments HTML lorsqu'il y a plusieurs style en concurrence. Les styles finaux sont choisis en fonction de règles spécifiques, mais ces derniers peuvent parfois entrer en **conflit**.

## Les conflits de styles :

- Un **conflit** de styles survient lorsque plusieurs règles visent le même élément et définissent des valeurs différentes pour une même propriété. La **cascade** utilise alors des critères pour résoudre ces conflits :
  - Position et ordre d'apparence
  - Spécificité des sélecteurs
  - Importance

# La cascade et les conflits de styles :

## ❖ Position et ordre d'apparence :

- Les règles sont appliquée dans l'ordre où elles apparaissent. Si deux règles s'appliquent à un même élément, la dernière règle dans la feuille de style l'emporte.

**Exemple :**

```
p { color: blue; }
p { color: green; } /* La couleur finale sera verte */
```

## ❖ Importance ( ! Important ):

- Lorsqu'une règle utilise !important , elle prend la priorité sur les autres styles, quelle que soit leur spécificité :

**Exemple :**

```
<style>
  p { color: blue !important; }
  #unique { color: red; }
</style>
<p id="unique">Ce texte sera bleu.</p>
```

# La cascade et les conflits de styles :

## ❖ Spécificité des sélecteurs :

- La spécificité mesure l'importance d'une règle CSS. Les règles plus spécifiques ont la priorité sur les règles moins spécifiques.
  - **Styles inline** (dans l'attribut style de l'élément HTML) : très spécifique.
  - **Sélecteur ID (#id)** : score élevé. un "score" de spécificité élevé, supérieur aux sélecteurs de classes ou d'éléments, mais inférieur aux styles inline.
  - **Sélecteurs de classe, pseudo-classes et attributs (.classe, :hover, [type="text"] )**: spécificité moyenne.
  - **Sélecteurs de type (p, div, etc.)** : spécificité basse.

**Exemple :**

```
<p id="paragraphe" class="texte">Bonjour !</p>
```

```
p { color: blue; }           /* Spécificité faible */
.texte { color: green; }     /* Spécificité moyenne */
#paragraphe { color: red; } /* Spécificité élevée */
```



# Héritage des propriétés de style :

- L'héritage en CSS est un mécanisme où certaines propriétés de style appliquées à un élément parent sont automatiquement transmises aux éléments enfants imbriqués. Ce principe simplifie la gestion des styles en évitant les répétitions pour chaque éléments.
- Propriétés héritables : celles qui sont appliquées aux enfants de manière automatique, comme color, font-family, et line-height.
- Propriétés non-héritables : celles qui ne se transmettent pas automatiquement, par exemple width, height, margin, et padding.

## Exemple :

```
<style>
  div { color: blue; }
  p { font-size: 16px; }
</style>

<div>
  Texte principal
  <p>Paragraphe imbriqué</p>
</div>
```



# Héritage des propriétés de style :**inherit**

- La valeur **inherit** permet à un élément enfant d'adopter la valeur d'une propriété de son parent, même si cette propriété n'est pas héritée par défaut.
- On utilise **inherit** pour forcer l'héritage même pour des propriétés non-héritables, comme background ou border.

**Exemple :**

```
.parent {  
    background-color: lightblue; /* Propriété non héritée */  
    padding: 20px; /* Propriété non héritée */  
}  
  
.child {  
    background-color: inherit; /* Hérite de la couleur de fond du parent */  
    padding: inherit; /* Hérite du padding (ceci ne fonctionnera pas comme prévu) */  
}
```

# Codage des couleurs:

Pour créer une couleur sur écran, le mode colorimétrique utilisé est le **RGB** (Red, Green, Blue) ou **RVB** en français (Rouge, Vert, Bleu).

## ■ **Code RGB :**

Le code RGB consiste à spécifier l'intensité de chaque couleur dans l'ordre suivant : Rouge, Vert, Bleu. Il existe plusieurs manières de le faire.

- Notation hexadécimale : chaque composante est exprimé en deux chiffre hexadécimaux, compris entre 00 et FF.
  - **Exemple :** #0000FF représente le bleu (00 pour le rouge, 00 pour le vert, et FF pour le bleu)

## ■ **Notation hexadécimale raccourcie :**

Il est également possible d'utiliser une notation hexadécimale raccourcie. Dans ce cas, chaque chiffre hexadécimal est doublé:

- **Exemple :** #oOF est équivalent à #0000FF (où o correspond au rouge, O au vert, et F au bleu).

# Codage des couleurs:

## ■ Notation décimale :

L'intensité de chaque couleur est exprimée à l'aide de trois valeurs comprises entre 0 et 255, a l'aide de la fonction **rgb()** :

### - Exemple :

```
rgb(255, 0, 0) /* Cela représente le rouge */
```

## ■ Notation en pourcentage :

Chaque valeur de couleur dans la fonction **rgb()** peut également être exprimée en pourcentage, allant de 0% à 100%

### - Exemple :

```
rgb(0, 100%, 0) /* Cela représente le vert */
```

## ■ Autres notations de couleurs :

Il existe d'autres méthodes pour définir les couleurs en CSS, telles que:

- **rgba()** : permet d'ajouter un canal alpha pour la transparence.
- **hsl()** : utilise la teinte, la saturation et la luminosité pour définir les couleurs.
- **hsla()** : comme hsl(), mais avec un canal alpha pour la transparence.

## ■ Noms des couleurs :

CSS reconnaît également des noms de couleurs standard, tels que bleu, white, et red. Ces mots-clés permettent d'utiliser des couleurs sans avoir à spécifier leurs valeurs.



Nom en français	Nom HTML	Décimal	Pourcentage	Hexadécimal
Noir	black	rgb(0, 0, 0)	rgb(0%, 0%, 0%)	#000000
Rouge	red	rgb(255, 0, 0)	rgb(100%, 0%, 0%)	#FF0000
Vert	green	rgb(0, 128, 0)	rgb(0%, 50%, 0%)	#008000
Jaune	yellow	rgb(255, 255, 0)	rgb(100%, 100%, 0%)	#FFFF00
Bleu	blue	rgb(0, 0, 255)	rgb(0%, 0%, 100%)	#0000FF
Cyan	cyan	rgb(0, 255, 255)	rgb(0%, 100%, 100%)	#00FFFF
Magenta	magenta	rgb(255, 0, 255)	rgb(100%, 0%, 100%)	#FF00FF
Blanc	white	rgb(255, 255, 255)	rgb(100%, 100%, 100%)	#FFFFFF
Gris	gray	rgb(128, 128, 128)	rgb(50%, 50%, 50%)	#808080
Vert clair	lightgreen	rgb(144, 238, 144)	rgb(56.5%, 93.3%, 56.5%)	#90EE90
Bleu clair	lightblue	rgb(173, 216, 230)	rgb(67.8%, 84.7%, 90.2%)	#ADD8E6
Orange	orange	rgb(255, 165, 0)	rgb(100%, 65%, 0%)	#FFA500
Violet	purple	rgb(128, 0, 128)	rgb(50%, 0%, 50%)	#800080
Marron	brown	rgb(165, 42, 42)	rgb(64.7%, 16.5%, 16.5%)	#A52A2A
Olive	olive	rgb(128, 128, 0)	rgb(50%, 50%, 0%)	#808000



# Quelques propriétés de textes :

Les propriétés CSS permettent de contrôler l'apparence du texte sur une page web. Voici quelques-unes des propriétés les plus courantes :

- **color** : définit la couleur de texte.
- **font-family** : spécifie la police de caractères à utiliser.
- **font-size** : définit la taille de la police (`small|medium|%|x pt`).
- **font-weight** : contrôle l'épaisseur du texte (`normal, bold, etc.`).
- **text-align** : aligne le texte (`left, right, center, right, justify`).
- **line-height** : définit l'espacement entre les lignes de texte.
- **Font-style** : définit le style du texte (`italic, oblique`).
- **text-indent** : définit le retrait de la première ligne d'un paragraphe.
- **letter-spacing** : définit l'espacement entre les lettres.
- **word-spacing** : définit l'espacement entre les mots.



# Quelques propriétés pour listes:

Les listes peuvent stylisées de différentes manières pour améliorer leur présentation.

- **list-style** : définir le style de la puce, la position et l'image d'une liste.
- **list-style-type** : Spécifie uniquement le type de puce pour les listes.
  - **Disc, circle, square** (pour les listes non ordonnées)
  - **Decimal, lower-alpha, upper-alpha** (pour les listes ordonnées)
- **list-style-position** : Définit la position des puces par rapport au texte de la liste.
  - **inside** : Les puces sont placées à l'intérieur du bloc de texte.
  - **outside** : Les puces sont placées à l'extérieur du bloc de texte.
- **list-style-image** : Permet d'utiliser une image personnalisée comme puce pour les listes.
  - **url ('chemin/vers/image.png')** : spécifie l'image à utiliser
  - **None** : supprime l'image.



# Quelques propriétés pour listes:

- **margin** : définit l'espace autour des listes (spécifié en unités comme px, %, em).
- **padding** : définit l'espace intérieur entre le contenu de la liste et ses bordures (px, %, em).
- **text-indent** : Définit le retrait de la première ligne des éléments de liste (px, %, em).
- **display** : Définit le type d'affichage des listes. Cela peut changer la manière dont les éléments de la liste sont rendus dans le document.
  - **liste-item** : Comportement par défaut pour les éléments de liste.
  - **bloc** : affiche chaque élément de la liste comme un bloc.
  - **inline** : affiche les éléments de la liste en ligne, comme des éléments de texte.



# Exemple:

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple de Liste et Texte</title>
    <link rel="stylesheet" href="exemplfruit.css">
</head>
<body>
    <h1>Mes Fruits Préférés</h1>
    <p>Voici une liste de mes fruits préférés, avec quelques détails :</p>

    <ul class="fruits-list">
        <li><strong>Pommes</strong>: Riches en fibres et en vitamines.</li>
        <li><strong>Bananes</strong>: Excellente source de potassium.</li>
        <li><strong>Fraises</strong>: Pleines d'antioxydants et délicieuses !</li>
    </ul>

    <p>J'aime particulièrement ces fruits pour leur goût et leurs bienfaits pour la santé.</p>
</body>
</html>
```

# Exemple:

```
body {  
    font-family: Arial, sans-serif;  
    line-height: 1.6;  
    margin: 20px;  
    color: #333; /* Couleur du texte principale */  
}  
h1 {  
    text-align: center;  
    color: #FF4500; /* Couleur du titre en orange vif */  
}  
p {  
    font-size: 16px;  
    margin-bottom: 20px;  
    color: #32CD32; /* Couleur du texte du paragraphe en vert lime */  
}  
.fruits-list {  
    list-style-type: square; /* Type de puce */  
    list-style-position: outside; /* Position des puces */  
    margin: 20px 0; /* Marges autour de la liste */  
    padding: 10px; /* Espacement intérieur */  
    background-color: #FFD700; /* Couleur de fond en jaune vif */  
    border: 2px solid #FF0000; /* Bordure en rouge vif */  
}  
.fruits-list li {  
    text-indent: 10px; /* Retrait de la première ligne */  
    margin-bottom: 10px; /* Espacement entre les éléments de la liste */  
    font-weight: bold; /* Texte en gras */  
    color: #1E90FF; /* Couleur du texte des éléments de la liste en bleu dodger */  
}
```

Exemple:

## Mes Fruits Préférés

Voici une liste de mes fruits préférés, avec quelques détails :

- **Pommes: Riches en fibres et en vitamines.**
- **Bananes: Excellente source de potassium.**
- **Fraises: Pleines d'antioxydants et délicieuses !**

J'aime particulièrement ces fruits pour leur goût et leurs bienfaits pour la santé.



# Les styles pour les boites:

## Le concept du modèle de boîte:

Une boîte est un conteneur rectangulaire qui peut contenir tout type de contenu : du texte, des images, des tableaux, des formulaires, mais aussi d'autres boîtes. Le modèle de boîte détermine la manière dont les éléments sont affichés et espacés sur une page. Chaque élément est composé de plusieurs couches :

- **Un contenu (content):** C'est la zone où le texte et les images apparaissent. Sa taille peut être contrôlée par la largeur (width) et la hauteur (height) de l'élément.
- **Un remplissage interne (Padding):** C'est l'espace entre le contenu de l'élément et sa bordure. Le remplissage ajoute de l'espace à l'intérieur de la boîte. Il peut être défini individuellement pour chaque côté (haut, droit, bas, gauche) ou de manière globale.

## Exemple:

**padding: 10px .** // Applique 10 pixels de remplissage sur tous les côtés.

**padding: 10px 20px;** // Applique 10 pixels en haut et en bas et 20 pixels à gauche et à droite.

# Les styles pour les boites:

- **Une bordure (border):** C'est une ligne qui entoure le remplissage (et le contenu) de l'élément. Vous pouvez définir l'épaisseur, le style et la couleur de la bordure.

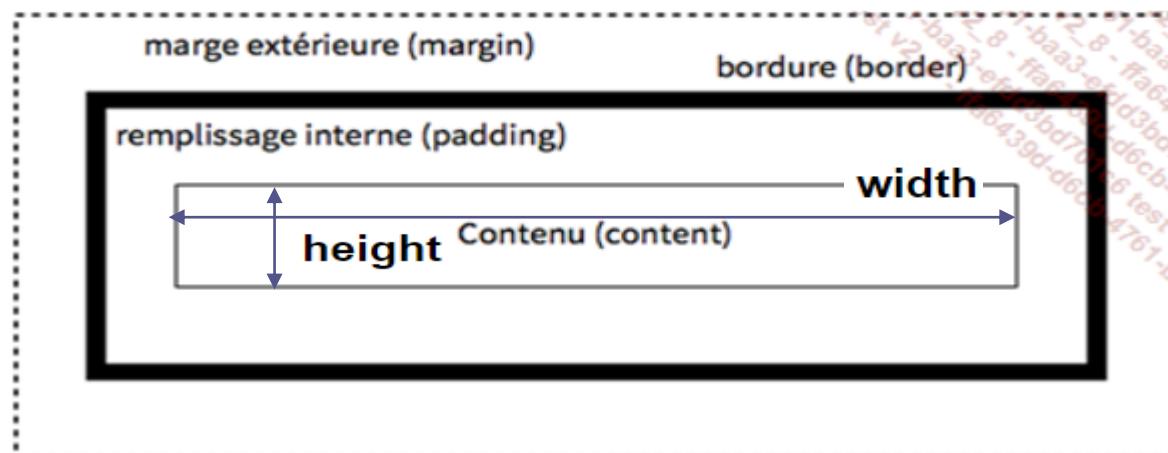
- **Exemple:**

**border: 2px solid black;** //Applique une bordure noire de 2 pixels d'épaisseur.

- **Une marge (margin):** C'est l'espace à l'extérieur de la bordure de l'élément, qui crée une séparation avec d'autres éléments. Les marges peuvent également être définies pour chaque côté de l'élément.

- **Exemple:**

Applique 20 pixels de marge sur tous les côtés.



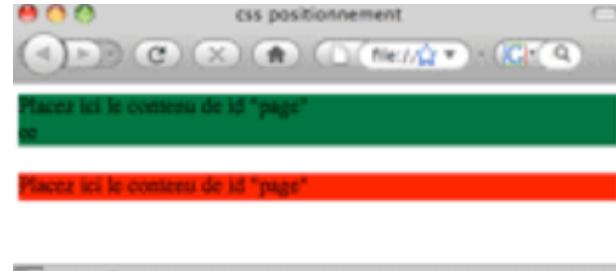
# Le positionnement des boites: Statique, Relatif, Absolu

- **Statique (par défaut):** il y a une boîte verte insérée dans la page, avec les marges de la page par défaut. Puis une deuxième boîte rouge, qui se place par défaut en dessous. Elles contiennent une ligne de texte. Par défaut les boîtes font 100% de leur conteneur et prennent la hauteur de leur contenu.



- **Relatif (Dans le flux) :** Il est possible de décaler horizontalement et/ou verticalement les éléments. Ceci peut générer des chevauchements ! Ici la boîte rouge a été "poussée" de 20pixels plus bas que l'élément précédent :

```
#rouge {
    background: red;
    position: relative;
    top: 20px; }
```





# Le positionnement des boites: Le Flottement

Le **flottement** (ou **float**) : Une technique utilisée pour positionner des éléments à gauche (float: left) ou à droite (float: right) dans leur conteneur.

## ➤ Utilisation du Flottement

Le flottement est souvent utilisé pour créer des mises en page où les éléments se disposent les uns à côté des autres. On peut flotter des images ou des boîtes de texte pour les faire s'enrouler autour.

Les éléments flottants doivent **OBLIGATOIUREMENT** avoir une **LARGEUR DEFINIE**.

## ➤ Propriété Float:

La propriété float peut prendre trois valeurs principales :

- **left** : Fait flotter l'élément à gauche de son conteneur.
- **right** : Fait flotter l'élément à droite de son conteneur.
- **none** : C'est la valeur par défaut. L'élément ne flotte pas.

# Le positionnement des boîtes: Le Flottement



En botanique, le lotus, dont le nom dérive du grec *lotos* par le latin *lotus*, désigne diverses plantes, arbres, arbustes ou

herbes, terrestres ou aquatiques, qui portaient déjà ce nom dans l'Antiquité.



L'image a le paramètre : **float: left** L'image se place le plus à gauche possible et le texte l'habille donc à droite.

```
/* Styles pour l'image */
img {
    float: left; /* Positionne l'image à gauche */
    margin: 10px; /* Ajoute de l'espace autour de l'image */
    width: 150px; /* Largeur de l'image */
    height: auto; /* Ajuste la hauteur automatiquement */
}
```

# Le positionnement des boîtes: Le Flottement

En botanique, le lotus, dont le nom dérive du grec *lotos* par le latin *lotus*, désigne diverses plantes, arbres, arbustes ou herbes, terrestres ou aquatiques, qui portaient déjà ce nom dans l'Antiquité.



L'image a le paramètre : **float: right** L'image se place le plus à droite possible et le texte l'habille donc à gauche.

```
/* styles pour l'image */
img {
    float: right; /* Positionne l'image à droite */
    margin: 10px; /* Ajoute de l'espace autour de l'image */
    width: 150px; /* Largeur de l'image */
    height: auto; /* Ajuste la hauteur automatiquement */
}
```



# Les types de boîte: Block, inline, inline-block

En CSS, les types de boîtes (ou "box types") permettent de contrôler la disposition et le comportement des éléments HTML dans une page web. Voici les principaux types de boîtes : block, inline, et inline-block .

Certains éléments se mettent les uns en dessous des autres, d'autres les uns à côté des autres. Les éléments qui se placent les uns en dessous des autres sont des éléments de type : **block**.

## Boîte de type block:

- ✓ Les éléments de type block occupent **toute la largeur** de leur conteneur, même si leur contenu est plus petit.
- ✓ Ils commencent toujours sur une nouvelle ligne et se placent les uns en dessous des autres.

## Exemples :

Les éléments HTML <div>, <p>, <h1> à <h6>, <ul>, <ol>, et <li> sont des éléments de type "block" par défaut.

Il sera toutefois possible de faire afficher certains de ces éléments grâce à la propriété : **display**.

# Éléments de type : block

```
<div class="block-element">Élément de type block 1</div>
<div class="block-element">Élément de type block 2</div>
```



```
<style>
  .block-element {
    display: block;
    width: 100%;
    background-color: lightblue;
    margin: 5px 0;
    padding: 10px;
  }
</style>
```



Élément de type block 1

Élément de type block 2



# Éléments de types : Inline

## Boîte de type Inline:

- ✓ Ceux qui s'affichent sur la même ligne que le texte ou d'autres éléments **inline** qui les entourent, sans commencer sur une nouvelle ligne.
- ✓ Les éléments de type inline occupent uniquement l'espace nécessaire à leur contenu.

## Caractéristiques :

- **Aucun saut de ligne** : Contrairement aux éléments block
- **Utilisation limitée du modèle de boîte** : Les éléments inline respectent uniquement les marges (margin) et le padding (padding) sur les côtés horizontal (gauche et droite).
- La largeur (width) et la hauteur (height) ne sont généralement pas applicables.
- **Texte et éléments graphiques** : Ce type d'élément est généralement utilisé pour les balises qui ajoutent du style, des liens, ou des images intégrées dans le texte.



# Éléments de types : Inline

## Exemples :

Les éléments HTML <span>, <a>, <strong>, <em>, et <img> sont tous des éléments de type "inline" par défaut.

CSS : Par défaut, ces éléments ont display: inline;.

```
<span class="inline-element">Élément inline 1</span>
<span class="inline-element">Élément inline 2</span>
<span class="inline-element">Élément inline 3</span>
```



```
.inline-element {
    display: inline;
    background-color: lightcoral;
    padding: 5px;
    margin: 2px;
}
```



Élément inline 1   Élément inline 2   Élément inline 3



# Éléments de types : Inline-block

## Boîte de type inline-block:

- ✓ Les éléments inline-block combinent les caractéristiques de inline et de block.
- ✓ Ils se placent les uns à côté des autres, mais peuvent également définir des marges, des paddings et une hauteur et largeur comme les éléments block.
- ✓ Utile pour créer des éléments de menu ou des boutons alignés horizontalement.
- ✓ CSS : Ils sont spécifiés avec display: inline-block;.



# Éléments de types : Inline-block

## Exemples :

```
<span class="boite">Boîte 1</span>
<span class="boite">Boîte 2</span>
<span class="boite">Boîte 3</span>
```



```
<style>
  .boite {
    display: inline-block;
    width: 100px; /* Largeur de la boîte */
    height: 100px; /* Hauteur de la boîte */
    background-color: #4CAF50; /* Couleur de fond verte */
    color: white; /* Couleur du texte */
    text-align: center; /* Centrer le texte horizontalement */
    line-height: 100px; /* Centrer le texte verticalement */
    margin: 10px; /* Espacement autour de chaque boîte */
  }
</style>
```

## Exemple avec Inline-Block

Les boîtes ci-dessous sont côte à côte grâce à `display: inline-block;`.





# Éléments de types : Inline-block

Type de boîte	Propriétés	Exemples d'utilisation
<code>block</code>	Occupe toute la largeur, commence sur une nouvelle ligne	<code>&lt;div&gt;</code> , <code>&lt;p&gt;</code> , <code>&lt;h1&gt;</code> , <code>&lt;ul&gt;</code>
<code>inline</code>	N'occupe que l'espace nécessaire, ne crée pas de saut de ligne	<code>&lt;span&gt;</code> , <code>&lt;a&gt;</code> , <code>&lt;img&gt;</code> , <code>&lt;strong&gt;</code>
<code>inline-block</code>	Comme <code>inline</code> , mais accepte les propriétés de taille et de marge	Menus, boutons alignés
<code>flex</code>	Disposition flexible en ligne ou colonne, avec contrôle de l'alignement	Barre de navigation, grilles de produits
<code>grid</code>	Disposition en lignes et colonnes, contrôle avancé du placement des éléments	Galeries d'images, mises en page complexes
<code>table</code>	Disposition en tableau avec lignes et cellules	Disposition de données tabulaires



# Eléments de type : non affichés

Les éléments de type "non affichés" (ou "hidden") sont des éléments HTML qui ne sont pas visibles dans la page. Bien qu'ils soient présents dans le code, ils ne prennent pas de place à l'affichage, contrairement aux éléments visibles.

## **Exemple:**

Certains éléments peuvent être rendus non visibles grâce à CSS avec `display: none;` ou `visibility: hidden;`

•**`display: none;`** : L'élément disparaît complètement de la page et ne prend aucune place.

•**`visibility: hidden;`** : L'élément devient invisible, mais l'espace qu'il occupe est toujours réservé.

# Eléments de type : non affichés

```
<p>Ceci est un paragraphe visible.</p>
<p class="non-affiche">Ce paragraphe est masqué avec <code>display: none;</code>.</p>
<p class="invisible">Ce paragraphe est invisible avec <code>visibility: hidden;</code>.</p>
```



```
<style>
    .non-affiche {
        display: none;
    }
    .invisible {
        visibility: hidden;
    }
</style>
```



Ceci est un paragraphe visible.



# Propriétés CSS pour les tableaux :

- Les tableaux sont généralement utilisés pour afficher des données tabulaires. Mais lorsque vous créer un tableau HTML sans style ni attribut, les navigateurs les affichent sans aucune bordure.
- Avec CSS, vous pouvez grandement améliorer l'apparence de vos tableaux.
- CSS fournit plusieurs propriétés qui nous permettent de contrôler la mise en page et la présentation des éléments du tableau.

Nom	Âge	Ville
Rim	25	Fes
Majd	20	Rabat



# Propriétés CSS pour les tableaux :

## Bordures :

Les bordures sont des lignes qui encadrent les éléments d'un tableau, permettant de délimiter visuellement les cellules et le tableau lui-même.

Nom	Âge	Ville
Rim	25	Fes
Majd	20	Rabat

```
table, th, td {  
    border: 1px solid black;  
    border-collapse: collapse;  
}
```

Nom	Âge	Ville
Rim	25	Fes
Majd	20	Rabat

## Propriété de bordure :

- **border** : Définit une bordure solide de **1 pixel** d'épaisseur et de type **solide**, et d'une **couleur noire** autour des éléments spécifiés.
- **border-collapse** : Fusionne les bordures adjacentes pour éviter les doubles bordures, ce qui donne un aspect plus propre.

# Propriétés CSS pour les tableaux: **Espacement et Marge**

## Espacement intérieur :

- L'espacement intérieur (ou padding) est la zone entre le contenu d'une cellule et la bordure de celle-ci. Cela améliore la lisibilité.

```
th, td {  
|   padding: 10px;  
}
```



Nom	Âge	Ville
Rim	25	Fes
Majd	20	Rabat

→ **padding** : Ajoute de l'espace à l'intérieur de chaque cellule, ce qui améliore la lisibilité en évitant que le texte ne soit collé aux bords des cellules.



# Propriétés CSS pour les tableaux: **Style de fond**

## Couleur de fond :

- Les couleurs de fond permettent de définir une couleur derrière le contenu d'une cellule, ce qui peut améliorer la distinction entre les éléments.

```
th {  
|   background-color: lightblue;  
}  
}
```



Nom	Âge	Ville
Rim	25	Fes
Majd	20	Rabat

→ **background-color** : Définit la couleur de fond des cellules. Les en-têtes auront une couleur bleu claire.



# Propriétés CSS pour les tableaux: Dimensions du tableau

## Largeur et hauteur :

- Les dimensions d'un tableau en CSS font référence à la largeur et à la hauteur du tableau, et peuvent être définies de plusieurs manières en fonction de la mise en page souhaitée. Voici quelques façons de contrôler les dimensions du tableau:

```
table {  
    width: 100%;  
}  
th, td {  
    height: 40px;  
}
```

- **width** : Définit que le tableau doit occuper 100% de la largeur de son conteneur.
- **height** : Définit une hauteur fixe pour les cellules, assurant une apparence uniforme.

Nom	Âge	Ville
Rim	25	Fes
Majd	20	Rabat



# Propriétés CSS pour les tableaux: **Coins arrondis**

## Arrondir les coins :

- Les coins arrondis ajoutent une courbure aux angles d'un élément, ce qui peut adoucir l'apparence générale du tableau.

```
table {  
    border-collapse: separate;  
    border-radius: 8px;  
    overflow: hidden;  
    border: 1px solid red;  
}
```

→ **border-collapse: separate:** Chaque cellule du tableau a une bordure distincte. Cela est particulièrement utile lorsque vous voulez appliquer un border-radius sur les coins du tableau sans que les bordures des cellules se chevauchent.

→ **border-radius :** Ajoute des coins arrondis au tableau de 8px. Cela rend les bords du tableau plus doux et moins anguleux.

→ **overflow: hidden :** Assure que les coins arrondis sont visibles même si le contenu déborde.

Nom	Âge	Ville
Rim	25	Fes
Majd	20	Rabat

# Propriétés CSS pour les tableaux: <caption>

- L'élément **<caption>** est utiliser pour ajouter une légende ou un titre à un tableau en HTML.
- Cet élément optionnel, s'il est présent, doit impérativement suivre la balise ouvrante **<table>**.
- Par défaut, la légende est graphiquement positionnée au-dessus du tableau.

```
<table>
  <caption>Liste des Employés</caption>
  <thead>
    <tr>
      <th>Nom</th>
      <th>Âge</th>
      <th>Ville</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Rim</td>
      <td>25</td>
      <td>Fes</td>
    </tr>
    <tr>
      <td>Majd</td>
      <td>20</td>
      <td>Rabat</td>
    </tr>
  </tbody>
</table>
```



Liste des  
Employés  
**Nom Âge Ville**  
Rim 25 Fes  
Majd 20 Rabat



# Propriétés CSS pour les tableaux: <caption>

- Contrôle de la position de la légende :

Vous pouvez définir la position verticale d'une légende de tableau à l'aide de la propriété **caption-side**. La légende peut être placée en haut (**top**) ou en bas (**bottom**) du tableau. La position par défaut est **top**

```
table {  
    width: 50%;  
    border-collapse: collapse;  
    margin: 20px 0;  
}  
caption {  
    caption-side: bottom;  
    font-size: 1.5em;  
    font-weight: bold;  
    text-align: center;  
    margin: 10px 0;  
}  
th, td {  
    padding: 10px;  
    border: 1px solid #ddd;  
    text-align: center;  
}  
th {  
    background-color: #f2f2f2;  
    font-weight: bold;  
}
```

```
caption {  
    |   caption-side: bottom;  
    | }  
}
```

Nom	Âge	Ville
Rim	25	Fes
Majd	20	Rabat

## Liste des Employés

# FLEXBOX

**Flexbox** est une méthode de mise en page selon un axe principal, permettant de disposer des éléments **en ligne** ou **en colonne**. Les éléments se dilatent ou se rétractent pour occuper l'espace disponible.

Pour faire de la mise en page avec Flexbox, il faut :

- Définir un conteneur.
- Placer à l'intérieur plusieurs éléments.

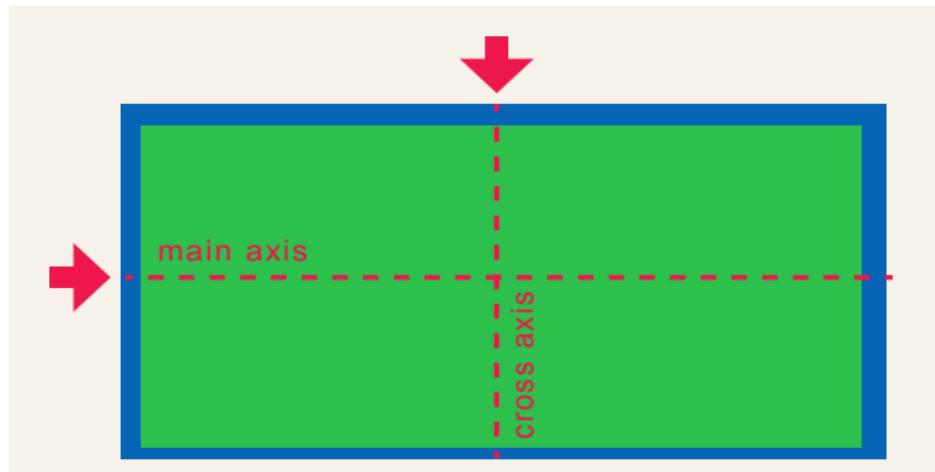
## Exemple:

Imaginez un carton dans lequel vous rangez plusieurs objets : c'est le principe ! Sur une même page web, vous pouvez avoir plusieurs conteneurs (plusieurs cartons, si vous préférez). Vous pouvez en créer autant que nécessaire pour obtenir la mise en page que vous voulez.



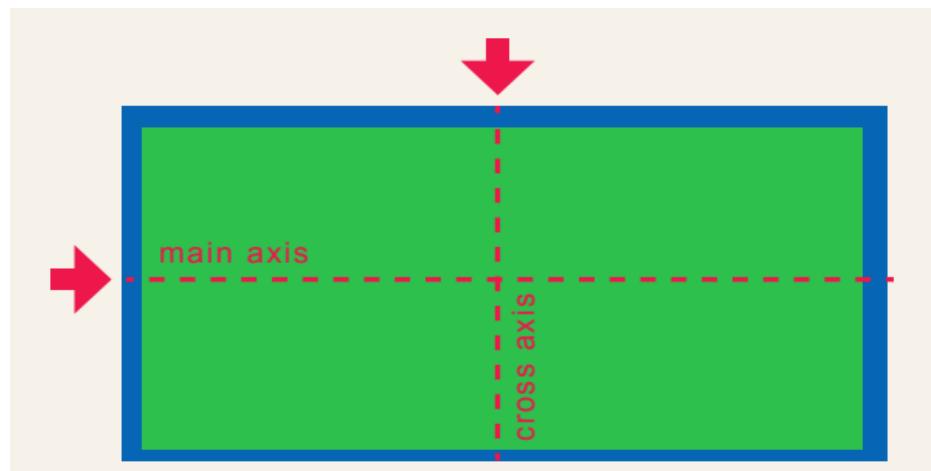
# FLEXBOX

- **L'axe principal (main axis):** L'axe de la direction dans laquelle sont disposés les éléments flex.
- Le début et la fin de cet axe sont appelés l'origine principale (**main start**) et la fin principale (**main end**).
- La direction de cet axe est définie par la valeur de flex-direction :
  - **Flex-direction: row;** place l'axe principal horizontalement (de gauche à droite).
  - **Flex-direction: column;** place l'axe principal verticalement (de haut en bas).



# FLEXBOX

- **L'axe croisé (cross axis):** l'axe perpendiculaire à l'axe principal, c'est-à-dire à la direction dans laquelle sont disposés les éléments flex.
- Le début et la fin de cet axe sont appelés le début (**cross start**) et la fin (**cross end**) de l'axe croisé.
  - Si l'axe principal est horizontal (row), l'axe croisé sera vertical, et inversement.
  - Début de l'axe croisé (cross start) : le point de départ de cet axe.
  - Fin de l'axe croisé (cross end) : le point d'arrivée de cet axe.
- **display: flex** est le **conteneur flex (flex container)**.
- Les éléments disposés en tant que boîtes flexibles à l'intérieur du conteneur flex sont appelés éléments flex (**flex items**).



# FLEXBOX

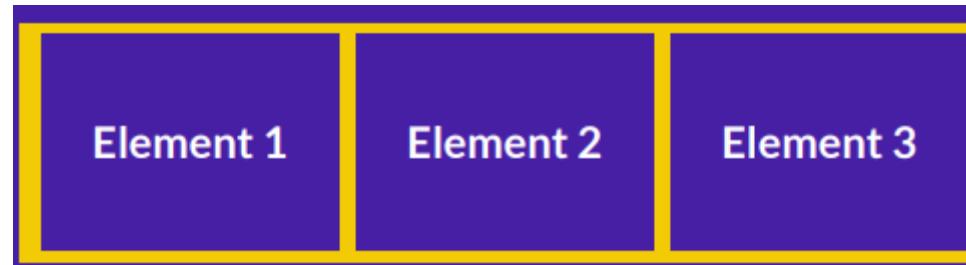
## Les éléments des boîtes flexibles:

Un conteneur (container) est une balise qui peut renfermer d'autres balises, comme du texte ou encore des images.

Les conteneurs les plus célèbres sont les balises `<div>` et `<span>`.

```
<div class="container">
<div class="element element1">Élément 1</div>
<div class="element element2">Élément 2</div>
<div class="element element3">Élément 3</div>
</div>
```

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



# Flexbox: Flex-direction

## La direction avec la propriété : Flex-direction

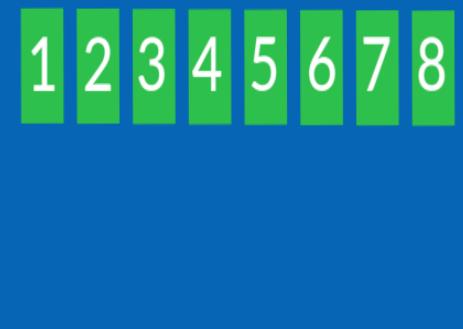
Par défaut, les blocs essaient de rester sur la même ligne s'ils n'ont pas la place, quitte à "s'écraser", et provoquer parfois des anomalies dans la mise en page (certains éléments pouvant dépasser de leur conteneur).

Les blocs aillent à la ligne lorsqu'ils n'ont plus la place, avec **flex-wrap**.

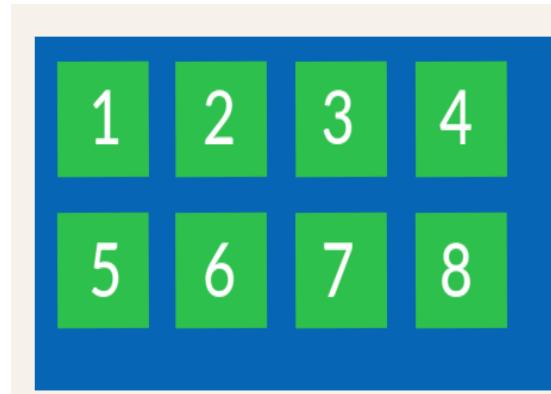
Voilà les différentes valeurs de flex-wrap :

- **nowrap** : Pas de retour à la ligne (par défaut)
- **wrap** : Les éléments vont à la ligne lorsqu'il n'y a plus la place
- **wrap-reverse** : Les éléments vont à la ligne, lorsqu'il n'y a plus la place, en sens inverse.

```
.container {
    display: flex;
    flex-wrap: nowrap;
    /* OU wrap;
    OU wrap-reverse; */
}
```



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



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



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



# Flexbox: Alignement des contenus

## Alignez les éléments sur un axe principal et secondaire:

Les blocs sont répartis sur plusieurs lignes avec **align-content**. S'il ya plusieurs lignes dans le Flexbox, choisir comment celles-ci seront réparties avec align-content. Cette propriété n'a aucun effet s'il n'y a qu'une seule ligne dans la Flexbox.

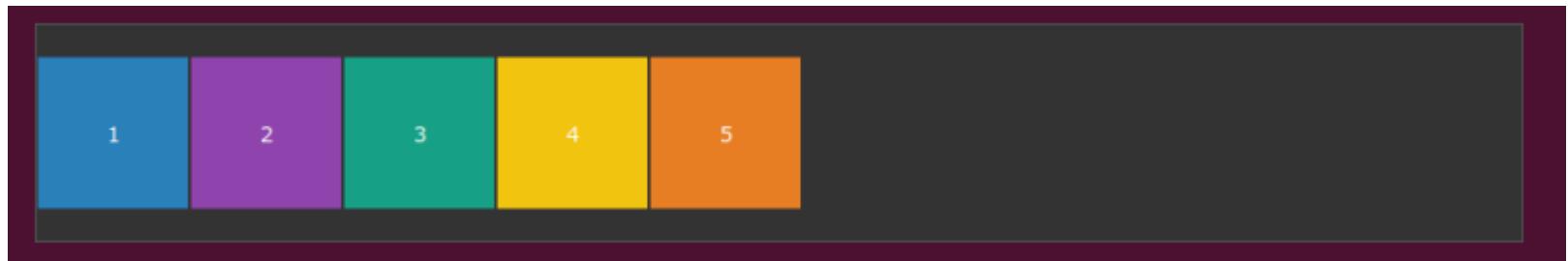
La propriété **align-content** peut prendre ces valeurs :

- stretch; flex-start ;flex-end ;center; space-between ; space-around

# Flexbox: Alignement des contenus

- **Flex-start** : les éléments sont placés au début

```
.parent {  
    justify-content: flex-start;  
}
```

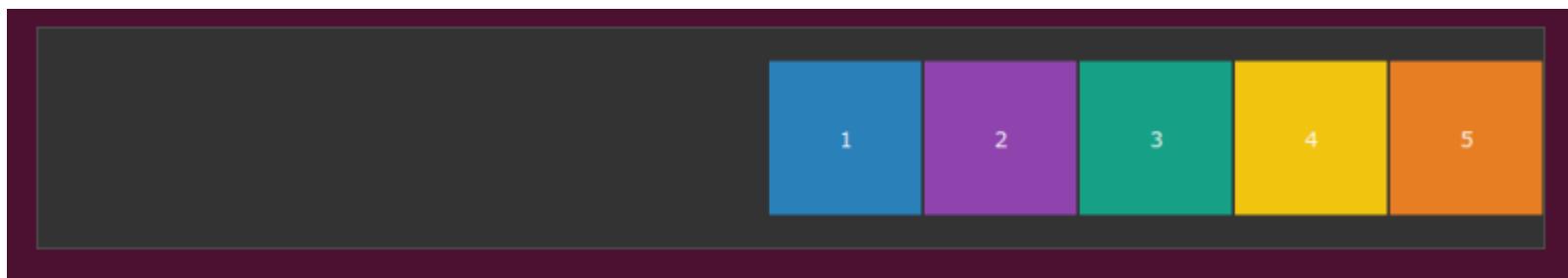




# Flexbox: Alignement des contenus

- **Flex-end** : les éléments sont placés à la fin

```
.parent {  
    justify-content: flex-end;  
}
```





# Flexbox: Alignement des contenus

- **Flex-center** : les éléments sont placés au centre:

```
.parent {  
    justify-content: center;  
}
```





# Flexbox: Alignement des contenus

- **Space-between:** les éléments sont séparés avec de l'espace entre eux.

```
.parent {  
    justify-content: space-between;  
}
```

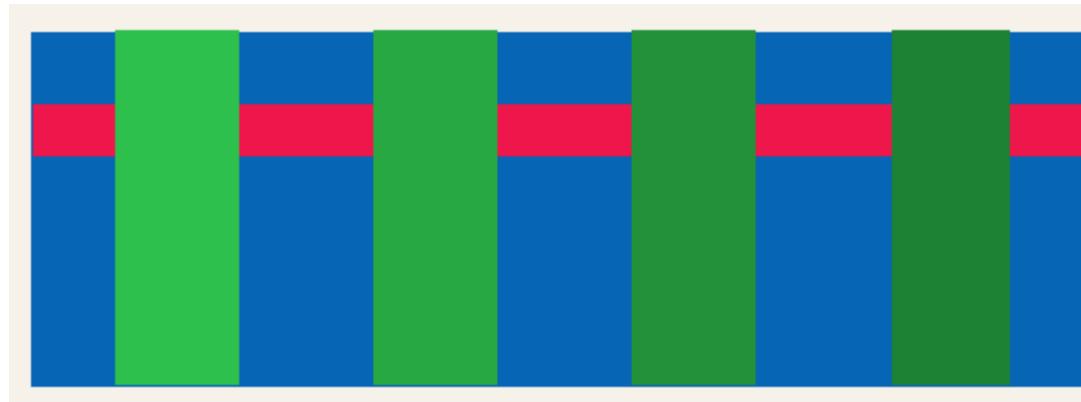




# Flexbox: Alignement des contenus

- **Space-around:** idem, mais il y a aussi de l'espace au début et à la fin..

```
.parent{  
    justify-content: space-around;  
}
```





# Flexbox: Alignement des contenus

flex-start

Element 1 Element 2 Element 3

flex-end

Element 1 Element 2 Element 3

center

Element 1 Element 2 Element 3

space-between

Element 1 Element 2 Element 3

space-around

Element 1 Element 2 Element 3



# Bootstrap

## Pourquoi le framework Bootstrap ?

- Développer des sites web
  - Très bonne connaissance des langages du web (HTML, CSS, JavaScript) , mais apprentissage long et difficile
    - Frameworks pour faciliter cette tâche
    - Collections de briques de code bien structurées et prêtes à l'emploi permettant de faciliter le développement et la maintenance
    - L'un des plus utilisés : Bootstrap <http://getbootstrap.com/>
      - Twitter → open source
    - Facilite l'utilisation des règles CSS pour concevoir des sites attractifs et adaptatifs (Responsive Web Design - RWD)
    - Grille d'affichage pour agencer les différentes boîtes d'affichage des sites et s'adaptant immédiatement aux écrans de diffusion (ordinateurs, tablettes, smartphones)



# Bootstrap

**Bootstrap**, le pionnier des frameworks CSS, demeure le plus utilisé à ce jour, reconnu pour sa facilité d'utilisation et sa vaste bibliothèque de composants. Il permet aux développeurs de construire rapidement des interfaces utilisateur élégantes et réactives.

The screenshot shows the official Bootstrap website. At the top, there's a purple navigation bar with links for 'Docs', 'Examples', 'Icons', 'Themes', and 'Blog'. A search bar is also present. The main content area has a gradient background transitioning from light blue to yellow. In the center, there's a large purple button containing a white 'B' inside a speech bubble-like shape. Below this, the text 'Build fast, responsive sites with Bootstrap' is displayed in a large, bold, black font. Underneath, a smaller paragraph describes Bootstrap as a 'Powerful, extensible, and feature-packed frontend toolkit'. At the bottom, there are two buttons: a grey one labeled 'npm i bootstrap@5.3.2' and a purple one labeled 'Read the docs'. A small note at the very bottom says 'Currently v5.3.2 · Download · 60 releases'.

# Bootstrap

## Pourquoi le framework Bootstrap ?

Portabilité (cross-browser) :

- Présentation similaire quel que soit le navigateur

- Homogénéité :

- Ensemble de styles prédéfinis partagés

- éléments complémentaires (barres de navigation, boutons...)

- Simplicité :

- Système de grille pour positionnement des éléments

- Adaptabilité (responsive) :

- Par défaut l'affichage s'adapte à la taille de l'écran (mobile-first)

- Facilité d'utilisation :

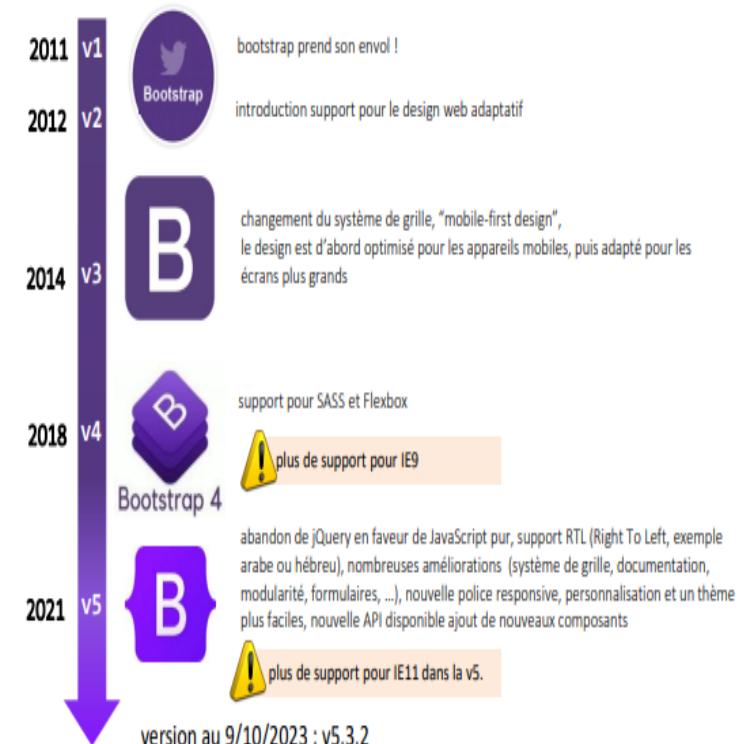
- Connaissances de base en HTML et CSS

- Temps d'apprentissage (learning curve)

- Uniformisation

- évolution rapide des versions

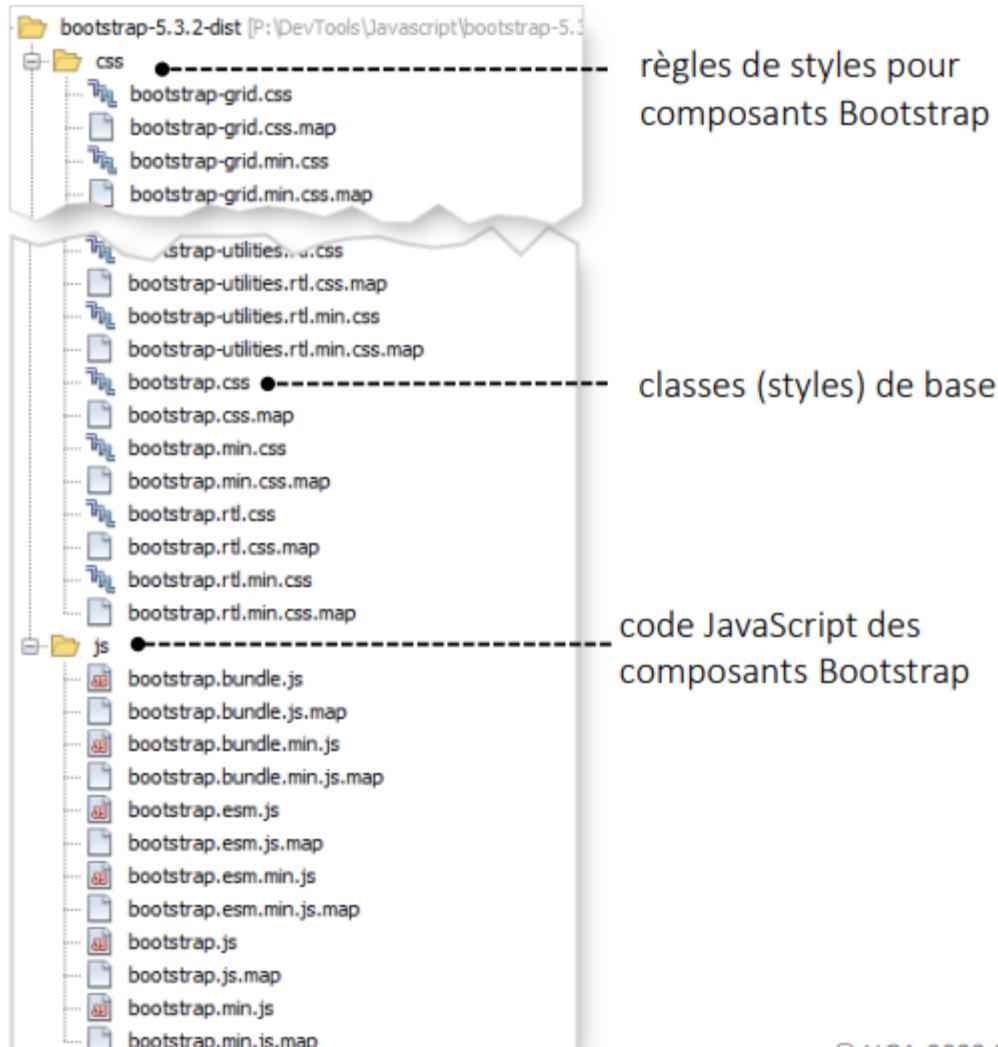
- Risque d'obsolescence des pages



# Bootstrap

Comment utiliser Bootstrap: version locale téléchargée.

1ère étape: télécharger une distribution <http://getbootstrap.com/>



# Bootstrap

## 2 ème étape: intégrer les éléments Bootstrap à une page HTML

index.html

```

<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple Bootstrap 5</title>

    <link rel="stylesheet" href="thirdparties/bootstrap5/css/bootstrap.css">
</head>

<body>
    <div class="container">
        <h1>Ma première page Bootstrap 5 </h1>
        <p>
            Lorem ipsum dolor sit amet consectetur,
            adipisicing elit. Officia autem nesciunt ipsum,
            modi dignissimos distinctio molestias illo
            optio recusandae. Accusantium expedita rerum
            quasi fuga suscipit esse cum optio eius ducimus.
        </p>
    </div>

    <script src="thirdparties/bootstrap5/js/bootstrap.bundle.js"></script>
</body>

</html>

```

les référencer (liens relatifs) depuis vos pages HTML

2

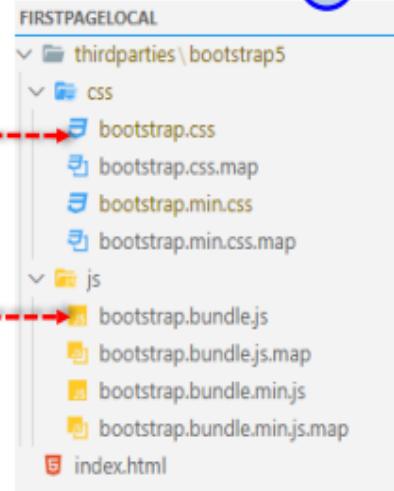


Certains composants de Bootstrap nécessitent également de référencer la librairie javascript de Bootstrap.

3

recopier dans votre site les éléments de Bootstrap dont vous avez besoin

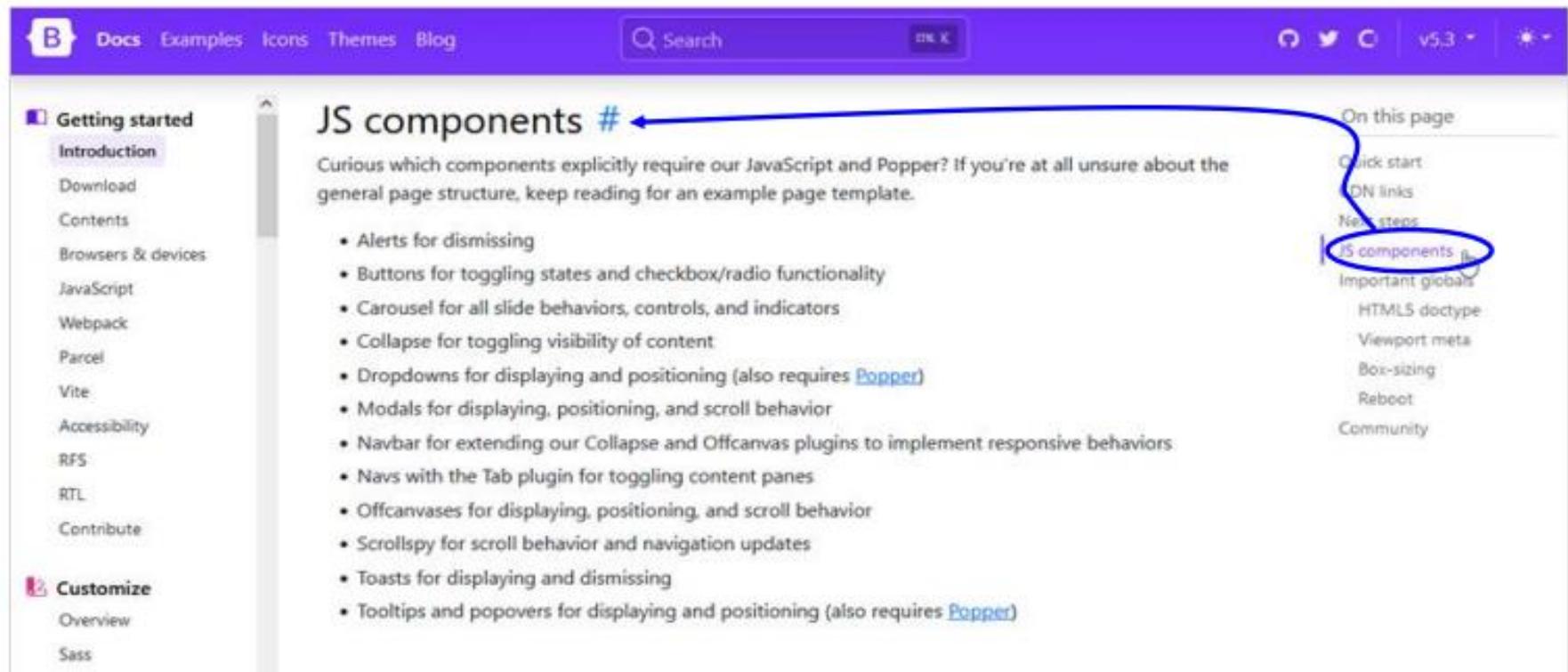
1



# Bootstrap

## Composants nécessitant Javascript

<https://getbootstrap.com/docs/5.3/getting-started/introduction>



The screenshot shows the Bootstrap documentation website with a purple header. The left sidebar has sections for "Getting started" (with links like "Introduction", "Download", "Contents", "Browsers & devices", "JavaScript", "Webpack", "Parcel", "Vite", "Accessibility", "RFS", "RTL", "Contribute") and "Customize" (with links for "Overview" and "Sass"). The main content area has a search bar and navigation icons. A blue arrow points from the heading "JS components" to a list of components. Another blue oval highlights the "JS components" link in the "On this page" sidebar.

**JS components #**

Curious which components explicitly require our JavaScript and Popper? If you're at all unsure about the general page structure, keep reading for an example page template.

- Alerts for dismissing
- Buttons for toggling states and checkbox/radio functionality
- Carousel for all slide behaviors, controls, and indicators
- Collapse for toggling visibility of content
- Dropdowns for displaying and positioning (also requires [Popper](#))
- Modals for displaying, positioning, and scroll behavior
- Navbar for extending our Collapse and Offcanvas plugins to implement responsive behaviors
- Navs with the Tab plugin for toggling content panes
- Offcanvases for displaying, positioning, and scroll behavior
- Scrollspy for scroll behavior and navigation updates
- Toasts for displaying and dismissing
- Tooltips and popovers for displaying and positioning (also requires [Popper](#))

On this page

- Quick start
- ON links
- New steps
- JS components** ↗
- Important globals
- HTML5 doctype
- Viewport meta
- Box-sizing
- Reboot
- Community

# Bootstrap

Utiliser Bootstrap: avec une alternative **un CDN**(Content Delivery Network).

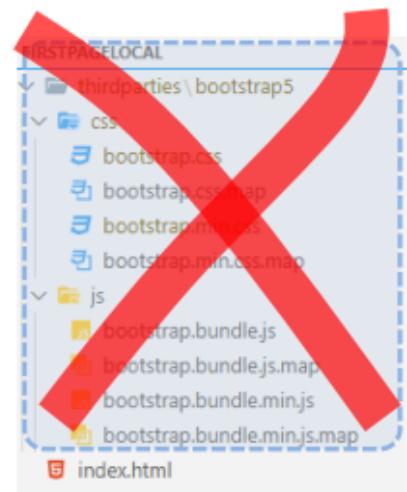
Un CDN ou réseau de diffusion de contenu, est un réseau de serveurs répartis dans différents endroits géographiques, conçu pour distribuer rapidement des contenus web aux utilisateurs. Inclure directement les liens vers les fichiers CSS et JavaScript de Bootstrap dans le <head> de le fichier HTML. Cela permet de charger Bootstrap sans avoir à télécharger les fichiers sur votre serveur.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple Bootstrap 5</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
          rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
          crossorigin="anonymous">
</head>

<body>
    <div class="container">
        <h1>Ma première page Bootstrap 5 </h1>
        <p>
            Lorem ipsum dolor sit amet consectetur,
            adipisicing elit. Officia autem nesciunt ipsum,
            modi dignissimos distinctio molestias illo
            optio recusandae. Accusantium expedita rerum
            quasi fuga suscipit esse cum optio eius ducimus.
        </p>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
           integrity="sha384-C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNxj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"
           crossorigin="anonymous"></script>
</body>

</html>
```

Les ressources statiques associées  
à Bootstrap sont chargées à partir  
de différents CND





# Bootstrap

## Avantages de l'utilisation d'un CDN

- **Chargement plus rapide** : Les CDNs sont optimisés pour la vitesse et sont souvent stockés en cache dans les navigateurs.
- **Pas besoin de téléchargement** : Pas besoin de télécharger Bootstrap sur votre serveur.
- **Facilité de mise à jour** : Vous pouvez facilement mettre à jour vers une version plus récente de Bootstrap en modifiant simplement l'URL du CDN.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple de page avec Bootstrap CDN</title>
    <!-- Lien CDN pour le CSS de Bootstrap -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css" integrity="sha384-xOolHFLh07PJGoPkLv1IbcE&t3P" crossorigin="anonymous">
</head>
<body>
    <div class="container">
        <h1 class="text-center mt-5">Bienvenue sur Bootstrap via CDN</h1>
        <p class="lead text-center">Cette page utilise Bootstrap en incluant les liens CDN pour le CSS et le JavaScript.</p>
    </div>

    <!-- Lien CDN pour jQuery, Popper.js et Bootstrap JS -->
    <!--SCRIPT Bootstrap-->
    <script src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.slim.min.js" integrity="sha384-Dfxz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+I&t3P" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-Fy6S3B9q64WdZWQuiU+q4/2Lc9npb8tCasX9F" crossorigin="anonymous"></script>
</body>
</html>
```