# Introduction au CSS

# Définition

# *Cascading Style Sheets*

Les CSS (feuilles de style) sont le code utilisé pour **mettre en forme une page web**

Tout comme HTML, il ne s'agit pas d'un langage de programmation mais plutôt de déclaration de styles

Grâce à CSS, on peut donc d'appliquer des styles sur **différents éléments HTML sélectionnés dans le document**

Par exemple, ce code permet de sélectionner tous les `<p>` du document et leur appliquer une couleur de texte en rouge

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

Nous reviendrons dans un instant sur la syntaxe du CSS ...

# Liaison à la page Web

On écrit généralement les CSS dans des fichiers portant l'extension `.css` (exemple : `styles.css`)

C'est la méthode recommandée de séparer les fichiers `.css` des fichiers `.html`

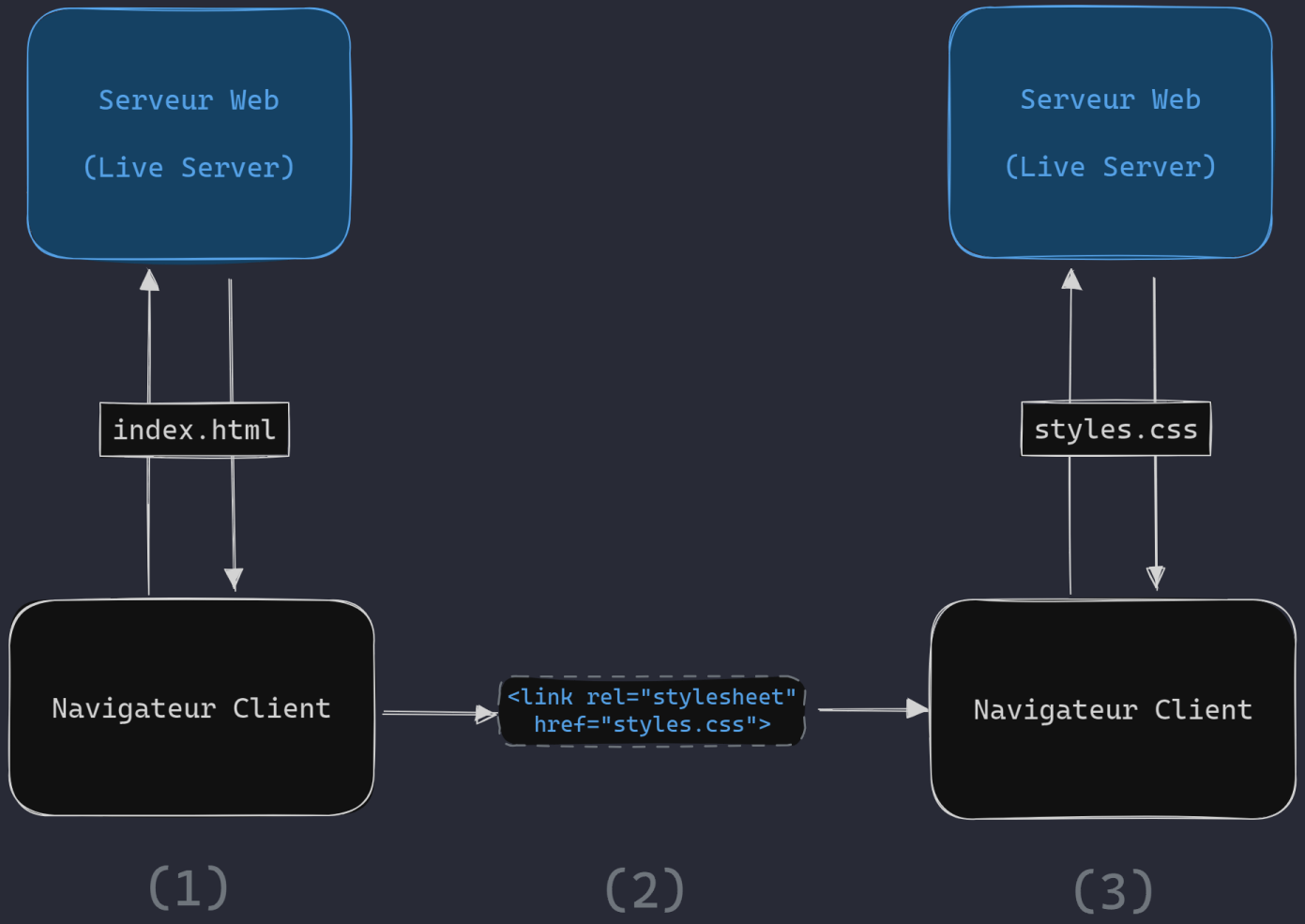
Il faut ensuite lier la page HTML au fichier `.css` créé

**C'est dans le fichier HTML que l'on va indiquer d'inclure le CSS**

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="utf-8">
  <title>Titre de la page Web</title>

  <link rel="stylesheet" href="styles.css">
</head>
<body>
  …
</body>
</html>
```

La balise <link> se place dans l'en-tête du document et déclare ici que cette page HTML fait appel au fichier de style `styles.css`


Serveur Web
(Live Server)
Serveur Web
(Live Server)
index.html
styles.css
Navigateur Client
<link rel="stylesheet"
href="styles.css">
Navigateur Client
(1)
(2)
(3)

# Exercice

Ouvrez un nouveau dossier de travail `/Intro CSS/` dans VSCode et créez une page HTML contenant un titre et un paragraphe, puis créez un fichier de style pour y placer le code suivant :

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

Liez ensuite le fichier CSS à votre page avec la balise :

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

Le paragraphe devrait s'afficher en rouge

# Correction

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="utf-8">
  <title>Titre de la page Web</title>

  <link rel="stylesheet" href="styles.css">
  <link rel="stylesheet" href="theme.css">
  <link rel="stylesheet" href="layout.css">
</head>
<body>
  …
</body>
</html>
```

On peut inclure autant de feuilles CSS que l'on souhaite

À noter qu'il est aussi possible d'écrire du CSS directement dans la page Web, mais cette méthode n'est pas recommandée car elle empêche la **réutilisation des styles dans les différentes pages HTML**

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="utf-8">
  <title>Titre de la page Web</title>

  <style>
    /* Code CSS ici */
  </style>
</head>
<body>
  …
</body>
</html>
```

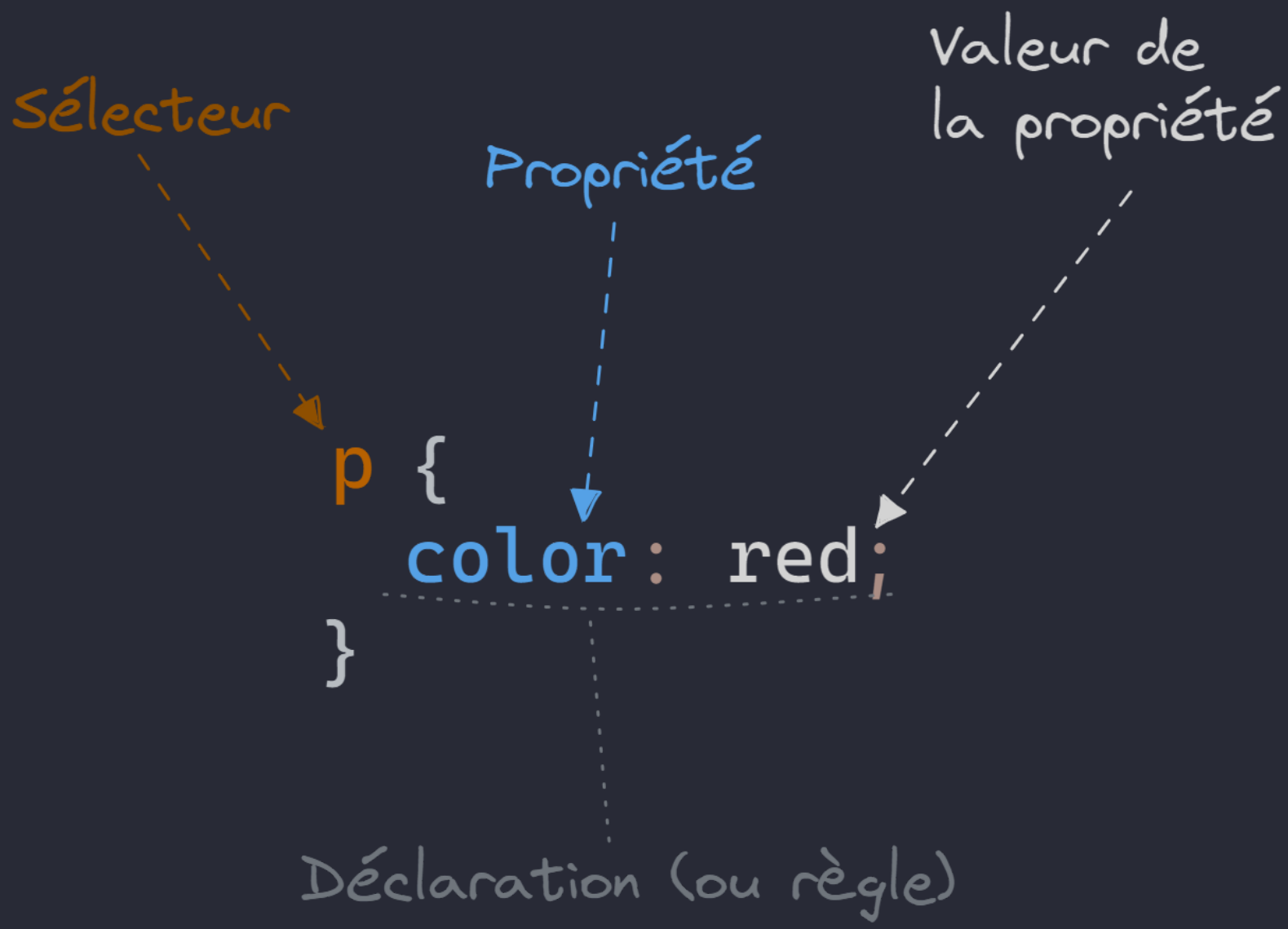
On l'utilise en pratique pour déclarer des styles additionnels spécifiquement pour cette page

# Anatomie du langage CSS

Les CSS déclare des **ensembles de règles** dont la syntaxe est bien précise

Analysons la structure de cet ensemble ...

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


Sélecteur
Propriété
Valeur de la propriété
p {
color: red;
}
Déclaration (ou règle)

Chaque ensemble de règles, à l'exception du sélecteur, doit être entre accolades { }

# Sélecteur

Le sélecteur permet de sélectionner les éléments sur lesquels appliquer le style souhaité

```css
p {
  color: red;
}

h1 {
  color: blue;
}
```

Tous les `<p>` seront en rouge, et tous les `<h1>` seront en bleu

# Déclaration

La déclaration est une règle simple qui détermine les *propriétés* de l'élément que l'on veut mettre en forme

⚠ Une déclaration se termine toujours par un « ; »

```
h1 {
  color: blue;
  font-size: 42px;
  text-decoration: underline;
}
```

Trois déclarations de règle indiquant que les <h1> sont bleus, soulignés et ont une taille de 42 pixels

# Propriété

La propriété est la partie gauche de la déclaration (avant les deux points `:`)

Il en existe un très grand nombre !

```
font-family
font-style
font-size
color
margin
padding
border
…
```

# Valeur de propriété

La valeur de propriété est la partie droite de la déclaration (après les deux points `:`)

Elle dépend de la propriété elle-même

```
font-family: Arial, sans-serif;
font-style: italic;
font-size: 18px;
color: gray;
margin: 8px;
padding: 8px;
border: 1px solid silver;
…
```

# Commentaires

Comme en HTML, il est possible d'annoter le code CSS avec des commentaires pour le développeur

On utilise les délimiteurs `/*` et `*/`. Tout ce qui se trouve entre est ignoré

```css
/* Ceci est un commentaire CSS */
p {
  color: red;
  /* text-decoration: underline; */
}
```

Pour styliser des éléments HTML, il faut donc :

1. sélectionner le·s élément·s concerné·s
2. leur déclarer des règles de style

Avant de voir les différentes règles possibles, commençons par quelques sélecteurs basiques …

# Les sélecteurs

Un sélecteur permet de cibler un groupe de balises HTML

Le plus simple est le sélecteur d'élément

```
section {

}
```

Les règles présentes dans cet ensemble seront appliquées à **toutes les balises de `<section>`**

On peut cibler plusieurs balises HTML en utilisant la virgule

```
h1, h2, h3 {

}
```

Ici, l'ensemble cible les balises <h1>, <h2> et <h3>

# Sélecteur de parenté

On peut sélectionner des éléments de façon plus spécifique via leur *liens de parenté*

Admettons le HTML suivant :

```
<h1>Titre de page</h1>
<article>
  <h1>Titre d'article</h1>
  <p>Contenu d'article</p>
</article>
```

Comment peut-on faire pour sélectionner en CSS le `<h1>` de l'article uniquement ?

```html
<h1>Titre de page</h1>
<article>
  <h1>Titre d'article</h1>
  <p>Contenu d'article</p>
</article>
```

```css
article h1 {

}
```

L'espace «  » entre `article` et `h1` signifie que l'on sélectionne tous les titre de niveau 1 se trouvant dans un article, **à quelque niveau que ce soit**

Il existe d'autres variantes du même genre comme les sélecteurs suivants :

```
1 article > h1
2 /* Tous les <h1> se trouvant DIRECTEMENT dans un article */
3
4 header + section
5 /* Toutes les <section> qui SUIVENT DIRECTEMENT un header */
6
7 header ~ section
8 /* Toutes les <section> qui se trouvent APRÈS un header */
```

# Sélecteur par attributs

Comme son nom l'indique, cette sélection permet de cibler des balises en fonction de leurs attributs

Imaginons que nous souhaitions sélectionner toutes les `<img>` qui disposent d'un attribut `alt`

```
1 <img src="fond.jpg">
2 <img src="chat.jpg" alt="Chat dormant sur le lit">
3 <img src="chien.jpg" alt="Chien jouant dans l'herbe">
```

```html
<img src="fond.jpg">
<img src="chat.jpg" alt="Chat dormant sur le lit">
<img src="chien.jpg" alt="Chien jouant dans l'herbe">
```

```css
img[alt] {

}
```

Les crochets [ ] en CSS désignent la sélection par attribut

# On peut même être plus précis et sélectionner des attributs par leur valeur

```
a[href='#']
/* Tous les liens avec un attribut href égal à '#' */

a[target='_blank']
/* Tous les liens avec un attribut target égal à '_blank' */
```

# Sélecteur par classe

Une classe CSS est un nom que l'on donne à un élément HTML pour le cibler plus facilement

On utilise l'attribut HTML `class`

```html
<a href="#" class="button-danger">Supprimer</a>
```

Comment cibler cet élément en CSS ?

```html
<a href="#" class="button-danger">Supprimer</a>
```

```css
.button-danger {

}
```

Un nom de classe en CSS commence **toujours par un point**

Une classe a pour vocation d'être **réutilisable et modulaire**

Évitez les noms de classe du style `.partie1` ou `.rouge`

# Règles de syntaxe

Un nom de classe doit **toujours** commencer par une lettre

Il peut ensuite contenir des lettres, des chiffres, des tirets et des underscores (mais pas d'espaces)

```
.badge          ✅
.badge big      ❌
.badge-big      ✅
.badge_big      ✅
.1-badge        ❌
.badge-1        ✅
```

# Sélecteur par ID

Un sélecteur par ID s'utilise de la même façon qu'une classe, sauf qu'on utilise l'attribut HTML `id="…"` et en CSS le signe `#` au lieu du point

La principale différence est qu'un ID ne peut être utilisé qu'une seule fois par document

```
<section id="jumbotron">…</section>
<section id="jumbotron">…</section>
❌ Erreur : deux éléments ont le même ID !
```

(Dans cet exemple, on devrait plutôt utiliser une classe)

On choisit généralement un ID pour désigner un élément unique dans le document

```
<article id="a-propos">…</article>
```

```
<article id="a-propos">…</article>
```

```
#a-propos {

}
```

Un ID a pour vocation d'être **unique et explicite**

Évitez les noms du style `#section1` ou `#partieB`

S'appliquent les mêmes règles de syntaxe que les classes

# Sélection plus précise

Il est possible de combiner la sélection d'élément avec les classes et les IDs

```
a.button {
  /* Tous les liens <a> avec la classe "button" */
}

section#about-us {
  /* Toutes les sections <section> avec l'ID "about-us" */
}
```

# CSS Diner

## Jeu des sélecteurs

🔗 https://flukeout.github.io/

Retrouvez plus de détails sur cet article 🔗 **du MDN à propos des sélecteurs CSS basiques**

Voyons maintenant quelques *propriétés* courantes en CSS pour appliquer des styles ...

# Propriétés CSS courantes

# Couleur de texte

La propriété `color` permet de définir la couleur du texte affiché

Elle prend une *couleur* comme valeur

Les valeurs de couleurs en CSS sont représentées de différentes manières

```css
p { color: red; }
p { color: #ff0000; }
p { color: rgb(255, 0, 0); }
p { color: hsl(0, 100%, 50%); }
p { color: rgba(255, 0, 0, 1); }
p { color: hsla(0, 100%, 50%, 1); }
```

🔗 fffuel.co/cccolor

# Taille de texte

La propriété `font-size` permet de définir la taille du texte affiché

Elle prend une *mesure de distance* comme valeur

# Il existe de nombreuses types de mesures en CSS

```css
p { font-size: 18px; }
p { font-size: 1.2em; }
p { font-size: 1.2rem; }
p { font-size: 1.2%; }
p { font-size: 1.5vw; }
p { font-size: 1.5vh; }
p { font-size: 1.5vmin; }
p { font-size: 1.5vmax; }
```

🔗 katydecorah.com/css-ruler

# Police de caractères

La propriété `font-family` permet de définir la police de caractères utilisée

```
p {
  font-family: Calibri;
}
```

# On peut définir une liste de polices séparées par des virgules pour gérer le *fallback*

```
p {
  font-family: Calibri, Arial, sans-serif;
}
```

Dans ce cas, si la police "Calibri" n'est pas disponible, on utilisera la police "Arial" à la place

Notez que les noms de polices contenant des espaces doivent être encadrés par des guillemets

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

Il est possible d'utiliser des polices personnalisées grâce à la règle CSS `@font-face` que nous découvrirons plus tard

# Alignement de texte

La propriété `text-align` permet de définir l'alignement du texte

```
p { text-align: center; }
p { text-align: left; }
p { text-align: right; }
p { text-align: justify; }
```

# Dimensions de blocs

Jusqu'à présent, nous avons vu des propriétés qui s'appliquent au texte

Il existe aussi des propriétés qui s'appliquent aux éléments de type *bloc* comme les `<div>` ou les `<section>`

Par défaut, un élément de type *bloc* va créer un bloc occupant la largeur totale qu'il lui est possible de prendre dans son contexte d'affichage

On peut agir sur ces dimensions grâce aux propriétés `width` et `height`

```html
<div class="avatar">
  <img src="https://i.pravatar.cc" alt="Photo utilisateur">
</div>
```

```css
.avatar {
  width: 150px;
  height: 150px;
}
```

Testons ce code ensemble ...

Les dimensions semblent être appliquées au conteneur `<div class="avatar">`, mais pas à l'image ...

L'image est un élément spécial qui n'est pas affecté par défaut par rapport à la taille de son parent. Il faut donc lui préciser une règle de style à part

```html
<div class="avatar">
  <img src="https://i.pravatar.cc" alt="Photo utilisateur">
</div>
```

```css
.avatar {
  width: 150px;
  height: 150px;
}

.avatar img {
  max-width: 100%;
  max-height: 100%;
}
```

Les propriétés `max-width` et `max-height` limitent la taille **maximale** d'un élément, alors que `width` et `height` indiquent des dimensions strictes

# Les bordures

La propriété `border` permet de définir une bordure autour d'un élément

Une bordure est décomposée en 3 sous-propriétés :

1. une taille
2. un style
3. une couleur

```css
.avatar {
  border-width: 5px; /* taille */
  border-style: solid; /* style */
  border-color: #007090; /* couleur */
}
```

On utilise généralement la propriété condensée `border` pour rassembler ces éléments en 1 ligne :

```css
.avatar {
  border: 5px solid #007090;
}
```

# Les marges

On peut définir deux types de marges sur des éléments de type *bloc* :

1. les marges internes (`padding`)
2. les marges externes (`margin`)

```html
<div class="avatar">
  <img src="https://i.pravatar.cc" alt="Photo utilisateur">
</div>
```

```css
.avatar {
  /* … */
  padding: 5px;
  margin: 50px;
}
```

Vu qu'il y a 4 côtés pour un éléments, on peut choisir de définir les 4 marges séparément :

```
.avatar {
  /* … */
  margin-left: 25px;
  margin-right: 25px;
  margin-top: 50px;
  margin-bottom: 50px;
}
```

On peut aussi le faire de façon condensée en séparant les différentes marges par des espaces :

```
1 .avatar {
2   /* … */
3   margin: 50px 25px 50px 25px;
4 }
```

`margin: <haut> <droite> <bas> <gauche>`

(moyen mémo-technique : « *dans le sens des aiguilles d'une montre* »)

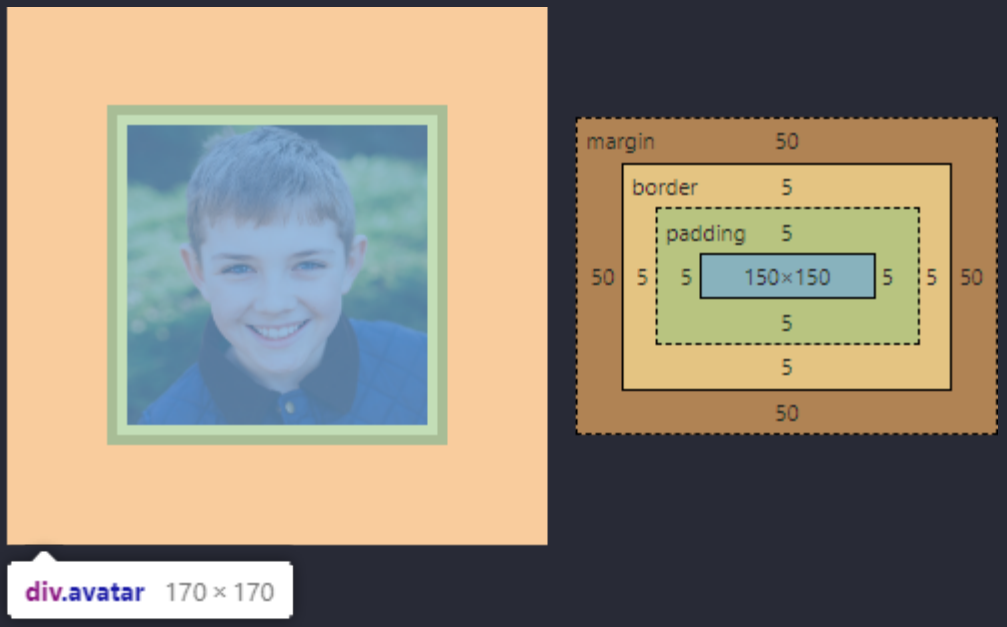
On peut même regrouper par axe vertical et horizontal, dans ce cas il y aura 2 valeurs :

```
.avatar {
  /* … */
  margin: 50px 25px;
}
```

`margin: <vertical> <horizontal>`

# Le modèle de boîtes

Dans le modèle standard, lorsqu'on spécifie les dimensions et marges d'un élément de type *bloc*, elles s'additionnent pour former la taille totale de l'élément




```css
.avatar {
  width: 150px;
  height: 150px;
  border: 5px solid #007090;
  padding: 5px;
  margin: 50px;
}
```

Au besoin, il est possible de modifier ce modèle de boîte via la propriété `box-sizing`

Pour plus de détails, allez consulter l'article suivant sur le MDN : 🔗 **Le modèle de boîte**

# Les arrière-plans

La propriété `background` permet de définir un arrière-plan à un élément

Tout comme les bordures ou les marges, elle se décline en sous-propriétés

L'utilisation la plus simple est la couleur d'arrière-plan, que l'on définit avec la propriété `background-color` qui prend une valeur de couleur

```
section {
  background-color: orange;
}
```

On peut aussi choisir une image d'arrière plan avec la propriété `background-image` qui prend comme valeur la *fonction url()*

```
section {
  background-image: url('images/fond.png');
}
```

⚠ Il est important de noter que le chemin vers le fichier image est **relatif à l'emplacement du fichier `.css`**

⚠️ Il est important de noter que le chemin vers le fichier image est **relatif à l'emplacement du fichier `.css`**

Si on considère l'arborescence suivante ...

```
index.html
├── styles/
│   └── global.css
└── images/
    └── fond.png
```

... il faudra alors préciser le chemin comme ceci :

```
background-image: url('../images/fond.png');
```

On peut choisir si l'image de fond doit se répéter ou non avec `background-repeat`

```
section {
  background-image: url('images/fond.png');
  background-repeat: no-repeat; /* repeat-x | repeat-y */
}
```

# On peut aussi choisir de repositionner l'image de fond grâce à `background-position`

```
1 section {
2   background-image: url('images/fond.png');
3   background-position: right center;
4 }
```

Ici, l'image sera positionnée dans le bloc à droite, et centrée verticalement

La propriété `background-attachment` définit si la position de l'image d'arrière-plan est fixée dans la zone d'affichage ou si celle-ci défile de façon normale

```
section {
  background-image: url('images/fond.png');
  background-attachment: fixed; /* scroll */
}
```

La propriété `background-size` définit la taille des images d'arrière-plan pour l'élément. La taille de l'image peut être contrainte, complètement ou partiellement afin de conserver ses proportions.

```
section {
  background-image: url('images/fond.png');
  background-size: cover; /* contain */
}
```

Pour toutes ces propriétés, on peut utiliser la version condensée :

```
section {
  background: orange url('images/fond.png') center/cover
              no-repeat fixed;
}
```

Il est toutefois déconseillé de l'utiliser avec trop de valeurs, afin d'éviter les difficultés de relecture

Toutes ces propriétés peuvent avoir des subtilités et comporter des déclinaisons de syntaxes

Il n'est pas possible de tout mémoriser, cependant lorsque vous souhaitez en utiliser une, **vous devrez aller vous documenter**

La méthode consiste à 🔗 rechercher sur le site du MDN la propriété en question