

# Vue3

Ibrahim ALAME

1<sup>er</sup> décembre 2024

## 1 Objectifs du chapitre

Ce chapitre va être une introduction approfondie au **HTML**. Nous allons voir toutes les bases du **HTML** dans ce chapitre : l'en-tête, les métadonnées les balises, les hyperliens, la structure des documents **HTML** etc.

Vous apprendrez entre autre à utiliser **Emmet** avec **Visual Studio Code**. Vous pouvez passer rapidement si vous connaissez déjà bien le langage mais un rafraîchissement ne peut pas vous faire de mal !

## 2 Structure d'une page HTML

### 2.1 Création de notre première page HTML

Créez un dossier pour notre cours où vous mettrez tous les projets que nous ferons ensemble : par exemple **dyma-htmcless**.

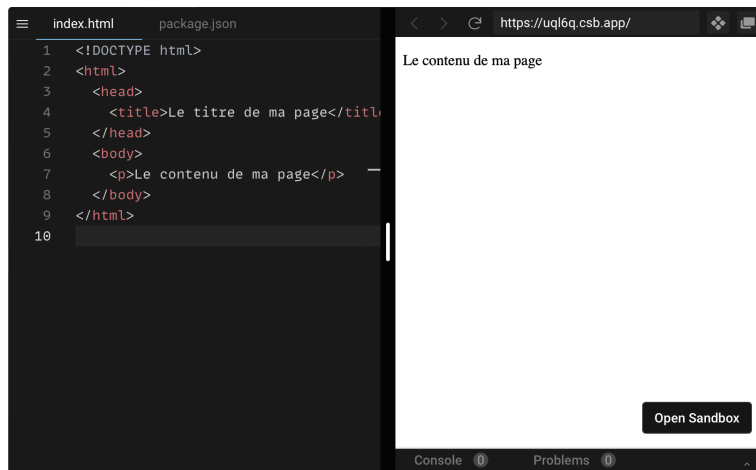
1. Dans ce dossier créez un fichier **index.html**.
2. Ouvrez ensuite le dossier avec l'éditeur **VS Code** : soit en cliquant du droit sur le dossier, soit en ouvrant **VS code** et en allant dans **File** ↗ **Open Folder**.

Dans cette leçon, nous allons étudier la structure d'un document **HTML**.

Une page HTML minimale ressemble à cela :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Le titre de ma page</title>
  </head>
  <body>
    <p>Le contenu de ma page</p>
  </body>
</html>
```

Ce qui donne cela dans un navigateur :



Nous allons utiliser à chaque fois un éditeur en ligne pour que vous puissiez exécuter directement le code. Mais nous vous invitons à le reproduire aussi sur votre ordinateur pour bien apprendre les notations en même temps. Nous allons d’abord voir ce que sont les balises en **HTML** puis nous étudierons la structure du document.

## 2.2 Les éléments **HTML**

Prenons un élément **HTML** par exemple :

```
<title>Le titre de ma page</title>
```

- La première partie, `<title>`, est une balise ouvrante. Il s’agit du nom de l’élément **HTML** ici `title` encadrés par des chevrons. Elle permet de déclarer le début de l’élément **HTML** ainsi que son type.
- La deuxième partie est le contenu, ici `Le titre de ma page`, qui est simplement du texte dans notre cas.
- La troisième partie, `</title>`, est une balise fermante. Elle permet de déclarer la fin de l’élément **HTML**.

## 2.3 L’imbrication

Vous pouvez voir que les éléments **HTML** que nous avons déclarés s’imbriquent les uns dans les autres. Par exemple, `<title>`, est imbriqué dans `<head>`, lui-même imbriqué dans `<html>`. Cette imbrication permet de créer la structure du document en déclarant quels éléments **HTML** sont inclus dans tels élément **HTML**.

## 2.4 La structure d’une page **HTML**

Nous allons maintenant étudier ligne par ligne la structure du document.

### 2.4.1 Le type de document

La première ligne d’un fichier **HTML** est la suivante :

```
<!DOCTYPE html>
```

Il signifie au navigateur qu'il s'agit d'un document **HTML** à la version au moins 5. Comme nous l'avons vu, il n'y a plus aujourd'hui de version **HTML**, il existe uniquement un **HTML Living Standard** qui évolue constamment. Cette ligne, et l'extension **.html** du fichier, permettent également aux éditeurs de pouvoir parser le document pour mettre les couleurs et activer l'autocomplétion **HTML**.

#### 2.4.2 L'élément racine **html**

La deuxième ligne est une balise ouvrante **<html>**. L'élément **<html>** contient tout le code de la page et est appelé élément racine.

#### 2.4.3 L'en-tête **head**

L'élément **HTML head** est l'en-tête du document. Il ne peut y en avoir qu'un par page. Il contient les éléments que vous voulez inclure dans la page **HTML** sans qu'ils soient affichés aux utilisateurs. Il peut contenir de nombreux éléments que nous verrons au fur et à mesure. Il contient notamment le titre de la page.

#### 2.4.4 Le titre de la page **title**

L'élément **title** permet de déclarer le titre de la page dans l'en-tête **head**. Il s'affiche dans l'onglet du navigateur. Il est également utilisé comme description de la page pour l'ajout en favori dans le navigateur.

#### 2.4.5 le corps **body**

L'élément **body** contient tout le contenu à afficher aux utilisateurs de votre page. Il contiendra le texte, les éventuelles images, vidéos, musiques etc.

### 2.5 Utilisation des extensions **Prettier** et **Emmet**

Nous allons commencer à utiliser deux extensions dans **VS Code**.

#### 2.5.1 **Emmet**

Cette extension est incluse dans **VS code** par défaut. Elle permet d'utiliser des raccourcis extrêmement utiles pour la génération de code **HTML** et **CSS**.

Nous allons l'utiliser tout le temps dans la formation et vous expliquerons à chaque fois comment gagner du temps en vous donnant les syntaxes appropriées.

#### 2.5.2 **Prettier**

**Prettier** permet de mettre en forme le code automatiquement lorsque vous le sauvegardez. C'est une extension très utile et nous vous invitons à l'installer dans **VS code** ;

1. Allez dans **Extensions** puis recherchez **Prettier**. Ensuite cliquez sur **Install**. **Prettier** a plus de 14 millions de téléchargement sur **VS code** ce qui en fait l'une des extensions les plus utilisées.
2. Dans **VS Code** allez ensuite dans **File** puis **Preferences** puis **Settings**. Recherchez **format on save** et cochez la case.

3. Toujours dans **VS Code**, et toujours dans **Settings**. Recherchez cette fois **Default Formatter** et dans la liste déroulante sélectionnez **Prettier - Code Formatter esbenp.prettier-vscode**. **Prettier** fonctionnera maintenant à chaque fois que vous sauvegardez !

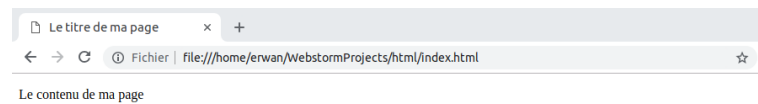
## 2.6 Affichage d'une page HTML

### 2.6.1 Ouvrir un document **HTML** local dans le navigateur

Vous pouvez afficher un document **HTML** à l'aide de n'importe quel navigateur.

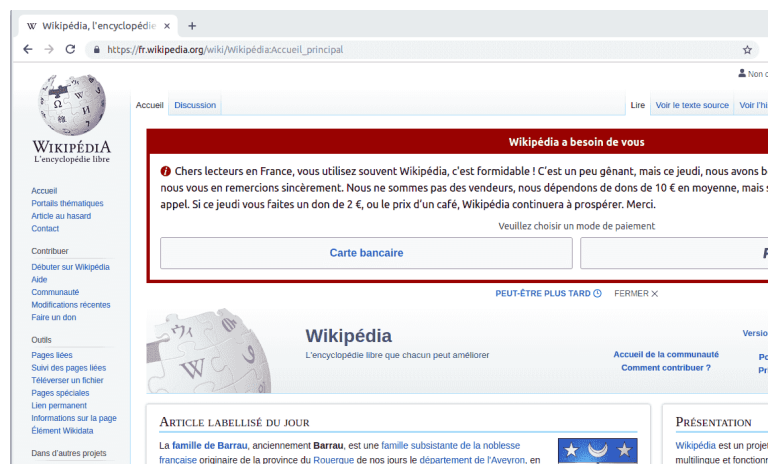
- Bien qu'il existe plus de 100 navigateurs différents, les plus performants sont actuellement probablement Chrome, Firefox et Brave.
- Dans la formation nous utiliserons Chrome, mais Firefox et Brave sont également très biens.
- Pour ouvrir un fichier **.html** avec un navigateur, cliquez droit et faites ouvrir avec **Chrome** par exemple. Ou ouvrir avec une autre application et sélectionnez un navigateur.

Vous aurez ensuite :



### 2.6.2 Ouvrir un document **HTML** depuis le **Web**

Lorsque vous ouvrez une page **HTML** en utilisant le **Web** et **Internet**, votre navigateur va envoyer une requête HTTP à l'adresse demandée, par exemple <https://fr.wikipedia.org/>. Les serveurs **Web** de **Wikipedia** vont alors traiter la requête et retourner une réponse **HTTP** que votre navigateur interprétera pour afficher la page. Par exemple :



### 2.6.3 Ouvrir un document **HTML** local en utilisant un serveur local

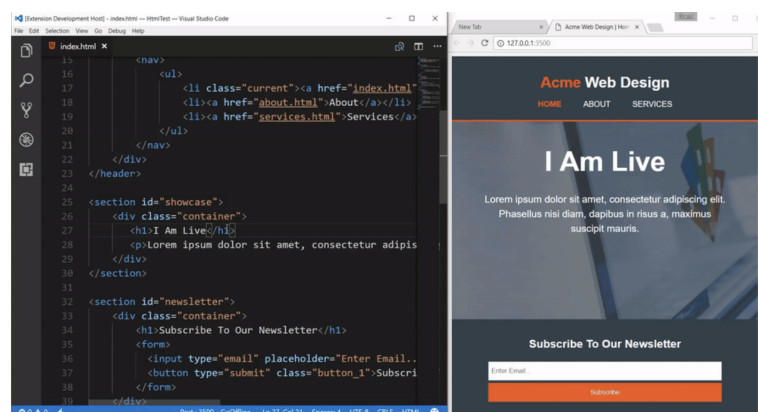
Il existe un inconvénient à ouvrir un fichier **HTML** directement dans votre navigateur : les changements ne sont pas pris en compte. Autrement dit, lorsque vous effectuez des modifications dans **VS code** du fichier **HTML** et que vous sauvegardez (avec **Ctrl + s**), la page affichée dans le navigateur ne se met pas automatiquement à jour. Il faut nécessairement rafraîchir la page dans le navigateur pour voir les changements.

Il existe heureusement une solution : utiliser un serveur **Web** local qui va suivre les changements effectués dans votre fichier et mettre automatiquement la page à jour. Pour cela, allez dans **VS code**, puis dans **Extensions** puis recherchez **Liver server** et installez la.

Cette extension permet de lancer un serveur sur **127.0.0.1** qui est une adresse **IP** spéciale faisant référence à votre ordinateur. La requête n'ira ainsi pas sur **Internet** et vous n'avez pas besoin de connaître l'adresse **IP** réelle de votre ordinateur. Cette adresse **IP** a un alias, c'est-à-dire un nom de domaine qui y équivaut, **localhost**. Tapez **localhost** ou **127.0.0.1** est donc équivalent.

**Live server** permet de lancer un serveur de développement local qui a permet automatiquement de mettre à jour la page en cas de changement dans le code (c'est ce qu'on appelle le **live reload**). Pour lancer une page **HTML** avec l'extension vous pouvez soit cliquer droit dans le fichier dans l'éditeur et faire **Open with Live Server**. Vous pouvez également cliquer sur **Go Live** en bas à droite. Vous pouvez enfin utiliser le raccourci : **Alt + L** puis **Alt + O**.

Voici un exemple :



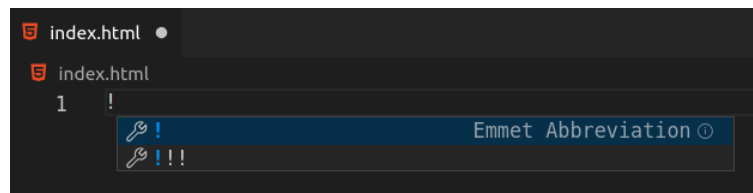
## 3 Les attributs HTML et les métadonnées

### 3.1 Génération d'un document HTML avec Emmet

1. Nous allons commencer par générer un document HTML avec Emmet.
2. Supprimez ce que vous aviez dans index.html.
3. Entrez ensuite :

!

Vous aurez une autocomplétion :



Vous pouvez alors presser la touche entrée et obtiendrez le document suivant :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Document</title>
  </head>
  <body></body>
</html>
```

- Si vous n'avez pas de suggestion qui apparaît, vous pouvez presser les touches **Ctrl + Espace**.
- Si ce raccourci ne se déclenche pas sur un Tab allez dans Fichier ↗ Préférences ↗ Paramètres et recherchez **Emmet**. Cochez la case **Emmet : Trigger Expansion On Tab**.

### 3.2 Les attributs

Les éléments HTML peuvent avoir des attributs. Un attribut est par exemple **lang** :

```
<html lang="fr">
```

Les attributs contiennent des informations supplémentaires sur l'élément HTML sans qu'elles n'apparaissent directement sur la page. Pour créer un attribut il y a trois règles :

1. Il faut mettre un espace entre le nom de l'élément HTML dans la balise ouvrante, et le nom de l'attribut, ou entre l'attribut précédent si il y a plusieurs attributs.
2. Il faut donner un nom à l'attribut et ajouter le signe égal.

3. Il faut donner une valeur à l'attribut. Il faut la placer à l'intérieur de guillemets, qui peuvent être simples ou doubles dès lors que la valeur contient un des caractères suivants " ' ' = < >. Cependant, vous trouverez quasiment systématiquement des guillemets autour de la valeur et nous vous invitons donc à toujours en mettre.

Parfois, il est possible d'utiliser une notation raccourcie avec les attributs booléens :

```
<input type="text" disabled>
```

Ces attributs ne peuvent avoir qu'une seule valeur qui est la plupart du temps le nom de l'attribut.

### 3.2.1 L'attribut **lang**

Dans le document généré par **Emmet**, nous utilisons donc l'attribut **lang** sur l'élément **html**. Cet attribut permet de définir la langue principale du document. Par défaut, il est défini à **en** pour **english** mais mettez **fr** pour une page en français. La langue est utilisée pour l'indexation par les moteurs de recherche.

## 3.3 Les métadonnées

Les métadonnées sont des données apportant des informations sur d'autres données. Le langage **HTML** permet d'utiliser des métadonnées dans un document en utilisant la balise **meta**.

### 3.3.1 Le **charset**

La métadonnée permet de définir l'encodage des caractères du document. C'est le jeu de caractères à utiliser pour convertir les valeurs numériques en caractères. Ici, nous utilisons **UTF-8** qui permet d'encoder n'importe quel caractère (latin, japonais, chinois, caractères spéciaux). C'est l'encodage le plus utilisé sur le **Web**, et plus de 95

### 3.3.2 Le **viewport**

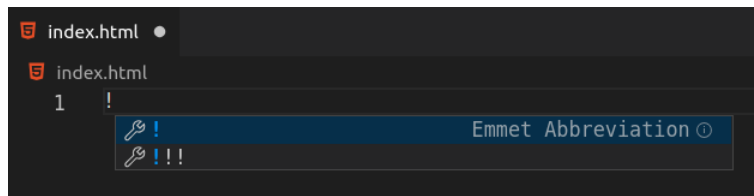
Le **viewport** est la surface de la fenêtre du navigateur. Nous allons étudier ce que veut dire cette balise **meta** :

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

Elle est utile pour l'affichage sur les mobiles et les tablettes. Pour bien comprendre il faut déjà voir les deux surfaces existantes pour les appareils mobiles et tablettes.

- La première est la surface physique qui est le nombre de pixels sur l'écran, qui est également appelée la définition de l'écran.
- La deuxième est la surface utilisable qui est le nombre de pixels virtuels que le terminal pense pouvoir afficher.

Ces surfaces ne sont pas égales sur les mobiles pour des raisons d'affichage. La taille du viewport d'un mobile n'est ni égale à sa surface physique, ni égale à sa surface virtuelle, et ce pour pouvoir afficher la plupart des pages Web. Elle est supérieure afin de pouvoir appliquer un dézoom de la page pour l'afficher en entier. Sans la balise viewport, le navigateur mobile réagit comme sur un ordinateur et met la page à l'échelle de l'écran, ce qui rend impossible la lecture du contenu. Sans la balise nous aurions l'affichage de gauche, et avec la balise l'affichage de droite :



La balise **viewport** permet donc d'indiquer au navigateur mobile comment ajuster les dimensions et l'échelle de la page à la largeur de l'appareil.

- **width="device-width"** permet de définir la largeur de la fenêtre du viewport à celle de l'appareil.
- **initial-scale="1.0"** permet de définir le niveau de zoom initial.

Autrement dit, ces deux paramètres permettent de forcer l'appareil mobile à ne pas changer le zoom ou prétendre que la largeur disponible est plus importante que réellement pour tenter d'afficher l'intégralité de la page. Il nous permettra plus tard dans la formation d'adapter notre page et nos styles en fonction de la largeur réelle disponible et donc de créer un site **responsive** comme sur l'image à droite.

### 3.4 La **meta X-UA-Compatible**

Ce tag permet d'empêcher le mode compatibilité sur **Internet Explorer** et de forcer le navigateur à utiliser le dernier moteur de rendu disponible (dernière version du moteur **Edge**). Ce **tag** n'est pas utile si vous ne souhaitez pas supporter Internet Explorer qui représente moins de 1% du marché des navigateurs.

#### 3.4.1 La **meta description**

La **meta** description sert à l'indexation de cette page par les moteurs de recherche et les annuaires. Elle doit contenir une description du contenu de la page en une ou deux phrases. Depuis 2017, Google affiche entre 260 et 300 caractères. Il faut donc adapter la description en conséquence. Elle s'utilise de cette manière :

```
<meta name="description" content="La description de la page." />
```

## 4 Les commentaires, et les principaux éléments HTML

### 4.0.1 Mettre en commentaire

### 4.0.2 Les titres

Chaque titre doit être contenu dans un élément titre.

Il y a 6 éléments pour les titres, suivant leur importance : **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>**, et **<h6>** .

- **<h1>** représente le titre principal de la page et il en faut normalement qu'un seul par page.
- **<h2>** représente des titres importants **<h3>** moins importants
- etc.

Par défaut les navigateurs vont adapter la taille de la police pour chaque niveau de titre, avec **h1** ayant la plus grande taille et ainsi de suite.



### 4.0.3 Les paragraphes

Les paragraphes sont contenus dans un élément `<p>` :

```
<p>
Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloremque
mollitia necessitatibus minus, illum sunt ducimus ipsam unde, perferendis
nisi dicta ullam nemo maxime impedit obcaecati quibusdam ab similique
minima eos.
</p>
```

## 4.1 Les éléments sémantiques

En **HTML** il existe plusieurs éléments de sémantique pour nous permettre de marquer l'importance d'un contenu textuel.

Ils sont très importants pour les personnes malvoyantes utilisant des lecteurs d'écrans. En effet, ceux-ci prennent en compte les éléments sémantiques lors de la lecture automatique.

### 4.1.1 L'emphase **em**

Dans le langage parlé nous utilisons un accent sur certains mots pour accentuer leur portée, cela s'appelle l'emphase.

Les navigateurs utilisent l'italique par défaut, mais il ne faut pas utiliser cette balise pour mettre en italique.

Il faut utiliser la balise `<i>` pour *italic* .

Voici un exemple :

```
<p>Je suis une <em>emphase</em> car je suis important.</p>
```

### 4.1.2 Très important **strong**

A l'oral, pour souligner des mots très importants, nous les accentuons.

En **HTML**, il faut utiliser l'élément `<strong>` comme balise pour donner une grande importance à un mot ou un groupe de mot.

Par défaut, les navigateurs mettent le texte marqué en gras mais il est recommandé d'utiliser du **CSS** pour mettre le mot en gras et ne pas compter sur ce comportement par défaut.

## 4.2 Sauts de ligne avec **br**

L'élément `<br />` crée un saut de ligne dans un paragraphe.

```
<p>J'écris cette ligne et je veux aller à la ligne<br>
J'écris sur la deuxième ligne et je veux aller à la ligne<br>
J'écris sur la troisième ligne et je veux aller à la ligne<br></p>
```

Sans éléments `<br />` , le paragraphe est rendu sur une seule longue ligne.

### 4.3 Ligne horizontale **hr**

L'élément `<hr />` crée un trait horizontal dans le document marquant un changement thématique dans le texte.

```
<p>Premier sujet très intéressant</p>
<hr>
<p>Deuxième sujet passionnant</p>
```

## 5 Liens hypertextes et images

### 5.1 Les hyperliens

Les hyperliens sont un des trois piliers du **Web** . C'est ce qui permet de naviguer entre différents sites.

Ils permettent de lier tout document **HTML** à un ou plusieurs documents **HTML** du même site ou non.

#### 5.1.1 La balise **a**

Un hyperlien se déclare avec `<a>` (pour ancre ou **anchor** ).

Il permet de déclarer un hyperlien vers un autre endroit de la même page **HTML** ou vers une autre page Web .

#### 5.1.2 L'attribut **href**

Il faut obligatoirement fournir l'attribut **href** (**H**ypertext **R**eference) qui indique la cible du lien sous la forme d'une **URL** (**U**niform **R**esource **L**ocator) :

```
<a href="https://www.esiee.fr/">Un lien vers l'Esiee</a>
```

Une **URL** est simplement une chaîne de caractères qui définit l'emplacement d'une ressource sur le **Web** : image, page **HTML** , fichier à télécharger etc.

Nous verrons plus en détails les valeurs de href mais sachez que par exemple :

```
<a href="#">Lien vers le haut</a>
```

Ce lien permet de remonter en haut de la page. En effet, **#** permet de désigner le document **HTML** courant.

#### 5.1.3 L'attribut **title**

Vous pouvez ajouter un attribut **title** pour préciser un titre qui s'affichera lorsque vous placerez la souris sur le lien ( **hover** ).

```
<a href="https://www.esiee.fr/" title="Apprenez les meilleures technologies !"> Un lien vers l'
```

#### 5.1.4 L'attribut **target**

Vous pouvez spécifier si le lien doit être ouvert dans un nouvel onglet avec **target** en utilisant la valeur **\_blank** :

```
<a href="https://www.esiee.fr/" target="_blank" rel="noopener"> Un lien vers l'Esiee </a>
```

Cela ouvrira un nouvel onglet, mais exécutera la page dans le même processus que le premier onglet source.

Si le nouvel onglet consomme de nombreuses ressources, les performances du premier onglet seront diminuées.

Nous allons voir immédiatement la solution.

#### 5.1.5 L'attribut **rel**

L'attribut **rel (relation)** permet de spécifier la relation entre la cible du lien et la page sur laquelle se trouve le lien.

Nous allons pas voir tous les détails qui ne sont pas intéressants mais seulement la valeur **noopener** .

Cette valeur permet d'indiquer au navigateur d'ouvrir le lien sans que le nouveau contexte de navigation créé ait accès au document qui contenait le lien.

Ce type est utile lorsqu'on ouvre un lien pour lequel on ne veut pas qu'il puisse interagir avec le document source. Autrement dit, il est recommandé de toujours mettre **rel="noopener"** pour les hyperliens vers d'autres sites.

En supposant que nous n'étions pas sur Dyma nous ferions :

```
<a href="https://www.esiee.fr/" target="_blank" rel="noopener"> Un lien vers l'Esiee </a>
```

## 5.2 Les images

Les images se déclarent avec la balise **<img />** .

Nous allons voir les deux attributs importants pour le moment.

#### 5.2.1 L'attribut **src**

L'attribut **src (source)** est obligatoire et contient l' **URL** de l'image qu'on souhaite afficher.

#### 5.2.2 L'attribut **alt**

L'attribut **alt (alternative)** n'est pas obligatoire mais fortement recommandé et contient une description textuelle de l'image.

Il sert de description pour les malvoyants et également lorsque l'image ne peut pas être chargée.

## 6 Listes et tableaux

Temps de lecture : 4 minutes

## 6.1 les tableaux

Un tableau est un ensemble structuré de données présentées en lignes et colonnes.

Un tableau **HTML** permet simplement d'établir la structure du tableau : lignes et colonnes. Mais sans CSS un tableau a un aspect visuel peu engageant.

La plupart du temps vous ne créez pas directement des tableaux en **HTML** mais utiliserez des bibliothèques. Il faut en effet beaucoup de style pour que l'aspect d'un tableau soit joli.

Nous n'allons donc pas trop nous attarder sur les tableaux pour le moment.

Voici un tableau minimaliste :

```
<table>
  <tr>
    <td>Rangée 1 Cellule 1</td>
    <td>Rangée 1 Cellule 2</td>
  </tr>
  <tr>
    <td>Rangée 2 Cellule 1</td>
    <td>Rangée 2 Cellule 2</td>
  </tr>
</table>
```

### 6.1.1 La balise **<table>**

L'élément **HTML <table>** permet de déclarer un tableau de données à deux dimensions.

### 6.1.2 La balise **<tr>**

L'élément **<tr>** (**table row**) permet de déclarer une rangée.

### 6.1.3 La balise **<td>**

L'élément **<td>** (**table data**) permet de déclarer une cellule du tableau.

### 6.1.4 La balise **<th>**

L'élément **<th>** (**table header**) permet de déclarer un entête.

### 6.1.5 La balise **<caption>**

L'élément **<caption>** permet de donner un titre au tableau.

## 6.2 Les listes

Il existe deux types de liste en **HTML** , les listes ordonnées et les listes non ordonnées.

### 6.2.1 Les listes ordonnées

Les listes ordonnées se déclarent avec la balise `<ol>` (ordered list) .

Les éléments de la liste sont ensuite déclarés avec `<li>` (list item) .

```
<ol>
  <li>Un</li>
  <li>deux</li>
  <li>trois</li>
  <li>quatre</li>
  <li>cinq</li>
  <li>six</li>
</ol>
```

Il est possible de commencer la liste à un autre nombre que 1 en utilisant l'attribut `start` :

```
<ol start="3">
  <li>trois</li>
  <li>quatre</li>
  <li>cinq</li>
  <li>six</li>
</ol>
```

Il est possible de commencer par la fin en inversant la numérotation avec l'attribut `reversed` .

Il est possible de changer le type de la numérotation en utilisant l'attribut `type` qui peut prendre comme valeur `a`, `A`, `i` ou `I` . Il est possible de changer la valeur de la numérotation d'un élément avec l'attribut `value` utilisé sur un list item .

Nous utiliserons tous ces attributs dans l'exemple à la fin de la leçon.

### 6.2.2 Les listes non-ordonnées

Les listes non-ordonnées n'ont pas de numérotation.

Les listes non-ordonnées se déclarent avec la balise `<ul>` (unordered list) . Les éléments de la liste sont également déclarés avec `<li>` (list item) .

```
<ul>
  <li>tomate</li>
  <li>concombre</li>
  <li>céleri</li>
  <li>betterave</li>
  <li>carotte</li>
  <li>chou</li>
</ul>
```

### 6.2.3 Un raccourci Emmet pour les listes

Pour imbriquer un élément dans un autre il faut utiliser `>`.

Par exemple :

```
ul>li
```

Pour multiplier un élément un certain nombre de fois il faut utiliser `*`.

Par exemple :

```
ul>li*6
```

## 7 Les formulaires

Les formulaires sont extrêmement importants et nous les approfondirons plusieurs fois : dans la partie CSS , dans les projets et dans le cours [JavaScript](#) .

Dans cette leçon, nous allons voir uniquement la partie [HTML](#) du formulaire qui est donc sa structure.

Les formulaires [HTML](#) sont la principale source d'interaction d'un utilisateur avec un site [Web](#) car ils permettent à l'utilisateur de fournir des données.

Ils sont en effet utilisés principalement pour que les utilisateurs puissent envoyer des données à un site [Web](#) .

Nous verrons dans les formations avancées que le formulaire n'est pas toujours utilisé pour envoyer des données, mais cela reste son utilisation principale.

Un formulaire est composé d'un ou de plusieurs [widgets HTML](#) .

Les [widgets](#) permettent d'entrer des informations par l'intermédiaire de zones de texte, de cases à cocher ( [checkbox](#) ), de boutons radio, de liste sélectionnable ( [select list](#) ) etc.

### 7.1 L'élément `<form>`

L'élément `<form>` permet de déclarer un formulaire qui va contenir un ou plusieurs widgets .

### 7.2 Les attributs `action` et `method`

Ces attributs sont utilisés pour envoyer les données du formulaire au serveur :

```
<form action="/gestion-formulaire" method="post">
</form>
```

- `action` est un attribut permettant de définir l' [URL](#) où doivent être envoyées les données du formulaire. Ici dans l'exemple elles sont envoyées sur le serveur courant.
- `method` est un attribut permettant de définir la méthode [HTTP](#) utilisée pour envoyer les données.

### 7.3 Les attributs communs à tous les [widgets](#)

Avant de voir chacun des [widgets](#) des formulaires [HTML](#) nous allons voir tous les attributs communs qui fonctionnent sur chacun d'entre eux.

- `autofocus`  
Cet attribut permet de sélectionner automatiquement le [widget](#) (lui donner le [focus](#) ) lors du chargement de la page.

- **disabled**  
Cet attribut permet de désactiver le **widget**. L'utilisateur ne peut alors pas interagir avec l'élément.
- **name**  
Cet attribut permet de nommer le **widget**. Le nom sera utilisé pour contenir les données du champ lors de l'envoi du formulaire.
- **value**  
Cet attribut permet de donner une valeur initiale au **widget** (sauf pour **textarea**).

## 7.4 Les champs pour saisir du texte

Il existe plusieurs types de **widgets** pour saisir du texte dans des formulaires : **input** et **textarea**.

### 7.4.1 Les attributs communs aux champs de saisie de texte

1. **readonly** : l'utilisateur ne peut pas modifier la valeur du champ.
2. **required** : indique que le champ est obligatoire avant de soumettre le formulaire (sauf pour certains types de champ **input**).
3. **tabindex** : valeur numérique qui est l'index (l'ordre) du champ dans le formulaire. Cela permettra à l'utilisateur de naviguer avec **tab** entre les champs dans l'ordre que vous souhaitez (par exemple si les champs sont sur plusieurs colonnes).
4. **placeholder** : permet de faire apparaître un texte indicatif dans le champ qui décrit ce qu'il doit contenir. Le texte disparaîtra lorsque l'utilisateur commencera à taper.

## 7.5 Le **widget input**

L'élément **input** permet de créer un champ permettant de saisir un texte sur une ligne. Il s'agit d'un élément vide qui n'a pas besoin de balise fermante :

```
<input type="text" name="prenom" placeholder="Prénom" />
```

Un **input** est généralement du **type text**, mais il existe un très grand nombre de type d' **input**. Nous étudierons plus en profondeur certains types plus loin dans la leçon.

## 7.6 Le **widget textarea**

L'élément **textarea** permet de définir un champ texte sur plusieurs lignes.

Ce champ prend en compte les retours à la ligne entrés par l'utilisateur.

Vous pouvez fixer la largeur visible de la zone de texte (avec l'attribut **cols**), ainsi que la hauteur (avec l'attribut **rows**).

Pour donner une valeur initiale au **textarea**, il suffit de placer du texte entre les balises ouvrante et fermante :

```
<textarea name="longtexte" cols="30" rows="10">Valeur initiale </textarea>
```

## 7.7 Les libellés labels

L'élément `<label>` est une légende pour un champ.

Il peut être associé à un champ avec l'attribut `for` ou en plaçant l'élément du champ à l'intérieur de l'élément `<label>`.

L'utilisateur peut cliquer sur le `label` pour `focus`, c'est-à-dire sélectionner, le champ. Pour associer le `label` à un champ avec `for`, il faut que le champ ait un `id`, à savoir un identifiant unique. Nous approfondirons cette notion plus bas.

## 7.8 Les listes sélectionnables `select`

Nous allons maintenant voir l'élément `select` qui permet de créer une liste déroulante permettant de sélectionner une option parmi une liste d'options.

Les éléments `option` permettent de définir les options sélectionnables dans la liste.

Si l'attribut `value` est défini sur un élément `option`, alors ce sera cette valeur qui est envoyée si l'option est sélectionnée.

Si il n'est pas défini, ce sera le texte contenu entre les balises `<options>`.

L'élément `optgroup` permet de définir un `label` qui est affiché pour un groupe d' `options`. Cet élément n'est pas sélectionnable.

L'attribut `selected` défini sur une `option` permet de définir l' `option` sélectionnée par défaut.

Il est possible d'utiliser l'attribut `multiple` sur `select` afin de rendre plusieurs options sélectionnables en utilisant les touches `ctrl` ou `maj`.

## 7.9 Les éléments à cocher

Nous allons voir deux `widgets` permettant de cocher une ou plusieurs cases : les `checkbox` et les boutons `radio` qui sont des types particuliers de champ `input`.

L'attribut `checked` permet de présélectionner un `input` de type `radio` ou `checkbox`.

### 7.9.1 L'élément `fieldset`

L'élément `HTML <fieldset>` permet de regrouper plusieurs champs en les encadrant. Il permet de définir une `<legend>` décrivant le groupe.

`id` permet de définir un identifiant unique pour un élément `HTML`, nous le verrons souvent car il est très important.

Vous pouvez désactiver tout un groupe en plaçant l'attribut `disabled` sur le `fieldset`.

### 7.9.2 Les `checkbox`

Une case à cocher est un élément `<input />` dont l'attribut `type` a pour valeur `checkbox`.

### 7.9.3 Les bouton `radio`

Un bouton `radio` est un élément `<input>` dont l'attribut `type` a pour valeur `radio`. Il est possible de lier plusieurs boutons `radio` ensemble en utilisant une valeur commune pour l'attribut `name`.

Un seul bouton `radio` peut être coché dans un groupe. Seule la valeur du bouton coché sera envoyée.



## 7.10 Compteurs et barres de progression

Nous allons voir très brièvement deux autres éléments : les `meter` et les `progress` :

### 7.10.1 L'élément `progress`

Une barre de progression, défini avec `<progress>` représente une valeur qui évolue dans le temps jusqu'à une valeur maximale définie par l'attribut `max` .

Cela permet par exemple d'afficher l'avancement d'un téléchargement.

Le contenu entre les deux balises est un affichage alternatif pour les vieux navigateurs (comme pour `alt` pour une image qui ne se charge pas).

### 7.10.2 L'élément `meter`

Un étalon, défini avec `<meter>` , est une valeur fixe dans une plage délimitée par une valeur minimale `min` et une valeur maximale `max`.

L'étalonnage est divisé en trois parties : entre `min` et `low` , entre `low` et `high` et entre `high` et `max` .

`optimum` permet de définir la valeur optimale pour l'étalonnage et défini ainsi la partie préférée, moyenne et moins bonne.

La partie `optimum` peut être basse, par exemple 10, et dans ce cas cela inversera les parties préférées. L'intervalle la plus basse sera le préféré etc.

La partie préférée s'affiche en vert, la partie moyenne en jaune et la partie moins bonne en rouge.

Le contenu entre les deux balises est un affichage alternatif pour les vieux navigateurs.

### 7.10.3 Éléments en ligne et éléments de bloc

## 7.11 Les éléments de bloc

Les éléments de bloc sont la catégorie des éléments `HTML` qui sont séparés par un saut de ligne avant et après l'élément.

Les éléments de bloc consomment tout l'espace disponible sur l'axe horizontal.

Les éléments de bloc que nous avons vu sont : `form` , `h1` à `h6` , `hr` , `li` , `ol` , `p` , `table` et `ul` .

Nous en apprendrons d'autres dans la formation.

## 7.12 Les éléments de ligne

Les éléments de ligne ( `inline` ) occupent l'espace délimité par leurs balises et n'ont pas de saut de ligne ni avant ni après.

Les éléments de ligne que nous avons vu sont : `a` , `br` , `button` , `em` , `i` , `img` , `input` , `label` , `meter` , `progress` , `select` , `strong` et `textarea` .

Nous en apprendrons également d'autres dans la formation.

## 8 Les éléments `div` et `span`

### 8.1 L'élément `div`

L'élément `HTML div (division)` est un conteneur permettant d'organiser du contenu. Il s'agit d'un élément de bloc.

Il permet d’imbriquer un ou plusieurs éléments pour permettre de leur appliquer un style particulier ou encore de leur appliquer un attribut en commun.

En théorie, l’élément `<div>` doit uniquement être utilisé lorsqu’il n’existe aucun autre élément dont la sémantique permet de représenter le contenu.

Nous verrons les éléments sémantiques dans la leçon suivante.

Cependant, en pratique, les éléments sémantiques ne sont pas très utilisés.

## 8.2 L’élément `span`

L’élément **HTML** `span` est un conteneur permettant d’organiser du contenu. Il s’agit d’un élément en ligne ( `inline` ).

Il permet également d’imbriquer un ou plusieurs éléments pour permettre de leur appliquer un style particulier ou encore de leur appliquer un attribut en commun.

La seule différence entre `div` et `span` est que `div` est un élément de bloc, et marque donc une division. A l’opposé, `span` peut s’utiliser dans un paragraphe par exemple, sans ajout de sauts de ligne avant et après.

# 9 Les éléments sémantiques

## 9.1 Qu’est-ce que la sémantique ?

La sémantique est l’étude du sens d’éléments linguistiques.

En programmation la sémantique est le sens d’un groupe de code.

Nous avons déjà vu quelques éléments sémantiques : `h1` , `strong` , `em` par exemple. Par exemple `h1` signifie en sémantique : titre le plus important de la page, c’est-à-dire de niveau 1.

La sémantique est utile pour les algorithmes des moteurs de recherche pour attribuer la dose d’importance à attribuer à chaque contenu. Elle contribue par exemple à répertorier les mots clé d’un site.

Nous allons étudier quelques éléments sémantiques supplémentaires.

## 9.2 L’élément `article`

L’élément **HTML** `<article>` représente du contenu autonome dans un document.

Cela peut être un poste de forum, un article de blog, un commentaire ou un avis utilisateur, ou tout autre élément indépendant.

Il est possible d’imbriquer un élément `article` dans un autre élément `article` . Dans ce cas, l’article imbriqué est relatif à l’élément parent.

Voici un exemple, nous verrons bien sûr tous les autres éléments : <https://codesandbox.io/embed/htmlcss-c1-l10-3522p>

## 9.3 L’élément `header`

L’élément **HTML** `<header>` représente un groupe de contenu introductif ou de contenu aidant à la navigation.

Voici un exemple :

```
<header>
  <h1>Titre principal</h1>
```

```

</header>
```

Remarquez que nous avons utilisé un **header** pour décrire le contenu du l' **article** de premier niveau.

Il peut en effet y avoir plusieurs **headers** par page. Il doit cependant y avoir un seul **header** de premier niveau (c'est-à-dire qui est imbriqué directement dans **body** ).

## 9.4 L'élément **footer**

L'élément **HTML** `<footer>` représente le pied de page de la **section** ou du groupe de contenu.

Les règles sont les mêmes que pour le **header** . Vous pouvez en mettre dans chaque groupe de contenu si nécessaire, mais un seul de premier niveau.

## 9.5 L'élément **section**

L'élément **HTML** `<section>` représente une **section** de contenu d'un document.

Une **section** ne doit pas être utilisée comme un conteneur générique. Pour cela, il faut utiliser `<div>` .

Les questions à se poser pour choisir entre **article** , **div** et **section** :

Le contenu pourrait apparaître dans le plan du document ? (table des matières) Si oui, il faut utiliser **section** .

Le contenu est-il indépendant ? Est-ce qu'il aurait un sens tout seul ? Si oui, utiliser un **article**

Est-ce que j'utilise l'élément seulement pour la structure ou l'application de styles ? Si oui, utiliser **div** . Sinon, trouvez l'élément sémantique approprié.

## 9.6 L'élément **aside**

L'élément **HTML** `<aside>` représente une partie d'un document dont le contenu n'a qu'un rapport indirect avec le contenu principal du document.

Voici un exemple :

```
<article>
  <p>
    Le film de David Fincher <cite>Fight Club</cite> est
    sorti en salles en 1999.
  </p>
  <aside>
    <p>
      Le film a gagné 100 millions de dollars pendant sa sortie initiale.
    </p>
  </aside>
  <p>
    Fight Club est un film américain de David Fincher, sorti en 1999 et adapté du roman éponyme
  </p>
</article>
```

## 9.7 L'élément **main**

L'élément **HTML** `<main>` représente le contenu principal du **body** du document.

L'élément **main** doit être unique dans le document, excluant tout contenu qui est répété sur plusieurs documents. Cet élément est utilisé par le mode lecture des navigateurs.

## 9.8 L'élément **nav**

L'élément **HTML** `<nav>` représente une section d'une page ayant des liens pour naviguer vers d'autres pages ou des fragments de cette page.

Il s'agit d'un élément destiné à la navigation dans un document.

Un document peut avoir plusieurs éléments **nav** : par exemple un pour la navigation sur entre les pages du site et un autre pour la navigation au sein de la page courante.

## 9.9 L'élément **time**

L'élément **HTML** `<time>` permet de représenter une période donnée. Cet élément permet d'utiliser l'attribut **datetime** afin de traduire la date ou l'instant dans un format que les navigateurs peuvent utiliser.

Voici un exemple :

```
<p>Linux est sorti initialement le <time datetime="1991-09-17">17 septembre 1991</time>.</p>
```

Les **datetime** peuvent être utilisés par un navigateur pour la gestion de calendrier et d'agenda.