

HTML : la base d'un site web.

Dans ce cours, je vous propose de découvrir de façon théorique le langage de base utilisé en programmation web : l'HTML.

Nous évoquerons ce qu'est l'HTML, sa syntaxe, la structure d'un document, et plein d'autres choses.

1 - Pour commencer... HTML c'est quoi ?

HTML est le langage qui constitue la base de la plupart des sites internet que vous consultez aujourd'hui. Derrière Wikipedia, Facebook, Youtube, Marmiton ou Google se cache de l'HTML.

Pour preuve, si vous vous rendez sur un des sites évoqués précédemment et que vous affichez le code source de la page (Ctrl+U), ce qui est affiché est de l'HTML.

Attention cependant, HTML n'est pas un langage de programmation!

C'est un langage qui permet de définir la structure d'un document informatique en spécifiant des sections, titres, sous-titres, images, liens, etc.

HTML n'est pas le seul outil utilisé dans le cadre du développement de sites internet. Nous verrons plus tard CSS et JavaScript qui se basent sur l'HTML pour respectivement ajouter de l'esthétisme et effectuer des interactions avancées sur une page web.

Il est également à noter que la technologie HTML est parfois utilisée dans le cadre du développement de logiciels et d'applications mobiles. Les applications Discord, Slack, Twitch & Messenger en sont de très bons exemples.

Dans ce cours je tâcherais de vous guider vers l'autonomie car c'est, je pense le meilleur moyen d'apprendre et de rester à jour. Nous verrons donc ensemble comment lire, comprendre et appliquer la documentation technique HTML fournie par le [Mozilla Developer Network](https://developer.mozilla.org/).

2 - La syntaxe et la structure d'un document HTML.

Il est important d'avoir quelques notions théoriques avant de pouvoir passer à la pratique.

HTML est l'acronyme de la locution anglaise “**H**yper**T**ext **M**arkup **L**anguage” qui en français se traduit littéralement par “Langage de Balisage HyperTexte”. On peut aisément, de par cette traduction, en déduire la syntaxe de l'HTML...

La syntaxe d'un code HTML :

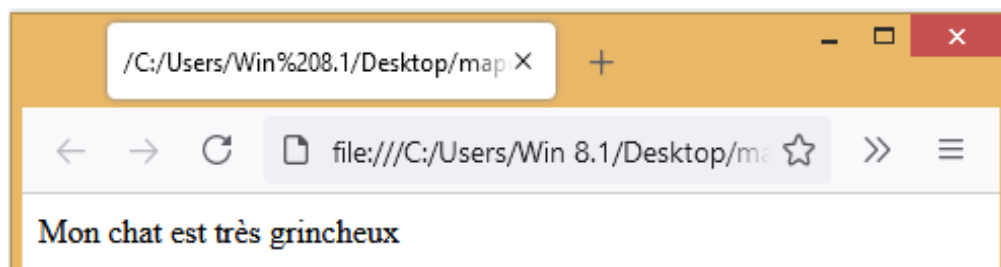
Les balises :

La syntaxe de base d'un code HTML est relativement “simple”.

Elle consiste à envelopper le contenu que l'on souhaite afficher sur une page web entre deux balises tel qu'illustré ci dessous :



Au chargement d'une page HTML, le navigateur web interprète les éléments HTML afin d'uniquement afficher son contenu selon des règles d'affichage préalablement définies :



Il est à noter que les balises vont systématiquement de paire (à l'exception des éléments vides évoqués peu après). L'une est dite ouvrante et l'autre fermante permettant respectivement de définir, dans le cas présent, le début et la fin d'un paragraphe ([accéder à la documentation de la balise p](#)).

Globalement la maîtrise de l'HTML passe par l'apprentissage de l'ensemble des balises référencées [ici](#), sur le Mozilla Developer Network.

Les attributs :

Certains éléments HTML peuvent être dotés d'attributs. Ceux-ci se spécifient toujours sur la balise ouvrante.

L'illustration ci-dessous est un exemple d'usage de l'attribut "[class](#)" qui, dans ce cas précis, a pour valeur "important".

```
<p class="important"> Mon texte </p>
```

⚠ Les attributs ne peuvent pas être interprétés par la technologie HTML. Ils sont uniquement utilisés par les développeurs afin de pouvoir :

- Précisément cibler un ou plusieurs éléments HTML afin de, par exemple, définir puis appliquer des règles d'affichage personnalisées (avec du code CSS).
- Transmettre des données à d'autres langages (tel que JavaScript) qui en feront une interprétation à l'aide d'algorithmes.

⚠ Veillez à attribuer des valeurs pertinentes à vos attributs afin de savoir ce à quoi ils font référence durant la phase de développement et ultérieurement à la livraison de vos projets (dans le cadre d'éventuelles maintenances). Cela fait partie des bonnes pratiques à adopter immédiatement.

Au long de cette formation nous ferons usage de plus en plus d'attributs. De ce fait, n'hésitez pas à aller consulter la documentation du [Mozilla Developer Network](#) si vous ne les connaissez pas ou bien que vous avez un doute sur leurs utilités respectives.

Les éléments vides :

Ce sont des éléments un petit peu particuliers. Ils n'ont pas besoin de texte ni de balise fermante pour être affichés. C'est par exemple le cas avec les illustrations :

```

```

En HTML, l'affichage d'une image s'effectue par l'intermédiaire de la balise [img](#). Dans cet exemple, les attributs src et alt permettent respectivement de spécifier le chemin d'accès à la ressource et de décrire textuellement ce qui est représenté. La balise [img](#) n'est pas fermée car le contenu à afficher se situe au sein même de la balise ouvrante. C'est typiquement ce qui définit un élément vide.

Il existe bien évidemment d'autres éléments vides qui sont référencés dans [cette documentation](#).

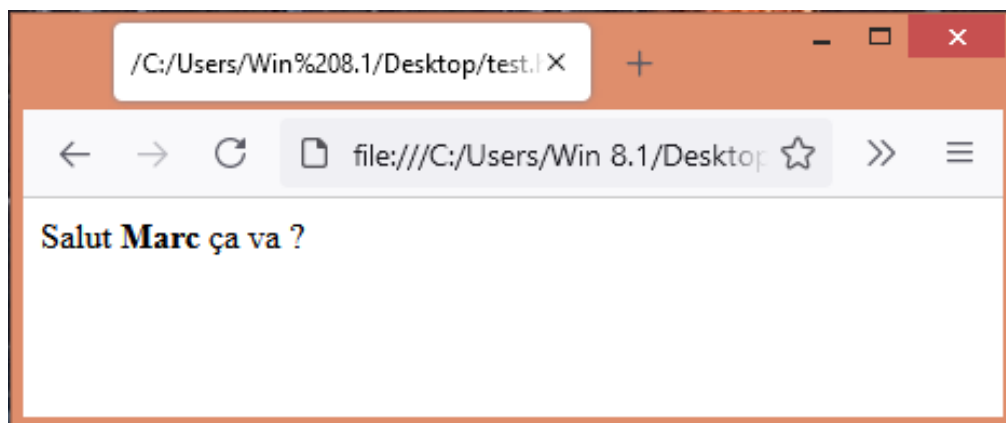
Les éléments imbriqués :

Vous devez ensuite savoir que les éléments HTML peuvent être imbriqués les uns dans les autres (un peu à la manière des poupées russes).

Voici un exemple concret :

```
<p>Salut <strong>Marc</strong> ça va ?</p>
```

Ici, la balise [strong](#) a été ouverte puis fermée au sein d'une balise p ce qui a pour conséquence visuelle d'afficher le prénom "Marc" en gras.



⚠ Il faut être vigilant au placement des balises imbriquées afin de prévenir d'éventuels dysfonctionnements. Ci-dessous figure un mauvais exemple d'enchevêtrement de balises.

```
<p>Salut <strong>Marc</p> ça va ?</strong>
```

Ici, la fermeture de la balise p interfère avec les balises strong ce qui a pour conséquence de causer des problèmes à l'affichage sur le navigateur. Avec la pratique c'est une erreur que l'on fait peu souvent au final.

La structure d'une page HTML :

Nous avons vu les règles syntaxiques de base de l'HTML pour afficher du texte et une image sur une page web. Nous allons désormais étudier la structure générique d'une page HTML afin de pouvoir développer de réels sites internet et applications web à vocation professionnelle.

Un document HTML est systématiquement composé des éléments suivants :

- une ligne précisant qu'il s'agit d'un document de type HTML en version 5,
- un élément html qui entoure l'ensemble de la page.
- un en-tête (élément head),
- un corps (élément body)

Exemple de structure de page HTML basique :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ma page</title>
  </head>
  <body>
    <p>Salut les gens</p>
  </body>
</html>
```

⚠ Les éléments imbriqués présentés sur la capture d'écran précédente, respectent un principe indispensable en programmation : l'indentation.

Indenter c'est le fait de décaler la saisie à chaque nouvelle sous-imbrication afin de retrouver plus aisément de façon visuelle les balises. Cela fait partie des bonnes pratiques à adopter immédiatement.

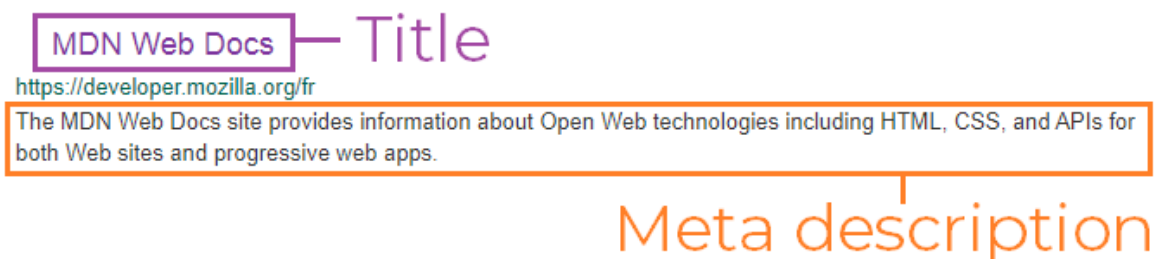
L'élément head :

L'en-tête se déclare systématiquement en début de document à l'aide des balises [head](#).

Les éléments contenus dans l'en-tête n'ont pas vocation à être affichés sur la page. Ils ont pour utilité de fournir au navigateur un ensemble d'informations au sujet de la page HTML à charger ainsi que des chemins d'accès à certaines ressources nécessaires pour l'affichage stylisé (via CSS) ou de tierces interactions (avec Javascript).

Dans l'exemple présenté précédemment, l'entête contient les éléments imbriqués [meta](#) et [title](#) qui ont respectivement pour utilité d'informer le navigateur du jeu de caractères ([charset](#)) utilisé lors de l'[encodage](#) de la page ainsi que de donner un titre à cette dernière (précisément pour les moteurs de recherche & onglets).

⚠ Une bonne pratique supplémentaire est de faire usage de la balise meta suivie de son attribut "description" afin de favoriser son indexation par les moteurs de recherche mais aussi dans le but de renseigner les utilisateurs d'internet sur les contenus qu'ils seront amenés à consulter sur la page web.



Au long de cette formation nous ferons davantage usage d'éléments imbriqués au sein de la balise head. De ce fait, n'hésitez pas à aller consulter la documentation du [Mozilla Developer Network](#) si vous ne les connaissez pas ou bien que avez un doute sur leurs utilités respectives.

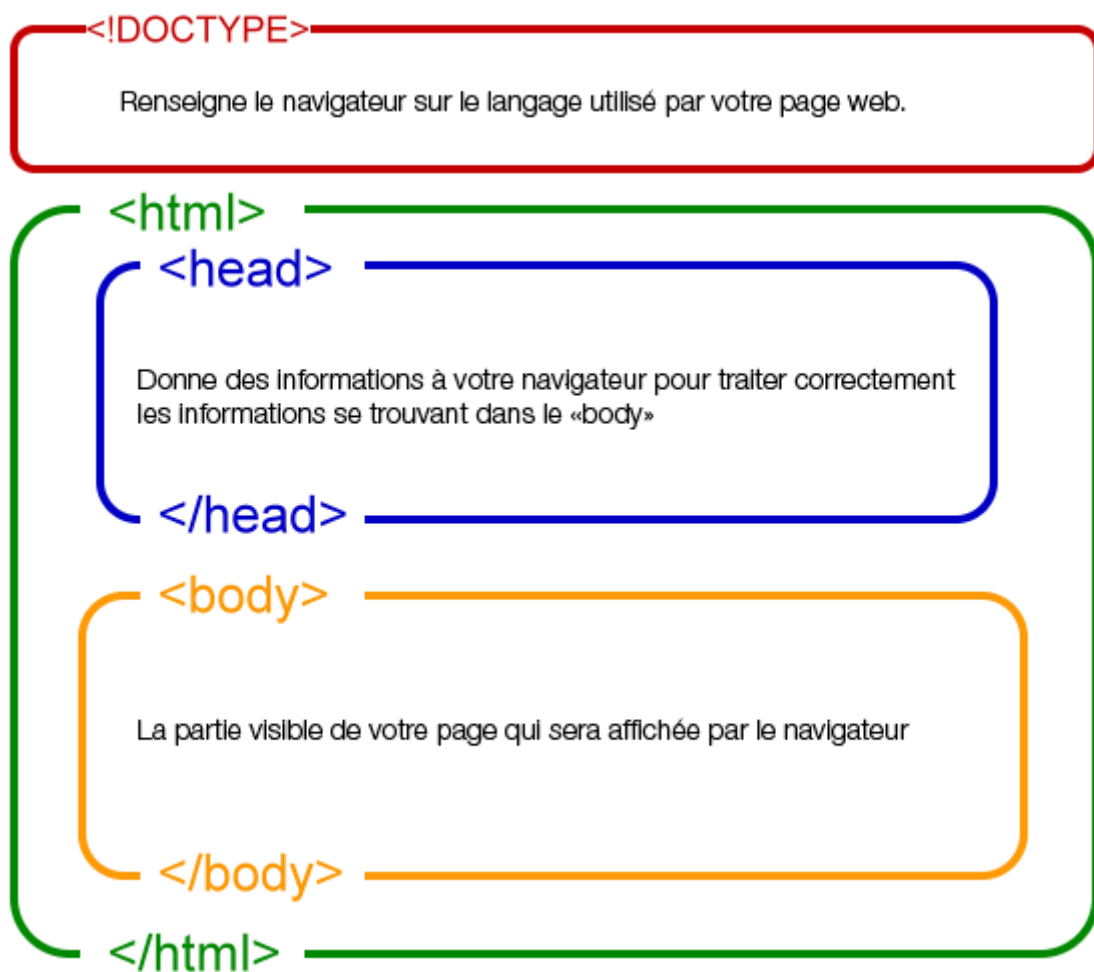
L'élément body (contenus textuels):

Il s'agit du corps (body en anglais) de la page.

C'est au sein de cet élément que, sont à énoncer, les contenus à afficher sur la page. Il se déclare avec la balise `body` et ce, systématiquement après la fermeture de l'élément head.

⚠ Une bonne pratique supplémentaire est de considérer qu'il ne peut y avoir qu'un seul élément body par document HTML.

En résumé :



3 - Les principaux éléments HTML composant le body.

Après avoir étudié la syntaxe et la structure basique d'un document HTML, nous allons maintenant découvrir tout un tas d'éléments qui servent principalement à afficher des contenus sur une page web.

Par nature, ces derniers sont à déclarer de façon systématique au sein du body et ils sont à connaître sur le bout des doigts.

Les éléments textuels :

Il s'agit des éléments qui permettent l'affichage de contenus de type texte (titres, paragraphes, liens, etc.) sur une page web.

Généralement après avoir rédigé un long texte sur Word ou Google Docs, on le hiérarchise afin de le rendre davantage accessible à la lecture.

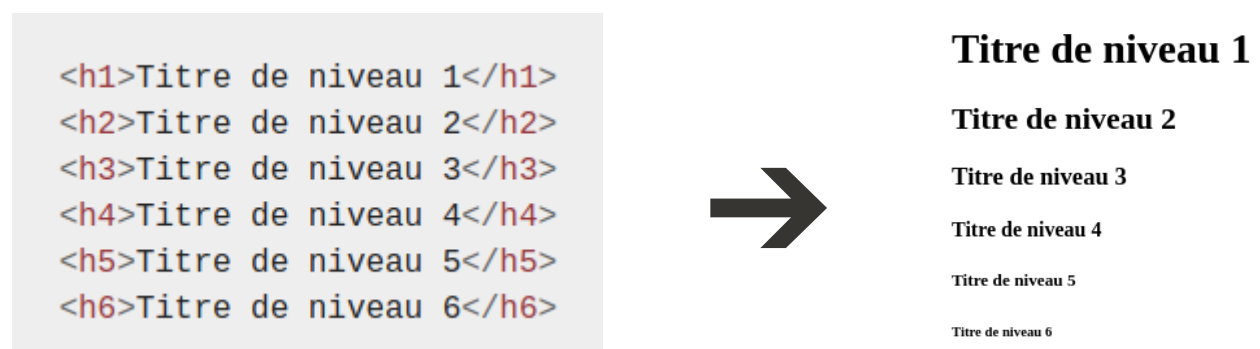
En HTML cela se fait également. Cette étape se nomme formatage et consiste globalement à organiser le contenu de façon logique en définissant des titres, des paragraphes et autres à l'aide de balises.

Les titres :

Les titres se déclarent avec les [balises h](#) suivi d'un numéro compris entre 1 et 6. Plus le titre est important et plus la valeur numérique qui accompagne la balise h est petite.

<h1> : Titre d'une importance capitale (ex : titre de la page).

<h2> à <h6> : Titres de niveau inférieur. (ex: titre de section)



⚠ Il est possible mais fortement déconseillé d'utiliser plusieurs titres de 1er niveau au sein d'une même page HTML sous peine de perdre en accessibilité ainsi qu'en critères de performances pour le référencement naturel (SEO). Attention également à ne pas sauter les niveaux de titres pour les mêmes raisons.

Les paragraphes et sémantique de base du texte en ligne :

Les paragraphes sont les éléments textuels les plus courants. Un paragraphe se déclare avec les [balises p](#) et permet l'affichage de texte sous forme de bloc.

- Le retour à la ligne au sein d'un paragraphe s'effectue à l'aide de [l'élément imbriqué br](#).
- La mise en évidence de certains termes situés au sein de paragraphes peut s'avérer utile pour l'accessibilité et le référencement naturel (SEO). Cette subtilité sémantique s'effectue par l'usage des éléments imbriqués [strong](#) et [em](#).
- Afficher du texte sous forme d'exposant mathématique (ex : 25cm²) s'effectue à l'aide de [l'élément imbriqué sup](#).
- L'élément vide [hr](#) permet d'effectuer une rupture thématique (changement de sujet) au sein d'un paragraphe ou d'une section.
- De nombreux autres éléments sémantiques existent et figurent sur la documentation du [Mozilla Developer Network](#). Soyez curieux!

Les liens hypertextes :

Un lien hypertexte permet d'effectuer une redirection vers une ressource annexe. (le plus souvent une autre page web).

Pour insérer un lien hypertexte au sein d'une page HTML, il suffit de placer du texte entre deux [balises a](#) puis, au sein de la balise ouvrante, d'ajouter l'attribut href. La valeur de cet attribut permet de spécifier l'URL de la ressource à charger lors du clic sur le lien.

```
<a href="https://www.mozilla.com">
  Mozilla
</a>
```



[Mozilla](https://www.mozilla.com)

⚠ Une bonne pratique consiste à privilégier l'usage de [liens relatifs](#) lorsque les ressources à charger se situent sur le même environnement de travail. Cela permet de faciliter la mise en production et les futures phases de maintenance. Les [liens absolus](#) ne sont à utiliser que lorsque les ressources se trouvent sur des machines distantes (sites externes).

L'ouverture de la ressource dans un nouvel onglet se spécifie par la déclaration, au sein de la balise a ouvrante, de l'attribut target ayant pour valeur _blank.

D'autres attributs spécifiques aux liens existent et sont référencés [ici](#).

Les listes :


En HTML, les listes existent sous deux formes distinctes.

- Les listes à puce (dites non ordonnées) qui se déclarent avec les [balises ul](#).
- Les listes ordonnées (incrémentales) qui se déclarent avec les [balises ol](#).

Dans les deux cas, chaque élément de liste se déclare à l'aide de la [balise li](#) et doit obligatoirement être contenu au sein d'un élément parent de type liste comme illustré ci dessous :

* Exemple de liste à puces (ingrédients d'une recette) :


```
<ul>
  <li>1 artichaut</li>
  <li>De l'essuie-tout</li>
  <li>200g de chocolat</li>
</ul>
```



- 1 artichaut
- De l'essuie-tout
- 200g de chocolat

* Exemple de liste ordonnée (étapes d'une recette) :

```
<ol>
  <li>Fee</li>
  <li>Fi</li>
  <li>Fo</li>
  <li>Fum</li>
</ol>
```



1. Fee
2. Fi
3. Fo
4. Fum

⚠ Les listes peuvent également s'imbriquer entre elles sur plusieurs niveaux.
Pour davantage d'informations au sujet des listes, n'hésitez pas à jeter un œil à [la documentation officielle](#).

Les éléments structurels :

Il s'agit d'éléments qui permettent de hiérarchiser les pages web de façon approfondie grâce à différentes balises de subdivision. Celles-ci permettent principalement d'avantage de structurer son travail mais aussi, pour certaines, de gagner en critères d'accessibilité & référencement naturel.

Les éléments structurels primaires :

- Header : élément HTML qui doit être composé de contenus dont les rôles sont à caractère introductif ou de navigation (titre, menu de navigation, court paragraphe d'intro, etc.).

Généralement situé en haut de document, [l'élément header](#) peut contenir autant de contenus HTML que souhaité dès l'instant où l'usage de ces derniers reste structurellement logique. De ce fait, il ne peut pas figurer autre part que dans le body ou au sein des éléments structurels suivants : articles et sections.

- Nav : élément HTML dont le rôle est la [navigation](#) principale entre différentes pages d'un même site web par l'intermédiaire de liens hypertextes.

Il peut figurer au sein du body ainsi que dans l'ensemble des éléments structurels à l'exception de lui-même (par souci logique). Il est toutefois à noter que [les balises nav](#) sont le plus souvent imbriquées dans un header.

- Article : élément HTML qui doit représenter une fonctionnalité indépendante et ré-utilisable (sidebar, widget, article de news, etc.).

Une page HTML peut être composée de plusieurs [éléments articles](#). Cet élément peut lui aussi contenir autant de contenus HTML que souhaité dès l'instant où l'usage de ces derniers reste structurellement logique. C'est par ailleurs pour cette raison que son imbrication avec d'autres éléments de type article est déconseillée.

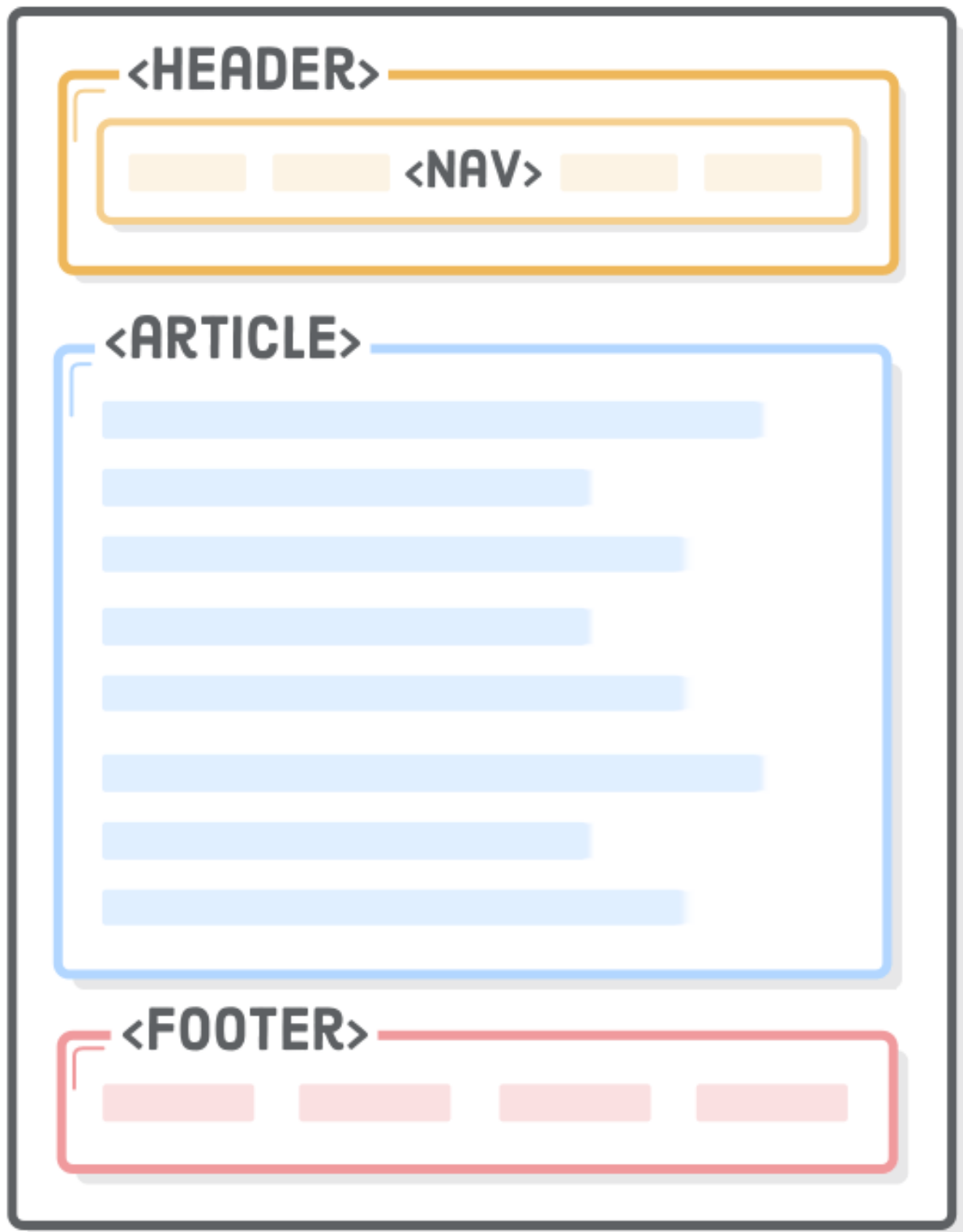
- Footer : élément HTML englobant des contenus de plus faible importance (auteur, copyright, ou documents annexes).

Généralement situé en fin de document ou de section, [l'élément footer](#) a uniquement pour vocation de contenir des éléments textuels.

Logiquement, aucun élément de structuration ne doit y figurer car ce n'est pas au sein d'un pied de page que s'introduisent et se développent des thématiques.

L'ensemble des éléments structurels se déclarent avec les balises associées évoquées dans leurs documentations respectives.

* Exemple de structuration classique d'une page web :



Les éléments structurels génériques :

- Div : élément HTML qui, par défaut, permet de [diviser](#) du contenu sous forme de blocs (systématiquement avec un retour à la ligne). Vous comprendrez l'utilité de cet élément très prochainement avec le cours sur le CSS.



- Span : élément HTML qui, par défaut, permet de [fractionner](#) du contenu sans retour à la ligne (affichage côte à côte). Le span est très utile pour esthétiquement mettre en valeur des mots ou groupes de mots au sein d'un paragraphe (via CSS).



⚠ Ces éléments génériques peuvent être déclarés n'importe où dans le body ainsi qu'au sein de l'ensemble des éléments structurels primaires précédemment étudiés. Il est à noter qu'ils n'ont aucune valeur sémantique c'est à dire qu'ils n'influent en rien sur l'accessibilité ou le référencement. De plus, ils sont imbriquables entre eux à l'infini comme illustré ci-après ce qui est très pratique.

```
<div class="more">
  <div class="gameinfo-container">
    <div class="gameinfo">
      <h3>Beaveroo</h3>
      <p>2D Platformer</p>
    </div>
    <div class="game-desc">
      <p><span>Beaveroo</span> est ...</p>
    </div>
  </div>
</div>
```

Autres éléments utiles :

Il s'agit d'éléments non catégorisable et très différents les uns des autres. Typiquement, les images, les tableaux et les formulaires en font partie.

Les images :

Pour intégrer une image sur un site internet il n'y a rien de plus simple : il suffit de déclarer une [balise vide img](#) au sein d'un document HTML.

Cette balise prends, par défaut, deux attributs :

- src : qui prend pour valeur un lien (absolu, relatif ou externe) qui pointe vers une ressource de type image.
- alt : qui sert à décrire l'image par souci d'accessibilité principalement pour les personnes souffrant de déficiences visuelles ou si la ressource est momentanément indisponible (404 - not found).

⚠ Si l'image est uniquement décorative, cet attribut est tout de même obligatoire. Il suffira simplement de lui affecter une valeur nulle comme illustré ci dessous :

```

```

Il est possible, dans certains cas spécifiques, d'imbriquer des images au sein d'autres éléments HTML tels que des paragraphes et liens hypertextes.

⚠ Enfin, pensez à optimiser et à utiliser des formats d'images adaptés afin de réduire les temps de chargement.

- Pour les photos, privilégiez les fichiers aux formats JPEG ou WEBP (car très léger).
- Pour les logos avec transparence, l'idéal est le SVG (car très léger) bien que le PNG puisse aussi faire l'affaire (bien que beaucoup plus lourd).
- Le format GIF n'est à utiliser que pour les petits contenus simples animés (ex: loader).
- Et surtout : redimensionnez vos images en amont!!!
Il est inutile de charger des images 8K pour les afficher sur un écran FHD.

Les tables :

Avec HTML, il est possible d'afficher des données sous forme de tableau sur une page web.

Cela est utile lorsque l'on souhaite représenter un emploi du temps, une grille salariale, une facture ou tout autre groupement d'informations qui se matérialise sous forme de lignes et de colonnes.

* Exemple de tableau HTML (stylisé avec du CSS):

ID.	Jeu	Prix
1124	Rayman 2	30.00€
2694	Sonic 3	50.00€
3845	Pokémon Rubis	50.00€
Totals		130,00€

L'ajout d'un tableau au sein d'une page web passe tout d'abord par la déclaration de [l'élément table](#) dans un document HTML.

On y imbrique ensuite d'autres éléments les uns dans les autres afin d'organiser les données sous forme de lignes segmentées en cellules tel qu'illustré ci-dessous

```
<table>
L  <tr>
i   <td>ID.</td>
n   <td>Jeu</td>
e   <td>Prix</td>
    </tr>
</table>
```

cellules 1 à 3

⚠ Le nombre de cellules par ligne d'un tableau HTML doit rester le même afin de conserver une cohérence sémantique et visuelle.

Pour certains cas très spécifiques il est tout de même possible d'étendre une cellule sur plusieurs colonnes à l'aide de l'attribut [colspan](#).

* Situation initiale (3 cellules dont une vide) :

```
<tr>
  <td>Totals</td>
  <td></td>
  <td>130,00€</td>
</tr>
```



Totals		130,00€
--------	--	---------

* Exemple colspan sur 2 colonnes :

```
<tr>
  <td colspan="2">Totals</td>
  <td>130,00€</td>
</tr>
```



Totals		130,00€
--------	--	---------

Il est également possible de regrouper une cellule sur plusieurs lignes à l'aide de l'attribut [rowspan](#).

* Situation initiale (articles à un prix similaire) :

```
<tr>
  <td>2694</td>
  <td>Sonic 3 </td>
  <td>50.00€</td>
</tr>
<tr>
  <td>3845</td>
  <td>Pokémon Rubis</td>
  <td>50.00€</td>
</tr>
```



2694	Sonic 3	50.00€
3845	Pokémon Rubis	50.00€

* Exemple rowspan sur 2 lignes (regroupement des prix) :

```
<tr>
  <td>2694</td>
  <td>Sonic 3 </td>
  <td rowspan="2">100.00€</td>
</tr>
<tr>
  <td>3845</td>
  <td>Pokémon Rubis</td>
</tr>
```



2694	Sonic 3	100.00€
3845	Pokémon Rubis	

Mémo associé à l'élément table :

Éléments de base & attributs associés :

<table> : Pour déclarer un nouveau tableau.

<tr> : Pour ajouter une nouvelle ligne dans un tableau.

<td> : Pour ajouter une nouvelle cellule au sein d'une ligne.

colspan : attribut dont la valeur permet de spécifier sur combien de colonnes doit d'étendre la cellule associée.

rowspan : attribut dont la valeur permet de spécifier sur combien de lignes doit d'étendre la cellule associée.

Éléments de structuration sémantique :

<th> : Pour ajouter une nouvelle cellule d'en-tête au sein d'une ligne.

<thead> : élément permettant d'identifier l'en-tête d'un tableau.

<tbody> : élément permettant d'identifier les données contenues au sein d'un tableau.

<tfoot> : élément permettant d'identifier les colonnes résumant un tableau.

Les formulaires :

Les formulaires sont des éléments qui permettent le traitement de données tel que l'envoi d'un email ou le stockage d'informations saisies par un utilisateur sur une page web.

Avec HTML il n'est cependant pas possible de traiter les données issues d'un formulaire. Cela s'effectue par l'intermédiaire de [langages orientés serveur](#) (PHP, Java, Ruby, etc.) que nous étudierons ultérieurement. Présentement, l'objectif est de comprendre comment interfacier un formulaire en suivant les règles sémantiques associées.

Le premier élément qu'il est indispensable de déclarer lorsque l'on souhaite créer un formulaire est [l'élément form](#). Celui-ci permet de spécifier le début et la fin d'un formulaire HTML.

```
<form action="/traitement.php" method="get">
  <!-- Corps du formulaire -->
</form>
```

⚠ Comme illustré ci-dessus, la balise ouvrante doit systématiquement accueillir les attributs :

- action : qui prend pour valeur une URL dont le rôle est la redirection des données vers un fichier de traitement écrit en langage orienté serveur.
- method : qui prend pour valeur GET ou POST selon l'usage.

La description de ces attributs est volontairement floue car dans le cadre de l'apprentissage de l'HTML, il n'est pas utile de les détailler. Ces attributs seront pour l'instant uniquement d'utilité sémantique. La partie fonctionnelle sera étudiée lorsque nous aborderons la partie serveur.

Ensuite, selon la complexité du formulaire, de nombreux sous éléments peuvent être déclarés au sein des balises form :

- Input : élément vide qui permet l'affichage d'un [champ](#) sur une page web. La représentation de ce dernier change selon la valeur de l'attribut "type", déclaré au sein de la balise ouvrante.

Les types de champs que vous rencontrerez le plus souvent sont : [text](#), [email](#), [password](#), [radio](#), [checkbox](#), [number](#), [hidden](#) & [submit](#).

Il en existe bien davantage référencés sur [la documentation officielle](#). Je vous conseille d'ailleurs d'y jeter un œil, on ne sait jamais!

- Textarea : élément HTML qui permet l'affichage d'un [champ de saisie sur plusieurs lignes](#).
- Select : élément HTML qui permet l'affichage d'un [champ de sélection](#) interfacé sous forme de liste déroulante composée d'options.
- Label : élément dont le rôle est l'affichage d'un [libellé](#), généralement situé au-dessus d'un champ de formulaire. Son utilité est principalement sémantique car il indique à l'utilisateur, le rôle du champ associé.

La déclaration de l'attribut "for" dont la valeur est celle de [l'attribut ID](#) du champ associé permet d'activer la saisie ou l'action de ce dernier, lors du clic sur le label.

```
<label for="abcd">Prénom</label>
<input type="text" id="abcd" name="prenom">
```

⚠ Chaque élément imbriqué au sein d'un formulaire (à l'exception des labels, des options et du bouton d'envoi) doit disposer d'un attribut name afin de rendre possible, côté serveur, la récupération et le traitement des informations renseignées. Nous en reparlerons ultérieurement.

⚠ Sur une même page web, il est interdit d'attribuer un même ID à plusieurs éléments.

* Exemple de la structure typique d'un formulaire de contact :

The diagram illustrates a typical contact form structure with the following components and labels:

- Input de type "radio"**: Points to the selection buttons "Mme." and "Mr." at the top left.
- Inputs de type "text"**: Points to the input fields for "Votre nom" and "Voter prénom".
- Input de type "email"**: Points to the input field for "Votre e-mail".
- Textarea**: Points to the large text area for "Message".
- Bouton de type "submit"**: Points to the "Envoyer" button at the bottom.

Ça y est! Vous en connaissez quasiment autant que moi au sujet de l'HTML!

Au long de votre formation ou de votre vie de développeur vous croiserez probablement d'autres balises, attributs et spécificités techniques relatives à l'HTML car l'informatique est un domaine en constante évolution.

En cas de nouveauté, ne paniquez surtout pas! Rendez-vous simplement sur la documentation du [Mozilla Developer Network](#)... Tout y sera détaillé.

Sur ce, je vous laisse mettre en pratique ce que vous avez appris!



4 - Normes et standardisation du code HTML.

Jusqu'à présent, nous avons couvert les bases de l'HTML d'un point de vue technique.

Vous pouvez cependant vous demander d'où proviennent les règles et standards précédemment évoqués, comment est gérée l'évolution de l'HTML ainsi que les moyens qui permettent de s'assurer que le code que vous avez produit est de qualité ?

Standards et évolution du langage:

C'est le [World Wide Web Consortium](#) (W3C) qui définit les standards et gère les évolutions liées au langage HTML.

Fondé en 1994, l'objectif de cet organisme est globalement d'assurer la compatibilité et l'accessibilité des technologies web à l'ensemble des usagers d'internet y compris aux personnes souffrant d'handicap ou atteintes de déficiences (visuelles, auditives, motrices, etc.) et ce, quels que soient leurs dispositifs d'accès (mobile, tablette, etc.) ou leurs conditions d'environnement (niveau sonore, éclairage, etc.).

Le slogan du W3C est d'ailleurs « Un seul web partout et pour tous ».

Contrôle qualité du code écrit :

L'organisme du W3C propose gratuitement de nombreuses ressources sur son site internet.

Parmi ces dernières figure [Validator](#), un outil à destination des développeurs qui souhaitent s'assurer de la qualité de leur code écrit en langages balisés (HTML, XML, SMIL, MathML, etc.).

Après soumission et analyse, cette plateforme retourne un rapport détaillé contenant, généralement, quelques avertissements et erreurs qu'il faut considérer avant le déploiement de l'application. L'objectif final est d'avoir un code exempt de défaut afin de maximiser l'accessibilité et le référencement.

Pour aller plus loin, l'outil [AccessibilityTest](#) (basé sur le [WCAG 2.0](#)) est pratique pour évaluer les performances uniquement liées à l'accessibilité sur le web. Nous reparlerons ultérieurement de cet utilitaire car l'accessibilité d'un site internet dépend de nombreux autres facteurs tels que le choix des typographies, des teintes, des contrastes, de l'adaptabilité sur différents formats d'écrans, etc.