HTML5 – LES BASES

TABLE DES MATIÈRES

Historique des versions

Version	Date	Modification
1.0.0	16/11/21	Création
1.0.1	17/11/21	Ajouts des listes, images, paragraphes, liens, meta, id
1.1.0	22/11/21	Corrections sur les ancres, lien vers un email et un n° de téléphone

NOTE : dans sa version actuelle, ce cours est très fortement inspiré/aspiré du cours « Apprendre à coder en HTML et CSS | Cours complet (2020) » de Pierre Giraud », accessible à l'adresse https://www.pierre-giraud.com/html-css-apprendre-coder-cours.

J'y ai apporté quelques modifications, ajouts et simplifications afin de mieux correspondre à notre contexte d'apprentissage.

Au fur et à mesure des versions, le contenu sera rendu plus spécifique.

Comme tu l'as peut-être lu dans l'introduction au développement web[1], **HTML** est un **langage** qui permet de **définir le contenu d'une page web et sa structure.** Les titres, les paragraphes, les liens, les images, formulaires… visibles sur une page web sont tous définis à l'aide du langage HTML.

Plus précisément, HTML veut dire **HyperText Markup Language**. En français : c'est un **langage de balisage hypertexte.**

HTML est donc un **langage** : il a sa propre **syntaxe** et ses **propres mots-clés**. Ce **langage** est essentiellement **constitué d'éléments** qui permettent de **hiérarchiser**, et **structurer** le contenu d'une page web.

Ça ressemble à ça :

```
<!DOCTYPE html>
<html>
        <head>
                <title>Titre affiché dans la barre de titre</title>
                <meta charset="utf-8">
        </head>
<body>

        <h1>Titre affiché dans la page</h1>
        <h2>Hello world!</h2>
        <p>Ceci est un paragraphe.</p>
        <p>Ceci est un autre paragraphe.</p>

</body>
</html>
```

LES ÉLÉMENTS HTML

Dans une page, nous allons utiliser **les éléments** en HTML **pour marquer du contenu**, c'est-à-dire pour lui **donner du sens** aux yeux des navigateurs et des moteurs de recherche. Selon l'élément utilisé, un navigateur va reconnaître le contenu comme étant de telle ou telle nature.

[1] Je sais, je suis sans doute un peu naïf ? 😊

Ainsi, on va **utiliser des éléments pour définir un paragraphe ou un titre** par exemple, ou encore pour insérer une **image** ou une **vidéo** dans un document.

L'élément `p`, par exemple, sert à définir un **paragraphe**, tandis que l'élément `a` va nous permettre de créer un lien, etc.

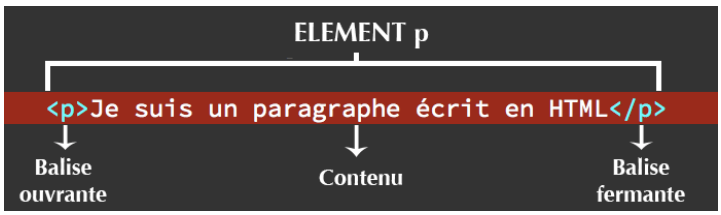
Aujourd'hui, il existe **plus de 120 éléments HTML différents** aux rôles très variés et qui font la richesse de ce langage. Nous allons étudier et nous servir d'une grande partie d'entre eux dans ce cours.

BALISES ET ÉLÉMENTS

Un **élément HTML** peut être soit constitué d'une **paire de balises** (**ouvrante et fermante**) **et d'un contenu**, soit d'une balise unique qu'on dit alors « **orpheline** ».

L'élément `p` (qui sert à définir un paragraphe) est par exemple constitué d'une balise ouvrante, d'une balise fermante et d'un contenu textuel entre les balises. L'idée ici est que le texte contenu entre les deux balises va être le texte considéré par le navigateur comme étant un paragraphe.

Voici comment on va écrire cela :

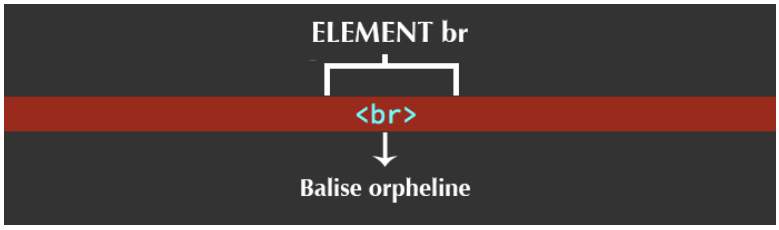


Comme vous pouvez le constater, **la balise ouvrante** de l'élément est constituée d'un **chevron ouvrant** `<`, du **nom de l'élément** en question et d'**un chevron fermant** `>`.

Notez bien ici la **différence** entre la balise ouvrante et la balise fermante de notre élément `p` : **la balise fermante contient un slash avant le nom de l'élément**.

Vous pouvez déjà retenir cette syntaxe qui sera **toujours la même en HTML**.

Certains éléments en HTML ne vont être constitués que d'**une balise** qu'on appelle alors **orpheline**. Cela va être le cas pour certains éléments qui ne possèdent pas de contenu textuel comme l'élément `br` par exemple, qui sert simplement à créer un retour à la ligne en HTML et qui va s'écrire comme ceci :



LES ATTRIBUTS

Finalement, **les éléments vont également pouvoir contenir des attributs** qu'on va alors placer **dans la balise ouvrante** de ceux-ci. Pour certains éléments, les attributs vont être **facultatifs** tandis que pour d'autres ils vont être **obligatoires** pour garantir le bon fonctionnement du code HTML.

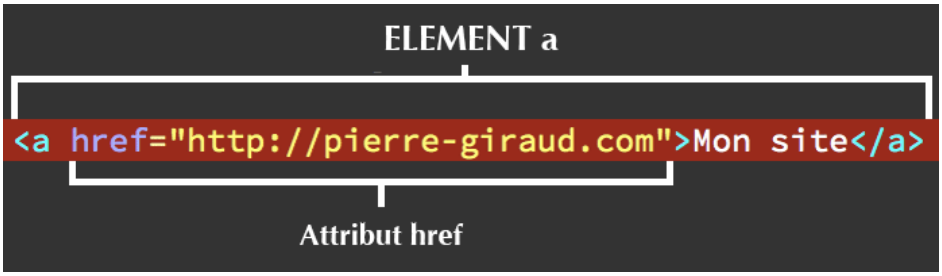
Un attribut doit se trouver **dans la balise ouvrante** d'un élément et a la forme `attribut="valeur"`.

Les attributs vont en effet venir compléter les éléments en les définissant **plus précisément** ou en apportant des informations supplémentaires sur le comportement d'un élément.

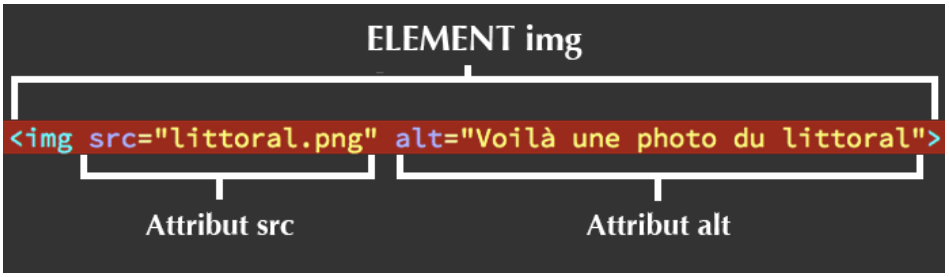
Un attribut contient toujours une valeur, qu'on peut cependant parfois omettre dans le cas des attributs ne possédant qu'une seule valeur (car la valeur est alors considérée comme évidente).

Prenons ici l'exemple de l'élément `a` qui est l'abréviation de « **a**nchor » ou « ancre » en français. Cet élément va principalement nous servir à créer des liens vers d'autres pages.

Pour le faire fonctionner correctement, nous allons devoir lui ajouter un attribut `href` (pour « **h**ypertexte **ref**erence » ou « référence hypertexte » en français) qui va nous permettre de **préciser la cible** du lien, c'est-à-dire la page de destination du lien en lui passant l'adresse de la page en question en valeur.



Voici un autre exemple pour l'élément `img` (pour insérer une image dans une page) :



Chaque élément aura des **attributs** qui lui sont propres et il existe quelques attributs qui sont **communs** à la majorité (voire la totalité) des éléments.

Je vous propose d'en découvrir 2 ci-après.

L'ATTRIBUT « ID »

Tous les éléments html peuvent avoir un attribut « **id** ». Cet attribut permet, comme son nom peut le laisser penser, à **identifier un élément de façon unique** dans la page courante.

L'**id** est **choisi par le concepteur** de la page web. C'est bien sûr mieux de choisir un id qui a du sens par rapport au contenu de l'élément qu'il identifie. Ce sera plus facile à utiliser par la suite.

Techniquement parlant (en théorie donc), la seule **restriction quant au format** d'un id est qu'il ne peut **pas comporter d'espace**. En pratique, il est généralement considéré comme une **très bonne pratique** de suivre les mêmes règles que pour nommer une variable dans la plupart des langages de programmation :

- **Commencer par une lettre**
- Contenir des **lettres** et ou des **chiffres**
- Pas de caractères spéciaux, sauf « _ » (le caractère souligné) et éventuellement un tiret (« - »)
- L'id est **sensible à la casse** (différence entre minuscules et majuscules), c'est-à-dire par exemple que « *Prenom* » et « *prenom* » sont considérés comme 2 id différents ! Et ceci sera valide aussi bien quand référencerez l'élément en JavaScript qu' en CSS !

En plus, en respectant ces bonnes pratiques, vous vous épargnerez bien des prises de têtes quand **vous devrez manipuler ces id en JavaScript et en CSS**. Vous êtes prévenus 😊

En vous basant sur ces recommandations, choisissez votre convention et respectez-là !

L'id est utile dans 3 situations :

- quand il fut pouvoir accéder à l'élément en **JavaScript**
- pour **appliquer un ensemble de règles CSS** à cet élément en particulier
- pour pouvoir **créer des liens/ancres internes** vers cet élément (dans la même page)

Vous n'êtes pas obligé de spécifier l'attribut id d'un élément, mais c'est vraiment conseillé si vous avez besoin de référencer ce dernier pour une des 3 raisons évoquées ci-dessus.

L'ATTRIBUT « CLASS »

Cet attribut sera très utile quand on verra les feuilles de styles (CSS) car il permet de spécifier une ou plusieurs **classes CSS** qui permettront de changer l'apparence de l'élément. On verra ça dans la partie CSS, patience !

LA STRUCTURE MINIMALE D'UNE PAGE HTML

Pour qu'une page HTML soit déclarée **valide**, elle doit obligatoirement comporter certains éléments et **suivre un schéma précis**.

Vos pages HTML devraient toujours être valides pour les raisons évoquées ci-dessus. En effet, une page non valide ne sera pas comprise par le navigateur qui va alors potentiellement mal l'afficher voire dans certains cas ne pas l'afficher du tout.

Voici ci-dessous le code minimum pour créer une page HTML valide. Nous allons dans la suite de ce chapitre expliquer le rôle de chaque élément et attribut.

STRUCTURE MINIMALE D'UNE PAGE HTML VALIDE

Voici le contenu minimal que doit contenir une page web pour être valide.

```
<!DOCTYPE html>
<html>

    <head>
        <title>Titre affiché dans la barre de titre</title>
        <meta charset="utf-8">
    </head>
```

```
        <body>

        </body>

</html>
```

→ A connaître par cœur ←

LE DOCTYPE

Tout d'abord, nous devrons toujours démarrer une page HTML en précisant le **doctype** de notre document. Comme son nom l'indique, le `doctype` **sert à indiquer le type du document**.

Faites bien attention à l'écriture du `doctype` : vous pouvez remarquer que la balise représentant le `doctype` commence par **un point d'exclamation. Ceci est un cas unique.**

Dans la balise de l'élément `doctype`, on va préciser le langage utilisé, à savoir le **HTML** dans notre cas.

L'ÉLÉMENT HTML

Après avoir renseigné le type du document, nous devons utiliser un élément **`html`**. Cet élément est composé d'une paire de balises ouvrante **`<html>`** et fermante **`</html>`**.

L'élément **`html`** va représenter notre **page** en soi. On va insérer tout le contenu de notre page (et donc les autres éléments) à l'intérieur de celui-ci.

À l'intérieur de l'élément html, nous devons à nouveau **indiquer obligatoirement** deux éléments qui sont les éléments **`head`** et **`body`** et qui vont avoir des rôles très différents.

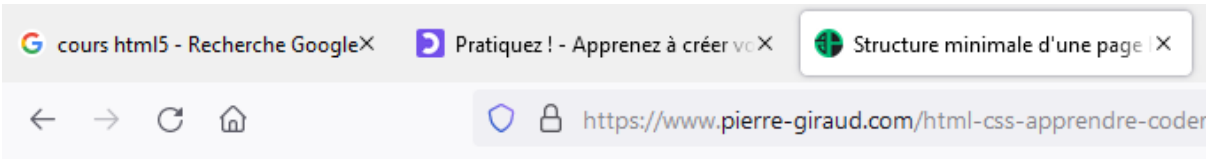
L'ÉLÉMENT HEAD

L'élément **`head`** est un élément d'**en-tête**. Il va contenir des éléments qui vont servir à fournir des informations sur la page au navigateur, comme le **titre** de la page.

L'ÉLÉMENT TITLE

Au sein de l'élément head, nous allons devoir à minima l'élément **title**.

L'élément **title** va nous permettre d'**indiquer le titre de la page** en soi, qui ne doit pas être confondu avec les différents textes définis comme des titres dans la page. Ce titre de page est le texte **visible sur le haut des onglets de votre navigateur :**



L'ÉLÉMENT « META CHARSET »

Le head peut avoir plusieurs éléments meta, qui servent à définir des méta informations qui seront utilisées par le navigateur (ou les moteurs de recherche) afin de mieux comprendre la page web. Une méta information, pour faire court (et un peu mystérieux) est une information à propos d'une information. Content ?

Dans le cas qui nous occupe ici, à savoir la structure minimale d'une page web, c'est l'élément meta avec l'attribut charset qui nous intéresse :

```
<meta charset="utf-8">
```

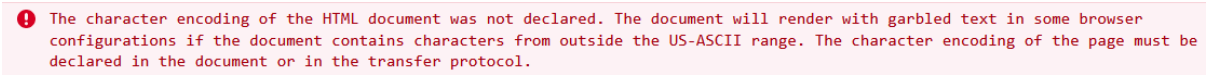
C'est technique, mais retenez simplement que les fichiers peuvent être enregistrés selon différents **encodages** (ou des jeux de caractères, des **char**acter **sets**, en anglais).

En spécifiant le **charset** **utf-8**, vous assurez une **compatibilité** plus **universelle** à votre page web. La valeur **utf-8** est de loin la valeur la plus utilisée sur le web et est la **valeur de référence pour tous les alphabets latins**. Cela va permettre à chacun de vos caractères de s'afficher correctement dans le navigateur.

Assurez-vous d'avoir spécifié le bon encodage au moment d'enregistrer votre page dans votre éditeur.

Si vous ne spécifiez pas cet élément, personne ne va mourir et la terre ne va pas exploser. Néanmoins, certains navigateurs vous avertiront en affichant une erreur dans la console développeur[2] :

[2] C'est un outil intégré à tous les navigateurs et que nous découvrirons très bientôt !



L'ÉLÉMENT BODY

L'élément `body` va lui contenir tous les éléments définissant **le contenu « visible »** de la page, c'est-à-dire les contenus à destination de l'utilisateur et notamment les différents textes présents dans la page, les images, etc.

Voilà tout pour la structure minimale d'une page HTML valide, qui représente toujours la première étape de création d'une vraie page web. Pour le moment, notre page ne possède pas de contenu visible. Nous allons en ajouter progressivement.

IMBRICATION DES BALISES ET DES ÉLÉMENTS

Un autre concept qu'il vous faut comprendre absolument pour coder en HTML est la façon dont les éléments HTML **s'imbriquent les uns dans les autres**.

Vous l'avez probablement remarqué : ci-dessus, nous avons placé des éléments HTML entre les balises ouvrantes et fermantes d'autres éléments (par exemple, l'élément `title` a été placé à l'intérieur de l'élément `head`).

On appelle cela **l'imbrication**. L'imbrication est l'une des fonctionnalités du HTML qui fait toute sa force (nous découvrirons réellement pourquoi plus tard, avec l'étude du CSS).

Cependant, **on ne peut pas imbriquer des éléments HTML n'importe comment** et il faut suivre des **règles précises**.

Ainsi, nous n'avons absolument pas le droit de « croiser » les balises des éléments[3]! Autrement dit, **le premier élément déclaré doit toujours être le dernier refermé**, tandis que **le dernier ouvert doit toujours être le premier fermé**.

```
<p>Mon chat est <strong>très</strong> grincheux.</p>
```

Il faut faire attention à ce que les éléments soient bien imbriqués les uns **DANS** les autres. Dans l'exemple précédent, on ouvre l'élément `<p>`, puis l'élément `<strong>`. Nous devrons donc fermer l'élément `<strong>` d'abord, puis l'élement `<p>`. Le code suivant est incorrect :

```
<p>Mon chat est <strong>très grincheux.</p></strong>
```

Les éléments doivent être ouverts et fermés correctement de façon à ce qu'ils soient **clairement à l'intérieur ou à l'extérieur les uns des autres**. S'ils se chevauchent, le navigateur essaiera de choisir la meilleure option, qui ne sera peut-être pas ce que vous vouliez dire et pourrait conduire à des résultats inattendus. Donc ne le faites pas !

ESPACES ET RETOURS À LA LIGNE

Les plus curieux d'entre vous auront, je suppose, déjà fait le test : **vous pouvez ajouter autant d'espaces** que vous le voulez au sein d'un paragraphe ou d'un titre ou effectuer des retours à la ligne dans votre code. En fait, en pratique, ces espaces **multiples** sont **convertis en un seul espace** par le navigateur et les retours à la ligne sont tout simplement ignorés. Donc pas de soucis à se faire à ce niveau-là.

Si vous voulez forcer visuellement un retour à la ligne dans un document html, vous pouvez utiliser l'élément **`<br>`** (de l'anglais *break*).

INDENTATION DU CODE

Indenter son code implique de le **décaler vers la droite** (= augmenter le retrait) à chaque fois qu'on **ouvre un nouvel élément** à l'aide d'une balise ouvrante.

[3] C'est comme dans Ghostbusters : il ne faut pas croiser les effluves 😊

Indenter va nous permettre d'avoir **un code plus propre et plus lisible**, donc plus **compréhensible**. Indenter permet également de plus facilement détecter les erreurs potentielles dans un code.

Comment ? Grâce à l'indentation, **la structure** du html **devient visible** dans le code.

Code non indenté	Code indenté
<!DOCTYPE html> <html> <head> <title>**Titre page**</title> </head> <body> <section> <article> **Titre** **Paragraphe** </article> </section> </body> </html>	<!DOCTYPE html> <html> <head> <title>**Titre page**</title> </head> <body> <section> <article> **Titre** **Paragraphe** </article> </section> </body> </html>

De plus, la plupart des éditeurs affichent des **indices visuels** qui permettent de visualiser facilement cette structure (regardez les lignes verticales grises) :

```
<body>
    <section>
        <article>
            <h1>Titre</h1>
            <p>Paragraphe</p>
        </article>
    </section>
</body>
```

Concernant l'indentation, **il n'y a pas de règle absolue**, notamment par rapport à la taille du retrait de chaque ligne et différents éditeurs peuvent d'ailleurs posséder différentes règles par défaut sur ce sujet.

Une bonne pratique assez courante consiste à effectuer **une nouvelle tabulation <u>après avoir</u> <u>ouvert</u> un nouvel élément qui n'est pas fermé sur la même ligne**. En effet, pas besoin d'indenter à l'intérieur d'un élément qui est fermé directement. On peut le voir dans l'exemple précédent avec les éléments `<h1>` et `<p>`.

La plupart des éditeurs sont configurés pour créer des retraits équivalents à **4 espaces**.

De même, **à chaque fois qu'on ferme un élément, on diminue l'indentation**.

LES BALISES COURANTES

LES TITRES

HTML permet de définir jusqu'à 6 niveaux de titres. Pour ce faire, vous pouvez utiliser les éléments `h1`, `h2`, `h3`... `h6`.

> Bon à savoir : « **h** » signifie « **heading** », soit l'équivalent du mot « titre » en français. **Les éléments en HTML portent souvent l'initiale de ce qu'ils représentent, en anglais**.

`h1` **est l'élément titre le plus important**, `h2` vient juste après, `h3` est moins important et ainsi de suite.

Vous verrez rapidement que les navigateurs affichent ces titres en grand et en gras. C'est un choix de présentation. Il faut néanmoins **impérativement résister** à la tentation d'utiliser les titres pour afficher du contenu en grand et en gras. Comme dit plus haut, **les éléments HTML doivent être utilisés pour structurer le contenu**, lui donner un sens, et **pas pour mettre le contenu en forme**. Les éléments html sont supposés être utilisés de **façon sémantique**. Nous nous chargerons de la mise en forme du contenu plus tard, grâce au CSS.

Ainsi, **il ne devrait y avoir qu'un seul titre `h1` sur une page car le titre `h1` est LE titre principal de la page**. Ceci n'est pas négociable. C'est faisable, visuellement ça marchera, mais sémantiquement, ça craint et c'est incorrect. C'est comme si tu donnais 2 titres à un livre : non !

En revanche, une page peut contenir **plusieurs titres de niveau inférieur**, tels des titres `h2` et `h3`. En pratique, les titres de niveau `h4`, `h5` et `h6` seront moins utilisés.

LES PARAGRAPHES ET PASSAGES À LA LIGNE

Pour créer des paragraphes en HTML, nous allons utiliser l'élément `p`.

On peut créer **autant de paragraphes que l'on souhaite** dans une page. À chaque nouveau paragraphe, il faut utiliser un nouvel élément `p`.

Pour chaque nouveau paragraphe, **un retour à la ligne va être créé automatiquement et affiché par votre navigateur**.

Ainsi le code suivant :

```
<p>Un développeur web est un informaticien spécialisé dans la programmation ou expressément impliqué dans le développement des applications du World Wide Web.</p>
<p>Les développeurs Web peuvent travailler dans différents types d'organisations, y compris les grandes sociétés et les gouvernements.</p>
<p>Certains développeurs web travaillent pour un organisme comme employés à temps plein</p>
<p>Les développeurs Web interviennent à la fois côté serveur et au niveau front-end.</p>
```

Donnera un résultat semblable à celui-ci :

Un développeur web est un informaticien spécialisé dans la programmation ou expressément impliqué dans le développement des applications du World Wide Web.

Les développeurs Web peuvent travailler dans différents types d'organisations, y compris les grandes sociétés et les gouvernements.

Certains développeurs web travaillent pour un organisme comme employés à temps plein

Les développeurs Web interviennent à la fois côté serveur et au niveau front-end.

Si vous voulez forcer visuellement un retour à la ligne dans un document html, dans un paragraphe ou en dehors, vous pouvez utiliser l'élément `<br>` (de l'anglais *break*).

LES BASES POUR INSÉRER DES IMAGES

Voici pour l'instant **la version simple**. On découvrira les spécificités plus tard dans le cours.

L'insertion d'images en HTML va se faire au moyen de l'élément HTML `img`. Cet élément est représenté par une **balise orpheline** (c'est-à-dire sans balise de fermeture).

Au sein de l'élément `img`, nous allons **obligatoirement** devoir préciser **deux attributs**: les attributs `src` et `alt`.

L'attribut `src` (pour **s**ou**rc**e) va prendre comme valeur **l'adresse** (url) de l'image (adresse relative ou absolue) tandis que l'attribut `alt` (pour ***alt****ernative*) va contenir un **texte** alternatif **décrivant l'image**. Ce texte va être affiché si l'image ne peut pas l'être pour une raison ou pour une autre, et est également très utile pour les non-voyants ou les mal voyants qui vont pouvoir « lire » notre image grâce à leurs lecteurs particuliers.

Pour insérer une image de votre site web, vous utiliserez des adresses relatives :

```
<img src="/images/home.png" alt="Home page">
```

Pour insérer une image depuis un site web externe, vous utiliserez des adresses absolues :

```
<img src="https://www.quizzfactory.be/images/boitiers.jpg" alt="Les boitiers">
```

> J'insiste ici sur l'importance de l'attribut alt : le web a été créé avec l'idée d'accessibilité à tous et il est donc de notre devoir de tout faire pour rendre chacune de nos pages lisibles pour tous et particulièrement pour les gens souffrant de déficiences.

L'attribut `src` fonctionne de la même manière que l'attribut `href` pour les liens. Ainsi, si vous enregistrez votre image dans un dossier différent de votre page HTML, pensez bien à en tenir compte dans la valeur donnée à `src`.

Une nouvelle fois, choisissez **un nom d'image sans espace ni caractère spécial** (pas d'accent notamment). Cela évitera des problèmes potentiels.

LES LIENS VERS DES PAGES

Les liens sont la base du système hypertexte : ils permettent de **créer la navigation** depuis votre page **vers d'autres pages** soit de votre site (ce sont alors des liens **internes**) ou d'autres sites (ce sont alors des liens **externes**).

En cliquant sur une **ancre** (= la partie visible d'un lien dans une page web), le visiteur sera redirigé vers la page indiquée par **l'url cible**.

Il existe aussi des liens qui permettent de naviguer au sein de la même page.

Quelque soit le type de lien utilisé, il sera toujours spécifié à l'aide de l'élément `a`, de son attribut « `href` » et du contenu qui sera affiché et cliquable, comme dans cet exemple :

```
<a href="https://www.wikipedia.com">Wikipédia</a>
```

L'élément HTML `a` est composé d'une paire de **balises** (balises ouvrante et fermante) et d'**un contenu** entre les balises que l'on va appeler "**ancre**". **Ce contenu peut être un texte, une image, etc. et sera la partie visible et cliquable du lien pour les utilisateurs**.

L'attribut `href` va nous servir à indiquer la cible du lien, c'est-à-dire l'endroit où l'utilisateur doit être envoyé après avoir cliqué sur le lien. Nous allons indiquer cette cible en valeur de l'attribut `href`.

Enfin, entre les deux balises ouvrante et fermantes, nous plaçons **la partie visible et cliquable (l'ancre)** de notre lien. Dans l'exemple plus haut, cela correspond au texte « Wikipédia ». Les utilisateurs vont pouvoir **cliquer sur ce texte** pour être renvoyés vers la page d'accueil de Wikipédia. Notez qu'ici nous avons choisi de placer **un texte** comme ancre de notre lien, mais rien ne nous empêche de **placer une image** à la place afin de créer une image cliquable.

Vous pouvez remarquer que le navigateur applique automatiquement des styles à la partie cliquable de nos liens :

- le texte (notre ancre) est de **couleur différente** (bleu avant de cliquer puis volet une fois le lien visité)
- le texte est **souligné**
- le **curseur** de notre souris change de forme lorsqu'on passe sur le lien.

Nous allons bien évidemment pouvoir changer ces comportements en **appliquant nos propres styles CSS** à nos éléments `a`. Nous verrons comment faire cela plus tard dans ce cours.

Par défaut, **les liens s'ouvrent dans la fenêtre courante du navigateur**. Quand vous cliquez sur lien, le contenu de la page en cours est remplacé par celui de la nouvelle page.

LIENS EXTERNES

Pour créer un lien externe qui pointe vers un autre site que le vôtre, il faut spécifier **une URL absolue[4] dans l'attribut href**. Exemple :

```
<a href="https://www.toyota.be">Site de Toyota</a>
```

LIENS INTERNES

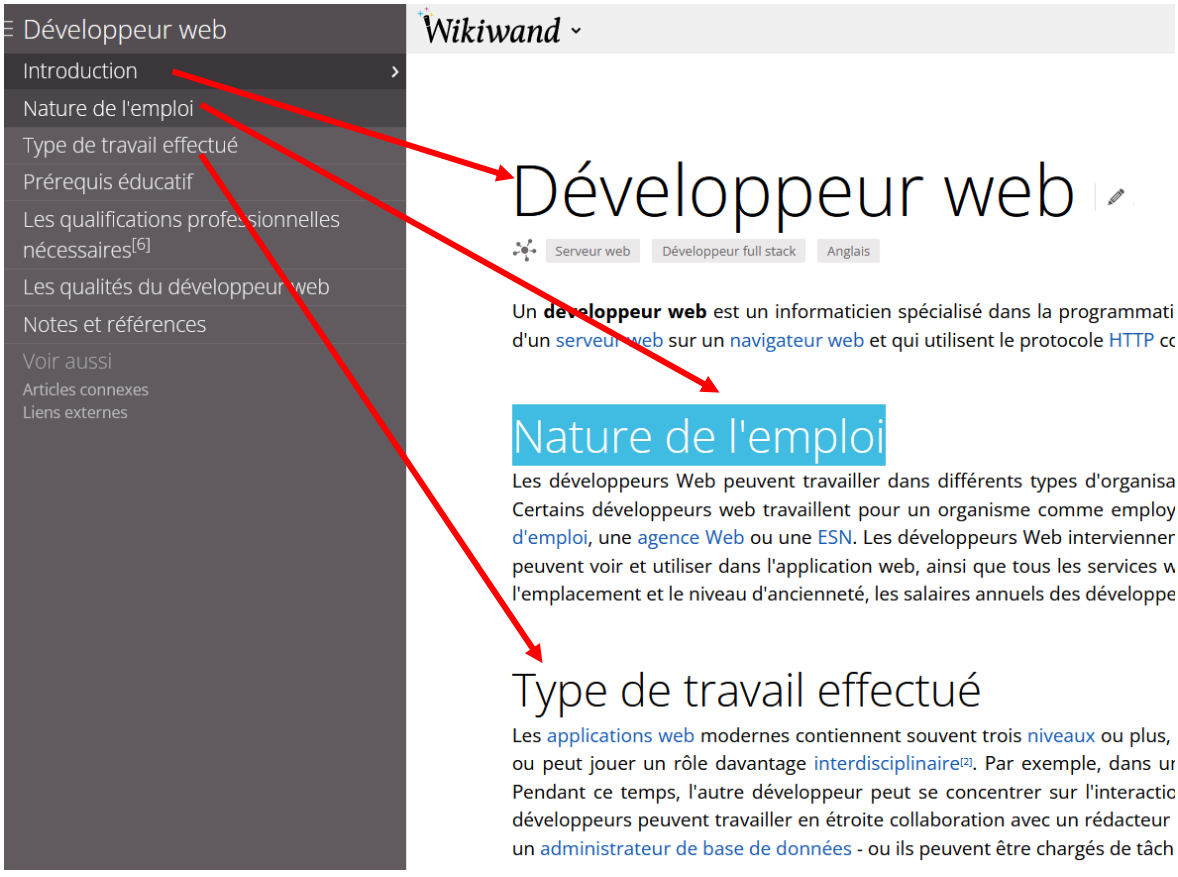
[4] Voir dans la partie « introduction » du cours

Pour créer un lien interne, qui pointe vers une autre page de votre site, il faudra utiliser une des 2 formes d'URL relatives[2] :

```
<a href="accueil.html">Site de Toyota</a>
<a href="/pages/contact.html">Site de Toyota</a>
```

LIENS VERS UN ENDROIT DANS UNE PAGE

Il est possible de créer des liens qui pointent **vers un endroit spécifique** au sein d'une page (que ce soit la même page ou une autre page). C'est utile par exemple pour créer une table des matières dans une page web, comme sur Wikipédia :



Un lien vers un endroit spécifique dans une page va **pointer** directement **vers un élément spécifique** de cette page. Pour ce faire, cet élément cible doit avoir son **attribut** `id` **spécifié**, qui devient alors ce que l'on appelle une **ancre**. La cible du lien pointera vers l'`id` de cet élément.

Imaginons le code HTML suivant :

```
<h1>Développeur web</h1>
<p>Un développeur web est un informaticien spécialisé dans la programmation ou expressément impliqué dans le développement des applications du World Wide Web...</p>

<h2 id="nature_emploi">Nature de l'emploi</h2>
<p>Les développeurs Web peuvent travailler dans différents types d'organisations...</p>
```

On voit que l'élément `h2` a un `id` « *nature_emploi* ».

Le lien pour « pointer » vers cet élément/cette ancre depuis la même page aura cette forme :

```
<a href="#nature_emploi" >En savoir plus</a>
```

C'est le caractère dièse/hashtag (**#**) dans le `href` qui indique que l'on va **pointer vers une ancre**. Dans ce cas-ci, on a une adresse relative (voir le cours sur les URL), et comme seule l'ancre est spécifiée, cette URL relative pointe vers une ancre qui se trouve **dans la page**.

Il est aussi possible de **pointer vers une ancre d'une autre page**, il suffit alors de faire précéder l'ancre par l'adresse (absolue ou relative) de la page où l'ancre est définie :

```
<a href="https://www.forem.org/developpeur#nature_emploi" >...</a>
```

Dans ce dernier cas, on a utilisé une URL absolue suivie d'une une ancre.

Les ancres permettent donc de faciliter la navigation sur une longue page en proposant des « raccourcis ».

Il existe un raccourci particulier qui permet de se rendre directement en haut de la page. Il prend l'une des 2 formes suivantes :

```
<a href="#top">Haut de la page</a>
```

ou

```
<a href="#">Haut de la page</a>
```

Au choix. Personnellement, je trouve que la version « `#top` » rend le lien plus explicite.

OUVRIR UN LIEN DANS UNE NOUVELLE FENÊTRE

Dans ces 3 cas, pour ouvrir un lien **dans une nouvelle fenêtre**, il suffit d'ajouter l'attribut « `target` » à l'élément `a` et de spécifier la valeur « `_blank` » En fait, en fonction de la configuration de votre navigateur, la plupart du temps ces liens s'ouvriront dans un nouvel onglet de navigation (tab).

Exemple :

```
<a href="/pages/contact.html" target="_blank">Nous contacter</a>
```

LIENS « VERS » UNE ADRESSE ÉLECTRONIQUE

Il est possible de créer des liens « vers une **adresse email** ». Cela permet à vos visiteurs vous envoyer facilement un email, sans devoir copier-coller votre adresse :

```
<a href="mailto:info@monsite.be">Envoyer un mail</a>
```

Il faut pour cela commencer l'attribut `href` par le mot-clé `mailto:` et de le faire suivre par une adresse email. Quand l'utilisateur clique sur ce lien, votre navigateur va ouvrir le programme de messagerie électronique configuré sur votre périphérique (ordinateur, téléphone…)

https://developer.mozilla.org/fr/docs/Web/HTML/Element/a#texte_de_lien_fort

LES LISTES

HTML permet de définir des **listes** d'**éléments**, qui permettent d'organiser le contenu.

De base, les listes vont ressembler à ceci :

- Timothée Chalamet
- Zendaya
- Oscar Isaac
- Jason Momoa
- Rebecca Ferguson

1) Ouvrir la page
2) Cliquer sur le lien
3) Attendre un peu
4) S'émerveiller devant cette splendeur

On peut le voir, il y a deux grands types de listes : les **listes ordonnées** (avec des numéros) et des listes **non-ordonnées** (avec des *puces*).

LES LISTES NON-ORDONNÉES

Les listes **non-ordonnées** vont être utiles pour lister des **éléments sans hiérarchie ni ordre explicite**.

Par exemple, si je souhaite lister les mots « pomme », « vélo » et « guitare », sans plus de contexte, j'utiliserai une liste non-ordonnée. En effet, on ne peut pas dégager de notion d'ordre, de hiérarchie ou de subordination entre ces trois termes (du moins pas sans un contexte précis).

Pour créer une liste non-ordonnée, nous allons avoir besoin d'un élément **ul** (pour « ***u**nordered **l**ist* », ou « liste non-ordonnée » en français) qui va représenter **la liste** en soi ainsi que d'un élément li (« list item » ou « élément de liste ») pour chaque élément de liste, ce qui donne :

```
<ul>
  <li>Timothée Chalamet</li>
  <li>Zendaya</li>
  <li>Oscar Isaac</li>
  <li>Rebecca Fergusson</li>
</ul>
```

- Timothée Chalamet
- Zendaya
- Oscar Isaac
- Rebecca Fergusson

Comme vous pouvez le remarquer, on va placer les **éléments li à l'intérieur de l'élément de liste ul**. C'est tout à fait logique puisque les éléments de liste « appartiennent » à une liste en particulier.

Visuellement, des **puces** (les points noirs) apparaissent automatiquement devant chaque élément d'une liste non-ordonnée par défaut. Nous allons pouvoir changer ce comportement et personnaliser l'apparence de nos listes en CSS plus tard dans ce cours.

LES LISTES ORDONNÉES

Au contraire des listes non-ordonnées, nous allons utiliser les listes ordonnées lorsqu'il y aura une notion d'ordre **explicite** ou de progression logique qui doit être rendue visible.

Par exemple, si l'on souhaite lister des étapes, un classement… on utilisera généralement des listes ordonnées.

Pour créer une liste ordonnée, nous allons cette fois-ci utiliser l'élément **ol** (pour « **o**rdered **l**ist » ou « liste ordonnée ») pour définir la liste en soi et à nouveau des éléments li pour chaque élément de la liste.

```
<ol>
  <li>Ouvrir la page</li>
  <li>Cliquer sur le lien</li>
  <li>Attendre un peu</li>
  <li>S'émerveiller devant…</li>
</ol>
```

1. Ouvrir la page
2. Cliquer sur le lien
3. Attendre un peu
4. S'émerveiller devant cette splendeur

Comme vous le voyez, ce sont cette fois-ci **des numéros** qui sont affichés par défaut devant chaque élément de la liste.

Encore une fois, nous allons pouvoir changer ce comportement et afficher différents styles de puces avec la propriété CSS.

IMBRIQUER DES LISTES

Finalement, sachez qu'il est tout-à-fait possible d'**imbriquer** (c'est-à-dire « insérer à l'intérieur ») une liste dans une autre en suivant quelques règles simples.

Pour imbriquer une liste dans une autre, il suffit de **définir une nouvelle liste à l'intérieur de l'un des éléments d'une autre liste**, juste avant la balise fermante de cet élément :

```
<!--Listes imbriquées-->
<ol>
  <li>Introduction</li>
  <li>Partie I
    <!--On imbrique une liste non-ordonnée dans une liste ordonnée-->
    <ul>
      <li>Définitions</li>
      <li>Auteurs</li>
      <li>Exemples</li>
    </ul>
  </li>
  <li>Partie II</li>
  <li>Conclusion</li>
</ol>
```

Ce qui donne comme résultat :

1. Introduction
2. Partie I
    - Définitions
    - Auteurs
    - Exemples
3. Partie II
4. Conclusion

Comme vous pouvez le voir, il devient ici **très important de bien indenter son code** afin de ne pas se perdre au milieu de nos listes !

Notez que l'on peut imbriquer **autant de listes que l'on souhaite** les unes dans les autres. Cependant, pour des raisons évidentes de lisibilité, il est conseillé de ne pas créer plus de niveaux de listes que ce qui est strictement nécessaire pour servir vos besoins.

LES TABLEAUX

Un tableau en HTML représente un **ensemble organisé de données structurées présentées en lignes et colonnes**. Pour créer un tableau en HTML nous allons utiliser l'élément `table` qui signifie « tableau » en anglais.

Couplé avec un peu de CSS pour styliser ces tableaux, HTML facilite l'affichage d'informations dans des tableaux sur le web comme les emplois du temps de l'école par exemple, les horaires d'ouverture de la piscine du quartier ou des statistiques à propos de votre équipe de football ou de dinosaures favorite.

On peut par exemple créer des tableaux comme ceux-ci :

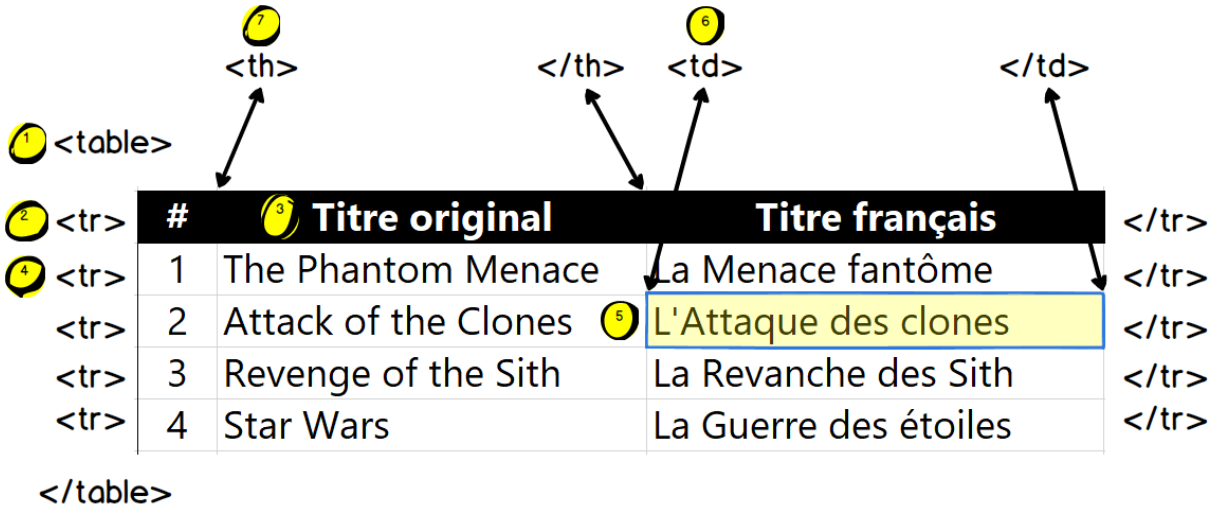
Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Public Swim 06:30 - 10:30	Public Swim 06:30 - 09:00	Public Swim 06:30 - 09:00	Public Swim 06:30 - 11:15	Public Swim 06:30 - 09:00	Lane Swim 08:00 - 09:00	Lane Swim 08:00 - 09:00
Aquacise 10:30 - 11:15	Aqua Jog 09:15 - 10:00	Education Swimming Lessons 09:00 - 12:00	Aquacise 11:15 - 12:00	Education Swimming Lessons 09:00 - 12:00	Oldham Active Kids Swimming Lessons 09:00 - 13:00	Public Swim 09:00 - 11:00
Lane Swim 11:30 - 13:00	Parent & Baby Class 09:30 - 10:15	Lane Swim 12:00 - 13:00	Lane Swim 12:00 - 13:00	Lane Swim 12:00 - 13:00	Parent and Baby 12:00 - 12:45	Aquacise 11:00 - 11:45
Education Swimming Lessons	Public Swim 10:00 - 11:45	Public Swim 13:00 - 16:00	Education Swimming Lessons	Oldham Active Kids Swimming	Public Swim 13:00 - 17:00	Public Swim 11:45 - 13:00

#	Titre original	Titre français	Année	Budget	Recettes
1	The Phantom Menace	La Menace fantôme	1999	115.000.000,00 €	924.317.558,00 €
2	Attack of the Clones	L'Attaque des clones	2002	120.000.000,00 €	649.398.328,00 €
3	Revenge of the Sith	La Revanche des Sith	2005	113.000.000,00 €	850.000.000,00 €
4	Star Wars	La Guerre des étoiles	1977	11.000.000,00 €	775.398.007,00 €
5	The Empire Strikes Back	L'Empire contre-attaque	1980	18.000.000,00 €	538.400.000,00 €
6	Return of the Jedi	Le Retour du Jedi	1983	32.350.000,00 €	572.700.000,00 €
7	The Force Awakens	Le Réveil de la Force	2015	**245.000.000,00 €**	**2.068.223.624,00 €**
8	The Last Jedi	Les Derniers Jedi	2017	200.000.000,00 €	1.332.539.889,00 €
9	The Rise of Skywalker	L'Ascension de Skywalker	2019	250.000.000,00 €	1.074.144.248,00 €

Caractéristiques	
Moteur :	V8 Essence
Cylindrée :	5,4L
DIN :	500 Chevaux
Couple maxi :	65 m/Kg à 5500
Boîte de vitesse :	BVM6
Poids à vide :	1779 Kg
Vitesse maxi :	260 Km/h
Consommation moyenne :	12 L/100Km

COMMENT ÇA MARCHE ?

Un tableau (`<table>`) a une ou plusieurs lignes (rows = `<tr>`) et chaque ligne est composée d'une ou plusieurs cellules/données (data = `<td>`), comme ceci :



La table est définie à l'aide de l'élément `<table>` [1]. Chaque **ligne** est délimitée par une paire de balises `<tr>` et `</tr>`. [2] [4] Dans une ligne, chaque **cellule** qui contient des **données** (=data) sera délimitée par un élément `<td>` [6].

La **première ligne** est généralement un peu particulière. En effet, la plupart du temps elle contient les **entêtes des colonnes** [2] [3]. Dans ce cas, les éléments <td> sont remplacés par des éléments <th> [7] (qui signifient ***t**able **h**ead,* soit entête de tableau).

Voici ce à quoi cela ressemble en :

```
<table>
	<tr>
		<th>#</th>
		<th>Titre Original</th>
		<th>Titre français</th>
	</tr>

	<tr>
		<td>1</td>
		<td>The Phantom Menace</td>
		<td>La Menace fantôme</td>
	</tr>

	<tr>
		<td>2</td>
		<td>Attack of the Clones</td>
		<td>L'Attaque des clones</td>
	</tr>

	<tr>
		<td>3</td>
		<td>Revenge of the Sith</td>
		<td>La Revanche des Sith</td>
	</tr>

</table>
```

QUAND NE PAS UTILISER DE TABLEAUX EN HTML ?

Les tableaux HTML ne doivent être utilisés que pour des **données tabulaires** — c'est pour cela qu'ils sont conçus. Malheureusement, beaucoup de gens ont utilisé les tableaux HTML ~~**pour organiser des pages Web**~~, par exemple : une ligne pour contenir l'en-tête, une ligne pour les colonnes de contenu, une ligne pour le pied de page, etc.

Il y a très très longtemps, dans une galaxie très très lointaine, cette solution avait son utilité, sans être une bonne pratique pour autant. Sans code CSS performant, les

développeurs HTML ont opté pour la facilité. **De nos jours**, **CSS est bien assez puissant pour** définir efficacement **la mise en page** des pages HTML. On verra ça plus tard.

MIEUX STRUCTURER LES TABLEAUX

Comme vos tableaux deviennent un peu plus structurellement complexes, il est utile d'en améliorer la définition. Une façon claire d'y parvenir consiste à utiliser les éléments `<thead>`, `<tfoot>` et `<tbody>`, qui vous permettent de marquer l'**en-tête**, le **pied** et le **corps** du tableau.

Ces éléments ne rendent pas le tableau plus accessible aux utilisateurs de lecteurs d'écran, et n'entraînent aucune amélioration visuelle par eux-mêmes. Ils sont cependant **très utiles pour la présentation et la mise en page** — agissant comme des accroches pour l'ajout des CSS. Pour vous donner quelques exemples intéressants, dans le cas d'un grand tableau, vous pouvez **répéter l'en-tête et le pied de page sur chaque page imprimée** ; vous pouvez prévoir l'affichage du corps sur une seule page et accéder au contenu par défilement vers le haut ou vers le bas.

Pour les utiliser :

- L'élément `<thead>` doit couvrir la partie du tableau qui est l'en-tête — ce sera en général la **première ligne** contenant les en-têtes de colonnes.
- L'élément `<tfoot>` doit envelopper la partie du tableau qui est le **pied de page** — ce peut être une dernière ligne contenant, par exemple, la somme des rangées précédentes. Vous pouvez inclure l'élément `<tfoot>` à la suite du code contenant le corps du tableau, là où vous souhaitez le trouver, ou juste en-dessous de l'élément <thead> (le navigateur l'affichera toujours en pied de tableau).
- L'élément `<tbody>` doit couvrir toutes les parties du tableau non contenues dans un `<thead>` ou un `<tfoot>`. Il pourra apparaître dans le code, sous la déclaration de l'en-tête ou du pied de page, selon la façon dont vous avez décidé de le structurer

Concrètement :

```
<table>
    <thead>
        <tr>
            <th>#</th>
```

```
                    <th>Titre Original</th>
                    <th>Titre français</th>
                </tr>
            </thead>

            <tbody>
                <tr>
                    <td>1</td>
                    <td>The Phantom Menace</td>
                    <td>La Menace fantôme</td>
                </tr>

                ...

                <tr>
                    <td>3</td>
                    <td>Revenge of the Sith</td>
                    <td>La Revanche des Sith</td>
                </tr>
            </tbody>

            <tfoot>
                <tr>...</tr>
            </tfoot>

        </table>
```