

Apprendre le HTML

Après notre introduction à HTML et CSS, il est temps de creuser un peu plus profondément dans le HTML et d'examiner les différents éléments qui composent ce langage.

Afin de commencer la construction de sites, nous avons besoin d'en apprendre un peu plus sur quels sont les éléments HTML les plus appropriés pour afficher différents types de contenu.

Il est également important de comprendre comment ces éléments vont s'afficher sur une page Web, ainsi que ce que signifient sémantiquement ces différents éléments.

Aperçu sur le code sémantique

Alors qu'est-ce qu'exactly la sémantique? La [sémantique en HTML](#) est la pratique de donner du sens au contenu et sa structure de la page en utilisant l'élément approprié. Le code sémantique décrit la *valeur* du contenu d'une page, quel que soit le style ou l'apparence de celui-ci. Il y a plusieurs avantages à utiliser les éléments sémantiques, permettre aux ordinateurs, lecteurs d'écran, moteurs de recherche, et autres appareils de lire correctement et d'analyser le contenu d'une page Web. En outre, il est plus facile de gérer et de travailler avec du HTML sémantique car il montre clairement quel est le sujet de chaque élément de contenu.

Au fur et à mesure que l'on va avancer dans ce cours et que de nouveaux éléments seront introduits, nous allons parler de ce que ces éléments signifient réellement et le type de contenu qu'ils représenteront le mieux.

Avant de passer à cela, regardons deux éléments `<div>` et `` qui de fait ne détiennent aucune valeur sémantique et existent seulement à des fins de mise en forme.

Identification des div & span

Les divisions ou `<div>` et `` sont des éléments HTML qui agissent comme des conteneurs et qui servent uniquement à des fins de mise en forme. Comme ce sont des conteneurs génériques, ils ne viennent pas avec un sens global ou une valeur sémantique. Les paragraphes sont sémantiques parce que le contenu enveloppé dans un élément `<p>` est connu et compris comme un paragraphe. `<div>` et `` ne détiennent pas un tel sens et sont tout simplement des contenant.

Elements « block » contre éléments « inline »

La plupart des éléments sont soit des éléments de type « block » -*bloc*- soit des éléments de type « inline » -*en ligne*-.

Quelle est la différence?

Les éléments **block** commencent sur une nouvelle ligne, s'empilent les uns sur les autres et occupent toute la largeur disponible. Les éléments « block » peuvent être imbriqués l'un dans l'autre et peuvent envelopper des éléments de type « inline ». Le plus souvent les éléments **block** seront utilisés pour de gros morceaux de contenu, comme par exemple un ensemble de paragraphes.

Les éléments **inline** ne commencent pas sur une nouvelle ligne. Ils restent dans le flux normal d'un document, se positionnant, s'alignant les uns après les autres. La largeur de l'élément correspond à la largeur de son contenu. Les éléments **inline** peuvent être imbriqués les uns dans les autres cependant ils ne peuvent pas envelopper des éléments de type *block*, à l'exception de l'élément ancre « **a** ». Nous verrons généralement que les éléments **inline** couvrent de petits morceaux de contenu, tels que quelques mots.

[La référence sur les balises inline et block sur le site de Mozilla](#)

Cependant les éléments `<div>` et `` sont extrêmement précieux lors de la construction d'un site Web en ce qu'ils nous donnent la possibilité d'appliquer des styles à un ensemble de contenus que l'on va pouvoir cibler.

`<div>` est un élément de niveau block qui est couramment utilisé pour identifier de grands groupes de contenu, et qui contribue à renforcer la mise en page d'une page de web.

`` d'autre part, est un élément de niveau inline couramment utilisé pour identifier de petits groupes de texte dans un élément de niveau bloc.

Nous allons souvent utiliser les `<div>` et `` avec les attributs `class` ou `id` à des fins de mise en forme -couleur, taille de la police, etc...-.

Le choix du nom d'une `class` ou d'un `id` nécessite un peu de soin. Nous voulons choisir une valeur qui fait référence au contenu d'un élément, pas nécessairement à l'aspect d'un élément.

Par exemple, si nous avons un `<div>` avec un fond orange qui contient des liens vers des réseaux sociaux, notre première pensée pourrait être de donner à la `<div>` une `class` ayant pour libellé `orange`.

Que se passe t'il si ce fond orange est plus tard changé en bleu ? Avoir une `class` libellée `orange` n'a plus de sens. Un choix plus judicieux pour le libellé de cette `class` serait `social` qui se référera au contenu de la `<div>`, et non pas au style.

```
<!-- Div -->
<div class="social">
  <p>Il a peut être trouvé sur ...</p>
  <p>En outre, j'ai un profil sur ...</p>
</div>

<!-- Span -->
<p>Bientôt nous serons les meilleurs en <span class="tooltip">écriture HTML</span> et de loin.</p>
```

Commentaires au sein du HTML & CSS

Le code précédent comprend des points d'exclamation dans le HTML, et pourtant tout va bien. Ce ne sont pas des éléments, ce sont des commentaires.

HTML et CSS nous donnent la possibilité de mettre des commentaires au sein de notre code, et tout contenu enveloppé dans un commentaire ne sera pas affichée sur la page Web. Les commentaires servent à aider à garder nos fichiers organisés, nous permettant d'y inscrire tout type d'informations, et de fournir un moyen pour nous de gérer plus efficacement notre code.

Les commentaires deviennent particulièrement utiles quand il y a plusieurs personnes qui travaillent sur les mêmes fichiers.

Les commentaires HTML commencent par `<!--` Et terminent par `-->`.

Les commentaires CSS commencent par `/*` et terminent par `*/`.

Utilisation des éléments de texte à base

Dans le web il y'a énormément de différents types de médias et de type de contenus en ligne, cependant le texte reste le type de contenu prédominant. Par conséquent, il existe un grand nombre d'éléments différents pour afficher du texte sur une page Web. Pour l'instant nous allons nous concentrer sur les éléments les plus populaires, titres, paragraphes, texte gras pour montrer "l'importance", et italique pour "l'émphase". Plus tard, dans une future leçon sur la typographie, nous jetterons un coup d'oeil plus précis sur la les styles applicables au texte.

Les Titres « H1 » à « H6 »

Les titres sont des éléments de niveau bloc, et il en existe six niveaux différents, de `<h1>` à `<h6>`. Ces rubriques aident à scinder le contenu et établir une hiérarchie. Ce sont des identificateurs clés pour les utilisateurs lors de la lecture d'une page. Ils aident également les moteurs de recherche car cette hiérarchie est prise en compte dans l'indexation et permet ainsi de déterminer le contenu d'une

page. Les titres doivent être utilisés dans un ordre qui est pertinent pour le contenu d'une page. Le titre principal d'une page ou section doit être marqué avec un `<h1>` et les rubriques suivantes devraient utiliser `<h2>`, `<h3>`, `<h4>`, `<h5>` et `<h6>`

Chaque rubrique de niveau doit être utilisée là où elle est sémantiquement juste, et ne doit pas être utilisée pour rendre texte gras ou gros il y a autres façons, bien meilleures de le faire.

Voici un exemple de code HTML pour tous les différents niveaux de titre.

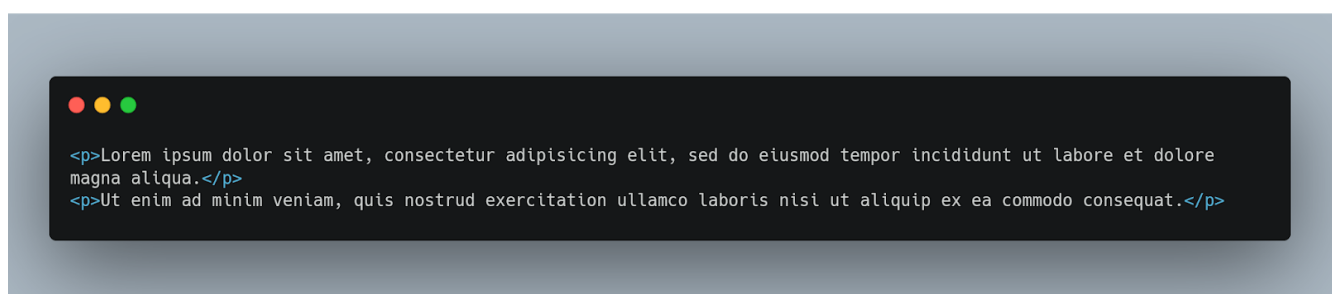
A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains six lines of HTML code for title tags, each on a new line.

```
<h1>Titre Niveau 1</h1>
<h2>Titre Niveau 2</h2>
<h3>Titre Niveau 3</h3>
<h4>Titre Niveau 4</h4>
<h5>Titre Niveau 5</h5>
<h6>Titre Niveau 6</h6>
```

Les Paragraphes « p »

Ces titres sont souvent suivis par des éléments paragraphes. Les paragraphes sont définis par l'élément de niveau bloc `<p>`. Les paragraphes peuvent apparaître l'un après l'autre, permettant l'ajout d'informations à une page.

Voici un exemple de façon de mettre place paragraphes.

A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains two lines of HTML code for paragraph tags, each on a new line.

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
<p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>
```

Texte gras avec « strong »

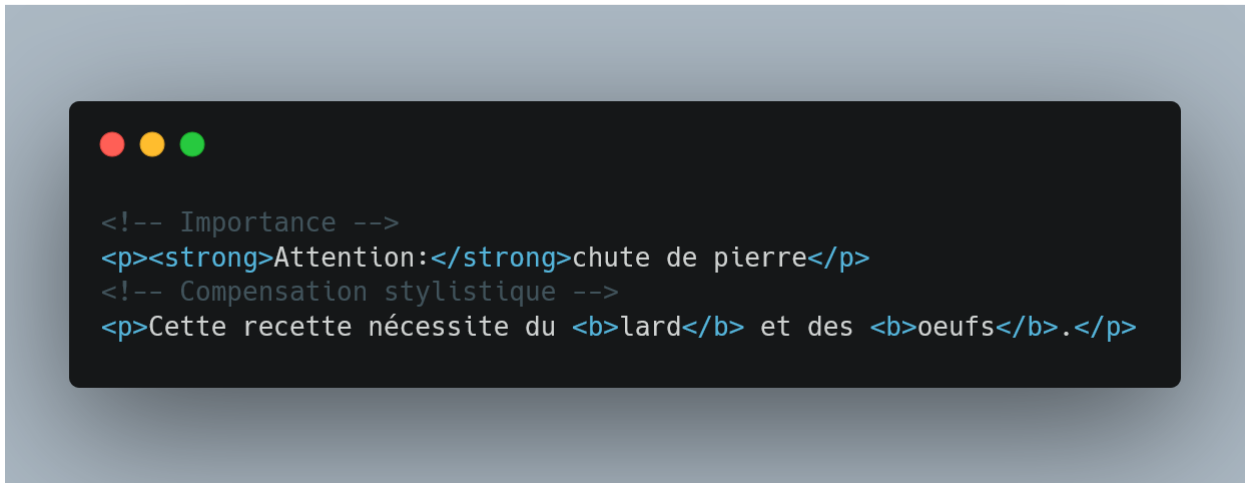
Pour mettre un texte en gras et accorder ainsi plus d'importance à celui ci, nous allons utiliser l'élément `` de niveau en ligne.

Il y a deux éléments qui peuvent mettre du texte en gras: `` et ``. Il est important de comprendre la différence sémantique entre les deux.

Le `` est sémantiquement utilisé pour donner une forte importance au texte, et est donc l'option la plus populaire pour mettre un texte gras.

Le ``, d'autre part, est de moindre sens sémantiquement parlant mais il compensera stylistiquement le texte, ce qui ne est pas toujours le meilleur choix pour que le texte soit mis au premier plan. Nous devons évaluer l'importance du texte que nous souhaitons mettre en gras et de choisir un élément en conséquence

Voici les deux options HTML pour créer texte gras dans action:

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains the following HTML code:

```
<!-- Importance -->
<p><strong>Attention:</strong>chute de pierre</p>
<!-- Compensation stylistique -->
<p>Cette recette nécessite du <b>lard</b> et des <b>oeufs</b>.</p>
```

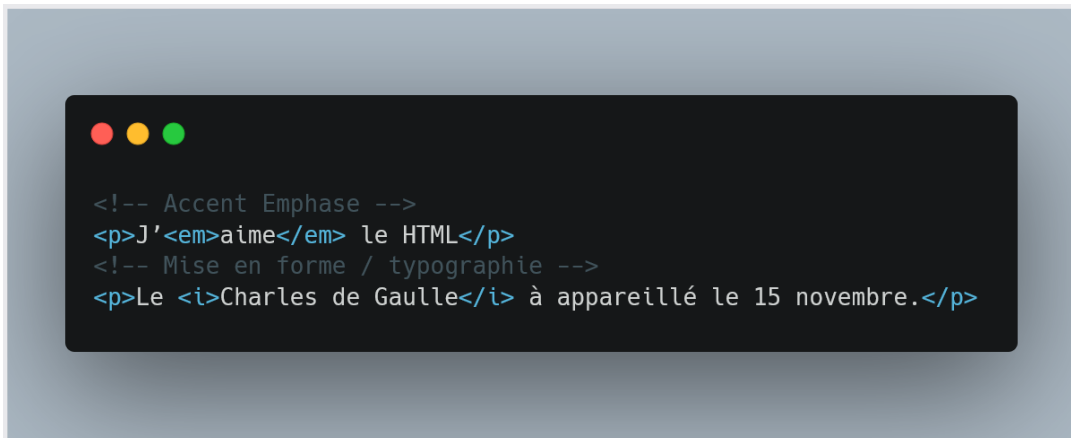
Les textes en italique avec « em »

Pour un texte en italique, mettant ainsi l'accent sur celui-ci, nous allons utiliser l'élément `` de niveau ligne. Comme les éléments de texte gras, il y a deux éléments différents qui italique le texte.

L'élément `` est utilisé sémantiquement de mettre l'accent et souligner un texte il est donc l'option la plus populaire pour mettre le texte en italique.

L'élément `<i>` traduit la seule mise en forme. Il ne s'agit pas de mettre l'accent sur l'expression mais bien de respecter une règle typographique.

Voici le code HTML pour :



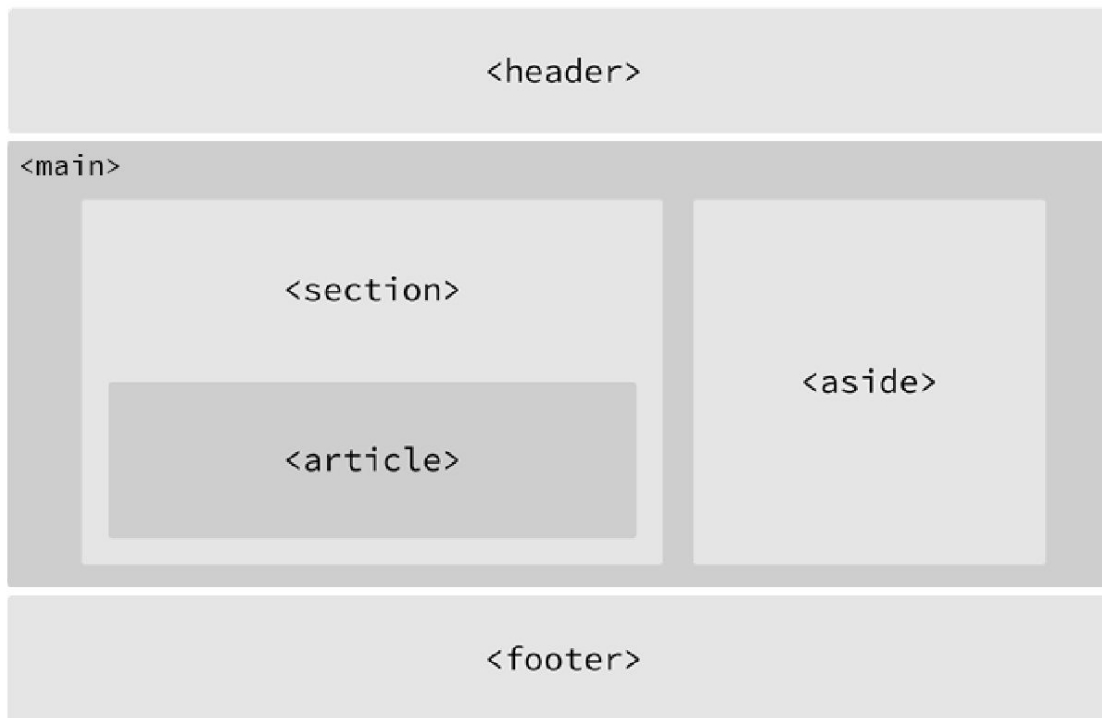
Construire la structure d'une page

Depuis fort longtemps la structure d'une page Web était construite à l'aide de `<div>`. Le problème est que les `<div>` ne fournissent aucune valeur sémantique, et il est donc assez difficile de déterminer ce à quoi servent ces `<div>`.

Heureusement HTML5 introduit de nouveaux éléments de base d'une structure de page comme les éléments `<header>`, `<nav>`, `<main>`, `<article>`, `<section>`, `<aside>` et `<footer>`.

Tous ces nouveaux éléments sont destinés à donner sens à l'organisation de nos pages et améliorer sémantiquement la déclaration de notre structure de page. Ce sont tous des éléments de niveau bloc qui n'ont pas de position quelconque ou un de style implicite. En outre, tous ces éléments peuvent être utilisés plusieurs fois par page, tant que chaque utilisation reflète la signification sémantique correcte.

Retroussons nos manches et regardons de plus près.



Un exemple possible de structure HTML5 donnant sens à l'organisation de nos pages

header

L'élément `<header>`, comme son nom l'indique, est utilisé pour identifier le haut d'une page, d'un article ou autre section d'une page. En général, `<header>` inclura un titre, un texte d'introduction ou même de navigation.



header, head et autres éléments titre de `<h1>` à `<h6>`

Il est facile de confondre l'élément `<header>` avec l'élément `<head>` ou les éléments de titre de rubrique `<h1>` à `<h6>`. Ils ont pourtant tous des significations différentes et pas que sémantiquement et doivent être utilisés en fonction de leurs significations et usage.

Pour référence:

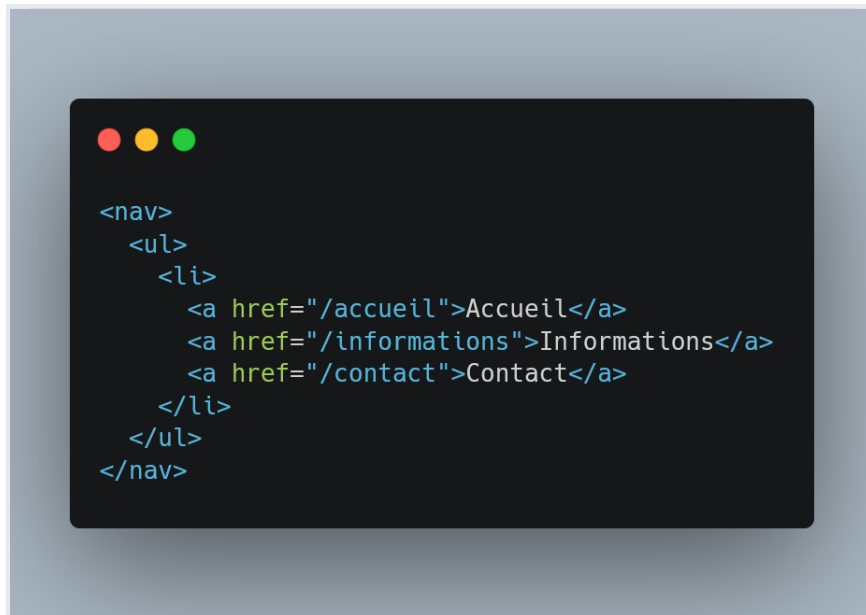
- `<header>` L'élément est un élément structurel qui décrit une rubrique d'un segment d'une page. Il se trouve dans le `<body>`.
- `<head>` L'élément est pas affiché sur une page et est utilisé pour décrire métadonnées, y compris le titre du document, et liens vers fichiers externes. Il tombe directement dans l'élément `<html>`.
- Les éléments de position `<h1>` à `<h6>`, sont utilisés pour désigner plusieurs niveaux de titres de texte à travers une page.

Nav

L'élément `<nav>` identifie une section des principaux liens navigation sur une page. L'élément `<nav>` doit être réservé pour les sections de navigation primaire seulement, comme la navigation globale du site, une table des matières, des liens précédents / suivant ou autres groupes remarquables de liens de navigation.

Le plus souvent, les liens inclus dans `<nav>` relieront à d'autres pages du même site ou à des parties de la même page Web.

Divers liens ponctuels ne doivent pas être insérés dans un élément `<nav>`, il faudra alors utiliser l'élément d'ancrage `<a>` et seulement l'élément d'ancrage.



Main

L'élément `<main>` représente le contenu principal d'une page Web. Bien que le contenu qui se répète sur plusieurs pages puisse être placé dans des en-têtes, des sections ou tout autre élément, l'élément `<main>` est réservé au contenu qui est spécifique et unique à la page en question. Par conséquent, avant HTML 5.2, l'élément `<main>` devait être unique dans le DOM pour que la page soit valide. Toutefois, compte tenu de la prédominance des demandes de site à page unique, il était difficile de s'en tenir à cette règle. C'est pourquoi depuis HTML 5.2, il est possible d'avoir plusieurs éléments `<main>` dans notre balisage, *tant qu'un seul est visible par l'utilisateur à un moment donné*.

Tout élément supplémentaire doit être masqué à l'aide de l'attribut `hidden`.

Article

L'élément `<article>` est utilisé pour identifier une section de contenu autonome indépendant qui peut être distribuée ou ré-utilisée de façon indépendante.

Nous allons souvent utiliser `<article>` pour marquer les billets de blog, articles éditoriaux ou contenu soumis par des utilisateurs...

Au moment de décider d'utiliser ou non l'élément `<article>`, nous devons déterminer si le contenu de l'élément pourrait être reproduit ailleurs sans aucune confusion.

Si le contenu au sein de l'élément `<article>` est retiré du contexte de la page et placé, par exemple, dans un courriel ou un document de travail imprimé le contenu devra encore faire sens.



Section

L'élément `<section>` est utilisé pour réaliser un regroupement thématique de contenu, qui généralement, mais pas toujours, correspond à une rubrique. Le regroupement de contenu au sein de `<section>` peut être aussi de nature générique.

La `<section>` est couramment utilisé pour séparer les contenus et fournir une hiérarchie à la page.



Décider entre les éléments `<article>` `<section>` ou `<div>`

Il est parfois assez difficile de décider lequel des éléments - `<article>` `<section>` ou `<div>`. L'astuce ici, comme avec chaque décision sémantique, est de regarder le contenu. Les deux éléments `<article>` et `<section>` contribuent à la structure d'un document et aident à le définir.

- Si le contenu est regroupé uniquement à des fins de style et ne fournit pas valeur à l'esquisse d'un document, utilisez l'élément `<div>`.
- Si le contenu ajoute au plan document et il peut être redistribué ou partagé indépendamment, utilisez l'élément `<article>`.
- Si le contenu ajoute au document de grandes lignes et représente un groupe thématique de contenu, utilisez l'élément `<section>`.

Informations connexes `<aside>`

L'élément `<aside>` contient du contenu, tels que du contenu complémentaire

-présenté sur le côté généralement-, des inserts, ou de brèves explications, qui seront connexes au contenu qui l'entoure. Lorsqu'il est utilisé dans un élément `<article>` par exemple `<aside>` peut identifier le contenu lié à l'auteur de l'article. Nous pouvons instinctivement penser à un `<aside>` comme un élément qui apparaît sur le côté gauche ou droit d'une page. Il faut se rappeler, cependant, que tous les éléments structurels, y compris l'élément `<aside>`, sont des éléments de niveau bloc et en tant que tels apparaîtront sur une nouvelle ligne, occupant la largeur disponible de la page ou de l'élément dans lesquels ils sont emboîtés, élément également connu comme élément parent.



Nous verrons la façon de changer la position d'un élément, le placer à droite ou à gauche d'un groupe de contenu, plus tard dans le cours.

Pied page `<footer>`

L'élément `<footer>` identifie la fermeture ou la fin d'une page ,article, section d'une page.

En général le `<footer>` se trouve au bas de son parent. Le contenu de l'élément `<footer>` doit être une information relative au contenu de l'élément contenant.




Avec ces éléments de structure et les éléments de base de texte, nos connaissances HTML commence vraiment à se réunir. Maintenant c'est le bon moment pour revenir sur notre site de test et voir si nous pouvons améliorer sa structure.

En pratique

Notre site Web débuté dans le cours précédent manque cruellement de structure et de contenu. Prenons un peu de temps pour étoffer notre page d'accueil.

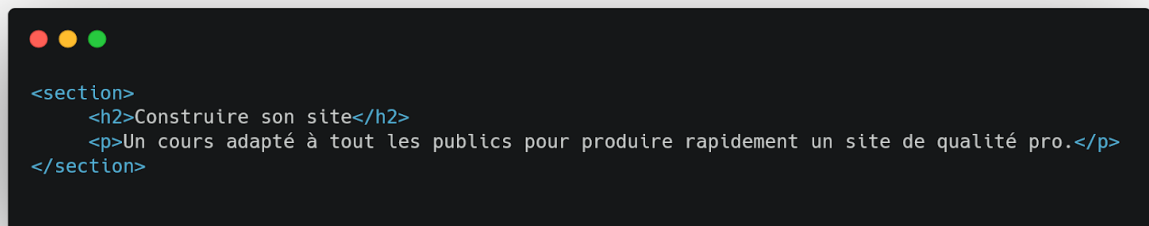
- A partir de notre fichier `index.html` existant, nous allons ajouter un élément `<header>`. Notre `<header>` inclura notre élément `<h1>` existant; et nous ajouterons également un `<h2>` comme un slogan pour soutenir notre titre `<h1>`.



```
<header>
  <h1>Cours HTML / CSS</h1>
  <h2>Octobre 2016 &ndash; Paris</h2>
</header>
```

- Après notre élément `<header>`, nous allons ajouter un nouveau groupe de contenu en utilisant l'élément `<section>` qui introduit notre cours. Nous allons commencer cette section avec un nouvel élément `<h2>` et finir avec un paragraphe.

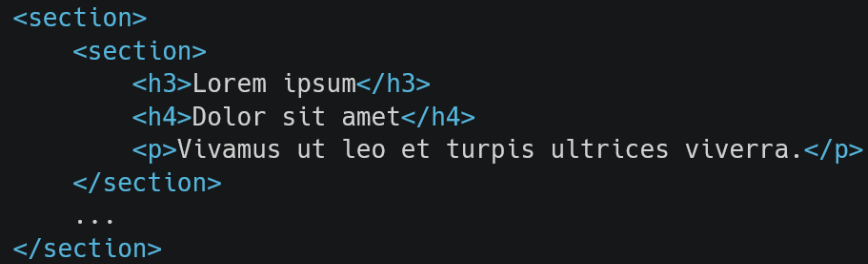
- Après l'introduction à notre cours, nous allons ajouter un autre groupe de



```
<section>
  <h2>Construire son site</h2>
  <p>Un cours adapté à tout les publics pour produire rapidement un site de qualité pro.</p>
</section>
```

contenu qui mettra en valeur les autres pages. Chacune des pages aura sa propre section et inclura un texte explicatif.


Nous allons regrouper tous ces présentations dans un élément `<section>`, et chaque présentation individuelle sera intégrée dans un autre élément `<section>`. En tout, nous aurons trois éléments `<section>` à l'intérieur d'une autre `<section>`.



```
<section>
  <section>
    <h3>Lorem ipsum</h3>
    <h4>Dolor sit amet</h4>
    <p>Vivamus ut leo et turpis ultrices viverra.</p>
  </section>
  ...
</section>
```

- Enfin ajoutons nos droits d’auteur dans l’élément `<footer>` à la fin de notre page. Pour ce faire nous allons utiliser l’élément `<small>`, qui représente sémantiquement un commentaire et affiche de petits caractères parfaits pour notre droit d’auteur.

En règle générale, le contenu dans l’élément `<small>` sera rendu comme, plus petit, sauf si une réinitialisation CSS fait en sorte que cela ne se produise pas !



```
<footer>
  <small>&copy; BCMA</small>
</footer>
```

Maintenant nous pouvons voir que notre page d'accueil commence à ressembler à quelque chose !



Cours HTML / CSS

Octobre 2016 - Paris

Lorem Ipsum

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim...

...ad minim veniam

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat

© BCMA

Notre page d'accueil après avoir ajouté plus de contenu et de structure

Encodage des caractères spéciaux

L'élément `<h2>` dans notre `<header>`, ainsi que l'élément `<small>` dans notre `<footer>`, contiennent des "choses" intéressantes, précisément des caractères spéciaux. Les caractères spéciaux sont la traduction codée de divers signes de ponctuation, lettres accentuées et symboles.

Lorsque l'on tape directement le caractère dans le code HTML, ils peuvent être mal interprétés ou confondus avec le mauvais caractère donc ils ont besoins d'être encodés.

Chaque caractère encodé commencera par une esperluette, « `&` », et terminera par un point-virgule « `;` ». Ce qui se trouve entre l'esperluette et la virgule est le codage unique d'un caractère, que ce soit un nom ou codage numérique. Par exemple, on pourra coder le mot « résumé » ainsi « `résumé` », au sein de notre entête nous avons codés un tiret court et dans notre pied de page nous avons codé le symbole du droit d'auteur « `copyright` ».

On peut trouver la liste de tous les caractères encodés en tapant « `html entities` » dans notre moteur de recherche préféré.

Notre page d'accueil prend forme, nous allons jeter un coup d'oeil à la création d'hyperliens afin que nous puissions ajouter des pages supplémentaires et de construire le reste de notre site Web.

Création des hyperliens

Avec le texte, un des composants de base de l'Internet est le lien hypertexte, qui offre la possibilité de lier des pages Web ou des ressources à d'autres. Les hyperliens sont établies en utilisant l'élément ancre, `<a>`, élément de niveau ligne. Afin de créer un lien d'une page (ou ressource) à un autre il faut utiliser l'attribut `href`, qui prendra comme valeur une référence de lien hypertexte. La valeur de l'attribut `href` identifie la destination du lien.

Par exemple si l'on clique sur le texte "Google !!!" qui est libellé à l'intérieur de l'élément d'ancrage avec pour valeur de l'attribut « `href: http://google.com` » les utilisateurs quitteront le site pour aller sur la page d'accueil de Google.



```
<a href="http://google.com">Google !!!!</a>
```

Insérer un élément de niveau block dans un lien

Par nature l'élément d'ancrage, `<a>`, est un élément en ligne, et, selon les standards du Web, un élément de niveau en ligne ne peuvent pas envelopper des éléments de niveau bloc. Avec l'introduction de HTML5, les éléments d'ancrage ont spécifiquement la possibilité d'envelopper des éléments de niveau ligne ou des éléments de niveau bloc. Ceci est donc une rupture de la convention, mais qui a été autorisée afin de permettre de faire des liens sur des blocs entiers de contenu sur une page.

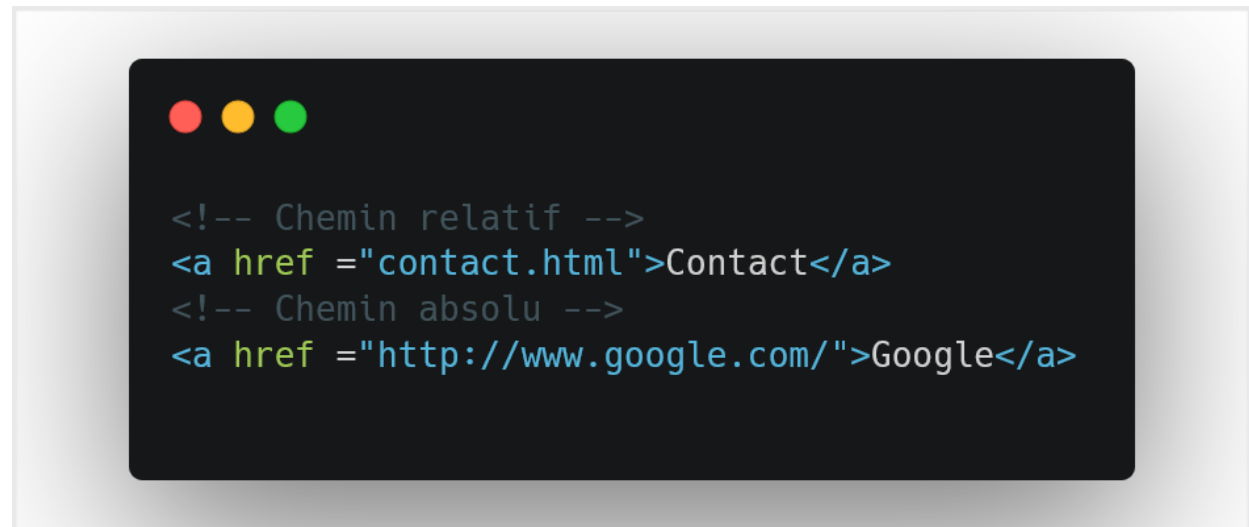
Chemins relatifs et absolus

Les deux types les plus courants de liens sont les liens vers d'autres pages du même site web et liens vers une page sur un autres sites. Les liens pointant vers d'autres pages du même site auront un *chemin* relatif, qui ne comprend pas le nom de domaine (.com, .org, .edu, etc...) dans la valeur de l'attribut `href`. Parce que le lien pointe vers autre page sur le même site, la valeur d'attribut `href` doit inclure uniquement le nom du fichier de la page vers laquelle le lien est fait, par exemple : `contact.html`.

Si la page liée se trouve dans un répertoire, dossier, différent, la valeur de l'attribut `href` doit aussi refléter cela. Par exemple si la page `contact.html` est placée dans le répertoire "pages", le chemin relatif serait alors `"pages/contact.html"`.

Les liens vers les sites extérieurs de la page, exigent un *chemin* absolu, où la valeur de l'attribut `href` doit inclure le domaine complet. Un lien vers Google aurait besoin dans la valeur de l'attribut `href` de <http://google.com>, commençant par `http` et incluant le nom de domaine, `.com` dans ce cas.

Dans l'exemple ci dessous si l'on clique sur le texte «Contact» ouvrira la page `contact.html` de votre site dans votre navigateur et si l'on clique sur le texte "Google" d'autre part, va ouvrir la page d'accueil de Google <http://google.com/> au sein de notre navigateur.



Pour créer un [lien email](#), la valeur d'attribut `href` doit commencer par `mailto:` suivi de l'adresse électronique à laquelle le message doit être envoyé. Pour créer un email qui sera à envoyer à `client@google.com`, par exemple, la valeur de l'attribut `href` sera <mailto:client@google.com>.

En outre, le corps de texte, et d'autres informations pour l'email peuvent être insérés. Pour ajouter une ligne d'objet, nous allons inclure le paramètre `subject=` après l'adresse email. Le premier paramètre suivant l'adresse email doit commencer par un point d'interrogation "?" pour se lier au chemin du lien hypertexte. S'il y'a plusieurs mots dans la ligne de l'objet cela que tous les espaces soient encodés en utilisant `%20`, exemple
"Réclamation%20facturation".

L'ajout de texte dans le corps du mail fonctionne de la même manière que l'ajout du sujet, cette fois utilisant le `body=` comme paramètre dans le `href` et parce que nous allons lier un paramètre à un autre nous devons utiliser l'esperluette "&" afin de séparer les deux. Comme avec le sujet, les espaces doivent être encodés en utilisant `%20` et les sauts de ligne doivent être encodés en utilisant

%0A., pour "url encoder/décoder" un texte voir le site <http://meyerweb.com/eric/tools/dencoder/>

Si on résume un lien vers `client@google.com` avec le sujet de «Réclamation facturation» et le corps de texte de «Bonjour, je fais une réclamation sur ma facture» nécessiterait une `href` valeur d'attribut de `"mailto:client@google.com?subject=Réclamation%20facturation&body=Bonjour%20,%20je%20fais%20une%20réclamation%20sur%20ma%20facture"`.

```
<a href="mailto:client@google.com?subject=Réclamation%20facturation&body=Bonjour%20,%20je%20fais%20une%20réclamation%20sur%20ma%20facture">
  Contacter le service facturation
</a>
```

Exemple complet

Liens et ouverture dans une nouvelle fenêtre

Une fonctionnalité disponible avec les hyperliens est la capacité de déterminer où un lien s'ouvre quand on clique dessus. En règle générale, les liens seront ouverts dans la fenêtre à partir de laquelle on aura cliqué. Cependant, les liens peuvent également être ouverts dans de nouvelles fenêtres.

Pour déclencher l'action d'ouverture d'un lien dans une nouvelle fenêtre, on va utiliser l'attribut `target` avec pour valeur `_blank`. L'attribut `target` détermine exactement où le lien sera affiché, et la valeur `_blank` spécifie une nouvelle fenêtre.

Pour ouvrir `http://google.com/` dans une nouvelle fenêtre, le code sera

```
<a href ="http://google.com/" target="_blank">Google</a>
```


Lier aux parties de la même page, l'ancrage

Périodiquement nous allons voir des liens hypertextes qui pointent vers une partie de la page où le lien apparaît. Un exemple courant de ces liens sur une même page sont les liens "Retour en haut page" qui renvoient un utilisateur vers le haut de la page en cours.

Nous pouvons créer un lien sur la page en définissant d'abord un attribut `id` sur l'élément vers lequel nous voulons créer lien, puis l'on va utiliser la valeur de cet `id` au sein d'un élément `href` comme attribut d'ancrage.

Utilisons le lien "Retour au début" comme exemple. Nous pouvons placer un attribut `id` avec la valeur "top" sur l'élément `<body>`. Maintenant nous pouvons créer un élément d'ancrage qui prendra pour valeur de l'attribut `href` l'identifiant top précédé du signe dièse "`#top`" pour créer le lien vers le début de l'élément `<body>`.

Notre code pour cette même page ressemblera à ce qui suit:

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains the following HTML code:

```
<body id="top">
    ...
    <a href="#top">Retour au début</a>
    ...
</body>
```

En pratique

Il est temps de faire de notre site « Cours HTML » un site web complet avec de multiples pages, qui seront reliées entre elles en utilisant des hyperliens.

Nous allons commencer par faire un lien sur notre texte « Cours HTML » dans la balise `<h1>` de notre `<header>` créant ainsi un lien vers la page `index.html`. Parce que nous sommes déjà sur la page `index.html`, cela peut sembler un peu bizarre, mais comme l'entête sera répliqué sur toutes les autres pages, créer dès maintenant un lien vers la page d'accueil prend tout son sens.

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains the following HTML code:

```
<h1>
    <a href="index.html">Cours HTML/CSS</a>
</h1>
```

Afin de naviguer dans les différentes pages, nous allons ajouter un menu de navigation en utilisant l'élément `<nav>` au sein de notre `<header>`.

Nous allons créer les entrées des pages suivantes: *Programme*, *Horaires*, *Lieu*, et *S'inscrire* à la suite de notre page d'accueil.

Nous devons donc créer des liens pour chacune d'entre elles dans le menu.

```

<header>
...
  <nav>
    <a href="index.html">Accueil</a>
    <a href="programme.html">Programme</a>
    <a href="horaires.html">Horaires</a>
    <a href="carte.html">Lieu</a>
    <a href="inscrire.html">S'inscrire</a>
  </nav>
</header>

```

Ajoutons également le même menu de navigation de notre élément `<header>` dans notre `<footer>` pour faciliter la navigation.

```

<footer>
...
  <nav>
    <a href="index.html">Accueil</a>
    <a href="programme.html">Programme</a>
    <a href="horaires.html">Horaires</a>
    <a href="carte.html">Lieu</a>
    <a href="inscrire.html">S'inscrire</a>
  </nav>
</footer>

```

Dans la `<section>` qui introduit notre cours, juste en dessous de notre entête, nous devrions également inclure un lien pour s'inscrire à ce cours. Placer un lien en dessous du paragraphe.

```

<section>
...
  <a href="inscrire.html">S'enregistrer maintenant</a>
</section>

```

Nous ne pouvons pas oublier d'ajouter des liens vers toutes les sections correspondantes à chacun de nos autres pages.

A l'intérieur de chaque section, nous allons entourer nos deux éléments `<h3>` et `<h4>` dans un élément d'ancrage reliant à la page adéquat.

```
1  <section>
2    <a href="programme.html">
3      <h3>Au programme...</h3>
4      <h4>...des kilomètres de code</h4>
5    </a>
6    <p>Le cours HTML CSS le plus complet à l'ouest du Pecos !</p>
7  </section>
8  <section>
9    <a href="horaires.html">
10     <h3>Horaires</h3>
11     <h4>...des heures de code</h4>
12   </a>
13   <p>Et des heures de code...</p>
14 </section>
15 <section>
16   <a href="carte.html">
17     <h3>Y aller...</h3>
18     <h4>...des kilomètres tout court</h4>
19   </a>
20   <p>Comment venir aux cours HTML/CSS</p>
21 </section>
```

Maintenant nous devons créer nos nouvelles pages. Créons nos fichiers `programme.html`, `horaire.html`, `carte.html` et `inscrire.html`. Ces fichiers seront dans le même dossier que le fichier `index.html`.

Pour faire en sorte que toutes nos pages aient le même aspect, nous allons créer ces nouveaux fichiers avec la même structure du document `<header>` et `<footer>` que le fichier `index.html`.

C'est officiel, nous ne travaillons plus avec une seule page mais bien avec un *site complet* !