

# HTML5

## Une brève histoire du Web

Il n'est pas nécessaire de faire un historique exhaustif de l'Internet, mais nous allons simplement indiquer quelques dates clés. Vous pourrez retrouver ces jalons à cette URL : <https://www.w3.org/History.html>

Vous le savez, l'Internet tel que nous le connaissons actuellement est une avancée technologique qui n'est pas si âgée que cela. En effet, c'est en mars 1989, au CERN (Centre Européen de Recherche Nucléaire) que Tim Berners-Lee rédige l'article « fondateur » de l'Internet. Dans son article titré « *Information Management : a proposal* », Tim Berners-Lee évoque dans son introduction la gestion de l'information à travers un système hypertexte distribué : « *It discusses the problems of loss of information about complex evolving systems and derives a solution based on a distributed hypertext system.* ».

En octobre 1990, Tim Berners-Lee travaille sur l'hypertexte avec un éditeur et un navigateur sur une station NeXT. Il donne le nom de World Wide Web à ce programme. En ce même mois d'octobre 1990, le belge Robert Cailliau rejoint l'équipe de Tim Berners-Lee et corédige la deuxième proposition nommée : « *WorldWideWeb: Proposal for a HyperText Project* ».

C'est à la fin de l'année 1990 que sont faites les démonstrations du premier serveur, du premier éditeur hypertexte et du premier navigateur. En décembre 1992 le premier serveur hors du CERN est installé à l'université de Stanford aux États-Unis. En 1993, le CERN rend libres les protocoles web.

Le 1er octobre 1994, le Word Wide Web Consortium (W3C) est créé au MIT (Massachusetts Institute of Technology). C'est en avril 1995 que l'INRIA (Institut National de Recherche en Informatique et Automatique) accueille le W3C en Europe et en septembre 1996 c'est au tour de l'université de Keio au Japon.

Si vous souhaitez avoir plus d'informations sur l'histoire de l'Internet et du Web, en 2004, pour le dixième anniversaire du W3C, ce dernier publia une iconographie retracant cette évolution : <https://www.w3.org/2005/01/timelines/timeline-2500x998.png>

## L'évolution du HTML

Une fois les principes d'une navigation par liens hypertextes acquis, Tim Berners-Lee s'attela au langage qui devait être utilisé pour créer et lier les documents.

En 1991, il rédige les premiers brouillons (*draft* en anglais) du HTML et en juin 1993, un premier document technique spécifie le langage HTML : "*Hypertext Markup Language (HTML) A Representation of Textual Information and MetaInformation for Retrieval and Interchange*". Ce document historique est toujours visible à cette URL : <http://www.w3.org/MarkUp/draft-ietf-iiir-html-01.txt>

Le 8 novembre 1993, le HTML+ sort : [http://www.w3.org/MarkUp/HTMLPlus/htmlplus\\_1.html](http://www.w3.org/MarkUp/HTMLPlus/htmlplus_1.html). La version 2 du HTML sort le 22 septembre 1995 ([http://www.w3.org/MarkUp/html-spec/html-spec\\_toc.html](http://www.w3.org/MarkUp/html-spec/html-spec_toc.html)) et il est spécifié comme une application SGML.

En mars 1995, le HTML 3 est publié (<http://www.w3.org/MarkUp/html3/>) comme une «extension» du HTML 2. Mais cette version est rapidement supplantée par la recommandation du HTML 3.2 le 14 janvier 1997 (<http://www.w3.org/TR/REC-html32.html>).

La recommandation du HTML 4.01 est publiée le 24 décembre 1999. Pour le W3C, cette version est la dernière concernant le HTML. Le W3C ne voit plus l'avenir des pages web avec le HTML, mais avec le XML. Le HTML est «mort» pour le W3C. La suite de l'histoire lui donnera tort.

Pour pallier les limites du HTML, le W3C propose le XHTML qui est basé sur le XML et qui «corrige» les ambiguïtés du HTML. La première recommandation du XHTML sort le 26 janvier 2000 (<http://www.w3.org/TR/2000/REC-xhtml1-20000126/>) et son évolution plus stricte d'un point de vue syntaxique, le XHTML 1.1, est publiée le 31 mai 2001 (<http://www.w3.org/TR/2001/REC-xhtml11-20010531/>). Le W3C propose le XHTML 2 en brouillon (*working draft*) en août 2002 (<http://www.w3.org/TR/2002/WD-xhtml2-20020805/>). Cela devait être une version «pure» XML, mais elle était incompatible avec les contenus web existants! De ce fait, le XHTML 2 ne sortira jamais en recommandation et le 17 décembre 2010, le W3C ferme officiellement le groupe de travail du XHTML 2.

Comme nous l'avons évoqué dans l'avant-propos, la recommandation actuelle du HTML est la version 5.2 (<https://www.w3.org/TR/html52/>), publiée le 14 décembre 2017. Pour conclure, le W3C se penche déjà sur la version 5.3 et un premier brouillon a été publié le 9 août 2018 (<https://www.w3.org/TR/html53/>).

## I. Les navigateurs

### L'évolution des navigateurs

Pour commencer ce chapitre, vous trouverez quelques dates importantes sur l'histoire des navigateurs. C'est à la fin de l'année 1990 que Tim Berners Lee développa le premier éditeur et le premier navigateur web au sein du CERN. En 1994, Mark Andreessen fonde Mosaic Communications Corp., qui deviendra le futur Netscape. Puis tous les éditeurs de plateforme créent leur propre navigateur, comme Microsoft Internet Explorer (1995) et Apple Safari (2003). Ensuite, des éditeurs indépendants arrivent et bousculent le marché, comme Mozilla Firefox en novembre 2004 et Opera qui devient gratuit en 2005.

Actuellement, sur les plateformes courantes, l'offre se limite à quatre acteurs principaux : Microsoft Edge, Google Chrome, Mozilla Firefox et Apple Safari. Mais bien sûr, il existe bien d'autres navigateurs web. Les supports mobiles devenant de plus en plus importants dans la vie de tous les jours, que ce soit personnellement ou professionnellement, tous les éditeurs ont publié une version mobile de leur navigateur. Entre ces différents éditeurs, la course à l'innovation est assez importante, ce qui fait que de nouvelles versions sont très régulièrement publiées. Dans ce livre, les versions d'octobre 2018 sont utilisées.

## Les outils de développement

Les outils destinés aux développeurs sont proposés nativement dans les quatre principaux navigateurs susnommés.

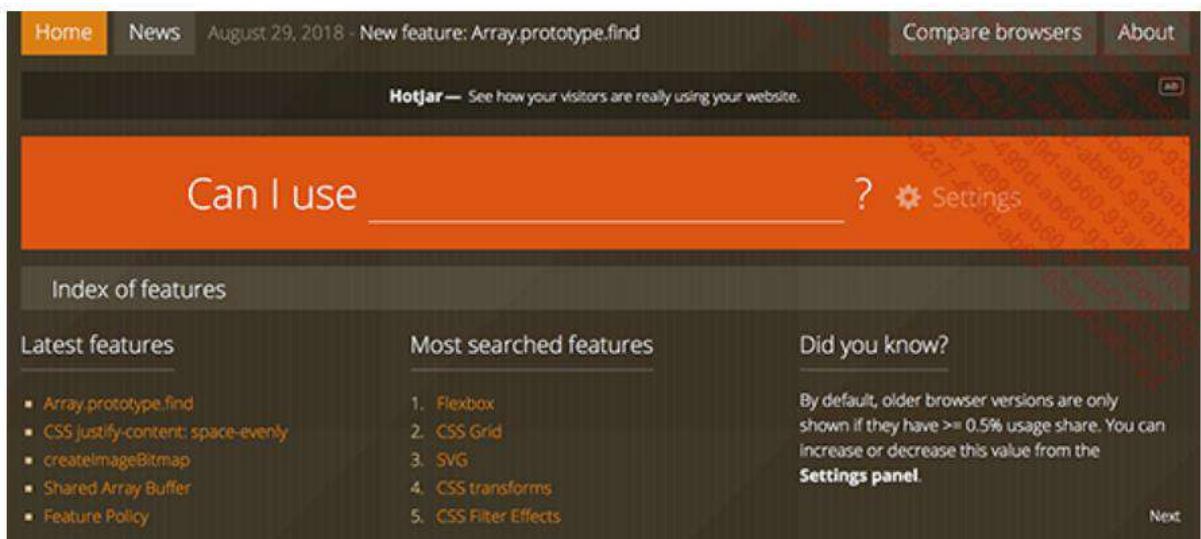
- Dans Microsoft Edge, allez dans le menu des options à droite et choisissez Outils de développement F12.
- Dans Google Chrome pour Windows, allez dans le menu des options à droite et choisissez Outils de développement. Dans Google Chrome pour macOS, allez dans le menu Afficher - Options pour les développeurs et vous avez accès aux items : Code source, Outils de développement et Console JavaScript.
- Dans Mozilla Firefox, allez dans le menu Outils - Développement web. Vous accédez alors à de nombreux items dédiés aux débogages des pages web. Notez que Mozilla propose Firefox Developer Edition qui est un navigateur spécialement dédié au développement des sites web, proposant de très nombreux outils dédiés à cette tâche : <https://www.mozilla.org/fr/firefox/developer/>
- Enfin, dans Apple Safari, vous devez d'abord aller dans les Préférences, dans l'onglet Avancées et cocher l'option Afficher le menu Développement dans la barre des menus. Une fois ceci fait, vous affichez le menu Développement avec ses très nombreux items dédiés aux débogages des pages web.

# La compatibilité des navigateurs

Nous venons de l'évoquer dans le titre précédent, c'est le W3C qui propose et ce sont les navigateurs qui disposent. Sachez que pour les CSS, les éditeurs des navigateurs peuvent proposer au W3C leurs propres propriétés, pour une éventuelle standardisation.

Pour le HTML5, il n'y a, pour ainsi dire, pas de problèmes de reconnaissance et d'interprétation, la norme est officialisée en recommandation depuis octobre 2014.

Pour les CSS 3, chaque module est développé à son propre rythme et son évolution est indiquée par l'intermédiaire des statuts évoqués dans le premier chapitre. Les navigateurs intègrent ces nouvelles propriétés CSS assez régulièrement. Pour connaître la compatibilité des propriétés CSS, la meilleure solution est de vous rendre régulièrement sur le site Can I Use : <http://caniuse.com>. Pour chaque propriété, dans un tableau clair, vous pourrez voir sa compatibilité avec les différentes versions des principaux navigateurs.



## II. Les bonnes pratiques

### Séparer le contenu de la mise en forme

Nous l'avons vu précédemment, à la sortie du HTML 4, le W3C a proposé les CSS 1. L'objectif était clair : séparer le contenu de la mise en forme et de la mise en page. Chaque langage a son objectif bien défini : le HTML décrit la

structure et le contenu des pages web et les CSS permettent de les mettre en forme et en page.

En travaillant de la sorte, vous n'aurez que des avantages :

- bien séparer les deux langages,
- avoir un code plus propre, plus rigoureux et plus lisible,
- séparer la gestion du contenu de la mise en forme et de la mise en page,
- centraliser la mise en forme et la mise en page en CSS,
- homogénéiser la mise en forme et la mise en page en CSS,
- avoir des mises à jour des CSS facilitées et rapides.

Dans la création des pages web, il faudra donc « éviter », autant que faire se peut, de « mélanger » la structure, le contenu et la mise en forme ; le code HTML et le code CSS.

## Utiliser une structure sémantique

Le HTML5 est un langage qui est parfaitement sémantique. Chaque élément HTML est fait pour contenir un type de contenu. Pour avoir un code propre, lisible, valide et accessible, vous devez respecter cette structure sémantique.

Voici quelques exemples de bonnes utilisations :

- L'élément `<p>` est fait pour contenir du texte courant dans des paragraphes.
- L'élément `<h1>` est fait pour afficher des titres de premier niveau, pour les titres les plus importants des pages.
- L'élément `<dl>` est fait pour concevoir des listes de définitions.

Dans votre code, utilisez à bon escient les éléments HTML et utilisez-les dans le cadre de leur définition.

## Optimiser le code et organiser vos fichiers

En tant que développeur, vous savez qu'il faut optimiser et bien organiser votre code. Voici quelques conseils de bon sens :

- Dans vos pages web, veillez à bien indenter les lignes de code. Cela sera toujours plus facile à reprendre par la suite, que ce soit pour vous ou une autre personne.

- Les commentaires sont indispensables pour bien expliquer votre code. À nouveau, c'est vous faciliter la tâche ou celles des personnes qui reprendront vos pages plus tard.
- Essayez de nommer les sélecteurs CSS de manière intelligible et logique. Il est toujours plus facile et rapide de reprendre du code bien créé.
- Pour l'organisation des fichiers, prenez la bonne habitude de créer des dossiers par type de fichier utilisé dans vos développements. Il est très classique d'avoir un dossier css pour tous les fichiers feuilles de style CSS, un dossier js pour tous les fichiers JavaScript et un dossier img pour les médias de type images.
- Et bien sûr, faites régulièrement des sauvegardes et utilisez, pourquoi pas, un outil de gestion des versions.

## Un exemple d'une page bien formée

Nous allons terminer ce chapitre en étudiant une courte et très simple page web "mal codée" et voir ce qu'il faut faire.

Voici le code de la page :

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<style>
.titre {
font-size: 16pt;
font-weight: bold;
}
</style>
</head>
<body>
<p class="titre">Maecenas faucibus mollis interdum</p>
<p>Curabitur blandit tempus porttitor...</p>
<p><center>Aenean eu leo quam. Pellentesque...</center></p>
Cras mattis consectetur purus sit amet...
```

```

<p>&nbsp;</p><p>&nbsp;</p>

<h2>Nullam quis risus eget urna mollis ornare vel eu leo</h2>

<p>Sed posuere consectetur est at lobortis...</p>

<table>

<tr>

<td></td>
<td></td>
<td></td>

</tr>

</table>

</body>

</html>

```

Voyons maintenant pourquoi cette page peut être qualifiée de "pas correctement codée".

- Il n'y a aucune indentation. Le code est donc peu lisible, mal organisé et mal structuré.
- Dans l'élément `<head>`, il n'y a pas l'élément `<title>` qui est pourtant indispensable.
- Le premier titre de la page est placé dans un élément `<p>`, fait pour contenir du texte courant dans un paragraphe.
- Pour ce titre, la mise en forme est faite en CSS (taille de caractère et mise en gras). Il fallait utiliser l'élément sémantique `<h1>`.
- Le deuxième paragraphe utilise l'élément HTML `<center>` qui est obsolète.
- Le troisième paragraphe de texte n'est placé dans aucun conteneur. Il ne peut donc être ciblé d'aucune manière, pour une mise en forme par exemple.
- Le deuxième titre, bien placé dans un élément `<h2>`, est espacé du texte précédent par des paragraphes `<p>` contenant une espace insécable `&nbsp;`. C'est du "pur bricolage", il faut utiliser les propriétés CSS faites pour cela.
- Les trois images sont placées dans un tableau. Les tableaux sont faits pour afficher des données tabulaires, pas pour faire de la mise en page.

Voici maintenant la même page, mais avec une syntaxe nettement plus rigoureuse :

```
<!DOCTYPE html>

<html>
<head>

    <title>Ma petite page web</title>
    <meta charset="UTF-8" />
    <style>

        .paragraphe-centre {
            text-align: center;
        }

        .espace-avant {
            margin-top: 68px;
        }

        .img-flotte-gauche {
            float: left;
            margin-right: 10px;
        }

    </style>
</head>
<body>

    <h1>Maecenas faucibus mollis interdum</h1>
    <p>Curabitur blandit tempus porttitor...</p>
    <p class="paragraphe-centre">Aenean eu leo quam.
    Pellentesque...</p>
    <p>Cras mattis consectetur purus sit amet...</p>
    <h2 class="espace-avant">Nullam quis risus eget urna mollis
    ornare vel eu leo</h2>
    <p>Sed posuere consectetur est at lobortis...</p>
    <div>
        <p>
        
        
    </p>

```

```
</div>  
</table>  
</body>  
</html>
```

Voici les corrections :

- Le code est bien indenté.
- La page possède bien l'élément `<title>`.
- Le premier titre utilise bien l'élément HTML sémantique `<h1>`. Il n'est donc pas nécessaire de le surcharger avec une mise en forme CSS.
- Le deuxième paragraphe utilise la propriété CSS adéquate pour centrer le paragraphe.
- Le troisième paragraphe est bien placé dans un élément HTML, `<p>` dans cet exemple.
- Le deuxième titre `<h2>` est bien espacé du texte précédent avec l'utilisation de la propriété CSS adéquate.
- Les trois images sont bien mises en page avec, à nouveau, les propriétés CSS adéquates.

## Un exemple d'une page bien formée

Nous allons terminer ce chapitre en étudiant une courte et très simple page web "mal codée" et voir ce qu'il faut faire.

Voici le code de la page :

```
<!DOCTYPE html>  
  
<html>  
  
<head>  
  
<meta charset="UTF-8" />  
  
<style>  
  
.titre {  
font-size: 16pt;  
font-weight: bold;  
}
```

```

</style>
</head>
<body>


Maecenas faucibus mollis interdum



Curabitur blandit tempus porttitor...



<center>Aenean eu leo quam. Pellentesque...</center>



Cras mattis consectetur purus sit amet...



&nbsp;</p><p>&nbsp;</p>



## Nullam quis risus eget urna mollis ornare vel eu leo



Sed posuere consectetur est at lobortis...



|  |  |  |
|--|--|--|
|  |  |  |
|--|--|--|


</body>
</html>

```

Voyons maintenant pourquoi cette page peut être qualifiée de "pas correctement codée".

- Il n'y a aucune indentation. Le code est donc peu lisible, mal organisé et mal structuré.
- Dans l'élément `<head>`, il n'y a pas l'élément `<title>` qui est pourtant indispensable.
- Le premier titre de la page est placé dans un élément `<p>`, fait pour contenir du texte courant dans un paragraphe.
- Pour ce titre, la mise en forme est faite en CSS (taille de caractère et mise en gras). Il fallait utiliser l'élément sémantique `<h1>`.
- Le deuxième paragraphe utilise l'élément HTML `<center>` qui est obsolète.
- Le troisième paragraphe de texte n'est placé dans aucun conteneur. Il ne peut donc être ciblé d'aucune manière, pour une mise en forme par exemple.

- Le deuxième titre, bien placé dans un élément `<h2>`, est espacé du texte précédent par des paragraphes `<p>` contenant une espace insécable `&nbsp;`. C'est du "pur bricolage", il faut utiliser les propriétés CSS faites pour cela.
- Les trois images sont placées dans un tableau. Les tableaux sont faits pour afficher des données tabulaires, pas pour faire de la mise en page.

Voici maintenant la même page, mais avec une syntaxe nettement plus rigoureuse :

```

<!DOCTYPE html>

<html>
  <head>
    <title>Ma petite page web</title>
    <meta charset="UTF-8" />
    <style>
      .paragraphe-centre {
        text-align: center;
      }
      .espace-avant {
        margin-top: 68px;
      }
      .img-flotte-gauche {
        float: left;
        margin-right: 10px;
      }
    </style>
  </head>
  <body>
    <h1>Maecenas faucibus mollis interdum</h1>
    <p>Curabitur blandit tempus porttitor...</p>
    <p class="paragraphe-centre">Aenean eu leo quam.
      Pellentesque...</p>
    <p>Cras mattis consectetur purus sit amet...</p>
    <h2 class="espace-avant">Nullam quis risus eget urna mollis
  
```

```
ornare vel eu leo</h2>

<p>Sed posuere consectetur est at lobortis...</p>

<div>

    <p>
        
        
    </p>

</div>

</table>

</body>

</html>
```

Voici les corrections :

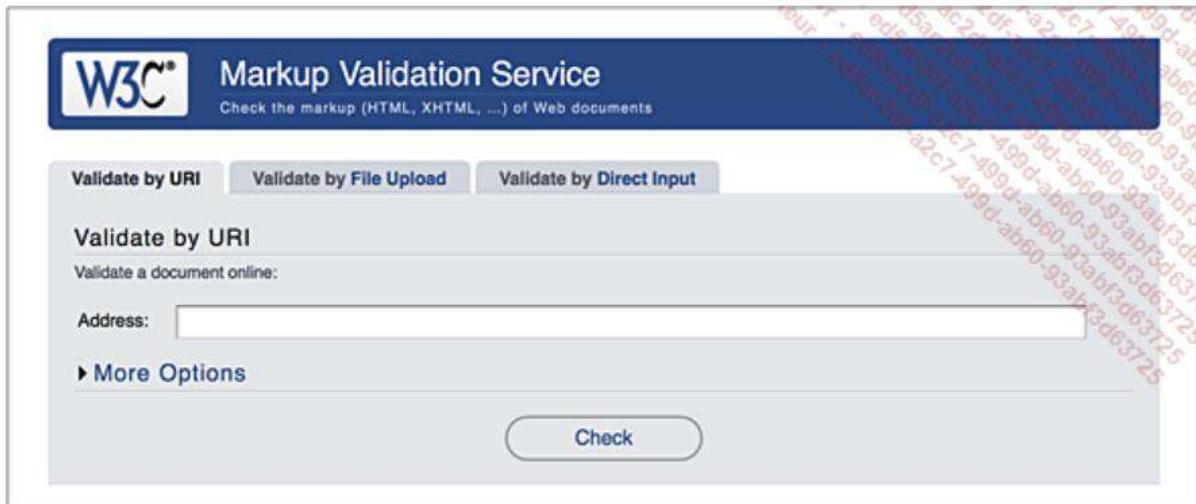
- Le code est bien indenté.
- La page possède bien l'élément `<title>`.
- Le premier titre utilise bien l'élément HTML sémantique `<h1>`. Il n'est donc pas nécessaire de le surcharger avec une mise en forme CSS.
- Le deuxième paragraphe utilise la propriété CSS adéquate pour centrer le paragraphe.
- Le troisième paragraphe est bien placé dans un élément HTML, `<p>` dans cet exemple.
- Le deuxième titre `<h2>` est bien espacé du texte précédent avec l'utilisation de la propriété CSS adéquate.
- Les trois images sont bien mises en page avec, à nouveau, les propriétés CSS adéquates.

## Valider le code de vos pages

Il est important de valider le code HTML de vos pages. En effet, vos clients s'attendent à recevoir un site ou une application web parfaitement fonctionnelle, sans erreur pouvant détériorer son utilisation. Et n'oubliez pas que les assistants vocaux pour les personnes malentendantes se basent sur des pages valides.

Pour toutes ces raisons, vous pouvez utiliser des validateurs de code. Certains éditeurs de code en intègrent un nativement, d'autres nécessitent une installation d'une extension.

Mais vous pouvez aussi utiliser les validateurs en ligne, comme celui du W3C qui est le plus rigoureux. Voici son URL : <https://validator.w3.org>.



Vous pouvez choisir de valider une page HTML publiée (onglet Validate by URI), d'importer une de vos pages (onglet Validate by File Upload), ou de copier/coller du code de votre page (onglet Validate by Direct Input).

### III. Les éléments html

#### Bien utiliser le HTML

Le langage HTML est un langage à balises, comme son nom l'indique : HyperText Markup Language, Langage à Balises Hypertextes. Notez aussi que le HTML est originellement une application SGML (*Standard Generalized Markup Language*).

Le langage HTML est saisi dans un document de type texte, ayant pour extension .html. Le HTML est interprété par un «agent utilisateur» pour reprendre la terminologie officielle (*user agent*). Ces agents utilisateurs du HTML sont la plupart du temps des navigateurs web, mais il existe d'autres formes d'applications capables d'interpréter le HTML, comme les lecteurs d'écran audio, les robots d'indexation des moteurs de recherche, mais aussi des systèmes embarqués dans des appareils électroniques.

L'objectif du HTML est de décrire la structure des pages web et d'indiquer le contenu sémantique de chaque élément constitutif de ces pages. Le HTML décrit le contenu des pages web à l'aide d'éléments HTML qui sont constitués de balises. Chaque élément est dédié à l'affichage d'un type de contenu donné. Vous avez des éléments HTML qui affichent les titres, les images, les tableaux, les formulaires...

Le HTML est donc bien un langage sémantique : chaque élément doit être utilisé dans le cadre de sa définition et les agents utilisateurs attendent un contenu donné pour chaque élément. Par exemple, l'élément HTML `<p>` est fait pour contenir un paragraphe de texte et non un tableau, l'élément HTML `<img>` est fait pour contenir une image et non un champ de formulaire. Pour que vos pages web soient valides et bien interprétées par les navigateurs, vous devez respecter cette sémantique.

## Les balises et les contenus

Chaque élément HTML est composé de plusieurs parties constitutives.

La première partie est la balise d'ouverture. Cette balise commence par le caractère `<` et est immédiatement suivie par le nom de l'élément et se termine par le caractère `>`. L'élément qui permet d'insérer un paragraphe de texte est nommé `p`, la syntaxe de sa balise d'ouverture est la suivante : `<p>`.

La deuxième partie concerne les contenus rédactionnels. C'est-à-dire les éléments HTML qui contiennent du texte, comme les titres, les paragraphes, les citations... Le texte est alors écrit de manière usuelle, sans aucune balise spécifique.

La plupart des éléments HTML ont une balise de fermeture. La syntaxe reprend le principe de la balise d'ouverture, mais avec en plus le caractère `/` qui précède le nom de l'élément. Ce caractère indique la fin de l'élément. Si nous reprenons notre exemple de paragraphe, voici la balise de fermeture : `</p>`.

Voici un exemple complet pour l'élément HTML insérant un paragraphe :

```
<p>Le texte de mon paragraphe.</p>
```

Il est important de noter que certains éléments HTML n'ayant pas de contenu rédactionnel ou pas d'éléments imbriqués n'ont logiquement pas de balise de fermeture. Prenons pour exemple l'élément `<hr>` qui permet d'insérer une ligne horizontale de séparation. Il n'y a pas de contenu rédactionnel, donc nous n'avons pas de balise de fermeture. Il en est de même avec l'élément `<img>` qui permet d'insérer une image. Dans ce cas, la balise d'ouverture est qualifiée d'autofermante.

## Les attributs des éléments

Les attributs permettent de modifier le comportement standard des éléments HTML. Les éléments HTML peuvent ne contenir aucun attribut, ou un seul et parfois plusieurs. Certains attributs sont obligatoires, d'autres sont facultatifs. La plupart des attributs ont des valeurs, mais pas tous. Les attributs qui ne nécessitent pas de valeur sont qualifiés de booléens. Dans tous les cas, les attributs se placent dans la balise d'ouverture des éléments.

Prenons comme premier exemple l'attribut facultatif qui permet d'identifier de manière unique un élément HTML. C'est l'attribut `id`. Cet attribut doit avoir une valeur qui est indiquée entre guillemets (non obligatoires, mais très fortement conseillés) et qui est précédée par le signe `=`. Voici un exemple :

```
<p id="introduction">Le contenu de mon paragraphe.</p>
```

Prenons comme deuxième exemple un attribut obligatoire. Pour insérer une image, nous utilisons l'élément `<img>` et son attribut obligatoire `src` qui permet d'indiquer le chemin d'accès à la source de l'image. Voici une syntaxe simple :

```

```

Si un élément HTML possède plusieurs attributs, ils sont séparés par un simple espace. Voici un exemple avec de nouveau l'élément `<img>` :

```

```

Notez que chaque élément HTML peut avoir des attributs qui lui sont propres. Mais sachez qu'il existe de très nombreux attributs universels (*Global attributes* en anglais) qui peuvent s'appliquer à tous les éléments HTML. Voici une URL où vous pourrez trouver la liste de ces attributs universels : [https://developer.mozilla.org/fr/docs/Web/HTML/Attributs\\_universels](https://developer.mozilla.org/fr/docs/Web/HTML/Attributs_universels)

## Le bon usage de la syntaxe

Le langage HTML est un langage assez permissif, mais il convient de respecter certaines règles pour offrir aux différents intervenants un code propre, lisible et valide.

Vous pouvez parfaitement utiliser des majuscules ou des minuscules pour saisir le nom des éléments et des attributs HTML. Les syntaxes `<p>Mon texte.</p>` et `<P>Mon texte.</P>` sont équivalentes. Mais l'usage prévaut l'utilisation des minuscules.

Certains éléments HTML ont une balise de fermeture optionnelle, comme l'élément `<p>`. Mais pour les mêmes raisons que précédemment, fermer toujours les éléments de contenu par leur balise de fermeture.

Les valeurs des attributs peuvent ne pas être indiquées entre guillemets. Mais à nouveau, privilégiez toujours les guillemets.

## L'imbrication des éléments

Les éléments HTML permettent de structurer le contenu de vos pages web. Cette structuration va de pair avec l'imbrication des éléments HTML. Par exemple dans un article, inséré avec l'élément `<article>`, nous pouvons imbriquer un élément d'en-tête `<header>`, un ou plusieurs paragraphes `<p>` et un pied de page `<footer>`. Nous avons alors une hiérarchie des éléments : `<header>`, `<p>` et `<footer>` sont imbriqués dans `<article>`. Ces éléments sont les enfants de l'article qui est leur parent. S'il y a plusieurs enfants `<p>`, ils sont frères.

Dans les paragraphes `<p>`, nous pouvons parfaitement appliquer une mise en forme sémantique avec l'élément `<strong>` qui permet d'appliquer une forte emphase. Dans ce cas, l'élément `<strong>` est enfant de l'élément `<p>`.

Avec ces exemples, nous voyons, en simplifiant, deux types d'éléments HTML. Les éléments de structure et les éléments de mise en forme du texte. Cette notion est héritée du HTML 4. Dans cette recommandation, les éléments HTML étaient typés en "*block*" et en "*inline*". Les éléments de type bloc (*block*) permettent de structurer la page, avec des éléments comme `<div>`, `<p>`, `<h1>`. Par défaut, les navigateurs doivent afficher ces éléments sur toute la largeur disponible et ils doivent commencer sur une nouvelle ligne. Les éléments en ligne (*inline*) permettent de mettre en forme le texte. Les navigateurs doivent donc les afficher dans la même ligne et ils peuvent s'enchaîner les uns aux autres. Bien sûr, ces deux types sont les plus utilisés et il en existe bien d'autres. Voici une URL qui vous liste tous les types d'affichage : [https://www.w3schools.com/cssref/pr\\_class\\_display.asp](https://www.w3schools.com/cssref/pr_class_display.asp).

Donc, en ce qui concerne l'imbrication des éléments, les éléments de type bloc peuvent contenir d'autres éléments de type bloc, des éléments de type en ligne et du texte. Les éléments de type en ligne peuvent contenir d'autres éléments en ligne ou du texte, mais pas d'éléments de type bloc.

Prenons un exemple très simple. Nous devons saisir un texte dont certains mots doivent être mis en évidence. Nous allons donc utiliser un élément `<p>` de type bloc, contenant du texte, dont un mot sera mis en évidence avec l'élément `<strong>` de type en ligne.

```
<p>Mon texte est mis en <strong>évidence</strong>  
avec une emphase forte.</p>
```

L'imbrication inverse n'est pas possible, c'est une évidence :

```
<strong>Mon texte est mis en <p>évidence</p>  
avec une emphase forte.</strong>
```

Ces notions de type *block* ([https://developer.mozilla.org/fr/docs/Web/HTML/Éléments\\_en\\_bloc](https://developer.mozilla.org/fr/docs/Web/HTML/Éléments_en_bloc)) et *inline* ([https://developer.mozilla.org/fr/docs/Web/HTML/Éléments\\_en\\_ligne](https://developer.mozilla.org/fr/docs/Web/HTML/Éléments_en_ligne)), et les autres types étaient parfaitement bien définis dans le HTML 4. Mais avec le HTML5 ces différences se sont un peu estompées. Par exemple, il est autorisé d'avoir un élément enfant `<p>`, de type bloc, dans un élément parent `<a>`, de type en ligne.

## Les commentaires

Comme dans tout langage informatique, il est vivement conseillé de commenter son code, que ce soit pour vous ou pour un autre développeur qui viendra reprendre vos pages. Les commentaires peuvent se placer n'importe où dans la page. Voici la syntaxe :

```
<!-- Le texte de mon commentaire -->
```

## IV. La structure des pages

### La structure générale des pages web

La totalité d'une page web est insérée dans l'élément `<html>`. Cet élément est qualifié d'élément racine de la page.

Nous avons ensuite les deux parties de contenu : l'en-tête et le corps. L'en-tête, avec l'élément `<head>`, permet de définir les propriétés globales des

pages, sachant que chaque page aura des propriétés différentes. Le corps de la page, élément `<body>`, permet d'insérer tout le contenu de la page qui pourra être affiché. Ces deux éléments, `<head>` et `<body>`, sont frères et ils sont des enfants de l'élément `<html>`.

Nous pouvons donc avoir cette structure minimale :

```
<!doctype html>
<html>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

## La déclaration doctype

Nous l'avons indiqué dans un chapitre précédent, le HTML est une application SGML. Ainsi, il faut que la toute première ligne d'une page web contienne l'indication du langage à balises utilisé. Ceci est fait pour le navigateur. La syntaxe est extrêmement simple :

```
<!doctype html>
```

Attention, notez bien que cette déclaration du type de document n'est pas un élément HTML.

## L'élément `<html>`

L'élément `<html>` est l'élément racine des pages HTML. La balise d'ouverture, `<html>`, fait immédiatement suite à la déclaration de type de document et la balise de fin, `</html>`, termine la page.

Parmi les attributs globaux, l'utilisation de l'attribut `lang`, non obligatoire, est très fortement recommandée. Cela permet tout simplement d'indiquer au navigateur quelle est la langue utilisée dans la page web. Cet attribut est exploité pour l'indexation par les moteurs de recherche et pour les navigateurs à synthèse vocale pour les personnes handicapées.

Voici la syntaxe usuelle :

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

## L'élément <head>

### 1. Les éléments enfant de l'en-tête

L'élément <head> est vraiment très important. Il contient toute une série de propriétés essentielles pour la page, sachant que ces propriétés ne sont indiquées nulle part ailleurs qu'ici.

Ces propriétés sont renseignées avec ces éléments enfants :

- <meta> : permet de renseigner plusieurs métadonnées sur le document. Vous pouvez n'avoir aucun élément, un ou plusieurs.
- <title> : c'est le seul élément qui soit obligatoire. Il donne le titre du document. Vous ne pouvez n'avoir qu'un seul élément de titre par page.
- <link> : donne les liens vers des ressources extérieures à la page, comme des fichiers de feuilles de styles CSS. Vous pouvez n'avoir aucun élément, un ou plusieurs.
- <style> : permet de déclarer les règles de style CSS incorporées dans la page. Vous pouvez n'avoir aucun élément, un ou plusieurs.
- <script> : permet de définir les scripts incorporés dans la page. Vous pouvez n'avoir aucun élément, un ou plusieurs.

### 2. Les éléments <meta>

L'élément <meta> permet de renseigner plusieurs métadonnées, dont certaines sont importantes comme l'encodage des caractères.

Il est important d'indiquer l'encodage des caractères juste après la balise d'ouverture <head>, car cet encodage va concerner tous les autres éléments à venir. Actuellement, l'encodage utilisé est l'UTF-8 (*Universal Character, Set Transformation Format*, sur 8 bits). Voici sa syntaxe :

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

Vous pouvez aussi utiliser des métadonnées pour renseigner des informations utilisées par les robots d'indexation des moteurs de recherche :

```
<meta name="description" content="La description de ma page">
```

```
<meta name="keywords" content="sites web, conception, html, css,  
javascript">  
<meta name="author" content="Christophe AUBRY">  
<meta name="generator" content="Mon logiciel de création">
```

### 3. L'élément <title>

L'élément <title> est obligatoire. Le contenu textuel de cet élément est repris pour être affiché dans la barre de titre des fenêtres ou dans les onglets des navigateurs. Il est aussi utilisé comme lien, dans les résultats des moteurs de recherche. Son contenu n'est donc pas à négliger, il faut bien travailler les mots qui y seront placés.

### 4. L'élément <link>

L'élément <link> permet de créer des liens vers des ressources externes à la page, comme vers des fichiers .css. Il permet aussi d'afficher une icône dans la barre d'adresse du navigateur. Voici les deux exemples de syntaxe :

```
<link rel="stylesheet" type="text/css" src="mes-styles.css">  
<link rel="icon" type="image/gif" href="icone-site.gif">
```

### 5. L'élément <style>

L'élément <style> permet de déclarer des styles CSS qui ne seront applicables qu'au sein de cette page. Voici un exemple de syntaxe :

```
<style>  
    .auteur {  
        color: #720868;  
        text-transform: uppercase;  
    }  
</style>
```

Notez qu'il n'est pas nécessaire d'indiquer l'attribut `type="text/css"` dans l'élément <style>, puisque les CSS sont considérées comme le type par défaut.

## 6. L'élément <script>

L'élément <script> permet de déclarer des scripts JavaScript qui ne seront applicables qu'au sein de cette page. Voici un exemple de syntaxe :

```
<script>
    alert ("Bonjour tout le monde !");
</script>
```

Notez qu'il n'est pas nécessaire d'indiquer l'attribut `type="text/javascript"` dans l'élément <script>, puisque le JavaScript est considéré comme le langage par défaut.

## L'élément <body>

L'élément <body> va contenir tous les éléments de contenu de la page web. Sa balise d'ouverture, <body>, est placée juste après la balise de fermeture de l'en-tête </head>. Sa balise de fermeture, </body>, est située juste avant celle du document, </html>.

## Exemple d'une structure simple

Voici un exemple d'une structure simpliste, minimale et valide d'une page web :

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Le titre de ma page</title>
    </head>
    <body>
        <p>Le contenu de ma page.</p>
    </body>
</html>
```

## V. Les conteneurs sémantiques

# Bien utiliser les conteneurs sémantiques

Pour concevoir vos pages, vous devez réfléchir en termes de « conteneur ». Un conteneur, comme son nom l'indique, inclut un contenu de type très varié.

Dans ces conteneurs, vous pourrez placer du texte, des images, des formulaires, des liens, des tableaux... Mais vous pourrez aussi avoir des conteneurs plus « petits » comme pour mettre en évidence un ou plusieurs mots, ou pour définir une cellule de tableau.

Les conteneurs servent aussi à structurer vos pages. Vous pourrez ainsi utiliser des conteneurs pour insérer un en-tête de page, une colonne latérale (une *sidebar* en anglais), un pied de page, une barre de navigation...

Vous l'avez compris, tous les contenus sont insérés dans des conteneurs. Chaque conteneur, en dehors des divisions (`<div>` et `<span>`), sont dédiés à recevoir un contenu spécifique. C'est pour cela qu'ils sont qualifiés de conteneurs sémantiques.

## L'élément `<div>`

L'élément `<div>` est un des conteneurs les plus anciens du HTML. Il permet de créer une division structurelle dans la page. Dans ces divisions, nous pouvons placer n'importe quel contenu et même d'autres conteneurs comme d'autres divisions `<div>`, des paragraphes, des listes... Ces divisions permettaient d'effectuer des mises en page à l'aide de conteneurs "neutres", c'est-à-dire sans contenu sémantique défini.

Le HTML5 a introduit de nouveaux conteneurs sémantiques qui ont réduit l'utilisation de l'élément `<div>`. Mais ce n'est pas parce que vous utilisez du HTML5 que vous devez abolir l'utilisation des boîtes `<div>` ! Elles sont toujours utilisables et ont toujours leur utilité. Les boîtes `<div>` sont souvent utilisées pour avoir des conteneurs "neutres" qui n'ont pas besoin d'avoir un sens sémantique précis.

## L'élément `<span>`

L'élément `<span>` est souvent utilisé, par exemple, pour avoir une division au sein d'un paragraphe de texte. Ceci est très pratique pour mettre en forme de manière particulière du texte dans un texte formaté d'une autre manière.

Voici un exemple précis : nous souhaitons mettre en fond gris et avec une fine bordure une partie d'un texte dans un paragraphe.

Voici le sélecteur CSS :

```
.fond-gris {  
    background-color: #eee;  
    padding: 0 5px;  
    border: 1px solid #333;  
}
```

Voici son application dans le code HTML :

```
<p>Donec ullamcorper nulla non metus auctor  
fringilla. Morbi leo risus, <span class="fond-gris">porta  
ac consectetur ac vestibulum</span> at eros. Donec sed odio...</p>
```

Voici l'affichage obtenu :

Donec ullamcorper nulla non metus auctor fringilla. Morbi leo risus, porta ac consectetur ac vestibulum at eros. Donec sed odio...

## L'élément <header>

L'élément <header> permet d'insérer une zone d'affichage pour les en-têtes. Ces en-têtes peuvent être utilisés à plusieurs endroits :

- au niveau de la page, c'est le classique en-tête de page, souvent placé en haut de l'écran, avec un logo, un slogan, une barre de navigation principale...
- au niveau des contenus, cela permet d'avoir une introduction au contenu qui va suivre, comme l'en-tête d'un article, par exemple.

Ce conteneur peut contenir tout type d'élément : des titres, des paragraphes, des liens...

Depuis le HTML 5.1, vous pouvez imbriquer des éléments <header> et <footer> dans un autre élément <header>, si les deux premiers éléments sont inclus dans un même élément parent.

Voici un exemple d'imbrication parfaitement valide :

```
<article>  
    <header>  
        <h2>Cras Vestibulum Sem Fermentum</h2>
```

```
<aside>
  <header>
    <h3>Inceptos Magna Vehicula Malesuada</h3>
    <p>Mollis Risus Sollicitudin Inceptos...</p>
  </header>
  <p>Sit Mattis Aenean Commodo...</p>
  <footer>
    <p>Parturient Pharetra Quam</p>
  </footer>
</aside>
</header>
<p>Adipiscing Ultricies Dapibus Mollis...</p>
</article>
```

## L'élément `<footer>`

L'élément `<footer>` permet d'insérer une zone d'affichage pour les pieds de page. Nous retrouvons la même sémantique que pour les en-têtes. Ces pieds de page peuvent être définis pour la page ou pour une autre zone d'affichage de celle-ci, comme un article. Notez que l'usage d'un `<footer>` n'implique pas forcément l'usage d'un `<header>`.

## L'élément `<aside>`

L'élément `<aside>` permet d'afficher un contenu lié à un contenu principal auquel il est associé. Cela correspond souvent aux très classiques barres latérales (*sidebar* en anglais), à des zones de composants d'interface (*widgets* en anglais), à des compléments sur des articles ou tout autre contenu rédactionnel.

Vous avez usuellement le contenu principal qui est affiché dans la partie centrale de la page, avec souvent, sur la droite, l'affichage du contenu associé avec l'élément `<aside>`.

## L'élément `<nav>`

L'élément `<nav>`, comme son nom le laisse supposer, est dédié à l'affichage d'une barre de navigation avec des liens hypertextes. Mais attention, ne vous sentez pas obligé de n'avoir qu'une seule zone de navigation par page. Vous pouvez créer autant d'éléments `<nav>` que vous avez de navigations différentes dans vos pages, à partir du moment où chacun d'entre eux est bien identifié. L'élément `<nav>` est plutôt dédié à la navigation principale du site, à la création de liens entre les pages du site.

Vous pouvez inclure une navigation principale `<nav>` dans un en-tête `<header>` et une navigation secondaire `<nav>` dans un pied de page `<footer>`, par exemple.

## L'élément `<main>`

L'élément `<main>` permet d'indiquer le contenu principal de la page. Ce contenu doit être unique et ne pas être répété dans la page. De plus, le W3C indique précisément son contexte d'utilisation : il ne doit pas être utilisé à l'intérieur, comme élément inclus, dans les éléments `<article>`, `<aside>`, `<footer>`, `<header>` ou `<nav>`.

## L'élément `<section>`

L'élément `<section>` permet de regrouper des éléments partageant une même thématique. Cela permet de regrouper dans un même élément un contenu structuré, avec son en-tête et son pied de page. L'utilisation de plusieurs éléments `<section>` distinguera plusieurs parties, plusieurs sections au sein d'une même page, avec d'autres éléments de structure imbriqués.

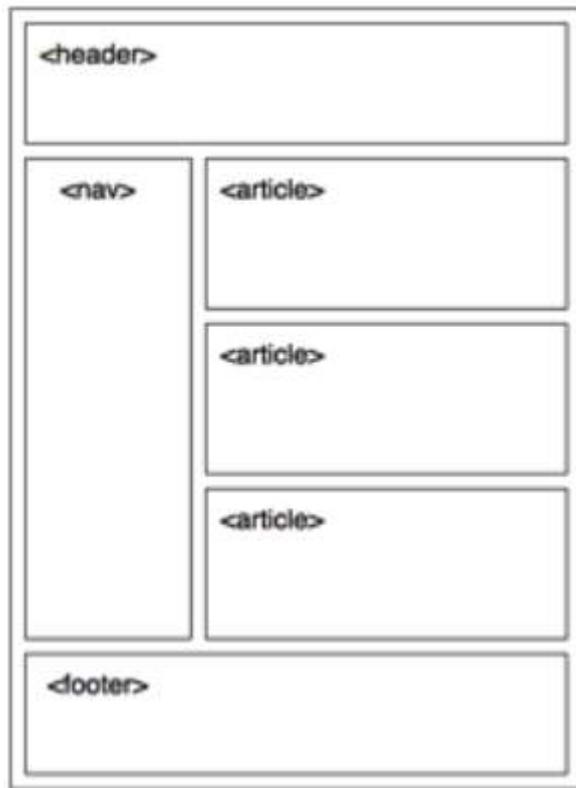
## L'élément `<article>`

L'élément `<article>` permet d'insérer un contenu autonome. Il est qualifié d'autonome car il peut être réutilisé ailleurs dans le site, sans que sa compréhension en soit affectée. L'usage le plus courant reprend le nom de l'élément : création d'articles de blog et d'actualités.

# Deux exemples de structure sémantique de page

## 1. Une structure sémantique simple

Voici la structure sémantique très simple d'un premier exemple :

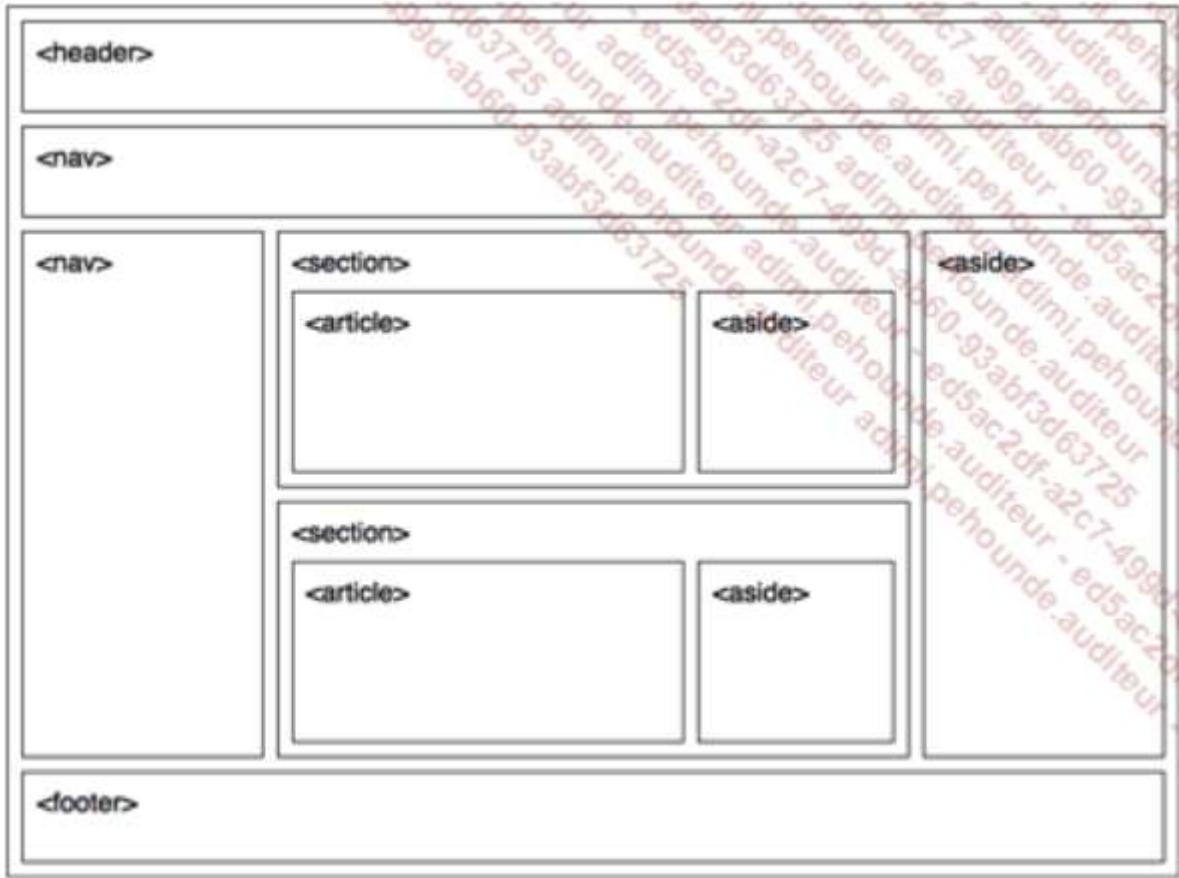


Nous avons :

- Un en-tête <header> dans la partie supérieure, avec un logo et un slogan par exemple.
- Une barre de navigation <nav> sur la partie gauche de la page.
- Toutes les actualités du site pourront être placées dans des éléments <article>.
- Enfin, le pied de page <footer> pourra contenir les mentions légales, les liens de contact, etc.

## 2. Une structure sémantique plus élaborée

Voici une deuxième structure plus élaborée :



Nous avons :

- Un en-tête `<header>` dans la partie supérieure.
- En dessous, une barre de navigation `<nav>` pour la navigation générale dans le site, pour naviguer de page en page.
- Sur la gauche, une deuxième boîte `<nav>` est faite pour la navigation secondaire, pour des liens liés directement au contenu de la page affichée.
- Sur la droite, un élément `<aside>` affiche des informations liées au contenu de la page, comme des liens promotionnels, des contenus en relation...
- Le contenu de la page est affiché dans deux éléments `<section>` permettant ainsi de bien différencier ces deux contenus. Chaque élément `<section>` contient un élément `<article>` pour le contenu rédactionnel et un élément `<aside>` pour afficher des éléments d'information supplémentaire liés à l'article (iconographie, liens...).
- Enfin, la page se termine par un pied de page `<footer>` pour afficher les informations légales, les conditions de vente, un lien de contact, un plan d'accès...

## Un exemple de structure sémantique d'un article

Voici un exemple d'un article, d'un contenu rédactionnel utilisant ces éléments de structure sémantique :



Nous avons :

- Un élément <article> comme conteneur général.
- Nos articles contiennent des en-têtes, des introductions, nous utilisons donc l'élément <header>. L'élément <header> contient le titre <h1> de l'article et son sous-titre <h2>.
- Le contenu rédactionnel de l'article est placé dans des éléments <p>. L'article contient des liens vers des compléments d'articles, listés dans une liste <ul>.
- L'article se termine par un pied de page <footer>, ou plutôt un pied d'article, avec la date de publication, la rubrique de l'article et le nom de l'auteur, par exemple.

## VI. Les conteneurs de texte

# Bien utiliser les conteneurs de texte

Après avoir vu les conteneurs de structure, attelons-nous aux conteneurs de texte. À nouveau, nous devons utiliser des conteneurs sémantiques pour insérer des types de textes précis dans nos pages web.

Notez que tous ces éléments sont de type `block`. Cela implique qu'ils s'affichent sur toute la largeur disponible dans leur conteneur parent, qui peut être l'élément `<body>` ou tout autre conteneur de type `block`. Donc le conteneur suivant, quel qu'il soit, s'affichera sur une nouvelle ligne, dans un nouveau bloc.

Sachez aussi que pour ces éléments en bloc, les navigateurs insèrent un espace avant et après. Vous pourrez modifier ces espaces avec des règles CSS.

## La langue et la direction du texte

Dans toute page web, vous pouvez parfaitement spécifier la langue utilisée dans un conteneur de texte avec l'attribut `lang`. Nous avons vu son utilisation dans l'élément `<html>` pour indiquer la langue utilisée dans l'ensemble de la page. Mais vous pouvez parfaitement faire une exception pour un conteneur de texte précis.

Nous pourrions parfaitement avoir cette syntaxe dans une partie spécifique d'un paragraphe rédigé en français, où la langue est de l'italien :

```
<p>Il m'a dit <span lang="it">Ciao a tutti! Come state ?</span>  
quand nous nous sommes vus.</p>
```

En plus de la langue, vous pouvez spécifier la direction d'écriture avec l'attribut `dir`. Cet attribut utilise ces trois valeurs :

- `ltr` : qui signifie *left to right*, écriture de gauche à droite, comme le français ou l'italien par exemple.
- `rtl` : qui signifie *right to left*, écriture de droite à gauche, comme l'arabe par exemple.
- `auto` : dans ce cas, c'est l'agent utilisateur qui détecte lui-même la direction de l'écriture.

Voici un exemple avec un paragraphe en arabe :

```
<p lang="ar" dir="rtl">مكلاج فيك ! اعیان حبیح </p>
```

## Les titres

Les éléments de titrage permettent d'insérer six niveaux de titre hiérarchique aux pages. Les éléments à utiliser sont `<h1>` à `<h6>`. Ces titres ont une très forte valeur sémantique. Le niveau de titre `<h1>` est le plus important dans la page et le niveau `<h6>` est le moins important.

Le W3C conseille fortement de les utiliser dans l'ordre descendant : vous devez d'abord utiliser un titre de niveau 1, puis un titre de niveau 2 et ainsi de suite. Il est déconseillé d'omettre un niveau, comme passer de `<h1>` à `<h3>`, en omittant le `<h2>`.

Notez bien que vous pouvez parfaitement utiliser plusieurs fois le même niveau de titre dans des conteneurs différents. Nous pouvons parfaitement imaginer avoir l'élément `<section>` avec un titre `<h1>` qui contient plusieurs éléments `<article>` ayant chacun un titre `<h2>`. Puis ailleurs, dans un élément `<aside>` par exemple, nous pouvons utiliser d'autres éléments `<h1>` et `<h2>`.

Les titres `<hx>` possèdent aussi une forte utilité pour l'indexation naturelle de vos pages web : SEO (*Search Engine Optimization*). Vous devez donc penser à utiliser des mots susceptibles d'être utilisés par les internautes lors de leurs recherches sur Internet.

Dernier point, tous les titres `<hx>` s'affichent en gras dans les navigateurs, avec la taille la plus grande pour le `<h1>` et la plus petite pour le `<h6>`.

Voici un exemple simple :

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Ma page web</title>
  </head>
  <body>
    <h1>Titre de niveau 1</h1>
    <h2>Titre de niveau 2</h2>
    <h3>Titre de niveau 3</h3>
```

```
<h4>Titre de niveau 4</h4>
<h5>Titre de niveau 5</h5>
<h6>Titre de niveau 6</h6>
</body>
</html>
```

Voici l'affichage obtenu :

# **Titre de niveau 1**

## **Titre de niveau 2**

### **Titre de niveau 3**

#### **Titre de niveau 4**

##### **Titre de niveau 5**

###### **Titre de niveau 6**

## **Les paragraphes**

L'élément `<p>` permet d'insérer du texte courant dans des paragraphes. Comme pour tous les écrits, chaque paragraphe pourra contenir une idée, un concept. Dans chaque paragraphe, vous pouvez mettre en évidence un ou des mots avec des éléments de mise en forme sémantique.

Voici un exemple simple :

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Ma page web</title>
  </head>
  <body>
```

```

<p>Morbi leo risus, porta ac consectetur...</p>
<p>Nullam quis risus eget urna mollis ornare...</p>
</body>
</html>

```

Voici l'affichage obtenu :

Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Vestibulum id ligula porta felis euismod semper. Aenean lacinia bibendum nulla sed consectetur. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit.

Nullam quis risus eget urna mollis ornare vel eu leo. Nullam quis risus eget urna mollis ornare vel eu leo. Donec sed odio dui. *Cras mattis consectetur purus sit amet fermentum. Nullam quis risus eget urna mollis ornare vel eu leo. Cras mattis consectetur purus sit amet fermentum. Cras mattis consectetur purus sit amet fermentum.*

## Les citations

Les blocs de citation permettent d'afficher un texte extrait d'une source externe. C'est l'élément `<blockquote>` qui est utilisé. L'élément `<blockquote>` sert de conteneur à d'autres éléments qui peuvent être de tout type : titre, paragraphe, image...

Voici un exemple simple :

```

<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Ma page web</title>
  </head>
  <body>
    <h1>Inceptos Consectetur Tristique Bibendum</h1>
    <p>Morbi leo risus, porta ac consectetur...</p>
    <blockquote>
      <h2>Fusce Mattis Ligula Etiam</h2>
      <p>Nullam id dolor id nibh...</p>
      
    </blockquote>
  </body>

```

```
<p>Nullam quis risus eget...</p>
</body>
</html>
```

Voici l'affichage obtenu :

## Inceptos Consectetur Tristique Bibendum

Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Vestibulum id ligula porta felis euismod semper. Aenean lacinia bibendum nulla sed consectetur. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit.

### Fusce Mattis Ligula Etiam

Nullam id dolor id nibh ultricies vehicula ut id elit. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec id elit non mi porta gravida at eget metus.



Nullam quis risus eget urna mollis ornare vel eu leo. Nullam quis risus eget urna mollis ornare vel eu leo. Donec sed odio dui. Cras mattis consectetur purus sit amet fermentum. Nullam quis risus eget urna mollis ornare vel eu leo. Cras mattis consectetur purus sit amet fermentum. Cras mattis consectetur purus sit amet fermentum.

## Les listes

### 1. Les différents types de liste

Les listes permettent d'insérer des énumérations sémantiques dans vos pages web. Vous avez trois types de liste à votre disposition : les listes non ordonnées (plus communément appelées listes à puces), les listes ordonnées (listes numérotées) et les listes de définitions.

Notez l'usage très répandu des listes pour créer des barres de navigation. C'est une utilisation parfaitement valide puisque ces barres sont des énumérations, des listes de liens.

## 2. Les listes non ordonnées

Les listes non ordonnées (*unordered list* en anglais) permettent de lister, d'énumérer des données qui s'afficheront avec une puce devant chaque élément de la liste. C'est l'élément `<ul>` qui est utilisé pour définir la liste. Ensuite, chaque élément, chaque item de la liste, sera placé dans un élément `<li>`.

Voici un exemple simple :

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Ma page web</title>
    </head>
    <body>
        <ul>
            <li>Pommes</li>
            <li>Poires</li>
            <li>Ananas</li>
        </ul>
    </body>
</html>
```

Voici l'affichage obtenu :

- Pommes
- Poires
- Ananas

### 3. Les listes ordonnées

Les listes ordonnées (*ordered list* en anglais) permettent de lister, d'énumérer des données qui s'afficheront avec chiffre devant chaque élément de la liste. C'est l'élément `<ol>` qui est utilisé pour définir la liste. Ensuite, chaque élément, chaque item de la liste, sera placé dans un élément `<li>`.

Voici un exemple simple :

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Ma page web</title>
  </head>
  <body>
    <ol>
      <li>Pommes</li>
      <li>Poires</li>
      <li>Ananas</li>
    </ol>
  </body>
</html>
```

Voici l'affichage obtenu :

- 
1. Pommes
  2. Poires
  3. Ananas

La liste `<ol>` possède plusieurs attributs :

- `start` permet de définir la valeur du début de la numérotation.
- `reversed` donne la possibilité d'inverser l'ordre des items de la liste.
- `type` permet de changer le type de l'énumération. Vous pouvez utiliser comme valeur `1` pour avoir des chiffres (valeur par défaut), `a` pour avoir

des lettres en minuscules, A pour avoir des lettres en majuscules et I pour avoir des chiffres romains.

Voici un exemple simple :

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Ma page web</title>
    </head>
    <body>
        <ol start="5" type="I">
            <li>Pommes</li>
            <li>Poires</li>
            <li>Ananas</li>
        </ol>
    </body>
</html>
```

Voici l'affichage obtenu :



V. Pommes  
VI. Poires  
VII. Ananas

#### 4. L'attribut de <li>

L'élément `<li>` peut utiliser l'attribut `value` pour spécifier sa valeur d'affichage dans les listes `<ol>`.

Voici un exemple simple :

```
<!doctype html>
<html lang="fr">
    <head>
```

```

<meta charset="UTF-8">
<title>Ma page web</title>
</head>
<body>
<ol>
<li value="5">Pommes</li>
<li>Poires</li>
<li value="2">Ananas</li>
</ol>
</body>
</html>

```

Voici l'affichage obtenu :

5. Pommes  
6. Poires  
2. Ananas

## 5. Les listes de définitions

Les listes de définitions permettent d'afficher les définitions de mots ou de termes qui vous semblent avoir une compréhension difficile. Pour créer une liste de définitions, nous avons trois éléments à notre disposition :

- dl (description *list*) permet de définir la liste de définition.
- dt (description *term*) indique le terme à définir.
- dd (description *definition*) donne la définition du terme. Par défaut, la définition sera indentée par rapport à son terme.

Voici un exemple simple :

```

<!doctype html>
<html lang="fr">
<head>
<meta charset="UTF-8">
<title>Ma page web</title>

```

```

</head>

<body>

    <dl>

        <dt>Sollicitudin</dt>
            <dd>Integer posuere erat...</dd>
        <dt>Etiam</dt>
            <dd>Vivamus sagittis lacus...</dd>
        <dt>Vulputate</dt>
            <dd>Vestibulum id ligula porta...</dd>
    </dl>

</body>

</html>

```

Voici l'affichage obtenu :

Sollicitudin	Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Donec ullamcorper nulla non metus auctor fringilla.
Etiam	Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Cras mattis consectetur purus sit amet fermentum.
Vulputate	Vestibulum id ligula porta felis euismod semper. Cras mattis consectetur purus sit amet fermentum.

## Les adresses

Il n'est pas rare d'avoir besoin d'insérer des adresses en tout genre dans vos pages web. Pour cela, vous devez utiliser l'élément `<address>` qui est dédié à cet usage. Vous pourrez y placer des adresses postales, des adresses mail etc. Dans cet élément, vous pouvez imbriquer les autres conteneurs que vous souhaitez.

Voici un exemple simple :

```

<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Ma page web</title>
    </head>

```

```

<body>
    <p>Voici nos coordonnées :</p>
    <address>
        <p>Agence Be Web</br>
        12 rue Tim Berners Lee<br>
        44000 NANTES</p>
        <p>Sur LinkedIn : <a href="https://linkedin.com/beweb/">
        Be Web</a>.</p>
    </address>
</body>
</html>

```

Voici l'affichage obtenu :

**Voici nos coordonnées :**

*Agence Be Web  
12 rue Tim Berners Lee  
44000 NANTES*

*Sur LinkedIn : [Be Web.](https://linkedin.com/beweb/)*

## Le texte préformaté

Le texte préformaté, inséré avec l'élément `<pre>`, permet d'insérer du texte qui sera mis en forme avec les conventions typographiques usuelles et non pas avec des éléments HTML. Cela veut dire que les espaces seront conservés, comme les marques de tabulation, et le texte sera usuellement affiché avec une police de caractère à espace constant, de type Courier.

Voici un exemple simple :

```

<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Ma page web</title>

```

```

</head>

<body>

    <p>Voici nos données :</p>

    <pre>
        Janvier    Février    Mars
        123        134        154
        245        276        287
        190        213        267
    </pre>

    <p>Donec ullamcorper nulla non metus auctor fringilla.</p>

</body>

</html>

```

Il y a une tabulation entre chaque entrée de ce pseudo-tableau.

Voici l'affichage obtenu :

Voici nos données :

Janvier	Février	Mars
123	134	154
245	276	287
190	213	267

Donec ullamcorper nulla non metus auctor fringilla.

## Les lignes horizontales

Cet élément `<hr>` ne contient pas de texte et n'affiche qu'une ligne horizontale qui permet de séparer différentes parties d'un contenu.

Voici un exemple simple :

```

<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">

```

```

<title>Ma page web</title>
</head>
<body>
    <h1>Magna Etiam Parturient Inceptos</h1>
    <p>Maecenas faucibus mollis interdum. Aenean eu leo quam.
    Pellentesque ornare sem lacinia quam venenatis vestibulum. Cum
    sociis natoque penatibus et magnis dis parturient montes, nascetur
    ridiculus mus. Duis mollis, est non commodo luctus, nisi erat
    porttitor ligula, eget lacinia odio sem nec elit. Cum sociis
    natoque penatibus et magnis dis parturient montes, nascetur
    ridiculus mus.</p>
    <hr>
    <p>Sed posuere consectetur est at lobortis. Cras justo odio,
    dapibus ac facilisis in, egestas eget quam. Lorem ipsum dolor sit
    amet, consectetur adipiscing elit. Aenean lacinia bibendum nulla
    sed consectetur. Morbi leo risus, porta ac consectetur ac,
    vestibulum at eros. Nullam quis risus eget urna mollis ornare vel
    eu leo. Morbi leo risus, porta ac consectetur ac, vestibulum at
    eros.</p>
</body>
</html>

```

Voici l'affichage obtenu :

## Magna Etiam Parturient Inceptos

Maecenas faucibus mollis interdum. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Sed posuere consectetur est at lobortis. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean lacinia bibendum nulla sed consectetur. Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Nullam quis risus eget urna mollis ornare vel eu leo. Morbi leo risus, porta ac consectetur ac, vestibulum at eros.

## VII. La mise en forme sémantique

Utiliser une mise en forme sémantique

Dans ce chapitre, nous allons aborder la mise en forme sémantique du texte. Vous allez pouvoir mettre en évidence un ou plusieurs mots d'un élément conteneur de type bloc, avec des éléments de type en ligne. Dans un même paragraphe, vous pourrez mettre des mots en gras, en italique, en souligné...

## Insérer des caractères spéciaux

Dans un texte, il est très courant d'avoir à insérer des caractères spéciaux, comme des flèches, des puces, des symboles mathématiques... Attention, il ne s'agit pas ici d'insérer des caractères accentués. N'oublions pas que nous avons déclaré dans l'en-tête <head> que l'encodage des caractères se fait en UTF-8, donc tous les caractères du clavier seront parfaitement reconnus par les navigateurs.

Les caractères spéciaux s'insèrent sous la forme d'entités de caractères, avec cette syntaxe précise :

- L'entité est préfixée par le caractère esperluette : &.
- Nous avons ensuite le code du caractère.
- L'entité est suffixée par le caractère point-virgule : ; .

Voici l'entité de caractère qui permet d'insérer une flèche vers la droite : &rarr;. rarr signifie *Right Arrow*.

Une autre entité très utile et très souvent utilisée est l'espace insécable : &ampnbsp. nbsp signifie *Non-breaking space*.

Voici une URL pour retrouver bon nombre d'entités de caractères : [https://fr.wikipedia.org/wiki/Liste\\_des\\_entités\\_caractère\\_de\\_XML\\_et\\_HTML](https://fr.wikipedia.org/wiki/Liste_des_entités_caractère_de_XML_et_HTML)

Voici un exemple très simple :

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Ma page web</title>
    </head>
    <body>
        <p>Des fruits :<br>
```

```

&rarr; Pommes<br>
&rarr; Poires<br>
&rarr; Ananas</p>

<p>Des symbole de carte :<br>
Le pic : &spades;<br>
Le trèfle &clubs;<br>
Le cœur &hearts;<br>
Le carreau : &diams;</p>

</body>
</html>

```

Voici l'affichage obtenu :

```

Des fruits :
→ Pommes
→ Poires
→ Ananas

Des symbole de carte :
Le pic : ♠
Le trèfle ♣
Le cœur ♥
Le carreau : ♦

```

## L'emphase forte

L'élément `<strong>` applique une forte emphase aux mots qui y sont inclus et l'affichage résultant usuel est une mise en gras. D'un point de vue sémantique, il s'agit bien d'une emphase forte et non d'une mise en gras.

Voici un exemple très simple :

```

<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Ma page web</title>

```

```
</head>

<body>

    <p>Nullam id dolor id nibh <strong>ultricies vehicula</strong>
ut id elit.</p>

</body>

</html>
```

Voici l'affichage obtenu :

Nullam id dolor id nibh **ultricies vehicula** ut id elit.

## L'emphase simple

Sur le même principe, une emphase simple s'affiche en italique et s'applique avec l'élément `<em>`.

Voici un exemple très simple :

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Ma page web</title>
    </head>
    <body>
        <p>Nullam id dolor id nibh <em>ultricies vehicula</em>
ut id elit.</p>
    </body>
</html>
```

Voici l'affichage obtenu :

Nullam id dolor id nibh *ultricies vehicula* ut id elit.

## Mettre en gras et en italique

Les éléments **<b>** et *<i>* permettent respectivement d'afficher du texte en gras et en italique, sans leur conférer un sens sémantique de type emphase. Il s'agit uniquement de mises en évidence pour distinguer ces mots des autres.

Voici un exemple très simple :

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Ma page web</title>
    </head>
    <body>
        <p>Un très <b>bon</b> livre et une belle
<i>découverte</i>.</p>
    </body>
</html>
```

Voici l'affichage obtenu :

Un très **bon** livre et une belle *découverte*.

## L'indice et l'exposant

Les éléments **<sub>** et **<sup>** permettent respectivement de mettre un ou des caractères en indice et en exposant.

Voici un exemple très simple :

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Ma page web</title>
    </head>
```

```
<body>  
    <p>Un bidon de 10m3, d'eau (H2O).</p>  
</body>  
</html>
```

Voici l'affichage obtenu :

Un bidon de 10m<sup>3</sup>, d'eau (H<sub>2</sub>O).

## Le souligné

Pour appliquer un souligné, utilisez l'élément `<u>`. Son sens sémantique est précis, il permet de mettre en évidence un texte sans importance particulière ou un texte dont l'orthographe ou la grammaire sont incorrectes.

Attention à l'utilisation du souligné, les visiteurs de la page pourraient confondre ces mots mis en évidence avec un lien hypertexte.

Voici un exemple très simple :

```
<!doctype html>  
<html lang="fr">  
    <head>  
        <meta charset="UTF-8">  
        <title>Ma page web</title>  
    </head>  
    <body>  
        <p>Un texte avec une <u>foto d'ortografe</u>.</p>  
    </body>  
</html>
```

Voici l'affichage obtenu :

Un texte avec une fote d'ortografe.

## Le barré

Applique un barré à du texte avec l'élément `<s>`, indique que celui-ci n'est plus exact ou n'est plus pertinent. Son affichage usuel est un texte barré.

Voici un exemple très simple :

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Ma page web</title>
    </head>
    <body>
        <p>Les pages web se créées avec le langage <s>HTML 4.01</s>, se conçoivent maintenant avec l'HTML5.</p>
    </body>
</html>
```

Voici l'affichage obtenu :

Les pages web se créées avec le langage **HTML 4.01**, se conçoivent maintenant avec l'**HTML5**.

## Réduire la taille de caractère

L'élément `<small>` a un sens sémantique très précis : il permet d'indiquer que le texte affiché en plus petit est de type "note en bas de page".

Voici un exemple très simple :

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Ma page web</title>
    </head>
```

```

<body>
    <p>Sed posuere consectetur est...</p>
    <hr>
    <p>Note : <small>Maecenas faucibus mollis...</small></p>
</body>
</html>

```

Voici l'affichage obtenu :

Sed posuere consectetur est at lobortis. Donec ullamcorper nulla non metus auctor fringilla. Sed posuere consectetur est at lobortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Donec ullamcorper nulla non metus auctor fringilla. Sed posuere consectetur est at lobortis. Donec sed odio du.

Note : Maecenas faucibus mollis interdum. Maecenas sed diam eget risus varius blandit sit amet non magna.

## Les titres d'œuvres et les citations courtes

L'élément `<cite>` permet d'indiquer le titre d'une œuvre et l'élément `<q>` spécifie une courte citation. Le texte inclus dans le titre est usuellement affiché en italique et le texte placé dans la citation courte est affiché entre guillemets.

Voici un exemple très simple :

```

<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Ma page web</title>
    </head>
    <body>
        <p>Dans l'ouvrage <cite>Le web en 2032</cite>, l'auteur nous dit <q>Eh bien je ne sais pas !</q>.</p>
    </body>
</html>

```

Voici l'affichage obtenu :

Dans l'ouvrage *Le web en 2032*, l'auteur nous dit «Ehbien je ne sais pas !».

## Les insertions et suppressions

Lorsque vous avez un document qui subit beaucoup de modifications suite à des relectures successives, il peut s'avérer très intéressant d'indiquer les textes qui ont été supprimés et ajoutés. Pour cela, utilisez les conteneurs sémantiques <ins> et <del>. Les insertions s'afficheront en souligné, les suppressions en barré.

Voici un exemple très simple :

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Ma page web</title>
  </head>
  <body>
    <p>Integer posuere erat a <ins>ante venenatis dapibus posuere velit aliquet. Donec sed odio dui.</ins> Cras justo odio, dapibus ac facilisis in, egestas eget quam. Nullam id dolor id nibh ultricies vehicula ut id elit. Donec ullamcorper nulla non metus auctor fringilla. Fusce dapibus, tellus ac cursus commodo,<del>tortor mauris condimentum nibh,</del> ut fermentum massa justo sit amet risus. Nulla vitae elit libero, a pharetra augue.</p>
  </body>
</html>
```

Voici l'affichage obtenu :

Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Nullam id dolor id nibh ultricies vehicula ut id elit. Donec ullamcorper nulla non metus auctor fringilla. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Nulla vitae elit libero, a pharetra augue.

## Le retour à la ligne

Vous pouvez dans un paragraphe par exemple, aller à la ligne avec l'élément <br>, tout en restant structurellement dans ce même paragraphe.

Voici un exemple très simple :

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Ma page web</title>
    </head>
    <body>
        <p>Integer posuere erat a ante venenatis dapibus posuere
velit aliquet. Donec sed odio dui. Cras justo odio, dapibus ac
facilisis in, egestas eget quam. Nullam id dolor id nibh ultricies
vehicula ut id elit.<br>Donec ullamcorper nulla non metus auctor
fringilla. Fusce dapibus, tellus ac cursus commodo, tortor mauris
condimentum nibh, ut fermentum massa justo sit amet risus. Nulla
vitae elit libero, a pharetra augue.</p>
    </body>
</html>
```

Voici l'affichage obtenu :

Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Nullam id dolor id nibh ultricies vehicula ut id elit.  
Donec ullamcorper nulla non metus auctor fringilla. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Nulla vitae elit libero, a pharetra augue.

## D'autres mises en forme sémantiques

Voici d'autres mises en forme sémantiques que vous pourrez rencontrer dans des pages web :

- `dfn` indique que le texte est une définition.
- `abbr` ajoute une abréviation.

Voici un exemple de son usage : <abbr title="Hypertext Markup Language">HTML</abbr>. Le texte de l'attribut title pourra s'afficher dans une infobulle.

- code indique que le contenu est du code informatique ou autre.
- var spécifie une variable mathématique ou informatique.
- samp signale l'utilisation d'un exemple ou d'un échantillon.
- kbd indique une saisie faite au clavier.
- mark met en surbrillance un texte.

## VIII. Les éléments d'interaction

### Afficher des détails

Lorsque vous disposez de peu de place dans une page web, il peut être intéressant de n'afficher une information que si le visiteur le souhaite. Pour cela, nous utilisons l'élément <details> qui est le conteneur général. L'élément <summary> contiendra le texte sur lequel les visiteurs devront cliquer pour afficher les détails. C'est dans l'élément <summary> que vous placez tous les éléments que vous souhaitez voir afficher.

Voici un exemple simple :

```
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="utf-8">
    <title>Détails et sommaire</title>
</head>
<body>
    <details>
        <summary>Affichez les informations détaillées</summary>
        <p>Morbi leo risus, porta ac consectetur...</p>
        <p>
    </p>
</details>

```

```
</details>  
</body>  
</html>
```

Voici l'affichage obtenu au chargement de la page. Le texte à cliquer est précédé par un triangle basculant. Mais c'est bien toute la ligne de texte qui est interactive.

► Affichez les informations détaillées

Voici l'affichage obtenu lorsque le visiteur a cliqué sur le texte :

▼ Affichez les informations détaillées

Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Cras mattis consectetur purus sit amet fermentum. Maecenas faucibus mollis interdum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec ullamcorper nulla non metus auctor fringilla.



## IX. Les liens

### Insertion de liens pour lier les pages

Par essence même, le Web est constitué de liens hypertextes : les pages sont reliées par des liens hypertextes. Et le HTML contient bien le mot hypertexte dans son abréviation, il correspond au H de HyperText Markup Language.

Vous allez pouvoir lier des pages entre elles, d'un site à l'autre, mais aussi au sein d'une même page de votre site. Vous insérerez des liens vers des adresses

de messagerie, vers des réseaux sociaux et pour télécharger des fichiers par exemple.

Enfin, notez que les liens sont des éléments de type en ligne (*inline*).

## La structure des liens

L'insertion d'un lien se fait avec l'élément `<a>` (a pour *anchor* en anglais). Cet élément possède plusieurs attributs :

- `href` est indispensable, il permet d'indiquer l'URL à atteindre.
- `hreflang` peut spécifier la langue de la cible. Les valeurs peuvent être par exemple `en`, `it`, `es...`
- `rel` permet d'indiquer le type de relation du lien qui est établi.
- `target` donne le contexte d'ouverture du lien dans le navigateur.

A minima, nous avons cette structure :

```
<a href="inscription.html">Inscrivez-vous</a>
```

Nous avons :

- La balise d'ouverture `<a>`.
- La référence du lien, `href`, est ici une page du même site, `inscription.html`.
- Le contenu du lien est `Inscrivez-vous`. Cela correspond au texte sur lequel les visiteurs devront cliquer pour activer le lien.
- La balise de fermeture `</a>`.

## Les liens vers des pages

Les liens que vous allez insérer auront plusieurs destinations possibles. Ils peuvent cibler une page de votre site ou une page d'un autre site. Dans le premier cas, l'URL pourra être relative, dans le second, l'URL devra être absolue.

Une URL relative se crée par rapport à la page qui contient le lien. Voici quelques exemples :

- La page ciblée se trouve dans le même dossier que la page contenant le lien : `<a href="inscription.html">Inscrivez-vous</a>`.

- La page ciblée se trouve dans un sous-dossier par rapport à la page contenant le lien : <a href="utilisateurs/inscription.html">Inscrivez-vous</a>.
- La page ciblée se trouve dans un dossier parent par rapport à la page contenant le lien : <a href="../inscription.html">Inscrivez-vous</a>.

Une URL absolue se crée en indiquant l'adresse complète de la page :

- <a href="http://www.le-site.fr/voyages/venise.html">Venise</a>.

## Les liens internes

Lorsque vous avez un long document, il peut être très utile pour les visiteurs de créer des liens internes à la page, de sorte à pouvoir se déplacer facilement et rapidement dans cette même page.

La première étape consiste à identifier, avec l'attribut `id`, les éléments HTML qui devront être ciblés par les liens.

Imaginons un long document comportant trois parties : Introduction, Résultats et Conclusion. Les titres de ces trois parties seront placés dans un élément `<h2>`. Nous aurons donc cette structure :

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Ma page web</title>
</head>
<body>
  <h1>Rapport</h1>
  <h2 id="introduction">Introduction</h2>
  <p>Donec id elit non mi porta gravida...</p>
  <h2 id="resultats">Résultats</h2>
  <p>Lorem ipsum dolor sit amet, consectetur...</p>
  <h2 id="conclusion">Conclusion</h2>
```

```
<p>Integer posuere erat a ante venenatis...</p>
</body>
</html>
```

La deuxième étape consiste à créer des liens vers les cibles identifiées précédemment. Pour cela, dans l'élément href, il faut indiquer la valeur de l'identifiant qui doit être préfixée par le caractère dièse #.

Voici les liens internes :

```
<p><a href="#introduction">Introduction</a> | <a href="#resultats">
Résultats</a> | <a href="#conclusion">Conclusion</a></p>
```

Voici l'affichage obtenu :

**Rapport**

[Introduction](#) | [Résultats](#) | [Conclusion](#)

**Introduction**

Donec id elit non mi porta gravida at eget metus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Nullam quis risus eget urna mollis ornare vel eu leo. Nullam id dolor id nibh ultricies vehicula ut id elit. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam id dolor id nibh ultricies vehicula ut id elit. Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Curabitur blandit tempus porttitor. Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Maecenas faucibus mollis interdum. Maecenas faucibus mollis interdum. Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Cras mattis consectetur purus sit amet fermentum. Aenean lacinia bibendum nulla sed consectetur. Sed posuere consectetur est at lobortis. Curabitur blandit tempus porttitor. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Etiam porta sem malesuada magna mollis euismod. Maecenas faucibus mollis interdum. Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Maecenas sed diam eget

[Haut de la page](#)

L'utilisateur peut alors cliquer sur l'un des liens de ce sommaire et il sera alors automatiquement dirigé vers la cible.

Pour vous déplacer facilement dans une page, vous pouvez parfaitement insérer un lien qui permet de revenir en haut de la page. Il suffit dans un premier temps d'identifier le titre principal : <h1 id="haut">Rapport</h1>. Ensuite, placez un lien à la fin de chaque partie qui cible cet identifiant : <p><a href="#haut">Haut de la page</a></p>.

Voici l'affichage obtenu :

## Résultats

*Consectetur adipiscing elit. Cras mattis consectetur purus sit amet fermentum. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Praesent commodo cursus magna, vel scelerisque nisl consectetur et. Maecenas sed diam eget risus varius blandit sit amet non magna. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam id dolor id nibh ultricies vehicula ut id elit. Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Curabitur blandit tempus porttitor. Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Maecenas faucibus mollis interdum. Maecenas faucibus mollis interdum. Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Cras mattis consectetur purus sit amet fermentum. Aenean lacinia bibendum nulla sed consectetur. Sed posuere consectetur est at lobortis. Curabitur blandit tempus porttitor. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Etiam porta sem malesuada magna mollis euismod. Maecenas faucibus mollis interdum. Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Maecenas sed diam eget risus varius blandit sit amet non magna.*

[Haut de la page](#)

Voici le code complet de cet exemple :

```
<!DOCTYPE HTML>

<html>
  <head>
    <meta charset="utf-8">
    <title>Ma page web</title>
  </head>
  <body>
    <h1 id="haut">Rapport</h1>
    <p><a href="#introduction">Introduction</a> | <a href="#resultats">Résultats</a> | <a href="#conclusion">Conclusion</a></p>
    <h2 id="introduction">Introduction</h2>
    <p>Donec id elit non mi porta gravida...</p>
    <p><a href="#haut">Haut de la page</a></p>
    <h2 id="resultats">Résultats</h2>
    <p>Lorem ipsum dolor sit amet, consectetur...</p>
    <p><a href="#haut">Haut de la page</a></p>
    <h2 id="conclusion">Conclusion</h2>
    <p>Integer posuere erat a ante venenatis...</p>
    <p><a href="#haut">Haut de la page</a></p>
  </body>
```

```
</html>
```

## Le contexte d'ouverture du lien

Lorsque vous créez un lien, vous pouvez spécifier le contexte d'ouverture de la cible avec l'attribut `target`. Cet attribut accepte quatre valeurs :

- `_blank` : la cible s'ouvre dans une nouvelle fenêtre ou un nouvel onglet.
- `_parent` : la cible s'ouvre dans l'élément parent de la cible. Cela est utile lorsque vous gérez des divisions d'écran avec l'élément `<iframe>`.
- `_self` : la cible s'ouvre dans la même fenêtre que le lien. Il n'y a donc pas de changement de contexte, c'est le contexte par défaut.
- `_top` : la cible s'ouvre dans le parent le plus élevé dans la hiérarchie des fenêtres.

## Les relations des liens

L'attribut `rel` permet de spécifier le type de relation du lien. Notez que l'on retrouve cet attribut dans l'élément `<link>` que nous avons vu précédemment. Le W3C propose un certain nombre de valeurs possibles pour l'attribut `rel`, mais il laisse la possibilité aux développeurs d'utiliser leurs propres valeurs. C'est donc un système ouvert.

Voici des valeurs utilisables :

- `alternate` indique que le lien cible une version alternative au document.
- `archives` précise que le document ciblé est une archive.
- `author` permet d'indiquer le nom de l'auteur du document.
- `first` signale que le document ciblé est le premier d'une série.
- `last` indique que le document ciblé est le dernier d'une série.
- `prev` signale que le document ciblé est le document précédent du document courant.
- `next` permet d'indiquer que le document ciblé est le document suivant du document courant.

- `nofollow` précise que le lien n'est pas approuvé par l'auteur. Cette valeur créée par Google indique, en principe, que l'indexation ne doit pas tenir compte des liens présents dans cette page ciblée.

## Les liens vers les messageries

Il est très courant d'utiliser un lien vers une adresse mail pour établir un contact entre le visiteur et les responsables d'un site. Lorsque le visiteur cliquera sur le lien, son application de messagerie s'ouvrira avec l'indication de l'adresse mail déjà renseignée.

Pour cela, il suffit d'indiquer dans l'attribut `href` le préfixe `mailto:` suivi par l'adresse mail souhaitée. Voici cette syntaxe :

```
<p><a href="mailto:jean@dupond.org">Contactez-moi</a></p>
```

À l'heure de la prédominance des réseaux sociaux, il est tout à fait possible de créer des liens vers vos comptes sur ces plateformes.

La syntaxe est très similaire à ce que nous avons précédemment vu :

- Lien vers Twitter : `<a href="https://twitter.com/mon-compte">`.
- Lien vers Facebook : `<a href="https://facebook.com/mon-compte">`.
- Lien vers LinkedIn : `<a href="https://fr.linkedin.com/in/mon-compte">`.

## Les liens de téléchargement

Malgré l'importance grandissante des sites de partage de fichiers, les fameuses *box*, vous pouvez parfaitement proposer à vos visiteurs de télécharger un fichier par l'intermédiaire d'un lien.

Pour ce faire, indiquez tout simplement le nom complet du fichier dans l'attribut `href`, sans omettre son extension. Voici un exemple :

```
<p>Téléchargez le fichier <a href="programme.pdf">PDF du programme</a>.</p>
```

Dans cet exemple, il s'agit d'un fichier PDF. Si l'ordinateur du visiteur possède un logiciel pour ouvrir ce type de document, il s'ouvrira. Si aucun logiciel n'est disponible, le navigateur proposera de télécharger le fichier.

## Des liens en image

Dans tous les exemples que nous avons étudiés précédemment, les liens étaient textuels. C'est-à-dire que les visiteurs devaient cliquer sur le texte placé comme contenu de l'élément `<a>`. Mais vous pouvez parfaitement utiliser des images.

Voici un exemple de l'utilisation d'une image pour créer un lien :

```
<p><a href="aide.html"></a></p>
```

La structure est très simple :

- Le lien `<a>` utilise l'attribut `href` pour indiquer l'URL ciblée.
- Le contenu du lien `<a>` est une image `<img>` qui possède son attribut `src` pour indiquer la source de l'image.

## X. Les tableaux

### La bonne utilisation des tableaux

Les tableaux sont uniquement faits pour afficher des données tabulaires. Toute autre utilisation n'est pas sémantique et n'est pas appropriée. Vous allez pouvoir créer des tableaux simples ou plus élaborés avec des regroupements de colonnes, des légendes... Enfin, notez que les tableaux s'affichent comme les éléments de type bloc, mais ils ont leur type spécifique qui est `table`.

### La structure des tableaux

Pour insérer un tableau, il faut utiliser l'élément `<table>`. Tous les autres éléments structurels des tableaux, lignes, colonnes et cellules, seront inclus dans cet élément.

Vous pourrez insérer :

- Des lignes avec `<tr>`.
- Des cellules avec `<td>` et `<th>`.
- Un titre avec `<caption>`.
- Un en-tête avec `<thead>`.

- Un pied de tableau avec `<tfoot>`.
- Un corps de tableau avec `<tbody>`.
- Des groupes de colonnes avec `<colgroup>` et `<col>`.

## Les lignes

Dans un tableau, vous allez devoir insérer des lignes avec l'élément `<tr>` (pour *table row*). Chaque ligne contiendra des cellules.

Voici une structure initiale simpliste avec juste trois lignes :

```
<table>
  <tr>...</tr>
  <tr>...</tr>
  <tr>...</tr>
</table>
```

Vous voyez que la structure interne des tableaux se crée avec des lignes et pas avec des colonnes.

## Les cellules

Dans les lignes de votre tableau, vous insérez ensuite des cellules simples avec l'élément `<td>` (pour *table data*) ou des cellules d'en-tête avec `<th>` (pour *table header*). Les cellules d'en-tête se différencient par un contenu centré et en gras.

Voici un exemple simple :

Janvier	Février	Mars
123		

```
<table>
  <tr>
    <th>Janvier</th>
    <th>Février</th>
    <th>Mars</th>
  </tr>
  <tr>
    <td>123</td>
  </tr>
</table>
```

```

<td>134</td>
<td>156</td>
</tr>
<tr>
<td>213</td>
<td>256</td>
<td>273</td>
</tr>
<tr>
<td>321</td>
<td>351</td>
<td>372</td>
</tr>
</table>

```

Voici l'affichage obtenu :

Janvier	Février	Mars
123	134	156
213	256	273
321	351	372

Si vous avez un tableau à double entrée, vous pouvez parfaitement utiliser <th> pour les en-têtes des lignes :

```

<table>
<tr>
<th>&nbsp;</th>
<th>Janvier</th>
<th>Février</th>
<th>Mars</th>
</tr>
<tr>
<th>Nantes</th>

```

```

<td>123</td>
<td>134</td>
<td>156</td>
</tr>
<tr>
    <th>Rennes</th>
    <td>213</td>
    <td>256</td>
    <td>273</td>
</tr>
<tr>
    <th>Vannes</th>
    <td>321</td>
    <td>351</td>
    <td>372</td>
</tr>
</table>

```

Vous remarquerez l'utilisation de l'entité de caractère &nbsp; dans la première cellule de la première ligne, pour avoir une cellule vide. Il est plus judicieux d'avoir un caractère "invisible" qu'aucune donnée dans une cellule.

Voici l'affichage obtenu :

Janvier Février Mars		
<b>Nantes</b>	123	134
<b>Rennes</b>	213	256
<b>Vannes</b>	321	351
		156
		273
		372

## La fusion des cellules

Vous pouvez parfaitement fusionner des cellules, verticalement ou horizontalement avec les attributs `rowspan` et `colspan`. La valeur de ces attributs indique le nombre de cellules que vous souhaitez fusionner.

Voici un exemple où deux cellules sont fusionnées verticalement avec rowspan :

```
<table>
  <tr>
    <th>Janvier</th>
    <th>Février</th>
    <th>Mars</th>
  </tr>
  <tr>
    <td>123</td>
    <td>134</td>
    <td>156</td>
  </tr>
  <tr>
    <td>213</td>
    <td rowspan="2">256</td>
    <td>273</td>
  </tr>
  <tr>
    <td>321</td>
    <td>372</td>
  </tr>
</table>
```

Pour plus de facilité d'affichage, j'ai ajouté des bordures aux cellules avec des propriétés CSS que nous étudierons dans le prochain module. Voici l'affichage obtenu :

Janvier	Février	Mars
123	134	156
213	256	273
321		372

Voici un exemple où deux cellules sont fusionnées horizontalement avec colspan :

```

<table>

  <tr>
    <th>Janvier</th>
    <th>Février</th>
    <th>Mars</th>
  </tr>

  <tr>
    <td>123</td>
    <td>134</td>
    <td>156</td>
  </tr>

  <tr>
    <td>213</td>
    <td colspan="2">256</td>
  </tr>

  <tr>
    <td>321</td>
    <td>351</td>
    <td>372</td>
  </tr>

</table>

```

Voici l'affichage obtenu :

Janvier	Février	Mars
123	134	156
213	256	
321	351	372

## Le titre

L'élément `<caption>` vous permet d'ajouter un titre à vos tableaux. Ce titre va s'afficher au-dessus du tableau. Cet élément se place juste après la balise d'ouverture de `<table>`.

Voici un exemple simple :

```
<table>

    <caption>Résultats 1e trimestre</caption>

    <tr>
        <th>Janvier</th>
        <th>Février</th>
        <th>Mars</th>
    </tr>

    <tr>
        <td>123</td>
        <td>134</td>
        <td>156</td>
    </tr>

    <tr>
        <td>213</td>
        <td>256</td>
        <td>273</td>
    </tr>

    <tr>
        <td>321</td>
        <td>351</td>
        <td>372</td>
    </tr>

</table>
```

Voici l'affichage obtenu, toujours avec l'utilisation des bordures :

Résultats 1e trimestre		
Janvier	Février	Mars
123	134	156
213	256	273
321	351	372

# Les groupes de colonnes

## 1. Regrouper des colonnes

Pour avoir des mises en forme spécifiques pour certaines colonnes, vous pouvez les regrouper avec l'élément `<colgroup>`. Cet élément se place juste après la balise d'ouverture de `<table>`.

Dans cet exemple, nous souhaitons avoir les deux premières colonnes avec un fond grisé. Cette mise en forme est obtenue avec une règle CSS placée directement dans l'élément `<colgroup>`. Pour cet exemple, nous indiquons le nombre de colonnes voulu avec l'attribut `span="2"` :

```
<table>
  <colgroup span="2" style="background-color: #eee;"></colgroup>
  <tr>
    <th>Janvier</th>
    <th>Février</th>
    <th>Mars</th>
    <th>Avril</th>
  </tr>
  <tr>
    <td>123</td>
    <td>134</td>
    <td>156</td>
    <td>186</td>
  </tr>
  <tr>
    <td>213</td>
    <td>256</td>
    <td>273</td>
    <td>298</td>
  </tr>
  <tr>
    <td>321</td>
```

```

<td>351</td>
<td>372</td>
<td>387</td>
</tr>
</table>

```

Voici l'affichage obtenu :

Janvier	Février	Mars	Avril
123	134	156	186
213	256	273	298
321	351	372	387

## 2. Cibler des colonnes

Avec l'élément `<col>`, nous allons pouvoir cibler plus précisément les colonnes à grouper. Les mises en forme indiquées dans les éléments `<col>` seront prioritaires par rapport à celles indiquées dans `<colgroup>`. Notez que nous pouvons parfaitement utiliser l'attribut `span` pour regrouper des colonnes. Dans l'élément `<colgroup>`, il suffit d'indiquer les colonnes à utiliser les unes après les autres, de gauche à droite, en effectuant des groupes si besoin est.

Dans cet exemple d'un tableau à double entrée, nous voulons mettre en forme la première colonne et la colonne du mois de mars. Voilà la syntaxe à utiliser :

```

<table>
  <colgroup>
    <col style="background-color: #eee;"></col>
    <col span="2"></col>
    <col style="background-color: #aaa;"></col>
  </colgroup>
  <tr>
    <th>&nbsp;</th>
    <th>Janvier</th>
    <th>Février</th>
    <th>Mars</th>
    <th>Avril</th>
  </tr>
  <tr>
    <td>351</td>
    <td>372</td>
    <td>387</td>
    <td>123</td>
    <td>134</td>
  </tr>
  <tr>
    <td>213</td>
    <td>256</td>
    <td>273</td>
    <td>156</td>
    <td>186</td>
  </tr>
  <tr>
    <td>321</td>
    <td>351</td>
    <td>372</td>
    <td>273</td>
    <td>298</td>
  </tr>
</table>

```

```

</tr>
<tr>
    <th>Nantes</th>
    <td>123</td>
    <td>134</td>
    <td>156</td>
    <td>186</td>
</tr>
<tr>
    <th>Rennes</th>
    <td>213</td>
    <td>256</td>
    <td>273</td>
    <td>298</td>
</tr>
<tr>
    <th>Vannes</th>
    <td>321</td>
    <td>351</td>
    <td>372</td>
    <td>387</td>
</tr>
</table>

```

Détaillons cette structure :

- Nous avons l'élément `<colgroup>` qui est sans mise en forme spécifique, qui ne sert que de conteneur aux éléments `<col>`.
- Le premier élément `<col>` cible la première colonne et lui applique un fond gris clair.
- Le deuxième élément `<col>` cible les deux colonnes suivantes avec l'attribut `span="2"` qui permet de les regrouper et ne leur applique aucune mise en forme.

- Le troisième élément `<col>` cible la colonne suivante, la quatrième du tableau et lui applique un fond gris sombre.
- La cinquième colonne du tableau n'est pas ciblée, elle utilise donc la mise en forme de l'élément parent `<colgroup>` qui est sans mise en forme.

Voici l'affichage obtenu :

	<b>Janvier</b>	<b>Février</b>	<b>Mars</b>	<b>Avril</b>
<b>Nantes</b>	123	134	156	186
<b>Rennes</b>	213	256	273	298
<b>Vannes</b>	321	351	372	387

## Les tableaux structurés

Vous allez pouvoir concevoir des tableaux très structurés avec les éléments `<thead>`, `<tbody>` et `<tfoot>`. Ces trois éléments sont utilisés conjointement. L'intérêt principal de l'utilisation des tableaux structurés est que si leur contenu est plus long que ce que l'affichage permet, l'en-tête et le pied de tableau seront toujours visibles à l'écran. Il en est de même en cas d'impression du tableau, où l'en-tête et le pied seront répétés en haut et en bas des pages d'impression.

L'élément `<thead>` constitue l'en-tête du tableau. Il doit être inséré après le titre `<caption>` et les groupes de colonnes `<colgroup>`. Dans cet élément, vous placerez les lignes `<tr>` du tableau.

L'élément `<tfoot>` constitue le pied du tableau. Notez bien que cet élément `<tfoot>` doit être placé avant le corps du tableau `<tbody>`.

L'élément `<tbody>` constitue le corps du tableau et il contiendra les lignes `<tr>` du tableau.

Voici un exemple de tableau structuré :

```
<table>
  <thead>
    <tr>
      <th>&nbsp;</th>
      <th>Janvier</th>
      <th>Février</th>
```

```
<th>Mars</th>
</tr>
</thead>
<tfoot>
<tr>
    <th>Synthèse</th>
    <td>234</td>
    <td>213</td>
    <td>189</td>
</tr>
</tfoot>
<tbody>
<tr>
    <th>Nantes</th>
    <td>123</td>
    <td>134</td>
    <td>156</td>
</tr>
<tr>
    <th>Rennes</th>
    <td>213</td>
    <td>256</td>
    <td>273</td>
</tr>
<tr>
    <th>Vannes</th>
    <td>321</td>
    <td>351</td>
    <td>372</td>
</tr>
</tbody>
</table>
```

Voici l'affichage obtenu :

	Janvier	Février	Mars	Avril
Nantes	123	134	156	186
Rennes	213	256	273	298
Vannes	321	351	372	387

Bien sûr, ce tableau est petit en nombre de lignes, mais s'il y avait beaucoup plus de lignes, les navigateurs devraient normalement adapter leur affichage lors de l'utilisation de la barre de défilement.

## XI. Les images

### Bien exploiter les images

Dans beaucoup de sites, les médias occupent de plus en plus de place. Ces médias, ces images doivent être choisis avec soins et optimisés au mieux pour un chargement rapide sur tous les supports actuels : écran d'ordinateur, de tablette et de smartphone. Il y a donc un travail important de la part des graphistes pour choisir la bonne image, définir ses dimensions en fonction des divers supports et déterminer un rapport compression/poids adéquat.

Dans les pages web, les images sont des contenus « embarqués », car elles ne sont pas directement décrites dans le contenu du fichier HTML. Les fichiers des images sont usuellement placés dans un dossier spécifique dans le dossier du site et elles sont insérées dans la page web en indiquant leur chemin d'accès.

### Comprendre les formats de compression

#### 1. Compresser les images

Lorsque vous prenez une photo, il n'est guère possible de la diffuser directement sur Internet sans une réduction de ses dimensions et sans une compression. Les images brutes de prise de vue sont toujours trop grandes et trop lourdes pour être publiées sur Internet. Il faut réduire leur dimension et les compresser pour réduire leur poids.

Actuellement, il existe trois principaux formats de compression d'image : GIF, JPEG et PNG. Cette compression se fait avec des logiciels dédiés, suite à des traitements pour retoucher les photos. Il existe d'autres formats de compression, mais qui ne sont pas encore totalement pris en charge par les navigateurs. Citons les formats JPEG 2000 et WebP de Google.

## **2. Le format GIF**

Le format GIF, pour Graphics Interchange Format, est un format de compression d'images conçu en 1987 par Compuserve et il n'est pas totalement libre. Son extension est .gif. Avec ce format, les images utilisent 256 couleurs au maximum et ne proposent qu'un seul niveau de transparence. Les pixels sont donc soit opaques, soit transparents, ce qui implique des effets d'escalier très disgracieux sur les bordures. Si l'image initiale possède moins de 256 couleurs, ce format est non destructif, s'il y en a plus de 256, la compression est destructive.

Le format GIF est donc réservé aux images de petites dimensions, comme des icônes, des boutons, des logos, possédants de larges aplats de couleurs.

## **3. Le format JPEG**

Le format JPEG, pour Joint Photographic Experts Group, est un format qui a été normé en 1991 et publié en 1992. Son extension usuelle est .jpg, plus rarement .jpeg. Les images compressées dans ce format peuvent conserver des millions de couleurs et ne peuvent pas utiliser la transparence. Notez bien que c'est un format de compression destructif qui provoque donc une destruction irrémédiable de couleurs et de nuances de l'image. C'est le graphiste qui décide du taux de compression des images et donc du rapport qualité visuelle/poids.

Le format JPEG est réservé aux images photographiques devant être affichées avec une bonne qualité visuelle.

## **4. Le format PNG**

Le format PNG a été créé pour améliorer le format GIF et pour être totalement libre. PNG est l'acronyme de Portable Network Graphics. Notez que le PNG est une recommandation du W3C datée de 2003 (<https://www.w3.org/TR/PNG/>). L'extension est .png. Le format PNG permet d'utiliser soit 265 couleurs, c'est le PNG-8, soit des millions de couleurs, c'est le PNG-24. De plus, le PNG peut accepter 256 niveaux de transparence, ce qui permet d'obtenir des bordures sans effet d'escalier.

Le format PNG peut être utilisé pour de nombreux types d'images, nécessitant de la qualité et plusieurs niveaux de transparence. Il convient aux graphistes de faire des tests de comparaison avec les autres formats pour opter pour celui qui donnera le meilleur rapport qualité/poids.

## **Insérer des images avec l'élément <img>**

# 1. L'utilisation des images

L'élément `<img>` est fait pour insérer des images d'illustration directement liées au contenu rédactionnel. D'un point de vue sémantique, cet élément n'est pas fait pour appliquer une image de fond à un en-tête ou toute autre zone d'affichage dans la page. Pour cela, il faut utiliser une règle CSS.

## 2. L'attribut `src`

Le premier attribut quasiment obligatoire est `src`. Cet attribut permet d'indiquer le chemin d'accès à l'image qui doit être affichée. Le chemin d'accès s'indique usuellement relativement à la page contenant l'élément `<img>`.

Voici quelques exemples :

- `` : l'image se trouve dans le même dossier que la page HTML.
- `` : l'image se trouve dans un sous-dossier nommé `images`, placé dans le dossier contenant la page HTML.
- `` : l'image se trouve dans le dossier parent du dossier contenant la page HTML.

Mais vous pouvez aussi choisir d'afficher une image déjà publiée sur le Web, avec un chemin absolu : ``. C'est une solution qu'il faut utiliser avec parcimonie, il faut être sûr des droits d'auteur et il se peut que l'image vienne à être supprimée du site dans laquelle elle se trouve.

## 3. L'attribut `alt`

L'attribut `alt` permet de renseigner le texte alternatif à l'image, si celle-ci ne peut être chargée, donc affichée. Son utilisation est aussi quasiment obligatoire pour passer la validation de W3C.

Voici son utilisation : ``.

Voici le code d'un exemple où l'image ne peut être chargée :

```
<!doctype html>
<html>
```

```

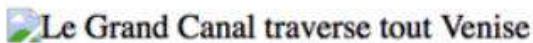
<head>
    <meta charset="utf-8">
    <title>Ma page web</title>
</head>
<body>
    <h1>Le Grand Canal à Venise</h1>
    <p>Aenean eu leo quam. Pellentesque....</p>
    <p></p>
</body>
</html>

```

Voici l'affichage obtenu dans Google Chrome :

## Le Grand Canal à Venise

Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Donec id elit non mi porta gravida at eget metus. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Donec id elit non mi porta gravida at eget metus. Donec ullamcorper nulla non metus auctor fringilla. Lorem ipsum dolor sit amet, consectetur adipiscing elit.



Nous avons une icône standard de Google Chrome qui indique que l'image ne peut être chargée et c'est le texte alternatif qui est bien affiché à la place de l'image.

## 4. Les attributs width et height

Les attributs `width` et `height` permettent, respectivement, d'indiquer l'espace alloué à l'affichage des images horizontalement et verticalement. Si ces deux attributs ne sont pas précisés, le navigateur doit attendre le chargement du fichier de l'image pour déterminer ses dimensions et pour réservé l'espace

d'affichage de l'image. Nous avons donc une perte dans les performances. C'est pour cela qu'il est conseillé de renseigner les deux attributs.

Si les valeurs des attributs sont différentes des dimensions de l'image, ce sont les valeurs d'attribut qui sont utilisées. Cela peut être pratique pour diminuer la taille d'affichage d'une image, si vous ne disposez pas du fichier source. Car il est toujours préférable de redimensionner une image dans un logiciel dédié, plutôt que le faire avec les attributs `width` et `height`. C'est là aussi une question de performance.

Notez que le HTML 5.1 autorise la valeur 0 pour les attributs `width` et `height`. Cela peut être très pratique pour masquer des images à la vue des visiteurs.

## 5. Les attributs `srcset` et `size`

Le HTML 5.1 a apporté le nouvel attribut `srcset` pour l'élément `img`. Cela permet de proposer plusieurs sources pour les images. Ces sources se différencient par la qualité et la résolution des images. Les navigateurs choisiront l'image la plus appropriée en fonction du média de diffusion, mais aussi selon la qualité de la connexion web.

Voici la syntaxe à utiliser :

```

```

Détaillons cette syntaxe :

- L'élément `<img>` utilise classiquement l'attribut `src` pour indiquer la source de l'image. Avec l'utilisation de plusieurs sources, vous pouvez indiquer l'image ayant la plus faible qualité pour une utilisation par défaut, si vous le souhaitez.
- L'attribut `srcset` permet l'exploitation des différentes variantes de l'image, en fonction de leur qualité. Chaque source d'images est séparée de la suivante par une virgule. Vous indiquez les différentes images avec leur qualité différente.

- L'attribut `srcset` permet l'utilisation du modificateur `x` qui indique la densité des pixels pour chaque image. Dans cet exemple, l'image `image-moyen-def.jpg` `2x` a donc une densité deux fois plus élevée que celle en basse définition.

Vous pouvez aussi utiliser le modificateur `w` qui indique la taille des images. Voici la syntaxe à utiliser :

```

```

Le modificateur `w` indique que la première image a une largeur de 300 pixels, la deuxième de 500 pixels et la dernière possède une largeur de 700 pixels.

Vous pouvez aussi utiliser le nouvel attribut `sizes` pour définir les conditions d'utilisation des sources d'images selon la taille de l'écran de diffusion.

Voici un exemple précis :

```

```

Voilà comment les navigateurs interprètent l'attribut `sizes` :

- Si la taille de l'écran est inférieure à 350 pixels (`max-width: 350px`), la place occupée par l'image est de 300 pixels.
- Même principe pour la deuxième taille : si la taille de l'écran est inférieure à 580 pixels (`max-width: 580px`), la place occupée par l'image est de 500 pixels.

- Dans tous les autres cas, si la taille de l'écran est supérieure à 580 pixels, la place occupée par l'image est de 700 pixels.

Les navigateurs notent la largeur du périphérique d'affichage, ensuite ils testent la condition de largeur de l'écran et enfin ils utilisent l'image ayant la taille la plus adaptée à l'écran.

## Insérer des illustrations avec l'élément <figure>

### 1. L'utilisation des illustrations

Les images permettent d'illustrer directement un contenu et les deux sont intimement liés. Les illustrations peuvent se suffire à elles-mêmes et n'ont pas besoin d'un texte rédactionnel pour être compréhensibles. De plus, la position de l'illustration peut être indépendante vis-à-vis du texte. Attention, le contenu d'une illustration peut être une ou des images photographiques, mais cela peut-être aussi un dessin, une vidéo, un tableau ou un long extrait de code. Voilà les différences sémantiques entre l'utilisation d'une image et d'une illustration.

### 2. L'élément <figure>

C'est l'élément <figure> qui permet d'insérer une illustration dans votre page web. Cet élément possède une balise d'ouverture et une balise de fermeture.

```
<figure>  
...  
</figure>
```

### 3. L'élément <figcaption>

L'élément <figcaption> permet d'afficher une légende à l'illustration. Cet élément se place dans l'élément <figure>. Dans la première version du HTML5, l'élément <figcaption> devait être utilisé comme premier ou dernier enfant de l'élément <figure>. Depuis le HTML 5.1, vous pouvez placer <figcaption> n'importe où dans l'élément <figure>.

Voici un exemple simple d'utilisation de ces deux éléments, avec l'utilisation de trois images :

```
<!doctype html>  
<html>
```

```
<head>
    <meta charset="utf-8">
    <title>Ma page web</title>
</head>
<body>
    <h1>Le Grand Canal à Venise</h1>
    <p>Aenean eu leo quam. Pellentesque ornare...</p>
    <figure>
        <br>
        <br>
        
        <figcaption>Des photos du Grand Canal à Venise</figcaption>
    </figure>
</body>
</html>
```

Voici l'affichage obtenu :

## Le Grand Canal à Venise

Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Donec id elit non mi porta gravida at eget metus. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Donec id elit non mi porta gravida at eget metus. Donec ullamcorper nulla non metus auctor fringilla. Lorem ipsum dolor sit amet, consectetur adipiscing elit.



Des photos du Grand Canal à Venise

## XII. Les formulaires

### La présence des formulaires dans les pages web

Les formulaires font partie intégrante des sites web actuels. Vous avez tous les jours fait l'expérience de saisir un formulaire, que ce soit pour réserver un voyage ou une place concert, vous inscrire à une conférence, payer vos achats en ligne ou publier un commentaire dans les pages des réseaux sociaux...

L'intégration d'un formulaire dans une page web fait très souvent appel à plusieurs compétences. Nous pouvons avoir un ergonome ou un designer d'interface pour la partie expérience utilisateur, un intégrateur qui crée le formulaire en HTML/CSS et un développeur qui conçoit le script qui va récupérer les données saisies, les valider et les gérer de manière sécurisée pour

le traitement (réservation, achat, inscription...). Le développeur utilisera le langage serveur qui sera exploité dans le projet général du site web (PHP, Ruby, C#, Java...).

Notez que vous pouvez envoyer les données du formulaire par mail, avec `action="mailto:adresse@mail.org"`. Dans ce cas, il n'y a aucun traitement et les données sont envoyées et reçues dans un format texte brut.

## La structure des formulaires

### 1. Le formulaire

Les formulaires sont insérés dans l'élément `<form>`. Dans cet élément, nous trouverons tous les champs utiles et les boutons de validation et d'annulation.

L'élément `<form>` accepte plusieurs attributs :

- `action` : indique l'URL du script qui va prendre en charge les données saisies du formulaire.
- `method` : spécifie si les données seront envoyées via HTTP, avec la méthode `get` ou `post`.
- `name` : nomme le formulaire.
- `enctype` : indique le type MIME des données envoyées. Le type MIME, pour *Multipurpose Internet Mail Extension*, permet d'indiquer la nature et le format d'un document envoyé par l'intermédiaire d'un formulaire. La valeur `application/x-www-form-urlencoded` est la valeur par défaut. Ces données sont encodées sous la forme de couple clé-valeur. Pour l'envoi de fichier, le type doit être `multipart/form-data`, format adapté pour les données binaires.

Voilà un exemple simple :

```
<form method="post" action="mon-script.php"
enctype="application/x-www-form-urlencoded" name="inscription">
  ...
</form>
```

## 2. Les étiquettes

L'élément `<label>` permet d'associer une étiquette à un champ. Cette étiquette va s'afficher devant le champ et permettre de le nommer. Cela facilitera son utilisation et offrira une meilleure accessibilité aux personnes handicapées. En effet, il suffira de cliquer sur l'étiquette d'un champ pour qu'il soit activé, avec, par exemple, le point d'insertion qui clignotera dans ce champ. C'est la méthode qui est la plus conseillée pour nommer les champs. Il vaut mieux éviter d'insérer un texte non sémantique devant le champ.

Voici un exemple simple avec deux champs de saisie :

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Ma page web</title>
</head>
<body>
    <p>Saisissez ce formulaire :</p>
    <form method="post" action="mon-script.php"
enctype="application/x-www-form-urlencoded" name="inscription">
        <p>
            <label for="nom">Votre nom : </label>
            <input type="text" id="nom" name="nom"/>
        </p>
        <p>
            <label for="prenom">Votre prénom : </label>
            <input type="text" id="prenom" name="prenom"/>
        </p>
    </form>
</body>
</html>
```

Le lien entre l'étiquette `<label>` et le champ de saisie `<input>` se fait avec les attributs `for` et `id`. Les deux valeurs doivent être identiques.

Voici l'affichage obtenu :

Saisissez ce formulaire :

Votre nom :

Votre prénom :

### 3. Grouper des champs

Pour une meilleure visibilité des champs de formulaire, vous pouvez en regrouper certains, avec l'élément <fieldset>. Ensuite, vous pourrez afficher une légende avec l'élément <legend>.

Voici un exemple simple avec le regroupement de boutons radio :

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Ma page web</title>
</head>
<body>
    <p>Saisissez ce formulaire :</p>
    <form method="post" action="mon-script.php"
enctype="application/x-www-form-urlencoded" name="inscription">
        <fieldset>
            <legend>Votre civilité : </legend>
            <input type="radio" name="civilite"
value="madame">Madame<br>
            <input type="radio" name="civilite"
value="mademoiselle">Mademoiselle<br>
            <input type="radio" name="civilite"
value="monsieur">Monsieur<br>
        </fieldset>
    </form>
</body>
</html>
```

```

        value="monsieur">Monsieur
    </fieldset>
</form>
</body>
</html>

```

Voici l'affichage obtenu :

Saisissez ce formulaire :

Votre civilité : \_\_\_\_\_

Madame  
 Mademoiselle  
 Monsieur

## 4. Les attributs communs

Les éléments HTML dédiés aux formulaires partagent un grand nombre d'attributs communs. Nous les étudierons dans les titres suivants, dans leur contexte d'utilisation. Voici les plus usités :

- **autocomplete** : permet l'autocomplétion dans les champs. L'autocomplétion permet de retrouver les saisies déjà effectuées dans une liste s'affichant sous le champ.
- **autofocus** : pour activer immédiatement un champ. Dans un champ de texte, le point d'insertion clignotera dans celui-ci.
- **checked** : indique que le bouton radio ou la case à cocher est déjà sélectionné.
- **disabled** : désactive le champ qui n'est donc plus utilisable par les visiteurs.
- **id** : permet d'identifier de manière unique chaque élément du formulaire.
- **max** et **min** : indiquent les valeurs maximale et minimale autorisées dans les champs de texte et calendaire.
- **name** : donne le nom du champ.
- **placeholder** : affiche un texte, une consigne dans le champ.
- **readonly** : indique que le champ est en lecture seule.

- `required` : permet de spécifier que la saisie dans le champ est obligatoire.
- `type` : permet de définir le type de champ.
- `value` : indique quelle est la valeur qui doit être envoyée au script et permet d'afficher le libellé des boutons d'action.

## Les champs de texte

### 1. La saisie de texte

Dans la plupart des formulaires, vous devrez saisir des informations sous forme de texte. Les formulaires proposent plusieurs champs de saisie de texte. Vous pourrez avoir des saisies simples, comme pour indiquer vos nom et prénom, des champs avec plusieurs lignes, pour saisir des commentaires par exemple, des champs pour y saisir un mot de passe...

### 2. Les champs de texte simples

C'est l'élément `<input>` qui permet de saisir un texte simple. L'attribut `type` détermine le type de champ que nous souhaitons. Pour un champ de texte simple, le type est `text` : `<input type="text">`.

Pour un exemple simple de champs de texte, reportez-vous à la section précédente nommée Les étiquettes.

### 3. Les champs de texte pour les mots de passe

Lorsque le visiteur a besoin d'effectuer une connexion à un espace réservé, il est plus prudent qu'il saisisse son mot de passe, sans que les caractères soient affichés. Dans ce cas, dans l'élément `<input>`, l'attribut `type` prend pour valeur `password` :

```
<p>
  <label for="motdepasse">Votre mot de passe : </label>
  <input type="password" id="motdepasse" name="motdepasse">
</p>
```

Voici l'affichage obtenu, avec une saisie dans le champ :

Votre mot de passe : .....

## 4. Les champs de texte multilignes

Si vous souhaitez proposer à vos visiteurs un champ de texte multiligne, utilisez l'élément <textarea>. Le champ de texte multiligne autorise la saisie d'un texte plus long, sur plusieurs lignes, pour y saisir un commentaire ou des renseignements complémentaires. Cet élément accepte plusieurs attributs :

- rows pour définir le nombre de lignes.
- cols pour indiquer le nombre de colonnes exprimé en largeur de caractères.
- wrap permet de préciser comment sont gérés les retours à la ligne dans le champ et dans l'envoi du formulaire. La valeur hard indique qu'un caractère de retour à la ligne est envoyé avec les données du formulaire et soft précise que le caractère retour à la ligne n'est pas transmis.

Voici un exemple simple :

```
<p>
    <label for="commentaire">Votre commentaire : </label>
    <br>
    <textarea id="commentaire" rows="6" cols="100"></textarea>
</p>
```

Voici l'affichage obtenu :

Votre commentaire :

## Les listes de valeurs

Pour faciliter le choix d'une valeur parmi d'autres, vous pouvez utiliser les listes déroulantes avec l'élément <select>. Les attributs sont :

- **size** : nombre d'éléments affichés. Si cet attribut n'est pas renseigné, les items de la liste sont affichés dans une liste déroulante. Si cet attribut possède une valeur, elle indique le nombre d'items qui sont affichés sur le nombre total des items. Les items non affichés sont accessibles en utilisant la barre de défilement.
- **multiple** est un attribut booléen. Sa présence indique que l'utilisateur peut sélectionner plusieurs valeurs.

Chaque item de la liste se place dans un élément <option>, en tant qu'élément enfant de l'élément <select>.

Voici des attributs utilisables :

- **value** est la donnée qui sera envoyée au script.
- **selected** est un attribut booléen qui indique que l'item est présélectionné.

Voici un exemple, avec une liste déroulante :

```
<p>
    <label for="ville">Choisissez la destination :</label>
    <br>
    <select id="ville">
        <option value="venise">Venise</option>
        <option value="florence">Florence</option>
        <option value="rome">Rome</option>
        <option value="verone">Vérone</option>
        <option value="parme">Parme</option>
        <option value="naples">Naples</option>
        <option value="genes">Gênes</option>
    </select>
</p>
```

Voici l'affichage obtenu, avec la liste non déroulée :

Choisissez la destination :

Venise

Vous remarquez que c'est le premier item qui est affiché.

Voici l'affichage obtenu, avec la liste déroulée :

Choisissez la destination :

- ✓ Venise
- Florence
- Rome
- Vérone**
- Parme
- Naples
- Gênes

Voici un deuxième exemple, avec une liste déroulée à sélection multiple :

```
<p>
<label for="ville">Choisissez la destination :</label>
<br>
<select id="ville" size="3" multiple>
    <option value="venise">Venise</option>
    <option value="florence">Florence</option>
    <option value="rome">Rome</option>
    <option value="verone">Vérone</option>
    <option value="parme">Parme</option>
    <option value="naples">Naples</option>
    <option value="genes">Gênes</option>
</select>
</p>
```

Voici l'affichage obtenu, avant la sélection :

Choisissez la destination :

Venise  
Florence  
Rome

Voici l'affichage obtenu, avec la sélection de deux items :

Choisissez la destination :

Venise  
Florence  
Rome

## Les boutons radio à choix unique

Dans les interfaces des applications et dans les formulaires, les boutons radio sont synonymes de choix unique. L'utilisateur ne pourra sélectionner qu'un seul choix. C'est avec l'élément `<input>` que nous insérons des boutons radio.

Voici des attributs utilisables :

- `name` permet de définir le groupe de boutons radio dans lequel les visiteurs ne pourront choisir qu'une seule option. La valeur de cet attribut doit être la même pour tous les boutons radio du groupe.
- `value` détermine la valeur du bouton radio sélectionnée qui est envoyée au script.
- `checked` spécifie si un bouton radio doit être sélectionné au chargement de la page.

Voici un exemple simple de demande de civilité :

```
<fieldset>
    <legend>Votre civilité : </legend>
    <input type="radio" name="civilite" value="madame">Madame<br>
    <input type="radio" name="civilite"
           value="mademoiselle">Mademoiselle<br>
    <input type="radio" name="civilite" value="monsieur">Monsieur
</fieldset>
```

Voici l'affichage obtenu :

Saisissez ce formulaire :

Votre civilité :

- Madame
- Mademoiselle
- Monsieur

## Les cases à cocher à choix multiple

Dans les interfaces des applications et dans les formulaires, les cases à cocher sont utilisées pour avoir un choix multiple. C'est à nouveau l'élément `<input>` qu'il faut utiliser, avec `type="checkbox"`. Les attributs utilisables sont les mêmes qu'avec les boutons radio, sans qu'il soit obligatoire que l'attribut `name` ait la même valeur, puisque nous voulons un choix multiple.

Voici un exemple simple :

```
<p>
    <p>Choisissez une ou plusieurs villes :</p>
    <input type="checkbox" name="venise" value="venise">Venise<br>
    <input type="checkbox" name="florence" value="florence"
checked>Florence<br>
    <input type="checkbox" name="rome" value="rome">Rome<br>
    <input type="checkbox" name="verone" value="Vérone">Vérone
</p>
```

Voici l'affichage obtenu :

Choisissez une ou plusieurs villes :

- Venise
- Florence
- Rome
- Vérone

## D'autres types de champs avec <input>

Nous venons de voir que l'élément <input> permet d'insérer des champs divers avec les types `text`, `password`, `radio` et `checkbox`. Mais il existe de nombreux autres types de champs :

- `hidden` masque un champ.
- `image` insère une image cliquable.
- `file` permet d'envoyer un fichier via le formulaire.
- `tel` spécifie que le contenu saisi doit être un numéro de téléphone.
- `url` spécifie que le contenu saisi doit être une URL.
- `email` indique que le contenu saisi doit être une adresse mail.
- `date`, `time`, `datetime`, `datetime-local`, `month`, `week` permettent de préciser que le contenu attendu est de type calendaire.
- `number` spécifie que le contenu doit être une valeur numérique.
- `range` permet d'indiquer que la valeur attendue est une valeur numérique comprise entre un intervalle de valeurs et qu'elle est indiquée à l'aide d'un curseur sur une réglette.
- `color` permet d'afficher un sélecteur de couleur.
- `search` précise que le contenu saisi est utilisé comme critère dans une recherche.

## Les aides à la saisie

### 1. Les objectifs

Les formulaires ne sont pas toujours un élément d'interface facile à appréhender pour les internautes. Pour les aider à bien utiliser les champs, nous avons à notre disposition, toute une série d'aides à la saisie.

### 2. La consigne de saisie

L'attribut `placeholder` permet d'afficher dans le champ une aide à la saisie qui disparaît dès que l'utilisateur saisit les premiers caractères. Par exemple, nous voulons préciser que le prénom qui doit être saisi correspond à celui qui est indiqué sur la carte d'identité. Voici la syntaxe à utiliser :

```
<p>
    <label for="prenom">Votre prénom : </label>
    <input type="text" id="prenom" name="prenom" size="30"
placeholder="Prénom de votre carte d'identité">
</p>
```

Voici l'affichage obtenu, avant la saisie :

Votre prénom : Prénom de votre carte d'identité

Voici l'affichage obtenu après les premières saisies :

Votre prénom : Chr

Vous pouvez aussi utiliser placeholder pour donner un exemple de saisie attendue :

```
<p>
    <label for="cp">Votre code postal : </label>
    <input type="text" id="cp" name="code-postal"
placeholder="Ex : 44600">
</p>
```

Voici l'affichage obtenu :

Votre code postal : Ex : 44600

### 3. Activer un champ

L'attribut booléen autofocus permet de faire clignoter le point d'insertion dans un champ donné. Rappelons qu'un attribut booléen n'a pas de valeur, sa seule présence suffit à son application.

```
<p>
    <label for="nom">Votre nom : </label>
```

```
<input type="text" id="nom" name="nom" autofocus>  
</p>  
<p>  
    <label for="prenom">Votre prénom : </label>  
    <input type="text" id="prenom" name="prenom" size="30"  
placeholder="Prénom de votre carte d'identité">  
</p>
```

Voici l'affichage obtenu au chargement de la page :

The screenshot shows a simple web form with two text input fields. The first field is labeled "Votre nom :" and contains a placeholder "Prénom de votre carte d'identité". The second field is labeled "Votre prénom :" and also contains the same placeholder text.

## 4. L'autocomplétion

L'attribut `autocomplete="on"` placé dans l'élément `<form>` permet d'indiquer au navigateur que si l'utilisateur saisit une donnée dans un champ, le navigateur doit lister les précédentes saisies dans une liste qui s'affiche sous le champ, pour que l'utilisateur puisse rapidement sélectionner une valeur déjà saisie. C'est le comportement par défaut des navigateurs. Par contre, rien ne vous empêche de placer cet attribut avec la valeur `off` pour un champ spécifique où vous ne souhaitez pas avoir de remplissage automatique.

## 5. Rendre un champ obligatoire

L'attribut booléen `required` permet d'indiquer qu'un champ est obligatoire. C'est à chaque navigateur d'informer, avec ses propres paramètres, qu'un champ obligatoire n'a pas été saisi, lorsque l'utilisateur clique sur le bouton d'envoi.

Voici un exemple simple :

```
<p>  
    <label for="nom">Votre nom : </label>  
    <input type="text" id="nom" name="nom" required>  
</p>
```

Voici l'affichage obtenu dans Google Chrome, lorsque le visiteur n'a pas renseigné ce champ au moment de l'envoi :

The screenshot shows a web form with a single input field labeled "Votre nom :". Below the input field is a red rectangular button with a white exclamation mark icon. To the right of the button, the text "Veuillez renseigner ce champ." is displayed in a red font, indicating a required field error.

## 6. Les saisies autorisées

L'attribut `pattern` permet d'indiquer très précisément quelle est la valeur attendue dans un champ. Un pattern est un modèle de saisie. Cet attribut se place dans un élément `<input>` et il utilise des expressions régulières. Les expressions régulières sont des chaînes de caractères, utilisant une syntaxe précise, qui permettent de décrire des ensembles de chaînes de caractères qu'il est possible d'utiliser. Il serait vain de vouloir tout connaître sur les expressions régulières dans cet ouvrage, tant le sujet est vaste et ce n'est pas le propos de ce livre. Vous trouverez sur Internet de nombreux sites traitant du sujet. Nous allons simplement évoquer quelques exemples.

Lorsque le visiteur renseigne les champs et qu'il clique sur le bouton d'envoi, c'est au navigateur de prendre en charge la vérification des champs possédant un pattern. Et c'est le navigateur qui décide de l'apparence du message à afficher lorsqu'il y a une erreur.

Dans ce premier exemple, nous souhaitons que la saisie comporte au maximum cinq lettres en majuscule.

Voici la syntaxe à utiliser :

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Ma page web</title>
</head>
<body>
    <p>Saisissez ce formulaire :</p>
    <form>
```

```

<p>
    <label for="nom">Votre pseudo : </label>
    <input type="text" id="pseudo" name="pseudo"
pattern="[A-Z]{5}">
</p>
<p>
    <input type="submit" name="envoi" value="Envoyer" />
</p>
</form>
</body>
</html>

```

Le pattern est très simple : `pattern="[A-Z]{5}"`. Nous avons entre crochets l'intervalle des lettres autorisées qui sont bien en majuscules : [A-Z]. Puis, entre accolades, nous avons le nombre de caractères maximum que l'utilisateur pourra saisir : {5}.

Voici l'affichage obtenu, lorsque le visiteur renseigne le champ :

Saisissez ce formulaire :

Votre pseudo :

Voici l'affichage obtenu avec Google Chrome, lorsque le visiteur clique sur le bouton d'envoi du formulaire alors que la saisie ne correspond pas au pattern :

Saisissez ce formulaire :

Votre pseudo :

 Veuillez respecter le format requis.

Si le visiteur saisit cinq lettres en majuscules et qu'il envoie le formulaire, il n'y a aucun message d'alerte.

Voici un deuxième exemple :

```
<input type="text" id="pseudo" name="pseudo" pattern="^ [A-Z] {2} [0-9]*">
```

Voici le pattern détaillé :

- Nous demandons à ce que la saisie commence par deux lettres en majuscules ^ [A-Z]. C'est le caractère spécial ^ qui indique, avec les caractères qui le suivent immédiatement, par quoi doit commencer la saisie.
- Puis, il peut y avoir entre 0 et n chiffres : [0-9]\*. C'est le caractère spécial \* qui impose la saisie entre 0 et n caractères.

Voici deux exemples de saisies erronées :

Saisissez ce formulaire :

Votre pseudo : AZE

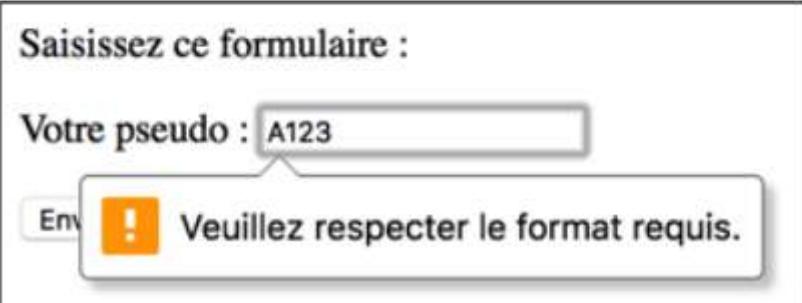
Envoyer  Veuillez respecter le format requis.



Saisissez ce formulaire :

Votre pseudo : A123

Envoyer  Veuillez respecter le format requis.



Et voici une saisie valide :

Saisissez ce formulaire :

Votre pseudo :

## Les boutons d'action

Lorsque les visiteurs ont renseigné tous les champs du formulaire, il faut l'envoyer au serveur, par l'intermédiaire du script pour effectuer le traitement. De plus, il faut toujours donner la possibilité d'annuler toutes les saisies faites afin de pouvoir recommencer à saisir le formulaire.

Pour insérer ces deux boutons d'action, nous allons utiliser l'élément `<input>`. C'est l'attribut `type` qui précise l'action à déclencher :

- Le type `type="submit"` déclenche l'envoi du formulaire au serveur pour son traitement, par l'intermédiaire du script.
- Le type `type="reset"` permet d'effacer toutes les saisies effectuées.

Voici un exemple simple :

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Ma page web</title>
</head>
<body>
    <p>Saisissez ce formulaire :</p>
    <form method="post" action="mon-script.php"
        enctype="application/x-www-form-urlencoded" name="inscription">
        <p>
            <label for="nom">Votre nom : </label>
            <input type="text" id="nom" name="nom">
        </p>
    </form>
</body>
</html>
```

```

<p>
    <label for="prenom">Votre prénom : </label>
    <input type="text" id="prenom" name="prenom">
</p>
<p>
    <input type="submit" name="envoyer" value="Envoyer">
    <br>
    <input type="reset" name="annuler" value="Annuler">
</p>
</form>
</body>
</html>

```

Voici l'affichage obtenu :

Saisissez ce formulaire :

Votre nom :

Votre prénom :

## Un exemple complet de formulaire

### 1. Le code complet du formulaire

Pour terminer ce chapitre, nous allons créer un formulaire complet avec divers types de champs.  
Voici le code utilisé :

```

<!doctype html>
<html>
<head>

```

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

<title>Ma page web</title>

</head>

<body>

    <p>Saisissez ce formulaire :</p>

    <form method="post" action="mon-script.php"
enctype="application/x-www-form-urlencoded" name="inscription">

        <fieldset>

            <legend>Votre civilité : </legend>

            <input type="radio" name="civilite"
value="madame">Madame<br>

            <input type="radio" name="civilite"
value="mademoiselle">Mademoiselle<br>

            <input type="radio" name="civilite"
value="monsieur">Monsieur

        </fieldset>

        <p>

            <label for="nom">Votre nom : </label>

            <input type="text" id="nom" name="nom">

        </p>

        <p>

            <label for="prenom">Votre prénom : </label>

            <input type="text" id="prenom" name="prenom">

        </p>

        <p>

            <label for="age">Votre âge : </label>

            <input type="number" id="age" name="age">

        </p>

        <p>

            <label for="email">Votre adresse mail : </label>

            <input type="email" id="email" name="email">

        </p>

    </form>

</body>
```

```

<p>
    <label for="date">Indiquez votre date de naissance :</label>
</p>
<input type="date" id="date" name="date">

<p>
    <label for="sport">Choisissez votre sport :</label>
    <select id="sport">
        <option value="liste">Liste des sports</option>
        <option value="tennis">Tennis</option>
        <option value="football">Football</option>
        <option value="handball">Handball</option>
        <option value="roller">Roller</option>
        <option value="arc">Tir à l'arc</option>
        <option value="athletisme">Athlétisme</option>
        <option value="natation">Natation</option>
    </select>
</p>
<fieldset>
    <legend>Choisissez une ou plusieurs options :</legend>
    <input type="checkbox" name="competition" value="competition">Compétition<br>
    <input type="checkbox" name="loisir" value="loisir" checked>Loisir<br>
    <input type="checkbox" name="accompagnateur" value="accompagnateur">Accompagnateur<br>
    <input type="checkbox" name="entraineur" value="entraineur">Entraîneur
</fieldset>
<p>
    <input type="submit" name="envoyer" value="Envoyer" />
    &nbsp;

```

```

        <input type="reset" name="annuler" value="Annuler" />
    </p>
</form>
</body>
</html>

```

## 2. Le formulaire

Commençons par détailler l'élément `<form>` :

```

<form method="post" action="mon-script.php"
enctype="application/x-www-form-urlencoded" name="inscription">
    • Le formulaire utilise la méthode post pour envoyer les données au script.
    • Le script qui gère le formulaire est nommé mon-script.php.
    • L'encodage, le type MIME, est celui par défaut : application/x-www-form-urlencoded.
    • Le nom du formulaire est inscription.

```

## 3. Les boutons radio

La civilité du visiteur est un groupe de boutons radio :

```

<fieldset>
    <input type="radio" name="civilite" value="madame">Madame<br>
    <input type="radio" name="civilite"
value="mademoiselle">Mademoiselle<br>
    <input type="radio" name="civilite" value="monsieur">Monsieur
</fieldset>

```

- Le groupe est placé dans un élément `<fieldset>`, avec une légende indiquée dans l'élément `<legend>`.
- Chaque bouton radio possède la même valeur pour l'attribut `name`, afin de n'avoir qu'un seul choix.
- Chaque bouton radio possède sa propre valeur `value` qui est envoyée au script.

Saisissez ce formulaire :

Votre civilité :

- Madame
- Mademoiselle
- Monsieur

#### 4. Les champs de texte

Ensuite, nous avons les champs de texte pour saisir le nom et le prénom :

```
<p>
    <label for="nom">Votre nom : </label>
    <input type="text" id="nom" name="nom">
</p>

<p>
    <label for="prenom">Votre prénom : </label>
    <input type="text" id="prenom" name="prenom">
</p>
```

- Chaque champ est placé dans un paragraphe, élément `<p>`.
- Chaque champ possède une étiquette, élément `<label>`.
- Les champs sont de type `text`, ils utilisent un identifiant, `id`, et un nom, `name`, propre.

Votre nom :	<input type="text"/>
Votre prénom :	<input type="text"/>

#### 5. Le champ numérique

Ensuite, nous demandons l'âge du visiteur avec un champ de type numérique :

```
<p>
    <label for="age">Votre âge : </label>
    <input type="number" id="age" name="age">
```

</p>

- Le champ est de type number.
- Le champ possède un identifiant et un nom.

L'intérêt d'insérer un champ numérique est l'utilisation d'un bouton affiché à droite dans le champ qui permet d'incrémenter ou de décrémenter la valeur.

Notez que vous pouvez utiliser d'autres attributs pour ce type de champ numérique :

- min permet de spécifier la valeur minimale admise.
- max permet de spécifier la valeur maximale admise.
- step indique la valeur d'incrémentation.

## 6. Le champ d'adresse mail

Dans le champ Votre adresse mail, nous demandons aux visiteurs de saisir leur adresse mail, dans un champ dédié :

```
<p>
    <label for="email">Votre adresse mail : </label>
    <input type="email" id="email" name="email">
</p>
```

Le type de ce champ est email, ce qui permet d'avoir une validation de la présence de l'arobase lorsque le visiteur enverra le formulaire. Si le champ ne possède pas d'arobase, le navigateur affichera un message d'alerte.

Voici un exemple avec Mozilla Firefox :

The screenshot shows a web form with three fields:

- A label "Votre adresse mail :" followed by an input field containing "christophe.aubry".
- A label "Indiquez votre d" followed by an error message "Veuillez saisir une adresse électronique valide."
- A label "Choisissez votre sport:" followed by a dropdown menu.

The input field for the email address has a red border, indicating it is invalid. The error message is displayed below the label "Indiquez votre d".

## 7. Le champ de date

Puis nous demandons la date de naissance à nos visiteurs, en spécifiant le type date au champ de texte.

<p>

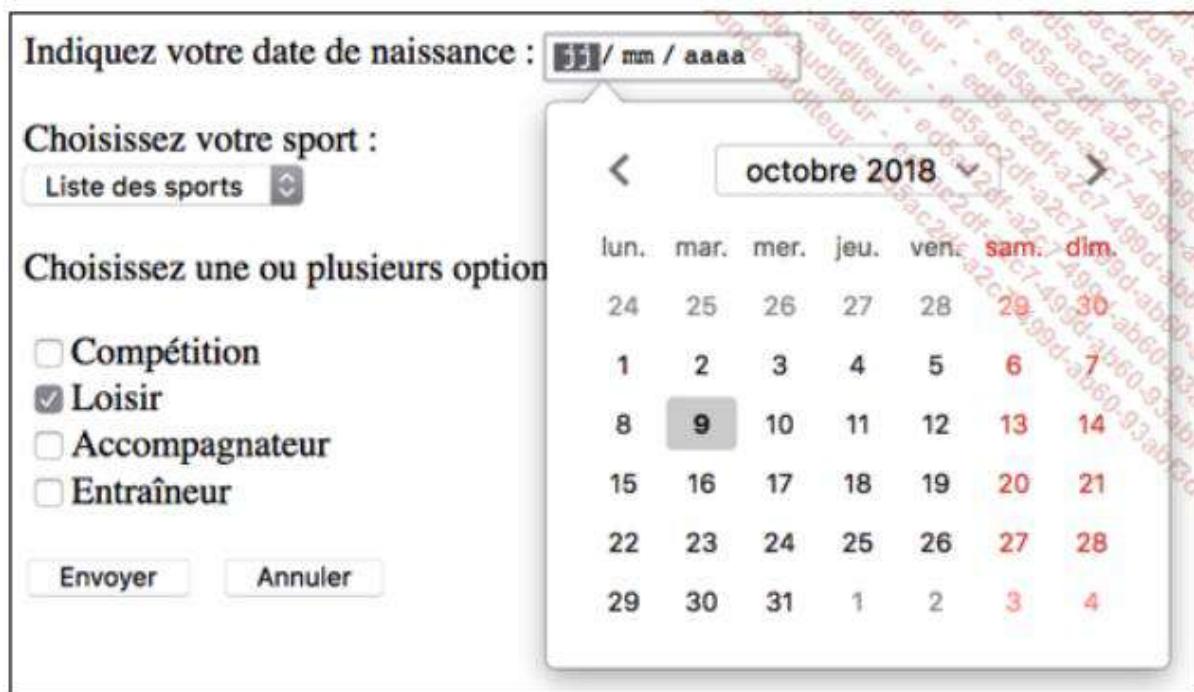
```
<label for="date">Indiquez votre date de naissance : </label>
<input type="date" id="date" name="date">
</p>
```

Le simple fait d'indiquer `type="date"` permet d'obtenir un calendrier graphique dans le navigateur. Bien sûr, chaque navigateur est libre de proposer l'interface qu'il souhaite. Le deuxième avantage est l'indication du modèle de date attendu dans le champ.

Voici l'exemple avec Mozilla Firefox, au chargement de la page :

Indiquez votre date de naissance : jj / mm / aaaa

Voici l'affichage obtenu lorsque le visiteur clique dans le champ :



Voici la date affichée lorsque le visiteur a saisi ou sélectionné la date :

Indiquez votre date de naissance : 12 / 04 / 1998

## 8. La liste déroulante

Nous demandons au visiteur de choisir un sport dans une liste déroulante. Voici le code HTML de cet élément :

```

<p>
    <label for="sport">Choisissez votre sport :</label>
    <select id="sport">
        <option value="liste">Liste des sports</option>
        <option value="tennis">Tennis</option>
        <option value="football">Football</option>
        <option value="handball">Handball</option>
        <option value="roller">Roller</option>
        <option value="arc">Tir à l'arc</option>
        <option value="athletisme">Athlétisme</option>
        <option value="natation">Natation</option>
    </select>
</p>

```

Notez simplement que la première valeur, `<option>`, de la liste déroulante n'est pas une valeur en soi, mais une indication d'interface.

Choisissez votre sport :  Liste des sports

Voici la liste déroulée pour sélectionner une seule valeur :

<b>Choisissez votre sport</b>	<input checked="" type="checkbox"/> Liste des sports
<b>Choisissez une ou plusieurs options</b>	
<input type="checkbox"/> Compétition	Liste des sports
<input checked="" type="checkbox"/> Loisir	Tennis
<input type="checkbox"/> Accompagnateur	Football
<input type="checkbox"/> Entraîneur	Handball
	Roller
	Tir à l'arc
	Athlétisme
	Natation

## 9. Les cases à cocher

Le formulaire se termine par un choix multiple avec des cases à cocher.

```

<fieldset>
    <legend>Choisissez une ou plusieurs options :</legend>

```

```

<input type="checkbox" name="competition"
value="competition">Compétition<br>
<input type="checkbox" name="loisir" value="loisir"
checked>Loisir<br>
<input type="checkbox" name="accompagnateur"
value="accompagnateur">Accompagnateur<br>
<input type="checkbox" name="entraineur"
value="entraineur">Entraîneur
</fieldset>

```

- Les cases à cocher sont regroupées avec l'élément `<fieldset>` qui possède une légende, `<legend>`, pour afficher un titre.
- La deuxième case à cocher est déjà sélectionnée avec l'attribut `checked`.

**— Choisissez une ou plusieurs options :**

<input type="checkbox"/> Compétition
<input checked="" type="checkbox"/> Loisir
<input type="checkbox"/> Accompagnateur
<input type="checkbox"/> Entraîneur

## 10. Les boutons d'action

Pour clore le formulaire, nous avons deux boutons d'action : le premier pour envoyer les données du formulaire au script et le deuxième pour annuler toutes les saisies.

```

<p>
<input type="submit" name="envoyer" value="Envoyer" />
&nbsp;
<input type="reset" name="annuler" value="Annuler" />
</p>

```

<input type="button" value="Envoyer"/>	<input type="button" value="Annuler"/>
--	--

# XIII. Le multimedia

# La présence du multimédia

Aujourd’hui, il est presque banal d’insérer de la vidéo et de l’audio dans les pages web. Il y a quelques années déjà, la lutte fut terrible entre les plugins qui permettaient de lire du multimédia, avec Adobe Flash, Microsoft Media Player et Apple QuickTime. De plus, chaque plugin ne lisait que les fichiers encodés avec des codecs spécifiques. Mais le HTML5 a bousculé tout cela en introduisant les nouveaux éléments audio et video.

## Les formats et les codecs

Il reste encore un vrai problème à l’uniformisation du multimédia sur le web, c’est la compression et le format de diffusion des fichiers. Comme pour les photos, vous ne pouvez pas directement insérer dans votre site web, une vidéo ou un fichier audio. Ces fichiers bruts ne sont pas adaptés à une diffusion sur Internet. Il faut au préalable les compresser avec un codec et les publier avec un format reconnu par les navigateurs.

Sans entrer dans des considérations trop techniques concernant le multimédia, abordons quelques points sur la compression et la diffusion du multimédia.

Pour compresser un fichier, il faut utiliser un codec. Cet acronyme signifie Coder-Décoder. Sachez qu’il existe de très nombreux codecs multimédias : VP9, MP3, AAC, H.265... Ensuite, pour diffuser ces fichiers compressés, il faut "emballer" ces fichiers dans un format de transport. Citons .ogg, .mp4 et .webm.

En ce qui concerne la diffusion, le W3C préconise d’utiliser un format open source et gratuit. Plusieurs solutions sont alors disponibles :

- Mozilla, Opera et Google utilisent les codecs Theora et Vorbis, et .ogg pour le format de diffusion.
- Apple et Microsoft utilisent les codecs H.264 et MP4, et .mp4 pour le format.
- Google propose en 2010 les codecs VP8 et Vorbis, et .webm pour le format. Ces solutions sont open source et gratuites.

Pour avoir une vision globale des codecs et des formats de diffusion du multimédia qui sont compatibles avec les navigateurs modernes, consultez cette URL : [https://developer.mozilla.org/fr/docs/Web/HTML/Formats\\_pour\\_audio\\_video](https://developer.mozilla.org/fr/docs/Web/HTML/Formats_pour_audio_video). Dans la section Compatibilité des navigateurs, vous avez un tableau récapitulatif avec de nombreux paramètres techniques.

Voilà l’état des lieux en octobre 2018 :

Ordinateur	Mobile	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Fonctionnalité						
Support simple		3.0	3.5 (1.9.1)	9.0	10.50	3.1
<audio> : PCM en WAVE		(Oui)	3.5 (1.9.1)	Pas de support	10.50	3.1
<audio> : Vorbis en WebM		(Oui)	4.0 (2.0)	Pas de support	10.60	3.1(1)
<audio> : Diffusion Vorbis/Opus en WebM via MSE		?	36.0 (36.0) <sup>[2]</sup>	?	?	?
<audio> : Vorbis en Ogg		(Oui)	3.5 (1.9.1)	Pas de support	10.50	Pas de support
<audio> : MP3		(Oui) <sup>[4]</sup>	(Oui) <sup>[5]</sup>	9.0	(Oui)	3.1
<audio> : MP3 en MP4		?	?	?	?	(Oui)
<audio> : AAC en MP4		(Oui) <sup>[6]</sup>	(Oui) <sup>[7]</sup>	9.0	(Oui)	3.1
<audio> : Opus en Ogg		27.0	15.0 (15.0)	?	?	?
<audio> : FLAC		56.0	51 (51)	Pas de support	Pas de support	11
<audio> : FLAC en Ogg		56.0	51 (51)	Pas de support	Pas de support	Pas de support
<video> : VP8 et Vorbis en WebM		6.0	4.0 (2.0)	9.0 <sup>[8]</sup>	10.60	3.1 <sup>[9]</sup>
<video> : VP9 et Opus en WebM		29.0	28.0 (28.0) <sup>[36]</sup>	?	(Oui)	?
<video> : Diffusion WebM via MSE		?	42.0 (42.0) <sup>[35]</sup>	?	?	?
<video> : Theora et Vorbis en Ogg		(Oui)	3.5 (1.9.1)	Pas de support	10.50	Pas de support
<video> : H.264 et MP3 en MP4		(Oui) <sup>[10]</sup>	(Oui) <sup>[10]</sup>	9.0	(Oui)	(Oui)
<video> : H.264 et AAC en MP4		(Oui) <sup>[4]</sup>	(Oui) <sup>[11]</sup>	9.0	(Oui)	3.1
<video> : FLAC en MP4		62	51 (51)	?	?	?
Tout autre format		Pas de support	Pas de support	Pas de support	Pas de support	3.1 <sup>[12]</sup>

La conclusion de ce bref survol est qu'il est souhaitable d'avoir plusieurs solutions pour publier du multimédia dans vos sites web :

- Pour la vidéo, proposez les formats .mp4 (H.264/MP4) et .webm (VP8/Vorbis).
- Pour l'audio, proposez les formats .mp3 (MP3) et .webm (Vorbis).

## L'insertion d'une vidéo

### 1. L'élément <video>

L'élément <video> permet d'insérer un fichier vidéo dans vos pages web :

```
<video>
  ...
</video>
```

### 2. La source de la vidéo

Dans l'élément <video>, c'est l'attribut `src` qui indique le chemin d'accès à la source du fichier vidéo à utiliser.

```
<video src="venise.mp4"></video>
```

Si vous affichez la page web, vous verrez seulement la première image de la vidéo :



Pour le moment, il n'y a aucun moyen d'utiliser la vidéo.

### 3. Les contrôles

Maintenant, il faut donner la possibilité de jouer cette vidéo. Pour cela, vous pouvez ajouter l'attribut booléen `autoplay` à l'élément `<video>` :

```
<video src="venise.mp4" autoplay></video>
```

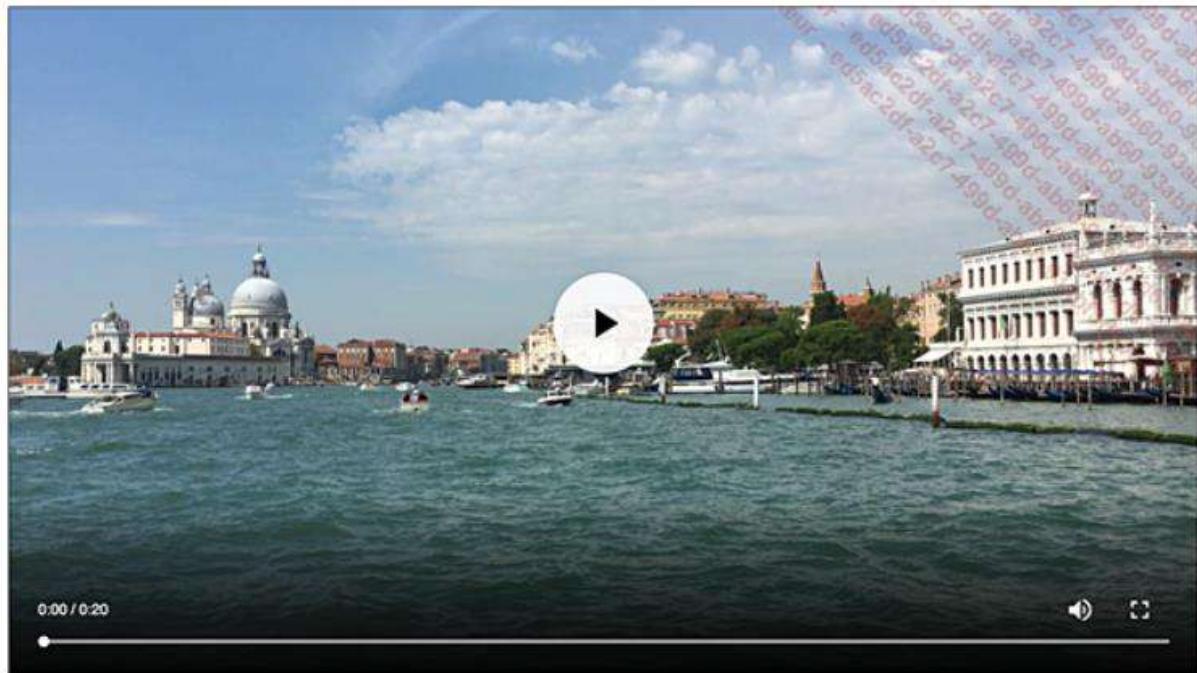
Cet attribut permet de jouer automatiquement la vidéo au chargement de la page. Mais le visiteur n'aura aucun moyen de gérer lui-même la diffusion de la vidéo.

Pour offrir les moyens de contrôler la vidéo, il faut plutôt utiliser l'attribut booléen `controls` :

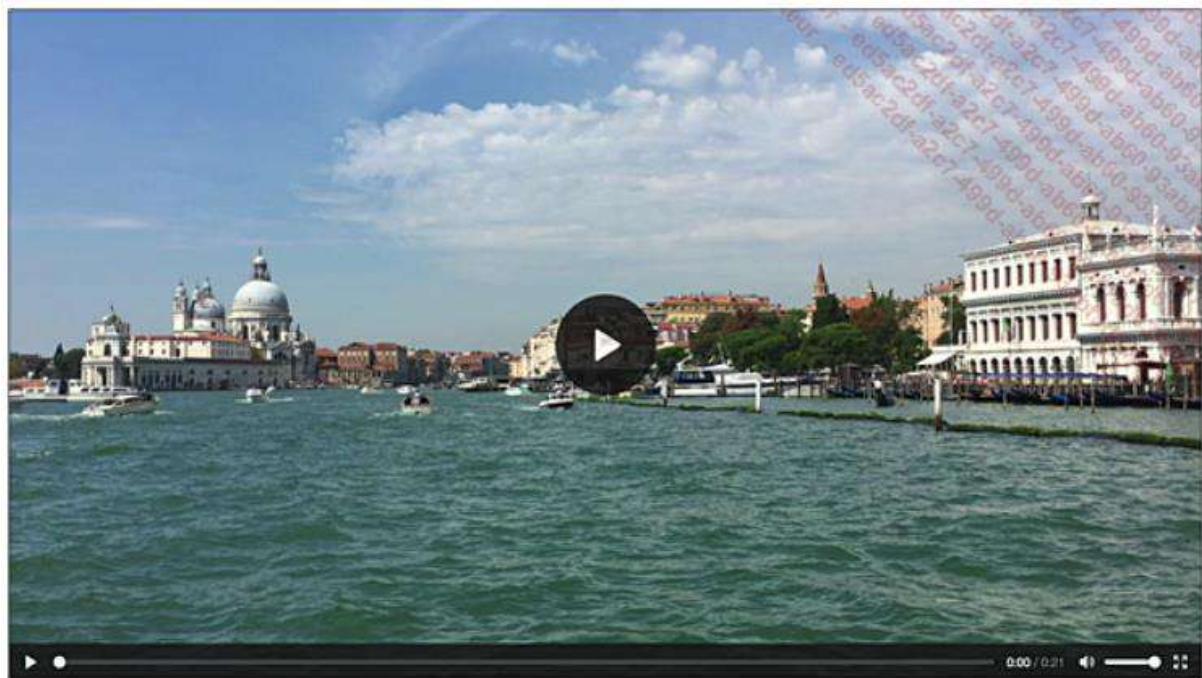
```
<video src="video.mp4" controls></video>
```

Dans ce cas, chaque navigateur devra afficher des contrôles, des boutons de gestion de la vidéo, selon sa propre interface.

Voici l'affichage obtenu avec Google Chrome :



Voici l'affichage obtenu avec Mozilla Firefox :



## 4. Précharger la vidéo

Les vidéos étant souvent des fichiers assez lourds, il est préférable de les précharger avant que l'utilisateur ne les joue. Il aura ainsi un temps de chargement moins long, donc une meilleure expérience utilisateur.

Pour ce faire, dans l'élément <video>, il faut ajouter l'attribut `preload` qui accepte trois valeurs :

- `auto` indique que c'est au navigateur de télécharger les données dont il a besoin pour précharger au mieux la vidéo.
- `metadata` spécifie au navigateur qu'il faut télécharger les métadonnées de la vidéo, afin d'avoir les renseignements techniques la concernant : dimension, durée...
- `none` indique qu'il ne faut pas précharger le fichier vidéo. Le navigateur n'aura alors aucune information technique sur le fichier, pas de dimension, pas de durée...

```
<video src="video.mp4" controls preload="auto"></video>
```

## 5. Afficher une image d'ouverture

Au chargement de la vidéo, vous pouvez souhaiter ne pas afficher la première image de la vidéo mais plutôt une image que vous avez créée avec un logiciel graphique. Une fois réalisée et exportée, vous allez pouvoir l'afficher avec l'attribut `poster` :

```
<video src="video.mp4" controls poster="intro-venise.jpg"></video>
```

## 6. Spécifier les dimensions

Si vous désirez réduire les dimensions de la vidéo, utilisez les attributs `width` et `height`, ou un seul de ces deux attributs :

```
<video src="video.mp4" controls poster="intro-venise.jpg"
width="320"></video>
```

Il vaut toutefois mieux le faire lors de l'exportation de la vidéo source pour des raisons évidentes de performance.

## 7. Proposer plusieurs sources

Nous l'avons vu en début de chapitre, il est préférable de proposer plusieurs sources pour la vidéo, pour une meilleure compatibilité avec les navigateurs. Pour ce faire, nous devons indiquer les sources comme contenu de l'élément `<video>` :

```
<video controls preload="auto">
  <source src="venise.mp4"/>
  <source src="venise.webm"/>
</video>
```

Dans cet exemple, nous avons deux sources disponibles : l'une au format .mp4, l'autre au format .webm. C'est le navigateur qui fera le choix du format à utiliser. Si les deux formats sont utilisables, la première source indiquée dans le code est utilisée.

Si le navigateur de certains internautes est trop ancien et n'interprète pas l'élément `<video>`, nous devons proposer une solution de repli. Pour cela, il suffit d'indiquer le problème dans un élément `<p>` par exemple.

```
<video controls preload="auto">
    <source src="venise.mp4"/>
    <source src="venise.webm"/>
    <p>Votre navigateur est trop ancien pour lire ces vidéos</p>
</video>
```

## 8. Jouer en boucle et désactiver le son

Dans l'élément `<video>`, vous pouvez utiliser les attributs booléens `loop` pour que la vidéo se joue en boucle et `muted` pour désactiver le son.

## L'insertion d'un fichier audio

L'insertion d'un fichier audio est très similaire à ce que nous venons de voir pour la vidéo.

L'élément à utiliser est `<audio>` et l'attribut `src` indique quel est le fichier à utiliser. Il faut ajouter l'attribut `controls` pour afficher les boutons de contrôle de l'audio. Nous retrouvons les attributs `autoplay`, `loop` et `preload`.

```
<audio src="musique.mp3" controls preload autoplay loop></audio>
```

Voici l'affichage obtenu dans Google Chrome



Comme précédemment, vous pouvez proposer plusieurs sources, avec différents formats de diffusion :

```
<audio controls>
    <source src="musique.mp3">
    <source src="musique.webm">
</audio>
```

Et pour finir, vous pouvez aussi indiquer un message pour les anciens navigateurs :

```
<audio controls>
    <source src="musique.mp3">
    <source src="musique.webm">
        <p>Votre navigateur est trop ancien pour lire ces fichiers audio</p>
</audio>
```