

# HTML5

## Premiers pas

Support de cours Version 1.2

## Sommaire

Sommaire .....	2
1 Introduction.....	5
1.1 Définition et Historique .....	5
1.2 Le HTML5 dans tout ça .....	7
1.3 Compatibilité des navigateurs.....	8
1.4 Pour les lecteurs pressés .....	10
2 Introduction à HTML5 .....	11
2.1 Concrètement, HTML, c'est quoi ? .....	11
2.2 Quels outils pour créer du HTML ?.....	14
2.2.1 Editeurs de code et IDE .....	14
2.2.1 Editeurs en ligne .....	16
2.3 Quelques principes de base .....	19
2.3.1 une balise fermante par balise ouverte .....	19
2.4 Présentation de balises usuelles .....	20
2.3.1 Balise « div » .....	20
2.3.2 Paragraphe .....	21
2.3.3 Saut de ligne .....	22
2.3.4 Listes .....	23
2.3.5 Liens.....	29
2.3.6 Tableaux .....	33
2.3.7 Images .....	41
2.3.8 Vidéo et Audio .....	43
2.3.9 Span .....	46
2.3.10 Autres balises .....	46
2.3.11 Les commentaires .....	48
2.3.12 iFrame.....	49
2.5 Anatomie d'une page HTML5 .....	51
2.5.1 Principes généraux .....	51
2.5.2 L'exception IE .....	53

2.6 Formulaire .....	57
2.6.1 Introduction.....	57
2.6.2 Les champs de saisie « historiques » .....	61
2.6.3 Attributs « historiques » importants .....	67
2.6.4 Nouveaux types « input » du HTML5 .....	69
2.6.5 Nouveaux attributs HTML5 .....	72
2.6.6 Nouvelles balises HTML5 .....	77
2.6.7 Balises auto-fermantes.....	79
2.6.8 Balises et attributs HTML dépréciés (obsolètes) .....	80
2.7 Structure d'une page HTML dans la « vraie vie » .....	81
2.7.1 Principes et vocabulaire .....	81
2.7.2 Structure type d'une page HTML .....	83
2.7.2 Outils de construction d'un mock-up .....	87
3. Liens utiles .....	89
4. Annexe.....	90
4.1 Lorem ipsum.....	90
4.2 Squelette de page HTML type .....	92
5. Changelog.....	94

Notes de l'auteur :

Je m'appelle Grégory Jarrige.

Je suis développeur professionnel depuis 1991. Après avoir longtemps travaillé sur des gros systèmes et des langages et technos propriétaires, j'ai fait le pari de me former aux technos et langage open source vers 2005-2006. J'ai commencé à développer des applications webs professionnelles à partir de 2007, avant d'en faire mon activité principale à partir de 2010. L'arrivée du HTML5 dans la même période a été pour moi une véritable bénédiction, et surtout un formidable terrain d'expérimentation (avec des API comme Canvas, WebAudio, etc...).

En plus de mon activité de développeur freelance, je suis également formateur - sur des sujets tels que PHP, HTML5, Javascript, SQL – tantôt en entreprise, tantôt dans le cadre de programmes de reconversion (GRETA notamment).

Ce support est une création réalisée en mai 2017 en vue de proposer une introduction rapide à la norme HTML5 pour des personnes débutant en programmation. On trouvera quelques redites dans ce support par rapport au support qui s'intitule « HTML5 APIs & Vanilla Javascript ». Ce dernier support couvre uniquement les changements intervenus avec HTML5 sans revenir aux bases du HTML, il est destiné à des développeurs ayant déjà des notions de HTML.

Ce document est disponible en téléchargement libre sur mon compte Github :

<https://github.com/gregja/JSCorner>

Il est publié sous Licence Creative Commons n° 6. Je souhaite que tout le monde puisse en bénéficier, et particulièrement les membres du meetup « Creative Coding Paris », que j'anime avec quelques amis :

<https://www.meetup.com/fr-FR/CreativeCodeParis>

# 1 Introduction

La norme HTML est une norme assez vieille puisque que ses origines remontent à 1989, et que ses caractéristiques ont été diffusées auprès du grand public à partir de 1991.

## 1.1 Définition et Historique

### Définition de HTML empruntée à Wikipédia :

*L'HyperText Markup Language, généralement abrégé HTML, est le format de données conçu pour représenter les pages web. C'est un langage de balisage permettant d'écrire de l'hypertexte, d'où son nom. HTML permet également de structurer sémantiquement et logiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie, et des programmes informatiques. Il permet de créer des documents interopérables avec des équipements très variés de manière conforme aux exigences de l'accessibilité du web. Il est souvent utilisé conjointement avec le langage de programmation JavaScript et des feuilles de style en cascade (CSS). HTML est initialement dérivé du Standard Generalized Markup Language (SGML).*

Source : [https://fr.wikipedia.org/wiki/Hypertext\\_Markup\\_Language](https://fr.wikipedia.org/wiki/Hypertext_Markup_Language)

Pour un historique complet de HTML, je vous invite à lire l'intégralité de la présentation proposée par Wikipédia.

On notera que HTML a été inventé par Sir Tim Berners-Lee :

[https://fr.wikipedia.org/wiki/Tim\\_Berners-Lee](https://fr.wikipedia.org/wiki/Tim_Berners-Lee)

Sir Tim Berners-Lee dirige le W3C (World Wide Web Consortium), un organisme de standardisation à but non lucratif qui définit et fait évoluer un certain nombre de normes très importantes pour le fonctionnement d'internet.



<http://www.w3.org/TR/html5/>

Le W3C (World Wide Web Consortium), est un organisme de normalisation à but non-lucratif, fondé en octobre 1994, chargé de promouvoir la compatibilité des technologies du World Wide Web telles que : HTML5, HTML, XHTML, XML, RDF, SPARQL, CSS, XSL, PNG, SVG et SOAP.

Un grand nombre d'entreprises partenaires participent plus ou moins directement aux activités du W3C. En règle générale, elles mettent à la disposition du W3C – pour une durée limitée - des experts de différents domaines, qui participent à des groupes de travail. On dénombrait 383 entreprises partenaires en 2013, elles sont 461 en avril 2017.

Pour une présentation détaillée du W3C et de ses activités, on recommandera la lecture de la page Wikipédia en anglais (plus à jour) :

[https://en.wikipedia.org/wiki/World\\_Wide\\_Web\\_Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium)

Le W3C est hébergé en Europe par l'INRIA (Institut national de recherche en informatique et en automatique), et il est activement soutenu par l'Union Européenne.



<http://www.whatwg.org>

Le WHATWG (Web Hypertext Application Technology Working Group) est une organisation fondée en 2004 par un groupe d'experts issus de la Fondation Mozilla, d'Opera Software, et d'Apple. Ce groupe dissident, mené par Ian Hickson, fut fondé en réaction à l'orientation prise par le W3C de développer de nouvelles normes pour HTML, incompatibles avec les anciennes normes HTML 4 et XHTML 1, et ne répondant pas aux besoins très concrets des développeurs et webdesigners.

Début 2007, et après l'annonce par le W3C de l'arrêt du projet de ce qui devait être le nouveau HTML, le WHATWG proposa au W3C qu'un nouveau groupe de travail du W3C prenne en compte les spécifications élaborées par le WHATWG, comme point de départ pour la définition de la nouvelle norme HTML 5. Le W3C accepta la proposition du WHATWG et le démarrage d'un nouveau groupe de travail au sein du W3C fut entériné en mai 2007.

Depuis 2007, le W3C et le WHATWG travaillent en bonne intelligence au développement de la norme HTML 5.

Depuis la mi 2012, les 2 organismes ont convenu d'une nouvelle répartition des tâches :

Le W3C s'occupe de la normalisation du HTML 5

Le WHATWG travaille plus particulièrement sur l'évolution de la norme HTML 5, et sur son

enrichissement par l'élaboration de nouvelles fonctionnalités (définition de nouvelles balises HTML, définition de nouvelles API, etc...).

L'objectif de ces 2 organisations est le suivant : à terme, on ne parlera plus de HTML 4, de XHTML, de HTML 5, mais d'une seule et même norme, HTML, qui couvrira l'ensemble des spécifications couvertes par HTML 5, et normalisées par le W3C.

Il est important de souligner ici que ce cours constitue un « instantané » d'une norme qui est en cours de rédaction et de stabilisation, et même d'évolution si l'on considère également les travaux en cours au sein du WHATWG. (Cf. le tableau de la « timeline » ci-dessous pris sur <http://en.wikipedia.org/wiki/HTML5> ).

Le W3C s'est efforcé de segmenter la spécification en différents chapitres, ou classes, de manière à en faciliter la compréhension. Chacune de ces classes s'est vue attribuer un nom et un logo distinct.

## ***1.2 Le HTML5 dans tout ça***



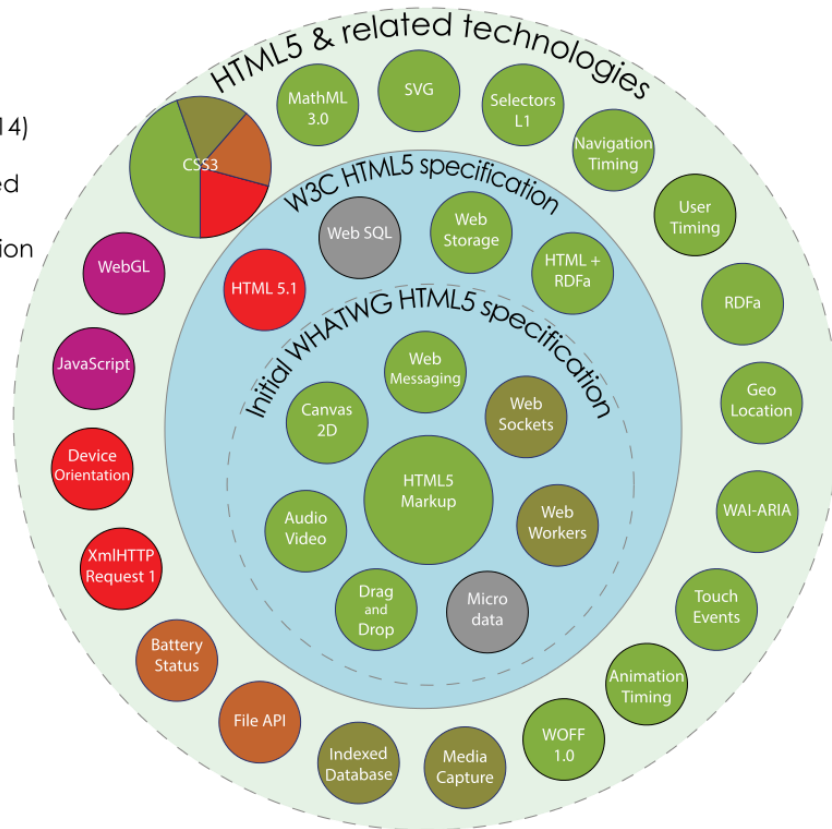
En résumé : La norme HTML5, est une évolution de la norme HTML, dont les premiers éléments ont été diffusés auprès des développeurs et du grand public à partir de 2009. Répondant à une attente forte du marché, et bénéficiant du renouvellement rapide du parc informatique (boosté notamment par le marché des smartphones et des terminaux mobiles), la norme HTML5 a été adoptée rapidement, et elle est aujourd'hui très bien supportée par les navigateurs internet en activité. Il n'y a guère que de vieux PC équipés de vieux navigateurs pas à jour, pour poser des difficultés avec HTML5. Le renouvellement rapide du parc informatique mondial, a contribué à réduire la part des navigateurs ne supportant pas HTML5, même s'il subsiste par ci par là quelques poches de résistance ☺.

Ce graphique emprunté à Wikipédia (<https://fr.wikipedia.org/wiki/HTML5>) montre la couverture fonctionnelle que propose la norme HTML5, soit au travers de sa spécification, soit au travers de projets parallèles venus se greffer dans un second temps (comme WebGL par exemple).

# HTML5

Taxonomy & Status (October 2014)

- Recommendation/Proposed
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated or inactive



Nous allons nous concentrer dans ce support sur l'étude exclusive du rond central qui s'intitule « HTML5 Markup ». C'est la partie qui définit la syntaxe et le fonctionnement des balises HTML5 de base qui composent l'essentiel des pages sur lesquelles nous « surfons » jour après jour.

## 1.3 Compatibilité des navigateurs

En ce qui concerne le support de « HTML5 Markup », les navigateurs récents assurent une compatibilité presque parfaite, même si nous verrons qu'il y a quelques « loupés » sur certaines nouvelles balises ou attributs.

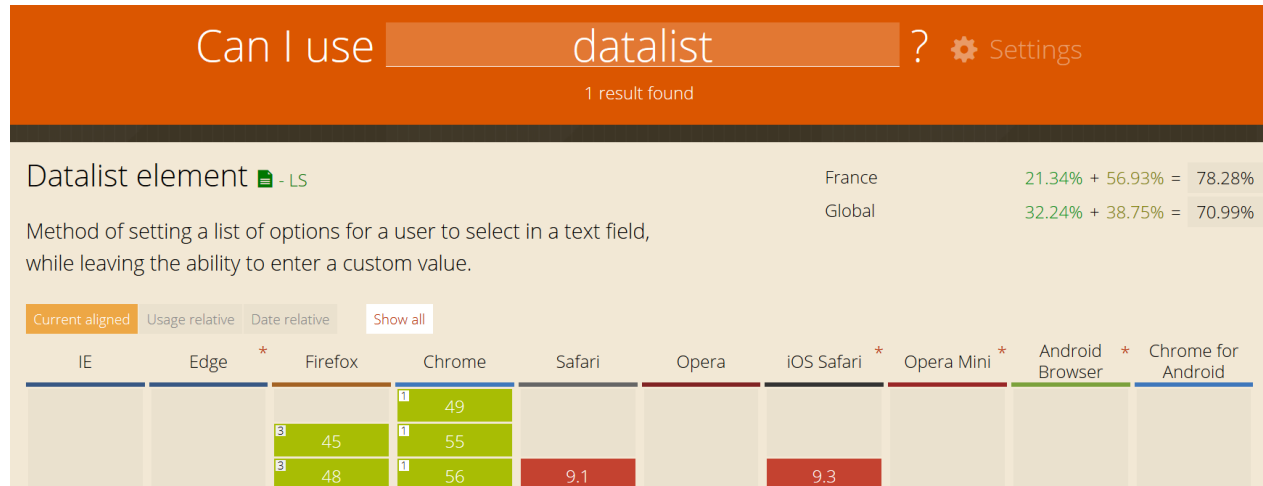
Tous les navigateurs ne sont pas égaux face aux nouvelles fonctionnalités proposées par la



norme HTML5.

Pour savoir si un navigateur - ou une version de navigateur - supporte une fonctionnalité du HTML5, le site de référence est :

<http://caniuse.com>



Autre site intéressant proposant des informations complémentaires :

<http://html5please.com>

**HTML5 PLEASE**

Use the new and shiny responsibly.

Look up HTML5, CSS3, etc features, know if they are ready for use, and if so find out how you should use them – with polyfills, fallbacks or as they are. [tell me more ►](#)

Search:  [link to this search >](#)

**Explore features**

- supported by [IE11+](#) • [IE10+](#) • [IE9+](#) • [IE8+](#) • [IE7+](#) • [no IE](#)
- not supported by [mobile devices](#) • [older mobile devices](#)
- requiring [prefixes](#) • [polyfill](#) • [fallback](#)
- that you should [use](#) • [use with caution](#) • [avoid](#)
- that are [css](#) • [html](#) • [api](#) • [js](#)

Dans ce support, nous nous focaliserons sur l'étude du HTML5, sans rentrer dans les problèmes de compatibilité multi-navigateurs.

## ***1.4 Pour les lecteurs pressés***

Si vous lisez ce document, c'est que vous souhaitez apprendre le HTML, que ce soit par envie ou par nécessité.

J'ai essayé en rédigeant ce document, de laisser de côté le superflu, et de me concentrer sur les éléments qui me semblent essentiels, dans cette norme HTML5. Et je l'ai fait avec mon point de vue de développeur d'application « métier ».

Si vous êtes pressé, vous n'avez pas besoin de maîtriser tout dans le détail, en tout cas pas tout de suite. Concentrez-vous en premier lieu sur les balises HTML5 usuelles (décrites dans le chapitre 2.4). Comprendre le principe des balises « div » et « span », des listes (balises « ul » et « ol »), des liens (balises « a ») et des tableaux (balise « table »), me paraît fondamental. Vous pouvez en revanche laisser de côté les chapitres sur les balises « video » et « audio », ainsi que les « iframe » (et revenir sur ces sujets plus tard, quand vous aurez plus de temps). Je vous recommande aussi de lire attentivement le chapitre sur les formulaires (chapitre 2.6).

Mais je ne peux pas préjuger de vos besoins et de vos priorités. Je vous recommande donc de faire une première lecture en diagonale, pour vous faire une première idée, et ainsi identifier les sujets qui vous intéressent en priorité. Concentrez-vous ensuite sur ces différents sujets. Mais n'oubliez pas que vous avez laissé certains sujets de côté, et que vous pourriez bien y découvrir quelques pépites dont vous n'aviez pas vu l'intérêt au premier abord. Donc, une seconde lecture ne sera probablement pas une perte de temps, mais elle peut intervenir plus tard.

N'oubliez pas aussi qu'une vraie pratique est nécessaire pour maîtriser un sujet comme le HTML. La syntaxe du HTML n'est pas compliquée, elle peut par contre sembler rébarbative (parfois à juste titre). Il n'y a pas de complexité algorithmique dans le HTML, mais construire un tableau HTML n'est pas évident la première fois qu'on en fait un (alors qu'avec un peu de pratique, cela devient presque amusant). De même, construire une liste HTML avec plusieurs niveaux d'imbrication est un gentil casse-tête la première fois qu'on s'y colle, mais avec un peu de pratique cela devient vite une promenade de santé.

Certains de mes étudiants m'ont demandé s'il fallait réellement construire un tableau HTML (ou une liste) à la main, s'il n'y avait pas une meilleure manière de faire... Bien sûr qu'il y a une meilleure manière de faire, elle consiste à utiliser un langage comme PHP ou Javascript, pour construire les tableaux et les listes à partir de jeux de données, qui peuvent par exemple provenir de bases de données SQL, ou de fichiers XML, ou d'autres sources encore. Mais quel que soit le langage utilisé pour la construction du HTML, c'est bien du HTML que l'on doit construire. Si vous maîtrisez les bases du HTML, vous constaterez que la construction de structures telles que des listes ou des tableaux HTML, à partir de langages comme PHP, Javascript, Ruby, Python ou autre... est un jeu d'enfant.

## 2 Introduction à HTML5

### 2.1 Concrètement, HTML, c'est quoi ?

HTML = Hyper Text Markup Language

... que l'on peut traduire en « Langage de balisage pour les hypertextes. »


Un hypertexte est un document numérique contenant des références à d'autres documents qui sont :

- soit accessibles par un clic de souris (hyperlien)
- soit directement inclus dans celui-ci (images, sons, vidéos...)
- soit répartis un peu partout sur internet (d'où la fameuse « toile d'araignée mondiale », ou « World Wide Web »)

Voici un exemple de page web, celle du W3C :

<https://www.w3.org/TR/html5/>

W3C Recommendation



## HTML5

A vocabulary and associated APIs for HTML and XHTML

W3C Recommendation 28 October 2014

**This Version:**  
<http://www.w3.org/TR/2014/REC-html5-20141028/>

**Latest Published Version:**  
<http://www.w3.org/TR/html5/>

**Latest Version of HTML:**  
<http://www.w3.org/TR/html/>

**Latest Editor's Draft of HTML:**  
<http://www.w3.org/html/wg/drafts/html/master/>

**Previous Version:**  
<http://www.w3.org/TR/2014/PR-html5-20140916/>

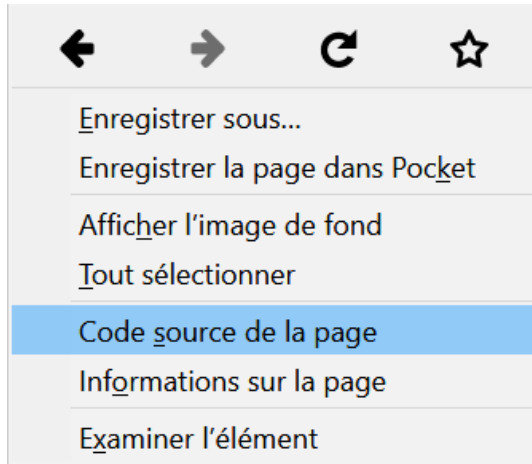
**Previous Recommendation:**  
<http://www.w3.org/TR/1999/REC-html401-19991224/>

**Editors:**  
WHATWG:  
[Ian Hickson](#), Google, Inc.

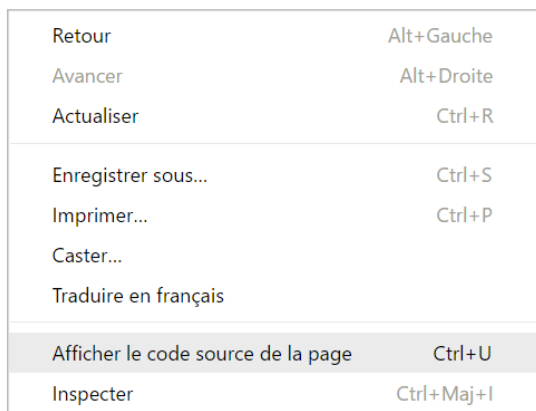
[www.w3.org/html/wg/drafts/html/master/](http://www.w3.org/html/wg/drafts/html/master/)

Les liens cliquables sont en bleu souligné, il s'agit d'une convention bien respectée par la plupart des navigateurs (ces liens pourraient être « relookés » via du code CSS, mais nous verrons ça plus tard 😊).

Faites un clic droit sur la page, vous devriez voir apparaître une fenêtre contextuelle comme celle-ci :



La plupart des navigateurs proposent un menu contextuel comme celui-ci, avec quelques petites différences. La fenêtre ci-dessus est fournie par Firefox, voici celle de Google Chrome :



On voit qu'il y a peu de différences, au moins pour ces options de base.

Sélectionnez l'option « Afficher le code source de la page », vous allez voir apparaître un nouvel onglet présentant un code un peu obscur :

```
1 <!DOCTYPE html><html lang="en-US-x-Hixie"><meta charset="utf-8"><title>HTML5</title><sc
2
3 .applies thead th > * { display: block; }
4 .applies thead code { display: block; }
5 .applies tbody th { white-space: nowrap; }
6 .applies td { text-align: center; }
7 .applies .yes { background: yellow; }
8
9 .matrix, .matrix td { border: hidden; text-align: right; }
10 .matrix { margin-left: 2em; }
11
12 .dice-example { border-collapse: collapse; border-style: hidden solid solid hidden
13 .dice-example caption { width: 30em; font-size: smaller; font-style: italic; paddi
14 .dice-example td, .dice-example th { border: solid thin; width: 1.35em; height: 1.
15
```

Il s'agit du code source de la page HTML que vous étiez en train de consulter il y a quelques instants.

C'est un code un peu confus (surtout pour un débutant), en tout cas ce n'est pas le meilleur endroit pour débiter notre apprentissage du HTML.

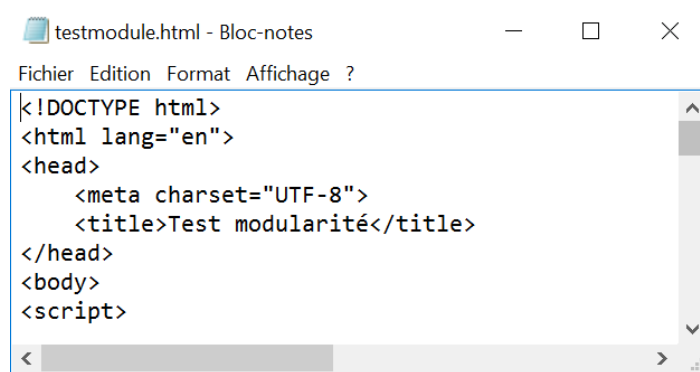
Nous allons voir de quelle manière nous pouvons créer notre propre code HTML, histoire d'apprendre en douceur.

## 2.2 Quels outils pour créer du HTML ?

### 2.2.1 Editeurs de code et IDE

Il existe de nombreux logiciels pour manipuler du HTML.

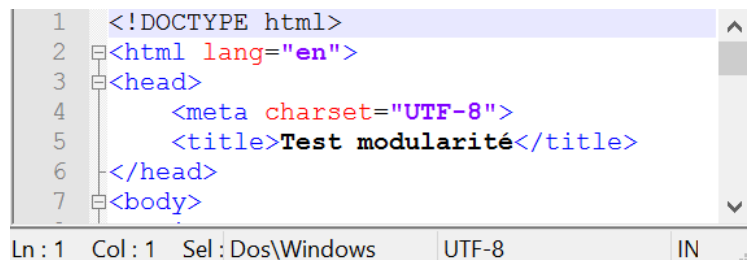
Si vous utilisez un PC sous Windows, vous pourriez utiliser le logiciel « bloc-notes » de Windows, mais ce n'est franchement pas une bonne idée. Voici un exemple de code HTML tel qu'il s'affiche dans le « bloc-notes » :



The screenshot shows a Windows Notepad window titled "testmodule.html - Bloc-notes". The menu bar includes "Fichier", "Edition", "Format", "Affichage", and "?". The text area contains the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Test modularité</title>
</head>
<body>
<script>
```

Voici le même extrait de code affiché avec le logiciel Notepad++ :



The screenshot shows the same HTML code in Notepad++, which applies syntax highlighting. The code is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Test modularité</title>
6 </head>
7 <body>
```

The status bar at the bottom indicates "Ln : 1 Col : 1 Sel : Dos\Windows UTF-8 IN".

Vous voyez que Notepad++ propose une « coloration syntaxique », plus agréable à l'œil. Elle nous permet d'identifier plus facilement les différentes parties du code HTML. Nous les reverrons en détail plus tard, mais notez déjà que :

- les parties en bleu sont des balises HTML
- les parties en rouge sont des attributs de balises HTML
- les parties en violet sont des valeurs d'attributs
- les parties en noir sont du « texte pur »

Notepad++ est un logiciel open-source sous licence GPL. Vous pouvez le télécharger et l'utiliser librement (même si son auteur ne serait pas mécontent de recevoir quelques dons de temps en

temps) :

<https://notepad-plus-plus.org/fr/>

Notepad++ n'est pas le seul logiciel entrant dans la catégorie de ce qu'on appelle les « éditeurs de code », il existe en effet beaucoup de logiciels, qui ont tous des avantages et des inconvénients, des aficionados et des détracteurs.

En voici quelques-uns qui sont excellents :

- Atom :
  - <https://atom.io/>
- SublimeText :
  - <https://www.sublimetext.com/>
- Brackets :
  - <http://brackets.io/>

D'autres logiciels, plus puissants mais aussi plus lourds à installer, entrent dans la catégorie des IDE (Integrated Development Environment => Environnement de Développement Intégré). Ils contiennent tous des éditeurs de code, mais aussi un certain nombre d'options permettant de travailler en mode projet.

- Webstorm et PHPStorm, tous deux édités par JetBrains, ils offrent les mêmes services en ce qui concerne l'écriture du HTML
  - <https://www.jetbrains.com/webstorm/>
- Netbeans :
  - <https://netbeans.org/downloads/>
- Eclipse :
  - <http://www.eclipse.org/downloads/packages/eclipse-php-developers/heliosr>

Chacun des logiciels ci-dessus propose une coloration syntaxique qui lui est propre, ainsi que des menus et raccourcis claviers différents. Et bien sûr, ils proposent tous des assistants, et une multitude d'options très intéressantes à découvrir et à utiliser au quotidien.

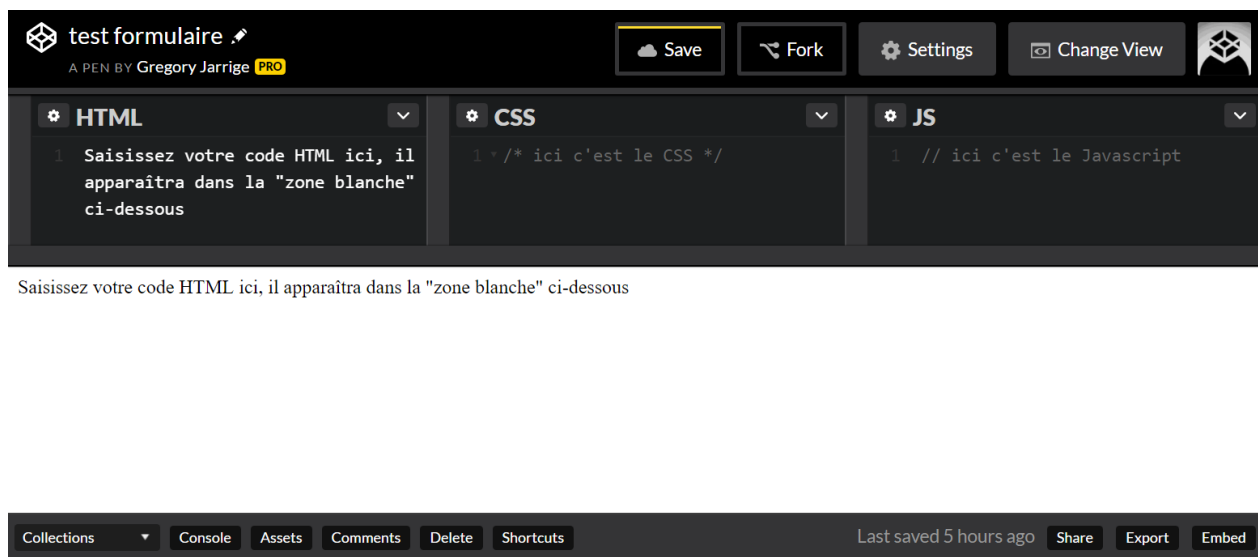
Il faut noter que la frontière entre éditeurs et IDE est assez floue, certains éditeurs précités ont des caractéristiques qui sont très proches de celles des IDE, alors il convient d'être prudent avec ces « étiquettes ».

## 2.2.1 Editeurs en ligne

Mais pour débiter en HTML, on n'est pas obligés d'utiliser l'un ou l'autre de ces logiciels. On peut aussi utiliser des solutions en ligne, telles que :

- Codepen : <https://codepen.io/>
- Codecircle : <https://live.codecircle.com/>

Codepen est un éditeur en ligne gratuit et puissant. Vous pouvez ainsi tester par vous-même tous les exemples de code qui vous intéressent. Codepen propose 3 zones pour saisir le code informatique (HTML, CSS, Javascript), ainsi qu'une vue qui affiche le résultat (la partie blanche en dessous).



Comme nous nous concentrerons dans un premier temps sur le code HTML, vous pouvez réduire la taille des zones « CSS » et « JS », histoire de bénéficier d'un meilleur confort de saisie pour votre code HTML. Vous disposez aussi de l'option « Change View » qui vous permet de choisir d'autres dispositions pour les zones de saisie et d'affichage (à vous de choisir celle qui vous convient le mieux) :

### Editor Layout

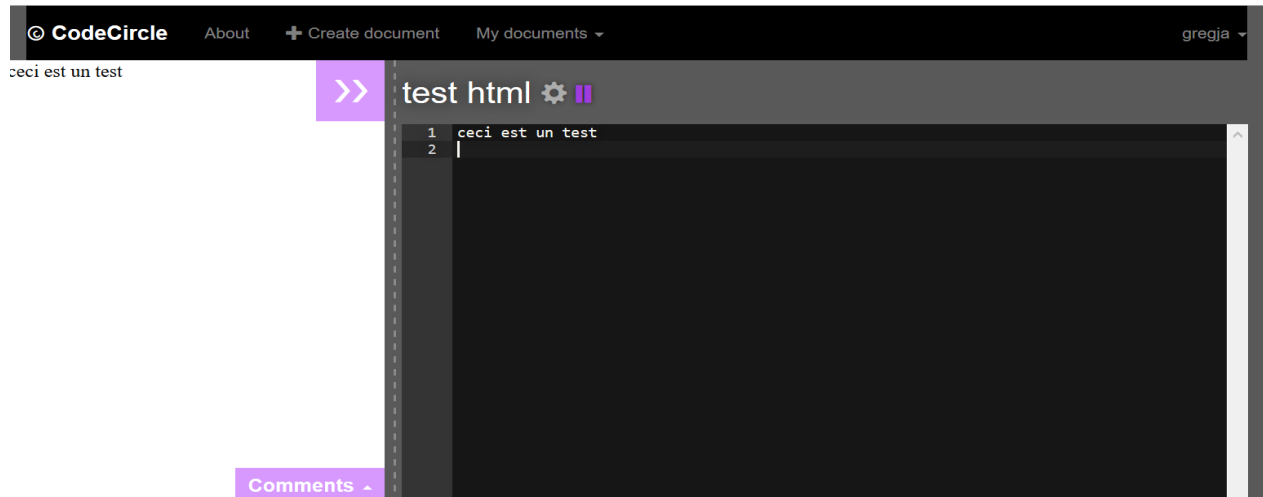




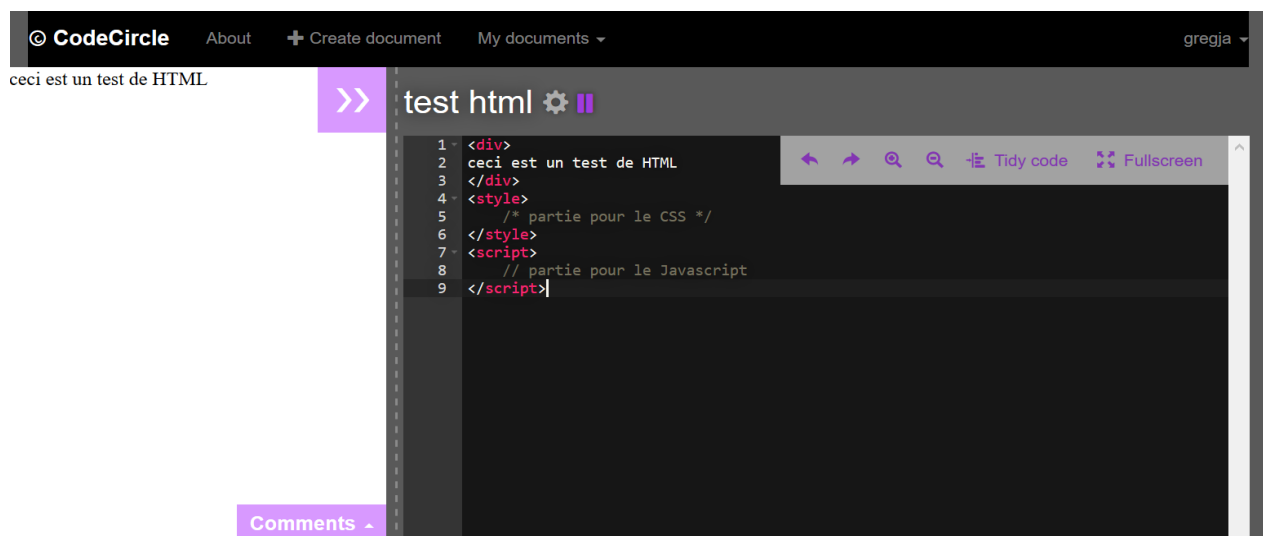
Il convient maintenant de parler de Codecircle.

Live.codecircle.com est une plateforme développée par la « Goldsmiths University of London » pour ses étudiants en programmation. Elle est ouverte gratuitement au grand public depuis mars 2017.

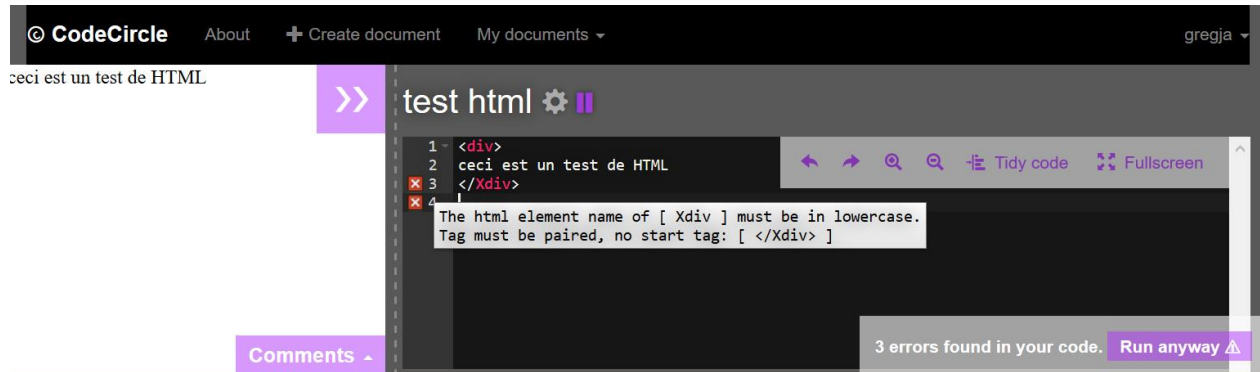
Cette plateforme est particulièrement bien adaptée à l'apprentissage du HTML, du CSS et du Javascript, je me devais donc de la présenter ici.



On peut aussi placer du code HTML, CSS et Javascript, comme avec Codepen, mais dans ce cas on définit nous même les parties de code au moyen des balises « style » et « script ». Pour la partie HTML, on utilise généralement la balise « div », mais c'est plus une convention qu'une obligation (ça permet surtout de « clarifier » son code).



Comme Codecircle est une plateforme conçue spécifiquement pour l'enseignement, il y a un « truc » vraiment bien, c'est la manière dont la plateforme nous signale les erreurs :



Nous détaillerons plus tard les causes de l'erreur ci-dessus.

Nous utiliserons alternativement Codepen et Codecircle, dans la suite de ce cours, histoire de nous familiariser avec ces deux solutions, et avec leurs avantages et inconvénients.

## 2.3 Quelques principes de base

### 2.3.1 une balise fermante par balise ouverte

Le composant de base du HTML, c'est la balise.

Exemple de balise : la « div » qui est une balise « fourre-tout »

```
<div>
```

Ceci est le contenu de la « div » que l'internaute verra apparaître dans sa page

```
</div>
```

Nous reviendrons dans le chapitre suivant sur le sens de cette balise, mais nous remarquons tout de suite un point important : une balise « ouverte » (comme ici la « div ») doit être « refermée » par une balise de fermeture (dans le cas d'une « div », la balise de fermeture est « /div »).

Autre exemple, la balise « p » (pour « paragraphe ») doit elle aussi être refermée par une balise de fermeture correspondante (dans ce cas, il s'agit de « /p »).

```
<p>
```

Ceci est le contenu du paragraphe que l'internaute verra apparaître dans sa page

```
</p>
```

Avec la vieille norme XHTML, cette règle – à toute balise doit correspondre une balise fermante – était une « règle d'or » (quoique pas toujours bien respectée).

Avec la vieille norme HTML 4, il y avait quelques exceptions à cette règle, mais en général les développeurs respectaient cette règle du « un pour un », car en cas d'oubli les conséquences pouvaient être désastreuses pour le rendu des pages webs.

Avec HTML5, de nombreuses exceptions ont été introduites à cette règle du « un pour un », on peut théoriquement – et même en pratique - oublier certaines balises de fermeture, les navigateurs le tolèrent et s'en débrouillent. Mais par habitude, par convention, et aussi parce qu'il est plus facile de lire et de maintenir du code HTML bien structuré, une grande majorité de développeurs continuent à respecter la règle du « un pour un » (une balise fermante pour chaque balise ouverte). Cela permet aussi de garantir un meilleur rendu des pages sur des navigateurs anciens n'ayant pas bénéficié de mise à jour depuis un certain temps.

## 2.4 Présentation de balises usuelles

Recommandation : Au fur et à mesure que vous avancez dans ce chapitre, prenez le temps de tester chacune des nouvelles balises que vous découvrirez, soit dans Codepen, soit dans Codecircle, voire dans les deux plateformes, et comparez les résultats obtenus.

### 2.3.1 Balise « div »

La balise « div » est une balise « fourre-tout » destinée à subdiviser la page en plusieurs parties plus ou moins indépendantes.

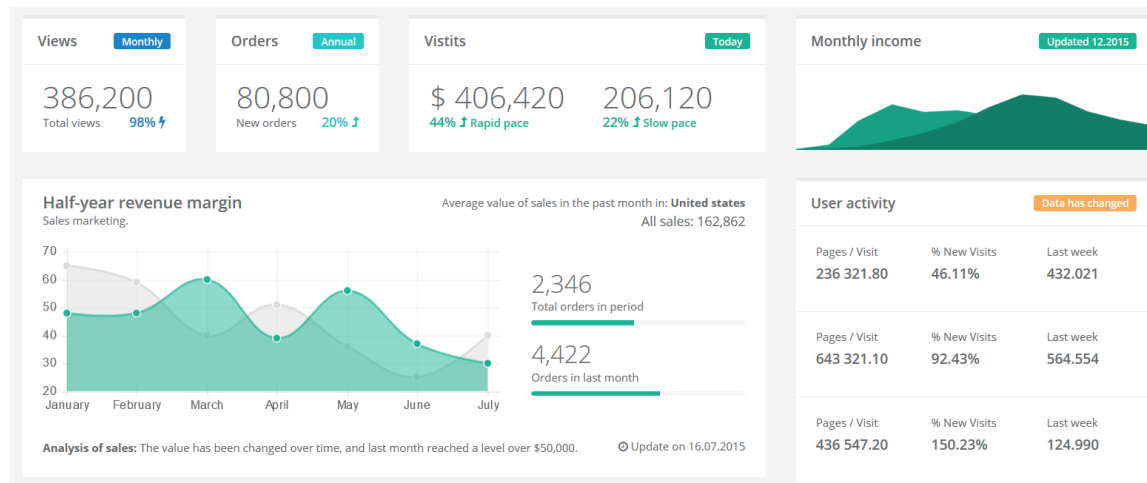
```
<div>  
Ceci est le contenu de la « div » que l'internaute verra apparaître dans sa  
page  
</div>
```

On peut placer à l'intérieur d'une « div » l'ensemble des balises HTML. Par exemple, on peut placer dans une « div » des paragraphes, des listes, des tableaux, des formulaires, etc...

Il est aussi possible d'imbriquer des « div » dans d'autres « div ».

```
<div>  
Ceci est la "div" principale  
  <div>  
    ceci est une "div" secondaire imbriquée dans la précédente  
  </div>  
  <div>  
    ceci est une seconde "div" elle aussi imbriquée dans la précédente  
  </div>  
</div>
```

Les « div » sont très pratiques. Associées à un peu de code CSS, on peut les utiliser pour créer plusieurs blocs distincts au sein d'une page, comme dans l'exemple suivant :



Exemple ci-dessus emprunté au template HTML suivant :

<https://wrapbootstrap.com/theme/inspinia-responsive-admin-theme-WB0R5L90S>

## 2.3.2 Paragraphe

La balise « p » est une balise « paragraphe ». Elle offre le même usage qu'un paragraphe dans un traitement de texte par exemple.

L'exemple ci-dessous présente deux paragraphes distincts se trouvant imbriqués dans la même « div ».

```
<div>
Ceci est la "div" principale
  <p>
    ceci est un paragraphe imbriqué dans une "div"
  </p>
  <p>
    ceci est second paragraphe imbriqué dans la même "div"
  </p>
</div>
```

### 2.3.3 Saut de ligne

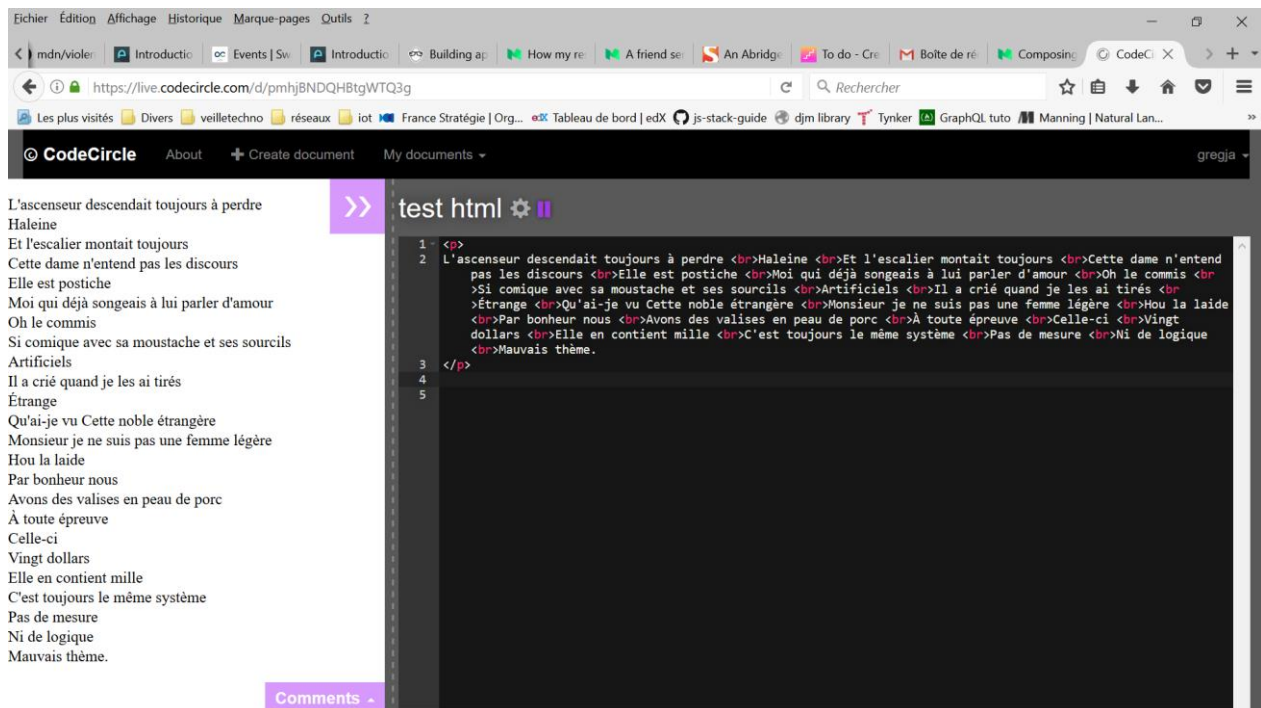
A l'intérieur d'une « div » ou d'un paragraphe, il peut être utile de placer des sauts de ligne, qui constitueront autant de « respirations ».

Pour illustrer notre propos, voici un poème de Louis Aragon (1897-1982), qui s'intitule « Charlot mystique » (extrait du recueil de poèmes : Feu de joie, 1920).

Le code HTML correspondant au poème est le suivant :

```
<p>
L'ascenseur descendait toujours à perdre <br>Haleine <br>Et l'escalier montait
toujours <br>Cette dame n'entend pas les discours <br>Elle est postiche
<br>Moi qui déjà songeais à lui parler d'amour <br>Oh le commis <br>Si comique
avec sa moustache et ses sourcils <br>Artificiels <br>Il a crié quand je les
ai tirés <br>Étrange <br>Qu'ai-je vu Cette noble étrangère <br>Monsieur je ne
suis pas une femme légère <br>Hou la laide <br>Par bonheur nous <br>Avons des
valises en peau de porc <br>À toute épreuve <br>Celle-ci <br>Vingt dollars
<br>Elle en contient mille <br>C'est toujours le même système <br>Pas de
mesure <br>Ni de logique <br>Mauvais thème.
</p>
```

Voici le rendu que l'on obtient dans Codecircle (la partie de gauche correspond à ce que verra l'internaute) :



Amusez-vous à retirer la balise « br » en plusieurs endroits du poème, et observez le résultat.

### 2.3.4 Listes

Les listes sont essentielles, on les utilise beaucoup dans les pages webs, y compris pour générer des menus (sujet que nous aborderons plus tard).

On dispose en HTML de deux balises distinctes :

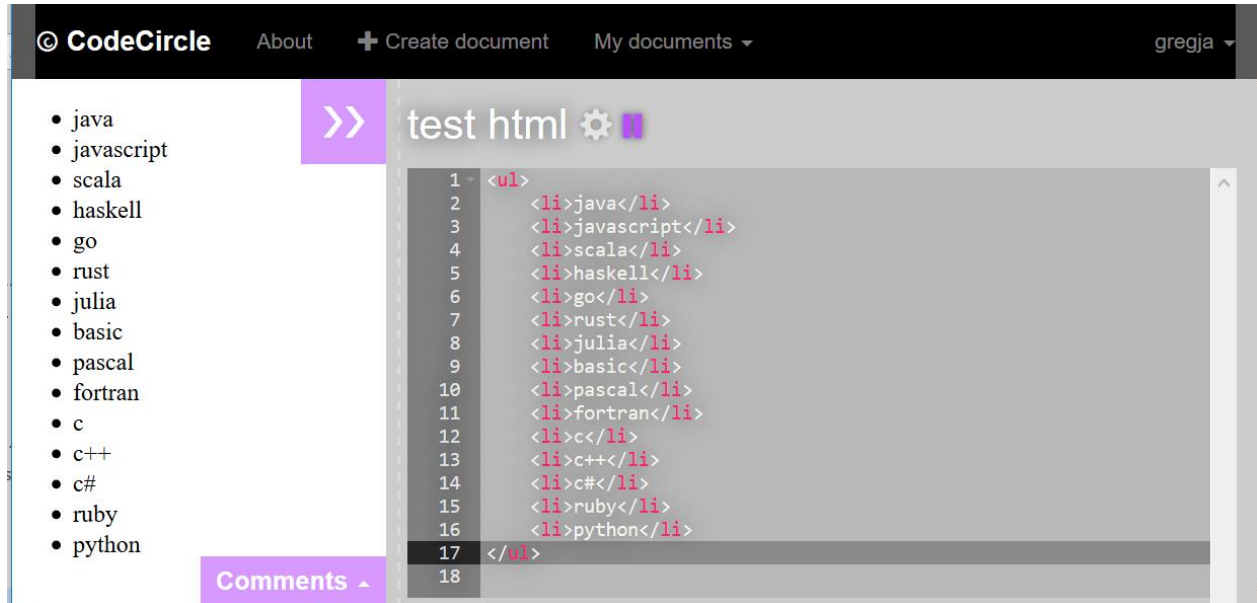
- Les balises « ul » (unordered list) pour les listes non numérotées
- Les balises « ol » (ordered list) pour les listes numérotées

Voici une liste de langages de programmation, intégrée dans un jeu de balises « ul » :

```
<ul>
  <li>java</li>
  <li>javascript</li>
  <li>scala</li>
  <li>haskell</li>
  <li>go</li>
  <li>rust</li>
  <li>julia</li>
  <li>basic</li>
  <li>pascal</li>
  <li>fortran</li>
  <li>c</li>
  <li>c++</li>
  <li>c#</li>
  <li>ruby</li>
  <li>python</li>
</ul>
```

La balise « li » qui est utilisée sur chacun des langages permet d'identifier précisément chacun des éléments de la liste.

Voici le résultat obtenu sur CodeCircle :



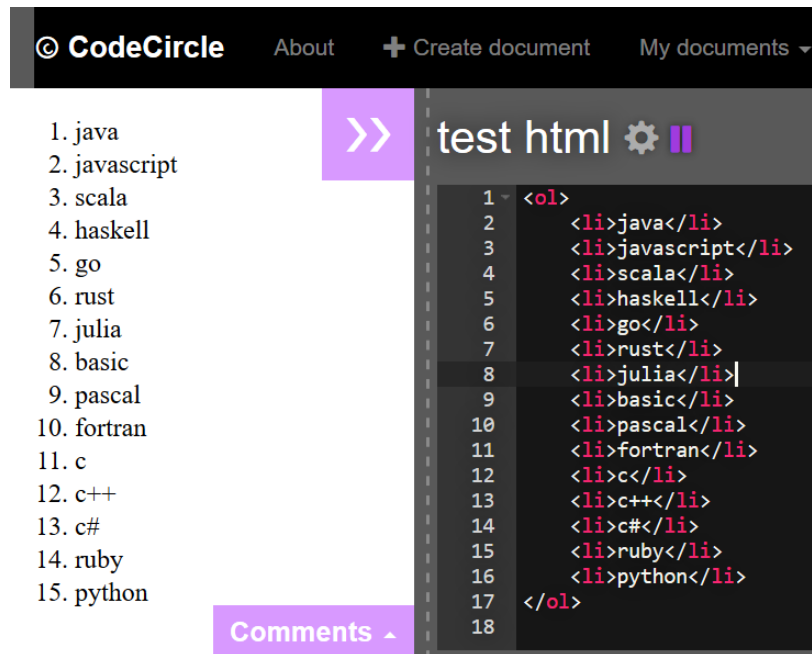
Vous noterez que, si vous enlevez dans le code source de votre liste les sauts de ligne, le résultat demeure inchangé pour le rendu final tel que le voit l'internaute. Par exemple, le code HTML ci-dessous produit le même résultat que le code précédent, il semble différent mais en réalité seuls les sauts de ligne ont été supprimés du code source, ce qui n'a aucune incidence sur le rendu tel que le voit l'internaute :

```
<ul>
<li>java</li><li>javascript</li><li>scala</li><li>haskell</li><li>go</li><li>r
ust</li><li>julia</li><li>basic</li><li>pascal</li><li>fortran</li><li>c</li><
li>c++</li><li>c#</li><li>ruby</li><li>python</li>
</ul>
```

NB : il est important de bien dissocier à ce stade, le code source HTML que vous avez saisi, du résultat final tel qu'il est affiché dans le navigateur de l'internaute. Prenez le temps de faire des essais : ajoutez des blancs et d'autres caractères entre certaines balises, observez comment réagit le navigateur. Vous pouvez faire ces tests dans CodeCircle, qui signale plus clairement que Codepen, d'éventuelles erreurs au niveau du code HTML. N'ayez pas peur de faire des erreurs, ce n'est pas grave, ce qui est intéressant ici c'est d'essayer de voir ce qui fonctionne et ce qui ne fonctionne pas d'un point de vue du code HTML.



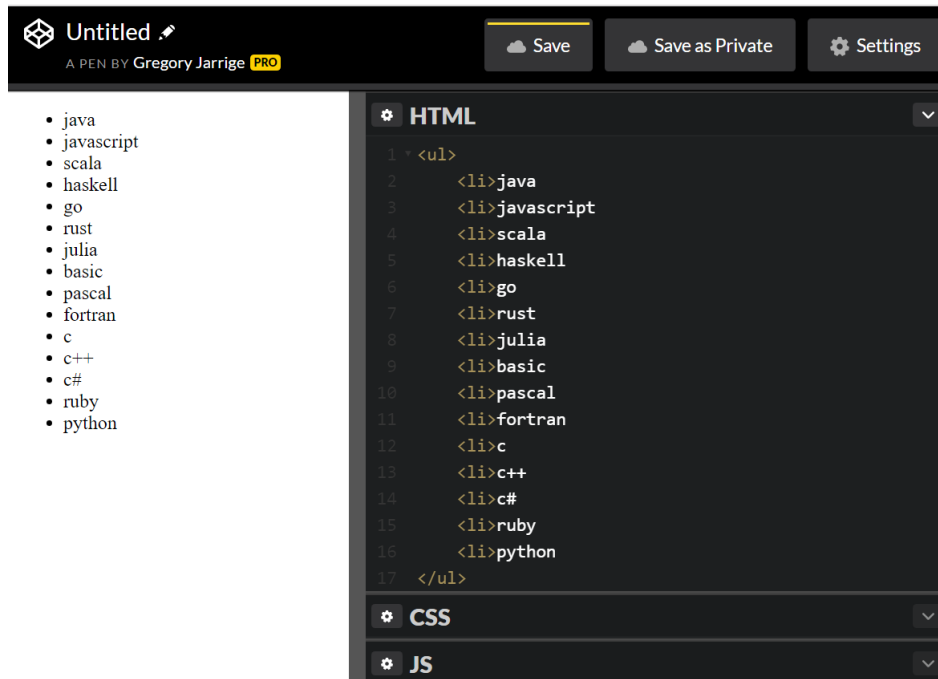
Voici le résultat obtenu quand on remplace la balise « ul » par la balise « ol ». On obtient une liste numérotée :



The screenshot shows the CodeCircle web editor interface. On the left, a list of 15 programming languages is displayed, numbered 1 through 15. The languages are: 1. java, 2. javascript, 3. scala, 4. haskell, 5. go, 6. rust, 7. julia, 8. basic, 9. pascal, 10. fortran, 11. c, 12. c++, 13. c#, 14. ruby, 15. python. On the right, the HTML code for this list is shown in a dark-themed editor. The code starts with an opening `<ol>` tag on line 1, followed by 15 `<li>` elements, each containing the name of a programming language, and ends with a closing `</ol>` tag on line 17. The editor title is 'test html'.

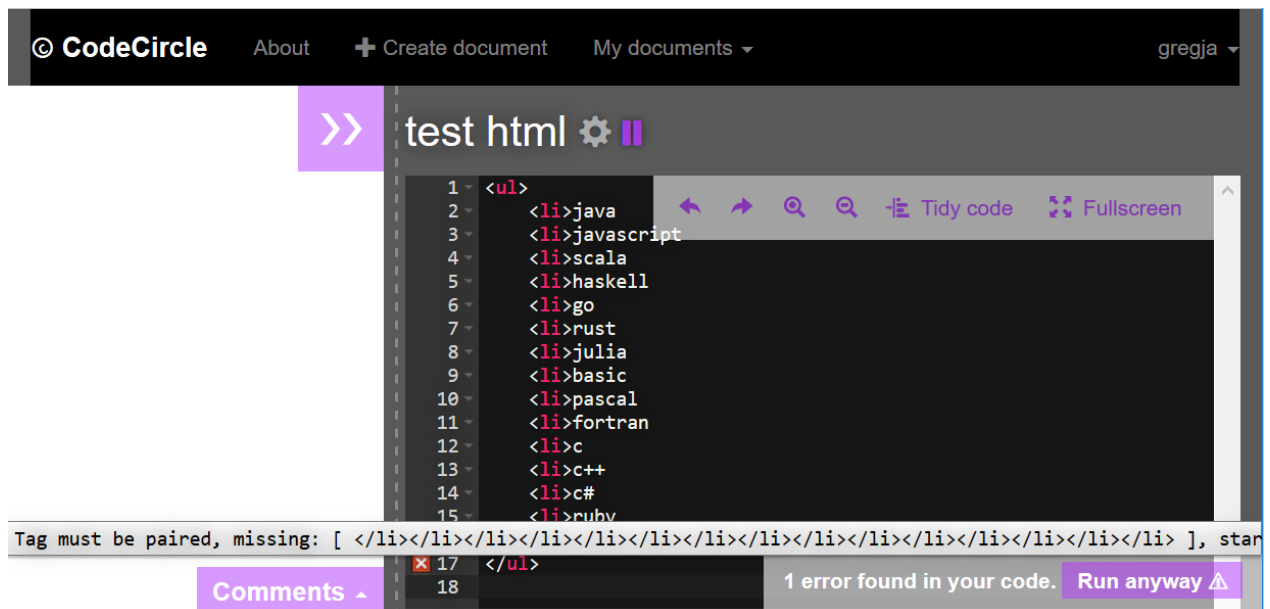
```
1 <ol>
2   <li>java</li>
3   <li>javascript</li>
4   <li>scala</li>
5   <li>haskell</li>
6   <li>go</li>
7   <li>rust</li>
8   <li>julia</li>
9   <li>basic</li>
10  <li>pascal</li>
11  <li>fortran</li>
12  <li>c</li>
13  <li>c++</li>
14  <li>c#</li>
15  <li>ruby</li>
16  <li>python</li>
17 </ol>
18
```

J'indiquais précédemment qu'en HTML5, les balises de fermeture étaient devenues facultatives dans de nombreux cas. Exemple ci-dessous, dans lequel toutes les balises de fermeture « /li » ont été retirées :



```
1 <ul>
2   <li>java
3   <li>javascript
4   <li>scala
5   <li>haskell
6   <li>go
7   <li>rust
8   <li>julia
9   <li>basic
10  <li>pascal
11  <li>fortran
12  <li>c
13  <li>c++
14  <li>c#
15  <li>ruby
16  <li>python
17 </ul>
```

Ca fonctionne bien avec Codepen, mais curieusement, CodeCircle signale une erreur pour ce même code :



```
1 <ul>
2   <li>java
3   <li>javascript
4   <li>scala
5   <li>haskell
6   <li>go
7   <li>rust
8   <li>julia
9   <li>basic
10  <li>pascal
11  <li>fortran
12  <li>c
13  <li>c++
14  <li>c#
15  <li>ruby
16  </ul>
```

Tag must be paired, missing: [ </li></li></li></li></li></li></li></li></li></li></li> ], star

1 error found in your code. Run anyway

CodeCircle nous indique dans son message que nous n'avons pas « appairé » les balises « li » avec des balises « /li ». Ceci est dû au fait que la plateforme CodeCircle a été conçue dans l'optique d'enseigner la programmation aux étudiants. Les concepteurs de la plateforme ont donc fait le choix d'imposer certaines règles, comme par exemple de créer un code HTML bien structuré, respectant la règle déjà évoquée. Ce choix se comprend.

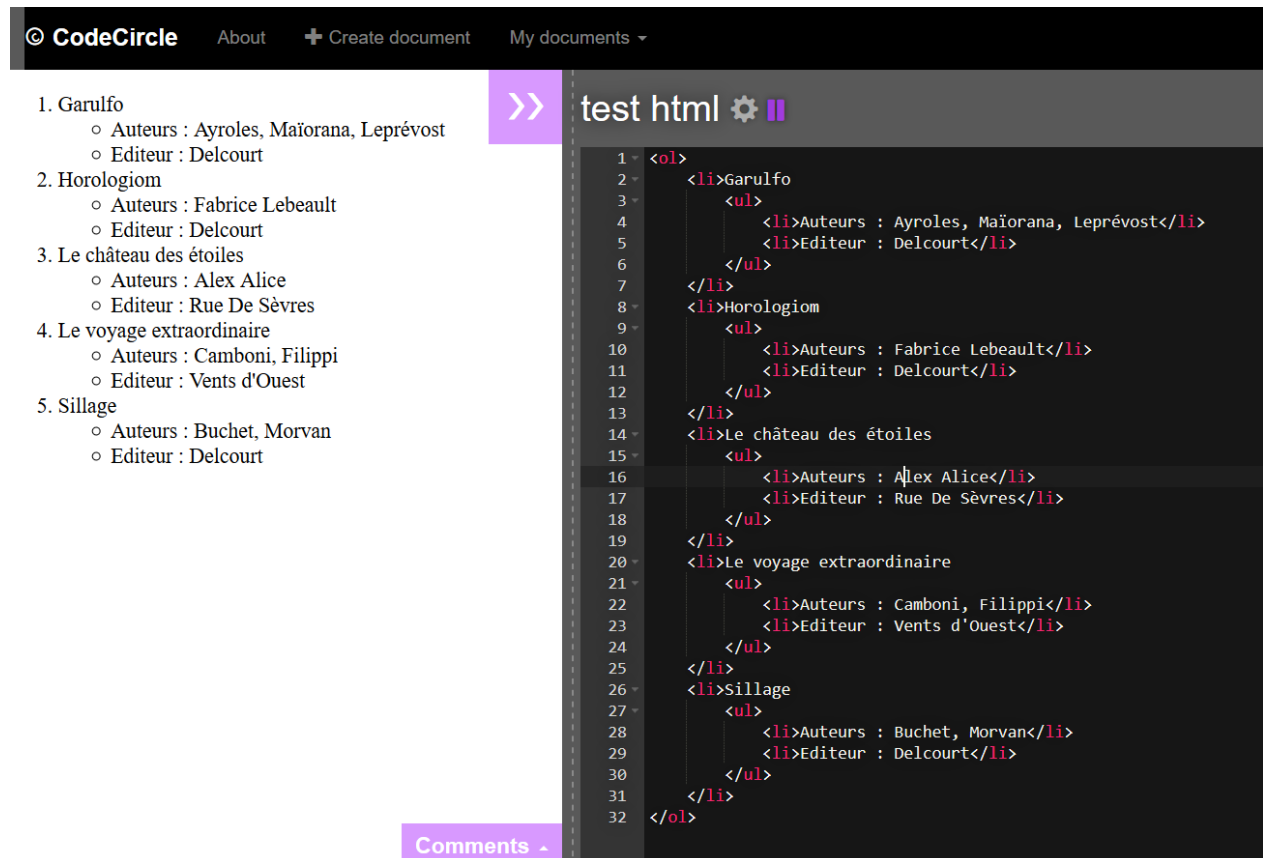
Pour ma part, je dirais que la première justification valable est surtout celle qui consiste à vouloir assurer un rendu optimal sur la majorité des navigateurs, y compris les plus anciens. C'est la raison pour laquelle j'ai tendance à préférer la règle d'une « balise fermante pour chaque balise ouverte ».

Il est possible d'imbriquer des listes dans d'autres listes.

Exemple ci-dessous avec une liste d'albums de bande dessinée :

```
<ol>
  <li>Garulfo
    <ul>
      <li>Auteurs : Ayroles, Maïorana, Leprévost</li>
      <li>Editeur : Delcourt</li>
    </ul>
  </li>
  <li>Horologiom
    <ul>
      <li>Auteurs : Fabrice Lebeault</li>
      <li>Editeur : Delcourt</li>
    </ul>
  </li>
  <li>Le château des étoiles
    <ul>
      <li>Auteurs : Alex Alice</li>
      <li>Editeur : Rue De Sèvres</li>
    </ul>
  </li>
  <li>Le voyage extraordinaire
    <ul>
      <li>Auteurs : Camboni, Filippi</li>
      <li>Editeur : Vents d'Ouest</li>
    </ul>
  </li>
  <li>Sillage
    <ul>
      <li>Auteurs : Buchet, Morvan</li>
      <li>Editeur : Delcourt</li>
    </ul>
  </li>
</ol>
```

Résultat obtenu dans CodeCircle :



Observez de près l'organisation du code HTML. Vous remarquerez notamment que, à l'intérieur de chaque balise « li », on démarre une nouvelle liste avec la balise « ul ». C'est grâce à cette imbrication de balises « ul » ou « ol » que l'on peut créer des listes sur plusieurs niveaux :

```

<li>Sillage
  <ul>
    <li>Auteurs : Buchet, Morvan</li>
    <li>Editeur : Delcourt</li>
  </ul>
</li>

```

Le nombre de niveaux d'imbrication n'est pas limité, mais il est rare que l'on dépasse 3 niveaux d'imbrication (au-delà cela devient complexe à gérer).

Il est intéressant de noter que nous pouvons changer le style des chiffres et des puces. Nous pouvons par exemple remplacer les nombres par des chiffres romains, ou les puces par des carrés. Ce point nécessite quelques notions de CSS, c'est un sujet que nous approfondirons plus

tard, mais pour les curieux et les pressés, je vous propose d'essayer ceci :

```
<li>Sillage
  <ul style="list-style-type:square">
    <li>Auteurs : Buchet, Morvan</li>
    <li>Editeur : Delcourt</li>
  </ul>
</li>
```

TODO : mauvaise pratique à commenter

Vous voyez que nous avons ajouté l'attribut « style » à notre balise « ul », et à cet attribut, nous avons donné la valeur « list-style-type : square ».

Testez ce bout de code et observez ce qui se passe.

Vous pouvez remplacer « square » par l'une des valeurs suivantes : circle, disc, none

Testez ces différentes valeurs, en les appliquant sur plusieurs balises « ul » de notre liste.

Il existe d'autres possibilités de personnalisation, nous les verrons plus tard.

Quelques liens utiles, que vous pourrez étudier à tête reposée :

<https://developer.mozilla.org/fr/docs/Web/CSS/list-style>

[https://www.w3schools.com/html/html\\_lists.asp](https://www.w3schools.com/html/html_lists.asp)

## 2.3.5 Liens

Une des grandes forces de HTML, c'est cette capacité à lier des pages entre elles au moyen de liens « hypertextes ».

La création d'un lien « hypertexte » se fait au moyen de la balise « a » :

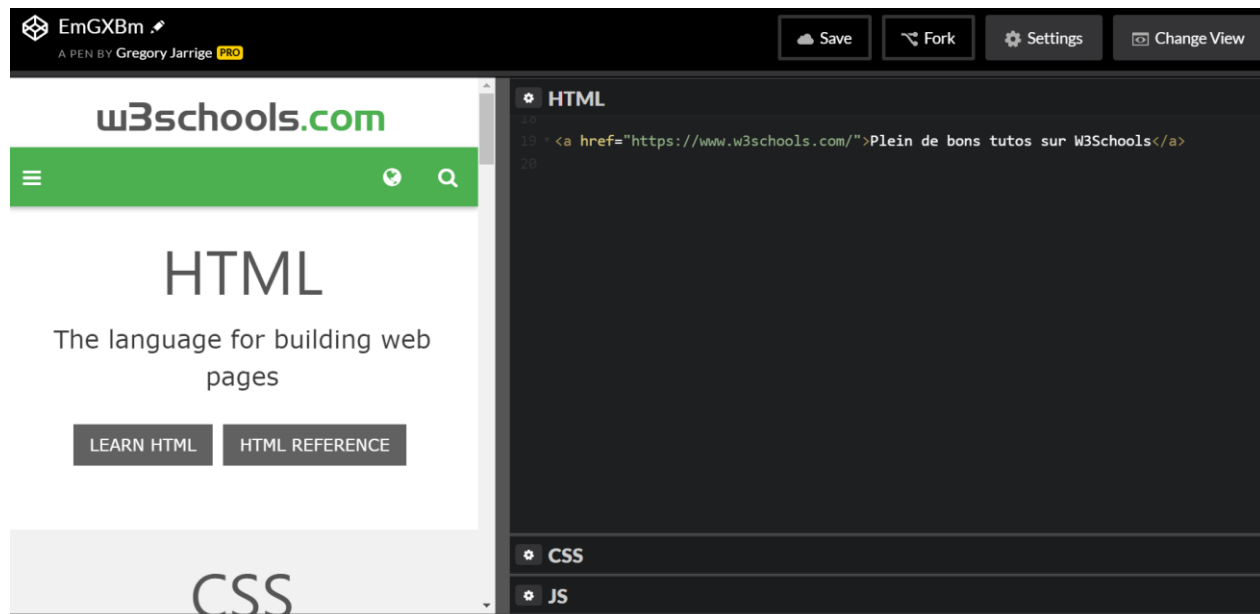
```
<a href="https://www.w3schools.com/">Plein de bons tutos sur W3Schools</a>
```

On voit que la balise « a » a sa propre balise de fermeture « /a ». Entre ces deux balises, on insère le texte de son choix. Avec les anciennes normes HTML4 et XHTML, on était limité à une simple chaîne de caractères, comme dans l'exemple ci-dessus. Mais HTML5 offre plus de possibilités puisque l'on peut désormais placer plusieurs paragraphes ou autres éléments HTML

(comme des images) entre les balises « a » et « /a ».

Autre élément important, l'attribut « href ». C'est avec cet attribut que l'on définit l'URL de la page à laquelle on va accéder quand on cliquera sur le lien.

Saisissez l'exemple précédent dans Codepen ou Codecircle, et observez ce qui se passe. Cliquez sur le lien, vérifiez qu'il fonctionne correctement.



Avec Codepen, vous constatez que la partie de gauche, contient le résultat de notre clic, soit la page de W3Schools que nous avons sélectionnée dans notre attribut « href ».

Dans Codepen, comme dans CodeCircle, la fenêtre « résultat » se comporte comme un mini-navigateur. Ajoutez un espace quelconque dans le code source à gauche, sauvegardez, et vous allez retrouver la page initiale dans laquelle apparaît le lien cliquable.

On voit que, par défaut, le fait de cliquer sur un lien nous faire « perdre » la page courante, qui est remplacée par la nouvelle page dans l'onglet du navigateur. Très souvent, on ne souhaite pas ce comportement, on préfère que le clic sur lien déclenche l'ouverture d'un autre onglet, ceci afin de conserver ouverte la page d'origine. Pour obtenir ce comportement, nous allons utiliser l'attribut « target » :

```
<a href="https://www.w3schools.com/"  
  target="_blank">Plein de bons tutos sur W3Schools</a>
```

Testez cet exemple dans Codepen ou CodeCircle, que constatez-vous ?

Il existe d'autres valeurs possibles pour l'attribut « target », mais c'est la valeur ci-dessus qui est la plus utilisée.

Il peut arriver que vous soyez en train de préparer une page mais que vous ne connaissiez pas encore l'URL d'un lien en préparation. Vous pouvez dans ce cas mettre un dièse comme valeur de l'attribut « href ». Les autres développeurs de votre équipe sauront tout de suite reconnaître le fait que ce lien n'est pas encore opérationnel :

```
<a href="#">Lien en préparation</a>
```

Plus important, il peut arriver qu'un lien pointe non pas vers une autre page, mais vers une autre partie de la page en cours. Pour obtenir cet effet, nous allons recourir à un attribut « id », que nous allons placer sur l'élément vers lequel nous souhaitons revenir si nous cliquons sur un lien. Pour illustrer notre propos, reprenons notre liste de bandes dessinées.

Sur la première balise « ol », placez un attribut « id » avec la valeur « debut » :

```
<ol id="debut">
  <li>Garulfo
    <ul>
      <li>Auteurs : Ayroles, Maïorana, Leprévost</li>
      <li>Editeur : Delcourt</li>
    </ul>
  </li>
  <li>Horologiom
    <ul>
      <li>Auteurs : Fabrice Lebeault</li>
      <li>Editeur : Delcourt</li>
    </ul>
  </li>
  <li>Le château des étoiles
    <ul>
      <li>Auteurs : Alex Alice</li>
      <li>Editeur : Rue De Sèvres</li>
    </ul>
  </li>
  <li>Le voyage extraordinaire
    <ul>
      <li>Auteurs : Camboni, Filippi</li>
      <li>Editeur : Vents d'Ouest</li>
    </ul>
  </li>
  <li>Sillage
    <ul style="list-style-type:square">
      <li>Auteurs : Buchet, Morvan</li>
      <li>Editeur : Delcourt</li>
    </ul>
  </li>
</ol>
```

Après le code HTML relatif à la liste d'albums, ajoutez une balise « a » avec pour attribut « href » la valeur « #debut » :

```
<a href="#debut">Début de la liste</a>
```

Le dièse (#) est ce qu'on appelle un « sélecteur ». Nous le retrouverons à de nombreuses reprises par la suite, aussi bien en CSS qu'en Javascript. Ce sélecteur dièse signifie que notre lien défini dans l'attribut « href » doit pointer vers un élément de la page ayant pour identifiant (attribut « id ») la valeur définie à droite du dièse (en l'occurrence « debut »).

Testez ceci dans Codepen. Dans CodeCircle le résultat est bizarre, la plateforme CodeCircle ne gère visiblement pas très bien cette technique, qui il est vrai est assez peu utilisée. Le résultat n'est pas flagrant car la liste est un peu courte. Pour mieux voir l'effet produit, dupliquez quelques éléments de la liste, par exemple dupliquez plusieurs fois la bande dessinée « Sillage », histoire d'avoir une liste dépassant la taille de la page, puis retestez le lien en cliquant dessus. Vous devriez obtenir un effet de « scrolling », c'est-à-dire de défilement (vers le haut), le navigateur repositionnant la page en haut de la liste des bandes dessinées.

Il y a un attribut que nous n'avons pas encore vu, c'est « title ». Cet attribut est facultatif et pas toujours nécessaire, mais il peut contribuer à améliorer l'accessibilité du site pour des personnes mal voyantes utilisant un logiciel de navigation spécialisé. On utilisera donc cet attribut si on souhaite apporter des informations supplémentaires qui ne sont pas visibles entre la balise « a » et sa balise de fermeture. Exemple :

```
<a href="https://www.w3schools.com/" title="Lien vers le site de W3Schools" target="_blank">Plein de bons tutos sur W3Schools</a>
```

Pour approfondir certains aspects de la balise « a », je vous invite à lire la documentation, et à tester les exemples disponibles sur cette page :

[https://www.w3schools.com/html/html\\_links.asp](https://www.w3schools.com/html/html_links.asp)



### 2.3.6 Tableaux

Les tableaux constituent un élément essentiel du HTML.

On les utilise pour présenter des données dans une page web, un peu à la façon d'un tableau Excel.

On les utilise aussi quelquefois pour présenter des blocs d'information sur une page, en les alignant tous de la même façon, ce qui est facile à faire dans un tableau, un peu plus difficile à faire quand on souhaite le faire avec du CSS. Certains développeurs s'en sont beaucoup servis pour gérer l'alignement de champs de saisie dans un formulaire, mais ce n'est pas forcément une bonne idée car cela a pour effet de rendre le code HTML confus et difficile à maintenir. Nous allons ici nous concentrer sur l'usage « normal » du tableau, à savoir la présentation de données.

Voici un exemple de tableau tel qu'on pourrait en afficher dans une page web :

Mois	Ventes	Situation
Janvier	10000 €	Ventes réalisées (16000 €)
Février	6000 €	
Mars	8000 €	Ventes en cours
Total: 24000 €		

Pour obtenir ce type de rendu dans une page web, nous devons utiliser les balises suivantes : « table », « tr », « td », « th », « tbody », « thead », « tfoot »

Certaines balises sont obligatoires, d'autres facultatives. Nous allons les étudier au travers d'un exemple plus simple, et tendre à nous rapprocher progressivement du tableau des ventes ci-dessus.

Voici un premier exemple de tableau HTML :

```
<table>
  <tr>
    <th>Mois</th>
    <th>Ventes</th>
  </tr>
  <tr>
    <th>Janvier</th>
    <td>10000 €</td>
  </tr>
  <tr>
    <th>Février</th>
    <td>6000 €</td>
  </tr>
  <tr>
    <th>Mars</th>
    <td>8000 €</td>
  </tr>
</table>
```

Voici ce que ça donne dans CodeCircle :

The screenshot shows the CodeCircle web editor interface. On the left, the rendered HTML table is displayed with the following content:

Mois	Ventes
Janvier	10000 €
Février	6000 €
Mars	8000 €

On the right, the source code is shown in a dark-themed editor. The code is as follows:

```
<table>
  <tr>
    <th>Mois</th>
    <th>Ventes</th>
  </tr>
  <tr>
    <th>Janvier</th>
    <td>10000 €</td>
  </tr>
  <tr>
    <th>Février</th>
    <td>6000 €</td>
  </tr>
  <tr>
    <th>Mars</th>
    <td>8000 €</td>
  </tr>
</table>
```

Il faut reconnaître que le rendu est plutôt moche 😞.

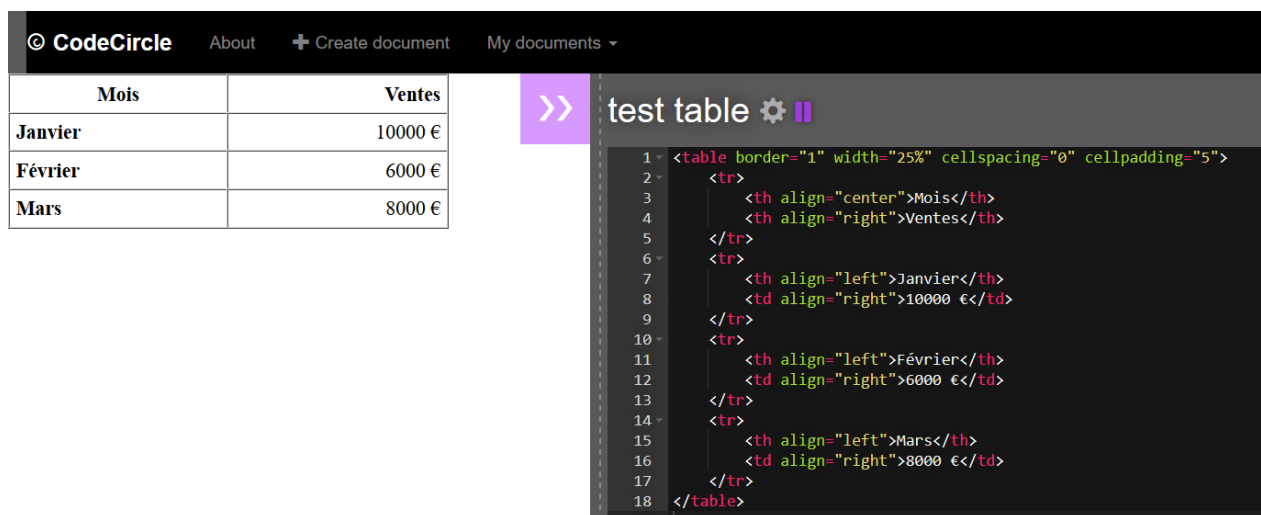
Voyons à quoi servent toutes ces balises :

- La balise « table » est la balise principale de notre tableau HTML, vous constatez qu'elle a une balise de fermeture « /table ». Amusez-vous à supprimer la balise « table » et sa copine « /table » et observez ce qui se passe.
- La balise « tr » permet d'identifier chaque ligne du tableau. Pensez à « tr » comme « table row » (« row » signifiant « ligne » en anglais).
- Les balises « th » et « td » sont des balises de « cellules », un peu comme les cellules d'un tableau Excel
  - o La balise « th » (pour « table head ») désigne une cellule « entête ». Cela peut être un entête de ligne ou de colonne (on a les 2 cas de figure dans notre tableau). Vous remarquez que les balises « th » sont affichées par défaut en gras par le navigateur.
  - o La balise « td » (pour « table detail ») désigne une cellule de donnée simple

Une première manière d'améliorer le rendu du tableau, c'est d'utiliser des attributs, comme par exemple « align » qui permet d'aligner les données dans les cellules, soit à gauche (left), soit à droite (right), soit de les centrer (center).

Nous allons aussi utiliser quelques balises complémentaires, comme « width », « border », « cellspacing » et « cellpadding ».

Exemple :



The screenshot shows a web editor interface with a table on the left and its corresponding HTML code on the right. The table has two columns: 'Mois' and 'Ventes'. The rows are: 'Janvier' with '10000 €', 'Février' with '6000 €', and 'Mars' with '8000 €'. The code on the right is as follows:

```
1 <table border="1" width="25%" cellspacing="0" cellpadding="5">
2   <tr>
3     <th align="center">Mois</th>
4     <th align="right">Ventes</th>
5   </tr>
6   <tr>
7     <th align="left">Janvier</th>
8     <td align="right">10000 €</td>
9   </tr>
10  <tr>
11    <th align="left">Février</th>
12    <td align="right">6000 €</td>
13  </tr>
14  <tr>
15    <th align="left">Mars</th>
16    <td align="right">8000 €</td>
17  </tr>
18 </table>
```

Ca a tout de suite plus d'allure 😊, voyons quelle est la signification de ces nouveaux attributs :

- border : Définit la bordure du tableau (en pixels).
- cellspacing : représente l'espace tout autour des cellules d'un tableau.

- cellpadding : représente la marge entre le contenu et le bord de la cellule.
- width : définit la largeur du tableau par rapport à la page

Cette manière de styliser un tableau HTML n'est pas la meilleure manière de faire, mais elle a le mérite d'être simple, et elle est encore pratiquée aujourd'hui, même si on tend à remplacer cette méthode par du code CSS (que nous étudierons plus tard).

Pour rappel, le premier tableau que je vous ai montré ressemblait à ceci :

Mois	Ventes	Situation
Janvier	10000 €	Ventes réalisées (16000 €)
Février	6000 €	
Mars	8000 €	Ventes en cours
Total: 24000 €		

On voit que la cellule « total » occupe 2 colonnes, et que la seconde cellule dans la colonne « situation » occupe 2 lignes. Pour obtenir cet effet, il nous faut étudier deux nouveaux attributs :

- colspan : définit le nombre de colonnes que la cellule va englober
- rowspan : définit le nombre de lignes que la cellule va englober

Je vous propose de prendre 5 minutes pour essayer d'imaginer de quelle manière vous pouvez modifier votre tableau dans CodeCircle (ou Codepen au choix), de manière à ce qu'il se rapproche de celui-ci (oubliez pour l'instant le fond grisé de certaines cellules, nous verrons ça plus tard). Résistez un petit peu à l'envie d'aller récupérer la solution sur la page qui suit 😊.

Pour info, le site W3Schools propose plusieurs exemples de tableaux utilisant les attributs « colspan » et « rowspan ». Vous pouvez y jeter un coup d'œil, cela devrait vous aider :

[http://www.w3schools.com/tags/att\\_td\\_rowspan.asp](http://www.w3schools.com/tags/att_td_rowspan.asp)

[http://www.w3schools.com/tags/att\\_td\\_colspan.asp](http://www.w3schools.com/tags/att_td_colspan.asp)

Voici le code source du tableau remanié :

```
<table border="1" width="25%" cellspacing="0" cellpadding="5">
  <tr>
    <th>Mois</th>
    <th>Ventes</th>
    <th>Situation</th>
  </tr>
  <tr>
    <th>Janvier</th>
    <td>10000 €</td>
    <td rowspan="2">Ventes réalisées<br>(16000 €)</td>
  </tr>
  <tr>
    <th>Février</th>
    <td>6000 €</td>
  </tr>
  <tr>
    <th>Mars</th>
    <td>8000 €</td>
    <td rowspan="1">Ventes en cours</td>
  </tr>
  <tr>
    <td colspan="2">Total: 24000 €</td>
    <td colspan="1">&nbsp;</td>
  </tr>
</table>
```

Et voici le résultat dans CodeCircle :

The screenshot shows the CodeCircle web editor interface. On the left, the rendered HTML table is displayed. It has three columns: 'Mois', 'Ventes', and 'Situation'. The rows are: 'Janvier' with '10000 €' and 'Ventes réalisées (16000 €)'; 'Février' with '6000 €'; 'Mars' with '8000 €' and 'Ventes en cours'; and a summary row with 'Total: 24000 €' and a blank cell. On the right, the source code is shown in a dark-themed editor, matching the code provided in the previous block. The editor title is 'test table'.

Mois	Ventes	Situation
Janvier	10000 €	Ventes réalisées (16000 €)
Février	6000 €	
Mars	8000 €	Ventes en cours
Total: 24000 €		

```
1 <table border="1" width="25%" cellspacing="0" cellpadding="5">
2   <tr>
3     <th>Mois</th>
4     <th>Ventes</th>
5     <th>Situation</th>
6   </tr>
7   <tr>
8     <th>Janvier</th>
9     <td>10000 €</td>
10    <td rowspan="2">Ventes réalisées<br>(16000 €)</td>
11  </tr>
12  <tr>
13    <th>Février</th>
14    <td>6000 €</td>
15  </tr>
16  <tr>
17    <th>Mars</th>
18    <td>8000 €</td>
19    <td rowspan="1">Ventes en cours</td>
20  </tr>
21  <tr>
22    <td colspan="2">Total: 24000 €</td>
23    <td colspan="1">&nbsp;</td>
24  </tr>
25 </table>
```

On peut encore améliorer un peu notre tableau, en terme de structure et de rendu, avec quelques balises complémentaires, qui sont apparues avec la norme HTML5 :

- caption : balise permettant de définir une légende à notre tableau (balise intéressante mais souvent peu utilisée)
- thead : balise définissant plus précisément l'« entête » du tableau
- tbody : balise définissant plus précisément le « corps » du tableau
- tfoot : balise définissant plus précisément le « pied » du tableau

Et nous allons aussi ajouter des classes CSS avec l'attribut « class », nous allons jouer un peu avec cette notion de CSS, même si nous verrons plus tard, dans le cas du cours CSS, comment ça fonctionne.

Voici ce que ça donne d'un point de vue de notre tableau HTML :

```
<table>
  <caption>Stats de ventes</caption>
  <thead>
    <tr>
      <th class="col_mois">Mois</th>
      <th>Ventes</th>
      <th>Situation</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>Janvier</th>
      <td class="prix">10000 €</td>
      <td rowspan="2">Ventes réalisées<br>(16000 €)</td>
    </tr>
    <tr>
      <th>Février</th>
      <td class="prix">6000 €</td>
    </tr>
    <tr>
      <th>Mars</th>
      <td class="prix">8000 €</td>
      <td rowspan="1">Ventes en cours</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="2" class="total">Total: 24000 €</td>
      <td colspan="1">&nbsp;</td>
    </tr>
  </tfoot>
</table>
```

Voici le code source CSS qui va nous permettre de relooker notre tableau :

```
<style>
  table, th, td {
    border: 1px solid black;
    margin: 5px;
    padding: 1px;
  }
  th {
    text-align: left;
  }
  .prix {
    text-align: right;
  }
  .total {
    text-align: center;
  }
  .col_mois {
    column-width: 100px;
  }
  tr > th {
    background-color: grey;
  }
</style>
```

Voilà le résultat obtenu dans CodeCircle :

© CodeCircle

About

+ Create document

My documents ▾

Stats de ventes

Mois	Ventes	Situation
Janvier	10000 €	Ventes réalisées
Février	6000 €	(16000 €)
Mars	8000 €	Ventes en cours
Total: 24000 €		

>>

test table ⚙ ||

```

1 <style>
2   table, th, td {
3     border: 1px solid black;
4     margin: 5px;
5     padding: 1px;
6   }
7   th {
8     text-align: left;
9   }
10  .prix {
11    text-align: right;
12  }
13  .total {
14    text-align: center;
15  }
16  .col_mois {
17    column-width: 100px;
18  }
19  tr > th {
20    background-color: grey;
21  }
22 </style>
23 <table>
24   <caption>Stats de ventes</caption>
25   <thead>
26     <tr>
27       <th class="col_mois">Mois</th>
28       <th>Ventes</th>
29       <th>Situation</th>
30     </tr>
31   </thead>
```

Comments ▾

Un nouvel attribut est apparu en HTML5, l'attribut "scope", qui est associé aux balises « th ». Cet attribut, qui peut prendre les valeurs "col" ou "row" permet de définir la portée de la ligne ou colonne considérée, par rapport aux cellules adjacentes. Cet attribut apporte une information précieuse aux personnes présentant des déficiences visuelles et qui se font aider par des logiciels d'assistance à la lecture.

Pour un exemple d'utilisation de l'attribut « scope » :

[https://www.w3schools.com/tags/att\\_scope.asp](https://www.w3schools.com/tags/att_scope.asp)

Nous avons vu qu'un tableau HTML se compose d'un certain nombre de balises et d'attributs. Pour synthétiser un peu, voici un squelette de tableau qui reprend les balises les plus couramment utilisées :

```
<table>
  <thead>
    <tr>
      <th>A</th>
      <th>B</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>C</td>
      <td>D</td>
    </tr>
    <tr>
      <td>C</td>
      <td>D</td>
    </tr>
  </tbody>
</table>
```

Je vous ai dit qu'un tableau HTML ressemblait un peu à un tableau Excel. Mais dans Excel, les cellules d'un tableau Excel sont modifiables, alors que ce n'est pas le cas avec notre tableau HTML. Peut-on rendre les cellules d'un tableau HTML modifiables ? Oui, grâce à l'attribut « contenteditable » qui est apparu avec la norme HTML5. Voici un exemple que vous pouvez tester dans CodeCircle ou Codepen :

```
<td class="prix" contenteditable="true">10000 €</td>
```

Vérifiez que le contenu de la cellule que vous avez retouchée est bien modifiable. Pour prendre en compte réellement les modifications faites dans le tableau par l'utilisateur, nous devrions ajouter du code Javascript, mais c'est en dehors du périmètre de ce support de cours HTML. Ce sujet sera abordé dans le cadre d'un autre cours, patience... 😊.



## 2.3.7 Images

Vous vous demandez peut être comment insérer une image dans une page web ?

Voici la réponse :

```

```

Nous verrons un peu plus loin qu'il existe une alternative avec la balise « picture ».

L'inclusion d'une image se fait au moyen de la balise « img », les 2 attributs principaux étant :

- « src » : définit l'emplacement de l'image : cela peut être une image se trouvant sur le PC, ou une image se trouvant sur internet, le format de l'image peut être JPEG, PNG ou Gif.
- « alt » : définit le texte alternatif qui s'affichera à la place de l'image, le temps qu'elle soit « chargée » par le navigateur.

Voici un exemple d'inclusion d'une image réalisée sur Codepen. Cette photo, prise au hasard sur internet, illustre un sport amusant qui nous vient d'Allemagne, le « sporthocker ». La photo est affichée 3 fois, dans 3 formats différents :



Voici le code HTML « copier-collable » pour ceux qui souhaiteraient le tester :

```
  
<br>  
  
<br>  

```

La première image est affichée dans sa taille réelle, les deux suivantes sont affichées dans des tailles plus petites selon deux techniques différentes, mais équivalentes.

Il convient d'être prudent avec le chargement des images. Une image de trop grande taille, même si elle est affichée dans un format plus petit, est chargée intégralement en mémoire, ce qui va avoir pour conséquence de ralentir le chargement global de la page concernée. Il est donc recommandé de redimensionner l'image au plus près du format souhaité à l'affichage.

Cette problématique d'optimisation des images est un sujet à part entière, qui justifierait un chapitre à lui tout seul. Vous pourrez l'approfondir par la suite, il existe de nombreux articles sur le sujet. Je vous invite à faire des recherches sur internet avec des termes tels que :

- optimiser le chargement des images
- lazy loading

Pour retoucher et optimiser vos images, il existe différentes solutions, vous pouvez notamment recourir au logiciel open source Gimp :

<https://www.gimp.org/>

La balise « picture » introduite par le HTML5 permet de gérer le problème des images responsives, c'est-à-dire qu'elle est capable de fournir au navigateur une image dont la résolution est adaptée à l'écran utilisé. La balise utilise pour ce faire la technique des « mediaqueries », qui nous vient du CSS3.

Exemple :

```
<picture>
  <source srcset="image1.jpg" media="(max-width: 500px)">
  <source srcset="image2.jpg" media="(min-width: 501px) and (max-width: 712px)">
  <source srcset="image3.jpg" media="(min-width: 713px)">
  
</picture>
```

La balise « picture » a fait couler beaucoup d'encre (virtuelle), car le fait qu'elle ne soit pas supportée par Internet Explorer, et qu'elle soit encore expérimentale sur certains navigateurs, est un véritable problème.

On notera que « picture » n'est pas utilisé uniquement pour gérer des problématiques de taille, car on peut l'utiliser aussi pour proposer au navigateur un panel de formats

d'images, à charge pour ce dernier de charger le format d'image qui lui convient le mieux :

```
<picture>
  <source type="image/svg" src="logo.svg" />
  <source type="image/png" src="logo.png" />
  
</picture>
```

On voit dans l'exemple ci-dessus qu'il est possible de charger un fichier SVG via cette technique.

Vous verrez quelquefois des sites dans lesquels les images (JPG ou PNG) sont incorporées directement dans le code source HTML de la page. Le navigateur fait ainsi l'économie d'une requête : il n'a en effet pas besoin d'aller rechercher l'image sur un serveur, puisqu'elle est intégralement intégrée au code HTML via l'attribut « src ». Voici le type de code que l'on obtient pour une image de type JPEG :

```

```

Pour de plus amples précisions sur le fonctionnement de la balise « picture » :

<https://developer.mozilla.org/fr/docs/Web/HTML/Element/picture>

<https://afarkas.github.io/lazysizes/>

[https://www.w3schools.com/tags/tag\\_picture.asp](https://www.w3schools.com/tags/tag_picture.asp)

<https://www.html5rocks.com/en/tutorials/responsive/picture-element/>

<https://webdesign.tutsplus.com/tutorials/quick-tip-how-to-use-html5-picture-for-responsive-images--cms-21015>

<https://la-cascade.io/images-responsives-picture-et-srcset/>

### 2.3.8 Vidéo et Audio

Il est possible de charger des sons et des vidéos dans une page web, au moyen des balises « audio » et « video », qui sont apparues avec la norme HTML5. Ces balises sont très amusantes à utiliser. Nous ne les étudierons pas ici dans le détail, mais nous verrons quelques exemples de

syntaxe, et je vous inviterai à lire différents articles si vous souhaitez approfondir ces sujets.

### 2.3.8.1 Vidéo

La balise « video » fonctionne de la façon suivante :

```
<video width="400" controls>  
  <source src="mov_bbb.mp4" type="video/mp4">  
  <source src="mov_bbb.ogv" type="video/ogg">  
  Votre navigateur ne supporte pas la balise video.  
</video>
```

Vous pouvez remplacer l'attribut « controls » par l'attribut « autoplay », éventuellement complété de l'attribut « loop » :

```
<video width="400" autoplay loop>  
  <source src="mov_bbb.mp4" type="video/mp4">  
  <source src="mov_bbb.ogv" type="video/ogg">  
  Votre navigateur ne supporte pas la balise video.  
</video>
```

Explication : tous les formats vidéos ne sont pas supportés par tous les navigateurs, mais la balise « video » permet de définir plusieurs sources, avec des formats différents. Le navigateur va rechercher dans les sources le premier format qu'il sait gérer, et télécharger le fichier vidéo correspondant.

On peut tester cette balise avec la vidéo du dessin animé « big Buck Bunny », disponible en téléchargement gratuit dans différents formats :

[http://download.blender.org/peach/bigbuckbunny\\_movies/](http://download.blender.org/peach/bigbuckbunny_movies/)

Pour de plus amples précisions sur le fonctionnement de la balise « video », je recommande la lecture des pages suivantes :

[https://www.w3schools.com/html/html5\\_video.asp](https://www.w3schools.com/html/html5_video.asp)

<https://developer.mozilla.org/fr/docs/Web/HTML/Element/video>

<http://thenewcode.com/820/Make-HTML5-Video-Adaptive-With-Inline-Media-Queries>

[https://developer.mozilla.org/fr/docs/Web/HTML/Formats pour audio video](https://developer.mozilla.org/fr/docs/Web/HTML/Formats_pour_audio_video)

### 2.3.8.2 Audio

La balise « audio » fonctionne de la façon suivante :

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
  Votre navigateur ne supporte pas la balise audio.
</audio>
```

Comme pour la balise « video », on peut remplacer l'attribut « controls » par l'attribut « autoplay », éventuellement complété de l'attribut « loop » :

```
<audio autoplay loop>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
  Votre navigateur ne supporte pas la balise audio.
</audio>
```

Explication : tous les navigateurs ne supportent pas tous les formats audios. Le navigateur va donc rechercher dans la liste des sources le premier format qu'il sait gérer, et il va charger le fichier audio correspondant.

Pour de plus amples précisions sur le fonctionnement de la balise « audio », je recommande la lecture des pages suivantes :

[https://www.w3schools.com/html/html5\\_audio.asp](https://www.w3schools.com/html/html5_audio.asp)

<https://developer.mozilla.org/fr/docs/Web/HTML/Element/audio>

[https://developer.mozilla.org/fr/docs/Web/HTML/Formats\\_pour\\_audio\\_video](https://developer.mozilla.org/fr/docs/Web/HTML/Formats_pour_audio_video)

Attention, il ne faut pas confondre la balise « audio » avec l'API Webaudio. Cette dernière se programme en Javascript et offre un panel de possibilités beaucoup plus large que la balise « audio ».

Pour faire des expérimentations autour du son, on peut télécharger des fichiers sons soumis à la licence Creative Commons, sur le site suivant :

<https://freesound.org/>

### 2.3.9 Span

Moins sexy que les balises précédentes, mais néanmoins très utile, la balise « span » est une balise destinée à mettre en relief certains mots ou expressions. On l'utilise généralement au sein d'un paragraphe ou d'une « div ».

Exemple :

```
<p>Mon chat a les yeux <span style="color:greenyellow">verts clairs</span> et  
le pelage <span style="color:rosybrown">marron</span>.</p>
```

Je vous invite à tester cette ligne dans Codepen ou CodeCircle.

Je rappelle qu'il n'est pas recommandé d'insérer du CSS à l'intérieur du HTML comme je l'ai fait ici. On peut néanmoins se le permettre quand on est comme ici dans un contexte pédagogique, ou que l'on est en train de créer un prototype pour tester un concept (ce que les anglophones désignent par le terme de « POC », pour « Proof Of Concept »).

La balise « span » est utilisée dans la pratique pour de nombreux usages, les développeurs Javascript s'en servent assez souvent, pour faire « ressortir » dynamiquement certaines informations. Mais cela sort du cadre de cette introduction au HTML, donc je n'en dis pas plus pour l'instant.

### 2.3.10 Autres balises

Nous nous sommes concentrés sur les balises les plus courantes, hormis celles relatives à la gestion des formulaires, que nous étudierons dans un chapitre ultérieur.

On peut néanmoins signaler la balise « hr » qui permet de tracer des lignes de séparation, et qui est très souvent utilisée pour créer une séparation nette entre le corps et le pied de la page HTML.

Exemple :

```
<h1>HTML</h1>  
<p>HTML est le langage de description de page.....</p>
```

```
<hr>
```

```
<h1>CSS</h1>  
<p>CSS définit la manière d'afficher les différents éléments d'une page  
HTML.....</p>
```

Pour de plus amples précisions sur la balise « hr » :

[https://www.w3schools.com/tags/tag\\_hr.asp](https://www.w3schools.com/tags/tag_hr.asp)

Signalons également l'existence des balises « h1 », « h2 », ..., à « h6 » qui permettent de définir des titres et sous-titres. La balise « h1 » désigne le titre de plus haut niveau, les balises « h2 » et suivantes désignant des balises de niveau inférieur. Les navigateurs appliquent un style par défaut à ces balises, avec la taille la plus grande pour la balise « h1 », et des tailles décroissantes pour les balises suivantes.

Exemple à tester sur Codepen ou CodeCircle :

```
<h1>Titre de niveau 1 (titre principal)</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>
```

Pour de plus amples précisions sur les balises « h1 » à « h6 » :

[https://www.w3schools.com/tags/tag\\_hn.asp](https://www.w3schools.com/tags/tag_hn.asp)

Il existe un certain nombre d'autres balises moins fréquemment utilisées, dont certaines sont carrément devenues obsolètes avec l'arrivée du HTML5. Le tableau récapitulatif proposé par W3Schools signale ces balises obsolètes :

<https://www.w3schools.com/tags/default.asp>

### 2.3.11 Les commentaires

On peut placer des commentaires dans le code HTML, au moyen des balises suivantes :

- début d'un commentaire : `<!--`
- fin du commentaire : `-->`

Exemple : le premier bloc de balises « h1 » à « h6 » est actif, le second bloc est en commentaire :

```
<h1>Titre de niveau 1 (titre principal)</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>
```

```
<!-- Début du commentaire  
<h1>Titre de niveau 1 (titre principal)</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>  
Fin du commentaire -->
```

Ces balises de mise en commentaire sont utilisées aussi en tant que balises conditionnelles, comme on le verra dans le chapitre qui s'intitule « exception IE ».



### 2.3.12 iFrame

La balise « iframe » permet d'afficher une page à l'intérieur d'une autre page.

La définition de cette balise donnée par MDN est intéressante :

*L'élément HTML <iframe> représente un contenu de navigation imbriqué qui permet en fait d'obtenir une page HTML intégrée dans la page courante. En HTML 4.01, un document peut contenir un élément <head> et un élément <body> ou un élément <head> et un élément <frameset> mais pas à la fois un élément <body> et un élément <frameset>. En revanche une <iframe> peut être utilisée à l'intérieur du corps d'un document normal. Le contexte de navigation qui contient le contenu intégré est appelé « contexte de navigation parent ». Le contexte de navigation le plus élevé (qui n'a pas de contexte parent) correspond généralement à la fenêtre du navigateur.*

Source : <https://developer.mozilla.org/fr/docs/Web/HTML/Element/iframe>

On peut ajouter à cette définition le fait que « iframe » signifie « inline frame », que l'on pourrait traduire par « cadre en ligne ».

Concrètement, une balise iframe s'écrit de la façon suivante :

```
<iframe src="http://...mon URL.../index.html"
height="317px" width="496px"></iframe>
```

Cette technique est souvent utilisée pour encapsuler dans une page un document multimédia (vidéo ou audio) provenant d'un autre site. Elle est très pratique, mais elle pose de nombreuses contraintes, et en particulier la contrainte de l'origine du site parent par rapport au site encapsulé dans l'iframe : si les deux pages n'appartiennent pas au même domaine (à la même origine), les interactions au moyen de Javascript, entre la page parente, et la page en iframe, seront impossibles.

On notera également que le paramétrage des attributs « height » et « width » n'est pas évident à réaliser, surtout dans le cadre d'affichage responsive (car il faudrait pouvoir communiquer des informations à la page en iframe, mais ce n'est possible que dans le cas d'une même origine). Or l'iframe est souvent utilisée pour encapsuler des pages « étrangères ».

Pour un bon tuto, à la fois sur la balise « iframe » et sur la manière de faire communiquer une iframe avec la page parente :

<https://www.xul.fr/html5/iframe.php>

<https://www.xul.fr/html5/iframe-communication.php>

Liens complémentaires :

[https://www.w3schools.com/tags/tag\\_iframe.asp](https://www.w3schools.com/tags/tag_iframe.asp)

<https://developer.mozilla.org/fr/docs/Web/HTML/Element/iframe>

NB : On trouvera un exemple d'utilisation de la balise « iframe » dans le cours d'introduction à CSS3.

## 2.5 Anatomie d'une page HTML5

### 2.5.1 Principes généraux

Nous avons étudié et testé différentes balises au moyen de Codepen et de CodeCircle.

Nous avons vu que Codepen nous propose 3 zones de saisie différentes : HTML, CSS et Javascript. Codepen s'appuie sur les informations que nous saisissons dans ces 3 zones, pour construire pour nous la page web correspondante. C'est très pratique pour créer de petits prototypes de pages et tester tout un tas de techniques.

Les concepteurs de CodeCircle ont opté pour une approche légèrement différente : le développeur saisit son code dans une seule zone, à charge pour lui de préciser ce qui est du HTML, du CSS et du Javascript. Si le développeur ne précise rien, le navigateur considère qu'il s'agit de code HTML par défaut. Le code Javascript et le code CSS doivent être identifiés précisément au moyen des balises « script » et « style ».

Pour comprendre comment fonctionnent Codepen et CodeCircle dans les coulisses, nous devons prendre un peu de hauteur et regarder comment se construit une page web.

Voici un squelette de page HTML :

```
<!DOCTYPE html> Déclaration du DOCTYPE simplifiée en HTML5
<html lang="fr">
<head>
  <meta charset="UTF-8"> charset déclaré avant l'élément "title"
  <title>mon titre</title>
  <script src="foo.js"></script> Plus besoin de préciser le type de source
  <link rel="stylesheet" href="foo.css"> Plus besoin de préciser le type de source
</head>
<body>
  <div>
    Ajouter du contenu ici
  </div>
</body>
</html>
```

La première ligne contient un DOCTYPE, c'est une information essentielle pour le navigateur, qui est ainsi informé du fait que le code qui suit est du HTML5. Il va donc adapter son fonctionnement aux caractéristiques de cette norme.

Les anciennes normes HTML4 et XHTML utilisaient des DOCTYPE particuliers, compliqués et verbeux, qu'il est intéressant de savoir reconnaître, même si on ne les utilise plus en HTML5.

Exemple de l'un des quelques DOCTYPE utilisables avec la norme HTML4 :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
```

Exemple de l'un des quelques DOCTYPE utilisables avec la norme XHTML :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

On voit donc qu'avec HTML5, c'est vraiment plus simple :

```
<!DOCTYPE html>
```

A la suite du DOCTYPE, nous avons une balise « html » qui englobe l'ensemble de la page, et à l'intérieur de cette balise, nous trouvons deux grandes parties, la partie « head » et la partie « body » :

```
<html lang="fr">
<head>
...
</head>
<body>
...
</body>
</html>
```

La partie « body », c'est la partie que voit l'internaute, sous la forme d'une page web bien structurée. C'est donc là que nous plaçons le code HTML relatif aux listes, aux liens, et à toutes les balises HTML que nous avons étudiées jusqu'ici. C'est également là que nous placerons le code des formulaires, partie que nous étudierons bientôt.

La partie « head » contient un certain nombre d'informations de paramétrage qui sont importantes pour le bon fonctionnement du navigateur :

- nous y trouvons une balise « meta » associée à l'attribut « charset », qui définit l'encodage de la page. C'est généralement de l'UTF-8, sauf cas particulier.
- Nous trouvons une balise « title » qui permet de définir le titre de la page, titre que nous retrouverons dans l'onglet correspondant du navigateur
- Nous pouvons y trouver aussi des balises « script » (une ou plusieurs) définissant du code Javascript
- Nous pouvons y trouver enfin des balises « style » (une ou plusieurs) définissant le code CSS de la page concernée

Exemple :

```
<head>
  <meta charset="UTF-8">
  <title>mon titre</title>
  <script src="foo.js"></script>
  <link rel="stylesheet" href="foo.css">
</head>
```

## 2.5.2 L'exception IE

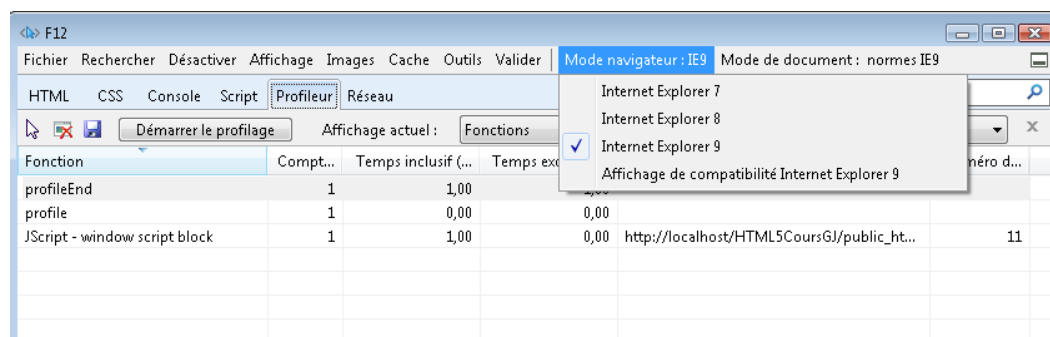
Microsoft a annoncé officiellement l'arrêt du développement du navigateur Internet Explorer (IE), lors de la sortie de Microsoft Edge, en 2015.

Entre 2001, année de sortie de IE 6, et 2015, année de l'arrêt officiel du projet, IE est passé par différentes versions : 7, 8, 9, 10, 11.

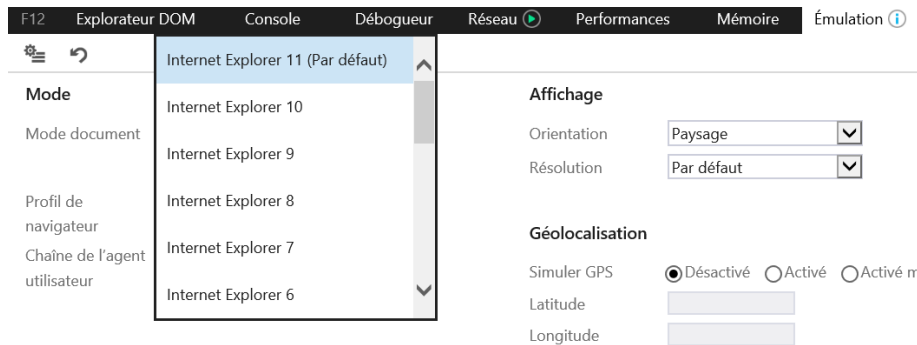
Bien que relativement mûre, et respectant un bon nombre des aspects de la norme HTML5, IE souffrait d'une mauvaise image, et sans aucun doute de fortes limitations techniques. Cette mauvaise image était due essentiellement aux carences des versions 6, 7 et 8, notamment en matière de CSS, mais aussi en matière de support du langage Javascript.

Edge est-il un simple réhabillage de IE 10 ou une véritable réécriture ? Peu importe, le problème qui se pose, c'est que certaines versions anciennes de IE sont encore utilisées dans certaines entreprises, et que le développement d'un code CSS et JS capable de fonctionner sur ces vieux navigateurs constitue un véritable tour de force.

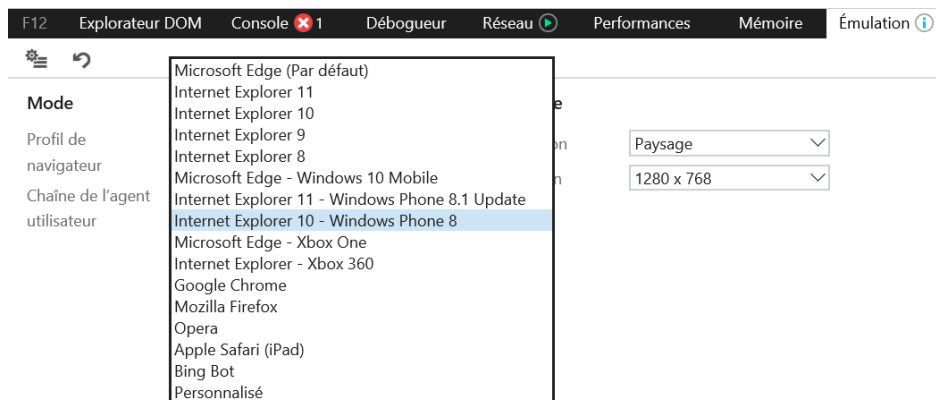
On notera que – à partir de IE 9 – on dispose d'une fonction d'émulation permettant de vérifier le rendu et le fonctionnement d'une page sur différentes versions de IE :



Microsoft Edge et IE 11 proposent à peu près les mêmes outils de développement, mais IE11 continue à proposer le support de l'émulation de IE6 et IE7 :



Edge lui propose un panel d'émulateurs assez large, mais excluant IE 6 et 7 :



Etant donné que de nombreuses incompatibilités existent entre les différentes versions de IE, et par rapport aux autres familles de navigateurs, comment faire pour gérer les exceptions et contourner ces incompatibilités ? Certaines solutions passent par l'utilisation de Javascript et de bibliothèques de polyfills (qui sont des sortes de rustines, ou de béquilles). Nous ne les aborderons pas dans cette introduction au HTML.

Mais une solution existe aussi au niveau du code HTML, avec la notion de commentaire conditionnel, qui nous permet d'exécuter certaines portions de code HTML uniquement si on est sur IE, ou sur une version particulière de IE.

Exemple :

```
<!--[if IE]>
ce code HTML ne s'exécutera que dans IE, et sera ignoré par les autres
navigateurs qui considèrent ça comme du commentaire
<![endif]-->
```

On voit qu'il y a une petite différence par rapport au commentaire HTML standard, avec la présence de crochets, du mot clé « if » et de la constante « IE » qui désigne toutes les versions de Internet Explorer.

Les webdesigners se servent souvent de cette technique pour charger un code CSS spécifique à IE, en complément du code CSS standard. On peut donc trouver dans l'entête d'une page HTML du code tel que celui-ci :

```
<link rel="stylesheet" type="text/css" href="style.css" />
<!--[if IE]>
<link rel="stylesheet" type="text/css" href="ie.css" />
<![endif]-->
```

On peut aussi cibler une version de IE en particulier, en utilisant dans la condition des mots clés tels que :

- lt : lower than. Strictement inférieure à ( < )
- lte : lower than equal. Inférieure ou égale à ( <= )
- gt : greater than. Strictement supérieure à ( > )
- gte : greater than equal. Supérieure ou égale à ( >= )

Si on ne précise pas de condition mais que l'on indique un numéro de version, cela équivaut à écrire « égal ». Voici un exemple :

```
<!--[if lte IE 7]> <html class="ie67 ie678" lang="fr"> <![endif]-->
<!--[if IE 8]> <html class="ie8 ie678" lang="fr"> <![endif]-->
<!--[if gt IE 8]><!--> <html lang="fr"> <!--<![endif]-->
```

Dans cet exemple, nous affectons à la balise « head » :

- un jeu de classes « CSS » particulières pour IE 6 et 7,
- un autre jeu de classes spécifiques uniquement pour IE 8,
- aucune classe spécifique pour les versions de IE supérieures à la 8

On voit dans cet exemple qu'il est possible de gérer de manière conditionnelle tout type de balise, y compris la balise « head ».

Il est également possible de définir une condition négative avec le symbole « ! » :

```
<!--[if !IE]> <-->
```

Ce texte s'affichera sur tous les navigateurs autres qu'IE.

```
<!--> <![endif]-->
```

Pour un tour d'horizon complet sur cette problématique de compatibilité spécifique à IE :

<https://openclassrooms.com/courses/les-hacks-css-pour-internet-explorer>

<https://www.lesintegristes.net/2008/04/08/cibler-internet-explorer-dans-une-css-oui-et-sans-hack/>

<https://www.alsacreations.com/astuce/lire/988-classes-conditionnelles-HTML.html>



## 2.6 Formulaire

### 2.6.1 Introduction

Un formulaire, c'est le regroupement d'un certain nombre de balises dédiées à la saisie d'informations.

Dans l'exemple ci-dessous, nous avons à gauche un formulaire de saisie, tel que vous pourriez en rencontrer en naviguant sur internet, et dans la partie de droite nous voyons une petite partie du code source ayant servi à générer ce formulaire :

Formulaire de test

Champ input1

Champ input2

Champ input3

Champ input4

☒ ☐ Cases à cocher

☒ ☐ ☐ Boutons radios

Champ liste1

Champ liste2

Champ areatxt 

Champ areatxt sur 4 lignes et 20 colonnes

#### HTML

```

1 <fieldset>
2   <legend>Formulaire de test</legend>
3   <form name="testform" method="POST" >
4     <label for="input1id">Champ input1</label>
5     <input type="text" id="input1id" name="input1" v
6     <br><br>
7     <label for="input2id">Champ input2</label>
8     <input type="text" id="input2id" name="input2" v
9     <br><br>
10    <label for="input3id">Champ input3</label>
11    <input type="text" id="input3id" name="input3" v
12          placeholder="champ input3 required" requi
13    <br><br>
14    <label for="input4id">Champ input4</label>
15    <input type="text" id="input4id" name="input4" v
16          placeholder="champ input4 avec 'plac
17    <br><br>
18    <label>
19      <input type="checkbox" name="checkboxbtn[]"
20            value="1" checked="checked" />

```

#### CSS

```

1

```

#### JS

```

1

```

Dans l'exemple de formulaire ci-dessus, nous avons des champs de saisie, des cases à cocher, des boutons radios, des listes déroulantes, et un champ de saisie « multi-lignes ». Nous avons également 3 boutons dont nous reparlerons. Tous ces éléments existent depuis longtemps dans la norme HTML, ils sont très supportés par l'ensemble des navigateurs. D'autres éléments sont venus se rajouter à cette liste, avec l'arrivée de la norme HTML5. Nous les étudierons aussi.

Au travers de différents exemples, nous allons étudier point par point comment construire des formulaires tels que celui ci-dessus.

Un formulaire est toujours encadré par les balises « form » et « /form » :

```
<form>
</form>
```

La balise « form » accepte plusieurs attributs qui sont les suivants : « name », « method » et « action »

```
<form name="testform" method="POST" action="api/toto.php">
</form>
```

A quoi servent les attributs définis dans l'exemple ci-dessus ?

- L'attribut « name » est facultatif, il peut être omis. On préfère quelquefois le remplacer par un attribut « id », plus pratique pour certaines manipulations réalisées en Javascript.
- L'attribut « method » peut prendre les valeurs « POST » ou « GET ». Cela définit le type de requête HTTP que le navigateur enverra au serveur, quand l'internaute cliquera sur le bouton de validation du formulaire.
- L'attribut « action » définit le nom du script qui sera appelé sur le serveur, lors de la soumission du formulaire. Dans notre exemple, il s'agit du script « toto.php » qui se trouve dans le répertoire « api » de notre application.

Nous verrons la signification de ces attributs plus tard, concentrons-nous sur les balises de saisie. Je vous invite à les tester au fur et à mesure que nous les découvrons, dans Codepen ou CodeCircle, selon votre préférence.

La balise « input » est l'une des plus importantes :

```
<form>
  <input type="text" name="champ1" value="" />
</form>
```

L'attribut « type » définit le type du champ de saisie. Si on l'oublie, ce n'est pas très grave, le navigateur considère dans ce cas que le champ de saisie est un type « texte » par défaut.

L'attribut « name » est important car c'est une information qui sera transmise au serveur, dans le cas où notre formulaire serait conçu pour envoyer les informations saisies vers un serveur. Ce point sera abordé plus en détail dans le cadre des cours PHP et/ou Javascript, aussi nous ne nous attarderons pas sur le sujet pour le moment.

L'attribut « value » définit quelle est la valeur par défaut du champ de saisie, lors de son premier affichage dans la page. On peut définir une autre valeur que blanc, je vous invite à tester d'autres valeurs pour le vérifier.

Le « slash » (/) placé à la fin de la balise est un caractère de fermeture de la balise. Cela équivaut à écrire une balise « /input », mais on n'écrit jamais cette balise, on préfère placer ce « slash » qui a le même effet pour le navigateur. En réalité ce « slash » est un résidu des anciennes normes HTML4 et XHTML, car en HTML5, on n'est plus obligé de l'écrire, les navigateurs considérant que la balise « input » est une balise « auto-fermante ». Dans le souci de garantir une bonne compatibilité avec d'anciens navigateurs, on recommande quand même de continuer à placer ce « slash » de fermeture.

Si vous trouvez que cela fait beaucoup d'informations pour un simple champ de saisie, dites-vous que ce n'est pas fini 😊. Mais vous avez tout à fait le droit de faire une pause 😊.

Voici deux attributs qui sont fréquemment utilisés sur les balises « input » : « id » et « placeholder ».

```
<form>

  <input type="text" id="champ1-id" name="champ1" value=""
        placeholder="saisissez ce que vous voulez" />

</form>
```

L'attribut « placeholder » est une nouveauté du HTML5. Cet attribut a pour effet d'afficher – à l'intérieur même du champ de saisie - un petit texte destiné à guider l'utilisateur dans sa saisie. Dès que l'utilisateur commence à saisir quelque chose, ce texte disparaît, et réapparaît uniquement dans le cas où l'utilisateur « vide » le contenu du champ de saisie. Je vous invite à tester cela dans Codepen ou CodeCircle.

L'attribut « id » est facultatif, mais quand il est défini, il permet de faciliter certaines interactions utilisateurs, notamment via du code Javascript. Nous verrons un tout petit exemple d'utilisation dans le cadre de ce cours, ce sujet sera approfondi ultérieurement dans le cadre du cours Javascript.

Il y a quand même un cas où l'attribut « id » est intéressant d'un point de vue HTML, c'est dans son association avec la balise « label ». Voici tout de suite un exemple :

```
<form>

  <label for="champ1-id">Champ de saisie 1</label>
  <input type="text" id="champ1-id" name="champ1" value=""
        placeholder="saisissez ce que vous voulez" />

</form>
```

Dans le navigateur, cela va ressembler à ceci :

Champ de saisie 1

La balise « label » a un attribut « for » dont la valeur est identique à la valeur de l'attribut « id » de la balise « input ». Cela signifie que les 2 balises sont liées. Cliquez une fois sur le libellé, vous allez constater que le curseur se positionne automatiquement sur le champ de saisie qui lui est associé. Cet effet est très pratique pour les internautes. Cela contribue à améliorer l'accessibilité du formulaire, notamment pour les personnes à mobilité réduite, ou ayant des déficiences visuelles, et qui utilisent des navigateurs spécialement adaptés pour elles.

Ce lien entre l'attribut « for » de la balise « label » et l'attribut « id » de la balise « input » n'est pas toujours bien compris des développeurs. On trouve quelquefois des développeurs qui pensent – à tort – que l'attribut « for » est lié à l'attribut « name » du champ de saisie, c'est une erreur.

On rencontre quelquefois une autre manière d'utiliser l'attribut « label », qui est tout à fait valable, et dont voici un exemple :

```
<label>Champ de saisie 1  
  <input type="text" id="champ1-id" name="champ1" value=""  
    placeholder="saisissez ce que vous voulez" />  
</label>
```

On voit que la balise « input » est imbriquée dans la balise « label », les deux balises se trouvent donc interdépendantes. L'attribut « for » au niveau de la balise « label » devient superflu, on peut donc le supprimer. L'accessibilité n'est pas remise en cause. Le résultat est le même d'un point de vue visuel.

Nous avons découvert dans ce chapitre les balises « form », « label » et « input ». Nous allons approfondir la balise « input », dans le chapitre suivant, et découvrir d'autres balises tout aussi intéressantes.

Nous avons

## 2.6.2 Les champs de saisie « historiques »

### Balise « input »

La balise « input » que nous avons étudiée au chapitre précédent accepte différents types :

- `<input type="text">` pour les champs texte simples
- `<input type="password">` pour la saisie des mots de passes
- `<input type="radio">` pour les boutons radio
- `<input type="checkbox">` pour les cases à cocher
- `<input type="hidden">` pour la gestion de champs cachés
- `<input type="file">` pour le chargement de fichier (fonction « upload »)
- `<input type="button">` pour les boutons simples (gérés via Javascript)
- `<input type="submit">` bouton de soumission du formulaire (déclenche l'envoi des données du formulaire vers un serveur)
- `<input type="reset">` bouton permettant de réinitialiser les champs du formulaire (à leur valeur initiale)

Tous ces types de champ sont bien supportés par les navigateurs. Il est vrai qu'ils existent depuis longtemps. Prenez le temps de les tester dans Codepen ou CodeCircle.

Les champs de type « file » ne seront pas étudiés dans ce support. Ils impliquent la mise en place d'interactions fortes avec un serveur (et un langage comme PHP par exemple), puisqu'ils sont utilisés pour envoyer des fichiers de différents formats vers le dit serveur. On sort du coup très vite de l'initiation HTML, et donc de l'objectif de ce support.

Les champs de type « hidden » sont pratiques pour insérer des données confidentielles, que l'utilisateur ne doit pas voir, mais que l'on a besoin de transmettre au serveur en même temps que les données saisies par l'utilisateur. Il peut s'agir d'un identifiant permettant d'authentifier l'utilisateur, ou d'un code produit dans le cas d'une application de prise de commande.

Les champs de type « password » permettent de saisir un mot de passe, sans qu'il apparaisse clairement sur la page (chaque caractère est généralement remplacé par un rond noir). Attention, ce n'est pas une garantie de sécurité, comme on le verra en cours PHP ou Javascript.

Le champ de type « submit » est sans aucun doute l'élément le plus important, puisque c'est lui qui – une fois cliqué – va déclencher l'envoi des données du formulaire vers le serveur dont l'URL est définie dans l'entête du formulaire (cf. balise « form » attribut « action »). Si vous oubliez ce bouton de soumission, le formulaire sera incapable d'envoyer des données où que ce soit :

```
<form name="testform" method="POST" action="api/toto.php">
```

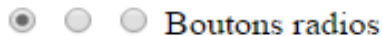
```
<input type="text" name="champ1" value="" />
<input type="submit" value="valider" name="valid" />
</form>
```

Nous allons étudier les champs « input » de type « checkbox » et « radio » au travers de quelques exemples.

### Boutons radios

Exemple de boutons radio :

```
<label>
  <input type="radio" name="radio1" value="1" checked />
  <input type="radio" name="radio1" value="2" />
  <input type="radio" name="radio1" value="3" />
  Boutons radios
</label>
```

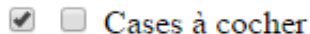


Le bouton radio a une caractéristique essentielle : une seule valeur peut être sélectionnée. C'est la raison pour laquelle tous les boutons radios de notre exemple ont le même attribut « name ». L'attribut « checked » détermine quelle est le bouton radio qui est sélectionné par défaut.

### Checkbox

Exemple de cases à cocher :

```
<label>
  <input type="checkbox" name="checkboxbtn[]" value="1" checked="checked" />
  <input type="checkbox" name="checkboxbtn[]" value="2" />
  Cases à cocher
</label>
```



Dans le cas des cases à cocher, l'utilisateur a la possibilité de cocher plusieurs cases pour un même choix. C'est la raison pour laquelle les cases de notre exemple ont toutes le même attribut « name » dans lequel une paire de crochets est définie (« checkbox[] »). Ceci est dû au

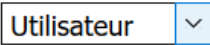
fait que le formulaire va envoyer au serveur non pas une seule valeur, mais un tableau de valeurs (les crochets symbolisent l'usage d'un tableau).

### Select

Il existe d'autres types de champ de saisie, et notamment les listes déroulantes :

Exemple de liste déroulante :

```
<select name="test">
  <option value="0">Développeur</option>
  <option value="1" selected>Utilisateur</option>
</select>
```

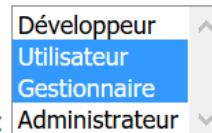
Exemple de liste déroulante : 

L'attribut « selected » est facultatif. Quand il est défini, il détermine quelle est la valeur sélectionnée par défaut.

Exemple de liste déroulante avec possibilité de sélections multiples :

```
<select name="test[]" size="4" multiple>
  <option value="0">Développeur</option>
  <option value="1" selected>Utilisateur</option>
  <option value="2" selected>Gestionnaire</option>
  <option value="4">Administrateur</option>
</select>
```

Exemple de liste déroulante avec sélections multiples :



C'est l'attribut « multiple », défini sur la balise « select » qui permet à l'utilisateur de sélectionner plusieurs valeurs. La présence de l'attribut « size » permet de définir combien d'éléments de la liste d'options on souhaite afficher en même temps (dans notre exemple c'est « 4 »).

Vous remarquerez la présence de l'attribut « selected » sur plusieurs options.

Comme pour les cases à cocher, on utilise des crochets dans l'attribut « name » : puisque l'utilisateur a la possibilité de sélectionner plusieurs valeurs, il faut que le formulaire renvoie l'information sous la forme d'un tableau et c'est la présence de ces crochets qui permet au navigateur de « savoir » qu'il doit gérer un tableau sur ce champ de saisie.





## Textarea

La balise « textarea » permet de définir un champ de saisie composé de plusieurs lignes et colonnes.

Exemple :

```
<textarea name="texte" cols="30" rows="5">
Le texte affiché dans la boîte de texte...
</textarea>
```

Prenez le temps de tester les différents types de champs que nous avons étudié jusqu'ici.

Le site W3Schools propose plusieurs petits exercices que je vous invite à faire. Ils sont relativement simples et plutôt amusants :

[https://www.w3schools.com/html/html\\_forms.asp](https://www.w3schools.com/html/html_forms.asp)

```
<form name="testform" method="POST" action="toto.php">
  <label for="nomid">Nom</label>
  <input type="text" id="nomid" name="nom" value=""
    placeholder="votre nom SVP" />
  <br><br>
  <label for="prenomid">Prénom</label>
  <input type="text" id="prenomid" name="prenom" value=""
    placeholder="votre prénom SVP" />
  <br><br>
  <input type="submit" value="valider" name="valid" />
  <input type="reset" value="Reinit" />
  <input type="button" value="bouton_for_JS"
    onclick="alert('Bonjour ' +
      document.getElementById('nomid').value + ' ' +
      document.getElementById('prenomid').value );" />
</form>
```



### 2.6.3 Attributs « historiques » importants

Ce chapitre présente quelques attributs importants associés aux balises « input ». Ces attributs existent depuis longtemps et sont bien supportés par les navigateurs (hormis « accesskey » qui présente quelques défauts comme le verra). On notera que l'attribut « tabindex » a fait l'objet d'une évolution en HTML5.

#### *Attributs size et maxlength*

Associé aux balises « input » de type « text » et aux balises « textarea », l'attribut « maxlength » permet de fixer la longueur maximale du texte pouvant être saisi par l'utilisateur. Si on ne précise rien, la longueur maximale par défaut est de 524288 caractères. Il ne faut pas confondre cet attribut « maxlength » avec l'attribut « size », qui lui définit le nombre de caractères apparaissant sur la page, indépendamment de la longueur maximale supportée par le champ de saisie :

```
<input type="text" size="40" maxlength="120" />
```

#### *Attribut disabled*

L'attribut « disabled » est compatible avec les balises « input », « textarea », « select » et « button ».

Avec cet attribut, le champ n'est pas modifiable et sa valeur n'est pas renvoyée vers le serveur lors de la soumission du formulaire

```
<input type="text" size="40" disabled="disabled" />
```

Avec le HTML5, la valeur de l'attribut « disabled » devient facultative, et on a le droit d'écrire ceci :

```
<input type="text" size="40" disabled />
```

#### *Attribut readonly*

L'attribut « readonly » est compatible avec les balises de formulaires « input » et « textarea » : Le champ n'est pas modifiable mais la valeur du champ est bien renvoyée vers le serveur lors de la soumission du formulaire

```
<input type="text" size="40" readonly="readonly" />
```

Avec le HTML5, la valeur de l'attribut « readonly » devient facultative, et on a le droit d'écrire ceci :

```
<input type="text" size="40" readonly />
```

#### *Attribut tabindex*

L'attribut « tabindex » est compatible avec les balises de formulaire « input », « textarea », « select » et « button »

Il permet de définir l'ordre dans lequel le curseur « enchaîne » les champs de saisie au moyen de la touche de tabulation. Dans les pages répondant à la norme HTML5, cet attribut peut être placé sur n'importe quel élément d'une page.

```
<input type="text" size="20" tabindex="2" />
```

**ATTENTION** : l'utilisation de l'attribut « tabindex » a changé en HTML5. En HTML 4.01, l'attribut « tabindex » ne pouvait être utilisé que sur les éléments recevant naturellement le focus (« a », « button », « input », « object », « select », « textarea »). Depuis HTML5, l'attribut « tabindex » peut être utilisé sur tous les éléments d'une page.

La valeur de cet attribut est un nombre entier, il peut prendre les valeurs suivantes :

- une valeur négative (par exemple -1) : l'élément ne peut être atteint via la navigation au clavier ;
- zéro (0) : l'élément peut être atteint via la navigation au clavier ;
- une valeur positive (1, 2, ..) : l'élément peut être atteint via la navigation au clavier et les éléments seront parcourus dans l'ordre croissant des valeurs de l'attribut. Si plusieurs éléments partagent la même valeur d'attribut, l'ordre de tabulation correspondra à l'ordre des éléments dans le document.

#### *Attribut accesskey*

cet attribut est compatible avec les balises de formulaires « input », « textarea », « select », « label », « legend » et « button ».

Il permet d'associer une touche du clavier à la touche [Alt], pour rediriger le curseur directement sur un champ prédéfini.

```
<input type="text" size="40" value="champ" accesskey="c" />
```

Attention, l'utilisation de l'attribut « accesskey » est problématique, car son fonctionnement n'est pas homogène au niveau des navigateurs. Pour en savoir plus sur ce sujet :

[http://openweb.eu.org/articles/accesskey\\_essai\\_non\\_transforme/](http://openweb.eu.org/articles/accesskey_essai_non_transforme/)

<http://www.alsacreations.com/article/lire/568-Accesskey-le-grand-echec-de-l-accessibilite-du-Web.html>

## 2.6.4 Nouveaux types « input » du HTML5

La norme HTML5 fournit plusieurs nouveaux types de champs de formulaires.

Ces nouveaux types ne sont pas adoptés de manière homogène par tous les navigateurs, mais ce n'est pas dramatique car les navigateurs ont pour particularité de traiter comme des champs de type "text" ceux des nouveaux types qu'ils ne reconnaissent pas encore.

Parmi les nouveaux types de champ, on trouve :

- `<input type="search">` pour la saisie dans une "boîte" de recherche (équivalent à un type « text » mais avec un rendu légèrement différent)
- `<input type="email">` pour la saisie d'email (contrôle à la saisie de la présence de l'arobase @)
- `<input type="url">` pour la saisie d'URL
- `<input type="tel">` pour la saisie de numéro de téléphone
- `<input type="number">` pour la saisie de nombre
- `<input type="range">` pour la saisie de donnée au moyen d'un curseur ("slider")
- `<input type="date">` pour la saisie de date
- `<input type="month">` pour la saisie de mois
- `<input type="week">` pour la saisie de semaines
- `<input type="time">` pour la saisie d'heure/minutes/secondes
- `<input type="datetime">` pour la saisie de timestamps
- `<input type="datetime-local">` saisie de timestamps en temps absolu (UTC)
- `<input type="color">` pour des effets de type "colorpicker"

Certains navigateurs appliquent un style par défaut à ces nouveaux types de champ.

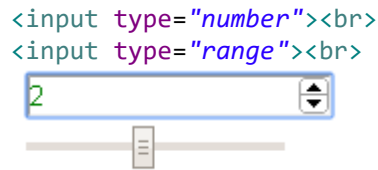
Mais surtout, certains terminaux mobiles personnalisent automatiquement l'affichage de la saisie en fonction du type de champ :

- en proposant un clavier virtuel pour les champs de type « text » ou « search »
- en proposant un clavier virtuel spécialement adapté à la saisie d'adresse email pour le type « email » (avec l'arobase @ mise en évidence)
- en proposant un calendrier pour les données temporelles (« date », « datetime », etc..)
- en proposant un pavé numérique sur les champs de type « number » et « tel ».

Les extraits de copie d'écran présentés page suivante ont été réalisés sur Google Chrome. Je vous invite à tester les différents types sur des navigateurs différents (Firefox, Chrome, Internet Explorer, Edge, Safari, Opéra), pour constater par vous-même que tous les navigateurs ne sont pas à jour par rapport à ces nouveaux types HTML5.

*Types "number" et "range"*

Dans le cas du type « number », on voit apparaître des « spinner » à droite du champs de saisie, permettant d'incrémenter ou décrémenter la valeur du champ de saisie. Le type « range » apparaît sous la forme d'un curseur :



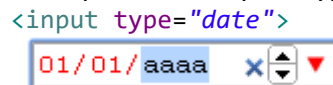
Les champs de type "number" bénéficient d'attributs complémentaires dont les noms sont assez parlants (« min », « max », « step ») :

```
<input type="number" max="100" min="10" step="10" value="50">
```

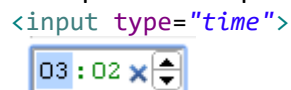
*Types "date", "time", "datetime", "month", "datetime-local"...*

HTML5 apporte de nouveaux champs permettant la saisie de données temporelles.

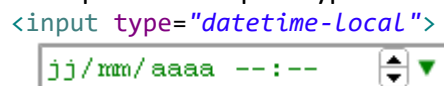
Exemple de champ de type "date" et sa représentation dans Google Chrome :



Exemple de champ de type "time" et sa représentation dans Google Chrome :



Exemple de champ de type "datetime-local" et sa représentation dans Google Chrome :



Exemple de champ de type "month" - combiné avec les attributs « min » et « max » - et sa représentation dans Google Chrome :

```
<input type="month" min="2017-01" max="2018-12" >
```

Calendar view for March 2017. The header shows 'mars 2017' with navigation buttons. The grid shows the following days:

lun.	mar.	mer.	jeu.	ven.	sam.	dim.
27	28	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

Exemple de champ de type "week" - combiné avec les attributs « min » et « max » - et sa représentation dans Google Chrome :

```
<input type="week" min="2017-01" max="2018-12" >
```

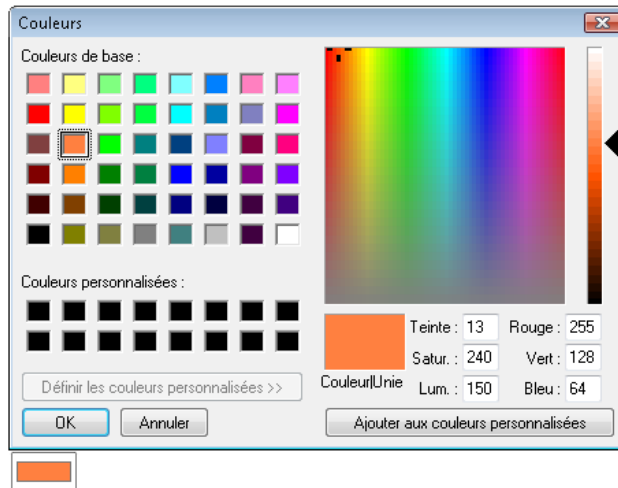
Calendar view for May 2017. The header shows 'Semaine 21, 2017' with navigation buttons. The grid shows the following weeks:

Semaine	lun.	mar.	mer.	jeu.	ven.	sam.	dim.
18	1	2	3	4	5	6	7
19	8	9	10	11	12	13	14
20	15	16	17	18	19	20	21
21	22	23	24	25	26	27	28
22	29	30	31	1	2	3	4

### Type "Color"

Exemple de champ de type "color" et sa représentation dans Google Chrome :

```
<input type="color">
```



Conclusion : le support des nouvelles balises HTML5 n'étant pas homogène d'un navigateur à l'autre, leur usage est souvent problématique, et les développeurs préfèrent souvent utiliser des composants Javascript, qui sont équivalents d'un point de vue fonctionnel, mais qui garantissent de bénéficier d'un comportement homogène sur la plupart des navigateurs. On peut néanmoins utiliser ces balises pour des prototypes d'applications, en gardant à l'esprit qu'il faudra les remplacer par une solution Javascript si le projet dépasse le stade du prototype.

## 2.6.5 Nouveaux attributs HTML5

La norme HTML5 apporte de nouveaux attributs pour les champs de formulaires.

Les nouveaux attributs HTML5 bénéficient d'une bonne prise en charge par les navigateurs tels que Firefox, Chrome, Safari (à vérifier au cas par cas pour les autres).

### Attribut autofocus

L'attribut autofocus peut être utilisé sur tous les types de champ saisie, et il permet - comme son nom l'indique - de placer le focus automatiquement sur le champ sur lequel cet attribut est défini.

Exemple :

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



### *Attribut placeholder*

L'attribut placeholder a été popularisé par certains frameworks et il fait maintenant partie intégrante de la norme HTML5. Il permet d'afficher un texte explicatif dans le champ de saisie, texte qui disparaîtra dès que l'utilisateur commencera à saisir à l'intérieur de ce champ.

Exemple :

```
<input type="email" placeholder="e.g. foo@bar.com">
```

### *Attribut autocomplete*

Par défaut, la plupart des navigateurs stockent dans un cache au minimum la dernière saisie effectuée sur un champ de formulaire.

Ce comportement peut être gênant sur certaines applications, aussi on peut maintenant le bloquer via l'attribut autocomplete.

Exemple :

```
<input type="email" autocomplete="off">
```

L'attribut "autocomplete" peut aussi être placé sur la balise "form", de manière à impacter tous les champs du formulaire.

### *Attribut spellcheck*

Certains navigateurs offrent une fonction de vérification orthographique, essentiellement sur les champs de type textarea.

On peut maintenant forcer les navigateurs à appliquer cette fonctionnalité sur d'autres types de champs, via l'attribut spellcheck.

Exemple :

```
<input type="text" spellcheck="true">
```

Le dictionnaire orthographique utilisé est celui correspondant à la langue par défaut du navigateur, mais on peut modifier ce comportement via l'attribut lang.

Exemple :

```
<input type="text" spellcheck lang="fr">
```

### *Attribut multiple*

On connaissant déjà l'attribut « multiple » avec la balise « select ». En HTML5, l'attribut « multiple » est utilisable avec le champ de saisie de type "file" pour permettre la saisie de plusieurs fichiers. On peut aussi l'utiliser avec le type de champ email, pour la saisie de plusieurs adresses email :

```
<input type="file" multiple>
```

### Attribut form

L'attribut "form" associé à un champ de formulaire permet de lier explicitement un champ de saisie à un formulaire. Il est dès lors possible de placer ce champ de saisie n'importe où dans la page, et surtout en dehors des balises <form> et </form> délimitant le formulaire en question.

```
<form id="foo">...</form>
<input type="text" id="bar" form="foo">
```

### Attribut required

C'est certainement l'un des nouveaux attributs les plus intéressants. Il permet de rendre un champ de sa saisie obligatoire, sans avoir besoin de recourir à du code Javascript. Exemple :

```
<input type="email" required>
```

### Attribut pattern

L'attribut pattern permet d'appliquer une expression régulière (ou regex) pour contrôler la validité d'une saisie.

Par exemple :

- pour forcer le navigateur à n'autoriser que la saisie de valeurs numériques dans un champ, on peut utiliser le pattern ci-dessous :

```
<input type="tel" pattern="\d*">
```

- pour définir une longueur minimale de 3 caractères et maximale de 30 caractères :

```
<input type="text" pattern=".{3,30}">
```

On peut personnaliser le message d'erreur associé au pattern, en ajoutant un second attribut, qui est l'attribut "title" :

```
<input type="tel" pattern="\d*" title="Numbers only">
```

Le site suivant propose de nombreux patterns prêts à l'emploi (pour les numéros de téléphone, les codes postaux, etc...) :

<http://html5pattern.com>

### AVERTISSEMENTS :

Il convient de rappeler que des attributs de validation tels que "required" et "pattern" ne peuvent en aucun cas constituer des outils suffisants pour contrôler et valider la qualité des données saisies par les utilisateurs. Pour des raisons de sécurité, ces contrôles doivent impérativement être "doublés" par des contrôles - au moins équivalents - effectués côté serveur (en PHP, Ruby, Python, Java, Javascript sur NodeJS, etc..).

### Attribut novalidate

On peut désactiver complètement la validation d'un formulaire au moyen de l'attribut novalidate :

```
<form action="foo" novalidate>...</form>
```

On peut aussi appliquer ce principe à des champs de saisie :

```
<button type="submit" formnovalidate>Go</button>
```

### Attribut contenteditable

Ce n'est pas véritablement un attribut destiné aux formulaires. Il est plutôt adapté pour rendre "éditable" des cellules de tableaux, ou des paragraphes à l'intérieur d'une page. Son utilisation nécessite l'utilisation de code Javascript, pour pouvoir prendre en compte les modifications de valeur, c'est pourquoi nous reverrons cet attribut dans le cadre du cours Javascript.

```
<td contenteditable="true">200</td>
```

### Attribut data-\*

Avant l'arrivée de la norme HTML5, quand on avait besoin de stocker une donnée "métier" dans le code HTML, comme par exemple un code produit, on détournait très souvent l'attribut "id" ou l'attribut "class", pour y stocker cette information. Cette utilisation détournée de certains attributs était souvent source d'incompréhensions et de bugs.

Avec HTML5, on est mesure de créer des attributs personnalisés, les attributs « data » : on peut créer autant d'attributs "data" que nécessaire, il faut simplement veiller à ce que leur nom commence par "data-". On peut par exemple écrire ceci :

```
<p data-id="443" data-SSN="XUL443XXU" data-garantie="3" data-prix=2500.50>Produit 443</p>
```

On peut aussi utiliser ces attributs sur des lignes et des cellules de tableau HTML :

```
<tr data-id="443">
  <td data-prix=2500.50>2500 €</td>
</tr>
```

Le fait que ces données métiers soient stockées dans des attributs rend plus aisée leur manipulation via le langage Javascript.

### *Autres attributs*

De nombreux types d'attributs sont apparus ces dernières années dans l'optique d'améliorer la qualité du code HTML, en y intégrant des notions de sémantique, c'est-à-dire la possibilité d'introduire du sens. Ces nouveaux attributs visent différents objectifs :

- amélioration du référencement des pages par Google et les moteurs concurrents
- amélioration de l'accessibilité pour les personnes handicapées utilisant des navigateurs adaptés
- amélioration de la lisibilité et de la maintenabilité du code produit par les équipes de développement (l'attribut data-\* est le grand gagnant dans cette catégorie)

Parmi les nouveaux attributs qui ne seront pas présentés dans ce support d'introduction au HTML5, on trouve :

- les microdonnées (en anglais « micro-data »)
- RDFa
- Les microformats
- WAI-ARIA

Pour les curieux 😊, ces différents sujets sont abordés de manière détaillée dans le support « Javascript et HTML5 » qui est librement téléchargeable sur ce dépôt :

<https://github.com/gregja/JSCorner>

## 2.6.6 Nouvelles balises HTML5

La norme HTML5 propose plusieurs nouvelles balises intéressantes, mais dont le support par les navigateurs est relativement hétérogène. Beaucoup de ces nouvelles balises sont manipulables via le langage Javascript, au travers d'API spécifiques, comme on le verra dans quelques exemples.

Pour une présentation plus détaillée de ces champs (incluant des exemples de manipulation en Javascript), on pourra se reporter au support « Javascript et HTML5 » qui est librement téléchargeable sur ce dépôt Github :

<https://github.com/gregja/JSCorner>

### *balise progress*

Les barres de progression sont couramment utilisées dans les applications, qu'elles soient "web" ou pas.

```
<progress max="20" min="10" value="15">15</progress>
```


Balise Progress 

### *Balise meter*

La balise "meter" se rapproche beaucoup, dans son mode de fonctionnement, de la balise "progress". Elle est plus particulièrement destinée à effectuer des mesures pour faciliter la production de tableaux de bord.

Exemple :

```
<p>
  Balises meter 1
  <meter min="35" max="43" value="37.5">37.5°C</meter>
</p>
<p>
  Balise meter 2
  <meter low="0" high="4" max="5" value="4.5" optimum="2.50">4.5 m</meter>
</p>
```

Balises meter 1 

Balise meter 2 

### *Balise output*

L'élément "output" affiche le résultat d'un calcul, éventuellement en interaction avec des saisies utilisateurs.

### Balise *datalist*

La balise « *datalist* » spécifie une liste d'options prédéfinies pour un élément de formulaire de type `<input>`. Elle fournit donc un mécanisme dit d'autocomplétion. Pour associer une « *datalist* » à un « *input* », on utilise l'attribut « *list* » au niveau de l'« *input* », attribut dans lequel on indique l'ID associé à la « *datalist* », comme dans l'exemple ci-dessous :

```
<label for="url_field">Saisissez ou sélectionnez une URL</label>
<input type="url" id="url_field" name="url_field" list="urls" />
<datalist id="urls">
  <option label="W3C" value="http://www.w3.org/">
  <option label="WHATWG" value="http://www.whatwg.org/">
  <option label="HTML 5 Specification" value="http://www.w3.org/TR/html5/">
  <option label="SVG Specification" value="http://www.w3.org/TR/SVG/">
  <option label="SPARSQL" value="http://fr.wikipedia.org/wiki/SPARQL">
</datalist>
```

Attention, la balise « *datalist* » semble séduisante au premier abord, mais elle présente de nombreux défauts qui sont largement commentés dans le support de cours « Javascript et HTML5 » indiqué au début de ce chapitre. On peut néanmoins utiliser cette balise pour des prototypes d'application, en gardant à l'esprit qu'elle devra être remplacée tôt ou tard par une solution plus « cross-browser » (c'est-à-dire « compatible avec un large panel de navigateurs »).

### Balise « *sémantiques* »

De nouvelles balises sont apparues avec la norme HTML5, comme par exemple :

« *article* », « *aside* », « *nav* », « *section* », « *footer* », « *header* », « *hgroup* »

Les navigateurs ont mis un certain temps à intégrer ces nouvelles balises, dont le succès est assez mitigé, et l'utilité controversée. On trouvera une présentation plus détaillée de ces balises dans le support « Javascript et HTML5 » qui est librement téléchargeable sur ce dépôt :

<https://github.com/gregja/JSCorner>

## 2.6.7 Balises auto-fermantes

Nous avons évoqué le fait que certaines balises étaient auto-fermantes. C'est le cas des balises « input », mais il y en a d'autres. Heureusement, les balises qui sont dans ce cas sont peu nombreuses, en voici la liste :

```
<br>
<embed>
<hr>
<img>
<input>
<link>
<meta>
<param>
<source>
<wbr>
```

La balise « input » n'a pas toujours été auto-fermante, ou plus exactement cela dépendait de la version de HTML utilisée. Si vous avez un doute, ou si vous souhaitez garantir une bonne compatibilité avec certains navigateurs anciens, rien ne vous empêche de placer le « slash » de fermeture en fin de balise, même si certaines balises n'en ont pas besoin. La norme HTML5 est plus souple que les versions antérieures (HTML4 et XHTML), les navigateurs supportant HTML5 se montrent donc très tolérants par rapport à ces règles syntaxiques.

### 2.6.8 Balises et attributs HTML dépréciés (obsolètes)

En établissant la norme HTML5, le W3C a fait le ménage dans les balises, et certaines balises se sont retrouvées dépréciées. La raison principale de cette dépréciation est que certaines balises étaient clairement destinées à positionner des éléments sur la page. Or le W3C a souhaité éliminer toute notion de style du HTML, pour confier ce rôle – sans ambiguïté – au CSS.

Parmi les balises dépréciées les plus connues, on trouve : « font », « basefont », « center », « s », « u », etc...

Ce ménage concerne aussi les propriétés HTML, dont certaines sont elles-aussi dépréciées en HTML5 : c'est le cas pour « align », « width » (associé à certaines balises), « height » (associé à certaines balises), « size », « color », « border », « background », « bgcolor », « border », « face », « target », etc...

Pour une présentation exhaustive des balises et propriétés dépréciées :

<https://www.w3.org/TR/html5-diff/#obsolete-elements>

<https://www.w3.org/TR/html5-diff/#obsolete-attributes>

Tableau de synthèse :

[https://www.tutorialspoint.com/html5/html5\\_deprecated\\_tags.htm](https://www.tutorialspoint.com/html5/html5_deprecated_tags.htm)

Sachant que certains éléments HTML dépréciés servaient à positionner des éléments dans la page, comme la balise « center » ou l'attribut « align », il faut trouver les solutions de substitution permettant de faire la même chose via CSS. Le site Openclassrooms propose un excellent tuto sur le sujet :

<https://openclassrooms.com/courses/centrer-en-css>

(on trouvera ce sujet traité plus en détail dans le support de cours « CSS3 – Premiers pas »).



## 2.7 Structure d'une page HTML dans la « vraie vie »

### 2.7.1 Principes et vocabulaire

La structure type d'une page HTML est généralement constituée de plusieurs éléments, tels que :

- Un bandeau « entête » (le « header »)
- Un menu de navigation horizontal
- Une partie facultative contenant généralement un texte de présentation du site et/ou de la page
- Le corps de la page (souvent désigné par le terme anglais « main » pour « principal »), lui-même constitué de :
  - o Un bandeau vertical gauche (facultatif) pouvant contenir un sous-menu vertical
  - o Un bandeau central plus large présentant les données importantes
  - o Un bandeau vertical droit (facultatif lui aussi, il est plutôt rare de combiner bandeau gauche et droit, question d'équilibre)
- Un bandeau « pied de page » (le « footer »)

En début de projet, la construction de modèle ou « template » de page est le plus souvent un travail conjoint du développeur et du webdesigner. Les deux définissent ensemble les principaux éléments de la page, ainsi qu'une première version de style qui sera enrichie par la suite.

Idéalement, il conviendrait que tout projet démarre avec l'élaboration d'une « bible » (ou « référentiel ») de tous les éléments graphiques pouvant être utilisés par les développeurs tout au long du projet. Mais dans la « vraie vie », et en particulier dans les projets développés en méthode agile, il est rare que toutes les fonctionnalités nécessaires au projet soient connues dès le démarrage, aussi de nombreuses retouches et compléments graphiques se révèlent nécessaires, impliquant l'intervention au fil de l'eau du webdesigner pour livrer les éléments manquants, ou retoucher le code HTML et CSS produit par les développeurs.

On entend souvent les webdesigners utiliser le terme « mock-up ». En voici une définition empruntée à Wikipédia :

*« le terme mock-up (qui vient du même mot anglais qui signifie une maquette à l'échelle 1:1) désigne un prototype d'interface utilisateur. Un mock-up a ainsi pour rôle de présenter les idées sur l'utilisation d'un logiciel. »*

Les webdesigner professionnels utilisent généralement un logiciel graphique tel que Adobe Photoshop pour élaborer un « mock-up ». Si le logiciel est effectivement produit avec Photoshop, le format du fichier résultant sera un PSD (format propriétaire du logiciel Photoshop). C'est soit le webdesigner lui-même, soit un intégrateur (profil intermédiaire entre webdesigner et développeur) qui effectuera le travail de conversion du PSD vers le format HTML.

La répartition des tâches peut varier d'une entreprise à l'autre, et même d'un projet à l'autre. Il arrive que le mock-up soit produit par le webdesigner d'une agence web (en anglais « web agency »), qui envoie le mockup au format PSD à une société de services, dans laquelle un intégrateur va transformer le PSD en une maquette HTML/CSS, maquette qui va être transmise au client final. Au sein de ce client final, un développeur va récupérer la maquette HTML/CSS pour l'intégrer dans une application en mettant le code HTML et CSS aux normes du projet concerné, et en ajoutant les règles métier spécifiques à l'application.

Dans les grosses structures, on peut aussi trouver des intégrateurs, dont le travail est d'intégrer la maquette HTML/CSS au sein de l'application concernée, de façon à ce que le développeur puisse se concentrer sur le développement des règles métier.

Exercice : les explications ci-dessus vous semblent sans doute théoriques. Essayez de vous représenter sous forme d'un schéma le chemin qu'emprunte un mock-up, de sa conception, jusqu'à son intégration dans un projet. Faites des recherches sur internet pour comprendre ce que sont les profils suivants :

- Web-designer
- Intégrateur web
- Développeur web

## 2.7.2 Structure type d'une page HTML

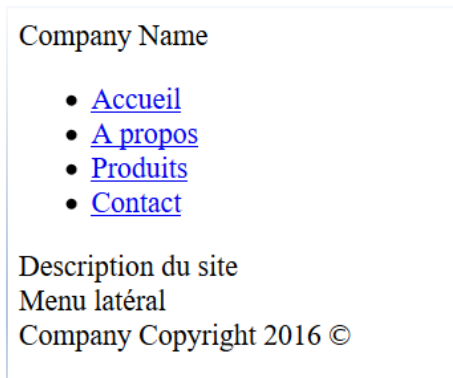
Pour rappel, voici la structure de page type que nous avons évoquée au chapitre précédent :

- Un bandeau « entête » (le « header »)
- Un menu de navigation horizontal
- Une partie facultative contenant généralement un texte de présentation du site et/ou de la page
- Le corps de la page (souvent désigné par le terme anglais « main » pour « principal »), lui-même constitué de :
  - o Un bandeau vertical gauche (facultatif) pouvant contenir un sous-menu vertical
  - o Un bandeau central plus large présentant les données importantes
  - o Un bandeau vertical droit (facultatif lui aussi, il est plutôt rare de combiner bandeau gauche et droit, question d'équilibre)
- Un bandeau « pied de page » (le « footer »)

Voici un squelette de page HTML qui se rapproche beaucoup de notre structure type :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Prototype de page</title>
</head>
<body>
  <header id="header">
    <div id="logo">Nom de la société</div>
  </header>
  <nav id="navbar">
    <ul>
      <li><a href="#">Accueil</a></li>
      <li><a href="#">A propos</a></li>
      <li><a href="#">Produits</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
  </nav>
  <div id="welcome"></div>
  <section class="container">
    <section class="content">
      Description du site
    </section>
    <aside class="sidemenu">
      Menu latéral
    </aside>
  </section>
  <footer id="footer">Nom de la société - Copyright 2016 &copy;</footer>
</body>
</html>
```

Si l'on observe le résultat obtenu dans un navigateur, on voit que cela ne ressemble pas à grand-chose :



Ajoutez les balises « style » et « style » dans la partie « head » du code HTML, et placez-y le code suivant :

```
<style>
    .container {
        padding-right: 15px;
        padding-left: 15px;
        margin-right: auto;
        margin-left: auto;
    }
</style>
```

Rafraîchissez la page dans votre navigateur, vous devriez constater que certains éléments sont mieux centrés. Ceci est dû à la définition de la classe CSS « container » que nous venons d'ajouter dans le code CSS de la page. Regardez dans le code HTML quels éléments sont affectés à cette classe « container ». Combien en dénombrez-vous ?

Ajoutez un peu de code CSS derrière la classe « container » :

```
<style>
    .container {
        padding-right: 15px;
        padding-left: 15px;
        margin-right: auto;
        margin-left: auto;
    }
    div, nav, section, footer {
        border: solid black 1px;
    }
</style>
```

Observez le résultat obtenu :

Nom de la société
<ul style="list-style-type: none"><li>• <a href="#">Accueil</a></li><li>• <a href="#">A propos</a></li><li>• <a href="#">Produits</a></li><li>• <a href="#">Contact</a></li></ul>
Description du site
Menu latéral
Nom de la société - Copyright 2016 ©

Explication : nous avons indiqué dans le CSS que nous souhaitons que les balises « div », « nav », « section » et « footer » soient entourés d'une bordure pleine, de couleur noire et d'un pixel de large :

```
div, nav, section, footer {  
    border: solid black 1px;  
}
```

C'est très pratique de disposer de cette bordure, au moins temporairement, pour mieux voir comment les différents éléments se répartissent dans la page.

Je vous invite à redéfinir ces éléments séparément au niveau du CSS, avec des couleurs différentes de manière à mieux les repérer dans la page.

Notre menu composé de 4 options est moche, on souhaite faire disparaître les puces, et on souhaite disposer les liens sur une seule ligne. Ajoutons le code CSS suivant :

```
nav ul {  
    list-style: none;  
}  
nav ul > li {  
    display: inline-block;  
}
```

Rafraîchissez la page :

Nom de la société
<a href="#">Accueil</a> <a href="#">A propos</a> <a href="#">Produits</a> <a href="#">Contact</a>
Description du site
Menu latéral
Nom de la société - Copyright 2016 ©

C'est pas mal, mais vous commencez à deviner qu'il va nous falloir beaucoup de travail avant d'obtenir un rendu de page digne de ce nom. Et surtout, vous devinez que nous devons comprendre le CSS, si nous voulons avancer dans notre maîtrise du développement web. C'est la raison pour laquelle je vous invite à poursuivre cette étude avec le support de cours « CSS3 – Premiers pas ».

## 2.7.2 Outils de construction d'un mock-up

Pour les lecteurs pressés qui auraient « sauté » allègrement le chapitre précédent, je signale que la définition du terme « mock-up » se trouve dans ce chapitre.

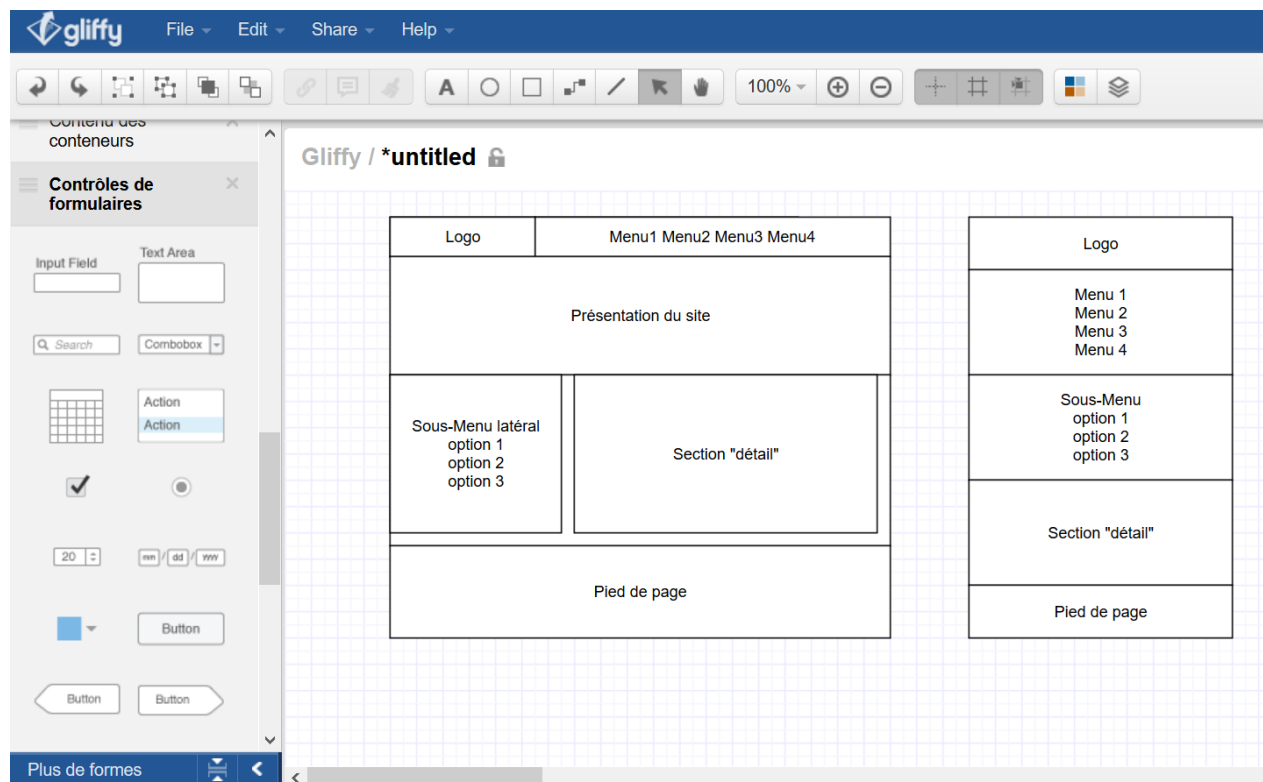
Lors du processus d'élaboration d'un mock-up, il peut être intéressant de s'aider d'outils graphiques.

L'application en ligne Gliffy peut se révéler un auxiliaire précieux :

<https://www.gliffy.com/go/html5/launch>

On voit ci-dessous deux exemples d'une même page :

- à gauche pour un affichage dans un navigateur en mode desktop
- à droite pour un affichage dans un smartphone



Si l'on souhaite disposer d'un logiciel desktop équivalent à Gliffy, et gratuit, le logiciel open-source Inkscape est sans aucun doute un bon choix :

<https://inkscape.org/fr/>



Et bien évidemment, il y a Photoshop, mais il s'agit là d'un logiciel nécessitant l'achat d'une licence.

Pour tout savoir sur la manière de maquetter un site web avec Photoshop :

<https://openclassrooms.com/courses/maquettez-votre-site-responsive-avec-photoshop>



### 3. Liens utiles

<https://developer.mozilla.org/fr/docs/Web/HTML>

<https://www.w3schools.com/html/>

<http://learn.shayhowe.com/html-css/building-your-first-web-page/>

<https://www.alsacreations.com/astuce/>

## 4. Annexe

### 4.1 Lorem ipsum

Les développeurs webs parlent quelquefois de « lorem ipsum ».

Exemple : « ...non, on ne sait pas encore ce que le client va vouloir dans cette « div »... en attendant tu n'as qu'à y coller un peu de « lorem ipsum »...

Bon, d'accord... mais c'est quoi « lorem ipsum » ?

Eh bien c'est du texte écrit en faux latin, bref, du texte « bidon » utilisé depuis fort longtemps par le monde de l'imprimerie, et plus récemment par le monde du développement web. On trouve sur internet des sites webs générant du « lorem ipsum », il suffit de copier-coller des portions de texte et de remplir ses pages HTML avec, et le tour est joué 😊.

Un site de référence en la matière, c'est celui-ci :

<http://fr.lipsum.com/>

La page d'accueil du site explique dans les grandes lignes les origines du lorem ipsum. En bas de cette même page, on trouve un petit formulaire permettant de générer le nombre de paragraphes « lorem ipsum » que l'on souhaite... Et voilà le travail :

*Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis elementum et odio non varius. Morbi semper luctus libero sed ornare. Pellentesque id metus urna. Etiam aliquet ut tellus egetas bibendum. Vestibulum urna sapien, malesuada at vestibulum vulputate, finibus eu orci. Fusce fringilla, elit ullamcorper euismod faucibus, magna quam tempus metus, venenatis auctor diam neque at sem. Integer porttitor bibendum eleifend. Vivamus vestibulum nisl vitae pellentesque imperdiet. Quisque sit amet felis felis. Vivamus ullamcorper finibus neque, at interdum lorem consequat in. Interdum et malesuada fames ac ante ipsum primis in faucibus. Aenean efficitur id*

*Nulla orci sapien, tincidunt eget odio non, euismod lacinia ipsum. Sed dignissim ipsum non ornare facilisis. Interdum et malesuada fames ac ante ipsum primis in faucibus. Aliquam accumsan tempus dui, sit amet porta metus imperdiet ac. Sed in lacus aliquam ex rhoncus consectetur nec ut erat. Phasellus placerat, neque sit amet elementum ultricies, ipsum augue malesuada nulla, ut hendrerit justo sem fermentum est. Nullam sed molestie tellus. In hac habitasse platea dictumst. Etiam vitae velit orci. Duis odio lorem, rutrum a volutpat a, tristique viverra tortor.*

Solution concurrente proposant le même type de service :

<http://www.blindtextgenerator.com/lorem-ipsum>



[GENERATOR](#) | 
 [HTML SNIPPETS](#) | 
 [ABOUT DUMMY TEXT](#) | 
 







## 4.2 Squelette de page HTML type

Voici le code source complet de la page type que nous avons construite au chapitre 2.7.2 :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Prototype de page</title>
  <style>
    .container {
      padding-right: 15px;
      padding-left: 15px;
      margin-right: auto;
      margin-left: auto;
    }
    div, nav, section, footer {
      border: solid black 1px;
    }
    nav ul {
      list-style: none;
    }
    nav ul > li {
      display: inline-block;
    }
  </style>
</head>
<body>
  <header id="header">
    <div id="logo">Nom de la société</div>
  </header>
  <nav id="navbar">
    <ul>
      <li><a href="#">Accueil</a></li>
      <li><a href="#">A propos</a></li>
      <li><a href="#">Produits</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
  </nav>
  <div id="welcome"></div>
  <section class="container">
    <section class="content">
      Description du site
    </section>
    <aside class="sidemenu">
      Menu latéral
    </aside>
  </section>
  <footer id="footer">Nom de la société - Copyright 2016 &copy;</footer>
</body>
</html>
```



## 5. Changelog

Version 1.2 publiée le 01/07/2017 :

- correction de quelques coquilles dans certains exemples de code HTML et CSS
- ajout d'un chapitre sur les balises et attributs HTML dépréciés
- ajout d'un chapitre pour les lecteurs pressés (chapitre 1.4)
- ajout de compléments d'infos sur les balises « audio » et « video »
- modification du titre du chapitre 2.7 (pour mieux le distinguer du chapitre 2.5)