



Rappels

Les versions HTML



C'est le World Wide Web Consortium (W3C) qui est chargé de l'évolution et de la normalisation du langage HTML. Depuis 1992, pas moins de cinq normes se sont succédées :

- La première version de HTML date de 1992. Bien loin de notre HTML actuel, elle ne comprenait que quelques balises de mise en forme, les balises de liens et quelques autres aujourd'hui obsolètes.
- Le HTML+, puis le HTML 2.0, sa version officielle, arrivent en 1994 et apportent deux grandes nouveautés : les images et les formulaires.
- La version suivante, de 1996, porte le numéro 3.2. C'est la version la plus utilisée aujourd'hui. Elle comprend la majorité des balises utilisées actuellement et offre, entre autres, les tables, la mise en forme de texte autour d'images ainsi que le support des applets Java.
- La version 4.0, officialisée en décembre 1997, a été corrigée en avril 1998 et décembre 1999. La grande nouveauté de cette version 4 est le support des frames et des feuilles de styles.
- En janvier 2000, le W3C publie les spécifications de XHTML 1.0, le futur langage pour la création de pages Web. Le XHTML, eXtended HyperTextMarkup Langage, permettra de définir ses propres balises, comme c'est déjà possible avec le XML, un autre langage de description de documents très puissant.

Rappels

Les versions HTML



Pour le W3C, le HTML était fini depuis la version 4. Il commença à développer le XHTML 2, avec le XML pour décor. Une rébellion se dessina au sein du W3C.

Les mécontents formèrent leur propre groupe : le Web Hypertext Application Technology Working Group, ou WHATWG pour faire court.

Au début, le gros travail du WHATWG portait sur deux spécifications : Web Forms 2.0 et Web Apps 1.0. Ces deux spécifications étaient destinées à compléter le HTML. Avec le temps, elles furent regroupées dans une seule spécification, simplement appelée HTML5.

Pendant que le WHATWG développait le HTML5, le W3C continuait à travailler sur le XHTML2.

En 2009, le W3C annonça que la charte du XHTML2 ne serait pas renouvelée.

Le HTML5 est la dernière itération en date.

Rappels

Les DTD

HTML4.01 strict

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

XHTML 1.0 strict

```
<!DOCTYPE html PUBLIC "-// W3C//DTD XHTML 1.0 strict//EN"  
" http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

HTML5

```
<!DOCTYPE html>
```

Une déclaration de type de document, ou doctype, est traditionnellement utilisée pour spécifier le type de balisage du document.

Le doctype de l'HTML 4.01 ressemble à ceci :

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

Ci-dessous, le doctype du XHTML 1.0 :

```
<!DOCTYPE html PUBLIC "-// W3C//DTD XHTML 1.0 strict//EN"  
" http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Ces déclarations sont difficilement compréhensibles, mais elles ne font que dire à leur façon ce document est écrit en HTML 4.01 ou ce document est écrit en XHTML 1.0.

On pourrait penser que le doctype qui déclare ce document est écrit en HTML5 comporte le numéro cinq quelque part, mais ce n'est pas le cas. La doctype de l'HTML5 ressemble à cela :

```
<!DOCTYPE html>
```

Rappels

Les différents composants d'un document HTML

Les bases d'un code HTML5 ressemblent beaucoup à celles rédigées à l'aide des versions précédentes HTML4 et XHTML. On y retrouve un *doctype* suivi des éléments les plus courants :

<html>, <head>, <title>, <meta>, <link>, <script> et <body>.

Un document HTML se compose de trois parties :

- une ligne contenant les **informations de version HTML**;
- une section en-tête déclarative (enclose par l'élément **HEAD**) ;
- un corps, qui emporte le contenu effectif du document. Le corps peut être implémenté par l'élément **BODY**.

Voici un exemple de document HTML simple :

```
<!doctype html>
<html lang="fr">
<head>
    <meta charset="utf-8">
    <title>Titre de la page</title>
    <link rel="stylesheet" href="style.css">
    <script src="script.js"></script>
</head>
<body>
    ...
    <!-- Le reste du contenu -->
    ...
</body>
</html>
```

Rappels

Les différents composants d'un document HTML

L'élément racine `<html>` va recueillir les deux principaux éléments de la hiérarchie :

`<head>` et `<body>`.



Les fameux `<head>` et `<body>`

À ce niveau, le code HTML peut être divisé en deux parties : l'en-tête située entre les balises `<head>` et `</head>`, qui regroupe toutes les méta-information, c'est-à-dire les données qui ne sont pas représentées directement à l'écran dans le rendu du document, mais qui lui sont tout de même liées : le titre dans `<title>` (extrêmement recommandé), les autres méta-information variées (facultatives : mots-clés, description générale de la page) dans zéro ou plusieurs `<meta>`, et éventuellement `<link>`, `<script>` ou `<style>`.

La deuxième partie comprend le corps à proprement parler, c'est-à-dire `<body>`. On y retrouvera tout le reste du contenu HTML, structuré par des balises variées - selon ce que l'on aura à y placer - et dont l'apparence sera affectée par les styles CSS chargés dans `<link>` et/ou `<style>`.

Rappels

Les commentaires

Un **commentaire** en HTML est un texte qui sert simplement de mémo. Il n'est pas affiché, il n'est pas lu par l'ordinateur, cela ne change rien à l'affichage de la page.

```
1 <!-- Ceci est un commentaire -->
```

Cela sert à *vous* et aux personnes qui liront le code source de votre page. Vous pouvez utiliser les commentaires pour laisser des indications sur le fonctionnement de votre page.

Quel intérêt ? Cela vous permettra de vous rappeler comment fonctionne votre page si vous revenez sur votre code source après un long moment d'absence.

Vous pouvez le mettre où vous voulez au sein de votre code source : il n'a aucun impact sur votre page, mais vous pouvez vous en servir pour vous aider à vous repérer dans votre code source.

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <!-- En-tête de la page -->
5         <meta charset="utf-8" />
6         <title>Titre</title>
7     </head>
8
9     <body>
10        <!-- Corps de la page -->
11    </body>
12 </html>
```

Rappels

La structure composite d'un document HTML

Le HTML peut utiliser, dans un même document, plusieurs vocabulaires de façon indépendante.

The screenshot shows a Microsoft Notepad window titled "style - Bloc-notes". The content of the file is a CSS style sheet:

```
background: #a77b7c;
color: #FFFFFF;
border-right:#FFFFFF solid 1px;
padding-top:0px;
font-size:12px;
}
#menu li a {
padding: 0px 30px;
text-align: center;
text-decoration: none;
background: #a77b7c;
color: #FFFFFF;
font-family:trebuchet ms;
font-weight:bold;
}
#menu li a:hover, #menu li a:focus, #menu li a:active {
background: #641616;
text-decoration: underline;
}
/* CONTENU DE LA PAGE */
#contenu {
padding:20px;
}
h1 {
font-family: "trebuchet ms";
font-size: 15px;
color:#FF0000;
text-align:center;
margin-bottom:20px;
}
h2 {
font-family: "verdana";
font-size: 15px;
color:#4d4dad;
font-variant: small-caps;
}
/* PIED DE PAGE */
#piedDePage{
background-color:#FFFFFF;
text-align:center;
font-family:verdana;
font-size:9px;
height:15px;
padding-top:2px;
color:#5a0000;
}
```

To the right of the code, there is a graphic featuring the Java logo (a steaming coffee cup) and the word "JavaScript" in red. Below these are two overlapping red 3D-style shapes with the letters "FL" on them, which likely represent the Flash logo.

Rappels

Le modèle de document (DOM)

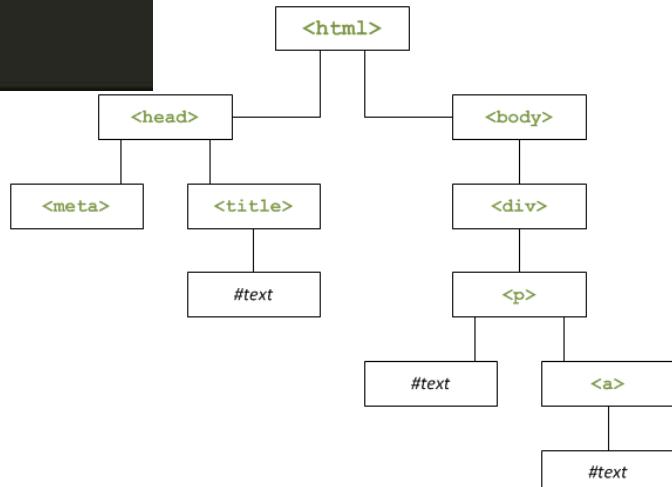
Le *Document Object Model* est une structure de données qui représente un document HTML comme une arborescence.

La racine de cet arbre est un nœud nommé *document* qui correspond grossièrement à la balise <html>.

Les balises HTML définies dans la page Web correspondent aux nœuds de l'arbre DOM et en constituent la structure.

Le langage jQuery est en mesure d'interroger le DOM pour connaître les caractéristiques (attributs et valeurs HTML, propriétés et valeurs CSS) des balises qui constituent un document HTML, mais aussi de modifier ces éléments pour changer l'allure et/ou le contenu du document.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Hello World!</title>
5   </head>
6
7   <body>
8
9     <script>
10    |   alert('Hello world!');
11
12    </script>
13
14   </body>
15 </html>
```

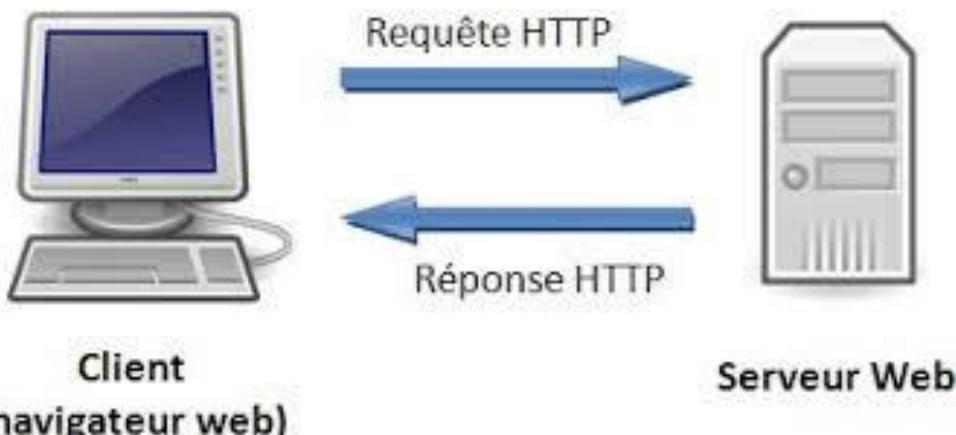


Rappels

Le protocole HTTP

Le Hypertext Transfer Protocol, plus connu sous l'abréviation HTTP, le « protocole de transfert hypertexte », est un protocole de communication client-serveur développé pour le World Wide Web.

Il est utilisé pour échanger toute sorte de données entre client HTTP et serveur HTTP.



Rappels

Interaction client / serveur HTTP

Un ensemble de lignes de commandes est envoyé au serveur par le biais du navigateur afin d'effectuer une requête HTTP.

La ligne de commande comprend la commande (HEAD, GET, POST...), l'URL (l'adresse de la page sur le serveur) et la version de protocole (la version de HTTP utilisée).

```
GET /fichier.ext HTTP/1.1
```

```
Host: www.site.com
```

```
Connection: Close
```

```
<nouvelle ligne>
```

Exemple de réponse :

```
HTTP/1.1 200 OK
```

```
Date: Thu, 11 Jan 2007 14:00:36 GMT
```

```
Server: Apache/2.0.54 (Debian GNU/Linux) DAV/2 SVN/1.1.4
```

```
Connection: close
```

```
Transfer-Encoding: chunked
```

```
Content-Type: text/html; charset=ISO-8859-1
```

Rappels

Interprétation du HTML par le navigateur

Le navigateur est le programme qui nous permet de voir les sites web. Le travail du navigateur est de lire le code HTML et CSS pour afficher un résultat visuel à l'écran. Si votre code CSS dit « Les titres sont en rouge », alors le navigateur affichera les titres en rouge. Le rôle du navigateur est donc essentiel !

Chaque navigateur interprète le code à sa manière donc il faudra vous y faire et prendre l'habitude de vérifier régulièrement que votre site fonctionne correctement sur la plupart des navigateurs.

Tous les navigateurs, même moderne, ne sont pas en phase pour prendre en compte les nouveautés HTML5.

Tester les différents navigateurs sur fmbip.com ou html5test.com ou canuse.com afin de s'apercevoir de la prise en charge ou non des nouveautés HTML5.

The screenshot displays three separate browser windows side-by-side, each showing results for browser support of various HTML5 features.

- findmebyIP.com:** A dark-themed tool for checking IP address. It shows the IP address 82.234.101.121. Below it, a large white banner says "What's my IP?" with a link to "Top". Navigation links include Browser, CSS3, HTML5, Forms 2.0, CSS3 Selectors, Script, and IP & Location.
- html5test.com:** A light-themed test site. At the top, it says "YOUR BROWSER SCORES 471 OUT OF 555 POINTS" and "You are using Firefox 32.0 on Windows 8.1". It includes sections for "semantics", "Parsing rules", and "multimedia". The "Parsing rules" section has a table:

	10	33/35
<!DOCTYPE html> triggers standards mode	Yes ✓	Yes ✓
HTML5 tokenizer	Yes ✓	No ×
HTML5 tree building	Yes ✓	No ×
HTML5 defines rules for embedding SVG and MathML, create a regular HTML document. Support for SVG and MathML is not required though, no actual points are awarded if your browser supports embedding these two		
video element	Yes ✓	Yes ✓
DRM support	Yes ✓	No ×
Media Source extensions	Yes ✓	No ×
Subtle support	Yes ✓	Yes ✓
Poster image support	Yes ✓	Yes ✓

- canuse.com:** A light-themed test site. It shows a "PROFESSIONAL HTTP/S SNIFTER" button and a "DOWNLOAD FREE" button. It also includes sections for "semantics" and "multimedia".

Rappels

Position des navigateurs

Depuis l'apparition des CSS3, la frilosité des développeurs à utiliser CSS3 du fait de la non prise en charge commune et généralisé par les navigateurs et surtout la non recommandation par le W3C. Tous les constructeurs proposent leur préfixage. Donc tous les navigateurs utilisent des moteurs de rendu différents.

WebKit est utilisé sur :

- ❖ Safari



Blink est utilisé sur :

- ❖ Chrome
- ❖ Opera

Gecko est utilisé sur :

- ❖ Firefox



Trident est utilisé sur :

- ❖ Internet explorer



Introduction au HTML5



- Les objectifs de HTML5
- Le WhatWG versus W3C
- Les principaux concepts et apports
- La " roadmap " de HTML 5
- Quand utiliser HTML 5 ?
- Les différents composants : balises, formulaires, audio, vidéo, API, etc.
- Outils de développement HTML 5
- Test de compatibilité, méthode de détection HTML 5

Introduction au HTML5

Les objectifs du HTML5

Répondre aux besoins d'hier :

- ❖ par l'amélioration des formulaires,
- ❖ par une meilleure structuration (amélioration de la sémantique).

Répondre aux besoins d'aujourd'hui :

- ❖ par l'intégration de nouveaux médias vidéo, audio, ...
- ❖ par la prise en compte de comportements tels que ceux réalisés en ajax.

Répondre aux besoins de demain :

- ❖ par le support du mode déconnecté,
- ❖ par le support d'événements serveur.

Beaucoup de carences sont comblées en HTML5 aussi bien du point de vu de la structure du document comme le prouvent les balises header et footer que du point de vu du contenu (par exemple la balise canvas ou encore le support du SVG) et du comportement (exemple de l'attribut focus dans les formulaires).

HTML



Introduction au HTML5

Le WhatWG versus W3C



Le W3C, pour World Wide Web Consortium, est un organisme de standardisation ayant joué un rôle prépondérant dans l'évolution du web dans son sens large. L'organisme est chargé d'analyser les nouvelles technologies pour en créer des standards auxquels les développeurs peuvent se référer. Le W3C collabore depuis des années avec un groupe de travail, le WHATWG. Mais ce dernier a décidé de faire cavalier seul.

Le WHATWG, dont fait également partie Google, va clairement avancer plus vite que le W3C. Les éditeurs de navigateur vont se référer au travail fait par le groupe, mais il ne s'agit pas pour autant de se détourner du W3C. Les deux travaux auront un certain degré de complémentarité. Il est simplement dommage que le manque d'accord sur les questions d'organisation mène à la création de deux versions différentes d'un même standard.

La séparation est d'autant plus regrettable que le HTML5 avait déjà ses propres difficultés. En effet, il ne faut pas oublier que la norme n'en est pas encore officiellement une : il s'agit d'un brouillon. Bien que l'on en soit à plusieurs versions, tous les aspects n'ont pas encore été finalisés. Le support réalisé par les navigateurs est donc forcément temporaire, les points finalisés étant mis à jour progressivement.

Introduction au HTML5

Les concepts et apports

La sémantique

Les médias riches

Les formulaires améliorés

Structure de la page

Les apports plus spectaculaires sont :

- ❖ l'intégration de l'audio et la vidéo,
- ❖ l'intégration de dessin en 2D par les canvas,
- ❖ la prise en compte des formats de dessin vectoriels SVG.

La structuration du document est grandement améliorée :

- ❖ une multitude de balises ont désormais des rôles précis,
- ❖ il ne devrait plus être nécessaire de recourir systématiquement aux div.

Les formulaires peuvent utiliser de nouveaux objets. Il existe des entrées :

- ❖ pour les dates,
- ❖ pour les couleurs,
- ❖ pour les E-mails,
- ...

Allié aux CSS3 des contraintes de présentations sont désormais révolues :

- ❖ possibilité de dimensionner une image de fond,
- ❖ possibilité de réaliser des animations.

Introduction au HTML5

La roadmap

Depuis mai 2011, les apports de fonctionnalités sont figés. Il est prévu que la norme soit finalisée en 2014.

Le W3C ayant émis le respect du standard, les navigateurs tentent de rattraper leur concurrents sur ce terrain.

Les smartphones ont eu un rôle primordial dans la mouvance vers un norme. Ils favorisent l'adoption de technologies standards au détriment de technologies propriétaires telles que *Flash*.

En 2014 il est prévu que le trafic web provienne majoritairement des téléphones mobiles.

Il est enfin révolu le temps où l'on trouvait judicieux de placer le message écrit : ce site est optimisé pour Internet Explorer en 800 par 600 et aucun concepteur web consciencieux ne s'en plaindra !

Introduction au HTML5

Quand utiliser l'HTML5?

si vous voulez commencer à utiliser les nouveaux éléments structurels de l'HTML5 dès aujourd'hui, rien ne vous empêche. La plupart des navigateurs vous permettront de ces nouveaux éléments.

Non pas parce que ces navigateurs supportent activement ces éléments, mais parce que la plupart des navigateurs vous permettent d'utiliser et de styler n'importe quel élément de votre création

Le HTML 5 n'est pas officiellement validé.

Il y aura des écarts entre le support actuel et la norme finale.

Tous les navigateurs modernes supportent les spécifications actuelles.

Partir sur des spécifications mouvantes présente un risque !

La prise en compte des spécifications actuelles par les navigateurs limite ce risque.

Il est donc conseillé de s'intéresser à cette technologie pour tout nouveau projet.

Il faut cependant s'assurer du rendu correct sur l'ensemble des navigateurs visés **et notamment les smartphones**.

Introduction au HTML5

Outils de développement HTML5

Pour mener à bien l'ensemble des tâches que nous allons réaliser, nous avons besoin d'éditeurs de code.

- ❖ Sur Windows, nous pouvons avoir recours à ***NotePad++***
- ❖ Sur Mac, depuis la plate-forme ***github***, il est possible de télécharger ***fraise***; également ***Bluefish*** sur MacOS.

Sinon des solutions plus musclées, ***Aptana*** qui est une structure plus lourde qui permet la gestion de sites ou encore ***BlueGriffon***.

Adobe Bracket est un projet qui vient de voir le jour, est un éditeur de code un peu particulier qui permet continuellement de pouvoir éditer son code ou que l'on soit.

Mozilla FireFox est un navigateur Web permettant de télécharger des modules complémentaires qui vous faciliteront la vie pour la réalisation de vos pages Web.

Modules complémentaires :

- ❖ Firebug
- ❖ Webdeveloper
- ❖ Cache statuts
- ❖ HTML validator

Les logiciels de la suite Adobe restent incontournables pour élaborer des projets innovants pour le web.

Introduction au HTML5

Méthode de détection



```
<!--[if lt IE 9]>
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->
```

Une des problématiques assez embêtantes de l'utilisation des balises HTM5, c'est qu'elles ne sont pas reconnues par Internet explorer.

Prenons un exemple, insérons un style sur la balise <article>, celui-ci est reconnu sur Internet explorer 9 par contre sur les versions antérieures ce dernier n'est pas reconnu. Faisons le test avec IEtester!!!

Il existe une approche permettant d'activer toutes les balises HTML5. Il est donc possible d'aider Internet Explorer à comprendre ces balises en insérant un commentaire conditionnel entre les balises et de notre document.

vous trouverez ce commentaire conditionnel sur le site suivant :

<https://code.google.com/p/html5shiv/>

Introduction au HTML5

Méthode de détection



Télécharger le fichier depuis <http://modernizr.com>

Au-delà de HTML5Shiv, Modernizr est une librairie qui est un peu le couteau suisse lorsqu'on fait du développement en HTML5.

Modernizr est un fichier Javascript très utile qui détecte le support des types de champs et des éléments audio, vidéo et canvas.

Ce script crée un objet en Javascript appelé Modernizr. En interrogeant les propriétés de cet objet, vous saurez si le navigateur supporte ou non tel type de champ.

```
<script>
if (Modernizr.canvas) {
    document.write("la balise canvas est utilisable");
} else {
    document.write("la balise canvas n'est pas utilisable");
}
</script>
```




Nouvelle structuration

- Les éléments traditionnels
- L'élément Header
- L'élément nav
- L'élément section
- L'élément article
- L'élément aside
- L'élément footer
- Modèle de contenu
- Les autres balises de structure
- Les nouvelles catégories de contenu
- La validation des pages
- Intérêt de ces nouvelles balises de structures
- Type en ligne et bloc

Nouvelle structuration

Les éléments traditionnels

Un document HTML5 minimal

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Titre</title>
  <link rel="stylesheet" href="style.css">
  <script src="script.js"></script>
  </head>
  <body>
    </body>
  </html>
```

La conception d'une page web en suivant la philosophie du standard HTML5 spécifié par le W3C répond à quelques réflexes de base. La structure d'un document est simplifiée pour répondre aux attentes pragmatiques des intégrateurs et webdesigners.

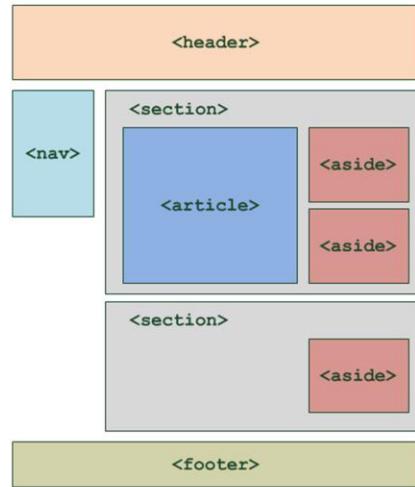
Sans surprise, les bases d'un code HTML5 ressemblent beaucoup à celles rédigées à l'aide des précédentes versions HTML 4 et XHTML. Rétro-compatibilité oblige. On y retrouve donc un *doctype* suivi des éléments les plus courants : `<html>`, `<head>`, `<title>`, `<meta>`, `<link>`, `<script>` et `<body>`.

Nouvelle structuration

Les éléments traditionnels

structuration avec les nouvelles balises HTML5

Les éléments de section ([section](#), [article](#), [nav](#), [aside](#), [header](#), [footer](#)) segmentent des portions du document ou de l'application web.



En général, une page web est constituée d'un en-tête (tout en haut), de menus de navigation (en haut ou sur les côtés), de différentes sections au centre... et d'un pied de page (tout en bas).

Nous allons nous intéresser aux nouvelles balises HTML dédiées à la structuration du site. Ces balises ont été introduites par HTML5 et vont nous permettre de dire : « Ceci est mon en-tête », « Ceci est mon menu de navigation »...

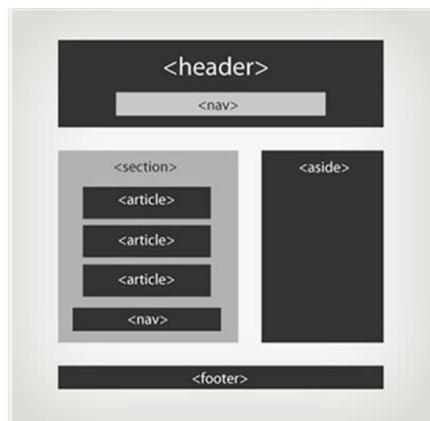
Notes :

Nouvelle structuration

Les éléments traditionnels

De nouveaux éléments sémantiques

Au-delà de la rénovation syntaxique, HTML5 introduit de nouveaux éléments inédits dotés de sens et apportant une alternative aux blocs génériques `<div>` et ``



Nom	Détails
<code><section></code>	Section générique regroupant un même sujet, une même fonctionnalité, de préférence avec un en-tête, ou bien section d'application web
<code><article></code>	Section de contenu indépendante, pouvant être extraite individuellement du document ou syndiquée (flux RSS ou équivalent), sans pénaliser sa compréhension
<code><nav></code>	Section possédant des liens de navigation principaux (au sein du document ou vers d'autres pages)
<code><aside></code>	Section dont le contenu est un complément par rapport à ce qui l'entoure, qui n'est pas forcément en lien direct avec le contenu mais qui peut apporter des informations supplémentaires.
<code><header></code>	Section d'introduction d'un article, d'une autre section ou du document entier (en-tête de page).
<code><footer></code>	Section de conclusion d'une section ou d'un article, voire du document entier (pied de page).

Nouvelle structuration

Les éléments traditionnels

Un Doctype simplifié

```
<!doctype html>
```

Ce doctype allégé a été mûrement testé et réfléchi : il permet toujours pour les anciens navigateurs de rester dans le mode d'interprétation conforme aux standards.

Elle vous est fidèle depuis que vos documents web sont conformes au W3C. Toujours en tête de vos pages, la *Déclaration de type de document* (DTD), appelée par le Doctype, annonce fièrement au navigateur quel est le langage et la grammaire que vous lui faites lire.

Voici à titre indicatif, deux Doctype couramment employés de nos jours, le HTML4.01 strict et le XHTML 1.0 transitionnel.

Doctype HTML 4.01 strict

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

Doctype XHTML 1.0 transitionnel

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitionnal//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitionnal.dtd">
```

A l'ère naissante de HTML5, la syntaxe du Doctype a été remaniée de façon tellement optimisée qu'elle en devient largement plus facile à retenir.

Nouvelle structuration

L'élément header

<header>

L'élément **au moment où une non** représente le bloc d'en-tête d'une section ou d'une page. Il remplace avantageusement son homologue classique `<div id="header">`, mais ne doit pas forcément être considéré comme un élément unique dans le document : toute section est susceptible de disposer de son header

`<header>`

Ici tout le contenu de l'en-tête de document ou de section

`</header>`

La spécification HTML5 décrit d'éléments **header** comme un conteneur pour « un groupe d'outils d'introduction ou de navigation ». Un document peut comporter plusieurs éléments **header**, vous pouvez par exemple utiliser un élément **header** dans un élément **section**. La spécification décrit l'élément **section** comme regroupement de contenu thématique, comprenant généralement un en-tête.

```
<section>
  <header>
    <h1>Spécification</h1>
  </header>
  <p>Support de cours HTML5 – CSS3</p>
</section>
```

Un **header** apparaîtra normalement en haut d'un document ou d'une section, mais ce n'est pas obligatoire. Il est défini par son contenu plutôt que par sa position.

Nouvelle structuration

L'élément nav

<nav>

<nav> a pour fonction de regrouper les liens de navigation considérés comme majeurs ou jugés suffisamment pertinents.

```
<nav>
  <ul>
    <li><a href="index.html">Page d'accueil</a></li>
    <li><a href="contact.html">Contact</a></li>
  </ul>
</nav>
```

L'élément <nav> fait exactement ce qu'on attend de lui. Il contient des informations de navigation, généralement une liste de liens.

L'élément <nav> est conçu pour les informations de navigations principales. Une simple liste de liens ne justifie pas l'utilisation de l'élément <nav>. En revanche, les liens permettant de naviguer sur un site doivent presque obligatoirement utiliser l'élément nav.

Bien souvent, un élément nav apparaîtra dans un élément header. Cela paraît logique si l'on considère que l'élément header peut être utilisé pour les aides à la navigation.

Nouvelle structuration

L'élément section

<section>

Une `<section>` représente un bloc générique de contenu ayant la même thématique. Cela concerne les chapitres, les en-têtes et pieds de page, ou tout autre partie dans un document. L'élément `<section>` peut contenir des éléments de titre à h1 à h6 pour une meilleure définition de la structure du document.

```
<section>
  <h1>Articles</h1>

  <article>
    ...
  </article>
</section>
```

L'élément `section` est utilisé pour regrouper du contenu de manière thématique. Cela rappelle l'élément `div`, souvent utilisé comme un conteneur de contenu générique. La différence, c'est que `div` n'a aucune signification sémantique. L'élément `section`, en revanche, est utilisé explicitement pour regrouper du contenu apparenté.

Vous pourrez remplacer quelques-uns de vos éléments `div` par des éléments `section`, mais demandez-vous toujours si tout le contenu est apparenté.

```
<section>
  <header>
    <h1>Spécification</h1>
  </header>
  <p>Support de cours HTML5 – CSS3</p>
</section>
```

Nouvelle structuration

L'élément article

<article>

L'élément `<article>` désigne une portion du document potentiellement autonome dans le sens où elle pourrait être reprise ou réutilisée, comme un article de journal, de bloc ou de forum.

```
<article>
  <h1>Titre de l'article</h1>
  <p>Contenu de l'article</p>
</article>
```

L'élément `article` est un autre type de section spécialisée. On l'utilise pour le contenu apparenté autonome. Le problème étant de déterminer ce qui constitue un contenu autonome.

Si le contenu a toujours un sens dans ce contexte, `article` est probablement l'élément à utiliser. En fait, l'élément `article` est conçu spécifiquement pour la syndication.

```
<article>
  <h1>Mon article</h1>
  <p>Support de cours HTML5 – CSS3</p>
</article>
```

Nouvelle structuration

L'élément aside

<aside>

L'élément `<aside>` représente une portion de contenu contextuelle, directement ou indirectement liées aux éléments qui l'entourent, tel un bloc d'archives relatives au contenu précédent. Par extrapolation, cet élément désigne fréquemment les barres latérales classiques du document et peut remplacer l'ancien `<div id="sidebar">`.

```
<aside>
  <h4>Sources de l'article</h4>
  <ul>
    <li><a href="#">Lien 1</a></li>
    <li><a href="#">Lien 2</a></li>
    <li><a href="#">Lien 3</a></li>
  </ul>
</aside>
```

L'élément `aside` est une sorte d'encadré. Quand je dis encadré, ce n'est pas une question de position. Il ne suffit pas que le contenu apparaisse à gauche ou à droite du contenu principal pour utiliser l'élément `aside`. Une fois de plus, c'est le contenu, et non la position, qui est important.

L'élément `aside` doit être utilisé pour du contenu indirectement apparenté. Si vous avez un morceau de contenu que vous considérez comme différent du contenu principal, l'élément `aside` est probablement le bon conteneur. Demandez-vous si le contenu de l'élément `aside` peut-être retiré sans affecter le sens du contenu principal du document ou de la section.

Nouvelle structuration

L'élément footer

<footer>

L'élément `<footer>` regroupe le contenu du pied d'une section ou d'un document et est destiné à recueillir les informations concernant l'auteur, les mentions légales, etc. Tel `<header>`, l'élément `<footer>` peut apparaître à divers endroits du document.

```
<article>
  ...
  <footer>
    <p>Posté par Simon, le
      <time datetime="2012-02-02">2 février 2012</time>
    </p>
  </footer>
</article>
```

Comme l'élément `header`, `footer` semble décrire une position, mais comme pour `header`, ce n'est pas le cas. L'élément `footer` contient plutôt des informations au sujet de l'élément qui le contient: auteur, informations de copyright, lien vers des contenus apparentés, etc.

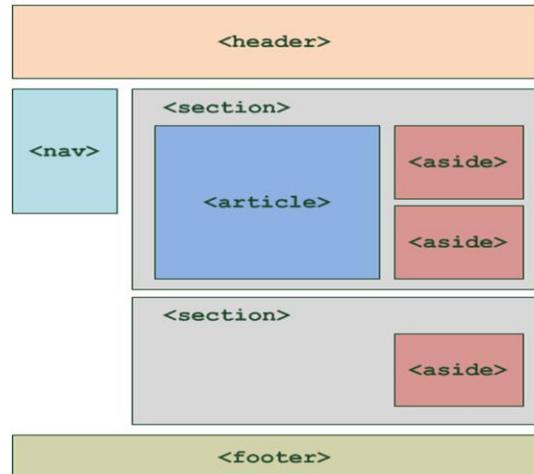
L'HTML5 nous permet également d'en placer à l'intérieur des sections.

```
<section>
  <header>
    <h1>Cours HCS</h1>
  </header>
  <p>HTML5 – CSS3</p>
  <footer>
    <p>Zack's Agency</p>
  </footer>
</section>
```

Nouvelle structuration

Modèle de contenu

Ce code peut vous aider à comprendre comment les balises doivent être agencées. Vous y reconnaîtrez un en-tête, un menu de navigation, un pied de page... Et, au centre, une section avec un article et un bloc aside donnant des informations sur l'auteur de l'article.



```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="style.css" />
    <title>Cours HCS</title>
  </head>

  <body>
    <header>
      <h1>Cours Zack's Agency</h1>
      <h2>Nouveaux éléments HTML5</h2>
    </header>

    <nav>
      <ul>
        <li><a href="#">Accueil</a></li>
        <li><a href="#">Blog</a></li>
        <li><a href="#">CV</a></li>
      </ul>
    </nav>

    <section>
      <aside>
        <h1>À propos de l'auteur</h1>
        <p>C'est moi, Zozor ! Je suis né un 23 novembre 2005.</p>
      </aside>
      <article>
        <h1>Je suis un grand voyageur</h1>
        <p>Bla bla bla (texte de l'article)</p>
      </article>
    </section>

    <footer>
      <p>Copyright Zozor - Tous droits réservés<br/>
        <a href="#">Me contacter !</a></p>
    </footer>
  </body>
</html>
```

Nouvelle structuration

Les autres balises de structure

<hgroup>

La balise `<hgroup>` est utilisée pour grouper des éléments de titres.

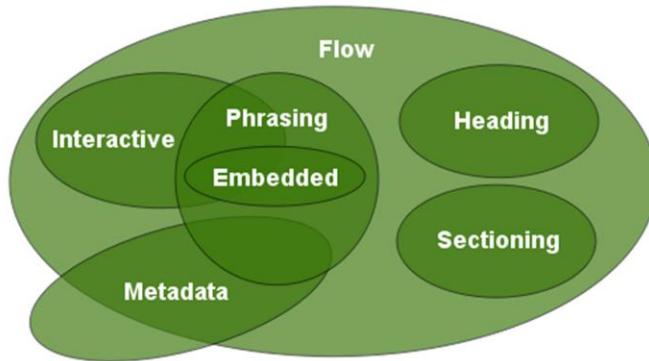
L'élément `<hgroup>` permet de regrouper un ensemble de balises `<h1>` à `<h6>`, quand un titre à plusieurs niveaux (sous-titre).

Exemple d'utilisation de la balise `<hgroup>`

```
<hgroup>
  <h1>Cours HCS</h1>
  <h2>HTML5 – CSS3</h2>
</hgroup>
```

Nouvelle structuration

Les nouvelles catégories de contenus



Le flux (**flow**) regroupe la plupart des éléments courants, c'est-à-dire les autres sous-modèles cités ci-après, ainsi que le contenu texte simple.

Le contenu **interactif** (**interactive**) est destiné à tout ce qui permet une interaction avec l'utilisateur : les liens, les éléments média, les contrôles de formulaire, bref, tout ce qui peut être piloté au clavier, à la souris ou au doigt.

Le contenu de **phrasé** (**phrasing**) correspond à la plupart des éléments pouvant apparaître dans un flux de texte, et qui sont pour la plupart d'entre eux affichés **en ligne** (par exemple en CSS `display:inline` ou `display:inline-block`). On peut y remarquer entre autres `<audio>`, `<video>`, `<iframe>`, `<canvas>`, ``, les éléments de formulaire tels que `<input>`, `<textarea>`, `<button>`, `<select>` et le balisage du texte ``, ``, `<i>`, ``, etc.

Le contenu **embarqué** (**embedded**) est plus spécialisé : `<audio>`, `<canvas>`, `<embed>`, `<iframe>`, ``, `<object>`, `<video>`, `<svg>`, `<math>`.

A titre indicatif, son apparence CSS est définie par des espaces possédant une hauteur et une largeur, souvent affichés avec un type `display:inline-block`.

La **titraille** (**heading**) comprend tous les titres hiérarchiques (`<h1>` à `<h6>` et `<hgroup>`), qui sont eux aussi par défaut affichés en **bloc** par les navigateurs.

Le **contenu sectionnant** (**sectioning**), définit les grandes zones du document HTML ou de l'application web : `<article>`, `<aside>`, `<nav>`, `<section>`. Par consensus, les navigateurs ont choisi de conférer à ces éléments un rendu CSS de type **bloc**.

Les **métadonnées** (**metadata**) ne relèvent pas du contenu principal mais participent à la définition des informations gravitant autour de ce dernier, par exemple le titre du document (`<title>`), le style (`<style>`), les relations externes (`<link>`), et les scripts (`<script>`).

Nouvelle structuration

La validation des pages

Utiliser à bon escient, un validateur est un outil très puissant pour un Webdesigner.

Henri Sivonen a créé un validateur [HTML5 complet à l'adresse](http://validator.nu/)
<http://validator.nu/>

Vous n'avez même pas besoin de mettre à jour l'adresse du validateur W3C (<http://validator.w3.org/>) dans vos favoris. Celui-ci utilise également le parseur d'Henri dès qu'il détecte le doctype HTML5

Nouvelle structuration

Intérêt de ces nouvelles balises structures



Au-delà de la rénovation syntaxique, HTML5 introduit de nouveaux éléments inédits dotés de sens et apportant une alternative au bloc générique `<div>` et ``. Ces derniers se déclinent dorénavant en un large panel d'éléments sémantiques tels que `<article>`, `<section>`, `<aside>`, `<hgroup>`, `<header>`, `<footer>`, `<nav>`, `<time>`, `<mark>`, `<figure>` et `<figcaption>`.

Tous ces éléments peuvent être mis en forme relativement via CSS sur l'ensemble des navigateurs modernes, même si leur fonction propre ne sera véritablement reconnue que sur les toutes dernières générations. Internet Explorer se distingue une fois encore, puisqu'il est l'unique navigateur qui ne parvient pas à se représenter ces éléments. Une alternative en JavaScript, que nous avons vu, lui sera nécessaire.

En termes d'accessibilité et de référencement, l'emploi de ces nouveaux éléments ne bouleverse pas la donne à l'heure actuelle : les agents utilisateurs et moteurs ne reconnaissant pas, il les interprète comme des éléments génériques (tels `<div>` et ``). En revanche, dans un futur encore incertain, le rôle confié à ces éléments peut s'avérer important.

Nouvelle structuration

Type en ligne et bloc

Exemples d'imbrications :

```
<div><p>Paragraphe 1</p><p>Paragraphe 2</p></div>
```

La balise `<div>` (type **block**) englobe les deux paragraphes (type **block**).
Bonne structure!!!

Par contre, nous n'aurions pas pu écrire :

```
<span><p>Paragraphe 1</p><p>Paragraphe 2</p></span>
```

car la balise `` (dont le type est **inline**) n'est pas autorisée à contenir des éléments de type **block** comme les paragraphes `<p>`.

Historiquement, **HTML** ne proposait que deux catégories d'éléments : **les éléments de niveau block et les éléments de niveau inline**. Cette catégorisation autorise ou non certaines imbrications (par exemple un niveau inline ne peut pas contenir de niveau block). En parallèle, les spécifications **CSS** proposent un vaste choix de modes de rendu pour les éléments via la propriété `display`. Parmi les valeurs les plus connues de `display`, l'on retrouve... `block` et `inline`. Forcément, c'est un peu confusant !

Les balises HTML ont **toutes par défaut des propriétés de rendu CSS particulières**. En fait, il existe initialement deux grands groupes principaux de balises : les balises de rendu CSS "bloc" et les balises de rendu CSS "en-ligne". Ces valeurs de rendu visuel coïncident généralement par défaut avec le groupe d'appartenance HTML.

- Un élément de type **block** peut contenir un ou plusieurs autres éléments **block** et/ou **inline**,
- Un élément de type **inline** ne peut contenir qu'un ou plusieurs autres éléments **inline**.

En **HTML5**, la gestion des boîtes imbriquées s'est assoupli...

Vous pouvez envelopper plusieurs éléments dans un élément `a`.

```
<a href=<< # >>></a>
  <h2>Zack's Agency</h2>
  <p>cours HTML5 – CSS3</p>
</a>
```


Les nouvelles balises HTML



- Les balises et attributs obsolètes
- Les images SVG
- Différence Canvas vs SVG
- Les nouvelles balises
- Les extensions de balises HTML existantes
- Les nouveaux attributs
- Les micros formats
- Les formulaires
- Les formats multimédias

Les nouvelles balises HTML

Les balises et attributs obsolètes



Les Has Been!

Adieu content de vous avoir connu

Un certain nombre d'éléments disparaissent de la spécification HTML 5. il s'agit généralement des éléments déjà abandonnés par XHTML :

<frame>, <frameset>, <noframes> (jugés néfaste à l'utilisabilité et à l'accessibilité);
<acronym> disparaît au profit de <abbr>;
accesskey (dans l'élément <a>, <area>, <button>, <input>, <label>, <legend> et <textarea>);
longdesc (dans et <iframe>);
name (dans , <form> et <a>);
language (dans <script>);
summary (dans <table>);
<basefont>, <big>, <center>, , <s>, <strike>, <tt>, <u>, etc.

Les nouvelles balises HTML

Les images SVG

HTML5 introduit un nouvel élément afin d'embarquer du contenu SVG dans une page web. L'élément `<svg>` permet d'intégrer avec les nœuds de SVG directement au sein du document HTML.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904
/DTD/svg10.dtd">
<svg width="300px" height="200px" xml:lang="fr"
xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <title>La kiwiParty, le web pulpeux</title>
    <desc>Un logo animé en SVG</desc>
</svg>
```

Dans les faits, le SVG n'a pas captivé l'imagination des développeurs comme `canvas` (que l'on verra un peu plus loin). Même si `canvas` est un petit nouveau, il bénéficie déjà d'une large compatibilité.

Le WHATWG recommande l'élément `canvas` pour le HTML5 alors que le SVG existe déjà.

Le SVG permet de créer des dessins vectoriels au sein des pages Web. Ces dessins peuvent être agrandis à l'infini.

Le logiciel **Inkscape** est connu pour permettre de dessiner des SVG.

Malheureusement, SVG n'est pas directement utilisable sur IE8 et inférieurs. Heureusement des personnes bien intentionnées ont réalisé des librairies pour combler ce manque. C'est notamment le cas du célèbre [Raphaël.js](#) qui rend praticable SVG sur IE6 et supérieur via JavaScript et l'utilisation de VML. D'autres librairies existent pour les navigateurs n'interprétant pas SMIL (ce qui est de plus en plus rare) notamment [FakeSmile.js](#).

Les nouvelles balises HTML

Canvas vs SVG

Depuis l'arrivée de HTML5 et de ses nouveaux éléments, les sites et applications web peuvent profiter de grandes avancées pour rendre ce média attractif : audio, vidéo et Canvas pour disposer de fonctionnalités de dessin en 2D et 3D dans le navigateur.

L'élément qui nous intéresse ici est `<canvas>`. Il s'agit d'un espace de pixels initialement transparents, armés de JavaScript pour réaliser un bon nombre de fonctions graphiques, partant du simple tracé de courbe.

Canvas étant un nouvel élément, il ne déroge pas à une structure HTML classique, avec des dimensions précisées par les attributs `width` et `height` (ou via CSS).

```
<canvas id="mon_canvas" width="350" height="350">
    Texte alternatif pour les navigateurs ne supportant
    pas Canvas.
</canvas>
```

À partir de ce moment, tout se passe du côté de JavaScript, qui va se servir de cet élément HTML pour accéder à la surface de dessin.

Pour ceci, deux fonctions sont appelées :

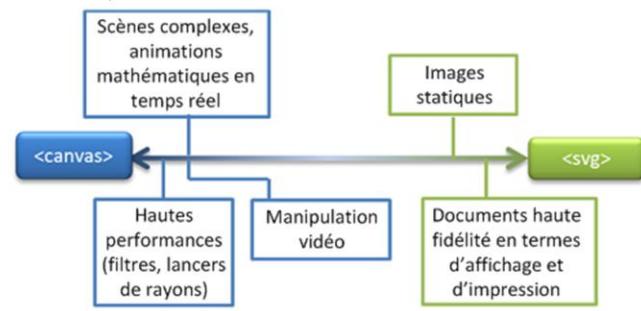
- `getElementById()` qui va permettre d'aller chercher et cibler l'élément `<canvas>` identifié par son attribut id unique (ici `mon_canvas`),
- `getContext()` de l'élément ainsi récupéré pour savoir dans quel contexte de dessin (2D ou 3D) le script va pouvoir agir, et de quelles fonctions il pourra disposer. Le contexte sera l'élément central de gestion de Canvas.

```
<script type="text/javascript">
    var c = document.getElementById("mon_canvas");
    var ctx = c.getContext("2d");
    // Le reste du script ici...
</script>
```

Les nouvelles balises HTML

Canvas vs SVG

Attention, il ne faut pas confondre **Canvas**, qui est une surface de dessin *bitmap* pilotable en JavaScript spécifique à HTML, et **SVG**, qui est un format de fichier pour le dessin vectoriel, décrit en XML. Bien souvent, on néglige de penser à SVG comme alternative à Flash, alors que celui-ci serait plus approprié (aussi conçu pour être vectoriel, animé, léger et modulaire).



Concernant l'utilisation de **Canvas** ou de **SVG**, il existe deux principaux facteurs de différentiation.

Une connaissance approfondie des API graphiques de bas niveau et un savoir limité des technologies Web, le choix sera probablement d'utiliser **Canvas**.

De plus, la performance est absolument essentielle pour les sites Web très fréquentés. Il peut être nécessaire de développer l'accessibilité, un style personnalisé ou des interactions utilisateur plus granulaires qui ne sont pas possibles avec Canvas.

Généralement, si la taille de l'écran augmente, la qualité de l'objet **Canvas** tend à se dégrader, car un nombre plus important de pixels doit être dessiné. En revanche, avec SVG, plus le nombre d'objets augmente à l'écran, plus les performances.

Les nouvelles balises HTML

Les nouvelles balises

<mark>

La balise `<mark>` permet de faire ressortir visuellement une portion de texte. L'extrait n'est pas forcément considéré comme important mais on veut qu'il se distingue bien du reste du texte. Cela peut être utile pour faire ressortir un texte pertinent après une recherche sur votre site par exemple

```
<p>Il y a quelques années, nous écrivions ceci :</p>
<blockquote>
  <p>A long terme, <mark> XHTML 2 remplacera HTML</mark>,
  et qu'il sera nécessaire pour les navigateurs et les
  intégrateurs de s'adapter à ce nouveau langage de balisage.
</p>
</blockquote>
<p>En réalité, nous avons vu que HTML5 a remplacé XHTML
2.</p>
```

L'élément `mark` n'accorde aucune importance à son contenu, sauf pour montrer que celui-ci présente temporairement un intérêt. La spécification nous dit que `mark` indique un morceau de texte marqué ou mis en surbrillance dans un document à des fins de référence, du fait de sa pertinence dans un autre contexte.

Vous auriez pu utiliser `em` ou `strong`, mais ce serait sémantiquement incorrect ; on ne veut pas donner une quelconque importance sur le terme recherché mais simplement trouver une façon de le mettre en surbrillance.

Les nouvelles balises HTML

Les nouvelles balises

<time>

La balise `<time>` formalise une information temporelle (date, heure), employée dans le contenu qu'elle encadre, par le biais de son attribut "datetime". Les dates saisies dans cet attribut doivent correspondre aux dates du calendrier grégorien.

```
<time datetime="1975-09-30">  
    30 septembre 1975  
</time>
```

L'élément arrime peut-être utiliser pour les dates, les heures, ou une combinaison des deux.

```
<time datetime="10:00">10 h</time>  
<time datetime="1975-09-30">30 septembre</time>  
<time datetime="1975-09-30T10:00">10 h le 30 septembre</time>
```

Propriétés	Définition	Valeurs
datetime	date au format datetime représentant l'information contenu dans l'élément	datetime
pubdate	Indique que la date portée par la balise <time> correspond à la dernière date de publication de l'élément parent article le plus proche ou du document lui-même sinon.	pubdate

Les nouvelles balises HTML

Les nouvelles balises

<figure>

```
<figure>
  
  <figcaption>Garfield, un petit chat tout mignon</figcaption>
</figure>
```

Selon le W3C, l'élément `<figure>` s'emploie combiné à `<figcaption>` pour structurer des illustrations, photos, diagrammes et portions de code entre autres. `<figcaption>` désignera la légende de la figure.

Notes :

Les nouvelles balises HTML

Les nouvelles balises

<meter>

L'élément meter peut être utilisé pour réaliser des mesures, du moment que ces mesures font partie d'une échelle avec un valeur minimum et une valeur maximum

```
<meter>
  9 pingouins sur 10
</meter>
```

Vous n'avez pas à exposer la valeur maximum si vous ne le souhaitez pas. Vous pouvez utiliser l'attribut **max** à la place :

```
<meter max="10"> 9 pingouins sur 10</meter>
```

Il existe un attribut **min** correspondant. Vous pouvez également jouer avec les attributs **high**, **low** et **optimum**. Si vous le souhaitez, vous pourrez même cacher la mesure elle-même dans un attribut **value**.

```
<meter low="-217" high="95" min="10" max="40" optimum="32" value="25">
Quelle chaleur en cette période de l'année.
</meter>
```

Les nouvelles balises HTML

Les nouveaux attributs

Draggable

Contenteditable

Hidden

De nouveaux attributs généraux

Le module de formulaire que nous verrons dans le chapitre suivant, comporte un certain nombre d'attributs innovants, mais il n'en a pas le monopole : les spécifications HTML 5 proposent quelques attributs polyvalents, que l'on peut affecter de manière générale à tous les éléments HTML. En l'état, ces attributs offrent un intérêt limité, mais se révèlent fort intéressants dès lors qu'ils sont manipulés via JavaScript.

Draggable

L'attribut **Draggable** confère à l' élément ciblé la possibilité d'être glissé-déposé via HTML5. La véritable dimension interactive nécessitera une couche de JavaScript afin de définir les modalités et conditions du comportement souhaité. Notez que la présence de cet attribut permet à elle seule de le cibler et le styler en CSS. En voici un exemple :

```
[draggable] {  
Outline: 1px dotted gray;  
Cursor : crosshair;  
}
```

Les nouvelles balises HTML

Les nouveaux attributs

```
<a title=""> <section title="">
    <abbr title="">
        <dfn title="cita"><span title="info">
            <aside title="Nota">
                ...angry titles "Menú principal"
            </aside>
        </dfn> <article title="Acerca de...">
            ...
        </article>
    </abbr>
</a>
```

Hidden

L'attribut **hidden** indique si un élément est suffisamment pertinent pour apparaître sur le document. Associé à la valeur **true**, **hidden** masque l'élément, les agents utilisateurs ne doivent plus en tenir compte.

Les spécifications W3C proposent l'exemple d'une page de connexion construite à l'aide de deux sections dans une boîte de login. Lorsque le visiteur est connecté, cette dernière zone devient masquée à l'aide de **hidden** :

```
<section id="login" hidden>
```

```
...</section>
```

Contenteditable

Reconnu depuis des lustres par Internet Explorer, cet attribut indique qu'une zone est éditable. L'utilisateur peut en changer le contenu et manipuler ainsi des chaînes de caractères.

```
<p contenteditable="true">Cliquez et modifiez ce texte</p>
```

Les nouvelles balises HTML

Les microformats



microformats

Les nouveaux éléments donnent davantage de sens à la structuration de la page et les microformats n'auront qu'à en tirer partie.

Plusieurs éléments vont permettre de se passer de certains Design Pattern ou certaines propriétés. Par exemple, l'élément `<time>` pourra remplacer Le [Design Pattern Datetime](#).

Autre petit désagrément livré avec HTML5, l'attribut `profile` a également été supprimé. On a déjà trouvé la parade en proposant un nouveau microformat : **rel-profile**. Celui-ci permet de définir le profil [XMDP](#) en le plaçant dans une balise `<a>` ou une balise `<link>` à l'endroit où il est nécessaire de définir les microformats.

Les nouvelles balises HTML

Les microformats



```
<!DOCTYPE html>
<html lang="fr">
<head>
    <title>Tony Starck</title>
</head>
<body>
    <section itemtype="http://microformats.org/profile/hcard" itemscope>
        <h1 itemprop="fn">Tony Stark</h1>
        <div itemprop="adr" itemscope>
            <p itemprop="street-address">56, rue du fer</p>
            <p itemprop="locality">Ironheim</p>
            <p itemprop="postal-code">67 999</p>
        </div>
        <a href="http://marvel.com/characters/bio/1009368/iron_man/" itemprop="url">Ma biographie</a>
    </section>
</body>
</html>
```

Microdata est un nouveau terme qui apparaît dans HTML5 . Ce sont des attributs supplémentaires permettant de préciser certains aspects sémantiques des éléments. Ils sont donc tout à fait dans la même ligne que les microformats puisqu'ils ajoutent des *microdonnées*.

En voici quelques-uns :

- L'attribut **itemscope** identifie et annonce des blocs qui seront balisés.
- L'attribut **itemprop** est une version spécifique aux noms de champs remplaçant l'attribut class.
- L'attribut **itemref** permet d'inclure des propriétés à un endroit quelconque de la page non descendant d'itemscope.
Il prend plusieurs id séparées par un espace (exemple : itemref="address phone" inclura les éléments ayant comme id address et phone).
- L'attribut **itemtype** permet de spécifier le type d'élément (exemple : itemtype="http://microformats.org/profile/hcard").

Les nouvelles balises HTML

Les formulaires



Dans le grand chantier de HTML5, une place d'honneur est réservée aux formulaires. Plutôt rustique jusque-là, les éléments de saisie de la dernière génération se voient doter de nouveaux types de champs, de nouvelles fonctions, voire de tests d'expressions régulières sur leur contenu textuel. Je vous en présente les plus prometteurs.

Email, url, tel, number, color...

HTML5 introduit de nouveaux types de champs de formulaires.

- **email** : le champ requiert un contenu au format d'adresse électronique
- **url** : le champ accueille des URL absolues
- **tel** : le champ est destiné aux numéros de téléphone
- **number** : le champ accepte uniquement les caractères numériques
- **color** : le champ est prévu pour les chaînes représentant une valeur de couleur

La syntaxe est similaire pour chacune des valeurs. Voici comment symboliser un champ d'adresse électronique :

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

Les nouvelles balises HTML

Les formulaires



Au cas par cas, certains attributs complémentaires peuvent élargir les fonctionnalités des nouveaux éléments de formulaire. Le type `number` accepte par exemple les attributs `min`, `max`, `step` et `value` :

```
<input type="number" min="0" max="10" step="2" value="6">
```

`type="number"` Indique qu'il s'agit d'un champ de type numérique

`min="0"` Spécifie la valeur minimale acceptable pour ce champ

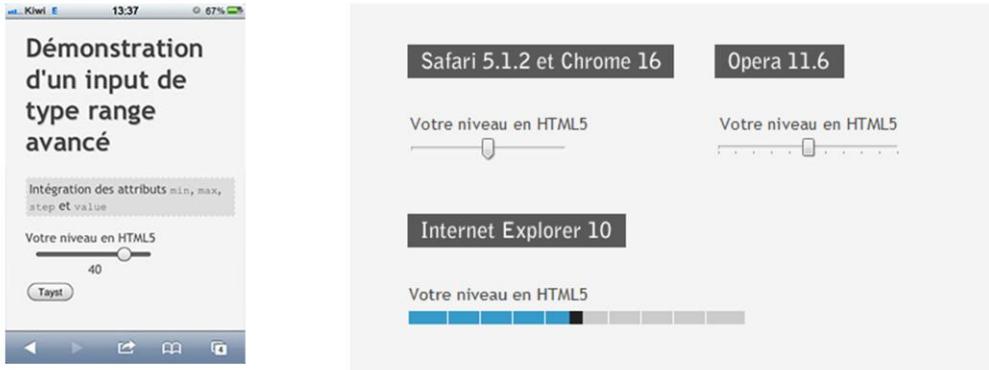
`max="10"` est le maximum acceptable

`step="2"` combiné à la valeur minimale indique que seuls les valeurs 0, 2, 4, 6, 8 et 10 sont acceptées, par pas de 2.

`value="6"` est la valeur par défaut au chargement de la page

Les nouvelles balises HTML

Les formulaires



range

Les champs de type `range` se présentent sous la forme d'un curseur défilant, ou slider.

```
<input type="range" min="0" max="50" value="0">
```

À l'instar du type `number`, un champ de type `range` accepte les propriétés complémentaires `min`, `max`, `step` et `value`.

```
<input type="range" min="0" max="10" value="6">
```

Les nouvelles balises HTML

Les formulaires



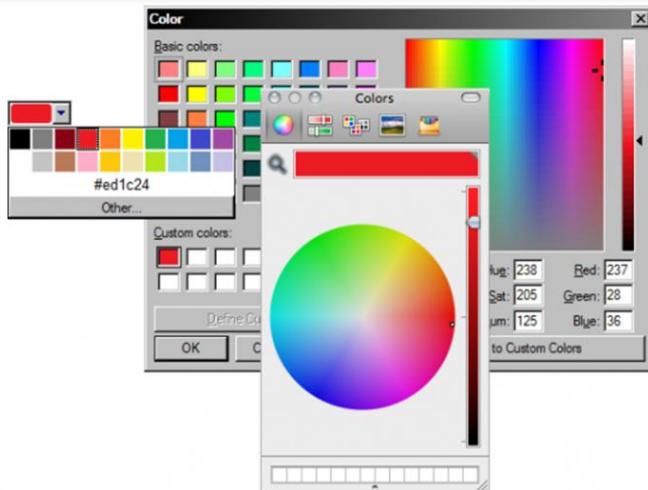
Date, datetime, month, week, time

Un certain nombre de nouveaux types de champ concernent les dates et l'heure.

- <input type="date"> affiche un contrôle de date (année, mois, jour).
- <input type="datetime"> affiche un contrôle de date associé à une heure
- <input type="month"> affiche un contrôle de mois
- <input type="week"> affiche un contrôle de numéro de semaine
- <input type="time"> affiche le contrôle d'heure

Les nouvelles balises HTML

Les formulaires



color

Ce type permet de transformer le champ pour sélectionner un code couleur dans une palette. La valeur attendue est une couleur au format hexadécimal (un dièse suivi de 6 caractères alpha-numériques compris entre A et F, et 0 et 9).

```
<input type="color" value="#fad345" name="textcolor">
```

Les attributs compatibles avec ce type de champ sont : `name`, `disabled`, `form`, `type`, `autocomplete`, `autofocus`, `list`, `value`.

Les nouvelles balises HTML

Les formulaires

Chrome

Effectuer une recherche

Kiwi

Safari

Effectuer une recherche

Kiwi

Search

Le type de champ `search`, comme son nom l'indique, désigne une zone de recherche au sein du formulaire.

```
<input type="search" name="recherche">
```

L'attribut complémentaire `results` affiche les dernières recherches récentes :

```
<input type="search" results="10" name="recherche">
```

Les nouvelles balises HTML

Les formulaires

L'élément `<output>` représente le résultat d'un calcul ou d'une *sortie* produite par le navigateur ou le serveur. Il accepte comme attribut : `for`, `name`, et `form`.

La valeur de l'attribut `for` de l'élément `<output>` reprend les valeurs des id de chaque champ `<input>` nécessaire au calcul.

L'attribut `form` de l'`<output>` reprend la valeur de l'id du formulaire qui comprend les différents champs servant au calcul.

Ce dernier point est important car l'élément `output` n'est pas forcément enfant d'un élément `form`.

```
<form action="scripts.php" method="post" id="tva_form"
onsubmit="ttc.value = ht.value * (1 + tva.value/100);
return false;">
<p>
    <label for="t_ht">Tarif HT</label>
    <input type="number" name="ht" id="t_ht"> €
</p>
<p>
    <label for="t_tva">TVA</label>
    <input type="number" name="tva" id="t_tva"
value="19.6"> %
</p>
<p>
    Prix TTC :
    <output for="t_ht t_tva" name="ht"
form="tva_form"></output> €
</p>
<input type="submit" value="Calculer">
</form>
```

Les nouvelles balises HTML

Les formulaires

L'attribut placeholder peut être placé sur les éléments :

- <input> : de type `text`, `search`, `password`, `url`, `tel`, `email`
- <textarea>

```
<input type="text" placeholder="Entrez un pseudo">
```

Placeholder

Placeholder est un attribut applicable à tout élément de formulaire. Sa fonction est d'afficher une chaîne de caractère temporaire au sein du champ. Lors du clic ou de l'activation du champ à l'aide du clavier, ce texte disparaît pour laisser la place au contenu à saisir.

Idéal par exemple pour indiquer qu'un champ est obligatoire ou pour remplir le rôle d'étiquette, placeholder s'applique à tous les champs de formulaires (<input>), même sur les éléments de champ type Password, ce qui n'était pas réalisable précédemment, puisque toute valeur dans un tel champ s'affichait avec des caractères masqués.

```
<input type="search" name="recherche" placeholder="Recherche">
```

Les nouvelles balises HTML

Les formulaires

HTML5 Form Attributes

The diagram shows a form titled "HTML5 Form Attributes". It contains three input fields: "Name:", "Website:", and "Email:". Below the "Name:" field, the word "autofocus" is written in blue, with a blue arrow pointing from it to the "Name:" field. A "Submit" button is located at the bottom of the form.

Name:	<input type="text"/>
Website:	<input type="text"/>
Email:	<input type="text"/>

Submit

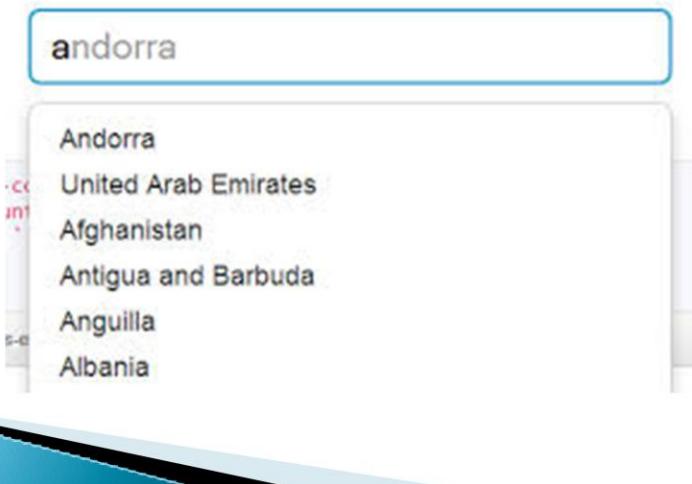
Autofocus

L'attribut autofocus confère la focalisation un champ déterminé dès le chargement de la page. Le site de Google s'avérerait être un cobaye parfait pour cette fonctionnalité : grâce à autofocus, le champ de recherche serait immédiatement opérationnel à l'affichage du site.

```
<input type="search" name="recherche" autofocus>
```

Les nouvelles balises HTML

Les formulaires



Autocomplete

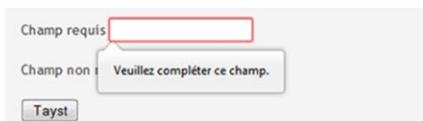
L'attribut `autocomplete` affiche une boîte contenant les dernières entrées de formulaires que vous avez saisies et qui ont été conservées en mémoire sur votre poste. La valeur par défaut est `on` (activé), mais il est possible de masquer ces termes à l'aide de la valeur `off`.

```
<input type="search" name="recherche" autocomplete="off">
```

Les nouvelles balises HTML

Les formulaires

L'attribut **required** permet de rendre obligatoire le remplissage d'un champ et bloquer la validation du formulaire si l'un des champs (concernés par cet attribut) n'a pas été renseigné.



```
<!-- sans valeur -->
<input type="text" required>

<!-- avec la valeur required -->
<input type="text" required="required">
```

Required

Les champs bénéficiant de l'attribut **required** doivent nécessairement être remplis lors de la soumission du formulaire, sans quoi la validation est refusée et le champ concerné mis en évidence.

Cette étape, à présent native dans HTML5, dispense de concevoir ces tests à l'aide de vérification via JavaScript ou un langage serveur.

Il est bien entendu possible de le cumuler avec d'autres attributs, comme **placeholder**, pour informer qu'un champ d'entrée est requis :

```
<input type="text" name="name" placeholder="champ requis" >
```

La reconnaissance de cet attribut par les navigateurs demeure relativement timide, mais, comme pour les autres attributs de cette nouvelle fournée HTML5, rien ne vous empêche dès aujourd'hui d'appliquer des styles distincts à cet élément :

```
[required] {border: 1px solid orange}
```

Les nouvelles balises HTML

Les formats multimédia



Nous sommes entourés de technologies multimédias fantastiques, qui nous permettent d'écouter à tout instant une musique sur notre baladeur MP3, ou encore de visionner un film sur notre mini PC, voir notre téléphone !

Dans le monde du Web, malgré des capacités de stockage et d'échanges chaque jour améliorées, la situation semble bloquée et complexe : point de salut hors technologies propriétaires ou plug-ins tels que Adobe Flash ou Apple QuickTime.
si

HTML5 change la donne en définissant trois nouveaux éléments destinés à gérer nativement des aspects visuels, sonores et graphiques de l'ère numérique moderne : `<audio>` et `<video>`.

Les nouvelles balises HTML

Les formats multimédia

À l'instar de la propriété font-family, l'élément `<audio>` offre la possibilité de proposer plusieurs types de formats au navigateur ; celui-ci va parcourir chacun des fichiers proposés et retenir le premier reconnu, sans changer les autres, comme le montre cet exemple :

```
<audio>
  <source src=la-musique.mp3>
  <source src=la-musique.ogg>
</audio>
```

`<audio>`

Élément audio permet de lecture d'un fichier audio, au format classique, sans recourir à des outils propriétaires. Sa syntaxe est aussi courte qu'efficace :

```
<audio src="la-musique.mp3"></audio>
```

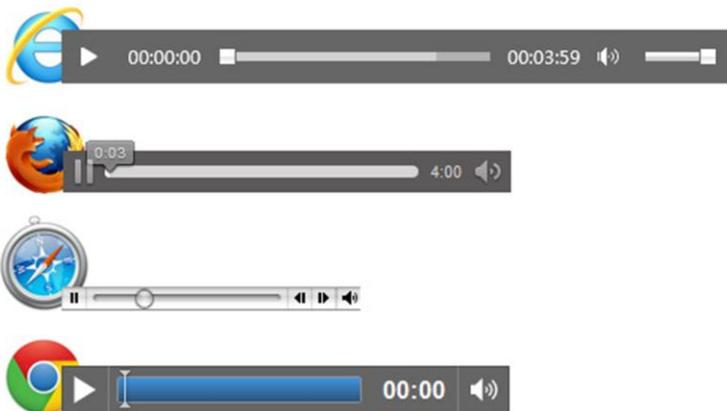
Quelques attributs peuvent se greffer à la propriété `<audio>`. Il s'agit, entre autres, de `autoplay` (lecture automatique) et `controls` (affichage des boutons de contrôle de lecture).

Voici comment proposer sur votre page Web un morceau musical lancé dès le chargement et muni de sa barre de contrôle :

```
<audio id="player" src="la-musique.mp3" autoplay controls></audio>
```

Les nouvelles balises HTML

Les formats multimédia



C'est un peu trop simple. Vous voudrez probablement être plus spécifique quant au comportement du fichier sonore.

Grâce à l'attribut `autoplay`, le fichier audio démarrera automatiquement.

```
<audio src="la-musique.mp3" autoplay></audio>
```

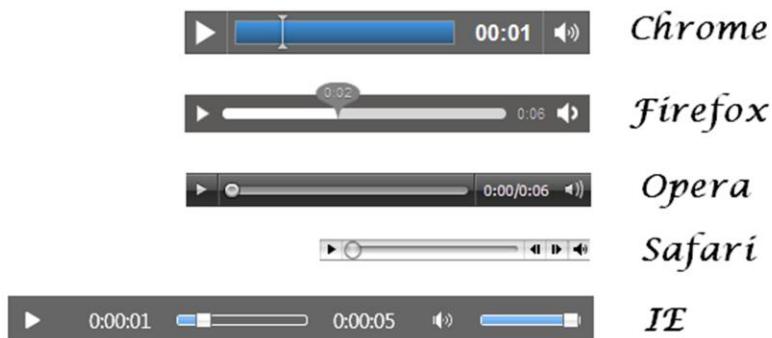
Remarquez que l'attribut `auto Play` ne prend pas de valeur. C'est un attribut booléen.

Si l'idée d'imposer votre fichier audio au démarrage ne vous paraît pas assez diabolique, vous pouvez infliger encore plus de souffrance en le jouant en boucle. Un autre attribut booléen appelé `loop` :

```
<audio src="la-musique.mp3" autoplay loop></audio>
```

Les nouvelles balises HTML

Les formats multimédia



L'élément audio peut représenter une bénédiction comme une malédiction. Il peut être judicieux de laisser à l'utilisateur le contrôle de la lecture du fichier audio. C'est ce qu'on fait avec l'attribut booléen `controls` :

```
<audio src="la-musique.mp3" controls></audio>
```

La présence de l'attribut `controls` invite le navigateur à afficher des contrôles natifs pour lire et mettre l'audio en pause, ainsi qu'ajuster le volume.

L'attribut `autobuffer` permettait aux auteurs de dire aux navigateurs que, même si le fichier audio ne devait pas démarrer automatiquement, il allait sans doute être lu à un moment ou à un autre, et que le navigateur devait donc commencer à le charger en arrière-plan. Cet attribut a maintenant été remplacé par `preload`. Ce n'est pas un attribut booléen. Il peut prendre trois valeurs différentes : `none`, `auto` et `metadata`. Avec `preload="none"`, vous pouvez maintenant demander explicitement au navigateur de ne pas précharger l'audio :

```
<audio src="la-musique.mp3" controls preload></audio>
```

Les nouvelles balises HTML

Les formats multimédia

Afin de faciliter sa reconnaissance par différents agents utilisateurs, l'élément `<video>` prévoit de définir plusieurs types de formats de fichiers :

```
<video>
  <source src="la-video.mp4" type="video/mp4">
  <source src="la-video.webm" type="video/webm">
  <source src="la-video.ogv" type="video/ogv">
</video>
```

`<video>`

HTML5 introduit l'élément `<video>` pour insérer des séquences vidéo dans une page Web, ce qui offre un alternative aux plug-ins propriétaire d'Apple (QuickTime) ou d'Adobe (flash) pour les plates-formes ne reconnaissant pas (iPhone, iPod, iPad...).

La syntaxe de base de l'élément vidéo est très simple :

```
<video src="la-video.mp4" ></video>
```

Des attributs permettent d'apporter des informations complémentaires :

- **controls** affiche les possibilités de contrôle de la vidéo (avance, pause, stop...)
- **autoplay** lance la lecture automatiquement dès le chargement de la page
- **preload** spécifie au navigateur de débuter le téléchargement de vidéo tout de suite, en anticipant sur le fait que l'utilisateur verra la vidéo

À l'heure actuelle, la difficulté d'implémentation et de standardisation de l'élément `<video>` tient au fait qu'un certain nombre de formats se livrent une bataille rangée (**mp4**, **ogv** et **webm**).

Les nouvelles balises HTML

Les formats multimédia



L'élément `video` fonctionne tout comme l'élément `audio`. Il comprend les attributs optionnels `autoplay`, `loop` et `preload`. On peut spécifier l'emplacement de la vidéo avec l'attribut `src` de l'élément `video`, ou remplaçant des éléments `source` entre les balises `<video>`. Vous pouvez laisser un navigateur le soin de fournir une interface utilisateur avec l'attribut `controls`, ou vous pouvez scripter vos propres contrôles.

La principale différence entre le contenu audio et le contenu vidéo, c'est que les films, par nature, prennent plus de place sur l'écran. Vous voudrez donc sans doute préciser des dimensions :

```
<video src="film.mp4" controls width="360" height="240"></video>
```

Vous pouvez choisir une image représentative de la vidéo et demander au navigateur de l'afficher à l'aide de l'attribut `poster`:

```
<video src="film.mp4" controls width="360" height="240" poster="apercu.jpg"></video>
```

Les nouvelles balises HTML

Les formats multimédia



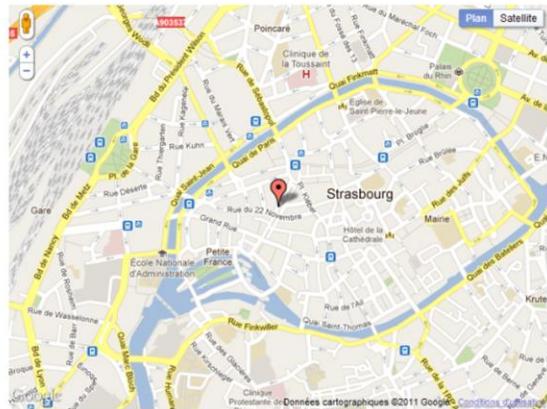
Une large partie des spécifications tentaculaires de HTML5 est allouée aux applications Web. Ces extensions du langage comportent nativement de nouvelles passerelles de communication vers les protocoles habituels, mais également des fonctionnalités de stockage, de géolocalisation, de travail hors ligne, de messagerie et de traitements de fichiers.

Ces applications, généralement encore à l'état de brouillon, sont ou seront en grande majorité pilotées par JavaScript. En clair, JavaScript, que l'on a cru enterrer il y a quelques années, et quelque peu remis au goût du jour grâce à Ajax et aux bibliothèques telles que Jquery, va dorénavant s'imposer comme langage de tout premier ordre grâce à HTML5.

Les nouvelles balises HTML

Les formats multimédia

Geolocalisation



Geolocalisation

L'API [geolocalisation](#) introduite par le W3C permet aux pages Web d'interroger le navigateur sur la position de l'utilisateur, de manière bien plus précise et fiable que les tests de positionnement basé sur l'interprétation de l'adresse IP de la machine.

L'application récupère au travers de l'objet JavaScript [navigator.geolocation](#) les coordonnées complètes de la position du poste de l'utilisateur, exprimé en [longitude](#), [latitude](#) et [altitude](#).

La géolocalisation repose sur deux méthodes :

- ✓ [getCurrentPosition](#) permettant un ciblage ponctuel
- ✓ [watchPosition](#) pour un suivi continu

Les nouvelles balises HTML

Les formats multimédia

L'attribut **draggable** rend un élément déplaçable



Drag and Drop

L'attribut **draggable** rend un élément déplaçable. Tout son cheminement est traçable via JavaScript: le clic sur l'élément, son trajet, ainsi que l'endroit où il est déposé sur la page à l'aide de l'événement **ondrop**.

Des objets indépendants du document peuvent également être déposés sur la page, par exemple un fichier de votre poste de travail que vous pouvez télécharger sur l'application en l'y faisant tout simplement glisser.

Les fonctionnalités de ce genre, déjà reconnu par l'ensemble des navigateurs y compris Internet Explorer 8 vont faciliter les échanges avec JavaScript au point de simplifier considérablement nos habitudes.

Les nouvelles balises HTML

Les formats multimédia

HTML 5 introduit la notion de stockage des données via ses API SessionStorage et LocalStorage



Web Storage

HTML 5 introduit la notion de stockage des données via ses API SessionStorage et LocalStorage :

- SessionStorage s'appuie sur JSON, qui autorise le stockage d'informations relatives à la navigation de l'internaute dans toutes les pages d'un site et ce, pour toute la durée de la session.
- LocalStorage permet, quant à lui, de conserver localement des données d'une session à l'autre : votre navigateur conserve ainsi sur votre machine ces données qui vous sont personnelles. Vous pouvez les réutiliser à tout moment, en toute sécurité.

Une application telle que LocalStorage permet de s'affranchir des techniques actuelles de cookies JavaScript vos sessions PHP pour le stockage des informations de l'utilisateur.

Les nouvelles balises HTML

Les formats multimédia

l'API Offline Web Application se charge de la mise en cache de constantes de vos activités



Applications Web hors-ligne

Complémentaire fonction de stockage Web Storage et comptant parmi les applications les plus prometteuses de notre ère du Web nomade, l'API Offline Web Application se charge de la mise en cache de constantes de vos activités, tant et si bien qu'il devient possible de continuer à faire tourner les fonctionnalités tout en étant déconnecté d'Internet.

Que vous soyez hors ligne ne vous empêche plus de rédiger des courriers de messagerie, de planifier votre agenda, ni de rédiger des documents ou présentations destinées à être partagés avec d'autres personnes : dès lors que votre terminal trouvera un point de connexion Internet, vos travaux en cours se synchroniseront ou seront expédiés automatiquement si le développeur le prévoit.

Les sélecteurs CSS 3



- Rappel sur la syntaxe : les sélecteurs, les règles
- Sélecteurs de répétition nthchild(even)
- Sélecteurs de cible target
- Sélecteur frère antérieur ~
- Sélecteurs d'enfants unique only-child

Les sélecteurs CSS 3

Rappel sur la syntaxe

Le style peut être indiqué :

❖ Dans chaque balise HTML en intégrant les attributs de style
`<p style="color:blue">`

❖ Dans l'en-tête `<head>` du HTML en précisant le style entre les balises `<style>`
`<style type="text/css">`
`p{color:blue}`
`</style>`

❖ Dans un fichier à part (lien intégré dans l'en-tête `<head>` du HTML)
`<link rel="stylesheet" type="text/css" href="style.css">`

La dernière méthode est celle à adopter. Elle contribuera à faciliter la maintenance de notre site Web.

Les fichiers comportera les règles de mise en forme de nos différentes pages et par convention aura l'extension **.CSS**

Exemple :

Notre fichier se nommera **style.css** et mentionnera la structure suivante :

```
p{  
    color: red;  
    background-color: #FFFFFF;  
}
```

Les sélecteurs CSS 3

Rappel sur la syntaxe

Remarquons que :

La balise impactée est indiquée en toutes lettres sans les chevrons <>, on l'appelle le sélecteur

Les règles sont entourées par des accolades {}

Chaque règle à une syntaxe du type propriété : valeur ;

La syntaxe générale est :

```
Sélecteur {  
    propriétés: valeur;  
}
```

Les sélecteurs CSS 3

Le CSS3 enrichi les sélecteurs

Nouveautés CSS3

Bien que les CSS2 permettent un choix assez riche de sélecteurs, certaines carences ont été comblées en CSS3 :

- ❖ des critères de sélection supplémentaires ont été apportés sur les attributs,
- ❖ des nouvelles combinaisons d'adjacence ont été définies,
- ❖ de nombreuses pseudo-classes ont été ajoutées.

Beaucoup de traitement de rendus peuvent être maintenant réalisés par le style.

Les sélecteurs CSS 3

Le CSS3 enrichi les sélecteurs

Univers de CSS3 ne se limite pas à un ensemble de nouvelles propriétés : en effet, s'il est gratifiant d'appliquer des styles révolutionnaires à un élément, il est tout aussi utile de pouvoir les cibler au mieux. C'est dans cette optique que de nombreux nouveau secteur et pseudo élément font leur apparition dans cette dernière mouture de CSS.

Selecteur adjacent général

Le sélecteur d'adjacence exister depuis CSS 2.1. Il s'écrit de la sorte **E + F** et cible le frère directement adjacent (ici l'élément F frère immédiatement successeur de E). La norme CSS3 élargit ce principe en introduisant un sélecteur adjacent général s'écrivant ainsi : **E ~ F**. La différence avec le sélecteur adjacent classique est que la version proposée par CSS3 ne se limite pas à la proximité directe : d'autres éléments frères peuvent se placer entre E et F.

Selecteur d'attribut

Tu arrêtes Stac

Vous connaissiez déjà le secteur d'attributs CSS2 permettant de désigner un élément en silence son attribut (**[attribut]**) ou la valeur de son attribut (**[attribut="valeur"]**). Là encore, trois nouvelles variantes de ses lecteurs d'attributs sont introduits dans la norme CSS3.

- ❖ **[attribut^="clafoutis"]** caractérise un attribut dont la valeur commence exactement par la chaîne clafoutis,
- ❖ **[attribut \$=clafoutis"]** caractérise l'attribut dont la valeur finie exactement par le suffixe clafoutis,
- ❖ **[attribut *= "clafoutis"]** désigne un attribut dans la valeur contient au moins une fois la sous-chaîne clafoutis.

Les sélecteurs CSS 3

Le CSS3 enrichi les sélecteurs

Sélecteurs par attribut

Les CSS2 permettent d'attribuer un style :

- ❖ en fonction de la présence d'un attribut `tag[attr]`,
- ❖ en fonction de la présence d'un attribut à une valeur `tag[attr="val"]`,
- ❖ en fonction de la présence d'un attribut dans une liste de valeur `tag[attr~="val"]`.

Les CSS3 permettent en plus d'attribuer un style :

- ❖ si un attribut commence par une expression `tag[attr^="deb"]`,
- ❖ si un attribut se termine par une expression `tag[attr$="fin"]`,
- ❖ si un attribut contient une expression `tag[attr*="milieu"]`.

Exemple :

Règles CSS

```
[title*="toto"] {  
background-color:red;  
color:black;  
}  
[title="toto"] {  
background-color:green;  
}
```

Appliqué au HTML

```
<h1 title="titi">Le titre</h1>  
<h2 title="titototi">le sous-titre</h2>  
<p title="toto">le contenu du paragraphe</p>
```

- ❖ affectera tous les styles sur le contenu de la balise `<p>`,
- ❖ affectera le premier style sur le contenu des balises `<p>` et `<h2>`,
- ❖ n'affectera pas le style sur le contenu de la balise `<h1>`.

Dans cet exemple la couleur de fond du paragraphe sera celle indiquée par la dernière règle de style (green), car la priorité des deux règles est équivalente, et dans ce cas, la dernière règle rencontrée est la règle appliquée.

Les sélecteurs CSS 3

Le CSS3 enrichi les sélecteurs

Sélecteurs de répétition

La pseudo-classe `tag:nth-child(p)` permet la sélection de la répétition en partant du début.

La pseudo-classe `tag:nth-last-child(p)` permet la sélection de la répétition en partant de la fin.

La pseudo-classe `tag:nth-of-type(p)` permet la sélection de la répétition d'éléments de même type en partant du début.

La pseudo-classe `tag:nth-of-last-type(p)` permet la sélection de la répétition d'éléments de même type en partant de la fin.

Dans tous ces cas le paramètre `p` correspond au contrôle de la répétition.

Exemple de répétition de couleur sur un tableau :

Règle CSS

```
tr:nth-child(6n+1), tr:nth-child(6n+2), tr:nth-child(6n+3) {  
background-color:red  
}
```

Appliqué au HTML

```
<table>  
<tr><td>1,1</td><td>1,2</td></tr>  
<tr><td>2,1</td><td>2,2</td></tr>  
<tr><td>3,1</td><td>3,2</td></tr>  
<tr><td>4,1</td><td>4,2</td></tr>  
<tr><td>5,1</td><td>5,2</td></tr>  
<tr><td>6,1</td><td>6,2</td></tr>  
<tr><td>7,1</td><td>7,2</td></tr>  
</table>
```

Rendra les 3 premières lignes et la dernière sur fond rouge.

Les sélecteurs CSS 3

Le CSS3 enrichi les sélecteurs

Sélecteurs de répétition

Les CSS3 permettent de localiser des éléments en fonction d'un critère de répétition. Cette répétition peut être contrôlée :

- ❖ en période,
- ❖ en phase,
- ❖ en type d'éléments.

Ce contrôle se fait par un paramètre **p** passé en argument au sélecteur.

Le paramètre **p** agit ainsi :

- ❖ **even** adressera les valeurs paires,
- ❖ **odd** adressera les valeurs impaires,
- ❖ **3n** adressera les valeurs 3, 6, 9 ... ,
- ❖ **6n+1** adressera les valeurs 1, 7, 13 ... ,
- ❖ **0n+7** adressera uniquement la valeur 7,
- ❖ **-n+2** adressera les valeurs 1 et 2.

Il faut voir **n** comme une valeur s'incrémentant indéfiniment avec une période indiquée par la valeur en préfixe. Ainsi **6n+3** n'incrémentera **n** que pour les valeurs **6, 12, 18...** tandis que **3** sélectionnera la phase, c'est à dire les éléments **3, 9, 15...**

La phase par défaut est égale à 0.

Les sélecteurs CSS 3

Le CSS3 enrichi les sélecteurs

Sélecteurs de cible target

tag:target permet la sélection d'un élément cible d'un lien.

Seul l'élément cible sera impacté par le style.

Exemple :

Règle CSS

```
*:target {  
background-color:green  
}
```

Appliqué au HTML

```
<h1>test sur <strong id="la_cible">les sélecteurs</strong> de <q>target</q>.</h1>  
<p>voici un lien sur <a href="#la_cible">la cible</a></p>
```

En chargeant la page, le titre sera sur fond incolore (donc blanc). En cliquant sur le lien, une partie du titre s'affichera en fond vert (celle dont l'id correspond à la cible).

Les sélecteurs CSS 3

Le CSS3 enrichi les sélecteurs

Sélecteurs d'adjacence

Les CSS2 permettent d'attribuer un style à un élément frère **adjacent** et suivant d'un élément donné : *tag1+tag2*

Les CSS3 permettent également de sélectionner tout élément frère suivant un élément donné : *tag1~tag2*

Exemple :

Règles CSS

```
li.toto~li {  
background-color:red  
}
```

Appliqué au HTML :

```
<ul>  
  <li>bonjour</li>  
  <li class="toto">S'il vous plaît</li>  
  <li>Merci</li>  
  <li>Au revoir</li>  
</ul>
```

❖ affectera les éléments situés après la seconde ligne de la liste.

Les sélecteurs CSS 3

Le CSS3 enrichi les sélecteurs

Sélecteurs d'enfants aux limites

Les CSS2 permettent de localiser un élément en tant que premier enfant d'un élément : [*tag:first-child*](#).

Les CSS3 permettent également le dernier enfant d'un élément : [*tag:last-child*](#).

Les CSS3 permettent aussi :

- ❖ de localiser un élément en tant que premier enfant du même type : [*tag:first-of-type*](#),
- ❖ de localiser un élément en tant que dernier enfant du même type : [*tag:last-of-type*](#).

Les sélecteurs CSS 3

Le CSS3 enrichi les sélecteurs

Sélecteurs d'enfant unique

tag:only-child permet la sélection d'un élément enfant unique.

tag:only-of-type permet la sélection d'un élément enfant unique du même type.

tag:empty permet la sélection d'éléments vides.

Exemple :

Règle CSS

```
em:only-child {  
background-color:red  
}  
strong:only-of-type {  
background-color:green  
}
```

Appliqué au HTML

```
<h1>test sur <strong>les sélecteurs</strong> d'<q>enfants</q> uniques.</h1>  
<p>voici un <em>texte qui devrait être sur fond rouge<em></p>
```

Rendra une partie du titre sur fond vert, et une partie du paragraphe sur fond rouge.

Les sélecteurs CSS 3

Le CSS3 enrichi les sélecteurs

[Les pseudo classes](#)



Les spécifications CSS3 sont très riches de pseudo-éléments et pseudo-classes inédites qui apportent une souplesse supplémentaire dans la sélection des éléments selon leur contexte. Ainsi il devient possible de cibler uniquement le dernier enfant d'un conteneur, le premier enfant d'un certain type, uniquement les éléments pairs ou impairs, ceux qui ne respectent pas une condition, etc.

Signalons que depuis CSS3, une convention d'écriture à été proposé par le W3C pour distinguer les pseudo-classes des pseudo-éléments. Ainsi ces derniers s'écrivent dorénavant à l'aide d'un double deux-points (`::first-line`, `::first-child`, `::after`, `::before`), tout en autorisant une rétrocompatibilité avec l'écriture CSS2.

Les sélecteurs CSS 3

Le CSS3 enrichi les sélecteurs



La pseudo-classe `:lang` permet de cibler un élément dont la langue correspond à une certaine valeur indiquée entre parenthèses.

Le sélecteur `:empty` fait référence à un élément vide de tout contenu.

Comme `:empty`, la pseudo-classe `:root` est reconnue sur plusieurs navigateurs. Cependant son intérêt est extrêmement limité : en HTML, ce secteur désigne uniquement, et toujours, l'élément racine `<html>`. La seule différence avec le sélecteur `html` est que le poids de `:root` est supérieur.

La pseudo-classe de négation `:not` cible un élément qui ne correspond pas au secteur déterminé entre parenthèses. Par exemple, `:not(a)` désignera tous les éléments de la page à l'exception des liens et `a:not(:visited)` pointera tous les liens sauf ceux déjà visités.

Les couleurs

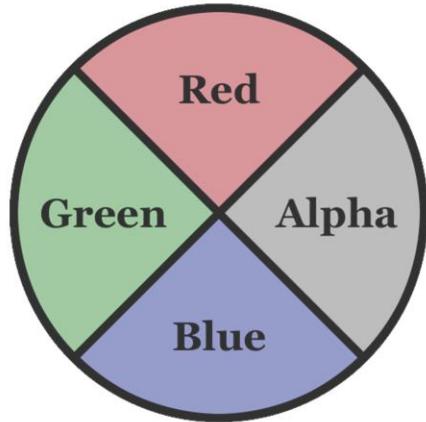
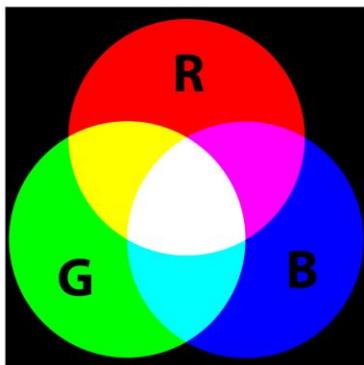


- Rappel sur les déclarations rgb et rgba
- Les modèles hsl et hsla
- La propriété Opacity



Les couleurs

Rappel sur les déclarations `rgb` et `rgba`



Bien qu'il ne s'agisse pas de propriétés, il est opportun d'évoquer la notation CSS3 RGBa qui introduit la notion de transparence pour toutes les propriétés relatives aux couleurs : arrière-plan, bordure, couleur des textes, etc.

Vous connaissez sans doute déjà la notation habituelle RGB qui se décompose ainsi :

- R(red) : composante rouge en pourcentage (de 0 à 100%) ou en valeur (0 à 255);
- G(green) : composante verte en pourcentage (de 0 à 100%) ou en valeur (0 à 255);
- B(blue) : composante bleue en pourcentage (de 0 à 100%) ou en valeur (0 à 255);

L'exemple suivant a fait une couleur bleue aux éléments de classe `bloc` :

```
.bloc {background-color:rgb(0,0,255);}
```

CSS3 ajoute une 4^{ème} composante (`a`), correspondant au degré de transparence et dont la valeur est fixée entre zéro et un

```
.bloc {background-color:rgba(0,0,255,0.5);} /* fond bleu à moitié transparent */
```

Les couleurs

Les modèles hsl et hsla



La notation HSLa traduit les couleurs via les canaux de teinte, de saturation, de luminosité et de transparence :

H (Hue) : nuances de couleur entre 0 et 360 (0 correspond au rouge, 120 au vert et 240 au bleu);

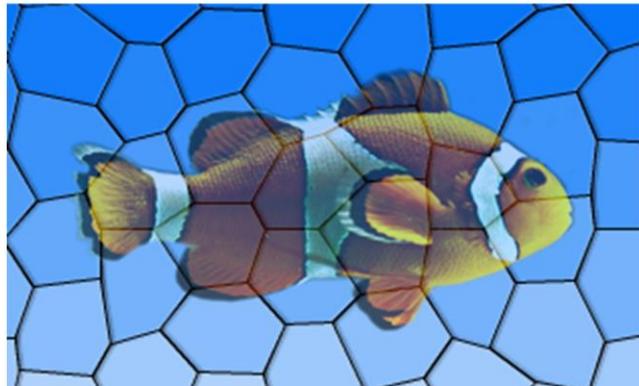
S(Saturation) : valeur en pourcentage (0 est complètement terne, 100 % est au maximum de la vibrance) ;

L(Luminosité) : valeur de 0 (sombre) à 100 % (Clair) ;

A(alpha) : degré de transparence, valeur de 0 à 1.

Les couleurs

La propriété Opacity



Comme son nom l'indique, opacity est une propriété agissant sur l'opacité d'un élément, c'est-à-dire son degré de transparence. Les valeurs acceptées par cette propriété sont situées entre 0 et 1. La valeur 0 dans l'élément et ses descendants entièrement invisible, tandis qu'avec la valeur par défaut de 1, il est totalement opaque. Il est possible d'appliquer cette propriété à tous les éléments HTML, qu'ils soient de type bloc ou non.

Voici un exemple d'élément à 60 % d'opacité :

```
.bloc {  
  opacity: 0.6;  
}
```

Un inconvénient fréquemment rencontré est que opacity s'applique à l'élément dans son intégralité, y compris tous ses descendants, ce qui n'est pas sans poser des problèmes si l'on désire que l'un des enfants ne bénéficie pas de ce lourd héritage.

Les couleurs

Les dégradés en CSS

Dégradé type linéaire

```
background : linear-gradient(x y, couleur1, couleur2,  
couleurN)
```

Dégradé type radial

```
background : radial-gradient(forme at x y, couleur1,  
..., couleurN)
```



De nouvelles propriétés permettent de réaliser et gérer des arrière-plans de teintes dégradées. Il s'agit de [linear-gradient](#) pour les dégradés linéaires et [radial-gradient](#) pour les dégradés radiaux.

Syntaxe pour linear-gradient

Les dégradés linéaires empruntent la valeur-fonction [linear-gradient\(\)](#) de la propriété [background-image](#) (ou [background](#) en raccourci). Il convient de préciser au minimum le point d'arrivée du dégradé ainsi que deux couleurs (ou plus).

Valeurs possibles

x et/ou y définissent la direction du dégradé. Exemple : "to right" ou "to left bottom",
couleur : le nom, la valeur hexadécimal ou la valeur RGBa

Syntaxe pour radial-gradient

Les dégradés radiaux sont réalisables à l'aide de la valeur-fonction [radial-gradient\(\)](#).
Elle nécessite d'indiquer la forme du dégradé : circulaire ([circle](#)) ou elliptique ([ellipse](#))
ainsi que son point de départ.

Valeurs possibles

x et/ou y définissent le point d'origine du dégradé radial,
couleur : le nom, la valeur hexadécimal ou la valeur RGBa

La mise en forme du texte



- Support des polices distantes @font-face
- Mise en forme du texte
- Gestion des débordements
- Gestion des enchaînements
- Effets de couleurs et d'ombre sur le texte
- Les modes multi-colonnes

La mise en forme du texte

Support des polices distantes @font-face

@font-face permet de définir une police qui sera téléchargée par le navigateur afin d'être utilisée pour le rendu typographique des éléments.

Déclaration de base pour @font-face

```
@font-face{  
    font-family: maPolice;  
    src: url('chemin/maPolice.ttf') format('ttf');  
}
```



@font-face

Utiliser des fontes originales sur le Web, cette possibilité a été pensée depuis longtemps, c'est le rôle de la règle @font-face. Elle est apparue au sein de CSS2 et était même implémentée dans Internet Explorer 4. Malheureusement, un format propriétaire unique, une interprétation différente des spécifications ainsi que le problème de gestion de propriété intellectuelle l'ont fait tomber aux oubliettes. C'est pourquoi elle a été retirée de CSS2.1. Mais, la voilà de retour en CSS3 et les navigateurs ont pris le train en marche !

Cette déclaration, à ajouter directement au sein de nos CSS, est défini par une @règle. Elle est composée de deux propriétés au minimum :

- ❖ **font-family** permet de nommer la police. C'est le nom qui sera utilisé ensuite pour l'appliquer aux textes.
- ❖ **src** permet de spécifier l'adresse de la fonte à télécharger. Le format de la police est également précisé, à l'aide de la fonction du même nom.

La mise en forme du texte

Support des polices distantes @font-face

Utilisation d'une police définie avec @font-face

```
h1{  
    font-family: maPolice, sans-serif;  
}
```

Dans ce cas simple, « maPolice » est appliquée sur le h1. Le type de famille est générique sans-serif est tout de même ajouté en fin de chaîne afin de gérer le cas où la fonte ne pourrait pas être affichée (problème de téléchargement, lenteur du réseau, navigateur ne prenant en charge [@font-face](#)).

La mise en forme du texte

Support des polices distantes @font-face

Il existe actuellement cinq formats de fontes disponibles pour le Web. Ces formats sont :

- ❖SVG/SVGZ (*Scalable Vector Graphics*)
- ❖TTF (*True Type Font*)
- ❖OTF (*Open Type Font*)
- ❖EOT (*Embedded Open Type*)
- ❖WOFF (*Web Open Font Format*)

Aujourd’hui, la majeure partie des navigateurs Web implémentent @font-face. Cependant, si nous reprenons notre exemple initial, la police utilisée est au format TTF, et ce format n'est pas interprété partout, il faut donc proposer des formats de polices différents pour un support au sein de chaque navigateur.

Le format WOFF, qui est le plus récent en date, est un format ouvert, créé à l'initiative d'Opéra, de Microsoft et de Mozilla, et a été spécialement pensé et conçu pour le Web. Il a donc vocation à être utilisé massivement, son support devenant de plus en plus important. WOFF est à présent supporté dans la version la plus récente de chaque navigateur.

La mise en forme du texte

Support des polices distantes @font-face

Déclaration pour @font-face, avec plusieurs formats de police

```
@font-face{  
    font-family: maPolice;  
    src: ('chemin/maPolice.woff') format('woff'),  
    src: ('chemin/maPolice.ttf') format('ttf'),  
    src: ('chemin/maPolice.svg#ancre') format('svg').  
}
```

Dans cet exemple, nous remarquons que toutes les versions de navigateurs trouveront une police qui leur convient.

Pour un support maximal de [@font-face](#), notre déclaration précédente doit alors s'enrichir pour faire appel aux différents formats vu précédemment. Pour cela, nous séparons le format de fonte par une virgule (,) au sein de la règle [src](#) en utilisant le principe classique de CSS.

De plus, il est préférable de faire appel en premier lieu au format WOFF, puisque lors du parcours de cette règle par le navigateur, si le format de la première police trouvée est reconnu, la fonte est téléchargée, et le reste de la déclaration de la règle [src](#) abandonnée. Le format WOFF est le format recommandé pour [@font-face](#).

La mise en forme du texte

Support des polices distantes @font-face



[Google Web fonts](#) nous promet des centaines de polices, libres de droit et optimisées pour le Web. Le choix d'une, ou de plusieurs polices, est décomposé en trois étapes :

- ❖ Choisir ses polices
- ❖ Comparer et vérifier le contenu
- ❖ Obtenir le code

Font squirrel propose également des centaines de polices, mais contrairement à [Google Web fonts](#), elles se trouvent sous forme de kit à télécharger. Ces kits sont multiples et sont classés par catégories de polices.

La mise en forme du texte

Mise en forme du texte

Les propriétés de police



Les nouveaux modules CSS3, encore à l'état de réflexion, étendent le nombre de propriétés liées à la gestion de la typographie. L'objectif premier étant d'avoir un plus grand contrôle concernant le rendu de la typographie elle-même, les lettres, les mots, les espaces blancs, les retours à la ligne, les alignements, les débordements et en règle générale, tout ce qui se rapporte aux textes.

La mise en forme du texte

Mise en forme du texte

La propriété font-stretch permet, comme son nom l'indique, de condenser ou à l'inverse, d'étirer les caractères d'une police. Le W3C définit neuf mots-clés allant du plus condensé au plus étiré :

- ❖ Ultra-condensed
- ❖ Extra-condensed
- ❖ Condensed
- ❖ Semi-condensed
- ❖ Normal
- ❖ Semi-expanded
- ❖ Expanded
- ❖ Extra-expanded
- ❖ Ultra-expanded

Ces valeurs sont applicables uniquement sur une famille de polices dont les styles condensés ou étirés existent. En clair, si aucune variante d'étirement n'existe au sein de la fonte, aucun de ces mots-clés ne fonctionnera. En revanche, prenons l'exemple d'une famille de police qui aurait deux styles : normal et condensé. Dans ce cas, le style condensé sera utilisé pour les valeurs [ultra-condensed](#) à [semi-condensed](#), et le style normal sera utilisé de [normal](#) à [ultra-expanded](#).

La mise en forme du texte

Mise en forme du texte

La propriété **font-size-adjust** permet d'ajuster la hauteur d'un caractère, afin d'uniformiser le rendu visuel des textes ou d'en garder la lisibilité, par exemple lorsque notre choix de police n'est pas présent chez l'utilisateur. **font-size-adjust** permet alors de faire correspondre la hauteur d'un caractère d'une typographie par rapport à un autre.



La mise en forme du texte

Mise en forme du texte

Le W3C nous propose d'utiliser la capacité qu'ont certaines familles de polices à pouvoir contrôler l'affichage ou non de ces glyphes et de ces caractéristiques stylistiques via CSS.

Cela se fait à l'aide de la propriété **font-variant** qui a été étendu par rapport à CSS2.1.

font-variant fonctionne maintenant comme la notation raccourcie d'un ensemble de règles, chacune définissant une particularité :

- ❖ **Font-kerning** active ou non le crénage des caractères
- ❖ **Font-variant-position** active la gestion des indices, exposant et ordinaux
- ❖ **Font-variant-ligatures** active la gestion des différents types de ligatures
- ❖ **Font-variant-caps** active les petites majuscules et les petites capitales
- ❖ **Font-variant-numeric** permet un contrôle pour l'affichage des nombres
- ❖ **Font-variant-alternates** active les glyphes alternatifs de caractère
- ❖ **Font-variant-east-asian** permet un contrôle de l'affichage et/ou la substitution des glyphes asiatiques.

La propriété font-feature-settings propose un contrôle encore plus fin de la gestion de ces caractéristiques, par le biais de mots-clés définis au sein du registre OpenType disponible à cette adresse :

URL : <http://www.microsoft.com/typography/otspec/featurelist.htm>

La mise en forme du texte

Gestion des débordements



La propriété `text-overflow` détermine la façon de traiter un texte lorsqu'il déborde de son parent, notamment lors de l'utilisation conjointe avec `white-space`. Le texte doit avoir une valeur d'`overflow` différente de `visible`. Les valeurs possibles sont les suivantes :

- ❖ **Clip** : c'est la valeur par défaut. Le texte est purement et simplement rogné aux limites de son parent. Le texte qui déborde et donc invisible.
- ❖ **Ellipsis** : le texte est également coupé, mais des points de suspension (...) apparaissent, afin de signaler que le texte affiché n'est pas complètement visible.

Attention, `text-overflow` ne fonctionne que si les lignes dépassent vers la droite du conteneur, et non vers le bas. Il est donc impossible d'utiliser cette règle sur un bloc de plusieurs lignes pour que les points de suspension s'affichent dès que la hauteur de l'élément parent devient trop petite.

La mise en forme du texte

Gestion des débordements

Y a des gens qui n'ont pas réussi parce qu'ils ne sont pas aware, ils ne sont pas "au courant". Ils ne sont pas à l'attention de savoir qu'ils existent cinématographiquement.
Moi je suis aware tu vois

Y a des gens qui n'ont pas réussi parce qu'ils ne sont pas aware, ils ne sont pas "au courant". Ils ne sont pas à l'attention de savoir qu'ils existent cinématographique
Moi je suis aware tu vois

Y a des gens qui n'ont pas réussi parce qu'ils ne sont pas aware, ils ne sont pas "au courant". Ils ne sont pas à l'attention de savoir qu'ils existent cinématographiquem
Moi je suis aware tu vois

La propriété `overflow-wrap` est strictement équivalent à la propriété CSS2.1 `word-wrap`. Elle permet de gérer la façon dont les mots trop longs sont affichés au sein de leur parent. Pour cela nous disposons de deux valeurs :

- ❖ **normal** : les mots trop longs peuvent déborder de leur conteneur si celui-ci est trop petit, sauf si `overflow: hidden` est spécifié.
- ❖ **break-word** : un saut de ligne est effectué arbitrairement au milieu des mots trop longs. Cependant, aucun trait d'union n'est ajouté, la césure n'est pas prise en charge.

Aucun navigateur n'implémente `overflow-wrap`. L'utilisation de `word-wrap` est donc obligatoire, surtout que cette règle est supportée depuis très longtemps dans les navigateurs.

La mise en forme du texte

Gestion des débordements

```
.bloc {  
    -webkit-hyphens: auto;  
    -moz-hyphens: auto;  
    -ms-hyphens: auto;  
    -o-hyphens: auto;  
    hyphens: auto;  
}
```

Y a des gens qui n'ont pas réussi parce qu'ils ne sont pas aware, ils ne sont pas "au courant". Ils ne sont pas à l'attention de savoir qu'ils existent cinématographiquement. Moi je suis aware tu vois

Hyphens

C'est sans nul doute la règle la plus prometteuse concernant la gestion des textes. En effet, la propriété **hyphens** permet d'activer la césure automatique des mots en fin de ligne par l'ajout de traits d'union. Cette gestion intelligente de l'affichage des paragraphes permet une aide à la lecture non négligeable, notamment sur des textes justifiés.

Cette règle accepte trois valeurs :

None : pas de césure, c'est le rendu par défaut.

Auto : la césure automatique effectuée en fonction de la langue de la page.

Manual : la césure effectuée seulement si un trait d'union conditionnel, tel ­, est présent dans un mot.

La mise en forme du texte

Gestion des enchaînements

W H I T E S P A C E

Présente dans la norme CSS2.1, la propriété [white-space](#) permet de préciser la façon dont les espaces présents dans le code source sont gérés. Les valeurs possibles sont [normal](#), [pre](#), [nowrap](#), [pre-wrap](#) et [pre-line](#).

En CSS3, deux nouvelles propriétés permettent d'affiner ce réglage: [text-wrap](#) et [text-space-collapse](#) :

[text-wrap](#) permet de spécifier la façon dont les sauts de ligne sont gérés : [normal](#), [none](#) et [avoid](#).

[text-space-collapse](#) permet de spécifier la façon dont les espaces blancs présents dans le code source sont gérés : [collapse](#), [preserve](#) et [preserve-breaks](#).

La mise en forme du texte

Effets de couleur et d'ombre sur texte

```
text-shadow: 0 0 7px rgba(0,0,0, 0.5);
```

Ombre simple

Depuis peu, la création des ombres a été simplifiée par l'ajout de règles CSS dédiées. Il est à présent possible d'ajouter une ombre sur une boîte ou sur un texte, grâce aux propriétés `text-shadow` et `box-shadow` (prochain chapitre).

La propriété `text-shadow` ajoute une ou plusieurs ombres sur un texte. Une ombre est créée à partir de quatre paramètres (trois longueurs et une couleur) où les valeurs correspondent à :

Décalage horizontal (obligatoire)

- ❖ Une valeur positive décale l'ombre vers la droite,
- ❖ Une valeur négative décale l'ombre vers la gauche.

Décalage vertical (obligatoire)

- ❖ Une valeur positive décale l'ombre vers le bas,
- ❖ Une valeur négative décale l'ombre vers le haut.

Valeur du flou (optionnel)

- ❖ Une valeur de 0 donne une ombre nette,
- ❖ Une valeur positive provoque le flou Gaussien,
- ❖ Les valeurs négatives sont interdites.

Couleur (optionnel-obligatoire)

- ❖ Toutes les définitions de couleur sont autorisées,
- ❖ La couleur peut aussi être écrite avant les trois autres paramètres.

La mise en forme du texte

Effets de couleur et d'ombre sur texte

La propriété raccourcie, combinant `column-width` et `column-count`, est la propriété `columns`

```
div { columns: 3 100px }
```

Le module de positionnement **multicolonnes** permet, comme son nom l'indique, de faire s'écouler du contenu sur plusieurs colonnes de largeurs égales, tel qu'on peut le voir dans le monde de l'imprimerie.

Le positionnement multicolonnes est - contre toute attente - plutôt bien supporté par le collège des navigateurs. Les anciennes versions d'Internet Explorer faisant bien entendu exception à la règle.

Les deux principales propriétés de ce module sont `column-width` et `column-count` :

column-width :

Valeurs : `auto` (par défaut) ou longueur supérieure à zéro (pas de valeur en pourcentage)

Correspond à la largeur "optimale" de la colonne. En réalité, s'adaptera à l'espace disponible et pourra être plus large ou plus réduite.

column-count :

Valeurs : `auto` (par défaut) ou nombre entier supérieur à zéro

Correspond au nombre "optimal" de colonnes. Si `column-width` est fixé, alors `column-count` représente le nombre *maximal* de colonnes

La mise en forme des boites



- Les ombres avec box-shadow
- Les coins arrondis avec border-radius

La mise en forme des boîtes

Les ombres avec box-shadow



```
box-shadow: -1px 2px 10px 3px rgba(0, 0, 0, 0.3) inset;
```

La propriété **box-shadow** ajoute une ou plusieurs ombres sur une boîte. Une ombre est créée à partir de six paramètres (quatre longueurs, une couleur et un mot-clé) ou les valeurs correspondent à :

Décalage horizontal (obligatoire)

- ❖ Une valeur positive décale l'ombre vers la droite,
- ❖ Une valeur négative décale l'ombre vers la gauche.

Décalage vertical (obligatoire)

- ❖ Une valeur positive décale l'ombre vers le bas,
- ❖ Une valeur négative décale l'ombre vers le haut.

Valeur du flou (optionnel)

- ❖ Une valeur de 0 donne une ombre nette,
- ❖ Une valeur positive provoque le flou Gaussien,
- ❖ Les valeurs négatives sont interdites.

Valeur d'étendue (optionnel)

- ❖ Une valeur positive augmente la taille de l'ombre, par rapport à celle de la boîte,
- ❖ Une valeur négative diminue la taille de l'ombre.

Couleur (optionnel-obligatoire)

- ❖ Toutes les définitions de couleur sont autorisées,
- ❖ La couleur peut aussi être écrite avant les trois autres paramètres.

Mot-clé inset (optionnel)

Crée une ombre interne, c'est-à-dire, à l'intérieur de la boîte.

La mise en forme des boîtes

Les ombres avec box-shadow

Ajout des préfixes pour un fonctionnement dans la majorité des navigateurs

```
div{  
    -webkit-box-shadow: 0 0 5px 3px black;  
    -moz-box-shadow: 0 0 5px 3px black;  
    box-shadow: 0 0 5px 3px black;  
}
```

Contrairement à `text-shadow` qui n'est jamais préfixé, `box-shadow` nécessite l'utilisation des préfixes pour fonctionner dans la majorité des navigateurs. Pour `box-shadow`, il faudra ajouter `-webkit-` et `-moz-`.

L'ajout des préfixes pour `box-shadow` doit être soumis à réflexion. En effet la majorité des navigateurs récents supporte la règle non préfixée. Ajouter les préfixes alourdit le code, et il n'est certainement plus utile de maintenir une telle syntaxe. Une ombre n'est qu'un élément de décor. Attention quand même au cas où le texte qui reçoit une ombre est de la même couleur que le fond, il ne soit plus lisible sans ombre.

La mise en forme des boîtes

Les coins arrondis avec border-radius



À l'instar des ombres, la création de coins arrondis est une pratique courante qui permet d'adoucir un élément de design. Sur une page Web, les coins arrondis permettent d'alléger le rendu visuel des blocs, en lieu et place des coins carrés.

La création de ces coins arrondis passe encore très souvent par la technique utilisée pour la création des ombres, à savoir, la création d'images de fond que nous appliquons sur différents blocs.

Cette technique est devenue obsolète depuis l'apparition de la propriété **border-radius** qui permet d'arrondir les coins d'une boîte beaucoup plus facilement et proprement.

La mise en forme des boîtes

Les coins arrondis avec border-radius

```
#cadre {  
    border-radius: 10px;  
}
```

ici un bloc arrondi

La propriété **border-radius** permet d'ajouter des coins arrondis sur une boîte, en spécifiant une taille de rayon pour ses angles. **Border-radius** accepte une, deux, trois ou quatre valeurs séparées par des espaces :

Une valeur : les quatre arrondis sont identiques ;

Deux valeurs :

- ❖ La première a défini l'angle en haut à gauche et l'angle en bas à droite ;
- ❖ La deuxième définit l'angle en haut à droite et l'angle en bas à gauche.

Trois valeurs :

- ❖ La première définit l'angle en haut à gauche ;
- ❖ La deuxième définit l'angle en haut à droite et l'angle en bas à gauche ;
- ❖ La troisième définit l'angle en bas droite.

Quatre valeurs :

- ❖ Les angles sont définis chacun leur tour, à partir de l'angle en haut à gauche puis en tournant dans le sens des aiguilles d'une montre.

La mise en forme des boîtes

Les coins arrondis avec border-radius

Comme pour box-shadow, et afin d'offrir un support maximal de coins arrondis, border-radius peut-être préfixée

```
div{  
    -webkit-border-radius: 20px;  
    -moz-border-radius: 20px;  
    border-radius: 20px;  
}
```

En revanche, et c'est encore plus vrai que pour [box-shadow](#), vérifier si l'ajout de ces préfixes est primordial puisque border-radius est compatible depuis assez longtemps en version non préfixée pour être utilisé tel quel. La majorité des navigateurs supporte la syntaxe officielle, les autres n'ajouteront simplement pas les coins arrondis.

La mise en forme du fond



- Les fonds multiples
- Les gradients de fond (-webkit-gradient)
- Les fonds ajustés aux conteneurs

L'utilisation des images d'arrière-plan en CSS est une fonctionnalité primordiale. Toutes les images concernant le design ne doivent pas être présentes sous forme de balises img dans le code HTML. Ces images n'apportent effectivement pas d'indication essentielle à une bonne compréhension du contenu. Elles n'ajoutent pas de signification à la page, elles se contentent de «rendre joli». Pour appliquer nos images en CSS, nous disposons de plusieurs règles :

[Background-image](#) ou [background](#) pour les blocs ;
[List-style-image](#) ou [list-style](#) pour les éléments de liste à puces ;
[Border-image](#) pour les images bordures ;
[Content](#) pour le contenu généré.

CSS3 apporte de nombreuses nouveautés concernant la gestion de ces images arrière-plan. Il est par exemple possible de choisir l'origine du positionnement d'une image appliquée en arrière-plan, de rogner le rendu de celle-ci, ou même de jouer sur sa taille. Mais la principale innovation est sans doute la gestion de plusieurs arrière-plans pour un même élément HTML.

La mise en forme du fond

Les fonds multiples

CSS3 rend possible l'affichage de **plusieurs images** en arrière-plan, dans un même élément en proposant de cumuler les valeurs au sein des propriétés **background-image**, **background-position** et **background-repeat**.

```
background-image: url("image1"), url("image2");  
background-position: x y, x y;  
background-repeat: no-repeat;
```

Pour utiliser plusieurs images arrière-plan sur un même élément, il suffit de les séparer par une virgule dans la définition de la règle CSS. Il n'est pas obligatoire de définir des fonds multiples composés uniquement d'images, la combinaison d'images et de couleurs est également possible, mais les arrière-plans multiples prennent tout leurs sens quand il s'agit d'appliquer plusieurs images en fond d'un élément unique.

La mise en forme du fond

Les gradients de fond

Le W3C définit deux fonctions, [linear-gradient](#) et [radial-gradient](#), qui nous permettent de créer des dégradés, et comme le résultat est une image, ces fonctions sont utilisables avec les mêmes propriétés que celles qui acceptent des images.

- ❖ [Background-image](#) ou [background](#)
- ❖ [List-style-image](#) ou [list-style](#)
- ❖ [Border-image](#)
- ❖ [Content](#)

Il existe également de fonctions supplémentaires, [repeating-linear-gradient](#) et [repeating-radial-gradient](#) qui sont, comme leurs noms l'indiquent, des dégradés linéaires ou radio qui se répètent.

L'utilisation des dégradés sur le Web est, comme pour les ombres, une manière de donner du relief un graphisme. Les dégradés, à condition qu'ils soient bien utilisés, sont quasiment non visibles pour l'utilisateur, mais son utilisation est nécessaire pour que les éléments d'une page ne paraissent plus plats. Ils sont très souvent employés pour la création de boutons, d'éléments de menus ou encore en arrière-plan d'un site, et souvent utilisés conjointement avec des ombres pour accentuer encore l'effet de lumière.

Commençons par comprendre ce que sont les dégradés. Dans la pratique, les dégradés CSS sont des images créées à la volée qui représente une transition douce entre différentes couleurs. Pour l'instant il n'en existe deux types, les dégradés linéaires et les dégradés radiaux.

La mise en forme du fond

Les gradients de fond

```
background: linear-gradient(to right, green,  
rgba(0,255,0,0));
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

La fonction `linear-gradient` permet de créer un dégradé linéaire et prend trois paramètres où les valeurs sont dans l'ordre :

Direction de la ligne du dégradé (optionnel) qui peut être :

- ❖ Un mot-clé (`top`, `bottom`, `left` ou `right`) qui indique le côté d'où commence le dégradé, en direction du côté opposé.
- ❖ Deux mots-clés (ex : `top left`) qui indique l'angle d'où commence le dégradé, en direction de l'angle opposé.
- ❖ Un angle, une valeur de `0deg` est identique à `left`, une valeur de `90deg` est identique à `bottom`.

Couleur de départ (obligatoire),

Couleur de fin (obligatoire).

La direction par défaut d'un dégradé est `top`, ce qui signifie que le dégradé est créé depuis le haut vers le bas si aucune direction n'est spécifiée.

La mise en forme du fond

Les gradients de fond

```
background: radial-gradient(ellipse at center top,  
green, blue);
```

 Lorem ipsum dolor sit amet, consectetur adipiscing elit.

La fonction `radial-gradient` permet de créer un dégradé radial, et prend quatre paramètres où les valeurs sont dans l'ordre :

Position du centre (optionnel)

- ❖ Un mot-clé (`top`, `bottom`, `left`, `right` ou `center`) ;
- ❖ Deux mots-clés (ex: `top left`)
- ❖ Valeurs numériques

Définition de la taille et de la forme de l'ellipse (optionnel)

- ❖ Choix de la forme : `circle` ou `ellipse`,
- ❖ Choix de la taille
- ❖ Valeurs numériques par rapport au centre déjà défini

Couleur de départ (obligatoire) ;

Couleur de fin (obligatoire).

La mise en forme du fond

Les fonds ajustés aux conteneurs

La propriété `background-clip` permet de définir les limites auxquelles une image d'arrière-plan est rognée et donc n'est plus visible.

```
div {  
    background-image: url("soif.jpg");  
    background-clip: content-box;  
}
```

```
div {  
    background-image: url("soif.jpg");  
    background-clip: padding-box;  
}
```

```
div {  
    border: 5px dotted #393838;  
    background-image: url(soif.jpg);  
    background-clip: border-box;  
}
```

La propriété `background-clip` a trois valeurs possibles :

- ❖ **Border-box** : le fond est visible jusqu'aux limites extérieures de la bordure;
- ❖ **Padding-box** : le fond est visible jusqu'aux limites extérieures de la marge interne;
- ❖ **Content-box** : le fond est visible uniquement sur le contenu. Il s'arrête à la limite intérieure de la marge interne. Cela correspond à la taille spécifiée par `width` et `height` dans le modèle de boîte classique.

La valeur par défaut est `background-clip : border-box` .

La mise en forme du fond

Les fonds ajustés aux conteneurs

La propriété `background-origin` permet de définir l'origine du positionnement initial des images d'arrière-plan.

```
div {  
  /* ... */  
  border: 20px dotted #393838;  
  background-origin: padding-box;  
}
```

```
div {  
  /* ... */  
  background-origin: border-box;  
}
```

```
div {  
  /* ... */  
  background: url(soif.jpg) no-repeat;  
  background-origin: content-box;  
}
```

La propriété `background-origin` a trois valeurs possibles, qui sont similaires à celles de `background-clip` :

- ❖ **Border-box** : l'origine est la limite externe de la bordure;
- ❖ **Padding-box** : l'origine est la limite externe de la marge interne ;
- ❖ **Content-box** : l'origine est la limite externe du contenu.

La valeur par défaut est `background-origin : padding-box`.

La mise en forme du fond

Les fonds ajustés aux conteneurs

La propriété background-size permet de spécifier la taille de l'image appliquée en arrière-plan.

```
div {  
    background-image:url(soif.jpg);  
    background-size: 540px 360px;  
}
```

Les valeurs possibles sont :

Valeurs numériques pour définir la largeur et/ou la hauteur. Le ratio est conservé si une seule valeur est fournie ;

Pourcentage : par rapport à la boîte elle-même. Une valeur de 100 % couvre la largeur de la boîte et le ratio est conservé ;

[Contain](#) : le fond est étendu à la taille de la boîte en ajustant le plus grand côté (largeur ou hauteur) au sein de celle-ci. Le ratio de l'image est conservé ;

[Cover](#) : le fond est étendu à la taille de la boîte en ajustant le plus petit côté (largeur ou hauteur) au sein de celle-ci. Le ratio de l'image est conservé.

La valeur par défaut est [background-size : auto](#).

Les transformations



- Utiliser les transformations
- Les transitions
- les animations : les animations, les transitions.

Les transformations

Utiliser les transformations

CSS3 apporte les transformations en 2 dimensions à travers la propriété **transform** et une liste de fonctions prédéfinies.

La syntaxe est simple d'emploi.

```
transform: function(value);
```

Il est également possible d'effectuer des transformations combinées en espaçant les fonctions d'un simple caractère blanc.

```
transform : function1(value1) function2(value2)  
function3(value3);
```

La propriété CSS **transform** permet de manipuler un élément HTML sur les axes X et Y (horizontal et vertical) grâce à des fonctions diverses de transformation graphique. Il est donc possible de modifier l'apparence d'un élément grâce à un ensemble fonctions 2D :

- ❖ Translation ([translate](#)),
- ❖ Mise à l'échelle ([scale](#)),
- ❖ Rotation ([rotate](#))
- ❖ Inclinaison ([skew](#))

Les transformations

Utiliser les transformations

Pour pouvoir appliquer des transformations, nous avons besoin de savoir quel est le point d'origine (d'ancre) de la transformation. La propriété `transform-origin` définit ce point d'origine.

La valeur initiale de cette propriété est le centre de l'élément, ce qui équivaut à la notation :

```
transform-origin: 50% 50%;
```

Il est possible de changer cette valeur en utilisant un mot-clef de position (`top`, `right`, `bottom`, `left`) suivi d'une valeur chiffrée dont l'unité peut varier (`px`, `%`, etc.)

```
div {  
    transform-origin: top 0 left 0;  
    transform: scale(1.25);  
}
```

Il s'agit là de la syntaxe proposée par le W3C. À l'heure actuelle (2012) aucun navigateur n'implémente cette syntaxe correctement. Cependant, il suffit de supprimer les mots-clefs de position pour obtenir des résultats sur tous les navigateurs récents (toujours à condition d'utiliser les préfixes vendeurs `-webkit-`, `-moz-`, `-ms-`, `-o-` selon les versions).

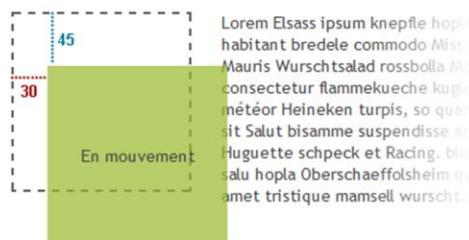
Les transformations

Utiliser les transformations

Une fois l'origine choisie, nous pouvons affecter des transformations à nos éléments avec la propriété `transform`.

La fonction `translate` permet d'effectuer une translation (un déplacement) de l'élément sur les axes X et Y.

`translate (30px, 45px)`



Il n'y a ici aucune notion de flux, l'élément part de son emplacement courant, quel que soit le type de positionnement que vous lui aurez attribué.

```
transform: translate(x, y);
```

y est une **valeur optionnelle** équivalente à 0 si elle n'est pas renseignée. Les deux valeurs peuvent être négatives.

Ces fonctions permettent de réaliser une translation sur l'axe X (`translateX`) ou Y (`translateY`).

```
transform: translateX(value) translateY(value);
```

Les transformations

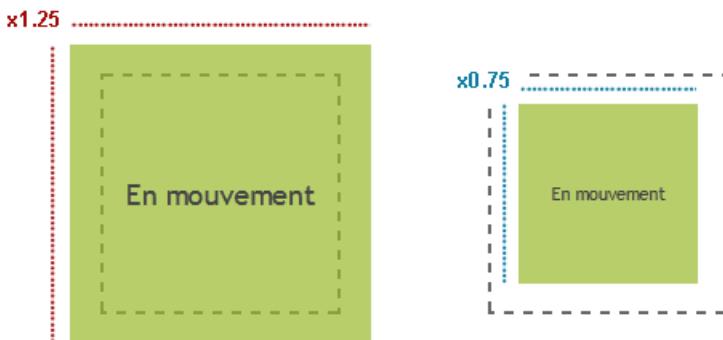
Utiliser les transformations

La fonction `scale` permet d'agir sur l'échelle (les dimensions) de l'élément. La valeur initiale est 1, tandis que les valeurs supérieures à 1 créent un effet d'agrandissement, et les valeurs inférieures créent un effet de réduction.

```
transform: scale(x, y);
```

La valeur `y` est optionnelle et sera égale à la valeur de `x` si elle est non renseignée, par exemple pour un agrandissement d'un facteur 1.25 :

```
transform: scale(1.25);
```



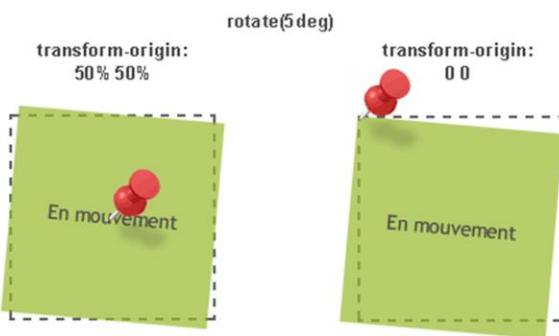
Les valeurs de `x` et `y` peuvent être aussi négatives. Il est possible d'effectuer ce "zoom" sur tous les éléments a priori... mais ce n'est pas parce que vous pouvez le faire qu'il faut le faire. N'oubliez pas qu'une image agrandie pourra être floue ou mal interpolée selon sa résolution initiale et celle de l'écran de destination.

Sur le même principe que pour les fonctions dérivées de `translate`, ces deux fonctions permettent de définir indépendamment les valeurs `x` et `y`.

Les transformations

Utiliser les transformations

Il s'agit d'une des plus simples fonctions ([rotate](#)) à comprendre. Comme son nom l'indique, elle permet d'effectuer une **rotation** de l'élément ciblé.



Cette rotation s'exprime en **degrés** (unité deg), et peut être négative et supérieure de manière absolue à 360. Ce dernier point n'a de réel intérêt que lors d'une animation d'un état à un autre afin de présenter, par exemple, une rotation de plusieurs tours d'un élément. Autrement, sans animation de la rotation, la valeur 380° équivaut visuellement à une rotation de 20°.

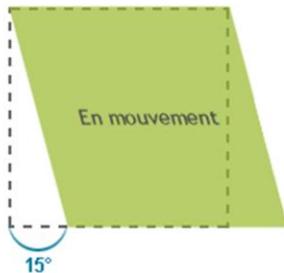
```
transform: rotate(5deg);
```

Les transformations

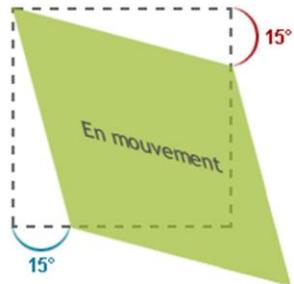
Utiliser les transformations

La fonction `skew` permet d'oblier la forme d'un élément. À ma grande surprise, la documentation du W3C ne parle que des fonctions `skewX` et `skewY`, et pour cause :

`skew(15deg, 0deg)`



`skew(15deg, 15deg)`



Il s'agit des fonctions dérivées de `skew`. Voici deux exemples de transformation en utilisant les deux fonctions. Vous aurez compris la syntaxe de base :

```
transform: skewX(15deg) ;
```

...ou en changeant les valeurs de X et Y :

```
transform: skewX(15deg) skewY(15deg) ;
```

Les transformations

Les transitions

CSS 3 Transitions



Le principe de base d'une **transition CSS3** est de permettre une transition douce entre l'ancienne valeur et la nouvelle valeur d'une propriété CSS lorsqu'un événement est déclenché :

- ❖ soit via une pseudo-classe telles que :hover, :focus ou :active
- ❖ soit via JavaScript

Précédemment, ce genre de comportement n'était possible qu'avec l'usage de JavaScript. Ce nouveau module CSS3 permet dorénavant de s'en affranchir au profit exclusif des feuilles de style.

Pour définir une nouvelle transition animée, il est nécessaire de préciser au minimum :

- ❖ La ou les propriété(s) à animer
- ❖ La durée de l'animation

Les transformations

Les transitions

Les propriétés spécifiques pour les transitions en CSS 3.

Propriété	Explication
<code>transition-property</code>	Précise les propriétés CSS à transformer
<code>transition-duration</code>	Précise la durée de la transition
<code>transition-timing-function</code>	Précise la fonction de transition à utiliser, le modèle d'interpolation (accélération...) Veuillez répéter
<code>transition-delay</code>	Précise le retard (ou l'avance) du départ de la transition

Les deux propriétés minimales nécessaires pour rendre fonctionnelle une transition en CSS 3 sont `transition-property` et `transition-duration`.

Il existe d'autres propriétés CSS spécifiques aux transitions : `transition-timing-function`, `transition-delay` et la notation raccourcie `transition`, que nous allons voir en détails.

Transition-property

Elle permet de définir la propriété qui sera animée lors de la transition. Les valeurs possibles sont :

- ❖ `All` : toutes les propriétés animables
- ❖ `None` : aucune transition.
- ❖ Une ou plusieurs propriétés animables séparées par une virgule.

`Transition-property: color, width;`

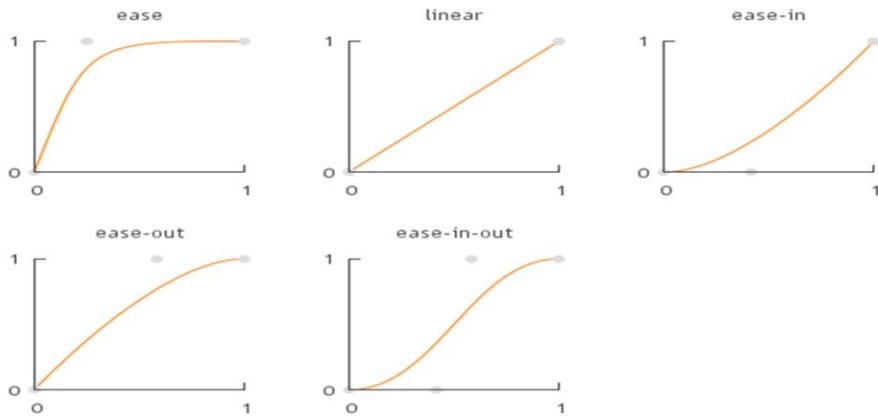
Transition-duration

Cette propriété définit tout simplement la durée totale de la transition exprimée en secondes. La valeur par défaut est zéro seconde ce qui ne provoque aucune transition lors du changement d'état. Voici la syntaxe pour une transition d'une durée de trois secondes.

`Transition-duration: 3s;`

Les transformations

Les transitions



Transition-timing-function

Cette propriété définit la progression ou l'évolution de la vitesse de transition. Une fonction mathématique va permettre le calcul des différentes valeurs interpolées lors de la transition.

Les fonctions prédéfinies sont les suivantes :

ease : Rapide sur le début et ralenti sur la fin.

linear : La vitesse est constante sur toute la durée de l'animation.

ease-in : Lent sur le début et accélère de plus en plus vers la fin.

ease-out : Rapide sur le début et décélère sur la fin.

ease-in-out : Le départ et la fin sont lents.

Les transformations

Les transitions

Il reste une propriété simple relative aux transitions, il s'agit de la propriété `transition-delay`. Cette propriété permet un départ retardé de la transition.

```
div{  
    -webkit-transition-delay: 500ms;  
    -moz-transition-delay: 500ms;  
    -ms-transition-delay: 500ms;  
    -o-transition-delay: 500ms;  
    transition-delay: 500ms;  
}
```

La propriété `transition-delay` permet un départ retardé de la transition. Si la valeur précisée positive, alors le départ de la transition sera décalé d'autant. Si cette valeur est négative, l'animation de la transition débute immédiatement mais depuis l'État interpolé correspondant au delta de temps indiqué en valeur négative. La durée totale de l'animation est diminuée de cette valeur.

Transition

Comme pour beaucoup de propriétés composées en CSS, il existe un raccourci qui permet de renseigner la plupart des valeurs avec une seule instruction. Il faut donc saisir les différentes valeurs séparées par un espace.

`transition: width 2s ease;`

Les transformations

Les animations

CSS 3 *Animations*



Les animations CSS3 sont semblables aux transitions. La majeure différence entre les deux est que **les animations permettront un contrôle très précis dans le temps** de la valeur que prendront les différentes propriétés CSS.

Une animation peut être lancée lors d'un changement de pseudo-classe (`:hover`, `:focus`, `:active`, `:target`, `:checked`, etc), lors d'un changement de class via Javascript, ou simplement lors du chargement de la page.

Si l'animation n'est répétée qu'une fois par défaut, elle peut être itérée à souhait. Pour réaliser une animation CSS, nous avons besoin de deux éléments distincts: une déclaration d'images-clés (`keyframe`) définie au sein d'une `@-rule`, et un lien d'un sélecteur vers cette animation au sein d'un bloc de code CSS.

Les animations étant encore à l'état d'ébauche, il vous faudra évidemment ajouter les préfixes navigateurs (présentement, `Webkit -webkit-`, `Gecko -moz-` et `Opera -o-` supportent les animations. Il faudra également pensé à `IE 10 -ms-`).

Ainsi, nous devrons écrire `@-moz-keyframes` et `-moz-animation` afin que nos animations soient comprises sur Firefox. Le même principe s'appliquera aux autres navigateurs.

Les transformations

Les animations

@keyframes va permettre de définir les différentes étapes de l'animation.

Voici l'exemple d'une règle définissant les étapes d'une transition classique :

```
@keyframes hide{  
    0%{  
        opacity: 1;  
    }  
    100%{  
        opacity: 0;  
    }  
}
```

Elle est pour l'instant encore préfixé pour Webkit, Gecko et IE10 et son implémentation est assez bonne dans les navigateurs concernés. Les étapes de l'animation vont être identifiées par un pourcentage ou l'un des mots-clés from qui équivaut à 0 % et to qui équivaut à 100 %. Pour chaque étape, comme lors d'une transition simple, nous allons définir les règles qui subissent une modification.

Il ne reste ensuite à donner un nom à cette @keyframes pour pouvoir l'associer avec une animation.

Les transformations

Les animations

Maintenant que nos keyframes sont créés et référencées par le nom qui leur a été donné, il ne nous reste plus qu'à appeler cette animation à partir d'un sélecteur. Les différentes propriétés d'animation citées ci-dessous :



Lancer les animations

- ❖ **animation-name:** Où on indique le nom de l'animation à utiliser. (obligatoire)
- ❖ **animation-duration:** Où on indique la durée en milliseconde (ms) ou en seconde (s) de l'animation. (Obligatoire)
- ❖ **animation-iteration-count:** Le nombre de fois que l'animation doit être effectué. La valeur devra être un nombre entier ou le mot clé infinite pour que l'animation se répète à l'infinie. La valeur par défaut est de 1. (Facultative)
- ❖ **animation-direction:** Définira si l'animation doit toujours jouer du début à la fin, ou si une fois rendu à la fin, elle doit refaire l'animation en sens inverse. Par défaut l'animation recommencera du début (normal), mais l'utilisation du mot clé alternate alternera la direction de l'animation. (Facultative)
- ❖ **animation-timing-function:** Définit un effet de timing à utiliser (le modèle d'interpolation) entre chaque keyframe. Comme vu précédemment, cette propriété est également modifiable au sein même d'une keyframe. (Facultative)
- ❖ **animation-delay:** Cette valeur définira un délai d'attente avant le début de l'animation, ou dans le cas d'une valeur négative, l'avance que doit prendre l'animation avant de débuter. (Facultative)

Les transformations

Les animations



Lancer les animations (suite)

❖ **animation-fill-mode:** Cette propriété définit l'état de départ et de fin de votre animation. Voici les différentes valeurs possibles:

- **forwards:** indique au navigateur de laisser l'élément dans son état final lors de la dernière itération. (l'élément ne revient donc pas à son état initial)
- **backwards:** indique au navigateur de placer l'élément dans son état définit au keyframe 0% au chargement de la page, même si un délai négatif est indiqué.
- **both:** appliquera les deux valeurs précédentes.
- **none:** indiquera au navigateur de styler l'élément selon son état à la première keyframe visible (dans le cas d'un délai négatif) et de ramener l'animation à la keyframe 0% après la dernière itération. Ceci est le comportement par défaut.

Les transformations

Les animations

Évidemment, autant de valeurs (avec les préfixes navigateurs en plus) vous créeront vite 21 lignes supplémentaires dans votre bloc de code. Ainsi, nous utiliserons plus souvent qu'autrement la syntaxe simplifiée:

```
selecteur {  
    animation: <name> <duration> <timing-function> <delay>  
    <iteration-count> <direction> <fill-mode>;  
}
```

Ou si vous désirez appliquer plusieurs animations sur un élément:

```
selecteur {  
    animation:  <name> <duration> <timing-function> <delay>  
    <iteration-count> <direction> <fill-mode>,  
    <name> <duration> <timing-function> <delay> <iteration-  
    count> <direction> <fill-mode>;  
}
```

Les transformations

Les animations

Mettre l'animation sur pause

Et pour clore ce chapitre sur les animations CSS3, il vous est également possible de mettre en pause une animation via la propriété `animation-play-state` qui prendra l'une des deux valeurs suivantes : `running` (par défaut) ou `paused` pour arrêter l'animation.

```
selecteur:hover {  
    -webkit-animation-play-state:paused;  
    -moz-animation-play-state:paused;  
    animation-play-state:paused;  
}
```