

Introduction à HTML5

par **Synbioz** ([Gardez le fil](#)) ([Blog](#))

Date de publication : 07 janvier 2012

Dernière mise à jour :

HTML5 est une évolution de la norme HTML dont le but avoué est de faciliter le développement d'interfaces utilisateur riches. HTML5 est beaucoup plus orienté applicatif que ses prédécesseurs et veut permettre de s'affranchir de plugins pour utiliser au maximum les technologies Web natives pour construire une application riche.

L'article original peut être lu sur le blog de **Synbioz** : [Introduction à HTML5](#).

HTML5, pourquoi ?.....	3
Nouvelle structuration.....	3
Microdata.....	5
Les formulaires.....	5
Audio et vidéos natifs.....	7
Canvas.....	7
API JavaScript enrichie.....	7
Trouver des éléments via l'API DOM.....	7
Trouver des éléments via les sélecteurs CSS (API Selectors).....	7
Attributs personnalisés data-*.....	8
Les listes de classes.....	8
Cookies on stéroïds: localStorage et sessionStorage.....	8
Bases de données WebSQL.....	9
API de cache.....	9
Les sockets Web.....	9
Géolocalisation.....	10
Accès à l'historique.....	10
Gestion des fichiers.....	11
Webworkers.....	12
CSS3.....	12
Les nouveaux sélecteurs CSS: nth-child(even), nth-child(odd), :not, first-child ?.....	12
La gestion des polices.....	13
Gestion de colonnes.....	13
Maîtrise de l'opacité.....	13
Bordures arrondies.....	13
Dégradés.....	14
Fonds redimensionnables.....	14
Fonds multiples.....	14
Ombrages.....	14
Transitions et transformation.....	15
Conclusion.....	15
Remerciements.....	16

HTML5, pourquoi ?

HTML5 est une évolution de la norme HTML regroupant un ensemble de technologies : SVG, CSS3, WebGL, File API, MathML...

Le but avoué de cette nouvelle version de HTML est de faciliter le développement d'interfaces utilisateur riches. HTML5 est beaucoup plus orienté applicatif que ses prédécesseurs et veut permettre de s'affranchir de plugins pour utiliser au maximum les technologies Web natives pour construire une application riche.

HTML5 arrive donc avec beaucoup de nouvelles capacités disponibles depuis une API vaste et variée. On pourra notamment communiquer avec des fonctions matérielles (micro, webcam, carnet d'adresses...) par l'intermédiaire d'API.

On peut facilement grâce à HTML5 visionner des vidéos sans plugin, ajouter des effets visuels aux textes, images, vidéos... On peut aussi utiliser des polices non standards, dessiner en SVG, faire de la 3D...

Voyons les choses qui me semblent être les plus intéressantes avec quelques exemples d'utilisation.

Nouvelle structuration

En premier lieu, HTML5 met largement à jour le système de balisage et donc la structuration d'une page. Voici à quoi pourrait ressembler une page-type utilisant certaines nouvelles balises pour améliorer la dimension sémantique du code :

```
<body>
  <header>
    <hgroup>
      <h1>Titre</h1>
      <h2>Sous-titre</h2>
    </hgroup>
  </header>

  <nav>
    <ul>
      Navigation
    </ul>
  </nav>
  <section>
    <article>
      <header>
        <h1>Titre</h1>
      </header>
      <section>
        Contenu...
      </section>
    </article>
    <article>
      <header>
        <h1>Titre</h1>
      </header>
      <section>
        Contenu
      </section>
    </article>
  </section>

  <aside>
    Liens externes
  </aside>

  <figure>
    
```

```
<figcaption>Graphique</figcaption>
</figure>

<footer>
  Copyright © Synbioz
  <time datetime="2011-12-08">2011</time>.
</footer>
</body>
```

Comme vous pouvez le voir, de nouvelles balises sémantiques sont utilisées pour regrouper les différents éléments en groupes logiques. Le principal avantage de ce balisage amélioré est de simplifier la lecture des pages par les moteurs de recherche qui seront alors en mesure de mieux classer et restituer les informations lors d'une recherche.

Les balises ajoutées l'ont aussi été pour simplifier l'écriture de la structure d'une application Web. Jusque-là on utilisait des div avec des id ou des classes pour représenter un entête, un menu, un article, une barre de navigation ou un pied de page. Pas très facile pour un moteur de recherche de faire la différence entre une pub, un menu ou l'article dans une page.

C'est donc pour pallier ce problème, faciliter et alléger le code que des balises sont apparues : header, section, article, nav, aside, footer...

L'attribut "rel", bien trop peu employé à mon goût, comporte lui aussi son lot de nouveautés :

- alternate : lien vers une version alternative de la page (version imprimable, traduction, miroir...);
- author : lien vers l'auteur du document ;
- bookmark : URL permanente pouvant servir de bookmark ;
- external : lien vers une page externe ;
- help : lien vers un document d'aide ;
- license : lien vers les copyrights du document ;
- next : prochain document dans la sélection (ex. : prochain article) ;
- prev : document précédent dans la sélection (ex. : article précédent) ;
- nofollow : lien vers un document tel qu'un lien commercial (Google utilise cet attribut pour ne pas suivre un lien) ;
- noreferrer : demande au navigateur de ne pas envoyer de référent dans les entêtes HTTP si l'utilisateur clique sur le lien ;
- prefetch : demande au navigateur de précharger la page liée ;
- search : lien vers un outil de recherche pour le document ;
- tag : permet d'affecter un tag (mot-clé) au document.

Quelques exemples :

```
<link rel="alternate" type="application/rss+xml" href="http://blog.com/feed"/>
<link rel="icon" href="/favicon.ico"/>
<link rel="pingback" href="http://blog.com/xmlrpc"/>
<link rel="prefetch" href="http://blog.com/main"/>

<a rel="archives" href="http://blog.com/archives">anciens articles</a>
<a rel="external" href="http://notmysite.com">lien externe</a>
<a rel="license" href="http://www.apache.org/licenses/LICENSE-2.0">licence</a>
<a rel="nofollow" href="http://notmysite.com/somewhere">pub</a>
<a rel="tag" href="http://blog.com/category/ruby">Articles à propos de Ruby</a>
```

Vous devez vous dire que l'intérêt de ces nouveautés est limité aux moteurs de recherche qui voient leur travail simplifié. C'est vrai en partie mais n'oubliez pas qu'être bien référencé c'est avoir une meilleure présence sur le Web et donc plus de visiteurs. Il est également plus simple pour vous de créer une ossature HTML compréhensible ce qui vous facilitera le travail de référencement, d'accessibilité pour vos visiteurs déficients et la maintenabilité.

Une structure claire, naturelle et légère est la base d'un bon site et je pense que HTML5 va largement contribuer à l'amélioration sémantique des applications développées.

Microdata

Les microdata sont un autre paramètre à prendre en compte pour un meilleur référencement. Les microdata sont un balisage sémantique avec un vocabulaire personnalisé qui permet de détailler très finement un élément dans la page. Ces détails pourront être repris par le moteur de recherche pour présenter des résultats plus pertinents et détaillés. On pourra donc par exemple détailler :

- un avis ;
- une note ;
- un fil d'Ariane ;
- des événements ;
- des personnes ou entreprises ;
- des produits ;
- ...

Un exemple vaudra mieux qu'un long discours :



Vous voyez donc que sous le lien, on retrouve des étoiles représentant une note, l'auteur et la date.

Le code de cette page pourrait être, dans sa forme la plus simple :

```
<div>
  <h1>Dragon Age</h1>
  Avis rédigé par GameSpot le 3 novembre
  Note: 5 sur 5
</div>
```

Vous pouvez tester vos pages utilisant des microdata via les **outils Google** pour voir ce que donnera un résultat de recherche pointant sur votre page.

Les formulaires

Certainement l'une des choses les plus utilisées dans toute application Web, les formulaires ne sont pas en reste côté améliorations. Jusque-là, les contrôles disponibles et surtout les types de données associées étaient très limités. HTML5 nous apporte quelques briques manquantes qui vont faciliter la récupération et le traitement de données.

Maintenant les formulaires peuvent plus ou moins être typés. L'apparition de ces nouveaux types rend le code plus compréhensible et plus accessible. Ce typage permet également au navigateur de proposer des contrôles adaptés à la situation. En effet le navigateur pourra faire du préremplissage ciblé, utiliser une interface appropriée (ex. : calendrier pour les dates, colorpicker...) ou encore afficher un clavier adapté au contexte lorsqu'on utilise un téléphone.

Voici les nouveaux types d'input disponibles :

- datetime, month, time, week...
- number ;

- search ;
- range ;
- color ;
- tel ;
- email.

Les widgets associés à ces nouveaux types sont particulièrement bien développés sur téléphone mobile où l'intérêt ergonomique est le plus perceptible.

Des attributs ont aussi été ajoutés :

- placeholder qui permet d'avoir un texte par défaut qui disparaîtra automatiquement lors de la saisie utilisateur sans avoir la moindre ligne de JavaScript à écrire ;
- autofocus qui donnera le focus à ce champ au chargement de page ;
- required qui définit le champ comme obligatoire ;
- pattern qui permet de définir un format qui devra être respecté pour valider le formulaire.

Voici quelques exemples d'utilisation :

```
<input type="text" required />

<input type="email" value="votre@email.com" />

<input type="date" min="2010-01-01" max="2011-12-31" value="2010-10-31"/>

<input type="range" min="0" max="50" value="10" />

<input type="search" results="10" placeholder="Recherche..." />

<input type="tel" placeholder="1234567890" pattern="^\d{10}$" />

<input type="color" placeholder="ex. #bbbbbb" />

<input type="number" step="1" min="-5" max="10" value="0" />
```

Parmi les possibilités moins souvent évoquées, les formulaires peuvent maintenant être découpés et être soumis de façons différentes.

En effet un champ peut maintenant se trouver hors du formulaire en lui-même, simplement en spécifiant l'attribut `form="id_formulaire"`.

Un même formulaire peut aussi être soumis de façon différente, en modifiant son comportement directement depuis un input de type submit.

Cela peut être très pratique par exemple dans un moteur de blog ou à l'écriture d'un article vous pouvez soit l'enregistrer soit en faire une prévisualisation.

Exemple :

```
<form id="new_post" action="/articles" method="post">
<input type="submit" formaction="/articles/preview" formmethod="get" value="Aperçu" />
</form>

<input type="submit" form="new_post" value="Publier" />
```

Audio et vidéos natifs

Plus besoin d'intégrer un lecteur flash pour pouvoir lire du son ou de la vidéo dans une page Web. HTML5 intègre cette possibilité nativement. Une simple balise permet de lire une vidéo et d'afficher ses contrôles ! Il vous est même possible de contrôler tout cela directement en JavaScript :

```
/* Joue un son en boucle au chargement de page en affichant ses contrôles */
<audio src="/audio/test.ogg" autoplay controls loop>

/* Joue une vidéo en boucle au chargement de page en affichant ses contrôles */
<video id="vid" src="/videos/test.ogv" autoplay controls loop>

<script type="text/javascript" charset="utf-8">
  var video = document.getElementById('vid');
  video.pause();
  alert(video.src);
  /* Durée du son */
  alert(video.duration);
  /* Position actuelle */
  alert(video.currentTime);
  /* On redéfinit la position courante dans la lecture */
  video.currentTime=35;
  /* On reprend la lecture */
  video.play();
</script>
```

Canvas

Je n'ai encore pas eu l'occasion de jouer avec les éléments de type Canvas, qui permettent de dessiner point par point via une API JavaScript, mais les démos disponibles sur le Web sont **vraiment impressionnantes**. On imagine facilement les applications possibles. Il devient facile de dessiner en temps réel des graphiques animés. Tout cela sans Flash ni plugin, simplement de l'HTML5, un peu de CSS3 et du JavaScript !

API JavaScript enrichie

Tout d'abord de nouveaux sélecteurs très pratiques font leur apparition.

Trouver des éléments via l'API DOM

```
var el = document.getElementById('section1');
el.focus();

var els = document.getElementsByTagName('div');
els[0].focus();

var els = document.getElementsByClassName('section');
els[0].focus();
```

Vous connaissiez déjà `getElementById()` qui retourne un élément du DOM. `getElementsByTagName()` permet d'obtenir un tableau d'éléments correspondant à la balise passée en paramètre. `getElementsByClassName()` est l'équivalent pour la classe passée en paramètre.

Trouver des éléments via les sélecteurs CSS (API Selectors)

```
var els = document.querySelectorAll("ul li:nth-child(odd)");
```

```
var tds = document.querySelectorAll("table.test > tr > td");  
var el = document.querySelector("table.test > tr > td");
```

querySelector() et querySelectorAll() permettent respectivement de récupérer le premier élément ou tous les éléments qui correspondent au sélecteur CSS3 passé en paramètre.

Attributs personnalisés data-*

Ces attributs que vous pouvez définir sur n'importe quelle balise permettent de définir, stocker et récupérer des données personnalisées directement dans le DOM.

```
<div id="test" data-id="12" data-name="nico" data-screen-name="bounga"></div>  
  
<script type="text/javascript" charset="utf-8">  
  // Ajout d'un attribut via JS.  
  var el = document.querySelector('#test');  
  el.setAttribute('data-foo', 'bar');  
  
  var html = [];  
  // On récupère toutes les données personnalisées et on les affiche  
  for (var key in el.dataset) {  
    html.push(key, ': ', el.dataset[key], '<br>');  
  }  
  
  el.innerHTML = html.join('');  
</script>
```

Le framework Ruby on Rails dans ses dernières versions fait déjà largement usage de ces attributs pour mettre en place du JavaScript non intrusif et indépendant de la bibliothèque JavaScript utilisée.

Les listes de classes

La manipulation des classes CSS appliquées à un élément a également été largement améliorée. Plus besoin d'un framework pour connaître les classes appliquées, en ajouter, en supprimer...

```
<div id="main" class="shadow rounded"></div>  
  
<script type="text/javascript" charset="utf-8">  
  var el = document.querySelector('#main').classList;  
  el.add('highlight');  
  el.remove('shadow');  
  el.toggle('highlight');  
  
  el.contains('highlight'); // false  
  el.contains('shadow'); // false  
  el.classList.toString() == el.className; // true  
</script>
```

Cookies on stéroïds: localStorage et sessionStorage

Un moyen bien plus puissant que les cookies de stocker des données. Le localStorage permet un stockage permanent alors que le sessionStorage permet un stockage par onglet (ou session).

```
// Enregistre le contenu d'un textarea au clic sur le bouton  
saveButton.addEventListener('click', function () {  
  window.localStorage.setItem('value', area.value);  
  window.localStorage.setItem('timestamp', (new Date()).getTime());  
}, false);
```



```
// Préremplit le textarea en fonction de la valeur stockée en local
textarea.value = window.localStorage.getItem('value');
```

Bases de données WebSQL

Encore plus puissant, il est maintenant possible d'avoir une base de données SQL côté client. On peut y stocker des données utilisables hors-ligne, relatives à l'utilisateur qui pourraient donc être chargées sans faire appel au serveur. Très pratique pour stocker des préférences utilisateur par exemple.

```
var db = window.openDatabase("ma_base", "1.0", "description", 5*1024*1024); //5MB

db.transaction(function(tx) {
    tx.executeSql("SELECT * FROM test", [], successCallback, errorCallback);
});
```

API de cache

Cette API permet de rendre disponibles certaines ressources en mode hors-ligne grâce au cache applicatif.

Il suffit de déclarer votre balise HTML en précisant l'emplacement du manifeste :

```
<html manifest="cache.appcache">
```

Le manifeste ressemblerait lui à quelque chose comme :

```
CACHE MANIFEST
# version 1.0.0

CACHE:
/html5/src/logic.js
/html5/src/style.css
/html5/src/background.png

NETWORK:
*
```

On rend donc disponibles hors-ligne les ressources listées dans la section CACHE pour tous les réseaux.

Il ne reste plus qu'à piloter via JavaScript :

```
window.applicationCache.addEventListener('updateready', function(e) {
    if (window.applicationCache.status == window.applicationCache.UPDATEREADY) {
        window.applicationCache.swapCache();
        if (confirm('A new version of this site is available. Load it?')) {
            window.location.reload();
        }
    }
}, false);
```

Les sockets Web

HTML5 ouvre la possibilité d'avoir des communications bidirectionnelles. Le serveur **et** le client peuvent envoyer des données dès qu'ils en ont besoin. Il est même possible de le faire en simultané. Seules les données sont envoyées sans en-têtes ce qui réduit l'utilisation de la bande passante et améliore donc les temps de réponse.

Côté serveur il vous faudra souscrire à un service Web qui propose la gestion des Websockets ou vous en charger de votre côté avec **Socket.IO-Node** par exemple.

```
var socket = new WebSocket('ws://site.org/echo');

socket.onopen = function(event) {
    socket.send('Hello, WebSocket');
};

socket.onmessage = function(event) { alert(event.data); }
socket.onclose = function(event) { alert('closed'); }
```

Notre JavaScript d'exemple contacte le serveur et crée une WebSocket. Une fois la socket ouverte (onopen) le client envoie un message au serveur. Si un message est reçu du serveur sur cette WebSocket, le client l'affiche. Si la socket est fermée, un message sera affiché par le client.

On peut donc imaginer des outils comme des chats en temps réel, ultraréactifs et beaucoup moins gourmands en bande passante que des modèles basés sur un appel en boucle du client au serveur. Avec les WebSockets, c'est le serveur qui contacte le client pour lui dire qu'il y a des données à traiter !

Géolocalisation

Il est maintenant possible de localiser un utilisateur (s'il l'autorise) via GPS, 3G ou IP. Une fonctionnalité qui semble clairement avoir été mise en place pour les appareils mobiles en vue de créer des applications sociales avec géolocalisation.

```
if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(function(position) {
        var latLng = new google.maps.LatLng(position.coords.latitude, position.coords.longitude);
        var marker = new google.maps.Marker({position: latLng, map: map});
        map.setCenter(latLng);
    }, errorHandler);
}
```

Avec ce simple morceau de code, sans bibliothèque externe ou appel d'une API, une carte centrée (Google Map) sur la position de l'utilisateur sera affichée.

Accès à l'historique

L'un des plus gros soucis rencontrés par les développeurs avec AJAX est la perte de l'historique navigateur. Il est aussi impossible de bookmarker un contenu chargé en AJAX. Avec l'API d'historique vous pourrez facilement pallier ces problèmes en manipulant les URL et l'historique depuis JavaScript.

```
// Parcourir l'historique
window.history.back();
window.history.go(-1);

window.history.forward();
window.history.go(1);

window.history.length;

// Modifier l'historique
var stateObj = { foo: "bar" };

// Ajout d'un élément à l'historique, modification de l'URL, modification du titre
// rien n'est chargé, seuls l'historique et la barre d'URL sont impactés
history.pushState(stateObj, "titre", "foo/bar.html");
```

```
// Ici on remplace l'élément d'historique plutôt que d'en ajouter un
history.replaceState(stateObj, "titre 2", "foo/bar2.html");

// Info sur l'élément courant
history.state;
```

Un événement `popstate` est envoyé à chaque fois que l'entrée courante change. Si cet élément avait été ajouté via un appel à `pushState` ou `replaceState`, alors l'événement contiendra une copie de l'état qui avait été passé en créant l'entrée. Ça peut être intéressant si à chaque changement de page vous devez remettre du contenu dans son état initial :

```
window.addEventListener('popstate', function(event) {
    document.querySelector('h1').innerText = event.state.title;
});
```

Gestion des fichiers

Depuis HTML5, pour peu que l'utilisateur vous accorde les droits, vous pouvez écrire dans un système de fichier virtuel cloisonné (*sandbox*). Cette nouveauté offre de nombreuses possibilités pour les applications Web. Dans notre exemple, nous écrirons des logs sur le disque du client :

```
/* On crée un fichier disponible de manière permanente d'une taille estimée de 5Mo */
window.requestFileSystem(window.PERSISTENT, 5*1024*1024, function(fs) {
    fs.root.getFile('log.txt', {create: true}, function(fileEntry) {

        fileEntry.createWriter(function(writer) {
            writer.onwrite = function(e) { };
            writer.onerror = function(e) { };

            var bb = new BlobBuilder();
            bb.append('je log !');

            writer.write(bb.getBlob('text/plain'));
        }, errorHandler);
    }, errorHandler);

/* Gestionnaire d'erreur d'accès au système de fichier virtuel */
function errorHandler(e) {
    var msg = '';

    switch (e.code) {
        case FileError.QUOTA_EXCEEDED_ERR:
            msg = 'QUOTA_EXCEEDED_ERR';
            break;
        case FileError.NOT_FOUND_ERR:
            msg = 'NOT_FOUND_ERR';
            break;
        case FileError.SECURITY_ERR:
            msg = 'SECURITY_ERR';
            break;
        case FileError.INVALID_MODIFICATION_ERR:
            msg = 'INVALID_MODIFICATION_ERR';
            break;
        case FileError.INVALID_STATE_ERR:
            msg = 'INVALID_STATE_ERR';
            break;
        default:
            msg = 'Unknown Error';
            break;
    }
};
```

```
console.log('Error: ' + msg);  
}
```

Il est vraiment possible de gérer ce pseudosystème de fichiers comme un vrai système de fichiers. Les possibilités sont nombreuses et je vous recommande de lire  [cette documentation](#) sur le sujet.

Webworkers

Les Webworkers peuvent, en quelque sorte, être comparés au multithreading.  **L'API des Webworkers** permet d'exécuter des scripts en tâche de fond, sans bloquer le client. Un script s'exécutant dans un Webworker ne sera donc pas bloquant pour l'affichage de la page.

```
/* index.js: */  
  
/* On crée un worker */  
var worker = new Worker('worker.js');  
  
/* On définit son comportement s'il reçoit un message */  
worker.addEventListener("message", function(event) {  
    alert(event.data);  
}, false);  
  
/* On envoie un message */  
worker.postMessage('Nico');  
  
/* worker.js: */  
  
self.onmessage = function(event) {  
    self.postMessage("Bonjour : " + event.data + " !");  
};
```

CSS3

Les nouveaux sélecteurs CSS: nth-child(even), nth-child(odd), :not, first-child ?

Un nombre impressionnant de sélecteurs ont été ajoutés et ils s'avèrent vraiment pratiques :

```
.row:nth-child(even) {  
    background: #dde;  
}  
/* Tous les éléments impairs ayant pour classe "row" */  
.row:nth-child(odd) {  
    background: white;  
}  
  
/* N'ayant pas la classe "box" */  
:not(.box) {  
    color: #00c;  
}  
/* Tous les éléments non span */  
:not(span) {  
    display: block;  
}  
  
/* Premier h2 si celui-ci est le premier élément du site */  
h2:first-child { }  
  
/* Sélectionne le premier div fils des div ayant la classe "text" */  
div.text > div { }
```

```
/* Sélectionne les éléments header suivant directement un h2 */
h2 + header { }

/* Tous les inputs de type text */
input[type="text"]{ }
```

La gestion des polices

CSS3 nous ouvre la possibilité d'utiliser des polices non standards, qu'elles soient installées ou non chez le client :

```
@font-face {
  font-family: 'LeagueGothic';
  src: url(LeagueGothic.otf);
}

@font-face {
  font-family: 'Droid Sans';
  src: url(Droid_Sans.ttf);
}

header {
  font-family: 'LeagueGothic';
}
```

Gestion de colonnes

Combien de fois avez-vous dû répartir un contenu sur plusieurs colonnes ? Une tâche fastidieuse en CSS qui peut s'avérer compliquée à mettre en œuvre de manière portable et flexible. CSS3 apporte une solution élégante :

```
-webkit-column-count: 2;
-webkit-column-rule: 1px solid #bbb;
-webkit-column-gap: 2em;
```

Maîtrise de l'opacité

En CSS3, il est possible de définir les couleurs en RGBA. On a donc en plus des habituels canaux rouge, vert et bleu accès au canal "alpha" qui permet de définir l'opacité de la couleur.

```
/* Opacité à 75 % */
color: rgba(255, 0, 0, 0.75);
background: rgba(0, 0, 255, 0.75);

/* Opacité à 50 % */
hsla(240, 100%, 70%, 0.5);
```

Bordures arrondies

Je rêvais de cette possibilité depuis des années. Étant clairement un manchot de Photoshop, je me suis longtemps demandé pourquoi il n'était pas possible d'arrondir les angles des div avec une simple ligne de CSS. Oubli réparé, fini les images de fond pour les coins de vos blocs. CSS3 intègre la gestion des angles des bordures :

```
/* Spécifique à webkit (Safari, Chrome) */
-webkit-border-radius: 3px;
-webkit-border-bottom-right-radius: 5px;
-webkit-border-bottom-left-radius: 5px;

/* Spécifique à Mozilla (Firefox) */
```

```
-moz-border-radius: 3px;
-moz-border-radius-bottomright: 5px;
-moz-border-radius-bottomleft: 5px;

/* Future norme ? */
border-radius: 3px;
border-bottom-right-radius: 5px;
border-bottom-left-radius: 5px;
```

Dégradés

Encore une fois vous allez pouvoir oublier les images utilisées en arrière-plan pour simuler un dégradé. CSS3 le gère nativement :

```
/* Dégradé linéaire du haut vers le bas, de vert à blanc */
background: -webkit-gradient(linear, left top, left bottom,
                             from(#00abee), to(white),
                             color-stop(0.5, white), color-stop(0.5, #66cc00));

/* Dégradé radial avec des valeurs absolues */
background: -webkit-gradient(radial, 430 50, 0, 430 50, 200, from(red), to(#000));
```

Fonds redimensionnables

Il était possible, jusque-là, d'ajouter une image de fond à un élément. On pouvait l'afficher une fois dans sa taille originale et également répéter le motif sur l'axe des x et/ou y. Une amélioration est apportée par CSS3 puisque maintenant il est possible de dire comment doit se comporter l'image. On peut donc la forcer à prendre toute la taille d'un bloc, à être contenue dans le bloc, avoir sa taille originale ou encore définie à un pourcentage.

```
#logo {
  background: url(logo.gif) center center no-repeat;
  /* Valeurs possibles : auto / contain / cover / 100%
  background-size: cover;
}
```

Fonds multiples

CSS3 permet de définir plusieurs images de fond pour un même élément :

```
div {
  background: url(/images/logo.png) 10px center no-repeat,
             url(/images/stripes.png) 0 center repeat-x;
}
```

Ombrages

Vous aimez ajouter des ombres à vos textes et blocs, plus besoin d'images ! Encore une fois, CSS3 apporte une solution élégante et légère à ce cas de figure :

```
/* Ombre ajoutée à un texte utilisant potentiellement une police spécifique */
text-shadow: rgba(64, 64, 64, 0.5);

/* Ombre ajoutée à un bloc (div, etc) */
box-shadow: rgba(0, 0, 128, 0.25);
```

Transitions et transformation

Vous êtes habitué à ajouter des effets de transition aux actions qui se produisent dans votre application ? Alors je suppose que vous utilisez JavaScript pour créer l'effet visuel de transition d'un état à l'autre ! En CSS3 :

```
/*
TRANSITIONS
Les éléments #box auront un effet de transition lorsque le margin-left sera modifié
*/
#box {
  -webkit-transition: margin-left 1s ease-in-out;
}

/*
TRANSFORMATIONS
Rotation d'éléments du DOM
utile par exemple sur un :hover
*/
-webkit-transform: rotateY(45deg);
-webkit-transform: scaleX(25deg);
-webkit-transform: translate3d(0, 0, 90deg);
-webkit-transform: perspective(500px)

/*
ANIMATIONS
Fait "clignoter" tous les divs en augmentant la taille de leur police.
*/

@-webkit-keyframes pulse {
  from {
    opacity: 0.0;
    font-size: 100%;
  }
  to {
    opacity: 1.0;
    font-size: 200%;
  }
}

div {
  -webkit-animation-name: pulse;
  -webkit-animation-duration: 2s;
  -webkit-animation-iteration-count: infinite;
  -webkit-animation-timing-function: ease-in-out;
  -webkit-animation-direction: alternate;
}
```

Conclusion

Dans cet article, nous n'avons malheureusement pas eu le temps de faire un tour complet des nouveautés qu'offre HTML5 mais vous trouverez de nombreux sites qui traitent du sujet. Une des références incontournables est évidemment un site à l'initiative de Google,  **HTML5 Rocks!**.  **HTML5 Demos** présente des cas concrets d'utilisation.

Si vous comptez utiliser HTML5, il vous faut absolument jeter un œil au projet  **Modernizr** qui vous permettra de connaître les fonctionnalités supportées par le client. Cet outil vous facilitera la mise en place de solutions alternatives.

Il est indéniable que HTML5 a été pensé et taillé pour créer des applications et non plus de simples sites. HTML5 propose même un mode hors-ligne qui permet de surfer et interagir avec des parties du site sans connexion. Une fois une connexion rétablie, les données peuvent être synchronisées. Les possibilités offertes nativement sont désormais impressionnantes. Un vrai régal pour les développeurs d'applications Web. Pour les développeurs de sites vitrines

c'est aussi une bonne nouvelle, les intégrations graphiques, les effets de transition et de transformation sont supportés nativement.

Remerciements

Cet article a été publié avec l'aimable autorisation de **Synbioz**, l'article original peut être vu **sur le blog de Synbioz**.

Nous tenons à remercier **kdbella** et **ClaudeLELOUP** pour leur relecture attentive de cet article.