WEBDESIGNER
Front-end developer

CSS 3.0

CascadingStylesheets

Bruxelles Formation | CEPEGRA | Olivier Céréssia | 2016

webdesigner.cepegra-labs.be

PRÉAMBULE

Le contenu de ce cours est grandement inspiré de l'excellent ouvrage de Raphaël Goetter « CSS avancées – Vers HTML5 et CSS3 » que je vous recommande vivement.

Nous vivons en ce moment une époque formidable : les technologies CSS 3 actuellement en cours de développement prévoient de faciliter notre quotidien de concepteur web plus que vous ne pourriez l'imaginer. Les techniques exploitables en production sont déjà très séduisantes : polices exotiques, positionnement intuitifs, effets décoratifs ou d'animation, propriétés avancées ou encore gestion des tailles d'écran sont au programme de ce cours.

Les premiers brouillons de CSS 3 ont été publiés en 1999 et comptent environ une trentaine de modules chacun contenant plusieurs dizaines d'éléments (propriétés, sélecteurs, valeurs).

À ce jour, les modules finalisés ne sont pas nombreux et certains experts pensent même que toutes les règles éditées dans le brouillon ne verront pas le jour, dommage.

Mais ça reste tout à fait suffisant pour vous ouvrir les portes de centaines de nouvelles possibilités qui vont vous faciliter la vie et rendre vos projets web encore plus séduisants.

Le seul problème dans CSS 3 demeure sont interprétation sur les différentes plateformes actuelles et surtout (peut-on dire uniquement ?) sur Internet Explorer … tiens tiens ☺

Il reste néanmoins possible de profiter dès aujourd'hui du plein potentiel de CSS 3 et de préparer le terrain pour que notre ami IE suive le pas … dans sa version 10 ?

TABLE DES MATIÈRES

CSS3 : LA RÉVOLUTION

• Les préfixes propriétaires

Étant donné que les composants CSS 3 sont, pour la plupart, en cours d'élaboration, le souci de l'implémentation dans tous les navigateurs est un véritable casse-tête.

Pour contrer ce problème, deux solutions :

- Soit on attend bien sagement que chacun des modules soit validé par le W3C et utilisable comme toute autre règle CSS 2
- Soit on va puiser dans les préfixes propriétaires à chaque navigateur pour faire fonctionner les règles correctement d'un navigateur à l'autre

Sur la page suivante …

http://www.w3.org/Style/CSS/current-work

… vous retrouverez toutes les spécifications CSS 3 et leur état d'avancement. Quand une spécification a un petit « CR » vis-à-vis d'elle, c'est bon, vous n'avez plus besoin de préfixe !

TABLE OF SPECIFICATIONS

Ordered from most to least stable:

Completed	Current	Upcoming	Notes	
CSS Snapshot 2010	NOTE	–	Latest stable CSS	
CSS Snapshot 2007	NOTE	–		
CSS Color Level 3	REC	REC	See Errata	
CSS Namespaces	REC	REC		
Selectors Level 3	REC	REC		
CSS Level 2 Revision 1	REC	REC	See Errata	
CSS Level 1	REC	–	Unmaintained, see Snapshot	
Stable	**Current**	**Upcoming**	**Notes**	
Media Queries	PR	REC		
CSS Style Attributes	CR	PR		
Testing	**Current**	**Upcoming**	**Notes**	
CSS Backgrounds and Borders Level 3	CR	PR		
CSS Image Values and Replaced Content Level 3	CR	PR		
CSS Marquee	CR	PR		
CSS Multi-column Layout	CR	CR		
CSS Speech	CR	PR		
CSS Mobile Profile 2.0	CR	PR	Status unknown	
CSS TV Profile 1.0	CR	?	Status unknown	
Refining	**Current**	**Upcoming**	**Notes**	
CSS Transforms	WD	WD		
CSS Transitions	WD	WD		
CSS Values and Units Level 3	LC	CR		
CSS Print Profile	LC	?	Status unknown	

Sinon, il vous faudra utiliser les préfixes suivants :

- -moz- pour Firefox
- -ms- pour IE
- -o- pour Opera
- -webkit- pour Chrome et Safari
- -khtml- pour les moteurs KHTML (Linux)

On doit donc décrire plusieurs fois la même règle mais avec un préfixe différent à chaque fois.

• Les propriétés CSS 3

Nous n'allons pas balayer ensemble toutes les propriétés CSS 3 car certaines sont trop techniques, d'autres ne verront peut-être pas le jour et certaines ont un intérêt assez relatif pour le moment.

Nous nous concentrerons sur celles qui offrent déjà un aperçu correct à l'heure actuelle, que vous pourrez utiliser déjà dès demain et qui, même si elles sont mal interprétées, ne poseront pas de problème sur les navigateurs (IE ☺) à la traine.

LES NOUVEAUX SÉLECTEURS

• Le sélecteur adjacent général(p ~ p {…})

On connaissait le sélecteur adjacent classique (ex : p + p {…} – cible tous les paragraphes qui suivent directement un autre paragraphe).

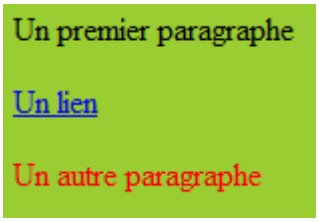
Avec CSS3, vous allez pouvoir utiliser le sélecteur adjacent général (p ~p {…}).

La différence avec le sélecteur classique (+) c'est que la version proposée par CSS3 ne se limite pas à la proximité directe : d'autres éléments peuvent se placer entre les paragraphes, le sélecteur fonctionnera quand même.

```
HTML
<p>Un premier paragraphe</p>
<a href="#">Un lien</a>
<p>Un autre paragraphe</p>

CSS
p ~ p {color:red;}
```

Ce qui nous donne comme résultat :



On arrive donc bien à cible le 2e paragraphe malgré le fait qu'une balise de lien soit venue se mettre entre les deux paragraphes.

La bonne nouvelle, c'est que ce sélecteur est supporté à partir de IE7, donc il est d'ores et déjà utilisable.

• Le sélecteur d'attribut – 3 nouvelles variantes

On connaissait le sélecteur d'attribut qui cible un attribut d'une balise html du genre :

```
a[href="index.html"] {…} /* Cible toutes les balises a qui ont pour lien
index.html */
```

Il y a moyen maintenant de cibler seulement une partie de la valeur qui se trouve dans un attribut.

Le sélecteur d'attribut qui vérifie le début de la valeur d'un attribut :

```
balise[attribut^="pouet"] {…} /* Cible la balise qui a pour valeur
d'attribut le mot « pouet » pour commencer – peu importe ce qui suit */
```

Le sélecteur d'attribut qui vérifie la fin de la valeur d'un attribut :

```
balise[attribut$=".pdf"] {…} /* Cible la balise qui a pour valeur
d'attribut le mot « .pdf » pour terminer – peu importe ce qui précède */
```

Le sélecteur d'attribut qui vérifie dans toute la valeur d'un attribut :

```
balise[attribut*="lol"] {…} /* Cible la balise qui a pour valeur d'attribut
le mot « lol » quelque part entre les " ", peu importe l'endroit */
```

La bonne nouvelle ?

Ces 3 nouveaux sélecteurs ne sont juste pas supportés sur IE6 mais partout ailleurs, ils fonctionnent parfaitement !

PSEUDO-CLASSES ET PSEUDO-ÉLÉMENTS CSS 3

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.

Depuis CSS 3, une convention d'écriture a été proposée par le W3C pour distinguer les pseudo-classes des pseudo-éléments.

Les pseudo-éléments s'écrivent dorénavant à l'aide d'un double deux-points (::first-line, ::first-child, ::after, ::before), mais vous pouvez toujours écrire cela comme avant (avec un seul deux-points).

• La pseudo-classe :empty

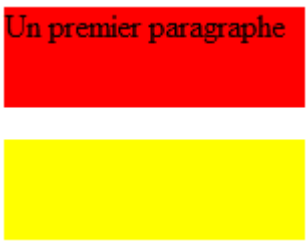
La pseudo-classe fait référence à un élément vide de tout contenu (balise ou texte).

Par exemple, sera concerné par le sélecteur :empty, tandis que , lol ou   ; ne le seront pas.

```
HTML
<p>Un premier paragraphe</p>
<p></p>


CSS
p {background:red; height:50px; width:150px;}
p:empty{background:yellow;}
```

Ce qui nous donne comme résultat :



Attention, ceci ne fonctionne pas dans les versions antérieures à IE9.

• La pseudo-classe :target

Si votre page web contient des ancres internes, c'est-à-dire des éléments nommés par des id, et des liens pointant vers ces éléments, alors la pseudo-classe :target désigne l'élément ciblé par l'ancre en question.

Dans l'exemple suivant, l'arrière-plan du titre <h2> devient jaune quand on clique sur le lien qui y mène :

```
HTML
<h2 id="joli_titre">Oh le joli titre</h2>
<a href="#joli_titre">Lien d'ancre qui ramène au titre</a>


CSS
h2:target {background:yellow;}
```

Ce qui nous donne comme résultat :

Raphaël Goetter a mis en place un site où il fait des tests de diverses propriétés CSS3 et techniques avancées qui nous concerne directement. :target y a sa place évidemment.

Le site :

http://ie7nomore.com/

Attention, ceci ne fonctionne pas dans les versions antérieures à IE9.

• La pseudo-classe :last-child

Le sélecteur :last-child fonctionne comme :first-child mais pour un élément qui est dernier enfant de son parent.

Ceci pourrait particulièrement être utile pour les menus pour lesquels la dernière item doit ne pas avoir de marge par exemple ou doit avoir un style différent.

```
HTML
<ul>
<li><a href="#">Accueil</a></li>
<li><a href="#">Portfolio</a></li>
<li><a href="#">CV</a></li>
<li><a href="#">Contact</a></li>
</ul>


CSS
body {background:yellowgreen;}
li {display:inline; background:red;}
ulli:last-child {background:yellow;}
```

Ce qui nous donne comme résultat :

Attention, ceci ne fonctionne pas dans les versions antérieures à IE9.

• La pseudo-classe :nth-child

Le sélecteur :nth-child s'applique au(x) n-ème(s) enfant(s) d'un élément en tenant compte de leur place en tant qu'enfant.

Les valeurs contenues au sein de la parenthèse de :nth-child() peuvent être :

- Un chiffre (entier positif ou négatif) – le premier enfant correspond à la valeur 1
- Une formule de type an+b, avec a et b deux chiffres ; n prendra toutes les valeurs à partir de 0
- Les mots-clés even ou odd qui symboliseront tous les fils pairs (even) ou impairs (odd) d'un parent, ce qui est idéal pour avoir des styles appliqués en alternance sur les lignes d'un tableau.

HTML

```
<ul>
<li><a href="#">Accueil</a></li>
<li><a href="#">Portfolio</a></li>
<li><a href="#">CV</a></li>
<li><a href="#">Contact</a></li>
</ul>
```

CSS

```
body {background:yellowgreen;}
li {display:inline; background:red;}
li:nth-child(2) {background:yellow;}
```

• La pseudo-classe :nth-of-type

Le sélecteur :nth-of-type s'applique au(x) n-ème(s) enfant(s) d'un élément en tenant compte du nombre de fois où ils apparaissent en tant qu'enfant. Pour le reste, elle fonctionne comme :nth-child

• Le pseud-élément ::selection

::selection permet de modifier la couleur de fond du texte d'une portion de contenu que le visiteur sélectionne en effectuant un cliquer-glisser par-dessus.

```
HTML
<p>Oh le joli paragraphe !</p>

CSS
p::selection {background:red;}
```

Ce qui nous donne comme résultat :

Attention, ceci ne fonctionne pas dans les versions antérieures à IE9.

D'autres sélecteurs ou pseudo-classes existent mais n'ont pas été abordées dans ce cours :

- :lang (permet de cibler un élément dont la langue correspond à une certaine valeur indiquée entre les parenthèses)

- :root (permet de cibler uniquement et toujours l'élément racine <html>. La seule différence avec le sélecteur <html> c'est que le poids de :root est supérieur)

- :not (cible un élément qui ne correspond pas au sélecteur déterminé entre les parenthèses)

- :nth-of-type (ne compte et ne sélectionne que les éléments du même type)

- :only-child (représente un élément qui n'a aucun frère)

- :only-of-type (renvoie tous les éléments qui sont seuls de leur type parmi leurs frères)

- :first-of-type et :last-of-type (désigne le premier ou le dernier élément de son type parmi ses frères)

- :enabled, :disabled et :checked (cible des éléments de formulaire. :enabled désigne les éléments actifs, :disabled les éléments inactifs et :checked les éléments cochés)

- :required et :optional (cible des éléments de formulaire. :required indique un champ considéré comme requis pour la validation du formulaire et :optional désigne un champ qui peut rester vierge)

- ::selection (permet de modifier la couleur de fond ou de texte d'une portion de contenu que le visiteur sélectionne en faisant un cliquer-glisser – une sélection)

LES FEUILLES DE STYLES POUR L'IMPRESSION

• Pourquoi une feuille de styles pour l'impression ?

L'avantage d'un périphérique « print »

Nous avons parfois la fâcheuse habitude de penser que le Web n'est bon à être restitué que sur un écran d'ordinateur. Pourtant, un grand nombre de documents web et d'informations en ligne sont parfaitement adaptés à l'impression. Cela facilite non seulement leur consultation, mais aussi leur conservation.

Le support imprimé offre des avantages non négligeables :

- Conserver des informations ;
- Faciliter la lecture hors ligne ;
- Disposer d'un document pouvant être annoté ou transmis physiquement

Au cours de votre existence d'internaute, vous avez très certainement imprimé des écrits tels que des réservations de vol ou d'hébergement, des documents à signer ou autres confirmations de commandes passées dans une boutique en ligne. Ce genre d'expérience laisse généralement des souvenirs mitigés, car le résultat obtenu via la commande Fichier>Imprimer est souvent bien aléatoire. Le plus souvent, on se retrouve avec trois pages à imprimer pour une seule vraie page de contenu.

Caractéristiques du format papier

Le Web sur écran n'est pas paginé, contrairement à l'impression sur papier. L'écran a ses propres règles, ses propres lois, qui s'expliquent tr !s facilement : non seulement un écran peut avoir différentes largeurs mais surtout les concepts de « page » et de « fin de page » n'existent pas : un document web utilise un ascenseur.

Si la restitution d'un contenu web sur un écran supporte peu ou mal les concepts de hauteur de page, les alignements verticaux et certaines valeurs de tailles, ces éléments sont en revanche parfaitement adaptés au format papier.

Grâce à leur gestion des périphériques de sortie, les feuilles de styles permettent de diffuser un même contenu HTML pour des supports différents. Il est donc possible de tirer parti au maximum des règles et mesure utilisées dans l'imprimerie (points, centimètres, sautes de page, césures, etc.) via un fichier CSS destiné uniquement à cet usage.

Les unités spécifiques

Toutes les unités de mesure recensées dans les spécifications et adaptées au Web sont susceptibles d'être employées en imprimerie. Toutefois, les unités les mieux adaptées à ce support paginé sont les points (pt), centimètre (cm), millimètres (mm) et pourcentage (%) .

• Gérer le support d'impression

Détecter le périphérique

Introduit dès la norme CSS 2, le print est destiné à un support paginé opaque ainsi qu'aux documents visionnés en mode Aperçu avant impression.

Deux méthodes ciblent spécifiquement ce support : un appel à un fichier CSS, ou des styles rédigés directement au sein d'un fichier CSS existant.

La première option nécessite une balise <link> placée dans le <head> et qui va inclure un fichier de styles consacré à l'imprimante par le biais de la syntaxe media="print".

```
HTML
<link href="print.css" rel="stylesheet" type="text/css" media="print" />
```

La seconde alternative s'appuie sur la règle @media incluse au sein de la feuille de styles globale, associée au mot-clé "print" et qui va servir de bloc de règles pour toutes les spécificités d'impression.

La feuille de styles « hôtesse » devra bien entendu couvrir explicitement ou non le support d'impression : une feuille déclarée sans media ou media="all" sera compatible, tandis qu'un fichier CSS s'adressant à un media="screen" ne pourra pas contenir de styles dédiés à l'impression.

```
CSS
@media print{/* ici les règles de styles pour l'impression */}
```

Notez qu'il est également envisageable d'importer une feuilel de styles CSS spécialisée au sein même d'une feuille commune à plusieurs supports :

```
CSS
@import url(print.css) print ;
```

Et cela ne fonctionne évidemment pas que pour les feuilles de styles destinées à l'impression.

Le principe est maintenant simple : dans votre feuille de style print.css, vous allez contredire toutes les règles initialement posées dans votre feuille de styles globale. Vous chargerez ensuite votre feuille de style print.css après votre feuille de styles globale pour que les styles énoncés dans print.css ne soient pas couverts par des déclarations générales :

```HTML
<link href="styles.css" rel="stylesheet" type="text/css" media="screen" />
<link href="print.css" rel="stylesheet" type="text/css" media="print" />
```

- # Méthodologie générale

Que faut-il imprimer ?

La première question qui se pose est celle de l'opportunité d'imprimer ou non certains éléments de la page. De nombreuses parties pertinentes à l'écran ne seront pas utiles sur le papier. Il s'agit alors de les identifier et de les masquer lors de l'impression à l'aide de notre propriété display et sa valeur none.

En général, les menus de navigation, les blocs publicitaires, les colonnes latérales sont des éléments que l'on n'affiche pas à l'impression, on pourrait donc imaginer ceci :

```CSS
#nav, #sidebar, #advertisement {display :none} ;
```

Plus intelligent encore, vous créez une classe spéciale pour cacher tous les éléments qui doivent l'être et vous appelez cette classe sur chaque élément HTML qui devra être caché lors de l'impression :

```HTML
<div id="nav" class="no-print"></div>
…
<div id="sidebar" class="no-print"></div>
…
<div id="advertisement" class="no-print"></div>
```

```CSS
.no-print {display :none} ;
```

MEDIA QUERIES : REQUÊTES DE MÉDIA CSS

Les media queries sont des règles CSS qui ne vont s'appliquer qu'à certains types de médias quand vous le leur demandez.

Vous pouvez décider de créer des styles différents selon que votre site soit consulté sur une tablette, un smartphone, un écran classique ou même lorsque vous décidez de l'imprimer.

C'est ça la force des media queries : offrir la possibilité de varier les styles selon le media dans lequel votre site est consulté ou imprimé.

CSS3 améliore la gestion des styles en fonction des périphériques de sortie et de leurs particularités en introduisant la notion de « requête de média » ou Media Query. Il devient possible de limiter la portée de styles CSS selon un environnement maîtrisé, par exemple uniquement sur les écrans dont la résolution est inférieure à 800 pixels.

Cette fonctionnalité, reconnue partout et prévue sur IE9, est particulièrement précieuse pour l'adaptation de son site au Web mobile.

• Syntaxe

Une Media Query se présente sous la forme d'une requête entre parenthèses, avec ou sans opérateurs logiques, associée au type de terminal.

Par exemple`, screen and (width :800px)` s'appliquera uniquement aux écrans dont la largeur occupe exactement 800 pixels.

La requête peut s'effectuer au sein du classique élément <link> :

```
HTML
<linkhref="color.css" rel="stylesheet" media="screen and (width:800px)"/>
```

Elle peut également prendre place dans une règle @ présente dans une feuille de styles :

```
CSS
@media screen and (width:800px) {…}
```

• Requête et préfixes

Couleurs, dimensions et orientation comptent parmi les critères de requêtes généralement définis au sein d'une requête de média. Ajoutons à ces requêtes deux préfixes facultatifs min et max faisant office de « plus petit ou égal à » ou « plus grand ou égal à ».

Critères	Description	Exemples
Color	Sans valeur associée, cette requête filtre les périphériques en couleur. Associée à une valeur, elle indique le nombre de bits de couleurs du périphérique.	(min-color :4)
Width	Largeur de la surface de rendu du périphérique de sortie.	(min-width :800px) désigne les terminaux dont la largeur est au moins de 800 pixels
Height	Hauteur de la surface de rendu du périphérique de sortie	(max-height :800px) désigne les terminaux dont la hauteur est inférieure ou égale à 800 pixels.
Device-width	Largeur physique du périphérique de sortie	(max-device-width :320px) désigne un terminal mobile dont la largeur est inférieure ou égale à 320 pixels, tel l'iPhone 3.
Device-height	Hauteur physique du périphérique de sortie.	(max-device-height :600px) désigne un terminal mobile dont la hauteur est inférieure ou égale à 600 pixels.
Orientation	Choisit un périphérique tourné dans le sens vertical (portrait) ou horizontal (landscape).	

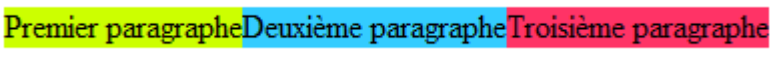
En pratique imaginons un code HTML simple :

```
HTML
<div>
<p>Premier paragraphe</p>
<p>Deuxième paragraphe</p>
<p>Troisième paragraphe</p>
</div>
```

Avec ce code CSS :

```
CSS
p {float:left; background:#CF0}
p + p {background:#3CF}
p + p + p {background:#F36}
```

Le résultat est le suivant sur notre écran large :



Et si maintenant, nous indiquons que nous aimerions que pour les résolutions inférieures à 600 pixels de large, les paragraphes ne soient plus alignés côte à côté mais bien placés les uns sous les autres, en 100% de large.

En CSS, vous ajouterez, après votre déclaration initiale :

```
CSS
@media screen and (max-width:600px) {
p {float:none;}
}
```

De cette façon, tous les éléments qui étaient floatés et mis côte à côte sont maintenant dé-floatés grâce à la propriété float et sa valeur none :

LES PROPRIÉTÉS CSS 3 LIÉES AU CONTENU

• Word-wrap

Dans un texte de contenu, quand un mot sans espace ni trait d'union est plus large que la dimension définie pour son parent, le comportement normal consiste à faire déborder le texte au-delà de la largeur normale du cadre, sans retour à la ligne et sans que la dimension du conteneur ne soit affectée.

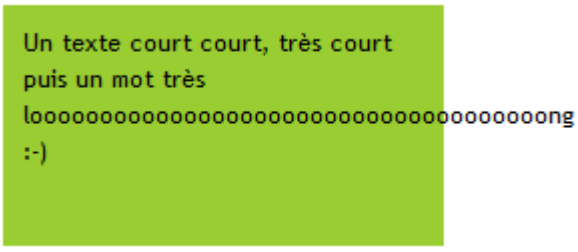
Il est toutefois possible de forcer la césure d'un mot long à l'aide de la propriété CSS word-wrap appliquée au parent et qui aura pour effet de couper le mot à un endroit arbitraire afin de provoquer un retour à la ligne.

Word-wrap	normal
Définit la césure d'un mot long et fait passer ce qui devait sortir du conteneur à la ligne.	Break-word

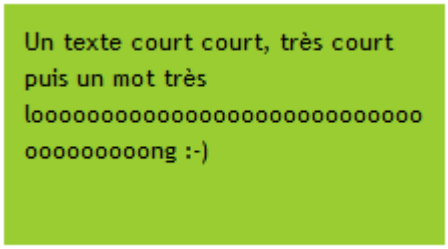
```
HTML
<div>Un texte court court, très court puis un mot très
looooooooooooooooooooooooooooooooooong :-)</div>

CSS
background:yellowgreen;
padding:10px;
width:200px;
height:100px;
word-wrap:break-word;
```

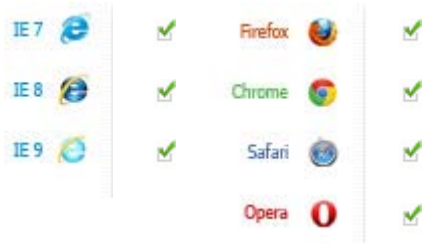
En temps normal, le mot « looooooooo..ng » serait sorti de la boite comme ceci :



Mais avec cette nouvelle propriété word-wrap, voici ce que ça donne :



Une bonne nouvelle maintenant : vous pouvez d'ores et déjà utiliser cette propriété CSS3 car elle est reconnue depuis … Internet Explorer 5.5 ! Pas de préfixe à utiliser pour cette propriété donc, cool !



• Text-overflow

Dans certains contextes particuliers, pour ne pas dénaturer la mise en page d'un document, nous sommes amenés à masquer les contenus d'un élément dimensionnés à l'aide de la règle overflow (hidden, scroll ou auto).

Les contenus qui débordent de ce bloc sont alors rognés et invisibles. Il peut être utile de laisser un indice visuel pour indiquer la présence de ce contenu masqué. C'est là qu'intervient la propriété `text-overflow` : associée à la valeur `ellipsis`, des points de suspension (…) sont ajoutés à l'endroit où le terme est rogné, mais il est possible de substituer cet indice par un autre de son cru à l'aide de la propriété `text-overflow-ellipsis`.

HTML
```
<div>Imaginons un bloc d'une certaine largeur avec un peu de texte, du joli texte et à la fin de ce texte, un lien qui, comme tous les liens, doit s'écrire sans espace blanc : <a href="http://www.ohmygoodweb.com/2012/04/le-project-glass-de-google-la-realite-augmentee-a-portee-de-vos-yeux/">http://www.ohmygoodweb.com/2012/04/le-project-glass-de-google-la-realite-augmentee-a-portee-de-vos-yeux/</a></div>
```

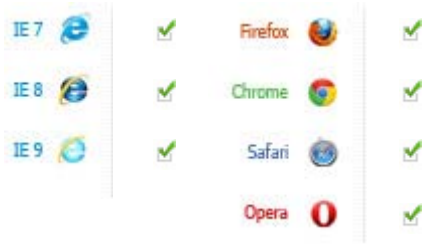
CSS
```
div {overflow:hidden; text-overflow:ellipsis; width: 200px; height:95px; background:yellowgreen; padding:5px;}
a {color:#FFF;}
```

Le résultat :

Imaginons un bloc d'une certaine largeur avec un peu de texte, du joli texte et à la fin de ce texte, un lien qui, comme tous les liens, doit s'écrire sans espace blanc : http://www.ohmygoodweb.com/2012…

Une autre excellente nouvelle, c'est que cette propriété text-overflow est supportée depuis IE 6 !

Donc y a plus qu'à en abuser quand vous en aurez besoin, elle a été d'ailleurs proposée par Microsoft à l'époque, comme quoi chez Microsoft, on ne fait pas toujours que de mauvais choix.



• Overflow-x et overflow-y

Pas grand-chose d'original à dire à propos de ces propriétés CSS3 que vous devriez comprendre sans même que je les explique.

En très court, disons que `overflow-x` n'agit que sur les débordements horizontaux tandis que `overflow-y` que sur les débordements verticaux. C'est assez clair je pense. Alors que leur papa, la propriété `overflow` agit sur tous les côtés de vois boîtes.

Les valeurs `visible, scroll, hidden et auto` restent d'actualité, rien de neuf à ce niveau-là.

Incroyable, mais au niveau de la compatibilité, Internet Explorer avait intégré cette propriété CSS dès la version 6 de son navigateur, donc de nouveau vous pouvez vous faire plaisir et utiliser dès aujourd'hui ces deux nouvelles variantes de la propriété overflow.

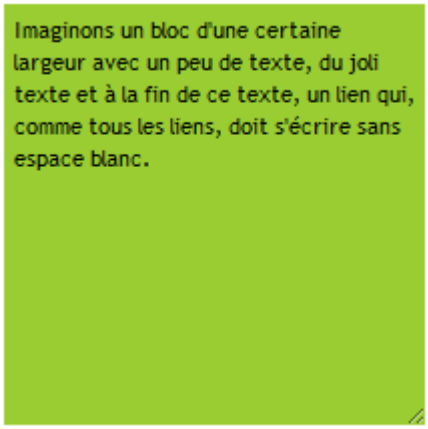
• Resize

La propriété CSS 3 resize permet de redimensionner un bloc à la souris (par un glisser-déposer du coin inférieur droit) !

Cette propriété trouve surtout son utilité pour les champs de formulaire `textarea` que le visiteur peut agrandir à loisir afin de disposer de suffisamment d'espace pour insérer son contenu sans être gêné.

4 valeurs sont tolérées : none, vertical, horizontal, both.

Quand vous aurez placé cette propriété sur une boite, vous trouverez, sur les navigateurs qui l'ont prévu, un petit élément graphique qui vous indique que la boite est étirable à partir de son coin inférieur droit :



Au niveau de la compatibilité, c'est plus compliqué pour cette propriété puisqu'Internet Explorer fait carrément l'impasse sur son utilisation à l'heure actuelle. Dommage.

LES PROPRIÉTÉS CSS 3 DÉCORATIVES

• @font-face

La règle @font-face avait déjà vu le jour en CSS 2, mais elle a été réintégrée à cette version de CSS.

L'avantage, c'est qu'à l'époque elle avait été intégrée par tous les navigateurs du marché et donc elle est déjà pleinement compatible à l'heure actuelle et ce sur tous les navigateurs ! (yipie)

Les différents formats de polices que vous allez devoir fournir pour que ça « passe » sur tous les navigateurs sont les suivants :

- .ttf pour Chrome, Firefox, Safari et Opera
- .otf pour Chrome, Firefox, Safari et Opera
- .eot pour Internet Explorer
- .svg pour l'iPhone

Théoriquement, le format .ttf, .eot et .svg sont suffisants pour couvrir un maximum de navigateurs.

Voici un exemple typique d'une police non standard (ou exotique), nommée ici Hit the Road :

```
@font-face {
font-family : "HitTheRoad";
/* Nom de la font – c'est comme ça que vous allez l'appeler + tard */

src : url('HitTheRoad-Regular.ttf') format("truetype") ,
      /* Ok partout sauf IE */
      url('HitTheRoad-Regular.eot') format("eot") ;
      /* Ok sur IE */
}
```

Attention
Dans la pratique, certaines fontes s'affichent de manière assez peu esthétique, présentant des arêtes visibles en forme de « marches d'escalier ». Ce phénomène, également connu sous le nom de aliasing (crénelage), dépend de la configuration de votre ordinateur, de la police affichée et surtout de la version de votre système d'exploitation (l'adoucissement – ou anticrénelage – est beaucoup mieux géré sur Mac OS ou un système récent).

• Border-radius

La propriété CSS 3 border-radius permet d'arrondir les angles d'un élément en indiquant la valeur de l'arrondi souhaité.

Ainsi, l'exemple suivant courbe les quatre coins de l'élément de classe bloc suivant un rayon de 10 pixels :

CSS
```
div {border-radius:10px; background:yellowgreen; width:100px;
height:100px;}
```

Il y a aussi moyen de courber indépendamment chaque coin de notre boite si vous codez la propriété border-radius sur 2 valeurs (haut/gauche + bas/droite et haut/droit + bas/gauche) ou 4 (haut/gauche, haut/droite, bas/droite, bas/gauche) . On commence toujours par le coin supérieur gauche et on tourne dans le sens des aiguilles d'une montre.
Un exemple avec les bords arrondis associés en diagonale (2 valeurs) :

CSS
```
div {border-radius:5px 20px; background:yellowgreen; width:100px;
height:100px;}
```

Un exemple avec les bords arrondis complètement dissociés (4 valeurs) :

CSS
```
div {border-radius:5px 10px 15px 20px; background:yellowgreen; width:100px;
height:100px;}
```

Un préfixe est nécessaire pour les anciennes versions de vos browsers, mais pas pour les nouvelles versions.

- # Opacity

Comme son nom l'indique, opacity agit sur la transparence d'un élément.

Les valeurs tolérées pour cette propriété doivent se situer entre 0 et 1, vous pouvez même aller dans les centièmes pour apporter plus de précision à votre transparence :

```
CSS
div {opacity:0.55; background:yellowgreen; width:100px; height:100px;}
```

J'ai donc appliqué pour cet exemple, une transparence de 55% au carré vert qui laisse donc maintenant bien voir l'image de fond qui se trouvait derrière lui (en l'occurrence ici le logo de Disney).

Attention
Un problème fréquemment rencontré avec la propriété CSS opacity, c'est qu'elle s'applique à l'élément dans son intégralité, y compris tous les éléments qui sont descendants, ce qui n'est pas sans poser de problèmes si l'on désire que l'un des enfants ne bénéficie pas de ce lourd héritage : **en pratique, et sans jouer avec des positionnements hors flux complexes, il est impossible d'annuler l'opacité de son ancêtre.**

Pour Internet Explorer de 6 à 8, il est possible d'émuler cette propriété opacity à l'aide de la fonction propriétaire filter. La syntaxe est sommaire pour les versions d'IE6 et 7 mais plus contraignante pour IE8 :

```
div {
      opacity:0.55; background:yellowgreen; width:100px; height:100px;
      }
```

- RGBa

Bien qu'il ne s'agisse pas ici d'une propriété CSS à proprement parler, évoquer RGBa semblait opportun à ce stade-ci de ce cours.

RGBa introduit la notion de transparence pour toutes les propriétés relatives aux couleurs : arrière-plans, bordures, couleurs de textes, etc.

Vous saviez évidemment que RGB se décompose en :
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 affecte une couleur bleue aux éléments div :

```
CSS
div {background:rgb(0,0,250); width:100px; height:100px;}
```

CSS 3 ajoute une quatrième composante (a), correspondant au degré de transparence (ici 55%) et dont la valeur est fixée entre 0 et 1 (comme pour opacity) :

```
CSS
div {background:rgba(0,0,250,0.55); width:100px; height:100px;}
```

Gros avantage par rapport à la propriété CSS Opacity : elle ne s'applique pas à un élément complet mais juste à une propriété (background, color, border, etc.).

Pour des raisons de compatibilité ascendante, il est toujours recommandé de prévoir une couleur en notation classique, pour assurer une alternative pour les vieux navigateurs qui ne reconnaitraient pas la syntaxe CSS 3 :

```
CSS
div {background-color:rgb(0,0,255);background-color:rgba(0,0,250,0.55);
width:100px; height:100px;}
```

On écrit d'abord la propriété qui fonctionne partout et, si le rgba fonctionne sur le navigateur, il prendra la propriété avec transparence et écrasera donc la déclaration précédente, par l'effet de la cascade.

• Box-shadow

La propriété CSS box-shadow a été incluse dans le module borders de CSS 3 et permet d'ajouter une ombre portée sur n'importe quel élément HTML.

Parmi les différentes valeurs utilisables, il est possible d'indiquer les décalages verticaux et horizontaux, ainsi que la force du dégradé, sans oublier la couleur évidemment.

La propriété s'applique sur la boîte de l'élément, et non sur sa bordure. L'ombrage n'affecte pas la taille de sa boîte.

En voici un exemple :

CSS
```
div {background:yellowgreen; width:100px; height:100px; box-shadow:8px 8px 8px #aaa;}
```

La première valeur indique le décalage horizontal, la deuxième le décalage vertical, la troisième l'adoucissement du dégradé et finalement la couleur (qui peut aussi être indiquée en rgba n'oubliez pas !).

Une dernière valeur, permet aussi de placer votre ombrage à l'intérieur de votre forme, il s'agit de inset :

CSS
```
div {background:yellowgreen; width:100px; height:100px; box-shadow:8px 8px 8px inset;}
```

2 ombres à la fois

Il est possible de placer 2 box-shadow d'un coup de cette façon :
```
box-shadow: -1px -1px 2px #000000, 1px 1px 2px #ffffff;
```

- # Text-shadow

La propriété CSS text-shadow permet, comme on peut s'en douter, de placer un ombrage d'une certaine couleur sous votre texte.

Comme pour le box-shadow, on peut spécifier le décalage horizontal et vertical ainsi que l'adoucissement de l'ombrage et sa couleur :

```
CSS
div {background:yellowgreen; width:100px; height:100px; text-shadow:1px 1px 0 white;}
```

Contrairement à box-shadow, la valeur inset n'est pas disponible pour la propriété text-shadow.

Au niveau compatibilité, ça passe vraiment mal sur IE, même en version 9.

• Les dégradés

De nouvelles propriétés permettent de réaliser et gérer des arrière-plans de teintes dégradées. Il s'agit de linear-gradient et radial-gradient.

Au niveau syntaxique ça donne :

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

- **x** et/ou **y** définissent la direction du dégradé. Exemple : "top right" ou "top left" – on peut aussi préciser l'angle du dégradé de cette façon -90deg,
- **couleur** : le nom, la valeur hexadécimal ou la valeur RGBa

```
body {
    background: linear-gradient(to bottom right, red, yellow);
};
```

Attention, la syntaxe pour écrire les dégradés est encore actuellement différente d'un browser à l'autre d'où la tartine de code ci-dessus pour que ça fonctionne sur tous les browsers.

• Background-size

CSS 3, via la propriété background-size, offre un moyen de spécifier les dimensions des images d'arrière-plan dans le but de les adapter à celles de l'élément sur lequel elles sont appliquées.Il suffit d'indiquer une valeur ou deux (horizontale et verticale) en pixels ou en pourcentage. Ces valeurs sont relatives aux dimensions du bloc, c'est-à-dire par défaut ses composantes de contenu et de marges internes. Les valeurs « cover » et « contain » permettent un « effet stretch » de l'image pour qu'elle s'adapte au conteneur dans lequel elle a été placée en background. Cover est indiqué pour les sites avec des background en fullscreen. Ci-dessous, un exemple qui redimensionne l'arrière-plan de manière à ce qu'il recouvre exactement tout l'élément :

```
CSS
div {width:300px; height:300px; background:url(wakfu.jpg) no-repeat;
background-size:100% 100%; border:1px solid gray}

HTML
<div></div>
```

Ci-dessous, un exemple avec la même image mais où chaque image prend 50% de la largeur et la valeur auto sur l'axe vertical pour conserver le ratio de l'image :

```
CSS
div {width:300px; height:300px; background:url(wakfu.jpg); background-
size:50% auto; border:1px solid gray}
```

• Arrière-plans multiples

CSS 3 permet d'afficher plusieurs images d'arrière-plan sur un même élément en cumulant les valeurs au sein des propriétés background-image, background-position et background-repeat, ces valeurs étant simplement séparées par une virgule.

Le résultat est similaire à des calques d'un logiciel graphique tel que Photoshop : la première image déclarée dans la liste sera au premier plan. Si une couleur de fond est déclarée, elle sera toujours reléguée au dernier plan.

Dans l'exemple ci-dessous, le logo HTML5 a été placé au milieu d'un conteneur et par-dessus une image de fond répétée :

```
CSS
div {width:300px; height:300px;
background:url(wakfu.jpg);background:url(html5.png) center center no-
repeat, url(bg_01.gif);}

HTML
<div></div>
```

Lorsque plusieurs images d'arrière-plan sont déclarées, l'ordre d'apparition dans la règle détermine leur empilement : la première de la liste apparaît en haut de la pile, puis la deuxième, et ainsi de suite jusqu'à la dernière qui occupera la couche la plus basse.

CSS Transformations

La propriété CSS 3 transform permet d'appliquer des transformations en deux dimensions sur un élément : rotation, décalage, zoom, déformation et perspective.

N'oubliez pas vos préfixes pour cette propriété CSS 3, c'est non négociable.

Scale, scaleX ou scaleY

Cette fonction agrandit ou réduit les dimensions d'un élément selon un ratio : une valeur inférieure à 1 aura pour conséquence de rapetisser l'élément, un chiffre supérieur à 1 va l'agrandir. La place occupée dans le flux demeure inchangée : si l'élément est agrandi, ses frères ne seront pas poussés en conséquence.

```
CSS
.milieu:hover {transform:scale(0.5); }

HTML
<imgsrc="html5.png" />
<imgsrc="html5.png" class="milieu" />
<imgsrc="html5.png" />
```

Il est également possible de ne faire agir le scale que sur l'un des deux axes (scaleX ou scaleY) :

```
CSS
.milieu:hover {transform:scalex(0.5); }
```

Ici avec un scaleX de 0.5 :

Rotate

Il est maintenant possible de faire tourner un élément sur lui-même avec la propriété rotate associée à tranform. On indique les valeurs de rotate en degrés (deg) ou en radians (rad).

```css
.milieu:hover {transform:rotate(45deg); }
```

Skew, skewX ou skewY

Skew modifie la perspective de l'élément en « tirant » sur ses coins de manière à obtenir au final une forme parallélépipédique. La première valeur concerne l'axe des X et la seconde l'axe des Y (si vous utilisez la forme raccourci skew – sinon vous pouvez utiliser skewX et skewY) :

```css
.milieu:hover {transform:skew(-25deg, 10deg); }
```

Translate

Transalte sert à faire bouger l'élément sur les axes X et Y (effectuer une translation).

```
CSS
.milieu:hover {transform:translate(100px, 100px); }
```

Propriété raccourcie

Pour finir en beauté, sachez que vous pouvez combiner toutes les propriétés ci-dessus qui font toutes partie de la propriété transform. Il suffit de les écrire bout à bout, simplement séparées par un espace, par exemple :

```
.milieu:hover {transform:scale(0.5) rotate(45deg) translate(20px,20px)
skew(-10deg, 25deg); }
```

• CSS Transitions

Vous en aviez rêvé ? Ils l'ont fait ! Il est maintenant possible d'animer vos pages web sans passer par du flash ou d'autres procédés javascriptesques. Grâce aux dernières évolutions du langage et au module CSS 3 Transitions, il est maintenant possible de réaliser des transitions basiques à l'aide de CSS dans les navigateurs très récents … et même dans IE 9 ! (yipie)

Le principe de base d'une transition CSS 3 est de permettre un passage en douceur de l'ancienne vers la nouvelle valeur d'une propriété CSS lorsqu'un événement est déclenché :

- Soit via une pseudo-classe telle que :hover, :focus ou :active ;
- Soit via JavaScript.

N'oubliez pas les préfixes pour cette propriété CSS 3 !

Concrètement, vous pouvez jouer sur 4 éléments lors d'une animation avec transition.

2 obligatoires :

- La (ou les) propriété(s) à animer (transition-property) ;
- La durée de l'animation (transition-duration).

2 facultatives :

- L'accélération de l'animation (transition-timing-function) ;
- Le délai avant que la transition ait effectivement lieu (transition-delay).

Transition-property

La propriété transition-property accepte trois valeurs :

- All (valeur par défaut) : toutes les propriétés possibles seront animées ;
- Propriété : le nom d'une propriété CSS pouvant être animée ;
- None : aucune propriété ne sera animée

Transition-duration

La propriété transition-duration indique la durée de la transition. Si plusieurs propriétés ont été définies à l'aide de transition-property, il est possible de préciser leurs valeurs en les séparant d'une virgule.

Les deux unités de temps acceptées sont :

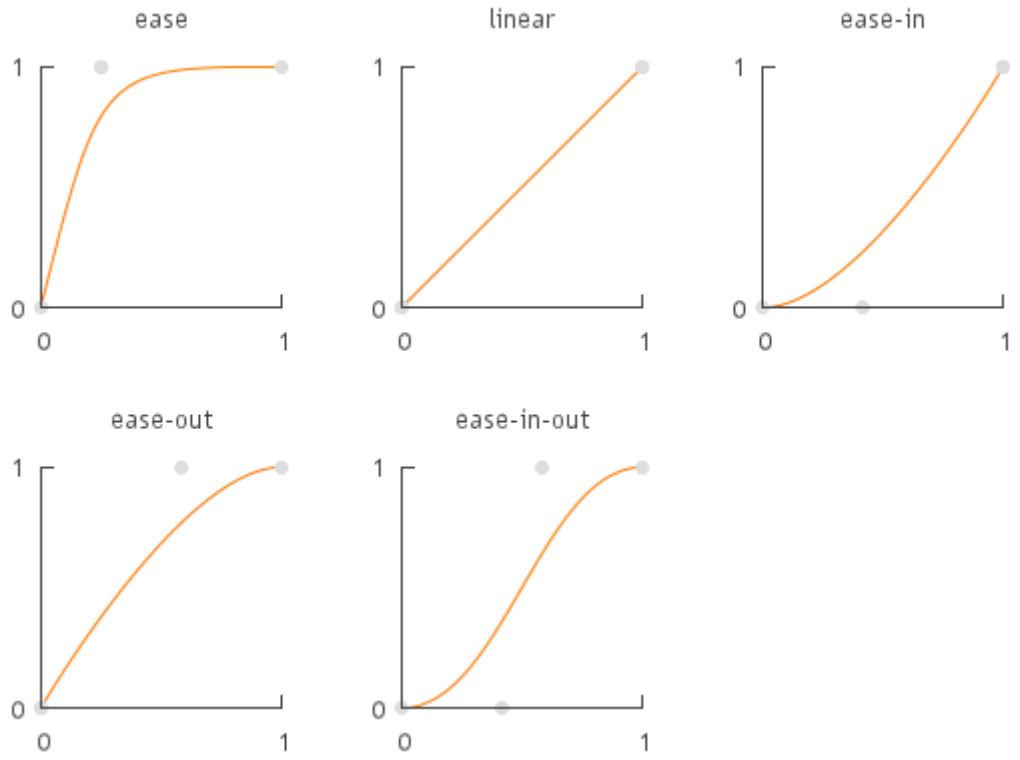
- s : la seconde ;
- ms : la milliseconde.

Transition-timing-function

La propriété transition-timing-function détermine la fluidité de l'animation. Les mots-clés suivants sont admis :

- ease : rapide sur le début et ralenti sur la fin ;
- linear : (valeur par défaut) vitesse constante sur toute la durée de l'animation ;
- ease-in : lent sur le début et accélère de plus en plus sur la fin ;
- ease-out : rapide sur le début et décélère sur la fin ;
- ease-in-out : le départ et la fin sont lents.

Une petite représentation sous forme de courbes :



Transition-delay

La propriété transition-delay indique la durée avant que se joue l'animation.

Les deux unités de temps acceptées sont :

- s : la seconde ;
- ms : la milliseconde.

Forme raccourcie

Comme c'était le cas avec tranform, transition bénéficie aussi d'une écriture raccourcie où il suffit d'indiquer les différentes valeurs des 4 propriétés ci-dessus séparées par un espace :

```
img {transition:all .3s ease .5s;}
img:hover {background:red;}
```

Notez que la transition se place **toujours sur l'état original** et pas sur l'état :hover sinon l'effet est un peu moins joli ☺

Propriétés compatibles

Toutes les propriétés CSS ou valeurs ne sont pas prévues pour bénéficier de transitions, mais la liste qui suit offre de belles possibilités :

- les couleurs ;
- toutes les valeurs numériques ;
- les transformations (avec des limitations) ;
- les dégradés (avec des limitations) ;

Voici un tableau complet des propriétés susceptibles d'être « transitionnées » :

PROPRIÉTÉS CSS	PROPRIÉTÉS CSS
Background-color	Margin-left
Background-position	Margin-right
Border-bottom-color	Margin-top
Border-bottom-width	Max-height
Border-color	Max-width
Border-left-color	Min-height
Border-left-width	Min-width
Border-right-color	Opacity
Border-right-width	Outline-color
Border-spacing	Outline-offset
Border-top-color	Outline-width
Border-top-width	Padding-bottom
Border-width	Padding-left
Bottom	Padding-right
Clip	Padding-top
Color	Right
Crop	Text-indent
Font-size	Text-shadow
Font-weight	Top
Grid	Vertical-align
Height	Visibility
Left	Width
Letter-spacing	Word-spacing
Line-height	z-index
Margin-bottom	

Attention

Les transitions ne fonctionnent pas avec les pseudo-éléments ::before et ::after.

• Compatibilité et conclusion

Les fonctions de transition sont encore un peu jeunes pour être exploitées en production : non seulement elles ne seront prises en compte qu'à partir d'IE9, mais même sur Firefox, il est nécessaire de disposer de la version 4 pour les voir en action.