

Collège
Ressources Informatiques

HTML5 et CSS3

**Maîtrisez les standards
des applications Web**

Luc VAN LANCKER

Téléchargement
www.editions-eni.fr

INFORMATIQUE TECHNIQUE



HTML5 et CSS3

Maîtrisez les standards des applications Web

Luc VAN LANCKER



Résumé

Ce livre sur le **HTML5 et CSS3** s'adresse à toute personne appelée à développer, mettre en place, faire vivre un site Web. En effet, pour débuter mais surtout pour progresser dans la conception de sites, il faut inévitablement passer par une bonne compréhension et une bonne maîtrise du **code source des applications Web**.

Le livre est conçu comme un réel outil de formation, pédagogique de la première à la dernière page, **abondamment illustré** d'exemples et de captures d'écran et constamment à l'affût des éléments réellement pratiques pour le webmestre.

Sont ainsi passés en revue le **HTML** (dans sa dernière version et ses nombreuses nouveautés), les **feuilles de style** avec l'avancée spectaculaire des **CSS3** en termes de présentation des pages Web et quelques éléments de **JavaScript**... Cet ouvrage n'est surtout pas une encyclopédie exhaustive de ces différentes techniques mais un **parcours structuré** de celles-ci. Il fournit aux concepteurs débutants, voire plus confirmés, les règles rigoureuses mais essentielles de la conception professionnelle d'un site Web. En effet, l'auteur s'est attaché à encourager l'élaboration d'un code **respectueux des prescriptions du W3C** et particulièrement de la séparation du contenu (HTML) et de la présentation (feuilles de style CSS) comme le préconise plus que jamais le HTML5.

Les nombreuses nouveautés abordées ne sont pas prises en compte que par les dernières versions des navigateurs (Internet Explorer 9, Firefox 3.6 et 4, Google Chrome ou Safari) mais l'auteur a été particulièrement attentif à fournir un code compatible avec des navigateurs moins évolués afin de pouvoir bénéficier dès à présent de ce pas important dans la conception des applications Web.

Les nombreux exemples du livre sont en téléchargement sur www.editions-eni.fr. Pour les lecteurs désirant reproduire les exemples à l'identique, les images et autres compléments y sont également à leur disposition.

Les chapitres du livre :

Avant-propos - Présentation du Htlm5 - Premiers éléments de feuilles de style - Le texte - La structuration du document - Les liens - Les tableaux - Les images et arrières-plans - Les balises sémantiques et d'organisation - Les formulaires - Le multimédia - Le dessin 2D - La géolocalisation - Présentation des feuilles de style - Notions de base des CSS - La police de caractères - Le texte en CSS - Les listes et les tableaux - Les arrière-plans - Les propriétés de boîte - Les pseudo-classes - Les propriétés d'affichage - Les propriétés d'impression - Les feuilles de style CSS3

L'auteur

Dès les débuts d'Internet, **Luc Van Lancker**, enthousiasmé par l'idée de communication universelle que véhiculait ce concept, s'est complètement investi dans ce domaine. C'est un formateur passionné, très au fait des nouvelles technologies liées au Web et grand pédagogue.

Ce livre numérique a été conçu et est diffusé dans le respect des droits d'auteur. Toutes les marques citées ont été déposées par leur éditeur respectif. La loi du 11 Mars 1957 n'autorisant aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les "copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective", et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, "toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayant cause, est illicite" (alinéa 1er de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal. Copyright Editions ENI

Ce livre numérique intègre plusieurs mesures de protection dont un marquage lié à votre identifiant visible sur les principales images.

Avant-propos

L'écriture de ce livre sur le Html5 et le CSS3 a été une entreprise très motivante. Le Html n'avait pas connu d'évolution majeure depuis 1998 avec le Html 4.0. Une éternité à l'échelle du développement du Web et d'Internet ! Il est important d'évaluer les évolutions et apports de cette nouvelle version du langage qui est à la base de cet outil devenu quotidien qu'est le Web. De leur côté, les feuilles de style CSS3 constituent une réelle avancée au niveau du graphisme des pages avec, par exemple, les bords arrondis, la transparence des images, les ombres ou la typographie.

Cet ouvrage sur le Html5 et les CSS3 se veut avant tout un ouvrage structuré et formatif plutôt qu'une exploration exhaustive des nouveautés apportées par ces deux technologies. Il vise deux publics bien distincts :

- Les débutants pourront appréhender les bases du langage source des pages de la toile au fil des chapitres grâce à de nombreux exemples complets, illustrés à chaque fois par des captures d'écran et la possibilité de les visionner sur leur écran, via l'espace de téléchargement consacré à cet ouvrage. Ils pourront ainsi entamer leur apprentissage de la conception de pages Web par la dernière évolution du Html qui est d'ores et déjà opérationnelle dans de nombreux navigateurs de la dernière génération et qui sera le standard des années futures.
- Ce livre s'adresse aussi aux concepteurs familiarisés au Html et qui souhaitent découvrir les apports et nouveautés du Html5 et des CSS3. Les différences par rapport au Html 4.0, parfois subtiles mais déterminantes, sont mises en évidence tout au long de l'ouvrage. Ils pourront aussi découvrir les réelles nouveautés du Html5 avec les chapitres dédiés aux nouveaux champs de formulaires, au dessin 2D, aux nouvelles balises audio et vidéo et à l'entrée du JavaScript avec, par exemple, la géolocalisation de l'utilisateur. L'important chapitre sur les CSS3 les convaincra de la révolution qu'elles vont apporter dans le design de la page, les débarrassant du passage obligé par des programmes graphiques.

Cet ouvrage marque également un tournant dans l'apprentissage du code source des pages de la toile. En effet, le Html5 fait table rase de tout aspect de présentation pour se consacrer exclusivement à la structure du document. Il n'est donc plus possible en Html5 d'aligner du texte ou autres éléments ou de dessiner des bordures de tableaux. Terminé aussi l'ajout de couleur au texte ou en arrière-plan. Le Html5 se décline désormais de façon austère, en noir et blanc ! Tous les éléments de présentation doivent obligatoirement être repris par les feuilles de style. Il n'est plus possible de concevoir un apprentissage de manière traditionnelle avec une partie exclusivement consacrée au Html et une seconde partie distincte pour les feuilles de style. Ces deux éléments sont maintenant intimement liés. Ainsi, il nous a semblé utile, voire indispensable, d'introduire dès l'apprentissage du Html des éléments basiques relatifs aux feuilles de style. Cela permettra à l'apprenant de concevoir ses premières applications d'une manière plus réaliste et plus attrayante.

Entrons plus profondément dans le détail des chapitres. La partie Html débute par les éléments basiques et traditionnels de l'apprentissage du code source. Le Html5 est en premier lieu une évolution du Html 4.0. Il reste donc un langage de balises. Les premiers éléments de l'apprentissage progressif concernent le texte, la structuration du document avec les titres et les listes, les tableaux, les images et leur insertion, sans oublier ce qui fait l'essence du langage Html c'est-à-dire les liens. Une série de chapitres aborde ensuite les nouveautés introduites par le Html5.

- Les nouvelles balises sémantiques qui permettent d'organiser le code source des applications afin d'en assurer une meilleure lisibilité.
- Il faut bien avouer que les formulaires traditionnels comme la ligne ou la zone de texte, les boutons radio, à choix multiples ou autres boutons de commandes entraînent une lassitude certaine dans l'aspect visuel de la page. Les apports du Html5 vont révolutionner les formulaires avec des lignes de texte avec suggestions, des calendriers, des compteurs numériques, des curseurs, une validation sans passer par le JavaScript et bien d'autres surprises à découvrir dans le chapitre qui leur est consacré.
- Les nouvelles balises audio et vidéo permettent désormais de lire de façon native les fichiers multimédia sans devoir passer par des plugins divers, souvent sources de complications.
- Le dessin 2D ouvre de nouvelles perspectives car il est désormais possible, par exemple, d'ajouter des graphiques directement par le code, sans passer par des captures d'écran de programmes comme Excel.
- Pour terminer, il fallait souligner que le Html5 marque aussi l'introduction du JavaScript comme un acteur privilégié dans la conception des applications Web. Ce dernier point est illustré par la géolocalisation de l'internaute.

Poursuivons avec la partie dédiée aux feuilles de style CSS. Dans un apprentissage progressif et structuré, il n'est pas possible d'ignorer les désormais élémentaires CSS1 et CSS2. Plusieurs chapitres sont ainsi consacrés aux feuilles de style relatives à la police de caractères, au texte, aux listes et tableaux, à la couleur, aux marges intérieures et extérieures, aux pseudo-classes et aux feuilles de style relatives à l'affichage ou à l'impression. Cette partie se termine

par les spécifications CSS3 qui vont profondément modifier l'aspect visuel de la navigation sur la toile. Ce chapitre ne manquera pas d'intéresser ceux qui, de près ou de loin, prennent en charge l'aspect graphique des sites. Parmi les apports de ces feuilles de style CSS3, citons :

- Les bords arrondis qui permettent de sortir de la présentation rectangulaire et angulaire des divisions.
- Les ombres que l'on peut apporter au texte ou aux images sans passer par une application graphique.
- Les polices que le concepteur pourra enfin choisir librement, marquant ainsi l'introduction tant attendue de la typographie dans la conception des pages web.
- La présentation d'un contenu textuel en colonnes.
- Les bordures imagées.
- Les arrière-plans multiples.
- Les dégradés de couleurs.
- La possibilité de jouer sur l'opacité ou la transparence d'éléments comme les images.

À n'en pas douter, cet ouvrage est dense mais présenté dans un langage que nous avons voulu clair, formatif et pédagogique.

En tant qu'auteur, je dois avouer que j'ai pris un énorme plaisir à écrire ce livre sur le Html5 et les CSS3. Je fais le souhait que ce plaisir soit partagé avec les lecteurs.

Bref historique du Html5

Nous débutons cet historique en 1998 avec la finalisation de la spécification Html 4.0. Mis à part une légère révision en 1999 avec le Html 4.01, plus rien de neuf à l'horizon de ce langage premier du Web. Le W3C (*World Wide Web Consortium*), l'organisme qui gère les standards d'écriture sur la toile, n'a-t-il pas annoncé à l'époque que le Html dans sa version 4.0 ne connaîtrait à l'avenir plus de développement, en bref que le Html était mort ?

Le W3C avait aussi édité en 1998 le XML (*eXtensible Markup Language* ou, en français, le langage extensible de balises) pour gérer de façon structurée des données de type texte, l'aspect extensible permettant à l'utilisateur de définir son propre langage avec ses éléments distincts. Le W3C recommandait à l'époque fortement le XML pour exprimer des langages de balisages spécifiques.

Une des premières applications concrètes du XML fut la reformulation du Html 4.0 selon la syntaxe stricte et formaliste du XML. Le Xhtml 1.0 apparut ainsi en 2000. Si les développeurs ont unanimement salué la rigueur que le Xhtml apportait à l'écriture du code, il faut bien constater que la migration vers le Xhtml n'a pas remporté le succès souhaité et que bon nombre de concepteurs sont restés fidèles au Html 4.0 transitional, plus souple.

Très rapidement, le W3C mit en chantier l'ambitieux projet du Xhtml 2.0 qui devait faire table rase du passé en matière de publication sur la toile. Cette position induisait que le Xhtml 2.0 ne serait pas rétrocompatible avec ce qui existait à l'époque et donc avec le Html.

Cette absence de rétrocompatibilité a engendré un mécontentement certain auprès des firmes qui développaient les navigateurs.

Ce qui eut pour conséquence la création d'un groupe dissident, le WHATWG (*Web Hypertext Application Technology Working Group*). Ce groupe de travail se présente notamment comme une réponse à la lenteur du développement des standards par le W3C et au caractère trop fermé de son processus interne d'élaboration de spécifications. Il se compose à l'origine de représentants de firmes comme Mozilla, Opera et Apple dont les navigateurs Firefox, Opera et Safari sont bien connus. Ils furent rejoints ensuite par Google. Leur position était diamétralement opposée à celle du Xhtml 2.0 en travaillant de façon pragmatique sur base des implémentations actuelles et donc du Html 4.0. Leurs premières réalisations portaient sur les Web Forms 2.0 pour renouveler les formulaires et le nouveau concept de publication sur le web, Web Apps 1.0 soit des applications Web.

Parallèlement le Hhtml 2.0, même si un brouillon (*working draft*) paraît en juillet 2006, connaît un développement particulièrement laborieux et la rumeur rapporte de nombreuses dissensions au sein de ce groupe de travail. En outre, les différents navigateurs précités boudaient délibérément le Html 2.0.

Il apparaissait clairement que le Xhtml 2.0 connaissait de gros problèmes. Sir Tim Berners-Lee, inventeur du Web et président du W3C, décida fin 2006, de rouvrir un groupe de travail sur le Html qui, de façon pragmatique, devait reprendre la suite des travaux du WHATWG. Ainsi, il y eut pendant tout un temps, le groupe de travail sur le Xhtml 2.0 et celui sur le Html5. Ce dernier publia le 22 janvier 2008, un premier brouillon de travail qui a connu de nombreuses évolutions depuis.

Mais les jours du Xhtml 2.0 étaient comptés. Fin 2009, le même Sir Tim Berners-Lee annonce la dissolution du groupe de travail sur le Xhtml 2.0 et l'abandon définitif de ce dernier.

On annonce pour 2012, la *candidate recommandation* du Html5 pour 2012 et, étant donné l'ampleur des nouveautés annoncées, la recommandation finale pour... 2022.

L'annonce du Html5 a fait grand bruit (un buzz dans le vocabulaire moderne). Cet intérêt s'est rapidement concrétisé à différents niveaux. Google a incorporé très rapidement, soit à partir de la version 5 de Chrome des éléments du Html5. Ses autres comparses du WHATWG également avec les versions récentes de Firefox, Safari et Opera. Même Microsoft, pourtant souvent à la traîne en matière d'adoption de nouveautés et de standards incorpore le Html5 dans la version 9 d'Internet Explorer. De leur côté, les smartphones adoptent également le standard Html5, que ce soit Apple avec son iPhone, RIM avec son BlackBerry, Google avec son Google Phone, etc. Est-ce encore une évolution ? N'est-ce pas une révolution ?

Parallèlement, une reformulation du Html5 selon les règles et la syntaxe du XML est en cours d'élaboration. Celle-ci porte le nom de Xhtml5.

De cette gestation mouvementée du Html5, on retiendra :

- Que l'on ne parle plus de pages Web mais du concept plus large d'applications Web.
- Que le Html5 est une évolution (importante certes) du Html 4.0.
- Que le Html5 a été conçu directement par les navigateurs les plus innovants du marché comme Firefox, Safari Opéra et Google. Ce qui est la garantie d'une adoption rapide des standards du Html5.
- Que l'on peut s'attendre avec le Html5 à de grands changements concernant les formulaires (voir les *Web Forms 2.0*).
- Qu'après plus d'une décennie sans réelles nouveautés, le Html5 correspond à un réel besoin des concepteurs

pour renouveler l'interface des applications Web.

Lignes de force du Html5

Avant de parcourir les nouveautés du Html5, dégageons quelques lignes de force de cette nouvelle version du langage Html.

- Le Html5 est une évolution du Html. Il en reprend les grands principes, quitte à en améliorer certains aspects. Il est conçu pour assurer une rétrocompatibilité avec ce qui a été fait jusqu'à présent, en termes de publication sur le Web.

Même si le codage est moins formaliste que le Xhtml5, l'exigence d'un code propre, respectueux des règles fondamentales du balisage continue d'être d'actualité.

- Le principe de la séparation du contenu et de la présentation reste de mise et se voit même renforcé. L'abandon de toute une série de balises et d'attributs de présentation (point F du présent chapitre) en est la meilleure preuve. Les feuilles de style CSS sont des partenaires indissociables du Html5. Celui-ci espère qu'avec la montée en puissance des nouvelles feuilles de style CSS3, l'interface de l'utilisateur soit complètement transformée dans les années à venir.

La simplification du code et le désir d'éviter toutes complications souvent inutiles sont deux principes suivis dans l'élaboration du Html5. On retrouve cette simplification et ce pragmatisme par exemple, dans un nouveau doctype simplifié et un code Html dépouillé de tout aspect de présentation.

- Une chasse aux plugins qui viennent encombrer le code et compliquer la tâche des concepteurs, par leur prise en charge directe dans le navigateur. Les nouvelles balises audio et vidéo en sont une belle illustration.
- La reconnaissance du JavaScript comme partenaire de la publication sur la toile. On pense ainsi aux scripts classiques pour vérifier l'encodage obligatoire ou la validité d'une adresse de courrier électronique dans les formulaires. Le Html5 évitera ces encodages redondants par la prise en charge de façon native par les navigateurs de ces fonctions. Autre exemple, le codage du dessin 2D et bientôt 3D (balise <canvas>) est beaucoup plus proche du JavaScript que d'un langage de balise.
- Le Html5 devient également une plate-forme d'interfaces d'applications (APIs) intégrant des fonctionnalités complexes comme la géolocalisation, l'édition en ligne ou le glisser-déposer (*drag/drop*).

Les nouveautés du Html5

Les nouveautés du Html5 et surtout du trinôme du concept "Html5 + CSS3 + applications JavaScript" sont nombreuses.

Html5

- Un nouveau doctype simplifié et unifié.
- La suppression des balises et attributs de présentation.
- De nouvelles balises sémantiques ou d'organisation.
- De nouvelles balises audio et vidéo qui ne nécessitent plus l'appel à des plugins dédiés.
- Le dessin 2D et bientôt 3D par la nouvelle balise <canvas>.
- Une profusion de formulaires novateurs comme les curseurs ou les calendriers et la prise en charge de façon native par les navigateurs de la validation des données.
- Etc.

L'objectif premier de cet ouvrage est bien entendu l'étude détaillée de l'héritage du Html 4.0 et des nouveautés du Html5. Il faut noter que ce dernier est déjà bien implanté dans les navigateurs de la dernière génération. L'implantation actuelle du Html5 permet déjà de le découvrir et de l'utiliser. Il faut cependant noter que l'intégration du Html5 n'est pas encore complète. De nouveaux apports sont encore attendus, particulièrement en ce qui concerne les formulaires.

CSS3

- De nouveaux sélecteurs.
- Les bords arrondis.
- Les bordures imagées.
- Les polices personnalisées permettant à la typographie de trouver enfin sa place dans les interfaces Web.
- La présentation en plusieurs colonnes d'un texte.
- Les ombres sur le texte comme sur les éléments.
- Les dégradés de couleur.
- Les arrière-plans multiples.
- L'opacité ou la transparence.
- Les transformations.
- Les transitions (sans JavaScript).

Ces nouvelles spécifications CSS3 sont déjà disponibles dans les versions récentes des navigateurs et en particulier dans les navigateurs de notre étude (voir la section "Les navigateurs de notre étude" dans le présent chapitre). Il ne fait aucun doute que les CSS3 vont révolutionner la façon de concevoir et d'exploiter les interfaces Web dans les prochains mois ou les prochaines années.

Les applications JsAPI

- La géolocalisation qui permet de localiser (avec sa permission) l'utilisateur par ses coordonnées de longitude et latitude.
- Des super cookies avec *Web Storage* qui permettra un stockage plus important de données dans le navigateur.
- L'utilisation des applications Web hors connexion après la mise en cache des ressources nécessaires.
- Les *Workers* qui peuvent exécuter des tâches de fond en parallèle du programme JavaScript principal dans un environnement totalement séparé de la page.
- Les *Websockets* qui permettent d'établir une communication bidirectionnelle asynchrone entre le navigateur et le serveur.
- Le glisser-déposer (*drag/drop*) en natif dans le navigateur.
- L'attribut *ContentEditable* qui permet l'édition en ligne du contenu d'un élément. Il fait apparaître un éditeur WYSIWYG basique qui permet donc d'éditer directement le contenu dans la page. Toutes les modifications apportées par l'utilisateur pourront être traitées à la fin de l'édition.
- Etc.

Ces applications dépassent largement le cadre de cet ouvrage dédié à l'apprentissage du Html5. Certaines sont déjà implémentées dans les navigateurs récents, d'autres non. Nous aborderons cependant brièvement la géolocalisation dans un chapitre pour son côté spectaculaire et intrigant.

Un certain nombre de ces fonctions sont déjà reprises dans les cadriels (framework) JavaScript comme Dojo et jQuery.

Le Html5 est un langage de balises

Digne successeur des versions précédentes du Html et du Html 4.0 en particulier, le Html5 est toujours un langage de balises.

1. Les balises

Les balises, aussi appelées éléments, sont des commandes à l'intention du navigateur et saisies entre des signes inférieur à (<) et supérieur à (>). Ainsi une balise s'écrit <balise>.

En règle générale, à toute balise d'ouverture correspond une balise de fermeture qui marque la fin de la commande annoncée par la balise d'ouverture. La balise de fermeture reprend le même énoncé que la balise d'ouverture mais ce dernier est précédé d'une barre oblique (/). Ainsi à la balise d'ouverture <balise> correspondrait la balise de fermeture </balise>. La syntaxe d'une balise est alors :

```
<balise> ... </balise>
```

Pour comprendre le mode de fonctionnement des balises, je vous propose le texte suivant :

Il est **important** d'apprendre le langage *Html5* !

Ceci peut se comprendre de cette façon :

- écrire "Il est" de façon normale puisque rien n'est spécifié,
- ensuite écrire le mot (et uniquement ce mot) "important" et le mettre en gras,
- rependre l'écriture normale pour "d'apprendre le langage",
- écrire cette fois-ci en italique le mot "Html5"
- et terminer par "!" en écriture normale.

Ce qui donne le résultat suivant :

Il est **important** d'apprendre le langage *Html5* !

Le langage Html n'est ni plus ni moins que cela. Chaque fois que l'on donne une instruction au navigateur, par exemple mettre un titre, commencer un tableau ou faire un lien vers une autre page, une balise de début est appliquée. La balise de fermeture signale au navigateur que la commande est terminée.

Une exception cependant, Html5 a hérité du Html des balises uniques, aussi appelées balises vides qui n'ont pas de balises de fermeture.

Par exemple, la balise image .

Afin d'assurer une rétrocompatibilité avec le Xhtml, ces balises peuvent également s'écrire en Html5 avec un signe de fermeture.

Ainsi, la balise image peut également s'écrire en Xhtml .

L'espace avant la barre oblique de fermeture est important pour des raisons de compatibilité avec les (très) anciens navigateurs.

Ce qui nous handicape un peu, nous les francophones, c'est le fait que ces balises font appel à des termes ou des abréviations de termes anglo-saxons qui les rendent (au premier abord) abstraites et donc complexes.

Exemples

	b pour <i>bold</i> ce qui signifie gras
<i>	i pour <i>italic</i> ce qui signifie italique
<p>	p pour <i>paragraph</i> ce qui signifie paragraphe

<div>	div pour <i>division</i> ce qui signifie ... division
<table>	<i>table</i> signifie tableau
<form>	<i>form</i> signifie formulaire
	<i>img</i> pour image
Etc.	

2. Les attributs de la balise

Il est parfois nécessaire de compléter une commande par des spécifications plus précises dans l'un ou l'autre domaine. Pour ce faire, le langage Html5 dispose des attributs de la balise. L'attribut s'insère dans la balise, entre le mot de commande et le signe > final.

La syntaxe complète d'une balise avec un attribut est : <balise attribut="valeur" ... </balise>

L'attribut comporte toujours une valeur, celle-ci s'indique en complément de l'attribut par un signe égal (=) suivi de la valeur mise entre des guillemets. Il est impératif de mettre cette valeur entre guillemets. Le strict respect de la syntaxe veut qu'il n'y ait pas d'espace avant et après le signe égal.

Il est possible d'utiliser plusieurs attributs, séparés par un espace, dans une même balise : <balise attribut1="valeur" attribut2="valeur" ... </balise>.

Le bon usage des balises

Voici quelques règles simples qu'il faudra respecter lors de l'écriture des balises Html5.

- En Html5, les balises ne sont pas sensibles à la casse. Ainsi, on peut écrire indifféremment <BALISE>, <Balise> ou <balise>. Certains voyaient même en l'utilisation des majuscules, un moyen efficace pour distinguer le code Html du contenu dans un document. Cependant, l'usage s'est généralisé d'écrire les balises en minuscules (comme en Xhtml).
- La règle générale veut que toute balise ouverte <balise> doit être fermée </balise>. Les libertés prises dans l'écriture du Html 3.2 à cause du fonctionnement plus ou moins permissif des navigateurs ne sont plus d'actualité. La rigueur apportée par le Html 4.0 strict et le Xhtml 1.0 doit continuer à guider votre écriture.
- Les balises doivent être correctement imbriquées. Lorsqu'on affecte plusieurs balises à un élément, l'ordre de fermeture de celles-ci est essentiel. La première balise de fermeture doit correspondre à la dernière balise d'ouverture non fermée. Un exemple et tout sera beaucoup plus clair :

Est correct : <a><c>contenu</c>.

Est incorrect : <a><c>contenu</c>.

- Les valeurs des attributs doivent toujours figurer entre des guillemets. Ici aussi, la rigueur dans le code reste de mise.

Les balises et attributs Html 4.0 disparus

Les différences, tant au niveau des balises que des attributs, sont nombreuses et importantes par rapport au Html 4.0.

1. Au niveau des balises

Notons tout d'abord la disparition complète des cadres. Les balises `<frame>`, `<frameset>` et `<noframes>` ont définitivement disparu des outils à la disposition des auteurs. Ceci ne constitue pas réellement une surprise car ils étaient décriés depuis de nombreuses années. Les concepteurs professionnels les évitaient comme la peste depuis que Google avait annoncé à l'attention des webmestres qu'il ne pouvait garantir la bonne indexation des sites comportant des cadres. La balise `<iframe>` quant à elle subsiste en Html5.

Ensuite, la disparition de certaines balises de présentation comme `<big>`, `<center>`, ``, `<strike>`, `<tt>` et `<u>`. Ici aussi, pas de grande surprise car le principe de la séparation de contenu et de la présentation est à présent bien établi. Ces balises de présentation doivent maintenant être prises en charge par des feuilles de style CSS.

2. Au niveau des attributs

Le Html5 va au bout du principe de la séparation du contenu et de la présentation. Ce qui étonne, c'est l'ampleur des attributs concernés. En résumé, ce sont tous les attributs relatifs à l'alignement, la largeur, les arrière-plans (de couleur ou avec image), les bordures et la numérotation des listes qui sont passés à la trappe.

Tous ces attributs doivent être pris en charge par les feuilles de style qui deviennent ainsi indissociables du code Html5.

De façon plus détaillée :

- align dans les balises `<caption>`, `<iframe>`, ``, `<input>`, `<object>`, `<legend>`, `<table>`, `<hr>`, `<div>`, `<hx>`, `<p>`, `<col>`, `<colgroup>`, `<tbody>`, `<td>`, `<tfoot>`, `<th>`, `<thead>` et `<tr>`.
- alink, link, text et vlink dans la balise `<body>`.
- background dans la balise `<body>`.
- bgcolor dans les balises `<table>`, `<tr>`, `<td>`, `<th>` et `<body>`.
- border dans les balises `<table>` et `<object>`.
- cellpadding et cellspacing dans la balise `<table>`.
- frameborder dans la balise `<iframe>`.
- height dans les balises `<td>` et `<th>`.
- hspace et vspace dans les balises `` et `<object>`.
- marginheight et marginwidth dans la balise `<iframe>`.
- noshade dans la balise `<hr>`.
- nowrap dans les balises `<td>` et `<th>`.
- rules dans la balise `<table>`.
- size dans la balise `<hr>`.
- type dans les balises ``, `` et ``.

- **valign dans les balises** `<col>`, `<colgroup>`, `<tbody>`, `<td>`, `<tfoot>`, `<th>`, `<thead>` et `<tr>`.
- **width dans les balises** `<hr>`, `<table>`, `<td>`, `<th>`, `<col>`, `<colgroup>` et `<pre>`.

Ainsi, il n'est plus possible de fixer, en Html5, les bordures, l'alignement, les arrière-plans éventuels et la largeur des tableaux sans avoir recours aux feuilles de style CSS.

-  Avec la disparition des attributs `color` ou `bgcolor`, le Html5 se décline en noir et blanc.
-

Cette absence complète d'attributs de présentation nous a obligés de revoir notre structure d'apprentissage traditionnelle du Html avec d'un côté les balises Html et par ailleurs les feuilles de style CSS. Ainsi, dès les premiers pas en Html, nous avons adjoint des notions élémentaires de feuilles de style pour la présentation. Pour permettre à l'apprenant de s'exercer, par exemple, dans l'apprentissage des tableaux, des éléments de feuilles de styles se révélaient indispensables pour au moins visualiser ceux-ci par une bordure. Une étude plus poussée et plus détaillée des propriétés CSS est abordée comme d'habitude à la suite de l'étude des balises et attributs du Html5.

Les navigateurs de notre étude

Ces technologies de pointe que sont le Html5 et les feuilles de style CSS3 ne sont reprises que par les dernières versions des navigateurs.

En outre, l'intégration du Html5 et des CSS3 n'est encore que partielle et varie d'un navigateur à l'autre. À ce stade, il est ainsi nécessaire de retenir pour notre étude les principaux acteurs de ces techniques innovantes pour assurer une vue d'ensemble des apports importants qu'ils entraînent ou entraîneront dans l'édition et la publication des documents Web.

Une intégration complète du Html5 et des CSS3 par les navigateurs de pointe prendra encore quelques mois, voire quelques années. Mais étant donné l'implication des firmes elles-mêmes dans l'élaboration du Html5, on peut présager que les évolutions seront rapides.

Il faudra également attendre un certain nombre d'années avant que, par le système de renouvellement ou de mises à jour des navigateurs, le Html5 et les feuilles de styles CSS3 deviennent des applications grand public.

Il est intéressant de noter l'importance donnée soudainement au JavaScript dans la conception des pages ou plutôt des applications Web. Cette prise de position se concrétise dans le fait que tous les navigateurs se sont dotés d'un nouveau moteur JavaScript. Que ce soit Opera 10.5 avec Carakan, Safari avec Nitro, Google Chrome avec V8 ou Internet Explorer 9 avec Chakra. On annonce même deux moteurs JavaScript dans Firefox 4 avec SpiderMonkey et le tout nouveau Narcissus.

Notons aussi l'apparition de l'accélération matérielle qui fait supporter par la carte graphique et non plus par le processeur, la restitution des animations et vidéos.

Internet Explorer 9

Internet Explorer 8 est arrivé à bout de course et présente un retard conséquent par rapport à la concurrence en raison de sa prise en charge quasi nulle des nouveaux standards Html5 et des feuilles de style CSS3.

Internet Explorer 9, annoncé courant 2011, se repositionne dans le groupe des navigateurs modernes en assimilant (enfin) du Html5 et une partie des spécifications CSS3.

Un problème de taille subsiste toutefois puisqu'Internet Explorer 9 n'est prévu que sous Microsoft Vista et Seven. Donc pas d'Internet Explorer 9 pour les utilisateurs de XP. Cet abandon de XP s'explique du côté de Microsoft par l'accélération matérielle d'Internet Explorer 9 qui réclame un système d'exploitation moderne. Pourtant, les spécialistes rétorquent que des navigateurs modernes comme Firefox, Chrome et Opera font bien de l'accélération matérielle sous XP. Cette prise de position est en tout cas regrettable pour les 50 % d'utilisateurs restés fidèles à XP. Mais c'est peut-être cette fidélité qui est reprochée...

Quelques mots pour terminer sur le cauchemar des développeurs Web. En effet, les statistiques indiquent que fin 2010, encore 5 % des internautes utilisent le totalement obsolète Internet Explorer 6.

Firefox 4

Attendu début 2011, Firefox 4 restera sans nul doute le navigateur préféré des concepteurs grâce à son système d'extensions qui en font le partenaire incontournable dans le développement d'applications Web. On pense spécialement à *Firebug* et *Web Developer* qui sont des extensions devenues quasi indispensables pour de nombreux développeurs. Les versions bêta disponibles à ce jour font apparaître de réelles avancées par rapport à son prédecesseur Firefox 3.6 que ce soit par son interface renouvelée, une meilleure prise en charge des standards Html5, les CSS3, l'accélération graphique, etc.

Firefox 3.6, vieillissant par rapport à la concurrence et spécialement celle de Google Chrome, ne sera pas totalement oublié pour autant. Il faut garder à l'esprit qu'il reprenait déjà de nombreux effets des CSS3 alors que d'autres navigateurs (lire Internet Explorer) les ignoraient complètement.

Google Chrome 7

Apparu en septembre 2008, Google Chrome n'a cessé depuis d'accroître ses parts de marché. On peut considérer qu'il est à l'heure actuelle le troisième navigateur le plus utilisé sur la toile. Depuis les versions se sont succédées à un rythme soutenu et toujours à la pointe de technologies comme le Html5 et la prise en charge progressive des feuilles de style CSS3. Il faut souligner que les versions récentes sont rapidement adoptées par les utilisateurs. Ce qui confirme la modernité et le dynamisme de ce jeune navigateur.

Safari 5

Safari est depuis 2003, le navigateur par défaut des plateformes Mac OS X. Depuis sa version 4 (juin 2009), il est également disponible pour Windows. Cet excellent navigateur, rapide et innovant, a cependant quelques difficultés à être adopté par les utilisateurs de Windows. Pourtant le navigateur Safari est à la pointe de la prise en charge du Html5 et des feuilles de style CSS3 et ne concède rien en performances à la concurrence.

Opera 10.6

Ce navigateur sympathique mais plutôt confidentiel nous sera très utile pour notre étude du Html5 car il est le plus novateur en la matière. Il nous permettra par exemple, d'illustrer les nouveautés en matière de formulaires du Html5 qui ne sont pas encore reprises à ce jour par les navigateurs précités.

Le document Html5 minimum

1. Le doctype et son importance

Tout document Html doit commencer par un doctype. Le Html5 propose un doctype unique et simplifié.

```
<!DOCTYPE html>
```

Pour se rendre compte de la simplification, il suffit de le comparer avec un doctype du Html 4.0.

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

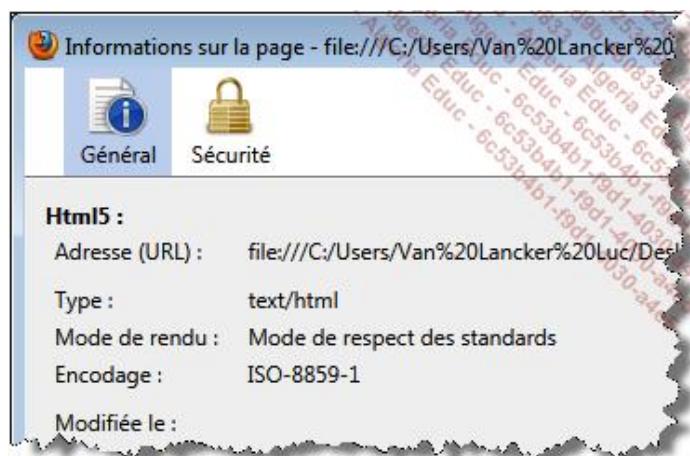
Nettement plus court et enfin mémorisable !

Le doctype, aussi appelé DTD sert à indiquer au navigateur à quelles règles d'écriture obéit le code source de la page Html ou Xhtml. Il utilisera celles-ci pour afficher la page selon les standards du W3C. Tous les navigateurs appliquant la même règle, vous êtes ainsi assuré d'un rendu identique entre les différents navigateurs.

En l'absence d'un doctype, le navigateur ignore selon quelles règles il doit traiter la page. Il se rabat ainsi sur des procédures qui lui sont propres pour afficher vaille que vaille votre page Html. Ce qui peut entraîner des différences de restitution sensibles entre les différents navigateurs. Ce mode bancal est appelé mode compatibilité ou *quirks mode*.

Vous pouvez aisément vérifier dans quel mode (quirks ou respect des standards) se trouve le navigateur.

Dans Firefox 3.6+, accédez au menu **Outils - Informations sur la page** - onglet **Général - Mode de rendu**.



Pour Internet Explorer 8+, encodez `javascript:alert(document.compatMode)` dans la barre d'adresse. Si une fenêtre d'alerte indique `css1Compat`, il s'agit du mode standard. Si la fenêtre d'alerte affiche `BackCompat`, il s'agit du mode *quirks*.

➤ Le doctype doit se situer à la première ligne du fichier Html. S'il y a n'importe quoi d'autre, même un simple espace ou une ligne blanche, certains navigateurs considéreront que la page n'a pas de doctype et afficheront celle-ci en quirks mode.

2. La balise Html

La balise `<html>` indique au navigateur qu'il s'agit d'un document Html.

La balise `<html>` est l'élément le plus haut ou l'élément racine du document. Elle prend donc place juste après la déclaration de doctype.

```
<!DOCTYPE html>
<html lang="fr">
...
</html>
```

La balise `<html>` peut prendre comme attribut `lang="fr"` qui spécifie que le document est en langue française. Cette information est fort prisée par les moteurs de recherche comme Google ainsi que par les synthèses vocales utilisées par les personnes malvoyantes.

À l'adresse www.w3schools.com/tags/ref_language_codes.asp, vous pouvez trouver une liste complète des autres langues référencées.

3. L'en-tête du document

La balise `<head> ... </head>` qui se positionne juste après le doctype et la balise `<html>` contient ce qu'on appelle l'en-tête du document.

Dans cet en-tête du document se situe toute une série d'informations relatives au document lui-même, indépendamment du contenu de celui-ci.

Ces informations peuvent être de natures très diverses :

- Le titre du document. Soit la balise `<title> ... </title>` reprise au point suivant.
- Des déclarations ou des appels de feuilles de style CSS.
- Des fonctions ou des liens vers des fichiers JavaScript.
- Des informations à l'intention des moteurs de recherche pour le référencement de la page.
- Des informations à l'intention des navigateurs.
- Des balises méta qui reprendront quant à elles l'encodage, la description de la page, les mots-clés associés à la page, le nom de l'auteur, des mentions de copyright, etc.

Dans un souci de simplification, le Html5 spécifie des valeurs par défaut pour l'attribut type des scripts, des styles et les éléments link. Sauf si vous avez expressément besoin d'une valeur différente que celles définies par défaut, vous pouvez omettre en Html5 cet attribut de type.

Le JavaScript est défini comme langage de script par défaut.

Ainsi, en Html 4.0, on devait écrire :

```
<script type="text/javascript">
...
</script>
```

Et

```
<script type="text/javascript" src="fichier.js"></script>
```

En Html5, on peut se contenter de :

```
<script>
...
</script>
```

Et

```
<script src="fichier"></script>
```

Les CSS sont définis comme le langage de feuilles de style par défaut.

Ainsi, en Html 4.0, on devait écrire :

```
<style type="text/css">
...
</style>
```

Et

```
<link rel="stylesheet" type="text/css" href="fichier.css">
```

En Html5, on peut se contenter de :

```
<style>
...
</style>
```

Et

```
<link rel="stylesheet" href="fichier.css">
```

Notre document minimal Html5 devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
...
</head>
...
</html>
```

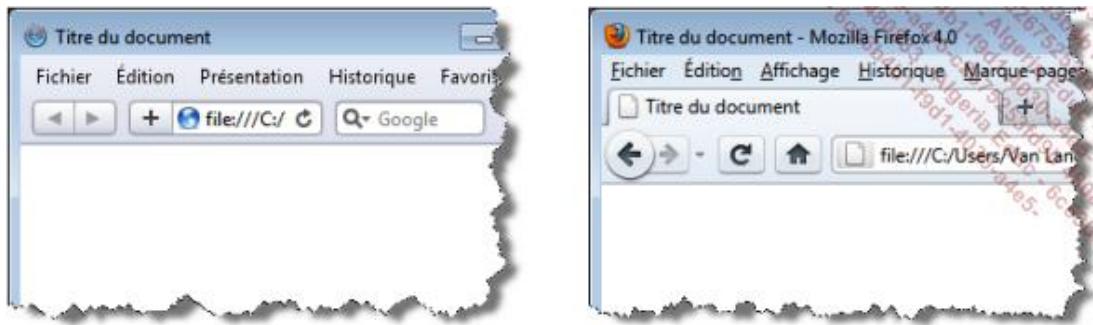
 Mis à part le titre de la page (voir ci-après), aucun élément compris entre les balises `<head> ... </head>` n'apparaît dans le navigateur.

4. Le titre du document

La balise `<title>`, incluse dans l'en-tête du document, est la seule balise obligatoire de celui-ci.

La balise `<title> ... </title>` donne un titre à votre page.

Ce titre est affiché dans la barre de titre, située en haut à gauche de tout navigateur et/ou dans les onglets ouverts par celui-ci.



La balise `<title>` a une place importante dans l'algorithme de référencement de Google. On veillera à avoir pour les documents Web, un titre accrocheur et synthétique. Il n'est pas inutile non plus d'y inclure un ou des mot(s)-clé(s) relatif(s) à la page.

Le document Html5 se complète ainsi :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
</head>
...
</html>
```

 Si par mégarde il y avait une erreur dans les balises <title> ... <title>, une simple page blanche est affichée.

5. L'encodage (*charset*) du document

La notion du jeu de caractères utilisé (*charset*) est une notion essentielle dans le développement de pages Web. Cette notion est pourtant parsemée d'embûches, surtout en termes d'interopérabilité. N'avez-vous jamais reçu d'emails où certains caractères étaient remplacés par d'autres signes cabalistiques comme des t ou vÃ©rification ?

Le jeu de caractères désigne la façon dont les caractères d'un alphabet donné sont convertis en octets dans un fichier informatique (et vice-versa). Certaines méthodes d'encodage sont spécifiques à un environnement informatique dans une langue ou un alphabet donné, d'autres sont multiplateformes et multilingues. Exemple d'encodage : ASCII, windows-1252, ISO-8859-1, ISO-8859-15, UTF-8, etc.

ASCII	Jeu de caractère basique. Il entraînera une interopérabilité maximale mais il faudra encoder les caractères accentués et spéciaux par des entités du genre é (pour le é) ou € (pour l'€).
windows-1252	Jeu de caractères utilisé par Windows et de nombreuses applications Microsoft. C'est la porte ouverte à des problèmes de compatibilité sur d'autres systèmes.
ISO-8859-1	L'ISO-8859-1 est le codage par défaut sur de nombreux protocoles réseau, ce qui est le gage d'une excellente interopérabilité.
ISO-8859-15	Version actualisée de l'ISO-8859-1 qui comporte entre autres le signe €.
UTF-8	Application de l'Unicode qui est présenté comme la solution de l'avenir. Son adoption est à l'heure actuelle freinée par sa prise en charge problématique dans certains langages de programmation (PHP).

Les navigateurs possèdent bien une fonction de détection automatique du jeu de caractères. Mais l'encodage ainsi retenu de façon arbitraire est parfois la porte ouverte à des erreurs d'interprétation.

Des surprises peuvent ainsi apparaître. Vous pouvez lire à ce sujet l'excellent article et la discussion qui s'en suit : "ISO-8859-1, ISO-8859-15, UTF-8, lequel choisir ?" à l'adresse <http://forum.alsacreations.com/topic.php?fid=17&tid=1201>.

Il fait ainsi partie des bonnes pratiques de la publication sur le web de définir prioritairement l'encodage du document. Celui-ci se réalise en Html5 par la balise :

```
<meta charset="iso-8859-1"> OU <meta charset="UTF-8">
```

L'écriture suivante, rappelant le Xhtml 1.0, est également acceptée.

```
<meta charset="iso-8859-1" /> OU <meta charset="UTF-8" />
```

Ainsi, il n'y a pas que le doctype qui est simplifié en Html5. Pour rappel, cette déclaration d'encodage prenait en Html 4.0 la forme :

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
```

Notre document Html5 minimal est alors :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
...
</html>
```

 La spécification Html5 du W3C, suivant en cela l'IETF (Internet Engineering Task Force) qui édicte les standards Internet, recommande l'usage de l'utf-8. Rien n'empêche cependant de retenir le format iso-8859-1 plus fréquemment utilisé en Europe occidentale.

6. Le corps du document

Les balises `<body> ... </body>` déterminent ce que l'on appelle le corps du document. C'est entre ces balises que prendra place le code Html5 qui sera utilisé pour élaborer le contenu de la page.

C'est cette partie du document Html qui sera affichée dans la fenêtre du navigateur et donc visible par l'internaute.

En Html5, les attributs dédiés à la présentation du corps du document de la balise `<body>` sont maintenant abandonnés pour laisser place à l'utilisation des feuilles de style.

Ces attributs qui déterminaient la couleur de l'arrière-plan (`bgcolor`), l'image d'arrière-plan (`background`) ou la couleur du texte (`text`) n'ont plus leur place en Html5.

Au final, notre document Html5 minimal se présente ainsi :

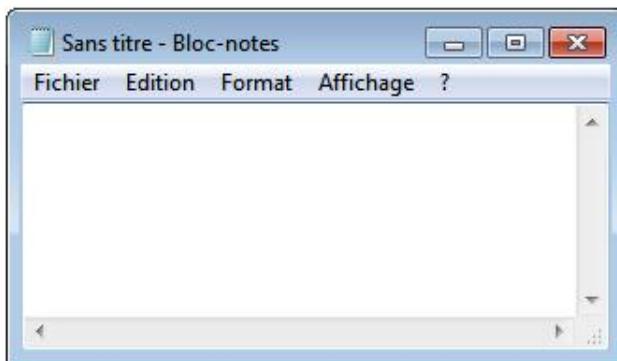
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
</body>
</html>
```

La façon de procéder

Les langages utilisés sur le Web ont cet avantage qu'ils ne réclament pas nécessairement un logiciel dédié. Pour écrire en code source, pas besoin de logiciel coûteux ! Un simple éditeur de texte, présent dans tous les systèmes d'exploitation, fait parfaitement l'affaire.

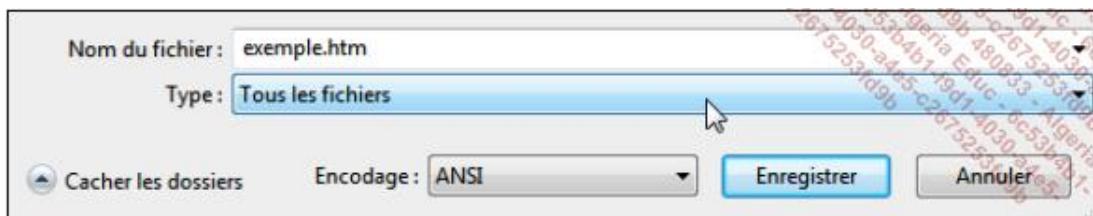
L'important est d'avoir du texte brut sans aucune mise en forme.

Sous Windows, vous pouvez utiliser le Bloc-notes ou Notepad. Pour rappel, on y accède par **Tous les programmes - Accessoires - Bloc-notes**.



Cet éditeur de texte aux fonctions rudimentaires est cependant "parfait" pour encoder du texte brut (sans aucun formatage).

Il est important de respecter une procédure d'enregistrement correcte. Après avoir fait **Fichier - Enregistrer sous...**, veillez à activer dans la zone **Type** l'option **Tous les fichiers**.



Dans le cadre de cet ouvrage consacré au Html5, les fichiers auront comme extension *.htm ou *.html. L'usage de l'extension *.htm semble se généraliser.

Pour afficher votre fichier, il suffit d'ouvrir votre navigateur et d'indiquer le chemin vers le fichier concerné.

Cette opération semble toute simple. Cependant, avec la tendance d'une interface de plus en plus simplifiée des navigateurs, elle n'est pas toujours intuitive. Pour Firefox, Safari et Opera, il faut dans un premier temps activer la barre des menus, puis retrouver le fichier par **Fichier - Ouvrir** un fichier. Pour Internet Explorer 9 et Google Chrome dont l'interface est encore plus dépouillée, il n'y a pas d'autre alternative que le raccourci clavier [Ctrl] O.

 En cas de modifications apportées au code source initial, n'oubliez pas de réactualiser la page pour que le navigateur tienne compte de celles-ci.

Exemple

À ce stade, tous les outils sont réunis pour réaliser notre premier document Html5.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
Un premier document Html5
</body>
</html>
```

Ce qui s'affiche dans Firefox 4.0 :



La validation du code Html5

Bien que le Html5 soit moins formaliste que le Html 4.0 strict ou le Xhtml 1.0, il reste utile voire indispensable de valider le code source.

Les avantages d'un code parfait et donc de la validation sont :

- Une vérification en profondeur du Html5 pour les codeurs débutants (et confirmés).
- L'assurance que votre page sera toujours correctement affichée par la plupart des navigateurs. Une page invalide peut amener les navigateurs à interpréter celle-ci de manière très différente.
- La démonstration de vos capacités professionnelles en fournissant un code de qualité, conforme au standard du html5.
- Les synthèses vocales et autres aides techniques à l'attention des personnes moins valides se basent sur un code valide pour restituer les pages de la toile.

Si le document soumis au validateur n'est pas correct, celui-ci vous fournira une analyse des problèmes qui vous permettra de corriger votre code. Les premières validations peuvent être très frustrantes mais constituent un excellent apprentissage du code source. Le rapport d'erreur est cependant souvent très technique.

 Pour effectuer leur fonction de vérification et de validation, les validateurs ont besoin du doctype du document (voir section Le document Html5 minimum - L'encodage (charset) du document du présent chapitre).

Les validateurs Html 5 (en ligne) disponibles sont :

- Le validateur du W3C lui-même.
- Le validateur du W3Québec.
- Le validateur validator.nu.

Le validateur du W3C

Le W3C lui-même présente un validateur de code (validator.w3c.org). Ce qui ne peut être qu'un gage de qualité.

La validation du Html5 est cependant encore annoncée comme étant au stade expérimental.



Notons :

- Qu'il est en anglais.
- Qu'il offre de valider un fichier par son adresse http, un fichier présent sur votre ordinateur ou du code inséré dans une zone de texte.
- Qu'il est très pointilleux.
- Que ses remarques et explications sont très techniques voire parfois sibyllines.
- Qu'il est "la" référence des professionnels.

Le validateur du W3Québec

Le W3Québec est un organisme sans but lucratif visant à promouvoir l'ensemble des normes, standards ouverts et bonnes pratiques du Web et du multimédia.



Il propose un validateur :

- En français.
- Avec la possibilité de valider un fichier par son adresse http, un fichier présent sur votre ordinateur ou du code inséré dans une zone de texte.
- Les avertissements ou erreurs sont fournis en français dans un langage technique mais accessible.
- Il manque parfois des explications plus explicites pour corriger le code soumis.
- Il s'agit en fait d'une traduction en français du validateur du W3C.

Il faut noter que le validateur du W3Québec (www.w3qc.org/validateur/) était en reconstruction lors de la conception de cet ouvrage.

Le validateur validator.nu

Le site validator.nu propose également au stade expérimental un validateur Html5 (html5.validator.nu/).

Signalons qu'il fut le premier à proposer un validateur Html5.

On peut souligner :

- Qu'il est en anglais.
- Que son interface est très simplifiée.
- Qu'il offre de valider un fichier par son adresse http, un fichier présent sur votre ordinateur ou du code inséré dans une zone de texte.
- Qu'il semble d'excellente qualité.

Pour les amateurs d'extensions Firefox, le module *Html5 Validator* vous permet, après installation, de valider votre page via le menu contextuel par le [validator.nu](#).

Test

Nous n'avons pas résisté à la tentation de réaliser un test à partir d'un code html5 erroné.

Le code erroné est :

```
1. <!DOCTYPE html>
2. <html lang="fr">
3. <head>
4. <title>Html5</title>
5. <meta charset="iso-8859-1">
6. </head>
7. <body>
8. <p align="center">
9. <s>Html5</s>
10. </p>
11. <table width="200px">
12. <tr>
13. <td>1</td><td>2</td>
14. </tr>
```

```
15. <tr>
16. <td>3</td><td>4</td>
17. </tr>
18. </table>
19. </body>
20. </html>
```

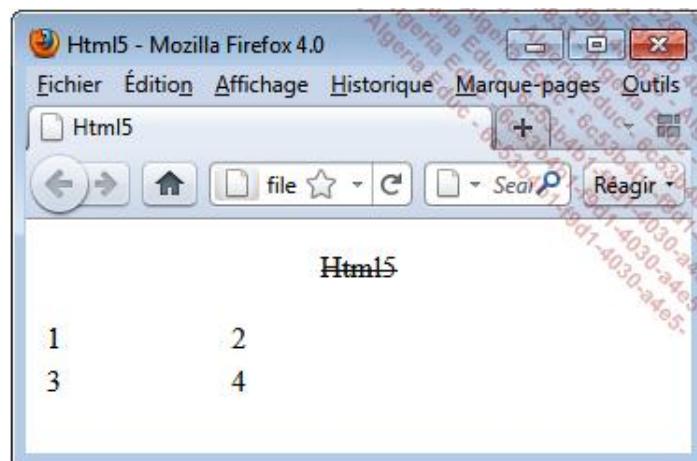
Les erreurs sont :

- À la ligne 8, l'usage de l'attribut `align` qui est supprimé (obsolète) en Html5.
- À la ligne 9, l'usage de la balise de soulignement `<s>` qui n'existe plus en Html5.
- À la ligne 11, l'usage de l'attribut `width` de la balise `<table>` est supprimé en Html5.

Le validateur du W3C détecte sans difficultés ces erreurs.

 Line 8, Column 18: <code><p align="center"></code>	The align attribute on the p element is obsolete. Use CSS instead.
 Line 9, Column 3: <code><s>Html5</s></code>	The s element is obsolete. Use CSS instead.
 Line 11, Column 21: <code><table width="200px"></code>	The width attribute on the table is obsolete. Use CSS instead.

L'affichage de ce code erroné nous fournit une belle justification de l'utilité d'une validation du code. En effet, tous les navigateurs de notre étude affichent le terme Html5 barré alors qu'un soulignement était prévu par la balise `<s>`.



Préambule

L'apprentissage de la publication sur le Web s'effectuait traditionnellement en distinguant le Html des feuilles de style CSS. Avec la disparition en Html5 de tous les attributs de présentation, il devient difficile, voire impossible, d'illustrer le code et les balises du Html. De façon logique, il apparaît nécessaire d'introduire quelques éléments de feuilles de style pour répondre à des contraintes de présentation afin de pouvoir illustrer les explications et les exemples.

Ce chapitre présente les éléments basiques des feuilles de style qui seront utilisés dans la partie consacrée au Html5. Les CSS seront bien entendu étudiées de façon détaillée dans la partie qui leur est consacrée.

L'important, à ce stade de notre étude, est simplement de comprendre les fonctions respectives du Html dédié à la structuration du contenu et des feuilles de style affectées à la présentation de celui-ci.

Les propriétés de style

Les feuilles de style prennent en charge tout ce qui concerne la présentation du document Html comme la couleur, l'alignement, les bordures, la taille, les arrière-plans ou l'interligne.

La déclaration d'un style s'effectue par le binôme propriété: valeur;.

Exemple

```
text-align: center;
```

Ce qui peut se lire : "aligner le texte de façon centrée".

Détaillons cette notation :

- propriété indique l'élément de présentation qui est concerné ; par exemple color pour la couleur, text-align pour l'alignement, border pour les bordures, font-size pour la taille des caractères.
- la propriété est séparée de sa valeur par un double-point.
- valeur spécifie par un mot-clé un pourcentage ou une grandeur en fonction de la propriété à laquelle elle est assignée.
- le point-virgule final qui indique la fin de la déclaration de style est obligatoire.
- les espaces sont permis. Ainsi, certains auteurs ont pris l'habitude de mettre un espace entre le double-point et la valeur pour la lisibilité du code.

Les sélecteurs

Il faut ensuite déterminer quels éléments du code Html seront concernés par cet effet de présentation. Ces éléments sont appelés les sélecteurs.

Sélecteur de base

Le premier sélecteur qui vient à l'esprit est la balise Html. Ainsi, on peut par exemple "accrocher" la déclaration de style précédente (aligner le texte de façon centrée) à la balise <h1>. La syntaxe devient : sélecteur { propriété: valeur; }

Exemple

```
h1 { text-align: center; }
```

Commentaires

- On reprend simplement la dénomination de la balise sans les signes inférieur (<) et supérieur (>). Donc uniquement le texte de la balise.
- La déclaration de style est comprise entre des accolades.
- Il ne faut pas oublier l'accolade fermante sinon le navigateur ne prend pas en compte la déclaration de style.

De cette façon, toutes les balises <h1> seront alignées de façon centrée.

Sélecteurs de classe

Dans certaines situations, on ne souhaite pas apporter un effet de présentation à toutes les balises d'un type donné mais à certaines balises librement choisies. Les feuilles de style ont alors introduit la notion de classes (*class*) permettant au concepteur de définir ses propres sélecteurs. D'où le nom de sélecteur universel.

La balise à laquelle on désire appliquer la déclaration de style sera distinguée des autres par l'attribut class="nom_classe" soit, par exemple, <h1 class="effet1"> ... </h1>.

La syntaxe est alors : .nom_classe { propriété: valeur; }

Soit le nom donné à la classe, précédé d'un point et suivi de la déclaration de style.

Exemple

```
.effet1 { text-align: center; }
```

Ici seule la balise <h1> avec l'attribut class="effet1" sera centrée. En outre, cet effet de présentation pourra être appliqué à d'autres balises du code comme un paragraphe ou une image.

Sélecteur d'identifiant

Son concept est proche des sélecteurs de classe mais ici, on ne souhaite appliquer l'effet de style qu'à un seul et unique élément de la page. Cet élément étant unique, il pourra être traité comme un objet qui sera éventuellement manipulé par du JavaScript.

Cet élément unique est identifié par l'attribut id="nom_identifiant" soit, par exemple, <h2 id="titre2"> ... </h2>.

La syntaxe est dans ce cas : #identifiant { propriété: valeur; }

Exemple

```
#titre2 { text-align: center; }
```

Soit le nom donné à l'identifiant, précédé du signe dièse (#) et suivi de la déclaration de style.

L'incorporation du style

Il existe de multiples façons d'incorporer des déclarations de style dans une page Html. À ce stade de notre étude des CSS, nous en retiendrons les plus élémentaires.

Style en ligne

La déclaration de style est ajoutée directement à une balise donnée dans le code Html par l'attribut style.

Exemple

```
<h1 style="text-align: center; > ... </h1>
```

Cette façon basique ne respecte pas la règle de la séparation du contenu de la présentation. Son usage doit donc rester exceptionnel.

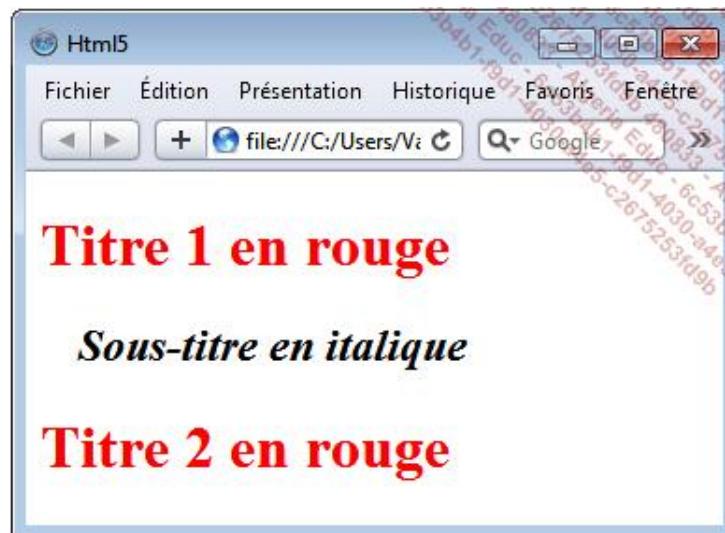
Style interne

Les différentes déclarations de style sont regroupées dans l'en-tête du document soit entre les balises `<head> ... </head>`. La syntaxe se présente ainsi :

```
<style type=""text/css">  
Déclarations de style  
</style>
```

Exemple

```
<!DOCTYPE html>  
<html lang="fr">  
<head>  
<title>Html5</title>  
<meta charset="iso-8859-1">  
<style type=""text/css">  
h1 { color: red; }  
.effet1 { padding-left: 20px;  
          font-style: italic; }  
</style>  
<body>  
<h1>Titre 1 en rouge</h1>  
<h2 class="effet1">Sous-titre en italique</h2>  
<h1>Titre 2 en rouge</h1>  
</body>  
</html>
```

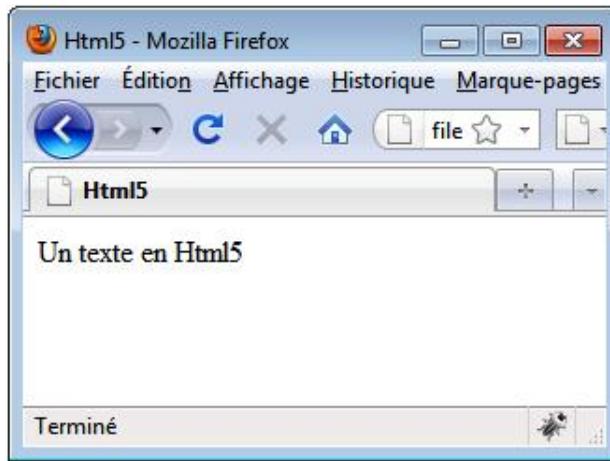


Le texte simple

Tout le contenu de votre page Web se place dans le corps du document Html5, soit entre les balises <body> ... </body>.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
Un texte en      Html5
</body>
</html>
```



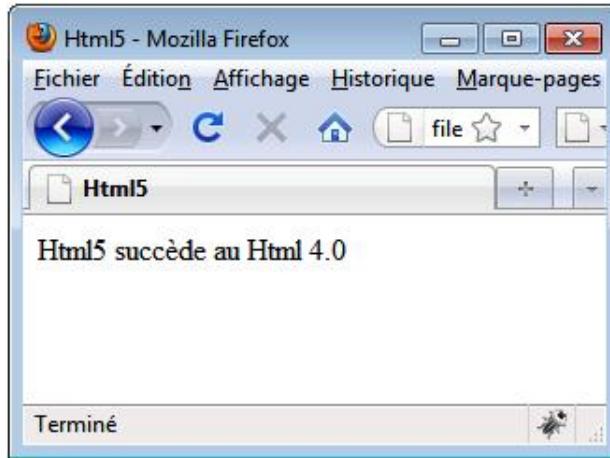
On notera qu'en Html, les espaces multiples sont ignorés. Seul un espace est retenu par le navigateur.

Le contrôle de passage à la ligne

En Html, les retours chariot, ou passages à la ligne dans le code, sont ignorés et assimilés à un espace.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
Html5 succède au
Html 4.0
</body>
</html>
```

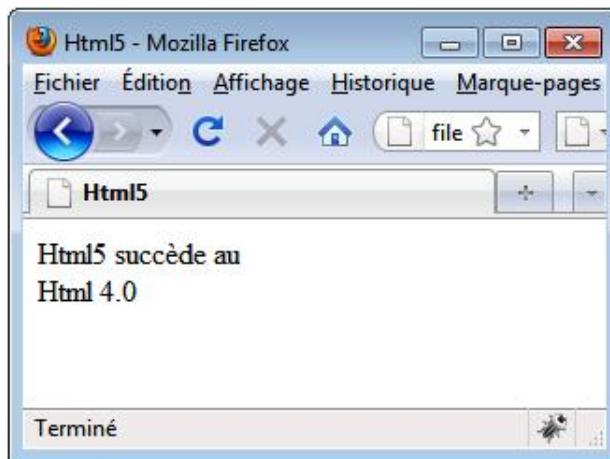


On remarque que le Html affiche tout sur une seule ligne.

Pour forcer le passage à la ligne, il faut utiliser la balise
.

Notre exemple devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
Html5 succède au<br>
Html 4.0
</body>
</html>
```



Commentaires

- La balise
 est une balise dite unique car elle ne comporte pas de balise de fermeture.
- La notation
, issue du Xhtml 1.0, est également acceptée en Html5.

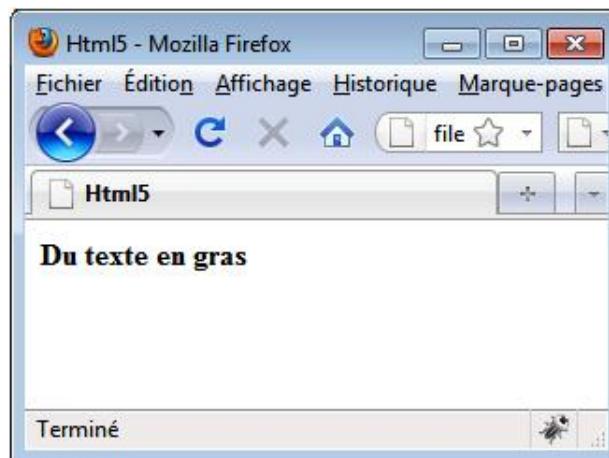
La mise en gras

Le W3C prône la stricte séparation du contenu et de la présentation. Le Html se chargeant de la seule structure du document et les feuilles de style CSS de tout l'aspect de mise en forme visuelle. Il peut ainsi sembler paradoxal de retrouver des balises de présentation ! Rassurez-vous, il ne s'agit que de quelques formatages de texte des plus basiques comme mettre du texte en gras, en italique, en exposant ou en indice, etc. L'étude détaillée des feuilles de style CSS permettra de plus grandes fantaisies de présentation.

Pour mettre du texte en gras (*bold* en anglais), il suffit d'inclure celui-ci entre les balises **...** .

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<b>Du texte en gras</b>
</body>
</html>
```



Dans le même ordre d'idée, la balise ** ... ** permet également de mettre en gras dans les navigateurs visuels.

La différence entre les balises **** et **** est pourtant sensible en Html. La balise **** induit un effet purement typographique et visuel. La balise **** a pour but de mettre en avant, de renforcer le texte. Si les navigateurs visuels ont adopté la même forme de mise en gras, l'interprétation en est fort différente dans les synthèses vocales qui lisent les pages Web à l'intention des personnes visuellement déficientes. La balise **** renforcera le contenu par une intonation de voix différente ou un niveau sonore plus élevé. Le sens donné au texte par la balise **** est ainsi fort différent.

La mise en italique

Pour mettre le texte en italique (*italic*), il faut l'inclure entre les balises *< i> ... </ i>*.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<i>Du texte en italique</i>
</body>
</html>
```



Commentaires

- Il est possible de mettre du texte en gras et en italique par le cumul des balises ** et *<i>*. Il importe cependant que ces balises soient alors correctement imbriquées. Ce qui signifie que l'ordre de fermeture des balises soit l'inverse de l'ordre d'ouverture de celles-ci. Ainsi, *<i>texte</i>* sont correctement imbriquées. Par contre, avec *<i>texte</i>*, les balises se chevauchent et l'ordre d'imbrication n'est pas respecté.
- Il existe aussi la balise ** (*emphase*). Celle-ci met également le texte en italique dans les navigateurs visuels. Il faut en trouver la différence avec la balise *<i>*, dans le sens donné à ces deux balises. La balise *<i>* est purement typographique et visuelle. La balise ** a pour but de mettre en avant, de mettre en exergue le texte qu'elle contient. Ce qui peut être rendu d'une autre façon dans les navigateurs non visuels.

La mise en exposant et en indice

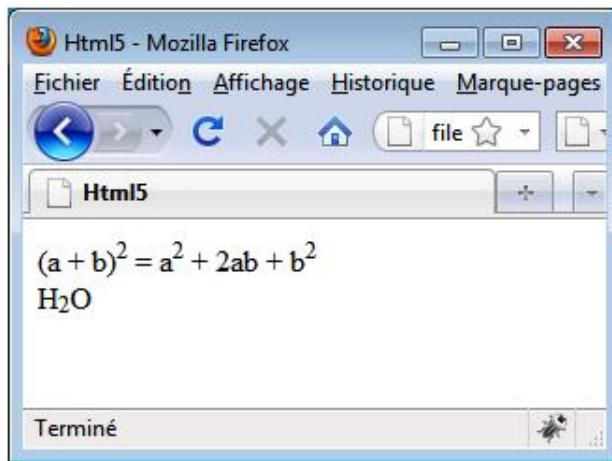
Il est parfois nécessaire de mettre des caractères en exposant ou en indice dans, par exemple, des formules mathématiques ou chimiques.

La balise `^{...}` mettra le contenu en exposant et la balise `_{...}` en indice.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
(a + b)2 = a2 + 2ab + b2  

H2O
</body>
</html>
```



Les autres balises de texte

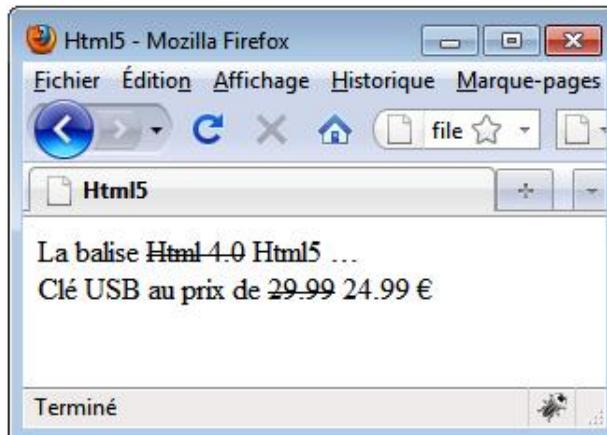
Il y a d'autres balises de texte dont l'emploi est moins fréquent sinon rare.

1. Le texte barré

La balise ` ... ` permet de marquer un élément de texte comme supprimé (*delete*) ou périmé, par exemple lors d'une mise à jour de prix dans un site commercial. Le texte apparaît alors barré à l'écran.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
La balise <del>Html 4.0</del> Html5 ...<br>
Clé USB au prix de <del>29.99</del> 24.99 €
</body>
</html>
```



Commentaire

La balise `<strike>` du Html 4.0, qui permettait de barrer du texte, n'a pas été reprise dans le Html5. La balise ``, qui présente le texte comme barré, peut être une alternative.

2. Le texte pré-formaté

La balise `<pre> ...</pre>` permet d'afficher le texte tel qu'il a été encodé dans l'éditeur de texte. Les espaces, les tabulations et les retours chariot sont respectés à l'écran. Si elle comporte du texte, celui-ci sera affiché avec une police à pas fixe.

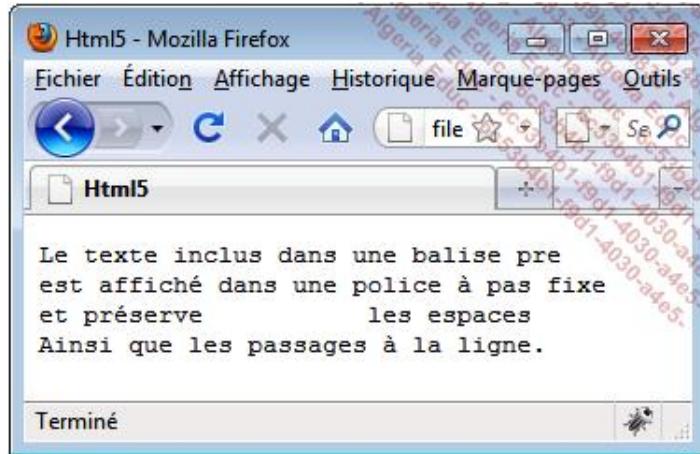
Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<pre>
```

```

Le texte inclus dans une balise pre
est affiché dans une police à pas fixe
et préserve les espaces
Ainsi que les passages à la ligne.
<pre>
</body>
</html>

```



3. La direction du texte

Certains systèmes d'écriture, tels que l'alphabet arabe et hébreu, s'écrivent de droite à gauche au contraire du sens d'écriture conventionnel de gauche à droite des langues utilisant l'alphabet latin (telles que le français). Le Html, langage universel, se devait de tenir compte de cette façon de spécificité.

La balise `<bdo> ... </bdo>` indique le sens d'affichage du texte (de la gauche vers la droite ou de la droite vers la gauche).

Les attributs sont :

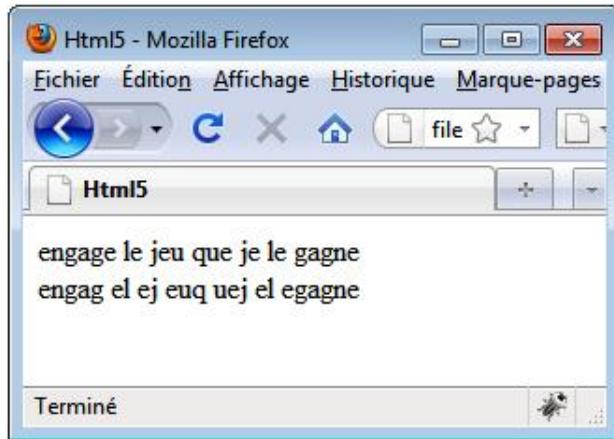
- `dir="ltr"` (*left to right*) pour le sens de lecture de la gauche vers la droite (défaut).
- `dir="rtl"` (*right to left*) pour le sens de la droite vers la gauche.

Exemple

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<bdo dir="ltr">engage le jeu que je le gagne</bdo><br>
<bdo dir="rtl">engage le jeu que je le gagne</bdo>
</body>
</html>

```



Les commentaires

Il est parfois utile de commenter le code Html (comme pour tout code de programmation d'ailleurs), pour en faciliter la compréhension lors d'une mise à jour.

En Html5, tout comme en Html 4.0, les commentaires doivent être précédés de la balise `<!--` et être fermés par la balise `-->`.

Tout ce qui sera écrit entre ces balises ne sera pas affiché à l'écran par le navigateur. Notons cependant que ces commentaires restent visibles lorsqu'on consulte le code source de la page.

Ajoutons que les commentaires peuvent occuper plusieurs lignes et peuvent prendre place indifféremment dans le `<head> ... </head>` ou dans le `<body> ... </body>`.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<b><i>Texte en gras et en italique</i></b>
<!-- commentaire particulier -->
</body>
</html>
```

Les caractères spéciaux

Il existe une foule de caractères spéciaux qu'il faut parfois encoder de façon particulière.

On pense dans un premier temps aux caractères qui sont utilisés par le code Html5. Par exemple, le signe inférieur à (<) qui marque l'ouverture d'une balise. À la lecture du code, le navigateur risque alors d'être mis en difficulté. Le validateur du W3C signale par ailleurs cette ambiguïté lors de la vérification de votre page.

Un peu comme pour les mots réservés dans un langage de programmation, il y a une liste d'éléments à éviter dans le codage du texte. Ces éléments par exemple :

- Le signe < qui marque le début d'une balise.
- Le signe > qui marque la fin d'une balise.
- Le signe " utilisé dans les attributs de balise.
- Le signe & qui marque le début d'une référence de caractère.

Ces différents signes doivent être encodés en utilisant les entités suivantes :

<	<
>	>
"	"
&	&

Remarquez bien le point-virgule final obligatoire.

Par ailleurs, il y a ensuite une série de caractères qui n'existent pas sur le clavier normal comme le © du copyright ou le ® de la marque déposée. D'autres caractères sont spécifiques aux mathématiques comme le signe ∫ pour les intégrales. Pour les personnes intéressées, une longue liste de caractères spéciaux utilisés en Html est disponible à l'adresse http://alexandre.alapetite.fr/doc-alex/alx_special.html.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
La balise &lt;b&gt; ... &lt;/b&gt; met le texte en gras.<br>
&spades; &clubs; &hearts; &diams;
</body>
</html>
```



Qu'en est-il des autres caractères spéciaux qui foisonnaient en Html ? Qu'en est-il en Html5 des ´ (pour le é), des ` (è), des ˆ (ê), des à (à), des (espace insécable), etc. ?

En principe, si le caractère existe dans le jeu de caractères (*charset*) déclaré dans le document Html5, il n'est pas nécessaire d'écrire les caractères spéciaux au moyen des entités.

Ainsi, dans les encodages iso-8859-1 et iso-8859-15, le é existe et il n'est donc pas nécessaire d'avoir recours à l'entité ´. Ce qui ne peut qu'améliorer la lisibilité du code.

Par contre, avec l'encodage iso-8859-1, le signe € n'existe pas car trop ancien pour supporter ce caractère dans son jeu. Il faut alors, obligatoirement, passer par son entité €.

Pour en savoir plus à ce sujet, vous pouvez consulter les prescriptions du W3C à l'adresse www.la-grange.net/w3c/html4.01/charset.html.

Il existe bien une détection automatique de l'encodage dans les navigateurs mais celle-ci peut dans certaines circonstances, réserver de mauvaises surprises. Il est recommandé de bien spécifier le *charset*.

Les balises Html 4.0 supprimées

Pour rappel, le Html5 va jusqu'au bout du principe de la séparation du contenu et de la présentation en supprimant toute une série de balises et d'attributs ayant trait avec la présentation.

Ainsi, les balises suivantes du Html 4.0 ont disparu dans la spécification Html5 :

- Les balises `<big>` et `<small>` qui permettaient d'afficher un texte de façon plus grande ou plus petite.
- La balise `` pour la police de caractères ainsi que ses attributs `size`, `color` et `face` qui déterminaient respectivement la taille, la couleur et le type de celle-ci.
- La balise `<s>` qui permettait de souligner du texte.
- La balise `<strike>` qui permettait de barrer du texte.
- La balise `<tt>` qui permettait d'afficher du texte dans une police à espacement fixe.

Le formatage du texte avec les feuilles de style

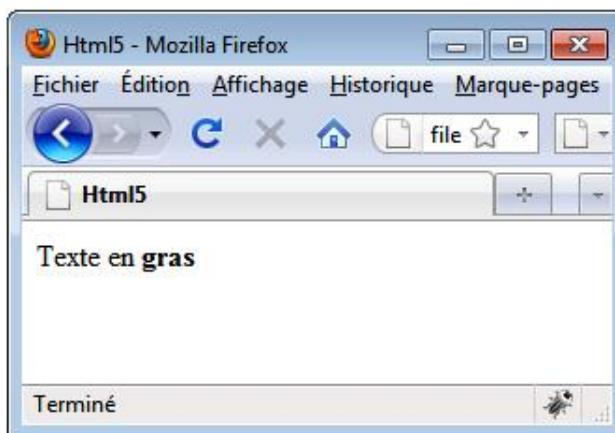
Comme expliqué au chapitre "Premiers éléments de feuilles de style", nous introduisons ci-après quelques éléments de feuilles de style afin de rendre vos premiers essais en Html5 plus agréables. En effet, le Html5 qui se préoccupe exclusivement de la structure du document, est pour le moins dépouillé.

L'important à ce stade est de comprendre le mécanisme des feuilles de style pour réaliser des éléments basiques de présentation. Leur étude détaillée sera abordée dans la partie 2 consacrée aux CSS.

1. Gras

Il est possible de mettre en gras avec les feuilles de style CSS sans passer par la balise .

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
span { font-weight: bold; }
</style>
</head>
<body>
Texte en <span class="gras">gras</span>
</body>
</html>
```



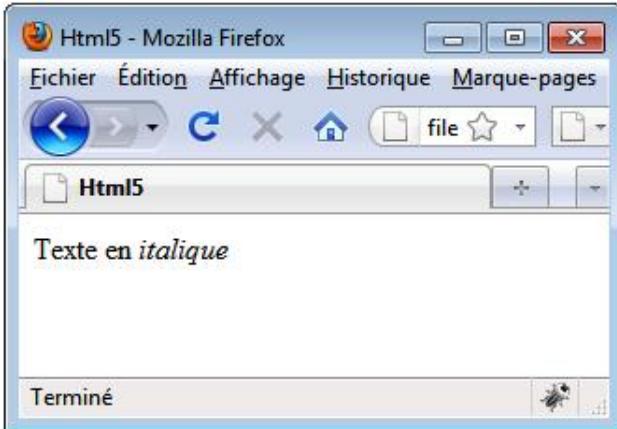
La propriété de style `font-weight` (textuellement, le poids ou l'importance de la police) est mise sur `bold` (soit en gras). Cette propriété CSS concerne les balises ``. La propriété complète `span { font-weight: bold; }` mettra toutes les balises `` du document en gras.

2. Italique

Il est possible de mettre en italique avec les feuilles de style CSS sans passer par la balise <i>.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.italique { font-style : italic; }
</style>
</head>
<body>
Texte en <span class="italique">italique</span>
</body>
```

```
</html>
```



Ici, la propriété de style `font-style` (le style de la police) est mise sur `italic` (en italique). Cette propriété CSS concerne les balises avec un attribut de classe que nous avons nommé `class="italique"`. La propriété complète `.italique { font-style : italic; }` mettra toutes les balises dotées de l'attribut `class="italique"` du document en italique.

3. La taille des caractères

Avec la suppression de la balise `` et de son attribut `size`, la taille des caractères est uniquement définie par la propriété de style CSS `font-size`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#grand { font-size: 36px; }
</style>
</head>
<body>
<div id="grand">Texte en grand</div>
</body>
</html>
```

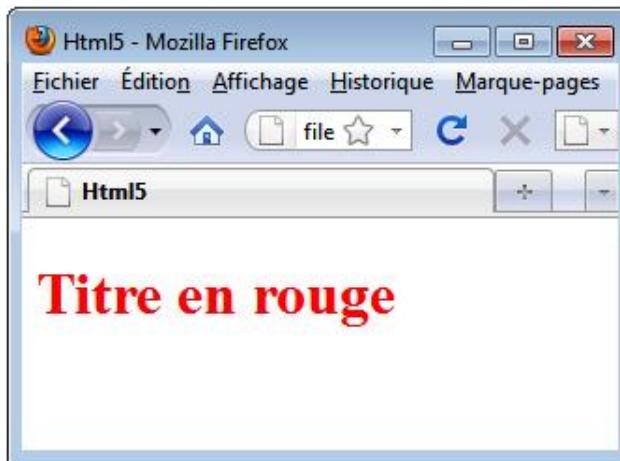


La taille des caractères (`font-size`) est fixée à 36 pixels (36px). Elle ne s'appliquera qu'à la seule balise `<div>` identifiée par `id="grand"`.

4. La couleur des caractères

Avec la suppression de l'attribut `color` de la balise ``, la couleur des caractères se définit par la propriété de style CSS `color`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
h1 {color: red;}
</style>
</head>
<body>
<h1>Titre en rouge</h1>
</body>
</html>
```

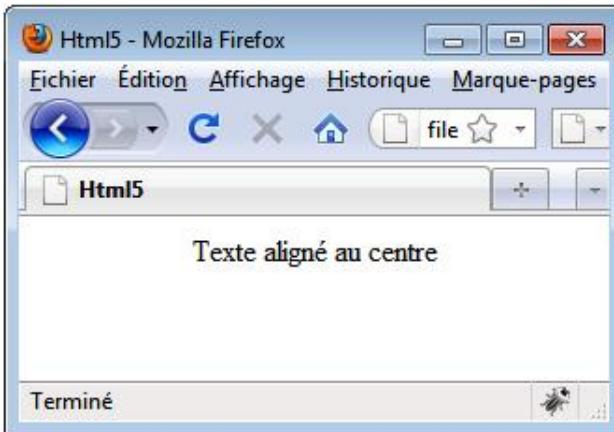


Les balises `<h1>` auront leur texte dans la couleur (`color`) rouge (`red`).

5. L'alignement

L'alignement du texte se réalise par la propriété de style CSS `text-align`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#centre { text-align: center; }
</style>
</head>
<body>
<div id="centre">Texte aligné au centre</div>
</body>
</html>
```



La propriété `text-align` indique que l'alignement du texte est centré (`center`).

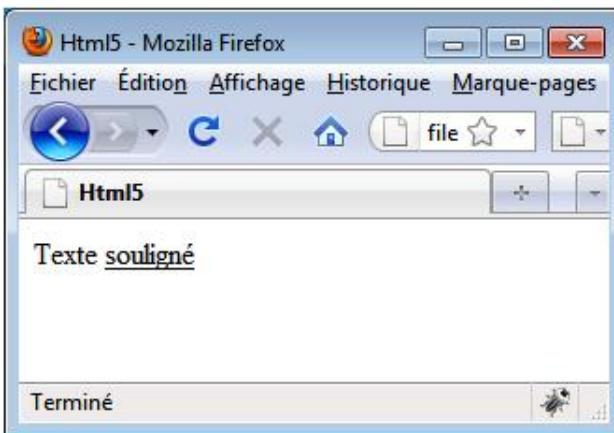
6. Le soulignement

Sur la toile, le soulignement est la convention adoptée pour signaler un lien. Tout autre soulignement de texte ne peut ainsi qu'induire votre visiteur en erreur. Ainsi, il doit être évité, sinon proscrit, lors de l'écriture du code Html.

La suppression de la balise de soulignement de texte `<s>` en est l'illustration.

Il est néanmoins possible de souligner un texte par une propriété de style.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.souligne { text-decoration: underline; }
</style>
</head>
<body>
Texte <span class="souligne">souligné</span>
</body>
</html>
```



La propriété de style indique que la décoration du texte (`text-decoration`) est le souligné (`underline`).

Les titres

Quoi de plus utile que de prévoir différentes tailles de titres pour structurer un contenu ? Il suffit de penser aux titres qui identifient une grande section de texte ou à une table des matières.

Le langage Html5 propose six grandeurs de caractères pour les titres. Quand on sait que titre se dit en anglais *heading*, l'élaboration de la balise de titre devient évidente ; *<hx>* pour *heading* suivi du numéro 1 (pour le plus grand), 2, 3, 4, 5, 6 (pour le plus petit) et le *>* de fermeture. Par la suite, la notation *<hx> ... </hx>* sera utilisée où *x* est un niveau de 1 à 6.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<h1>Titre de niveau 1</h1>
<h2>Titre de niveau 2</h2>
<h3>Titre de niveau 3</h3>
<h4>Titre de niveau 4</h4>
<h5>Titre de niveau 5</h5>
<h6>Titre de niveau 6</h6>
</body>
</html>
```



Commentaires

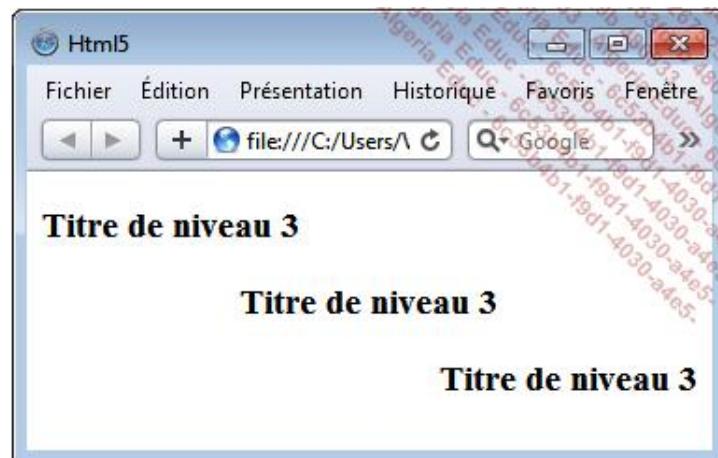
Malgré la simplicité de cette balise, plusieurs remarques s'imposent :

- Cette balise datant du tout début du Html est parfaitement compatible avec tous les navigateurs.
- Par défaut, avec la balise *<hx> ... </hx>*, le texte est mis en gras.
- Par défaut également, comme pour tous les éléments de bloc aussi appelés les éléments de division (voir point F du présent chapitre), la balise ajoute une ligne vide entre le contenu de la balise et la suite du document.

- Pour obtenir des caractères plus grands ou plus petits que ceux proposés par défaut ou une autre police que la police par défaut du navigateur, il faut passer par une feuille de style CSS.
- Les balises de titre en début de page sont fortement prisées par les moteurs de recherche comme Google et influenceront de façon pertinente le référencement et le classement du site, particulièrement lorsqu'elles contiennent des mots-clés pertinents.
- Le W3C insiste pour qu'en HTML5, les balises de titre <hx> retrouvent toutes leurs qualités de structuration des pages Web. L'usage des balises de titres doit donc être réservé à leur fonction originelle. Il importe de ne pas les détourner de leur but premier pour, par exemple, mettre du texte en gras ou avec une taille de caractères déterminée. Les feuilles de style CSS sont là pour réaliser ces présentations.
- L'attribut align, disponible dans le HTML 4.0, n'est pas repris dans le HTML 5. L'alignement du paragraphe se réalise ainsi par une propriété de style CSS.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>HTML5</title>
<meta charset="iso-8859-1">
<style>
#centre { text-align: center; }
#droite { text-align: right; }
</style>
</head>
<body>
<h3>Titre de niveau 3</h3>
<h3 id="centre">Titre de niveau 3</h3>
<h3 id="droite">Titre de niveau 3</h3>
</body>
</html>
```



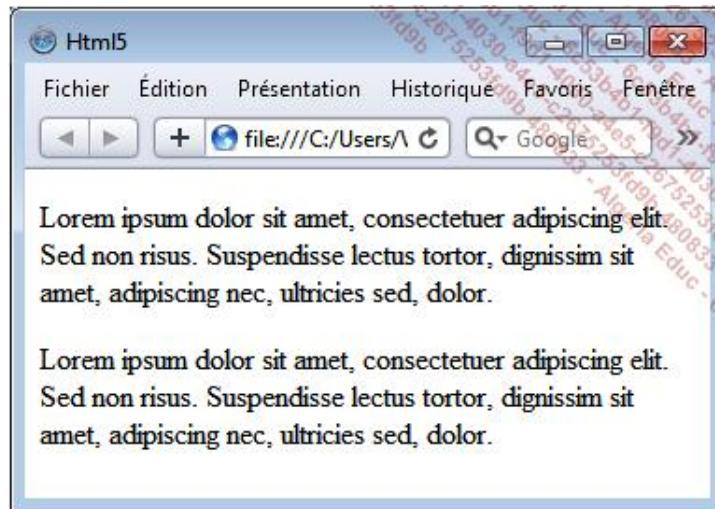
Les paragraphes

Un contenu textuel gagne en lisibilité et compréhension lorsqu'il est divisé en différents paragraphes.

Pour définir un paragraphe, la balise `<p> ... </p>` est utilisée.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor.</p>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor.</p>
</body>
</html>
```



Commentaires

- Cette balise est parfaitement compatible avec les navigateurs anciens et modernes.
- Par défaut, chaque paragraphe est précédé et suivi d'un espace correspondant à un saut de ligne pour marquer la séparation avec le contenu précédent et suivant. Cet espace peut être modifié par les feuilles de style.
- L'attribut `align` du Html 4.0 a disparu dans la spécification Html5. L'alignement (gauche, centré, droite et justifié) doit ainsi être réalisé par une propriété de style CSS.

Les citations

Les citations sont mises en avant dans le contenu textuel par les balises `<blockquote> ... </blockquote>`.

Exemple

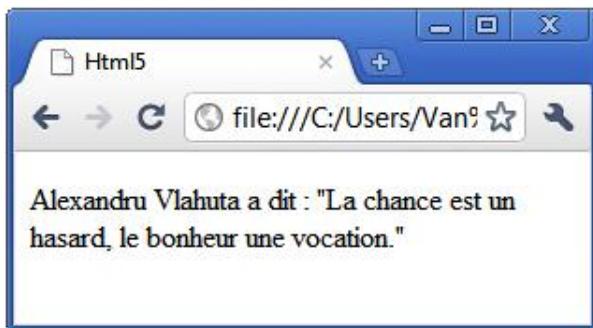
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
Alexandru Vlahuta a dit :<br>
<blockquote>La chance est un hasard,<br>
le bonheur une vocation.</blockquote>
</body>
</html>
```



Commentaires

- La citation est affichée avec un saut de ligne avant et après celle-ci comme pour les paragraphes.
- Elle s'affiche également avec un léger retrait par rapport à la marge gauche. L'usage de la balise `<blockquote>` est parfois détourné pour obtenir ce retrait. Préférez plutôt la propriété de style `margin-left`.
- Il existe aussi la balise `<q> ... </q>` pour les citations. Celle-ci n'affiche pas d'espace avant et après, ni de retrait. Elle est cependant reprise entre des guillemets.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<p>Alexandru Vlahuta a dit :<br>
<q>La chance est un hasard,<br>
le bonheur une vocation.</q>
</p>
</body>
</html>
```

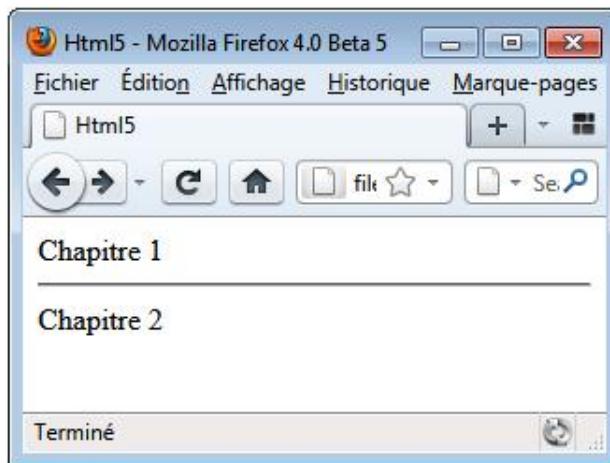


Les séparateurs horizontaux

Le trait horizontal peut se révéler fort utile pour définir un changement dans le contenu. La balise `<hr>` réalise cette fonction.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
Chapitre 1
<hr>
Chapitre 2
</body>
</html>
```



Commentaires

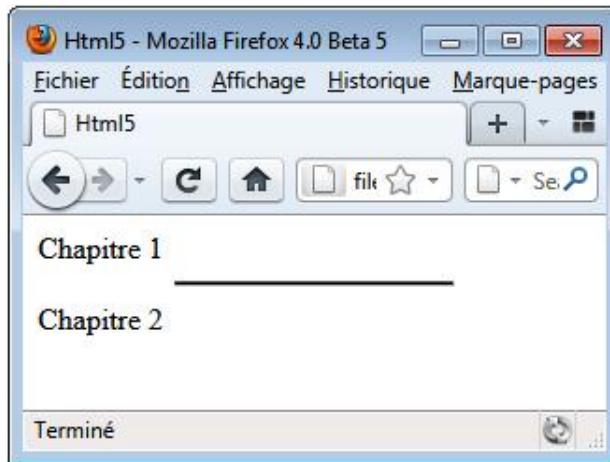
- Pas de problèmes de compatibilité pour cette balise assez ancienne.
- La balise `<hr>` est une balise unique qui ne comporte pas de balise de fermeture. Le Html5 admet aussi la notation `<hr />` dérivée du Xhtml 1.0.
- Par défaut, cette ligne horizontale occupe toute la largeur de la fenêtre du navigateur.
- Les designers préfèrent souvent remplacer ce trait plutôt basique sur le plan esthétique par une image graphiquement plus élaborée.
- Tous les attributs `align`, `noshade`, `size`, `width` déjà dépréciés (*deprecated*) dans le Html 4.0 n'existent plus en Html5. Il faut passer par les feuilles de style pour en déterminer l'alignement, la taille et la largeur.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
hr { text-align: center;
```

```
height: 3px;
background-color: black;
width: 100px; }

</style>
</head>
<body>
Chapitre 1
<hr>
Chapitre 2
</body>
</html>
```



Les listes

Les listes, bien connues des utilisateurs de traitement de texte, sont une façon efficace pour structurer un contenu textuel.

1. Les listes ordonnées

Une liste ordonnée est mise en place par la balise ` ... `.

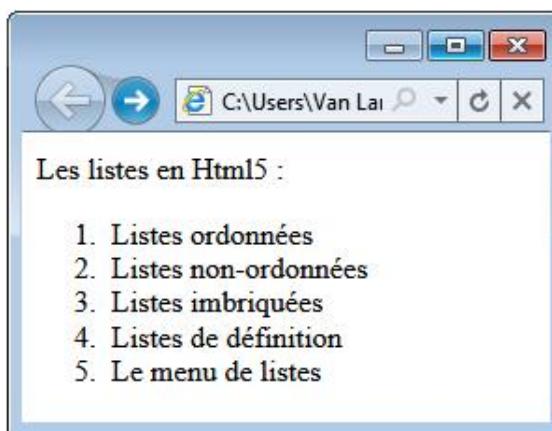
On détermine ensuite, à l'intérieur de ces balises, les éléments ou les items de la liste, soit les balises ` ... `.

La structure générale d'une liste ordonnée ou numérotée est donc :

```
<ol>
<li>Chapitre 1</li>
<li>Chapitre 2</li>
<li>Chapitre 3</li>
<li>Chapitre 4</li>
</ol>
```

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
Les listes en Html5 :
<ol>
<li>Listes ordonnées</li>
<li>Listes non-ordonnées</li>
<li>Listes imbriquées</li>
<li>Listes de définition</li>
<li>Le menu de listes</li>
</ol>
</body>
</html>
```



Commentaires

- La balise `` est un classique du Html. Elle est ainsi bien reconnue par les navigateurs.
- L'attribut `type`, qui permettait de remplacer les chiffres par les lettres (majuscules ou minuscules) ou des chiffres romains, ne fait plus partie du Html5. Nous y reviendrons dans la partie consacrée aux feuilles de style CSS pour modifier les numérotations.

- L'attribut `start` permet de commencer la numérotation à un autre chiffre que 1. Pour les amateurs de précision, cet attribut était déprécié (*deprecated*) en Html 4.0 mais il est pourtant bien repris dans le Html5.
- Le nouvel attribut Html5 `reversed` permet une numérotation descendante (5, 4, 3, 2, 1) au lieu de la numérotation ascendante habituelle (1, 2, 3, 4, 5). À ce jour, ce nouvel attribut n'est repris par aucun navigateur.

2. Les listes non-ordonnées

Les listes non-ordonnées, aussi appelées listes à puces, sont introduites par la balise ` ... `.

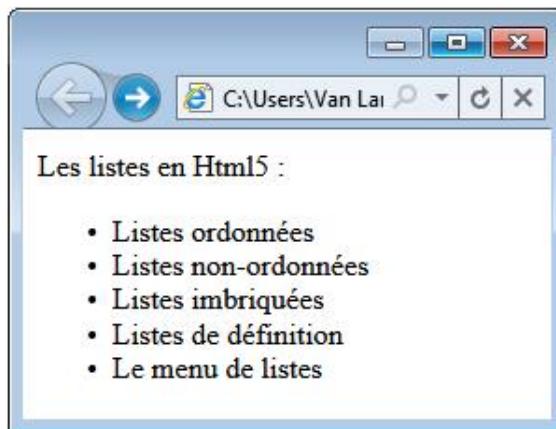
On détermine ensuite, à l'intérieur de cette balise, les éléments ou les items de celle-ci, soit les balises ` ... `.

La structure générale d'une liste à puces est donc :

```
<ul>
<li>Chapitre 1</li>
<li>Chapitre 2</li>
<li>Chapitre 3</li>
<li>Chapitre 4</li>
</ul>
```

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
Les listes en Html5 :
<ul>
<li>Listes ordonnées</li>
<li>Listes non-ordonnées</li>
<li>Listes imbriquées</li>
<li>Listes de définition</li>
<li>Le menu de listes</li>
</ul>
</body>
</html>
```



Commentaires

- La balise `` est un classique du Html. Elle est ainsi bien reconnue par les navigateurs.

- L'attribut `type` qui permettait de changer quelque peu la forme de la puce (ronde pleine, ronde vide, carrée) n'existe plus en Html5. Nous y reviendrons dans la partie consacrée aux feuilles de style CSS pour modifier la forme des puces ou les remplacer par des images.
- La forme des puces (disque plein ou disque vide) est déterminée par le navigateur.

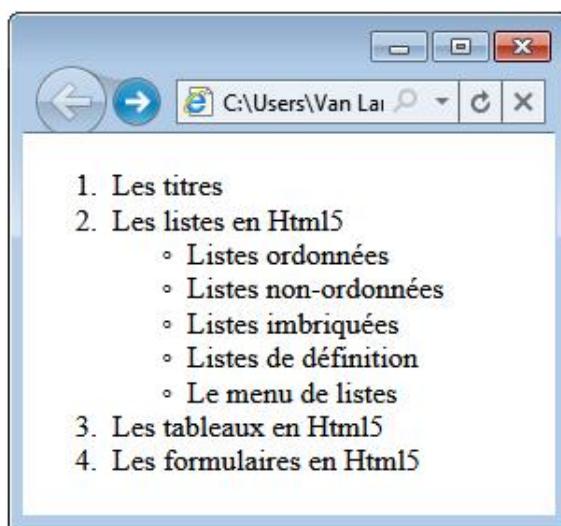
3. Les listes imbriquées

Rien ne s'oppose à élaborer des listes ordonnées et des listes non-ordonnées imbriquées. Le codage est simplement un peu plus complexe.

Exemple

Soit une liste numérotée dans laquelle s'imbrique une liste à puces.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<ol>
<li>Les titres</li>
<li>Les listes en Html5
<ul>
<li>Listes ordonnées</li>
<li>Listes non-ordonnées</li>
<li>Listes imbriquées</li>
<li>Listes de définition</li>
<li>Le menu de listes</li>
</ul></li>
<li>Les tableaux en Htm15</li>
<li>Les formulaires en Htm15</li>
</ol>
</body>
</html>
```



4. Les listes de définition

Le Html propose une dernière sorte de liste, particulièrement utile pour présenter, par exemple, des glossaires. Cette liste, dite de définition, présente une liste de termes, chacun d'entre eux étant suivi de sa description.

La liste se construit d'abord par la déclaration d'une liste de définition (*definition list*), soit `<dl> ... </dl>`.

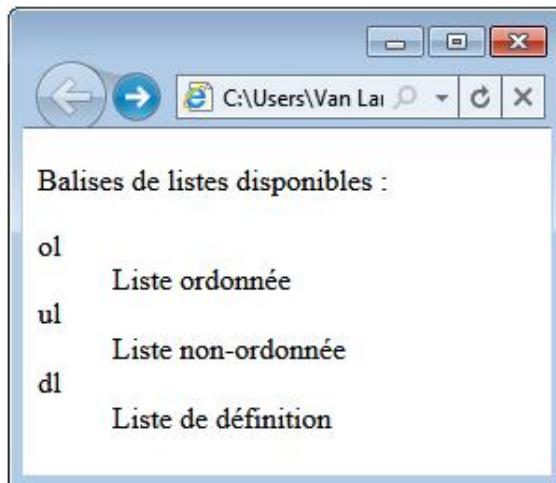
Entre ces balises, on reprend ce qu'on a appelé le terme par labalise <dt> ... </dt> suivi par la description dans labalise <dd> ... </dd>.

La structure générale d'une liste de définition est donc :

```
<dl>
<dt>Terme 1<dt>
    <dd>Definition 1</dd>
<dt>Terme 2<dt>
    <dd>Definition 2</dd>
<dt>Terme 3<dt>
    <dd>Definition 3</dd>
</dl>
```

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<p>Balises de listes disponibles :</p>
<dl>
<dt>ol</dt>
<dd>Liste ordonnée</dd>
<dt>ul</dt>
<dd>Liste non-ordonnée</dd>
<dt>dl</dt>
<dd>Liste de définition</dd>
</dl>
</body>
</html>
```



Les divisions de page

1. Les éléments de type bloc

Comme nous avançons dans notre étude du Html, il nous paraît opportun de parler de la division en bloc et en ligne des éléments.

Chaque balise Html est interprétée par les navigateurs qui en déterminent le rendu visuel ou l'affichage par l'intermédiaire de feuilles de style CSS appliquées par défaut.

Il existe deux grands groupes de balises : les balises de rendu CSS bloc (*block*) et les balises de rendu CSS en ligne (*inline*).

Les balises de rendu CSS bloc occupent par défaut toute la largeur de la fenêtre du navigateur. Elles se placent ainsi les unes au-dessus des autres. Généralement, le navigateur insère automatiquement un espace au-dessus et en dessous du contenu de la balise

C'est le cas de la balise de titre `<hx>` et de toutes les balises de ce chapitre. Elles sont toutes des balises de type bloc.

Une balise de titre ou de paragraphe occupe toute la largeur de la page et est séparée par un espace plus ou moins important.

Le code `<p>Paragraphe 1</p><p>Paragraphe 2</p>` va s'afficher sur deux lignes car la balise `<p>` est une balise de type bloc. Chaque paragraphe va occuper une ligne.

Les principaux éléments de division de type bloc qui seront détaillés tout au long de notre étude sont :

- Les balises de titres `<hx> ... </hx>`.
- Les balises de paragraphe `<p> ... </p>`.
- Les balises de citation `<blockquote> ... </blockquote>`.
- La balise `<hr>` qui introduit une ligne horizontale.
- Les balises ` ... ` des listes ordonnées.
- Les balises ` ... ` des listes à puces.
- Les balises `<dl> ... </dl>` des listes de définitions.
- Les balises de tableaux `<table> ... </table>`.
- Les balises de déclaration de formulaires `<form> ... </form>`.
- La balise `<div> ... </div>` spécialement conçue pour introduire une division (`div` pour division) sans passer par une des balises citées ci-avant. Cette balise est particulièrement utile pour appliquer une déclaration de feuilles de style CSS.

Notons qu'une balise de type bloc peut contenir une (ou plusieurs) balise(s) de type bloc. Ainsi, par exemple, une citation incluse dans un paragraphe.

L'extension Web Developer de Firefox permet de visualiser les différents éléments bloc d'une page et par conséquence la structure de celle-ci : **Outils - Web Developer - Entourer - Entourer les éléments de type block**.

Soit le code :

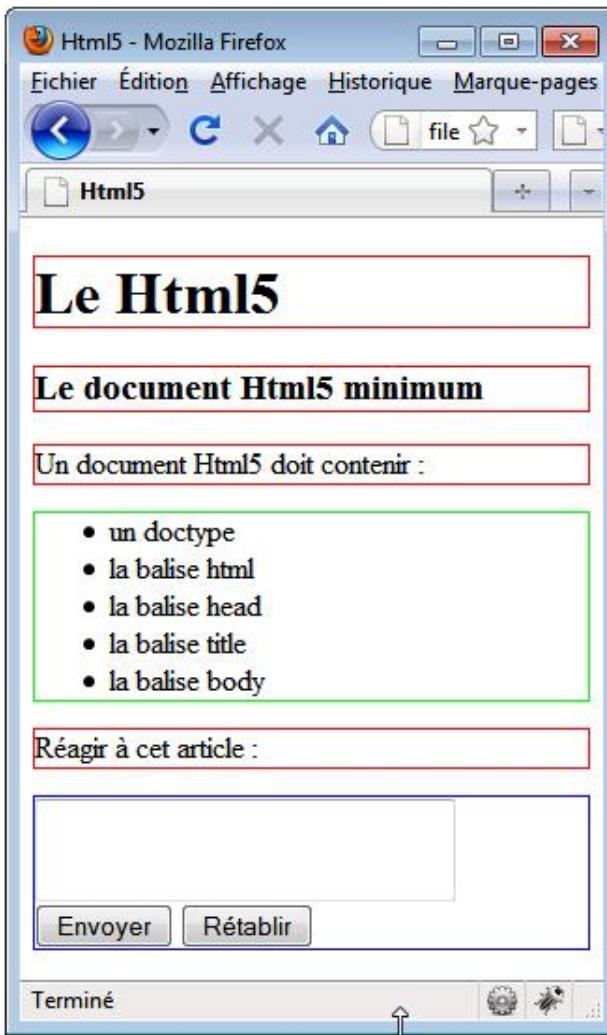
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
```

```

<body>
<h1>Le Html5</h1>
<h3>Le document Html5 minimum</h3>
<p>Un document Html5 doit contenir :</p>
<ul>
<li>un doctype</li>
<li>la balise html</li>
<li>la balise head</li>
<li>la balise title</li>
<li>la balise body</li>
</ul>
<p>Réagir à cet article :</p>
<form action="">
<textarea rows="2" cols="25"></textarea><br>
<input value="Envoyer" type="submit">
<input value="Rétablir" type="reset">
</form>
</body>
</html>

```

Dans Firefox avec l'extension Web Developer :



2. Les éléments de type en ligne

Au contraire des éléments de type bloc, les éléments en ligne se placent toujours l'un à côté de l'autre afin de rester dans le flux du texte.

Ainsi, le code texte en `gras` et en `<i>italique</i>` est restitué sur une seule ligne, sans interrompre le flux du texte : texte en **gras** et en *italique*.

Le chapitre précédent contenait beaucoup de balises en ligne car elles n'affectaient que du texte.

Les principales balises en ligne sont :

- La balise ** ... ** de mise en gras du texte.
- La balise **<i> ... </i>** de mise en italique du texte.
- La balise **
** de passage à la ligne.
- La balise **<a>** pour les liens.
- La balise **** pour les images.
- Les balises de champs de formulaires **<input>, <select> ... </select>** et **<textarea> ... </textarea>**.
- La balise **** pour introduire une division en ligne dans le texte. Cette balise est particulièrement utile pour appliquer une déclaration de feuilles de style CSS.

L'insertion d'un lien

Les liens constituent l'essence du langage Html et des pages Web. La richesse de la fonction hypertexte permet de tisser cette toile gigantesque qu'est le Web.

La balise `<a> ... ` introduit un lien. Sa syntaxe de base est :

```
<a href="destination_du_lien">Texte de lien</a>
```

La destination du lien peut être :

- Un endroit de la page en cours.
- Une autre page du site.
- Un endroit dans une autre page du site.
- Une page d'un autre site situé sur le Web.
- Une adresse électronique.
- Un fichier à télécharger.

La balise de lien possède différents attributs :

href

L'attribut `href` définit l'adresse (*url*) de la cible du lien.

hreflang

Précise la langue du document cible si celle-ci est différente de celle du document de départ.

ping

Nouveau en Html5. L'attribut `ping` peut contenir une liste d'adresses url (séparées par un espace) qui recevront une notification lorsque l'utilisateur suit le lien.

```
<a href="html5.htm" ping="http://www.monsite.com/data/visites.php">  
Html5</a>
```

Le navigateur va alors envoyer une requête POST à l'adresse url spécifiée dans l'attribut. Ce nouvel attribut sera particulièrement utile pour les statistiques d'un site. Dans le cas présent, il permettra de connaître le nombre de visiteurs de la partie consacrée au Html5.

rel

Nouveau en Html5. Spécifie la relation entre le document de départ et le fichier cible du lien.

Les valeurs sont nombreuses : alternate, archives, author, bookmark, contact, external, first, help, icon, index, last, license, next, nofollow, noreferrer, pingback, prefetch, prev, search, stylesheet, sidebar, tag, up.

Cet attribut n'est que très partiellement implémenté dans les navigateurs récents.

target

Spécifie au navigateur la façon d'afficher la cible du lien. Cela peut être dans une nouvelle instance du navigateur ou un nouvel onglet du navigateur (`target="_blank"`), dans la même fenêtre que la page de départ du lien (`target="-self"`) et dans la même fenêtre mais en occupant la totalité de la fenêtre du navigateur.

Nous y reviendrons plus longuement à la section "Le lien vers une fenêtre spécifique" du présent chapitre.

type

Indique au navigateur le type MIME de la cible si celui-ci n'est pas un document Html5, par exemple un fichier son ou une image.

Les attributs Html 4.0 suivants ont disparu de la spécification Html5 : charset, coords, name, rev et shape.

Le plus fréquemment utilisé était l'attribut `name` pour définir les ancrés pour les liens à l'intérieur d'un document. Nous y

reviendrons à la section "Le lien à l'intérieur d'une page" du présent chapitre.

Le lien vers une autre page

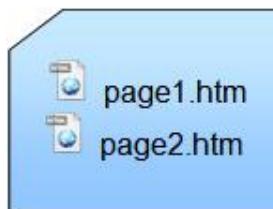
Les premiers liens lors de l'apprentissage de la publication Web, consistent à pointer vers une autre page située dans le site (adressage relatif).

Pour la création d'un site, il est fortement recommandé de regrouper tous les fichiers relatifs au site dans un seul dossier. Libre à vous de créer dans ce dossier des sous-dossiers.

1. Lien vers une page située dans le même dossier

L'adressage relatif n'est pas sans poser de problèmes dans le cas d'une arborescence complexe.

Débutons notre étude par le cas le plus simple. Soit deux fichiers (page1.htm et page2.htm) situés dans le même dossier.



Effectuons dans la page 1 un lien vers la page 2.

Le code du lien dans la page 1 devient :

```
<a href="page2.htm">Lien vers page 2</a>
```

 Il est impératif de respecter scrupuleusement les majuscules et minuscules dans le nom du fichier ainsi que d'éviter les espaces car ce qui est reconnu par Windows installé sur votre ordinateur ne le sera peut-être pas par l'hébergeur de votre site qui utilise, par exemple, un serveur Unix.

2. Lien vers une page située dans un autre dossier

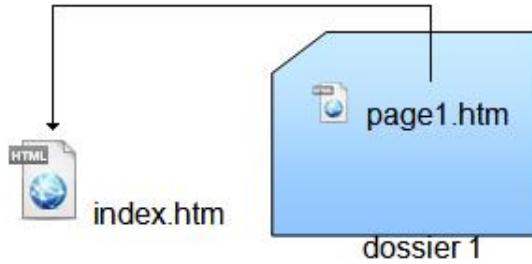
Dans les sites d'une certaine importance, on ne peut se contenter de mettre tous les fichiers dans un seul et unique répertoire. Pour structurer le site, il n'est pas rare d'avoir dans le même dossier de départ plusieurs sous-dossiers.

La conception du chemin d'accès au fichier peut se révéler quelque peu déroutante d'autant que la méthode d'adressage s'inspire de celle adoptée par Unix, peu familière aux concepteurs habitués à Windows.

Soit la situation suivante. Le dossier de départ comporte à la racine un fichier index.htm, un sous-dossier dossier1 avec le fichier page1.htm et un autre sous-dossier dossier2 avec le fichier page2.htm. Cette arborescence nous permettra d'aborder plusieurs cas de figure.



a. Lien à partir de index.htm vers page1.htm



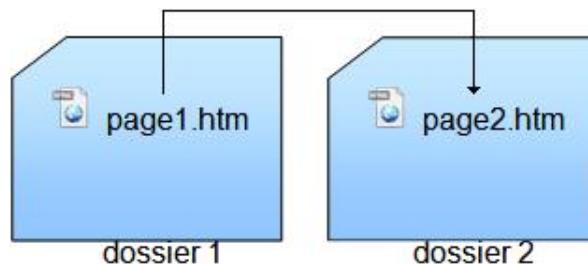
Le code dans index.htm devient :

```
<a href="dossier1/page1.htm">Texte du lien </a>
```

Commentaires

- Le fichier index.htm et le dossier dossier1 se situent au même niveau de l'arborescence. Une fois dans dossier1, il suffit de "descendre" vers page1.htm.
- C'est bien la barre oblique / qui est utilisée et non la barre oblique inverse \ que l'on rencontre parfois avec Windows.

b. Lien à partir de page1.htm vers page2.htm



Le code dans page1.htm devient :

```
<a href="../../dossier2/page2.htm">Texte du lien</a>
```

Commentaires

Pour aller de page1.htm vers page2.htm, il faut d'abord remonter d'un niveau pour sortir de dossier1. Pour remonter d'un niveau, on utilise la notation "../". Une fois remonté d'un niveau, il faut descendre vers dossier2 et une fois dans dossier2, descendre pour aller chercher page2.htm.

c. Lien à partir de page2.htm vers index.htm



Le code dans page2.htm devient :

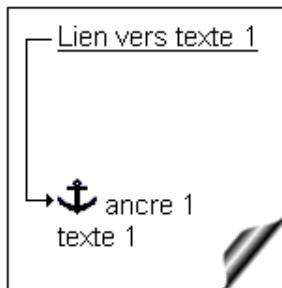
```
<a href="../../index.htm">Texte du lien</a>
```

Commentaires

Pour aller de page2.htm vers index.htm, il faut d'abord remonter d'un cran dans l'arborescence pour sortir de dossier2, d'où la notation ../. En "redescendant", on trouve directement index.htm.

Le lien à l'intérieur d'une page

Dans une page d'une certaine importance, généralement d'essence textuelle, il est intéressant de pouvoir mener l'utilisateur à un endroit précis du document. On utilise alors la technique des ancrés (*anchor*) ou des signets.



Ce lien à l'intérieur de la page s'exécute en deux temps :

- La déclaration de l'ancre.
- Le lien vers l'ancre.

La déclaration de l'ancre

Les ancrés vont définir un endroit dans la page, ce qui permettra de faire un lien vers celui-ci.

La syntaxe de création d'une ancre est :

```
<a id="nom de l'ancre"></a>
```

Avec la disparition de l'attribut `name` en Html5, la déclaration de l'ancrage se réalise par un identifiant `id`. C'était déjà le cas en Xhtml 1.0.

Notons aussi que l'ancre n'est pas affichée dans le navigateur.

Les liens vers une ancre située dans la même page

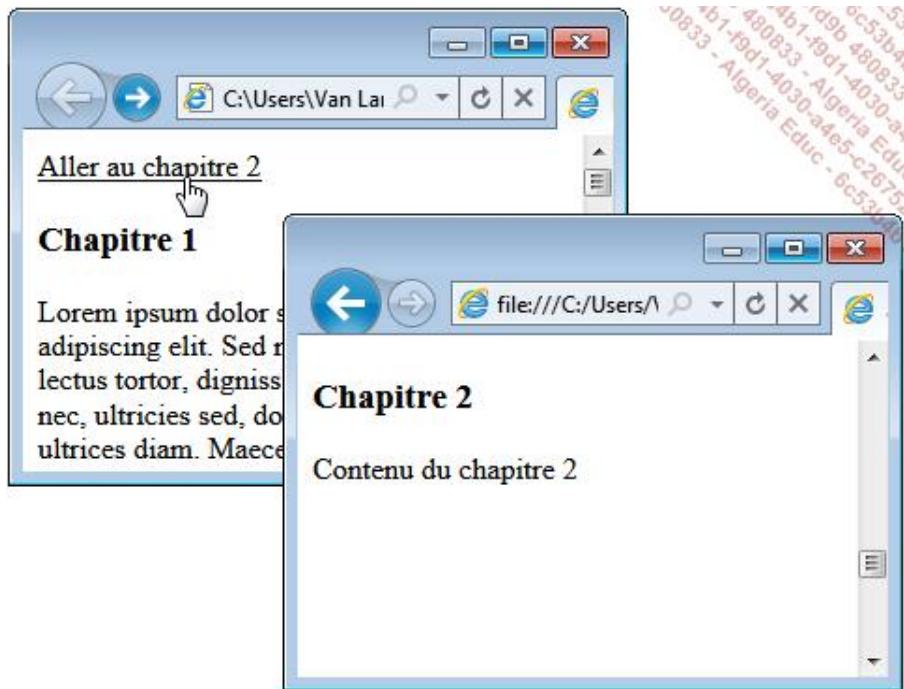
Après avoir défini l'ancre, il faut effectuer le lien vers celle-ci.

Pour créer ce lien, il suffit de reprendre dans l'attribut `href` le nom de l'ancre directement précédé du signe dièse (#).

```
<a href="#nom de l'ancre>Texte du lien</a>
```

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<a href="#chapitre2">Aller au chapitre 2</a>
<h3>Chapitre 1</h3>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam...</p>
<h3>Chapitre 2</h3>
<a id="chapitre2"></a>
<p>Contenu du chapitre 2</p>
</body>
</html>
```



Les liens vers une ancre située dans une autre page

Le principe est le même mais il faudra également définir l'adressage (relatif ou absolu) de la page.

Exemple

À partir du fichier index.htm, effectuez un lien vers une ancre du fichier page1.htm du dossier dossier1.

Après avoir ajouté une ancre, soit `` dans le fichier page1.htm, le lien dans le fichier index.htm devient :

```
<a href="dossier1/page1.htm#ancre">Texte du lien</a>
```

Le lien vers un autre site

Un lien peut faire référence à des pages d'autres sites, situées à une autre adresse sur le Web. La destination dans la balise de lien sera alors l'adresse complète du site ou de la page. On parle alors d'adressage absolu.

Soit par exemple :

```
<a href="http://www.w3.org/standards/index.htm">Lien</a>
```

Précisons qu'il s'agit bien de l'adresse complète, avec la mention du protocole http://. En effet, avec les navigateurs récents, cette mention est devenue facultative et est ignorée par la plupart des utilisateurs.

Le lien vers une adresse email

Pour ajouter un élément d'interactivité, il est intéressant de permettre aux visiteurs de vous contacter par courrier électronique.

La destination dans la balise de lien sera alors l'adresse électronique, précédée du protocole du courrier électronique (*mail*), soit `mailto:` (avec simplement un double-point).

```
<a href="mailto:editions@eni.com">L'auteur</a>
```

L'activation du lien ouvrira une fenêtre de l'application de messagerie électronique par défaut du visiteur. Par exemple, Microsoft Outlook.

Bien qu'il ne s'agisse pas de code Html à proprement parler, il est aussi possible de prédefinir un objet du message ainsi envoyé ou prévoir l'envoi d'une copie à un autre destinataire.

- Pour prédefinir un objet à l'e-mail :

```
<a href="mailto:editions@eni.com?subject=Formation Html5">Editions ENI</a> où le contenu de subject est l'objet (prédefini). Dans le cas de notre exemple, il s'agit de "Formation Html5".
```

- Pour définir une copie du message :

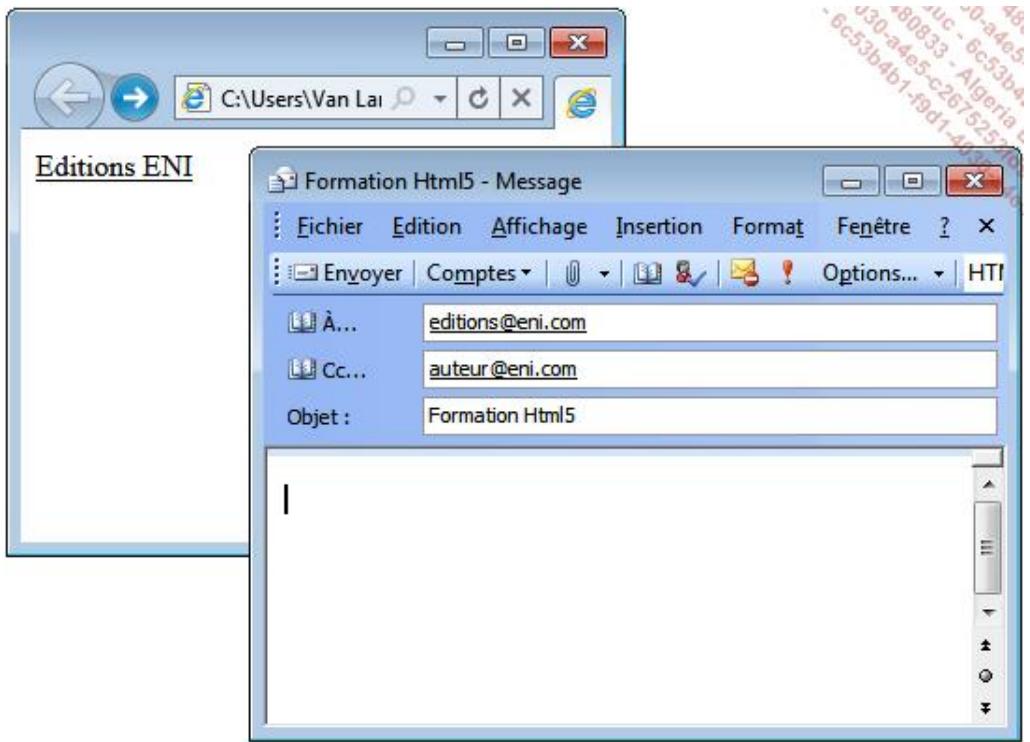
```
<a href="mailto:editions@eni.com?cc=auteur@eni.com">Editions ENI</a> où le contenu de cc est l'adresse électronique destinée à la copie.
```

- Pour combiner les possibilités :

```
<a href="mailto:editions@eni.com?subject=Formation Html5&cc=auteur@eni.com">Editions ENI</a>
```

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<a href="mailto:editions@eni.com?subject=Formation
Html5&cc=auteur@eni.com"> Editions ENI</a>
</body>
</html>
```



Le lien pour télécharger un fichier

Pour permettre de télécharger un fichier, il suffit de spécifier le nom (avec l'extension) de celui-ci. Cette façon de procéder est valable pour autant que le fichier Html et le fichier offert en téléchargement soient situés dans le même dossier ou sous-dossier.

Soit un fichier "formation.pdf" situé dans le même dossier. Le code devient :

```
<a href="formation.pdf">Version PDF</a>
```

Si une application qui lit les fichiers pdf (Adobe Acrobat Reader par exemple) n'est pas installée sur l'ordinateur du visiteur, le navigateur de celui-ci propose la fenêtre d'invite de téléchargement. Par contre, si l'application Adobe Acrobat Reader est présente sur le poste du visiteur, le navigateur ouvre l'application et affiche le fichier.

Il en sera de même pour tous les autres types de fichiers. Quand il n'y a pas d'application définie par défaut pour l'extension du fichier, le navigateur télécharge le fichier, après avoir proposé la fenêtre d'invite de téléchargement.

Le lien vers une fenêtre spécifique

Les pages ciblées par un lien peuvent s'ouvrir dans des fenêtres spécifiques grâce à l'attribut `target`.

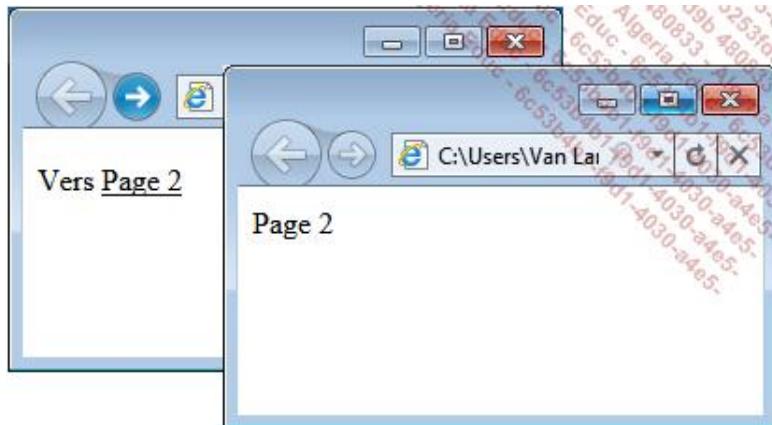
Passons en revue les différentes possibilités :

- `target="_self"` (défaut). La page cible du lien s'ouvre dans la même fenêtre que la page de départ du lien.
- `target="_top"`. La page cible du lien est affichée dans la même fenêtre mais occupera la totalité de la fenêtre d'affichage. À ce stade de notre étude, `self` et `top` ont un effet équivalent.
- `target="_blank"`. La page cible du lien s'ouvre dans une nouvelle instance du navigateur ou un nouvel onglet de celui-ci.

Nous illustrons cet attribut `target` et spécialement `target="_blank"`.

Dans le fichier Page 1 :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
Vers <a href="page2.htm" target="_blank">Page 2</a>
</body>
</html>
```



La couleur et le soulignement des liens

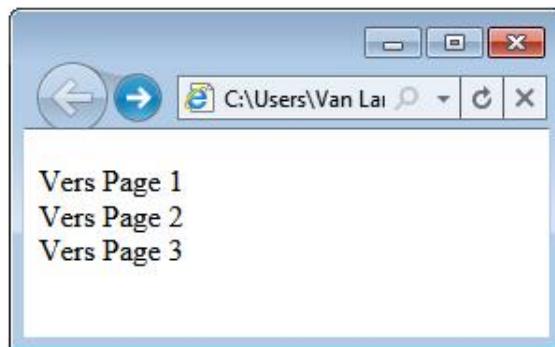
Héritage datant du début du Html, les liens apparaissent par défaut soulignés et de couleur bleue pour le lien non visité, de couleur pourpre pour le lien visité et rouge pour le lien actif.

Le soulignement est une convention majeure du Web pour indiquer les liens hypertextes. Pour ne pas dérouter l'internaute débutant, celle-ci doit être maintenue le plus souvent possible. Pourtant sous le plan esthétique, le soulignement systématique dans une page comportant de nombreux liens, peut alourdir la présentation de la page.

Il est possible d'enlever le soulignement par une simple feuille de style (`text-decoration: none`).

De même, pour uniformiser la couleur des liens, la propriété de style `color: valeur` peut être utilisée.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<style type="text/css">
a { text-decoration: none;
    color: black; }
</style>
<meta charset="iso-8859-1">
</head>
<body>
Vers <a href="page1.htm">Page 1</a><br>
Vers <a href="page2.htm">Page 2</a><br>
Vers <a href="page3.htm">Page 3</a><br>
</body>
</html>
```



Une infobulle sur un lien

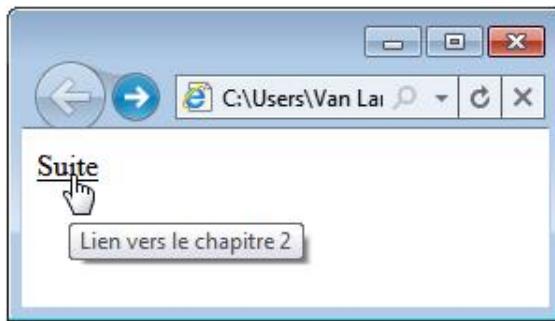
L'attribut title="texte" (qui n'est pas spécifique à la balise de lien) permet de proposer une infobulle explicative au survol du lien par le curseur de la souris. Ce petit "plus" en ergonomie est surtout apprécié en termes d'accessibilité, par les interfaces vocales qui guident les surfeurs non-voyants.

Initialement reprise par Internet Explorer, cette infobulle apparaît dans tous les navigateurs de notre étude.

```
<a href="page2.htm" title="lien vers le chapitre 2">Suite</a>
```

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<a href="page2.htm" title="Lien vers le chapitre 2">Suite</a>
</body>
</html>
```

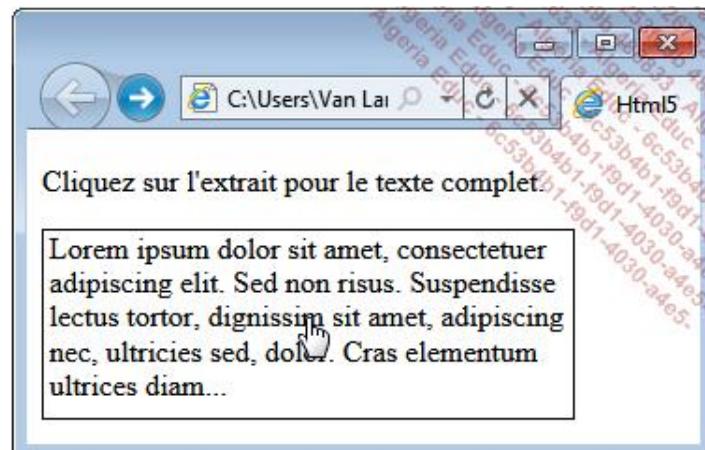


Les nouvelles possibilités de liens

Les liens peuvent maintenant porter sur n'importe quel élément de type bloc comme les divisions (`<div>`) et items de liste (``).

Ainsi, le code suivant est valide en Html5 alors qu'il ne l'est pas en Html 4.0.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="fr">
<head>
<title>Html5</title>
<style>
div { width: 280px;
      height: 100px;
      border: 1px solid black;
      padding-left: 3px;}
a { text-decoration: none;
    color: black; }
</style>
<meta charset="iso-8859-1">
</head>
<body>
<p>Cliquez sur l'extrait pour le texte complet.</p>
<a href="page2.htm">
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam...</div>
</a>
</body>
</html>
```



Le curseur en forme de main atteste l'existence d'un lien sur l'ensemble de la division.

Voilà qui risque d'être déroutant pour l'utilisateur quand ceci se mettra en place !

Préambule

Le mouvement s'était déjà amorcé avec le Html 4.0 et le Xhtml 1.0. Il se renforce avec le Html5. Les tableaux sont exclusivement réservés à la présentation de données. Il n'est plus question dans une application Web moderne d'utiliser les tableaux pour réaliser la disposition de la page. Ces tableaux de présentation, souvent imbriqués dans d'autres tableaux, étaient le défaut de la conception de pages Web dans ce que l'on peut appeler à postériori le Web 1.0. Ces multiples tableaux alourdissaient le code source et en rendaient la lecture quasi impossible. Par ailleurs, ces tableaux complexes compliquaient grandement la lecture des aides vocales utilisées par les personnes non voyantes. La disposition des éléments de la page est maintenant assurée par des balises <div>.

Rien de bien nouveau dans le Html5 par rapport au Html 4.0. Il faut cependant noter la disparition de très nombreux attributs de présentation qui doivent maintenant être impérativement repris par des feuilles de style CSS. Tout ceci sera détaillé dans la suite de ce chapitre.

Toutes les balises et tous les attributs du Html5 relatifs aux tableaux sont parfaitement compatibles avec les navigateurs du marché.

La création d'un tableau

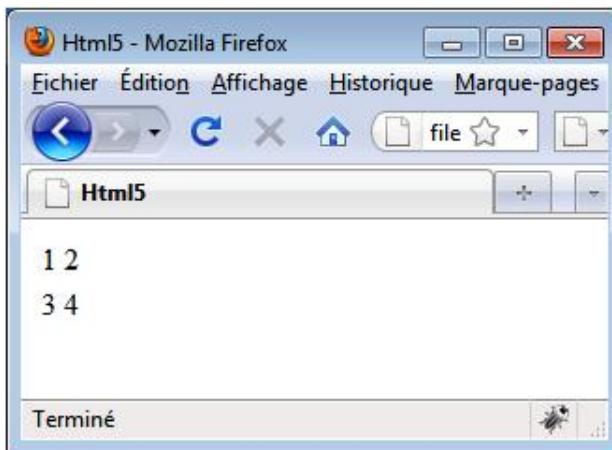
Un tableau (`<table>`) comporte des lignes `<tr>`. Ces lignes contiennent des cellules `<td>`.

L'agencement de ces différentes balises pour afficher un tableau suit une logique que l'exemple suivant nous permettra d'illustrer.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<table>
<tr>
<td>1</td><td>2</td>
</tr>
<tr>
<td>3</td><td>4</td>
</tr>
</table>
</body>
</html>
```

Le tableau en Html se construit ainsi :

- On déclare que l'on ouvre un tableau avec la balise `<table>`.
- Ce tableau comporte une première ligne introduite par la balise `<tr>`.
- Cette ligne comporte une première cellule que l'on ouvre avec la balise `<td>`.
- Soit 1 comme unique contenu de cette cellule.
- Fermeture de cette première cellule `</td>`.
- Passons à la seconde cellule `<td>2</td>`.
- Fin de la première ligne, donc `</tr>`.
- Vient la ligne suivante soit à nouveau `<tr>`.
- Cette ligne contient les cellules `<td>3</td><td>4</td>`.
- Fin de la seconde ligne `</tr>`.
- Et enfin la balise de fermeture du tableau `</table>`.



Commentaires

- Cette balise, basique en Html, est parfaitement compatible entre les différents navigateurs.
- La balise `<table>` doit obligatoirement être fermée. En cas d'oubli de la balise de fermeture `</table>`, le tableau risque de ne pas s'afficher ou ne pas s'afficher correctement.

Passons aux attributs de la balise `<table>`. À notre grande surprise, cette balise ne comporte plus d'attributs en Html5 ! En fait, nous aurions pu écrire, ne comporte plus d'attributs. Le Html 4.0 proposait les attributs `width`, `border`, `cellpadding`, `cellspacing`, `frame` et `rules`. Tous ces attributs ont disparu en Html5 et doivent être pris en charge par des feuilles de style CSS.

Les balises `<tr>` et `<td>` suivent le mouvement avec la disparition des attributs `align` et `valign`.

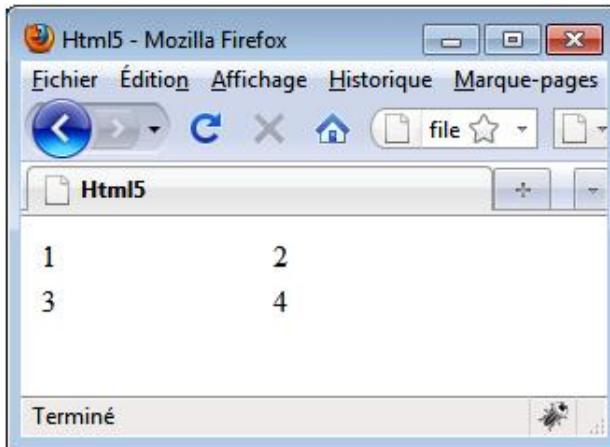
1. La largeur du tableau

La largeur du tableau est déterminée par la propriété de style CSS `width` (largeur).

Exemple

Le code précédent pour une largeur de tableau de 250 pixels devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { width: 250px; }
</style>
</head>
<body>
<table>
<tr>
<td>1</td><td>2</td>
</tr>
<tr>
<td>3</td><td>4</td>
</tr>
</table>
</body>
</html>
```



Il est également possible de déterminer la hauteur du tableau par la propriété de style `height`.

```
<style>
table { height: 150px; }
</style>
```

2. Les bordures du tableau

Les bordures d'un tableau et d'autres éléments Html sont introduits par la propriété de style CSS `border` (pour bordure).

Pour obtenir des bordures classiques d'un tableau, la démarche est un peu plus particulière.

Notre but est d'ajouter une bordure de 1 pixel (`1px`), de couleur noire (`black`) et avec un trait plein (`solid`).

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
        width: 250px; }
</style>
</head>
<body>
<table>
<tr>
<td>1</td><td>2</td>
</tr>
<tr>
<td>3</td><td>4</td>
</tr>
</table>
</body>
</html>
```

1	2
3	4

Le résultat ne répond pas tout à fait à nos attentes et c'est logique car nous avons demandé par la propriété de style d'ajouter une bordure au tableau (`table`). Il faut également demander d'ajouter une bordure aux cellules `<td>` qui constituent le tableau (`td`).

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
```

```

<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
         width: 250px;}
td { border: 1px solid black;
      width: 33%;}
</style>
</head>
<body>
<table>
<tr>
<td>1</td><td>2</td>
</tr>
<tr>
<td>3</td><td>4</td>
</tr>
</table>
</body>
</html>

```

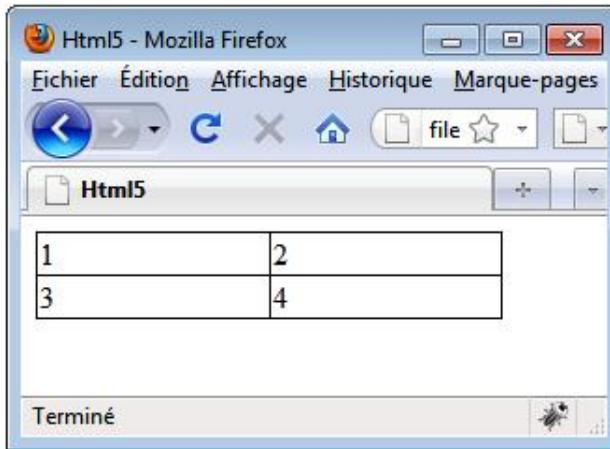
1	2
3	4

Ce qui ne correspond toujours pas au résultat escompté car il subsiste un espace entre la bordure du tableau et les bordures des cellules (vestige de l'attribut `cellspacing` pour les initiés). Pour réduire ces deux bordures en une seule, on utilise la propriété de style `border-collapse`.

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
        border-collapse: collapse;
        width: 250px;}
td { border: 1px solid black; }
</style>
</head>
<body>
<table>
<tr>
<td>1</td><td>2</td>
</tr>
<tr>
<td>3</td><td>4</td>
</tr>
</table>
</body>
</html>

```



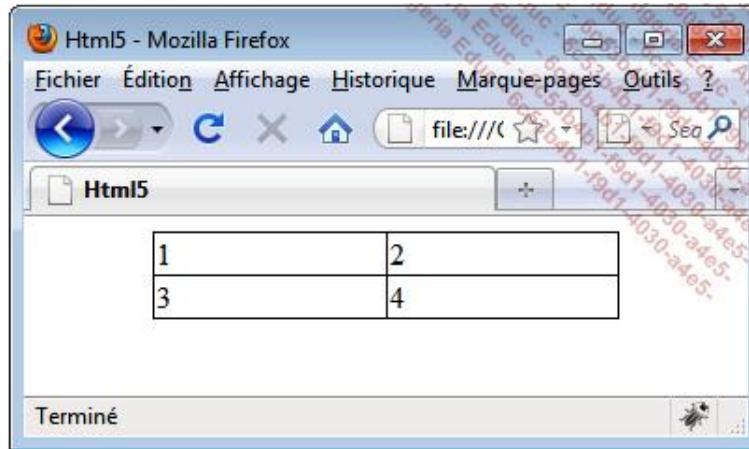
Il existe en CSS d'autres formes de bordures. Celles-ci seront détaillées dans la partie de cet ouvrage consacrée aux feuilles de style.

3. L'alignement du tableau

Le tableau est ici aligné à gauche. Pour un alignement centré, il faut passer par une astuce. Le fait de mettre une marge (margin) automatique à gauche (left) et à droite (right) du tableau provoque le centrage.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { margin-left: auto;
         margin-right: auto;
         border: 1px solid black;
         border-collapse: collapse;
         width: 250px;}
td { border: 1px solid black;}
</style>
</head>
<body>
<table>
<tr>
<td>1</td><td>2</td>
</tr>
<tr>
<td>3</td><td>4</td>
</tr>
</table>
</body>
</html>
```

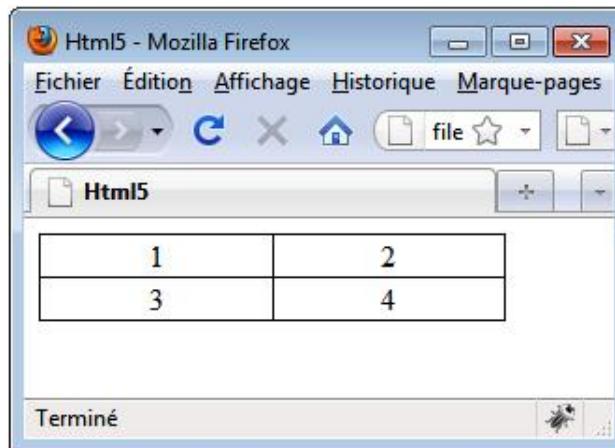


4. L'alignement des cellules du tableau

Le contenu des cellules est ici aligné à gauche. Il serait plus esthétique de le centrer (`center`) dans les cellules. La propriété de style `text-align` est alors utilisée.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { text-align: center;
        border: 1px solid black;
        border-collapse: collapse;
        width: 250px;}
td { border: 1px solid black;}
</style>
</head>
<body>
<table>
<tr>
<td>1</td><td>2</td>
</tr>
<tr>
<td>3</td><td>4</td>
</tr>
</table>
</body>
</html>
```



Commentaire

Avec `text-align: center` appliqué au tableau (`<table>`), toutes les cellules du tableau auront un contenu centré dans celles-ci.

Les cellules de tableaux

Les cellules peuvent contenir tous les éléments définis par le Html, soit du texte, des images, des liens, des arrière-plans et même des tableaux.

En Html5, les attributs `width`, `height`, `align`, `valign`, `abbr`, `axis` et `scope` du Html 4.0 ont disparu.

1. La largeur des cellules

Par défaut, le navigateur adapte la largeur des cellules selon leur contenu.

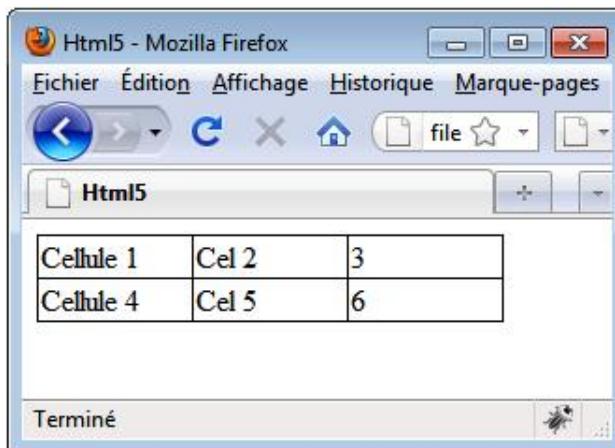
Soit l'exemple suivant d'un tableau de trois colonnes sans spécifications particulières.

Cellule 1	Cel 2	3
Cellule 4	Cel 5	6

Il faudra avoir recours à la propriété de style `width` mais cette fois appliquée à la balise `<td>` pour uniformiser la largeur des cellules.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
         border-collapse: collapse;
         width: 250px; }
td { border: 1px solid black;
     width: 33%; }
</style>
</head>
<body>
<table>
<tr>
<td>Cellule 1</td><td>Cel 2</td><td>3</td>
</tr>
<tr>
<td>Cellule 4</td><td>Cel 5</td><td>6</td>
</tr>
</table>
</body>
</html>
```



2. L'alignement horizontal

L'alignement horizontal du contenu d'une cellule peut être à gauche, centré ou à droite. Celui-ci est obtenu par la propriété de style `text-align` appliquée à une cellule. La propriété `text-align` peut prendre la valeur `left` (gauche), `center` (centré) ou `right` (droite). La valeur par défaut est `left`.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
         border-collapse: collapse;
         width: 290px;}
td { border: 1px solid black;
     width: 33%;}
</style>
</head>
<body>
<table>
<tr>
<td style="text-align: left;">Produit 1</td>
<td style="text-align: center;">En stock</td>
<td style="text-align: right;">3.43 €</td>
</tr>
<tr>
<td style="text-align: left;">Produit 2</td>
<td style="text-align: center;">En commande</td>
<td style="text-align: right;">6.72 €</td>
</tr>
</table>
</body>
</html>
```



La propriété de style a été appliquée ici directement dans le code à la balise `<td>` par `style="text-align: xxx;"` (style en ligne). Il y a d'autres façons d'implémenter cette propriété de style. Vous les découvrirez lors de l'étude plus détaillée des feuilles de style CSS.

3. L'alignement vertical

Par défaut, l'alignement vertical du contenu d'une cellule est le milieu de la cellule. Ce qui risque d'entraîner des situations peu esthétiques lors de l'élaboration de votre tableau.

1	xxxxx xxxx xxxx
2	xxxxx xxxx
3	xxxxx xxxx xxxx
	xxxxx

Il est prudent de prévoir un alignement vertical vers le haut par la propriété de style `vertical-align: top` appliquée aux cellules `<td>`.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
         border-collapse: collapse;
         width: 140px;}
td { border: 1px solid black;
     vertical-align:top;}
</style>
</head>
<body>
<table>
<tr>
<td style="width: 10%;">1</td>
<td>xxxxx xxxx xxxx xxxx xxxx</td>
</tr>
<tr>
<td style="width: 10%;">2</td>
<td>xxxxx xxxx</td>
</tr>
<tr>
<td style="width: 10%;">3</td>
<td>xxxxx xxxx xxxx xxxx xxxx xxxx xxxx</td>
</tr>
</table>
</body>
</html>
```



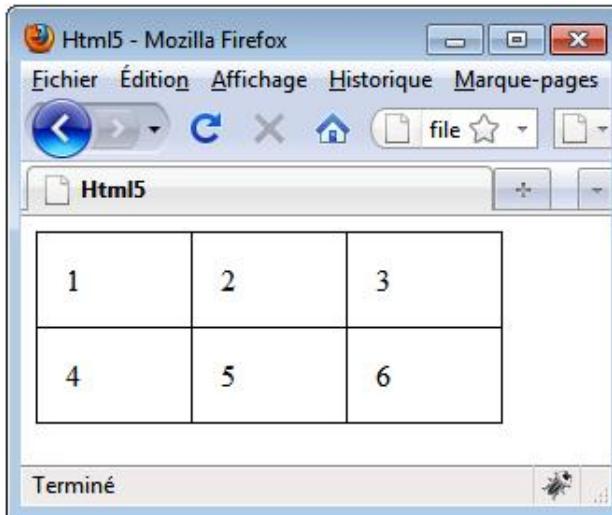
4. La marge intérieure des cellules

Dans nos captures d'écran, le contenu de la cellule est accolé à la bordure de celle-ci. Ce qui n'est pas très esthétique et nuit à la lisibilité des données. La propriété de style `padding` permet d'ajouter de l'espace entre le contenu de la cellule et la bordure de celle-ci.

Il est important de distinguer la marge extérieure qui est désignée par `margin` et la marge intérieure qui est reprise sous le vocable `padding`.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
         border-collapse: collapse;
         width: 250px;}
td { border: 1px solid black;
     padding: 15px;}
</style>
</head>
<body>
<table>
<tr>
<td>1</td><td>2</td><td>3</td>
</tr>
<tr>
<td>4</td><td>5</td><td>6</td>
</tr>
</table>
</body>
</html>
```



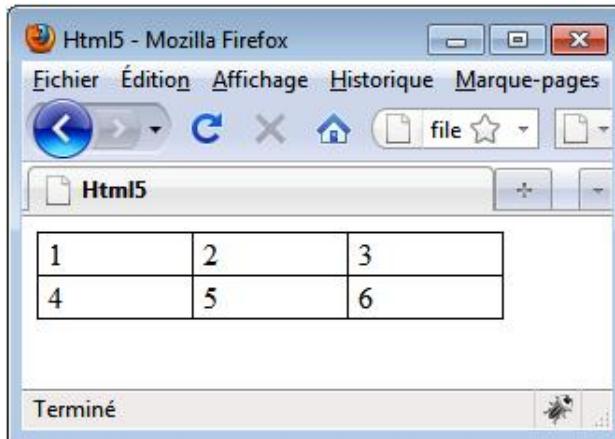
On remarque que la propriété de style `padding` a ajouté de l'espace tout autour du contenu de la cellule. Il est possible de n'ajouter un espace qu'entre la bordure gauche et le contenu par la propriété `padding-left`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { text-align: center;
         border: 1px solid black;
```

```

border-collapse: collapse;
width: 250px;}
td { border: 1px solid black;
padding-left: 5px;}
</style>
</head>
<body>
<table>
<tr>
<td>1</td><td>2</td><td>3</td>
</tr>
<tr>
<td>4</td><td>5</td><td>6</td>
</tr>
</table>
</body>
</html>

```



5. L'arrière-plan de couleur

Depuis la disparition de l'attribut `bgcolor`, l'ajout d'une couleur d'arrière-plan doit passer par la propriété de style `background-color`.

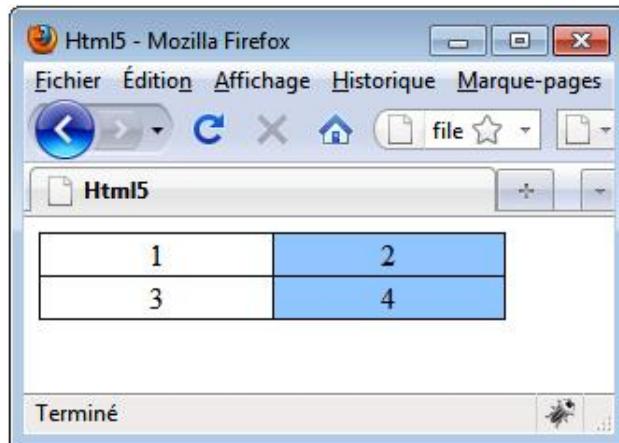
Exemple

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { text-align: center;
border: 1px solid black;
border-collapse: collapse;
width: 250px;}
td { border: 1px solid black;}
.couleur { background-color: #99ccff; }
</style>
</head>
<body>
<table>
<tr>
<td>1</td>
<td class="couleur">2</td>
</tr>
<tr>
<td>3</td>
<td class="couleur">4</td>
</tr>

```

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



Cette propriété de style d'arrière-plan `background-color` peut aussi s'appliquer aux balises `<table>` et `<tr>`.

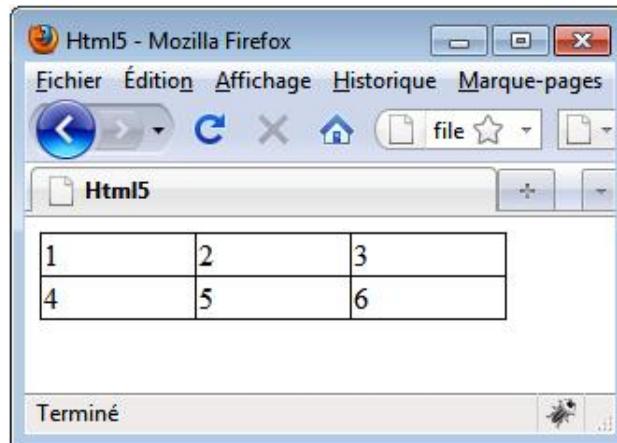
Nous en apprendrons plus sur la notation des couleurs au chapitre Notions de base des CSS - La notation des couleurs.

La fusion des cellules

Il est possible de fusionner horizontalement ou verticalement des cellules.

Soit un tableau de départ de deux lignes et trois colonnes.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
         border-collapse: collapse;
         width: 250px;}
td { border: 1px solid black;}
</style>
</head>
<body>
<table>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</table>
</body>
</html>
```



1. La fusion de colonnes

Pour fusionner des colonnes, le Html5 dispose de l'attribut de cellule `colspan="x"` où `x` correspond au nombre de colonnes que l'on souhaite fusionner horizontalement.

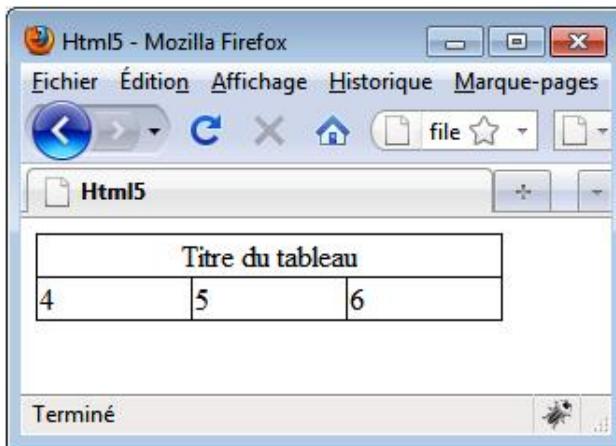
Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
```

```

table { border: 1px solid black;
    border-collapse: collapse;
    width: 250px;}
td { border: 1px solid black;}
</style>
</head>
<body>
<table>
<tr>
<td colspan="3" style="text-align: center;">Titre du tableau</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</table>
</body>
</html>

```



2. La fusion de lignes

Pour fusionner des lignes, le Html5 dispose de l'attribut de cellule `rowspan="x"` où `x` correspond au nombre de lignes que l'on souhaite fusionner verticalement.

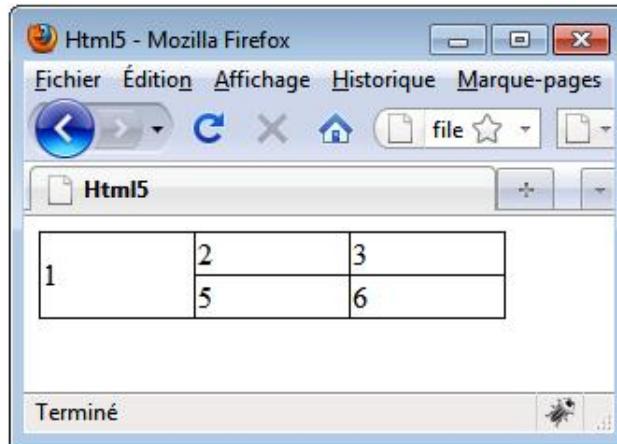
Exemple

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
    border-collapse: collapse;
    width: 250px;}
td { border: 1px solid black;}
</style>
</head>
<body>
<table>
<tr>
<td rowspan="2">1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>6</td>

```

```
</tr>
</table>
</body>
</html>
```



Commentaire

La première colonne des deux lignes est fusionnée. Remarquez l'alignement vertical centré par défaut.

Les cellules d'en-tête

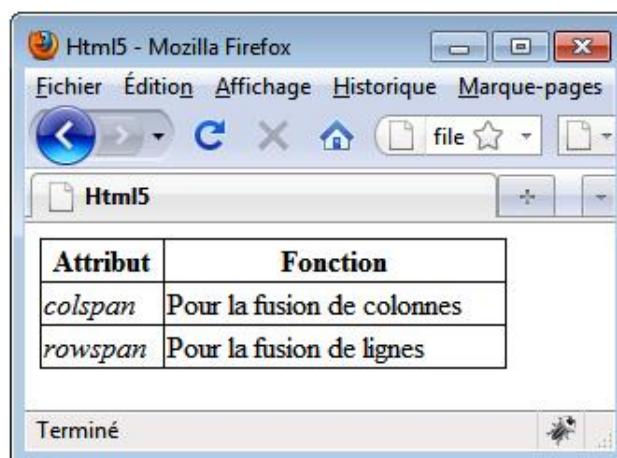
Les cellules d'en-tête de colonnes ou de lignes sont définies par la balise `<th> ... </th>`. Elles fournissent, en quelque sorte, un titre aux données de la colonne ou de la ligne.

Dans les navigateurs visuels, le contenu de la balise `<th>` apparaît en gras et en centré. C'est peut-être pour cette raison que celle-ci est souvent négligée par les concepteurs. Elle fait pourtant partie historiquement des balises d'encodage des tableaux en Html. Notons que leur apparence peut toujours être modifiée par des feuilles de style CSS.

La balise `<th>` a été gardée dans la spécification du Html5 car elle est d'une grande utilité avec les lecteurs d'écran utilisé par les personnes non voyantes. Elle contribue ainsi à l'accessibilité des sites web et particulièrement des tableaux qui, s'ils sont d'une certaine ampleur, posent vite problème lorsqu'ils sont découverts par l'audition.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
         border-collapse: collapse;
         width: 250px; }
td { border: 1px solid black; }
</style>
</head>
<body>
<table>
<tr>
<th>Attribut</th>
<th>Fonction</th>
</tr>
<tr>
<td><i>colspan</i></td>
<td>Pour la fusion de colonnes</td>
</tr>
<tr>
<td><i>rowspan</i></td>
<td>Pour la fusion de lignes</td>
</tr>
</table>
</body>
</html>
```



Le résumé du tableau

L'attribut `summary` de la balise `<table>` permet de fournir un résumé du tableau.

Ce résumé n'apparaîtra en aucune façon dans la présentation visuelle de la page. Celui-ci n'est pris en charge que par les synthèses vocales et autres barrettes braille pour fournir un résumé du tableau avant l'exploration des données par les personnes visuellement déficientes. C'est une fois de plus un outil pour l'accessibilité des sites.

Notons qu'en Html5, l'attribut `summary` est le seul et unique attribut de la balise `<table>`.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
         border-collapse: collapse;
         width: 250px; }
td { border: 1px solid black; }
</style>
</head>
<body>
<table
summary="Le tableau présente les attributs utilisés pour fusionner
les lignes et les colonnes d'un tableau Html">
<tr>
<th>Attribut</th>
<th>Fonction</th>
</tr>
<tr>
<td><i>colspan</i></td>
<td>Pour la fusion de colonnes</td>
</tr>
<tr>
<td><i>rowspan</i></td>
<td>Pour la fusion de lignes</td>
</tr>
</table>
</body>
</html>
```

La capture d'écran est identique à celle du point précédent étant donné que cet attribut n'a aucune action visuelle.

La légende d'un tableau

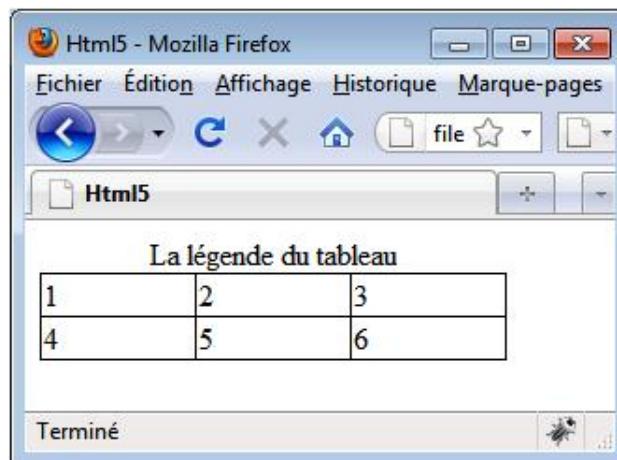
La balise `<caption>` permet d'associer un titre ou une légende au tableau.

La balise `<caption>` doit être placée juste après la balise ouvrante `<table>` et ne peut apparaître qu'une fois dans le tableau.

Par défaut, la plupart des navigateurs affichent le contenu de la balise `<caption>` de façon centrée au-dessus du tableau. Il est possible de mettre cette légende en dessous du tableau à l'aide de la propriété de style CSS `caption {caption-side: bottom;}`.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
         border-collapse: collapse;
         width: 250px; }
td { border: 1px solid black; }
</style>
</head>
<body>
<table>
<caption>La légende du tableau</caption>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</table>
</body>
</html>
```



Le groupement de colonnes

La balise `<colgroup>` est utilisée pour grouper des colonnes en vue de leur appliquer une mise en forme à l'ensemble des balises du groupe par l'entremise d'une feuille de style CSS.

L'attribut `span="x"` détermine le nombre de colonnes ainsi groupées. Par défaut, la valeur de `x` est 1. Cet attribut `span` est le seul attribut de la balise `<colgroup>`. Les attributs `align`, `valign`, `width` du HTML 4.0 ont disparu.

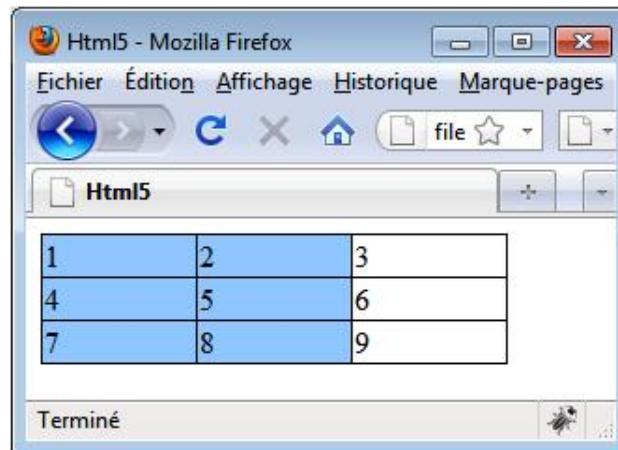
En outre, la balise `<colgroup>` se positionne juste après la balise `<table>` (ou juste après la balise `<caption>` si celle-ci est utilisée) et avant toute balise `<tr>` ou `<td>`.

Par définition, la balise `<colgroup>` ne s'applique que pour les tableaux.

Exemple 1

Soit un tableau de trois lignes et trois colonnes. Un arrière-plan de couleur ainsi qu'un alignement centré sont appliqués aux deux premières colonnes.

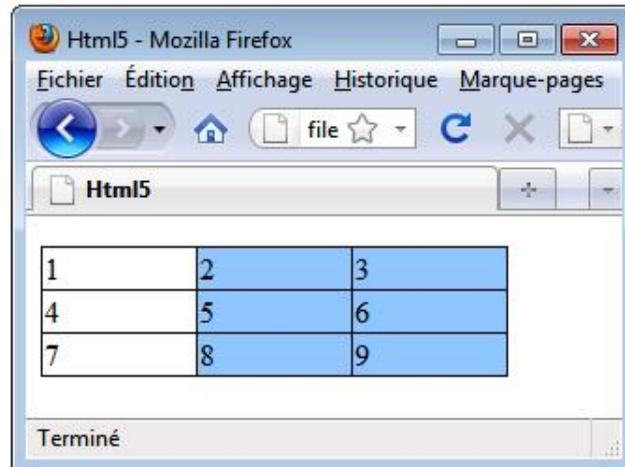
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
         border-collapse: collapse;
         width: 250px; }
td { border: 1px solid black; }
colgroup { background-color : #99ccff;
           text-align: center; }
</style>
</head>
<body>
<table>
<colgroup span="2"></colgroup>
<tr>
<td>1</td><td>2</td><td>3</td>
</tr>
<tr>
<td>4</td><td>5</td><td>6</td>
</tr>
<tr>
<td>7</td><td>8</td><td>9</td>
</tr>
</table>
</body>
</html>
```



Exemple 2

Si l'on souhaite regrouper les deux dernières colonnes, il faut déclarer plusieurs balises <colgroup>.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
         border-collapse: collapse;
         width: 250px; }
td { border: 1px solid black; }
#colonnes { background-color : #99ccff;
            text-align: center; }
</style>
</head>
<body>
<table>
<colgroup span="1"></colgroup>
<colgroup id="colonnes" span="2"></colgroup>
<tr>
<td>1</td><td>2</td><td>3</td>
</tr>
<tr>
<td>4</td><td>5</td><td>6</td>
</tr>
<tr>
<td>7</td><td>8</td><td>9</td>
</tr>
</table>
</body>
</html>
```



La structuration d'un tableau

Le HTML5 prévoit des balises pour structurer de façon logique le contenu d'un tableau.

Ces balises sont :

- `<thead>` pour regrouper les informations concernant l'en-tête du tableau comme par exemple le titre et l'intitulé des colonnes.
- `<tbody>` pour le corps du tableau, soit l'ensemble des données de celui-ci.
- `<tfoot>` pour le contenu de bas de page comme par exemple des remarques ou une légende.

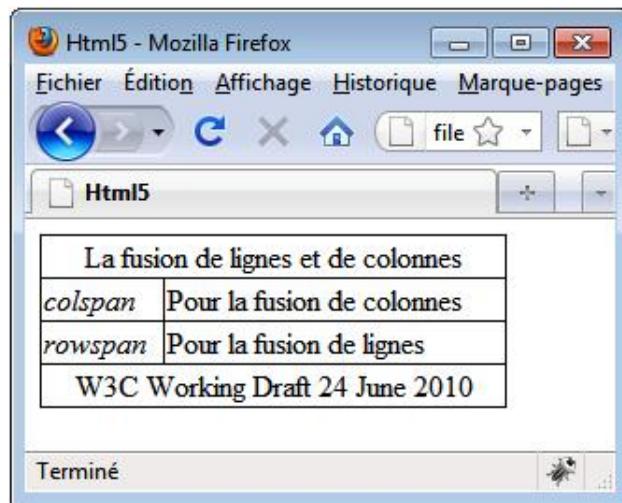
Ces divisions logiques n'affectent en rien la présentation par défaut du tableau mais elles peuvent être reprises par des propriétés de style pour agrémenter la présentation du tableau.

On notera que la balise `<tfoot>` doit se situer avant la balise `<tbody>` de sorte que le navigateur puisse prévoir le pied de page avant la réception des lignes de données.

Exemple

Appliquons ces balises à un tableau.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>HTML5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
         border-collapse: collapse;
         width: 250px; }
td { border: 1px solid black; }
</style>
</head>
<body>
<table>
<thead>
<tr>
<td colspan="2" style="text-align: center;">La fusion de lignes et
de colonnes</td>
</tr>
</thead>
<tfoot>
<td colspan="2" style="text-align: center;">W3C Working Draft 24
June 2010</td>
</tfoot>
<tbody>
<tr>
<td><i>colspan</i></td>
<td>Pour la fusion de colonnes</td>
</tr>
<tr>
<td><i>rowspan</i></td>
<td>Pour la fusion de lignes</td>
</tr>
</tbody>
</table>
</body>
</html>
```



Ce tableau ne présente rien de particulier. Le même tableau avec des propriétés de style affectant les balises `<thead>` et `<tfoot>` devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
table { border: 1px solid black;
         border-collapse: collapse;
         width: 250px; }
td { border: 1px solid black; }
thead { background-color: #99ccff;
         font-size: 1.4em; }
tfoot { font-size: 0.7em; }
</style>
</head>
<body>
<table>
<thead>
<tr>
<td colspan="2" style="text-align: center;">La fusion</td>
</tr>
</thead>
<tfoot>
<tr>
<td colspan="2" style="text-align: center;">Source : W3C Working
Draft 24 June 2010</td>
</tr>
</tfoot>
<tbody>
<tr>
<td><i>colspan</i></td>
<td>Pour la fusion de colonnes</td>
</tr>
<tr>
<td><i>rowspan</i></td>
<td>Pour la fusion de lignes</td>
</tr>
</tbody>
</table>
</body>
</html>
```



Commentaire

La balise <tbody> est indispensable pour accéder au contenu d'un tableau en JavaScript.

Les formats d'image

Pour réduire au maximum le temps de téléchargement, le Web réclame des formats de fichier particuliers ; le format GIF, JPEG et PNG. Ces trois types de format comportent tous un algorithme de compression.

Sans déborder sur le sujet du traitement des images numériques qui dépasserait le cadre de cet ouvrage, détaillons brièvement ces trois formats.

Le format GIF

Le format GIF (pour *Graphics Interchange Format*) a connu son heure de gloire aux débuts du Web. Il était alors parfaitement adapté aux cartes graphiques de l'époque (souvent maximum 256 couleurs) et aux faibles capacités des lignes de téléphone analogiques (taux de compression élevé).

Caractéristiques

Le format GIF ne permet de coder les images qu'en maximum 256 couleurs. Ce qui n'est plus très réaliste avec les développements et les possibilités des images numériques.

- C'est un format de compression efficace et rapide. Les images GIF sont d'un poids en octet réduit.
- La compression s'effectue sans perte de données et sans perte de qualité.
- La fonction dite entrelacée permet un affichage rapide d'images qui deviennent progressivement de plus en plus nettes.
- La possibilité, sous la spécification GIF89a, de définir une des 256 couleurs comme étant transparente. À l'affichage apparaissent alors les éléments qui sont placés sous le graphique aux endroits où l'image est transparente.
- Les images sont facilement éditables et modifiables vu le nombre limité de couleurs.
- Seul format qui permette de construire des animations, sortes de petits dessins animés, avec ce qu'on appelle les GIFS animés.
- Le format GIF fait l'objet d'un brevet détenu par UNISYS.

Conclusion

En raison de sa limitation à 256 couleurs, le format GIF n'est pas du tout approprié aux graphiques à haute résolution comme les photos, les images nuancées et les dégradés. Par contre, il excelle encore pour les petits logos, boutons, puces, barres, symboles, et autres cliparts fréquents pour certains graphiques du Web qui ne nécessitent pas une palette de couleurs très étendue. Il est cependant de plus en plus supplanté par le format PNG.

Le format JPEG

Le format JPEG figure pour *Joint Photographic Expert Group*, donc non pas pour le nom du format graphique mais pour le nom de la corporation qui a développé ce format. C'est le type de format le plus répandu dans l'univers de la photo numérique.

Caractéristiques

- Le format JPEG permet de sauvegarder jusqu'à 16,7 millions de couleurs par image.
- Son taux de compression est efficace en ce qui concerne la taille résultante.
- Il permet de faire varier le taux de compression selon les besoins du concepteur. Une compression de 15 à 20 % fournira une image de qualité satisfaisante pour une taille de fichier réduite.
- Revers de la médaille, il y a une perte de qualité en fonction du taux de compression. Plus le facteur de compression est élevé, plus la qualité de l'image est mauvaise. C'est donc une compression destructive.

- Des déformations (*artifacts* ou artefacts) apparaissent souvent dans les passages de couleurs tranchés, donc sur les coins et les bords des objets.
- Depuis quelques années, la fonction entrelacée est implémentée, ce qui permet un affichage progressif et plus rapide sur le Web.
- Le format JPEG n'offre pas de possibilité de transparence ni d'animations.

Conclusion

- Le JPEG est un excellent format pour les photos, spécialement celles où il y a des tonalités diverses et des dégradés de couleurs. Mais il n'est pas exempt de défauts avec ses pertes de données et ses déformations.
- À cause des pertes de données et des déformations, il est impératif de respecter la règle de sauvegarder l'image originale et de travailler uniquement sur des copies de celle-ci.

Le format PNG

Le format PNG, pour *Portable Network Graphic*, est présenté comme le format du futur. Il s'agit d'un format graphique conçu spécialement pour la mise en œuvre sur le Web. La recommandation du W3C concernant le PNG remonte à 1996. Le PNG doit théoriquement rassembler tous les avantages de GIF et du JPEG. Il commence à être largement retenu par les concepteurs d'applications Web.

Caractéristiques

- Le format PNG soutient 16,7 millions de couleurs comme le format JPEG.
- Le format PNG est une spécification conçue spécialement pour Internet et est l'objet d'une recommandation du W3C.
- Sa compression est performante. Le taux de compression du format PNG est de 5 % à 25 % supérieur à celui du format GIF.
- Cette compression s'effectue sans pertes de données et de qualité.
- Il permet la transparence jusqu'à 256 couleurs.
- Le format PNG est un format ouvert et n'est pas breveté.
- Sa fonction entrelacée permet un affichage progressif.
- Il ne peut générer des images animées.

Conclusion

Le format PNG est appelé à être de plus en plus utilisé à l'avenir car il représente un excellent compromis entre le format GIF et JPEG. Les professionnels l'adoptent largement en lieu et place du GIF et pour les petites photos.

Le format WebP

Le WebP est un nouveau format d'image qui risque de prendre de l'importance dans les prochaines années. Ce format est développé par Google à partir d'un des formats de la nouvelle balise `<video>` du HTML5 (le WebM). Selon les premiers essais, le WebP réduirait la taille des fichiers de 39 % par rapport aux formats JPEG, PNG et GIF, sans perte de qualité perceptible.

Le poids des photos

Il semble illusoire de charger directement une photo de votre appareil photo numérique sur un site. Certaines "pèsent" jusqu'à 4 Mo, ce qui, même avec une connexion rapide, prendra un temps certain de chargement.

On peut conseiller :

- De recadrer les images pour ne recueillir que ce qui est utile.
- De redimensionner cette image recadrée.
- D'adopter un taux de compression d'environ 15 à 20 % pour les images JPEG. La perte de qualité n'est quasiment pas perceptible à l'œil normal.
- De diminuer le nombre de couleurs de l'image, si l'opération s'y prête.

Ces différentes manipulations devraient ramener la taille du fichier à une dimension beaucoup plus raisonnable et améliorer sensiblement le temps de téléchargement.

L'insertion d'une image

L'insertion d'une image est obtenue par la balise :

```

```

La balise `` est une balise unique qui ne comporte pas de balise de fermeture. La notation Xhtml 1.0 `` est aussi acceptée en Html5.

En Html5, la balise image `` ne comporte qu'un seul attribut obligatoire :

L'attribut `src="adresse du fichier image"`. En Html, l'image n'est pas directement incluse dans le document mais provient d'un fichier externe, appelé par l'attribut `src`. Celui-ci précise l'adresse relative ou absolue du fichier image qui doit être affiché dans la page.

Ainsi, pour une image située dans le même répertoire que la page Html (adressage relatif), on écrira :

```

```

ou

```

```

Pour une image située sur un site du Web (adressage absolu), on écrira :

```

```

ou

```
<img src=http://www.html5.com/images/image.png />
```

Les règles d'adressage sont identiques à celles abordées au chapitre "Les liens".

Les attributs facultatifs de la balise image sont :

- Les attributs `height` et `width`. Ces attributs permettent de définir en pixels, respectivement la hauteur et la largeur de l'image à l'écran. La définition de ces attributs permet au navigateur de réserver un emplacement pour l'image avant son téléchargement complet, tout en lui permettant de continuer la composition de la page et à afficher le texte de celle-ci. En définissant la dimension de l'image, le chargement est accéléré et donne matière à patienter aux internautes impatients. Bien que facultatifs, ces attributs sont considérés comme essentiels par les concepteurs professionnels.

Si le fichier n'est pas disponible ou si l'adressage est erroné, une icône sera affichée en lieu et place de l'image souhaitée.

- L'attribut `alt="texte associé"`. Cet attribut contient une brève description de l'image. Ce texte est surtout prévu pour les personnes non voyantes car il sera lu par les interfaces vocales. Plus récemment encore, le contenu de l'attribut `alt` est aussi utilisé par les moteurs de recherche (dont Google) pour le référencement et le classement d'un site ainsi que pour alimenter leur banque d'images.

Obligatoire en Html 4.0 et en Xhtml 1.0, est (re)devenu facultatif en Html5. Ce caractère facultatif alimente de nombreuses réactions, discussions et controverses dans les forums spécialisés. Rappelons que la recommandation Html5 n'est pas encore définitive au moment de l'écriture de cet ouvrage.

Les attributs de présentation `align`, `border`, `hspace` et `vspace`, déjà dépréciés (*deprecated*) en Html 4.0, ne sont plus repris dans le Html5. Il faudra faire appel à des feuilles de style CSS pour remplacer ces attributs disparus.

Pour les spécialistes du Html, signalons également que l'attribut `longdesc` qui permettait d'ajouter une description détaillée à l'attention des synthèses vocales a également disparu.

Exemple

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

```
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>

</body>
</html>
```



L'image globe.jpg est disponible dans l'espace de téléchargement.

L'insertion d'un lien sur une image

Les liens à partir d'une image se réalisent simplement en entourant celle-ci par la balise de lien `<a> ... `.

Certains navigateurs continuent à entourer cette image d'une bordure (inesthétique) pour signaler le lien. Internet Explorer 8 et 9 ainsi que Firefox 3.6 perpétuent cette façon de procéder. Les autres navigateurs de notre étude (Firefox 4, Opera 10, Safari 5 et Chrome 7) n'ajoutent plus cette bordure.



Pour enlever cette bordure, il faut passer en Html5 par une déclaration de style. Dans l'attente de l'étude détaillée des feuilles de style, il suffit d'ajouter à la balise `` la déclaration `style="border: none;"`.

Le code complet devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<a>

</a>
</body>
</html>
```

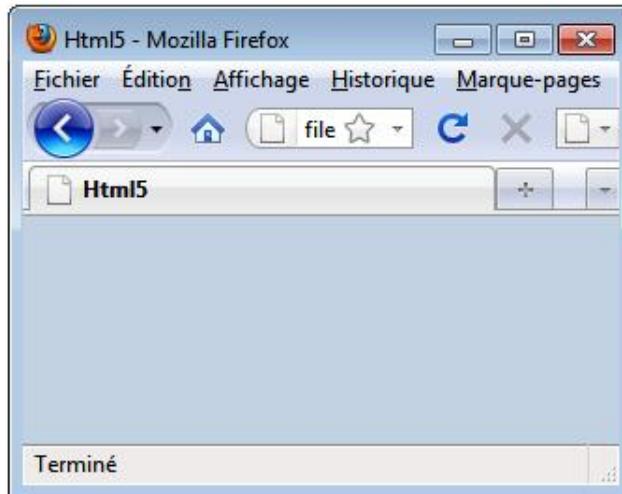


L'insertion d'une couleur d'arrière-plan

Depuis la disparition de l'attribut `bgcolor`, il n'est plus possible d'ajouter un arrière-plan de couleur à un élément. Il faut impérativement passer par les propriétés de style CSS, `background-color` en l'occurrence.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<style>
body {background-color: rgb(200,215,230);}
</style>
<body>
</body>
</html>
```



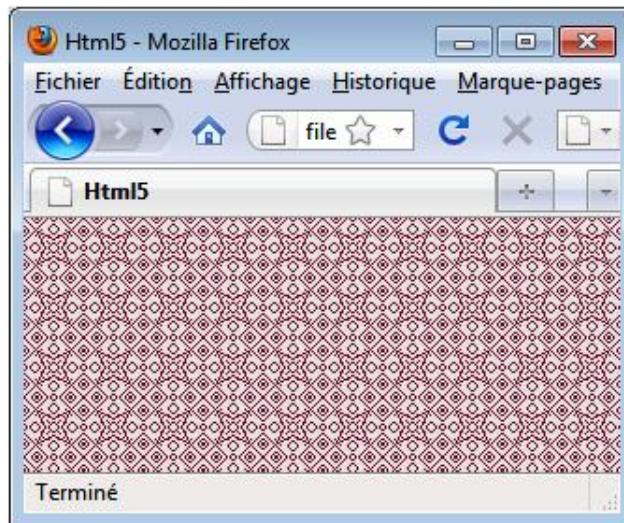
Les feuilles de style relatives aux arrière-plans seront étudiées en détail dans la partie consacrée aux CSS.

L'insertion d'une image d'arrière-plan

L'attribut `background` étant déclaré obsolète en HTML5, il n'y a d'autre solution que de passer par la propriété de style `background-image` pour insérer une image d'arrière-plan.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>HTML5</title>
<meta charset="iso-8859-1">
</head>
<style>
body {background-image: url('bg.gif');}
</style>
<body>
</body>
</html>
```



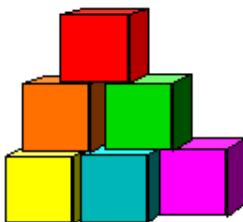
Les images réactives

1. Notion

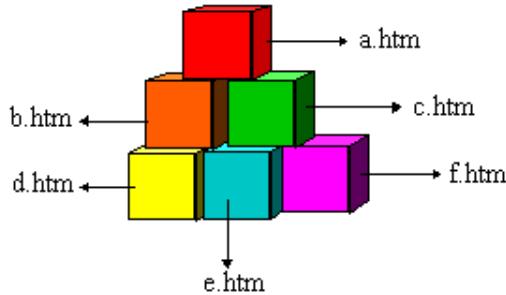
Les images réactives permettent d'effectuer des liens distincts en fonction d'une partie d'image ou d'une zone (*area*) prédefinie de celle-ci. Ainsi, par exemple, sur une carte de la France, on peut imaginer qu'un clic sur celle-ci mènerait le visiteur vers un fichier avec le nom et les informations du département concerné.

Il faut signaler que l'emploi des images réactives devient de plus en plus rare dans les applications Web actuelles. Les problèmes qu'elles posent pour un référencement efficace par les moteurs de recherche y sont pour beaucoup.

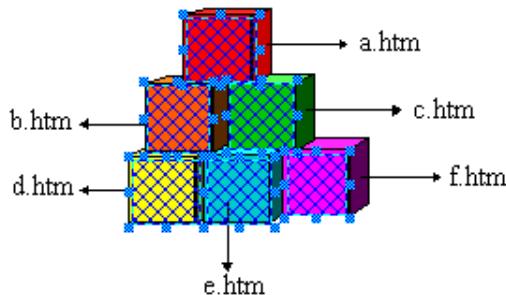
On prend une image.



Pour chaque zone retenue (ici un cube), on va associer un fichier.



Pour ce faire, on va définir des zones dans l'image, un peu comme avec une carte de géographie (*map* en anglais) et associer à chacune de ces zones un fichier Html.



On obtient ainsi une image réactive, dite aussi "mapée", car elle est découpée en zones à l'instar des cartes géographiques. On appellera la carte (*map*), les balises Html qui reprennent à la fois les zones de l'image et les fichiers associés.

Il existe plusieurs méthodes pour réaliser cet effet d'images réactives : les méthodes NCSA, CERN et CSIM. La méthode CSIM (*Client Side Image Map*) est de loin la plus utilisée à l'heure actuelle car elle fait partie à part entière du langage Html. Avec cette méthode, les fichiers de la carte sont inclus dans la page Html et ne nécessitent pas l'appel à des ressources additionnelles côté-serveur pour réaliser leurs fonctions. C'est cette méthode que nous utiliserons.

2. Les balises et attributs

Les balises des images réactives peuvent sembler très hermétiques pour l'utilisateur moyen. Cependant, une fois décodées, elles déterminent, en peu d'éléments, tout ce dont le navigateur a besoin pour les traiter.

La balise de l'image réactive

```

```

En fait, on ajoute simplement à la balise classique de l'image l'attribut `usemap` pour avertir le navigateur qu'il doit employer pour celle-ci une carte (`usemap`) et le nom de la carte en question. Cette carte sera alors incluse directement dans le fichier, un peu comme une ancre de lien. Le code reprend d'ailleurs la notation avec le dièse #, propre aux ancrées.

Les balises de la carte

Pour rappel, la carte (`map`) est découpée en zones (`area`).

```
<map>
```

```
<map id="nom_de_la_carte">
<area shape="forme" coords="coordonnées" href="destination">
... autres balises area ...
</map>
```

Reprenons les éléments un par un :

- La balise `<map id="nom_de_la_carte"> ... </map>`.

On donne un nom à la carte (`map`) par l'identifiant `id` et c'est ce nom qui sera retenu dans l'attribut `usemap`, vu quelques lignes plus haut.

- La balise `<area shape="forme" coords="coordonnées" href="destination">`.

```
<area>
```

Cette balise détermine les zones réactives de l'image.

- L'attribut `shape` de `<area>`

L'attribut `shape="forme"` détermine la forme de la zone :

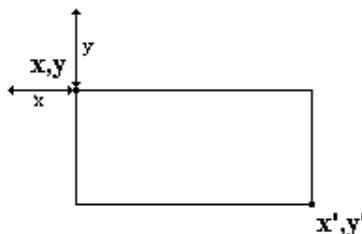
- `rect` pour un rectangle.
- `circle` pour un cercle.
- `polygon` pour un polygone irrégulier.
- `default` pour gérer les clics effectués en dehors d'une des zones réactives.

- L'attribut `coords` de `<area>`

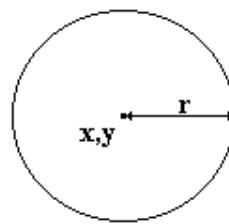
L'attribut `coords="coordonnées"` note les coordonnées qui permettront au navigateur de reconstituer la forme géométrique.

Nous détaillons ci-après la façon, spécifique à chaque méthode, de noter ces coordonnées. La véritable difficulté pour le webmestre est de trouver les coordonnées des points dans l'image. C'est ici qu'une application de traitement de l'image, comme Adobe Photoshop ou Gimp, se révèle utile.

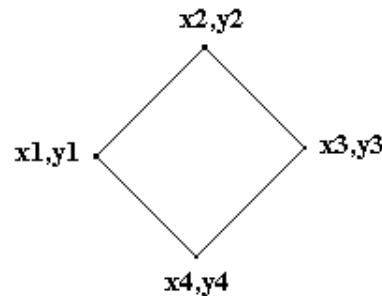
Pour un rectangle : `coords="x,y,x',y"`.



Pour un cercle : coords="x,y,r" soit le point central et le rayon.



Pour un polygone : coords="x1,y1,x2,y2,x3,y3,x4,y4" et ce pour autant de points qu'il y a de sommets dans le polygone.



- L'attribut href de <area>

L'attribut href="destination" spécifie le fichier associé à la zone sélectionnée. L'adressage se fait de la façon tout à fait classique en Html. Facultatif si le concepteur souhaite qu'un clic sur une zone n'entraîne aucune action.

- L'attribut alt de <area>

L'attribut alt="commentaire" (facultatif en Html5) permet d'ajouter une illustration textuelle.

3. Un exemple complet

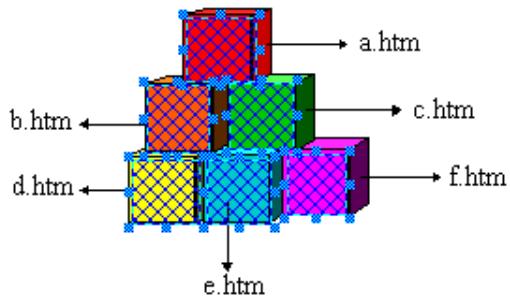
Soit le fichier Html5 suivant :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<p>

</p>
</body>
</html>
```

Et la carte nommée "cartons" de l'image "cubes.gif" :

```
<map id="cartons">
<area shape="rect" coords="37,9,72,40" href="a.htm" alt="">
<area shape="rect" coords="18,46,47,79" href="b.htm" alt="">
<area shape="rect" coords="61,43,93,78" href="c.htm" alt="">
<area shape="rect" coords="9,84,36,119" href="d.htm" alt="">
<area shape="rect" coords="48,85,77,116" href="e.htm" alt="">
<area shape="rect" coords="89,81,123,115" href="f.htm" alt="">
</map>
```



Incluons la carte (`map`) dans le fichier Html5 pour créer l'image réactive.

Cette inclusion se réalise en deux temps :

On ajoute d'abord l'information pour le navigateur `usemap="#cartons"` dans la balise de l'image.

On inclut ensuite la carte n'importe où dans le fichier Html5.

Le fichier final devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<p>

</p>
<map id="cartons">
<area shape="rect" coords="37,9,72,40" href="a.htm" alt="">
<area shape="rect" coords="18,46,47,79" href="b.htm" alt="">
<area shape="rect" coords="61,43,93,78" href="c.htm" alt="">
<area shape="rect" coords="9,84,36,119" href="d.htm" alt="">
<area shape="rect" coords="48,85,77,116" href="e.htm" alt="">
<area shape="rect" coords="89,81,123,115" href="f.htm" alt="">
</map>
</body>
</html>
```

L'image et les fichiers sont disponibles dans l'espace de téléchargement.

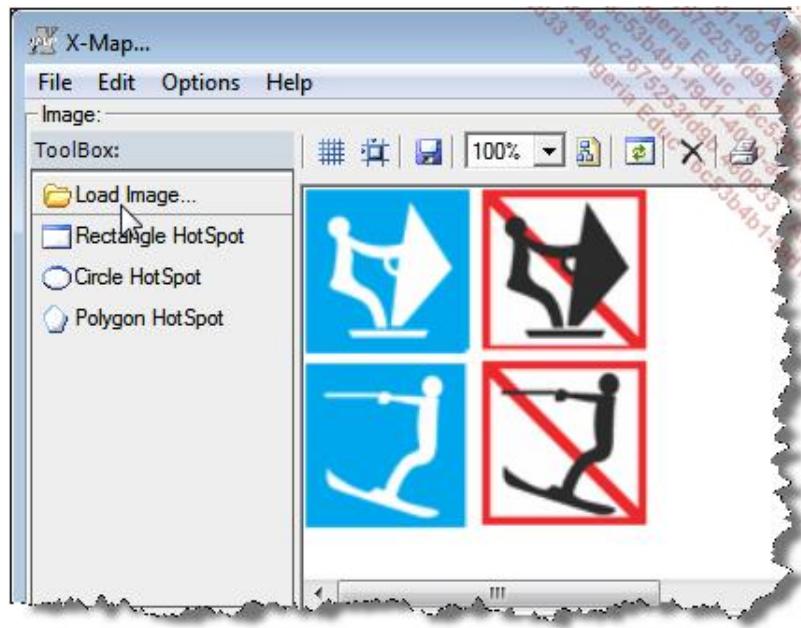
4. Les logiciels disponibles

Les éditeurs Html évolutifs, comme par exemple Dreamweaver, disposent de fonctions pour réaliser directement ces images réactives.

Pour ceux qui préfèrent encoder manuellement, il existe quelques applications qui peuvent aider à les réaliser. Citons MapThis, que vous pouvez télécharger à l'adresse www.lehtml.com/htmlplus/mapthis.htm. Ce logiciel est freeware, en anglais, efficace mais son look devient désuet. Retenons Xmap, un freeware également en anglais mais assez intuitif. Il peut être téléchargé à l'adresse www.carlosag.net/Tools/XMap/.

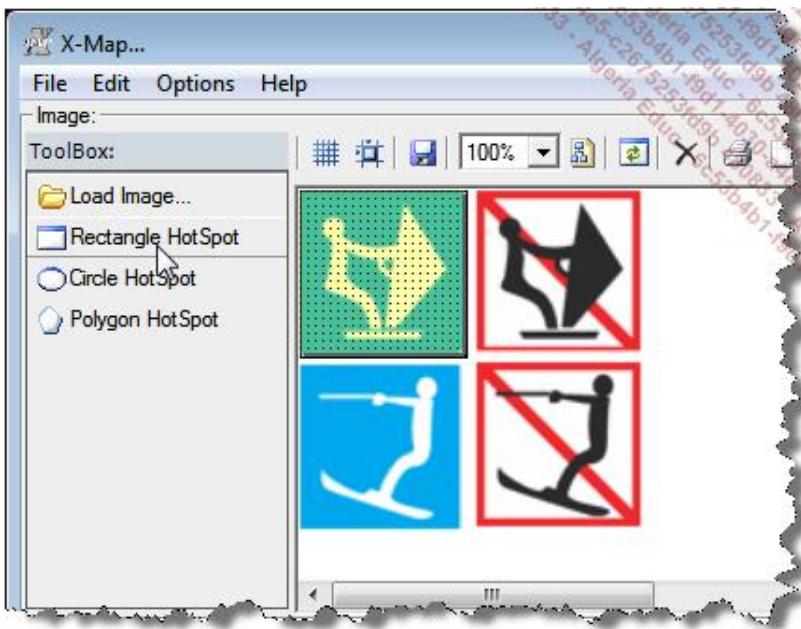
- Ouvrons l'application.

Commençons par charger l'image (**Load image**) grâce au menu à l'extrême gauche de la fenêtre. Nous avons retenu l'image map.png disponible dans l'espace de téléchargement.

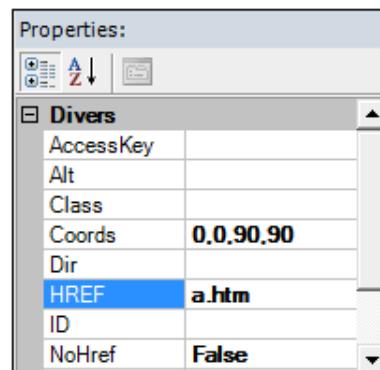


- Appliquons une zone réactive à chaque panneau.

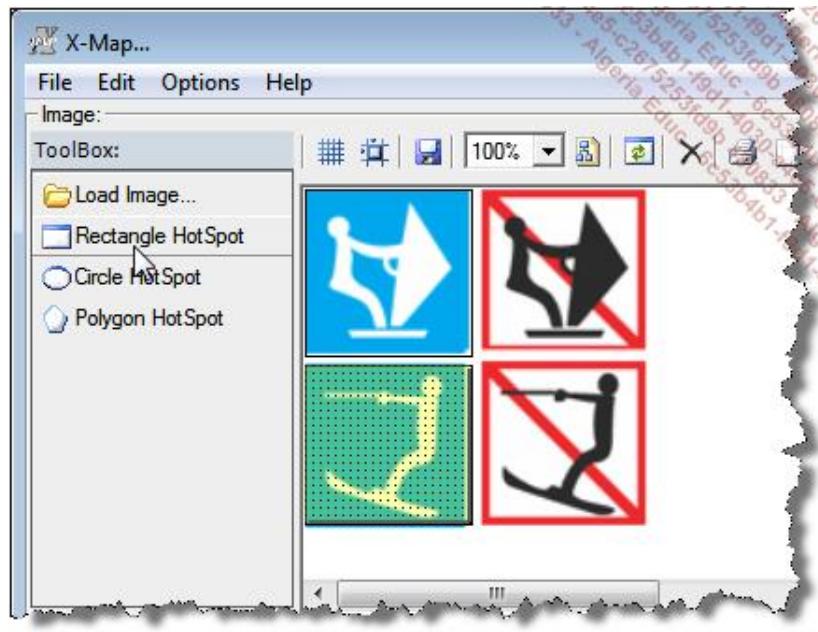
Dans le même menu, retenons l'option rectangle (**Rectangle HotSpot**) et positionnons le rectangle sur le panneau supérieur gauche.



Remarquons la boîte de propriétés (**Properties**) à l'extrême droite de votre écran. Elle nous permettra de déterminer le fichier cible de la zone en précisant son attribut HREF.



- Faisons de même pour un second panneau. Le lien attribué est b.htm.



- Et ensuite les panneaux suivants.

Le code de l'image réactive est généré au fur et à mesure des manipulations dans la sous-fenêtre au bas de l'application.

```
<html>
  <head>
    <title>Image Map Sample</title>
  </head>
  <body>
    <map name='map'>
      <area shape='rect' coords='0,0,90,90' href='a.htm' />
      <area shape='rect' coords='0,94,90,184' href='b.htm' />
      <area shape='rect' coords='94,0,180,90' href='c.htm' />
      <area shape='rect' coords='94,94,180,184' href='d.htm' />
    </map>
  </body>
</html>
```

Le code complet devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<p>

</p>
<map name='map'>
<area shape='rect' coords='0,0,90,90' href='a.htm'>
<area shape='rect' coords='0,94,90,184' href='b.htm'>
<area shape='rect' coords='94,0,180,90' href='c.htm'>
<area shape='rect' coords='94,94,180,184' href='d.htm'>
</map>
</body>
</html>
```


Préambule

Il est souvent difficile de percevoir le pourquoi du comment de ces balises dites sémantiques. Surtout que leur implémentation ne répond en rien à la définition académique de la sémantique comme relation entre le signifié et le signifiant.

Ces balises sémantiques décrivent la nature de certains éléments du contenu d'une page. Ces derniers pourront ainsi être plus facilement récupérés et réutilisés par des programmes automates, soit des moteurs de recherche.

Exemple

La nouvelle balise sémantique ou d'organisation de la page `<nav> ... </nav>` indique simplement que son contenu est un menu de navigation.

```
<nav>
<a href="#">Accueil</a> |
<a href=".htm">Item 1</a> |
<a href=".htm">Item 2</a> |
<a href=".htm">Item 3</a> |
<a href=".htm">Contact</a>
</nav>
```

Il est d'autant plus difficile à en comprendre l'utilité que ces balises Html n'ont aucune influence sur la structure du contenu et bien souvent aucune influence sur la présentation et l'affichage de la page.

Ce concept du Web sémantique est une idée maîtresse de Tim Berners-Lee (l'inventeur du Web) qui le défend avec conviction depuis de nombreuses années.

Les anciennes balises sémantiques

Le Html5 a hérité des anciennes balises sémantiques du Html 4.0. Ces balises possèdent une mise en forme qui leur est spécifique.

Ces balises sémantiques sont :

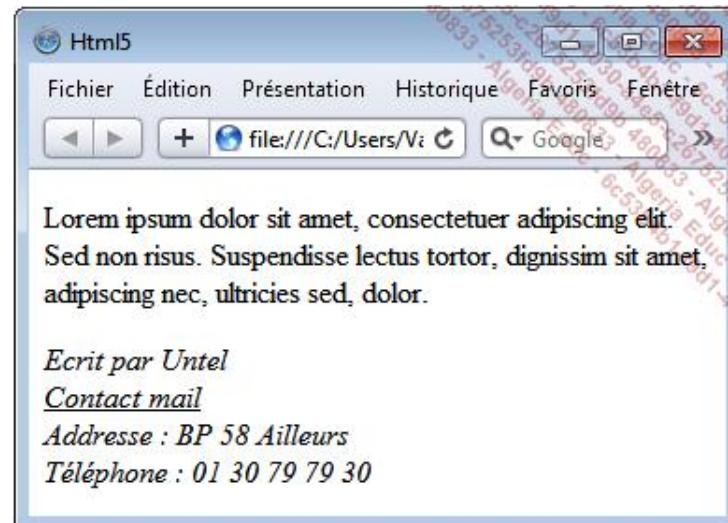
- `<abbr> ... </abbr>` qui signale une abréviation, par exemple SARL ou TVA. Outre sa fonction sémantique, cette balise signale aux synthèses vocales de ne pas tenter de lire le mot tel qu'il se présente mais de l'épeler.
- `<acronym> ... </acronym>` qui signale un acronyme, soit un mot formé d'initiales ou de syllabes de plusieurs mots comme ONU pour Organisation des Nations Unies, n'est plus reprise par le Html5. On utilisera la balise `<abbr>`.
- `<address> ... </address>` qui affiche une adresse de contact. Son contenu est affiché par défaut en italique et en petits caractères. Précisons qu'aucun lien vers cette adresse n'est effectué par cette balise.
- `<cite> ... </cite>` qui affiche une citation. Par défaut cette citation est affichée en italique.
- `<code> ... </code>` qui signale une syntaxe ou du code informatique. Par défaut, son contenu est affiché dans une police à espacement fixe.
- `<samp> ... </samp>` qui met en évidence un texte d'exemple. Par défaut, son contenu est affiché dans une police à espacement fixe.
- `<dfn> ... </dfn>` qui donne la définition d'un terme. Par défaut, cette définition est affichée en italique.
- `<kbd> ... </kbd>` qui indique à l'utilisateur les saisies au clavier à effectuer. Par défaut, son contenu est affiché dans une police à espacement fixe.
- `<var> ... <var>` qui contient une variable. Par défaut, le texte de la variable est représenté dans une police italique.
- ` ... ` définit un texte important. Par défaut, son contenu est affiché en gras.
- ` ... ` marque un texte sur lequel on veut insister. Par défaut, son contenu est affiché en italique.

Commentaire

On peut donner une autre allure que la présentation par défaut en utilisant les feuilles de style.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam. Maecenas ligula massa, varius a, semper congue, euismod non,
mi.</p>
<address>
Ecrit par Untel</b>
<a href="mailto:untel@example.org">Contact mail</a><br>
Adresse : BP 58 Ailleurs<br>
Téléphone : 01 30 79 79 30
</address>
</body>
```



Les nouvelles balises d'organisation

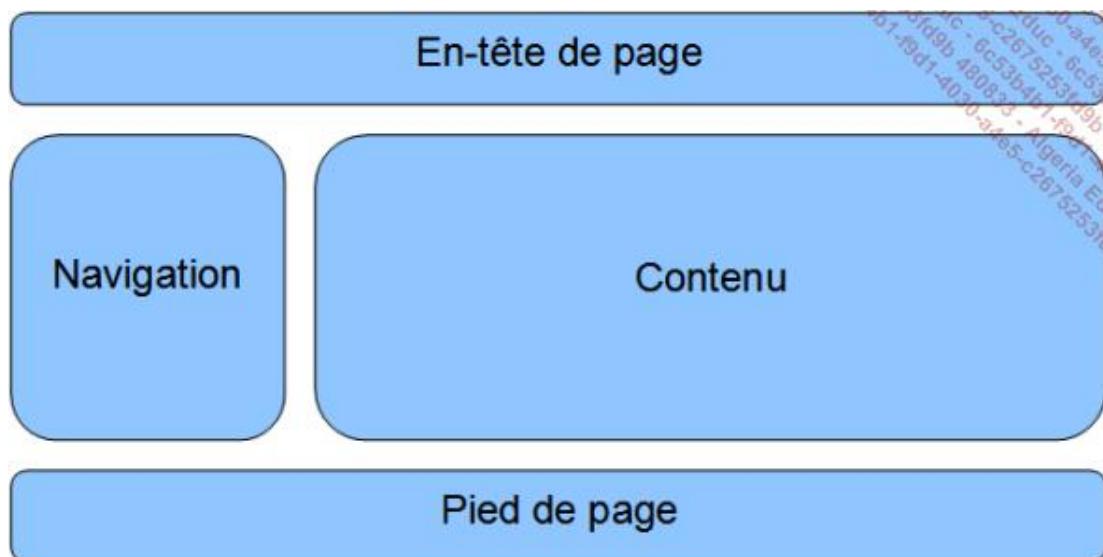
Ces balises d'organisation sont plutôt destinées au concepteur de la page afin qu'il puisse distinguer plus aisément des portions de code.

1. Les balises <header>, <nav>, <footer> et <aside>

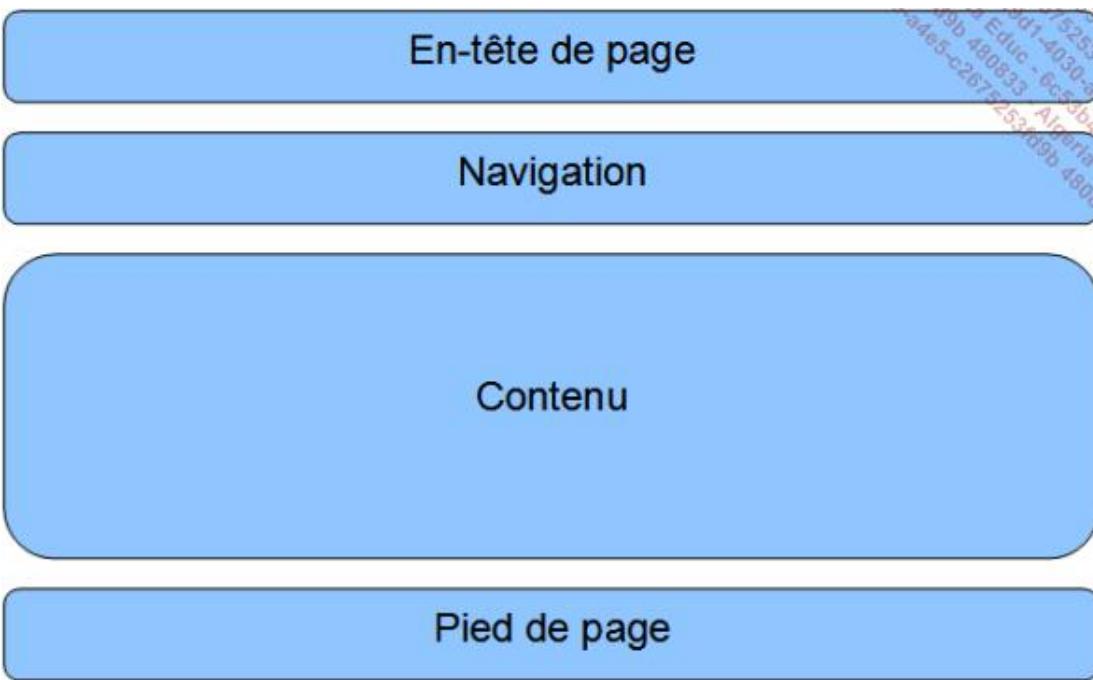
Si l'on effectue une synthèse des pages existantes sur la toile, on peut constater qu'elles comportent la plupart du temps, en totalité ou en partie, les éléments suivants :

- Un en-tête de page avec, par exemple, un logo, une bannière, le nom du site, un slogan ou un champ de recherche.
- Les outils de navigation, précieux et indispensables pour la consultation des différentes parties du site.
- Une partie consacrée au contenu.
- Une zone annexe qui permet d'apporter des éléments accessoires au contenu proprement dit, comme une publicité.
- Un pied de page avec par exemple la mention d'un copyright, le plan du site, les mentions légales, les règles d'accessibilité, etc.

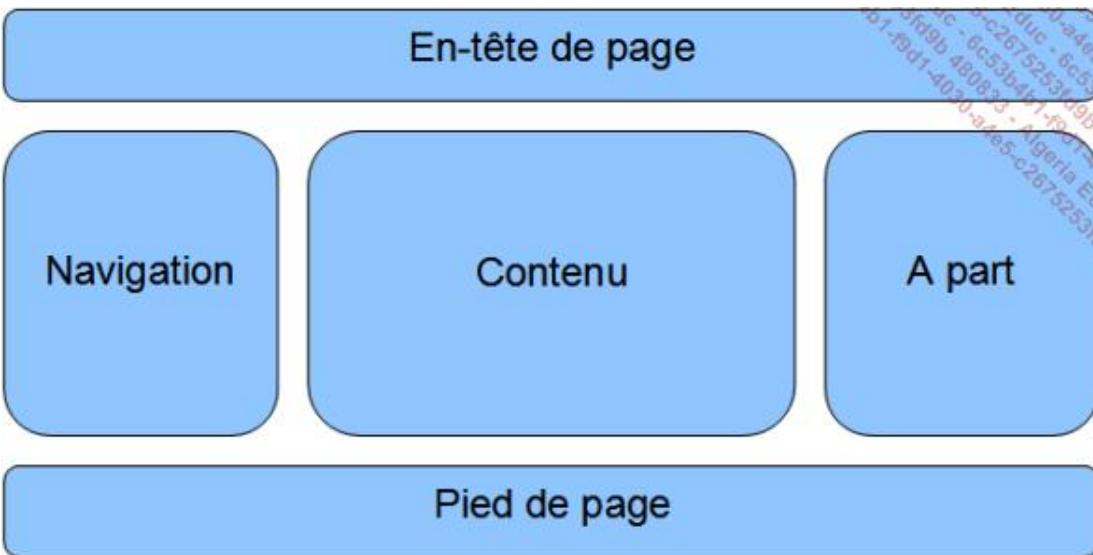
Ce qui visuellement peut prendre les formes suivantes :



Ou encore :



Ou encore :



L'idée du Html5 a été de créer de nouvelles balises pour permettre d'identifier plus clairement et plus rapidement ces grandes parties du design de la page. Ainsi :

- `<header> ... </header>` regroupe les éléments de l'en-tête de la page. Cette balise peut remplacer utilement les `<div id="header">` que l'on rencontre fréquemment.
- `<nav> ... </nav>` indique les éléments d'un menu de navigation.
- `<footer> ... </footer>` signale les éléments de pied de page. Cette balise peut remplacer utilement les `<div id="footer">` que l'on rencontre fréquemment.
- `<aside> ... </aside>` avertit qu'il s'agit d'éléments annexes au contenu.

Prenons un exemple de code :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
```

```

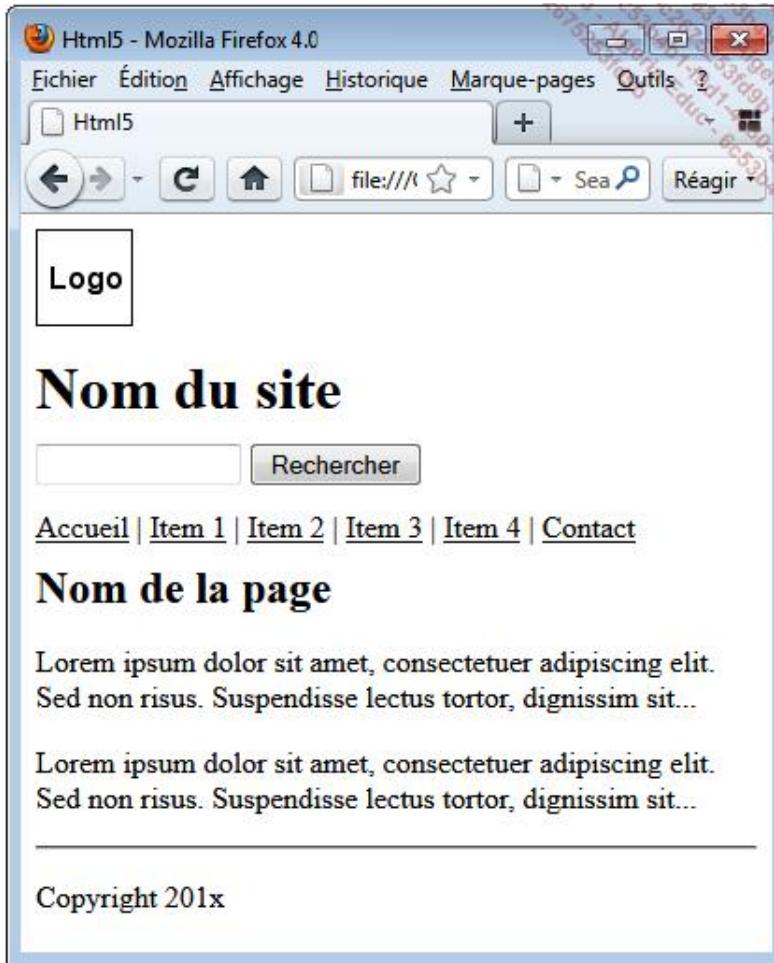
<meta charset="iso-8859-1">
</head>
<body>

<h1>Nom du site</h1>
<form action="http://www.google.com/search">
<input type="text" size="15" value="">
<input type="submit" value="Rechercher">
</form>
<div>
<a href="index.htm">Accueil</a> |
<a href="item1.htm">Item 1</a> |
<a href="item2.htm">Item 2</a> |
<a href="item3.htm">Item 3</a> |
<a href="item4.htm">Item 4</a> |
<a href="contact.htm">Contact</a>
</div>
<h2>Nom de la page</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam.</p>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam. </p>
<hr>
<p>Copyright 201x</p>
</body>
</html>

```

Soit une page assez commune avec :

- Un logo.
- Le nom du site.
- Un formulaire de recherche.
- Des liens de navigation dans le site.
- Le titre de la page.
- Du contenu.
- Une ligne horizontale.
- Une mention de copyright.



Ce code est correct, c'est même du Html5 tout à fait valide, mais rien n'est incorporé pour organiser la page et en distinguer des éléments majeurs comme le code relatif à l'en-tête, aux menus de navigation et du pied de page. Cette information est apportée par les nouvelles balises Html5 `<header>`, `<nav>` et `<footer>`.

Avec les nouvelles balises d'organisation du Html5, ce code pourrait prendre l'allure suivante :

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<header>

<h1>Nom du site</h1>
<form action="http://www.google.com/search">
<input type="text" size="15" value="">
<input type="submit" value="Rechercher">
</form>
</header>
<nav>
<a href="index.htm">Accueil</a> |
<a href="item1.htm">Item 1</a> |
<a href="item2.htm">Item 2</a> |
<a href="item3.htm">Item 3</a> |
<a href="item4.htm">Item 4</a> |
<a href="contact.htm">Contact</a>
</nav>
<h2>Nom de la page</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam.</p>

```

```

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam.</p>
<footer>
<hr>
<p>Copyright 201x</p>
</footer>
</body>
</html>

```

Le code apparaît ainsi plus lisible et l'on peut distinguer du premier coup d'œil les éléments de code relatifs à l'en-tête de la page, aux outils de navigation et au pied de page.

Voilà la seule et unique fonction de ces nouvelles balises Html5. Il ne faut pas leur chercher des fonctions de mise en page ou de présentation. Elles ne servent qu'à organiser la page et à rendre la lecture du code plus aisée principalement pour le concepteur.

La capture d'écran est par ailleurs totalement identique à la précédente.

C'est à l'auteur d'associer éventuellement des feuilles de style CSS à ces balises pour en assurer la présentation.

2. Les balises <section> et <article>

Le contenu principal peut à son tour être organisé en diverses parties :

- La balise <section> détermine une partie du contenu de la page se rapportant à un thème déterminé.
- La balise <article> définit un contenu indépendant du document qui possède une identité à part entière dans la page comme l'article d'un blog, un post dans un forum ou un produit dans un site commercial.

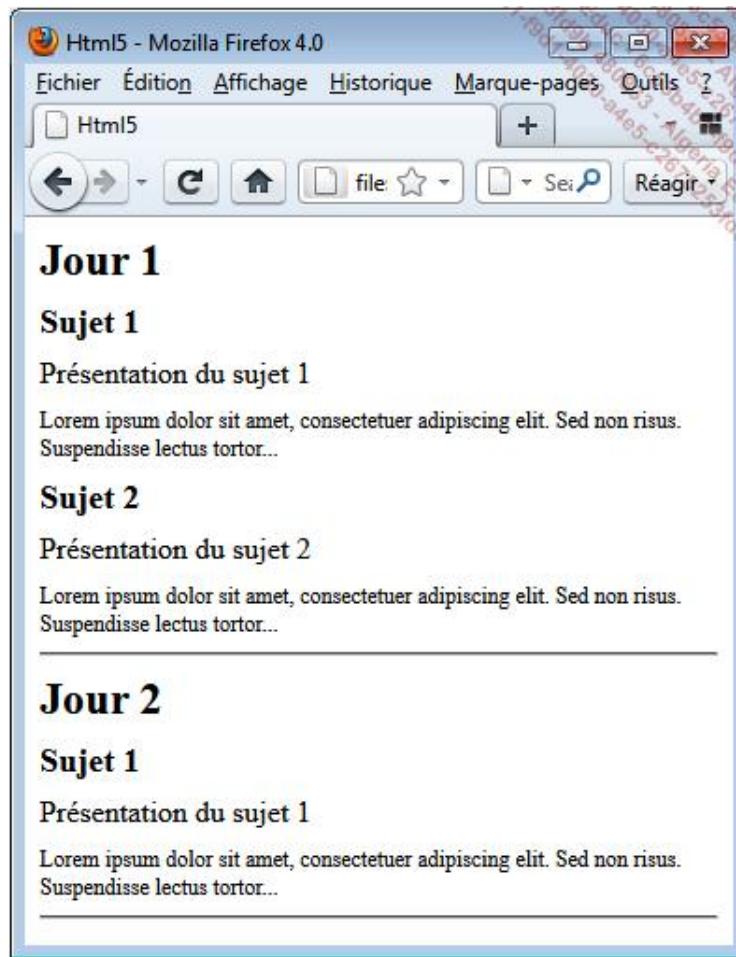
Prenons comme exemple un blog avec différents sujets abordés à chaque entrée journalière.

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<h1>Jour 1</h1>
<h2>Sujet 1</h2>
<div>Présentation du sujet 1</div>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam.</p>
<h2>Sujet 2</h2>
<div>Présentation du sujet 2</div>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam.</p>
<hr>
<h1>Jour 2</h1>
<h2>Sujet 1</h2>
<div>Présentation du sujet 1</div>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam.</p>
</body>
</html>

```

Ce qui s'affiche comme suit dans le navigateur :



Appliquons des balises `<section>` et `<article>`.

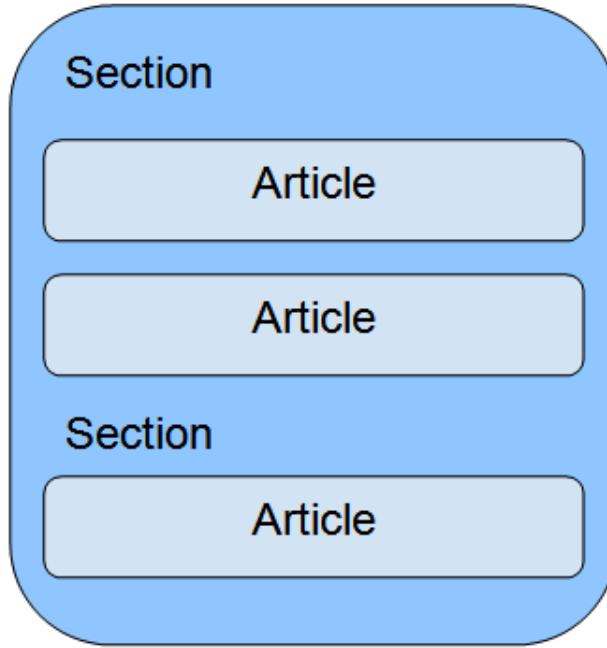
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<section>
<h1>Jour 1</h1>
<article>
<h2>Sujet 1</h2>
<div>Présentation du sujet 1</div>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam.</p>
</article>
<article>
<h2>Sujet 2</h2>
<div>Présentation du sujet 2</div>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam.</p>
</article>
<hr>
</section>
<section>
<h1>Jour 2</h1>
<article>
<h2>Sujet 1</h2>
```

```

<div>Présentation du sujet 1</div>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam.</p>
</article>
<hr>
</section>
</body>
</html>

```

Cette organisation du code peut se visualiser ainsi :



3. Un exemple général

Organisons maintenant une page complète avec les balises `<header>`, `<nav>`, `<footer>`, `<aside>`, `<section>` et `<article>`.

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<header>

<span>Nom du site</span>
</header>
<nav>
<ul>
<li><a href="">Accueil</a></li>
<li><a href="">Item 1</a></li>
<li><a href="">Item 2</a></li>
<li><a href="">Item 3</a></li>
<li><a href="">Contact</a></li>
</ul>
</nav>
<section>
<h1>Sujet de la page</h1>
<article>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed

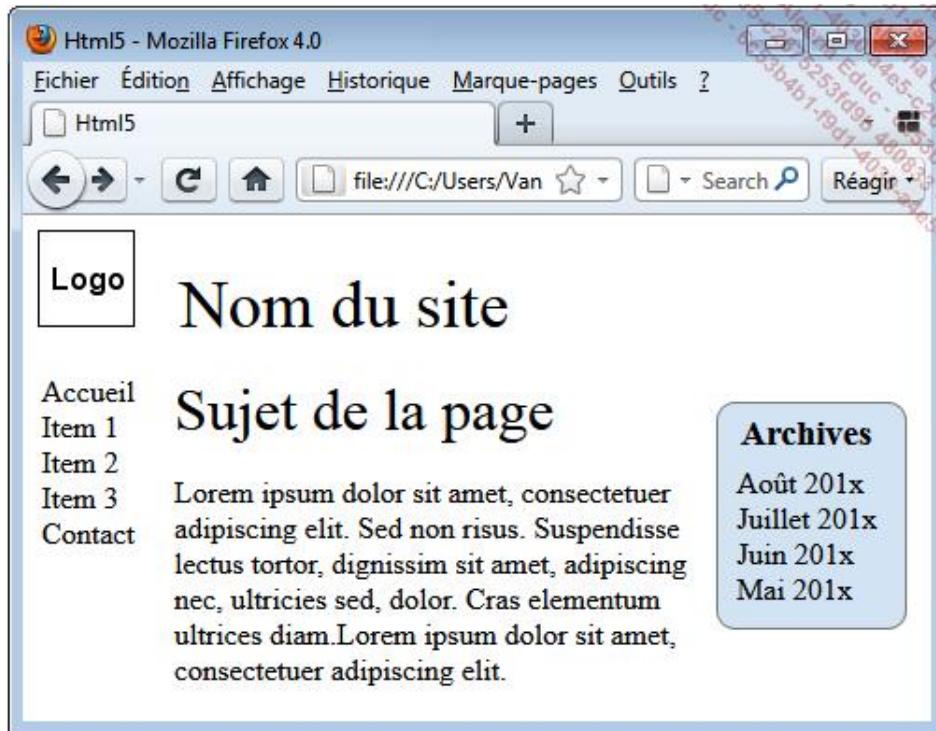
```

```

non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam.Lorem ipsum dolor sit amet, consectetuer adipiscing
elit...</p>
</article>
</section>
<aside>
<h3>Archives</h3>
<ul>
<li><a href="">Août 201x</a></li>
<li><a href="">Juillet 201x</a></li>
<li><a href="">Juin 201x</a></li>
<li><a href="">Mai 201x</a></li>
</ul>
</aside>
</body>
</html>

```

Le tout agrémenté de quelques feuilles de style peut donner, par exemple, le résultat suivant :



Là où cela se complique, c'est que rien n'interdit de mettre des `<nav>` dans des `<footer>`, des `<header>` dans des `<section>`, des `<header>` dans des `<article>`. Comme ces balises sont nouvelles et emblématiques du Html5, certains auteurs en usent et en abusent.

L'exemple suivant tente d'illustrer que l'usage intensif de ces balises d'organisation risque finalement de plus embrouiller le code que de l'organiser...

```

<section>
<header>
<h1>Les animaux de l'arctique</h1>
</header>
<article>
<header>
<h3>L'ours polaire</h3>
</header>
<p>L'ours blanc ou ours polaire (Ursus maritimus) est un grand
mammifère carnivore originaire des régions arctiques. C'est, avec
l'ours kodiak, le plus grand des carnivores terrestres et il
figure au sommet de sa pyramide alimentaire.</p>
<footer>
Source : Wikipédia
</footer>

```

```
</article>
<article>
<header>
<h3>Les phoques</h3>
</header>
<p>On rencontre communément six espèces de phoques dans les eaux canadiennes : le Phoque commun (Phoca vitulina), le Phoque annelé (P. hispida), le Phoque du Groenland (P. groenlandica), le Phoque barbu (Erignathus barbatus), le Phoque à capuchon (Cystophora cristata) et le Phoque gris (Halichoerus grypus).</p>
<footer>
Auteur : IAN MCTAGGART-COWAN
Source : L'encyclopédie canadienne
</footer>
</article>
<footer>
Compilation d'articles
</footer>
</section>
```

Pour que ces balises d'organisation atteignent leur objectif, elles doivent donc être utilisées avec modération.

De nouvelles balises sémantiques

Le HTML5 introduit également une foule de nouvelles balises sémantiques.

La balise <hgroup>

Cette balise `<hgroup>` est destinée à contenir un groupe de balises `<hx>` (au moins deux). Elle est une forme dérivée de la balise `<header>`.

```
<hgroup>
<h1>Titre de l'article</h1>
<h2>Sous-titre de l'article</h2>
</hgroup>
```

La balise <time>

La balise `<time>` définit une date et/ou une heure. L'attribut facultatif `datetime` indique la date et l'heure selon la représentation numérique internationale de la norme ISO8601.

```
<time>Demain</time> est un autre jour.
<time datetime="2012-10-05">Le 5 octobre</time>
<time datetime="2012-04-13T00:19:00+02:00">Samedi 13 mars à 19h</time>
```

La balise <mark>

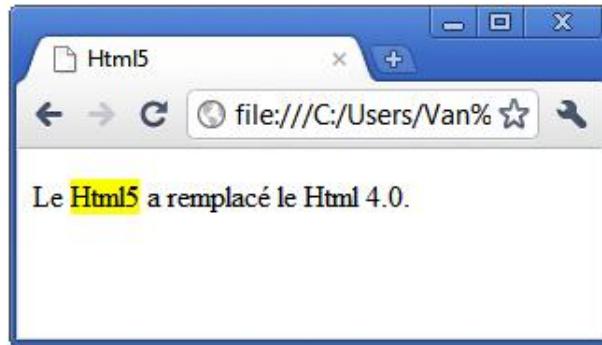
La balise `<mark>` met en évidence une partie du texte.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>HTML5</title>
<meta charset="iso-8859-1">
</head>
<body>
<mark>HTML5</mark> a remplacé le HTML 4.0.
</body>
</html>
```

Le contenu de cette balise devrait être souligné sur fond jaune, à l'identique des résultats de recherche dans les navigateurs.

À ce jour, seules les versions récentes de Google Chrome et de Firefox interprètent cette balise `<mark>`.



La balise <meter>

La balise `<meter>` définit une mesure. Utilisée uniquement pour les mesures avec un minimum connu et une valeur maximale.

Cette balise possède six attributs :

- value : la valeur de cette donnée sur l'échelle.

- min : le minimum possible.

- low : le minimum atteint.

- high : le maximum atteint.

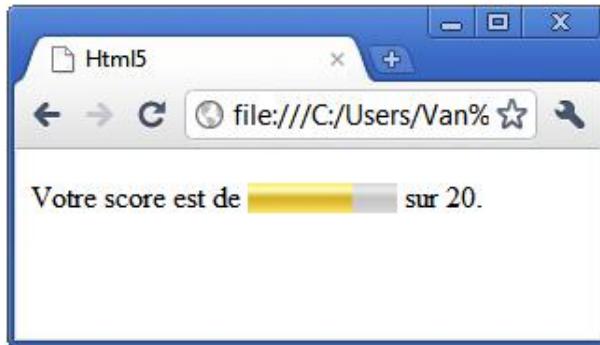
- max : le maximum possible.

- optimum : le maximum idéal.

À ce jour, seul Google Chrome interprète cette balise par une jauge du plus bel effet.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<p>Votre score est de
<meter value="14" min="0" max="20" low="8" high="18"
optimum="19.5"></meter> sur 20.
</body>
</html>
```



La balise <figure>

La balise `<figure>` peut être utilisée pour regrouper des éléments tels que des images, des vidéos ou même du texte qui viennent en illustration du contenu principal.

La balise `<figcaption>`, utilisée conjointement à la balise `<figure>`, fournit une légende aux éléments ainsi regroupés.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<h3>Les éoliennes</h3>
<p>Une éolienne est une machine utilisant la force motrice du vent. Cette force peut être utilisée mécaniquement (dans le cas d'une éolienne de pompage), ou pour produire de l'électricité (dans le cas d'un aérogénérateur) [Source Wikipédia].</p>
<figure>


<figcaption>Quelques éoliennes</figcaption>
</figure>
</body>
```



Les balises <details> et <summary>

La balise `<details>` définit des détails ou un contenu accessoire. La balise `<summary>` fournit un résumé de ces éléments accessoires.

Exemple

```
<details>
<summary>Plus d'informations...</summary>
<p>Voici les réactions de nos lecteurs concernant le
développement... </p>
</details>
```

Aucun navigateur ne reconnaît encore ces balises `<details>` et `<summary>`. Pourtant les perspectives sont intéressantes car selon les spécifications du W3C, au clic du texte spécifié par la balise `<summary>`, le contenu de la balise `<details>` devrait apparaître à la façon d'un menu déroulant. Et ceci de façon native, sans faire appel à du JavaScript.

Présentation

Les formulaires occupent une place prépondérante dans la conception et l'exploitation d'une application ou un site Web. C'est en effet la seule façon de recevoir en retour des informations provenant directement de l'utilisateur final, et en outre, structurées selon les besoins du concepteur. Il suffit de penser à tous les sites à vocation commerciale pour lesquels ces formulaires sont indispensables.

La préoccupation première des développeurs d'application Web est de récupérer des données valides. Citons par exemple un code postal français avec cinq caractères numériques ou une adresse email dont la syntaxe est valide. Une des applications premières du JavaScript a été (et est toujours) la validation en direct (côté client) des données encodées par l'utilisateur. Ainsi, tous les formulaires d'une certaine importance comportent, outre le code HTML, une bonne dose de JavaScript pour la validation des données. Une des idées majeures du HTML5 est de faire prendre en charge par le HTML (et donc par les navigateurs) la validation des champs de formulaires, libérant ainsi le code source de scripts redondants. Et ce, même dans le cas où la prise en charge du JavaScript aurait été désactivée.

Les formulaires font partie du langage HTML depuis une quinzaine d'années et n'ont pas été modifiés depuis. Il faut avouer que graphiquement, même avec des ajouts de propriétés de style, une lassitude certaine s'est installée autant chez les designers que les usagers. En outre avec le Web 2.0, l'utilisateur s'est habitué à une présentation plus proche des applications logicielles que des pages Web classiques. Citons les calendriers qui sont devenus usuels dans les sites de réservation en ligne ou les curseurs utilisés dans divers logiciels. En intégrant dans le langage HTML les spécifications Xforms, le HTML5 marque indéniablement les esprits en ajoutant aux formulaires hérités du HTML 4.0 un nombre impressionnant de formulaires inédits, se rapprochant de ce qui est utilisé dans les applications logicielles. L'intégration de ces nouveaux formulaires dans les navigateurs se fait hélas progressivement et il faudra peut-être encore attendre quelques années avant que tous les navigateurs usuels ne les gèrent complètement. Mais il est indéniable que le HTML5 ouvre en la matière des perspectives considérables aux concepteurs et designers.

Le but ultime des formulaires est le traitement automatisé des données récoltées. Il faut alors faire appel, côté serveur, à des langages de programmation et de gestion de bases de données comme par exemple PHP et MySQL. Un apprentissage qui dépasse largement le cadre de cet ouvrage se révélera indispensable. On entre là dans un domaine réservé aux amateurs très avertis, sinon aux professionnels. Heureusement, il existe pour les débutants le protocole mailto, qui permet l'envoi des données des formulaires à une adresse de courrier électronique. Si ce système présente l'avantage de pouvoir être utilisé par tout un chacun, sans faire appel à des ressources externes, il comporte néanmoins des limites certaines. En outre, le traitement ultérieur des données devra être effectué manuellement.

La déclaration de formulaire

La balise `<form> ... </form>` a comme unique fonction de déclarer au navigateur qu'il doit mettre en place un formulaire. Elle englobera les éléments ou champs de formulaires comme une ligne de texte, des cases à cocher, des listes déroulantes, etc.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
Eléments du formulaire
</form>
</body>
</html>
```

En soi, la balise `<form>` n'affiche rien dans la fenêtre du navigateur. Une capture d'écran n'a donc pas de sens.

Les attributs usuels de la balise `<form>` sont :

name

Pour attribuer un nom (*name*) au formulaire.

action

Lorsque vous donnez l'ordre au navigateur de transmettre les données du formulaire, il a besoin de connaître l'action qu'il doit effectuer.

Cette action sera :

- soit l'adresse d'un programme de traitement des données, situé sur le serveur, en CGI, Perl, PHP, ASP... Par exemple, `action = "http://www.serveur/traitement.php"`.
- soit une adresse de courrier électronique pour récupérer simplement les données. Le protocole mailto est alors utilisé. Par exemple, `action="mailto:mon_email@serveur"`.
- soit, lorsque les données d'un formulaire sont traitées en interne (côté-client) par du JavaScript, l'attribut `action` reste vide. Par exemple `action=""`.

Si vous utilisez le validateur du W3C (<http://validator.w3.org>), la présence de l'attribut `action` est obligatoire.

enctype

L'attribut `enctype` spécifie sous quel format informatique (*mime type*) seront transmises les données du formulaire. Par défaut, la valeur est `application/x-www-form-urlencoded`. Pour l'envoi de fichiers (voir la section "Les formulaires de transfert de fichiers" dans ce chapitre), celle-ci doit être `multipart/form-data` et enfin, pour l'envoi vers une adresse électronique avec le protocole mailto, la valeur sera `text/plain`.

method

La transmission des données d'un formulaire s'effectue selon la méthode GET ou la méthode POST. La méthode GET effectue le transfert en caractères ASCII et les données ne peuvent excéder 100 caractères. La méthode POST gère les caractères non-ASCII et une quantité de caractères illimitée. Dans la pratique, la méthode POST s'est imposée pour des raisons de facilité et d'efficacité.

Lorsque les données du formulaire sont traitées en interne (`action=""`), les attributs `method` et `enctype` sont inutiles car il n'est pas fait appel au serveur.

Commentaires

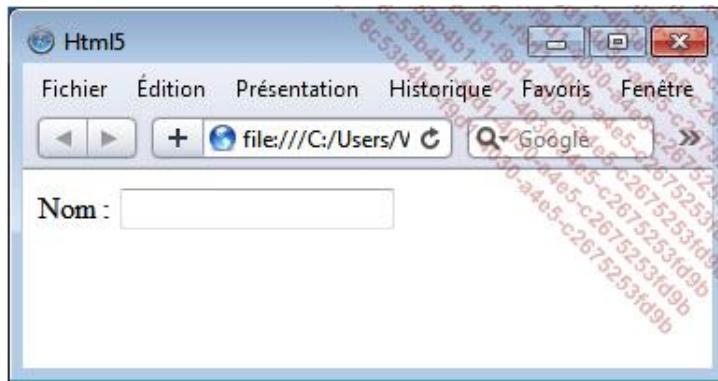
- Cette balise, basique en Html, est parfaitement compatible entre les différents navigateurs.
- La balise `<form>` doit obligatoirement être fermée. En cas d'oubli de la balise de fermeture `</form>`, aucun élément de votre formulaire ne sera affiché dans le navigateur.

La ligne de texte

Cette ligne de texte permet d'accueillir l'encodage de données tant alphabétiques que numériques.

Exemple

Proposons à l'utilisateur une ligne de texte pour encoder son nom.



```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
Nom : <input type="text">
</form>
</body>
</html>
```

Commentaires

- Cette balise, basique en Html, est parfaitement compatible entre les différents navigateurs.
- La balise `<input>` n'a pas de balise de fermeture. L'écriture `<input type="text" />`, inspirée du Xhtml, est également acceptée en Html5.

Les attributs possibles sont :

name

Définit un nom (*name*) unique pour cet élément. Cet attribut est utilisé pour collecter le contenu de la ligne de texte lors de la soumission du formulaire.

size

Définit le nombre de caractères visibles de la ligne de texte et, par conséquent, la largeur de la zone de texte. L'utilisateur peut néanmoins encoder autant de caractères qu'il souhaite même s'ils débordent du champ de la zone visible. La valeur par défaut de size est de 20.

maxlength

Détermine le nombre maximal de caractères que l'utilisateur peut encoder dans la ligne de texte. Cet attribut est particulièrement utile pour des données avec un nombre défini de caractères comme, par exemple, cinq chiffres pour le code postal français.

value

Définit la valeur par défaut de la ligne de texte. Celle-ci apparaît dans la ligne de texte au chargement de la page. Par exemple, `<input type="text" value="Votre nom ici !" />`.

readonly

Indique que la valeur attribuée par défaut à la ligne de texte ne pourra être modifiée par l'utilisateur.

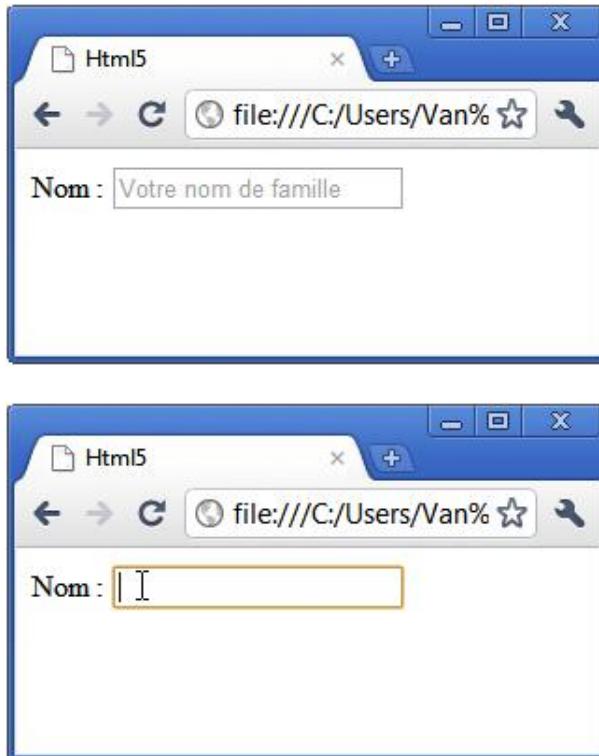
De nouveaux attributs sont apparus en Html5. Leur prise en charge par les navigateurs est à ce jour nulle ou en cours d'implémentation. En voici quelques-uns :

placeholder

Permet d'afficher une suggestion de saisie qui apparaît en grisé dans le champ de formulaire au chargement de la page. Ce texte disparaît dès que l'utilisateur donne le focus à l'élément concerné.

Cet attribut n'est pas encore repris par tous les navigateurs récents mais est déjà repris par Firefox 4+, Safari4+ et Chrome 3+. Cet attribut sera ignoré par les navigateurs qui ne le prennent pas encore en charge.

```
<input type="text" placeholder="votre nom de famille">
```



Certains pourraient avancer que l'attribut `value` effectue la même fonction. La différence est réelle mais subtile. Avec `value`, l'utilisateur doit effacer le texte avant d'effectuer son propre encodage. Avec `placeholder`, le texte est effacé automatiquement dès que le focus est donné à l'élément. En outre, avec l'attribut `value`, en cas de soumission du formulaire sans modification de cette valeur par défaut, c'est celle-ci qui sera transmise.

autofocus

Donne le focus à l'élément lors du chargement de la page. Cet attribut épargne quelques lignes de JavaScript qui réalisaient cette fonction.

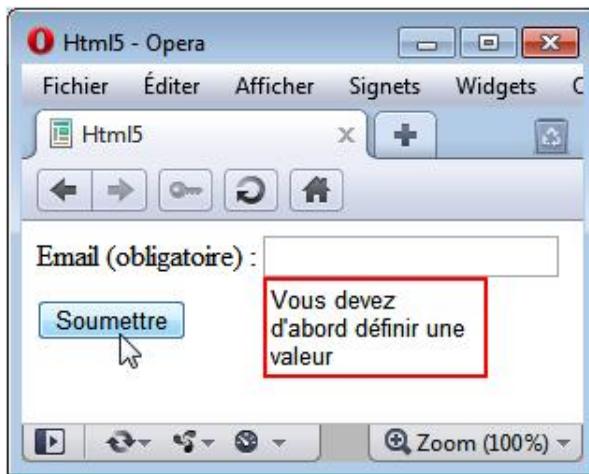
Cet attribut ne semble pour l'instant n'être repris que par Safari4+, Chrome 3+ et Opera 10+. Cet attribut sera ignoré par les navigateurs qui ne le prennent pas encore en charge.

required

Rend l'encodage de l'élément obligatoire par l'utilisateur pour la soumission et le traitement du formulaire. Très utile pour des éléments essentiels comme le nom ou l'adresse email de l'utilisateur.

Opera 10+ nous fournit une illustration de la réaction du navigateur lorsqu'un champ `required` serait laissé vide à la soumission du formulaire.

```
<form action="">
Email (obligatoire) :
<input type="text" name="mail" required><br>
<input type="submit" value="Soumettre">
</form>
```



Souhaitons cependant une infobulle plus élaborée sur le plan esthétique.

pattern

Définit l'expression régulière JavaScript qui est utilisée pour la validation de l'encodage. Par exemple, `pattern="[0-9]"` signifie que la valeur de l'élément doit être un nombre compris entre 0 et 9.

height

Détermine, en pixels ou en pourcentage, la hauteur de la ligne de texte. Peut être remplacé par la propriété CSS `height`.

width

Détermine, en pixels ou en pourcentage, la largeur de la ligne de texte. Peut être remplacé par la propriété CSS `width`.

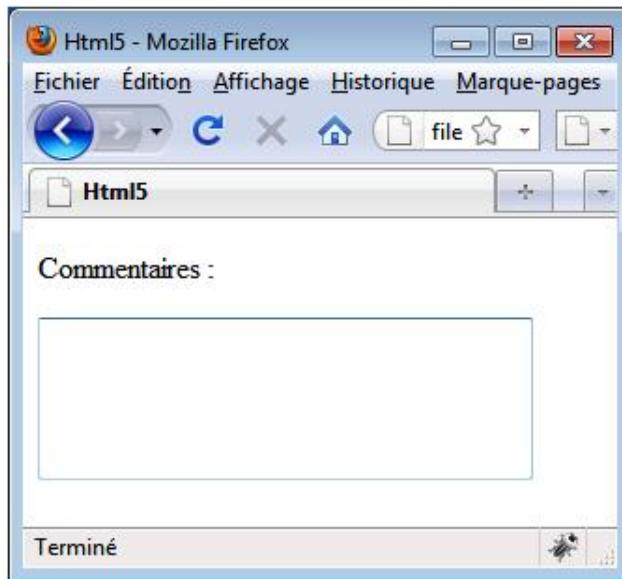
La zone de texte

Dans certaines situations, il faut prévoir plus d'espace à l'utilisateur pour s'exprimer. C'est le cas, par exemple, pour des commentaires, remarques ou suggestions. On utilise alors la balise `<textarea> ... </textarea>` qui introduit une zone de texte de plusieurs lignes.

Exemple

Une zone de texte dédiée aux commentaires de l'utilisateur :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
<p>Commentaires : </p>
<textarea rows="4" cols="30"></textarea>
</form>
</body>
</html>
```



Commentaires

- Cette balise, basique en Html, est parfaitement compatible entre les différents navigateurs.
- La balise `<textarea>` comporte une balise de fermeture `</textarea>`.
- Une valeur par défaut peut être prévue dans la zone de texte, non par l'attribut `value` mais de la façon suivante : `<textarea rows="4" cols="25">Vos suggestions ici! </textarea>`.

Les attributs possibles sont :

name

Définit un nom pour cet élément.

cols

Définit le nombre de caractères visibles en largeur de la zone de texte. Peut être remplacé par la propriété CSS `width`.

rows

Définit le nombre de lignes visibles en hauteur de la zone de texte. Peut être remplacé par la propriété CSS `height`.

readonly

Indique que la valeur attribuée par défaut à la ligne de texte ne pourra être modifiée par l'utilisateur.

De nouveaux attributs sont apparus en Html5. Leur prise en charge par les navigateurs est à ce jour nulle ou en cours d'implémentation. En voici quelques-uns :

autofocus

Donne le focus à l'élément lors du chargement de la page.

maxlength

Détermine le nombre maximal de caractères que l'utilisateur peut encoder dans la zone de texte. Permet de limiter les utilisateurs prolixes.

required

Rend l'encodage de l'élément obligatoire.

wrap

Spécifie la façon de gérer les retours automatiques à la ligne du texte lors de la soumission de la zone de texte. Avec `wrap="hard"`, un caractère de passage à la ligne est transmis avec le texte. Avec `wrap="soft"` (défaut), aucun caractère de changement de ligne n'est transmis.

Pour les puristes, cet attribut ne faisait plus partie du Html 4.0. Il a été réintroduit dans la spécification Html5, sous une forme modifiée cependant.

placeholder

Affiche un texte dans le champ de formulaire au chargement de la page. Ce texte s'efface automatiquement dès que l'utilisateur donne le focus à la zone de texte

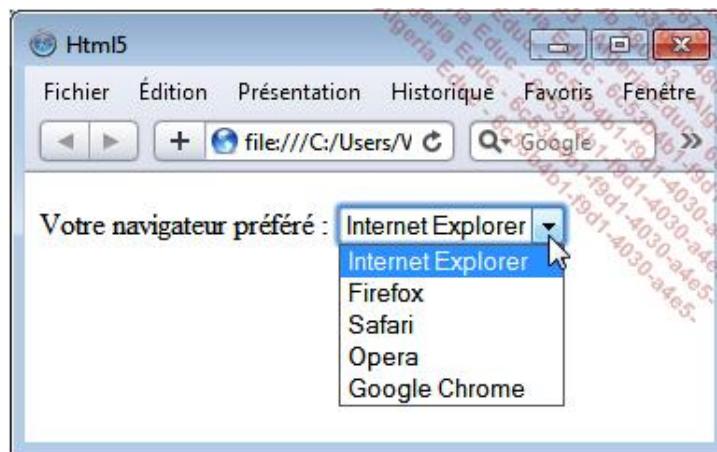
La liste déroulante

La liste déroulante propose différentes options à l'utilisateur.

Exemple

Demandons à l'utilisateur son navigateur préféré :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
<p>votre navigateur préféré :</p>
<select>
<option value="1">Internet Explorer</option>
<option value="2">Firefox</option>
<option value="3">Safari</option>
<option value="4">Opera</option>
<option value="5">Google Chrome</option>
</select>
</p>
</form>
</body>
</html>
```



Commentaires

- La balise `<select>... </select>` indique au navigateur l'usage d'une liste déroulante. Les éléments de la liste sont introduits par les balises `<option> ... </option>`.
- Cette balise est parfaitement compatible entre les différents navigateurs.

Les attributs possibles sont :

name

Définit un nom pour la liste déroulante en vue d'un traitement ultérieur.

size

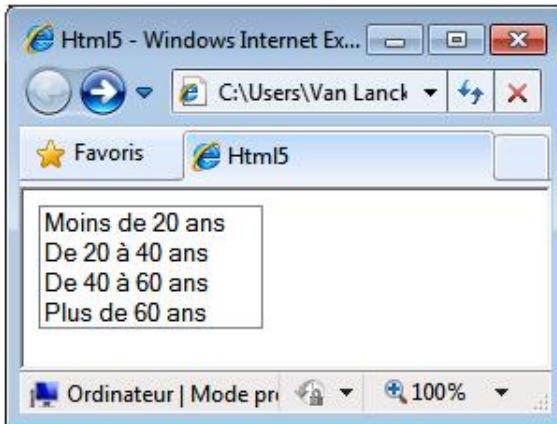
Par défaut, l'attribut `size` de la balise `<select>` est égal à `1`, ce qui est assez pratique car cela permet un gain de place appréciable dans la disposition de la page. Cependant par l'attribut `size="x"`, vous pouvez définir le nombre d'éléments du menu qui sera visible. Votre menu n'aura alors plus rien de déroulant !

```
<body>
```

```

<form action="">
<select size="4">
<option>Moins de 20 ans</option>
<option>De 20 à 40 ans</option>
<option>De 40 à 60 ans</option>
<option>Plus de 60 ans</option>
</select>
</form>
</body>

```



multiple

Par défaut, l'utilisateur ne peut choisir qu'un élément du menu déroulant. Avec l'attribut multiple de la balise `<select>`, plusieurs choix peuvent être effectués. Pour ce faire, l'utilisateur doit maintenir la touche [Ctrl] du clavier et cliquer sur les éléments avec la souris. Il est alors préférable de rappeler dans la page cette façon de procéder peu commune pour l'internaute moyen.

L'attribut `size` doit être spécifié et égal au nombre des balises `<option>`.

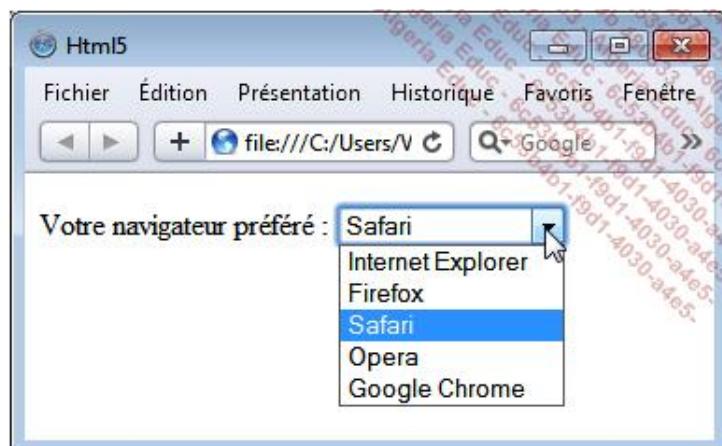
selected

Par défaut, le premier élément de la liste est retenu. Mais il est possible de présélectionner un autre élément par l'attribut `selected` de la balise `<option>`.

```

<body>
<form action="">
<p>Votre navigateur préféré :</p>
<select>
<option value="IE">Internet Explorer</option>
<option value="FF">Firefox</option>
<option value="S" selected>Safari</option>
<option value="O">Opera</option>
<option value="GC">Google Chrome</option>
</select>
</p>
</form>
</body>

```



value

En principe, c'est le texte de l'élément choisi placé derrière `<option>` qui est transmis lors de l'envoi du formulaire. Vous pouvez toutefois spécifier qu'une autre valeur (généralement numérique) soit transmise avec l'attribut `value="valeur"`.

Mis à part `autofocus`, il n'y a pas de nouveaux attributs significatifs en Html5.

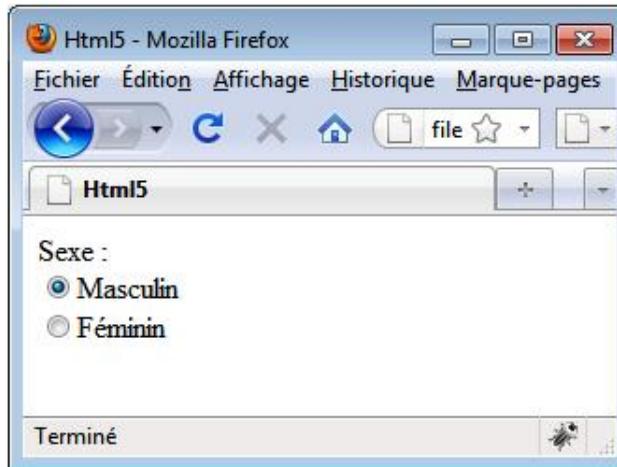
Les boutons de choix unique (radio)

Les boutons de choix unique, aussi appelés boutons radio, ont comme particularité qu'une seule option à la fois peut être activée (le "ou" exclusif).

Exemple

Informons-nous du sexe de l'interlocuteur :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
Sexe :<br>
<input type="radio" name="sexe">Masculin
<input type="radio" name="sexe">Féminin
</form>
</body>
</html>
```



Commentaires

- Cette balise est parfaitement compatible entre les différents navigateurs.
- La balise `<input>` n'a pas de balise de fermeture. L'écriture `<input type = "radio"/>`, inspirée du Xhtml, est également acceptée en Html5.

Les attributs possibles sont :

name

Ici l'attribut `name` est obligatoire. En outre, dans le cas de boutons radio, le nom doit être identique pour tous les boutons.

checked

Permet de présélectionner un bouton radio.

value

En vue d'un traitement ultérieur, on attribue une valeur à chaque bouton radio par l'attribut `value="valeur"`.

Mis à part `autofocus`, il n'y a pas de nouveaux attributs significatifs en Html5.

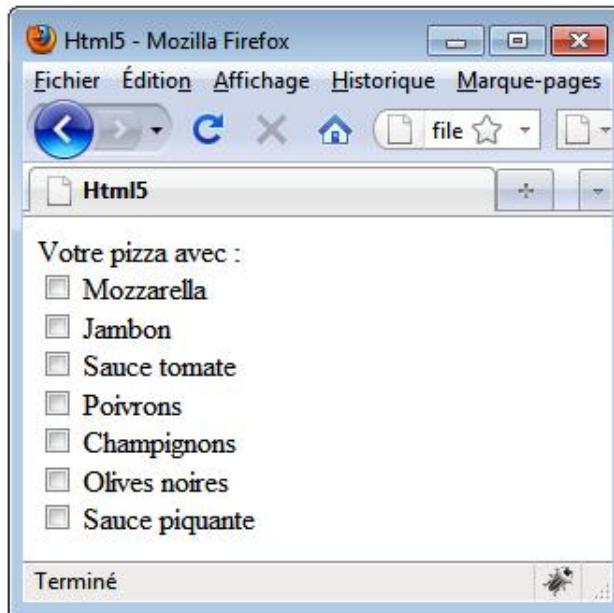
Les boutons de choix multiples (checkbox)

La mise en œuvre de ces boutons de choix multiples, aussi appelés boutons checkbox, est proche des boutons à choix unique mais, dans le cas présent, plusieurs choix simultanés peuvent être réalisés.

Exemple

Demandons de préciser les garnitures d'une pizza.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
Votre pizza avec :<br>
<input type="checkbox" name="n1"> Mozzarella<br>
<input type="checkbox" name="n2"> Jambon<br>
<input type="checkbox" name="n3"> Sauce tomate<br>
<input type="checkbox" name="n4"> Poivrons<br>
<input type="checkbox" name="n5"> Champignons<br>
<input type="checkbox" name="n6"> Olives noires<br>
<input type="checkbox" name="n7"> Sauce piquante
</form>
</body>
</html>
```



Commentaires

- Cette balise est parfaitement compatible entre les différents navigateurs.
- La balise `<input>` n'a pas de balise de fermeture. L'écriture `<input type="checkbox" />`, inspirée du Xhtml, est également acceptée en Html5.

Les attributs possibles sont :

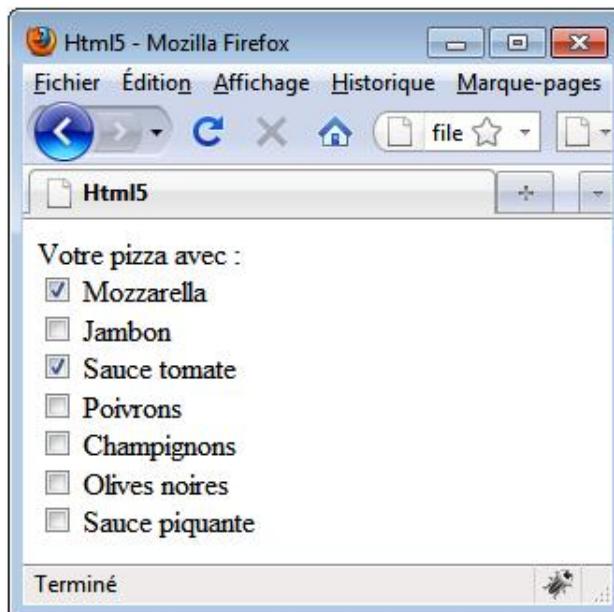
name

L'attribut `name="nom"` est obligatoire. Les règles sont moins précises que pour les boutons d'option. Vous pouvez utiliser des noms identiques ou des noms différents pour chaque case à cocher. Cependant des noms différents sont indispensables pour leur utilisation dans un script.

checked

L'attribut checked permet de présélectionner une case à cocher.

```
<form action="">
Votre pizza avec :<br>
<input type="checkbox" name="n1" checked > Mozzarella<br>
<input type="checkbox" name="n2"> Jambon<br>
<input type="checkbox" name="n3" checked > Sauce tomate<br>
<input type="checkbox" name="n4"> Poivrons<br>
<input type="checkbox" name="n5"> Champignons<br>
<input type="checkbox" name="n6"> Olives noires<br>
<input type="checkbox" name="n7"> Sauce piquante
</form>
```



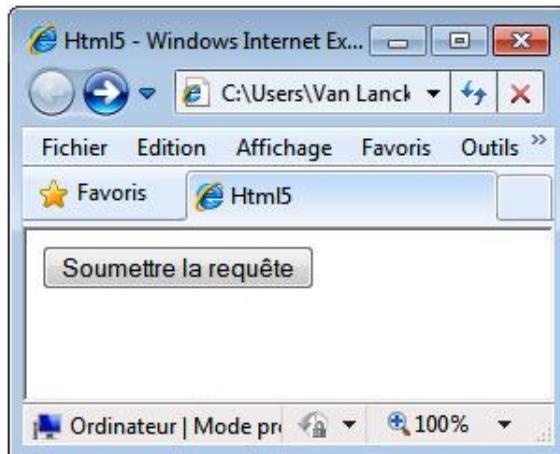
value

En vue d'un traitement ultérieur, on attribue bien entendu une valeur à chaque bouton checkbox par l'attribut value="valeur".

Mis à part autofocus, il n'y a pas de nouveaux attributs significatifs en Html5.

Le bouton d'envoi

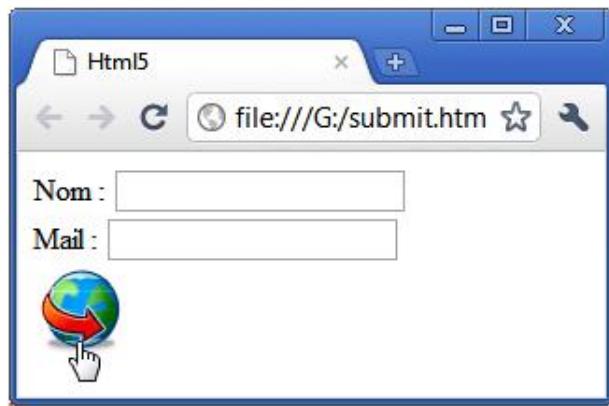
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
<input type="submit">
</form>
</body>
</html>
```



Commentaires

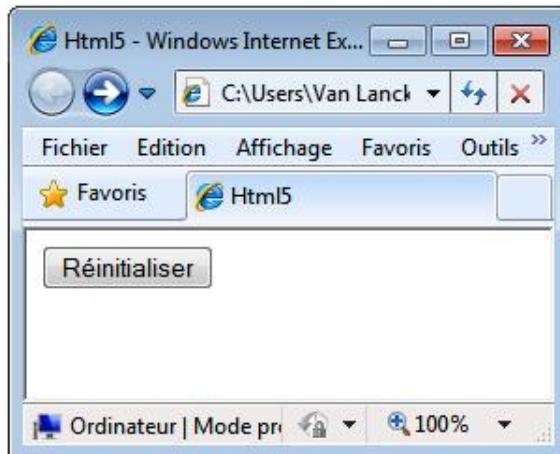
- Cette balise est parfaitement compatible entre les différents navigateurs.
- La balise `<input>` n'a pas de balise de fermeture. L'écriture `<input type="submit" />`, inspirée du Xhtml, est également acceptée en Html5.
- Le texte du bouton est choisi par le navigateur. Force est de constater que celui-ci varie d'un navigateur à l'autre : **Soumettre la requête** pour Internet Explorer, **Envoyer** pour Firefox et Opera, **Soumettre** pour Safari et **Valider** pour Google Chrome. Il est possible de modifier le texte par défaut du bouton par l'attribut `value`. La balise devient alors, par exemple, `<input type="submit" value="Envoyer le formulaire">`.
- Il est possible de remplacer le bouton de soumission par une image grâce à la balise `<input type="image">`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
Nom : <input type="text"><br>
Mail : <input type="text"><br>
<input type="image" src="submit.gif" alt="Soumettre" width="50"
height="50">
</form>
</body>
</html>
```



Le bouton d'annulation

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
<input type="reset">
</form>
</body>
</html>
```



Commentaires

- Cette balise est parfaitement compatible entre les différents navigateurs.
- La balise `<input>` n'a pas de balise de fermeture. L'écriture `<input type="reset" />`, inspirée du Xhtml, est également acceptée en Html5.
- L'intitulé du bouton est choisi par le navigateur. La plupart de ceux-ci ont adopté **Réinitialiser** sauf Firefox qui préfère **Effacer**. Il est possible de modifier le texte par défaut du bouton par l'attribut value. La balise devient alors, par exemple, `<input type="reset" value="Recommencer">`.

Le bouton de commande

En matière de boutons (*button*), il y a aussi la balise `<button> ... </button>`. Elle permet de déclencher, au clic de celui-ci, une action spécifique définie par le concepteur du site, généralement à l'aide de JavaScript. Elle offre également la possibilité de réaliser la soumission et l'annulation du formulaire (`submit` et `reset`).

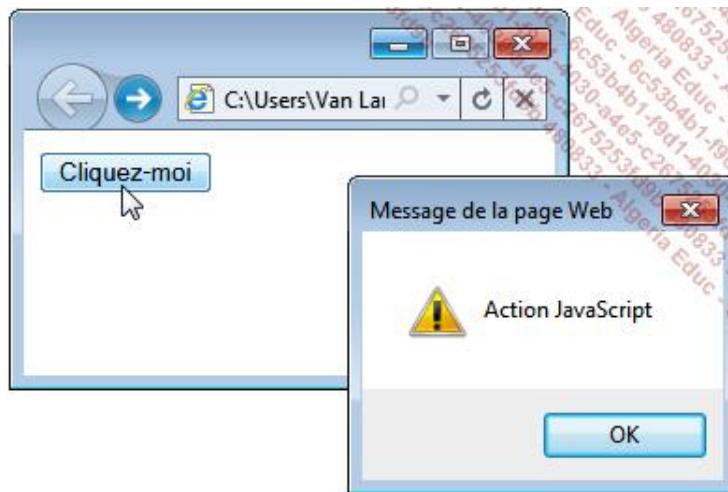
En outre, cette balise `<button>` a l'avantage de posséder une balise d'ouverture et de fermeture. Il est ainsi possible d'y inclure du texte, des images ou du contenu Html. Notons également que cette balise `<button>` n'a pas besoin d'être imbriquée dans un formulaire `<form>` et peut ainsi être utilisée dans de multiples contextes.

Tout ceci en fait une balise polyvalente et justifie son succès auprès des développeurs.

Exemple

Au clic d'un bouton, une fenêtre d'alerte s'affiche :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<button onclick="alert('Action JavaScript')">
Cliquez-moi</button>
</body>
</html>
```



Commentaires

- Cette balise est parfaitement compatible entre les différents navigateurs.
- La balise `<button>` a bien une balise de fermeture, soit `</button>`.
- Le texte du bouton prend place entre les balises et non par l'attribut `value`. Par exemple, `<button>Cliquez-moi</button>`.

Les attributs possibles sont :

name

Spécifie un nom pour le bouton.

type

L'attribut `type` admet trois arguments :

- `button` : utilisé lorsque le bouton déclenche un script (généralement repris comme défaut par les navigateurs).
- `submit` : pour soumettre le formulaire.
- `reset` : pour réinitialiser le formulaire.

Les sites et les forums spécialisés recommandent de toujours spécifier l'attribut `type` car l'argument `button` n'est pas repris comme valeur par défaut par tous les navigateurs (par exemple, Internet Explorer 8 en mode Xhtml Strict).

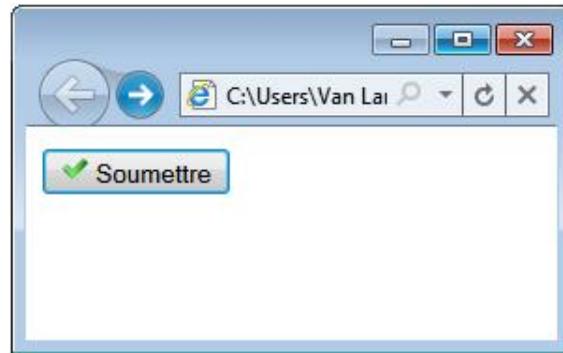
disabled

Permet de désactiver le bouton au chargement de la page. Celui-ci apparaît alors en grisé. Un script permettra de l'activer (`enabled`) après une action de l'utilisateur, par exemple, après avoir accepté des conditions d'utilisation.

Mis à part `autofocus`, il n'y a pas de nouveaux attributs significatifs en Html5.

Illustrons la possibilité d'ajouter une image à la balise `<button>` par l'ajout d'une petite icône (ok.png) disponible dans l'espace de téléchargement réservé à cet ouvrage.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<button>
Soumettre
</button>
</body>
</html>
```



Les formulaires cachés

Les formulaires permettent de stocker des données qui ne seront pas visibles par le visiteur de la page. Ce type de champ est parfois utilisé par les programmeurs confirmés (spécialement en JavaScript) car il peut contenir des données d'un formulaire précédent, des données brutes ou des données issues d'un autre script.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<input type="hidden" name="c" value="3.1416">
</body>
</html>
```

Une capture d'écran n'a pas de sens puisque le champ de formulaires est caché (`hidden`).

Commentaires

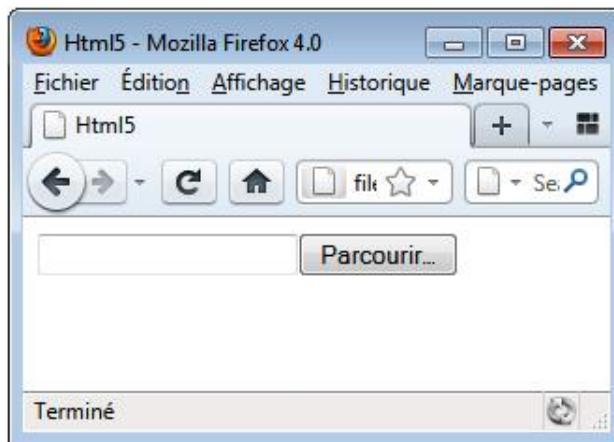
- Cette balise est parfaitement compatible entre les différents navigateurs.
- La balise `<input>` n'a pas de balise de fermeture. L'écriture `<input type="hidden" />`, inspirée du Xhtml, est également acceptée en Html5.

Les formulaires de transfert de fichiers

La balise `<input type="file">` prend en charge le transfert de fichiers (*file*) du poste de l'utilisateur vers un ordinateur de type serveur.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="" method="post" enctype="multipart/form-data">
<input type="file">
</form>
</body>
</html>
```



Le clic du bouton **Parcourir** (intitulé variable selon les navigateurs) ouvre l'arborescence de votre ordinateur pour sélectionner le fichier à transférer.

Commentaires

- Cette balise est parfaitement compatible entre les différents navigateurs.
- La balise `<input>` n'a pas de balise de fermeture. L'écriture `<input type="file" />`, inspirée du Xhtml, est également acceptée en Html5.
- Les attributs habituels de la balise `<input>` peuvent être utilisés. Signalons cependant l'attribut `maxlength` qui permet de fixer la taille maximale de fichier à transférer.
- Dans la déclaration de la balise `<form>`, il faut impérativement utiliser les attributs `method="post"` et `enctype="multipart/form-data"` pour le transfert au bon format du fichier.
- Il y a aussi l'attribut `accept` qui permet de limiter le transfert à certains types de fichiers. On peut citer des fichiers texte (txt), Word (doc), Excel (xls), pdf, des fichiers image (jpeg, gif ou png), etc. Dans la désignation des fichiers, le joker * est accepté.
- Il faut noter que cette balise Html ne sert qu'à sélectionner le fichier à transférer. Le transfert lui-même doit être pris en charge par des applications côté-serveur comme par exemple PHP.

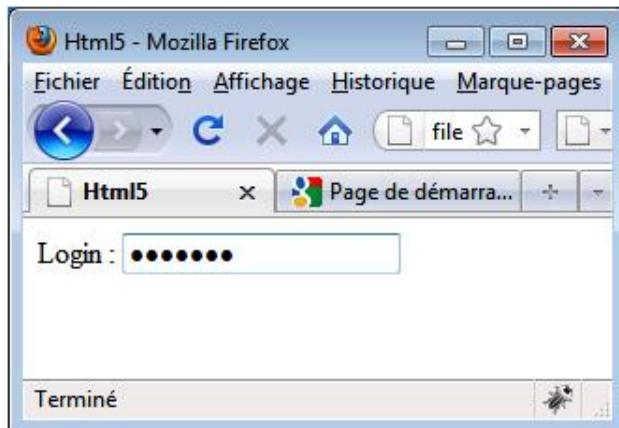
Les formulaires de mot de passe

Ce type de formulaire est en fait une simple ligne de texte mais dont l'encodage est remplacé, à l'affichage, par des puces ou des astérisques.

Ce formulaire de mot de passe ne protège en aucun cas les données car elles seront transmises en clair. Elles vous protègent uniquement contre les personnes qui pourraient vous lire durant l'encodage.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
Login :
<input type="password">
</form>
</body>
</html>
```



Commentaires

- Cette balise est parfaitement compatible entre les différents navigateurs.
- La balise `<input>` n'a pas de balise de fermeture. L'écriture `<input type="password" />`, inspirée du Xhtml, est également acceptée en Html5.
- Les attributs usuels de la ligne de texte `name`, `size`, `maxlength`, etc. peuvent être utilisés.

Mis à part `autofocus`, il n'y a pas de nouveaux attributs significatifs en Html5.

L'organisation des champs de formulaires

Dans le cas de formulaires longs et complexes, il est parfois utile de regrouper graphiquement certains éléments pour organiser la page de façon logique. Les balises `<fieldset>` et `<legend>` permettent d'améliorer sensiblement l'ergonomie et l'usabilité des formulaires.

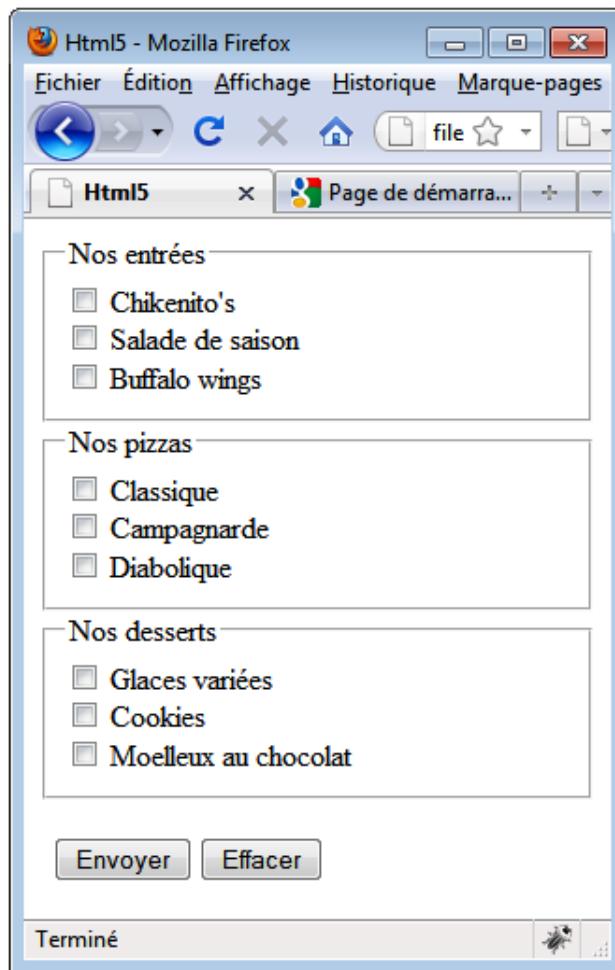
La balise `<fieldset> ... </fieldset>` englobe les champs de formulaires que vous déterminez. Ces champs sont alors visualisés à l'écran par une bordure.

La balise `<legend> ... </legend>`, qui se place directement derrière la balise `<fieldset>`, ajoute une légende qui vient s'insérer dans la bordure dessinée par le `<fieldset>` (voir capture d'écran suivante).

Exemple

Regroupons les entrées, les pizzas et les desserts dans une liste à choix multiples :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
<fieldset>
<legend>Nos entrées</legend>
<input type="checkbox" name="n1"> Chikenito's<br>
<input type="checkbox" name="n2"> Salade de saison<br>
<input type="checkbox" name="n3"> Buffalo wings<br>
</fieldset>
<fieldset>
<legend>Nos pizzas</legend>
<input type="checkbox" name="n4"> Classique<br>
<input type="checkbox" name="n5"> Campagnarde<br>
<input type="checkbox" name="n6"> Diabolique<br>
</fieldset>
<fieldset>
<legend>Nos desserts</legend>
<input type="checkbox" name="n7"> Glaces variées<br>
<input type="checkbox" name="n8"> Cookies<br>
<input type="checkbox" name="n9"> Moelleux au chocolat<br>
</fieldset>
<input type="submit"> <input type="reset">
</form>
</body>
</html>
```



Commentaires

- Cette balise est parfaitement compatible entre les différents navigateurs.
- Il ne faut pas oublier les balises de fermeture.
- Il n'y a plus d'alignement gauche ou droit de la légende en Html5. Cette opération doit se faire par une feuille de style CSS.

L'étiquetage des champs de formulaires

La balise `<label>` associe explicitement l'intitulé à un champ de formulaire particulier. Un peu comme si on collait une étiquette (`label`) en face d'un élément de formulaire.

Dans un premier temps, la balise `<label>` améliore grandement l'ergonomie des formulaires en permettant d'activer un élément de celui-ci, par exemple un bouton radio, en cliquant sur le bouton radio lui-même ou sur l'intitulé de celui-ci.

Mais les balises `<label>` sont aussi particulièrement utiles dans le domaine de l'accessibilité des sites Web aux personnes non voyantes. Ces "étiquettes" sont prises en charge par les aides techniques (barrettes braille ou synthèses vocales) et facilitent grandement l'utilisation des formulaires par les personnes visuellement déficientes.

Dans un premier temps, le texte assigné à un élément de formulaire doit être placé entre les balises `<label> ... </label>`.

```
<label>Nom</label> :  
<input type="text"><br>
```

Il faut ensuite relier cette étiquette `label` au contrôle de formulaire. Pour ce faire, l'élément de formulaire sera défini par un identifiant de type `id`.

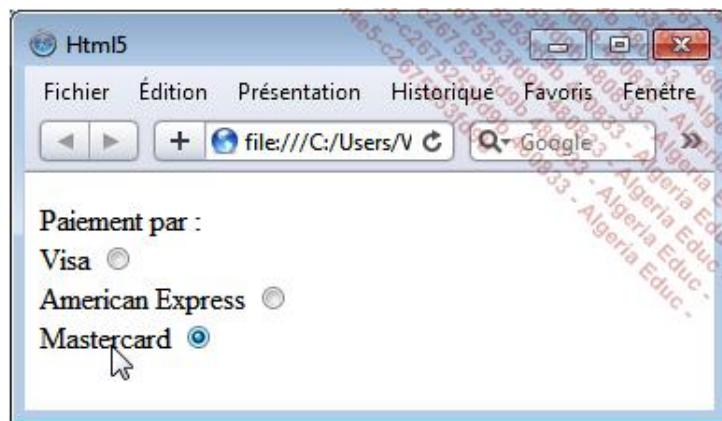
```
<label>Nom</label> :  
<input type="text" id="f1"><br>
```

L'attribut `for="..."` ajouté à la balise `<label>` permet de raccorder directement l'étiquette au champ de formulaire en faisant référence à cet identifiant.

```
<label for="f1">Nom</label> :  
<input type="text" id="f1"><br>
```

Exemple

```
<!DOCTYPE html>  
<html lang="fr">  
<head>  
<title>Html5</title>  
<meta charset="iso-8859-1">  
</head>  
<body>  
<form action="">  
<p>Paiement par :<br>  
<label for="cash1">Visa</label>  
<input type="radio" name="cash" id="cash1"><br>  
<label for="cash2">American Express</label>  
<input type="radio" name="cash" id="cash2"><br>  
<label for="cash3">Mastercard</label>  
<input type="radio" name="cash" id="cash3">  
</p>  
</form>  
</body>  
</html>
```



Il n'y a pas de nouveaux attributs significatifs en Html5.

La ligne de texte avec liste de suggestions

La balise `<datalist>` ajoutée à une ligne de texte ouvre une liste de suggestions d'encodage au focus de celle-ci. L'utilisateur peut retenir une suggestion ou encoder une valeur de son choix. Ceci est proche de ce que Google Suggest présente lors de l'entrée d'un mot-clé dans sa barre de recherche.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
    Choisir un fruit :
    <input type="text" list="fruits">
    <datalist id="fruits">
        <option value="Orange"></option>
        <option value="Pomme"></option>
        <option value="Poire"></option>
        <option value="Prune"></option>
    </datalist>
</form>
</body>
</html>
```

La balise `<datalist>` contient une liste de balises `<option>` reprenant les suggestions d'encodage. Ces balises `<option>` doivent toujours posséder un attribut `value`.

La balise `<datalist>` est reliée à un champ de formulaire par un identifiant `id` qui renvoie à l'attribut `list` de celui-ci, ici la ligne de texte.

Cette balise n'est pas encore reprise par les navigateurs, sauf par Opera 10+.



La ligne de texte d'adresse email

Combien de fois ne vous a-t-on pas demandé votre adresse de courrier électronique lors de votre utilisation de la toile ?

Quel casse-tête aussi pour les concepteurs d'applications Web afin de disposer d'une adresse mail valide !

Le Html5 répond à ce besoin avec la balise <input type="email">.

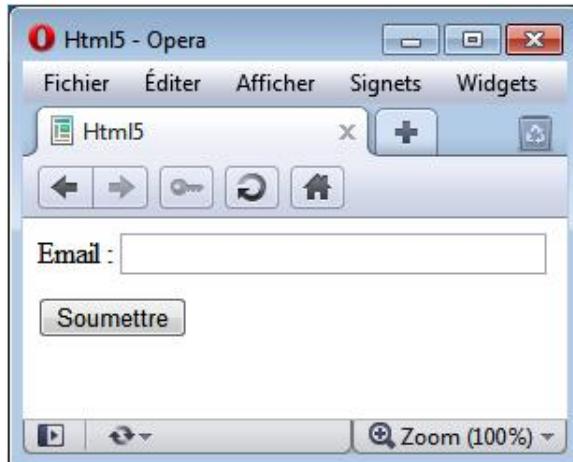
Cette balise n'est encore reprise à ce jour que par le navigateur innovant mais assez confidentiel qu'est Opera. Il nous permettra d'illustrer les perspectives ouvertes par celle nouvelle balise introduite par le Html5.

Elle se révèle particulièrement intéressante par la validation directe par le navigateur, sans l'ajout de JavaScript par le concepteur.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
Email : <input type="email" name="mail" required>
<input type="submit" value="OK">
</form>
</body>
</html>
```

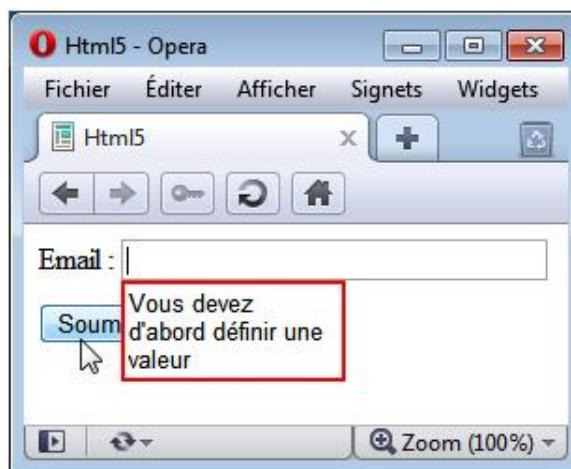
A priori, la ligne de texte introduite par cette balise ne se distingue pas d'une ligne de texte normale.



Cependant, à la soumission du formulaire, une infobulle annonce que l'adresse email est incorrecte.

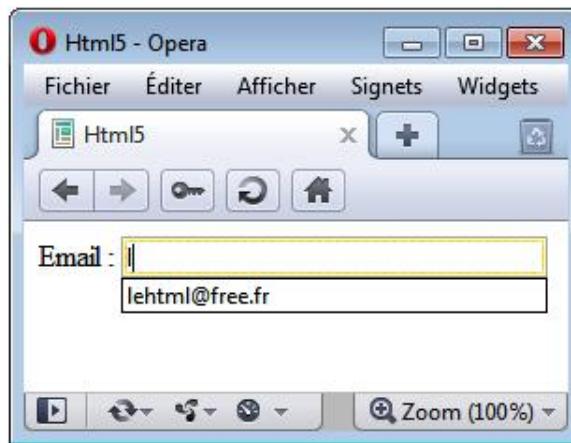


En outre, de par l'attribut `required`, une autre infobulle signale que le champ doit être obligatoirement complété.



Espérons cependant qu'à l'avenir ces infobulles soient plus esthétiques.

Profitons de cette balise `email` pour parler de l'attribut `autocomplete` du Html5 disponible dans plusieurs éléments de formulaires. Pour autant que l'option soit activée dans les préférences du navigateur, cet attribut propose de compléter automatiquement la ligne de formulaire à partir de votre adresse de courrier électronique.



La ligne de texte d'url

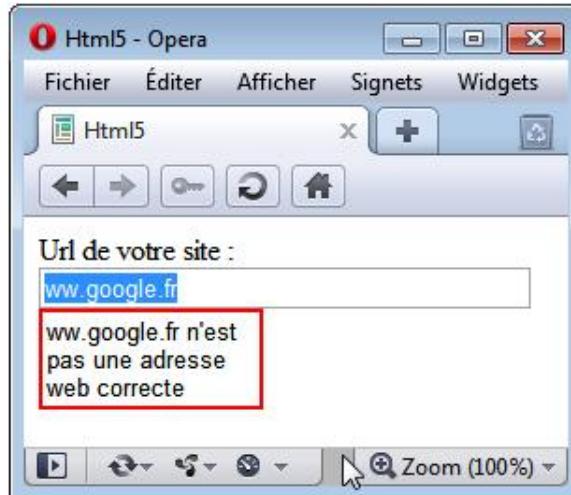
Assez proche de la balise précédente <input type="email">, la balise <input type="url"> est destinée à recueillir des adresses Web (*url*).

Cette balise n'est encore reprise à ce jour que par Opera. Il nous permettra cependant d'illustrer les perspectives ouvertes par cette nouvelle balise introduite par le Html5.

Ici aussi une validation est réalisée en direct (sans JavaScript) par le navigateur lui-même.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
Url de votre site : <input type="url" name="web">
<input type="submit" value="OK">
</form>
</body>
</html>
```



La ligne de texte de format numérique

Autre nouveau formulaire apporté par le Html5, les compteurs numériques. Ces compteurs, que l'on retrouve fréquemment dans les applications logicielles comme les traitements de texte, sont à présent utilisables dans les formulaires Web.

Cette nouvelle balise `<input type="number">` a des attributs spécifiques :

min

Indique la valeur minimale du compteur.

max

Indique la valeur maximale du compteur.

step

Détermine le pas d'avancement à chaque pression de la souris sur la flèche ou .

value

La valeur de départ du compteur.

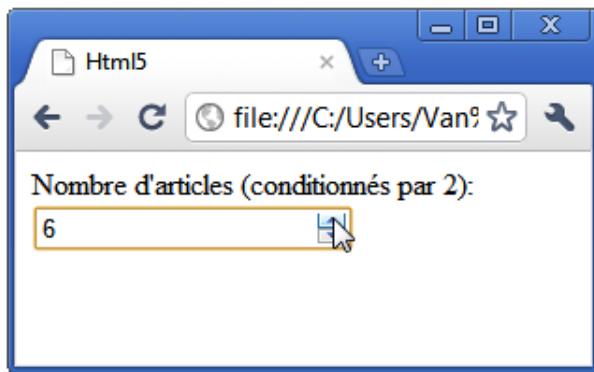
Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
Nombre d'articles (conditionnés par 2) : <br>
<input type="number" name="q" min="2" max="10" step="2" value="2">
</form>
</body>
</html>
```

Au départ :



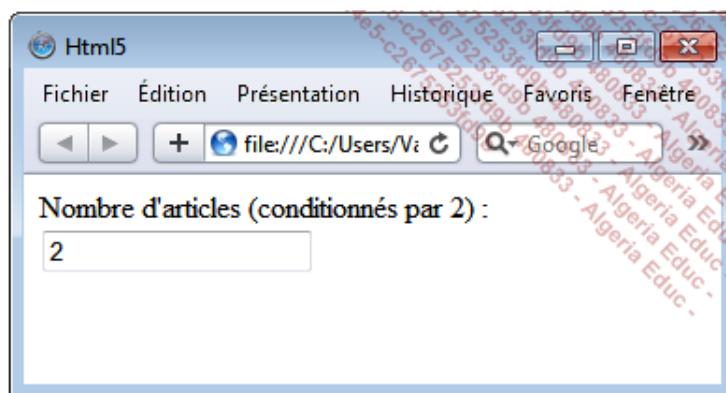
Par une pression des petites flèches, l'utilisateur peut augmenter ou diminuer le compteur.



En utilisant le fichier Html5 disponible dans l'espace de téléchargement, on remarquera que la valeur initiale est de 2, qu'il n'est pas possible de descendre en dessous de 2 et au-dessus de 10 et que chaque pression sur les (petites) flèches augmente ou diminue le compteur de 2 unités (voir le code source de l'exemple).

Commentaires

- Au stade de l'écriture de ce livre, ce compteur n'est implanté que dans les versions récentes d'Opera 10+.
- Le code reste néanmoins compatible avec les navigateurs qui ne prennent pas encore en charge cette nouvelle balise en affichant une simple ligne de texte à compléter manuellement.



Gageons qu'à l'avenir ce nouveau champ de formulaires connaîtra un succès certain notamment dans les sites de vente en ligne.

La ligne de texte de format de date

Les calendriers ont pris une place prépondérante dans les applications du Web 2.0. Leur utilité est indéniable dans tous les sites de réservation en ligne, que ce soit pour un billet d'avion ou d'une chambre d'hôtel.

Pour les développeurs, les formats de date posent un réel problème, surtout pour les sites à vocation internationale (jj/mm/aa ou mm/jj/aa).

À ce jour, ces calendriers étaient affichés en faisant appel à des scripts complexes ou des frameworks JavaScript (par exemple Dojo).

Le HTML5 propose (à l'avenir) des champs de formulaires spécialement adaptés à l'encodage de dates et de faire apparaître automatiquement par navigateur interposé, sans plugins particuliers, un calendrier où l'utilisateur aura le choix dans la date au format souhaité par le concepteur.

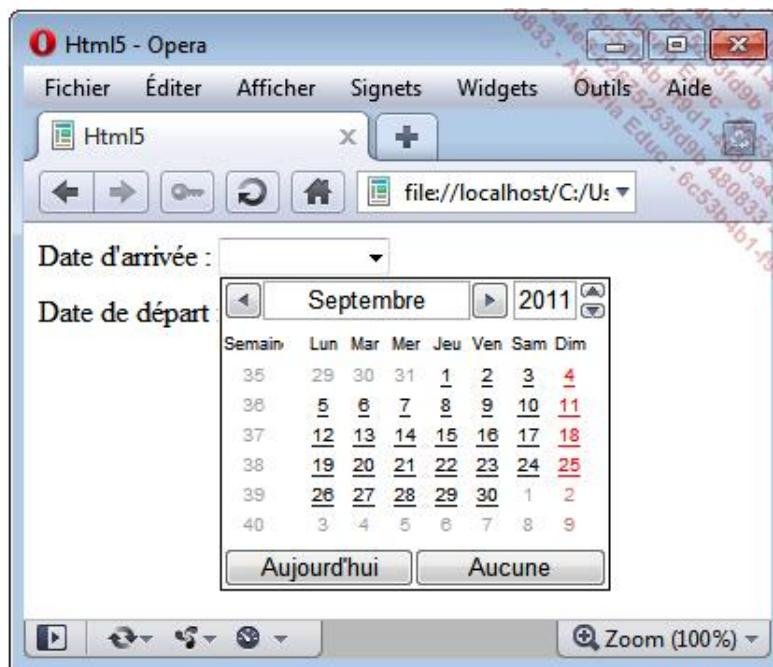
Il s'agit d'une des avancées majeures du HTML5 en termes de conception de sites.

À ce jour, seul Opera 10+ permet d'illustrer cette nouvelle fonctionnalité.

Au départ :



Au focus du champ, le calendrier apparaît :



Le code de ceci tient en quelques lignes :

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

```

<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<form action="">
Date d'arrivée : <input type="date" name="in"><br>
Date de départ : <input type="date" name="out">
</form>
</body>
</html>

```

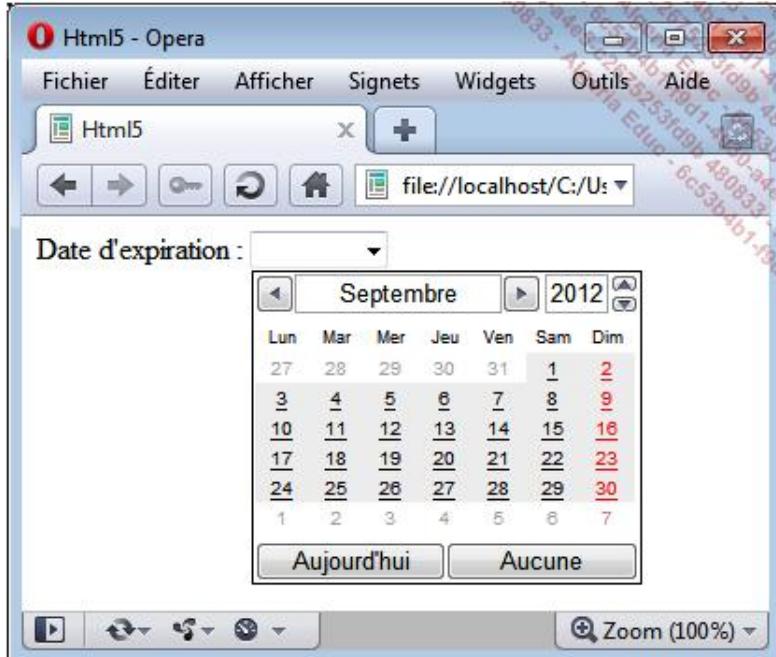
Cette balise `<input>` de format de date se décline sous plusieurs formes.

<input type="date">

La plus générale, illustrée ci-avant. Elle permet de sélectionner l'année, le mois et le jour.

<input type="month">

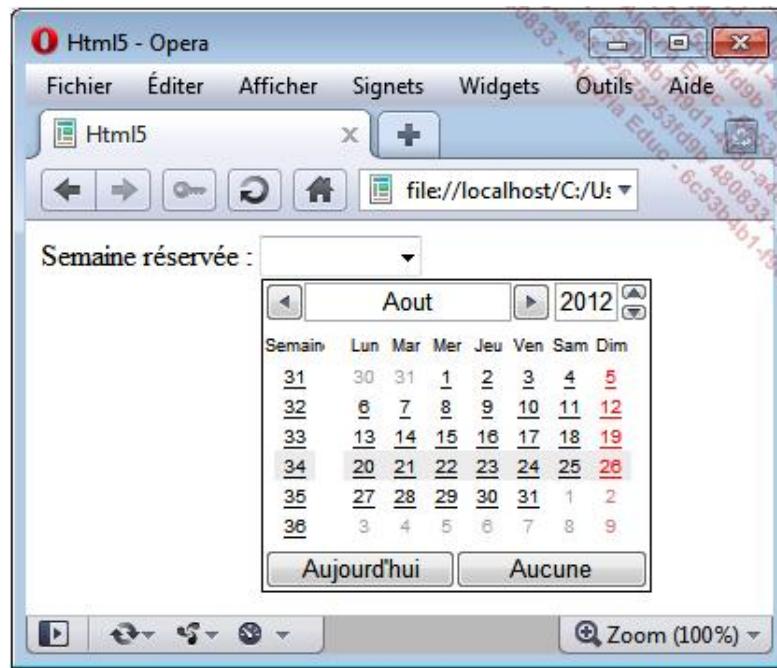
Pour le mois et l'année. Utile par exemple pour la date d'expiration d'une carte de crédit.



Date d'expiration : <input type="month" name="in">

<input type="week">

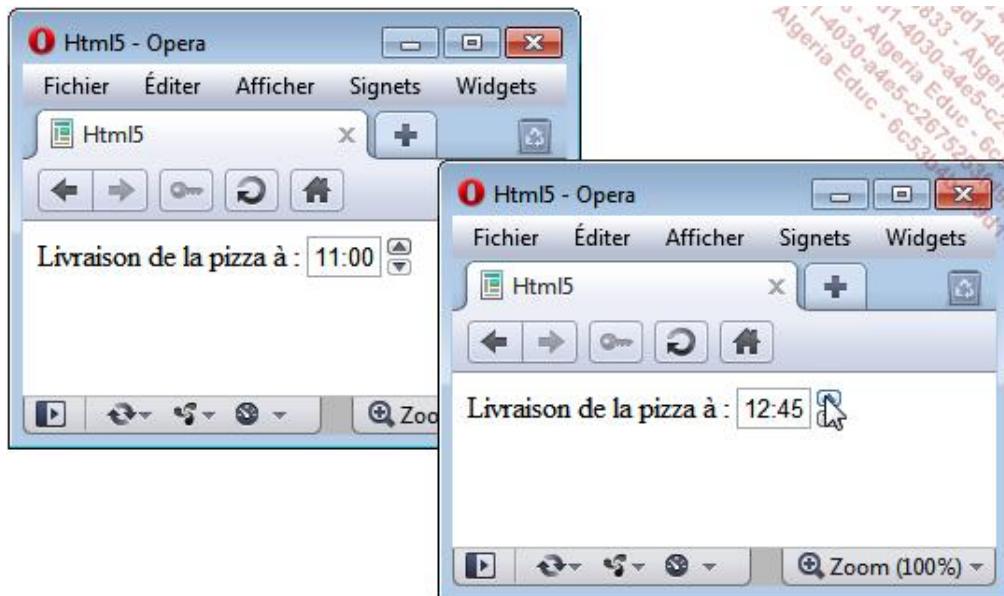
Pour une semaine déterminée.



Semaine réservée : <input type="week" name="in">

Mais Html5 n'en reste pas là. Il propose également un outil pour le format horaire :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<form action="">
Livraison de la pizza à :
<input type="time" min="11:00" max="23:00" step="900"
value="11:00">
</form>
</body>
</html>
```



L'attribut `min` permet de fixer une heure de début du compteur, `max` une heure de fin et `step` l'avancement, ici de 15 en 15 minutes (900 secondes) à chaque pression sur la petite flèche.

Et enfin il est possible de combiner le calendrier et le compteur horaire avec les balises `<input type="datetime">` et `<input type="datetime-local">`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<form action="">
Jour et heure d'arrivée :
<input type="datetime" name="in">
</form>
</body>
</html>
```



La ligne de texte de recherche

Très nombreux sont les sites ou applications Web qui proposent un outil de recherche. Celui-ci prend souvent la forme d'une ligne de texte.

Le Html5 propose avec la balise `<input type="search">` une ligne de texte spécialement dédiée à cette possibilité de recherche.

Cette balise n'est pas encore reprise par les navigateurs.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<form action="">
Moteur de recherche :
<input type="search">
<input type="submit" value "Go">
</form>
</body>
</html>
```

Il se dit dans les blogs spécialisés dans le Html5 que cette ligne de texte dédiée aux outils de recherche pourrait prendre une forme particulière. Pourquoi pas une ligne de texte à bords arrondis, ce qui lui donnerait un look Macintosh OS X ?

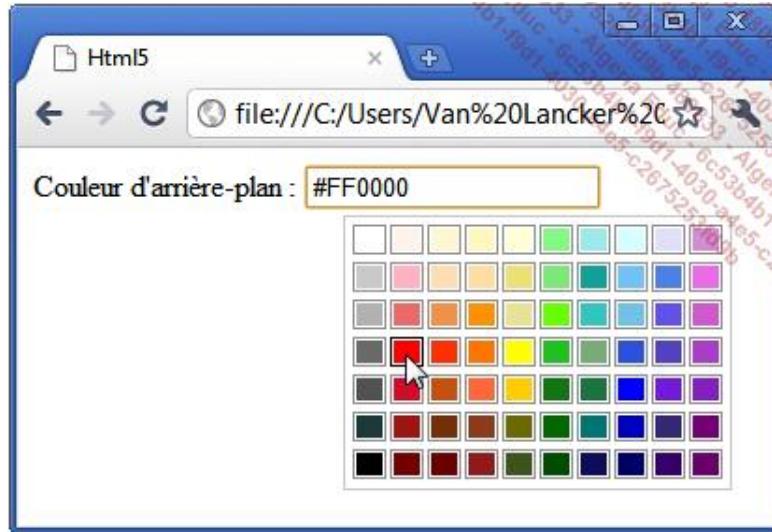


La ligne de texte de couleur

HTML5 prévoit également une ligne de texte destinée à recevoir l'encodage d'une couleur (en hexadécimal) avec la balise `<input type="color">`.

Au focus de cette ligne de texte, une palette de couleur devrait apparaître automatiquement et au choix d'une couleur, la valeur de celle-ci serait reportée.

Nous parlons au conditionnel car à ce jour cette fonctionnalité n'est implémentée dans aucun navigateur. La capture d'écran suivante n'est donc qu'une simulation réalisée à partir du cadriel Dojo.

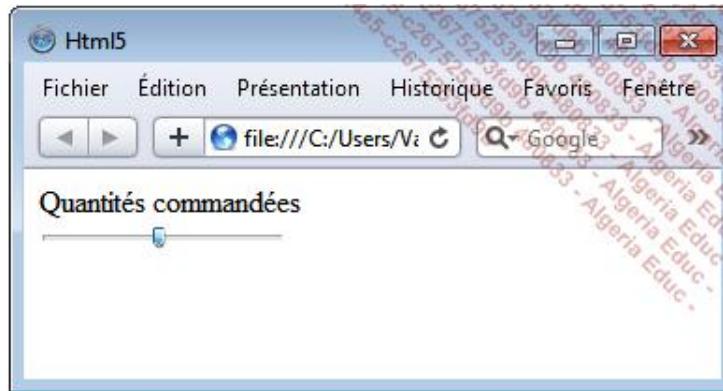


Les curseurs

Les curseurs sont des outils conviviaux et intuitifs pour fixer dans les applications logicielles une valeur, généralement numérique. Ils sont présents ainsi par exemple dans les traitements de texte ou autres programmes graphiques comme Photoshop.

Les curseurs sont introduits dans le Html5 par la balise `<input type="range">`. Gageons que ce contrôle original est appelé à connaître un grand succès auprès des concepteurs et qu'il envahira les applications Web 2.0 dans un avenir proche.

Les curseurs ne sont pour l'instant opérationnels que dans les dernières versions de Chrome 5+, Safari 5+ et Opera 10+. Les autres navigateurs traitent les balises `<input type="range">` comme de simples lignes de texte (`<input type="text">`). Il n'y a donc pas de raisons de s'en priver et de les adopter dès à présent.



Le code est :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<form action="">
Quantités commandées<br>
<input name="curseur" type="range">
</form>
</body>
</html>
```

Cette nouvelle balise `<input type="range">` a des attributs spécifiques, identiques à ceux de `<input type="number">`.

min

Indique la valeur minimale du compteur.

max

Indique la valeur maximale du compteur.

step

Détermine le pas d'avancement à chaque pression de la souris sur la flèche augmenter ou diminuer (voir capture d'écran suivante).

value

La valeur de départ du compteur.

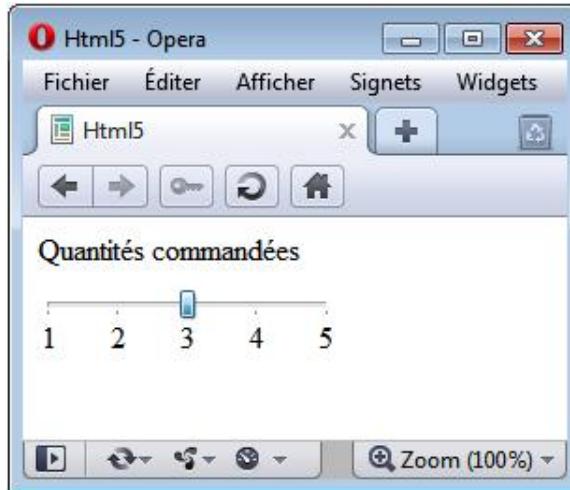
Habillé de quelques feuilles de style, l'exemple suivant met en place un curseur qui se positionne au départ sur 3 (attribut `value`) et dont les valeurs minimales et maximales sont 1 et 5.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
```

```

<style>
span {margin-right: 25px;}
</style>
</head>
<body>
<form action="">
Quantités commandées<br>
<input name="curseur" type="range" min="1" max="5" value="3"
step="1" style="margin-top: 10px"><br>
<div style="margin-top: -2px">
<span style="margin-left: 2px;">1</span>
<span>2</span>
<span>3</span>
<span>4</span>
<span>5</span>
</div>
</form>
</body>
</html>

```



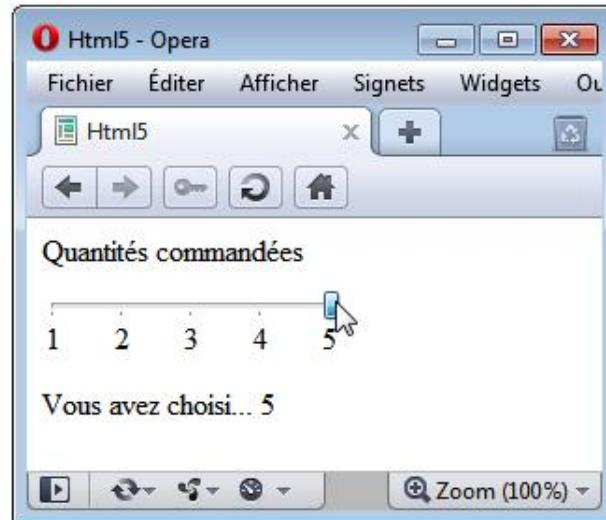
Pour convaincre le lecteur ou confirmer à l'utilisateur le choix effectué, quelques lignes de JavaScript feront apparaître la valeur ainsi retenue.

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
span {margin-right: 25px;}
</style>
<script type="text/javascript">
function afficher(choix){
document.getElementById("valeur").innerHTML = choix;
}
</script>
</head>
<body>
<form action="">
Quantités commandées<br>
<input name="curseur" type="range" min="1" max="5" value="3"
step="1" style="margin-top: 10px"
onchange="afficher(this.value)"><br>
<div style="margin-top: -2px">
<span style="margin-left: 2px;">1</span>
<span>2</span>
<span>3</span>
<span>4</span>
<span>5</span>
</div>
</form>

```

```
<p>Vous avez choisi... <span id="valeur">3</span></p>
</form>
</body>
</html>
```



Une application complète

Élaborons un formulaire pour une activité commerciale. Les premiers champs du formulaire ont trait aux coordonnées du client ; nom et prénom, adresse, code postal et pays (France exclusivement). Les autres éléments ont trait à la commande (le modèle Road 66), la taille et les quantités.

Les groupes **Client** et **Commande** peuvent être regroupés par une balise `<fieldset>`.

Le protocole mailto, malgré ses limites, est utilisé pour l'envoi du formulaire pour des raisons pratiques d'apprentissage.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style>
fieldset { border : 1px solid black; }
</style>
</head>
<body>
<form action="mailto:votre_adresse_email" method="post"
enctype="text/plain">
<fieldset>
<legend><i>Client</i></legend>
Nom et prénom : <input type="text" name="Nom"><br>
Adresse: <input type="text" name="Adresse"><br>
Code Postal : <input type="text" name="CP" size="4"
maxlength="5"><br>
Pays : <input type="text" readonly name="Pays" value="France">
</fieldset>
<br>
<fieldset>
<legend><i>Commande</i></legend>
Modèle : <input type="text" readonly name="Modele" value="Road
66"><br>
Taille : <br>
<input type="radio" name="Taille" value="S">S<br>
<input type="radio" name="Taille" value="M">M<br>
<input type="radio" name="Taille" value="L">L<br>
<input type="radio" name="Taille" value="XL">XL<br>
Quantité : <input type="text" name="Quantite" value="1"><br>
</fieldset>
<br>
<input type="submit" value="Commander">
<input type="reset" value="Recommencer">
</form>
</body>
</html>
```

Client

Nom et prénom :

Adresse:

Code Postal :

Pays : France

Commande

Modèle : Road 66

Taille :

S

M

L

XL

Quantité : 1

Buttons:

À la réception, le courrier peut prendre la forme suivante :

```

Nom=Dumoulin Adrien
Adresse=Rue de la gare, 22 Perpignan
CP=66000
Pays=France
Modele=Road 66
Taille=L
Quantite=2
  
```

Soit à chaque fois le nom donné (`name`) au champ de formulaire, le signe égal = et les données encodées par l'utilisateur.

Ce formulaire est en soi assez similaire à un formulaire HTML 4.0. Il nous paraît intéressant d'y adjoindre des balises et attributs propres au HTML5 et de mettre en avant les nouvelles fonctionnalités apportées par le HTML5.

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>HTML5</title>
<meta charset="iso-8859-1">
<style>
fieldset { border : 1px solid black; }
</style>
</head>
<body>
<form action="mailto:votre_adresse_email" method="post"
enctype="text/plain">
<fieldset>
<legend><i>Client</i></legend>
Nom et prénom : <input type="text" name="Nom" required
placeholder="Nom et prénom"><br>
Adresse: <input type="text" name="Adresse" required><br>
Code Postal : <input type="text" name="CP" size="4" maxlength="5"
required pattern="[0-9]"><br>
Pays : <input type="text" readonly name="Pays" value="France" >
</fieldset>
  
```

```

<br>
<fieldset>
<legend><i>Commande</i></legend>
Modèle : <input type="text" readonly name="Modele" value="Road
66"><br>
Taille : <br>
<input type="range" name="Taille" min="1" max="4" step="1"
value="2">
<div>
<span>S</span>
<span>M</span>
<span>L</span>
<span>XL</span>
</div>
<br>
Quantité : <input type="number" name="Quantite" min="1" max="6"
step="1" value="1"><br>
</fieldset>
<br>
<input type="submit" value="Commander">
<input type="reset" value="Recommencer">
</form>
</body>
</html>

```

The screenshot shows a web browser window titled "Html5 - Opera". The address bar indicates the page is "file:///localhost/C:/U:". The main content area displays a form with two sections: "Client" and "Commande".

Client Section:

- Nom et prénom :** A text input field containing "Nom et prénom".
- Adresse :** A text input field containing "Adresse complète".
- Code Postal :** An empty text input field.
- Pays :** A text input field containing "France".

Commande Section:

- Modèle :** A text input field containing "Road 66".
- Taille :** A range input field with a slider set between S and XL. Below the slider are buttons for "S", "M", "L", and "XL".
- Quantité :** A number input field containing "1".

Buttons:

- Commander**
- Recommencer**

Toolbar:

- Back, Forward, Stop, Home, Refresh, Zoom (100%)

Ce formulaire faisant appel au Html5 est déjà visuellement fort différent avec l'ajout d'un curseur pour les tailles et d'un compteur pour les quantités.

Les champs **Nom et prénom** et **Adresse** se sont vus complétés par une suggestion de saisie qui apparaît en grisé (attribut placeholder).

Divers champs se sont vus complétés par l'attribut required qui les rend obligatoires. À la soumission, une infobulle apparaît pour compléter le champ.

Le code postal s'est vu doté d'une expression régulière (attribut `pattern`) qui réclame un encodage exclusivement numérique. Une infobulle apparaît si l'encodage n'est pas valide.

Le compteur pour les quantités permet d'imposer une quantité maximale, ici 6 pièces, par l'attribut `max`.

Voilà qui devient fort différent du Html 4.0 et ouvre de nombreuses perspectives pour les futures applications Html.

L'insertion d'un fichier audio

1. La balise <audio>

Jusqu'à présent, il n'y avait pas de standard pour ajouter du son dans une application Web. Cette opération s'effectuait par l'appel à un plugin, par exemple Flash, mais tous les navigateurs ne disposaient pas des mêmes plugins.

Le HTML5 propose à présent une nouvelle balise pour reproduire les fichiers sonores, quels que soient les plugins installés chez l'utilisateur.

L'insertion d'un fichier audio se réalise très simplement par la balise :

```
<audio src="fichier_son"></audio>
```

Étant donné l'implémentation encore très limitée de cette balise `<audio>` dans les navigateurs existants, il n'est pas inutile de prévoir au moins un message pour les navigateurs qui ne prennent pas en charge cette balise.

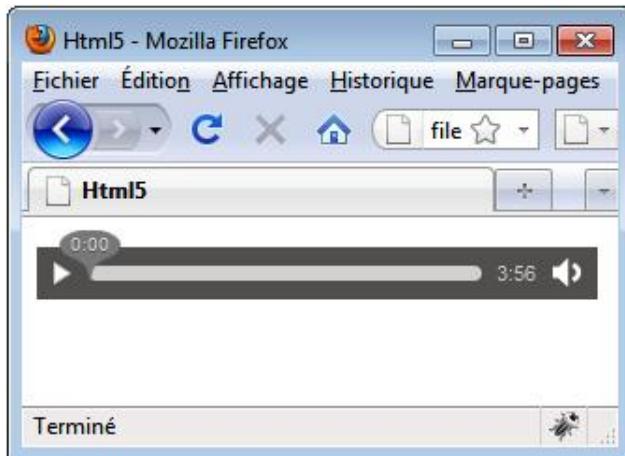
```
<audio src="fichier_son">
Votre navigateur ne supporte pas la balise audio.
</audio>
```

Affichons un premier exemple :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>HTML5</title>
<meta charset="iso-8859-1">
</head>
<body>
<audio src="piano_ogg.ogg" controls>
Votre navigateur ne supporte pas la balise audio.
</audio>
</body>
</html>
```

L'adressage du fichier audio suppose ici que celui-ci se trouve dans le même répertoire que le fichier HTML contenant la balise audio.

Résultat dans Firefox 3.6, qui reconnaît la balise `<audio>` et le format ogg :



Résultat dans Internet Explorer 8, qui ne reconnaît pas la balise `<audio>` :



Les attributs de la balise <audio> sont :

src

Obligatoire. Définit l'adresse du fichier son.

controls

Affiche les contrôles du lecteur audio (voir capture d'écran). Ceux-ci comportent les fonctions de lecture, d'arrêt, d'avancement et de volume.

```
<audio src="son.ogg" controls>
```

En l'absence de l'attribut `controls`, les contrôles du lecteur ne seront pas affichés par le navigateur.

La notation Xhtml des attributs `controls="controls"` est aussi rencontrée.

autoplay

Définit la lecture automatique du fichier audio dès que celui-ci sera disponible, soit au chargement de la page.

```
<audio src="son.ogg" autoplay>
```

Faut-il rappeler que cette lecture automatique du fichier audio n'est pas forcément appréciée par tous les utilisateurs. Ceux-ci écoutent peut-être leur propre musique ou consultent votre site dans un espace où le silence est de rigueur. En outre, cette introduction sonore sera très gênante pour les utilisateurs d'une synthèse vocale.

La notation Xhtml de l'attribut `autoplay="autoplay"` est également acceptée.

loop

Spécifie que le fichier son sera joué en boucle. Ainsi, le morceau sonore est joué à nouveau lorsqu'il se termine.

```
<audio src="son.ogg" autoplay>
```

La notation `loop="loop"` est aussi acceptée.

preload

Indique au navigateur qu'il doit télécharger le fichier audio au chargement de la page de sorte qu'il soit disponible pour une lecture immédiate lors de la demande de l'utilisateur.

```
<audio src="son.ogg" preload>
```

Cet attribut peut être précisé :

- `preload="none"`. Pas de préchargement.
- `preload="metadata"`. Préchargement des métadonnées (metadata) attachées au fichier.
- `preload="auto"`. Préchargement automatique.

Cet attribut `preload` est ignoré si l'attribut `autoplay` est présent.



Il faut noter la concision de l'insertion d'un fichier sonore grâce à cette balise `<audio>` et ses attributs en nombre réduit.

2. Les formats de fichier audio

La situation idéale aurait été qu'un seul format de fichier soit repris et qu'en outre ce format soit un format libre.

Le format ogg

Ce fut l'option de Firefox qui le premier implémenta la balise `<audio>` dans son navigateur. Le format libre ogg a été retenu.

Ogg est le nom du principal projet de la fondation Xiph.org dont le but est de proposer à la communauté des formats et codecs multimédias ouverts, libres et dégagés de tout brevet. L'extension .ogg est une des extensions possibles pour les fichiers au format ogg. Par abus de langage, on appelle couramment "fichier Ogg" un fichier audio au format ogg contenant des données audio compressées en Vorbis, l'un des codecs du projet ogg (source Wikipédia).

Les fichiers ogg constituent une alternative libre au format mp3, beaucoup plus connu et répandu.

Le format ogg est reconnu par Firefox 3.6+, Chrome 5+ et Opera 10.6+. Il ne l'est pas par Safari 5+ et Internet Explorer 9.

Le format mp3

Difficile de se passer de l'emblématique format mp3 pour les fichiers son.

Le MPEG-1/2 Audio Layer 3, plus connu sous son abréviation de MP3, est la spécification sonore du standard MPEG-1/MPEG-2, du *Moving Picture Experts Group* (MPEG). C'est un algorithme de compression audio capable de réduire drastiquement la quantité de données nécessaire pour restituer de l'audio, mais qui, pour l'auditeur, ressemble à une reproduction du son original non compressé, c'est-à-dire avec perte de qualité sonore significative mais acceptable pour l'oreille humaine (source Wikipédia).

Le mp3 est souvent perçu par l'utilisateur final comme une technologie gratuite. Celui-ci peut en effet coder ou décoder sa musique de manière tout à fait légale pour peu que l'enregistrement original lui appartienne ou qu'il soit une copie à usage privé. Pourtant, cette technologie propriétaire fait l'objet de brevets et d'une licence commerciale.

Le format mp3 est reconnu par Chrome 5+, Safari 5+ et Internet Explorer 9. Il ne l'est pas par Firefox 3.6 et 4 et Opera 10.6.

Le format acc

Il y a plus performant en termes de compression que le format ogg et mp3. C'est le format acc.

Le format ACC (*Advanced Audio Coding*) est un algorithme de compression audio avec perte de données ayant de meilleures performances en termes de compression que le format plus ancien MPEG-1/2 Audio Layer 3 (plus connu sous le nom de mp3). Pour cette raison, il a été choisi par différentes firmes comme Apple ou RealNetworks (source Wikipédia).

Le format acc est le format des fichiers audio supportés par Apple au sein de son baladeur numérique iPod et de son logiciel iTunes.

Il s'agit ici également d'un format propriétaire.

Le format acc est reconnu par Chrome 5+, Safari 5+ et Internet Explorer 9. Il ne l'est pas par Firefox 3.6 et 4, Opera 10.6.

Le format wav

Citons pour mémoire l'antique format wav mais qui, par son absence de compression, n'apparaît pas des plus adaptés à la toile. En effet, la taille des fichiers wav est souvent très (ou trop) volumineuse.

Le format wav est reconnu par Firefox 3.6+, Safari 5+ et Opera 10.6+. Il ne l'est pas par Chrome 5+ et Internet Explorer 9.

Tableau comparatif

	Chrome 5+	Firefox 3.6+	Opera 10.6+	Safari 5+	Explorer 9
ogg	oui	oui	oui	non	non

mp3	oui	non	non	oui	oui
acc	oui	non	non	oui	oui
wav	non	oui	oui	oui	non

En guise de conclusion :

- La balise `<audio>` est bien implémentée dans les navigateurs récents.
- Google Chrome 5+ apparaît de loin le plus polyvalent en ce qui concerne les formats audio.

Le Html5 n'est encore qu'au stade naissant. Un nouveau format audio peut encore apparaître. Un navigateur peut encore reconnaître un format à ce jour non retenu. L'auteur vous conseille de consulter régulièrement la rubrique Html5 Audio Codecs du site www.findmebyip.com/litmus/ pour les développements futurs possibles.

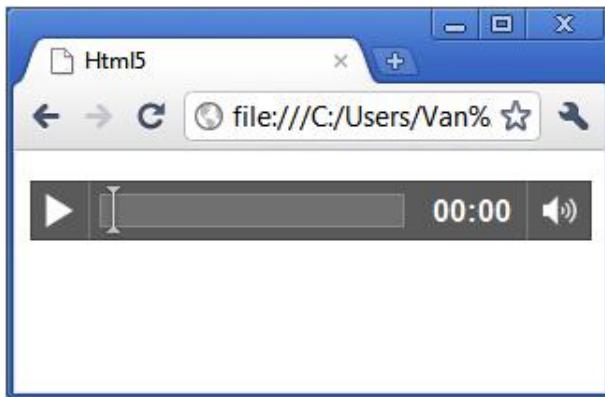
3. La balise `<source>`

La balise `<source>` va nous permettre de résoudre cette problématique des formats de fichiers différents.

La balise `<source>` est utilisée pour spécifier plusieurs ressources audio. À charge du navigateur de choisir le format qu'il prend en compte.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<audio controls>
<source src="piano_ogg.ogg">
<source src="piano_mp3.mp3">
<source src="piano_acc.acc">
Votre navigateur ne supporte pas la balise audio.
</audio>
</body>
</html>
```



Les fichiers sont disponibles dans l'espace de téléchargement.

Il est important de comprendre le fonctionnement de ces différentes balises `<source>`.

Supposons que cette page est lue par le navigateur Safari qui, pour rappel, ne reconnaît pas le format ogg. Safari ignore la première balise `<source>` qui renvoie à un fichier ogg. Il passe à la seconde balise `<source>` qui propose un fichier mp3. Comme le format mp3 est reconnu par Safari, c'est cette ressource qui sera utilisée pour reproduire le fichier son.

Par contre, Firefox, qui gère les fichiers ogg, reprend la première ressource. Les autres balises `<source>` sont ignorées.

Les balises `<source>` sont lues dans l'ordre d'apparition dans le code. Le navigateur utilisera la première ressource qu'il prend en charge.

Ces multiples balises `<source>` assurent une parfaite prise en charge des différents formats de fichier audio. Cependant, le concepteur du site ou de l'application Web doit encoder le fichier audio plusieurs fois afin de le convertir dans les différents formats.

Les attributs de la balise `<audio>` sont :

src

Obligatoire. Définit l'adresse du fichier son.

type

Définit le type MIME du contenu. Soit :

- `type="audio/ogg"`.
- `type="audio/mpeg"`.
- `type="audio/acc"`.

On peut également spécifier le codec utilisé. L'attribut `type` devient alors : `type="audio/ogg; codecs=vorbis"`.

En spécifiant l'attribut `type`, vous accélérez le processus de prise en charge de la balise `<source>` la plus adéquate pour le navigateur.

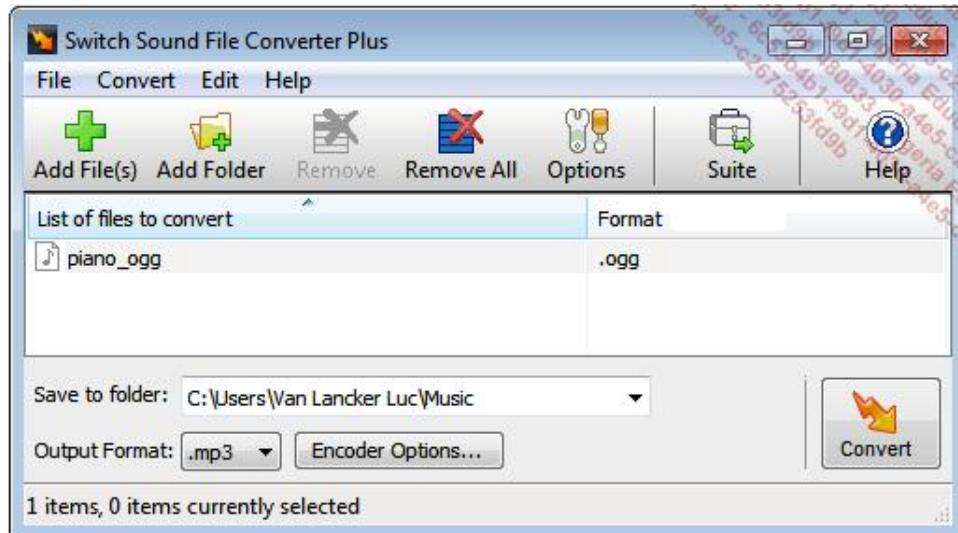
4. Des convertisseurs audio

De nombreux convertisseurs audio, gratuits ou en version d'évaluation sont disponibles sur la toile. Épinglons les deux logiciels suivants.

Switch Audio Converter

Logiciel gratuit. Téléchargeable à l'adresse www.nch.com.au/switch/index.html. Existe pour Windows et pour Mac.

Switch permet de convertir au format ogg, mp3 et acc mais aussi en bien d'autres formats audio (wma, flac, raw...). En outre, la qualité du fichier de sortie peut être ajustée afin d'en réduire la taille.



Audacity

Avec Audacity, vous disposez d'un éditeur audio relativement complet et facile d'utilisation. Il est disponible pour Windows, Mac OS X, Linux et bien d'autres systèmes d'exploitation.



Il vous permet d'éditer les fichiers audio par copier/coller, mixer plusieurs pistes, ajouter des effets de fondu en ouverture et/ou en fermeture, ajouter des effets sonores, corriger le bruit et le volume, etc.

Au niveau de la conversion des formats audio, Audacity propose de très nombreux formats d'exportation dont bien entendu les formats ogg, mp3 et ACC de la balise <audio>.

5. Compatibilité avec les anciens navigateurs

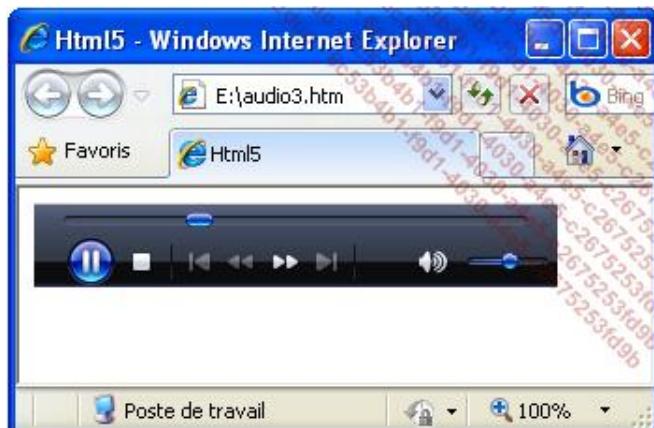
Il est possible d'assurer une compatibilité pour les navigateurs qui ne supportent pas la balise <audio> comme par exemple Internet Explorer 8.

Nous avons vu qu'un contenu alternatif pouvait être fourni à la balise <audio>. Plutôt qu'une simple phrase annonçant que le navigateur ne prend pas en charge cette balise, il est tout à fait possible d'inclure les balises <object> et <param> nécessaires à cette ancienne génération de navigateurs pour lire les fichiers audio.

Le code perd ainsi en simplicité mais la compatibilité sera assurée.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<audio controls>
<source src="piano_ogg.ogg">
<source src="piano_mp3.mp3">
<source src="piano_acc.acc">
<object type="audio/mpeg" data="piano_mp3.mp3" width="280"
height="45"
classid='CLSID:22d6f312-b0f6-11d0-94ab-0080c74c7e95'>
<param name="src" value="piano_mp3.mp3">
<param name="autoplay" value="false">
<param name="loop" value="false">
<param name="controller" value="true">
</object>
</audio>
</body>
</html>
```

Le fichier est alors compatible dans Internet Explorer 8.



L'insertion d'un fichier vidéo

1. La balise <video>

Depuis quelques années, la vidéo a envahi le Web. Il suffit, par exemple, de penser au succès de YouTube. Pourtant, en Html 4.0, il n'y avait pas de standard pour ajouter de la vidéo dans une application Web. Cette opération s'effectuait par l'appel à un plugin : QuickTime, RealPlayer ou Flash. Le tout était d'avoir le bon plugin au bon moment.

Le Html5 propose à présent une nouvelle balise pour reproduire les fichiers vidéo quels que soient les plugins installés chez l'utilisateur.

L'insertion d'un fichier vidéo se réalise très simplement par la balise :

```
<video src="fichier_video"></video>
```

Étant donné l'implémentation encore très limitée de cette balise <video> dans les navigateurs existants, il n'est pas inutile de prévoir au moins un message pour les navigateurs qui ne prennent pas en charge cette balise.

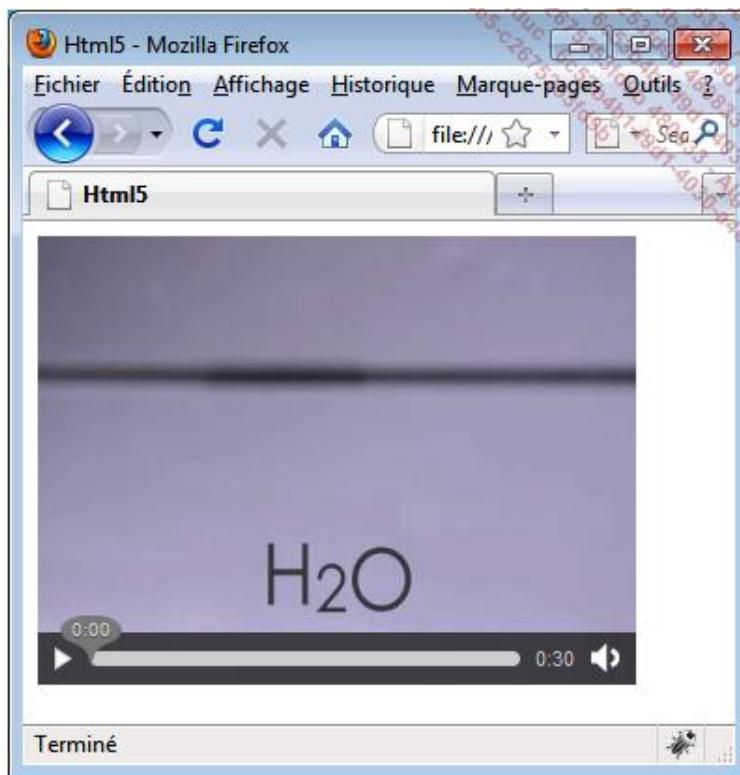
```
<video src="fichier_video">
Votre navigateur ne supporte pas la balise vidéo.
</video>
```

Affichons un premier exemple.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<video src="video_ogv.ogv" controls>
Votre navigateur ne supporte pas la balise vidéo.
</video>
</body>
</html>
```

L'adressage du fichier vidéo suppose ici que celui-ci se trouve dans le même répertoire que le fichier Html contenant la balise vidéo.

Résultat dans Firefox 3.6 qui reconnaît la balise <video> et le format ogv.



Les attributs de la balise <video> sont :

src

Obligatoire. Définit l'adresse du fichier vidéo.

width

Détermine la largeur de la vidéo.

height

Détermine la hauteur de la vidéo.

Comme pour la balise , vous pouvez définir explicitement les dimensions de votre vidéo. Sinon, l'élément sera par défaut affiché à la largeur et la hauteur du fichier vidéo. Si vous spécifiez une dimension mais pas l'autre, le navigateur va ajuster la taille de la dimension non spécifiée afin de préserver les proportions de la vidéo.

poster

L'attribut poster permet de spécifier une image que le navigateur utilisera alors que la vidéo est en cours de téléchargement ou jusqu'à ce que l'utilisateur commence la lecture de la vidéo. Si cet attribut n'est pas spécifié, la première image de la vidéo sera utilisée à la place.

```
<video src="video_ogv.ogv" poster="image.png">
```

controls

Affiche les contrôles du lecteur de vidéo (voir capture d'écran). Ceux-ci comportent les fonctions de lecture, d'arrêt, d'avancement et de volume.

```
<video src="video_ogv.ogv" controls>
```

En l'absence de l'attribut controls, les contrôles du lecteur ne seront pas affichés par le navigateur.

La notation Xhtml des attributs controls="controls" est aussi rencontrée.

autoplay

Définit la lecture automatique du fichier vidéo dès que celui-ci sera disponible, soit au chargement de la page.

```
<video src="video_ogv.ogv" autoplay>
```

Nous rappelons que cette lecture automatique n'est pas appréciée par tous les utilisateurs surtout si, dans le cas

présent, la page comporte un contenu textuel.

La notation Xhtml de l'attribut `autoplay="autoplay"` est également acceptée.

loop

Spécifie que le fichier vidéo sera joué en boucle. Ainsi, la vidéo est jouée à nouveau lorsqu'elle se termine.

```
<video src="video_ogv.ogv" loop>
```

La notation `loop="loop"` est aussi acceptée.

preload

Indique au navigateur qu'il doit télécharger le fichier vidéo au chargement de la page de sorte qu'il soit disponible pour une lecture immédiate lors de la demande de l'utilisateur.

```
<video src="video_ogv.ogv" preload>
```

Cet attribut peut être précisé :

- `preload="none"`. Pas de préchargement.
- `preload="metadata"`. Préchargement des metadata attachées au fichier.
- `preload="auto"`. Préchargement automatique.

Cet attribut `preload` est ignoré si l'attribut `autoplay` est présent.

2. Les formats de fichier vidéo

Tout comme pour les fichiers audio, il existe plusieurs formats de fichiers vidéo.

Format ogv

Le format ogv est le pendant vidéo de l'extension ogg pour les fichiers audio.

Les fichiers de vidéo ogv reprennent le codec Theora pour la vidéo et Vorbis pour la partie audio.

Rappelons que les formats ogg et ogv sont des formats libres.

Ce format fut repris par Firefox 3.6 dès l'implémentation de la balise vidéo (juin 2009). Opera 10.5+ et Chrome 3+ le prennent également en compte.

Format H.264

Le navigateur Safari pour Mac et PC fait largement appel à QuickTime qui reconnaît de nombreux formats mais pas le format ogv et les codecs Theora et Vorbis. Il se tourna alors vers le codec H.264.

H.264 est un codec de compression vidéo numérique des images et vidéo haute définition à la norme MPEG-4, développé par le VCRG (*Video Coding Experts Group*) en partenariat avec le MPEG (*Moving Picture Experts Group*), aussi connu sous l'appellation AVC (*Advanced Video Coding*). Le format H.264 est aussi repris sous le vocable MPEG-4 AVC ou encore MPEG-4 Part 10.

Ce format, soutenu par Apple, connaît un succès indéniable surtout depuis son adoption par YouTube.

Mais le format H.264 n'est pas un format libre et gratuit. C'est un format propriétaire qui est soumis à des redevances et royalties.

Devant les protestations des défenseurs d'un Web libre, le consortium MPEG-LA a d'abord libéralisé le format H.264 pour les sites et les plates-formes proposant gratuitement leurs contenus vidéo jusqu'en 2011. Cette échéance fut repoussée ensuite jusqu'en 2016. Finalement sous la poussée d'un nouveau format libre, le WebM, soutenu par Google, le consortium a décidé de rendre le format libre de royalties dans les cas où la vidéo est gratuitement accessible par l'utilisateur final.

Belle illustration de la guerre des formats vidéo à laquelle nous assistons pour l'instant...

Le format H.264 est reconnu par Chrome 5+, Safari 5+ et Internet Explorer 9. Il ne l'est pas par Firefox et Opera 10.6.

Format WebM

Google est devenu propriétaire du codec vidéo VP8 depuis le rachat de l'entreprise On2 Technologies.

Pour contrer l'ascension fulgurante du format H.264, Google a publié en mai 2010 le code source du codec VP8 sous licence libre. Il s'est associé à Mozilla et Opera pour proposer un format audio-vidéo ouvert, le WebM, qui rassemble VP8 pour les flux vidéo et Vorbis pour les flux audio.

Le format WebM est reconnu par Firefox 4+, Chrome 6+ et Opera 10.6+. Il ne l'est pas par Safari 5+ et Internet Explorer 9.

Tableau comparatif

	Chrome		Firefox		Opera 10.6+	Safari 5+	Explorer 9
	5	6	3.6	4			
ogv	oui	oui	oui	oui	oui	non	non
H.264	oui	oui	non	non	non	oui	oui
WebM	non	oui	non	oui	oui	non	non

En guise de conclusion :

- La balise <video> est bien implémentée dans les navigateurs récents.
- Google Chrome 6+ apparaît de loin le plus polyvalent en ce qui concerne les formats vidéo.

Le Html5 n'est encore qu'au stade naissant. Un nouveau format vidéo peut encore apparaître. Un navigateur peut encore reconnaître un format à ce jour non retenu. L'auteur vous conseille de consulter régulièrement la rubrique Html5 Video Codecs du site www.findmebyip.com/litmus/ pour les développements futurs possibles.

3. La balise <source>

La balise <source> permet de résoudre cette problématique des formats de fichiers différents.

La balise <source> est utilisée pour spécifier plusieurs ressources vidéo. À charge du navigateur de choisir le format qu'il prend en compte.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<video controls>
<source src="video_ogv.ogv">
<source src="video_mp4.mp4">
<source src="video_webm.webm">
Votre navigateur ne supporte pas la balise vidéo.
</video>
</body>
</html>
```



Les fichiers sont disponibles dans l'espace de téléchargement.

Il est important de comprendre le fonctionnement de ces différentes balises `<source>`.

Soumettons ce fichier au navigateur Safari qui pour rappel, ne reconnaît pas le format ogv. Safari ignore la première balise `<source>` qui renvoie à un fichier ogv. Il passe à la seconde balise `<source>` qui propose un fichier mp4. Comme le format mp4 est reconnu par Safari, c'est cette ressource qui sera utilisée pour reproduire le fichier vidéo.

Par contre, Firefox, qui gère les fichiers ogv, reprend la première ressource. Les autres balises `<source>` sont ignorées.

Les balises `<source>` sont lues dans l'ordre d'apparition dans le code. Le navigateur utilisera la première ressource qu'il prend en charge.

Ces multiples balises `<source>` assurent une parfaite prise en charge des différents formats de fichier vidéo. Cependant, le concepteur du site ou de l'application Web doit encoder le fichier vidéo plusieurs fois afin de le convertir dans les différents formats.

Les attributs de la balise `<source>` sont :

src

Obligatoire. Définit l'adresse du fichier vidéo.

type

Définit le type MIME du contenu. Soit :

- `type="video/ogg"`.
- `type="video/mp4"`.
- `type="video/webm"`.

On peut également spécifier les codecs utilisés. L'attribut `type` devient :

- `type='video/ogg; codecs='theora, vorbis'"'`.
- `type='video/webm; codecs='vp8, vorbis'"'`.
- `type='video/mp4; codecs='avc1.42E01E, mp4a.40.2'"'`.

En spécifiant l'attribut `type`, vous accélérez le processus de prise en charge de la balise `<source>` la plus adéquate pour le navigateur.

4. Des convertisseurs vidéo

De nombreux convertisseurs gratuits ou en version d'évaluation sont disponibles sur la toile. Nous en avons retenu quelques-uns.

Miro Video Converter

Miro Video Converter est logiciel libre pour Windows et Mac. Il peut être téléchargé à l'adresse : www.mirovideoconverter.com/

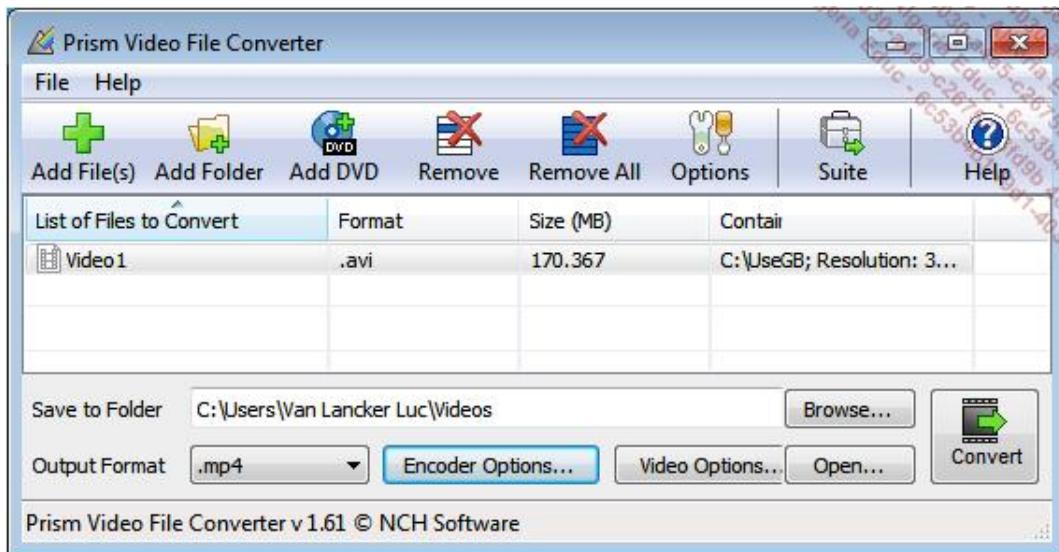
Il convertit n'importe quelle vidéo pour permettre sa lecture sur iPhone, Android, PSP, etc. Glissez/déposez simplement le fichier à transformer sur l'interface du logiciel puis sélectionnez le format souhaité (MP4, Ogg Theora, WebM, etc.) ou l'appareil de destination : Android, PSP, iPhone, iPad, etc. Enfin, cliquez sur **Convert!** et le tour est joué !



Il est difficile de faire plus simple en matière de conversion de fichiers vidéo. Seul inconvénient, il n'est pas possible de configurer précisément les paramètres de conversion.

Prism Video Converter

Prism Video Converter est un convertisseur vidéo gratuit qui prend en charge la plupart des formats vidéo. Vous pourrez ainsi encoder vos fichiers AVI, DivX, MPG, VOB, WMV, 3GP... aux formats compatibles PSP, iPod, mobiles, lecteur de salon, etc. Très simple d'utilisation, cette application comporte également des paramètres de configuration et de personnalisation tels que le taux de compression et la résolution des fichiers sortants, ou encore l'accès direct par le menu contextuel.



Firefogg

Firefogg est une extension de Firefox qui convertit automatiquement les vidéos vers un format ouvert (Ogv et WebM). Téléchargeable à l'adresse <http://firefogg.org/>.



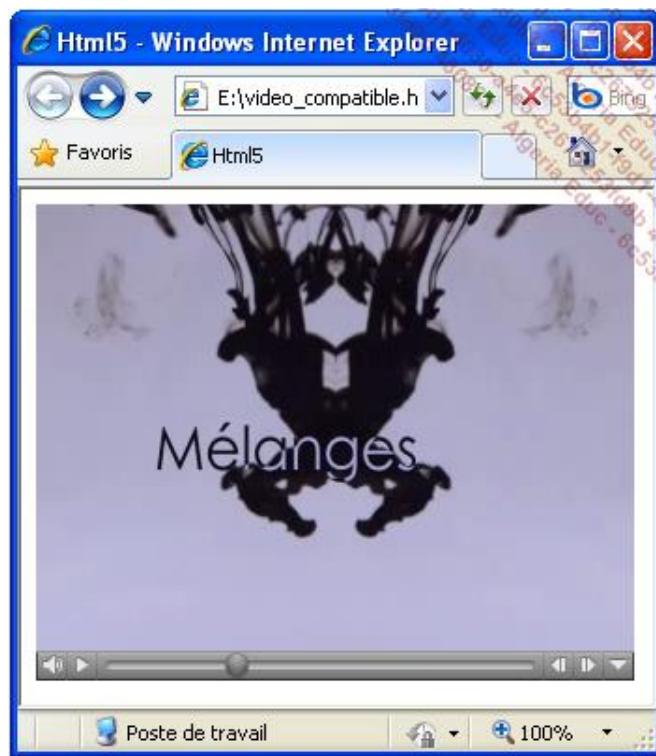
Les conversions avec Firefogg sont parfaitement paramétrables.



Comme Firefogg est une extension de Firefox, l'utilisateur est ainsi assuré de disposer des derniers progrès en ce qui concerne les formats retenus.

5. Compatibilité avec les anciens navigateurs

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<video controls>
<source src="video_ogv.ogv">
<source src="video_mp4.mp4">
<source src="video_webm.webm">
<object classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
codebase="http://www.apple.com/qtactivex/qtplugin.cab" width="550"
height="320" standby="Chargement...">
<param name="src" value="video_mp4.mp4">
<param name="type" value="application/x-mplayer2">
<param name="autoplay" value="true">
<param name="quality" value="high">
<p>La vidéo ne peut être affichée. Lien vers le <a href="http://www.apple.com/quicktime/download/">plug-in</a> à télécharger ou lien vers le <a href="video_mp4.mp4">fichier vidéo</a> directement.</p>
</object>
</video>
</body>
</html>
```



Introduction

La nouvelle balise `<canvas>` permet d'intégrer des zones de dessin en 2D dans la page. Ces dessins permettent, par exemple, d'afficher des graphiques du genre Excel sans passer par des captures d'écran. En outre, ces éléments pourront être rendus dynamiques à partir de scripts JavaScript.

Cette nouvelle fonctionnalité `canveas` est bien reconnue par les navigateurs modernes de notre étude : Safari 5+, Chrome 7+, Opera 10.6+, Internet Explorer 9+, et n'oublions pas Firefox 3.6+, qui a été l'instigateur de cette balise `<canvas>`.

Celle-ci laisse entrevoir de belles opportunités en termes de design et de graphisme, surtout que le dessin 3D est annoncé. Certains n'hésitent pas à y voir un concurrent sérieux de Flash.

La balise <canvas>

La balise <canvas> introduit une zone qui accueillera les dessins et autres graphismes.

Les dimensions de cette zone sont introduites par les attributs width et height qui en fixent la largeur et la hauteur.

```
<canvas width="200px" height="200px"></canvas>
```

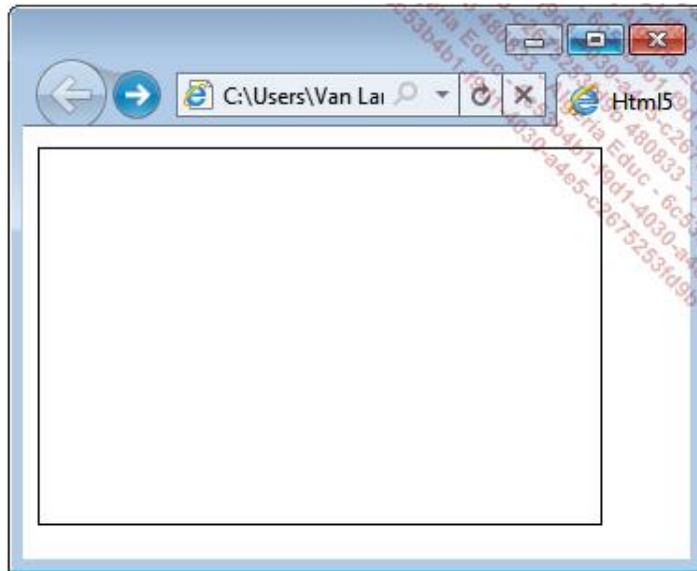
Cette zone de rendu est par défaut transparente. Une capture d'écran à ce stade est inutile.

Pour visualiser cette zone introduite par la balise <canvas>, ajoutons-lui, par une propriété de style, une simple bordure de 1px.

En outre, prévoyons un identifiant de type id afin de pouvoir utiliser cette zone graphique initiée par la balise <canvas>.

Le code devient ainsi :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="zone" width="200px" height="200px" style="border: 1px solid black">
Votre navigateur ne supporte pas la balise canvas.
</canvas>
</body>
</html>
```



La balise <canvas> n'est compatible qu'avec les toutes dernières versions des navigateurs. Il est ainsi prudent de prévoir un contenu alternatif pour les navigateurs plus anciens qui ne prennent pas en charge cette dernière :

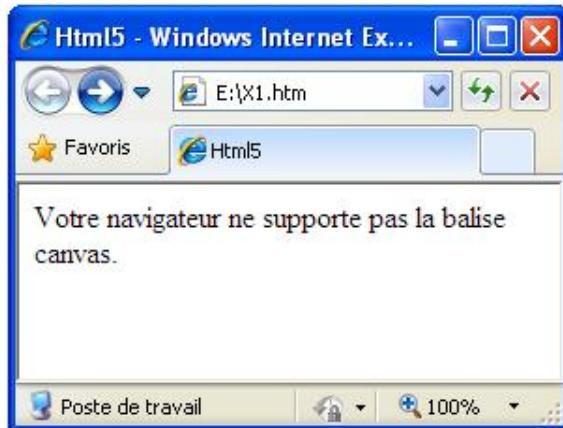
```
<canvas id="zone" width="200px" height="200px" style="border:
1px solid black">
Votre navigateur ne supporte pas la balise canvas.
</canvas>
```

Le code complet est alors :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
```

```
</head>
<body>
<canvas id="zone" width="200px" height="200px" style="border: 1px
solid black">
Votre navigateur ne supporte pas la balise canvas.
</canvas>
</body>
</html>
```

En voici le résultat dans Internet Explorer 8, qui ne reconnaît pas la balise <canvas> :



Commentaires

- Lorsque les attributs `width` et `height` ne sont pas spécifiés, le canevas sera initialement de 300 pixels de large et 150 pixels de haut.
- Il est possible d'avoir plusieurs balises `<canvas>` sur la même page. L'identifiant `id` prend dans ce cas toute son utilité.
- L'élément `<canvas>` peut être stylé de la même façon que n'importe quelle image normale (marges, bordures, fond, etc.). Ces règles n'affectent en rien le dessin du canevas lui-même.

Appel de l'API de dessin

Et c'est tout pour le Html. Celui-ci passe la main au JavaScript pour mettre en œuvre les dessins et autres graphismes de la zone de rendu.

Ceci peut être déconcertant pour ceux qui découvrent l'écriture de pages Web par le Html. JavaScript est un langage de programmation qui vient parfois s'ajouter au code Html. Les instructions JavaScript sont gérées par le navigateur lui-même (côté-client) et apportent une dose d'interactivité aux applications.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="zone" width="200px" height="150px" style="border: 1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>
<script type="text/javascript">
var canevas = document.getElementById( "zone" );
var contexte = myCanvas.getContext( "2d" );
// Suite du script
</script>
</body>
</html>
```

Détaillons ces nouvelles lignes de code.

```
<script type="text/javascript">
...
</script>
```

Les scripts JavaScript sont introduits par la balise `<script type="text/javascript"> ... </script>`. Cette balise peut se placer dans le `<head>` ou le `<body>` du document Html.

```
var canevas = document.getElementById("zone");
```

La variable canevas contient l'élément du document (`document`) identifié par l'identifiant zone (`getElementById ("zone")`). Il faut respecter scrupuleusement la casse de `getElementById`.

```
var contexte = canevas.getContext("2d");
```

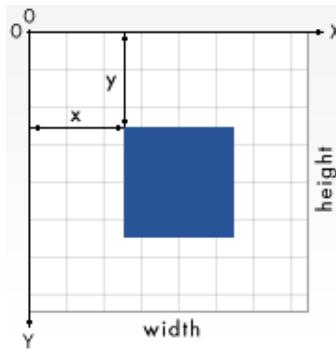
La variable contexte va chercher l'application API de dessin 2D (`getContext("2d")`) à appliquer à la variable canevas.

Commentaires

Les spécifications du Html5 prévoient pour l'avenir des dessins en 3D. Des applications, au stade expérimental, fonctionnent déjà sous Opera 10 mais on est encore loin d'une quelconque standardisation.

Dessiner un rectangle

Avant de passer au code relatif aux dessins, il est nécessaire de spécifier la notation utilisée pour les coordonnées.



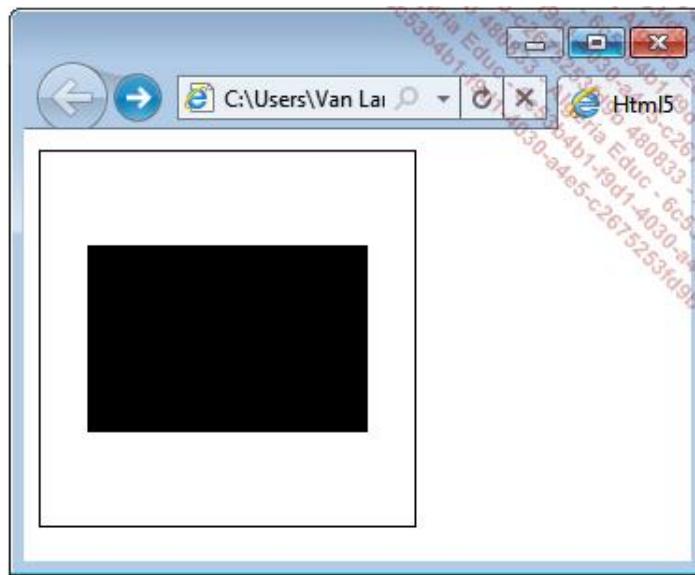
L'origine des axes x et y est placée dans le coin supérieur gauche [coordonnées (0,0)]. Tous les éléments du canevas sont définis relativement à ce point d'origine. Ainsi, l'arête supérieure gauche du carré colorié est définie à x pixels horizontalement à partir de la gauche et y pixels verticalement depuis le haut [coordonnées (x,y)].

Un rectangle plein est dessiné par la fonction `fillRect(x,y,width,height)`.

Exemple

Dessinons un rectangle plein dont l'arête supérieure gauche est située à 25 pixels du bord gauche et 25 pixels du bord haut. Notre rectangle a une largeur (`width`) de 150 pixels et une hauteur (`height`) de 100 pixels.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="zone" width="200px" height="200px" style="border: 1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>
<script type="text/javascript">
var canevas = document.getElementById( "zone" );
var contexte = canevas.getContext( "2d" );
contexte.fillRect(25,50,150,100);
</script>
</body>
</html>
```



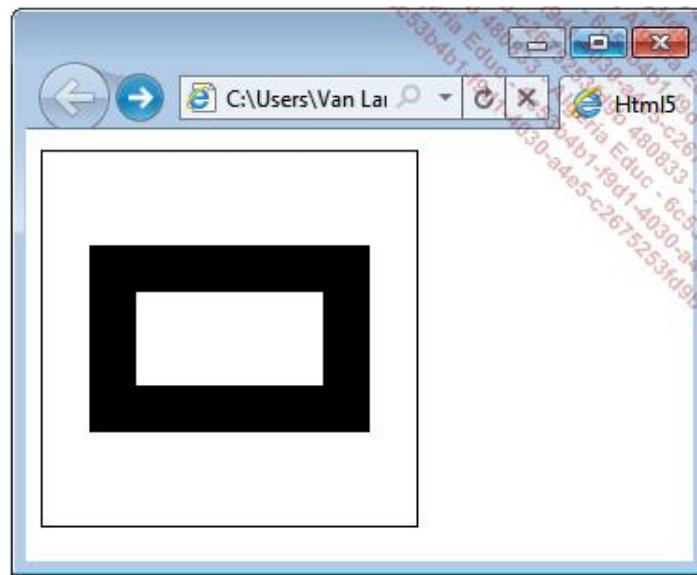
Par défaut, la couleur est le noir. Dès le point suivant, nous découvrirons comment colorier les dessins.

La fonction `clearRect(x,y,width,height)` efface la zone spécifiée et la rend entièrement transparente.

Exemple

Découpons une zone dans le rectangle défini ci-avant.

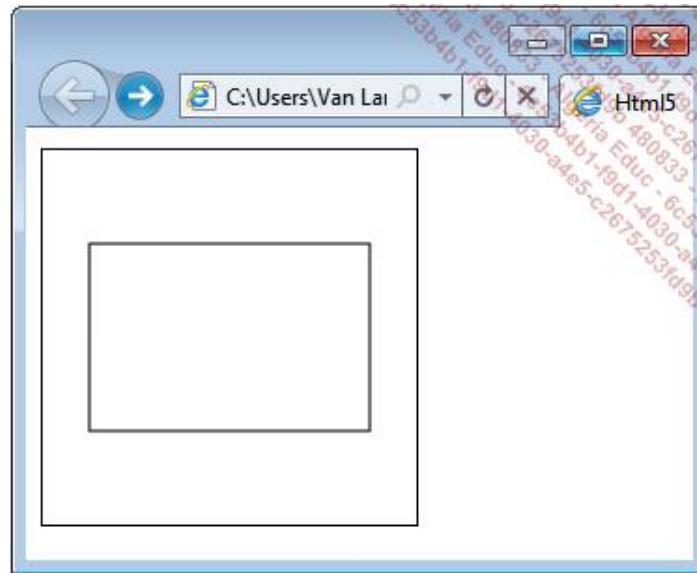
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="zone" width="200px" height="200px" style="border:
1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>
<script type="text/javascript">
var canevas = document.getElementById( "zone" );
var contexte = canevas.getContext( "2d" );
contexte.fillRect(25,50,150,100);
contexte.clearRect(50,75,100,50);
</script>
</body>
</html>
```



Et enfin, la fonction `strokeRect(x,y,width,height)` dessine un rectangle vide dont seule la bordure est visible.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="zone" width="200px" height="200px" style="border: 1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>
<script type="text/javascript">
var canevas = document.getElementById("zone");
var contexte = canevas.getContext( "2d" );
contexte.strokeRect(25,50,150,100);
</script>
</body>
</html>
```



Ajouter de la couleur

Pour ajouter de la couleur, vous disposez des propriétés

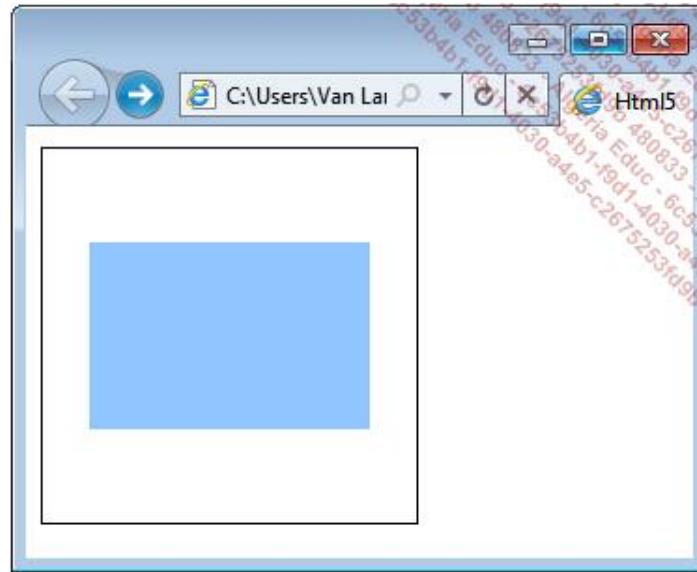
`fillStyle="couleur"` et `strokeStyle="couleur"`.

La notation des couleurs accepte toutes les spécifications de couleurs CSS et même celles des CSS3 (voir plus loin dans ce livre). Ainsi, la couleur orange peut se définir par :

```
- ctx.fillStyle = "orange";
- ctx.fillStyle = "#FFA500";
- ctx.fillStyle = "rgb(255,165,0)";
- ctx.fillStyle = "rgba(255,165,0,1);
```

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="dessin" width="200px" height="200px" style="border:
1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>
<script type="text/javascript">
var canvas = document.getElementById( "dessin" );
var contexte = canvas.getContext( "2d" );
contexte.fillStyle = "#99ccff";
contexte.fillRect(25,50,150,100);
</script>
</body>
</html>
```



Commentaire

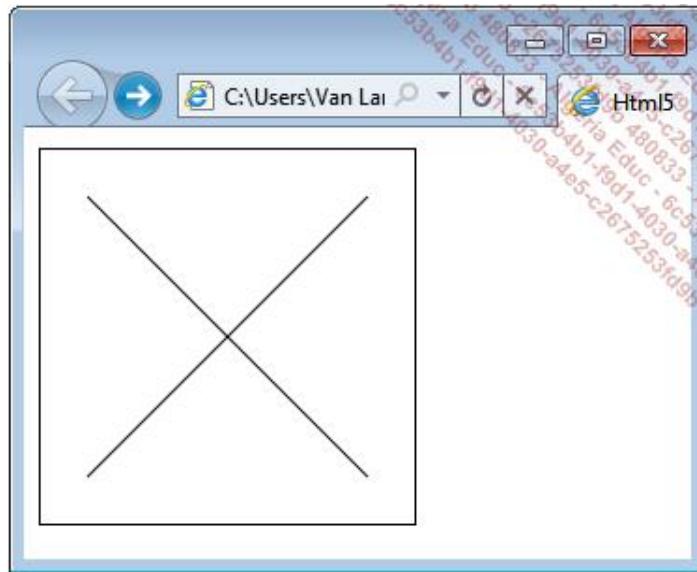
Si vous définissez la propriété `strokeStyle` et/ou `fillStyle`, la nouvelle valeur devient celle par défaut pour toutes les formes dessinées à partir de cet instant. Pour chaque forme qui doit être dans une couleur différente, vous devrez réassigner les propriétés `fillStyle` ou `strokeStyle`.

Tracer une ligne droite

La méthode `lineTo(x,y)` permet de dessiner des lignes droites. Les arguments `x` et `y` sont les coordonnées de l'extrémité finale de la ligne. Le point de départ dépend des chemins dessinés précédemment, puisque le dernier point d'un chemin est le point de départ du suivant, etc. Le point de départ peut également être changé à l'aide de la méthode `moveTo`.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="zone" width="200px" height="200px" style="border: 1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>
<script type="text/javascript">
var canevas = document.getElementById("zone");
var contexte = canevas.getContext("2d");
contexte.beginPath();
contexte.moveTo(25, 25);
contexte.lineTo(175, 175);
contexte.moveTo(175,25);
contexte.lineTo(25, 175);
contexte.stroke();
</script>
</body>
</html>
```



Détaillons le code :

```
contexte.moveTo(25, 25);
contexte.lineTo(175, 175);
```

Le script positionne le stylo virtuel à son point de départ (`moveTo(25, 25)`) et dessine la première diagonale (`lineTo(175, 175)`).

```
contexte.moveTo(175,25);
contexte.lineTo(25, 175);
```

Ensuite, le stylo est déplacé jusqu'à l'extrémité de la seconde diagonale (`moveTo(175, 25)`) et il trace celle-ci (`lineTo(25, 175)`).

```
contexte.stroke();
```

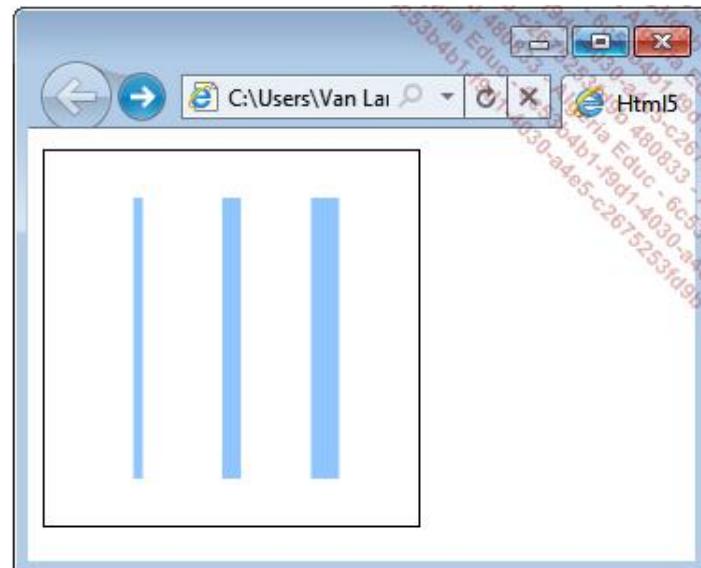
Avec les rectangles, toutes les opérations (dessin, couleur, etc.) étaient appliquées directement dans le canevas. Pour toutes les autres formes, c'est comme si vous utilisiez un stylo sans encre. Il faut en quelque sorte faire apparaître l'encre à la fin du dessin, d'où le code `stroke()`.

Plusieurs propriétés permettent de changer le style des lignes.

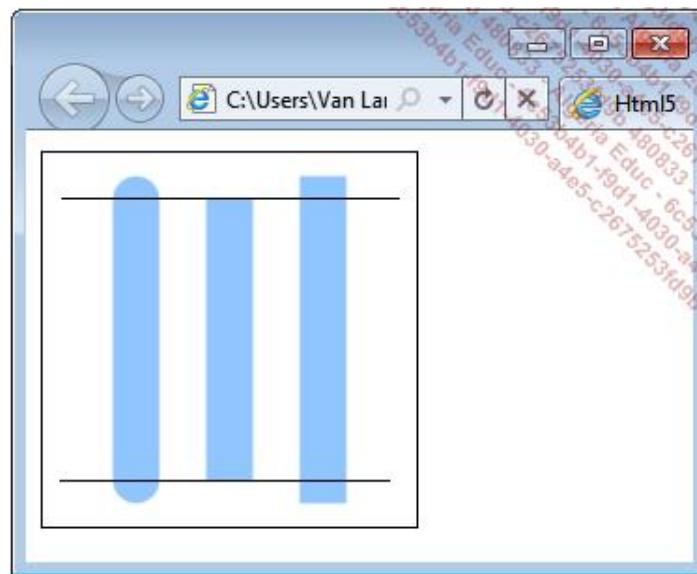
La propriété `lineWidth=valeur` définit l'épaisseur du trait de la ligne courante. La valeur par défaut est de 1 pixel.

Dessinons plusieurs lignes verticales d'épaisseurs différentes.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="zone" width="200px" height="200px" style="border: 1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>
<script type="text/javascript">
var canevas = document.getElementById("zone");
var contexte = canevas.getContext("2d");
contexte.strokeStyle = "#99ccff";
contexte.beginPath();
contexte.lineWidth = 5;
contexte.moveTo(50, 25);
contexte.lineTo(50, 175);
contexte.stroke();
contexte.beginPath();
contexte.moveTo(100, 25);
contexte.lineWidth = 10;
contexte.lineTo(100, 175);
contexte.stroke();
contexte.beginPath();
contexte.moveTo(150, 25);
contexte.lineWidth = 15;
contexte.lineTo(150, 175);
contexte.stroke();
</script>
</body>
</html>
```



La propriété `lineCap=valeur` détermine comment sont dessinées les extrémités de chaque ligne. Les trois valeurs possibles pour cette propriété sont : `butt`, `round` et `square`. Par défaut, celle-ci est définie à `butt`.



La capture d'écran et l'exemple suivant illustrent chacune des valeurs de la propriété `lineCap` :

- La valeur `round` ajoute un demi-cercle à l'extrémité de la ligne. Le rayon de celui-ci est de la moitié de l'épaisseur de la ligne.
- La valeur `butt` (défaut) se termine exactement à l'endroit défini par le code.
- La valeur `square` ajoute une boîte de la largeur de la ligne et haute de la moitié de l'épaisseur de celle-ci.

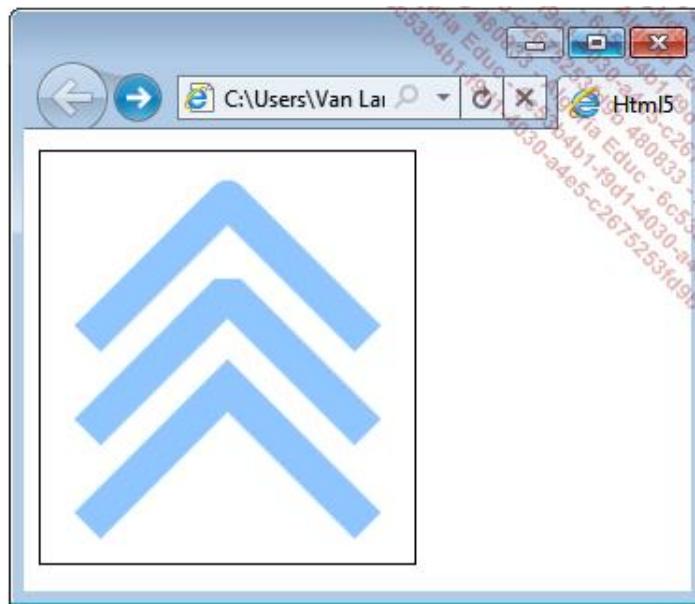
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="zone" width="200px" height="200px" style="border: 1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>
<script type="text/javascript">
var canevas = document.getElementById("zone");
var contexte = canevas.getContext("2d");
contexte.strokeStyle = "#99ccff";
contexte.lineWidth = 25;
contexte.beginPath();
contexte.moveTo(50, 25);
contexte.lineCap="round";
contexte.lineTo(50, 175);
contexte.stroke();
contexte.beginPath();
contexte.lineCap="butt";
contexte.moveTo(100,25);
contexte.lineTo(100, 175);
contexte.stroke();
contexte.beginPath();
contexte.moveTo(150,25);
contexte.lineCap="square";
contexte.lineTo(150, 175);
contexte.stroke();
</script>
</body>
```

```
</html>
```

La propriété `lineJoin=valeur` définit la façon dont deux lignes qui se rejoignent sont reliées entre-elles. Les trois valeurs possibles pour cette propriété sont : `round`, `bevel` et `miter`. Par défaut, celle-ci est définie à `miter`.

La capture d'écran et l'exemple suivant illustrent chacune des valeurs de la propriété `lineJoin` :

- La valeur `round` arrondit les coins de la forme. Le rayon de l'arrondi est égal à l'épaisseur de la ligne.
- La valeur `bevel` n'effectue aucune opération.
- La valeur `miter` (défaut) fait prolonger les lignes pour qu'elles se rejoignent en un seul point.



```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="zone" width="200px" height="220px" style="border: 1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>
<script type="text/javascript">
var canevas = document.getElementById("zone");
var contexte = canevas.getContext("2d");
contexte.strokeStyle = "#99ccff";
contexte.lineWidth = 20;
contexte.beginPath();
contexte.lineJoin="round";
contexte.moveTo(25,100);
contexte.lineTo(100,25);
contexte.lineTo(175,100);
contexte.stroke();
contexte.beginPath();
contexte.lineJoin="bevel";
contexte.moveTo(25,150);
contexte.lineTo(100,75);
contexte.lineTo(175,150);
contexte.stroke();
contexte.beginPath();
contexte.lineJoin="miter";
contexte.moveTo(25,200);
```

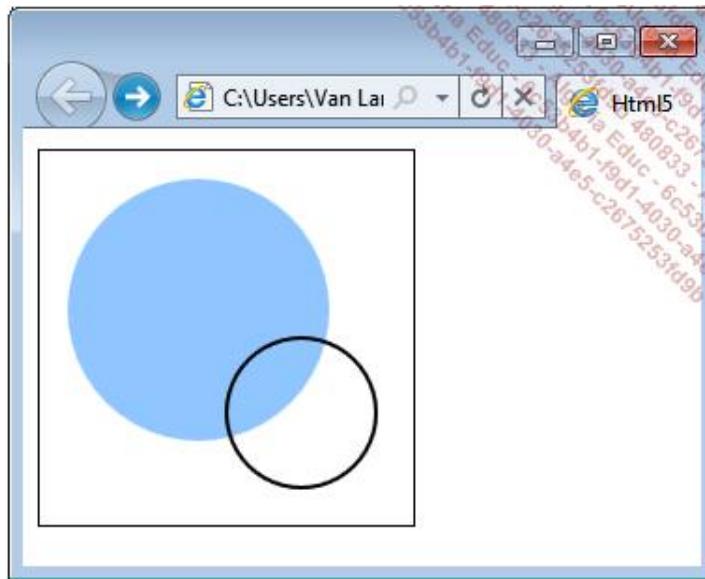
```
contexte.lineTo(100,125);
contexte.lineTo(175,200);
contexte.stroke();
</script>
</body>
</html>
```

Dessiner des formes géométriques

Pour dessiner des arcs et des cercles, il faut utiliser la méthode `arc(x, y, rayon, angleDépart, angleFin, sensInverse)`.

Cette méthode prend cinq paramètres : `x` et `y` sont les coordonnées du centre du cercle et `rayon` est bien entendu son rayon. Les paramètres `angleDépart` et `angleFin` définissent les points de départ et d'arrivée de l'arc en radians. Ceux-ci sont mesurés à partir de l'axe `x`. Le paramètre `sensInverse` est une valeur booléenne qui, lorsqu'elle est mise à `true`, dessine l'arc dans le sens inverse des aiguilles d'une montre. Autrement (`false`), le dessin se fait dans le sens des aiguilles d'une montre.

Notons que les angles dans la fonction `arc` sont exprimés en radians et non en degrés.



```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="zone" width="200px" height="200px" style="border: 1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>
<script type="text/javascript">
var canevas = document.getElementById("zone");
var contexte = canevas.getContext("2d");
contexte.fillStyle = "#99ccff";
contexte.lineWidth = 2;
contexte.beginPath();
contexte.arc(85, 85, 70, 0, 2*Math.PI, true);
contexte.fill();
contexte.beginPath();
contexte.strokeStyle = "black";
contexte.arc(140, 140, 40, 0, 2*Math.PI, true);
contexte.stroke();
</script>
</body>
</html>
```

Pour les formes complexes, les courbes de Bézier sont également disponibles dans les fonctions de `canevas`. Les courbes de Bézier dessinent les courbes à partir de formules mathématiques. Leur apprentissage est aussi complexe que les courbes dessinées et réclame une formation en infographie poussée. Elles sont reprises ici pour information.

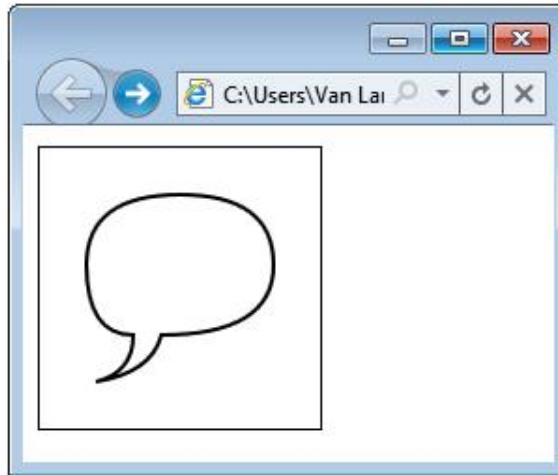
Deux fonctions sont disponibles : `quadraticCurveTo` et `bezierCurveTo`. Une courbe de Bézier quadratique a un point de départ, un point d'arrivée et un point unique de contrôle. Une courbe de Bézier cubique dispose par contre de deux points de contrôle.

L'utilisation de courbes de Bézier quadratiques et cubiques peut s'avérer relativement ardue car, contrairement aux logiciels de dessin vectoriel comme Adobe Illustrator, on ne dispose pas d'un retour visuel direct sur ce que l'on est en train d'élaborer. Cela rend le dessin de formes complexes assez difficile.

Voici quelques exemples simples.

Le premier exploite la fonction quadraticCurveTo.

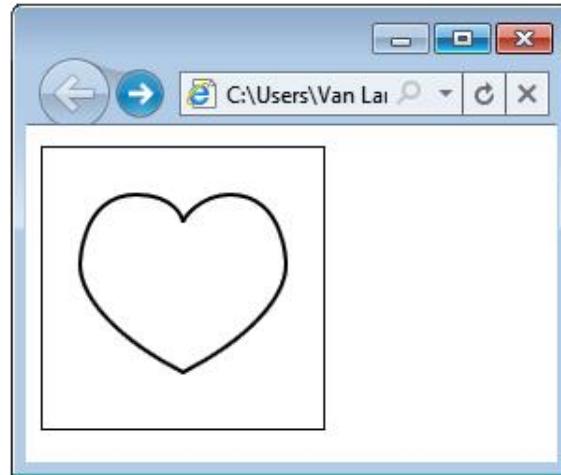
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="zone" width="150px" height="150px" style="border: 1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>
<script type="text/javascript">
var canevas = document.getElementById("zone");
var contexte = canevas.getContext("2d");
contexte.beginPath();
contexte.lineWidth = 2;
contexte.moveTo(75,25);
contexte.quadraticCurveTo(25,25,25,62.5);
contexte.quadraticCurveTo(25,100,50,100);
contexte.quadraticCurveTo(50,120,30,125);
contexte.quadraticCurveTo(60,120,65,100);
contexte.quadraticCurveTo(125,100,125,62.5);
contexte.quadraticCurveTo(125,25,75,25);
contexte.stroke();
</script>
</body>
</html>
```



Le second exemple reprend bezierCurveTo.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="zone" width="150px" height="150px" style="border: 1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>
<script type="text/javascript">
var canevas = document.getElementById("zone");
```

```
var contexte = canevas.getContext("2d");
contexte.beginPath();
contexte.lineWidth = 2;
contexte.moveTo(75,40);
contexte.bezierCurveTo(75,37,70,25,50,25);
contexte.bezierCurveTo(20,25,20,62.5,20,62.5);
contexte.bezierCurveTo(20,80,40,102,75,120);
contexte.bezierCurveTo(110,102,130,80,130,62.5);
contexte.bezierCurveTo(130,62.5,130,25,100,25);
contexte.bezierCurveTo(85,25,75,37,75,40);
contexte.stroke();
</script>
</body>
</html>
```



Importer des images

L'importation des images se réalise en deux étapes :

- Il faut d'abord créer une image comme objet JavaScript. Il n'est pas possible d'inclure une image simplement par l'attribut Html `src`.
- La fonction `drawImage` est utilisée pour dessiner l'image dans le canevas. Il est important de veiller à ce que l'image soit bien chargée avant de faire appel à la fonction `drawImage`.

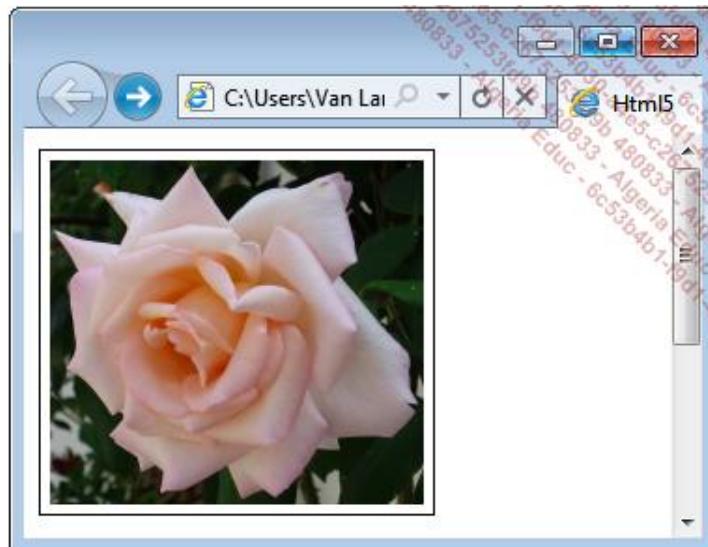
La syntaxe de cette fonction `drawImage(image, x, y)` où `image` est une référence à l'image et `x, y` les coordonnées du placement de l'image dans le canevas.

Les images peuvent être au format GIF, JPEG ou PNG.

Exemple 1

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<script type="text/javascript">
function dessin() {
var canevas = document.getElementById("zone");
var contexte = canevas.getContext("2d");
var img = new Image();
img.src = 'rose.png';
contexte.drawImage(img,5,5);
}
</script>
<body onload="dessin()">
<canvas id="zone" width="210px" height="194px" style="border: 1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>

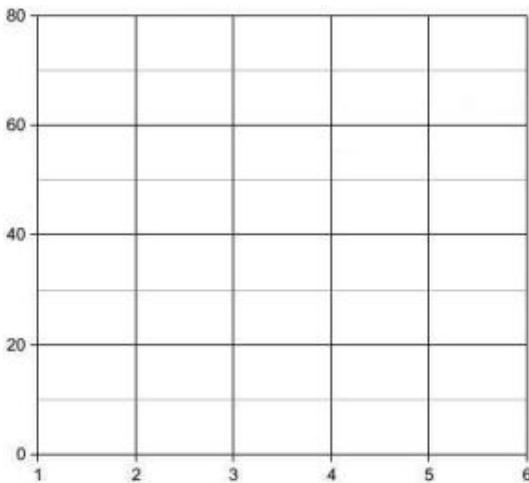
</body>
</html>
```



Exemple 2

Il est bien entendu possible de superposer des lignes sur une image.

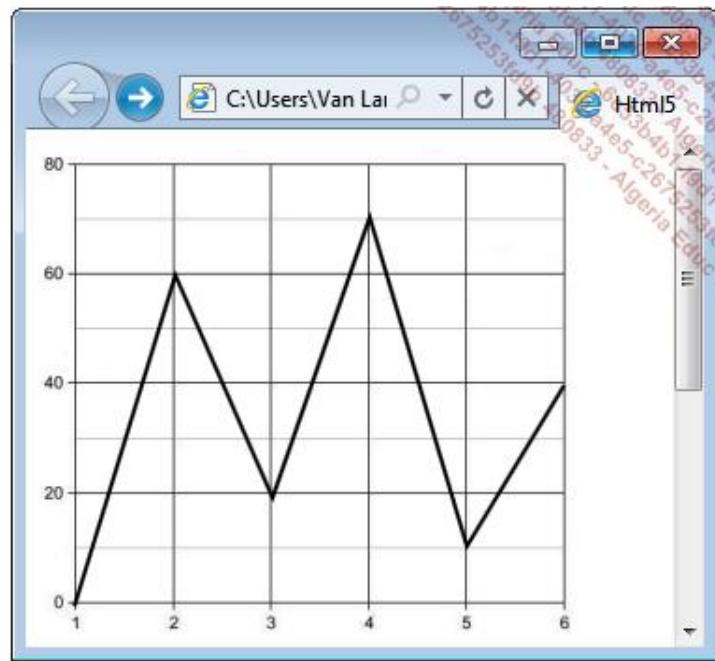
Soit une image de fond de graphique :



À cette image, incluse dans le canevas, nous allons ajouter une ligne de graphique.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<script type="text/javascript">
function dessin() {
var canevas = document.getElementById("zone");
var contexte = canevas.getContext("2d");
var img = new Image();
img.src = 'graphe.png';
contexte.drawImage(img,0,0);
contexte.beginPath();
contexte.lineWidth = 2;
contexte.moveTo(18,244);
contexte.lineTo(72,67);
contexte.lineTo(124,186);
contexte.lineTo(176,36);
contexte.lineTo(228,212);
contexte.lineTo(280,126);
contexte.stroke();
}
</script>
<body onload="dessin()">
<canvas id="zone" width="285px" height="260px">
Votre navigateur ne supporte pas la balise canvas
</canvas>

</body>
</html>
```



Ajouter du texte

Pour ajouter du texte dans le canevas, le concepteur dispose des méthodes `fillText("texte", x, y)` et `strokeText("texte", x, y)` où `texte` est le texte à inclure, `x` la coordonnée horizontale du début du texte et `y` la coordonnée verticale du début du texte.

Ces méthodes comportent trois propriétés :

La propriété `font` définit la police de caractères à utiliser lors du dessin. La syntaxe est identique à celle de la propriété de style CSS `font`. La valeur par défaut est `10px sans-serif`.

Exemple

```
contexte.font = "20pt Arial";
```

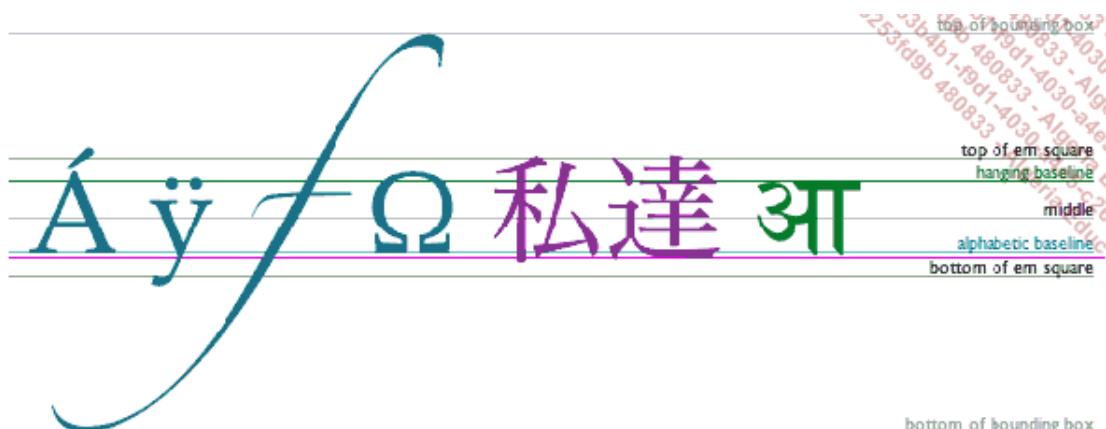
La propriété `textAlign` détermine l'alignement du texte. Les valeurs possibles sont :

- `left` pour un alignement à gauche.
- `right` pour un alignement à droite.
- `center` pour un alignement centré.
- `start` (défaut) pour l'alignement du début de ligne pour l'écriture de gauche à droite.
- `end` pour l'alignement à la fin de ligne pour les écritures de droite à gauche.

Exemple

```
contexte.textAlign = "left";
```

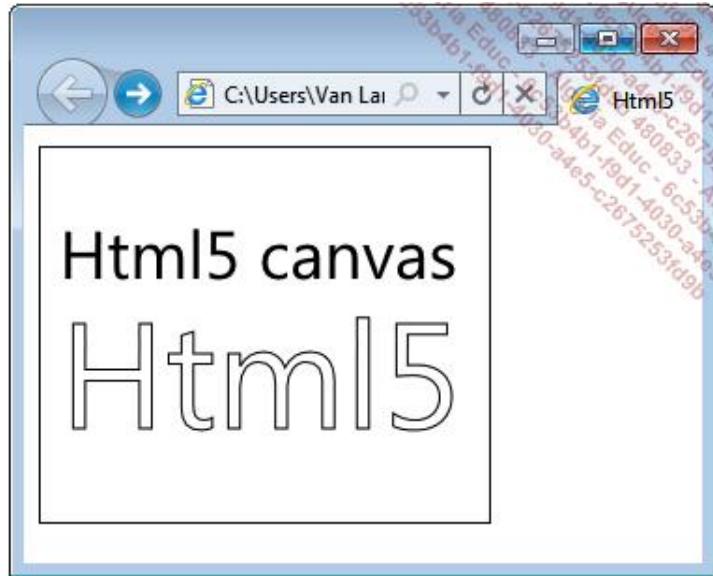
Et enfin, la propriété `textBaseline` définit la ligne de référence de l'écriture du texte. Les valeurs possibles, illustrées par l'image suivante, sont `top`, `middle`, `alphabetic` (défaut) et `bottom`.



Exemple d'ajout de texte :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="dessin" width="240px" height="200px" style="border:
1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>
<script type="text/javascript">
```

```
var canvas = document.getElementById( "dessin" );
var contexte = canvas.getContext( "2d" );
contexte.textBaseline = "alphabetic";
contexte.textAlign = "left";
contexte.font = '36px "Segoe UI" bold';
contexte.fillText("Html5 canvas", 10, 70);
contexte.font = "80px \"Segoe UI\" bold";
contexte.strokeText("Html5", 10, 150);
</script>
</body>
</html>
```



Ajouter de l'ombre

Les effets d'ombre sont ajoutés par les propriétés `shadowOffsetX`, `shadowOffsetY`, `shadowBlur` et `shadowColor`.

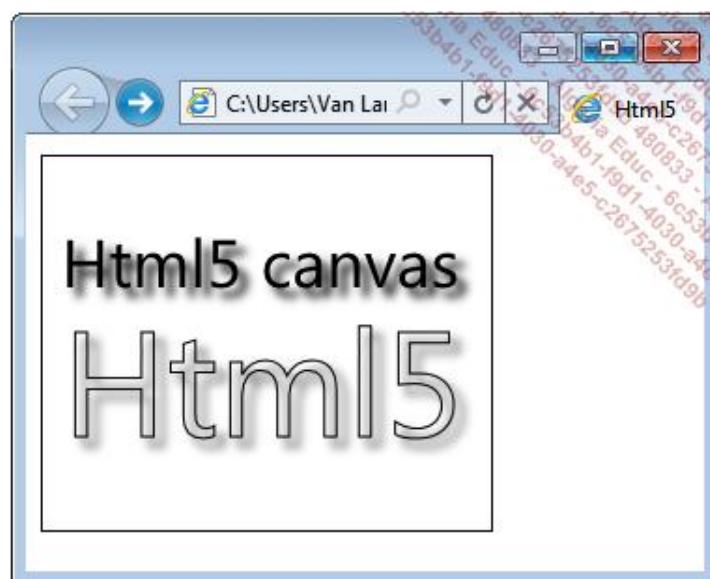
Les propriétés `shadowOffsetX=valeur` et `shadowOffsetY=valeur` indiquent le déport de l'ombre par rapport à l'axe horizontal et vertical. Des valeurs négatives peuvent être reprises pour une ombre dirigée vers le haut et vers la gauche. Les valeurs positives induisent un effet d'ombrage vers le bas et vers la droite. La valeur par défaut est 0 pour les deux propriétés.

La propriété `shadowBlur=valeur` détermine l'effet de dispersion de l'ombre. La valeur par défaut est 0.

La propriété `shadowColor=couleur` indique la couleur de l'effet d'ombrage. Cette couleur est indiquée selon la notation des feuilles de style CSS. Par défaut, la couleur est le noir (`black`).

La tentation est grande d'ajouter un effet d'ombre au texte de l'exemple précédent :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<canvas id="dessin" width="240px" height="200px" style="border:
1px solid black">
Votre navigateur ne supporte pas la balise canvas
</canvas>
<script type="text/javascript">
var canvas = document.getElementById( "dessin" );
var contexte = canvas.getContext( "2d" );
contexte.textBaseline = "alphabetic";
contexte.textAlign = "left";
contexte.font = '36px "Segoe UI" bold';
contexte.shadowOffsetX = 5;
contexte.shadowOffsetY = 5;
contexte.shadowBlur = 5;
contexte.shadowColor = "black";
contexte.fillText("Html5 canvas", 10, 70);
contexte.font = "80px \"Segoe UI\" bold";
contexte.strokeText("Html5", 10, 150);
</script>
</body>
</html>
```



Transformations

Jusqu'à présent le dessin initié par la balise `<canvas>` était statique. Les différentes transformations rendent les éléments dynamiques.

Translation

La méthode `translate(x,y)` déplace un élément du canevas de son origine vers un nouveau point dont les coordonnées sont fournies par `x` et `y`.

Rotation

La méthode `rotate(angle)` est utilisée pour faire tourner un élément autour de son point d'origine. Le paramètre `angle` est exprimé en radians et effectue la rotation dans le sens des aiguilles d'une montre.

Notons que le point central de la rotation est toujours le point d'origine. Pour modifier ce dernier; il est nécessaire de déplacer l'élément à l'aide de la méthode `translate`.

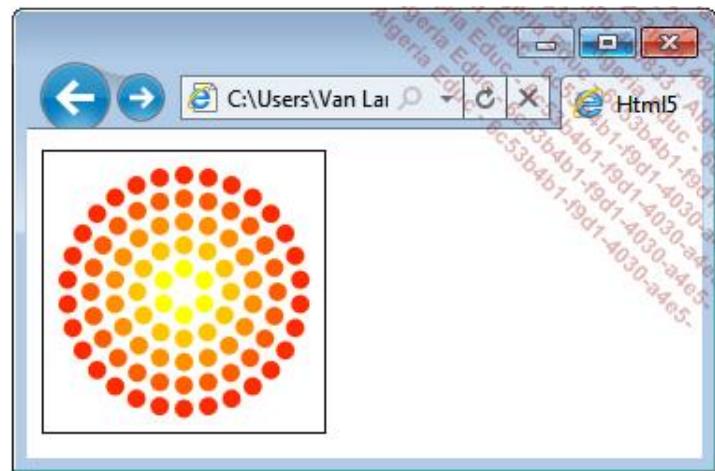
Mise à l'échelle

La mise à l'échelle permet d'agrandir ou diminuer la taille des éléments dans le canevas. On utilise la méthode `scale(x,y)` où `x` est le facteur d'échelle dans la direction horizontale et `y` celui de la direction verticale. Les deux paramètres doivent être des nombres positifs. Les valeurs plus petites que 1.0 réduisent la taille de l'unité, et les valeurs plus grandes que 1.0 augmentent la taille de l'unité. Positionner le facteur d'échelle à exactement 1.0 n'a aucun effet sur la taille de l'unité.

Ainsi, comme l'unité de base est le pixel, si l'on applique un facteur d'échelle de 0.5, l'unité résultante devient 0.5 pixels et les formes dessinées diminuent à la moitié de leur taille. D'une manière similaire, en positionnant le facteur d'échelle à 2.0 on augmente la taille de l'unité et celle-ci devient alors égale à deux pixels. Le résultat est que les formes dessinées augmentent de deux fois leur taille.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<script type="text/javascript">
function draw() {
var ctx = document.getElementById('canvas').getContext('2d');
ctx.translate(75,75);
for (i=1;i<6;i++){
ctx.save();
ctx.fillStyle = 'rgb(255,'+Math.abs(306-51*i)+',0)';
for (j=0;j<i*6;j++){
ctx.rotate(Math.PI*2/(i*6));
ctx.beginPath();
ctx.arc(0,i*12.5,5,0,Math.PI*2,true);
ctx.fill();
}
ctx.restore();
}
}
</script>
<style type="text/css">
canvas { border: 1px solid black; }
</style>
</head>
<body onload="draw();">
<canvas id="canvas" width="150" height="150"></canvas>
</body>
</html>
```



Préambule

Le HTML5 comporte de façon native de nombreuses interfaces de programmation (*Application Programming Interface* ou API). Citons :

- La géolocalisation, qui permet de localiser l'utilisateur par ses coordonnées de longitude et latitude.
- Des super cookies, avec *Web Storage* qui permettra un stockage plus important de données dans le navigateur.
- L'utilisation des applications Web hors connexion après la mise en cache des ressources nécessaires.
- Les *Workers*, qui peuvent exécuter des tâches de fond en parallèle du programme JavaScript principal dans un environnement totalement séparé de la page.
- Les *Websockets*, qui permettent d'établir une communication bidirectionnelle asynchrone entre le navigateur et le serveur.
- Le glisser-déposer (*drag/drop*) en natif dans le navigateur.
- L'attribut `ContentEditable`, qui permet l'édition en ligne du contenu d'un élément. Il fait apparaître un éditeur WYSIWYG basique qui permet donc d'éditer directement le contenu dans la page. Toutes les modifications apportées par l'utilisateur pourront être traitées à la fin de l'édition.
- Etc.

La plupart dépassent largement le cadre de cet ouvrage. Intéressons-nous néanmoins à la fonction de géolocalisation.

Concept et utilité

La géolocalisation (*geolocation*) est un procédé permettant de positionner un objet ou une personne sur une carte ou un plan. Au niveau du Web, la géolocalisation permet à un site de connaître les coordonnées géographiques (longitude et latitude) d'un utilisateur.

Ce concept n'est pas nouveau sur la toile. La nouveauté réside dans le fait que cette fonctionnalité est incluse de façon native dans le Html5 soit sans bibliothèque ou API supplémentaire.

Les applications de la géolocalisation peuvent être multiples. Un site commercial peut ainsi proposer des revendeurs de son produit, situés à proximité immédiate de l'endroit où réside l'utilisateur. Un réseau social peut vous indiquer des affiliés correspondant à votre profil qui résident près de chez vous. Un moteur de recherche peut également vous proposer des hôtels, restaurants ou lieux de loisirs dans les environs d'une ville étape. En fonction de votre lieu de résidence, un site de vente en ligne international pourra afficher les prix dans votre devise ou afficher les dates selon votre modèle habituel (jj/mm/aaaa ou mm/jj/aaaa).

Le fonctionnement

D'où proviennent ces données qui permettent de vous localiser ?

Tout d'abord, de l'adresse IP de l'utilisateur. C'est une procédure simple de connaître l'adresse IP de l'utilisateur. Il suffit alors de consulter des registres d'attribution pour connaître votre adresse physique. La précision de la géolocalisation est parfois sujette à surprise car, dans certains cas, c'est l'adresse de votre fournisseur d'accès qui sera reprise.

Ensuite, de votre réseau wifi. La géolocalisation de votre réseau wifi s'obtient par triangulation par rapport à des points d'accès ou bornes wifi aux alentours. Rappelez-vous qu'en mai 2010, la société Google a révélé que ses véhicules chargés de prendre des photos pour son application *Google Street View* engrangeaient également des informations relatives aux réseaux wifi rencontrés. La géolocalisation ainsi obtenue est assez précise, surtout dans les zones urbaines.

Et enfin, pour certains téléphones cellulaires de la dernière génération (iPhone), par les coordonnées GPS. La précision de la géolocalisation est alors quasi absolue.

Les navigateurs

La plupart des navigateurs reconnaissent cette fonctionnalité de géolocalisation en Html5. Citons Chrome5+, Firefox 3.6+, Opera10.6 et Safari 5+. Internet Explorer 8 et 9 ne reconnaissent pas la géolocalisation.

Il faut cependant signaler que l'état d'avancement de l'inclusion de cette API varie grandement d'un navigateur à l'autre.

Protection de la vie privée

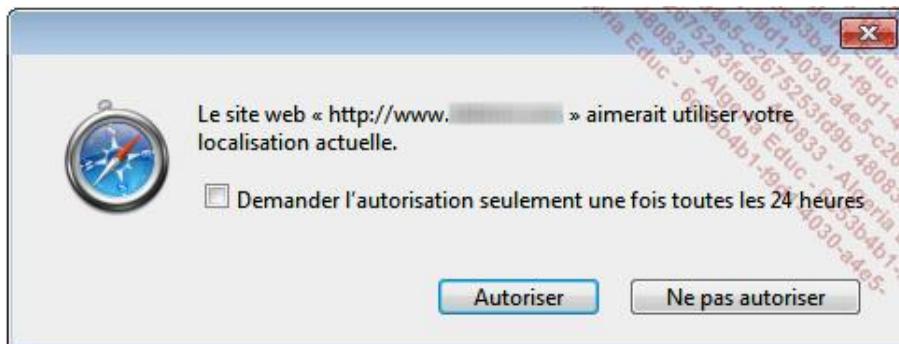
Cette géolocalisation peut se révéler assez indiscrete et tous les utilisateurs ne souhaiteront pas partager leurs données de géolocalisation. Le W3C émet des règles d'utilisation très strictes.

La géolocalisation doit être une démarche volontaire de l'utilisateur. Avant toute procédure de géolocalisation de la part d'un site, une fenêtre popup ou un bandeau doit apparaître dans le navigateur demandant l'autorisation expresse de l'utilisateur.

Capture d'écran de la demande de partage de la localisation dans Firefox 4 :



La même fenêtre d'autorisation dans Safari :



Quand un navigateur procède à une géolocalisation, les informations échangées se réalisent par l'intermédiaire d'une connexion chiffrée.

À aucun moment le nom et l'adresse du site Web visité ne doivent être partagés.

L'identifiant client aléatoire attribué pour une procédure de géolocalisation doit expirer après un délai de deux semaines.

Faisons le vœu que ces procédures soient scrupuleusement respectées par tous les intervenants de la toile...

Longitude et latitude

Cette bibliothèque (API) de géolocalisation fait largement appel au JavaScript.

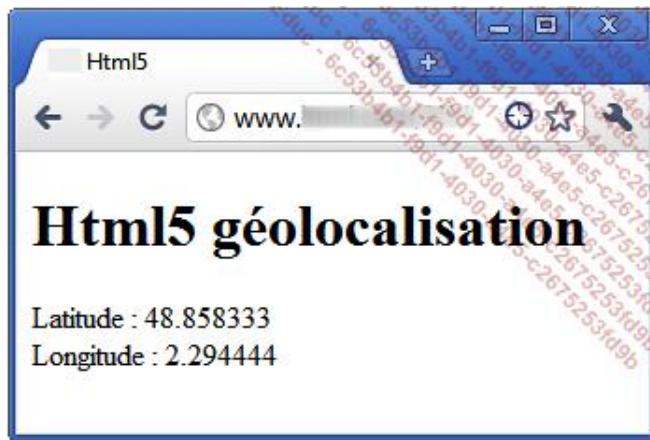
Le JavaScript `position.coords` retourne les coordonnées mais aussi bien d'autres valeurs :

- `position.coords.longitude` renvoie la longitude de la position.
- `position.coords.latitude` renvoie la latitude de la position.
- `position.coords.altitude` retourne l'altitude de la position.
- `position.coords.accuracy` indique la précision des coordonnées.
- `position.coords.altitudeAccuracy` fournit la précision de l'altitude.
- `position.coords.heading` donne la position en degré par rapport au nord.
- `position.coords.speed` qui correspond à la vitesse de l'utilisateur par rapport à sa dernière position.

Ces valeurs ne sont pas encore toutes présentes dans les navigateurs mais donnent une idée du potentiel de cette application.

Élaborons un exemple qui retourne la longitude et la latitude :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<script type="text/javascript">
if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition(function(position) {
var latitude = position.coords.latitude;
var longitude = position.coords.longitude;
document.getElementById("latitude").innerHTML = latitude;
document.getElementById("longitude").innerHTML = longitude;
});
}
else {
alert("Votre navigateur ne supporte pas la géolocalisation");
}
</script>
</head>
<body>
<h1>Html5 géolocalisation</h1>
Latitude : <span id="latitude">Loading...</span><br>
Longitude : <span id="longitude">Loading...</span>
</body>
</html>
```



Pour préserver les données privées de l'auteur, les coordonnées de longitude et de latitude correspondent à celles de la tour Eiffel.

Le script est disponible dans l'espace de téléchargement réservé à cet ouvrage.

Pour illustrer l'aspect encore expérimental de cette API, le script fonctionne en local sous Firefox 3.6+ et Opera 10.6+. Pour Chrome, la page doit être en ligne et Safari (sous Windows) bloque sur l'écran d'accueil.

Explications du script :

```
<script type="text/javascript">
...
</script>
```

Les balises qui introduisent un script JavaScript :

```
if (navigator.geolocation){
...
}
```

Le script teste si la fonction de géolocalisation (`geolocation`) est prise en charge par le navigateur (`navigator`).

```
var latitude = position.coords.latitude;
var longitude = position.coords.longitude;
```

Si c'est bien le cas, la variable `latitude` est obtenue par `position.coords.latitude` et la variable `longitude` par `position.coords.longitude`.

```
document.getElementById("latitude").innerHTML = latitude;
document.getElementById("longitude").innerHTML = longitude;
});
```

On reporte dans le Html (`innerHTML`) la variable `latitude` à l'intérieur dans la balise `` dotée de l'identifiant Id `id="latitude"`. On procède de même pour la `longitude`.

```
Else {
alert("Votre navigateur ne supporte pas la géolocalisation");
}
```

Si le navigateur n'accepte pas la géolocalisation, une fenêtre d'alerte apparaît.

Message de la page Web



Votre navigateur ne supporte pas la géolocalisation

OK

Localisation sur une carte

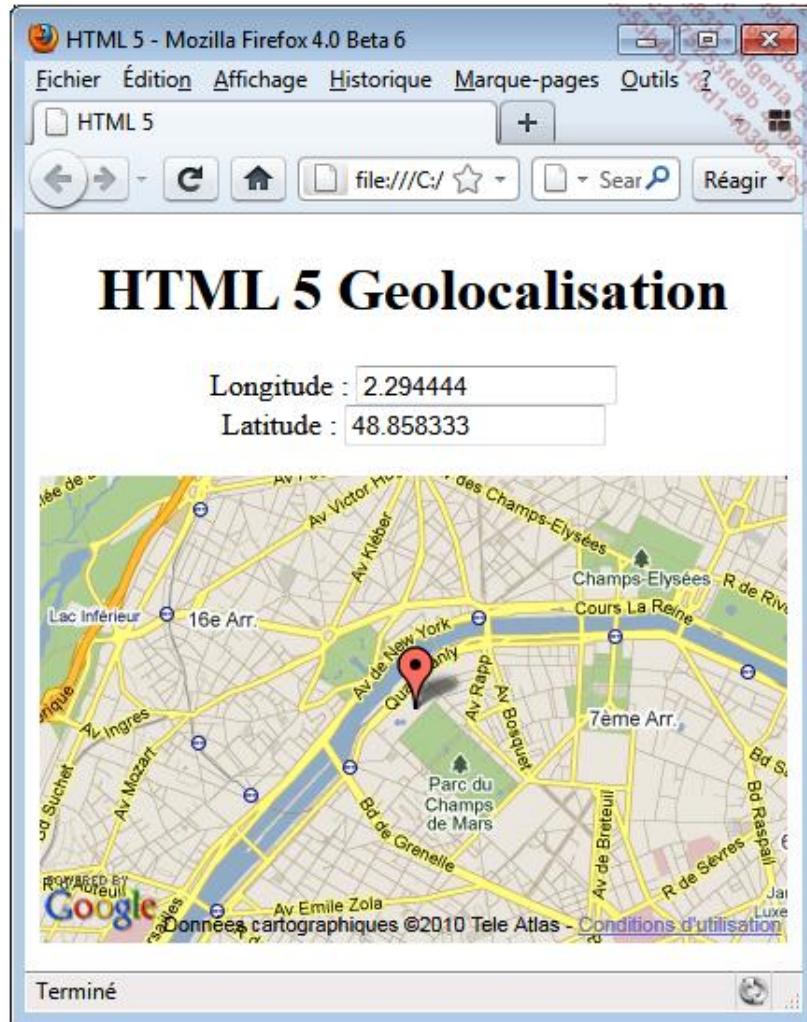
Une fois la longitude et la latitude connues, la tentation est grande de reporter cette position sur une carte en faisant appel à Google Maps.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<script
src="http://maps.google.com/maps?file=api&v=2&sensor=false
&key=ABQIAAAA8JXb0YDVa4ot0LnM95w50BSeC_rwpfX9fQb-
nbMGMDH8BB4BVRTjxWS14T5WLZf7TpXaaAtk_SIb-Q"
type="text/javascript">
</script>
<script type="text/javascript">
var map;
var geocoder;
function init() {
map = new Gmap2(document.getElementById("carte"));
map.setCenter(new GLatLng(34, 0), 1);
geocoder = new GclientGeocoder();
}
function sur_carte(response) {
map.clearOverlays();
if (!response || response.Status.code != 200) {
alert("Désolé, nous ne pouvons géolocaliser votre adresse.");
}
else {
place = response.Placemark[0];
point = new GLatLng(place.Point.coordinates[1],
place.Point.coordinates[0]);
marker = new Gmarker(point);
map.setCenter(point, 13);
map.addOverlay(marker);
}
}
function geolocalisation() {
var adresse = document.form.latitude.value + "," +
document.form.longitude.value
geocoder.getLocations(adresse, sur_carte);
}
if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition(function(position) {
latitude = position.coords.latitude;
longitude = position.coords.longitude;
document.form.longitude.value = longitude;
document.form.latitude.value = latitude;
geolocalisation();
});
}
else {
alert("Votre navigateur ne supporte pas la géolocalisation");
}
</script>
</head>
<body onload="init()">
<div style="text-align:center; width: 400px;">
<h1>HTML 5 Geolocalisation</h1>
<form name="form" action="">
<p>
Longitude : <input type="text" name="longitude" value=""
size="20"><br>
Latitude : <input type="text" name="latitude" value="" size="20">
</p>
</form>
```

```

<div id="carte" style="width: 400px; height: 250px;"></div>
</div>
</body>
</html>

```



- Pour préserver les données privées de l'auteur, les coordonnées de longitude et de latitude correspondent à celles de la tour Eiffel.

La page Html et le script sont disponibles dans l'espace de téléchargement.

Quelques explications :

La géolocalisation s'effectue comme au point précédent par le script suivant :

```

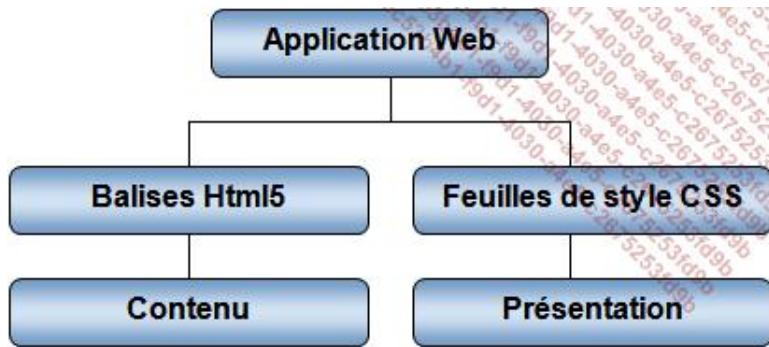
if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(function(position) {
    latitude = position.coords.latitude;
    longitude = position.coords.longitude;
    document.form.longitude.value = longitude;
    document.form.latitude.value = latitude;
    geolocalisation();
  });
}
else {
  alert("Votre navigateur ne supporte pas la géolocalisation");
}

```

Les coordonnées trouvées sont envoyées à la fonction geolocalisation. Cette dernière fait à son tour appel à la fonction sur_carte qui exploite la bibliothèque (API) Google Maps Api. Pour plus d'informations sur l'insertion d'une carte Google maps dans un site, consultez l'adresse <http://code.google.com/intl/fr-FR/apis/maps/>.

Concept des feuilles de style

Les feuilles de style sont des ajouts de code au langage Html qui vont prendre en charge la présentation du document ou de l'application Web.



Le concept de feuilles de style repose sur le principe de la séparation du contenu et de la présentation dans l'élaboration d'applications Html.

Ainsi un même contenu pourrait être utilisé, selon la feuille de style adoptée, pour un affichage sur des médias aussi divers qu'un écran traditionnel, l'écran d'un ordinateur de poche ou d'un mobile, des feuilles de papier imprimées, une barrette braille, etc.

C'est ainsi tout le domaine de la présentation qui est pris en charge par les feuilles de style, le rôle du Html se limitant alors à la structure et à l'encodage de l'information brute.

Utilité des feuilles de style

Les feuilles de style représentent un outil puissant dont les implications dans la conception des applications Web sont multiples.

Un passage obligé en Html5

Le Html5 va au bout du concept de la séparation du contenu et de la présentation. Presque tous les éléments de présentation des balises ou des attributs présents dans le Html 4.0 ont ainsi disparu. Un document Html5 sans feuille de styles pour en assurer la présentation n'est alors plus qu'une morne structure du contenu textuel. Citons, par exemple, l'absence en Html5 de l'attribut `border` qui assurait la présence d'une bordure à la balise de tableau `<table>`, rendant ainsi la représentation des tableaux problématiques en pur Html.

Les feuilles de style vont beaucoup plus loin dans le domaine de la présentation que les anciennes balises du Html. De par leur richesse et leur variété, elles ont profondément modifié l'aspect visuel des applications en décuplant la créativité des webgraphistes.

Une simplification du code

Il semble déjà bien lointain, le temps où le code source des pages Web était un imbroglio incompréhensible de tableaux imbriqués et d'astuces en tout genre pour assurer une présentation agréable.

Les feuilles de style ont permis d'alléger considérablement le code source en le rendant plus lisible et plus accessible.

Une maintenance de site facilitée

De par l'utilisation croissante du Web comme source d'information, la dimension des sites Web a considérablement augmenté en termes de nombre de pages par site. Il n'est plus rare de rencontrer des sites de plusieurs centaines de pages. Une mise à jour graphique peut ainsi constituer un travail gigantesque.

Plutôt que de devoir reprendre et retravailler, une par une, chacune des pages du site, les feuilles de style permettent, par la seule modification d'un unique fichier, de modifier la présentation graphique de l'ensemble d'un site, tout en lui assurant une cohérence graphique.

Une voie vers l'accessibilité

Depuis quelques années, l'accent a été mis sur l'accessibilité des sites Web pour les personnes présentant des déficiences visuelles. Beaucoup de concepteurs de sites ont pris conscience de l'importance du Web comme media d'informations irremplaçable pour ces personnes comportant un handicap de la vue. Pour rendre un site accessible, soit procurer aux malvoyants un accès immédiat à la même information et au même volume d'information, l'usage des feuilles de style s'avère être un outil essentiel.

Sans entrer dans les détails, les feuilles de style offrent, par exemple, la possibilité d'afficher une page de texte avec des caractères plus grands et donc plus lisibles.

Bref historique

Lorsque, début 1991, Tim Berners-Lee inventa le Web, le langage Html était principalement conçu pour faciliter l'échange d'informations entre scientifiques. Les premières pages Web étaient pour le moins austères avec leur fond de page gris, un texte de couleur noire et structuré uniquement avec des titres et des listes. Il fallut même attendre quelques années pour voir apparaître les premières images... en 256 couleurs ! On était loin de nos applications Web actuelles.

Assez rapidement cependant, dès 1994, devant les relatives faiblesses du Html en matière de présentation, l'idée d'ajouter au Html des styles, comme dans les logiciels de traitement de texte, fit son apparition.

Lorsqu'en 1995 le W3C (pour *World Wide Web Consortium*) commença à fonctionner, une de ses premières réalisations concrètes fut la publication en novembre 1995 d'une première ébauche de travail (*working draft*) sur les feuilles de style. Ce document devint une recommandation officielle le 17 décembre 1996 sous le nom de "Feuilles de style en cascade niveau 1" (*Cascading Style Sheets, level 1*) ou de manière abrégée CSS1.

Dans la foulée, un groupe de travail spécifique fut créé avec comme mission d'étendre le concept des feuilles de style. Une nouvelle recommandation fut publiée le 12 mai 1998 sous le vocable CSS2.

Quelques corrections et des extensions mineures sont proposées courant 2006 sous la dénomination CSS2.1.

Depuis 2009, il est beaucoup question des CSS3. Mais on ne peut plus parler de recommandation CSS3 car le groupe de travail a scindé la spécification en une multitude de modules qui suivent chacun leur évolution dans le temps. Citons le module CSS3 des couleurs (*CSS Color Module Level 3*), le module CSS3 de texte (*CSS Text Level 3*), le module qui s'occupe des arrière-plans et des bordures (*CSS Backgrounds and Borders Level 3*), etc.

Les différentes marques de navigateurs portent un intérêt évident pour ces nouvelles CSS3 et les intègrent rapidement. Parfois, celles-ci sont même implantées de façon provisoire en attendant la spécification définitive.

Compatibilité avec les navigateurs

On peut considérer que les feuilles de style CSS1 et CSS2 sont bien reprises par les navigateurs présents sur le marché, même s'il subsiste quelques bémols et bugs dans les navigateurs plus anciens.

Pour les navigateurs de notre étude (pour rappel, Internet Explorer 9, Firefox 3.6+, Safari 5+, Chrome 7+ et Opera 10.6+), les feuilles de style CSS1 et CSS2 ne posent plus de problèmes de compatibilité.

La situation est loin d'être la même pour les CSS3 (voir chapitre "Les feuilles de style CSS3"). Il importe dès à présent d'insister sur le côté nouveau voire expérimental de l'incorporation de ces feuilles de style CSS3 dans les navigateurs modernes. Ainsi, une propriété est reprise par tel navigateur mais pas par un autre, une propriété n'est incorporée que dans la version X d'un autre ou encore chaque navigateur traite celle-ci avec un préfixe particulier. Nous avons cependant veillé à fournir un code avec une compatibilité maximale même si l'étude de ces CSS3 est encore ardue.

Ne manquez cependant pas de percevoir l'avancée marquante des CSS3 dans la présentation du document car les nouveautés sont nombreuses et devraient modifier radicalement l'aspect des applications Web dans les prochaines années. Citons l'arrivée de la typographie avec les polices personnalisées, la présentation d'un contenu textuel en colonnes multiples, les ombres sur le texte et les éléments boîte, les bords arrondis, les dégradés de couleurs, la transparence de celles-ci, etc.

Outils pour les feuilles de style

Tout comme pour l'étude du Html5, un simple éditeur de texte brut comme NotePad ou SimpleText fait parfaitement l'affaire. D'autres petits logiciels d'édition peuvent être utilisés mais l'important est d'avoir du texte brut sans aucune mise en forme.

Parallèlement au validateur de documents Html (Présentation du Html5 - La validation du code Html5), il existe également un validateur dédié aux feuilles de style.

Le validateur en ligne du W3C pour les CSS (<http://jigsaw.w3.org/css-validator/>) est un outil de grande valeur et dont les qualités sont unanimement reconnues. Il permet de fournir des pages parfaites, scrupuleusement respectueuses des normes éditées par le W3C.



Cependant, son usage ne se révèle pas très adapté dans le cadre d'un apprentissage. La conception de ce validateur a été guidée par le principe de la stricte séparation entre la structure et la présentation. Ce faisant, il présuppose que les informations de présentation contenues dans le document Html sont intégrées par des feuilles de style externes (Validation par téléchargement du fichier vers le serveur et validation par saisie directe), ce qui est rarement adopté lors d'un apprentissage. En effet, l'encodage des feuilles de style sous forme de code CSS interne au document Html se révèle plus abordable et plus convivial. Le validateur du W3C offre bien la possibilité de valider un fichier Html5 avec CSS mais dans ce cas, le fichier doit être présent sur un serveur Web (Validation par URI), ce qui réclame une procédure longue et peu pratique pour une formation.

À l'heure actuelle, le validateur n'intègre pas encore les feuilles de style CSS3.

Le validateur CSS du W3C se révèle donc être plus un outil destiné aux professionnels qu'aux apprentis concepteurs.

La déclaration de style CSS

La déclaration d'un style s'effectue par le binôme propriété: valeur;

Exemple

```
background-color: red;
```

Ce qui pourrait se lire : "mettre la couleur de l'arrière-plan à la valeur rouge".

Détaillons cette déclaration :

- La propriété identifie ce qui sera défini dans le style adopté. Ces propriétés sont énumérées dans les spécifications CSS. Il existe de nombreuses propriétés portant, par exemple, sur la police de caractère (*font*), sur le texte (*text*), sur l'arrière-plan (*background*), sur la bordure (*border*), etc.
- La propriété est séparée de sa valeur par un double point.
- Les espaces sont permis. Ainsi certains auteurs ont pris l'habitude de mettre un espace entre le double point et la valeur pour une meilleure lisibilité du code.
- La valeur identifie la nature de l'effet de style souhaité. La valeur s'exprime par un mot-clé, un pourcentage, une grandeur en fonction de la propriété à laquelle elle est assignée.
- Une déclaration de style se termine toujours par un point-virgule.
- Il est possible de définir plusieurs déclarations de style pour un sélecteur. Par exemple : propriété1:valeur; propriété2:valeur; propriété3:valeur;.

Les sélecteurs de base

1. Les sélecteurs de balise

Dans le chapitre précédent, nous avons défini les feuilles de style comme des ajouts au code Html5.

La propriété de style vient donc "s'accrocher" à un élément ou une balise Html. C'est ce qu'on appelle un sélecteur.

La syntaxe d'un sélecteur de balise est la dénomination de la balise suivie de la déclaration de style comprise entre des accolades ouvrantes et fermantes.

```
sélecteur { déclaration de style}
```

Exemple

```
div { background-color: red; }
```

Remarquez que l'on reprend simplement la dénomination de la balise sans ses signes inférieur (<) et supérieur (>).
Donc uniquement le texte de la balise.

Ce qui pourrait se lire : "appliquer à toutes les balises de division <div> l'effet de style décrit entre les accolades".

Il est aussi possible d'affecter le même effet de style à différents sélecteurs de balise. Dans ce cas, chacune des balises concernées devra être séparée par une virgule.

Soit :

```
h1,h2,h3 { déclaration de style}
```

Ce qui signifie que les balises <h1>, <h2> et <h3> auront le même effet de style décrit entre les accolades.

2. Les sélecteurs de classe

Nous avions vu jusqu'à présent que les feuilles de style s'appliquaient à un sélecteur et que ce sélecteur pouvait être une balise Html.

Mais cette seule notion de sélecteur est trop limitative. Il faudrait pouvoir appliquer plusieurs styles différents à une même balise. C'est ce qu'apportent les classes. Avec celles-ci, le concepteur pourra, en quelque sorte, définir lui-même ses propres sélecteurs.

La définition d'une classe est :

```
.nom_de_la_classe { déclaration(s) de style}
```

Soit un point, suivi du nom que vous voulez attribuer à la classe, suivi de la déclaration de style entre des accolades.

Exemple

```
.rouge { background-color: red; }
```

Cette définition de classe pourra être utilisée pour n'importe quelle balise du document ou de l'application. D'où le terme de classe universelle.

Cependant, la classe ainsi définie n'a aucune influence aussi longtemps qu'elle n'est pas appelée dans le document.

L'appel de la classe de style se fait par :

```
<balise class="nom_de_la_classe">
```

Exemple

```
<div class="rouge"> ... </div>
```

Commentaires :

- Le point du sélecteur de classe n'est pas utilisé pour l'appel de la classe `class` (erreur fréquente).
- Une même classe peut être appelée plusieurs fois dans le document Html. Par exemple :

```
<div class="rouge">Item 1<div>
<div>Item 2<div>
<div class="rouge">Item 3<div>
```

Ici, seules les divisions avec la classe rouge auront un arrière-plan de couleur rouge.

- On peut mélanger les déclarations `class` et `id`.
- Un nom de classe peut comporter des lettres, des chiffres, le tiret et le caractère de soulignement. Le premier caractère ne peut être un nombre, un tiret ou un caractère de soulignement. Les espaces sont à éviter ainsi que les noms réservés du JavaScript.

Il existe une autre façon de définir une classe, mais elle est moins pratique (et donc moins utilisée) que la précédente :

```
nom_balise.nom_de_la_classe {déclaration(s) de style}
```

Soit un nom de balise, suivi d'un point, suivi du nom que vous voulez donner à la classe, suivi de la déclaration de style entre des accolades.

Exemple

```
blockquote.rouge { background-color: red; }
```

Avec cette définition de classe, celle-ci ne pourra être appliquée qu'à la seule balise désignée, soit ici `blockquote`.

3. Les sélecteurs d'identifiant

Le sélecteur `id`, aussi appelé identifiant `id`, permet d'appliquer une feuille de style (comme le sélecteur `class`) mais il ne pourra être appelé qu'une seule fois dans le document Html.

Le sélecteur `id` permet donc d'identifier un élément unique dans la page. Cette distinction prend toute son importance quand on fera appel à du JavaScript ou du Dhtml. En effet, `id` ne pouvant être appelé qu'une fois dans le document par un élément, celui-ci pourra être traité comme un objet unique qui sera par la suite manipulé par le JavaScript.

La définition d'un sélecteur `id` est :

```
#nom_de_l'identifiant {déclaration(s) de style}
```

Soit un dièse (#), suivi du nom que vous voulez donner à l'identifiant `id`, suivi de la déclaration de style entre des accolades.

Exemple

```
#rouge { background-color: red; }
```

Cette sélection ne pourra être utilisée qu'une seule fois dans le document.

Le sélecteur ainsi défini n'a aucune influence, aussi longtemps qu'il n'est pas appelé dans le document.

L'appel se fait par :

```
<balise id="nom_de_l'identifiant">
```

Exemple

```
<div id="rouge"> ... </p>
```

Commentaires

- Un identifiant `id` ne peut figurer qu'une fois dans le document Html. Ainsi, ce qui suit serait incorrect !

```
<p id="rouge"> ... </p>
...
<p id="rouge"> ... </p>
```

- Un document peut bien entendu comporter plusieurs identifiants `id` de noms différents mais ils ne pourront être appelés chacun qu'une seule fois.
- On peut mélanger les déclarations `class` et `id`.
- Un nom d'identifiant peut comporter des lettres, des chiffres, le tiret et le caractère de soulignement. Le premier caractère ne peut être un nombre, un tiret ou un caractère de soulignement. Les espaces sont à éviter ainsi que les noms réservés du JavaScript.

Il existe une autre façon de définir une classe mais elle est moins pratique (et donc moins utilisée) que la précédente :

```
nom_balise#nom_de_l'identifiant {déclaration(s) de style}
```

Soit un nom de balise, suivi d'un dièse (#), suivi du nom que vous voulez donner à l'identifiant, suivi de la déclaration de style entre des accolades.

Exemple

```
blockquote#rouge { background-color: red; }
```

Avec ce sélecteur, celui-ci ne pourra être appliqué qu'à la seule balise `<blockquote>` désignée par l'identifiant `rouge`.

-
-  Signalons que les feuilles de style CSS3 ont ajouté de nombreux sélecteurs. Ceux-ci seront abordés en détail au chapitre Les feuilles de style CSS3 - Les sélecteurs CSS3.
-

Les commentaires

Tout langage de description ou de programmation permet l'insertion de commentaires pour la compréhension du code et sa maintenance ultérieure. Dans une feuille de style CSS, un commentaire commence par les caractères barre oblique et étoile (soit /*) et se termine par étoile et barre oblique (soit */). On peut placer des commentaires n'importe où dans la feuille de style, excepté à l'intérieur d'une chaîne de caractères.

Exemple

```
/* Ceci est un commentaire */
```

Les unités de mesure

Les feuilles de style CSS permettent d'utiliser de nombreuses unités de mesure soit en pouces (*inches*), en centimètres, en millimètres, en points, en picas, en pixels et en pourcentage.

On distingue les valeurs relatives qui peuvent varier selon l'ordinateur utilisé et les valeurs absolues qui restent constantes quel que soit le matériel ou le software utilisé.

Les valeurs absolues sont :

Unité	Nom	Description	Valeur	Exemple
pt	point	72 pt = 1 inch	entier	48pt
pc	pica	1 pc = 12 pt	réel	4.5pc
mm	millimètre	1 mm = .24 pc	entier	60mm
cm	centimètre	1 cm = 10 mm	entier	6cm
in	inch	1 in = 2.54 cm	réel	0.1in

Les valeurs relatives sont :

Unité	Description	Valeur	Exemple
em	Unité relative se basant sur la taille de police par défaut de la page.	réel	1.8em
ex	Unité relative à la hauteur de la minuscule de l'élément sélectionné.	réel	1.3ex
px	Le pixel est la plus petite partie d'une image. Dépend de la résolution d'écran.	entier	220px
%	Pourcentage	entier	80%

Commentaires

- Les unités de mesure sont toujours notées par une abréviation comportant deux lettres.
- Il ne peut y avoir d'espace entre la valeur et l'unité. Si un espace est mis entre la valeur et l'unité, la feuille de style ne sera pas acceptée et ne sera donc pas affichée.
- Certaines propriétés acceptent des valeurs négatives.
- Il est généralement recommandé d'utiliser l'unité em pour décrire la taille des polices de caractères pour une plus grande stabilité entre les différents systèmes d'exploitation et navigateurs.

La notation des couleurs

Les feuilles de style CSS proposent de multiples notations pour déclarer une couleur. Soit :

- La notation hexadécimale classique soit `#ffcc00`. Cette manière est déjà bien connue de ceux qui sont habitués aux couleurs en Html. Elle définit la couleur, ou plutôt ses trois composantes rouge (r pour *red*), vert (g pour *green*) et bleu (b pour *blue*), via une notation hexadécimale de type `#rrggbb`.
- La notation hexadécimale abrégée soit `#fd3`. Cette notation particulière permet de gagner quelques caractères. Chaque chiffre est alors implicitement dupliqué. Par exemple, `#fd3` correspond à la notation classique `#ffffdd33`. On comprend donc qu'on ne pourra pas abréger une couleur comme `#cfe4f5`.
- La notation décimale soit, par exemple, `color: rgb(0, 0, 255)`. Le code RGB (RVB pour les francophones) de la couleur n'est plus encodé en valeur hexadécimale mais par un nombre entier compris entre 0 et 255. C'est donc l'équivalent décimal de la notation hexadécimale.
- La notation en pourcentage soit, par exemple, `color: rgb(25%, 50%, 0%)`. La valeur 0% signifie l'absence de la composante, 100% qu'elle est à son maximum.
- Les mots-clés soit, par exemple, `color: red`. Ces couleurs, au nombre de 17, sont désignées par un nom en anglais et constituent les couleurs basiques. On y retrouve : green (vert), yellow (jaune), blue (bleu moyen), orange (orange), white (blanc), red (rouge), black (noir), maroon (brun marron), lime (vert citron), aqua (turquoise), teal (cyan foncé), navy (bleu marine), olive (olive), fuchsia (fuchsia), purple (pourpre), silver (gris clair) et gray ou grey (gris foncé).

À ces notations, la spécification CSS3 a ajouté :

- La notation RGBa, qui obéit aux mêmes règles de fonctionnement que la notation classique RGB, mis à part qu'une composante est ajoutée à la valeur : `rgb(0,0,0)`. Celle-ci devient donc `rgba(0,0,0,0)`. La dernière valeur indiquant le degré d'opacité ou de transparence entre 0 et 1.
- La notation HSL (*Hue Saturation Luminance*), qui correspond à la notation TSL en français (Teinte Saturation Luminosité). La notation HSL consiste en trois valeurs. La première est exprimée en degrés de 0° à 359° (mais le symbole degré ° n'apparaît pas dans la notation). Elle correspond à une couleur dans la roue chromatique : le rouge (0°), le jaune (60°), le vert (120°), le cyan (180°), le bleu (240°) et le magenta (300°). Les secondes et troisièmes valeurs sont exprimées en pourcentage et notent respectivement la saturation et la luminosité. Par exemple, `color: hsl(0, 100%, 50%)` pour le rouge.
- La notation HSLa, qui ajoute une valeur comprise entre 0 et 1 pour la transparence ou l'opacité. Par exemple, `color: hsl(0, 100%, 50%, 0.5)` pour un rouge semi-transparent.

Pour vous aider dans le codage des couleurs, les sites suivants vous seront d'une grande utilité :

- <http://code-couleur.com/>
- http://fr.wikipedia.org/wiki/Liste_de_couleurs
- <http://www.ficml.org/jemimap/style/color/couleur.html>
- <http://www.webmaster-hub.com/outils/color.html>
- http://www.gratos.be/webmaster/code_couleur.htm

Une convention d'écriture

Les feuilles de style ne sont pas sensibles à la casse (*case insensitive*). Peu importe donc qu'elles soient écrites en majuscules et minuscules. Cependant, les éléments qui ne sont pas sous le contrôle des feuilles de style comme les noms de police ou les URLs peuvent être sensibles aux majuscules et aux minuscules (*case sensitive*).

L'usage veut que l'on encode les feuilles de style en minuscules.

Les CSS intégrées à un élément Html5

En Html, il est envisageable d'ajouter directement dans le code, une déclaration de style à une balise déterminée.

Ce style en ligne se présente ainsi :

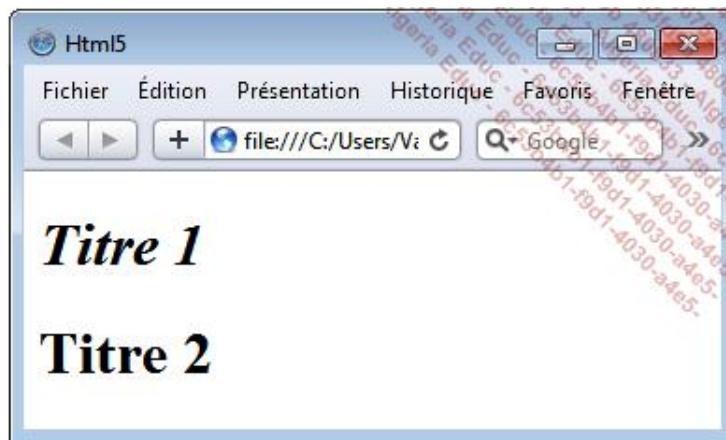
```
<h1 style="color: red;">Votre titre de niveau 1</h1>
```

Cette façon de procéder est cependant à éviter en vertu du principe de la séparation du contenu et de la présentation.

Elle peut cependant être utile pour donner la priorité à une propriété de style par rapport à ce qui a été spécifié dans la déclaration interne ou externe.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<h1 style="font-style: italic;">Titre 1</h1>
<h1>Titre 2</h1>
</body>
</html>
```



Seule la première balise `<h1>` avec la déclaration de style est affichée en italique.

Les CSS internes au document Html5

L'utilité principale des feuilles de style CSS est de déterminer un style qui s'applique à l'entièreté d'une page Html5.

À cet effet, le code des feuilles de style sera regroupé dans l'en-tête du document, soit entre les balises `<head>` et `</head>`. On parle alors de style interne.

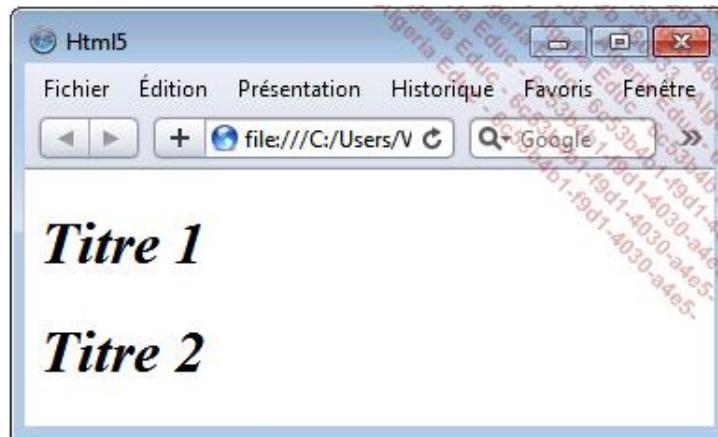
La syntaxe se présente alors ainsi :

```
<head>
<style type="text/css">
h1 { color: red; }
</style>
</head>
```

Cette déclaration de style aura comme effet de mettre en rouge le texte de toutes les balises `<h1>` du document Html.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
h1 { font-style: italic; }
</style>
</head>
<body>
<h1>Titre 1</h1>
<h1>Titre 2</h1>
</body>
</html>
```



On remarque que tous les titres de niveau 1 de cette page seront en italique.

Les CSS externes au document Html5

Il est également possible de regrouper les déclarations de style dans un fichier externe (distinct) au fichier Html.

Cette façon de procéder respecte au mieux la séparation du contenu et de la présentation.

Une des potentialités de ces feuilles de style externe est qu'il est possible de créer une feuille de style qui s'applique non pas à une seule page Html mais à l'ensemble des pages d'un site ou d'une application.

Cette technique met en œuvre deux fichiers :

- Un fichier qui contient la déclaration de style. Ce fichier aura une extension .css.
- Un document Html qui contient un lien vers le fichier CSS ainsi créé.

La feuille de style externe

```
h1 { text-decoration: overline; }
```

Il faut noter :

- Que ce fichier en texte brut peut se créer, par exemple, avec le Bloc-notes de Windows.
- Qu'il contient uniquement des déclarations de style.
- Qu'il ne peut donc pas contenir de balises Html et en particulier les balises `<style type="text/css"> ... </style>` des feuilles de style internes.
- On enregistrera ce fichier sous un nom de fichier quelconque avec une extension .css. Soit ici exemple.css.
- Pour la simplicité, ce fichier se situera souvent dans le même dossier que le document Html (adressage relatif ou local).

Le document Html5

On ajoutera dans l'en-tête, soit entre les balises `<head>` et `</head>`, un lien vers la feuille de style en question.

```
<link rel="stylesheet" type="text/css" href="exemple.css">
```

Étudions cette ligne de code :

- `link` indique au navigateur que ce qui suit est un lien.
- `rel="stylesheet"` précise que ce lien est relatif à une feuille de style.
- `href="exemple.css"` est l'écriture classique d'un lien en Html.
- Le lien peut être absolu (commençant par `http://...`) ou relatif.
- Rien n'empêche de mettre plusieurs balises `<link>` vers des feuilles de style externes différentes.
- Rien n'empêche non plus d'utiliser également une feuille de style interne.

Exemple

Le code du fichier exemple.css est tout simplement :

```
h1 { text-decoration: overline; }
```

Le code du fichier Html5 :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<link rel="stylesheet" type="text/css" href="exemple.css">
</head>
<body>
<h1>Titre de niveau 1</h1>
</body>
</html>
```

Tous les titres h1 de toutes les pages faisant référence à cette feuille de style seront surlignés.



► Même si cette méthode respecte au mieux les prescriptions du W3C (séparation du contenu et de la présentation), nous adopterons cependant, dans la suite de cet ouvrage, la technique des feuilles de style internes, moins abstraites et plus compréhensibles.

CSS importée (@import)

Une autre façon d'utiliser des feuilles de style externes est d'utiliser `@import`.

```
<style type="text/css">
@import url(exemple.css);
</style>
```

`@import` est une propriété de style CSS 2 (alors que la balise `<link>` est du Html). L'avantage est que l'on pourra non seulement l'utiliser pour appeler une feuille de style externe dans un document Html mais aussi pour importer une autre feuille de style dans la feuille de style externe.

Commentaires

- Les balises `<style> ... </style>` sont nécessaires car `@import` est une propriété de style (CSS 2).
- Il n'y a pas d'espace entre l'arobase et import.
- L'adresse de la feuille de style peut-être globale (`http:// ...`) ou locale.
- N'oubliez surtout pas le point-virgule final ! Pour rappel, `@import` est une propriété de style CSS.
- La balise `<link>` n'est plus nécessaire.

Le code complet pourrait être :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
@import url(exemple.css);
</style>
</head>
<body>
<h1>Titre de niveau 1</h1>
</body>
</html>
```

Tout comme pour les styles externes, tous les titres `<h1>` de toutes les pages du site seront surlignés.

La capture d'écran est identique à la précédente.

La notion de cascade

À la lecture des points précédents, on constate que l'on peut avoir plusieurs définitions de style soit en ligne, interne(s) ou externe(s). En cas de concurrence entre plusieurs éléments de style, intervient alors la notion de "cascade" (le Cascading de Cascading Style Sheets) ou d'ordre de priorité.

Le navigateur prend d'abord en considération les spécifications des feuilles de style externes (avec l'extension css), ensuite celles de style internes (soit celles à l'intérieur des balises <head>) et ensuite des feuilles de style en ligne (celles liées à un élément Html5).

Ainsi, en cas de conflit entre une spécification de style définie à la fois dans une feuille de style externe et dans une feuille de style interne, c'est la spécification de la feuille de style interne qui sera retenue par le navigateur. De même, en cas de conflit entre une feuille de style interne et en ligne, c'est cette dernière qui l'emportera.

L'ordre croissant de priorité (de la plus basse à la plus haute) est :

1. Propriétés du navigateur.
2. Les feuilles de style externes.
3. Les feuilles de style internes.
4. Les feuilles de style en ligne.

Les styles déclarés dans la feuille de style en ligne ont donc bien la plus haute priorité.

 La règle de priorité, pour l'affichage du document par le navigateur, sera d'utiliser la feuille de style la plus proche de l'élément.

Exemple

Définissons une feuille de style externe avec les titres de niveau 1 soulignés et une feuille de style interne avec ces mêmes titres de niveau 1 surlignés.

Le fichier de style externe (style1.css) :

```
h1 { text-decoration: overline; }
```

Le document Html5 :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<link rel="stylesheet" type="text/css" href="style1.css">
<style type="text/css">
h1 { text-decoration: overline; }
</style>
</head>
<body>
<h1>Titre de niveau 1</h1>
</body>
</html>
```



Le titre de niveau 1 est bien affiché comme surligné (`text-decoration: overline`), soit avec l'effet du style interne, le plus proche de l'étape d'affichage.

Il est cependant possible de passer outre ces règles de priorité par défaut en utilisant la valeur `!important` et redonner la préséance à une déclaration, peu importe les déclarations qui peuvent la suivre.

Exemple

On donnera la priorité au style externe soit au soulignement par la valeur `!important`.

Le fichier de style externe (style2.css) :

```
h1 { text-decoration: underline !important; }
```

Le document Html5 :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<link rel="stylesheet" type="text/css" href="style2.css">
<style type="text/css">
h1 { text-decoration: overline; }
</style>
</head>
<body>
<h1>Titre de niveau 1</h1>
</body>
</html>
```



La notion d'héritage

Le principe de l'héritage peut être expliqué par un rappel du Html et de ses balises imbriquées.

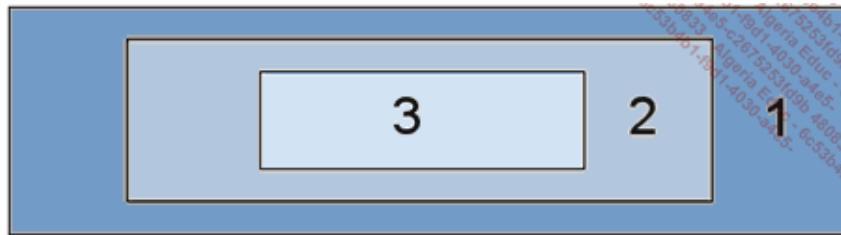
Soit le bout de code :

< b > Les éditions Eni < i > Collection < code > Ressources informatiques<code></ i >< / b >

Ressources informatiques sera non seulement sous une police à pas fixe (les balises `<code> ... </code>`) mais aussi en italique (compris dans des balises `<i> ... </i>`) et également en gras (balises ` ... `).

On pourrait dire que Ressources informatiques a hérité du formatage italique de la balise *<i>* et du formatage gras de la balise ****.

Il en est de même pour les styles.



Le style 2 se caractérise par son propre style et celui du style 1 dont il a hérité.

Le style 3 se caractérise par son propre style et celui du style 1 et 2 dont il a hérité.

Dans la même logique, on utilisera le terme de "parent" et "enfant".

Le style 1 est bien entendu parent du style 2. De même, le style 2 et le style 1 sont parents du style 3. Ou encore, le style 2 est enfant du style 1 et le style 3 est enfant des styles 2 et 1.

Ces notions de cascade et d'héritage sont assez subtiles et délicates à comprendre. Elles se préciseront en cours d'étude des feuilles de style.

Type de police

La police de caractères est un élément important de la présentation d'un document, que ce soit à l'écran ou à l'impression.

Le véritable problème pour les concepteurs Web est qu'ils sont dépendants des polices de caractères réellement disponibles sur la machine de leurs visiteurs. Les feuilles de style apportent une plus grande diversité dans le choix de la police mais sans contourner le problème des polices installées chez l'utilisateur.

La syntaxe de la déclaration de style pour le type de police du texte est :

font-family:	nom de la police ou famille de la police
--------------	--

Nom de la police

Exemple

```
font-family: Arial;
```

Commentaires

- Si l'on indique plusieurs polices, le nom de celles-ci doit être séparé par une virgule. Soit par exemple : font-family: Arial, Helvetica, Georgia;

Dans ce cas, le navigateur utilisera la première police nommée, à condition qu'elle soit présente sur le système du visiteur. Sinon, il passera à la suivante, etc. Si aucune des polices désignées n'existe sur le système, le navigateur revient à la police par défaut.

- Si le nom de la police comporte des espaces, le nom complet doit être mis entre guillemets (simples ou doubles) : font-family : 'Courrier New', 'Trebuchet MS', 'Lucida Console' .
- Le nom des polices n'est pas sensible à la casse (*case insensitive*). Il est équivalent d'écrire "Times New Roman" ou "times new Roman".

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<meta charset="iso-8859-1">
<style type="text/css">
h1 {font-family: 'Lucida Console';}
</style>
</head>
<body>
<h1>HTML5 + CSS</h1>
</body>
</html>
```



Le résultat peut varier d'un ordinateur à l'autre en fonction des polices installées.

Famille de police

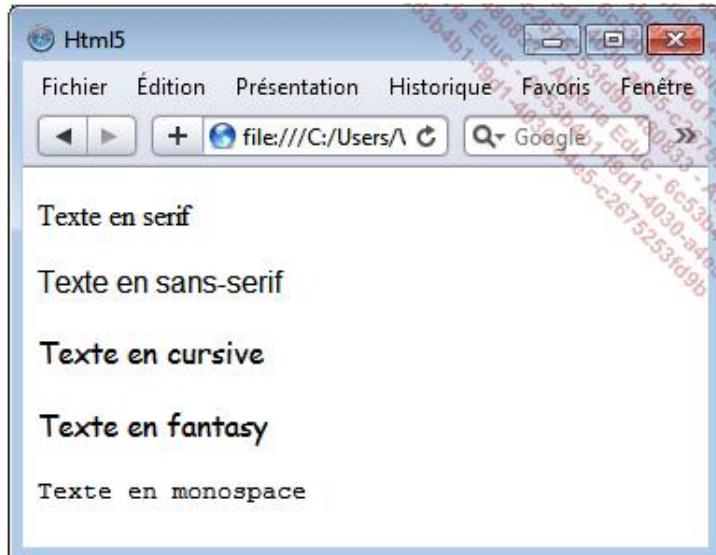
Au lieu de spécifier le nom de la police, on peut utiliser des familles de police ou des noms de polices génériques :

- **serif**. Les polices de type serif ont des terminaisons au bout de leurs traits et un espacement des lettres proportionnel (Times New Roman, Bodoni, Georgia, Garamond...).
- **sans-serif**. Les polices sans serif ont les bouts des traits pleins, sans évasement et un espacement des lettres proportionnel (Arial, Verdana, Helvetica, Trebuchet...).
- **cursive**. Famille de polices dont le résultat est proche d'une écriture manuscrite (Script, Adobe Poetica...).
- **fantasy**. Les polices fantaisies ou décoratives (Wingdings...).
- **monospace**. Toutes les lettres ont les mêmes dimensions. L'aspect est semblable à celui obtenu avec une machine à écrire manuelle et on emploie souvent ce type de police pour écrire du code informatique (Courier, Courier New).

Le navigateur choisira lui-même la police disponible sur la machine de l'utilisateur qui correspond aux critères du nom générique.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.serif {font-family: serif;}
.sans-serif {font-family: sans-serif;}
.cursive {font-family: cursive;}
.fantasy {font-family: fantasy;}
.monospace {font-family: monospace;}
</style>
</head>
<body>
<p class="serif">Texte en serif</p>
<p class="sans-serif">Texte en sans-serif</p>
<p class="cursive">Texte en cursive</p>
<p class="fantasy">Texte en fantasy</p>
<p class="monospace">Texte en monospace</p>
</body>
</html>
```



Le résultat peut varier d'un ordinateur à l'autre en fonction des polices installées.

Commentaire

Il est recommandé de spécifier un nom de police générique après une énumération de noms de police afin que l'affichage respecte le genre de police retenu plutôt que la police par défaut du navigateur.

Exemple : font-family: Arial, Verdana, Helvetica, sans-serif;

► Ne manquez pas de découvrir, au chapitre "Les feuilles de style CSS3 - Les polices personnalisées", la possibilité offerte par les CSS3 d'importer et d'afficher des polices personnalisées. Cette nouveauté marque l'entrée de la typographie dans les applications Web.

Italique

Cette propriété permet d'écrire un texte en italique, ou avec une inclinaison des caractères ou en écriture normale.

font-style:	italic; ou oblique; ou normal;
-------------	--------------------------------------

Commentaires

La distinction entre italic et oblique est assez subtile et peu visible à l'affichage. La valeur italic reprend la version en italique de la police retenue tandis que la valeur oblique est simplement l'inclinaison vers la droite des caractères de la police, par le navigateur.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.italic {font-style: italic;}
.oblique {font-style: oblique;}
</style>
</head>
<body>
<h2 class="italic">Titre en italique</h2>
<h2 class="oblique">Titre en oblique</h2>
</body>
</html>
```



Petite majuscule

Cette propriété permet d'afficher un texte normal en petites majuscules.

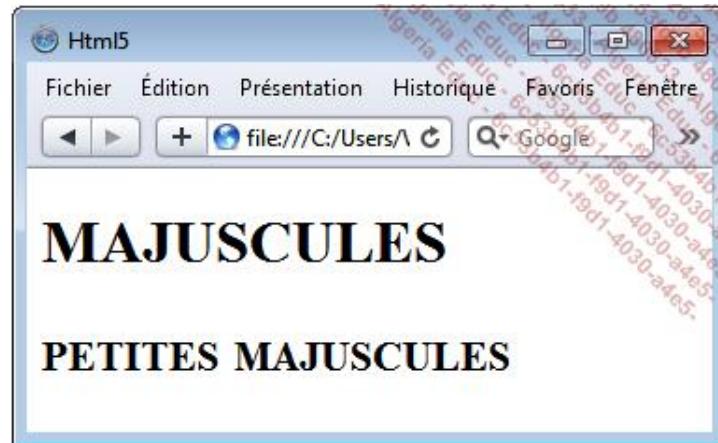
font-variant:	small-caps; ou normal;
---------------	---------------------------

Commentaire

La valeur `small-caps` affiche le texte en majuscules mais de dimension plus petite (*small*) que les majuscules normales. Ce qui donne un résultat moins "agressif".

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.caps { font-variant:small-caps; }
</style>
</head>
<body>
<h1>MAJUSCULES</h1>
<h1 class="caps">petites majuscules</h1>
</body>
</html>
```



Graisse

Cette propriété permet de mettre le texte en gras. Notons, même si les différences ne sont pas toujours perceptibles, que les possibilités sont nettement plus nombreuses qu'avec la balise .

font-weight:	bold; ou bolder; ou lighter; ou normal; ou valeur entre 100 et 900
--------------	--

Commentaires

- La valeur **bold** (gras) met la police de caractères en gras.

La valeur **bolder** (plus gras) accentue la graisse par rapport à l'élément parent.

La valeur **lighter** (plus léger) diminue la graisse par rapport à l'élément parent.

- Une valeur numérique entre 100 et 900. Les valeurs disponibles sont 100, 200, 300, 400, 500, 600, 700, 800, 900.

La valeur 500 correspond souvent à la graisse "normal".

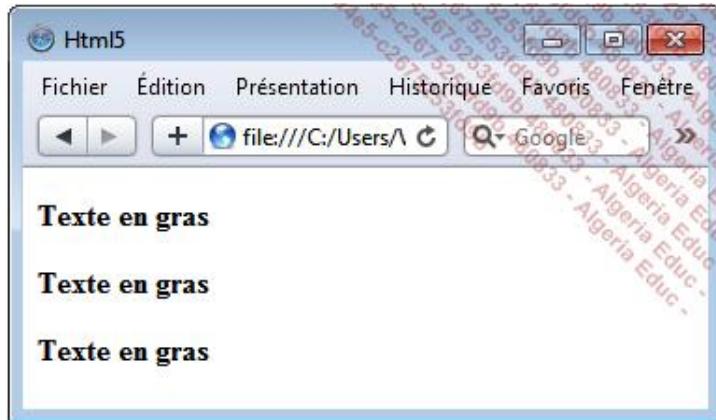
La valeur 700 correspond à gras.

La valeur 900 correspond (théoriquement) à super gras.

- Si cette propriété `font-weight` autorise de nombreuses valeurs, la totalité de ces valeurs ne sera cependant pas prise en compte par toutes les polices car elles n'ont pas toutes plusieurs versions de caractères en gras.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.bold { font-weight: bold; }
.bolder { font-weight: bolder; }
.bold900 { font-weight: 900; }
</style>
</head>
<body>
<p class="bold">Texte en gras</p>
<p class="bolder">Texte en gras</p>
<p class="bold900">Texte en gras</p>
</body>
</html>
```



Dans le cas présent, la différence n'est pas perceptible.

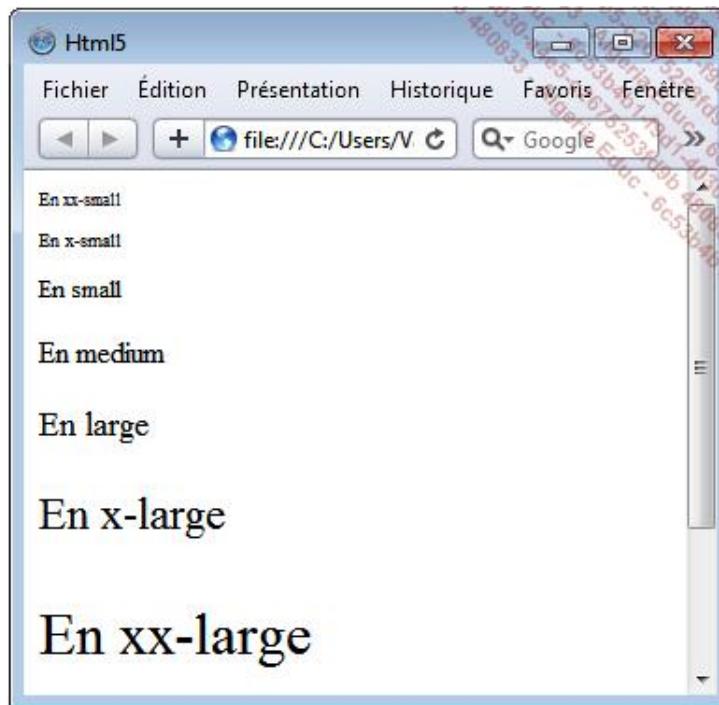
Taille des caractères

Alors que le Html 4.0 se limitait à sept tailles de caractères dans les balises <hx>, les feuilles de style permettent de définir un nombre illimité de tailles de caractères et ceci avec de multiples notations.

font-size:	valeur en pt ou valeur en px ou valeur en em ou xx-small, x-small, small, medium (défaut), large, x-large, xx-large ou smaller, larger soit plus petit, plus grand par rapport à l'élément parent ou en pourcentage de la taille de l'élément parent
------------	---

Commentaires

- La taille en valeur peut être indiquée en points (pt) ou en pixels (px). La notation en point est plus utilisée pour les pages destinées à être imprimées et le pixel est plus répandu pour l'affichage à l'écran. On s'accorde à considérer la taille de 12px comme plus ou moins équivalente à celle de 12pt.
- Les valeurs peuvent aussi être indiquées en mm, cm, ex, in ou pica mais elles sont peu fréquentes pour l'affichage des caractères.
- Il n'y a pas d'espace entre la valeur et l'unité, soit 12px est correct, 12 px ne l'est pas.
- On peut aussi déterminer la valeur par rapport à la hauteur em d'une police. 1 em correspond à 100 % de la taille en cours de la police, 1.2 em à 120 %, 0.8 em à 80 %. L'usage de em est limité aux polices de caractères.
- Les valeurs xx-small, x-small, small, medium, large, x-large, xx-large sont illustrées ci-après :



- Les valeurs smaller (plus petit), larger (plus grand) et en pourcentage sont aussi appelées valeurs relatives car elles dépendent de la taille de caractères de l'élément parent.
- La taille peut varier d'un système d'exploitation à l'autre. Ainsi, une taille de caractères de 12 points apparaîtra plus grande sous Windows qu'en Macintosh.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.taille {font-size: 96px;}
</style>
</head>
<body>
<h1>En Html5</h1>
<p class="taille">En CSS</p>
</body>
</html>
```



Limité pour des raisons de mise en page du livre, on aurait pu afficher, avec les feuilles de style, des tailles de caractères encore plus grandes.

Raccourci font

Bien que les propriétés de police peuvent être définies de façon indépendante, le raccourci font permet de regrouper, en une seule déclaration, les différentes propriétés font-style, font-variant, font-weight, font-size et font-family. Ce qui représente, dans la pratique de l'écriture, un gain de temps appréciable.

Un exemple serait :

```
font: italic bold small-caps 24pt Arial, sans-serif;
```

Cette seule déclaration correspond à :

```
font-style: italic;
font-weight: bold;
font-variant: small-caps;
font-size: 24pt;
font-family: Arial, sans-serif;
```

Commentaires

- Les différents attributs doivent être indiqués à la suite, séparés d'un espace.
- L'ordre n'a pas d'importance. Il n'est pas obligatoire de définir chaque propriété de police. Pour les propriétés non définies, les valeurs par défaut seront retenues.
- Le raccourci font permet également de définir l'interligne (propriété de texte, abordé au chapitre Le texte - Interligne). La notation est un peu particulière. On indique la valeur de taille de caractère (propriété font-size), suivie d'une barre oblique, suivie de la valeur de l'interligne (propriété line-height). Ainsi, par exemple, pour font: italic bold small-caps 24pt/1.5 Arial, sans-serif;, la taille de la police serait de 24 points et l'interligne de 1½ .

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
p {font: italic bold small-caps 24pt Arial, sans-serif;}
</style>
</head>
<body>
<p>Le raccourci font</p>
</body>
</html>
```



Les polices système

Depuis la spécification CSS 2, il est possible d'utiliser les polices système définies sur l'ordinateur de l'utilisateur.

Il suffit pour cela d'utiliser une déclaration de style font suivie d'un des mots-clés suivants :

- font: `caption`, police associée aux boutons.
- font: `icon`, police associée aux étiquettes des icônes.
- font: `menu`, police associée aux menus.
- font: `message-box`, police associée aux fenêtres de dialogue.
- font: `small-caption`, police associée aux étiquettes plus petites.
- font: `status-bar`, police associée à la barre de statut.

Cette possibilité est peu retenue dans la pratique.

Couleur

Cette propriété permet de définir la couleur du texte.

color:	nom de couleur ou notation hexadécimale sous sa forme #rrvvbb ou notation hexadécimale abrégée sous la forme #rvb ou notation en mode RVB avec des entiers de 0 à 255 ou notation en mode RVB avec des % de 0 à 100 ou notation en mode RVBa où a est compris entre 0 et 1 ou notation en mode HSL ou notation en mode HSLA où a est compris entre 0 et 1 ou transparent
--------	--

Ces diverses notations des couleurs ont déjà été abordées au chapitre Notions de base des CSS - La notation des couleurs.

Ainsi, il est équivalent de noter pour la couleur rouge :

```
color: red;  
color: #ff0000;  
color: #f00;  
color: rgb(100%,0%,0%);  
color: rgb(255,0,0);  
color: rgb(255,0,0,1);  
color: hsl(0, 100%, 50%);  
color: hsl(0, 100%, 50%,1);
```

Notons que la couleur par défaut du texte est le noir.

Exemple

```
<!DOCTYPE html>  
<html lang="fr">  
<head>  
<title>Html5</title>  
<meta charset="iso-8859-1">  
<style type="text/css">  
body { color: #ff0000; }  
</style>  
</head>  
<body>  
<h1>Texte en rouge</h1>  
</body>  
</html>
```



Commentaire

Bien qu'étudiée ici pour le texte, cette propriété `color` peut s'appliquer à bien d'autres éléments comme, par exemple, les lignes horizontales, les éléments de formulaires ou les arrière-plans.

Décoration

Cette propriété permet de personnaliser l'apparence du texte, par exemple avec le soulignement ou le surlignement.

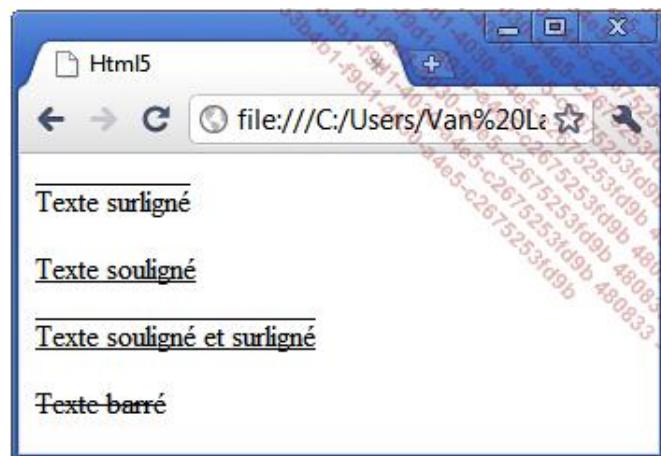
text-decoration:	underline. Souligne le texte. overline. Surligne le texte. line-through. Barre le texte en son milieu. none. Enlève toute apparence définie par ailleurs.
------------------	--

Commentaires

- Rien n'empêche de définir plusieurs valeurs de décoration. Il suffit de les séparer par un espace, par exemple, `text-decoration: underline overline;`. Dans le cas présent, le texte sera en même temps souligné et surligné.
- La valeur `line-through` est particulièrement utile dans les sites commerciaux pour indiquer notamment une révision de prix. Par exemple, ~~maintenant à 100 € 85 €~~. Rappelons la balise `` qui réalise le même effet.
- Appliquée à la balise de lien (`<a>`), la déclaration de style `text-decoration: none;` enlèvera le soulignement par défaut des liens. C'est assurément là sa principale utilisation.
- N'utilisez que très rarement le soulignement dans la conception de vos pages et applications Html5. En effet, le soulignement sur le Web indique par convention un lien. Il n'y a rien de plus agaçant que de cliquer sur ce que l'on pense être un lien et qui ne se révèle être qu'un soulignement inapproprié. Il existe d'autres façons de mettre des éléments en évidence. Pour rappel, la balise de soulignement `<s>` a été retirée du Html5.
- Pour les amateurs de précision, la valeur `blink` qui faisait clignoter le texte a (heureusement) disparu dans la spécification CSS3.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.deco1 { text-decoration: overline; }
.deco2 { text-decoration: underline; }
.deco3 { text-decoration: underline overline; }
.deco4 { text-decoration: line-through; }
</style>
</head>
<body>
<p class="deco1">Texte surligné</p>
<p class="deco2">Texte souligné</p>
<p class="deco3">Texte souligné et surligné</p>
<p class="deco4">Texte barré</p>
</body>
</html>
```



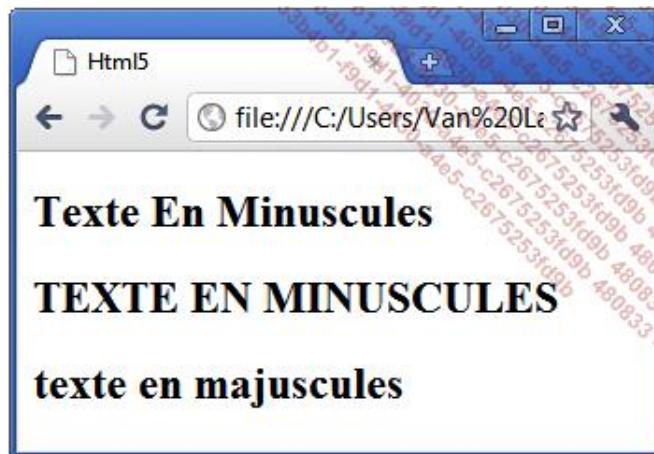
Transformation

Cette propriété affiche du texte en majuscules ou en minuscules, quelle que soit la façon dont les caractères figurent dans le code source.

text-transform:	capitalize. Met la première lettre de chaque mot en majuscule. uppercase. Met toutes les lettres en majuscules. lowercase. Met toutes les lettres en minuscules. none. Laisse les lettres inchangées.
-----------------	--

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.transform1 { text-transform: capitalize; }
.transform2 { text-transform: uppercase; }
.transform3 { text-transform: lowercase; }
</style>
</head>
<body>
<h2 class="transform1">texte en minuscules</h2>
<h2 class="transform2">texte en minuscules</h2>
<h2 class="transform3">TEXTE EN MAJUSCULES</h2>
</body>
</html>
```



Commentaire

Si cette propriété ne paraît pas d'une utilité évidente dans le domaine de l'édition, elle sera appréciée pour formater les données destinées aux ou provenant de bases de données.

Indentation

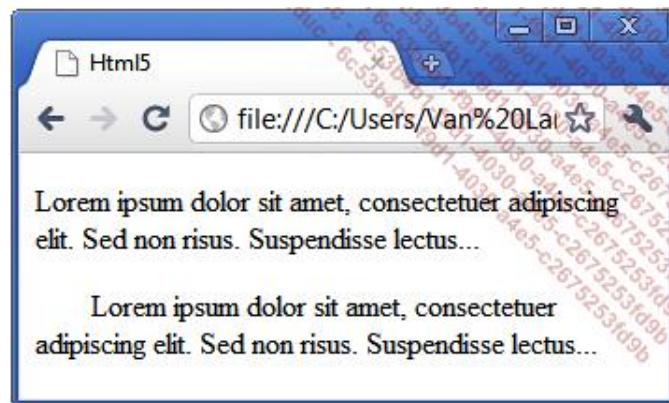
Cette propriété permet d'indenter la première ligne de texte d'un paragraphe.

text-indent:	valeur précise (par exemple 20px) ou valeur relative en pourcentage par rapport à la largeur du paragraphe.
--------------	--

Ce léger retrait, non prévu dans le Html, était obtenu par la répétition disgracieuse dans le code de nombreux espaces insécables ().

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.indentation { text-indent: 30px; }
</style>
</head>
<body>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed
non risus. Suspendisse lectus...
</p>
<p class="indentation">Lorem ipsum dolor sit amet, consectetuer
adipiscing elit. Sed non risus. Suspendisse lectus...
</p>
</body>
</html>
```



Commentaire

La valeur de l'indentation peut être négative. Ceci permet de réaliser des effets inattendus mais le résultat n'est pas toujours garanti.

Espace entre les lettres

Cette propriété permet de faire varier l'espacement entre les caractères. Ce qui, utilisé sans exagération, peut favoriser la lisibilité d'un texte.

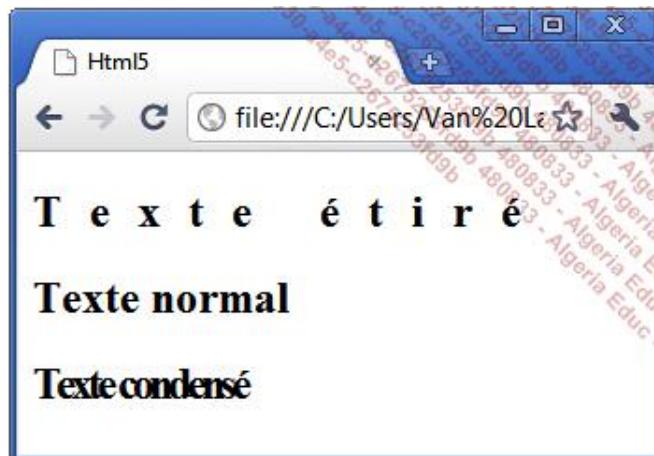
letter-spacing:	valeur de longueur (par exemple 8px)
	normal

Commentaires

- Les valeurs de longueur doivent être déterminées en valeurs absolues. Les valeurs relatives, par exemple en pourcentage, ne sont pas admises ici.
- Une valeur positive ajoute de l'espace entre les lettres. Une valeur négative diminue l'espace entre les lettres.
- La justification du texte permet au navigateur de modifier l'interlettrage. Une valeur de letter-spacing mise à 0 empêchera la justification du texte.
- Pour empêcher que la propriété letter-spacing ne rapproche trop les mots et rende ainsi le texte illisible, on peut la coupler avec un espacement normal de mots word-spacing: normal; (voir propriété ci-après).

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.espace1 { letter-spacing: 15px; }
.espace2 { letter-spacing: -3px; }
</style>
</head>
<body>
<h2 class="espace1">Texte étiré</h2>
<h2>Texte normal</h2>
<h2 class="espace2">Texte condensé</h2>
</body>
</html>
```



Espace entre les mots

Cette propriété, très proche de la précédente concernant l'espace entre les lettres, permet ici de faire varier l'espacement entre les mots. Ce qui, utilisé sans exagération, peut favoriser la lisibilité d'un texte.

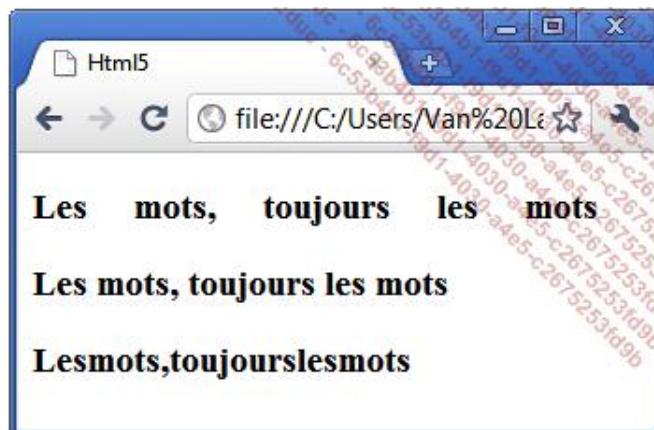
word-spacing:	valeur de longueur (par exemple 8px)
	normal

Commentaires

- Les valeurs de longueur doivent être déterminées en valeurs absolues. Les valeurs relatives, en pourcentage par exemple, ne sont pas admises ici.
- Une valeur positive ajoute de l'espace entre les mots. Une valeur négative diminue l'espace entre les mots.
- La justification du texte permet au navigateur de modifier l'espace entre les mots. Une valeur de word-spacing définie de 0 empêchera la justification du texte.
- Les effets de cette propriété sur la lisibilité du texte sont parfois désastreux.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.espace1 { word-spacing: 20px; }
.espace2 { word-spacing: -5px; }
</style>
</head>
<body>
<h3 class="espace1">Les mots, toujours les mots</h3>
<h3>Les mots, toujours les mots</h3>
<h3 class="espace2">Les mots, toujours les mots</h3>
</body>
</html>
```



Interligne

Cette propriété offre la possibilité de modifier les interlignes.

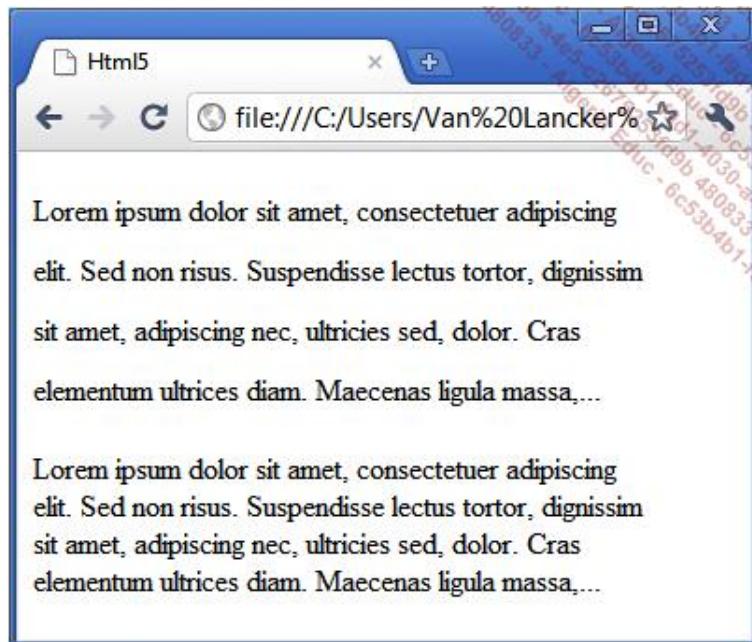
line-height:	un nombre ou une valeur de longueur ou un pourcentage ou normal
--------------	--

Commentaires

- Le nombre de `line-height` multiplie l'interligne induit par la taille de la police de caractères. Par exemple 2 pour un double interligne.
- Une valeur de longueur. Celle-ci détermine l'interlignage sans tenir compte de la police de caractères. Par exemple, 18px.
- Un pourcentage. L'interligne sera déterminé en pourcentage de l'interligne induit par la taille de la police. Ainsi un double interligne peut être déterminé par 200 %.
- La valeur donnée peut être négative. Ce qui diminuera l'interligne normal et bien souvent la lisibilité.
- La valeur `normal` correspond à l'interligne normal.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.interligne { line-height: 200%; }
</style>
</head>
<body>
<p class="interligne">
Lorem ipsum dolor sit amet, consectetur adipiscing<br>
elit. Sed non risus. Suspendisse lectus tortor, dignissim<br>
sit amet, adipiscing nec, ultricies sed, dolor. Cras<br>
elementum ultrices diam. Maecenas ligula massa,....
</p>
<p>
Lorem ipsum dolor sit amet, consectetur adipiscing<br>
elit. Sed non risus. Suspendisse lectus tortor, dignissim<br>
sit amet, adipiscing nec, ultricies sed, dolor. Cras<br>
elementum ultrices diam. Maecenas ligula massa,....
</p>
</body>
</html>
```



Il faut rappeler que le raccourci font (chapitre La police de caractères - Écriture raccourcie) permet également de définir l'interligne. L'interligne est défini avec la taille de caractères. La notation s'indique par la valeur de taille de caractères (propriété `font-size`), suivi d'une barre oblique, suivi de la valeur de l'interligne (propriété `line-height`).

Exemple

```
font: italic bold small-caps 24pt/1.5 Arial, sans-serif;
```

Espaces vides

Cette propriété permet de contrôler les espaces vides.

Depuis ses origines, le langage Html ignore les espaces vides, qu'il ramène à un seul espace, ainsi que les retours chariot. Les espaces vides multiples et les retours chariot sont aussi pris en charge en Html5 par la balise `<pre> ... </pre>`. Par contre l'attribut `nowrap` du Html 4.0, qui empêche la rupture de ligne automatique, n'est pas repris en Html5. Il est donc impératif dans ce cas de passer par la présente déclaration de style.

white-space:	pre OU nowrap OU normal.
--------------	--------------------------------

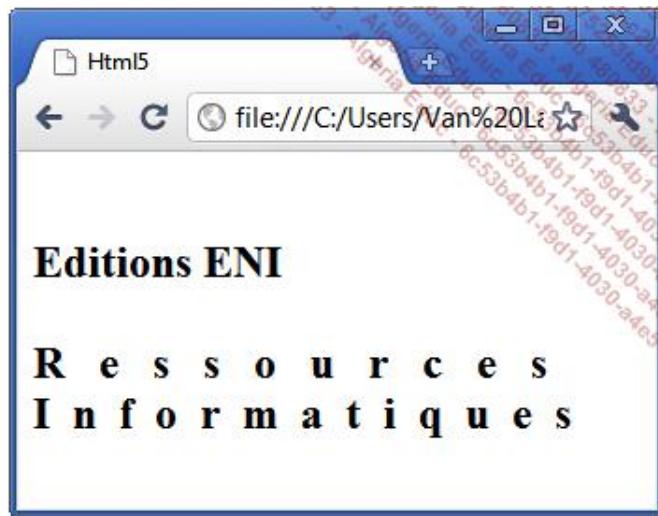
Commentaires

- La valeur `pre` préserve les espaces multiples et les retours chariot.
- La valeur `nowrap` empêche la rupture de ligne et affiche le texte horizontalement quelle que soit la place à utiliser dans le navigateur. Dans certains cas, l'utilisateur devra utiliser la barre de défilement horizontale pour lire la totalité du texte.
- La valeur `normal` laisse le navigateur gérer à son gré les espaces multiples, les retours chariot et les ruptures de ligne.

Exemple de `pre`

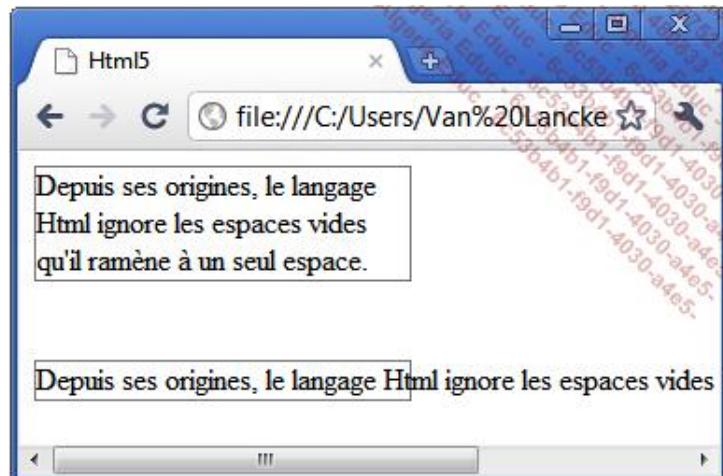
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
h2 { white-space: pre; }
</style>
</head>
<body>
<h2>
Editions ENI

R   e   s   s   o   u   r   c   e   s
I   n   f   o   r   m   a   t   i   q   u   e   s
</h2>
</body>
</html>
```



Exemple de nowrap

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
div { width: 200px;
      border: 1px solid gray}
.exemple { white-space: nowrap;}
</style>
</head>
<body>
<div>
Depuis ses origines, le langage Html ignore les espaces vides
qu'il ramène à un seul espace.
</div>
<pre>
</pre>
<div class="exemple">
Depuis ses origines, le langage Html ignore les espaces vides
qu'il ramène à un seul espace.
</div>
</body>
</html>
```



Alignment horizontal

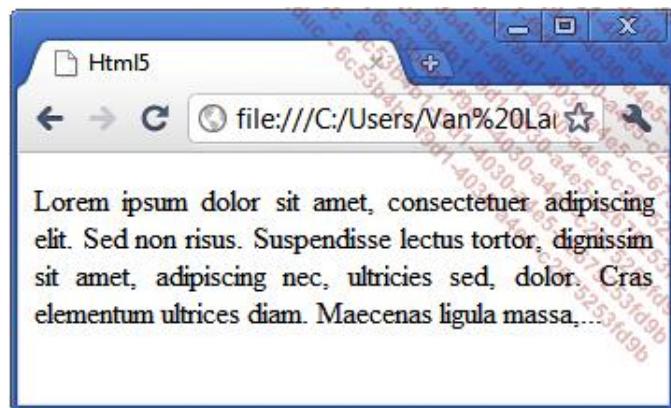
Cette propriété permet de réaliser un alignement comme dans le plus banal traitement de texte.

text-align:	left; Texte aligné à gauche (par défaut). right; Texte aligné à droite. center; Texte centré. justify; Texte justifié. auto; Alignement par défaut.
-------------	---

La justification, qui donne à chaque ligne de texte la même longueur, entraîne parfois à l'écran des résultats inattendus, voire décevants. Les opinions diffèrent quant à la lisibilité.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
p { text-align: justify; }
</style>
</head>
<body>
<p>
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non
risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing
nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas
ligula massa,....
</p>
</body>
</html>
```



Alignment vertical

Cette propriété détermine l'alignement vertical d'un texte par rapport à un autre élément du code.

vertical-align:	baseline; (par défaut). sub; super; top; middle; bottom; sans mention. valeur de longueur. pourcentage.
-----------------	---

Commentaires

- Les valeurs `baseline`, `sup` et `super` s'utilisent par rapport à une ligne de texte.
- La valeur `baseline` aligne le texte par rapport au bas de la ligne de texte (ligne de base).
- La valeur `sub` met le texte concerné en indice, soit en dessous de la ligne de base.
- La valeur `super` met le texte concerné en exposant, soit au-dessus de la ligne de base.
- Cet alignement sans mention de valeur de longueur ou de pourcentage laisse la main au navigateur pour l'affichage.
- Les valeurs `top`, `middle`, `bottom` alignent relativement à l'élément parent, qui peut être du texte ou encore une ou des cellule(s) de tableau.
- La valeur `top` aligne par rapport au plus haut de l'élément parent.
- La valeur `middle` aligne par rapport au milieu de l'élément parent.
- La valeur `bottom` aligne par rapport au plus bas de l'élément parent.
- Une valeur de longueur positive aligne au-dessus de la ligne de base. Une valeur négative, en dessous. Par exemple, `vertical-align: 1.7em` mettra en exposant de 1.7 em.
- Un pourcentage positif aligne au-dessus de la ligne de base. Une valeur négative, en dessous. Par exemple, `vertical-align: -20%` mettra en indice de 20 %.

Exemple 1

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.align1 { vertical-align:super; }
.align2 { vertical-align:baseline; }
```

```

.align3 { vertical-align:sub; }

```

```

</style>
</head>
<body>
<h2>Ressources <i class="align1">Informatiques</i></h2>
<h2>Ressources <i class="align2">Informatiques</i></h2>
<h2>Ressources <i class="align3">Informatiques</i></h2>
</body>
</html>

```



Exemple 2

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
table { border: 1px solid gray;
         border-collapse: collapse;
         width: 200px; }
td { border: 1px solid gray;
      width: 100px; }
.top { vertical-align:top; }
.middle { vertical-align: middle; }
.bottom { vertical-align: bottom; }
</style>
</head>
<body>
<table>
<tr>
<td><br>&nbsp;</td>
<td class="top">texte</td>
</tr>
<tr>
<td><br>&nbsp;</td>
<td class="middle">texte</td>
</tr>
<tr>
<td><br>&nbsp;</td>
<td class="bottom">texte</td>
</tr>
</table>
</body>
</html>

```



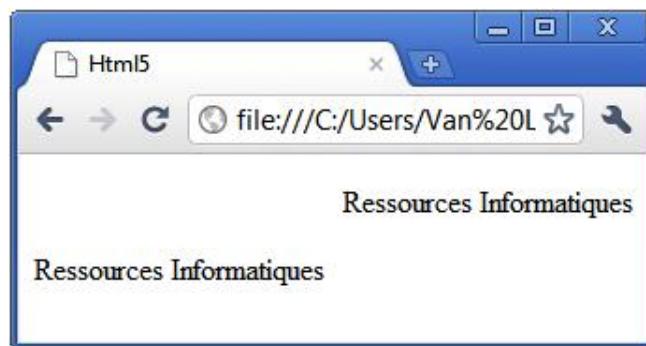
Direction du texte

Cette propriété permet de changer la direction du texte. Si le français se lit de gauche à droite, cette propriété de style sera particulièrement utile pour les langues qui se lisent de droite à gauche comme l'arabe et l'hébreu.

direction:	ltr (left to right). Écriture de gauche à droite. rtl (right to left). Écriture de droite à gauche.
------------	--

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.direction1 { direction: rtl; }
.direction2 { direction: ltr; }
</style>
</head>
<body>
<p class="direction1">Ressources Informatiques</p>
<p class="direction2">Ressources Informatiques</p>
</body>
</html>
```



Longueur et hauteur

Les propriétés de style `width` et `height` vont permettre d'associer, à certaines balises de texte, respectivement la largeur et la hauteur d'une zone entourant le contenu textuel.

Cette propriété est très utile car elle évite de passer par l'astuce de tableaux factices pour afficher du texte dans une zone rectangulaire.

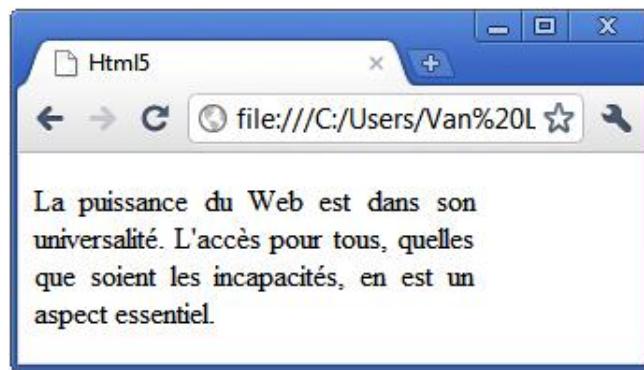
width:	valeur de longueur ou pourcentage
height:	valeur de longueur ou pourcentage

Commentaires

- Les valeurs `width` (largeur) et `height` (hauteur) déterminent la largeur et la hauteur du contenu.
- La valeur de longueur spécifie une valeur fixe.
- Le pourcentage spécifie une valeur relative.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
p { width: 235px;
   text-align:justify;}
</style>
</head>
<body>
<p>La puissance du Web est dans son universalité. L'accès pour
tous, quelles que soient les incapacités, en est un aspect
essentiel.</p>
</body>
</html>
```



Les propriétés `width` et `height` nécessitent de nombreux commentaires. Nous réservons ceux-ci lors de leur étude détaillée au chapitre Les propriétés de boîte - Largeur et hauteur.

Sortes de marqueurs

Cette propriété permet de déterminer l'apparence de la puce ou le style de numérotation de la liste.

list-style-type:	disc; Cercle plein. circle; Cercle vide. square; Carré. ou decimal; Soit 1, 2, 3, etc. decimal-leading-zero; Soit 01, 02, 03, etc. upper-roman; Soit I, II, III, etc. lower-roman; Soit i, ii, iii, etc. upper-alpha; Soit A, B, C, etc. lower-alpha; Soit a, b, c, etc.
------------------	---

Commentaires

Les éléments `disc`, `circle`, `square` sont identiques à ceux de la balise `` en Html5. Avec celle-ci cependant, la forme est déterminée par défaut par le navigateur.

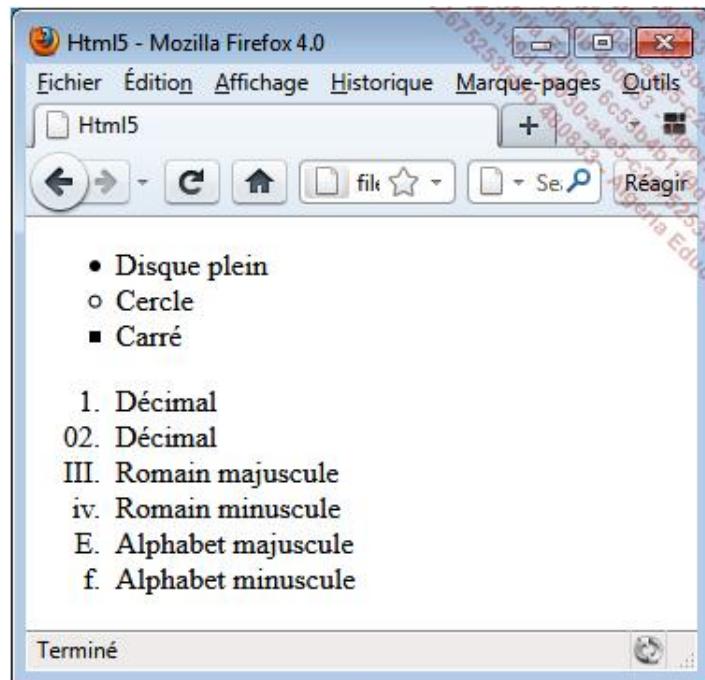
Même constatation pour les numérotations de la balise ``. Celle-ci est uniquement décimale.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.style1 { list-style-type:disc; }
.style2 { list-style-type:circle; }
.style3 { list-style-type:square; }
.style4 { list-style-type:decimal; }
.style5 { list-style-type:decimal-leading-zero; }
.style6 { list-style-type:upper-roman; }
.style7 { list-style-type:lower-roman; }
.style8 { list-style-type:upper-alpha; }
.style9 { list-style-type:lower-alpha; }
</style>
</head>
<body>
<ul>
- Disque plein
- Cercle
- Carré

</ul>
<ol>
- Décimal
- Décimal
- Romain majuscule
- Romain minuscule
- Alphabet majuscule
- Alphabet minuscule

</ol>
</body>
</html>
```



Marqueur graphique

Cette propriété de CSS permet de remplacer les puces par une image. Ce qui apporte un peu de diversité par rapport aux puces traditionnelles du Html.

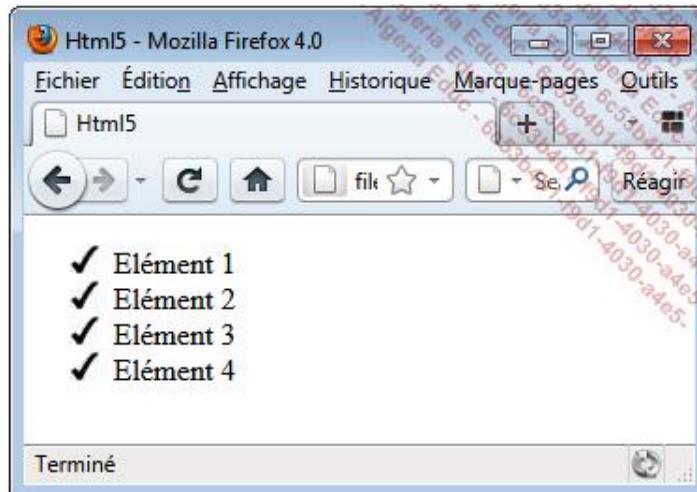
list-style-image:	url(votre_puce.gif);
	none;

Commentaires

- La valeur `url(votre_puce.gif)` désigne l'adresse (URL) de l'image à utiliser comme puce. On peut employer une adresse relative vers une image comprise dans le même dossier (comme dans notre exemple) ou une adresse absolue (avec `http://...`).
- Les images peuvent être au format GIF, JPEG ou PNG.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
ul { list-style-image: url(puce.gif); }
</style>
</head>
<body>
<ul>
<li>Elément 1</li>
<li>Elément 2</li>
<li>Elément 3</li>
<li>Elément 4</li>
</ul>
</body>
</html>
```



L'image puce.gif est disponible dans l'espace de téléchargement.

Retrait

Cette propriété détermine la position de la seconde ligne (et les suivantes) d'un élément de liste par rapport à la puce ou la numérotation.

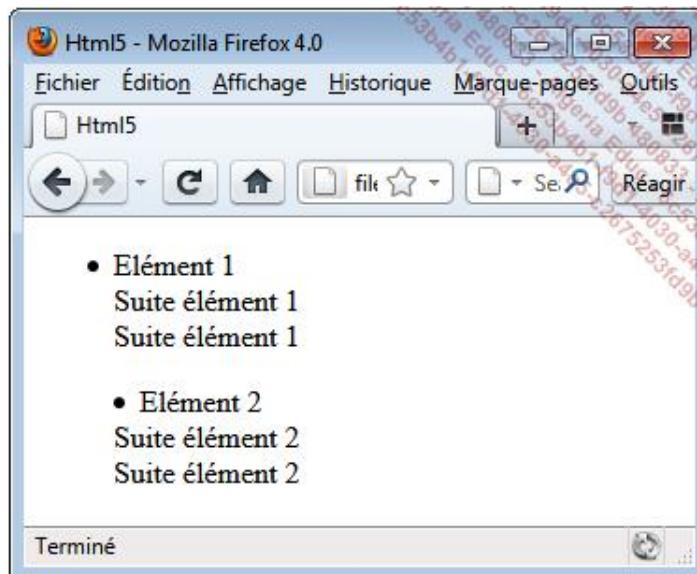
list-style-position:	outside; (par défaut)
	inside;

Commentaires

- La valeur `outside` est la présentation classique des listes avec toutes les lignes de l'élément en retrait par rapport à la puce ou au numéro.
- La valeur `inside` est une présentation un peu particulière avec un retrait uniquement sur la première ligne de l'élément de la liste. Les autres lignes viennent s'aligner sur le marqueur.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.position1 { list-style-position: outside; }
.position2 { list-style-position: inside; }
</style>
</head>
<body>
<ul>
<li class="position1">Elément 1<br>
Suite élément 1<br />
Suite élément 1</li>
</ul>
<ul>
<li class="position2">Elément 2<br>
Suite élément 2<br />
Suite élément 2</li>
</ul>
</body>
</html>
```



Écriture de liste raccourcie

Cette propriété permet de reprendre dans une seule déclaration de style les valeurs précédentes (`list-style-type`, `list-style-image`, `list-style-position`).

Exemple

```
list-style: url(puce.gif) circle inside;
```

Commentaires

- Les valeurs sont regroupées et séparées par des espaces.
- Il peut sembler bizarre de déterminer une puce en image et une puce de type cercle. En CSS, cela signifie que si le navigateur ne trouve pas l'image à l'adresse spécifiée, il affichera une puce en forme de cercle.
- Aucune de ces valeurs n'est obligatoire et on peut aussi ne définir qu'une seule valeur.

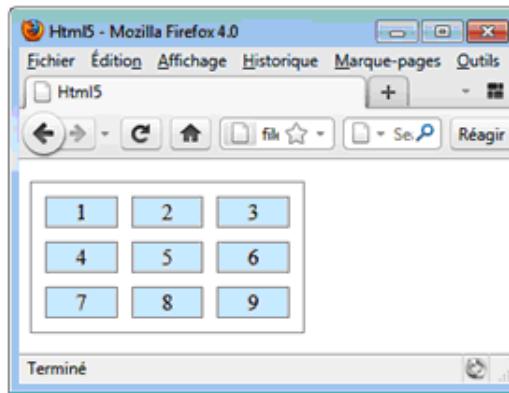
Espace entre les cellules de tableaux

Cette propriété permet de spécifier l'espacement entre chaque cellule. C'est l'équivalent de l'attribut `cellspacing` du Html 4.0.

border-spacing:	valeur numérique en px. valeur numérique en pt. pourcentage.
-----------------	--

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
table { width: 200px;
        text-align: center;
        border : 1px solid gray;
        border-spacing: 10px; }
td { border : 1px solid gray;
      background-color: rgb(213,230,245); }
</style>
</head>
<body>
<table>
<tr>
<td>1</td><td>2</td><td>3</td>
</tr>
<tr>
<td>4</td><td>5</td><td>6</td>
</tr>
<tr>
<td>7</td><td>8</td><td>9</td>
</tr>
</table>
</body>
</html>
```



Sans la propriété de style `border-spacing`, le tableau se présente par défaut ainsi :

1	2	3
4	5	6
7	8	9

Il est possible de n'avoir qu'une seule bordure de tableau avec border-spacing: 0px.

1	2	3
4	5	6
7	8	9

Les bordures des tableaux

Nous avons vu que par défaut, le navigateur affiche deux bordures aux tableaux : la bordure qui en fait le contour et les bordures qui entourent les cellules.

La propriété de style `border-collapse` permet de gérer cette situation.

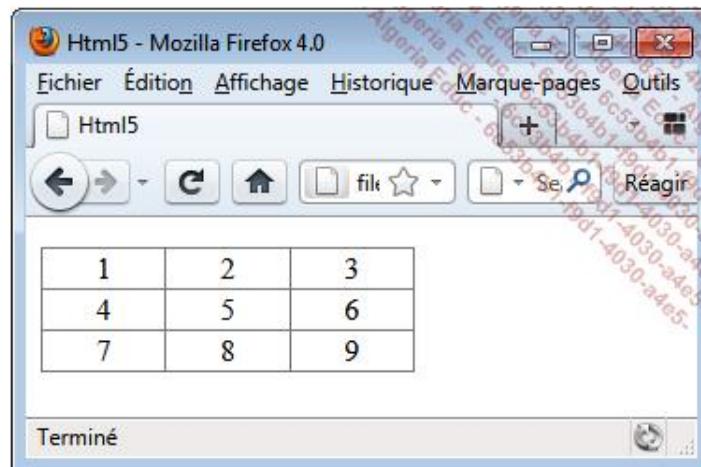
border-collapse:	collapse;
	separate;

Commentaires

- La valeur `collapse` fusionne les deux bordures adjacentes, ce qui donne l'aspect d'une bordure unique.
- La valeur `separate` affiche les deux bordures séparément, ce qui est la situation par défaut.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
table { width: 200px;
        text-align: center;
        border : 1px solid gray;
        border-collapse: collapse;}
td { border : 1px solid gray;}
</style>
</head>
<body>
<table>
<tr>
<td>1</td><td>2</td><td>3</td>
</tr>
<tr>
<td>4</td><td>5</td><td>6</td>
</tr>
<tr>
<td>7</td><td>8</td><td>9</td>
</tr>
</table>
</body>
</html>
```



Les cellules vides de tableaux

Cette propriété permet de déterminer le comportement du navigateur lorsqu'il rencontre une cellule vide dans un tableau.

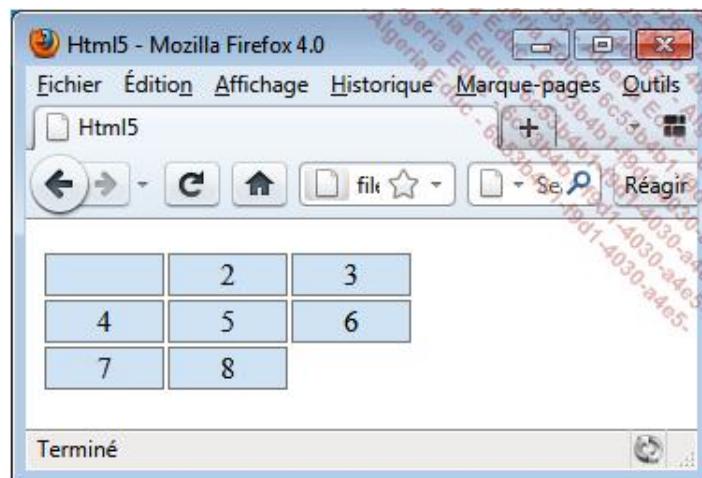
empty-cells:	show; hide;
--------------	----------------

Commentaires

- La valeur `show` affiche la cellule vide. Les bordures et l'arrière-plan sont ainsi visibles.
- La valeur `hide` n'affiche pas la cellule vide. Les bordures et l'arrière-plan sont ainsi invisibles.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
table { width: 200px;
    text-align: center; }
td { border : 1px solid gray;
    background-color: rgb(213,230,245); }
.show {empty-cells: show;}
.hide {empty-cells: hide;}
</style>
</head>
<body>
<table>
<tr>
<td class="show"></td><td>2</td><td>3</td>
</tr>
<tr>
<td>4</td><td>5</td><td>6</td>
</tr>
<tr>
<td>7</td><td>8</td><td class="hide"></td>
</tr>
</table>
</body>
</html>
```



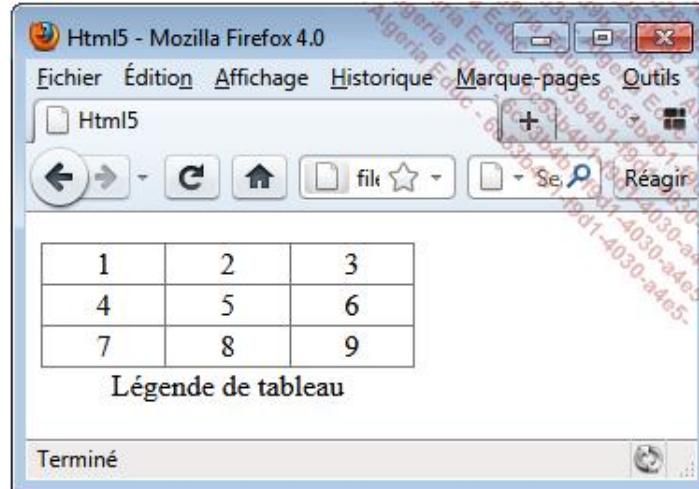
Position de la légende

L'attribut align de la balise <caption> en Html 4.0 ayant disparu, il faut faire appel à cette propriété de style pour positionner la légende du tableau.

caption-side:	top; Légende au-dessus du tableau (défaut). bottom; Légende en dessous du tableau.
---------------	---

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
table { width: 200px;
        text-align: center;
        border : 1px solid gray;
        border-collapse: collapse;}
td { border : 1px solid gray;}
caption { caption-side: bottom; }
</style>
</head>
<body>
<p></p>
<table>
<caption>Légende de tableau</caption>
<tr>
<td>1</td><td>2</td><td>3</td>
</tr>
<tr>
<td>4</td><td>5</td><td>6</td>
</tr>
<tr>
<td>7</td><td>8</td><td>9</td>
</tr>
</table>
</body>
</html>
```



Préambule

Le Html 4.0 offrait, par certains attributs, la possibilité d'ajouter une couleur ou une image de fond aux pages et éventuellement aux tableaux ou aux cellules de tableaux. Le Html5 ne propose quant à lui aucun attribut pour réaliser ces effets de présentation. Il faut, en Html5, passer impérativement par les feuilles de style CSS pour ajouter un arrière-plan à un élément. Il faut signaler cependant que les possibilités offertes par les feuilles de style sont plus étendues. Elles permettent, par exemple, de contrôler au gré de votre créativité la répétition de l'image de fond qui ne sera plus ainsi nécessairement disposée en mosaïque.

Couleur d'arrière-plan

Cette propriété permet de donner une couleur de fond à un élément.

background-color:	nom de couleur ou notation hexadécimale sous sa forme #xxxxxx ou notation hexadécimale abrégée sous la forme #xxx ou notation en mode RVB avec des entiers de 0 à 255 ou notation en mode RVB avec des % de 0 à 100 ou notation en mode RVBa où a est compris entre 0 et 1. notation en mode HSL notation en mode HSLa où a est compris entre 0 et 1. transparent
-------------------	---

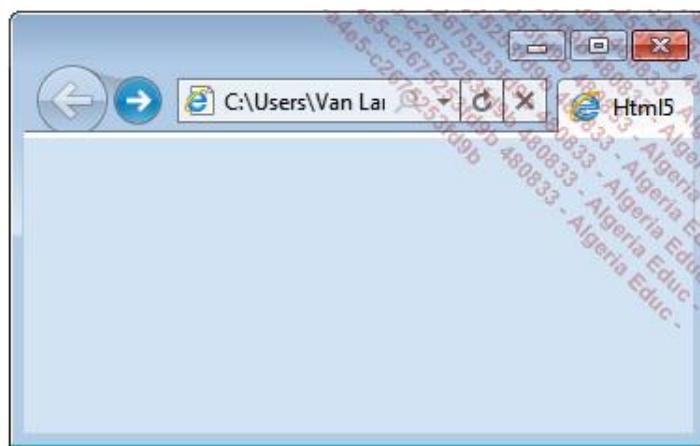
Les différentes notations des couleurs ont été abordées en détail au chapitre Notions de base des CSS - La notation des couleurs.

Notons que la valeur transparent, lorsqu'elle est appliquée à l'arrière-plan de la page, ne fait que reprendre la couleur par défaut du navigateur.

Exemple 1

Appliquer une couleur de fond à la page.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
body { background-color: rgb(215,230,245); }
</style>
</head>
<body>
</body>
</html>
```

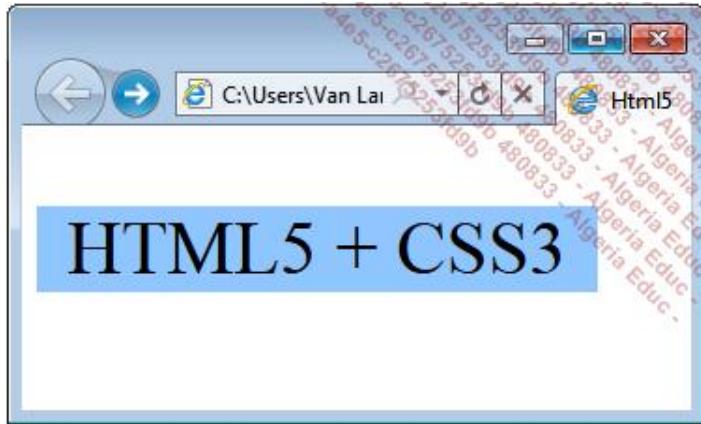


Exemple 2

Appliquer une couleur de fond à une division <div>.

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

```
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
div { width: 300px;
      background-color: #9cf;
      font-size: 250%;
      text-align: center; }
</style>
</head>
<body>
<div>
<p>HTML5 + CSS3</p>
</div>
</body>
</html>
```



Insertion d'une image d'arrière-plan

Cette propriété insère une image de fond disposée de façon classique, soit en mosaïque.

background-image:	url(fichier_image);
	none;

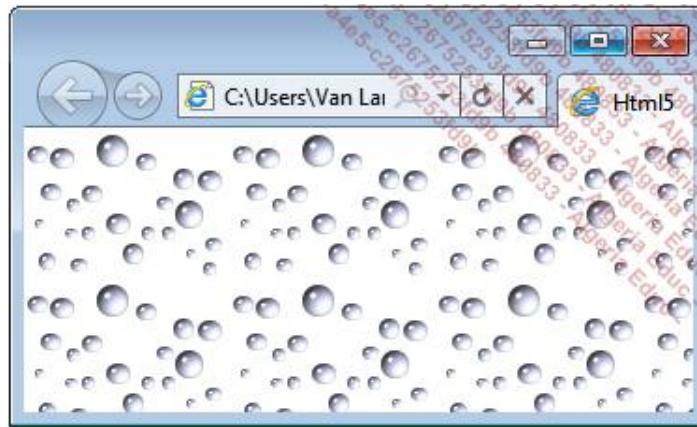
Commentaires

- On peut utiliser l'adressage relatif ou absolu pour indiquer l'adresse (URL) de l'image.
- Les images peuvent être au format GIF, JPEG ou PNG.
- La valeur `none` n'affiche aucune image d'arrière-plan.

Exemple

Mettre une image de fond à la page (présentation classique).

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
body { background-image: url(bulles.jpg); }
</style>
</head>
<body>
</body>
</html>
```



L'image bulles.jpg est disponible dans l'espace de téléchargement.

Répétition de l'image

Cette propriété donne le contrôle au concepteur quant à la répétition de l'image qui ne sera ainsi plus nécessairement disposée en mosaïque.

background-repeat:	repeat; repeat-x; repeat-y; no-repeat;
--------------------	---

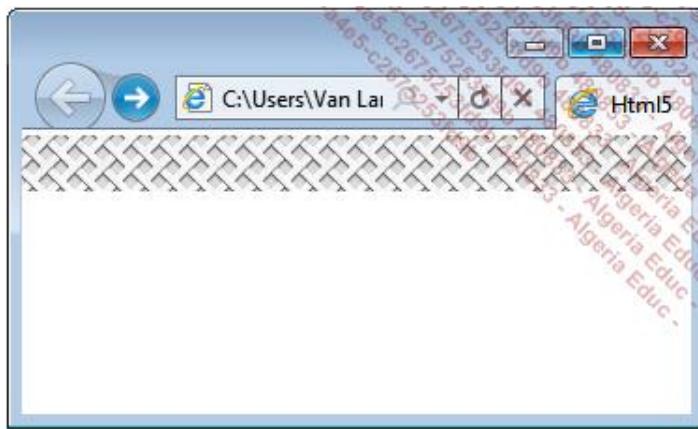
Commentaires

- Cette déclaration de style n'a d'effet que si une image de fond a été définie par la propriété `background-image: url(fichier_image)`.
- La valeur `repeat` répète l'image horizontalement et verticalement. Le résultat est des plus classiques et en tout point semblable à la déclaration `background-image: url(fichier_image)`.
- La valeur `repeat-x` répète l'image seulement horizontalement (selon l'axe des x). L'image forme ainsi une ligne horizontale dans le haut de l'élément.
- La valeur `repeat-y` répète l'image seulement verticalement (selon l'axe des y). L'image forme ainsi une ligne verticale le long du côté gauche de l'élément.
- La valeur `no-repeat` affiche l'élément une seule fois sans le répéter.
- Rien n'empêche de définir (par `background-color`) une couleur d'arrière-plan en même temps qu'une image d'arrière-plan. Ceci permettra certains effets assez créatifs. En outre, l'arrière-plan de couleur sera toujours affiché même si l'image n'est pas trouvée à l'adresse indiquée.

Exemple 1

Illustrons la valeur `repeat-x`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
body { background-image: url(gris.jpg);
       background-repeat: repeat-x; }
</style>
</head>
<body>
</body>
</html>
```

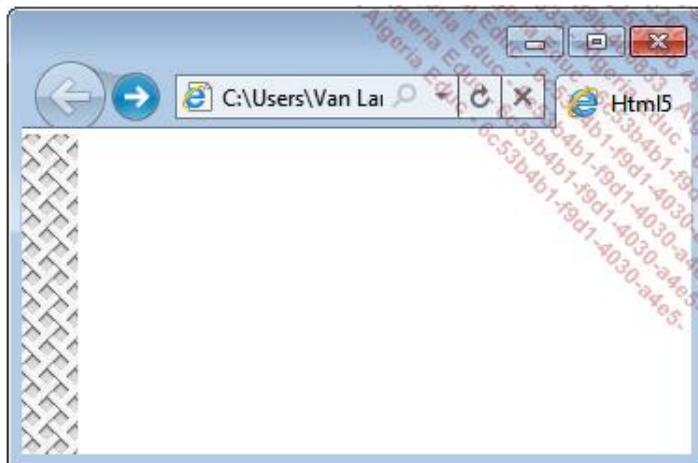


L'image gris.jpg est disponible dans l'espace de téléchargement.

Exemple 2

Illustrons la valeur repeat-y.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
body { background-image: url(gris.jpg);
       background-repeat: repeat-y;}
</style>
</head>
<body>
</body>
</html>
```

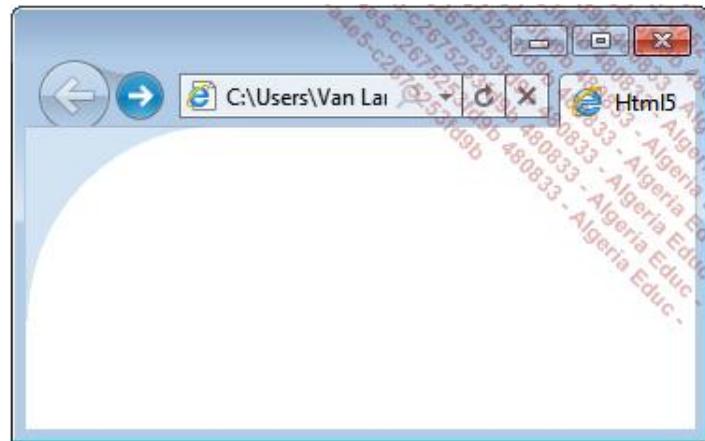


Exemple 3

Illustrons la valeur no-repeat.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
body { background-image: url(coin.gif);
       background-repeat: no-repeat;}
</style>
</head>
```

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



L'image coin.gif est disponible dans l'espace de téléchargement.

Positionnement de l'image

Cette propriété va permettre de positionner précisément l'image de fond d'un élément, au pixel près.

background-position:	valeur de longueur valeur de longueur; pourcentage pourcentage; left ou center ou right / top ou center ou bottom;
----------------------	--

Soit par exemple :

```
background-position: 50px 100px;  
background-position: 20% 50%;  
background-position: center center;
```

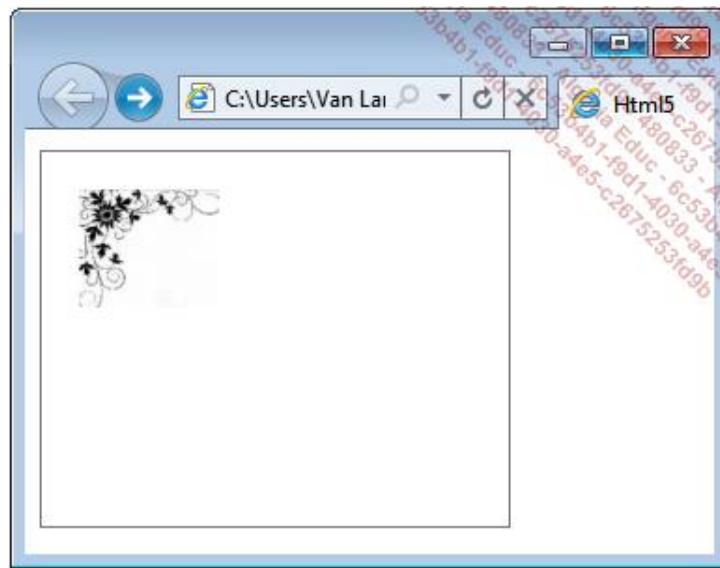
Commentaires

- Cette déclaration de style n'a de sens que si une image de fond a été définie par la déclaration background-image: url(fichier_image).
- Cette déclaration ne s'utilise généralement que lorsque l'image n'est pas répétée (background-norepeat).
- La première valeur a trait à la position par rapport au bord gauche (axe horizontal) et la seconde valeur à la position par rapport au bord supérieur (axe vertical). Les deux valeurs sont séparées par un espace.
- Les pourcentages respectent le même principe mais évidemment en pourcentage de la taille de l'élément parent.
- On peut ne donner qu'une valeur. Dans ce cas, celle-ci sera interprétée comme étant la valeur horizontale (la première valeur). La valeur verticale ou seconde valeur sera alors automatiquement définie par défaut à la valeur center ou 50%.
- Les valeurs en pourcentage et de longueur peuvent être combinées. Ainsi, background-position: 50px 30%; est correct.
- Les valeurs left (gauche), center (centre) ou right (droite) déterminent la position horizontale.
- Les valeurs top (haut), center (centre), bottom (bas) déterminent la position verticale.
- Les valeurs négatives sont acceptées.

Exemple 1

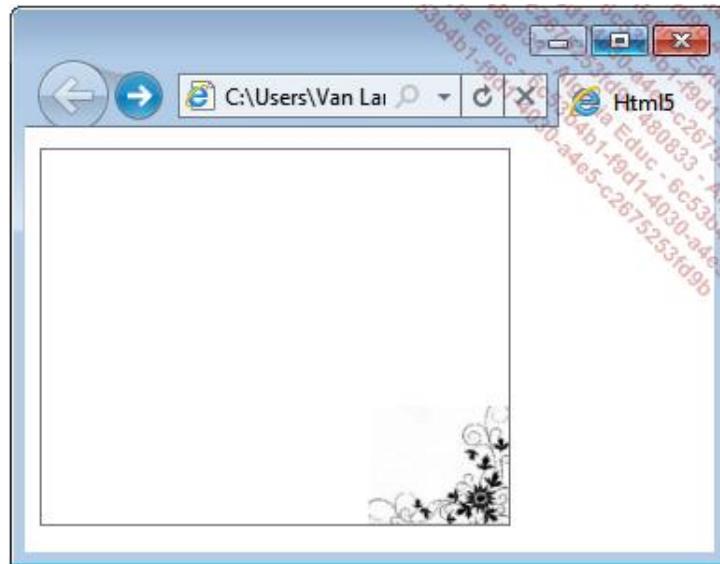
```
<!DOCTYPE html>  
<html lang="fr">  
<head>  
<title>Html5</title>  
<meta charset="iso-8859-1">  
<style type="text/css">  
div { width: 250px;  
      height: 200px;  
      border: 1px solid gray;  
      background-image: url(paper1.png);  
      background-repeat: no-repeat;  
      background-position: 20px 20px; }  
</style>  
</head>  
<body>  
<div>  
</div>
```

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



Exemple 2

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
div { width: 250px;
      height: 200px;
      border: 1px solid gray;
      background-image: url(paper3.png);
      background-repeat: no-repeat;
      background-position: bottom right; }
</style>
</head>
<body>
<div>
</div>
</body>
</html>
```



Défilement de l'image

Cette propriété permet de fixer une image de fond qui ne défile pas avec le reste de la page.

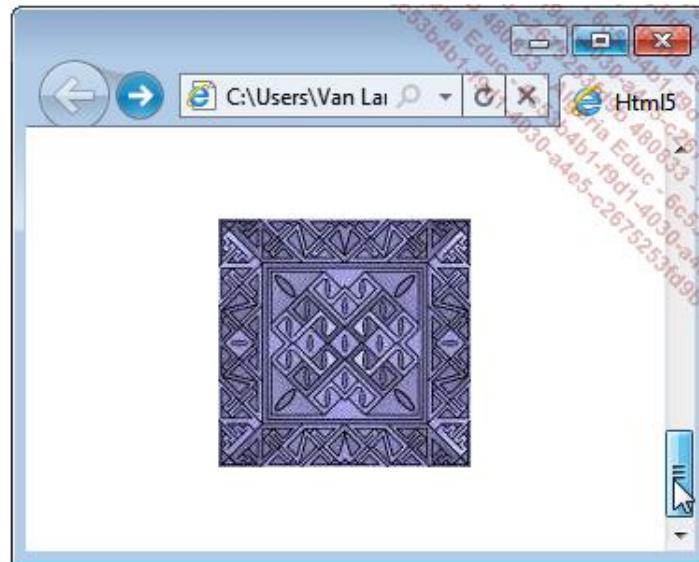
background-attachment:	scroll; fixed;
------------------------	-------------------

Commentaires

- La valeur `scroll` fait défiler l'image d'arrière-plan avec le contenu de la page.
- La valeur `fixed` laisse l'image d'arrière-plan fixe alors que le contenu de la page peut défiler.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
body { background-image: url(gris.jpg);
      background-repeat: no-repeat;
      background-position: center center;
      background-attachment: fixed; }
</style>
</head>
<body>
</body>
</html>
```



Écriture raccourcie

La propriété `background` permet de regrouper les différentes propriétés en une seule ligne en ne reprenant que les valeurs de celle-ci.

Comme les différentes propriétés et valeurs d'arrière-plan sont plutôt longues à encoder manuellement, ce raccourci permet de gagner du temps mais réduit peut-être la lisibilité du code.

```
background: url(format_image) white no-repeat right top fixed;
```

Cette seule déclaration correspond à :

```
background-image: url(xxx.gif);
background-color: white;
background-repeat: no-repeat;
background-position: right top;
background-attachment: fixed;
```

Commentaires

- Les différents attributs doivent être indiqués à la suite, séparés d'un espace.
- L'ordre n'a pas d'importance.
- Il n'est pas obligatoire de définir chaque propriété d'arrière-plan ; pour les propriétés non définies, les valeurs par défaut seront retenues.

Exemple

Le code de l'exemple de la section Défilement de l'image pourrait ainsi s'écrire :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
body { background: url(gris.jpg)no-repeat center center fixed; }
</style>
</head>
<body>
</body>
</html>
```

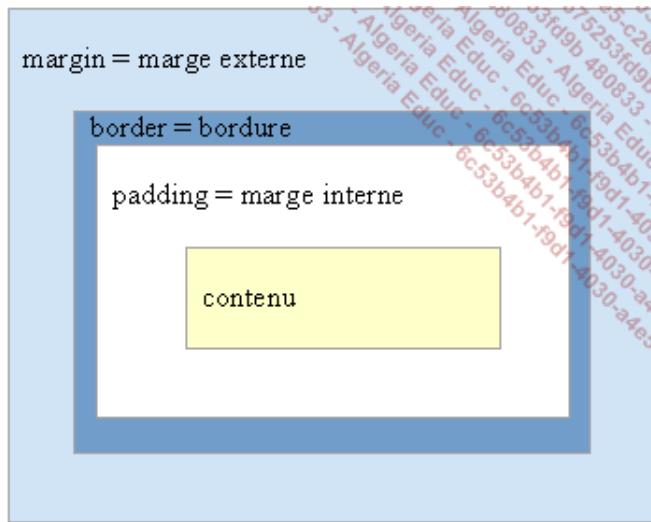
Concept du modèle de boîte

L'élément boîte est une notion importante en CSS. Cette notion, un peu abstraite, doit être correctement intégrée pour la suite de notre étude et tout particulièrement en vue du positionnement (chapitre Les propriétés d'affichage - Positionnement).

Cet élément boîte est aussi appelé un conteneur ou un bloc (*block*).

Le W3C définit un élément boîte comme une zone rectangulaire constituée :

- d'un contenu,
- d'une marge intérieure (*padding*),
- d'une bordure (*border*),
- et d'une marge extérieure (*margin*).



Précisons que :

- Le contenu peut être, par exemple, le texte d'un paragraphe.
- Celui-ci est entouré d'un espace allant du texte aux bordures du cadre, que l'on appellera la marge intérieure ou interne.
- Les bordures du cadre (facultatives).
- Un espace entourant le tout, appelé marge extérieure ou externe.

Les feuilles de style vont reprendre et étendre cette notion de boîte pour donner au concepteur le contrôle total de la marge intérieure et extérieure et de la bordure.

Avec les feuilles de style CSS, on pourra, par exemple, régler distinctement :

- Les quatre marges extérieures et ce dans toutes les directions,
- Les quatre bordures (dimension, style, couleur),
- Les quatre marges intérieures et ce dans toutes les directions,
- Les dimensions (largeur et hauteur) du contenu.

La source principale de confusion est que, après avoir précisé la largeur et/ou la hauteur du contenu, on n'a pas explicitement défini la dimension totale de l'élément boîte, qui comprend la dimension du contenu, la marge intérieure, la dimension de la bordure et la marge extérieure.

Ainsi, la propriété `width` appliquée au contenu (le petit rectangle central) n'est qu'un élément de la dimension de

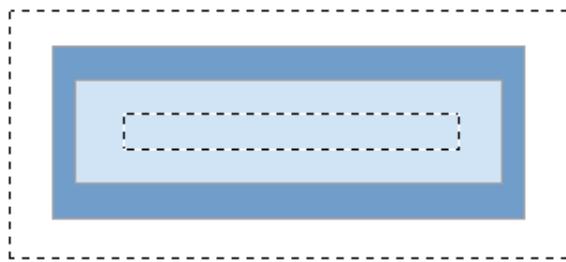
l'élément boîte, il faut y ajouter la marge intérieure, la bordure et la marge extérieure pour obtenir l'espace total réel de l'élément boîte.

C'est bien entendu la même chose pour la hauteur déterminée par la propriété `height`.

La dimension totale de l'élément boîte se calcule donc ainsi : contenu + marges intérieures + bordures + marges extérieures.

Soit une boîte avec les propriétés suivantes :

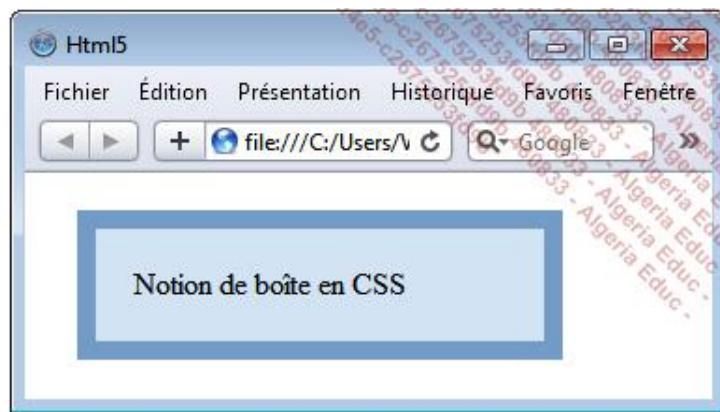
- largeur de contenu 200 pixels,
- marge intérieure 20 pixels,
- bordure d'une largeur de 10 pixels,
- marge extérieure 20 pixels.



La largeur de l'élément boîte est donc bien, dans notre exemple, de 300 pixels qui se décomposent ainsi (de gauche à droite) : 20 pixels de marge extérieure + 10 pixels pour la bordure + 20 pixels de marge intérieure + 200 pixels de contenu + 20 pixels de marge intérieure + 10 pixels de bordure + 20 pixels de marge extérieure. Le total est bien de 300 pixels.

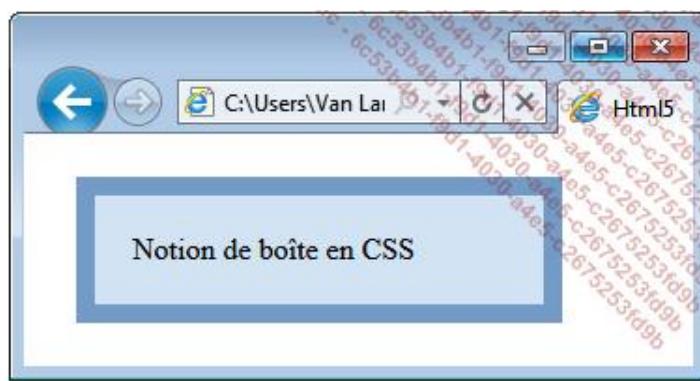
Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
p { width: 200px;
    border: 10px solid rgb(125,165,205);
    padding: 20px;
    margin: 20px;
    background-color: rgb(215,230,245); }
</style>
</head>
<body>
<p>Notion de boîte en CSS</p>
</body>
</html>
```



Commentaires

- Internet Explorer 9 avec le doctype du Html5 (`<!DOCTYPE html>`) affiche le modèle de boîte conformément aux spécifications du W3C.



- Cependant, sans le doctype ou avec un doctype erroné, Internet Explorer 9 passe en mode de compatibilité arrière (*quirks mode*) et reprend son modèle de boîte de Microsoft erroné. En effet, Microsoft a longtemps considéré que la marge intérieure et les bordures prennent place à l'intérieur de la zone de contenu. La largeur apparente d'une boîte sera alors égale à la seule largeur spécifiée pour le contenu à laquelle on ajoutera simplement les marges extérieures. La boîte précédente n'occupera donc à l'écran qu'une largeur de 240 pixels. Sa zone de contenu se trouvera réduite des dimensions de bordure et de remplissage, soit ici 60 pixels par rapport au modèle standard.



Tout ceci illustre bien l'importance du doctype.

Élément bloc ou en ligne

Nous avons déjà abordé précédemment la notion d'élément bloc ou en ligne.

La propriété de style `display` permet de redéfinir un élément en ligne comme étant un élément bloc et inversement.

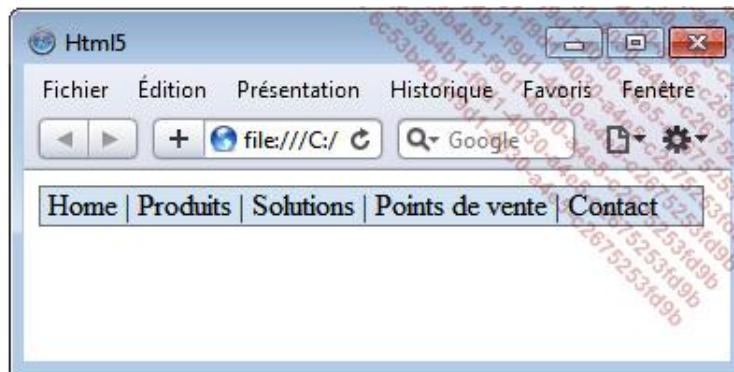
display:	inline; (en ligne)
	block; (de type bloc)

Cette propriété est très riche en possibilités de présentation et est souvent utilisée pour des feuilles de style CSS complexes.

Exemple 1

Les nouvelles balises sémantiques (chapitre "Les balises sémantiques et d'organisation") sont des éléments en ligne. Pour leur attribuer une fonction de présentation, il est impératif de les définir comme des éléments de type bloc. Appliquons ceci à la balise `<nav>`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
nav { display: block;
      width: 350px;
      border: 1px solid gray;
      background-color : rgb(215,230,245);
padding-left: 4px;}
a { text-decoration: none; }
</style>
</head>
<body>
<nav>
<a>Home</a> |
<a>Produits</a> |
<a>Solutions</a> |
<a>Points de vente</a> |
<a>Contact</a>
</nav>
</body>
</html>
```

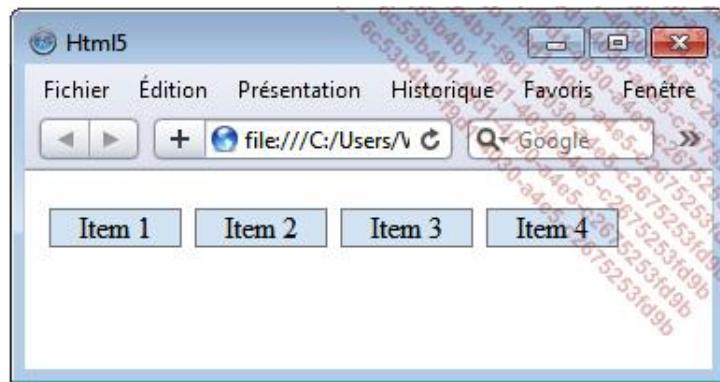


Exemple 2

Les items de listes sont des éléments de type bloc. Ils apparaissent d'ailleurs les uns en dessous des autres. Pourtant avec la propriété `display: inline`, il est possible de les afficher sur une ligne.

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

```
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
ul { list-style: none ;
      padding: 5px;;}
li { display: inline;
      border: 1px solid gray;
      background-color: rgb(215,230,245);
      text-align: center;
      margin-right: 3px;
      padding-right: 15px;
      padding-left: 15px;}
</style>
</head>
<body>
<ul>
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
<li>Item 4</li>
</ul>
</body>
</html>
```



LARGEUR ET HAUTEUR

Les propriétés de style `width` et `height` vont fixer respectivement la largeur et la hauteur de l'élément boîte.

<code>width:</code>	auto ou valeur de longueur ou pourcentage
<code>height:</code>	auto ou valeur de longueur ou pourcentage

Commentaires

- Les valeurs `width` et `height` spécifient la largeur et la hauteur du contenu (sans marges intérieures, sans bordures et sans marges extérieures).
- La valeur `auto` ajuste automatiquement la dimension au contenu.
- La valeur de longueur spécifie une valeur fixe.
- Le pourcentage spécifie une valeur relative.

Notons qu'il existe en CSS2 des propriétés pour déterminer la taille minimale et maximale d'un élément de type bloc ou d'un élément remplacé dynamiquement par du JavaScript.

<code>max-height:</code>	Fixe la hauteur maximale
<code>min-height:</code>	Fixe la hauteur minimale
<code>max-width:</code>	Fixe la largeur maximale
<code>min-width:</code>	Fixe la largeur minimale

La propriété `max-height` est généralement utilisée conjointement avec la propriété `min-height` pour fixer une marge de manœuvre pour l'affichage des éléments concernés. Ceci vaut aussi pour les propriétés `max-width` et `min-width`. À l'aide de ces propriétés CSS `max-width` et `min-width`, vous aurez ainsi la possibilité de contrôler la largeur de votre design de façon à ce qu'il s'adapte aux résolutions d'écran des visiteurs.

Ainsi, si la largeur dépasse celle spécifiée par `max-width`, le navigateur recalcule la largeur de l'élément en se basant sur la valeur de `max-width`. De même, si la largeur est inférieure à celle de `min-width`, le navigateur recalcule la largeur en fonction de la valeur déterminée dans `min-width`. Le navigateur procède de même pour les propriétés `max-height` et `min-height`.

Il faut noter que les propriétés `max-height`, `min-height`, `max-width` et `min-width` n'incluent pas les marges intérieures, les bordures et les marges extérieures.

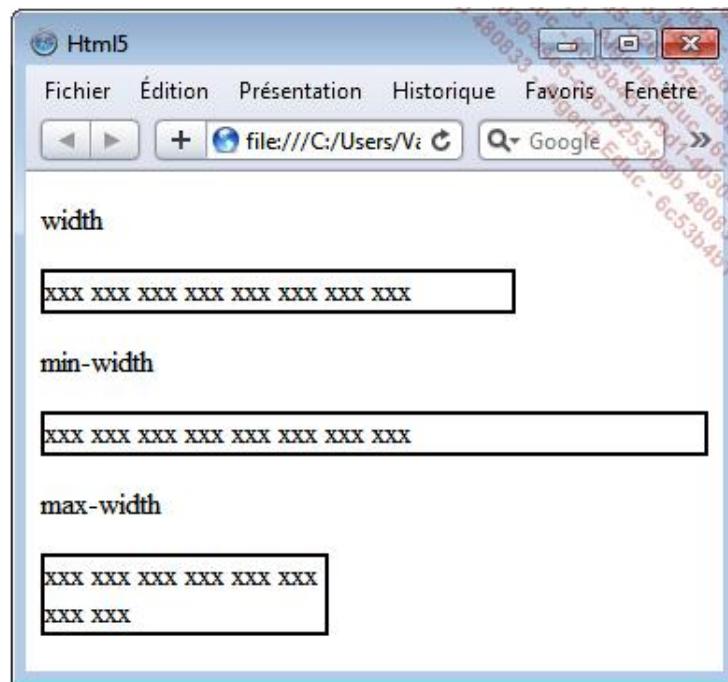
Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<p>width</p>
<div style= "border: 2px solid black;
              width: 250px;">
xxx xxx xxx xxx xxx xxx xxx
</div>
<p>min-width</p>
```

```

<div style="border: 2px solid black;
            min-width: 250px;">
    xxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
</div>
<p>max-width</p>
<div style="border: 2px solid black;
            width: 250px;
            max-width: 150px;">
    xxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
</div>
</body>
</html>

```



À la première division, la largeur est fixée par `width: 250px`. À la seconde, celle-ci est déterminée par `min-width: 250px`. Comme cette valeur minimale n'est pas atteinte, la division prend toute la largeur de la fenêtre. À la dernière division, la largeur répond à la valeur maximale `max-width: 150px`.

Marges externes

Cette propriété permet de définir la marge extérieure de l'élément boîte.

margin:	auto ou
margin-top:	valeur de longueur ou pourcentage
margin-right:	
margin-bottom:	
margin-left:	

Commentaires

- La valeur `auto` laisse la main au navigateur pour afficher les marges externes.
- Une valeur de longueur définit précisément les marges extérieures.
- Un pourcentage définit la longueur par rapport à l'élément parent.
- La marge extérieure de chaque côté de l'élément boîte peut également être définie individuellement à l'aide des propriétés `margin-top`, `margin-right`, `margin-bottom` et `margin-left`. Les directions `top`, `right`, `bottom`, `left` désignent respectivement la bordure supérieure, droite, inférieure et gauche. Ces propriétés sont des spécifications CSS2.
- On peut raccourcir l'écriture des propriétés de marge externe `margin-top`, `margin-right`, `margin-bottom` et `margin-left`, en désignant une, deux, trois ou quatre valeurs à la propriété `margin`.

Si une seule marge externe est indiquée, elle sera appliquée aux quatre côtés.

Avec deux marges, la première sera appliquée aux côtés supérieurs et inférieurs.

La seconde aux côtés latéraux (droite et gauche).

Avec trois marges, la première sera appliquée au côté supérieur, la seconde aux côtés latéraux et la troisième au côté inférieur.

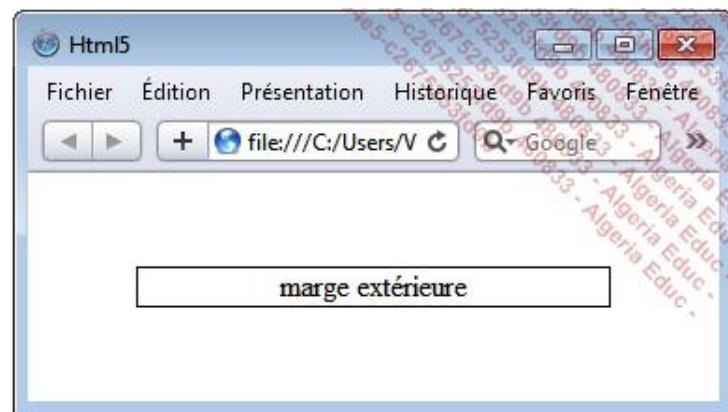
Avec quatre marges, celles-ci sont attribuées dans le sens des aiguilles d'une montre en partant du côté supérieur, soit côtés supérieur, droit, inférieur et gauche.

- Les marges peuvent être définies avec des valeurs négatives, ce qui peut engendrer des effets visuels intéressants, mais ces valeurs négatives risquent d'être interprétées différemment selon les navigateurs.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.body { margin: 0px; }
.bloc { border: 1px solid black;
        margin: 20px 20px 0px 20px;
        text-align: center; }
</style>
</head>
<body>
<div class="bloc">
marge extérieure
</div>
```

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



Marges internes

Cette propriété permet de définir la marge intérieure de l'élément boîte.

padding:	auto ou
padding-top:	valeur de longueur ou
padding-right:	pourcentage
padding-bottom:	
padding-left:	

Commentaires

- La valeur `auto` laisse la main au navigateur pour afficher les marges internes.
- Une valeur de longueur définit précisément les marges intérieures.
- Un pourcentage définit la longueur par rapport à l'élément parent.
- La marge intérieure de chaque côté de l'élément boîte peut également être définie individuellement à l'aide des propriétés `padding-top`, `padding-right`, `padding-bottom` et `padding-left`. Les directions `top`, `right`, `bottom`, `left` désignent respectivement la bordure supérieure, droite, inférieure et gauche.
- On peut raccourcir l'écriture des propriétés de marge interne `padding-top`, `padding-right`, `padding-bottom` et `padding-left`, en désignant une, deux, trois ou quatre valeurs à la propriété `padding`.

Si une seule marge interne est indiquée, elle sera appliquée aux quatre côtés.

Avec deux marges, la première sera appliquée aux côtés supérieurs et inférieurs. La seconde aux côtés latéraux (droite et gauche).

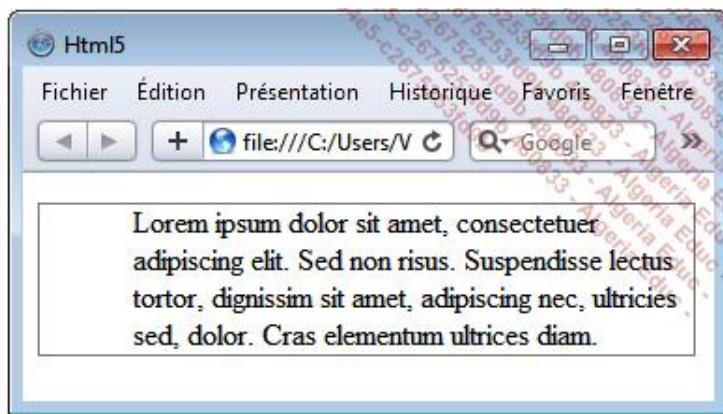
Avec trois marges, la première sera appliquée au côté supérieur, la seconde aux côtés latéraux et la troisième au côté inférieur.

Avec quatre marges, celles-ci sont attribuées dans le sens des aiguilles d'une montre en partant du côté supérieur, soit côtés supérieur, droit, inférieur et gauche.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.marge { width: 300px;
          padding-left: 50px;
          border: solid 1px gray; }

</style>
</head>
<body>
<div class="marge">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non
risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing
nec, ultricies sed, dolor. Cras elementum ultrices diam.
</div>
</body>
</html>
```



Couleur de la bordure

Ces propriétés de style permettent de définir la couleur des quatre bordures de l'élément boîte ou de chaque bordure prise individuellement.

border-color:	nom de couleur
border-top-color:	notation hexadécimale sous sa forme #xxxxxx
border-right-color:	notation hexadécimale abrégée sous la forme #xxx
border-bottom-color:	notation en mode RVB avec des entiers de 0 à 255
border-left-color:	notation en mode RVB avec des % de 0 à 100 ou notation en mode RVBa où a est compris entre 0 et 1.
	notation en mode HSL
	notation en mode HSLa où a est compris entre 0 et 1.
	transparent

Commentaires

- La propriété CSS1 `border-color` définit la couleur (unique) des quatre côtés de l'élément boîte. Ainsi, avec la propriété `border-color: red`, les quatre bordures auront la couleur rouge.
- La couleur ne peut être appliquée que si le type de la bordure (`border-style`) et l'épaisseur de la bordure (`border-width`) ont été définis, sans quoi la bordure sera inexistante.
- Les couleurs de chaque côtés de l'élément boîte peuvent également être définies individuellement à l'aide des propriétés `border-top-color`, `border-right-color`, `border-bottom-color` et `border-left-color`. Les directions `top`, `right`, `bottom`, `left` désignent respectivement la bordure supérieure, droite, inférieure et gauche.
- On peut faire l'impasse sur l'écriture (un peu longue) des propriétés `border-top-color`, `border-right-color`, `border-bottom-color` et `border-left-color`, en désignant une, deux, trois ou quatre couleurs à la propriété `border-color`.

Si une couleur est indiquée, elle sera appliquée aux quatre côtés.

Avec deux couleurs, la première sera appliquée aux côtés supérieurs et inférieurs. La seconde aux côtés latéraux (droite et gauche).

Avec trois couleurs, la première sera appliquée au côté supérieur, la seconde aux côtés latéraux et la troisième au côté inférieur.

Avec quatre couleurs, celles-ci sont attribuées dans le sens des aiguilles d'une montre en partant du côté supérieur, soit côtés supérieur, droit, inférieur et gauche.

- Avec la valeur `transparent`, la couleur des bordures est transparente mais celles-ci existent bel et bien.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
h1 { border-style: solid;
      border-width: 7px;
      border-top-color: rgb(215,230,245);
      border-bottom-color: rgb(215,230,245);
      border-left-color: rgb(125,165,205);
      border-right-color: rgb(125,165,205);
```

```
    text-align: center;}  
</style>  
</head>  
<body>  
<p></p>  
<h1>Html5 + CSS3</h1>  
</body>  
</html>
```



Épaisseur de la bordure

Cette propriété permettra de définir l'épaisseur des quatre bordures simultanément ou de chaque bordure prise individuellement.

border-width:	valeur de longueur;
border-top-width:	thin;
border-right-width:	medium;
border-bottom-width:	thick;
border-left-width:	

Commentaires

- La valeur `thin` intervient pour une bordure fine, `medium` pour une bordure moyenne et `thick` pour une bordure épaisse. L'interprétation de l'épaisseur à partir de ces mots-clés varie d'un navigateur à l'autre.
- L'épaisseur ne sera affichée que si le type de bordure (`border-style`) a été défini.
- L'épaisseur de chaque côté peut également être définie individuellement à l'aide des propriétés `border-top-width`, `border-right-width`, `border-bottom-width` et `border-left-width`. Les directions `top`, `right`, `bottom`, `left` désignent respectivement la bordure supérieure, droite, inférieure et gauche.
- On peut faire l'impasse sur l'écriture (un peu longue) des propriétés `border-top-width`, `border-right-width`, `border-bottom-width` et `border-left-width`, en désignant une, deux, trois ou quatre épaisseurs à la propriété `border-width`.

Si une épaisseur est indiquée, elle sera appliquée aux quatre côtés.

Avec deux épaisseurs, la première sera appliquée aux côtés supérieurs et inférieurs. La seconde aux côtés latéraux (droite et gauche).

Avec trois épaisseurs, la première sera appliquée au côté supérieur, la seconde aux côtés latéraux et la troisième au côté inférieur.

Avec quatre épaisseurs, celles-ci sont attribuées dans le sens des aiguilles d'une montre en partant du côté supérieur, soit côtés supérieur, droit, inférieur et gauche.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
h1 { border-style: solid;
      border-color: rgb(125,165,205);
      border-width: thin thick;
      text-align: center;}
</style>
</head>
<body>
<h1>Html5 + CSS3</h1>
</body>
</html>
```



Style de la bordure

Cette propriété CSS apporte une variété de bordures par rapport au seul style de bordures du Html.

border-style: border-top-style: border-right-style: border-bottom-style: border-left-style:	solid ou dashed ou dotted ou double ou groove ou ridge ou inset ou outset ou hidden ou none;
---	--

Commentaires

- Il convient d'illustrer ces différentes bordures, qui pour la plupart sont nouvelles dans le domaine de la conception de pages Web. Ainsi, le mot clé affiche :

`solid` : un trait plein,

`dashed` : des tirets,

`dotted` : des pointillés,

`double` : des doubles traits pleins,

`groove` : effet 3D gravé dans la page (opposé de `ridge`),

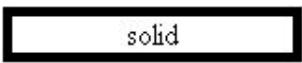
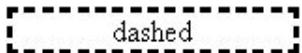
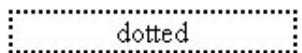
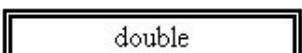
`ridge` : effet 3D sortant de la page (opposé de `groove`),

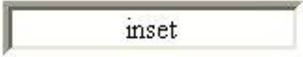
`inset` : bordures rentrantes, incrustées dans la page (inverse de `outset`),

`outset` : bordures sortantes extrudées de la page (inverse de `inset`),

`hidden` : pas de bordure (influe sur la bordure adjacente),

`none` : équivaut à `border-width: 0px` soit, pas de bordure.

solid	
dashed	
dotted	
double	
groove	
ridge	

inset	
outset	

- Pour `double`, la largeur définie par `border-width` correspondra à la taille des deux traits et de l'espace entre ceux-ci.
- Le style de bordure ne sera affiché que si l'épaisseur de bordure (`border-width`) a été définie.
- Le style de chaque côté peut également être défini individuellement à l'aide des propriétés `border-top-style`, `border-right-style`, `border-bottom-style` et `border-left-style`. Les directions `top`, `right`, `bottom`, `left` désignent respectivement la bordure supérieure, droite, inférieure et gauche.
- On peut faire l'impasse sur l'écriture (un peu longue) des propriétés `border-top-style`, `border-right-style`, `border-bottom-style` et `border-left-style`, en désignant une, deux, trois ou quatre épaisseurs à la propriété `border-style`.

Si le style est indiqué, il sera appliqué aux quatre côtés.

Avec deux styles, le premier sera appliqué aux côtés supérieurs et inférieurs. Le second aux côtés latéraux (droite et gauche).

Avec trois styles, le premier sera appliqué au côté supérieur, le second aux côtés latéraux et le troisième au côté inférieur.

Avec quatre styles, ceux-ci sont attribués dans le sens des aiguilles d'une montre en partant du côté supérieur, soit côtés supérieur, droit, inférieur et gauche.

Le résultat au niveau des raccords n'est pas garanti.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
h2 { border-style: double;
      border-width: 10px;
      border-color: gray;
      text-align: center; }
</style>
</head>
<body>
<h2>Une bordure qui a du style</h2>
</body>
</html>
```



Écriture raccourcie de bordure

Cette propriété de style permet de noter de façon raccourcie les différentes propriétés permettant de définir les bordures d'un élément, soit `border-color`, `border-style` et `border-width`.

Exemple

```
border: red double 5px;
```

Commentaires

- Ceci ne vaut que dans le cas où les quatre bordures sont identiques.
- Les valeurs de ces différentes propriétés sont regroupées et peuvent être énoncées à la suite, peu importe l'ordre, mais séparées par des espaces.
- Pour une compatibilité optimale des navigateurs, il est recommandé que les trois propriétés soient renseignées.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
h1 { border: dotted 1px black;
      text-align: center; }
</style>
</head>
<body>
<h1>Raccourci border</h1>
</body>
</html>
```



Notion de pseudo-classe

Il existe, dans un document Html, des éléments non définis par une balise ou par un attribut qui pourraient se révéler intéressants pour leur appliquer un effet de style comme, par exemple, le lien visité (attribut absent en Html5) ou la première lettre d'un paragraphe.

Le W3C a introduit, pour certains de ces éléments non référencés de la page, la notion de pseudo-classes. On parle aussi de pseudo-style ou pseudo-élément.

La syntaxe des pseudo-classes est :

```
sélecteur:pseudo-classe { déclaration(s) de style }
```

Soit le sélecteur, double point, le nom de la pseudo-classe suivi par la déclaration de style entre accolades.

Les pseudo-classes peuvent également être utilisées avec des classes. Elles se noteront alors :

```
sélecteur.classe:pseudo-classe { déclaration(s) de style }
```

Soit le sélecteur, point, le nom de la classe, double point, le nom de la pseudo-classe suivi par la déclaration de style entre accolades.

Tout comme pour les classes, les pseudo-classes sont définies à l'intérieur de balises `<style type="text/css"> ... </style>` dans le `head` de la page ou dans une feuille de style externe.

Pseudo-classes de lien

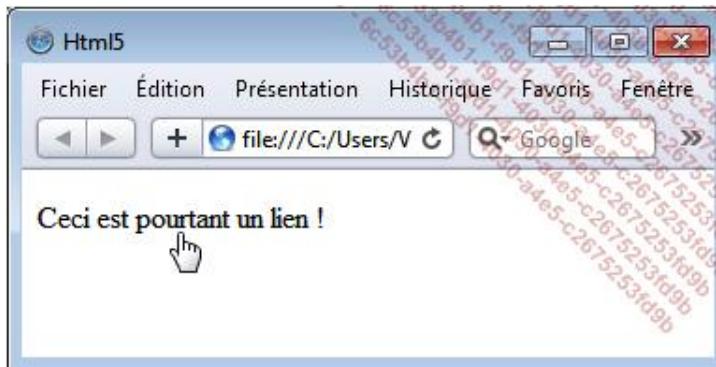
1. a:link

La pseudo-classe `a:link` permet de définir l'apparence d'un lien qui n'a pas encore été sélectionné.

On ne peut pas manquer ce grand classique des feuilles de style qui consiste à enlever le soulignement des liens par défaut.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
a:link { text-decoration: none;
          color: black; }
</style>
</head>
<body>
<p><a href="test.htm">Ceci est pourtant un lien !</a></p>
</body>
</html>
```



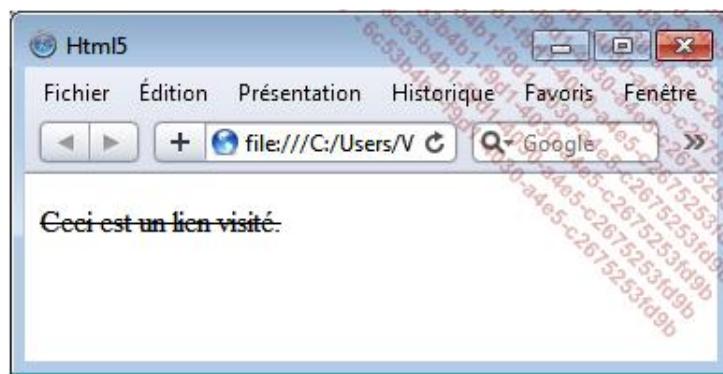
Notons que la déclaration de style `a {text-decoration: none;}` aurait été possible. Elle aurait enlevé le soulignement à tous les états du lien (`link`, `active`, `hover` et `visited`).

2. a:visited

La pseudo-classe `a:visited` permet de définir l'apparence d'un lien qui a déjà été visité.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
a:visited {text-decoration: line-through;
           color: black; }
</style>
</head>
<body>
<p><a href="test.htm">Ceci est un lien visité.</a></p>
</body>
```

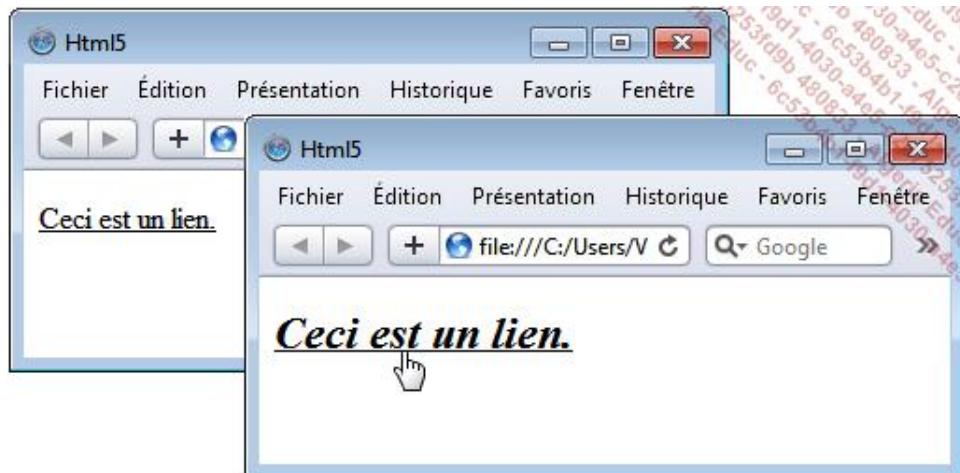


3. a:hover

La pseudo-classe a:hover permet de définir l'apparence d'un lien au passage du curseur de la souris.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
a:hover { font-weight: bold;
            font-style: italic;
            font-size: 1.5em; }
</style>
</head>
<body>
<p><a href="test.htm">Ceci est un lien.</a></p>
</body>
</html>
```



4. a:active

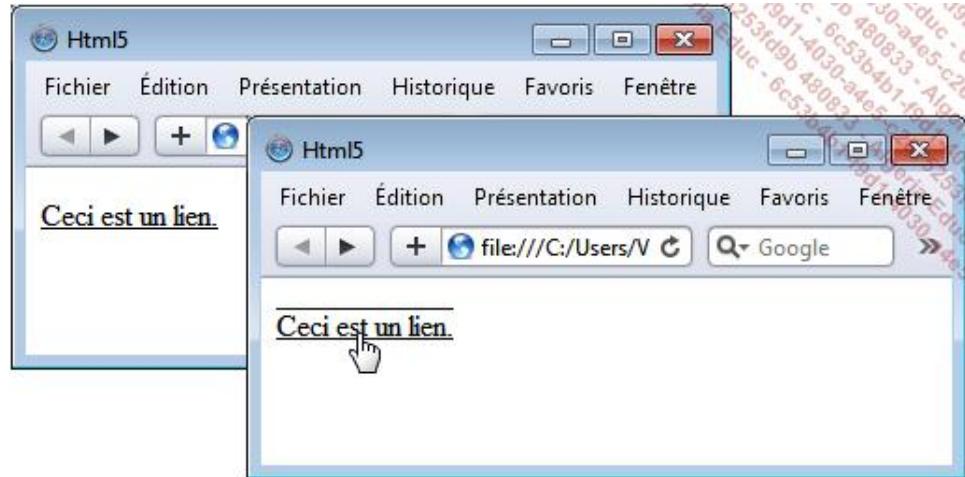
La pseudo-classe a:active permet de définir l'apparence d'un lien au moment où il est cliqué par le visiteur.

Exemple

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
a:active { text-decoration: underline overline; }
</style>
</head>
<body>
<p><a href="test.htm">Ceci est un lien.</a></p>
</body>
</html>

```



Commentaires

- La pseudo-classe :active s'utilise généralement conjointement avec :visited, :hover et :link.

Pour obtenir un résultat correct, il est indispensable d'énoncer les apparences de lien dans l'ordre suivant :

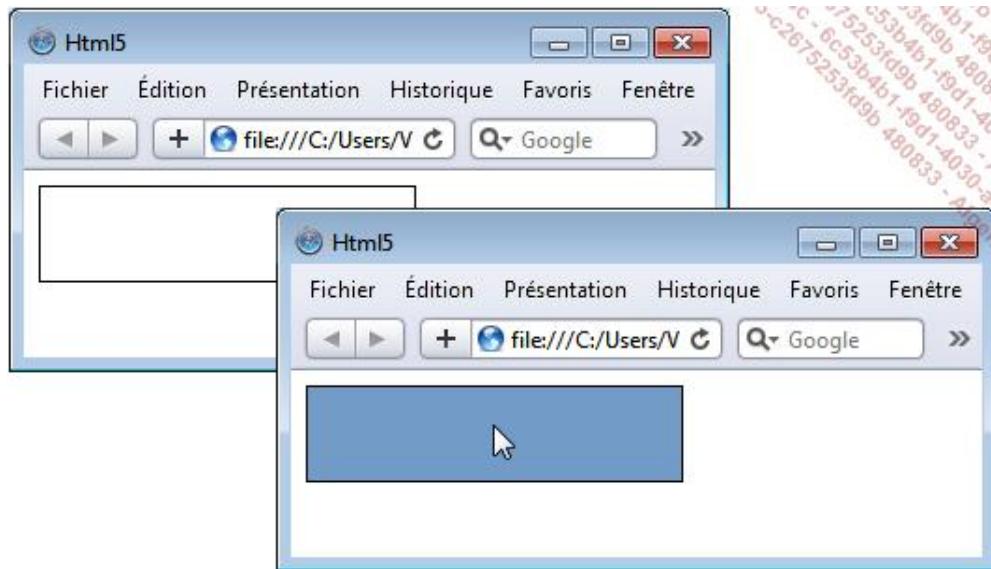
- :link
- :visited
- :hover
- :active

- La pseudo-classe :hover peut s'utiliser sur d'autres éléments que les liens.

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<style type="text/css">
div { width: 200px; height: 50px;
      border: 1px solid black; }
div:hover { background-color: rgb(125,165,205); }
</style>
</head>
<body>
<div></div>
</body>
</html>

```



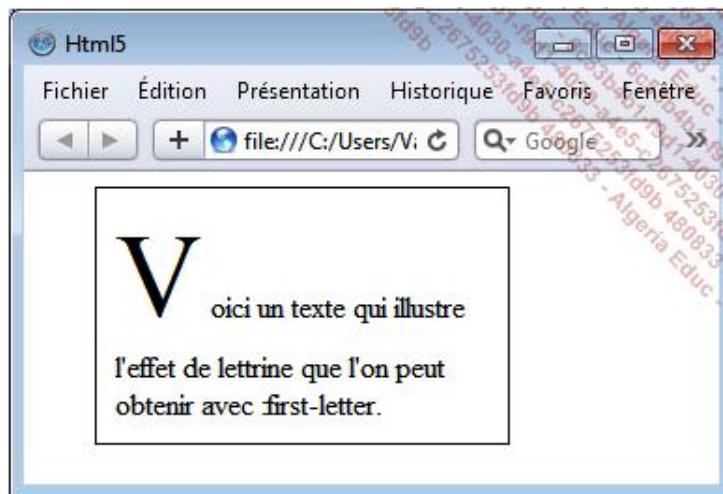
Pseudo-classes de paragraphe

1. :first-letter

Cette pseudo-classe `:first-letter` permet de définir l'apparence de la première (et uniquement la première) lettre d'un paragraphe. On crée ainsi un effet de lettrine.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#box { border: 1px solid black;
        margin-left: 30px;
        width: 200px;
        padding: 10px;}
.lettrine:first-letter { margin-right: 5px;
                        font-size: 4em;}
</style>
</head>
<body>
<div id="box">
<div class="lettrine">Voici un texte qui illustre l'effet de
lettrine que l'on peut obtenir avec :first-letter.</div>
</div>
</div>
</body>
</html>
```



2. :first-line

Cette pseudo-classe `:first-line` permet de définir l'apparence de la première (et uniquement la première) ligne d'un paragraphe.

La longueur de la première ligne est variable, elle dépendra de la largeur de l'élément boîte, de la taille de police, etc.

Exemple

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

```

<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.box { border: 1px solid black;
       margin-left: 30px;
       width: 200px;
       padding:10px; }
.ligne1:first-line { font-variant: small-caps;
                      font-weight: bold; }
</style>
</head>
<body>
<div class="box">
<p class="ligne1">Voici un texte qui illustre l'effet de style que
l'on peut obtenir avec :first-line.</p>
</div>
</body>
</html>

```



-  On voit apparaître en CSS3 la notation `::first-letter` et `::first-line` afin de ne pas confondre ces pseudo-classes avec les pseudo-classes de sélection, par exemple `:first-child` ou `:first-of-type` (voir chapitre Les feuilles de style CSS3 - Les sélecteurs CSS3).

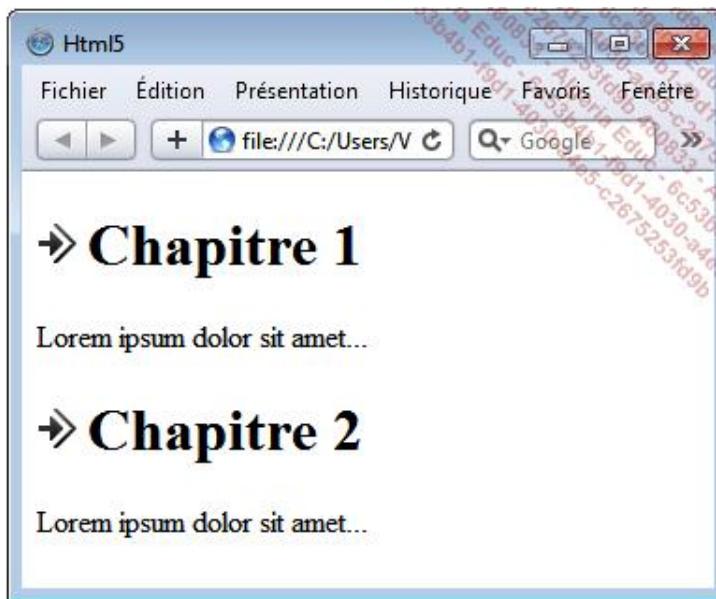
Insérer un contenu

1. :before

La pseudo-classe `:before`, associée à la propriété `content`, permet d'insérer un contenu avant un élément.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
h1:before {content:url(arrow.gif);}
</style>
</head>
<body>
<h1>Chapitre 1</h1>
<p>Lorem ipsum dolor sit amet...</p>
<h1>Chapitre 2</h1>
<p>Lorem ipsum dolor sit amet...</p>
</body>
</html>
```



La propriété de style `content` ajoute une flèche (arrow.gif) devant chaque titre de niveau `<h1>` (`h1:before`).

L'image arrow.gif est disponible dans l'espace de téléchargement consacré à cet ouvrage.

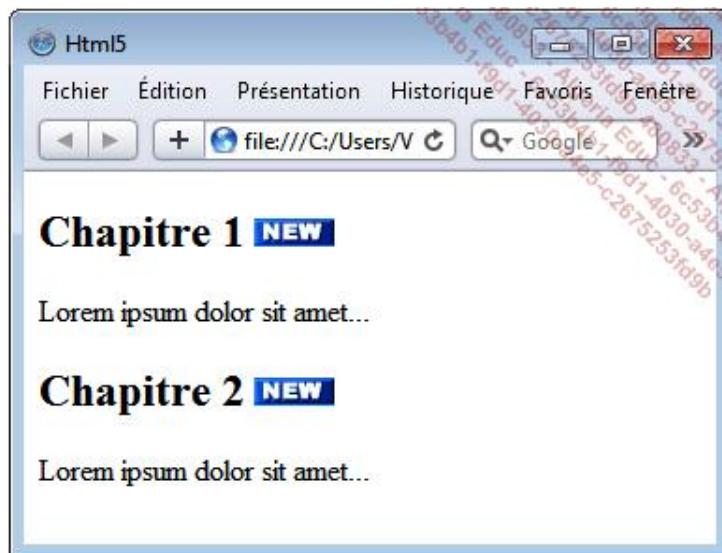
2. :after

La pseudo-classe `:after`, associée à la propriété `content`, permet d'insérer un contenu après un élément.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
```

```
<meta charset="iso-8859-1">
<style type="text/css">
h2:after {content:url(new.gif);}
</style>
</head>
<body>
<h2>Chapitre 1</h2>
<p>Lorem ipsum dolor sit amet...</p>
<h2>Chapitre 2</h2>
<p>Lorem ipsum dolor sit amet...</p>
</body>
</html>
```



La propriété de style `content` ajoute une flèche (`new.gif`) après chaque titre de niveau `<h2>` (`h2:after`).

L'image `new.gif` est disponible dans l'espace de téléchargement consacré à cet ouvrage.

-
- On voit apparaître en CSS3 la notation `::before` et `::after` afin de ne pas confondre ces pseudo-classes avec les pseudo-classes de sélection (voir chapitre Les feuilles de style CSS3 - Les sélecteurs CSS3).
-

Préambule

Dans le passé, le positionnement exact d'un élément en Html tournait rapidement au cauchemar et il ne pouvait généralement se réaliser que grâce à un puzzle de tableaux imbriqués.

Les feuilles de style CSS proposent maintenant des outils pour positionner, au pixel près, un élément dans votre document Html. Éditées à l'origine sous le terme CSS-P, ces propriétés de positionnement ont été reprises sous la spécification CSS2.

Ces éléments ainsi positionnés par les feuilles de style CSS pourront éventuellement être rendus dynamiques par l'apport du JavaScript et du Dhtml.

Positionnement

Un élément peut se positionner de quatre façons différentes.

1. Positionnement statique

C'est le positionnement normal de l'élément selon la façon habituelle de procéder du navigateur.

Le positionnement statique se détermine par :

position: static;	
-------------------	--

Le concepteur n'a donc pas le contrôle. L'élément ne peut être positionné ou repositionné et sa visibilité ne peut être modifiée. Il n'est également pas possible d'utiliser du JavaScript pour changer la position de l'élément.

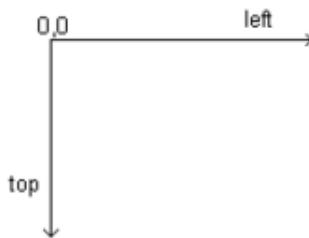
2. Positionnement relatif

C'est le positionnement d'un élément par rapport à sa position normale ou statique.

Cet élément reste dans le flux des données mais est en quelque sorte excentré par rapport à sa position normale.

La position est définie par les coordonnées (x,y) où :

- x est la distance par rapport au bord gauche de l'élément parent ou de la fenêtre du navigateur (axe horizontal). Ainsi, `left` détermine la distance entre la gauche de l'élément et la gauche de la page et `right`, la distance entre la droite de l'élément et la droite de la page.
- y est la distance par rapport au bord supérieur de l'élément parent ou de la fenêtre du navigateur (axe vertical). Ainsi `top` détermine la distance entre le bord supérieur de l'élément et le bord supérieur de la page et `bottom`, la distance entre le bord inférieur de l'élément et le bord inférieur de la page.



Le positionnement relatif se détermine par :

position: relative;	left: valeur ou %; top: valeur ou %; right: valeur ou %; bottom: valeur ou %;
---------------------	--

Commentaires

- En pratique, une seule spécification sur l'axe horizontal (`left` ou `right`) et une seule spécification sur l'axe vertical (`top` ou `bottom`) suffisent.
- Les valeurs de `top`, `left`, `right` et `bottom` peuvent être négatives.

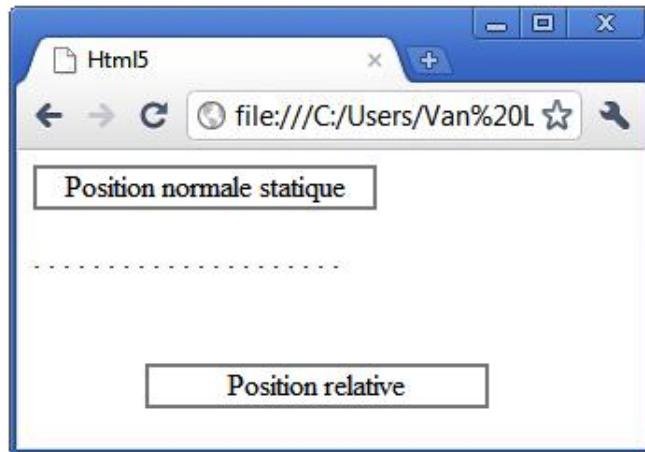
Exemple

<!DOCTYPE html>

```

<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.normale { width: 180px;
            border: 2px solid gray;
            text-align: center;}
.relative { position: relative;
            top: 30px;
            left: 60px;
            width: 180px;
            border: 2px solid gray;
            text-align: center;}
</style>
</head>
<body>
<div class="normale">
Position normale statique
</div>
<p>. . . . . . . . . . . . . . . .</p>
<div class="relative">
Position relative
</div>
</body>
</html>

```



3. Positionnement absolu

Le positionnement absolu crée un élément indépendant du reste du document. Les éléments définis en position absolue sont retirés du flux normal et se positionnent à l'emplacement exact défini par le concepteur.

La position est définie par les coordonnées (x,y) où :

- x est la distance par rapport au bord gauche de l'élément parent ou de la fenêtre du navigateur (axe horizontal). Ainsi, `left` détermine la distance entre la gauche de l'élément et la gauche de la page et `right`, la distance entre la droite de l'élément et la droite de la page.
- y est la distance par rapport au bord supérieur de l'élément parent ou de la fenêtre du navigateur (axe vertical). Ainsi `top` détermine la distance entre le bord supérieur de l'élément et le bord supérieur de la page et `bottom`, la distance entre le bord inférieur de l'élément et le bord inférieur de la page.

Le positionnement absolu se détermine par :

<code>position: absolute;</code>	<code>left: valeur ou %;</code>
	<code>top: valeur ou %;</code>
	<code>right: valeur ou %;</code>

bottom: valeur ou %;

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.absolu { position: absolute;
    left: 50px; top: 50px;
    width: 180px;
    border: 2px solid black;
    text-align: center;}
</style>
</head>
<body>
<div class="absolu">
Position absolue
</div>
</body>
</html>
```



Commentaires

- Le positionnement absolu n'est pas sans risque pour l'affichage correct de la page sous d'autres résolutions d'écran que celle utilisée pour la conception car l'élément ainsi positionné peut s'afficher en empiétant sur le flux normal des autres données de la page.
- Il faut se rappeler que les navigateurs ajoutent par défaut une marge au corps de la page Html et que cette marge par défaut varie d'un navigateur à l'autre. Il est ainsi prudent en cas de positionnement de spécifier les marges de la balise <body>.

4. Positionnement fixe

Le positionnement fixe crée aussi un élément indépendant dont on peut définir la position. Ici l'élément reste fixe même lorsque l'on fait défiler le document.

position: fixed;

Ce positionnement fixe a longtemps été boudé par les navigateurs. Il est à présent parfaitement intégré dans les navigateurs de notre étude.

Exemple

```
<!DOCTYPE html>
```

```
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.fixe { position: fixed;
         top: 50%;
         left: 50px;
         width: 180px;
         border: 2px solid black;
         text-align: center; }

</style>
</head>
<body>
<div class="fixe">
Position fixe
</div>
</body>
</html>
```



Flottement

La propriété `float` retire un élément boîte du flux normal pour la placer le plus à droite ou le plus gauche possible dans son élément parent, soit son conteneur.

float:	right; ou left; ou none;
--------	--------------------------------

Commentaires

- La valeur `right` aligne l'élément concerné à droite, poussant les autres éléments à s'aligner à gauche.
- La valeur `left` aligne l'élément concerné à gauche, poussant les autres éléments à s'aligner à droite.
- La valeur `none` ne spécifie rien et rend la main au navigateur.
- La position `float` ne peut s'appliquer en cas de positionnement absolu.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<div style="float:right;">

</div>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam. Maecenas ligula massa, varius a, semper congue, euismod non,
mi.</p>
</div>
</body>
</html>
```



L'élément conteneur est la division box. La propriété de style `float: right;` force l'image à se positionner à droite dans le conteneur. Le texte, quant à lui, s'affiche alors à gauche.

L'image cup.png est disponible dans l'espace de téléchargement.

Dégagement

La propriété `clear` permet d'annuler le flottement introduit par la propriété `float`.

clear:	right; ou left; ou both; ou none;
--------	--

Commentaires

- la valeur `right` annule le flottement à droite.
- la valeur `left` annule le flottement à gauche.
- la valeur `both` annule le flottement des deux côtés.
- la valeur `none` annule toute propriété de flottement.

Exemple

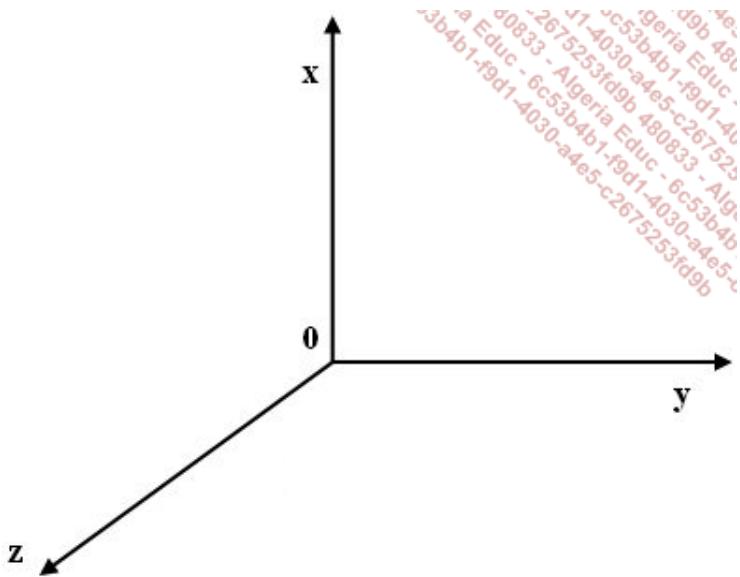
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<div style="float:right;">

</div>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</p>
<p style="clear:right">Sed non risus. Suspendisse lectus tortor,
dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras
elementum ultrices diam. Maecenas ligula massa, varius a, semper
congue, euismod non, mi.</p>
</div>
</body>
</html>
```



Superposition

La propriété `z-index` ajoute un axe en profondeur permettant de positionner des éléments au-dessus ou en dessous d'un autre élément.



Ainsi, l'élément avec une propriété `z-index: 2` apparaîtra avant ou au-dessus de l'élément avec une propriété `z-index: 0`.

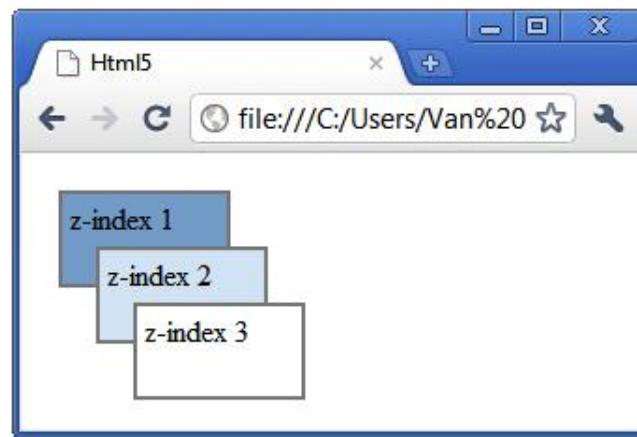
<code>z-index:</code>	un nombre entier (positif);
-----------------------	-----------------------------

Cette propriété ne fonctionne qu'avec un positionnement absolu des éléments.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.boitel { position: absolute;
    left: 20px; top: 20px;
    width: 80px; height: 40px;
    padding: 4px;
    border: 2px solid gray;
    background-color: rgb(125,165,205);
    z-index: 1;}
.boite2 { position: absolute;
    left: 40px; top: 50px;
    width: 80px; height: 40px;
    padding: 4px;
    border: 2px solid gray;
    background-color: rgb(215,230,245);
    z-index: 2;}
.boite3 { position: absolute;
    left: 60px; top: 80px;
    width: 80px; height: 40px;
    padding: 4px;
    border: 2px solid gray;
    background-color: white;
    z-index: 3;}
</style>
</head>
```

```
<body>
<div class="boite1">
z-index 1
</div>
<div class="boite2">
z-index 2
</div>
<div class="boite3">
z-index 3
</div>
</body>
</html>
```



Dépassement

La propriété `overflow` détermine ce que le navigateur doit faire lorsqu'un élément est plus grand que l'élément parent qui le contient.

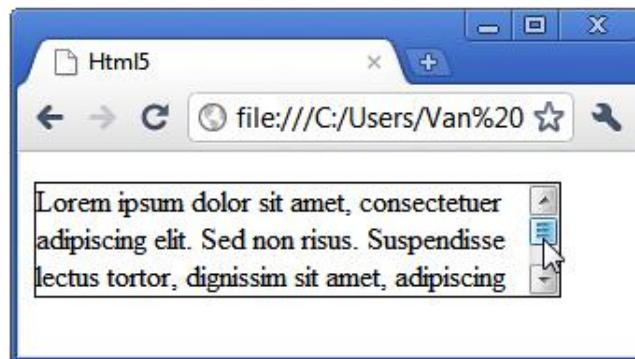
overflow:	hidden; ou scroll; ou visible; ou auto;
-----------	--

Commentaires

- Avec la valeur `hidden`, la partie qui dépasse est cachée sans possibilité de la voir ou d'y accéder.
- Avec la valeur `scroll`, la partie qui dépasse est cachée mais accessible par une barre de défilement.
- Avec la valeur `visible`, la partie qui dépasse est affichée, ignorant les spécifications de l'élément parent qui le contient.
- Avec la valeur `auto`, on rend la main au navigateur.

Exemple

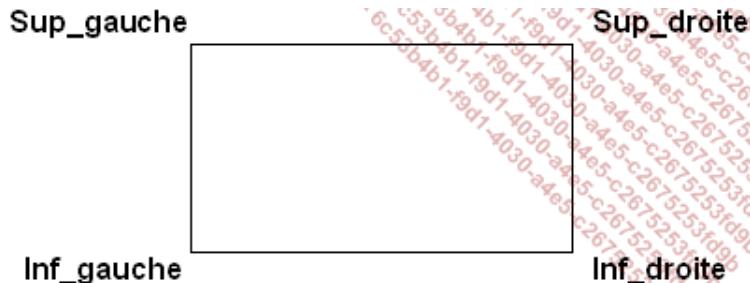
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.boite { width: 280px; height: 60px;
         border: 1px solid black;
         overflow: auto;}
</style>
</head>
<body>
<div class="boite">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non
risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing
nec, ultricies sed, dolor. Cras elementum ultrices diam. </div>
</body>
</html>
```



Découpage

Cette propriété `clip` détermine la partie visible de l'élément, généralement une image. Ainsi, l'image entière sera bien présente dans le document mais seulement une partie (rectangulaire) de celle-ci sera visible.

```
clip: rect(sup_gauche sup_droit inf_droit inf_gauche);  
auto;
```



Soit par exemple :

```
clip: rect(10px 270px 190px 50px);
```

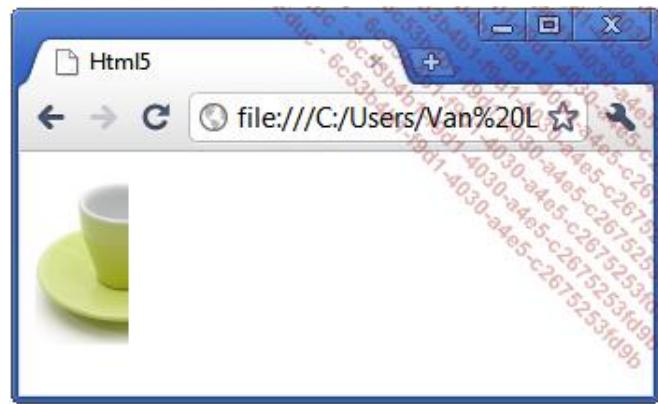
Commentaires

- Les coordonnées du rectangle découpé sont données par les sommets supérieur-gauche, supérieur-droit, inférieur-droit et inférieur-gauche par rapport à l'image initiale.
- On recommande d'inclure l'élément qui sera ainsi découpé dans une balise `<div>` ou ``.
- Cette propriété ne fonctionne qu'avec un positionnement absolu de l'élément.

Pour l'instant, seule la forme rectangulaire est implémentée. On prévoit un plus grand choix de formes à l'avenir.

Exemple

```
<!DOCTYPE html>  
<html lang="fr">  
<head>  
<title>Html5</title>  
<meta charset="iso-8859-1">  
<style type="text/css">  
.coupe { position: absolute;  
        clip: rect(0px 50px 115px 0px);}  
</style>  
</head>  
<body>  
<p></p>  
<div class="coupe">  
  
</div>  
</body>  
</html>
```



Affichage

La propriété `display` permet de contrôler l'affichage des éléments dans la page.

<code>display:</code>	<code>block;</code> ou <code>inline;</code> ou <code>none;</code>
-----------------------	---

Au chapitre Les propriétés de boîte - Élément bloc ou en ligne, nous avons déjà traité de la propriété `display` qui permet de redéfinir un élément en ligne comme un élément bloc et inversement (`display: block` et `display: inline`)

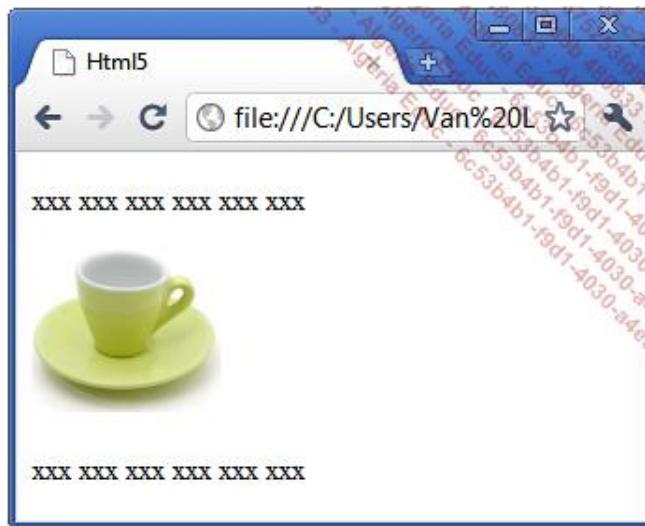
C'est la propriété `display: none` qui nous intéresse spécialement ici. Elle définit qu'un élément ne sera pas affiché. Cet élément est alors retiré du document et de la mise en page. Voir la section Visibilité ci-après, pour la différence avec la propriété `visibility: hidden`.

Exemple 1

L'affichage normal de l'image.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<p>xxx xxx xxx xxx xxx xxx</p>
<div>

</div>
<p>xxx xxx xxx xxx xxx xxx</p>
</body>
</html>
```



Exemple 2

Avec la propriété `display:none`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
```

```
.displaynone { display: none; }

```

```
</style>

```

```
</head>

```

```
<body>

```

```
<p>xxx xxx xxx xxx xxx</p>

```

```
<div class="displaynone">

```

```


```

```
</div>

```

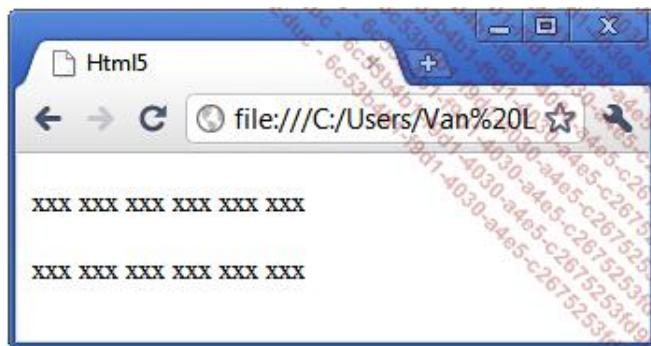
```
<p>xxx xxx xxx xxx xxx</p>

```

```
</body>

```

```
</html>
```



La capture d'écran illustre bien que l'image a été retirée comme si elle était absente du code source.

Visibilité

La propriété CSS `visibility` détermine si un élément est visible ou caché.

visibility:	visible; ou hidden;
-------------	------------------------

Commentaires

- La valeur `visible` affiche l'élément.
- La valeur `hidden` cache l'élément.

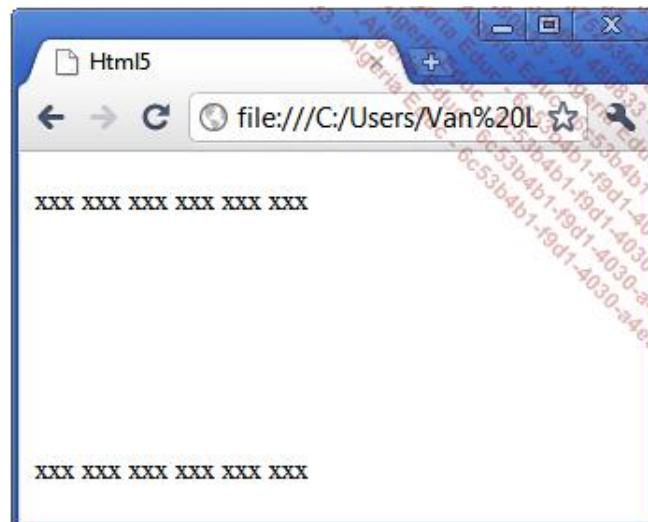
Attention ! Malgré la valeur `hidden`, l'élément occupe toujours sa place dans le document et un rectangle blanc apparaît là où il pourrait être affiché. De cette manière, la mise en page est conservée malgré l'absence (apparente) de l'image et les autres éléments de la page conservent leur position relative initiale. La valeur `hidden` diffère en cela de la propriété `display: none;` qui retire purement et simplement l'élément de la page et ne réserve pas de place pour celui-ci.

Exemple

Avec la propriété `visibility: hidden;`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.hidden { visibility: hidden; }
</style>
</head>
<body>
<p>xxx xxx xxx xxx xxx</p>
<div class="hidden">

</div>
<p>xxx xxx xxx xxx xxx</p>
</body>
</html>
```



Curseur de la souris

Cette propriété `cursor` permet de modifier le curseur de la souris.

cursor:	mot-clé;
---------	----------

Les différents mots-clés sont :

- `pointer`. Le curseur représente un doigt indiquant un lien.
- `move`. Le curseur indique un objet qu'on peut déplacer.
- `e-resize`. Curseur pointant vers l'est.
- `ne-resize`. Curseur pointant vers le nord-est.
- `nw-resize`. Curseur pointant vers le nord-ouest.
- `n-resize`. Curseur pointant vers le nord.
- `se-resize`. Curseur pointant vers sud-est.
- `sw-resize`. Curseur pointant vers le sud-ouest.
- `s-resize`. Curseur pointant vers le sud.
- `w-resize`. Curseur pointant vers l'ouest.
- `text`. Le curseur indique qu'on peut sélectionner le texte. Souvent un .
- `wait`. Le curseur indique une progression. Souvent une montre ou un sablier.
- `help`. Le curseur indique une aide. Souvent un point d'interrogation.
- `default`. Le curseur par défaut selon la plate-forme. Souvent une flèche.
- `crosshair`. Le curseur indique une croix.
- `progress`. Le curseur en forme de flèche avec sablier.
- `not-allowed`. Le curseur en forme de rond barré.
- `no-drop`. Le curseur en forme de main avec un doigt déplié et un rond barré.
- `col-resize`. Curseur fait de deux traits verticaux avec une flèche de chaque côté.
- `row-resize`. Curseur fait de traits horizontaux.
- `auto`. Le navigateur détermine lui-même le curseur selon le contexte.
- `url`. Spécifie un fichier image spécial pour le curseur. Le fichier image de l'URL doit être au format cur (curseur) ou ani (curseur animé).

Ou quelques curseurs sous forme imagée :

pointer	
move	
e-resize	
ne-resize	
nw-resize	
n-resize	
se-resize	
sw-resize	
s-resize	
w-resize	
text	
wait	
help	
crosshair	

Attention, l'apparence du curseur peut varier selon les navigateurs et le système d'exploitation.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
h2 { cursor: help; }
</style>
</head>
<body>
<h2>Les feuilles de style CSS</h2>
</body>
</html>
```



Numérotation automatique

La spécification CSS2 a introduit la notion de compteurs (*counter*) dans les propriétés de style. Grâce à ceux-ci, il sera possible de générer, par exemple, une numérotation automatique de sections et sous-sections.

counter-reset:	nom du compteur;
counter-increment:	nom du compteur valeur d'incrémentation;

Commentaires

- La propriété counter-reset crée ou réinitialise le compteur désigné.
- La propriété counter-increment incrémente le compteur désigné.
- Le pas d'incrémentation est par défaut de 1. Mais une autre valeur peut être spécifiée.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
body { counter-reset: section;
       font-size: 50%; }
h1 { counter-reset: item; }
h1:before { counter-increment: section;
            content: "Chapitre " counter(section) " : ";}
h2:before { counter-increment:item;
            content: counter(section) "." counter(item) " ";}
</style>
</head>
<body>
<h1>Partie Html</h1>
<h2>Html 4.0</h2>
<h2>Xhtml 1.0</h2>
<h2>Xhtml5</h2>
<h1>Partie scripts</h1>
<h2>JavaScript</h2>
<h2>Dhtml</h2>
<h1>Partie cadriels</h1>
<h2>jQuery</h2>
<h2>Dojo</h2>
</body>
</html>
```

Ces propriétés de style nécessitent quelques explications.

```
body { counter-reset: section; }
```

On initialise un compteur appelé `section` dont le champ d'action est le corps du document.

```
h1 { counter-reset: item; }
```

On initialise un autre compteur `item` dont le champ d'action est les titres de niveau 1.

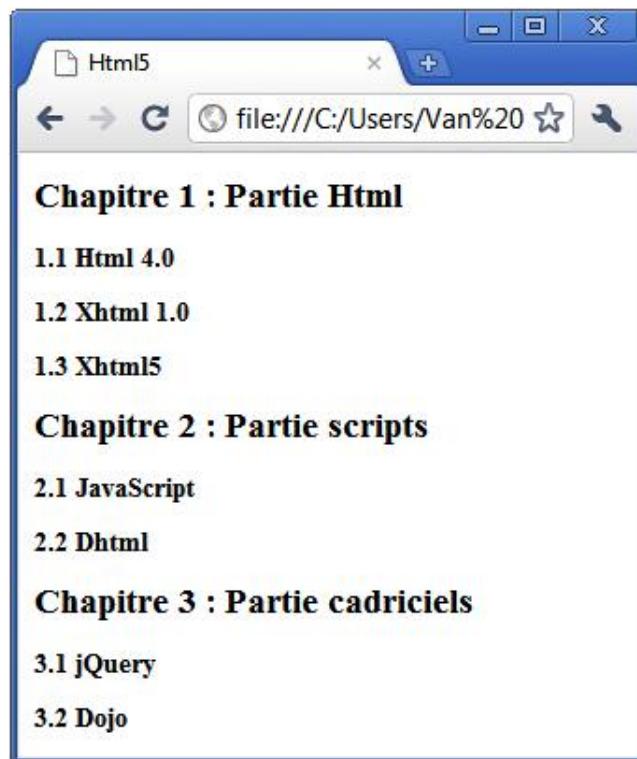
```
h1:before { counter-increment: section;
             content: "Chapitre " counter(section) " : "; }
```

À chaque titre `<h1>`, le compteur `section` s'incrémentera d'une unité (`counter-increment: section`). La propriété de style écrit devant les titres `<h1>` (`h1:before`) le contenu (`content`) repris entre les guillemets. Soit le texte "Chapitre " suivi de

la valeur du compteur section.

```
h2:before { counter-increment:item;
    content: counter(section) "." counter(item) " " ; }
```

À chaque titre `<h2>`, le compteur item s'incrémenté d'une unité (`counter-increment: item`). La propriété de style écrit devant les titres `<h2>` (`h2:before`) le contenu (`content`) repris entre les guillemets. Soit le texte "Chapitre " suivi de la valeur du compteur item.



Utilité des feuilles de style pour l'impression

Vous avez inévitablement déjà été déçu lors de l'impression d'une page Web. Entre autres désagréments, on peut citer :

- La page imprimée a épuisé une bonne partie de vos cartouches d'encre avec des images ou des bannières inutiles.
- Les menus de navigation, si utiles sur la toile, encombrent votre feuille imprimée avec des informations inutilisables.
- La police utilisée à l'écran se révèle difficilement lisible sur la version papier.
- La page Web est mal imprimée car des parties du document sont absentes de votre feuille.
- Etc.

Les feuilles de style viennent à votre secours pour prévoir une version imprimable sans devoir écrire une page distincte à cet effet.

Il ne sera pas nécessaire d'imprimer la page pour faire des essais ou des mises au point. Un aperçu avant impression permet de visualiser la page comme si elle était imprimée.

Feuilles de style selon le média de sortie

1. Présentation

Les feuilles de style permettent des représentations différentes selon le media de sortie.

Si nous sommes des habitués du Web à partir de l'écran, d'autres internautes le sont à partir d'un ordinateur de poche, d'un smartphone, d'une interface vocale, d'une barrette Braille ou d'un document imprimé.

Avec l'attribut `media`, qui vient s'ajouter à la déclaration de style, on peut concevoir des feuilles de style pour différents médias de sortie.

<code>media="all"</code>	Les styles définis s'appliquent à tous les types de médias (valeur par défaut).
<code>media="screen"</code>	Les styles s'appliquent à l'affichage sur un écran classique.
<code>media="print"</code>	Les styles s'appliquent à l'impression sur papier. Les navigateurs utiliseront ces spécifications lorsque l'utilisateur demande d'imprimer la page Web.
<code>media="projection"</code>	Les styles s'appliquent à la projection de données avec des projecteurs.
<code>media="handheld"</code>	Les styles s'appliquent à l'affichage sur des ordinateurs de poche ou des smartphones permettant la navigation sur le Web.
<code>media="aural"</code>	Les styles s'appliquent à des systèmes de restitution vocale assistée par ordinateur.
<code>media="braille"</code>	Les styles s'appliquent à des barrettes Braille conçues pour les non-voyants.
<code>media="embossed"</code>	Les styles s'appliquent à des imprimantes à perforation Braille conçues pour les non-voyants.

Ces différentes possibilités ne sont pas encore toutes prises en charge par les navigateurs récents. On pense aux propriétés vocales (`media="aural"`) dont les feuilles de style sont éditées depuis longtemps mais qui n'ont pas encore été reprises par les navigateurs grand public. Seul Opera 8+ les prend partiellement en charge.

2. Incorporation

Il existe différents moyens pour incorporer dans un document Html5 ces styles spécifiques à un média.

Style interne

Les propriétés de style s'encodent ici dans l'en-tête du document Html.

Exemple

```
<head>
<style type="text/css" media="print">
/* Feuilles de style qui s'appliquent au document imprimé */
</style>
<style type="text/css" media="screen">
/* Feuilles de style qui s'appliquent lors de l'affichage à l'écran */
</style>
<style type="text/css" media="all">
/* Feuilles de style qui s'appliquent en toutes circonstances */
</style>
</head>
```

Style externe

Les propriétés de style s'encodent ici dans un document distinct du document Html5. C'est sans aucun doute la méthode la plus utilisée par les concepteurs de site car la plus compatible et la plus proche de l'esprit du Html5.

Exemple

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

Déclaration qui appelle la feuille de style externe print.css, préparée pour l'impression (*print*).

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

Déclaration qui appelle la feuille de style ecran.css, destinée à l'affichage à l'écran (*screen*).

Style importé

Exemple

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

Déclaration qui appelle la feuille de style externe print.css, préparée pour l'impression (*print*).

```
@import url(ecran.css) screen;
```

Déclaration qui appelle la feuille de style ecran.css, destinée à l'affichage à l'écran (*screen*).

@media

Outre la possibilité d'incorporer diverses feuilles de style séparées pour différents médias de sortie dans une page Html, il existe également la possibilité de définir différents modes de sorties dans une seule déclaration de style.

```
<style type="text/css">
@media print
{
/* définition de formats pour l'impression */
}
@media screen
{
/* définition de formats pour une sortie écran ... */
}
</style>
```

Exemple

Ce mode de notation peu habituel en CSS donnerait en pratique :

```
<style type="text/css">
@media print
{ body { background-color: white; }
}
@media screen
{ body { background-color: black; }
}
</style>
```

Optimiser une page d'impression

Avant d'explorer les propriétés de style propres à l'impression, il convient de passer la page en revue afin de l'optimiser. On procédera en trois temps :

- déceler et éliminer tous les éléments qui se révéleront inutiles voire dérangeants lorsque la page sera imprimée.
- revoir le formatage du contenu textuel en vue de la lecture sur papier.
- ajouter des informations, habituellement réservées à l'affichage à l'écran.

 On peut dire qu'une page Web affichée à l'écran doit être plaisante à l'œil et qu'une information papier doit être plaisante à l'esprit. Plus que jamais, c'est la richesse de l'information ou du contenu qui est recherchée sur la page imprimée.

Sans être limitatif, il faudra veiller :

- aux arrière-plans de couleur ou avec image.
- à la couleur des caractères.
- au type de police de caractères.
- à la taille des caractères.
- aux liens.
- aux images.
- au menu de navigation.
- aux animations Flash, JavaScript et Dhtml.
- aux formulaires.
- aux marges...

Reprenons ces différents points en détail :

- Modifier les arrière-plans. Par défaut, la plupart des navigateurs n'impriment pas les couleurs et les images d'arrière-plan. Cependant, si votre visiteur a coché l'option avancée **Imprimer les couleurs et les images d'arrière-plan**, il risque de ne pas apprécier votre page spécialement conçue pour l'impression et de vous considérer (à tort) comme un piètre concepteur. Ainsi, il est prudent de définir pour cette page d'impression des arrière-plans blancs.

```
body { background: white;  
background-image: none; }
```

- Mettre en noir les polices de couleur. Parfois plaisants à l'écran, des caractères de police coloriés risquent de consommer inutilement de l'encre et n'apportent pas d'enrichissement à l'information sur papier.

```
body { color: black; }
```

- Changer la police en serif. Bien que ce soit une question d'appréciation personnelle, d'aucuns trouvent les polices sans-serif (Arial, Verdana, Helvetica) plus lisibles à l'écran que les polices de type serif (Times New Roman, Garamond), plus adaptées à la lecture sur papier.

```
font-family : Arial, Verdana, sans-serif;
```

- Veiller à la taille de la police. Si votre page Web est écrite avec une police de petite dimension, il est prudent d'adopter une taille de caractères de 11 pt ou 12 pt, voire plus grande en fonction de votre audience. Il est conseillé de travailler avec des points (pt), plus spécifiques à l'impression, plutôt que des pixels (px).

```
font-size : 12pt;
```

- Identifier les liens. Question délicate que celle des hyperliens. De toute évidence, ils ne sont pas aussi utiles sur papier qu'à l'écran, mais il est souvent important d'indiquer qu'il y avait des liens dans l'original.

Pour ce faire, plusieurs politiques peuvent être adoptées. On peut forcer le soulignement des liens dont celui-ci avait été effacé par la propriété `text-decoration: none;`

```
a { text-decoration : underline;}
```

Certains préconisent de compléter le lien avec son adresse.

```
a:after { content: " (" attr(href) " ) " ;}
```

Les mentions "cliquez ici", déjà peu plaisantes à l'écran, sont tout à fait inutiles en version papier.

- Enlever toute image et bannière inutile à la compréhension du contenu. Il est cependant conseillé de conserver les logos pour vous identifier auprès de l'utilisateur.

```
.noprint { display: none; }  
...  
<div class="noprint"><img scr="xxx.jpg" alt=""></div>
```

- Enlever les outils de navigation. Les menus de navigation, particulièrement s'ils sont placés sur les côtés, occupent de la place inutilement sur la feuille d'impression. En les enlevant, vous permettrez au texte de s'imprimer plus efficacement.

```
#navigation { display: none; }  
...  
<div id="navigation"> ... </div>
```

- Les marges. Les marges ont peut-être été modifiées par rapport aux marges par défaut pour l'affichage. Pour l'impression papier, celles-ci devront éventuellement être revues.

```
margin-left: 20pt;
```

- Enlever les animations Flash, JavaScript et Dhtml, car elles s'impriment très mal voire pas du tout. Même façon de procéder avec la propriété `display: none;`.
- Les formulaires ne sont daucune utilité en version papier. Même façon de procéder avec la propriété `display: none;`.
- Les ajouts. Certaines informations risquent d'être absentes de la page imprimée, comme l'indication de l'URL originale du site pour permettre à l'utilisateur de retourner sur votre site Web, votre logo et la mention d'un copyright si nécessaire.

Exemple

Soit la page :



```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
body { background-color: rgb(200,215,230); }
.titre { font-size: 36px;
         font-weight: bold; }
.texte { width: 350px;
         background-color: rgb(125,165,205);
         color: white;
         padding: 5px;
         font-size: 9pt;
         font-family: Arial, sans-serif; }
.formulaire { width: 350px;
         background-color: rgb(125,165,205);
         color: white;
         font-size: 9pt;
         font-family: Arial, sans-serif;
         padding: 5px; }
</style>
</head>
<body>
<p>
<span class="titre">Les poissons</span></p>
<div class="texte">
Les poissons sont des animaux aquatiques à sang froid, pourvus de
nageoires et dont le corps est couvert d'écaillles. Ils respirent
grâce à des branchies. Jusqu'à présent, on a répertorié plus de 27
315 espèces de poissons.
<br>
</div>
<div class="formulaire">
<form action="">
Documentation gratuite - Email : <input type="text" size="20"><br>
<input type="submit" value="Envoyer">
<input type="reset" value="Rétablir">
</form>
</div>
</body>
</html>
```

Cette page et les images sont disponibles dans l'espace de téléchargement.

Pour une page d'impression, on fera en sorte :

- d'enlever l'image du poisson, inutile pour le contenu.
- d'adopter des arrière-plans blancs. Bien que par défaut, les navigateurs n'impriment pas les couleurs et images d'arrière-plan, on prendra cependant cette précaution pour les utilisateurs qui auraient désactivé cette option.
- d'imprimer le texte dans une police plus grande et donc plus lisible.
- d'écrire le texte en noir plutôt qu'en blanc.
- d'abandonner le formatage en gras du texte pour une écriture normale.
- de retenir une police de type serif au lieu de la sans-serif initiale.
- d'ôter l'arrière-plan de couleur du texte.
- de ne pas imprimer le formulaire, inutile en version papier.

Le code devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css" media="screen">
body { background-color: rgb(200,215,230); }
.titre { font-size: 36px;
          font-weight: bold; }
.texte { width: 350px;
          background-color: rgb(125,165,205);
          color: white;
          padding: 5px;
          font-size: 9pt;
          font-family: Arial, sans-serif; }
.formulaire { width: 350px;
          background-color: rgb(125,165,205);
          color: white;
          font-size: 9pt;
          font-family: Arial, sans-serif;
          padding: 5px; }
</style>
<style type="text/css" media="print">
body { background-color: white; }
img { display: none; }
.titre { font-size: 36px;
          font-weight: bold; }
.texte { background-color: white;
          color: black;
          font-size: 12pt;
          font-family: sans-serif; }
.formulaire { display: none; }
</style>
</head>
<body>
<p>
<span class="titre">Les poissons</span></p>
<div class="texte">
Les poissons sont des animaux aquatiques à sang froid, pourvus de nageoires et dont le corps est couvert d'écaillles. Ils respirent grâce à des branchies. Jusqu'à présent, on a répertorié plus de 27 315 espèces de poissons.
<br>
</div>
```

```
<div class="formulaire">
<form action="">
Documentation gratuite - Email : <input type="text" size="20"><br>
<input type="submit" value="Envoyer">
<input type="reset" value="Rétablir">
</form>
</div>
</body>
</html>
```

À l'impression :



Saut de page avant

Cette propriété forcera un saut de page avant un élément de la page.

page-break-before:	always; avoid; auto;
--------------------	----------------------------

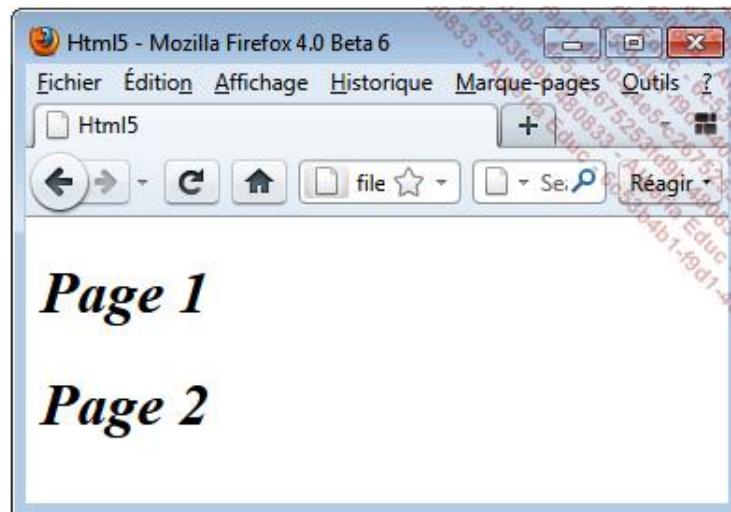
Commentaires

- Le paramètre `always` force un saut de page avant l'élément où il est défini.
- Le paramètre `avoid` empêche un saut de page avant l'élément où il est défini.
- Le paramètre `auto` laisse la main au navigateur.
- Cette propriété est reconnue par tous les navigateurs du marché.

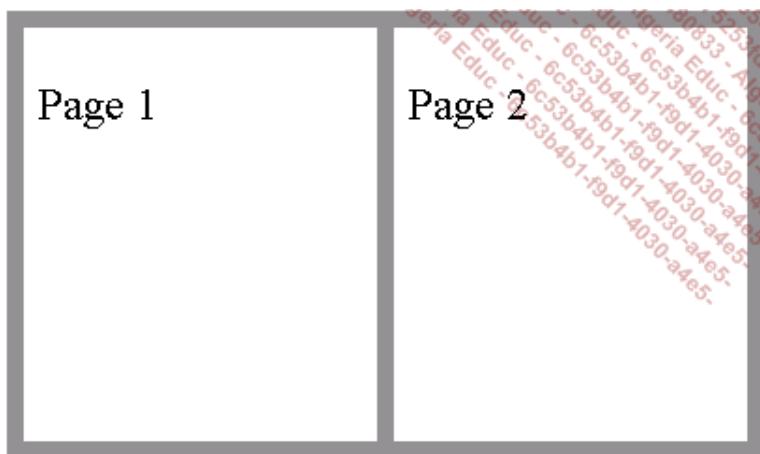
Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css" media="screen">
h1 { font-style: italic; }
</style>
<style type="text/css" media="print">
h1 { font-style: normal; }
.normal { font-size: 120pt; }
.break { font-size: 120pt;
         page-break-before:always; }
</style>
</head>
<body>
<h1 class="normal">Page 1</h1>
<h1 class="break">Page 2</h1>
</body>
</html>
```

Ce qui donne à l'écran :



L'aperçu avant impression révèle :



Saut de page arrière

Cette propriété forcera un saut de page après un élément de la page.

page-break-after:	always; avoid; auto;
-------------------	----------------------------

Commentaires

- Le paramètre `always` force un saut de page après l'élément où il est défini.
- Le paramètre `avoid` empêche un saut de page après l'élément où il est défini.
- Le paramètre `auto` laisse la main au navigateur.
- Cette propriété est reconnue par tous les navigateurs du marché.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css" media="screen">
h1 { font-style: italic; }
</style>
<style type="text/css" media="print">
.normal { font-size: 120pt; }
.break { font-size: 120pt;
    page-break-after:always; }
</style>
</head>
<body>
<h1 class="break">Page 1</h1>
<h1 class="normal">Page 2</h1>
</body>
</html>
```

L'aperçu après impression fournit le même résultat.

Éviter les sauts de page

Cette propriété évitera le saut de page dans l'élément désigné. Particulièrement utile pour éviter qu'un tableau soit imprimé sur plusieurs pages.

page-break-inside:	always; avoid; auto;
--------------------	----------------------------

Commentaires

- Le paramètre `always` force un saut de page après l'élément où il est défini.
- Le paramètre `avoid` empêche un saut de page après l'élément où il est défini.
- Le paramètre `auto` laisse la main au navigateur.
- À ce jour, cette propriété n'est reconnue que par le navigateur Opera.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css" media="screen">
h1 { font-style: italic; }
</style>
<style type="text/css" media="print">
table { page-break-inside:avoid; }
</style>
</head>
<body>
<table>
<tr>
<td>1</td><td>2</td><td>3</td><td>4</td><td>5</td>
...
<td>96</td><td>97</td><td>98</td><td>99</td><td>100</td>
</tr>
</table>
</body>
</html>
```

Le sélecteur @page

La spécification CSS2 a ajouté le sélecteur @page, réservé aux propriétés de style d'impression spécifiques.

Force est de constater que, même des années après, peu de ces propriétés spécifiques sont implémentées dans les navigateurs même récents alors que les spécifications du W3C à ce sujet sont nombreuses et a priori fort pratiques.

On pense au sélecteur de page @page, qui était implanté dans le navigateur Opéra 8+ et totalement ignoré par les autres navigateurs. Pourtant, ces fonctions sont pour le moins alléchantes. Citons les possibilités :

- d'imprimer en mode portrait ou mode paysage.
- de dimensionner la page imprimée.
- d'ajouter des marques de découpe.
- de prévoir un style pour la première page, les pages de gauche et les pages de droite.
- etc.

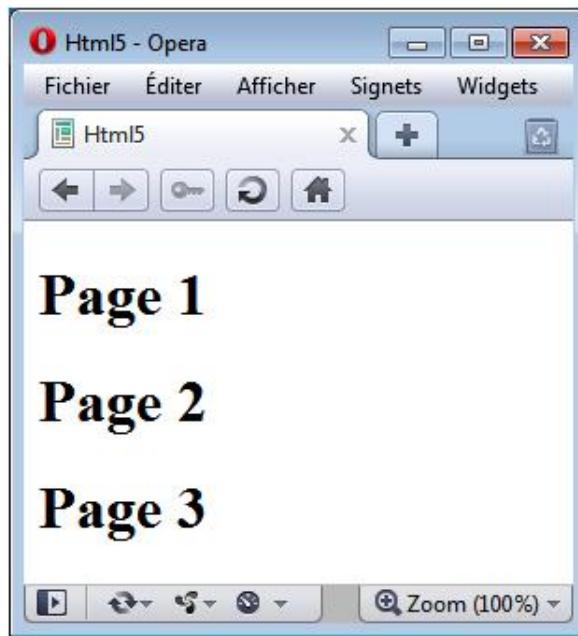
Profitons de l'occasion pour explorer ces possibilités dans Opera 10.6.

Exemple 1

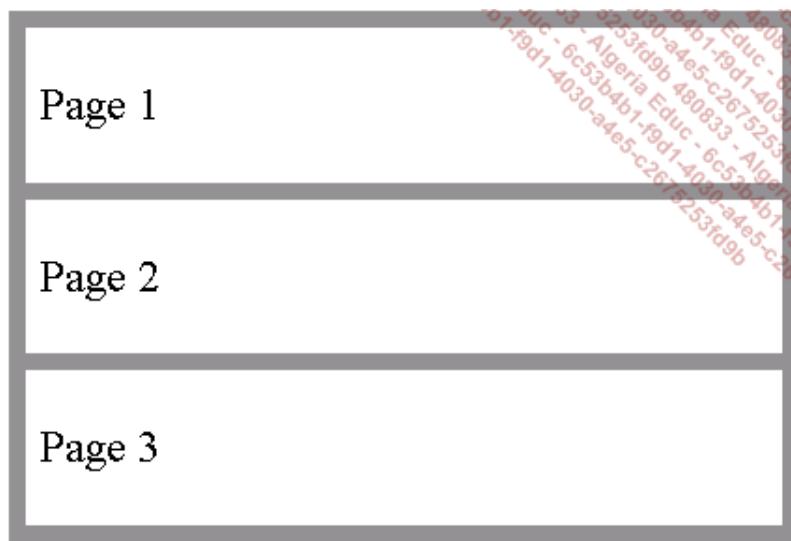
Forcer l'impression en mode paysage par @page{size: landscape; }.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css" media="print">
@page{ size: landscape; }
.normal { font-size: 120pt; }
.break { font-size: 120pt;
         page-break-after:always; }
</style>
</head>
<body>
<h1 class="break">Page 1</h1>
<h1 class="break">Page 2</h1>
<h1 class="normal">Page 3</h1>
</body>
</html>
```

À l'écran :



À l'impression (simulation) :



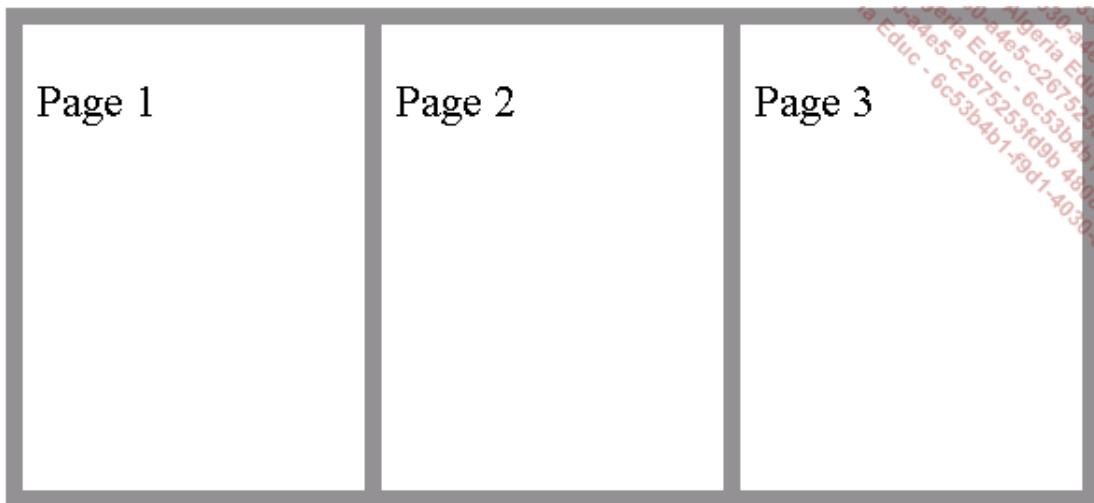
Exemple 2

Appliquer un style différent aux pages de gauche et aux pages de droite par respectivement @page:left et @page:right pour tenir compte de la reliure par exemple.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css" media="print">
@page:left{ margin-right: 420px; }
@page:right{ margin-left: 420px; }
.normal { font-size: 120pt; }
.break { font-size: 120pt;
         page-break-after:always; }
</style>
</head>
<body>
<h1 class="break">Page 1</h1>
<h1 class="break">Page 2</h1>
<h1 class="normal">Page 3</h1>
</body>
```

</html>

À l'impression (simulation) :



Les sélecteurs CSS3

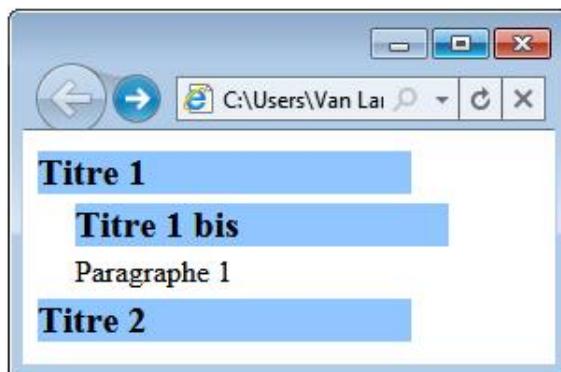
Les sélecteurs de base du CSS1 sont assez limités. Les spécifications CSS2 et CSS3 ajoutent une multitude de sélecteurs.

1. Les sélecteurs hiérarchiques

A B	Descendants	Sélectionne tout élément B qui est descendant d'un élément A.
A > B	Enfants	Sélectionne tout élément B qui est enfant direct d'un élément A.
A + B	Frère adjacent	Sélectionne tout élément B immédiatement précédé d'un élément A.

Exemple 1

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#division h1 { width: 200px;
               background-color: #9cf; }
h1 { margin-top: 5px;
      margin-bottom:5px;
      font-size: 20px; }
p { margin-top: 5px;
      margin-bottom:5px; }
</style>
</head>
<body>
<div id="division">
<h1>Titre 1</h1>
<div id="contenu" style="padding-left: 20px;">
<h1> Titre 1 bis</h1>
<p>Paragraphe 1</p>
</div>
<h1>Titre 2</h1>
</div>
</body>
</html>
```

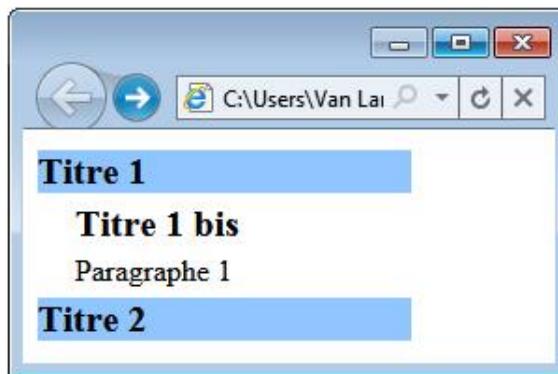


Exemple 2

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#division > h1 { width: 200px;
    background-color: #9cf; }
h1 { margin-top: 5px;
    margin-bottom:5px;
    font-size: 20px; }
p { margin-top: 5px;
    margin-bottom:5px; }
</style>
</head>
<body>
<div id="division">
<h1>Titre 1</h1>
<div id="contenu" style="padding-left: 20px;">
<h1> Titre 1 bis</h1>
<p>Paragraphe 1</p>
</div>
<h1>Titre 2</h1>
</div>
</body>
</html>

```

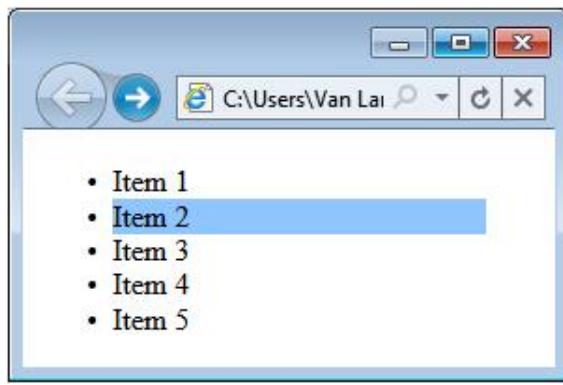


Exemple 3

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
li.un + li { width: 200px;
    background-color: #9cf; }
</style>
</head>
<body>
<ul>
<li class="un">Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
<li>Item 4</li>
<li>Item 5</li>
</ul>
</body>
</html>

```



2. Les pseudo-classes de sélection

:root	Représente l'élément racine ou l'élément le plus haut d'un document. Par exemple, en Html5, la balise <html>.
:empty	Correspond aux éléments vides et qui n'ont donc pas d'enfants.
:only-child	Reprend l'enfant unique. Celui-ci n'a donc pas d'élément frère.
:first-child	Sélectionne le premier élément enfant.
:last-child	Sélectionne le dernier élément enfant.
:nth-child(n)	Désigne le nième élément enfant où n est un nombre ou les mots-clés even (pairs) ou odd (impairs).
:nth-last-child(n)	Sélectionne le nième enfant en partant du dernier élément.
:only-of-type	Reprend l'unique élément de ce type.
:first-of-type	Représente le premier élément de ce type.
:last-of-type	Représente le dernier élément de ce type.
:nth-of-type(n)	Désigne le nième élément de ce type où n est un nombre ou les mots-clés even (pairs) ou odd (impairs).
:nth-last-of-type(n)	Sélectionne le nième élément de ce type en partant du dernier élément.

Ces sélecteurs sont implémentés dans Firefox 3.6+, Safari 5+, Chrome 7+, Opera 10.6+ et Internet Explorer 9+.

 La numérotation de n commence à 1 et non pas à 0 comme en JavaScript.

Exemples

Les exemples se basent sur le document Html5 suivant. Il comprend une balise parent et cinq balises enfant .

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
```

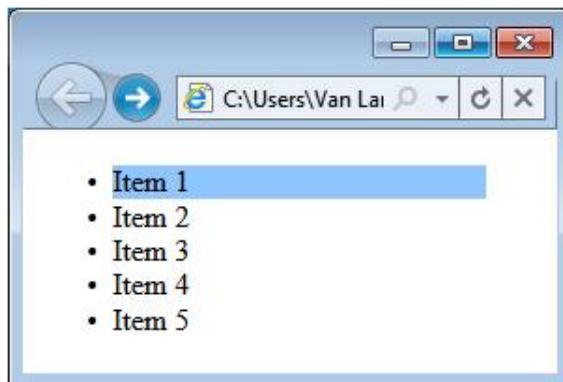
```
<ul>
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
<li>Item 4</li>
<li>Item 5</li>
</ul>
</body>
</html>
```

Le premier item de la liste

```
ul :first-child { background-color: #9cf; }
```

ou

```
li:first-of-type { background-color: #9cf; }
```

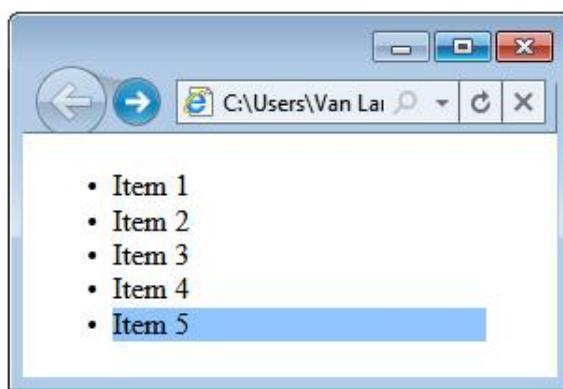


Le dernier item de la liste

```
ul :last-child { background-color: #9cf; }
```

ou

```
li:last-of-type { background-color: #9cf; }
```

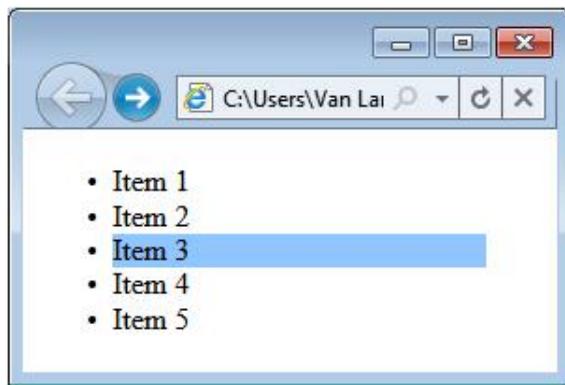


Le troisième item de la liste

```
ul :nth-child(3) { background-color: #9cf; }
```

ou

```
li:nth-of-type(3) { background-color: #9cf; }
```

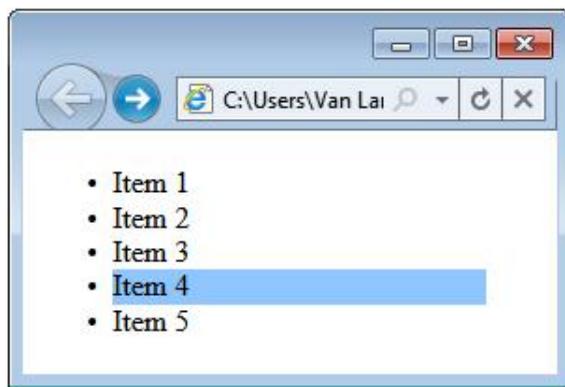


Le quatrième item de la liste (ou le second en partant du dernier élément)

```
ul :nth-last-child(2) { background-color: #9cf; }
```

ou

```
li:nth-last-of-type(2) { background-color: #9cf; }
```

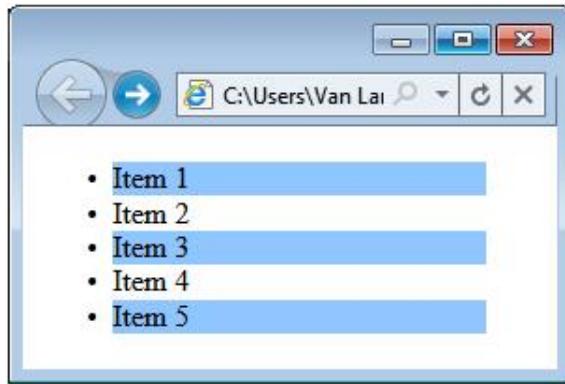


Les éléments impairs

```
ul :nth-child(odd) { background-color: #9cf; }
```

ou

```
li:nth-of-type(odd) { background-color: #9cf; }
```

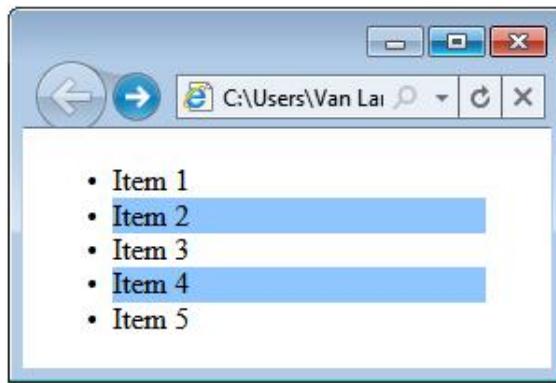


Les éléments pairs

```
ul :nth-child(even) { background-color: #9cf; }
```

ou

```
li:nth-of-type(even) { background-color: #9cf; }
```



3. Les sélecteurs d'attributs

[attr]	Désigne un élément qui comporte l'attribut attr indiqué.
[attr="valeur"]	Désigne un élément qui comporte l'attribut attr fourni par la valeur indiquée.
[attr~="valeur"]	Correspond à tout élément dont l'attribut attr contient une liste de valeurs séparées par des espaces et dont l'une d'elles est valeur.
[attr = "valeur"]	Correspond à tout élément dont l'attribut attr contient une liste de valeurs séparées par des tirets débutant par valeur.
[attr^="valeur"]	Représente un élément dont l'attribut attr commence avec le préfixe fourni par la valeur indiquée.
[attr\$="valeur"]	Représente un élément dont l'attribut attr se termine avec le suffixe fourni par la valeur indiquée.
[attr*="valeur"]	Représente un élément dont l'attribut attr contient une instance de la valeur indiquée.

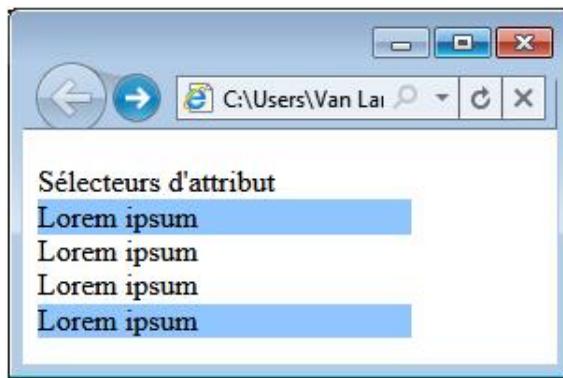
Ces sélecteurs d'attributs sont implémentés dans Firefox 3.6+, Safari 5+, Chrome 7+, Opera 10.6+ et Internet Explorer 9+.

Exemples

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
</head>
<body>
<div class="titre">Sélecteurs d'attribut</div>
<div name="un">Lorem ipsum</div>
<div title="deux">Lorem ipsum </div>
<div title="trois">Lorem ipsum </div>
<div name="dernier">Lorem ipsum </div>
</body>
</html>
```

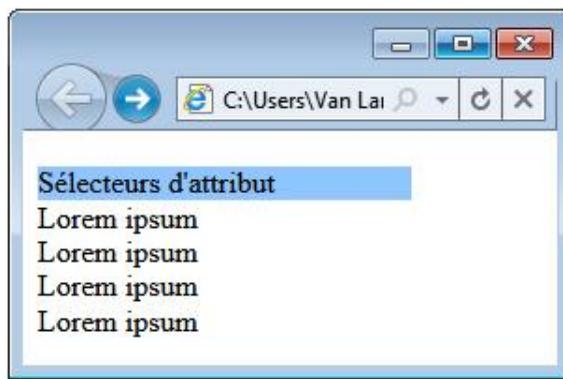
Élément(s) avec l'attribut name

```
div[name] { background-color: #9cf; }
```



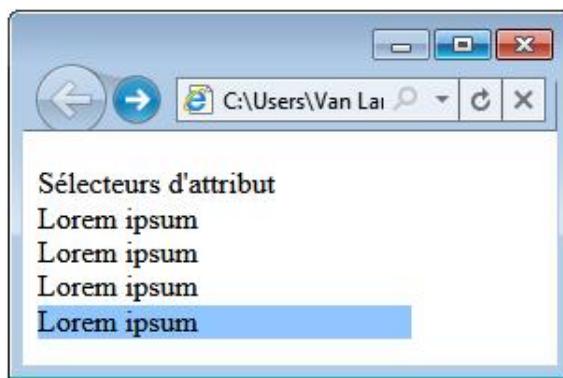
Élément avec l'attribut class="titre"

```
div[class="titre"] { background-color: #9cf; }
```



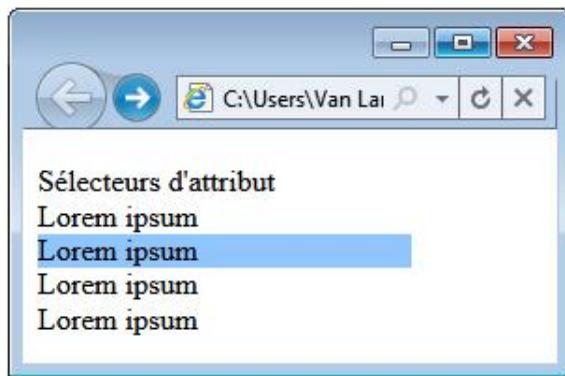
Élément avec un attribut name qui commence par "de"

```
div[name^="de"] { background-color: #9cf; }
```



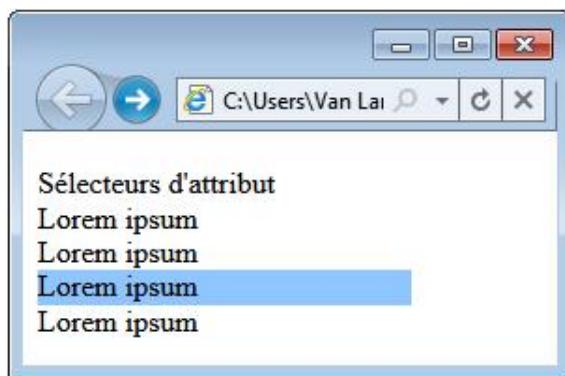
Élément dont l'attribut title se termine par "x".

```
div[title$="x"] { background-color: #9cf; }
```



Élément dont l'attribut title contient "roi"

```
div[title*="roi"] {background-color: #9cf;}
```



Notons qu'il est possible de combiner les sélecteurs.

```
div[name="un"][class="titre"]
```

Les bords arrondis

Après presque 20 ans de formes rectangulaires avec des angles droits et pointus, voici venir enfin les bords arrondis, tout en rondeurs.

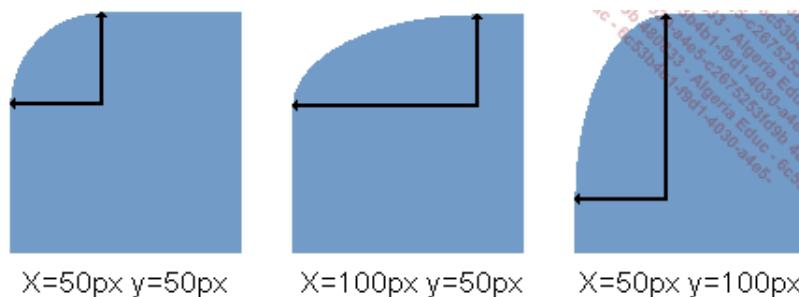
Jusqu'à présent, les bords arrondis ne pouvaient être réalisés que par une gymnastique qui relevait plus du rayon des trucs et astuces avec de multiples divisions `<div>` et des images placées dans leur arrière-plan. Avec le CSS3, ces bords arrondis sont réalisés directement avec quelques lignes de propriétés de style.

Les bords arrondis des quatre extrémités sont introduits par la propriété :

border-radius	x y; où x et y sont une valeur ou un pourcentage
---------------	---

Les valeurs de x et y déterminent les rayons horizontaux et verticaux d'un quart d'ellipse, ce qui induira la courbure de l'angle.

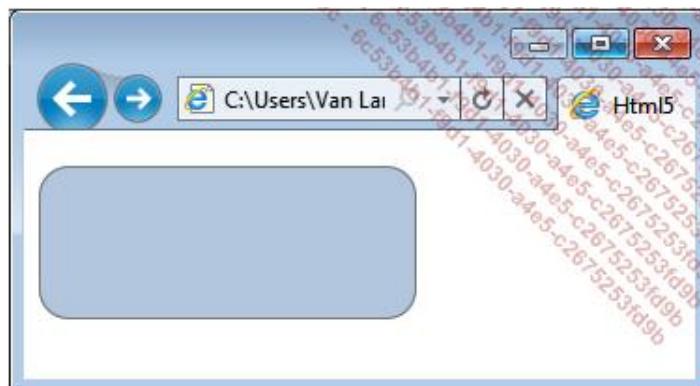
Les exemples suivants sont plus parlants.



Une seule valeur peut être indiquée. Dans ce cas, la valeur de x est égale à la valeur de y.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#round { width: 200px; height: 80px;
          background-color: rgb(185,205,225);
          border: 1px solid gray;
          border-radius: 1em; }
</style>
</head>
<body>
<div id="round"></div>
</body>
</html>
```



Cette propriété CSS3 est reconnue en l'état par Internet Explorer 9+, Opera 10.6+, Safari 5+ et Chrome 7+.

Et Firefox ? Une petite complication se présente. Firefox a introduit la propriété `radius` depuis de nombreuses années, sans attendre les recommandations CSS3. Ainsi pour Firefox 3.6+, on utilisera la propriété `border-radius` précédée du préfixe `-moz`. Soit :

```
-moz-border-radius: 1em;
```

Safari et Chrome, qui reprennent le moteur de rendu Webkit, ont également anticipé cette propriété CSS3 et leurs versions antérieures à celles signalées ci-avant nécessitent le préfixe `-webkit`. Soit :

```
-webkit-border-radius: 1em;
```

Une version parfaitement compatible de notre exemple devient alors :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#round { width: 200px; height: 80px;
          background-color: rgb(185,205,225);
          border: 1px solid gray;
          border-radius: 1em;
          -moz-border-radius: 1em;
          -webkit-border-radius: 1em; }
</style>
</head>
<body>
<div id="round"></div>
</body>
</html>
```

Comme souvent en CSS, il est possible de spécifier indépendamment un angle pour chaque extrémité.

border-top-right-radius border-bottom-right-radius border-bottom-left-radius border-top-left-radius	x y; où x et y sont une valeur ou un pourcentage
--	---

```
border-top-left-radius: 10px 5px;
```

Il est également possible de définir l'arrondi de chacun des angles, à l'aide d'une écriture raccourcie qui se lit comme à l'accoutumée dans le sens des aiguilles d'une montre en débutant par le haut (`top, right, bottom, left`).

```
border-radius: liste valeurs / liste valeurs
```

La première liste de valeurs représente les rayons horizontaux des angles, la deuxième liste représente les rayons verticaux des angles.

```
border-radius: 5px 10px 5px 10px / 10px 5px 10px 5px;
```

Les écritures avec les préfixes `-moz` et `-webkit` sont :

```
-moz-border-radius-topright  
-moz-border-radius-bottomright  
-moz-border-radius-bottomleft  
-moz-border-radius-topleft
```

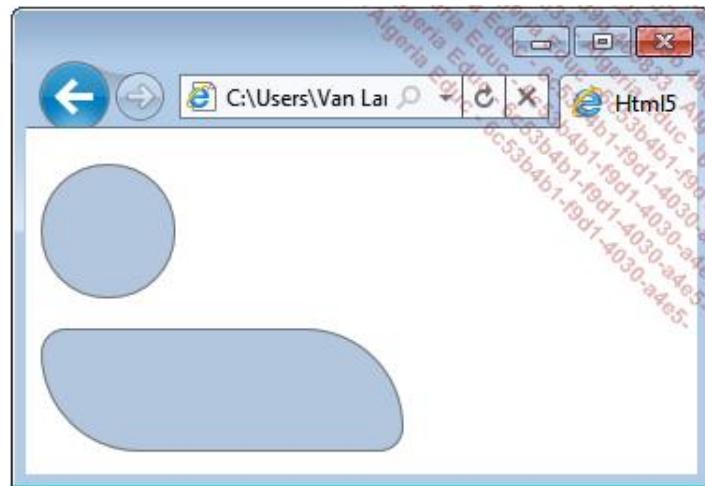
```
-webkit-border-top-right-radius  
  
-webkit-border-bottom-right-radius  
  
-webkit-border-bottom-left-radius  
  
-webkit-border-top-left-radius
```

Il faut souligner la position différente du mot `radius` et la différence entre `top-right` et `topright`.

Tout ceci permet des figures assez inhabituelles dans le design des pages.

Exemple 1

```
<!DOCTYPE html>  
<html lang="fr">  
<head>  
<title>Html5</title>  
<meta charset="iso-8859-1">  
<style type="text/css">  
#round1 { width: 70px; height: 70px;  
    background-color: rgb(185,205,225);  
    border: 1px solid gray;  
    border-radius: 35px;  
    -moz-border-radius: 35px;  
    -webkit-border-radius: 35px; }  
#round2 { width: 12em; height: 4em;  
    background-color: rgb(185,205,225);  
    border: 1px solid gray;  
    border-radius: 1em 4em 1em 4em;  
    -moz-border-radius: 1em 4em 1em 4em;  
    -webkit-border-radius: 1em 4em 1em 4em; }  
</style>  
</head>  
<body>  
<p></p>  
<div id="round1"></div>  
<p></p>  
<div id="round2"></div>  
</body>  
</html>
```



Exemple 2

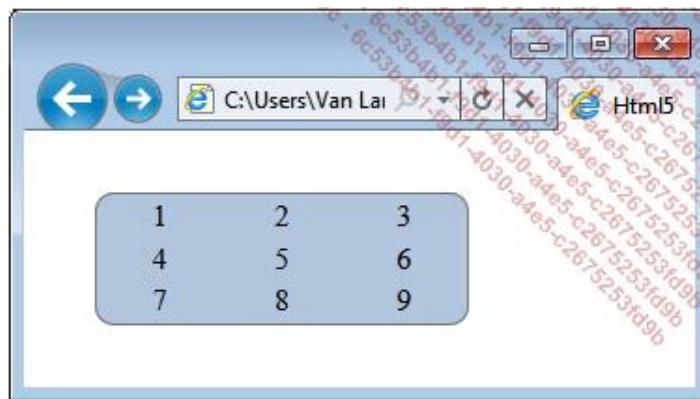
Les bords arrondis ne s'appliquent pas qu'aux divisions `<div>`. Voici une application dans un tableau.

```
<!DOCTYPE html>
```

```

<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type='text/css'>
table { width: 200px;
    background-color: rgb(185,205,225);
    text-align: center;
    margin:30px;
    border:1px solid gray;
    border-radius:10px;
    -moz-border-radius:10px;
    -webkit-border-radius:10px; }
</style>
</head>
<body>
<table>
<tr><td>1</td><td>2</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td>6</td></tr>
<tr><td>7</td><td>8</td><td>9</td></tr>
</table>
</body>
</html>

```



Les bords arrondis ne sont pas possibles avec la propriété collapse.

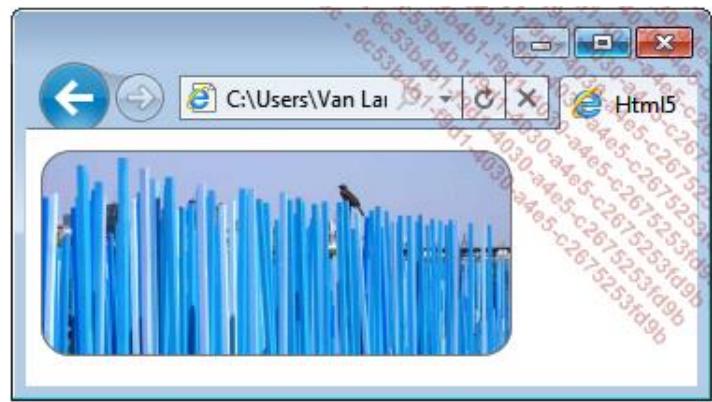
Exemple 3

Les bords ronds peuvent aussi s'appliquer à des images et ce, sans passer par un programme graphique.

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#round { width: 250px; height: 108px;
    border-radius: 1em;
    -moz-border-radius: 1em;
    -webkit-border-radius: 1em;
    border: 1px solid gray;
    background-image: url(blue.png); }
</style>
</head>
<body>
<div id="round"></div>
</body>
</html>

```

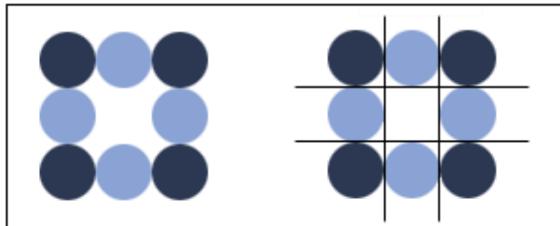


L'image blue.png est disponible dans l'espace de téléchargement consacré à cet ouvrage.

Les bordures imagées

Une des nouveautés des CSS3 est de permettre l'utilisation d'une image pour l'appliquer à la bordure d'un élément boîte.

La propriété `border-image` prend une image rectangulaire et la divise en 9 parties (voir capture d'écran ci-dessous). Les 8 parties du contour sont alors utilisées pour les angles et les côtés. Le centre est caché pour permettre l'affichage du contenu.



L'écriture de cette propriété est assez complexe.

border-image	url(fichier_image) a b c d valeur valeur;
	où
	url(fichier_image) est l'adresse relative de l'image servant à confectionner les bordures.
	a, b, c et d sont la dimension d'un morceau de la grille dans les côtés de la bordure.

valeur soit `round`, `repeat` ou `stretch`.

Soit par exemple :

```
border-image: url(border.png) 27 27 27 27 round round;
```

Commentaires

- On peut ne signaler qu'un paramètre. Dans ce cas, celui-ci est appliqué pour les quatre côtés. Avec deux paramètres, ceux-ci déterminent la dimension des côtés supérieur/inférieur et gauche/droite. Trois paramètres, respectivement le côté supérieur, gauche/droite et inférieur. Et enfin quatre paramètres, dans l'ordre, bord supérieur, bord droit, bord inférieur et bord gauche.
- Le mot-clé `round` reproduit les images et redimensionne celles-ci afin qu'elles s'ajustent exactement à la largeur et la hauteur de l'élément. Le mot-clé `repeat` (non repris par certains navigateurs) effectue le même travail que `round` mais sans ajustement. Et enfin, `stretch` (défaut) étire l'image aux dimensions de la boîte. Une seule valeur détermine les quatre côtés. Deux valeurs s'appliquent respectivement au côté supérieur/inférieur et gauche/droite.

La propriété `border-image` est reprise par Firefox 3.6+, Safari 3+, Chrome 2+ et Opera 10.6+. Elle n'est pas intégrée dans Internet Explorer 9.

Exemple 1

Au départ de l'image (border.png) reproduite ci-avant et disponible dans l'espace de téléchargement.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#box { width:220px; height: 80px;
       border-width:20px;
       border-image:url(border.png) 30 30 round;
```

```

-moz-border-image:url(border.png) 30 30 round;
-webkit-border-image:url(border.png) 30 30 round;
padding: 10px;
text-align: center; }

</style>
</head>
<body>
<div id="box"><h1>Html5 et CSS3</h1></div>
</body>
</html>

```

Remarquez qu'il est nécessaire de fixer la largeur de la bordure (`border-width`). Ce qui paraît logique.



Exemple 2

Avec la valeur d'étirement `stretch` :

```

<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#box { width:220px; height: 80px;
       border-width:20px;
       border-image:url(border.png) 30 30 round stretch;
       -moz-border-image:url(border.png) 30 30 round stretch;
       -webkit-border-image:url(border.png) 30 30 round stretch;
       padding: 10px;
       text-align: center; }

</style>
</head>
<body>
<div id="box"><h1>Html5 et CSS3</h1></div>
</body>
</html>

```



Exemple 3

Soit l'image de départ (border_image.png),



```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#box { width:220px; height: 80px;
       border-width:20px;
       border-image:url(border_image.png) 30 30 round;
       -moz-border-image:url(border_image.png) 30 30 round;
       -webkit-border-image:url(border_image.png) 30 30 round;
       padding: 10px;
       text-align: center; }
</style>
</head>
<body>
<div id="box"><h1>Htm15 et CSS3</h1></div>
</body>
</html>
```



Les ombres

Avec les propriétés CSS3, il est à présent possible d'ajouter un effet d'ombre sur le texte ou les éléments boîte.

1. Les ombres sur le texte

text-shadow:	x y z couleur où - x est le déport de l'ombre vers la droite. - y est le déport de l'ombre vers le bas. - z est l'intensité du dégradé ou du flou (facultatif, par défaut 0). - couleur est la couleur de l'ombre.
--------------	--

Les paramètres x et y admettent des valeurs négatives. Le déport de l'ombre est alors respectivement vers la gauche et vers le haut.

Cette propriété de style est intégrée par Firefox 3.6+, Safari 3+, Chrome 2+ et Opera 10+.

Pour Internet Explorer (même la version 9), il faut passer par le filtre propriétaire Microsoft Shadow. Pour un effet d'ombrage compatible, il faut ainsi ajouter par exemple :

```
.ombrage {  
filter:progid:DXImageTransform.Microsoft.Shadow(color='#999999',  
Direction=135,  
Strength=4);}
```

Exemple

```
<!DOCTYPE html>  
<html lang="fr">  
<head>  
<title>Html5</title>  
<meta charset="iso-8859-1">  
<style type='text/css'>  
.ombre { text-shadow: 2px 2px 4px #999;  
filter:progid:DXImageTransform.Microsoft.Shadow(color='aaaaaa',  
Direction=135, Strength=6);}  
.relief { color : white;  
text-shadow: 2px 2px 3px black;  
filter:progid:DXImageTransform.Microsoft.Shadow(color='aaaaaa',  
Direction=135, Strength=12);}  
</style>  
</head>  
<body>  
<h1 class="ombre">Html5 + CSS3</h1>  
<h1 class="relief">Editions Eni</h1>  
</body>  
</html>
```



Pour les amateurs de précision, `text-shadow` était inclus dans la spécification CSS2 (d'où l'absence de préfixes). Il a été retiré de la CSS2.1 et revient avec les CSS3.

2. Les ombres sur les éléments boîte

box-shadow	<p>x y z couleur où</p> <ul style="list-style-type: none">- x est le déport de l'ombre vers la droite.- y est le déport de l'ombre vers le bas.- z est l'intensité du dégradé ou du flou (facultatif, par défaut 0).- couleur est la couleur de l'ombre.
------------	---

Les paramètres x et y admettent des valeurs négatives. Le déport de l'ombre est alors respectivement vers la gauche et vers le haut.

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

Cette propriété de style est intégrée par Firefox 3.6+, Safari 3+, Chrome 2+ et Opera 10.6+.

Comme propriété CSS3, il faut ajouter les préfixes `-moz` pour Firefox et `-webkit` pour Safari et Chrome.

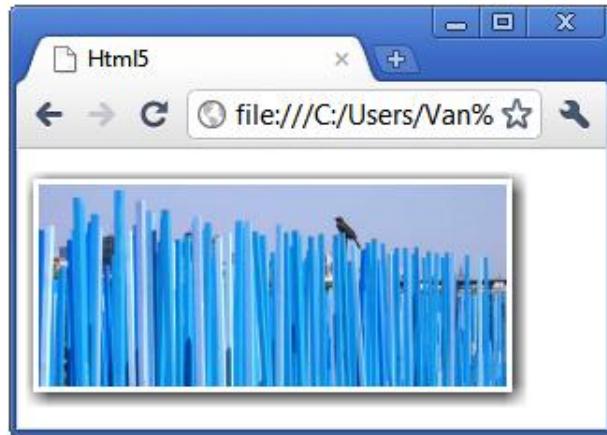
Pour Internet Explorer (même la version 9), il faut passer par le filtre propriétaire Microsoft `Shadow`. Pour un effet d'ombrage compatible, il faut ajouter par exemple :

```
.ombrage { filter:progid:DXImageTransform.Microsoft.Shadow(color='#aaaaaa', Direction=135, Strength=12); }
```

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.ombre { border: 3px solid white;
    box-shadow: 2px 2px 12px black;
    -moz-box-shadow: 2px 2px 12px black;
    -webkit-box-shadow: 2px 2px 12px black;
    filter:progid:DXImageTransform.Microsoft.Shadow(color='#aaaaaa', Direction=135, Strength=12); }
</style>
</head>
<body>

</body>
</html>
```



L'effet d'ombre aurait pu être directement appliqué sur l'image.

Les polices personnalisées

Jusqu'à présent les pages Web n'étaient pas une merveille de typographie. En effet, comme les polices dépendent de celles installées sur l'ordinateur de l'internaute, peu de polices sont à la fois disponibles sur les postes équipés de Windows ou Mac.

Les CSS3 permettent d'importer et d'utiliser une police personnalisée même si celle-ci n'est pas installée chez l'utilisateur. Cette propriété `@font-face` constitue ainsi une innovation significative pour les webgraphistes qui pourront ainsi reprendre des typographies plus créatives.

L'adoption d'une police particulière s'effectue en deux temps.

Il faut d'abord télécharger la police de caractères pour l'incorporer dans la page.

```
@font-face { font-family: "nom_police";
  url("nom_police.extension_police"); }
```

Commentaires

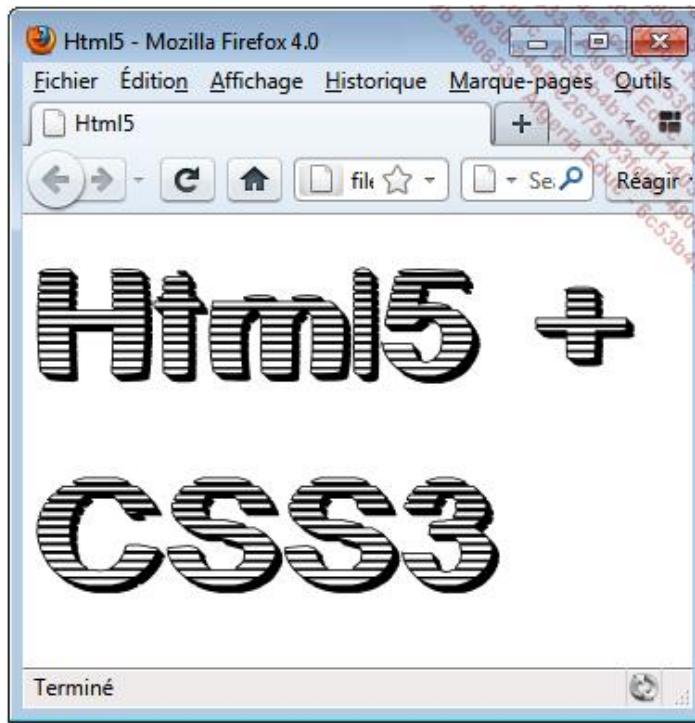
- `font-family: "nom_police"` indique sous quel nom la police sera utilisée dans le code de la page.
- `url("nom_police.extension_police")` indique l'url en adressage relatif ou absolu de la police personnalisée. En adressage relatif, la police sera ainsi présente dans le répertoire du site comme une image ou un fichier de feuille de style externe. On retiendra de préférence des polices adaptées au Web (webfonts) dont la taille réduite ne pénalisera pas le temps de téléchargement.
- `extension_police` détermine l'extension d'une police de caractères, par exemple l'extension ttf (*True Type Fonts*).

Une fois la police téléchargée avec `@font-face`, celle-ci peut être appliquée à un élément, comme n'importe quelle police, avec la propriété `font-family`.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
@font-face { font-family: "mapolice";
  src: url('fantaisie.ttf');}
.fontface { font: 100px mapolice, fantasy;}
</style>
</head>
<body>
<div class="fontface">Html5 + CSS3</div>
</body>
</html>
```

La police fantaisie.ttf est disponible dans l'espace de téléchargement dédié à cet ouvrage.



Ceci fonctionne parfaitement avec Firefox 3.6+, Safari 5+, Chrome 5+ et Opera 10.6+.

Internet Explorer 9 reconnaît lui aussi la propriété `@font-face` mais ne prend en compte que la police au format eot (*Embedded OpenType*), propriétaire Microsoft.

Un script compatible pourrait être :

```
@font-face { font-family: "Boston";
    src: url('boston_traffic.eot');
    src: url('boston_traffic.ttf');}
```

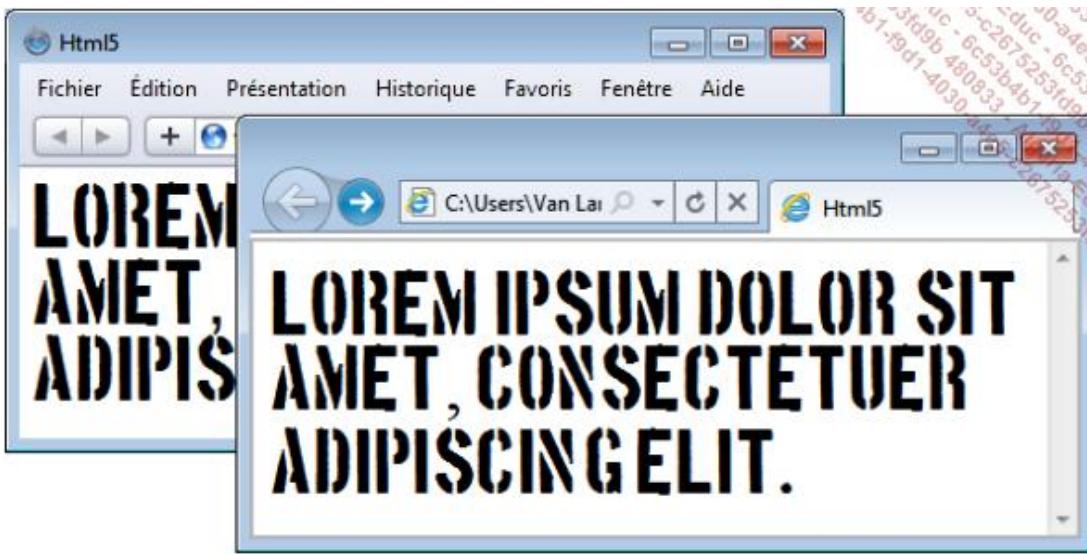
Mais, Internet Explorer a quelques difficultés avec l'attribut `src` lorsqu'il appelle tantôt une police eot et tantôt une police ttf.

Il faudra mettre en place une astuce pour rendre la feuille de style compatible avec tous les navigateurs de notre étude.

```
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
@font-face {
font-family: 'Boston';
src: url('boston_traffic.eot');
src: local('x'),
url('boston_traffic.ttf') format('TrueType');");
.police {font: 40px 'Boston', Arial, sans-serif;}
</style>
</head>
<body>
<p></p>
<p class="police">Lorem ipsum dolor sit amet, consectetur
adipiscing elit.</p>
</body>
</html>
```

Parmi les attributs que l'on peut ajouter à la propriété `@font-face`, il existe l'attribut (facultatif) `local("nom_police")` qui demande au navigateur de vérifier s'il existe une version locale de la police avant de la télécharger. Internet Explorer télécharge bien la police eot mais bloque sur le `local("x")` qu'il ne reconnaît pas et ne se préoccupera donc pas de la police ttf. Les autres navigateurs, ne trouvant pas en local la police x, téléchargeront alors la police ttf.

Les polices `boston_traffic.eot` et `boston_traffic.ttf` sont disponibles dans l'espace de téléchargement.



Il existe un autre moyen d'utiliser des polices particulières sur une page Web sans avoir à s'occuper de @font-face et parfaitement compatible avec les différents navigateurs. Il consiste à passer par Google Font Api. Les polices proposées par ce service sont parfaitement adaptées à l'affichage dans un navigateur et d'un poids réduit.

Il suffit simplement d'ajouter une feuille de style à votre page Html afin d'obtenir la nouvelle police.

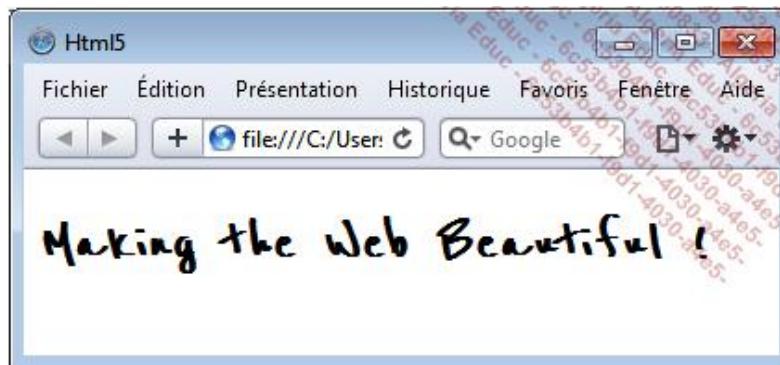
```
<link
href='http://fonts.googleapis.com/css?family=Reenie+Beanie&subset=latin'
rel='stylesheet' type='text/css'>
```

Il suffit alors de l'appliquer dans une classe CSS avec la propriété font-family.

Il ne faut donc pas passer par la propriété @font-face, ni s'occuper du format de la police. Tout est géré par Google afin d'assurer le meilleur affichage de celle-ci dans le navigateur.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<link
href='http://fonts.googleapis.com/css?family=Reenie+Beanie&subset=
latin' rel='stylesheet' type='text/css'>
<style>
h1 { font-family: 'Reenie Beanie', arial, serif; }
</style>
<body>
<h1>Making the Web Beautiful !</h1>
</body>
</html>
```



Pour trouver des polices de caractères adaptées au Web, une recherche sur Google avec le mot-clé *webfont* vous

fournira l'adresse de sites spécialisés.

Citons néanmoins :

- www.fontsquirrel.com/
- kernest.com/
- www.webfontlist.com/
- www.smashingmagazine.com/2007/11/08/40-excellent-freefonts-for-professional-design/

N'oubliez pas qu'il peut exister un copyright sur les polices et que certaines sont payantes.

Il subsiste la question de savoir comment convertir des polices ttf en format eot ou d'autres formats de polices.

Il y a bien entendu le logiciel vieillissant et peu ergonomique Microsoft Weft pour convertir en format eot. On lui préférera un convertisseur en ligne à l'adresse www.kirsle.net/wizards/ttf2eot.cgi ou le programme eotfast (<http://eotfast.com/>).

Un convertisseur multiformats en ligne, en français, est quant à lui disponible à l'adresse <http://www.convertissez.fr/convertisseur-police.php>.

Ajuster la hauteur des caractères

Des polices différentes, malgré une taille de caractères identique, peuvent paraître plus grandes ou plus petites que les polices avoisinantes.



En outre, avec la propriété `font-family`, vous n'avez pas la maîtrise de la police réellement affichée. En effet, lorsqu'une police de caractères n'existe pas sur l'ordinateur de l'utilisateur, les navigateurs utilisent une police de remplacement. Celle-ci n'a pas forcément les mêmes caractéristiques dimensionnelles que la police originale, et ce, même si la taille des caractères est identique.

La propriété `font-size-adjust` permet d'obtenir des caractères toujours de la même taille, aussi bien pour les majuscules que pour les minuscules, en figeant le coefficient d'aspect. La propriété `font-size-adjust` doit être définie à la valeur du coefficient d'aspect de la police préférentielle.

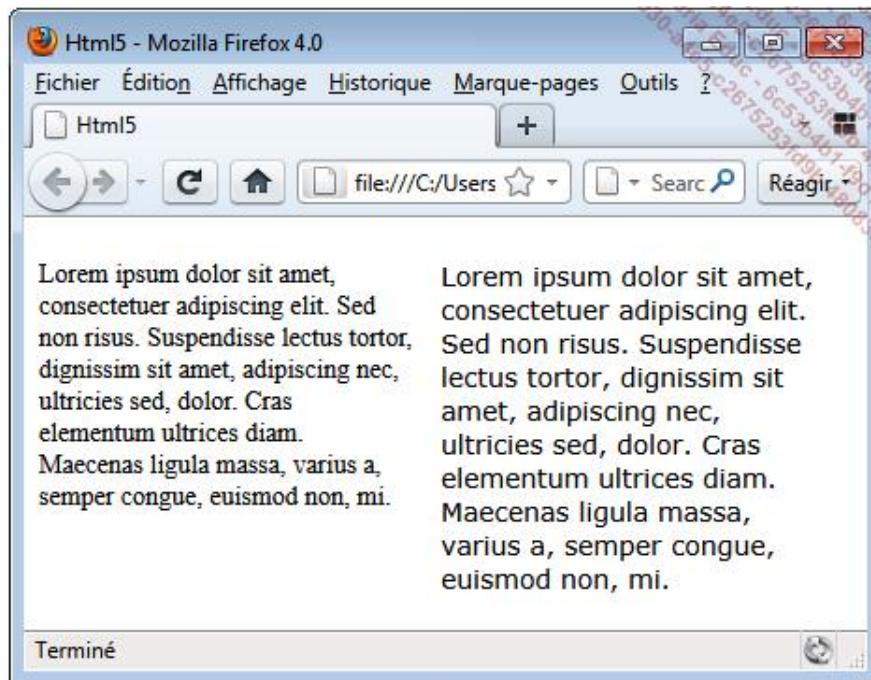
```
font-size-adjust: 0.465;
```

Ce coefficient d'aspect peut se définir par tâtonnements, mais il est plus simple de consulter des sites comme www.brunildo.org/test/aspect-table.html, qui sont dédiés à cette tâche.

Pour l'instant, seul Firefox 3.6+ a intégré la propriété `font-size-adjust`, pourtant bien utile.

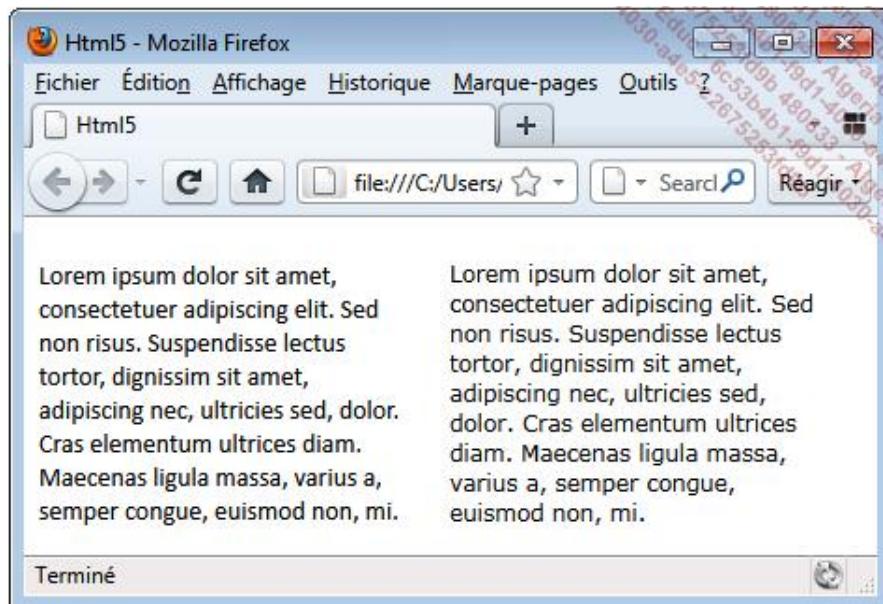
Exemple sans `font-size-adjust`

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
div { width: 200px;
      float: left;
      margin-right: 15px; }
#calibri p { font: 14px Calibri sans-serif; }
#verdana p { font: 14px Verdana, sans-serif; }
</style>
</head>
<body>
<div id="calibri">
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam. Maecenas ligula massa, varius a, semper congue, euismod non,
mi. </p>
</div>
<div id="verdana">
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam. Maecenas ligula massa, varius a, semper congue, euismod non,
mi. </p>
</div>
</body>
</html>
```



Exemple avec font-size-adjust

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Htm5</title>
<meta charset="iso-8859-1">
<style type="text/css">
div { width:200px;
      float:left;
      margin-right:20px;}
#calibri p { font:14px Calibri, sans-serif;}
#verdana p { font: 14px Verdana, sans-serif;
              font-size-adjust: 0.465;}
</style>
</head>
<body>
<div id="calibri">
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam. Maecenas ligula massa, varius a, semper congue, euismod non,
mi. </p>
</div>
<div id="verdana">
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor. Cras elementum ultrices
diam. Maecenas ligula massa, varius a, semper congue, euismod non,
mi. </p>
</div>
</body>
</html>
```



Pour une information plus fouillée sur la propriété font-size-adjust, consultez la page www.babylon-design.com/site/index.php/2010/04/12/262-mais-c-est-quoi-au-juste-font-size-adjust.

Les arrière-plans multiples

Les CSS 3 rendent possible l'affichage de plusieurs images dans un même arrière-plan en permettant de cumuler les valeurs au sein des propriétés `background-image`, `background-position` et `background-repeat` (Partie CSS - Les arrière-plans).

Cet effet n'était possible qu'en superposant des divisions `<div>` définies en position absolue, un peu à la manière des calques dans les applications graphiques.

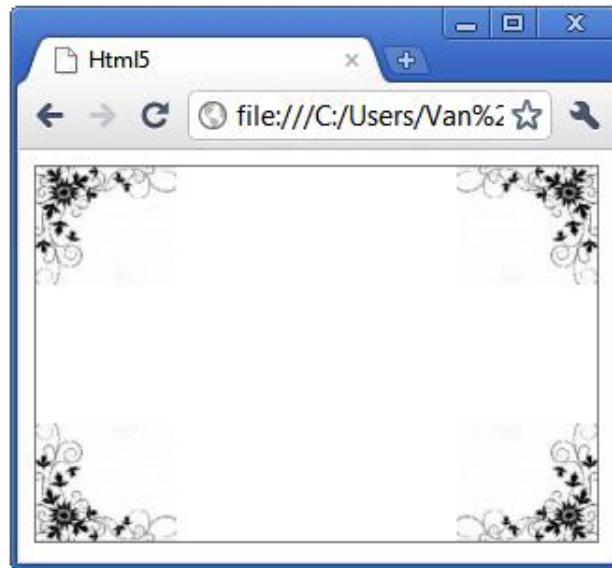
Tous les navigateurs de notre étude reconnaissent les arrière-plans multiples, soit Internet Explorer 9, Firefox 3.6+, Opera 10.1+, Safari 4+ et Chrome 4+.

Exemple

Soit les images flore1.png, flore2.png, flore3.png et flore4.png :



```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#box { width:300px; height: 200px;
       border: 1px solid gray;
       background: url(flore1.png) top left no-repeat,
                   url(flore2.png) top right no-repeat,
                   url(flore3.png) bottom right no-repeat,
                   url(flore4.png) bottom left no-repeat;}
</style>
</head>
<body>
<div id="box"></div>
</body>
</html>
```



Le dégradé de couleurs

Les standards CSS pour les dégradés de couleurs ne sont pas encore finalisés. Ce qui n'a pas empêché Firefox, Safari et Google Chrome de les implanter.

Qu'en est-il des préfixes `-moz` et `-webkit`? Pour cette propriété, la définition des préfixes est fort différente.

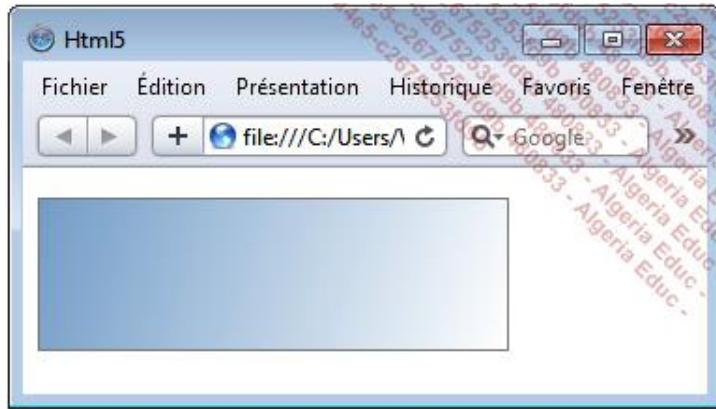
Pour webkit

Les dégradés linéaires, pour les navigateurs Safari et Chrome, s'écrivent selon la propriété :

background:	<code>-webkit-gradient(linear, origine, fin, from(couleur), to(couleur));</code>
	<p>où :</p> <ul style="list-style-type: none">• <code>linear</code> pour les dégradés linéaires.• <code>origine</code> est le point de départ du dégradé. Celui-ci est déterminé par deux valeurs séparées par un espace. La syntaxe supporte des nombres, des pourcentages ou les mots-clés <code>top</code>, <code>right</code>, <code>bottom</code> et <code>left</code>. Soit, par exemple, <code>top left</code>.• <code>fin</code> est le point de fin du dégradé. Celui-ci est déterminé par deux valeurs séparées par un espace. La syntaxe supporte des nombres, des pourcentages ou les mots-clés <code>top</code>, <code>right</code>, <code>bottom</code> et <code>left</code>. Soit, par exemple, <code>bottom right</code>.• <code>from(couleur)</code>, la couleur de début du dégradé.• <code>to(couleur)</code>, la couleur de fin du dégradé.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#box { width:250px; height: 80px;
       border: 1px solid gray;
       background: -webkit-gradient(linear, left top, right
                                     bottom, from(#7DA5CD), to(white));}
</style>
</head>
<body>
<div id="box"></div>
</body>
</html>
```

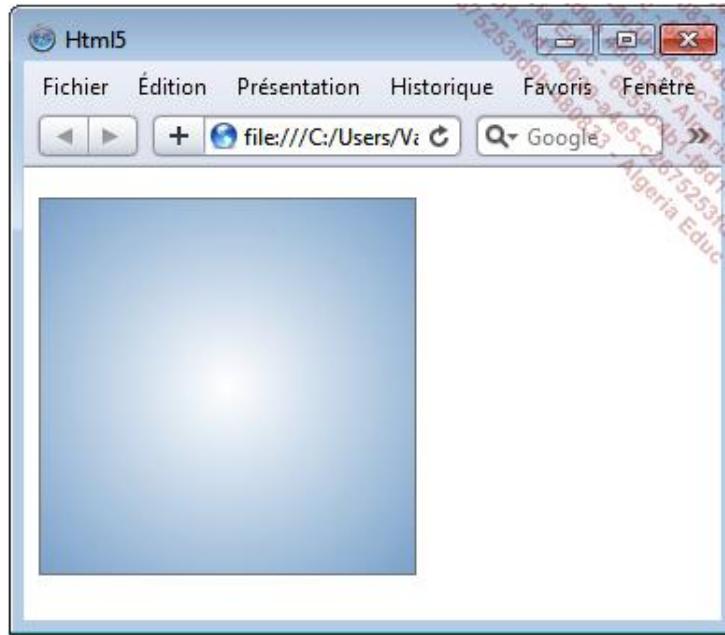


Pour les dégradés circulaires, la propriété est :

<p>background:</p>	<pre>-webkit-gradient(radial, origine, radius1, fin, radius2, from(couleur), to(couleur));</pre> <p>où :</p> <ul style="list-style-type: none"> • radial pour les dégradés circulaires. • origine est le point de départ du dégradé, soit du cercle intérieur. Celui-ci est déterminé par deux valeurs séparées par un espace. La syntaxe supporte des nombres, des pourcentages ou les mots-clés <code>top</code>, <code>right</code>, <code>bottom</code>, <code>left</code> et <code>center</code>. Soit, par exemple, <code>center center</code>. • radius1 est le radius du cercle intérieur. • fin est le point de fin du dégradé. Celui-ci est déterminé par deux valeurs séparées par un espace. La syntaxe supporte des nombres, des pourcentages ou les mots-clés <code>top</code>, <code>right</code>, <code>bottom</code>, <code>left</code> et <code>center</code>. • radius 2 est le radius du cercle extérieur. • from(couleur), la couleur de début du dégradé, soit du cercle intérieur. • to(couleur), la couleur de fin du dégradé, soit du cercle extérieur.
--------------------	--

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#box { width:200px; height: 200px;
        border: 1px solid gray;
        background: -webkit-gradient(radial, center center, 0,
                                      center center, 150, from(white), to(#7DA5CD)); }
</style>
</head>
<body>
<p></p>
<div id="box"></div>
</body>
</html>
```



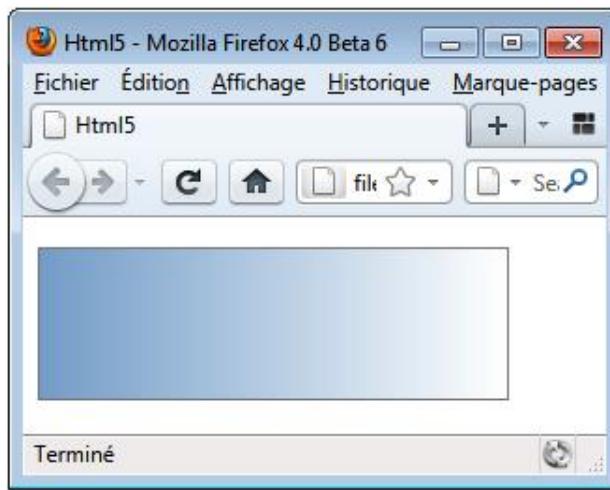
Pour Mozilla Firefox

Pour les dégradés linéaires :

background:	-moz-linear-gradient(point d'origine, couleur de départ, couleur fin);
où	<ul style="list-style-type: none">• linear pour les dégradés linéaires.• point d'origine est à choisir entre top, right, bottom ou left ou une paire de valeurs (séparées par un espace) élaborée à partir de top, right, bottom ou left.• couleur d'origine pour couleur de départ du dégradé.• couleur fin pour la couleur avec laquelle se termine le dégradé.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#box { width:250px; height: 80px;
       border: 1px solid gray;
       background: -moz-linear-gradient(top left, #7DA5CD,
                                         #ffffff);}
</style>
</head>
<body>
<div id="box"></div>
</body>
</html>
```

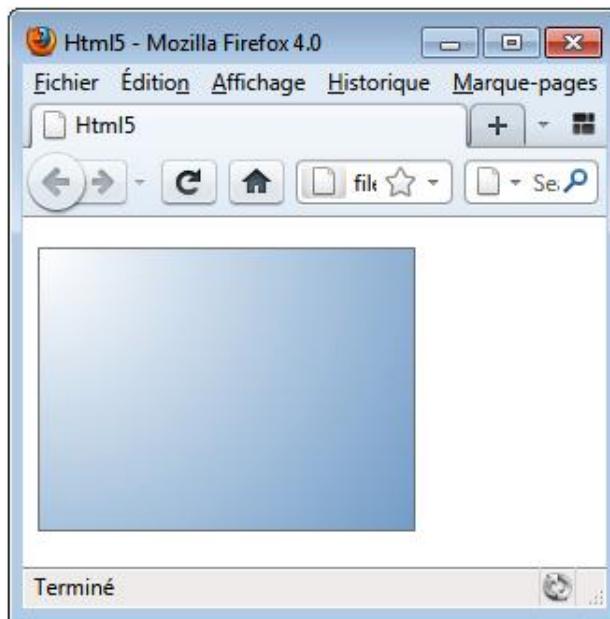


Pour les dégradés circulaires :

background:	-moz-radial-gradient(position, forme, couleur début, couleur fin); où <ul style="list-style-type: none">• radial pour les dégradés circulaires.• position est le point central du gradient radial. Mot-clé à choisir entre top, right, bottom, left, center ou une paire de valeurs (séparées par un espace) élaborée à partir de top, right, bottom, left ou center. Par exemple, center center.• forme, la forme du gradient radial soit circle (circulaire) ou ellipse (elliptique).• couleur début, couleur fin. La couleur de début du dégradé et celle de fin.
-------------	--

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#box { width:200px; height: 150px;
       border: 1px solid gray;
       background: -moz-radial-gradient(top left, circle, white,
                                         #7DA5CD);}
</style>
</head>
<body>
<p></p>
<div id="box"></div>
</body>
</html>
```



Mozilla Firefox ajoute encore bien d'autres options permettant de créer des gradients de plus en plus complexes.

Pour Internet Explorer 8+

Internet Explorer ne reprend pas la propriété de style `gradient`. Il se rabat sur un filtre propriétaires Microsoft.

```
filter: progid:DXImageTransform.Microsoft.Gradient(StartColorStr='#00FF00',
EndColorStr='#000000', GradientType=1);
```

Avec `StartColorStr` pour la couleur de départ, `EndColorStr` pour la couleur de fin et `gradientType` avec une valeur 0 pour les dégradés verticaux et une valeur de 1 pour les dégradés horizontaux.

Le code d'un dégradé compatible serait :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#box { width:250px; height: 80px;
       border: 1px solid gray;
       background-color: #7DA5CD;
       filter:progid:DXImageTransform.Microsoft.Gradient(
       StartColorStr='#7DA5CD', EndColorStr='#ffffff',
       GradientType=1);
       background: -moz-linear-gradient(top left, #7DA5CD,
       #ffffff);
       background: -webkit-gradient(linear, left top, right
       bottom, from(#7DA5CD), to(white));}
</style>
</head>
<body>
<div id="box"></div>
</body>
</html>
```

L'opacité ou la transparence

Avec les CSS3, il est possible de faire varier l'opacité ou la transparence d'un élément.

Cette propriété a été adoptée très tôt et est ainsi compatible avec Firefox 1+, Safari 1+, Chrome 1+ et Opera 9+.

Il n'est donc pas nécessaire de se préoccuper des préfixes `-moz` et `-webkit`.

opacity:	Valeur comprise entre 1 et 0. Avec la valeur 1, l'opacité est complète et la transparence nulle. Avec 0, l'opacité est nulle et l'élément complètement transparent.
----------	---

Commentaires

- Le résultat de la propriété `opacity` est assez proche de celui de la notation de couleur `RGBa` avec transparence. Il faut cependant noter que la propriété `opacity` s'applique à l'élément dans son intégralité, soit à tous ses descendants. `RGBa` ne s'applique uniquement qu'à l'élément sélectionné.
- L'élément doté de la propriété `opacity` est translucide. Il laisse apparaître l'élément qui se trouve éventuellement situé en dessous de lui (voir exemple ci-après).

À ce stade de notre étude des CSS3, on ne s'étonnera plus du fait qu'Internet Explorer ne reprenne pas la propriété `opacity`. Il est nécessaire de passer par le filtre `alpha`, propriétaire Microsoft.

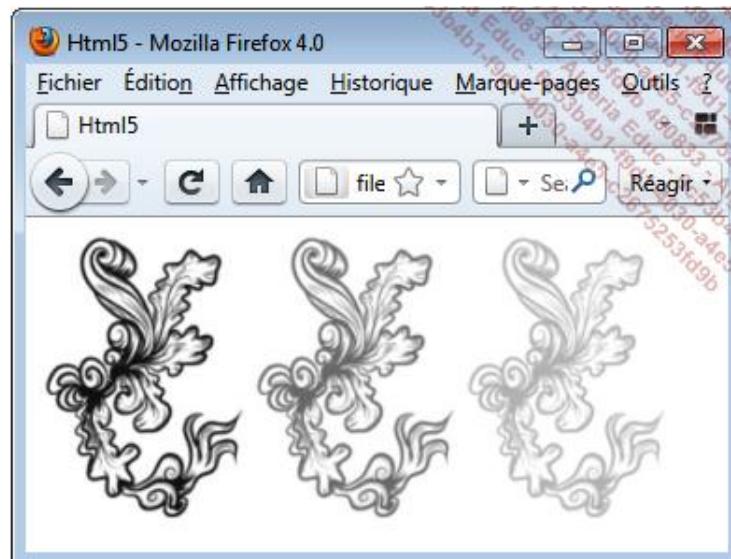
`filter:alpha(opacity=x)` où `x` est une valeur comprise entre 100 et 0.

Exemple 1

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<body>



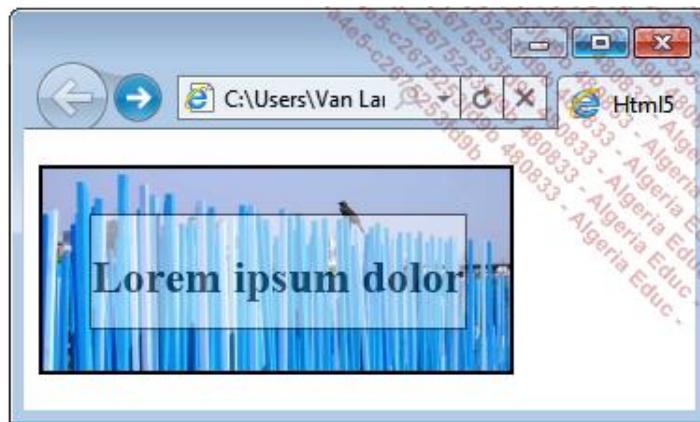
</body>
</html>
```



Exemple 2

Illustrons la translucidité.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
div.background { width: 250px; height: 108px;
    background:url(blue.png);
    border:2px solid black;}
div.box { width: 200px; height: 60px;
    margin:24px 25px;
    border:1px solid black;
    background-color:#ffffff;
    filter:alpha(opacity=50);
    opacity:0.5;
    text-align: center;}
</style>
</head>
<body>
<p></p>
<div class="background">
<div class="box">
<h2>Lorem ipsum dolor</h2>
</div>
</div>
</body>
</html>
```

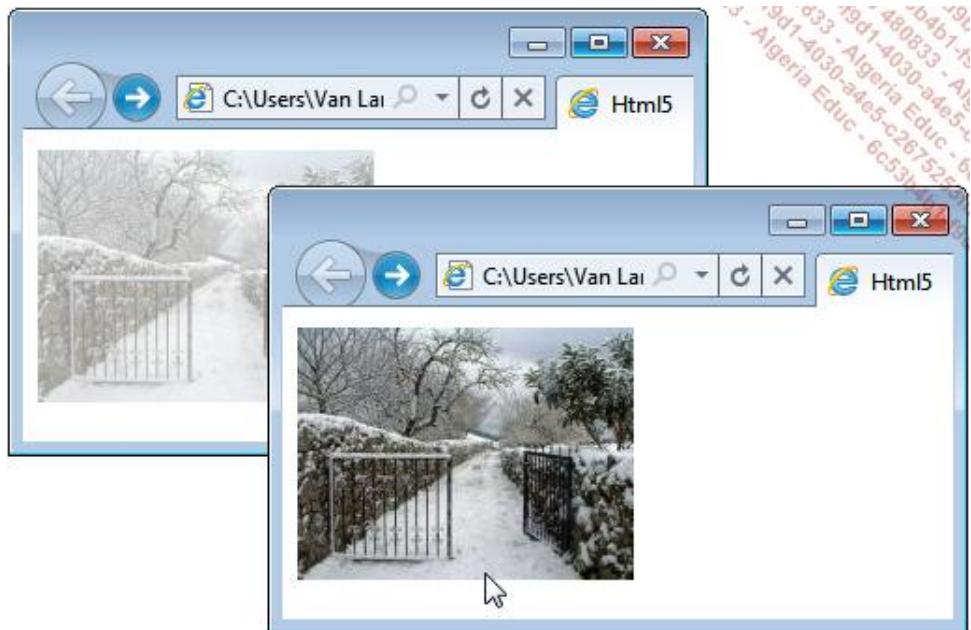


Exemple 3

Il est tentant de modifier l'opacité de l'image au survol du curseur de la souris.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<body>

</body>
</html>
```



Dans un premier temps, l'image s'affiche avec une opacité de 0.4. Au survol du curseur de la souris (`onmouseover`), celle-ci s'affiche avec une opacité maximale. Lorsque le curseur quitte celle-ci (`onmouseout`), l'image revient dans son état initial avec une opacité de 0.4.

Les colonnes multiples

Présenter un contenu en plusieurs colonnes comme avec Adobe InDesign est une autre nouveauté des CSS3.

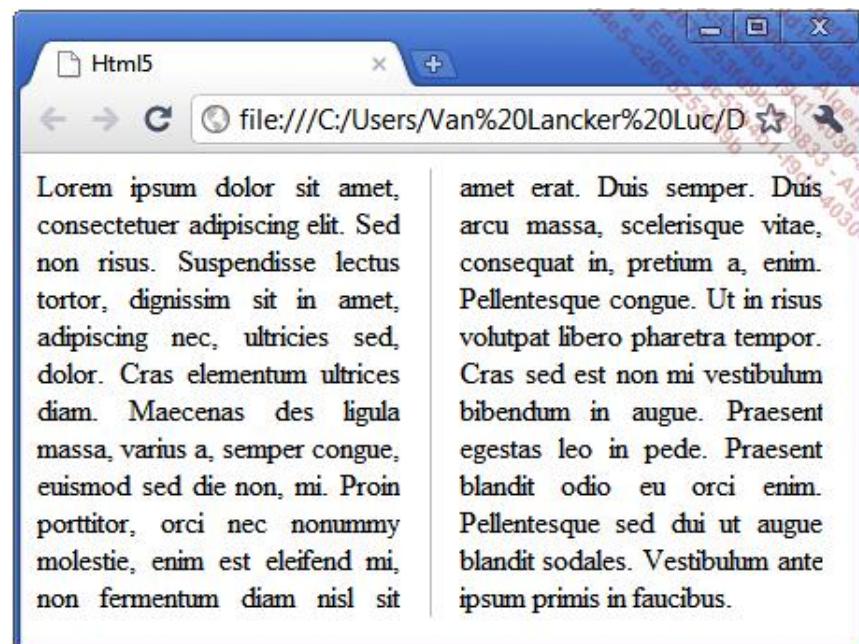
column-count:	Entier qui détermine le nombre de colonnes dans lesquelles sera affiché le contenu de l'élément.
column-width:	Valeur qui décrit la largeur optimale de chaque colonne (facultatif).
column-gap:	Valeur qui détermine l'espace ou le padding entre les colonnes.
column-rule:	Définit une bordure entre les colonnes.

À ce jour, les colonnes multiples ne sont pas reprises par Internet Explorer 9 et Opera 10.6. Seuls Firefox 3+, Safari 3.1+ et Chrome 4+ permettent cette mise en page mais avec respectivement les préfixes `-moz` et `-webkit`.

-moz-column-count:	-webkit-column-count:
-moz-column-width:	-webkit-column-width:
-moz-column-gap:	-webkit-column-gap:
-moz-column-rule:	-webkit-column-rule:

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#box { width:420px; height: 240px;
        text-align: justify;
        column-count: 2;
        column-gap: 2em;
        column-rule: 1px solid silver;
        -moz-column-count: 2;
        -moz-column-gap: 2em;
        -moz-column-rule: 1px solid silver;
        -webkit-column-count: 2;
        -webkit-column-gap: 2em;
        -webkit-column-rule: 1px solid silver; }
</style>
</head>
<body>
<div id="box">
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non
    risus. Suspendisse lectus tortor, dignissim sit in amet,
    adipiscing nec, ...
</div>
</body>
</html>
```



Transformations

Encore une propriété CSS3 dont les résultats sont pour le moins spectaculaires. En effet, elles brisent le carcan rectangulaire du design des pages Web. Cette propriété permet d'appliquer des transformations sur un élément comme la rotation, le décalage, le zoom ou la déformation oblique. Associé aux transitions (voir point suivant), ce binôme de choc risque de modifier profondément la mise en page des sites du futur.

En outre, si pour l'instant ces transformations sont en 2D, la 3D est déjà annoncée.

transform:	rotate(angle), rotateX(angle), rotateY(angle) scale(nombre), scaleX(nombre), scaleY(nombre) skew(angle), skewX(angle), skewY(angle) translate(longueur), translateX(longueur), translateY(longueur)
------------	--

Comme toujours avec les CSS3 expérimentales, la propriété `transform` se décline avec un préfixe selon les navigateurs :

-webkit-transform.
-moz-transform.
-o-transform.

La propriété `transform` est prise en charge par presque tous les navigateurs de notre étude (Firefox 3.6+, Opera 10.6+, Safari 4+, Chrome 4+). Elle est par contre encore ignorée par Internet Explorer 9.

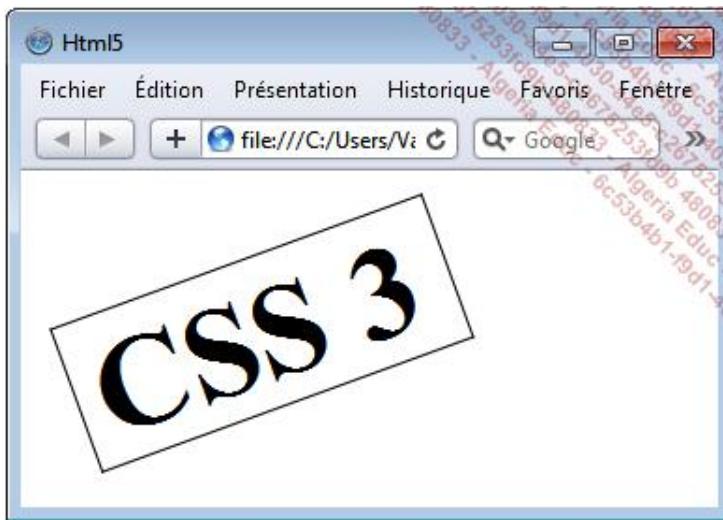
Passons en revue les différents paramètres de cette propriété `transform`.

Rotation

La rotation d'un élément est obtenue par le paramètre `rotate(angle)` où l'angle peut être défini en degrés (deg), radians (rad) ou gradients (grad). Les valeurs positives effectuent la rotation de l'élément dans le sens des aiguilles d'une montre et les valeurs négatives tournent celui-ci dans le sens inverse. Lorsqu'une seule valeur est indiquée, celle-ci prévaut pour les deux axes X et Y. Avec deux valeurs, celles-ci s'appliquent à l'axe des X et des Y.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
body { padding-left: 15px; }
#box { width:200px; height: 80px;
        font-size: 70px;
        border: 1px solid black;
        padding-left: 10px;
        -moz-transform: rotate(-20deg);
        -webkit-transform: rotate(-20deg);
        -o-transform: rotate(-20deg);
        transform: rotate(-20deg); }
</style>
</head>
<body>
<h1 id="box">CSS 3</h1>
</body>
</html>
```

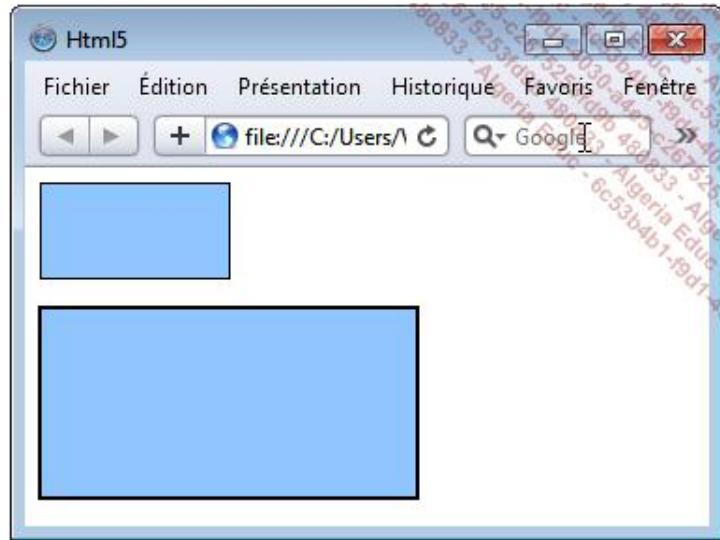


Modification d'échelle

La modification de l'échelle d'un élément est obtenue par le paramètre `scale(nombre)` où le nombre peut être un entier ou un décimal, positif ou négatif. Les valeurs positives effectuent un zoom avant et les valeurs négatives, un zoom arrière. Lorsqu'une seule valeur est indiquée, celle-ci prévaut pour les deux axes X et Y. Avec deux valeurs, celles-ci s'appliquent à l'axe des X et des Y.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
div { border: 1px solid black;
      background-color: #9cf;
      width:100px; height: 50px;}
#box { margin: 40px 50px;
        -moz-transform: scale(2);
        -webkit-transform: scale(2);
        -o-transform: scale(2);
        transform: scale(2);}
</style>
</head>
<body>
<div></div>
<div id="box" style=""></div>
</body>
</html>
```

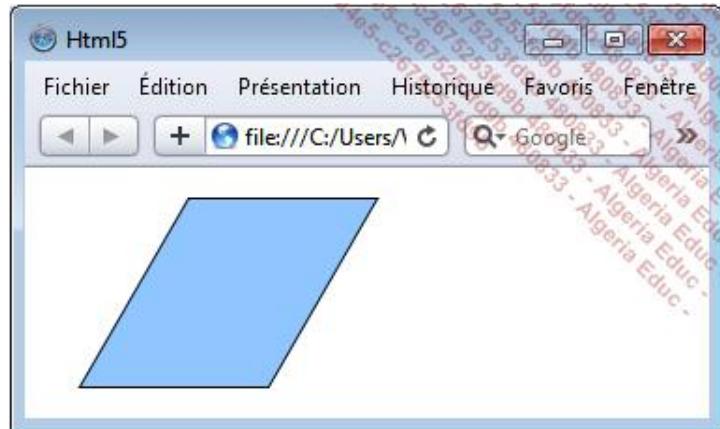


Déformation oblique

Une déformation oblique d'un élément est obtenue par le paramètre `skew(angle)` où l'angle peut être défini en degrés (deg), radians (rad) ou gradiants (grad). Les valeurs positives effectuent la rotation de l'élément dans le sens des aiguilles d'une montre et les valeurs négatives tournent celui-ci dans le sens inverse. Lorsqu'une seule valeur est indiquée, celle-ci prévaut pour les deux axes X et Y. Avec deux valeurs, celles-ci s'appliquent à l'axe des X et des Y.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#box { width: 100px; height: 100px;
       border: 1px solid black;
       background-color: #9cf;
       margin-left: 50px;
       -moz-transform: skew(-30deg);
       -webkit-transform: skew(-30deg);
       -o-transform: skew(-30deg);
       transform: skew(-30deg); }
</style>
</head>
<body>
<p></p>
<div id="box" style=""></div>
</body>
</html>
```

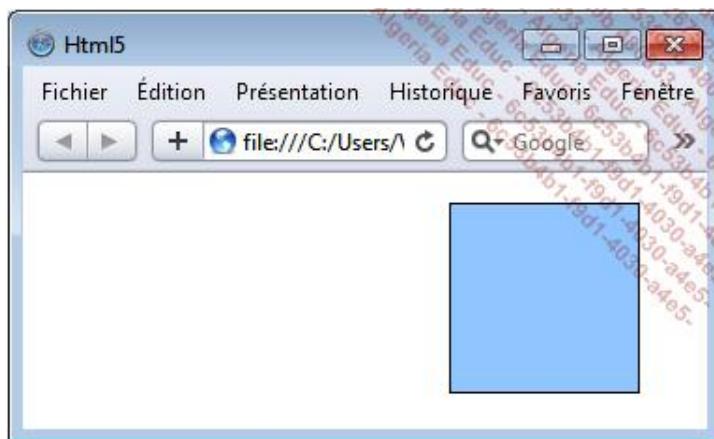


Déplacement

Le déplacement d'un élément est obtenu par le paramètre `translate(longueur)` où la longueur peut être définie en px, %, em, in, mm, cm. Les valeurs positives effectuent, sur l'axe des X, un déplacement vers la droite et vers la gauche pour les valeurs négatives. Pour l'axe des Y, une valeur positive effectue le déplacement vers le haut et une valeur négative vers le bas. Lorsqu'une seule valeur est indiquée, celle-ci prévaut pour les deux axes X et Y. Avec deux valeurs, celles-ci s'appliquent à l'axe des X et des Y.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
#box { width: 100px; height: 100px;
       border: 1px solid black;
       background-color: #9cf;
       -moz-transform: translate(220px, 0);
       -webkit-transform: translate(220px, 0);
       -o-transform: translate(220px, 0);
       transform: translate(220px, 0);
}
</style>
</head>
<body>
<p></p>
<div id="box" style=""></div>
</body>
</html>
```



Pour tester toutes les transformations possibles, ne manquez pas de consulter le site <http://www.westciv.com/tools/transforms/>.

Les transitions

Les propriétés CSS3 transition apportent du mouvement dans le design de la page, à l'instar de ce qui se réalise avec du JavaScript.

Les animations se réalisent principalement à partir de quatre propriétés :

transition-property	Précise les propriétés CSS à animer, par exemple, la couleur (<code>color</code>) et la largeur (<code>width</code>). Ces propriétés sont précisées en les listant, séparées par des virgules. Le mot-clé <code>all</code> (défaut) désigne toutes les propriétés animables de l'élément.
transition-duration	Précise la durée de la transition. Celle-ci est exprimée en s (seconde) ou ms (milliseconde).
transition-timing-function	Précise la fonction de transition à utiliser : <code>ease</code> : rapide sur le début et ralenti sur la fin. <code>linear</code> : vitesse constante sur toute la durée de l'animation. <code>ease-in</code> : lent au début et accélère de plus en plus vers la fin. <code>ease-out</code> : rapide sur le début et décélère sur la fin. <code>ease-in-out</code> : le départ et la fin sont lents.
transition-delay	Précise le retard (ou l'avance) du départ de la transition. Celui-ci est exprimé en s (seconde) ou ms (milliseconde).

Le raccourci `transition` évite de reprendre chaque propriété.

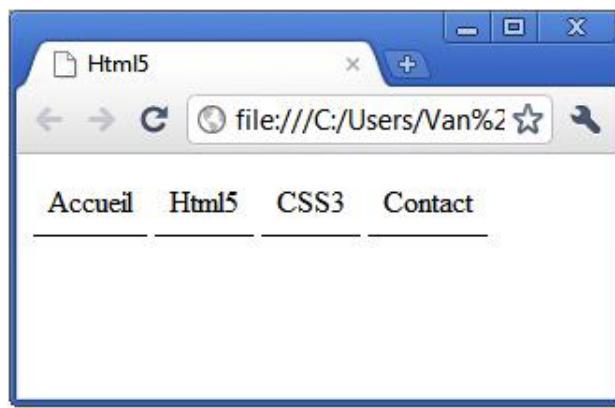
```
transition: width 2s ease, height 3s linear;
```

La propriété `transition` est prise en charge par presque tous les navigateurs de notre étude (Firefox 4+, Opera 10.6+, Safari 4+, Chrome 4+). Elle est par contre ignorée par Internet Explorer 9.

Exemple 1

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.transition a { padding: 8px;
    text-decoration:none;
    color: black;
    border-bottom:1px solid black;
    -webkit-transition: border 1s ease-out ;
    -moz-transition: border 1s ease-out;
    -o-transition: all 1s ease-out;
    transition: border 1s ease-out;}
.transition a:hover{ border-bottom: 50px solid #9cf; }
</style>
</head>
<body>
<nav class="transition">
<a href="#">Accueil</a>
<a href="#">Html5</a>
<a href="#">CSS3</a>
<a href="#">Contact</a>
</nav>
</body>
</html>
```

La situation de départ :



Au survol de la souris :



Exemple 2

La présentation de contenu sous forme d'accordéon connaît un franc succès sur la toile. La propriété CSS3 transform permet de réaliser cet effet uniquement par des feuilles de style, sans faire appel à du code JavaScript.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Html5</title>
<meta charset="iso-8859-1">
<style type="text/css">
.verticalaccordion>ul { margin: 0;
                        padding: 0;
                        list-style:none;
                        width: 300px; }

.verticalaccordion>ul>li {
display:block;
overflow: hidden;
margin: 0;
padding: 0;
list-style:none;
height:30px;
width: 300px;
background-color: rgb(215,230,245);
-webkit-border-radius: 7px;
-moz-border-radius: 7px;
border-radius: 7px;
transition: height 0.3s ease-in-out;
-moz-transition: height 0.3s ease-in-out;
-webkit-transition: height 0.3s ease-in-out;
-o-transition: height 0.3s ease-in-out; }

.verticalaccordion>ul>li>h3 {
display:block;
```

```

margin: 0;
padding:10px;
height:19px;
border-top:#f0f0f0 1px solid;
font-family: Arial, Helvetica, sans-serif;
font-size: 80%;
color: #000;
background: rgb(185,205,225);}
.verticalaccordion>ul>li>div { margin:0;
                                overflow: auto;
                                padding:10px;
                                height:150px;}
.verticalaccordion>ul>li:hover { height: 150px;}
.verticalaccordion:hover>ul>li:hover>h3 {
color:#fff;
background: rgb(125,165,205);
font-size: 80%; }
.verticalaccordion>ul>li>h3:hover { cursor:pointer; }
</style>
</head>
<body>
<div class="verticalaccordion">
<ul>
<li>
<h3>Item 1</h3>
<div>


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non  
risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing  
nec, ultricies sed, dolor. Cras elementum ultrices diam.


</div>
</li>
<li>
<h3>Item 2</h3>
<div>


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non  
risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing  
nec, ultricies sed, dolor. Cras elementum ultrices diam.


</div>
</li>
<li>
<h3>Item 3</h3>
<div>


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non  
risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing  
nec, ultricies sed, dolor. Cras elementum ultrices diam.


</div>
</li>
<li>
<h3>Item 4</h3>
<div>


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non  
risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing  
nec, ultricies sed, dolor. Cras elementum ultrices diam.


</div>
</li>
</ul>
</div>
</body>
</html>

```

