13

Créer une mise en page : le dimensionnement et le positionnement

Les méthodes de mise en page au moyen de cadres (telles que nous les avons vues au chapitre 8) ou à l'aide de tableaux (chapitre 6), outre qu'elles sont aujourd'hui considérées comme obsolètes, ne manquent pas de présenter des inconvénients importants. Si les cadres gênent considérablement l'indexation correcte des pages web, les mises en page à base de tableaux, pour pratiques qu'elles aient l'air à première vue, impliquent des structures trop complexes qui nuisent à la lisibilité de l'organisation et à la maintenance des sites. Nous allons voir dans ce chapitre que tous ces problèmes peuvent être contournés en utilisant conjointement le dimensionnement et le positionnement des éléments qui contiennent les différentes zones d'une page. Ces méthodes apportent enfin des réponses simples permettant de créer les mises en page les plus diverses, et qui plus est en respectant le principe, fondamental en XHTML, de séparation du contenu et de la présentation. Nous pouvons par exemple obtenir des présentations différentes à partir du même code XHTML en modifiant simplement les styles CSS attachés à ces différents composants.

Le dimensionnement des éléments

Au chapitre 11, dans la définition du modèle de boîte de CSS, nous avons vu que les propriétés `width` et `height` permettent de déterminer respectivement la largeur et la hauteur d'un élément. Elles s'appliquent à tous les éléments sauf les éléments en ligne non remplacés et les lignes et groupes de lignes des tableaux. Nous rappelons leurs syntaxes :

```
width: <longueur> | NN% | auto | inherit
height: <longueur> | NN% | auto | inherit
```

Si la définition de la largeur ne pose généralement pas de problème, celle de la hauteur peut engendrer des effets particuliers, voire conflictuels entre les boîtes créées pour les éléments successifs et leurs contenus. L'exemple 13-1 en est une illustration concrète. Le corps du document contient une division `<div>` (repère ❹) qui inclut successivement un titre `<h1>` (repère ❺), deux paragraphes `<p>` (repères ❻ et ❼). Les paragraphes incluent pour leur part du texte ordinaire. L'élément `<div>` est suivi d'un second paragraphe ne contenant que du texte (repère ❽).

Les définitions de styles fixent les dimensions de l'élément `<div>` qui a une largeur de 80 % et une hauteur de 500 pixels (repère ❶). Les éléments `<p>` ont une largeur de 75 % de celle de leur conteneur (repère ❷), soit 75 % de 80 % (donc 60 % de la largeur de la page) pour le premier et 75 % de la page pour le second. Ces largeurs sont effectivement respectées par tous les navigateurs (voir la figure 13-1). De même, l'élément `<h1>` a une largeur réelle de 80 % de celle de son parent `<div>`, donc 80 % de 80 %, soit 64 % de celle de la page.

Mais c'est au niveau de la définition des hauteurs que se posent les problèmes. Nous pouvons remarquer d'emblée que la hauteur de l'élément `<div>` est supérieure à la somme des hauteurs de ses éléments enfants `<p>` (200 pixels chacun) et `<h1>` (50 pixels). Le résultat obtenu visible à la figure 13-1 montre tout d'abord que la hauteur définie pour `<div>` est bien respectée. Celles des deux paragraphes sont bien de 200 pixels, la couleur de fond qui leur est attribuée nous permettant de le vérifier. Il en est de même pour la hauteur de l'élément `<h1>`. En revanche, si le texte du premier paragraphe s'affiche bien dans la boîte assignée à l'élément `<p>`, celui du deuxième déborde, et pire encore il se superpose au contenu du troisième. Le résultat obtenu est évidemment catastrophique au niveau de la présentation.

Exemple 13-1. Les problèmes de dimensionnement en hauteur

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Dimensionnement des éléments</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
```

```
 <style type="text/css" >
  body {font-family: Arial sans-serif;font-size: 16px;}
  div{width:80%;height:500px;background-color:#EEE;}❶
  p{width:75%;height:200px;background-color:#BBB;}❷
  h1{width: 80%; height: 50px;background-color:#888;color:white;}❸
 </style>
</head>
<body>
 ❹<div>
  ❺<h1>Dimensionnement des éléments</h1>
  ❻<p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
  ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
  </p>
  ❼<p>Deus duo magna luminaria luminare maius ut praeesset diei et luminare minus
  ➥ ut praeesset nocti et stellas et posuit eas in firmamento caeli . . .</p>
    </div>
  ❽<p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
  ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas . . .
 </p>
</body>
</html>
```

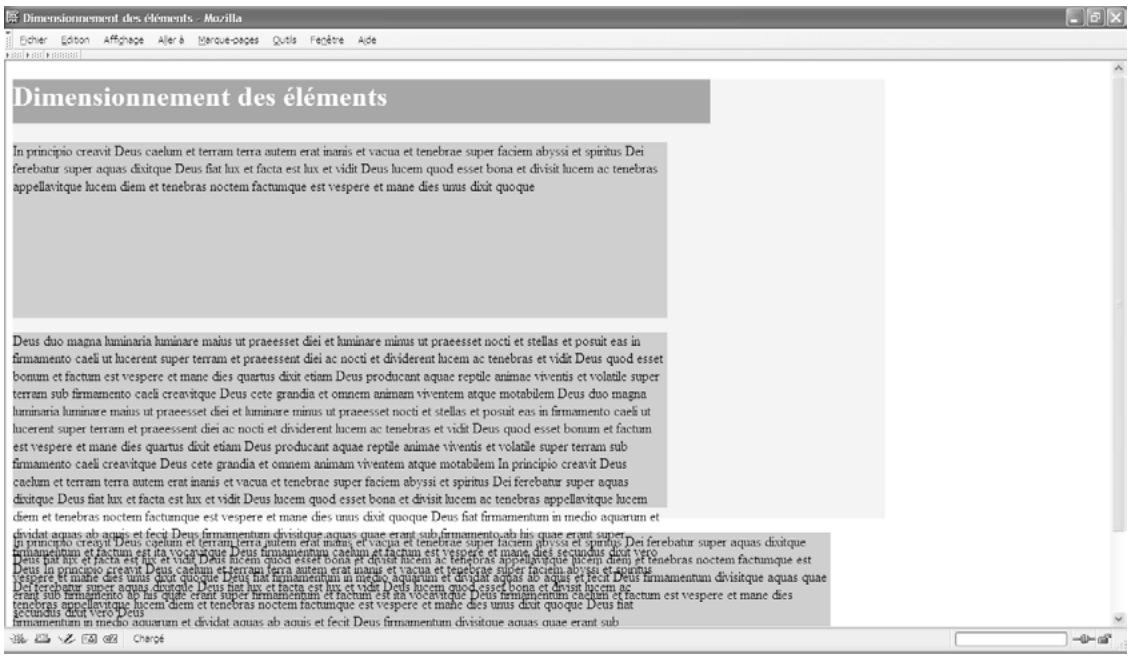


Figure 13-1

Les problèmes dus au dimensionnement

Nous pouvons corriger facilement ce résultat en donnant la valeur `auto` à la propriété `height` des éléments `<div>` et `<p>` (repères ❶ et ❷). L'élément `<style>` devient donc :

```
<style type="text/css" >
 body {font-family: Arial sans-serif;font-size: 16px;}
 div{width:80%;height:auto ❶ ;background-color:#EEE;}
 p{width:75%;height:auto ❷ ;background-color:#BBB;}
 h1{width: 80%; height: 50px;background-color:#888;color:white;}
</style>
```

Nous obtenons alors le dimensionnement présenté à la figure 13-2, dans lequel chaque paragraphe occupe la place qui lui est nécessaire pour afficher son contenu.

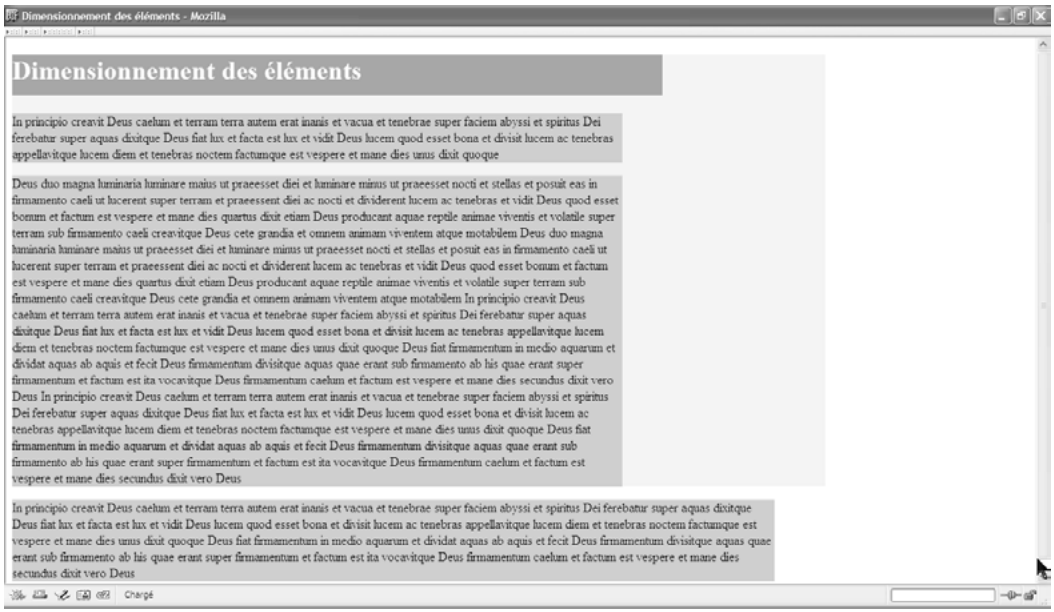


Figure 13-2

Le dimensionnement avec la valeur auto

L'utilisation de la valeur `auto` ne permet pas une mise en page précise des différents éléments dans le document, en particulier si leur contenu est créé dynamiquement, en provenance d'une base de données, suite à une recherche déclenchée par le visiteur par exemple. S'il semble nécessaire au programmeur de définir impérativement une hauteur à chaque élément, il est possible de conserver cette présentation en appliquant la propriété `overflow` pour gérer les éventuels débordements des contenus. Sa syntaxe est la suivante :

```
overflow : visible | hidden | scroll | auto | inherit
```

Voici le rôle imparti à chacune de ses valeurs :

- `visible` : le contenu débordant est affiché.

- `hidden` : le contenu débordant est caché et donc illisible.
- `scroll` : des barres de défilement horizontales et verticales apparaissent sur les côtés droit et bas de la boîte de l'élément, ce qui permet d'accéder au contenu débordant. Cette valeur a l'inconvénient de laisser apparaître ces barres même si le contenu ne déborde pas.
- `auto` : le navigateur fait apparaître les barres de défilement en cas de débordement uniquement.

En modifiant les styles de l'exemple 13-1 de la manière suivante :

```
<style type="text/css" >
 body {font-family: Arial sans-serif;font-size: 16px;}
 div{width:80%;height:500px;background-color:#EEE;}
 p{width:75%;height: 200px;background-color:#BBB;overflow: auto;❶}
 h1{width: 80%; height: 50px;background-color:#888;color:white;}
</style>
```

dans laquelle nous définissons la propriété `overflow` pour les paragraphes avec la valeur `auto` (repère ❶), nous obtenons le résultat présenté à la figure 13-3 qui montre que la hauteur définie pour les éléments `<p>` est conservée à 200 pixels, mais que le contenu du deuxième paragraphe est lisible en utilisant les barres de défilement. Les intentions de mise en page sont donc préservées et le contenu est entièrement accessible.

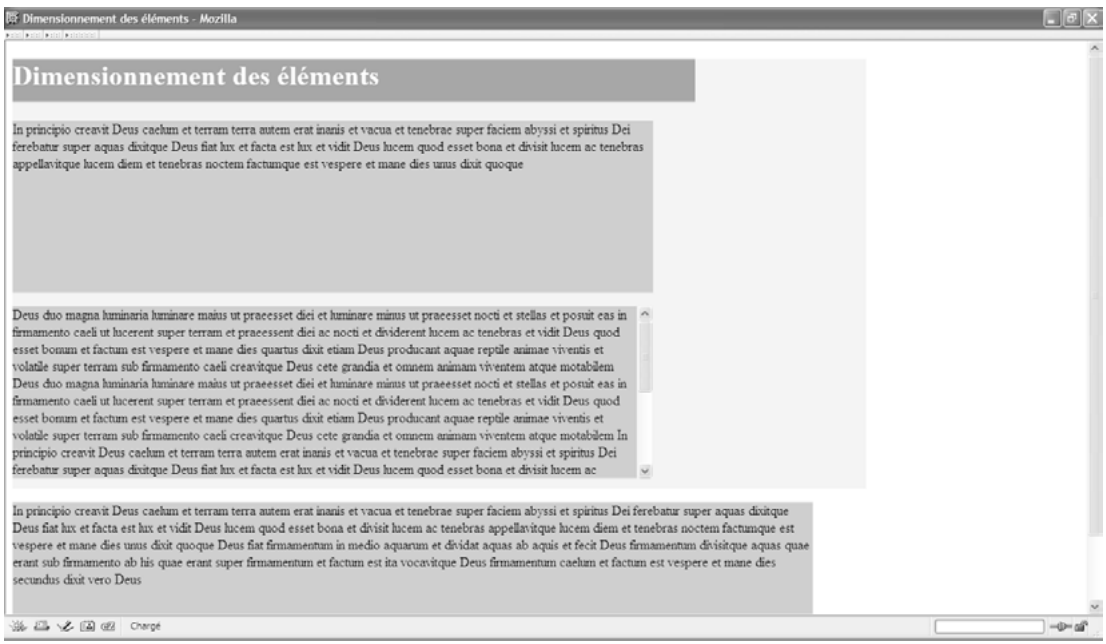


Figure 13-3

Gestion des débordements

Dans les cas précédents, et en gérant les débordements avec la propriété `overflow`, tous les paragraphes ont les mêmes dimensions quels que soient leurs contenus. Dans notre dernier exemple, le premier et le second ont une hauteur de 200 pixels alors que le premier ne contient que quelques lignes. Pour pouvoir intervenir sur les hauteurs des éléments sans pour autant les fixer de manière absolue, nous pouvons définir une hauteur minimale et une hauteur maximale pour tous les éléments, à l'exception des éléments en ligne non remplacés et des éléments de tableau.

Nous disposons pour cela des propriétés `min-height` et `max-height` dont la syntaxe est la suivante :

```
min-height: <longueur> | NN% | inherit
max-height: <longueur> | NN% | none |inherit
```

Le paramètre `<longueur>` est donné comme d'habitude par un nombre et une unité, et les pourcentages sont donnés en référence à la hauteur du bloc conteneur. La valeur `none` (qui est la valeur par défaut pour la hauteur maximale) indique que la valeur limite est celle du bloc conteneur.

De même, une largeur minimale et une largeur maximale peuvent être indiquées pour les mêmes éléments à l'aide des propriétés `min-width` et `max-width` dont les syntaxes sont similaires aux précédentes :

```
min-width: <longueur> | NN% | inherit
max-width: <longueur> | NN% | none |inherit
```

L'exemple 13-2 nous permet d'affiner les mises en pages obtenues dans les exemples précédents. L'élément `<div>` se voit affecté une largeur maximale de 900 pixels et une hauteur maximale de 800 pixels (repère ❶). Les paragraphes ont une hauteur minimale de 80 pixels et maximale de 250 pixels (repères ❷ et ❸), ainsi qu'une largeur comprise entre 500 et 600 pixels (repères ❹ et ❺). Les éventuels débordements sont gérés en utilisant la propriété `overflow` (repère ❻). L'élément `<h1>` a pour sa part une largeur comprise entre 400 et 500 pixels au maximum (repères ❼ et ❽). Comme nous pouvons le constater sur la figure 13-4, le contenu du titre `<h1>` est affiché sur deux lignes car sa largeur est limitée à 400 pixels. Le premier élément `<p>` (repère ❾) occupe juste la hauteur nécessaire à son contenu limité à quatre lignes. En revanche, le deuxième (repère ❿) a un contenu beaucoup plus long et il apparaît avec une hauteur de 250 pixels, ce qui est sa limite supérieure, et, comme son contenu déborde, des barres de défilement apparaissent automatiquement. Quant au troisième paragraphe (repère ⓫), il a le même comportement que le premier et sa hauteur est exactement adaptée à son contenu en respectant les contraintes de hauteur.

Exemple 13-2. Largeurs et hauteurs minimales et maximales

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Dimensionnement des éléments</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
```

```
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   body {font-family: Arial sans-serif;font-size: 16px;}
   div{max-width:900px;max-height:800px;background-color:#EEE;}❶
   p{min-height:80px ❷; max-height:250px ❸; min-width:500px ❹;
   ➥ max-width:600px ❺; background-color:#BBB;overflow: auto ❻;}
   h1{min-width: 400px ❼; max-width: 500px ❽; background-color:#888;color:white;}
  </style>
 </head>
 <body>
  <div>
  <h1>Dimensions mini et maxi des éléments XHTML</h1>
  ❾<p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
  ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
  ➥ . . . </p>
  ❿<p>Deus duo magna luminaria luminare maius ut praeesset diei et luminare minus
  ➥ ut praeesset nocti et stellas et posuit eas in firmamento caeli ut lucerent
  ➥ super terram et praeessent diei ac nocti et dividerent. . . </p>
  </div>
  ⓫<p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
  ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
  ➥ Deus fiat lux et facta est lux et vidit Deus lucem quod . . .</p>
 </body>
</html>
```

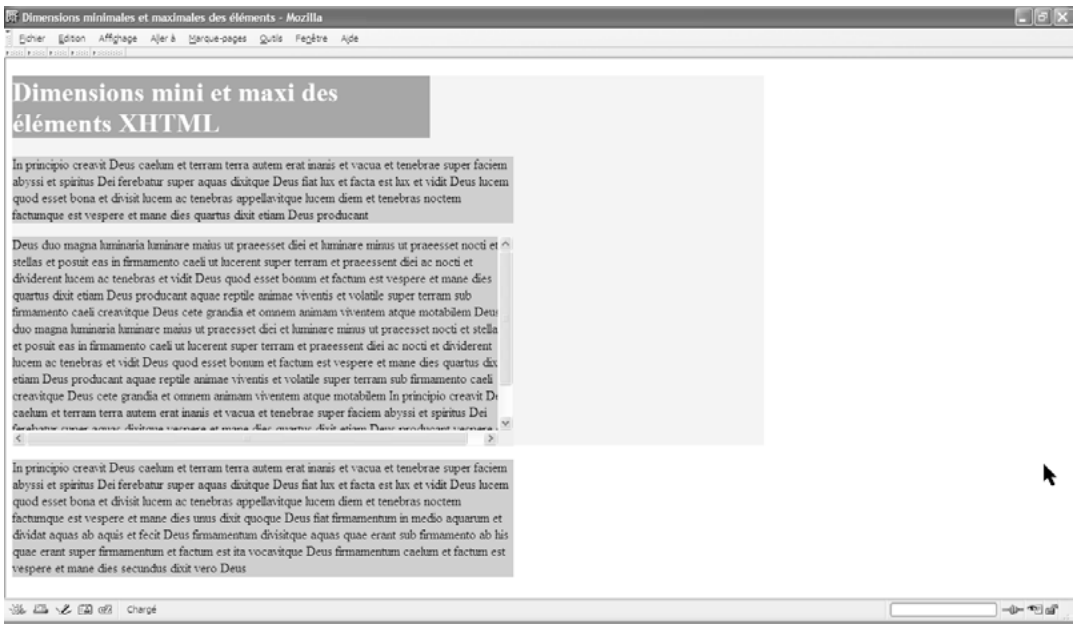


Figure 13-4

Les dimensions minimales et maximales

Le rendu des éléments

Dans la première partie de cet ouvrage, nous avons passé en revue l'ensemble des éléments XHTML et nous avons vu que chacun d'eux est associé par défaut à une présentation spécifique (à l'aide d'une feuille de style par défaut incluse dans les navigateurs : voir l'annexe B). Tous les éléments peuvent donc être classés en grands groupes de mise en page comme les blocs, les éléments en ligne, les listes ou les tableaux avec les caractéristiques qui s'y rattachent. Pour chaque élément, nous pouvons intervenir sur l'appartenance à un de ces groupes et modifier le rendu d'un élément en fonction des besoins. Un élément `<h1>` peut par exemple être transformé en élément en ligne ou en élément de liste alors qu'il est par défaut de niveau bloc.

Ces modifications sont opérées au moyen de la propriété `display` dont la syntaxe est la suivante :

```
display : none |inline | block | list-item |table | inline-table | table-row-group |
➥ table-header-group | table-footer-group | table-row | table-column-group | table-
➥ column | table-cell | table-caption | inherit
```

Sa valeur par défaut est `inline` mais la feuille de style par défaut la modifie et donne à chaque élément le style que nous lui connaissons. Ce sont ces valeurs par défaut sur lesquelles nous intervenons. Les principales valeurs qu'elle peut prendre ont la signification suivante :

- `none` : l'élément n'est pas affiché et la boîte qui lui est associée n'est pas créée. Tout se passe comme s'il n'existait pas dans le code XHTML.
- `inline` : l'élément devient du type en ligne (comme `<span>` par exemple).
- `block` : l'élément devient du type bloc (comme `<h1>`, `<p>`, `<div>`…).
- `list-item` : l'élément devient du type liste (équivalent de `<li>`).

Les autres valeurs sont rarement utilisées en XHTML, mais plutôt réservées à la création de styles destinés à des documents XML pour induire des comportements identiques à ceux des éléments de tableau XHTML. Le tableau suivant donne les correspondances entre les valeurs de la propriété `display` et l'élément XHTML dont le comportement est induit par celles-ci.

Tableau 13-1. Correspondances entre les valeurs de la propriété display et les éléments XHTML

Valeur	Élement	Valeur	Élement
table	<table>	table-row-group	<tbody>
table-header-group	<thead>	table-footer-group	<tfoot>
table-row	<tr>	table-column-group	<colgroup>
table-column	<col />	table-cell	<td> ou <th>
table-caption	<caption>	inline-table	<table> dont la propriété display vaut inline

L'exemple 13-3 offre une illustration de la propriété `display`. La page comprend un paragraphe `<p>` qui inclut quatre éléments `<span>` (repère ❹). Ces éléments, qui sont normalement de type en ligne, sont affichés sous forme de liste en donnant la valeur `list-item` à la propriété `display` (repère ❶). La page contient également une liste non ordonnée (repère ❺). En donnant la valeur `inline` à `display` pour les éléments `<li>` (repère ❷), ils constituent un menu sur une seule ligne. La page contient enfin une division (repère ❻) incluant une image (repère ❾), deux titres `<h1>` (repères ❻ et ⓫) et un paragraphe (repère ⓬). Pour ces derniers, la propriété `display` prend la valeur `inline`. Le contenu des titres est donc affiché comme du texte en ligne dans le paragraphe. Quant à l'image incluse dans `<div>`, le gestionnaire d'événements `onclick` permet de la faire disparaître quand le visiteur clique sur la division en donnant à l'aide de code JavaScript la valeur `none` à la propriété `display` (repère ❼) ; en gérant l'événement `ondblclick`, un double-clic la fait réapparaître en lui octroyant la valeur `inline` (repère ❽).

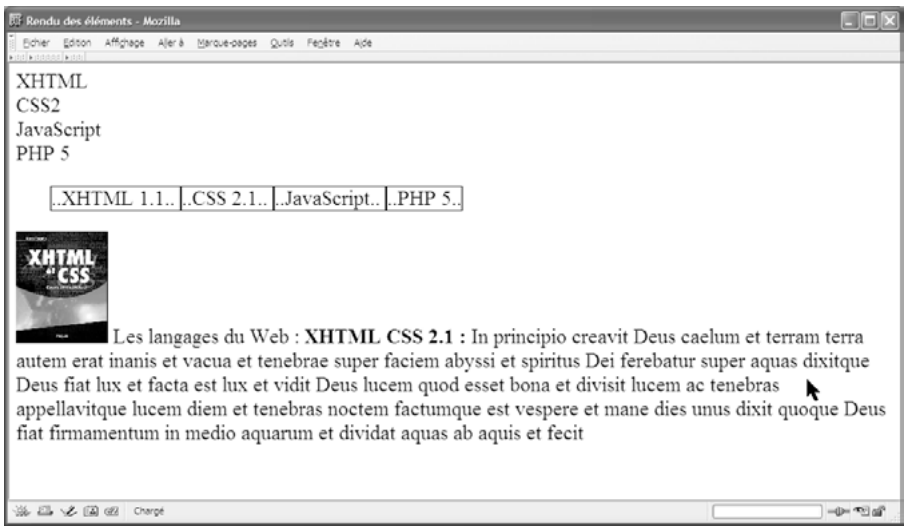
Exemple 13-3. Modification du rendu des éléments

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Rendu des éléments</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   body,h1,p{font-size: 24px;}
   span{display: list-item; }❶
   li{display: inline;border: solid 1px black;}❷
   h1,p{display: inline;}❸
  </style>
 </head>
 <body>
  ❹<p><span> XHTML </span> <span> CSS2 </span> <span> JavaScript </span>
  ➥ <span> PHP 5 </span>  </p>
  ❺<ul>
   <li><a  href="lien1.html" ></a>..XHTML 1.1.. </li>
   <li><a  href="lien2.html" ></a>..CSS 2.1.. </li>
   <li><a  href="lien3.html" ></a>..JavaScript.. </li>
   <li><a  href="lien4.html" ></a>..PHP 5.. </li>
  </ul>
  ❻<div id="division"
  ➥ ❼onclick="document.getElementById('couv').style.display='none'"
  ➥ ❽ondblclick="document.getElementById('couv').style.display='inline'" >
   ❾<img id="couv"  src="couvxhtml_gd.jpg" alt="XHTML CSS" height="130"
   ➥ width="107" /> Les langages du Web :
```

```
⓾ <h1>XHTML </h1>
⓫ <h1> CSS 2.1 : </h1>
⓬ <p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
➥ Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona et divisit
➥ lucem ac tenebras appellavitque lucem diem et tenebras noctem factumque est
➥ vespere et mane dies unus dixit quoque Deus fiat firmamentum in medio aquarum
➥ et dividat aquas ab aquis et fecit </p>
  </div>
 </body>
</html>
```

La figure 13-5 montre le résultat obtenu avant la disparition de l'image.

Figure 13-5
Les modifications du rendu normal des éléments



Le positionnement des éléments

Le principe du positionnement permet de définir l'emplacement des boîtes générées pour chaque élément XHTML présent dans le code de la page. Il existe plusieurs schémas de positionnement en CSS 2.1.

Le positionnement selon le flux normal : il inclut le positionnement par défaut opéré par les navigateurs sans définition de styles particuliers. Avec ce positionnement, les éléments `<p>`, `<div>` ou `<h1>`, par exemple, sont associés à une boîte de bloc, et les éléments `<img />` ou `<span>` à une boîte en ligne. Il inclut également le positionnement relatif des éléments de type bloc ou en ligne.

Le positionnement flottant : dans ce cas, la boîte de l'élément est générée comme dans le flux normal. Elle conserve pour exemple les dimensions qui lui ont été attribuées, soit par

ses attributs, soit par les propriétés `width` et `height`. En revanche, cette boîte est déplacée afin d'être positionnée le plus haut et le plus à gauche possible dans le contenu.

Le positionnement absolu : avec ce type de positionnement, le bloc généré par l'élément devient complètement indépendant du flux normal. Sa position est en particulier sans aucun rapport avec son ordre d'apparition dans le code XHTML. Des propriétés particulières doivent être définies pour placer la boîte de l'élément par rapport à son contenu, qu'il s'agisse de `<body>` ou d'un autre élément.

Le positionnement fixe : dans ce cas, un élément occupe une position fixe dans la fenêtre du navigateur et ne défile pas avec le reste de la page.

Le flottement

Le concept du modèle de boîtes vu au chapitre 11 suppose que les blocs apparaissent dans la page les uns à la suite des autres, de haut en bas, selon leur ordre d'apparition dans le code XHTML. De même, un élément remplacé, comme une image incluse dans un paragraphe au milieu d'un texte par exemple, est affiché comme un sous-bloc en provoquant un retour à la ligne avant et après l'image. Si sa largeur est inférieure à celle du paragraphe, cela entraîne l'apparition d'une zone blanche à sa droite, car elle est alignée à gauche par défaut. Cela crée un effet évidemment disgracieux dans la page. Il en est de même pour un élément de bloc, par exemple un tableau dont la largeur est faible par rapport à celle de son conteneur.

Pour améliorer cet état de fait, nous pouvons rendre n'importe quel élément flottant. Quand un élément est déclaré flottant, le contenu des autres éléments voisins se dispose autour de lui comme l'eau s'écoule autour d'un rocher placé au milieu d'une rivière. L'espace laissé libre autour de cet élément est comblé, créant ainsi une meilleure occupation de la surface de la page.

Quand un élément est déclaré flottant, la boîte qui correspond à l'élément est déplacée vers la gauche ou la droite de son conteneur selon la valeur choisie pour le flottement. L'alignement vertical est tel que la boîte est déplacée sur le haut de la ligne qui contient l'élément, ou sur le bas du bloc qui le précède. On procède à la définition du flottement d'un élément au moyen de la propriété `float` qui peut s'appliquer à tous les éléments XHTML et dont la syntaxe est la suivante :

```
float:left|right|none|inherit
```

Les valeurs qu'elle peut prendre ont la signification suivante :

- `left` : la boîte de l'élément est déplacée en haut et à gauche de son conteneur ;
- `right` : la boîte de l'élément est déplacée en haut et à droite ;
- `none` : la boîte ne flotte pas (c'est la valeur par défaut) ;
- `inherit` : la boîte hérite du comportement de son élément parent (ce qui n'est pas le cas par défaut).

Pour qu'un élément soit flottant, il est nécessaire qu'il soit dimensionné explicitement avec la propriété `width`, ou implicitement par ses dimensions intrinsèques (celles d'une image par exemple), ou par ses attributs `width` et `height` dans le cas d'un élément `<img />` ou `<object>`. Dans le cas contraire, le navigateur risque de l'afficher avec une largeur minimale, si ce n'est nulle.

Si un élément flottant possède des marges, ces dernières ne fusionnent pas avec celles des éléments voisins, mais elles s'ajoutent à celles-ci. Quand plusieurs éléments sont flottants du même côté, celui qui apparaît en seconde position dans le code peut être amenée à butter sur le bord droit ou gauche du premier selon que la propriété `float` a respectivement la valeur `left` ou `right`.

Dans l'exemple 13-4, la page contient un élément `<div>` (repère ❹) qui inclut du texte brut puis une image (repère ❺), à nouveau du texte puis un paragraphe (repère ❻). La division a une couleur de fond, des marges de 15 pixels et un alignement justifié (repère ❶). L'image est positionnée flottante à gauche et a des marges de 20 pixels. Le paragraphe est flottant à droite, dimensionné avec une largeur de 200 pixels, et est muni d'une bordure qui permet de bien le distinguer du reste du texte. La figure 13-6 montre le résultat obtenu. Nous y constatons que les éléments flottants n'apparaissent que relativement à l'ordre dans lequel ils sont écrits dans le code XHTML. Par exemple, l'image n'apparaît qu'une fois que la ligne dans laquelle l'élément `<img />` est situé a été complétée par du texte qui se situe après cet élément.

Exemple 13-4. Les éléments flottants

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les éléments flottants</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   div{background-color: #EEE;margin: 15px;text-align: justify;}❶
   img{float:left;margin:20px;}❷
   p.droit{float: right;border:2px solid red;width: 200px;margin:10px;
   ➥ padding: 10px;}❸
  </style>
 </head>
 <body>
  ❹<div>XHTML ET CSS : In principio creavit Deus caelum et terram terra autem erat
   ➥ inanis et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur
   ➥ super aquas dixitque Deus fiat lux et facta est lux et vidit Deus
```

```
    ❺ <img src="couvxhtml_gd.jpg" alt="XHTML CSS" />
    lucem quod esset bona et divisit lucem ac tenebras appellavitque lucem diem et
    ➥ tenebras noctem factumque est vespere et mane dies unus dixit quoque Deus fiat
    ➥ firmamentum . . .
    ❻ <p class="droit">Le W3C <br />aquas quae erant sub firmamento ab his quae erant
    ➥ super firmamentum et factum est ita vocavitque Deus firmamentum caelum et
    ➥ factum est vespere et mane dies secundus dixit vero Deus congregentur aquae
    ➥ quae sub caelo sunt in locum unum et appareat  </p>
    in firmamento caeli et inluminent terram et factum est ita fecitque Deus duo
    ➥ magna luminaria luminare maius ut praeesset diei et luminare minus . . .
  </div>
 </body>
</html>
```

Figure 13-6

Le flottement des éléments

L'exemple suivant permet d'appréhender les subtilités du positionnement flottant dans des cas particuliers. L'élément `<div>` (repère ❸) inclus dans la page contient maintenant trois images incorporées au milieu d'un texte (repères ❹, ❺ et ❻). Vient ensuite un paragraphe ne contenant que du texte (repère ❼). Les deux premières images sont flottantes à gauche (repère ❶), et la troisième à droite en utilisant la classe `img.droit` (repère ❷).

Seule la deuxième image est dimensionnée explicitement avec ses attributs `height` et `width`, les autres ayant les dimensions originales de l'image.

Exemple 13-5. Cas particuliers de flottement

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les éléments flottants</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   div{background-color: #EEE;}
   img{float:left;margin:20px;}❶
   img.droit{float: right;}❷
   p{background-color: #DD2;}
  </style>
 </head>
 <body>
  ❸<div>XHTML ET CSS : In principio creavit Deus caelum et terram terra autem erat
   ➥ inanis et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur
   ➥ super aquas dixitque Deus fiat lux et facta est lux et vidit Deus
   ❹<img src="couvxhtml_gd.jpg" alt="XHTML CSS" />
   lucem quod esset bona et divisit lucem ac tenebras appellavitque lucem diem et
   ➥ tenebras noctem factumque est vespere et mane dies unus dixit quoque Deus
   ➥ fiat . . .
   ❺<img src="couvxhtml_gd.jpg" alt="XHTML CSS" height="130" width="107" />
   herbam virentem et adferentem semen iuxta genus suum lignumque faciens fructum et
   ➥ habens unumquodque sementem secundum speciem suam et vidit Deus . . .
   ❻<img class="droit" src="couvxhtml_gd.jpg" alt="XHTML CSS" />ut praeesset nocti
   ➥ et stellas et posuit eas in firmamento caeli ut lucerent super terram et
   ➥ praeessent diei ac nocti et dividerent lucem ac tenebras et vidit Deus . . .
  </div>
  ❼<p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
   ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
   ➥ Deus fiat lux et facta est lux et vidit Deus . . . </p>
 </body>
</html>
```

On peut le vérifier à la figure 13-7, la première image est positionnée comme dans l'exemple précédent, et la deuxième flottante à gauche vient se positionner le plus à gauche possible, selon les principes énoncés plus haut. Mais comme l'espace est occupé par la première, elle vient se placer contre le bord droit de cette dernière, et non plus sur le

bord gauche de son conteneur. Nous pouvons remarquer à cette occasion que les marges de chacune des images sont conservées et ne fusionnent pas. Le cas de la troisième image flottante à droite est le plus particulier. En effet, l'élément `<div>` ne pouvant la contenir en entier, elle déborde et empiète sur le paragraphe qui suit et se comporte comme si elle était flottante dans ce paragraphe, le texte de ce dernier l'entourant.

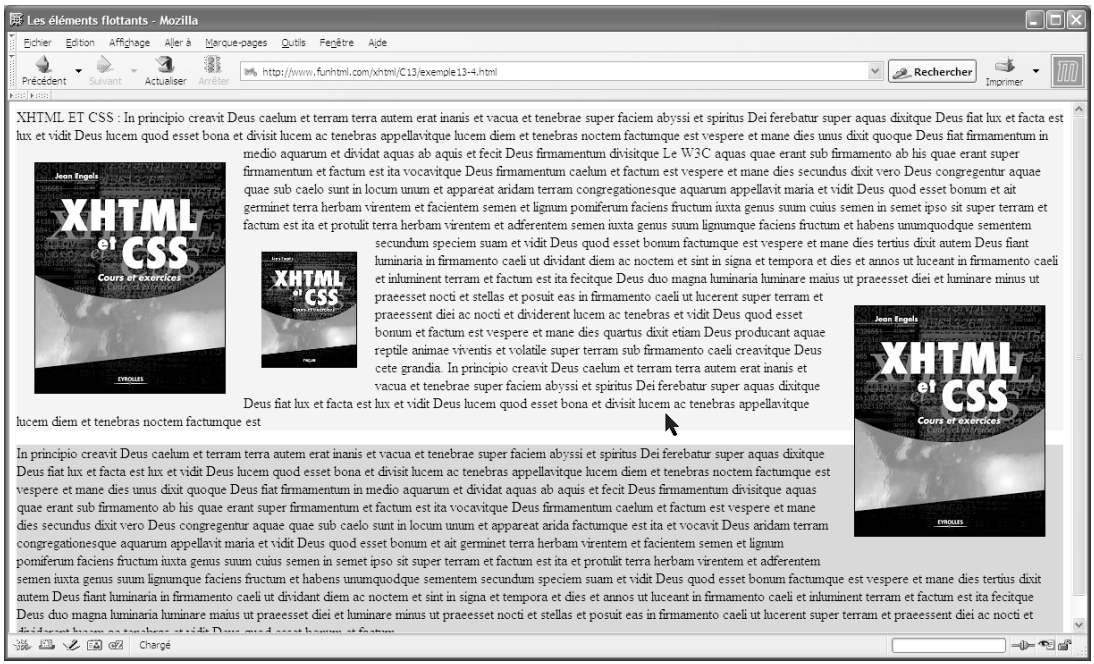


Figure 13-7

Cas particuliers de flottement

Empêcher le flottement

Nous pouvons empêcher le type de comportement de la troisième image qui flotte sur le paragraphe `<p>` en appliquant pour ce dernier la propriété `clear`. Ce procédé n'est possible que pour les éléments de bloc, mais il permet d'éviter le flottement d'un élément sur la gauche ou la droite de celui pour lequel la propriété est définie. Ainsi, l'image a la place nécessaire à son affichage et le texte du paragraphe est repoussé vers le bas, laissant apparaître un espace vide. Cela peut constituer un choix de présentation dans certains cas, mais la solution précédente est en l'occurrence plus compacte. Voici la syntaxe de la propriété `clear` :

```
clear: none | left | right | both | inherit
```

Les valeurs qu'elle peut prendre ont pour signification :

- `none` : le flottement est autorisé ;
- `left` : le flottement d'un élément sur la gauche est interdit ;
- `right` : le flottement d'un élément sur la droite est interdit ;
- `both` : le flottement d'un élément est interdit à gauche et à droite.

En conservant le code XHTML de l'exemple 13-5 et en modifiant seulement les styles de la manière suivante (repère ❶) :

```
<style type="text/css" >
 div{background-color: #EEE;}
 img{float:left;margin:20px;}
 img.droit{float: right;}
 p{background-color: #DD2;clear:right;}❶
</style>
```

dans laquelle la propriété `clear` est définie avec la valeur `right` pour l'élément `<p>`, nous obtenons le résultat présenté à la figure 13-8 qui montre que la troisième image ne flotte plus sur le paragraphe.

Figure 13-8

Suppression du flottement sur le côté d'un élément

Le positionnement relatif

Dans le positionnement relatif, les navigateurs analysent un document et déterminent un emplacement pour chacun des éléments comme dans le flux normal, puis déplacent ceux qui ont un positionnement relatif par rapport à la position qu'ils auraient dû occuper, mais en ne déplaçant pas les autres éléments. En conséquence, l'emplacement initial reste vide et il en résulte des chevauchements.

En positionnement relatif, l'ordre d'écriture des éléments est important car il conditionne l'emplacement de chaque élément (avant le déplacement relatif) et l'ordre de superposition en cas de chevauchement des boîtes ; la dernière écrite dans le code se superpose à celle de l'élément écrit avant.

Le positionnement relatif est spécifié au moyen de la propriété `position`, munie de la valeur `relative`. Prise isolément, cette propriété n'a aucun effet. Il faut ensuite préciser le décalage voulu au moyen des propriétés `left`, `top`, `right`, `bottom`, dont les significations sont les suivantes :

- `left` : décale l'élément vers la droite (si sa valeur est positive) ou vers la gauche (si sa valeur est négative) ;
- `top` : décale l'élément vers le bas (si sa valeur est positive) ou vers le haut (si sa valeur est négative) ;
- `right` : décale l'élément vers la gauche (si sa valeur est positive) ou vers la droite (si sa valeur est négative) ;
- `bottom` : décale l'élément vers le haut (si sa valeur est positive) ou vers le bas (si sa valeur est négative).

Les quatre propriétés `left`, `top`, `right`, `bottom` ont la même syntaxe. Par exemple :

```
left: <longueur> | NN% | auto | inherit
```

Le paramètre `<longueur>` est donné comme à l'habitude par un nombre et une unité ; les pourcentages se réfèrent aux dimensions du bloc conteneur. Avec le mot-clé `auto`, la valeur de la propriété est calculée en fonction de la valeur de celle qui lui est complémentaire (les couples `left/right` et `top/bottom` sont complémentaires) de la manière suivante :

- Si les deux propriétés complémentaires ont la valeur `auto`, leur valeur calculée est `0`.
- Si l'une vaut `auto` et que l'autre a une valeur numérique explicite, la première prend la valeur opposée à celle de la seconde.
- Si les deux propriétés complémentaires sont définies avec des valeurs numériques (donc différentes de `auto`) et que ces deux valeurs ne sont pas opposées, la valeur qui l'emporte dépend du sens de lecture du texte définie par l'attribut `dir`. S'il vaut `ltr`, c'est la propriété `left` qui l'emporte, et `right` prend la valeur opposée ; sinon, quand il vaut `rtl`, c'est `right` qui l'emporte.

L'exemple 13-6 illustre le positionnement relatif. L'élément `<div>` (repère ❺) inclut un paragraphe qui contient du texte brut, puis trois images (repères ❼, ❽ et ❾) dans un ordre donné. Si aucun de ces éléments n'est positionné, nous obtenons le résultat habituel présenté à la figure 13-9. Si nous positionnons chacune de ces images en leur appliquant