respectivement les classes `img.un`, `img.deux` et `img.trois`, nous obtenons les déplacements relatifs suivants :

- image 1 : déplacement de 30 pixels vers le bas (`top` vaut `30 px`) et de 40 pixels vers la droite (`left` vaut `40 px`) (repère ❶) ;
- image 2 : déplacement de 50 pixels vers le haut (`top` a une valeur négative de `- 50 px`) et de 40 pixels vers la gauche (`left` a une valeur négative de `-40 px`) (repère ❷) ;
- image 3 : déplacement de 80 pixels vers le haut (`top` a une valeur négative de `- 80 px`) et de 40 pixels vers la droite (`left` vaut `40 px`) (repère ❸).

Le paragraphe est quant à lui déplacé de 20 pixels vers le bas (`bottom` a une valeur négative de `- 20 px` équivalente au style `top:20 px`) et de 6 % de la largeur de son conteneur (l'élément `<div>`) vers la droite. Notons ici que les valeurs négatives de pourcentage ne fonctionnent pas.

Exemple 13-6. Le positionnement relatif

```
<?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>Positionnement relatif</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{margin:20px;}
   img.un{position:relative;top:30px;left:40px;}❶
   img.deux{position:relative; top:-50px; left:-40px;}❷
   img.trois{position:relative; top:-80px; left:40px;}❸
   p{background-color: #DD2;position:relative; bottom:-20px; left: 6% ;}❹
  </style>
 </head>
 <body>
  ❺<div>
   ❻<p><big>XHTML </big>: 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 . . . </p>
   ❼<img src="couvxhtml_gd.jpg" alt="XHTML CSS" class="un"/>
   ❽<img src="couvxhtml_gd.jpg" alt="XHTML CSS" height="130" width="107"
   ➥ class="deux"/>
   ❾<img class="trois" src="couvxhtml_gd.jpg" alt="XHTML CSS" />
  </div>
 </body>
</html>
```

La figure 13-10 donne le résultat obtenu après le positionnement relatif. Nous constatons que les éléments déplacés relativement à leur position dans le flux normal peuvent se superposer aux autres éléments voisins, la deuxième image étant placée au dessus de la première et la troisième se superposant au texte du paragraphe.

Figure 13-9
La page sans positionnement

Figure 13-10
La page avec positionnement

Le positionnement absolu

Dans le positionnement absolu, qui peut s'appliquer à tous les éléments, la boîte créée pour l'élément concerné n'apparaît plus dans le flux normal du document. Autrement dit, si un bloc `<div>` est positionné de manière absolue, il peut être écrit n'importe où dans le document XHTML sans que cela ne modifie la position qui va lui être assignée avec un style CSS. Le positionnement d'un élément est effectué par rapport au bloc de son conteneur (il s'agit souvent de l'élément `<body>` mais pas nécessairement). Chaque bloc d'un élément positionné de manière absolue devient à son tour le conteneur de ses éléments enfants. Si ces éléments sont eux-mêmes en position absolue, ils le sont par rapport à leur bloc parent direct et non par rapport au bloc qui contient leur parent. Nous pouvons donc créer des blocs `<div>` et les positionner, puis écrire normalement leur contenu.

L'indépendance de la position par rapport au flux normal fait que deux éléments en position absolue peuvent occuper la même zone dans leur bloc parent commun. Dans ce cas, aucun d'eux ne repousse l'autre comme il en va pour les éléments flottants. Le bloc d'un élément peut alors recouvrir l'autre en fonction de leur ordre d'empilement (voir la propriété `z-index`). Cette propriété peut alors être utilisée pour créer des effets dynamiques d'apparition et de disparition gérés par des scripts JavaScript.

Comme le positionnement relatif, le positionnement absolu est défini en utilisant encore la propriété `position`, mais en lui donnant cette fois la valeur `absolute` (ou `fixed` sur laquelle nous reviendrons par la suite). Il faut ensuite définir la position de l'élément par rapport à son conteneur à l'aide des propriétés `left`, `top`, `right`, `bottom`, qui définissent la position des bords de l'élément respectivement par rapport aux bords gauche, haut, droit et bas du conteneur. Nous ne définissons généralement que deux de ces propriétés, le plus souvent `left` et `top` (les propriétés symétriques `right` et `bottom` prenant une valeur opposée), et la boîte de l'élément doit être dimensionnée avec les propriétés `width` et `height`. L'utilisation conjointe des propriétés `position` et `float` est impossible, et si c'est le cas la propriété `float` prend automatiquement la valeur `none`.

Dans les exemples suivants, nous allons étudier divers cas de mise en page correspondant à des types de présentation couramment rencontrés sur le Web.

Le premier cas présenté dans l'exemple 13-7 consiste à diviser la page en deux zones horizontales. La zone supérieure est un bandeau contenant le titre du site et un menu composé de liens vers les différentes pages. Pour permettre une navigation aisée, toutes les pages contiennent ce même bandeau dont il suffit de copier le code et les styles qui lui sont associés dans chacune des pages. Il est contenu dans un élément `<div>` muni d'un attribut `id` (repère ❺) et il inclut un titre `<h1>` (repère ❻) et le menu créé par une liste non ordonnée `<ul>` (repère ❼) dont chaque item contient un lien vers les différentes pages.

Ce premier élément `<div>` est positionné de manière absolue en haut et à gauche de la page en définissant les propriétés `top` et `left` avec la valeur `0`. Il est dimensionné à 100 % en largeur et à 110 pixels en hauteur (repère ❶). La propriété `display` appliquée à l'élément `<li>` permet d'obtenir le menu des liens en ligne sous forme horizontale (repère ❸). Les items sont, de plus, munis de bordures pour en améliorer l'aspect.

La zone de contenu est également incluse dans un second élément `<div>` qui contient un titre `<h1>`, un paragraphe `<p>` et une image (repère ❽), ces éléments n'étant pas positionnés. Comme la première, cette zone est dimensionnée à 100 % de la largeur de la page et positionnée de manière absolue en étant placée à 110 pixels du bord supérieur de la page (correspondant à la hauteur du bandeau) et collée à son bord gauche en mettant la propriété `left` à la valeur `0` (repère ❷).

Exemple 13-7. Positionnement en deux blocs horizontaux

```
<head>
  <title> La référence des éléments  XHTML 1.1 </title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <style type="text/css">
   body{font-size: 18px;}
   h1{font-size: 2em;margin-top: 5px;}
   div#menu {position: absolute ; width:100%; height: 110px; left:0px; top:0;
   ➥ background-color:rgb(255,102,5);color:white; }❶
   div#corps { position: absolute ;width:100%; left:0; top:110px;background-color:
   ➥ #EEE;color:black;}❷
   li {display:inline;  border: solid 1px white;padding: 0 10px 0 10px;}❸
   img {float:right;}❹
   a{text-decoration: none;color: white;}
  </style>
 </head>
 <body>
  <div id="menu">❺
   <h1>   Référence XHTML 1.1 </h1>❻
   <ul>❼
     <li> <a  href="page1.html" tabindex="1" accesskey="A"
     ➥ title="Structure">Structure</a> </li>
     <li> <a  href="page2.html" tabindex="2" accesskey="B" title="Texte">Texte</a>
     ➥ </li>
  <!-- Suite de la liste -->
   </ul>
  </div>
  <div id="corps">❽
   <h1>La déclaration &lt;!DOCTYPE&gt;</h1>
   <img src="couvxhtml_gd.jpg" alt="XHTML CSS" />
   <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 divis
   ➥ . . . </p>
  </div>
 </body>
</html>
```

La figure 13-11 montre le résultat obtenu.

Figure 13-11

Le positionnement en deux blocs horizontaux

Nous pouvons également envisager une variante de ce premier cas qui va nous montrer que nous pouvons positionner des éléments de manière absolue à l'intérieur d'un élément lui-même positionné de cette façon. Nous réalisons cette opération à l'intérieur du second élément `<div>` de l'exemple 13-7 en conservant intégralement son code XHTML.

Pour positionner les éléments enfants du second élément `<div>`, nous pouvons par exemple modifier simplement les styles CSS de la manière suivante :

```
<style type="text/css">
   body{font-size: 18px;}
   h1{font-size: 2em;margin-top: 5px;}
   div#menu {position: absolute ; width:100%; height: 110px; left:0px; top:0;
   ➥ background-color:rgb(255,102,5);color:white; }
   div#corps { position: absolute ;width:100%; left:0; top:110px;color:black;}❶
   li {display:inline;  border: solid 1px white;padding: 0 10px 0 10px;}
   div#corps h1{position: absolute; width:600px; height:40px;top: 20px;
   ➥ left: 300px;background-color: #AAA;margin: 0;}❷
   img {position: absolute; top: 70px;right:20px;}❸
   p {position: absolute;width: auto; top: 70px; right:260px; left:30px;margin: 0;
   ➥ text-align: justify;background-color: #BBB;}❹
```

```
    a{text-decoration: none;color: white;}
</style>
```

Le second élément `<div>` garde son positionnement par rapport à la page (repère ❶), mais les éléments qu'il contient sont à leur tour positionnés. Ses éléments enfants `<h1>`, `<img />` et `<p>` étant eux-mêmes positionnés de manière absolue, ils ne le sont pas par rapport à la page mais par rapport à leur parent. L'élément `<h1>` a une largeur de 600 pixels et une hauteur de 40 pixels ; il est placé à 300 pixels du bord gauche de son parent et à 20 pixels de son bord supérieur (soit 110 + 20 = 130 pixels du bord supérieur de la page) (repère ❷).

L'image est placée à 70 pixels du bord haut de son conteneur et à 20 pixels de son bord droit (repère ❷). Elle n'est pas dimensionnée explicitement et conserve ses dimensions intrinsèques qui sont celles du fichier image. Le paragraphe `<p>` n'est pas non plus dimensionné explicitement et sa largeur est fixée avec la valeur `auto`. Il sera ainsi redimensionné automatiquement sans empiéter sur l'image si la fenêtre est elle-même redimensionnée. En pratique, c'est son positionnement à 260 pixels du bord droit et à 30 pixels du bord gauche qui conditionne sa largeur.

Notons que si nous déplacions le conteneur de tous ces éléments en modifiant les propriétés `left` et `top` du second élément `<div>`, la position de ces trois éléments à l'intérieur de leur conteneur resterait inchangée. La figure 13-12 montre le résultat obtenu avec ces positionnements.

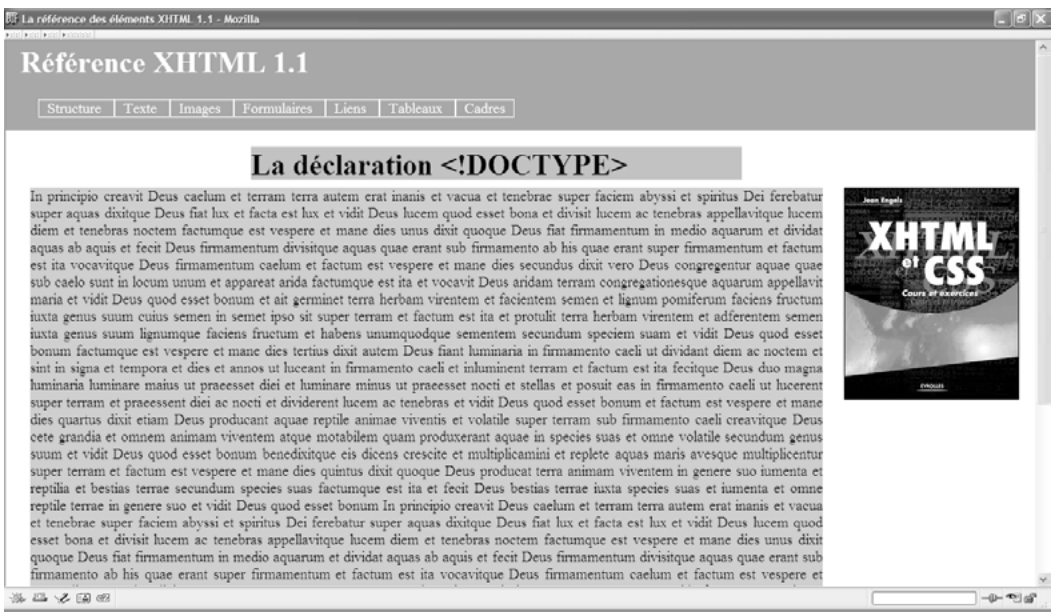

Figure 13-12

Positionnement dans un élément lui-même positionné

L'exemple suivant permet de créer un autre type de présentation très classique puisqu'il permet de diviser la page en trois zones. La première est un bandeau qui peut contenir le titre du site ou une publicité, l'espace restant étant divisé en deux colonnes de la même façon que nous l'avons réalisé avec des cadres au chapitre 8. Le contenu est identique à celui des exemples précédents mais il est structuré différemment. À chaque zone correspond un élément `<div>` qui va être positionné de manière absolue dans la page. Nous n'avons pas défini ici d'attributs `id` pour chacun d'eux, mais nous appliquons des classes différentes à chaque division.

Le premier élément `<div>` (repère ❹) qui contient un titre `<h1>` (repère ❺) est positionné comme précédemment en haut de la page après avoir été dimensionné à 100 % en largeur et 100 pixels en hauteur. On évite ici de définir une hauteur en pourcentage pour qu'elle ne dépende pas du contenu de la page. Ces styles sont définis dans la classe `div.tete` (repère ❶).

Le reste de la page est partagé en deux colonnes. La colonne de gauche (repère ❻) contient le même menu (repère ❽) que dans l'exemple 13-7, mais dans un affichage vertical, soit son style par défaut. Cet élément `<div>` est d'abord dimensionné avec une largeur de 20 % de celle de la page et une hauteur de 100 %. Il est positionné de manière absolue au moyen de la classe `div.menu` à 100 pixels du bord haut de la page et à 0 pixel de son bord gauche (repère ❷).

Le contenu éditorial de la page constitue la colonne de droite créée avec le troisième élément `<div>` (repère ❾). Il contient un titre (repère ❿), une image (repère ⓫) et un paragraphe (repère ⓬) comme dans les exemples précédents. Son traitement est assuré par la classe `div.contenu` dans laquelle il est dimensionné en largeur à 78 % et avec la valeur `auto` en hauteur. Son positionnement est absolu et il est placé à 20 % de la largeur de la page du bord gauche et à 100 pixels du bord supérieur (repère ❸). Avec leurs dimensions et positionnement respectifs, les trois zones sont bien collées les unes aux autres.

Exemple 13-8. Division de la page en trois zones

```
<?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 xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title> La référence des éléments  XHTML 1.1 </title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
   <style type="text/css">
   body{font-size: 18px;}
   div.tete {position: absolute ; left:0px;top:0px; width:100%; height:100px;
   ➥ background-color:rgb(0,0,153);margin:0; }
   div.menu {position: absolute ; width:20%; height: 100%; left:0px; top:100px;
   ➥ background-color:rgb(255,102,51);color:white; }
```

```
  div.contenu {margin: 1% ; position: absolute ;width:78%; left:20%;
  ➥ top:100px;background-color:white;color:black;}
  div h1 {font-size:50px;font-style:italic;color:rgb(255,102,51);
  ➥ margin-top:0px;margin-left:200px;}
  li {padding: 15px;}
  p{text-align: justify;}
  img{float: left;margin: 0 10px 0 0;}
 </style>
</head>
<body>
 <div class="tete">
  <h1>   Référence XHTML 1.1 </h1>
 </div>
 <div class="menu">
  <ul>
 <li> <a  href="page1.html" tabindex="1" accesskey="A"
 ➥ title="Structure">Structure</a> </li>
 <li> <a  href="page2.html" tabindex="2" accesskey="B" title="Texte">Texte</a>
 ➥ </li>
 <li> <a  href="page3.html" tabindex="3" accesskey="C" title="Images">Images</a>
 ➥ </li>
 <li> <a  href="page4.html" tabindex="4" accesskey="D"
 ➥ title="Formulaires">Formulaires</a> </li>
 <li> <a  href="page5.html" tabindex="5" accesskey="E" title="Liens">Liens</a>
 ➥ </li>
 <li> <a  href="page6.html" tabindex="6" accesskey="G" title="Tableaux">Tableaux
 ➥ </a> </li>
 <li> <a  href="page7.html" tabindex="7" accesskey="H" title="Cadres">Cadres</a>
 ➥ </li>
  </ul>
 </div>
 <div class="contenu">
  <h1>La déclaration &lt;!DOCTYPE&gt;</h1>
  <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 . . .</p>
 </div>
</body>
</html>
```

La figure 13-13 montre le résultat obtenu avec ces définitions de styles. Nous pouvons remarquer une fois de plus l'intérêt des styles CSS car la présentation n'est pas la même que celle de la figure 13-12 alors que le contenu est le même.

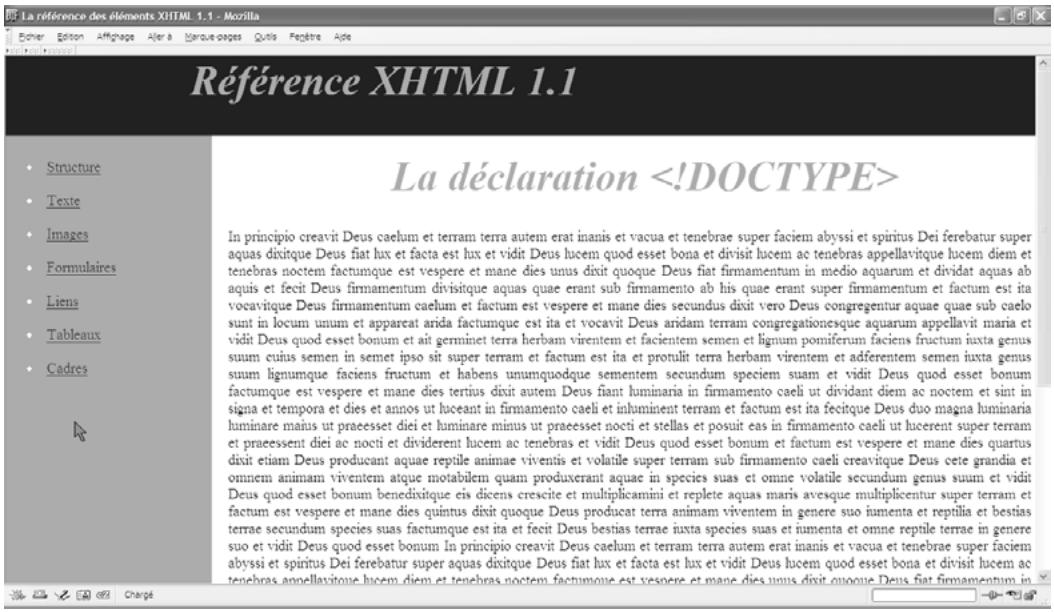

Figure 13-13

Positionnement en trois zones

Notre dernier exemple constitue une structure de page complexe comportant un bandeau, trois colonnes, la première contenant le menu, la deuxième un contenu textuel et un pied de page qui inclut l'adresse de contact. La troisième colonne contient une liste de liens utiles. La figure 13-14 montre la présentation de la page que nous allons obtenir.

Chacune de ces zones est créée par un élément `<div>` dimensionné puis positionné de manière absolue en leur appliquant les classes `div.haut`, `div.gauche`, `div.droit`, `div.contenu` et `div.bas`. La seule division qui représente un élément nouveau par rapport aux cas antérieurs est celle qui permet de placer la division d'adresse en bas de la zone centrale de contenu. Cette division est incluse dans la division précédente et non plus directement dans `<body>`.

Dans le code de l'exemple 13-9, les éléments `<div>` sont volontairement placés dans le désordre afin de démontrer que leur position dans le code XHTML n'a aucune importance. Les définitions et les rôles des classes sont les suivants :

- `div.haut` : (repères ❶ et ⓬) largeur 100 %, hauteur 70 pixels, positionné en haut et à gauche (`top:0` et `left:0`).

- `div.gauche` : (repères ❷ et ⓫) largeur 15 %, hauteur 100 % (ou auto), positionné à 70 pixels du haut et à 0 du bord gauche.
- `div.droit` : (repères ❸ et ❼) largeur 15 %, hauteur 100 % (ou `auto`), positionné à 70 pixels du haut et à 0 du bord droit.
- `div.contenu` : (repères ❹ et ❽) largeur 70 %, hauteur `auto` car la hauteur du paragraphe peut varier si on réduit la fenêtre du navigateur.
- `div.bas` : (repères ❺ et ⓬) elle s'applique à la division incluse dans le contenu. Sa largeur est de 100 % de celle de son parent (soit 70 % de celle de la page) et sa hauteur de 60 pixels. Elle est positionnée à gauche et en bas de son parent.

Afin que son contenu ne cache pas la fin du texte, on notera qu'il faut que le paragraphe (repère ❾) ait une marge basse d'au moins 60 pixels (repère ❻).

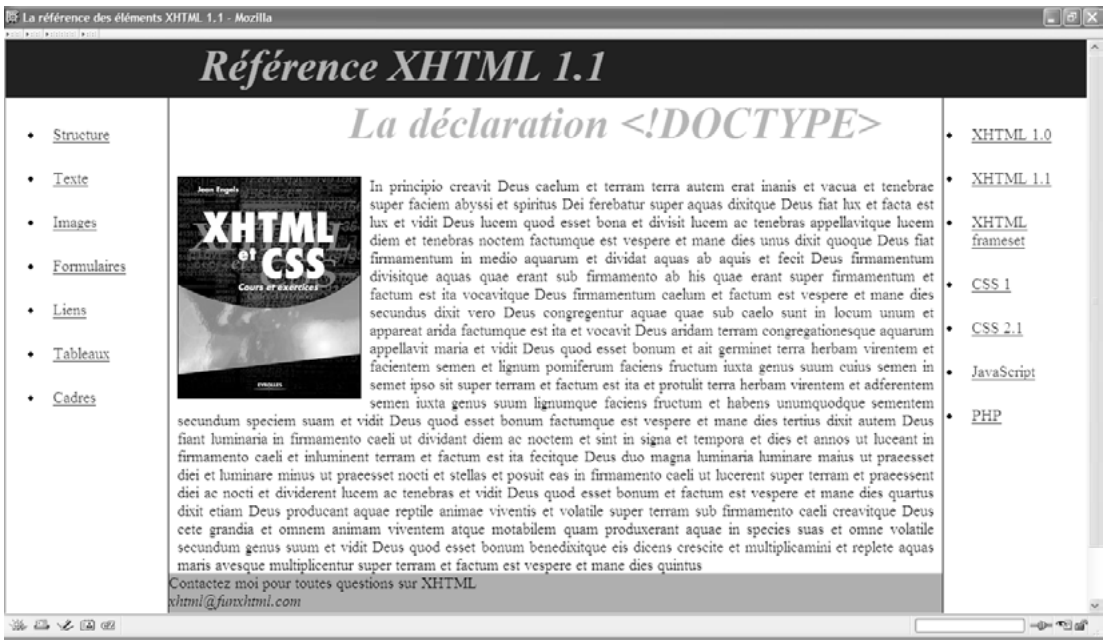

Figure 13-14

Organisation d'une page complexe en cinq zones

Exemple 13-9. Création d'une structure complexe en positionnement absolu

```
<?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 xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
```

```
<head>
 <title> La référence des éléments  XHTML 1.1 </title>
 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
 <style type="text/css">
  body{font-size: 18px;background-color:white;color:black;}
  div.haut {position: absolute ; left:0px;top:0px; width:100%; height:70px;
  ➥ background-color:rgb(0,0,153);margin:0; }❶
  div.gauche {position: absolute ; width:15%; height: auto; left:0px; top:70px;}❷
  div.droit{position: absolute ; width:15%; height: 100%; right:0; top:70px; }❸
  div.contenu {position: absolute ;width:70%; height:auto;left:15%;
  ➥ top:70px;padding: 0 10px 0 10px;border-left: 1px solid black;
  ➥ border-right: 1px solid black; }❹
  div.bas{position: absolute ; left:0px; bottom:0px; width:100%;
  ➥  height:60px;background-color:rgb(0,220,153);}❺
  div h1 {font-size:50px;font-style:italic;color:rgb(255,102,151);
  ➥ margin-top:0px;margin-left:200px;}
  li {padding: 15px;}
  p{text-align: justify;margin-bottom:60px;}❻
  img{float: left;margin: 0 10px 0 0;}
 </style>
</head>
<body>
 ❼<div class="droit">
   <ul>
   <li> <a  href="xhtml10.html" tabindex="1" accesskey="A" title="Structure">XHTML
   ➥ 1.0</a> </li>
   <li> <a  href="page2.html" tabindex="2" accesskey="B" title="Texte">XHTML 1.1
   ➥ </a> </li>
   <li> <a  href="page3.html" tabindex="3" accesskey="C" title="Images">XHTML
   ➥ frameset</a> </li>
   <li> <a  href="page4.html" tabindex="4" accesskey="D" title="Formulaires">CSS
   ➥ 1</a> </li>
   <!—Suite de la liste -->
  </ul>
 </div>
 ❽<div class="contenu">
  <h1>La déclaration &lt;!DOCTYPE&gt;</h1>
  <img src="couvxhtml_gd.jpg" alt="XHTML CSS" />
  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
  ➥ Deus fiat lux et facta est lux et vidit Deus lucem quod esset . . .</p>
 ❿<div class="bas">
  Contactez-moi pour toutes questions sur XHTML <address>xhtml@funxhtml.com
  ➥ </address>
```

```
  </div>
  </div>
  ⓫<div class="gauche">
   <ul>
  <li> <a  href="page1.html" tabindex="1" accesskey="A"
  ➥ title="Structure">Structure</a> </li>
  <li> <a  href="page2.html" tabindex="2" accesskey="B" title="Texte">Texte</a>
  ➥ </li>
  <li> <a  href="page3.html" tabindex="3" accesskey="C" title="Images">Images</a>
  ➥ </li>
  <!—Suite de la liste -->
   </ul>
  </div>
  ⓬<div class="haut">
   <h1>   Référence XHTML 1.1 </h1>
  </div>
 </body>
</html>
```

Le positionnement fixe

Au chapitre 8, nous avons envisagé la manière de créer des cadres pour diviser la page en zones qui demeurent fixes dans la fenêtre du navigateur. Outre que cette méthode, considérée aujourd'hui comme obsolète, présente plusieurs inconvénients déjà abordés, elle nécessite l'emploi de la DTD XHTML 1.0 frameset. Comme nous l'avions signalé au chapitre 8, en utilisant des styles CSS appropriés, il est possible de créer une présentation quasi similaire.

Le positionnement fixe n'est qu'un cas particulier du positionnement absolu. La détermination des dimensions et de la position d'un élément se réalisent de la même façon à l'aide des propriétés `width` et `height` pour les dimensions, et `top`, `left`, `right` et `bottom` pour déterminer la position. La seule différence réside dans le fait que le bloc conteneur utilisé n'est plus forcément l'élément parent mais la fenêtre du navigateur. De cette façon, un élément positionné de manière fixe reste à sa place même en cas de défilement de la page quand la hauteur de celle-ci dépasse celle de la fenêtre du navigateur. En d'autres termes, les propriétés de positionnement sont appliquées non plus par rapport au conteneur mais par rapport aux bords de la fenêtre.

Ce type de positionnement peut être réalisé en utilisant encore la propriété `position` en lui donnant la valeur `fixed`. Elle est bien gérée par les navigateurs actuels comme Mozilla, Firefox, Netscape, ce qui n'a pas toujours été le cas, mais très incomplètement par Internet Explorer.

Si nous reprenons exactement le même code XHTML que celui de l'exemple 13-9, nous pouvons réaliser un positionnement fixe des zones haute, gauche, droite et basse qui sont

des parties de navigation qu'il peut être utile de voir en permanence. Pour ce faire, nous écrivons les styles suivants :

Exemple 13-10. Le positionnement fixe

```
<style type="text/css">
   body{font-size: 18px;background-color:white;color:black;}
   div.haut {position: fixed ❶; left:0px;top:0px; width:100%; height:10%;
   ➥ background-color:rgb(0,0,153);margin:0; }
   div.gauche {position: fixed ❷; width:15%; height: auto; left:0px; top:10%;}
   div.droit{position: fixed ❸; width:15%; height: 100%; right:0; top:10%; }
   div.contenu {position: absolute ❹;width:70%; height:auto;left:15%;
   ➥ top:10%;padding: 0 10px 0 10px;border-left: 1px solid black;
   ➥ border-right: 1px solid black; }
   div.bas{position: fixed ❺; left:0px; bottom:0px; width:100%;
   ➥ height:60px;background-color:rgb(0,220,153);}
   div h1 {font-size:50px;font-style:italic;color:rgb(255,102,151);
   ➥ margin-top:0px;margin-left:200px;}
   li {padding: 15px;}
   p{text-align: justify;margin-bottom:60px;}
   img{float: left;margin: 0 10px 0 0;}
  </style>
```

Les seules modifications effectuées consistent à remplacer le mot-clé `absolute` par `fixed` dans toutes les classes sauf celle du contenu qui doit pouvoir défiler pour être lisible. La figure 13-15 montre le résultat obtenu dans une fenêtre redimensionnée et après avoir effectué un défilement vertical. Nous y constatons que seule la zone centrale a défilé alors que les autres n'ont pas bougé et restent entièrement visibles. Nous pouvons remarquer également que la zone basse est maintenant bien positionnée par rapport à la fenêtre car elle occupe toute la largeur de l'écran contrairement au cas précédent de positionnement absolu.

Le positionnement fixe représente donc une alternative crédible à la création de cadres comme nous l'avons fait au chapitre 8. Nous parlerons plutôt de simili-cadres qui donnent l'illusion d'une page avec cadres et en présente les avantages visuels, mais sans interactivité entre les différentes zones comme c'est le cas des cadres. La procédure à suivre, relativement simple, est la suivante :

- Créer autant d'éléments `<div>` que l'on désire obtenir de zones différentes dans la page. Chaque zone est l'équivalent des éléments `<frame />` utilisés dans la méthode avec cadres. Pour chacune de ces zones, définir la propriété `position` avec la valeur `fixed`.
- Dimensionner chacune de ces divisions en utilisant les propriétés `width` (largeur) et `heigth` (hauteur).
- Positionner les éléments `<div>` en définissant les propriétés `left`, `top`, `right` et `bottom` qui nous permettent de placer les éléments `<div>` par rapport aux bords de la fenêtre.
- Définir éventuellement les propriétés de couleur de fond, de bordure, de marge et d'espacement pour améliorer la présentation du contenu de chaque élément `<div>`.

Figure 13-15
Le positionnement fixe

Pour retrouver une interactivité entre les « simili-cadres », par exemple afin que le clic sur un lien du menu affiche un contenu adapté dans la zone centrale, nous pouvons créer plusieurs pages qui ont toutes en commun les divisions positionnées de manière fixe et dont le contenu de la zone centrale est variable. L'illusion d'un site avec cadres est alors complète.

Visibilité et ordre d'empilement

Nous avons pu constater qu'en positionnant des éléments de manière relative, absolue ou fixe, il était possible que plusieurs éléments occupent partiellement ou totalement le même espace dans la fenêtre du navigateur. Dans ce cas, le dernier apparu dans l'ordre du code XHTML se superpose au précédent. Pour pouvoir intervenir sur cet état de fait et gérer volontairement ces superpositions, CSS définit un placement des éléments selon trois dimensions, les deux premières dans le plan de l'écran sur lesquelles nous pouvons intervenir avec les propriétés `left`, `top`, `right` et `bottom` comme nous l'avons déjà vu, et la troisième dimension est définie selon un « axe des z » perpendiculaire à l'écran et dirigé vers le spectateur. Nous pouvons gérer cet ordre d'empilement au moyen de la propriété `z-index` dont la syntaxe est la suivante :

```
z-index : auto | <Nombre> | inherit
```

Elle ne s'applique qu'aux éléments positionnés, et les valeurs qu'elle peut prendre sont les suivantes :

- `auto` : l'élément a la même valeur que celle de son parent direct, qu'il soit défini explicitement par un nombre ou implicitement par son ordre d'apparition dans le code XHTML (le dernier ayant la priorité).

- `<Nombre>` : un nombre entier positif ou négatif, sachant que plus le nombre est grand, plus l'élément est placé en avant, et se superpose à ceux dont la valeur est inférieure.

Nous pouvons intervenir dynamiquement sur l'ordre d'empilement au moyen de code JavaScript en modifiant la valeur de la propriété CSS `z-index` (qui en JavaScript porte le nom de `zIndex`) en réponse à une action du visiteur (survol de l'élément par la souris, clic…). Cette modification est gérée par les attributs gestionnaires d'événements correspondants (`onmouseover`, `onclick`…) ou par la pseudo-classe `:hover` par exemple.

Dans le même ordre d'idées, nous pouvons appliquer à tous les éléments la propriété `visibility` qui permet de les cacher ou de les rendre visibles. Sa syntaxe est la suivante :

```
visibility : visible | hidden | collapse | inherit
```

- `visible` : l'élément est visible normalement et c'est la valeur par défaut.

- `hidden` : l'élément est caché mais la différence de comportement avec la propriété `display`, quand elle prend la valeur `none`, c'est que la boîte de l'élément est ici maintenue dans la page mais qu'elle est simplement vide et non retirée de la page comme avec `display`.

- `collapse` : son comportement est identique à la valeur `hidden` mais elle s'applique particulièrement aux cellules des tableaux.

Cette propriété est héritée par défaut et elle s'applique donc aux éléments enfants.

L'exemple 13-11 donne une illustration de la gestion de l'ordre d'empilement et de la visibilité de plusieurs éléments. La page comporte une division (repère ❻) qui contient un paragraphe `<p>` positionné (repères ❺ et ❼), lui-même incluant du texte et une image flottante (repères ❽ et ❸). La division inclut également deux images (repères ❾ et ❿) qui y sont positionnées de manière absolue (repères ❶ et ❷). La figure 13-16 montre le résultat obtenu initialement lors de l'affichage de la page.

Compte tenu de l'ordre d'apparition des éléments enfants de la division `<div>`, l'ordre d'empilement sans utilisation de la propriété `z-index` devrait être de l'arrière vers l'avant, le paragraphe, l'image « un » puis l'image « deux » au premier plan. Cependant, comme nous attribuons à la propriété `z-index` de la première image la valeur `2` et à la suivante la valeur `1`, cet ordre est inversé.

Figure 13-16

L'état initial de la page avec des éléments empilés

Si la propriété `z-index` ne servait qu'à définir un ordre d'empilement fixe, elle n'aurait qu'un intérêt limité car il suffirait de placer en dernier dans le code XHTML l'élément que l'on veut voir se positionner au premier plan. En gérant l'événement `onclick` par exemple pour le paragraphe ou chacune des images avec le code JavaScript suivant :

```
onclick="this.style.zIndex++"
```

nous permettons au visiteur de placer au premier plan l'élément qu'il désire et éventuellement d'inverser cet ordre à chaque nouveau clic. Notons de nouveau qu'en JavaScript le nom de la propriété devient `zIndex` et que l'opérateur ++ signifie simplement qu'il faut augmenter la valeur de la propriété de 1 unité. De même le texte du paragraphe, recouvert partiellement par l'image « un », peut être mis au premier plan pour être entièrement lisible.

La visibilité de l'image incluse dans le paragraphe (repère ❽) est contrôlée par la pseudo-classe `:hover`. Si le visiteur positionne le curseur sur cette image (en le laissant immobile, sinon on obtient un effet de clignotement), elle devient invisible en laissant l'espace qu'elle occupait vide dans le paragraphe. Cet effet est obtenu en donnant à la propriété `visibility` la valeur `hidden` en cas de survol (il est annulé automatiquement quand le curseur quitte la zone de l'image).

Exemple 13-11. Visibilité et empilement

```
<?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>Visibilité et ordre d'empilement</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-size:18px;}
   img.un{position:absolute;top:160px;left:350px;}❶
   img.deux{position:absolute;top:260px;left:450px;}❷
   img#couv{float: left;}❸
   img#couv:hover{visibility:hidden;}❹
   p{background-color: #EEE; position:absolute; top:40px;  left:50px;}❺
  </style>
 </head>
 <body>
  ❻<div>
   ❼<p onclick="this.style.zIndex++"><big>XHTML </big>:
   ❽<img id="couv" src="couvxhtml_gd.jpg" alt="XHTML CSS" height="130"
   ➥ width="107" />
    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 </p>
   ❾<img src="couvxhtml_gd.jpg" alt="XHTML CSS" height="285" width="240" class="un"
   ➥onclick="this.style.zIndex++" />
   ❿<img src="couvphp5gd.png" alt="XHTML CSS" height="285" width="240" class="deux"
   ➥onclick="this.style.zIndex++" />
  </div>
 </body>
</html>
```

La figure 13-17 donne le résultat obtenu après avoir successivement mis le paragraphe devant la première image (en cliquant sur le paragraphe), la seconde image devant la première (en cliquant sur la seconde), puis en plaçant le curseur sur l'image incluse dans le paragraphe. Par une série de nouveaux clics, il est possible de revenir en arrière.

Figure 13-17

L'état de la page après la modification de l'empilement et de la visibilité de certains éléments

Exercices

Exercice 1 : Incorporez trois images dans une page en définissant pour l'élément `<img />` les propriétés CSS `width` et/ou `height` (sans utiliser les attributs de même nom). Que se passe-t-il si les dimensions intrinsèques des images dépassent ces valeurs ?

Exercice 2 : Dans l'exercice précédent, comment procéder au dimensionnement CSS de façon que chaque image occupe la place qui lui est nécessaire d'après les dimensions du fichier image ?

Exercice 3 : Si les dimensions d'une image sont définies en pourcentage de celles de son conteneur, et que ses proportions largeur/hauteur sont inconnues, comment faire pour qu'elle ne soit pas déformée ?

Exercice 4 : Incluez trois éléments `<div>` contenant du texte, l'un dans l'autre, et définissez la largeur à 70 % du précédent pour chacun d'eux. Le premier doit avoir 800 pixels de haut et les suivants doivent correspondre à 80 % de la hauteur du précédent.

Exercice 5 : Créez deux paragraphes d'une hauteur de 300 pixels et d'une largeur de 700 pixels. Incluez-y un texte très long et gérez le débordement du texte afin qu'il soit entièrement lisible.

Exercice 6 : Créez cinq titres de niveau 2 et affichez les sous-formes de liste (indice : voir la propriété `display`).

Exercice 7 : Dans un élément `<div>`, incluez un élément `<span>` contenant du texte et donnez-lui le style bloc.

Exercice 8 : Créez un menu vertical dont les éléments sont des liens `<a>`.

Exercice 9 : Créez une page contenant un paragraphe incluant du texte et deux éléments `<img />` qui se suivent. Faites flotter les images, la première à gauche et la seconde à droite.

Exercice 10 : Reprenez l'exercice précédent et empêchez le flottement de la deuxième image.

Exercice 11 : Placez trois images de tailles initiales différentes dans une page. Écrivez les styles pour qu'elles s'affichent avec la même taille. Ensuite, positionnez-les afin d'obtenir un effet de cascade avec un décalage horizontal et vertical constant pour chaque image par rapport à la précédente. L'utilisateur doit pouvoir mettre chacune d'elles au premier plan en cliquant dessus (voir la propriété `z-index`).

Exercice 12 : Créez une mise en page à trois colonnes de largeurs respectives de 20 %, 65 % et 15 %. La première et la troisième doivent contenir respectivement un menu et une liste de liens créés à partir d'images. La colonne centrale doit avoir un contenu éditorial.

Exercice 13 : Créez une mise en page selon le modèle de la figure ci-dessous :

La colonne de gauche (repère ❶) a une largeur de 200 pixels et le bandeau (repère ❷) une hauteur de 150 pixels. Le bandeau contient le titre du site, la colonne de gauche un menu et la zone principale (repère ❸) du texte et des images au choix. Le premier lien du menu doit afficher une page ayant la même structure, le même contenu dans les zones ❶ et ❷, mais un contenu éditorial différent dans la zone ❸.

Exercice 14 : Reprenez l'exemple précédent de façon à ce que les zones ❶ et ❷ soient fixes dans la fenêtre du navigateur.