Jean Engels

HTML 5 et CSS 3

Cours et exercices corrigés

 ISBN : 978-2-212-13400-1

EYROLLES

2

Structure d'un document HTML 5

Avant de créer des pages web et de leur donner un contenu, nous allons déterminer une structure générale commune à toute page en conformité avec les spécifications HTML 5. En fonction des besoins, les codes des exemples 2-1 à 2-4 serviront de base à la constitution de toutes nos pages. Il suffira donc de les copier dans votre éditeur préféré, puis de les compléter avec un contenu particulier pour chaque page.

Les éléments de base

Le langage HTML 5 est une amélioration du langage HTML 4, avec des simplifications par rapport à la version XHTML qui était de mise avant lui. Tout document peut donc débuter de la même manière par la déclaration suivante (exemple 2-1 repère ❶) :

```
<!DOCTYPE html>
```

Vient ensuite l'élément racine `<html>` (repère ❷) qui inclut les éléments `<head>` (repère ❸) et `<body>` (repère ❻). Chacun de ces éléments a un contenu et donc une balise d'ouverture et une balise de fermeture, `<head>` incluant obligatoirement l'élément `<title>` (repère ❺) et un élément `<meta />` (repère ❹) qui contient la définition du jeu de caractères utilisé dans la page, et `<body>` ayant au moins un élément enfant ; ici, il s'agit de `<h1>` (repère ❼). La structure minimale d'un document HTML 5 est donc semblable à celle de l'exemple 2-1. Le fichier contenant ce code doit avoir une extension `.html` ou `.htm`.

Exemple 2-1 Structure minimale d'un document HTML 5

```
<!DOCTYPE html> ❶
<html> ❷
 <head> ❸
  <meta http-equiv="Content-type" content="text/html;charset=UTF-8" /> ❹
  <title> HTML 5 et CSS 3 </title> ❺
 </head>
 <body> ❻
  <!-- Tout le contenu de la page -->
  <h1>Le corps de la page minimale</h1> ❼
 </body>
</html>
```

Nous retrouvons bien dans cet exemple la structure arborescente décrite au chapitre 1. L'élément racine, au sens XML du terme, est `<html>` et inclut les éléments `<head>` et `<body>`. L'élément `<head>` contient l'élément `<title>` qui est obligatoire ainsi que la déclaration du jeu de caractères dans un élément `<meta />` ; l'élément `<body>`, qui ne doit pas être vide (ce qui est évident), contient un titre de niveau 1 `<h1>` sur lequel nous reviendrons plus loin. Du point de vue hiérarchique, `<html>` est bien le parent ou l'ancêtre de tous les autres.

Les commentaires

Tout ce qui est contenu entre les symboles `<!--` et `-->` est considéré par le navigateur comme des commentaires et n'est pas affiché dans la page, même s'ils se trouvent dans l'élément `<body>`. Comme pour tout langage de programmation, nous avons tout intérêt à commenter le code HTML afin d'en faciliter la relecture a posteriori. Notez toutefois que les commentaires seront visibles par l'internaute si celui-ci choisit d'afficher le code source de la page dans son navigateur. Évitez donc d'y inclure des informations confidentielles et d'y faire figurer des informations privées.

Un document HTML 5 peut incorporer du code PHP pour créer des pages dynamiques (interaction avec une base de données et création automatique de pages). Dans ce cas, le code PHP est compris entre les balises `<?php` et `?>`. Notez que le code de l'exemple 2-2 passé au validateur n'est pas déclaré conforme car il ne reconnaît pas ces balises, mais le code créé par l'exemple sera conforme.

Pour éviter les problèmes d'interprétation divergente entre les différents navigateurs, nous utiliserons systématiquement la déclaration du jeu de caractères avec l'élément `<meta />` dans chaque document. La structure minimale de ce type de page est donc celle de l'exemple 2-2. Notez que le fichier PHP a une extension propre, du type `.php` ou `.php5` par exemple, toujours en fonction de la configuration du serveur. Pour que le document HTML, que le serveur va finalement envoyer au navigateur, soit conforme au standard, il faut que le premier script placé au début du document (repère ❶) ne crée aucun code HTML (il peut par exemple ne contenir que des fonctions) et que le second (repère ❷) ne crée que du code HTML conforme. En respectant ces conditions, il n'y a aucune limite à l'utilisation de scripts PHP à l'intérieur d'un document.

Exemple 2-2. Structure d'une page HTML 5 contenant un script PHP

```
<?php ❶
// Placez ici du code PHP
?>
<!DOCTYPE html>
<html lang="fr">
 <head>
   <meta http-equiv="Content-type" content="text/html;charset=UTF-8" />
   <title> HTML 5 et CSS 3 </title>
  </head>
 <body>
  <!-- Le corps de la page -->
  <h1>Le corps de la page</h1>
  <?php ❷
   echo "<h2> Placez ici du code PHP créant un titre</h2>";
   ?>
 </body>
</html>
```

Dans toute la suite de cet ouvrage, nous n'utiliserons que la structure de base présentée à l'exemple 2-1.

La déclaration DOCTYPE

Nous avons déjà indiqué que le code d'une page HTML devait se conformer à des règles précises. La déclaration `DOCTYPE`, obligatoire dans tout document, précise le type de document qui va être créé. Dans HTML 5, cette déclaration est désormais réduite à sa plus simple expression par rapport à XHTML :

```
<!DOCTYPE html>
```

Dans ce code, la partie `html` donne le nom de l'élément racine du document.

L'élément racine <html>

L'élément `<html>` est l'élément racine du document, au sens XML du terme. C'est donc lui qui est le parent de tous les autres, soit directement, comme `<head>` et `<body>`, soit indirectement par l'intermédiaire de ces derniers. Il est donc le conteneur de premier niveau placé en haut de la hiérarchie de tous les éléments du document. Il n'existe que deux éléments enfants de l'élément `<html>`. En HTML 5, son contenu est constitué de l'en-tête du document, introduit par la balise `<head>` et terminé par la balise `</head>`, puis par le corps du document introduit par `<body>` et terminé par `</body>`, comme nous pouvons le vérifier dans les exemples.

L'élément racine possède les attributs communs dont les plus utiles sont :

- l'attribut `lang` dont la valeur est un code de langue normalisée qui indique la langue utilisée par défaut dans la page. Cette valeur sera reconnue par les moteurs de recherche

pour leur permettre d'indexer les pages du site en effectuant un tri par langue. Elles n'apparaîtront dans Google par exemple que si l'utilisateur a choisi le bouton France ;

- l'attribut `dir` qui indique le sens de lecture du texte de la page. Il peut prendre les valeurs `ltr` pour le texte qui se lit de gauche à droite (langues européennes) ou `rtl` pour le texte qui se lit de droite à gauche (langues orientales : hébreu, arabe).

Un élément `<html>` complet tel que nous pouvons l'utiliser s'écrira donc :

```
<html lang="fr" dir="ltr">
 <!--suite des éléments inclus -->
</html>
```

En pratique, pour des sites ayant un contenu dans une langue européenne, nous omettrons l'attribut `dir` ; dans nos exemples, nous n'utiliserons pas systématiquement l'attribut `lang`.

L'en-tête d'un document : l'élément <head>

L'élément `<head>` englobe un certain nombre d'informations utiles au bon affichage de la page web. Ces informations dites métadonnées sont contenues dans six éléments différents qui ont chacun un rôle bien déterminé. Il s'agit des éléments `<base/>`, `<link />`, `<meta />`, `<script>`, `<style>` et `<title>` dont nous allons étudier les rôles respectifs.

Aucun d'eux n'a de répercussion directement visible dans la page et seul le contenu de l'élément `<title>` sera visible, non dans la page mais dans la zone de titre du navigateur. Le bloc d'en-tête a donc la structure suivante, dans laquelle seuls les éléments `<title>` et `<base />` ne doivent figurer qu'une seule et unique fois, les autres n'ayant pas de limites.

```
<head>
  <title>Titre de la page</title>
  <meta http-equiv="Content-type" content="text/html;charset=UTF-8" />
  <base href="http://www.monsite.com" />
  <link rel="shortcut icon" type="images/x-icon"   href="../images/favicon.ico" />
  <meta name="Author" content="Jean ENGELS" />
  <script  type="text/javascript">
   <!-- Scripts JavaScript -->
  </script>
  <style type="text/css">
   <!-- Styles CSS -->
  </style>
</head>
```

Remarquons d'emblée que seuls les éléments `<title>`, `<script>` et `<style>` ont un contenu, et donc une balise fermante.

Nous allons maintenant détailler le rôle de chacun des éléments inclus dans `<head>`.

L'adresse de base : l'élément <base />

Il s'agit d'un élément vide et n'a donc pas de balise de fermeture. L'information qu'il contient est donnée dans son unique attribut `href` dont l'utilisation est obligatoire. Le

contenu de cet attribut est une URL qui fournit l'adresse de base de tous les fichiers utilisés dans la page quand leur adresse est transmise de manière relative. Si nous écrivons le code suivant :

```
<base href="http://www.funhtml.com/" />
```

le navigateur ira chercher une image dont l'URL est indiquée sur le serveur par `/html/images/monimage.gif` à l'adresse :

```
http://www.funhtml.com/html/images/monimage.gif
```

L'élément `<base />` possède également l'attribut commun `id`, qui ne peut servir qu'à modifier la valeur de l'attribut `href` au moyen d'un script JavaScript, selon la syntaxe suivante :

```
document.getElementById(id_element).href='valeur'
```

Les documents liés : l'élément <link />

Comme le précédent élément, il s'agit d'un élément vide dont l'information est contenue dans ses attributs. Il permet d'établir un lien entre la page HTML 5 en cours et un autre document externe nécessaire à la page. Nous l'utiliserons particulièrement pour lier la page à une feuille de style CSS contenue dans un fichier ayant l'extension `.css` ou un script JavaScript contenu dans un fichier sous l'extension `.js`.

L'utilisation de l'élément `<link />` crée l'incorporation virtuelle de ces documents dans le code de la page web. On parle d'incorporation virtuelle car la page se comportera comme si le code des fichiers externes faisait partie de la page, le contenu de ces fichiers n'étant pas visible, même en affichant le code source de la page.

La liaison avec les fichiers externes est déterminée par les attributs `rel`, `type`, `href`, `hreflang`, `media` et `sizes`.

- L'attribut `rel` indique le nom de la relation à établir avec le fichier externe. Il peut prendre les valeurs suivantes :
  - `rel="stylesheet"` si le fichier externe est une feuille de style.
  - `rel="alternate"` si le fichier est une page alternative (de rechange, proposée aux visiteurs dans les navigateurs).
  - `rel="shortcut icon"` ou `"icon"` pour faire référence à l'icône identifiant le site et qui s'affiche devant l'adresse dans les navigateurs les plus conformes.
  - `rel="previous"` ou `rel="prev"` si le fichier désigné est la page précédente dans l'ordre normal de consultation du site.
  - `rel="next"` si le fichier est la page suivante dans l'ordre normal de consultation du site.
  - `rel="help"` si le fichier est la page d'aide.

- L'attribut `type` précise le type de contenu du fichier externe. Il peut par exemple prendre les valeurs suivantes :

```
type = "text/css" pour une feuille de style CSS
type = "text/javascript" pour un script JavaScript
type = "text/html" ou "text/xml"
type = "images/x-icon" pour créer une icône
```

- L'attribut `href` contient l'adresse relative ou absolue de la ressource externe associée.
- L'attribut `hreflang` – qui, comme `lang`, prend pour valeur un code de langue – précise la langue utilisée dans le document cible.
- L'attribut `media` indique le type de média concerné par le document externe. Nous l'utiliserons en particulier pour lier une feuille de style en précisant le type de média visé par les styles du document CSS. Les valeurs de l'attribut `media` sont `screen` (écran d'ordinateur), `print` (imprimante), `tty` (télétype), `tv` (téléviseur), `projections` (rétro ou vidéoprojecteur), `handheld` (PDA, téléphone), `braille` (terminal braille), `aural` (navigateurs oraux) et `all` (tous les médias).
- L'attribut `sizes` indique la taille des icônes en particulier quand l'attribut `rel` vaut `icon`. Sa valeur doit comporter deux nombres entiers séparés par les caractères `x` ou `X`.

À titre d'exemple, nous pouvons écrire les codes suivants pour l'élément `<link/>` :

- Pour lier une feuille de style :

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

- Pour lier plusieurs feuilles de styles en précisant le média concerné :

```
<link rel="stylesheet" type="text/css" href="styleWeb1.css" media="screen"/>
<link rel="stylesheet" type="text/css" href="styleWeb2.css" media="print"/>
```

- Pour lier un script JavaScript :

```
<link rel="script" type="text/javascript" href="code.js"/>
```

- Pour créer une icône dans la barre d'adresse :

```
<link rel="icon" type="images/x-icon" href="/fashion.icon"/>
```

- Pour créer un lien de dépendance vers un document :

```
<link rel="next" type="text/html" href="page3.html" />
<link rev="prev" type="text/html" href="page1.html" />
```

- Pour créer un lien vers la page d'accueil :

```
<link rel="start" type="text/html" href="index.html" />
```

Les commentaires

Il est toujours utile de commenter votre code HTML 5, comme tout code informatique d'ailleurs, pour en permettre une meilleure compréhension, en particulier pour le relire un certain temps après l'avoir écrit. Tous les commentaires que vous écrirez seront ignorés par le navigateur, et vous pouvez donc vous exprimer librement. Pour placer un commentaire, vous devez l'ouvrir avec les symboles `<!--`, et le fermer avec les symboles `-->`. N'oubliez pas de fermer vos commentaires, sinon tout ce qui suit sera encore interprété en tant que tel, provoquant une erreur. Par exemple, on aura :

```
<!-- Voici un commentaire HTML
qui peut comporter plusieurs lignes
sans problème -->
```

Les méta-informations : l'élément <meta />

L'élément `<meta />` est également un élément vide pour lequel l'information est contenue dans ses attributs. Ces informations ne sont donc pas visibles dans la page mais elles sont destinées au serveur web, aux navigateurs et aux moteurs de recherche. Chaque information est identifiée par un nom et un contenu. Le nom de l'information est défini dans les attributs `name` ou `http-equiv` dont les rôles sont similaires, et la valeur associée est contenue dans l'attribut `content` sous la forme suivante :

```
<meta name="nom1" content="valeur1" />
<meta http-equiv="nom2" content="valeur2" />
```

Si nous utilisons l'attribut `http-equiv`, l'information indiquée dans l'attribut `content` sera présente dans les en-têtes HTTP envoyés par le serveur au navigateur sous la forme de couple nom/valeur.

La plupart des valeurs des attributs `name` et `http-equiv` sont des mots-clés. Nous allons nous arrêter sur la signification et l'utilité des plus courants d'entre eux.

- `name="author"` désigne le nom de l'auteur de la page sans pour autant créer un copyright.

  ```
  <meta name= "author" content="Jean Engels" />
  ```

- `name="keywords"` : cette valeur est très importante pour le créateur de site car son incorporation dans un document sert à l'indexation des pages web par les moteurs de recherche et les annuaires. L'attribut `content` associé à `name` contient la liste des mots-clés que vous allez choisir comme les plus représentatifs du contenu du site. Chaque mot ou expression est séparé des autres par une virgule. Il n'est pas rare d'utiliser plusieurs lignes de mots-clés dans l'attribut `content`. L'utilisation de l'élément `<meta />` à cette fin est donc des plus utile car il va permettre une mise en valeur de votre site qui apparaîtra dans les réponses fournies par les moteurs de recherche si vos mots-clés correspondent à la demande formulée par un internaute. Il est important de bien choisir ses mots-clés pour qu'ils correspondent vraiment au contenu du site et d'en multiplier les variantes dans la liste de l'attribut `content`. On pourra retrouver le même

mot au singulier et au pluriel, au masculin et au féminin. En revanche, il serait contre-productif d'utiliser les mots les plus demandés dans les moteurs de recherche sous prétexte d'attirer du public, alors qu'ils ne correspondent pas au contenu réel de votre site. Exemple de code :

```
<meta name="keywords" content="HTML 4, XHTML, HTML 5, CSS 2, CSS 3, design web,
➥ création de sites />
```

- `name="description"`. Dans le même ordre d'idée que la valeur précédente, il indique une brève description de l'information contenue dans le site. C'est cette description qui apparaît dans le moteur de recherche et il est donc essentiel qu'elle soit courte et correcte. Inutile de donner une description de 10 lignes alors que Google par exemple n'en affiche que deux. Il est également fortement recommandé d'utiliser cet élément `<meta />` car si vous ne fournissez pas vous-même une description de la page, Google et les autres moteurs de recherche utilisent les premières lignes de votre page qui ne sont pas nécessairement les plus explicites. Exemple de code :

```
<meta name="Description" content="Le site du livre &raquo; HTML 5 et CSS 3
➥ &raquo; de Jean Engels Editions Eyrolles" />
```

- `http-equiv="refresh"`. Quand il est associé à l'attribut `content` qui a pour valeur un nombre de `N` secondes, son utilisation permet de forcer le navigateur à recharger la page toutes les `N` secondes. On procédera ainsi pour un site aux informations renouvelées très fréquemment, par exemple un site de cotation boursière, ou pour afficher l'heure régulièrement par exemple à l'aide d'un script JavaScript ou PHP. Vous pouvez également utiliser cet élément pour rediriger automatiquement le visiteur vers une autre page du même site ou encore d'un autre site. On appliquera cette technique si le contenu du site a changé de nom de domaine. Pour cela, l'attribut `content` doit contenir le nombre `N` de secondes avant lequel la redirection sera effectuée, suivi d'un point-virgule (;) et de l'URL absolue de la nouvelle page. Par exemple, pour rediriger vers une autre page au bout de dix secondes, écrivez le code suivant :

```
<meta http-equiv="refresh" content="10; http:/www.funhtml.com/index/>
```

- `http-equiv="Content-type"`. Cette valeur de l'attribut permet de définir simultanément le type du document et le jeu de caractères employé dans la page. Comme nous l'avons déjà signalé, il faut l'utiliser impérativement si le jeu de caractères n'a pas été précisé, ou si cette information est absente, comme dans le cas des pages PHP. Si cette déclaration n'est pas effectuée, l'utilisation de cet élément `<meta />` est indispensable sous peine de voir le document non validé. L'attribut `content` contient alors le type du document suivi d'un point-virgule puis le code du jeu de caractères. Ce qui donne par exemple le code suivant :

```
<meta http-equiv="Content-type" content="text/html;charset=UTF-8" />
```

Les scripts internes : l'élément <script>

Nous avons vu qu'il était possible de lier la page HTML 5 avec un fichier externe contenant du code JavaScript au moyen de l'élément `<link />`. Si cette solution correspond bien au concept de séparation des fichiers ayant chacun un rôle différent, il est également possible de réaliser la même opération avec l'élément `<script>` qui sera alors vide. Le type de langage utilisé est précisé dans l'attribut `type` qui est obligatoire et contient généralement la valeur `text/javascript` ou `application/javascript` pour JavaScript. Dans le cas d'un fichier externe, il faut employer l'attribut `src` qui donne l'URL du fichier externe du script qui possède l'extension `.js` pour JavaScript. Nous aurions par exemple le code suivant :

```
<script type="text/javascript" src="http://www.funhtml/html/fichiercode.js">
</script>
```

Il est toujours possible d'incorporer du code JavaScript dans une page de code au moyen de l'élément `<script>` mais, à la différence de l'utilisation précédente, le code du script est inclus entre les balises `<script>` et `</script>`. Dans ce cas, seul l'attribut `type` est requis. Il est d'usage courant d'inclure tout le code JavaScript dans un commentaire situé à l'intérieur de l'élément `<script>` pour qu'il ne soit pas interprété par les navigateurs dépourvus d'interpréteur (s'il en existait encore !) mais aussi et surtout si le visiteur a volontairement interdit l'exécution des scripts JavaScript.

Nous pouvons par exemple écrire le code suivant :

```
<script  type="text/javascript">
  <!--
  function debut()
  {alert('Bonjour HTML 5');}
  -->
</script>
```

En plus de l'attribut `id` commun à la plupart des éléments, l'élément `<script>` possède également les attributs suivants dont le rôle est annexe par rapport aux précédents.

- `charset` pour préciser le jeu de caractères utilisés dans l'élément.
- `defer` dont la seule valeur autorisée est `defer`. S'il est utilisé, l'interpréteur JavaScript du navigateur n'interprète pas le code contenu dans l'élément avant l'affichage de la page, ce qui rend l'affichage plus rapide. Cet attribut ne sera utilisé que si le code ne contient pas d'instructions provoquant un affichage direct dans la page du type de celles réalisées au moyen de la fonction `write` ().

À la différence des autres éléments présents dans l'en-tête du document, l'élément `<script>` peut être utilisé également dans le corps du document, directement inclut entre les balises `<body>` et `</body>`, ou indirectement dans de nombreux autres éléments inclus eux-mêmes dans `<body>` et dont la liste figure dans le tableau 2-1.

Tableau 2-1

Tous les éléments de la catégorie Phrasing (voir le tableau 2-3) ainsi que `<head>`.

Le code suivant utilise l'élément `<script>` et tous ses attributs :

```
<script type="text/javascript" charset="UTF-8" defer="defer">
<!-- Code JavaScript -- >
</script>
```

L'incorporation des styles : l'élément <style>

L'utilisation des styles CSS est étroitement liée à HTML 5 comme elle l'était déjà avec XHTML. Nous avons vu qu'ils pouvaient le plus souvent être écrits dans un fichier externe lié à la page avec l'élément `<link />`. Cependant, comme pour les scripts, il est possible de les placer directement dans le code HTML en tant que contenu de l'élément `<style>`. Il n'est pas interdit d'écrire tous les styles dans cet élément mais cette possibilité sera réservée aux cas où les styles sont peu nombreux. Il est aussi envisageable de lier un fichier externe contenant les styles communs à toutes les pages d'un site avec `<link />` et d'ajouter des styles particuliers à une page dans l'élément `<style>` de celle-ci.

En plus des attributs `id`, `lang`, `dir` et `title` que nous avons déjà rencontrés, et peu utiles ici, l'élément `<style>` possède les attributs suivants.

- L'attribut `type`, dont la présence est obligatoire, précise le type de feuilles de styles utilisées. Pour les documents HTML, il prend toujours la valeur `text/css`.
- L'attribut `media` précise le type de média concerné par la feuille de style. Il est donc possible d'incorporer plusieurs éléments `<style>` dans l'en-tête de la page, chacun étant destiné à un média différent (voir l'exemple 2-4). Il prend les valeurs suivantes : `screen` (écran d'ordinateur), `print` (imprimante), `tty` (télétype), `tv` (téléviseur), `projections` (rétro ou vidéoprojecteur), `handheld` (PDA, téléphone), `braille` (terminal braille), `aural` (navigateur oral) et `all` (tous les médias).

Le code suivant définit une couleur de fond jaune pour la page et une couleur bleue pour le texte pour le media écran :

```
<style type="text/css" media="screen">
 body{background-color:yellow;color:blue;}
</style>
```

Nous reviendrons en détail sur l'écriture des styles dans la seconde partie de cet ouvrage. Il nous faut simplement retenir ici la localisation des styles.

Le titre de la page : l'élément <title>

Chacun a pu remarquer dans son navigateur qu'avant l'affichage complet d'une page web, un titre apparaît dans la barre de titre située en haut de la fenêtre du navigateur. Ce texte est défini dans l'élément `<title>` qui est l'un des deux seuls dont la présence est obligatoire

dans l'élément <head>. Son contenu est un simple texte qui doit résumer le contenu de la page en une ligne maximum. Il est important de bien réfléchir à ce contenu car c'est aussi lui qui apparaîtra comme titre principal du site dans les moteurs de recherche. Il doit donc être accrocheur et bien correspondre à l'esprit de la page. Nous aurons par exemple le code suivant :

```
<title>Le site de HTML 5 et CSS 3 </title>
```

L'élément <title> possède les attributs globaux dont lang, dir et id que nous avons déjà rencontrés.

L'exemple 2-4 crée une page en utilisant la plupart des éléments possibles de l'en-tête <head> dont <title> (repère ❶), <base /> (repère ❷), un grand nombre d'éléments <meta /> (repère ❸) et <link /> (repère ❹), les éléments <style> (repère ❺) et <script> (repère ❻). L'élément <body> (repère ❼) contient un minimum d'éléments pour obtenir un affichage, à savoir un titre (repère ❽) et une image (repère ❾). Nous reviendrons dans la section suivante sur cet élément important et sur ce qu'il peut contenir.

Exemple 2-4 Les éléments de l'en-tête du document

```
<!DOCTYPE html>
<html>
 <head>
  <meta http-equiv="Content-type" content="text/html;charset=UTF-8" />
    <!-- Element <title> --> ❶
    <title> HTML 5 et CSS 3 </title>
    <!-- Element  <base /> --> ❷
    <base href="http://www.funhtml.com/html/" />
    <!--Elements <meta />  --> ❸
    <meta name="Author" content="Jean ENGELS"/>
    <meta name="Keywords" content="HTML 5, CSS 3, Web" />
    <meta name="Description" content="Le site du livre &raquo; HTML 5 et CSS 3
    ➥ &raquo; de Jean Engels Editions Eyrolles" />
    <meta http-equiv="default-style" content="text/css" />
    <meta http-equiv="Refresh" content="1250; URL=http://www.funhtml.com/xhtml">
    <!-- Elements <link /> --> ❹
    <link rel="shortcut icon" type="images/x-icon" href="/html/images/
    ➥ favicon.ico" />
    <link rel="stylesheet" type="text/css" href="/xhtml/C2/messtyles.css" />
    <link rel="next" title="PHP 5" type="text/html" href="http://www.funhtml.com/
    ➥ php5" />
    <link rel="previous" title="Document HTML" type="text/html" href="/html/C2/
    ➥ exemple2-1.xml" />
    <link rel=alternate href="/en/html" hreflang=en type=text/html title="English
    ➥ HTML">
    <!-- Element <style> --> ❺
    <style type="text/css" media="screen">
      body {background-color:yellow;color:blue;font-size:40px}
    </style>
    <style type="text/css" media="print">
```

```
      body {background-color:white;color:green;}
    </style>
    <!-- Element <script> --> ❻
    <script  type="text/javascript">
     function debut()
     {alert('Bonjour HTML 5');}
    </script>
    <script type="text/javascript" src="http://www.funhtml/html/fichiercode.js">
    </script>
  </head>
  <body onload="debut()"> ❼
    <!-- Tout le contenu de la page -->
   <h1>Le corps de la page</h1> ❽
    <p>
      <img src="/html/images/html5.png" alt="HTML 5" /> ❾
    </p>
  </body>
</html>
```

La figure 2-1 montre le résultat minimal obtenu.

Figure 2-1

L'utilisation des éléments de l'en-tête

Le corps du document : l'élément <body>

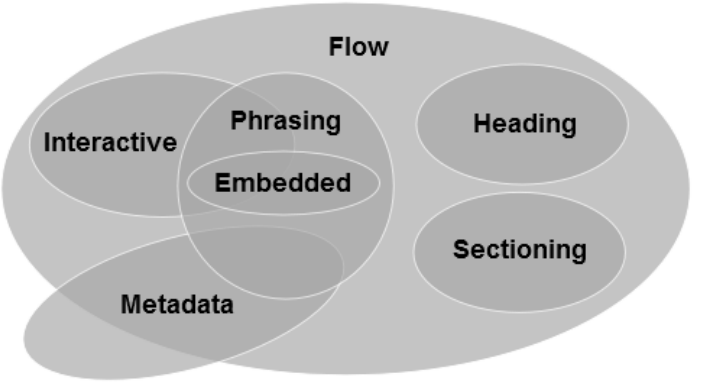
L'essentiel des éléments de l'en-tête `<head>` que nous venons d'aborder ont un contenu invisible dans le navigateur. L'élément `<body>` au contraire est le conteneur de l'ensemble des éléments textuels et graphiques d'une page web.

Les catégories d'éléments

La presque totalité des éléments HTML 5 sont désormais regroupés en catégories qui précisent leur rôle. Celles-ci ne sont pas indépendantes les unes des autres et il existe de nombreuses intersections entre ces différents ensembles comme le montre la figure 2-2.

Figure 2-2

Les catégories d'éléments



Les tableaux 2-2 à 2-8 donnent pour chaque catégorie la liste exhaustive des éléments qui la composent. Le tableau 2-9 donne la liste des éléments HTML 5 ne figurant dans aucune catégorie précise.

Tableau 2-2. L'ensemble des éléments de catégorie Flow

a, abbr, address, article, aside, audio, b, bdo, blockquote, br, button, cite, code, del, dfn, div, dl, em, embed, fieldset, figure, footer, form, h1, h2, h3, h4, h5, h6, header, hgroup, hr, i, iframe, img, input, ins, kbd, label, map, mark, math, menu, meter, nav, noscript, object, ol, output, p, pre, progress, q, s, samp, script, section, select, small, span, strong, sub, sup, svg, table, textarea, u, ul, var, video, wbr, texte brut

Tableau 2-3. L'ensemble des éléments de catégorie Phrasing

abbr, audio, b, bdo, br, button, cite, code, dfn, em, embed, i, iframe, img, input, kbd, label, mark, math, meter, noscript, object, output, progress, q, s, samp, script, select, small, span, strong, sub, sup, svg, textarea, u, var, video, wbr, texte brut

Tableau 2-4. L'ensemble des éléments de catégorie Interactive

a, button, embed, iframe, label, select, textarea, audio (si l'attribut controls est présent), img (si l'attribut usemap est présent), input (sauf pour un composant hidden), object (si l'attribut usemap est présent), video (si l'attribut controls est présent)

Tableau 2-5. L'ensemble des éléments de catégorie Heading

h1, h2, h3, h4, h5, h6, hgroup

Tableau 2-6. L'ensemble des éléments de catégorie Sectioning

article, aside, nav, section

Tableau 2-7. L'ensemble des éléments de catégorie Embedded

audio, embed, iframe, img, object, svg, video

Tableau 2-8. L'ensemble des éléments de catégorie Metadata

base, link, meta, noscript, script, style, title

Tableau 2-9. L'ensemble des éléments hors catégorie

caption, col, colgroup, dd, dt, figcaption, head, html, legend, li, optgroup, option, param, source, tboby, tfoot, th, thead, tr

Au fur et à mesure de leur étude, nous préciserons la catégorie des éléments qui peuvent être inclus dans tel élément. Ceci est utile pour savoir, comme en XHTML, quel peut être le contenu (les enfants) d'un élément et dans quels éléments il peut être inclus (ses parents). Il est important pour être conforme aux prescriptions HTML 5 de respecter les différentes inclusions. C'est donc le premier contrôle auquel vous devez procéder. Par exemple, l'élément `<body>` peut contenir tous les éléments de la catégorie Flow mais l'élément `<caption>` ne peut avoir comme parent que l'élément `<table>` et rien d'autre. Notons malgré tout que même si la catégorie Flow contient le texte brut, il est déconseillé d'inclure directement du texte dans l'élément `<body>` ; il vaut mieux le placer dans un élément enfant de `<body>`, ne serait-ce que pour lui appliquer simplement un style particulier et aussi pour qu'il apparaisse clairement dans la hiérarchie des éléments.

Exemple 2-5 Exemple d'inclusion des éléments dans une page HTML 5

```
<!DOCTYPE html>
<html>
 <head>
  <meta http-equiv="Content-type" content="text/html;charset=UTF-8" />
  <title>Les éléments du corps de la page</title>
  <meta name="Author" content="Jean ENGELS" />
  <meta name="Keywords" content="HTML 5" />
  <meta name="Description" content="Des inclusions" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="style1">
       body{background-color:   #EEE;}
  </style>
 </head>
 <body>
  <div>
   <address>Contact : html@funhtml.com</address>
   <blockquote>
    <hgroup>
     <h1>titre de niveau 1</h1>
    </hgroup>
    <dl>
     <dt>Liste de d&eacute;finition</dt>
     <dd>UN</dd>
    </dl>
   </blockquote>
  </div>
  <!--  -->
  <form action="script.php">
   <fieldset>
    <legend>commentaires</legend><br />
    <textarea cols="30" rows="5">texte</textarea>
   </fieldset>
  </form>
  <!--  -->
  <table border="1">
   <tbody>
    <tr>
     <td>Cellule 1 de tableau</td>
    </tr>
    <tr>
     <td>Cellule 2 de tableau</td>
    </tr>
   </tbody>
  </table>
  <!-- The document is valid HTML5 + ARIA + SVG 1.1 + MathML 2.0  -->
 </body>
</html>
```

L'exemple 2-5 présente plusieurs exemples d'inclusions successives d'éléments qui créent une hiérarchie arborescente présentée ci-dessous. L'élément `<body>` a trois enfants directs qui ont chacun à leur tour un ou plusieurs enfants, et ainsi de suite.

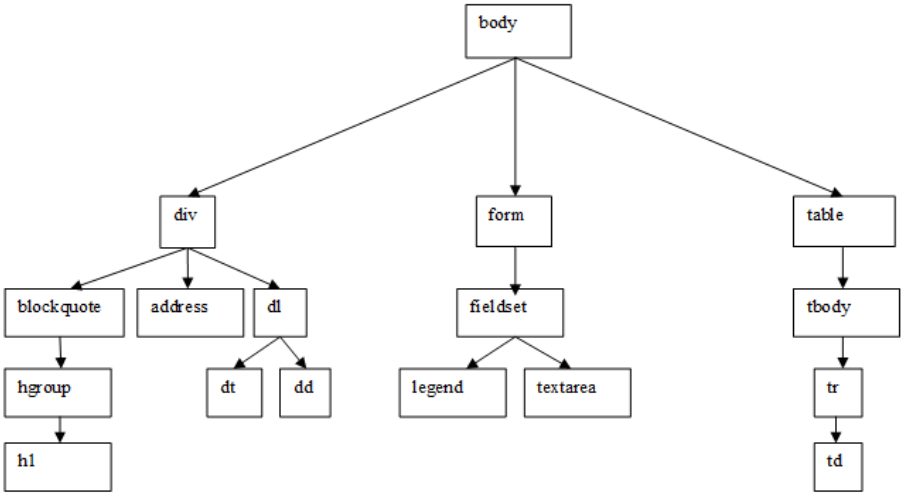


Figure 2-3

Arborescence d'un fichier HTML

Exercices

Exercice 1

La déclaration `DOCTYPE` est-elle obligatoire ? Où doit-elle être placée ? Quelles informations contient-elle ?

Exercice 2

Que faut-il faire pour que le code HTML 5 soit validé ?

Exercice 3

À quel emplacement est défini le jeu de caractères utilisé dans le document ?

Exercice 4

Où est déclaré le nom de l'élément racine du document ? Quel est son rôle et son nom ?

Exercice 5

Quels sont les éléments enfants de l'élément `<html>` ?

Exercice 6

À quoi sert l'élément `<head>` ? Quels sont ses éléments enfants ? Peuvent-ils être employés plusieurs fois dans le même document ?

Exercice 7

Quel élément est obligatoire dans l'élément `<head>` ? À quoi sert-il ?

Exercice 8

Écrire le code nécessaire à la liaison d'une feuille de style avec un document.

Exercice 9

Écrivez le code nécessaire à la liaison d'un document externe contenant des scripts JavaScript avec un document.

Exercice 10

Comment déclarer les mots-clés associés au site ? Quelle est l'utilité de cette déclaration ? Écrivez-en un exemple.

Exercice 11

Quel est le rôle de l'élément `<base />` ? Écrivez-en un exemple.

Exercice 12

Peut-on écrire le code suivant dans une page ?

```
<body>
 Bienvenue dans notre site
 <h1>Le site du HTML 5 et de CSS 3</h1>
</body>
```

Exercice 13

Peut-on inclure les éléments `<tbody>`, `<form>` et `<li>` directement dans l'élément `<body>` ?

Exercice 14

Écrivez un script qui affiche un message d'alerte quand l'utilisateur arrive sur le site.

Exercice 15

Écrivez le code CSS suivant à l'endroit adéquat :

```
body {background-color:white;color:green;font-size :20px}
```

Incluez ensuite un texte dans la page et testez le résultat. Vous devez obtenir un fond rouge, un texte bleu avec des caractères de 20 pixels.