CSS

Les boites

Ce document est un complément qui aborde uniquement la notion fondamentale de boite dans le langage CSS. Tout ce que nous avons étudié en cours ne sera pas repris ici, mais vous trouverez ci-dessous quelques rappels des notions que nous avons abordés et des liens vers des pages de site web cliquables pour complément.

Le principal site web cité dans ce document est celui de la ® Fondation Mozilla Mdn. En effet, il est considéré par les utilisateurs des langages liés au web comme une référence, bien qu'il existe un consortium officiel, plus technique, qui édicte les standards du web : le W3C.

Voici une petite astuce : si vous recherchez une information liée aux langages du web, il vous suffit de saisir dans un moteur de recherche le mot clé « Mdn » suivi des termes de votre recherche. Les résultats obtenus sont le plus souvent pertinents dès les premiers liens qui s'affichent. Tous les liens mentionnés dans ce document sont présentés en utilisant cette astuce au cas où vous souhaiteriez l'imprimer pour le lire.

1 - Le langage CSS

1.1 - La syntaxe du langage

```
Sélecteur(s) {
   Propriété-1 : valeur(s);
   Propriété-2 : valeur(s);
   …
   Propriété n : valeur(s);
}
```

Comme son nom l'indique, le sélecteur permet de désigner un élément (ou plusieurs) sur lequel on souhaite intervenir. À minima, cet élément est une balise HTML. Par exemple, le sélecteur simple `p { … : … ; }` permettra d'appliquer des propriétés à l'ensemble les balises `<p>…</p>` situées dans un ou de plusieurs fichiers HTML.

Important : la syntaxe du CSS impose que le sélecteur soit toujours suivi du caractère `"{"` (touches `"Option" + "("` sur Mac), et le bloc des couples `"propriété : valeur(s);"` doit se terminer par le caractère `"}"` (touches `"Option" + ")"` sur Mac).

Plusieurs règles et options gèrent l'utilisation des sélecteurs. Elles permettent de rendre la désignation des élément la plus précise possible. Vous les trouverez dans la documentation officielle Mdn : Mdn - Sélecteurs. On peut les regrouper dans les grandes catégories suivantes :

- Les sélecteurs simples
- Les combinateurs
- Les pseudo-classes
- Les pseudo-éléments

Le couple `"propriété : valeur(s);"` permet de mettre en forme le ou les éléments désignés par le sélecteur.

Important : la syntaxe du CSS impose qu'une propriété soit toujours suivie du caractère ":" et que la ou les valeurs se terminent par le caractère ";".

Il existe un grand nombre de propriétés, ce qui fait la richesse de ce langage et du design web. Vous en trouverez la liste sur Mdn - Propriétés CSS. Quand nous aurons détaillé la notion de boite, nous verrons que ces propriétés peuvent être classées en deux grandes catégories :

- Celles qui permettent d'intervenir sur une boite en tant que conteneur, comme sa dimension, ses marges, son fond, etc.
- Celles qui agissent sur le contenu d'une boite comme du texte, des images, l'organisation des autres boites qu'elle peut contenir, etc.

Il est important de noter que plus le sélecteur est précis, plus les propriétés qui lui sont associées seront importantes. En cas de conflit de propriété, seule la celle qui sera associée au sélecteur le plus précis sera retenue. Considérons l'exemple suivant :

```
a { color : blue;}
li a {color: green;}
a.alert {color: red}
```

Dans cet exemple, ces trois sélecteurs colorisent le texte des balises `<a>`. Ainsi, tous les textes à l'intérieur d'une balise `<a>` sont bleus (`a {color:blue;}`), à l'exception des textes suivants :

- Les textes des balises `<a>` qui ont comme ancêtre une balise `<li>` qui sont verts (`li a {color:green;}`).
- Les textes des balises `<a>` avec une classe `alert` (`<a class="alert">texte lien</a>`) qui sont rouges (`a.alert {color: red}`).

1.2 - Où écrire le CSS ?

1.2.1 Les styles en ligne

Les couples "`propriété:valeur(s);`" peuvent apparaître à l'intérieur d'une balise ouvrante HTML à l'aide de l'attribut `style`. Ils deviennent alors la valeur de l'attribut HTML `style`. Dans ce cas, les propriétés seront uniquement appliquées à la balise concernée. Par exemple :

```
<h1 style="width:400px; color:red; background-color:yellow;">Mon titre</h1>
```

Le texte de « Mon titre » est rouge, le fond de la boite est jaune et sa largeur est de 400 pixels.

1.2.2 Dans la balise `<style>` d'un fichier

Les sélecteurs et les propriétés sont écrits dans la balise `<style>` de la balise `<head>` du fichier HTML. Par exemple :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8"/>
    <title>Titre de la page</title>
```

```
    <style>
       h1 {
         width:400px;
         color:red;
         background-color:yellow;
       }
    </style>
  </head>

  </body> ... </body>

</html>
```

Dans cet exemple, le texte de toutes les balises `<h1>` de ce fichier HTML (s'il en contient plusieurs) est rouge, le fond des boites est jaune et leur largeur est de 400 pixels.

1.2.3 La feuille de style externe

Une feuille de styles externe est un fichier (avec l'extension « .css ») qui ne contient que du langage CSS (sélecteurs, propriétés, valeurs). Pour qu'elle soit active dans une page web, il est nécessaire de la mettre en relation avec le fichier HTML de cette page. Le lien est réalisé à l'aide de la balise `<link>` à placer dans la balise `<head>` du fichier HTML.

Voici un exemple d'un fichier HTML lié à une feuille de style externe `style.css` :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8"/>
    <link rel="stylesheet" type="text/css" href="style.css" media="screen"/>
    <title>Titre de la page</title>
   </head>
  </body> ... </body>

</html>
```

La balise `<link>` doit, à minima, comporter deux attributs :

- `rel` avec comme valeur `stylesheet` si le lien concerne une feuille de style. En effet, on peut utiliser la balise `<link>` pour lier le fichier HTML à des fichiers qui contiennent d'autres langages comme le JavaScript.
- `href` pour indiquer le nom du fichier et l'adresse absolue ou relative pour le trouver.

Les deux autres attributs de l'exemple sont optionnels :

- `type` : par défaut la valeur est `text/css`. On peut donc s'abstenir de l'écrire.
- `media` : précise le contexte d'affichage qui rendra active la feuilles de style (`all` par défaut).

Parmi les nombreuses valeurs possibles, on notera :

- `all` : tous les contextes d'affichage.
- `screen` : affichage sur un écran.
- `print` : lors de l'impression.
- etc.

Le fichier style.css ne contient que le langage CSS. Ainsi, si l'on écrit dans ce fichier :

```
h1 {
    width:400px;
    color:red;
    background-color:yellow;
}
```

Le texte de toutes les balises `<h1>` du fichier HTML (s'il en contient plusieurs) lié à `style.css` est rouge, le fond des boites est jaune et leur largeur est de 400 pixels.

Remarques importantes :

- Un fichier HTML peut être lié à plusieurs feuilles de styles externes. Dans ce cas, il faut autant de balises `<link>` qu'il y a de feuilles de style externes.
- L'ordre d'écriture de ces balises `<link>` dans la balise `<head>` a de l'importance, car il peut réserver des surprises lors de l'affichage s'il existe des conflits de propriétés. La règle est d'écrire en dernier les balises `<link>` faisant référence aux feuilles de styles les plus importantes.
- Dans la pratique, tous les fichiers HTML d'un site web sont liés à une ou plusieurs feuilles de styles externes. Cela permet de gérer plus facilement la mise en forme de l'ensemble des pages d'un site web en intervenant que sur ces feuilles de styles externes.
- L'écriture du CSS en ligne ou dans la balise `<style>` permettent la personnalisation de certaines pages d'un site web en cas de besoin. Ces écritures sont également très utilisées pour la mise en page des newsletters (envoi par email d'un fichier HTML mis en page).

2 - Qu'est-ce qu'une boite ?

Commençons par énoncer un principe simple, mais très important, car c'est sur lui que repose l'essentiel des règles du langage CSS :

1 balise HTML = 1 boite

Pour bien comprendre ce principe, il est nécessaire de biens définir les concepts que nous allons utiliser :

- Le terme de balise HTML est également connu sous les noms d'élément, de nœud ou de boite dans

d'autres langages informatiques, tels que le XML, le CSS et le JavaScript. Tous ces termes renvoient en fait au même objet, à la même entité.

- En CSS, on se représente une boite comme un rectangle. Si on utilise le terme de boite plutôt que de rectangle, c'est pour insister sur l'idée qu'une boite peut avoir un contenu, mais sa forme est rectangulaire.

De ce principe, nous pouvons déjà en déduire les conséquences suivantes que nous détaillerons tout le long de ce document :

- Toutes les propriétés du langage CSS agissent sur la disposition des boites dans une page, sur leur mise en forme (contenu et conteneur) ainsi que sur leurs animations.
- Puisque les propriétés du langage CSS peuvent être appliquées seulement aux boites et qu'une balise est une boite, alors le langage CSS nécessite un fichier qui contient des balises. Pour un site web, il s'agit d'un fichier HTML.
- Les sélecteurs du langage CSS permettent de désigner les boites sur lesquelles on souhaite intervenir pour les positionner, les mettre en forme et les animer à l'aide des propriétés.
- Tout fichier HTML contient des balises, donc des boites. Lors de la rédaction d'un fichier HTML, on choisit les balises en fonction du contenu (textes et média) dans le but de permettre à la « machine » de comprendre, même superficiellement, ce contenu (ceci est un en-tête, un pied de page, le contenu principal, un article, une section, un menu de navigation, un titre, un sous-titre, un paragraphe, un lien, une image, une vidéo, un son, une citation, un texte important, une emphase, etc.).
  Donc, lorsque l'on met en forme un fichier HTML à l'aide du CSS, on ne part pas de rien. Ce fichier contient forcément un certain nombre de balises imbriquées les unes dans les autres.
- Pour des besoins de manipulation, de mise en forme et de disposition des boites, il peut arriver que les balises contenues dans un fichier HTML ne suffisent pas. Le HTML a prévu deux balises neutres (elles n'ont pas de signification particulière) que l'on peut utiliser, avec parcimonie, pour ne pas trop s'écarter du rôle du HTML :
  - La balise `<div>` : balise de type bloc (notion que nous détaillerons plus bas) qui permet de regrouper des balises en vue de les manipuler comme un bloc.
  - La balise `<span>` : balise de type en ligne (notion que nous détaillerons plus bas) qui peut être utile pour mettre en forme une portion particulière de texte.
  - Ces balises sont souvent associées à des sélecteurs de classes (cf. Mdn - Sélecteurs de classe).

3 - Comment se représenter une page web ?

En reprenant le principe énoncé précédemment, le webdesigner étudie scrupuleusement le fichier HTML de la page web, avant de commencer la mise en page pour identifier les balises et leurs imbrications.

Son étude débouche sur une représentation visuelle de la page web à l'aide de boites rectangulaires.

Nous allons partir d'un exemple de fichier HTML dont vous trouverez le contenu en page 6 qui sera un support pour expliquer les notions que nous aborderons tout au long de ce document.

index.html

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

    <head>
        <meta charset="utf-8"/>
        <meta name="description" content="Feuille de style en cascade - CSS : Les boites et le box model"/>
        <link rel="stylesheet" href="reset.css">
        <link rel="stylesheet" href="style.css">
        <title>CSS - Les boites</title>
    </head>

    <body>

        <header>
            <p><em><u>Complément des cours en présentiel © Éric NÊME</u></em></p>
            <nav>
                <ul>
                    <li><a href="index.html">Accueil</a></li>
                    <li><a href="phoenix-code.html">Rappel sur l'éditeur de texte
                        <strong><em><small>®</small>Phœnix Code</em></strong></a></li>
                    <li><a href="balise-html.html">Les balises
                        <abbr title="HyperText Markup Language">HTML</abbr></a></li>
                    <li><a href="structure-html.html">La structure de base d'une page
                        <abbr title="HyperText Markup Language">HTML</abbr></a></li>
                </ul>
            </nav>
        </header>

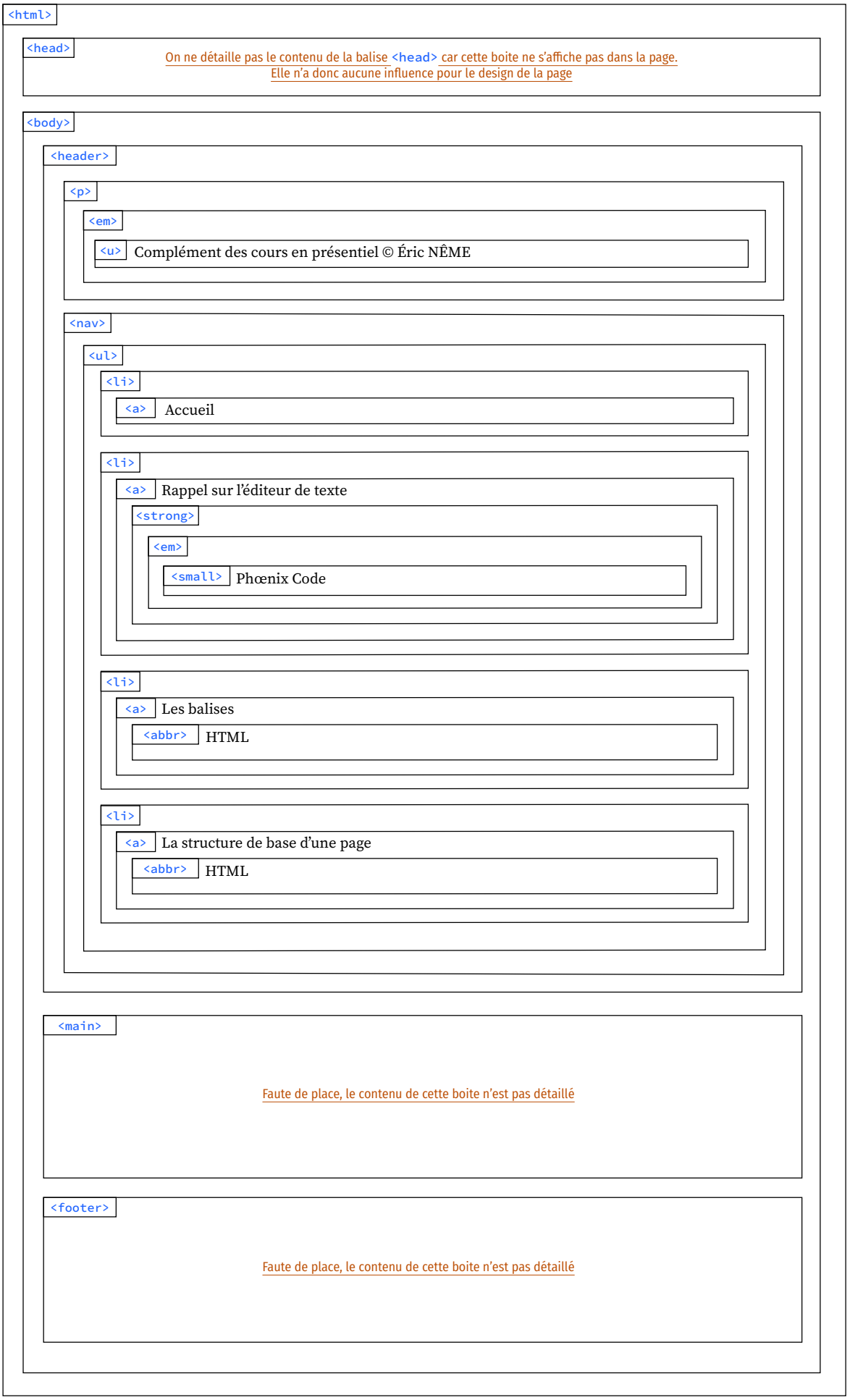
        <main>
            <h1>Formation au <abbr title="HyperText Markup Language">HTML</abbr></h1>
            <img src="html5.jpg" alt="logo html 5">
            <p>Ces quelques pages web reprennent l'ensemble du cours sur le <abbr title="HyperText Markup Language">HTML</abbr>.</p>
            <h2>Le code source des fichiers fournis</h2>
            <p>Vous pouvez ouvrir les fichiers dans un <strong>éditeur de texte</strong> comme <a href="https://phcode
            .dev/download/">Phoenix Code</a> que nous avons utilisé pendant les TD pour étudier leur contenu.<br/>
            Vous pourrez ainsi étudier dans le détail la structure des pages et les balises utilisées.</p>
            <h2>Lecture des fichiers depuis le navigateur</h2>
            <p>Comme ces fichiers reprennent le cours sur le langage <abbr title="HyperText Markup Language">HTML</abbr>, vous
            pouvez également les lire depuis un <strong>navigateur (client web)</strong>, mais il est conseillé, comme mentionné
            ci-dessus, de bien les étudier également en les ouvrant dans votre éditeur de texte.</p>
            <h2>Le design des pages</h2>
            <img src="css3.jpg" alt="logo css 3">
            <p>Aucun des fichiers n'est mis en page, car le <abbr title="HyperText Markup Language">HTML</abbr> n'a pas ce
            rôle.<br/>
            Nous étudierons l'année prochaine le langage dédié au design des pages web : le <abbr>CSS</abbr>.<br/>
            Mais vous remarquerez que les navigateurs intègrent par défaut quelques propriétés de ce langage <abbr>CSS</abbr> pour
            réaliser une mise en page sommaire dans le but de « traduire visuellement » la signification de certaines balises,
            comme les en-têtes (h1, h2, …, h6), les éléments importants (strong), les emphases (em), les listes (ul, ol, dl, li),
            etc.</p>
            <p>Tout rédacteur pour le web doit s'attacher, en plus de la création de son texte, à choisir les balises <abbr
            title="HyperText Markup Language">HTML</abbr> les plus adaptées à la signification du contenu. Le design n'entre pas
            en ligne de compte à cette étape de la conception d'une page web. Le but est de faire comprendre à la machine,
            notamment aux moteurs de recherche (SEO), de quoi parle la page en mettant en évidence des mots clés et des blocs de
            contenu à l'aide des balises <abbr title="HyperText Markup Language">HTML</abbr>.</p>
        </main>

        <footer>
            <p>Contact (courriel) : <a href="mailto:eric.neme@univ-lyon3.fr">Éric NÊME</a></p>
        </footer>

    </body>

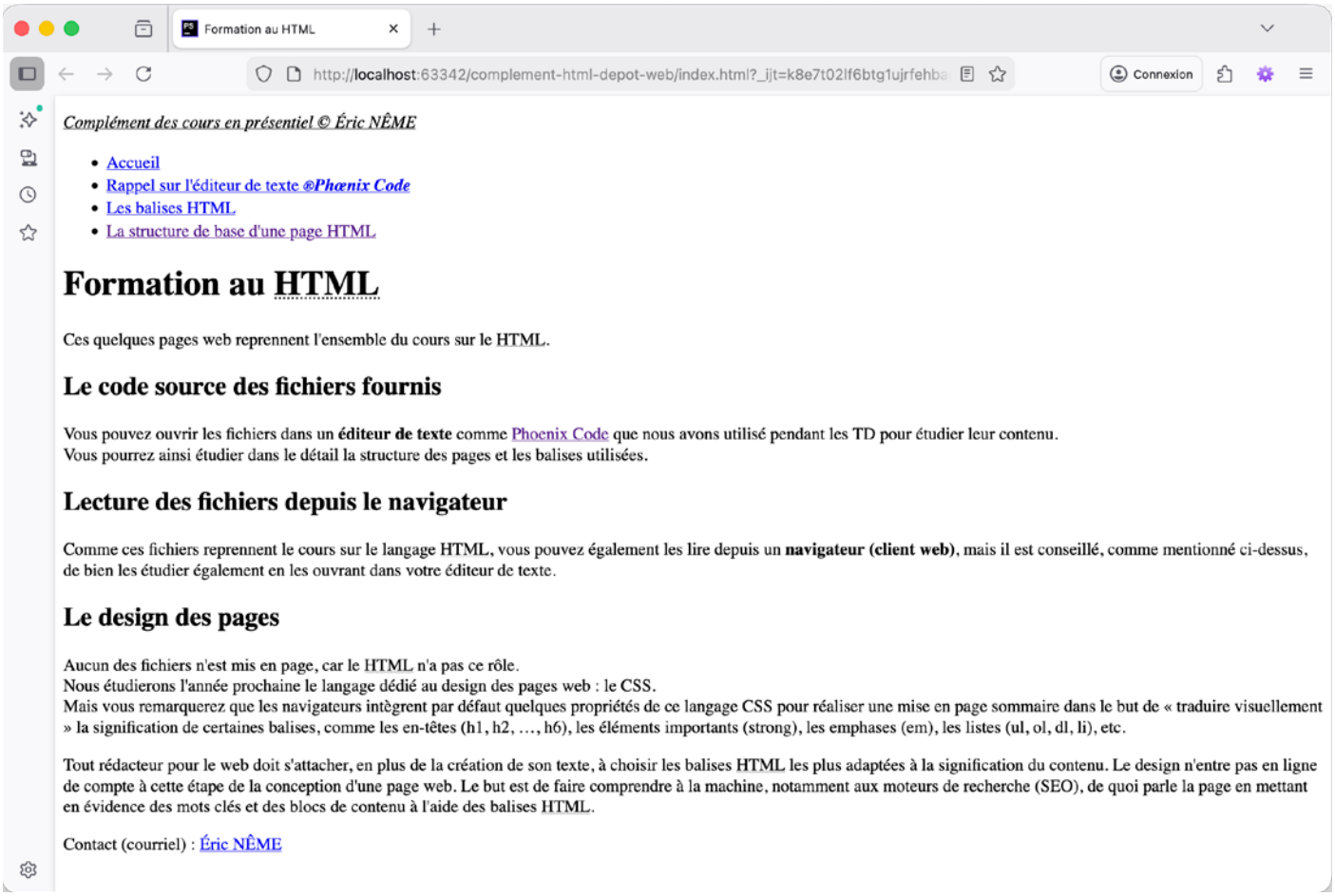
</html>
```

Voici la représentation partielle (faute de place, désolé...) du fichier HTML.



Prenez le temps de bien étudier la représentation de cette page web. Vous pouvez aussi compléter les balises `<main>` `<footer>` que je n'ai pas détaillées pour vous exercer.

Malgré l'imbrication de très nombreuses boites, le résultat d'affichage du fichier dans un navigateur semble très simple, mais comme le dit la célèbre formule : un train peut en cacher beaucoup d'autres 😉... :



3.1 Description de la structure du fichier

En CSS, on fait souvent référence à des concepts issus de la généalogie pour décrire l'imbrication des boites les unes dans les autres. De nombreuses règles touchant les sélecteurs et les propriétés reposent sur ces concepts, notamment la règle d'héritage des propriétés que nous présenterons plus loin (notion de cascade faisant partie du nom même de ce langage, Cascading Style Sheet - CSS).

On utilise cinq termes pour décrire l'imbrication des boites :

- Ancêtres ou ascendants.
- Parent.
- Enfants.
- Descendants.
- Frères ou sœurs.

À l'aide de ces concepts, toute boite peut être située dans ce contexte d'imbrication.
Par exemple, on peut dire de la boite `<header>` :

- Elle a deux ascendants : `<body>` et `<html>`.

- Elle a un parent : `<body>`.
- Elle a deux enfants : `<p>` et `<nav>`.
- Elle a 18 descendants (eh oui, il y a du monde...) : `<p>`, `<em>`, `<u>`, `<nav>`, `<ul>`, `<li>` x 4, `<a>` x 4, `<strong>`, `<em>`; `<small>`, `<abbr>` x 2.
- Elle a deux frères ou sœurs : `<main>` et `<footer>`.

Vous pouvez vous exercer en pointant n'importe quelle boite de ce fichier pour faire cet exercice.

Remarque importante : dans les documentations et tutos que vous pouvez rencontrer, certains auteurs n'utilisent pas les termes d'ascendants et de descendants, mais utilisent les mots de parents et d'enfants et, quand ils font référence au premier niveau d'ascendance ou de descendance, il rajoute l'adjectif «directe».
Ils parlent alors de parent direct ou d'enfants directes. Donc, pour eux :

- Ascendants ou ancêtres = parents
- Parent = parent direct
- Enfants = enfants directes
- Descendants = enfants

Malheureusement, cette imprécision peut provoquer des confusions, c'est la raison pour laquelle j'apporte ces précisions, notamment pour suivre les tutoriels proposés par le site ©grafikart en référence dans la webographie.

3.2 Quelques boites particulières

De cette structure de boites, on peut remarquer les particularités suivantes :

- La boite `<html>` est l'ascendante de toutes les boites : c'est la boite qui représente toute la fenêtre du navigateur. On la qualifie aussi d'élement racine (root).
  Il faut avoir à l'esprit que, lorsque l'on fait varier la taille de la fenêtre du navigateur, on fait varier les dimensions de la boite `<html>`.
- La boite `<head>` n'est jamais prise en compte dans un contexte de mise en page, puisque, par défaut, elle n'est pas affichée dans la fenêtre du navigateur.
- Comme la boite `<head>` n'est pas affichée, la boite `<body>` est la seule enfant de la boite `<html>`. Bien que ce ne soit pas vraiment le cas, on peut considérer que la boite `<body>` représente aussi la fenêtre du navigateur et, au même titre que la boite`<html>`, les dimensions de la boite `<body>` correspondent à la taille de la fenêtre du navigateur.

3.3 Que contient une boite ?

Il faut rester très attentif à la structure des boites et décrire scrupuleusement leur contenu. En effet, leur affichage dans la page dans un navigateur masque souvent la présence d'éléments intermédiaires, ce qui peut nous induire en erreur lorsque l'on manipule les boites à l'aide des propriétés du langage CSS.

En observant attentivement la représentation de notre page web, on constate qu'une boite peut contenir :

- Uniquement des boites, une ou plusieurs (ex. les boites `<body>`, `<header>`, `<nav>`, etc.).
- Uniquement du texte (ex. les boites `<u>`, `<small>`, `<abbr>`).

- Un mélange de texte et de boites (ex. les boites `<a>` sauf la première qui ne contient que du texte).

En conséquence, dans notre exemple, si l'on s'intéresse uniquement le contenu de la boite `<header>`, on s'aperçoit que le texte n'apparaît que dans quelques boites (`<a>`, `<u>`, `<small>`, `<abbr>`) alors qu'elle en comporte beaucoup plus (`<p>`, `<nav>`, `<ul>`, `<li>`, `<em>`, `<strong>`).

Le principe à garder à l'esprit est toujours le même : dès que l'on intervient sur une boite avec une propriété, il faut connaître son contenu.

4 - La notion d'héritage

Comme déjà mentionnée plus haut, la notion d'héritage est fondamentale dans le langage CSS. Prenez le temps de bien comprendre le mécanisme en jeu, il vous sera d'une grande utilité.

On peut classer l'ensemble des propriétés du CSS en deux catégories :

- Les propriétés héritées : si une propriété héritée est appliquée à une boite particulière, elle sera aussi appliquée individuellement à l'ensemble les descendants de cette boite. Autrement dit, les descendants de cette boite vont hériter de cette propriété.
- Les propriétés non héritées : si l'on applique une propriété non héritée à une boite particulière, seule cette boite sera affectée par la propriété, ses descendants ne seront pas concernés.

Pour illustrer le concept d'héritage, considérons l'exemple de la propriété `font-family`, qui permet d'afficher le texte avec une police particulière.
Nous avons abordé en cours les problèmes liés aux polices de caractères dans le cadre du web. Je vous rappelle simplement que vous ne pouvez pas connaître à l'avance les fichiers de polices de caractères installés sur l'ordinateur de l'internaute qui consulte votre page web. La propriété `font-family` permet de sélectionner un ensemble de polices afin de s'assurer du résultat d'affichage du texte.

Imaginons que l'ensemble du texte de notre page web doit s'afficher avec une police en bâton, on pourrait écrire :

```
font-family : tahoma, helvetica, arial, sans-serif;
```

Mais soyez très attentifs aux boites qui contiennent réellement du texte, nous en avons repéré quelques-unes au §3.3. Si cette propriété n'était pas héritée, il faudrait alors faire intervenir un sélecteur qui liste précisément toutes les balises qui contiennent du texte !...

Grâce au mécanisme d'héritage et parce que cette propriété est héritée, nous allons nous contenter d'utiliser `body` comme sélecteur et écrire :

```
body {
  font-family : tahoma, helvetica, arial, sans-serif;
}
```

La boite `<Body>` étant l'ancêtre de toutes les boites affichées dans la fenêtre du navigateur, toutes les boites descendantes héritent individuellement de cette propriété et toutes celles qui contiennent du texte l'afficheront avec une police en bâton (Tahoma, Helvetica, Arial ou tout autre sans-serif c'est-à-dire en bâton).

Remarque importante : un descendant qui reçoit de son ancêtre une propriété héritée peut redéfinir cette propriété. Par exemple, imaginons que nous souhaitons que le texte des boites `<a>` s'affiche en Times. Comme elles ont hérité de la propriété `font-family` de leur ancêtre `<body>`, elles s'affichent actuellement en Tahoma, Helvetica ou Arial. Pour obtenir du Times, on pourrait écrire :

```
a {
  font-family : times, serif;
}
```

Dès lors, tous les textes à l'intérieur des boites `<a>` s'affichent en Times même si elles sont toutes des descendantes de la boite `<body>`.

Étudions maintenant le cas d'une propriété non hérité comme la propriété `border` qui permet de placer une bordure autour d'une boite. Imaginons que nous souhaitons en placer une (trait bleu de 5 pixels d'épaisseur) autour de la boite `<header>` en écrivant :

```
header {
  border : 5px, solid, blue;
}
```

Comme la propriété `border` n'est pas héritée, seule la boite `<header>` aura une bordure bleue, tous ses descendants ne seront pas concernés.

Remarque : toute documentation CSS indique si une propriété est héritée (inherited en anglais).
Pour plus d'information sur l'héritage : Mdn - héritage.

5 - Les boites de type bloc, les boites de type en ligne

Les boites peuvent être classées en deux types :

- Les boites de type bloc qui, à l'affichage, provoque un retour à la ligne : ex. `<h1>` à `<h6>`, `<p>`, `<ul>`, `<li>`, `<table>`, `<div>` etc. Leur comportement est de refuser la présence d'autres boites à leur gauche ou à leur droite, même si l'espace est suffisant pour les accueillir.
- Les boites de type en ligne qui ne provoque pas de retour à la ligne : ex. `<a>`, `<em>`, `<strong>`, `<u>`, `<img>`, `<span>`, etc.

Le comportement de ces deux types de boites est à l'origine des règles d'affichage par défaut du contenu d'une page web dans un navigateur. On parle de flux naturel d'affichage, et il existe des propriétés CSS qui nous permettent de contrarier localement ce flux naturel.

En conséquence, dans le flux naturel d'affichage :

- Les boites de type bloc s'affichent les unes en dessous des autres quand elles sont frères ou sœurs.
- Les boites de type en ligne s'affichent les unes à côté des autres ou à l'intérieur d'un texte si elles sont entourées de texte.

D'autres comportements sont sous-entendus, comme :

- Une boite de type bloc occupe toute la largeur que lui autorise son parent, même si elle a peu de contenu.
- Une boite de type en ligne n'occupe que l'espace dont elle a besoin pour afficher son contenu.

6 - Les propriétés CSS fondamentales pour une boite

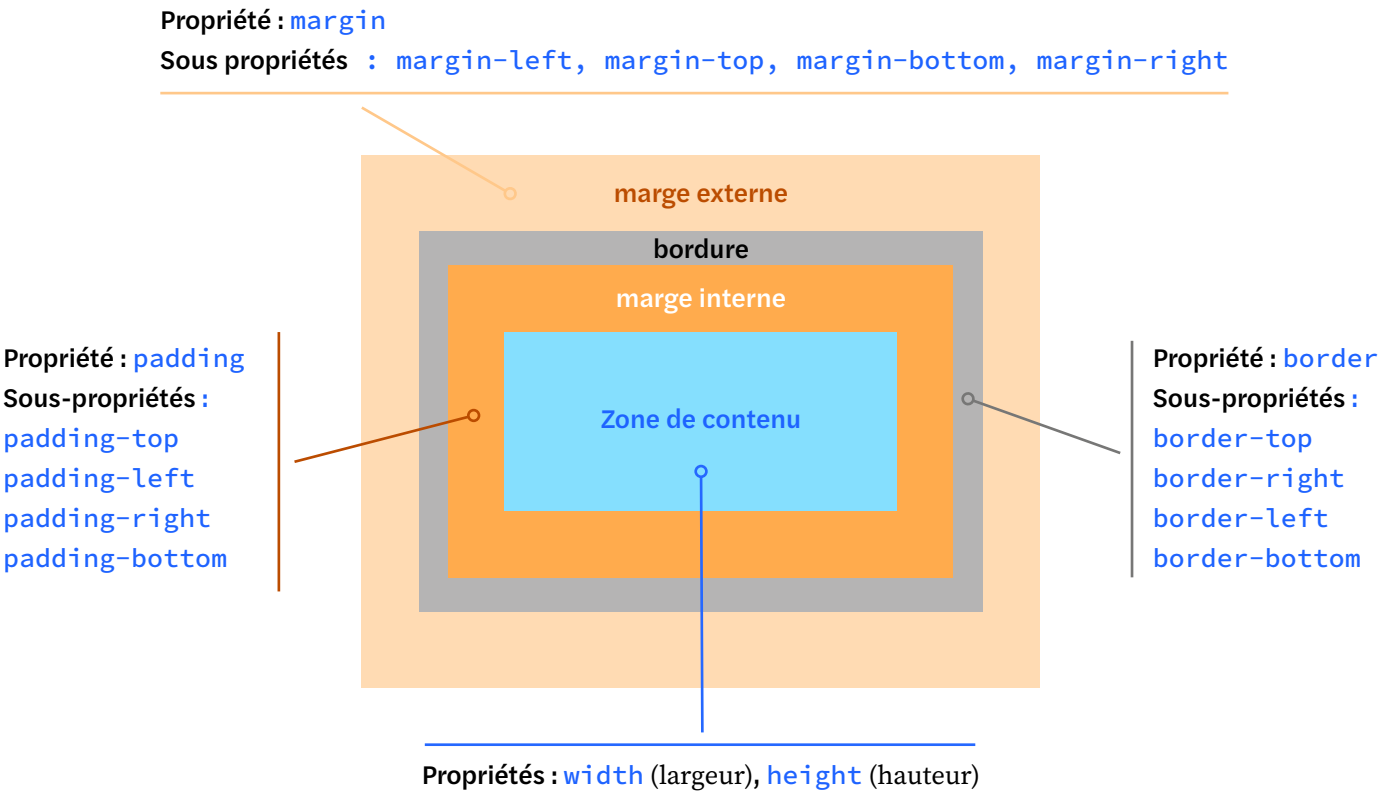
Il existe un grand nombre de propriétés qui affectent une boite en tant que conteneur. Nous allons étudier les principales qui sont des incontournables. On parle du modèle de boite CSS (box model) qu'il faut absolument comprendre et maîtriser :

- La propriété `width` : largeur.
- La propriété `height` : hauteur.
- La propriété `border` : bordure.
- La propriété `padding` : marge interne.
- La propriété `margin` : marge externe.
- La propriété `background` : le fond d'une boite (cette propriété ne fait pas partie du modèle de boite, mais je la mentionne, car elle est très utilisée dans le design d'une boite)

Important : toutes ces propriétés peuvent être appliquées à une boite de type bloc sans restriction. Pour les boites de type en ligne, certaines d'entre elles peuvent être utilisées, mais avec des limitations que nous détaillerons plus loin.

6.1 - Pour les boites de type bloc

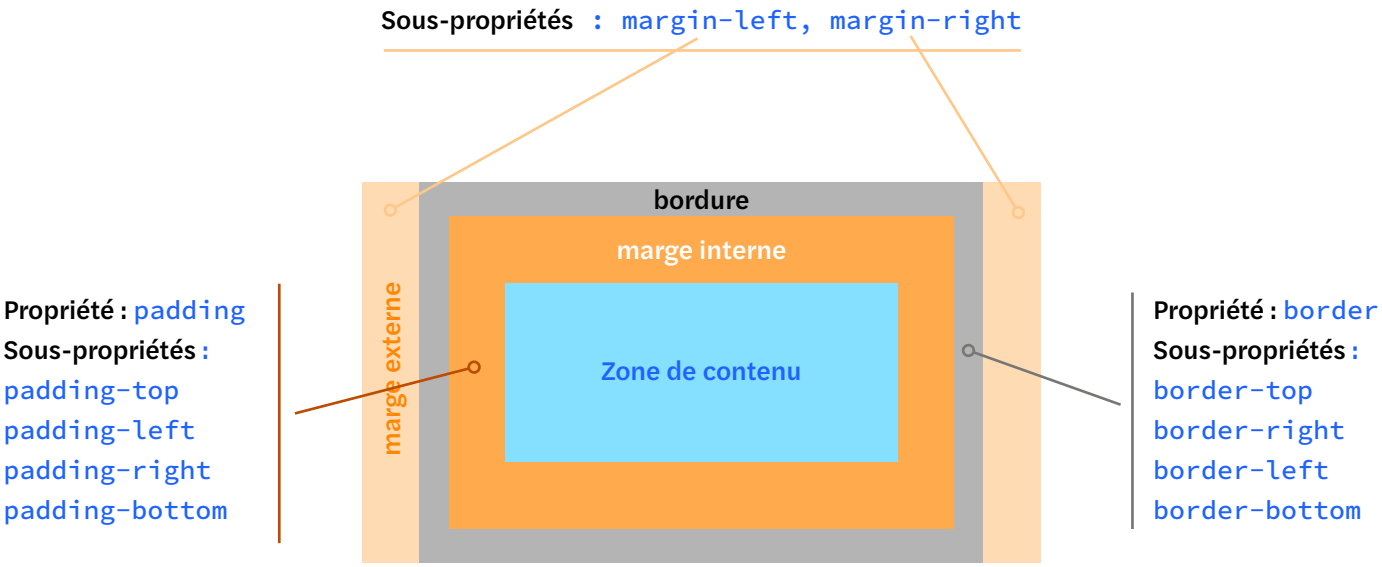
Le schéma suivant représente le modèle de boite et les propriétés concernées :



Important : pour la propriété `background`, la couleur ou les images de fond se répandent par défaut dans tout l'espace délimité par les bords externes de la bordure. La marge externe, quant à elle, est toujours transparente. Pour plus de précision sur la propriété background et ses nombreuses valeurs : Mdn - Background.

6.2 - Pour les boites de type en ligne

Pour les boites de type en ligne, les propriétés sont moins nombreuses, ce qui correspond au schéma suivant :



- Pas de propriétés `width` et `height` : les dimensions de la zone de contenu s'adaptent au volume du contenu (le texte, l'image, la vidéo, etc.).
- Pas de propriétés `margin-top` et `margin-bottom`.
- Les propriétés `padding` et `border` sont accessibles, mais avec des limitations : mal utilisées, ces propriétés peuvent provoquer des chevauchements avec les éléments qui entourent la boite.
- Pour la propriété `background`, la remarque précédente sur la zone d'affichage est identique.

6.3 - Cas particulier de la boite `<img>`

Les concepteurs du CSS ont pensé à tout pour nous faciliter la mise en page... Bien que la boite `<img>` soit de type en ligne, on peut lui appliquer toutes les propriétés du modèle de boite. Rappelons toutefois qu'il n'est pas recommandé de redimensionner une image avec les propriétés `width` et `height` ou les attributs HTML de même nom. Il est préférable de le faire depuis un logiciel dédié avant leur intégration dans la page web.

6.4 - La zone de contenu

Comme son nom l'indique, cette zone est l'espace que le contenu d'une boite va pouvoir occuper. Par conséquent, la première règle à respecter est de s'assurer de laisser assez d'espace pour que le contenu s'affiche entièrement. Si cet espace est insuffisant, le contenu excédentaire déborde de la boite et chevauche les éléments qui entourent la boite... En cas de débordement, la propriété `overflow` peut aider à maîtriser le contenu excédentaire (cf. Mdn - Overflow).

Pour les boites de type bloc, les propriétés `width` et `height` permettent de déterminer les dimensions de la zone de contenu. Utiliser ces deux propriétés en même temps est risqué, il faut s'assurer que le volume du contenu est adapté et invariable (ce qui est rare pour des pages de site web modifiables à tout moment).

La plupart du temps, on utilise une seule de ces propriétés (ex. `width`) laissant la liberté à l'autre propriété de s'adapter au volume du contenu en cas de variation.

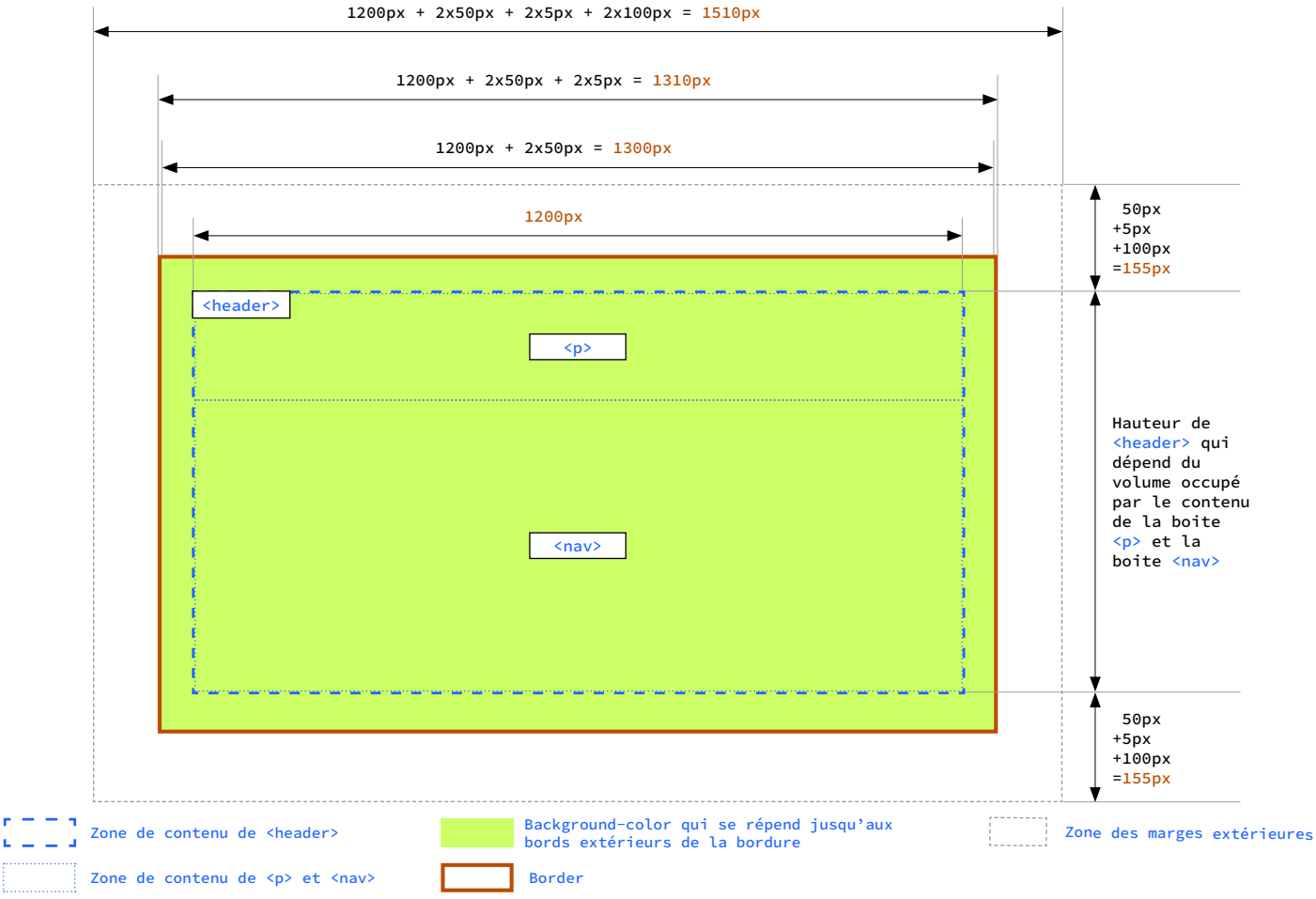
Pour bien comprendre toutes les caractéristiques de cette zone de contenu, il faut avoir à l'esprit tout ce qui a été dit au §.3, à savoir :

- Une balise = une boite, et chaque boite à forcément une zone de contenu.
- Une boite peut contenir :
  - Que des boites
  - Que du texte, une image, une vidéo, etc.
  - Un mélange de boites, de texte, d'images, de vidéos.

Il faut bien retenir ceci : pour une boite donnée, quels que soient le nombre et les caractéristiques de ses enfants, ces derniers n'ont pas d'autre possibilité que de se placer dans la zone de contenu de cette boite.

Prenons un exemple avec la boite `<header>` de notre fichier HTML qui a deux enfants `<p>` et `<nav>` et ajoutons à cette boite les propriétés suivantes :

```
header {
      width : 1200px;
      padding : 50px;
      border : 5px solid #d76700;
      margin : 100px;
      background-color : #ccff66;
}
```



On constate avec ce schéma que :

- La largeur de la zone de contenu `<header>` est de `1200px,` car on a fixé la propriété `width` à cette valeur.
- Ses deux enfants `<p>` et `<nav>` ne peuvent pas dépasser la largeur imposée par la zone de contenu de leur parent `<header>` (`1200px`).
- Les deux boites `<p>` et `<nav>` étant des boites de type bloc, leur comportement est d'occuper tout l'espace de la zone de contenu de leur parent. Elles ont donc toutes les deux une largeur de `1200px` (cf. §.5).
- La boite `<nav>` apparait dans le fichier HTML après la boite `<p>`. Elle n'a pas d'autre choix que de se placer sous la boite `<p>` puisque cette dernière occupe toute la largeur de la zone de contenu de `<header>` qu'elle ne peut pas franchir.
- Comme nous n'avons pas fait intervenir la propriété `height` pour la boite `<header>`, la hauteur de sa zone de contenu s'adapte aux hauteurs que ses deux enfants `<p>` et `<nav>` réclament pour afficher leur propre contenu.
- La propriété `margin` de la boite `<header>` définie des limites que les boites frères ou sœurs (les boites `<main>` et `<footer>` dans notre exemple) ne peuvent pas franchir.
- Finalement, la largeur occupée par la boite `<header>` n'est pas de `1200px` (propriété `width`), mais `1510px` : `width` + `padding-left` + `padding-right` + `border-left` + `border-right` + `margin-left` + `margin-right`, à savoir :
  `1200px` + `50px` + `50px` + `5px` + `5px` + `100px` + `100px` = `1510px`.
- Cette largeur de `1510px` doit donc être disponible dans la zone de contenu de son parent (`<body>` dans notre cas).
  Que se passe-t-il si la largeur de la zone de contenu de la boite `<body>` est inférieure à `1510px` ?
  La largeur `<body>` pouvant être considérée comme la largeur de la fenêtre du navigateur (cf. §.3.2), une largeur de fenêtre plus petite que `1510px` provoque l'apparition de la barre de défilement horizontale en bas de la fenêtre pour pouvoir afficher le contenu masqué de la boite `<header>`.

Cet exemple nous a permis de détailler finement toutes les règles du modèle de boite CSS qui est incontournable pour gérer la mise en page des boites que contient tout fichier HTML.

Dans le but de bien le maîtriser et pour apporter quelques remarques complémentaires, nous allons rajouter les propriétés suivantes à notre exemple :

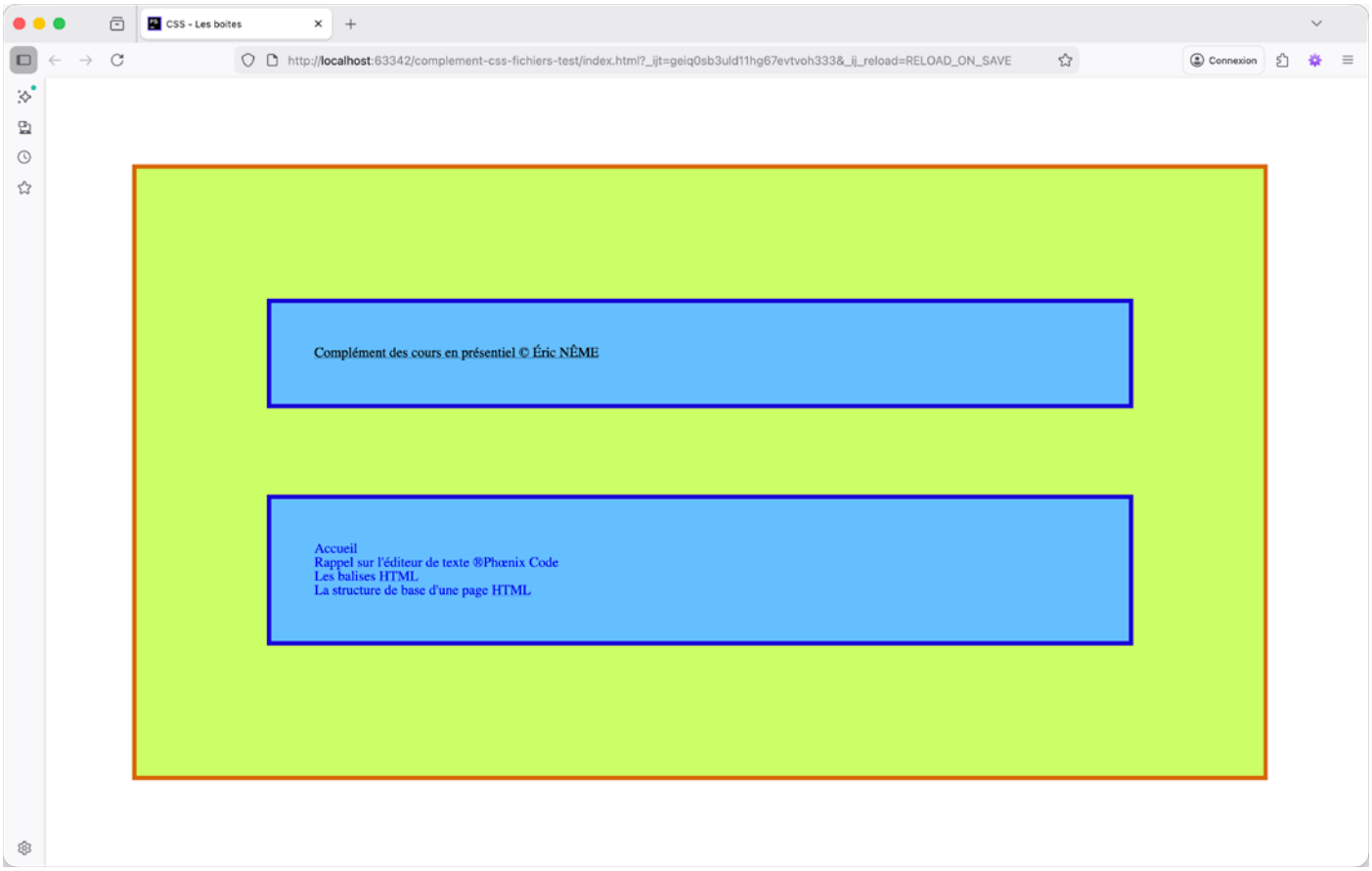
```
header p, header nav {
    padding : 50px;
    border : 5px solid #1900d7;
    margin : 100px;
    background-color : #66bfff;
}
```

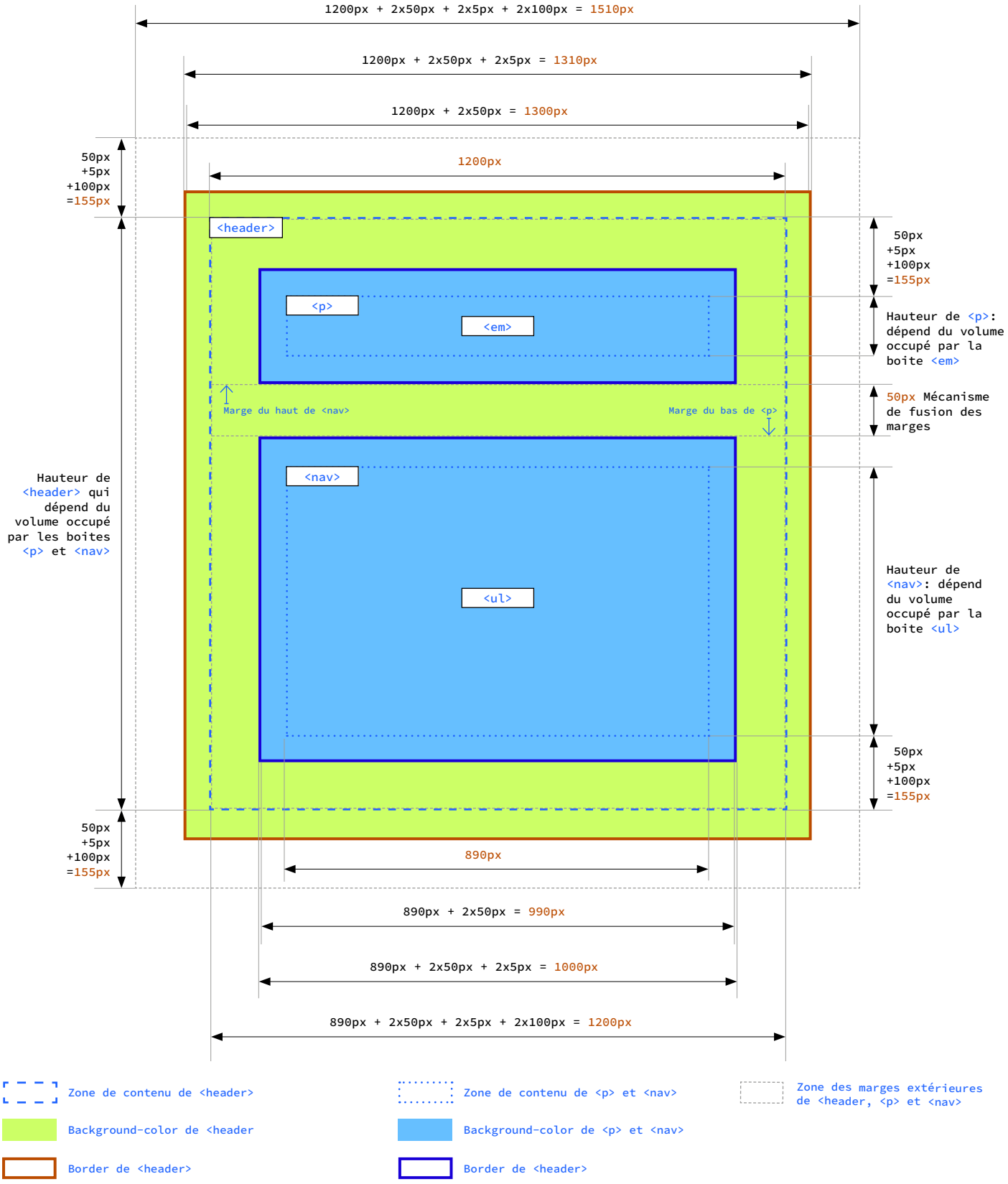
Explications du sélecteur :

- `header p` : toutes les boites `<p>` qui ont comme ancêtre une boite `<header>`.
- `,` : et.
- `header nav` : toutes les boites `<nav>` qui ont comme ancêtre une boite `<header>`.

Dans notre cas, ce sélecteur (judicieusement choisi 😉) pointe uniquement les 2 enfants de `<header>` : `<p>` et `<nav>`.

Voici une capture d'écran de la fenêtre du navigateur affichant la boite `<header>` de notre fichier HTML (le schéma explicatif de ce résultat d'affichage se trouve en page suivante) :





Toutes les règles du modèle de boite énoncées plus haut qui concernaient la boite `<header>` s'appliquent également aux boites `<p>` et `<em>`.
Nous leur avons ajouté les mêmes propriétés, sauf une : la propriété `width` qui est absente.
Accessoirement, les couleurs de la bordure et du fond ont été modifiées pour mieux visualiser ces boites à l'affichage de la page dans le navigateur.

Puisque la propriété `width` est absente pour ces deux boites, on laisse le navigateur calculer lui-même sa valeur. Mais rien ne nous empêche de connaître précisément sa valeur : à vos calculettes !...

La largeur de la zone de contenu de la boite parente (`<header>`) à nos deux boites (`<p>` et `<nav>`) est de `1200px` (propriété `width` de `<header>`). On sait que ces deux boites en tant que type bloc occupent toute la largeur de cette zone de contenu. De plus, on connaît les propriétés suivantes pour nos deux boites :

- Marge externe de gauche : `100px`.
- Bordure de gauche : `5px`.
- Marge interne de gauche : `50px`.
- Marge interne de droite : `50px`.
- Bordure de droite : `5px`.
- Marge externe de droite : `100px`.

Toutes ces valeurs additionnées (`310px`) vont déjà occuper une partie de la largeur de la zone de contenu de `<header>`. L'espace restant sera alors attribué à la largeur de la zone de contenu de nos deux boites (la propriété `width`), à savoir : `1200px` - `310px` = `890px`.

Conclusion : les enfants des boites `<p>` et `<nav>` ne pourront se répandre que sur une largeur de `890px`. Si vous attribuez une propriété `width` supérieure à cette valeur à un enfant de ces deux boites, il sortira de la boite...

6.5 - Le mécanisme de fusion des marges (propriété `margin`)

Dans notre exemple, on observe l'effet de ce mécanisme : l'espace entre les boites `<p>` et `<nav>` devrait être de `200px` alors qu'elle est de `100px`. En effet, nous avons attribué à ces deux boites une marge externe de `100px`, donc, pour chacune d'entre elles, nous avons :

- Marge externe du haut : `100px`.
- Marge externe de droite : `100px`.
- Marge externe du bas : `100px`.
- Marge externe de gauche : `100px`.

En toute logique, la marge externe du bas de la boite `<p>` et la marge externe du haut de la boite `<nav>` devraient s'additionner (`100px` + `100px` = `200px`).

Le mécanisme de fusion des marges annule ce cumul et ne retient que la valeur de la marge la plus grande pour l'affecter à l'espace entre boites. Dans notre cas, les marges sont identiques (`100px`), il affecte donc `100px` d'espace entre ces deux boites.

Les concepteurs du CSS ont mis en place ce mécanisme pour nous faciliter la mise en page, car, dans la pratique, on cherche à obtenir une homogénéité des espaces comme ceux que l'on place entre les titres (boites `<h1>` à `<h6>`), les paragraphes (boite `<p>`), etc.

Dernière remarque, ce mécanisme de fusion des marges s'applique aussi pour des boites situées les unes à côté des autres, donc pour les marges externes droites et gauches.

6.6 La propriété `box-sizing`

Le modèle de boite que nous avons longuement détaillé est celui qui est respecté par défaut par tous les navigateurs (sauf pour les très anciennes versions d'Internet Explorer notamment).

Certains webdesigners (de plus en plus nombreux...) utilisent la propriété `box-sizing` qui modifie légèrement les règles du modèle de boite CSS. Cela permet notamment de faciliter la représentation des dimensions des boites et de moins utiliser la calculatrice...

Vous n'êtes pas obligés d'utiliser cette propriété. Je vous la présente, car vous risquez très probablement de la rencontrer, puisqu'elle est de plus en plus utilisée. De plus, elle peut éventuellement mieux vous convenir dans votre travail de mise en page.

La modification du modèle de boite apportée par cette propriété est en fait très simple : elle redéfinit les propriétés `width` et `height` pour qu'elles correspondent non plus à la zone de contenu, mais aux longueurs entre les bords externes des bordures.

Détaillons cette propriété. Elle peut prendre deux valeurs :

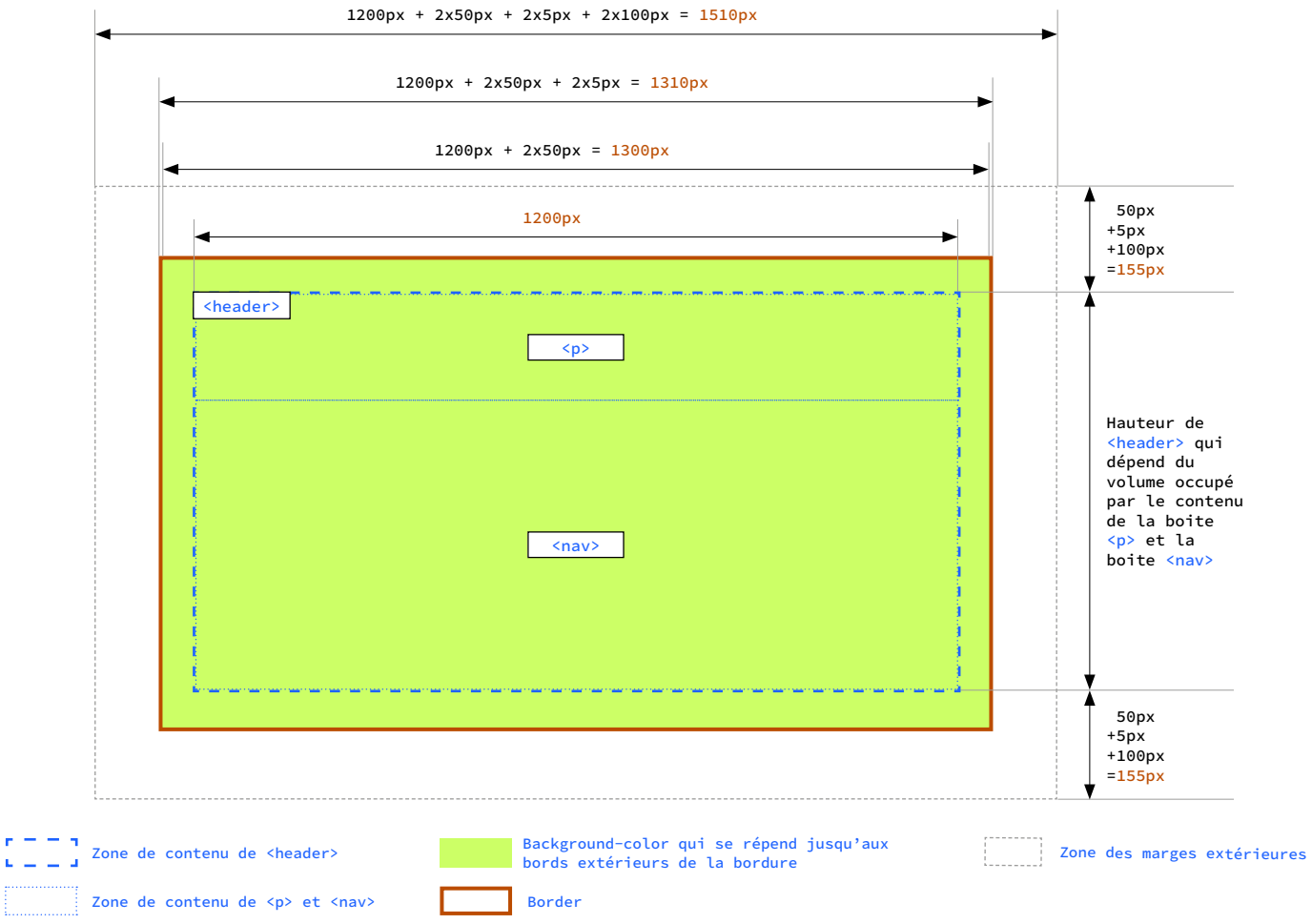
- `box-sizing : content-box;`
  C'est le comportement par défaut des navigateurs que nous avons expliqué. Il n'est donc pas utile de le préciser si vous restez dans le modèle de boite par défaut.
- `Box-sizing : border-box;`
  Cette fois-ci, les propriétés `width` et `height` se rapportent aux bordures (`border-box`) et plus à la zone de contenu (`content-box`).
  Dans ce cas :
  - La propriété `width` = largeur de la zone de contenu + la marge interne de gauche (`padding-left`) + la marge interne de droite (`padding-right`) + l'épaisseur de la bordure de gauche (`border-left`) + l'épaisseur de la bordure de droite (`border-right`).
  - La propriété `height` = hauteur de la zone de contenu + la marge interne du haut (`padding-top`) + la marge interne du bas (`padding-bottom`) + l'épaisseur de la bordure du haut (`border-top`) + l'épaisseur de la bordure du bas (`border-bottom`).

Cette redéfinition des propriétés `width` et `height` permet de laisser au navigateur le soin de calculer lui-même les dimensions de la zone de contenu. En fixant de cette manière la largeur d'une boite ou sa hauteur ou les deux, on peut mieux se rendre compte visuellement de leurs dimensions, puisqu'elles correspondent à l'espace délimité par les bordures.

Attention, il ne faut pas oublier les marges externes qui vont s'ajouter aux propriétés `width` et `height`.

Reprenons notre exemple et changeons notre CSS concernant la boite `<header>` pour obtenir les mêmes dimensions que précédemment:

```
header {
      box-sizing : border-box;
      width : 1310px;
      padding : 50px;
      border : 5px solid #d76700;
      margin : 100px;
      background-color : #ccff66;
}
```



À cause de la propriété `box-sizing` : `border-box`, on a changé la valeur de `width` (`1310px` au lieu de `1200px` précédemment) pour ne rien changer des dimensions initiales de la boite. Si nous avions conservé la propriété `width` à `1200px`, la boite aurait été réduite.

Remarque importante : pour éviter de se faire des nœuds au cerveau 😉, il est préférable d'appliquer cette propriété pour toutes les boites si l'on décide de l'utiliser.
Dans notre exemple, nous avons appliqué cette propriété à la boite `<header>` et pas aux boites `<p>` et `<nav>`. Nous sommes donc dans la situation suivante :

- Pour la boite`<header>`, `width` correspond à la largeur entre les bordures.
- Pour les boites `<p>` et `<nav>`, `width` correspond à la largeur de la zone de contenu.

Ce mélange n'est pas très confortable et peut rendre la mise en page complexe.
Donc, si l'on choisit d'utiliser `box-sizing` : `border-box`, il est préférable de l'appliquer à toutes les boites du fichier HTML, ce qui veut dire rajouter cette propriété pour chaque sélecteur, ce qui risque d'être un peu long.

Pour éviter cette répétition, on peut écrire au début de la feuille de style :

```
* {
    box-sizing : border-box;
}
```

Pour rappel, le sélecteur « * » est le sélecteur universel qui cible toutes les boites, c'est magique 😃!...

6.7 Rappels sur les propriétés `margin`, `padding` et `border`

Nous avons vu en cours que de nombreuses propriétés CSS sont composées d'une propriété générique et de sous-propriétés. Celles qui interviennent dans le modèle de boite n'échappent pas à la règle : les propriétés `margin`, `padding` et `border` possèdent elles aussi des sous-propriétés qui permettent d'attribuer des valeurs différentes pour les quatre côtés de la boite. Ainsi :

- Pour `margin` : `margin-top`, `margin-right`, `margin-bottom`, `margin-left`, sont les sous-propriétés qui permettent de personnaliser les quatre marges externes de la boite.
- Pour `padding` : `padding-top`, `padding-right`, `padding-bottom`, `padding-left`, sont les sous-propriétés qui permettent de personnaliser les quatre marges internes de la boite.
- Pour `border` : `border-top`, `border-right`, `border-bottom`, `border-left`, sont les sous-propriétés qui permettent de personnaliser les quatre bordures de la boite.

Pour les propriétés `margin` et `padding`, on peut également personnaliser les valeurs des quatre côtés de la boite en utilisant la propriété générique avec plusieurs valeurs. Voici comment cela fonctionne pour `margin` (la logique est la même pour `padding`) :

- `margin` : `10px` : les quatre côtés de la boite ont une marge externe de `10px`.
- `margin` : `10px 20px 30px 40px` : la marge externe du haut est de `10px`, celle de droite `20px`, celle du bas `30px` et enfin celle de gauche `40px`.
  On s'aperçoit que la première valeur correspond au côté haut de la boite, ensuite on tourne dans le sens des aiguilles d'une montre (haut, droite, bas et gauche).
- `margin` : `10px 20px` : les marges externes du haut et du bas sont de `10px`, celles de droite et de gauche sont de `20px`.
- `margin` : `10px 20px 30px` : la marge externe du haut est de `10px`, celles de droite et de gauche sont de `20px`, celle du bas de `30px`.

La propriété générique `border` n'est pas concernée, car elle contient déjà plusieurs valeurs (épaisseur, type de trait et couleur). Il faut donc passer par les sous-propriétés énoncées plus haut pour personnaliser les bordures des quatre côtés de la boite.
Il existe d'autres sous-propriétés pour `border` : cf. Mdn - border.

6.8 La valeur `auto` pour la propriété `margin`

La valeur `auto` signifie automatique. Elle permet de demander au navigateur de calculer lui-même cette valeur. On peut faire appel à cette valeur pour certaines propriétés CSS.
Concernant `margin` elle est très utile pour centrer une boite de type bloc dans la zone de contenu de son parent.

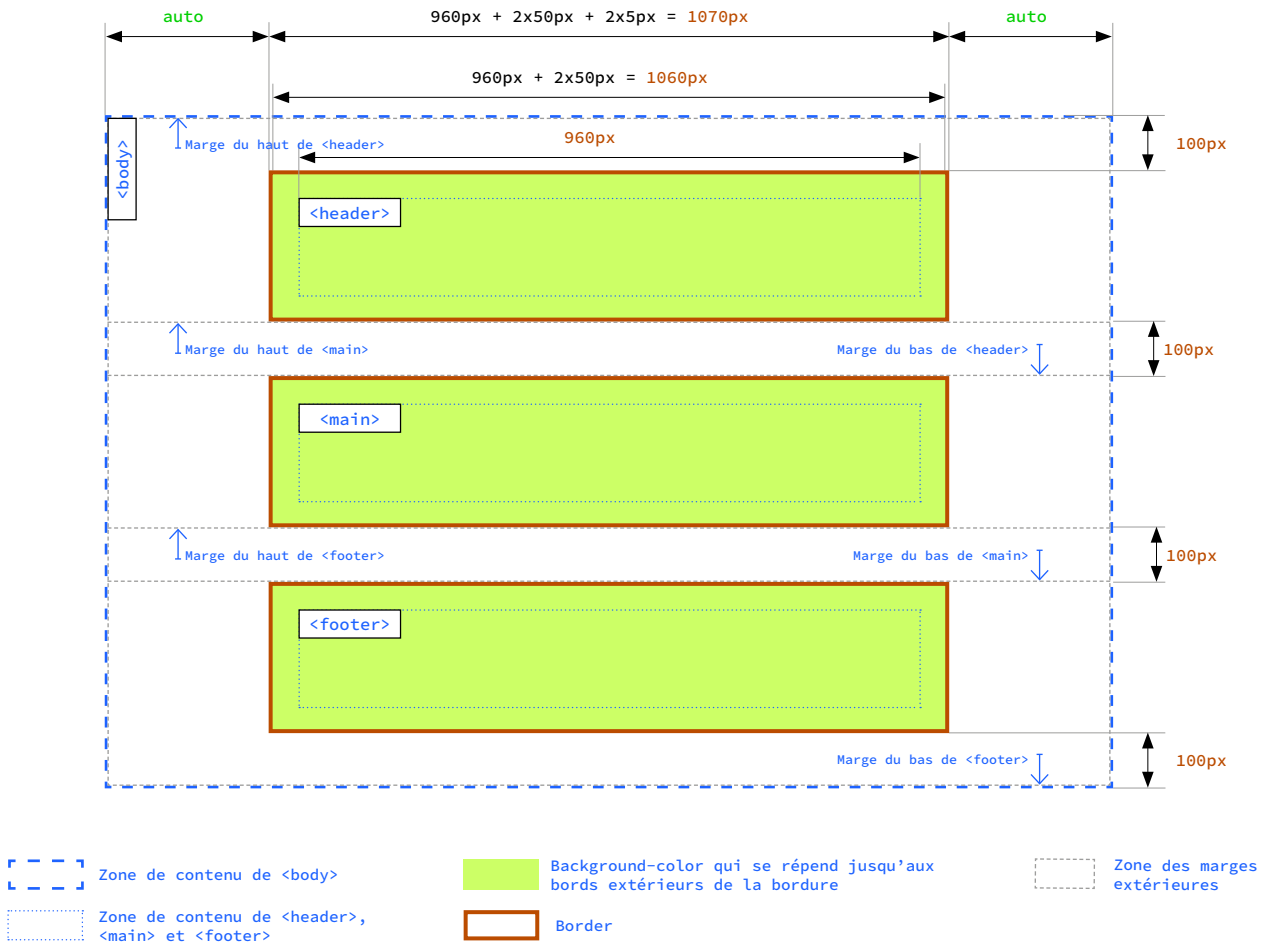
Reprenons un exemple à partir de notre fichier HTML avec comme objectif :

- Fixer la largeur des trois enfants`<header>`, `<main>` et `<footer>` de la boite `<body>` à `960px` (on choisit d'utiliser le modèle de boite par défaut).
- Les trois boites doivent être centrées dans la page.

Côté CSS on pourrait écrire :

```
header, main, footer {
     width : 960px;
     padding : 50px;
```

```
        border : 5px solid #d76700;
        margin : 100px auto;
        background-color : #ccff66;
}
```



Explication du schéma. Selon les propriétés attribuées aux boites`<header>`, `<main>` et `<footer>` nous pouvons dire que :

- La largeur de la zone de contenu de la boite `<body>` correspond à la largeur de la fenêtre du navigateur, on ne peut donc pas connaître à l'avance sa valeur.
- La distance qui sépare les bords extérieurs de droite et de gauche des boites`<header>`, `<main>` et `<footer>` est de `1070px`.
  Comme nous n'utilisons pas la propriété `box-sizing`, leur zone de contenu a une largeur de `960px` (propriété `width`) à laquelle on rajoute les marges internes et la bordure pour trouver cette largeur de `1070px`.
- La propriété `margin` contient deux valeurs :
  - `100px` qui correspond à la valeur des marges externes du haut et du bas. Avec le mécanisme de fusion des marges, les boites sont séparées verticalement de `100px`. De plus, la boite `<header>` est à `100px` du haut de la fenêtre du navigateur.
  - `Auto` qui correspond à la valeur des marges externes de gauche et de droite. Ces deux marges sont identiques et leur valeur est calculée par le navigateur lui-même de la manière suivante : (largeur de la zone de contenu de `<body>` - `1070px`) / 2.
    Par exemple, si la fenêtre du navigateur a une largeur de `2040px`, les marges de gauche et de droites seront de : (`2040px` - `1070px`) /2 = `485px`.

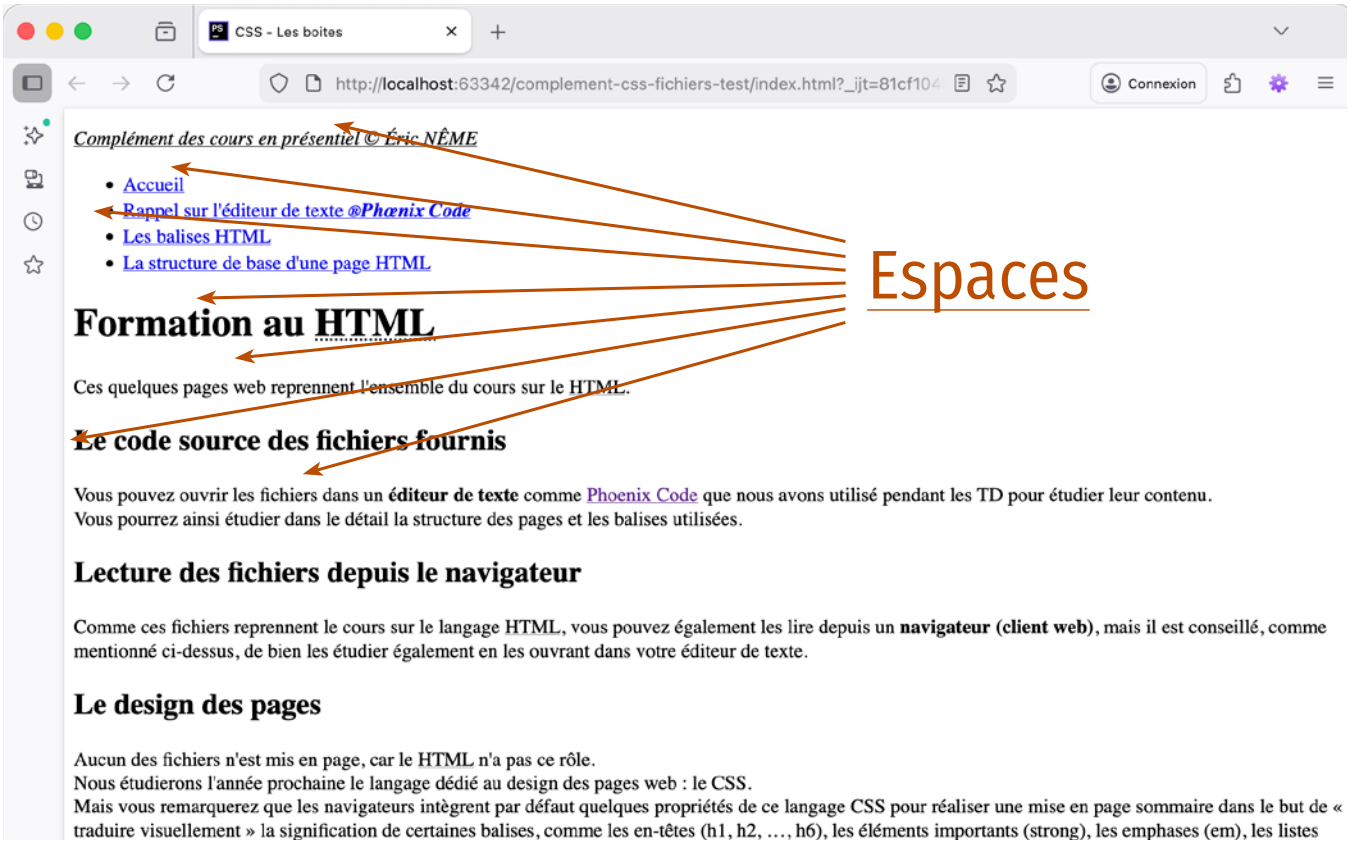
Comme ces deux marges externes sont identiques (`485px`) les boites `<header>`, `<main>` et `<footer>` sont centrées dans la boite `<body>`, et elles le resteront quelle que soit la largeur de la fenêtre du navigateur !

En conclusion, pour centrer une boite de type bloc dans la zone de contenu de son parent, on utilise la valeur `auto` de la propriété `margin`.

6.9 La notion de reset ou reboot

Pour en finir avec cette présentation du modèle de boite CSS, il faut mentionner que tous les navigateurs intègrent par défaut une feuille de style qui attribue aux boites des valeurs particulières aux propriétés du modèle que nous venons d'étudier.

Voici l'affichage de notre fichier HTML dans le navigateur Firefox sans ajouts de styles :



Tous les espaces qui entourent les textes sont déterminés par les propriétés `margin` et `padding` appliquées, par défaut, aux boites.

Ces espaces ne facilitent pas la maîtrise de la mise en page, notamment quand on souhaite placer les boites à des endroits précis et gérer leur distribution, au pixel près, dans la page.

La solution consiste à redéfinir toutes ces propriétés, en début de CSS, en mettant leur valeur à `0`. On appelle cette action un reset ou un reboot. On crée, le plus souvent, un fichier CSS que l'on nomme `reset.css` ou `reboot.css` dans lequel on redéfinit ces propriétés pour un grand nombre de boites.

Attention : dans le fichier HTML, la balise `<link>` faisant le lien avec ce fichier doit apparaître avant les autres balises `<link>` au risque d'écraser tout votre design :

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

    <head>
        <meta charset="utf-8"/>
        <meta name="description" content="Feuille de style en cascade - CSS : Les
         boites et le box model"/>
        <link rel="stylesheet" href="reset.css">
        <link rel="stylesheet" href="style.css">
        <title>CSS - Les boites</title>
    </head>

    <body...>

</html>
```

Dans cette capture d'écran de notre fichier HTML, nous avons rajouté deux feuilles de style externes. Le fichier `reset.css` sera lu par le navigateur avant le fichier `style.css` que nous utilisons pour la mise en forme de notre page.

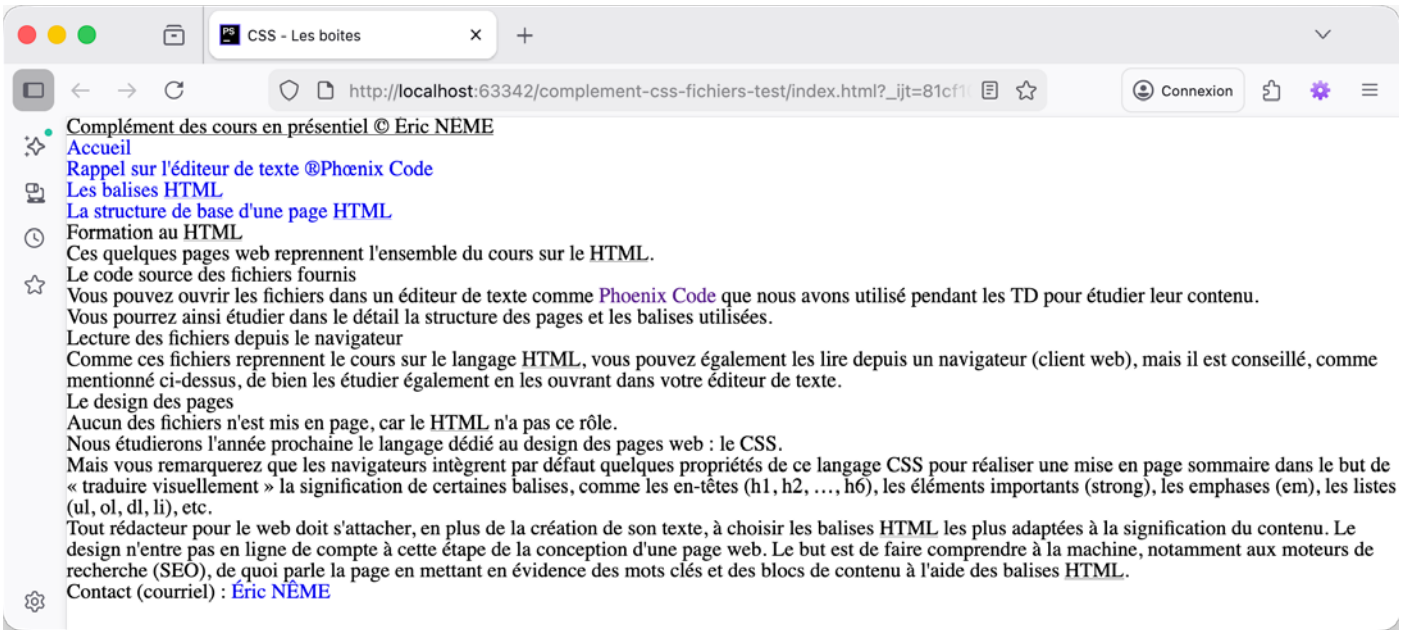
Vous pouvez créer vous-même votre fichier reset, mais on peut en trouver sur le web comme celui-ci : © Eric Meyer - Reset CSS.

```css
/* http://meyerweb.com/eric/tools/css/reset/
   v2.0 | 20110126
   License: none (public domain)
*/

html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
   margin: 0;
   padding: 0;
   border: 0;
   font-size: 100%;
   font: inherit;
   vertical-align: baseline;
}
/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
   display: block;
}
body {
   line-height: 1;
}
ol, ul {
   list-style: none;
}
blockquote, q {
   quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
   content: '';
   content: none;
}
table {
   border-collapse: collapse;
   border-spacing: 0;
}
```

Vous pouvez étudier dans le détail le contenu de ce fichier proposé par Eric Meyer et vous référer à la documentation Mdn si vous voulez des informations complémentaires sur des propriétés que vous ne comprenez pas.
J'attire surtout votre attention en ce qui concerne le modèle de boite : l'auteur a listé un grand nombre de boites (cf. premier sélecteur) et a mis à 0 les propriétés `margin`, `padding` et `border`.

Voici le résultat d'affichage de notre fichier HTML dans Firefox après un tel traitement :



Il n'y a plus d'espace d'entre les boites, nous pouvons maintenant travailler sur notre fichier style.css pour faire notre mise en page sans que les styles par défaut des navigateurs interfèrent.

Remarque : on constate également à l'affichage que le fichier d'Éric Meyer a réduit la taille de la police des titres et leur graisse. Les propriétés `font` et `font-size` et leurs valeurs qu'il a utilisées sont responsables de ce résultat (cf. Mdn - font).

Il existe d'autres possibilités de reset que vous pouvez rencontrer dans vos recherches sur le web. Les webdesigners qui les proposent ont leurs propres habitudes de mise en page et profitent de ce fichier pour mettre en place des propriétés adaptées à leur propre modèle de mise en page. C'est le cas de la grille CSS ® Bootstrap que je vous ai présentée en cours. Parmi les fichiers CSS de cette grille que l'on incorpore à nos fichiers HTML, il y a le fichier `bootstrap-reboot.css` qui est un reset.

Le reset d'Éric Meyer a un atout majeur : il se contente du strict minimum, mais il reste suffisant pour partir sur une très bonne base.

7 - Les unités de mesure

Les unités de mesure proposées par le CSS sont nombreuses et permettent de contrôler au mieux les tailles des polices de caractère, les dimensions des boites, des marges, des bordures, les positionnements, les espaces, les déplacements, les agrandissements, les rotations, les animations, etc. Autant dire qu'il est important de bien comprendre le rôle et le fonctionnement de ces unités de mesure pour maîtriser la mise en page d'une page web.

Vous pouvez vous référer à la documentation Mdn - Valeurs et unités et plus particulièrement au chapitre « Nombres, longueurs et pourcentages » de cette page.

Cette autre page de la documentation Mdn liste toutes les unités disponibles : Mdn - Listes valeurs et unités.

Les unités de mesure du CSS peuvent être classées en deux catégories :

- Les unités absolues comme :
  - `px` (pixel) qui est l'unité matérielle de base d'un écran. La résolution d'un écran correspond au nombre de pixels qui le compose. Pour une même surface, plus il y a de pixels, plus la résolution est importante et l'affichage précis. Autrement dit, un écran qui a une haute résolution contient plus de pixels qu'un écran de basse résolution et la taille des pixels est plus petite que de ceux qui composent un écran de plus faible résolution. Donc, quand on utilise cette unité, par exemple `400px`, on désigne un nombre de pixels matériels (400) à utiliser et donc une longueur précise même si elle dépend de la grosseur du pixel.
  - `mm`, `cm`, `pt` (point) qui sont des unités de longueur que l'on utilise en imprimerie. On peut faire appel à ces unités si l'on souhaite imprimer la page web.
    Je vous rappelle que l'on peut personnaliser la mise en forme d'une page web en vue de son impression en utilisant une feuille de style dédiée qui ne sera active que dans le contexte de l'impression (cf. §1.2.3 et l'attribut `media`).
    Ces unités correspondent également à des grandeurs physiques précises, le mètre est représenté par un étalon universel, et ses sous-unités aussi...
  - `deg` (degré), `rad` (radian), `grad` (gradian) : unité d'angle.
  - `turn` (nombre de tours) : unité de rotation. `1turn` = 1 tour, `0.25turn` = 1/4 de tour, `2turn` = 2 tours, etc.
- Les unités relatives comme :
  - `em` : unité relative à la taille de la police de la boite parente.
  - `rem` : unité relative à la taille de la police de la boite racine (boite `<html>` ou taille de la police par défaut du navigateur)
  - `%` (pourcentage) : unité relative aux propriétés du modèle de boite CSS de la boite parente (`width`, `height`, `margin`, `padding`).
    On peut aussi utiliser cette unité pour les tailles de police (propriété `font-size`) bien qu'il soit préférable d'utiliser les unités `em` et `rem` plus spécifiques aux polices de caractère.

Je n'ai pas listé ci-dessus toutes les unités absolues et relatives possibles en CSS, vous pouvez en découvrir d'autres en consultant la documentation Mdn. Celles que je vous ai présentées sont les plus utilisées, à ce titre, elles sont incontournables.

7.1 Les unités absolues

Les unités absolues ont l'avantage de la précision, mais rendent difficile l'adaptation dynamique de la mise en page des pages web, qui doit être assez souple pour pouvoir s'ajuster à des dimensions de fenêtre de navigateur extrêmement variables, notamment à cause de la profusion des tailles d'écran.

Malgré cette rigidité, ces unités absolues sont très utilisées pour des objets dont on veut stabiliser la taille et pour la mise en page en vue de l'impression.

Pour l'affichage écran et dans la pratique, on constate que les webdesigners font appel à ces unités absolues pour fixer les épaisseurs des bordures des boites et pour les espacements fins qui ne doivent pas varier. Ce n'est pas une règle absolue, mais ceci montre que nous allons plutôt privilégier l'utilisation des unités relatives plus adaptées à une mise en page dynamique pour gérer le texte, le modèle de boite CSS, le placement des boites, etc.

7.2 Les unités relatives

Les unités relatives nécessitent quelques explications complémentaires pour comprendre leur fonctionnement. Si l'on qualifie ces unités de relatives, c'est qu'elles sont forcément en relation avec quelque chose d'autre...

À l'exception de l'unité relative `rem` que nous détaillerons plus loin, la règle est toujours la même : pour une boite donnée, si une propriété a une valeur dont l'unité est relative, c'est la même propriété du parent qui devient la référence pour calculer la grandeur de cette valeur. Cette règle s'applique que la propriété du parent soit explicite ou implicite.

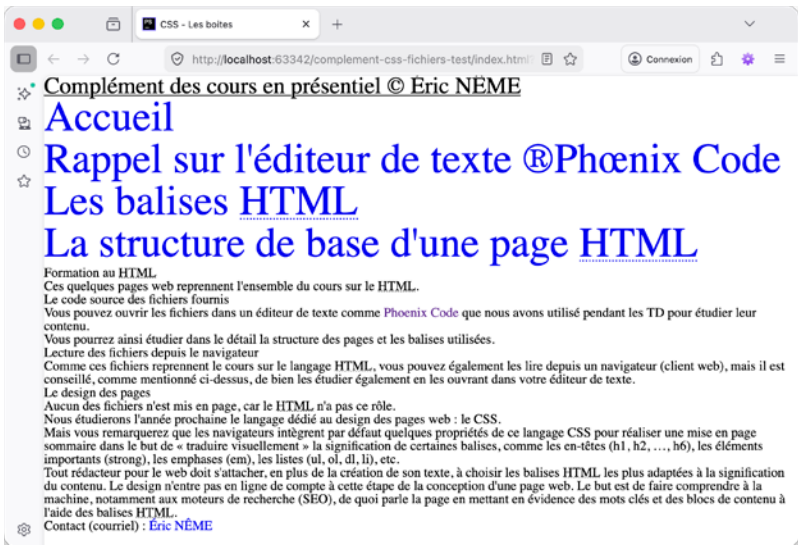
7.2.1 L'unité relative `em`

Nous allons nous appuyer sur un exemple à partir notre fichier HTML pour détailler le plus précisément possible le principe que nous venons d'énoncer avec l'unité relative `em` :

```
header {
   font-size : 28px;
}

nav {
   font-size : 1.8em;
}
```

- Nous faisons appel à la propriété `font-size` (taille de la police de caractère) qui est une propriété héritée. Donc, tous les descendants des boites `<header>` et `<nav>` qui contiennent du texte sont impactés.
- Commençons par la boite `<header>` : la propriété `font-size` est explicite, et tous les descendants qui contiennent du texte ont une taille de `28px`. Dans notre cas, cela concerne le texte des boites `<u>`, `<a>`, `<small>` et `<abbr>`.
- Passons à la boite `<nav>` qui est un enfant de `<header>`. La propriété `font-size` est redéfinie avec une valeur dont l'unité est relative (`1.8em`), donc la taille de la police dépend de celle du parent `<header>` (`28px`). La taille du texte sera calculée comme suit : `1.8em` x `28px` = `50px`. La boite `<nav>` ne contient aucun texte, mais, comme la propriété `font-size` est héritée, tous les descendants de `<nav>` qui contiennent du texte l'affiche maintenant en `50px`.
- Le résultat est le suivant : la boite `<u>` a un texte de `28px` et les textes des boites `<a>` `<small>` `<abbr>` ont une taille de `50px`. Le reste du texte de la page qui n'est dans une boite descendante de `<header>` ne change pas de taille. La preuve à l'écran :



Il faut bien comprendre que l'unité relative `em` utilisée ici rend dépendante la taille de la police de caractère de la boite `<nav>` et de ses descendants de celle de la sa boite parente `<header>`. Ainsi, si on change la valeur de la propriété `font-size` de `<header>`, par exemple `20px`, les textes descendants de `<nav>` seront aussi impactés : `1.8em` x `20px` = `36px`.

Que se passera-t-il si nous supprimons la propriété `font-size` attribuée à `<header>` ?

```
header {
   font-size : 28px;
}

nav {
   font-size : 1.8em;
}
```

La règle de l'unité relative `em` ne change pas, et la taille du texte des descendants de `<nav>` est toujours calculée en fonction de la taille du texte de son parent `<header>`. Il nous faut donc toujours connaître la taille du texte de la boite`<header>`, c'est-à-dire sa propriété `font-size`. Mais nous l'avons supprimée !... Sommes-nous dans une impasse ?

Pas du tout, elle est maintenant implicite. Il ne faut pas oublier la règle de l'héritage de la propriété `font-size`. Donc, cette propriété de la boite `<header>` hérite de celle de son parent `<body>`. Mais connait-on la valeur de la propriété `font-size` de la boite `<body>` ? Toujours pas, nous n'avons pas attribué explicitement cette propriété à cette boite.

Continuons de suivre la règle de l'héritage, et passons à son parent `<html>` auquel nous n'avons pas attribué non plus de propriété `font-size`. Cette fois-ci, nous ne pouvons pas remonter plus haut, `<html>` n'a pas de parent. À ce stade, nous sommes vraiment dans une impasse...

Mais non, les concepteurs du CSS ont pensé à tout... Il reste un endroit où l'on peut trouver la taille de la police `<html>` : les préférences des navigateurs auxquelles chaque internaute a accès. Par défaut, lorsque vous installez un navigateur web sur votre ordinateur, la police est du `Times` et la taille d'affichage est de `16px`. Vous pouvez changer ces valeurs, c'est d'ailleurs une possibilité donnée aux personnes souffrant de déficiences visuelles pour agrandir le texte.

Dans notre cas, si les préférences du navigateur n'ont pas été modifiées, la taille du texte des descendants de la boite `<nav>` sera donc de : `1.8em` x `16px` = `29px`. Dès lors, tout le texte de la page s'affiche en `16px` sauf les descendants de la boite `<nav>` qui sont en `29px`. Et voici la résultat d'affichage :



L'étude détaillée de cet exemple nous a permis de comprendre le mécanisme en jeu lorsque l'on utilise l'unité relative `em` que l'on pourrait résumer ainsi :

- Cette unité fait toujours référence à la propriété font-size (taille de la police de caractère), raison pour laquelle ont l'utilise pour gérer les tailles des polices, bien que l'on puisse l'utiliser pour d'autres propriétés.
- Cette unité fait dépendre la propriété qui l'utilise de la propriété font-size de son parent, que cette dernière soit explicite ou implicite.

7.2.2 L'unité relative `rem`

Cette unité, également très utilisée par les webdesigners, fait aussi référence à la propriété `font-size`, mais cette fois-ci, elle ne se rapporte plus à la propriété `font-size` de la boite parente, mais à celle de la boite racine du fichier HTML (`rem` siginifie root em).

La boite racine d'un fichier HTML est celle qui contient toutes les autres boites (c'est en cela qu'elle est « racine »). Par conséquent, l'unité `rem` fera toujours référence à la propriété `font-size` de la boite `<html>`.

Pour maîtriser les grandeurs faisant appel à l'unité `rem`, il est préférable de fixer explicitement la valeur de la propriété font-size de la boite `<html>` en écrivant, par exemple, au début de la feuille de style :

```
html {
   font-size : 20px;
}
```

Dans ce cas, toutes les grandeurs des propriétés qui utilisent l'unité `rem` seront calculées comme suit : `valeur en rem` x `20px`.

Reprenons le premier exemple du § 7.2.1 en utilisant l'unité `rem`.

```
html {
   font-size : 20px;
}

header {
   font-size : 28px;
}

nav {
   font-size : 1.8rem;
}
```

Cette fois-ci, les textes de toutes les boites descendantes de la boite `<nav>` sont en `36px` (`1.8rem` x `20px` = `36px`). Si nous avions laissé l'unité `em` la valeur aurait été `50px` (`1.8em` x `28px` = `50px`) puisque `em` fait référence à la propriété `font-size` du parent `<header>`.

Le grand avantage de l'unité `rem` est qu'elle se réfère toujours à une même valeur, celle de la propriété `font-size` de la boite `<html>`. Dans notre exemple, toutes les propriétés dont la valeur utilise l'unité `rem` ont une grandeur proportionnelle à `20px`.

Que se passe-t-il si nous n'attribuons pas explicitement une valeur à la propriété `font-size` de la

boite `<html>` ? Comme expliqué précédemment, par le mécanisme d'héritage, c'est la taille de la police précisée dans les préférences du navigateur qui donnera cette valeur à la propriété `font-size` de la boite `<html>`. Dès lors, la taille de la police des préférences du navigateur (`16px` par défaut) est celle à laquelle se réfèrent toutes les propriétés dont la valeur utilise l'unité `rem`.

Il est important de remarquer que, dans ce cas, une personne souffrant de déficience visuelle, susceptible de changer cette valeur, pourra bénéficier de l'avantage d'une mise en page plus adaptée à sa situation. Par exemple, si la taille de la police de vos titres est fixée à `1.5 rem`, leur taille réelle d'affichage en pixel sera calculée en prenant en compte la valeur de la taille de police que cette personne aura indiquée dans les préférences de son navigateur. Ainsi, s'il paramètre la taille à `30px`, vos titres s'afficheront en `1.5rem` x `30px`, soit `45px`, pour ceux qui ne modifient pas ces paramètres, la taille des titres sera de `1.5rem` x `16px`, c'est-à-dire `24px`.

7.2.3 L'unité relative `%`

Voici une autre unité relative très utilisée par les webdesigners. Elle reprend la règle présentée au début du § 7.2, mais est plus générale, car elle s'applique à toutes les propriétés du modèle de boite, mais aussi à la propriété `font-size`.

Toutefois, pour bien comprendre cette unité relative, il faut préciser :

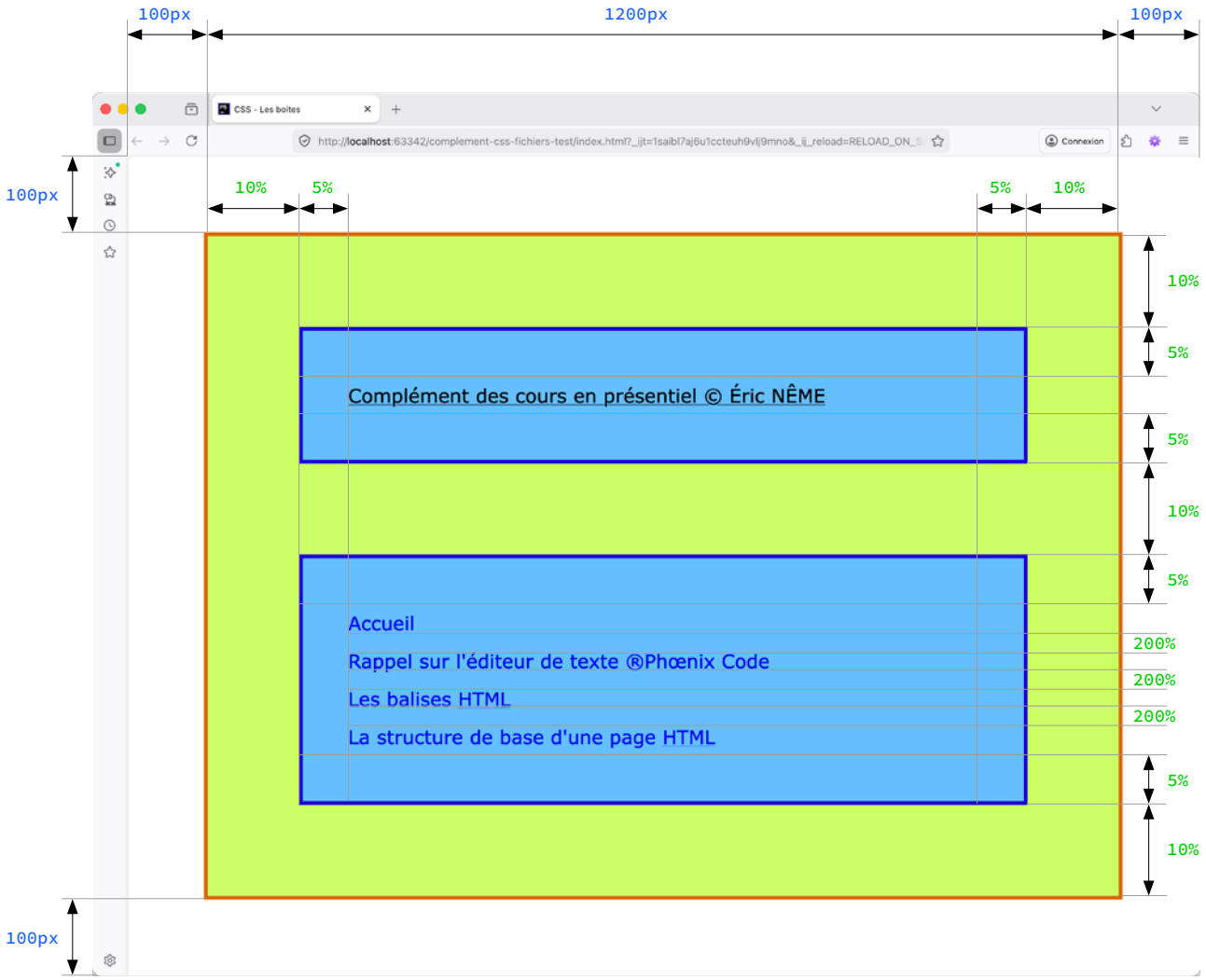
- Si la propriété qui utilise cette unité ne concerne pas le texte, c'est la valeur de la propriété `width` ou `height` de la boite parente qui sera la référence. Dès lors, les dimensions de la zone de contenu de la boite parente sont la référence pour calculer la valeur en pixel.
- Si la propriété qui utilise cette unité concerne le texte, c'est la propriété `font-size` de la boite parente qui sera la référence. Dès lors, la taille de la police de la boite parente est la référence pour calculer la valeur en pixel.

Prenons un exemple pour comprendre cette unité :

```
header {
    width : 1200px;
    margin : 100px;
    border : 5px solid #d76700;
    background-color : #ccff66;
    font-family : verdana, helvetica, arial, sans-serif;
    font-size : 25px;
}

header p, header nav {
    margin : 10%;
    padding : 5%;
    line-height : 200%;
    border : 5px solid #1900d7;
    background-color : #66bfff;
}
```

Pour vous exercer, étudiez attentivement les sélecteurs, les propriétés, les valeurs et les unités utilisées et essayez, avant de regarder la page suivante, de vous représenter le résultat d'affichage dans le navigateur.




```
En bleu : valeurs des propriétés attribuées à la boite <header>.
En vert : valeurs des propriétés attribuées aux boites <p> et <nav>, enfants de la boite <header>.
```

Voici l'explication du résultat d'affichage ci-dessus des éléments impactés par l'utilisation de l'unité relative `%` :

- Une seule propriété concerne le texte, `line-height` (interlignage du texte) : `line-height` : `200%`. La référence est donc la propriété `font-size` de son parent `<header>`. La valeur calculée est de `25px` x `200%` = `50px`. L'espace entre les lignes des textes des descendants de `<p>` et `<nav>` est donc de `50px`.
- Toutes les autres propriétés dont la valeur fait appel à l'unité `%` ne concernent pas le texte :
  - `margin` : `10%`. La référence est donc la propriété `width` de leur parent `<header>`. La valeur calculée est de `1200px` x `10%` = `120px`. La marge externe des boites `<p>` et `<nav>` est donc de `120px`.
  - `padding` : `5%` : la référence est donc la propriété `width` de leur parent `<header>`. La valeur calculée est de `1200px` x `5%` = `60px`. La marge interne des boites `<p>` et `<nav>` est donc de `60px`.

Il existe des subtilités, que je ne détaillerais pas ici, concernant l'unité `%` lorsqu'elle se réfère à la propriété `height` de la boite parente. En pratique, on fait rarement appel à cette propriété pour laisser la boite adapter sa hauteur en fonction du volume de son contenu. Les concepteurs du CSS n'ont pas déterminé de manière précise les règles de l'unité `%` pour calculer les valeurs des dimensions faisant référence à la propriété `height` du parent, ce qui peut déboucher sur des résultats d'affichage inattendus.

8 - Le placement des boites

Le modèle de boite CSS que nous avons étudié dans le détail permet d'intervenir sur les dimensions des boites, des bordures et des marges. Il nous reste à présenter les propriétés qui permettent de disposer les boites dans la page, c'est-à-dire de les placer les unes par rapport aux autres.

Rappelons le flux naturel de l'affichage des boites :

- Pour les boites de type bloc, elles s'affichent les unes en dessous des autres en fonction de leur ordre d'apparition dans le fichier HTML. De plus, elles occupent la largeur de la zone de contenu de leur parent.
- Pour les boites de type en ligne, elles s'affichent les unes à côté des autres.

Si nous souhaitons contrarier ce comportant naturel et disposer nos boites selon nos souhaits pour mettre en forme nos pages web, le langage CSS met à notre disposition les propriétés suivantes :

- La propriété `display` : Mdn - Display.
- La propriété `float` (flottement) : Mdn - Float.
- La propriété `position` (positionnement) : Mdn - Position.

Chacune de ces propriétés est accompagnée d'un ensemble d'autres propriétés qui leur sont dépendantes :

- La propriété `display` propose de très nombreuses valeurs très puissantes. Par exemple, la valeur `none` permet de ne pas afficher une boite (essayez `body { display : none; }` et tout le contenu de votre page disparaît comme par magie...).
  Parmi ces valeurs, deux sont apparues il y a une dizaine d'années. Elles ont permis des évolutions majeures dans la maîtrise de la disposition des boites :
  - La valeur `flex` (on parle de modèle de boite flexible ou flexbox) : Mdn - flexbox.
    De manière générale, si une boite a la propriété `display : flex`, tous ses enfants peuvent être manipulés et disposés selon les règles et propriétés propres à ce modèle.
  - La valeur `grid` (on parle de modèle de grille ou grid) : Mdn - Grid.
    Si une boite a la propriété `display : grid`, tous ses enfants peuvent être manipulés et disposés selon les règles et propriétés propres à ce modèle.

  Je ne présenterai pas ces deux modèles dans ce document, mais vous trouverez dans la webographie fournie plus bas, des tutoriels vidéo et des sites qui les détaillent.

- La propriété `float`, que nous étudierons dans le détail, a été très utilisée avant l'apparition des modèles grid et flexbox pour manipuler la disposition des boites. Les webdesigners l'utilisent maintenant principalement pour habiller les images par le texte, mais elle peut toujours être utilisée pour n'importe quelle boite.
  La propriété `clear` dépend de cette propriété.
- La propriété `position`, dont nous détaillerons le comportement, permet de placer précisément les boites dans la page en faisant abstraction de tout ou partie du contexte d'affichage, c'est-à-dire des autres boites environnantes.
  Les propriétés, `top`, `left`, `bottom`, `right` et `z-index` sont dépendantes de cette propriété.

Pour résumé, les propriétés `float` et `position` que nous allons détaillées étaient les propriétés majeures pour positionner les boites dans une page web avant l'apparition des modèles grid et flexbox. Aujourd'hui, on utilise la propriété `float` pour habiller les boites <img> par le texte et la propriété `position` pour placer les boites dans une page en la soustrayant totalement ou partiellement de l'influence des autres boites.

Nous allons utiliser notre fichier HTML et les styles suivants pour préparer la présentation des propriétés `display`, `float` et `position` :

style.css

```css
* {
    box-sizing: border-box;
}

html {
    font-size : 20px;
}

body {
    width : 1200px;
    margin : 50px auto;
    font-family: tahoma, verdana, arial, sans-serif;
    font-size: 1rem;
    line-height: 1.5rem;
}

header, main, footer {
    margin : 50px 0;
    border: 1px dotted #00bbff;
}

main h1 {
    margin: 40px 0;
    font-size : 3rem;
    color: #2a6ea5;
}

main h2 {
    margin: 30px 0;
    font-size : 2rem;
    color: #2a9ba5;
}
```

Vous devriez être en mesure de comprendre toutes les propriétés mises en place. Un bon exercice est d'essayer de se représenter le résultat d'affichage du fichier avant de vérifier le résultat en page 34.

Le résultat n'est pas à la hauteur de ce que l'on peut attendre d'un design professionnel, mais le but est de ne pas trop surcharger la mise en page pour vous permettre de vous concentrer sur les explications qui vont suivre.

Dans cet esprit, j'ai rajouté des bordures fines en pointillées pour mieux identifier les trois boites enfants de la boite `<body>` : `<header>`, `<main>` et `<footer>`.

Dernières précisons : pour la suite, nous allons surtout intervenir sur certains enfants et descendants de la boite `<main>`. Nous rajouterons des propriétés à l'aide de sélecteurs qui pointeront ces éléments.

Voici le résultat d'affichage, l'aviez-vous prévu ? Attardez-vous sur toutes les propriétés du modèle de boite et, si vous notez des incohérences, reprenez les pages précédentes, vous trouverez certainement les réponses à vos questions. Si vraiment vous n'y parvenez pas, je suis disponible par email.



8.1 - La propriété `display`

Nous allons présenter certaines valeurs de cette propriété, qui, comme nous l'avons dit plus, sont nombreuses et puissantes et dignes d'intérêt.

- La valeur `none` : une boite concernée par `display : none` disparait à l'affichage et pour les autres boites qui occuperont sa place laissée vacante. Pour la faire réaparaître, il suffit de donner à la

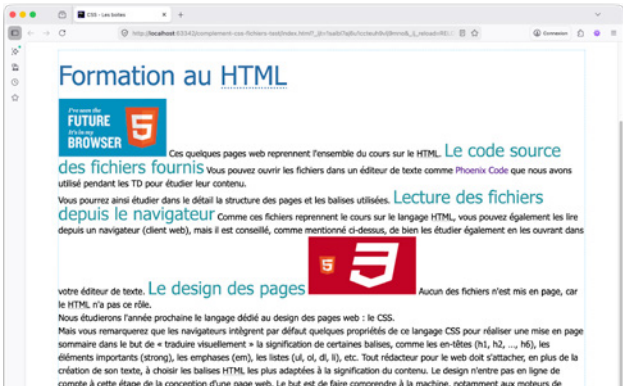
boite une des deux valeurs présentées ci-après.
Pour information, il existe une autre propriété (`visibility`) pour masquer une boite uniquement à l'affichage. L'espace que la boite occupe, même si son contenu ne s'affiche plus dans la page, ne peut pas être empli par les autres boites. Les deux valeurs de cette propriété sont les suivantes : `hidden` pour masquer et `visible` pour afficher.
Ces propriétés sont souvent associées aux évènements de survol ou de clic (ex. les pseudo-classes `:hover`, `:active`, etc. , cf. Mdn - Pseudo-classes ou contrôlés en JavaScript).

› La valeur `inline` : toute boite de type bloc concernée par `display : inline` devient une boite de type inline ! La boite ne provoquera plus de retour à la ligne.

Voici un exemple :

```
h2, p{
    display : inline;
}
```

Ci-contre, le résultat d'affichage : les sous-titres et les paragraphes ne provoquent plus de retour à la ligne.

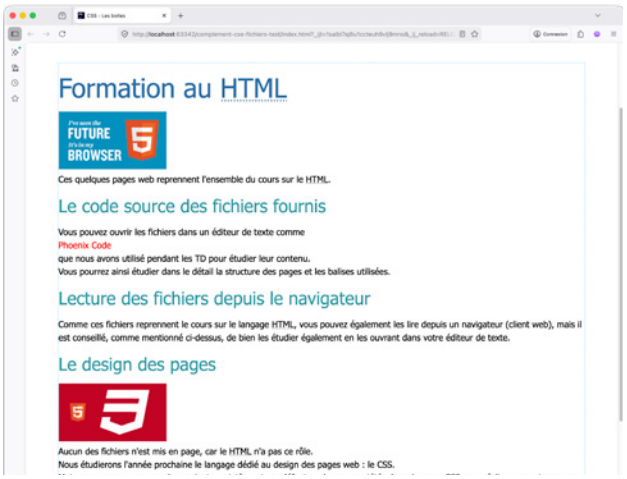


› La valeur `block` : toute boite de type en ligne concernée par `display : block` devient une boite de type bloc ! La boite provoquera un retour à la ligne.

Voici un exemple (pour mieux voir les liens ils sont colorisés en rouge) :

```
a{
  display : block;
  color : red;
}
```

Ci-contre, le résultat d'affichage : le lien `<a href="https://phcode.dev/download/">Phoenix Code</a>` provoque des retours à la ligne.

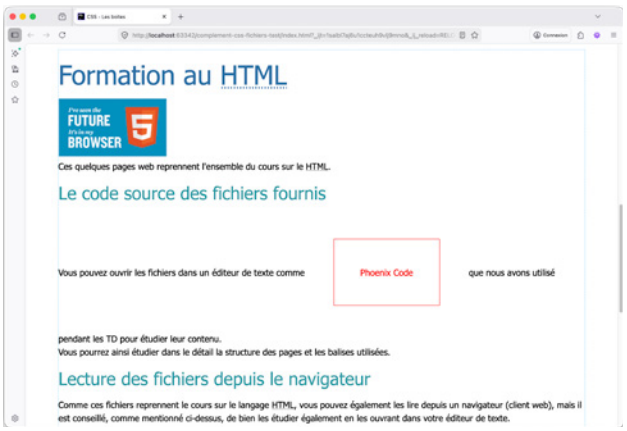


› La valeur `inline-block` : toute boite concernée par `display : inline-block` devient une boite de type mixte ! La boite ne provoque pas de retour à la ligne (`inline`) mais on peut utiliser toutes les propriétés du modèle de boite CSS pour la mettre en forme (`block`).

Voici un exemple :

```
a{
    display : inline-block;
    margin : 5%;
    padding : 5%;
    border : solid 1px red;
    color : red;
}
```

Ci-contre, le résultat d'affichage : il n'y a pas de retour à la ligne, mais la boite <a> provoque un espacement important entre les lignes.



8.2 - La propriété `float`

La propriété `float` (flottement) a trois valeurs principales :

- `left` : la boite est flottante à gauche.
- `right` : la boite est flottante à droite.
- `none` : la boite ne doit pas flotter.

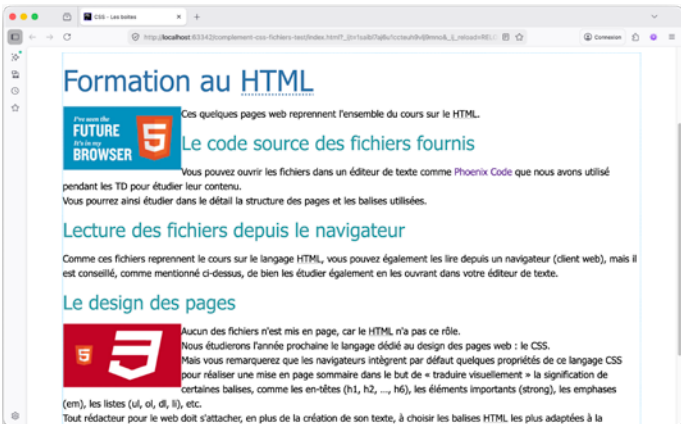
Avant de précisier ces valeurs il faut mentionner plusieurs choses pour comprendre le mécanisme du flottement :

- Lorsqu'une boite est flottante, elle signale aux autres boites qu'elle n'existe plus bien qu'elle soit toujours visible à l'affichage ! Si la boite n'existe plus, alors elle laisse la place qu'elle occupe vacante et les autres boites se précipitent pour l'occuper... En effet, une boite, quelle qu'elle soit, cherche toujours à se positionner le plus haut et le plus à gauche possible dans la page. Elle ne sera bloquée dans sa tentative que par une boite (ou du texte pour les boites de type en ligne) située au-dessus d'elle ou à sa gauche. Pour information, concernant les langues s'écrivant de la droite vers la gauche (rtl, right to left), les boites se placent le plus à droite possible.
- La boite flottante n'existe plus pour les autres boites, sauf pour le texte ?!? Ce comportement est plutôt étonnant, et pourtant, les auteurs du CSS l'ont volontairement mis en place pour permettre le mécanisme d'habillage d'une boite par du texte. L'habillage consiste à « faire tourner le texte » autour de la boite. On l'utilise souvent pour les images. Pour que la boite soit habillée par du texte, il faut bien qu'elle soit toujours considérée comme présente pour lui.
- Le texte des autres boites « tourne autour » de la boite flottante. Sa dimension comprend toutes les propriétés du modèle de boite CSS, ainsi, les textes ne pourront pas empiéter sur la zone des marges externes de la boite flottante.
- La valeur `left` indique que la boite flottante doit se placer le plus à gauche possible, en restant dans la zone de contenu de sa boite parente, et que le texte des autres boites se situera à sa droite.
- La valeur `right` indique que la boite flottante doit se placer le plus à droite possible, en restant dans la zone de contenu de sa boite parente, et que le texte des autres boites se situera à sa gauche.

Passons à la pratique en ajoutant les propriété `float : left` à nos images :

```
main img {
    float : left;
}
```

Les boites situées en dessous de nos boites images ont été remontées pour occuper leur place, sauf les textes qui tournent autour d'elles sur leur droite.



Comme le texte ne peut pas empiéter sur les marges externes de la boite flottante, nous allons rajouter une petite marge externe pour le décaler légèrement :

```
main img {
    float : left;
    margin : 7px 15px 0 0;
}
```

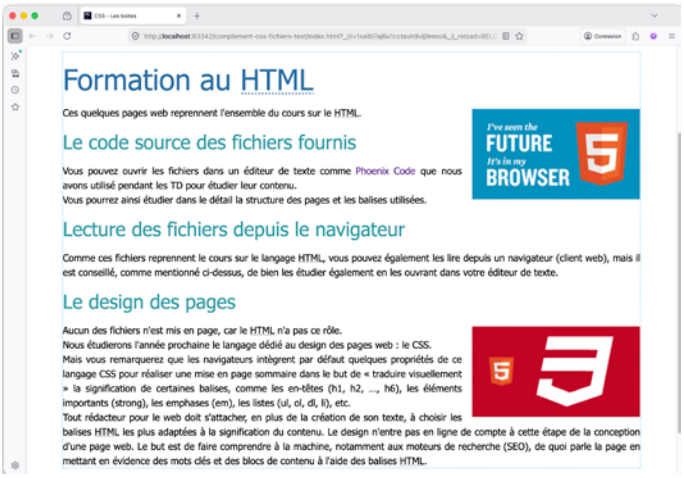


Utilisons maintenant la propriété `float : right` toujours pour nos images :

Pour mieux visualiser les effets, nous allons justifier les textes des paragraphes (boites `<p>`) et agrandir légèrement les images. On en profite pour rajouter une petite marge pour que les textes se décalent légèrement de les bords des images.

```
main p {
    text-align : justify;
}

main img {
    float : right;
    width : 350px;
    margin : 7px 0 0 20px;
}
```



Le mécanisme est le même avec les images placées à droite.

Pour les boites de la page qui ne sont pas flottantes, nous pouvons contrôler leur comportement par rapport aux boites flottantes à l'aide de la propriété `clear` qui annule l'effet de flottement. Ainsi :

- Une boite concernée par `clear : right` annule l'effet des boites flottantes à droite.
- Une boite concernée par `clear : left` annule l'effet des boites flottantes à gauche.
- Une boite concernée par `clear : both` annule l'effet des boites flottantes à gauche ou à droite.

Passons à la pratique : nous allons annuler les effets de flottement des images pour deux boites `<p>` particulières de notre fichier HTML (cf. p.6, lignes 32 et 43). Pour les désigner plus facilement, il aurait été plus judicieux d'ajouter une classe à ces deux paragraphes (ex. `<p class="no-float">...</p>`) ce qui aurait permis d'utiliser le sélecteur de classe `.no-float {…}` dans notre feuille de style. Nous allons utiliser une autre méthode, moins souple, mais qui fonctionne :

```
main p:nth-child(5), main p:nth-child(11) {…}
```

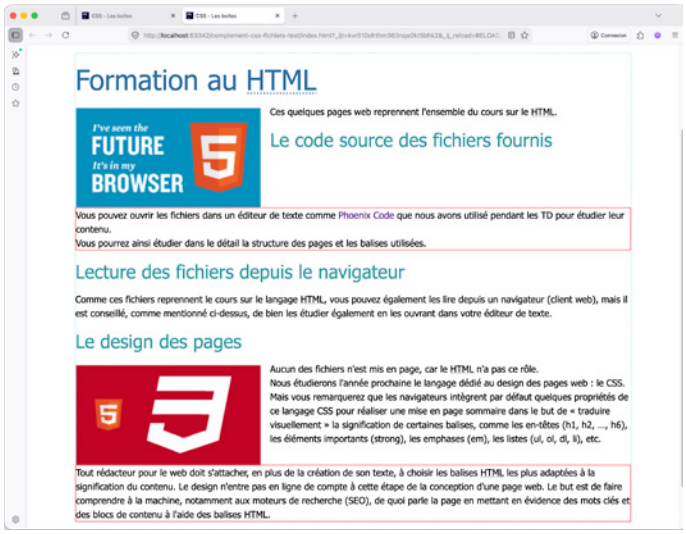
Ce sélecteur a la signification suivante : le 5e enfant et le 11e enfant de la boite `<main>` qui sont des boites `<p>`.
Vous trouverez le détail de la pseudo classe :nth-child très utilisée ici : Mdn - :nth-child.

Nous allons également leur rajouter une fine bordure rouge pour mieux les repérer à l'affichage.

› Cas des images flottantes à gauche avec la propriété `clear : left`:

```
main img {
    width : 400px;
    margin : 7px 20px 0 0;
    float : left;
}

main p:nth-child(5),
main p:nth-child(11) {
    border : 1px solid red;
    clear : left;
}
```

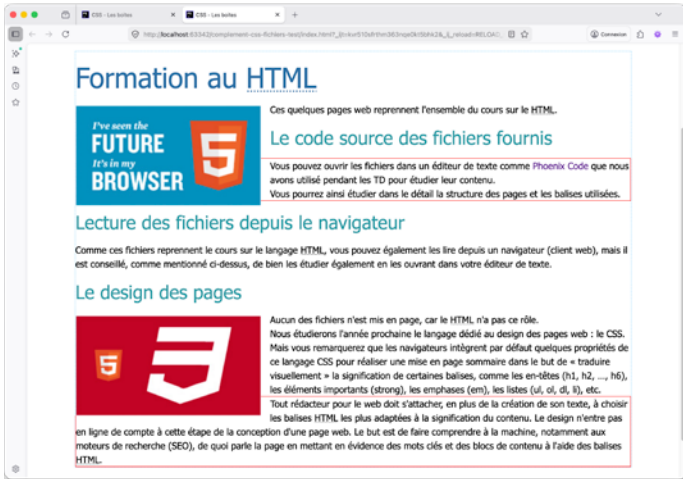


Toutes les boites respectent le flottement des images, sauf les deux boites `<p>` que nous avons sélectionnées qui annulent l'effet du flottement et se placent sous les images.

› Cas des images flottantes à gauche avec la propriété `clear : right`:

```
main img {
    width : 400px;
    margin : 7px 20px 0 0;
    float : left;
}

main p:nth-child(5),
main p:nth-child(11) {
    border : 1px solid red;
    clear : right;
}
```

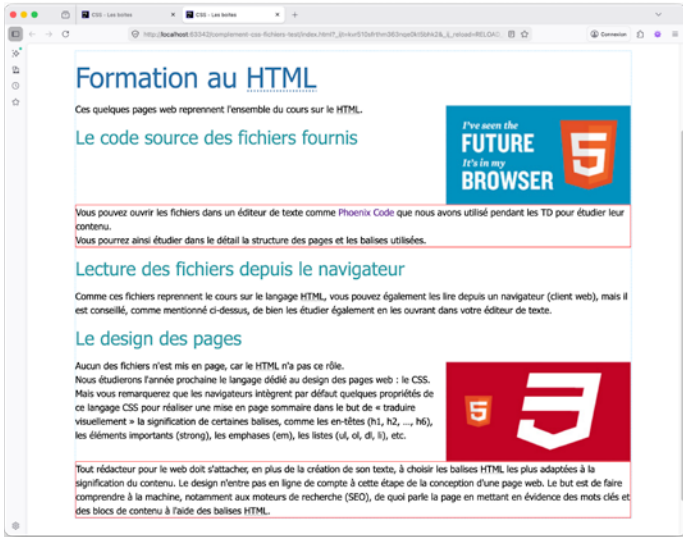


Toutes les boites respectent le flottement des images. Les deux boites `<p>` que nous avons sélectionnées annulent uniquement l'effet du flottement des boites flottantes à droite.

› Cas des images flottantes à droite avec la propriété `clear : right`:

```
main img {
    width : 400px;
    margin : 7px 0 0 20px;
    float : right;
}

main p:nth-child(5),
main p:nth-child(11) {
    border : 1px solid red;
    clear : right;
}
```

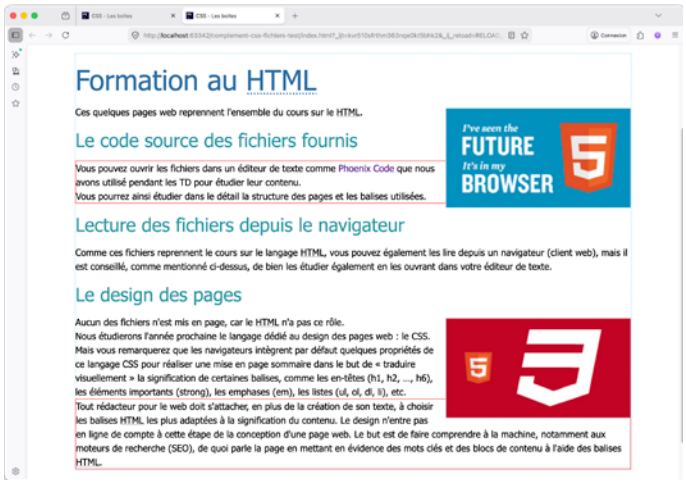


Seules les deux boites `<p>` sélectionnées annulent l'effet du flottement et se placent sous les images.

› Cas des images flottantes à droite avec la propriété `clear : left`:

```
main img {
    width : 400px;
    margin : 7px 0 0 20px;
    float : right;
}

main p:nth-child(5),
main p:nth-child(11) {
    border : 1px solid red;
    clear : left;
}
```

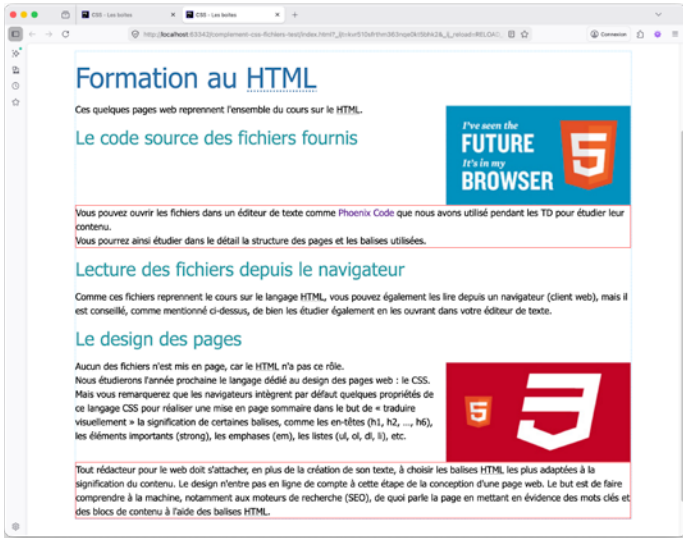


Toutes les boites respectent le flottement des images. Les deux boites `<p>` que nous avons sélectionnées annulent uniquement l'effet du flottement des boites flottantes à gauche.

› Cas des images flottantes à droite avec la propriété `clear : both`:

```
main img {
    width : 400px;
    margin : 7px 0 0 20px;
    float : right;
}

main p:nth-child(5),
main p:nth-child(11) {
    border : 1px solid red;
    clear : both;
}
```

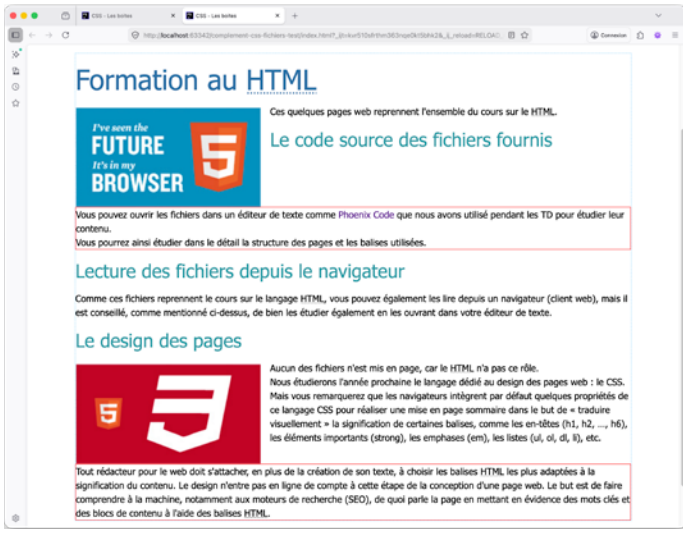


Seules les deux boites `<p>` sélectionnées annulent l'effet du flottement des boites flottant à droite ou à gauche et se placent sous les images.

› Cas des images flottantes à gauche avec la propriété `clear : both`:

```
main img {
    width : 400px;
    margin : 7px 20px 0 0;
    float : left;
}

main p:nth-child(5),
main p:nth-child(11) {
    border : 1px solid red;
    clear : both;
}
```



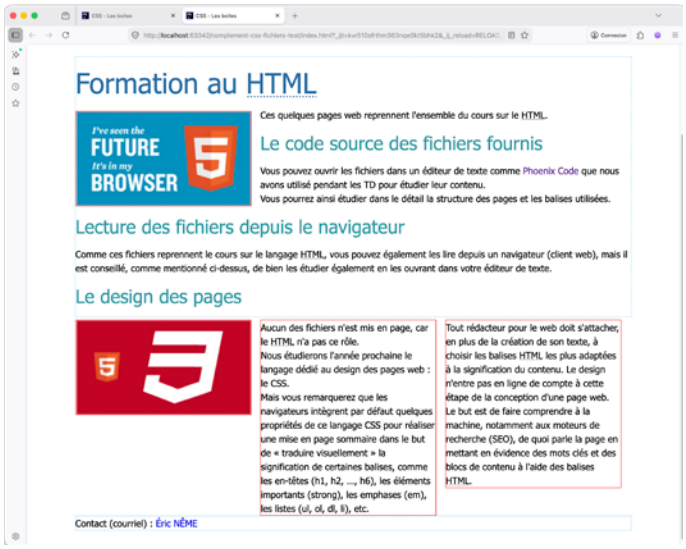
Seules les deux boites `<p>` sélectionnées annulent l'effet du flottement des boites flottant à droite ou à gauche et se placent sous les images.

Nous venons d'utiliser la propriété de flottement pour les boites `<img>`, c'est-à-dire des images. Mais, comme nous l'avons dit plus haut, nous pouvons l'appliquer à n'importe qu'elle boite.

Testons cette possibilité en rendant flottants à gauche une boite `<img>` et deux boites `<p>` (cf. p.6, lignes 39, 40 et 43) :

```
main p:nth-child(10),
main p:nth-child(11),
main img {
    width : 379px;
    margin : 7px 20px 0 0;
    border : 1px solid red;
    float : left;
}

footer {
    clear : both;
}
```



La boite `<img>` et les deux boites `<p>` se placent les unes à côté des autres.
On obtient à l'affichage trois colonnes de `425px` de large.

Ce test démontre que la propriété `float` peut servir à aligner des boîtes les unes à côté des autres. Cependant, comme je l'ai mentionné précédemment, nous ne l'utilisons plus pour atteindre ce type de résultat. On préfère aujourd'hui utiliser les propriétés `display : grid` ou `display : flex` qui sont plus souples et qui offrent plus de possibilités (cf. Webographie). Par conséquent, on fera appel à la propriété `float` principalement pour habiller les images par du texte.

8.3 - La propriété `position`

La propriété `position` donne à la boite à laquelle on l'applique une très grande liberté de placement dans la page web. C'est une liberté qu'il faut savoir maîtriser, car, dans la plupart des cas, une boite positionnée n'existe plus pour les autres boites environnantes, même pour les textes, ce qui peut provoquer des chevauchements malheureux.

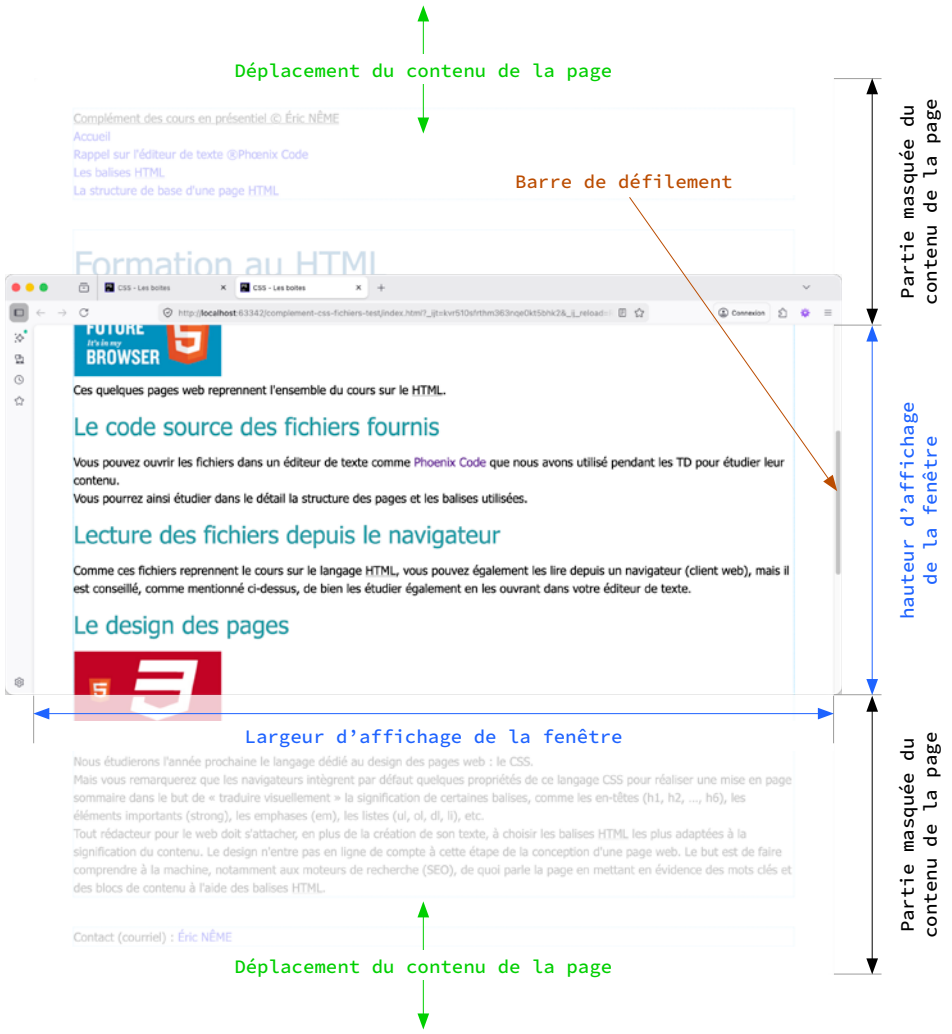
Nous allons étudier dans le détail cette propriété avec toutes ses valeurs et un ensemble d'autres propriétés qui s'y rapportent.

La propriété `position` a cinq valeurs possibles :

- `static` : c'est le positionnement par défaut de toutes les boites.
- `relative` : la boite occupe la position que lui octroie le flux naturel d'affichage des boites, mais on peut la déplacer par rapport à cette position.
- `absolute` : la boite est totalement retirée du flux naturel. Son positionnement se fait par rapport à l'ancêtre le plus proche qui a une propriété position autre que static. Si aucun de ses ancêtres ne possède cette propriété, c'est la fenêtre du navigateur qui devient la référence de placement.
- `fixed` : la boite est totalement retirée du flux naturel. Son positionnement se fait par rapport à la fenêtre du navigateur qui devient la référence pour son placement. La boite reste fixe et visible même en cas de défilement du contenu de la page.

- `sticky` : la boite occupe la position que lui octroie le flux naturel. Lors du défilement du contenu de page, elle suivra le déplacement sans changer de position par rapport aux autres boites. Mais, lors du défilement de la page, lorsque la boite parente aura atteint et dépassé un bord des quatre côtés de la fenêtre et qu'elle-même l'aura aussi atteint, elle se figera. Elle restera bloquée contre ce bord jusqu'à ce que le côté opposé de la boite parente l'ait aussi dépassé. La boite sera libérée de sa position figée pour retrouver sa place initiale. L'explication est un peu complexe, mais nous aurons l'occasion de préciser ce comportement par des exemples.

Pour les deux valeurs `fixed` et `sticky` il est fait mention du défilement du contenu de la fenêtre. En effet, nous faisons défiler ce contenu à l'aide des barres de défilement du navigateur lorsque qu'il dépasse les dimensions de la zone d'affichage de la fenêtre. Il est important d'avoir une représentation correcte de la situation pour mieux comprendre les comportements de la propriété `position` quand elle a ces deux valeurs.



La zone d'affichage de la fenêtre du navigateur peut devenir une référence pour le placement des boites affectées par la propriété `position`.

Dès qu'une boite reçoit la propriété `position` avec une valeur autre que `static`, de nouvelles propriétés deviennent disponibles pour la déplacer par rapport à sa position initiale ou à celle de la boite ascendante la plus proche ayant la propriété position, ou à la zone d'affichage de la fenêtre du navigateur :

- La propriété `top` qui fixe le placement par rapport au haut.
- La propriété `bottom` qui fixe le placement par rapport au bas.

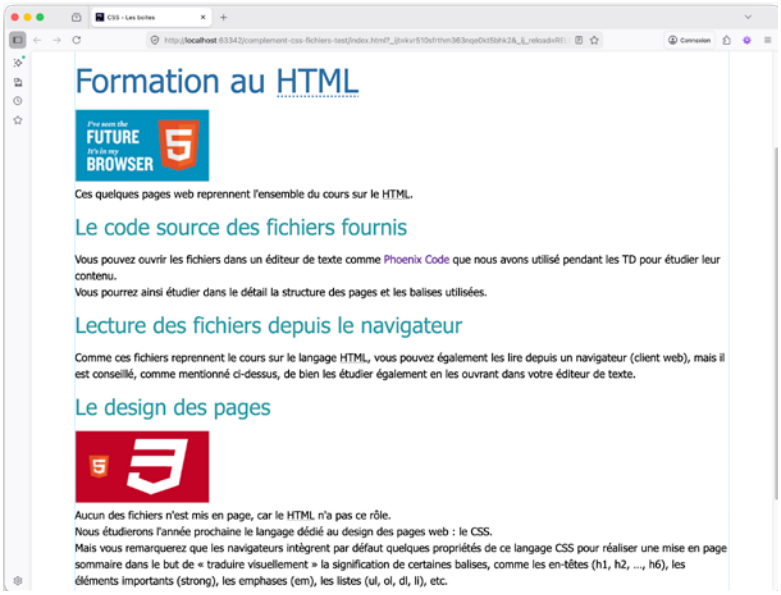
- La propriété `left` qui fixe le placement par rapport à la gauche.
- La propriété `right` qui fixe le placement par rapport à la droite.
- La propriété `z-index` qui permet d'organiser la superpostion des boites les unes par rapport aux autres. En effet, lorsque les boites ont la propriété `position`, elles peuvent se chevaucher. L'ordre d'empilement peut être gérer à l'aide de cette propriété en attibuant un nombre quelconque positif ou négatif. Si une valeur de z-index est supérieur à une autre, la boite passe au-dessus. Par exemple, si on attribue à trois boites positionnées les propriétés `z-index : 10` pour la boite 1 , `z-index : -5` pour la boite 2, `z-index : 200` pour la boite 3, la boite 2 sera en dessous de la boite 1 qui sera en dessous de la boite 3.
  Par défaut, toutes les boites peuvent être consédérée comme ayant un `z-index` à `0`.
  Si des boites ont la même valeur de `z-index`, elles s'empileront selon le flux naturel d'affichage, c'est-à-dire qu'une boite qui apparaît après une autre dans le fichier HTML passera au-dessus.

Passons à la pratique. Nous allons manipuler uniquement les images de notre page pour illustrer le fonctionnement de ces propriétés et valeurs, mais, bien évidemment, nous pouvons les appliquer à n'importe quelles boites.

8.3.1 La valeur `relative` de la propriété `position`

- Testons la propriété `position` avec la valeur `relative` :

```
main img {
    position : relative;
}
```



Malgré la propriété `position : relative`, les deux boites `<img>` ne se déplacent pas !... Elles restent à leur position initiale.

Ce comportement est très important, car il va nous être particulièrement utile pour comprendre le positionnement des boites descendantes qui auront la propriété `position : absolute` que nous détaillerons juste après.

Il faut donc bien retenir que toute boite qui a uniquement la propriété `position : relative` ne change pas sa position initiale ni celle des autres boites qui l'entourent.
À l'affichage, rien de change !

Maintenant que la boite a la propriété `position`, nous pouvons utiliser les propriétés `top`, `bottom`, `left`, `right` et `z-index`. Par exemple :

```
main img {
    position : relative;
    top : 100px;
    left : 150px;
}
```



Les deux propriétés que nous venons d'ajouter permettent de déplacer les boites par rapport à leur position initiale.

Il faut remarquer que les espaces qu'occupaient ces deux boites sont toujours présents. Ces boites, en tant que conteneurs, restent à leur position initiale et c'est le contenu de celles-ci qui se déplace par rapport aux conteneurs. Les autres boites n'ont pas bougé, elles ne sont pas remontées pour occuper l'espace initial des images.

Par contre, on constate que les images passent au-dessus des autres boites et masquent leur contenu. C'est le danger que nous avons signalé plus haut, lorsqu'une boite utilise la propriété `position`, elle peut provoquer des chevauchements. Il est donc important de l'utiliser à bon escient pour éviter ce type de problème.

› Introduisons la propriété `z-index` :

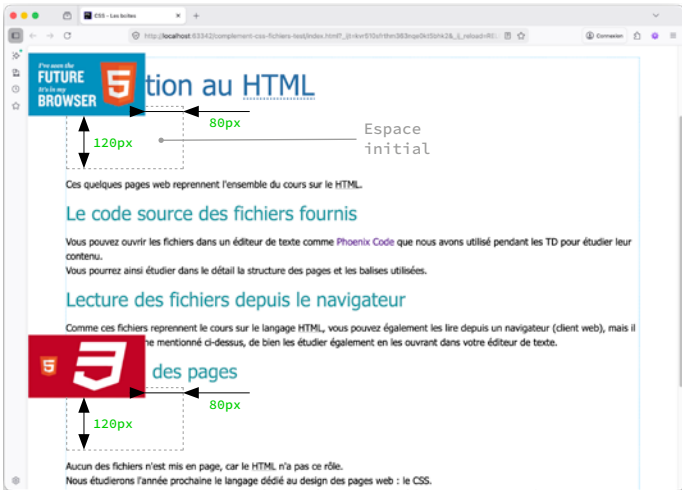
```
main img {
    position : relative;
    left : 150px;
    top : 100px;
    z-index : -10;
}
```



Les deux images sont maintenant au dessous des autres boites et le texte leur passe par-dessus. Notons la valeur `-10` utilisées. Comme nous l'avons vu, on peut considérer que, par défaut, les boites ont un `z-index` à `0`, il suffit donc de donner n'importe quelle valeur inférieure à `0` pour que les images se placent en dessous des boites qui n'ont pas la propriété `position`.

› Étudions les propriétés `right` et `bottom` :

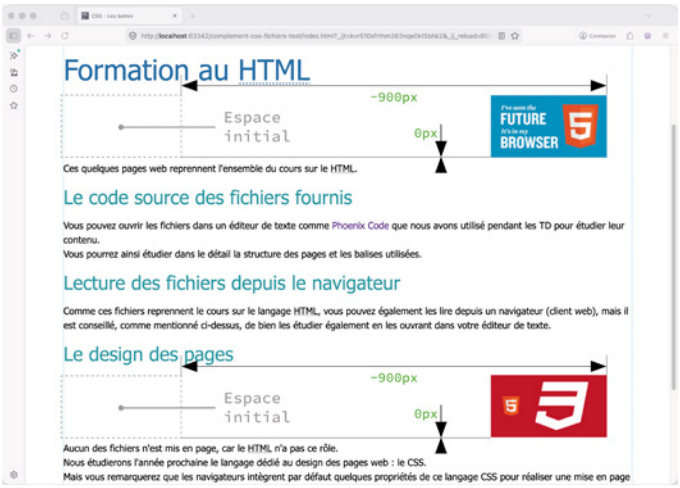
```
main img {
    position : relative;
    right : 80px;
    bottom : 120px;
}
```



Les côtés bas des contenus se déplacent à `120px` au-dessus du bas de leur conteneur.
Les côtés droits des contenus se déplacent à `80px` de la gauche des côtés droits de leur conteneur

› Étudions le cas d'une valeur négative :

```
main img {
    position : relative;
    right : -900px;
    bottom : 0;
}
```



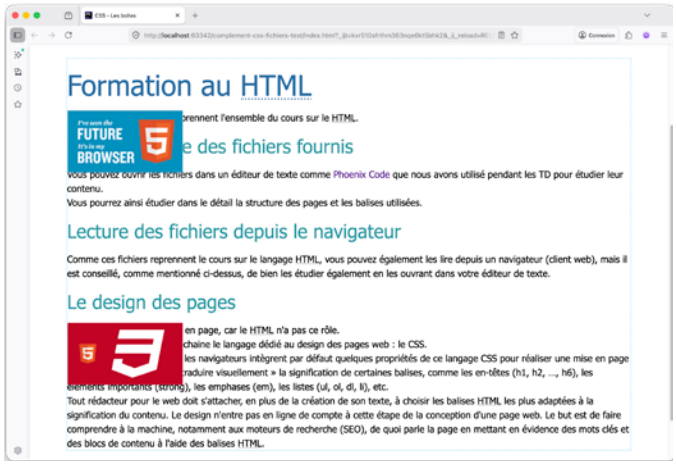
Le bas des contenus des deux boites ne se déplacent pas (`0px`).

Les côtés droits des contenus se déplacent à `900px` de la droite des côtés droits de leur conteneur.

8.3.2 La valeur `absolute` de la propriété `position`

› Testons la propriété `position` avec la valeur `absolute` :

```
main img {
    position : absolute;
}
```

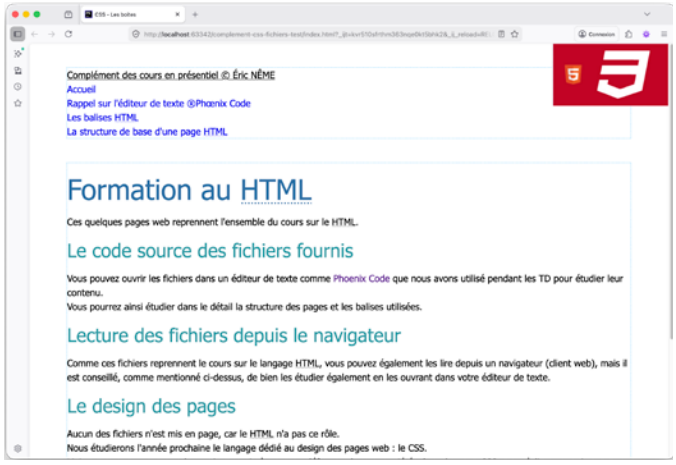


Les deux boites `<img>` ne bougent pas !... Elles restent à leur position initiale, mais, contrairement à la valeur `relative`, les boites sont extraites totalement du flux naturel d'affichage. Les autres boites en profitent pour occuper leur place laissée vacante.

Sur ce dernier point, on retrouve le mécanisme du flottement, à une différence près, et pas des moindres : le texte des autres boites ne tourne plus autour des images, mais passe par-dessous ou par-dessus si on rajoute la propriété `z-index` avec une valeur inférieure à `0`.

› Maintenant que nos boites `<img>` on la propriété `position`, ajoutons, par exemple, les propriétés `top` et `right` pour étudier leur comportement :

```
main img {
    position : absolute;
    top : 0;
    right : 0;
}
```



On constate que l'image « logo CSS 3 » s'est déplacée tout en haut (à `0px` du bord haut) et tout à droite (à `0px` du bord droit) de la fenêtre du navigateur et que l' image « Logo HTML 5 » a disparu...

En fait, l'image « Logo HTML 5 » est bien présente, mais elle est sous l'image « logo CSS 3 », puisqu'elle est aussi concernée par les propriétés top et right. Comme elles ont les mêmes dimensions et que l'image « logo CSS 3 » apparaît après dans le fichier HTML, l'image « Logo HTML 5 » est placée dessous et cachée.

Mais une autre question se pose : pourquoi ces boites `<img>` se sont-elles placées contre les bords supérieur et droit de la fenêtre du navigateur ? En d'autres termes, pourquoi les propriétés `top` et `right` font-elles référence à la fenêtre du navigateur ?

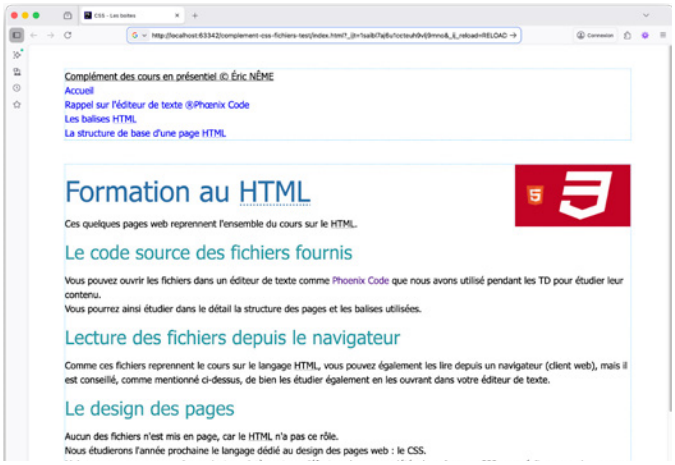
La règle qu'il faut absolument retenir est que, pour une boîte ayant la propriété `position` : `absolute`, les propriétés `top`, `left`, `right` et `bottom` font toujours référence à l'ancêtre le plus proche ayant la propriété position, peu importe sa valeur, sauf static.

Dans notre cas, aucun ancêtre à une propriété `position`, donc c'est la boite `<body>` qui devient la référence, c'est-à-dire la fenêtre du navigateur.

› Ajoutons une propriété `position` à la boite `<main>` :

```
main {
    position : relative;
}

main img {
    position : absolute;
    top : 0;
    right : 0;
}
```



La boite `<main>` possède désormais la propriété `position` et, comme nos deux images appartiennent à sa descendance, elle devient leur référence. Les boites `<img>` se déplacent donc à `0px` du bord supérieur et `0px` du bord droit de la boite `<main>` (la fine bordure bleue permet de mieux repérer ses bords).

J'attire votre attention sur l'astuce que nous avons utilisée et qui est très souvent exploitée par les webdesigners : nous avons attribué la valeur `relative` à la propriété `position` de la boite `<main>`. En effet, comme nous l'avons vu précédemment, cette valeur n'affecte pas la position de la boite `<main>`, ce qui permet de faire d'elle une référence pour nos images sans risque de changement dans la mise en page.

8.3.3 La valeur `fixed` de la propriété `position`

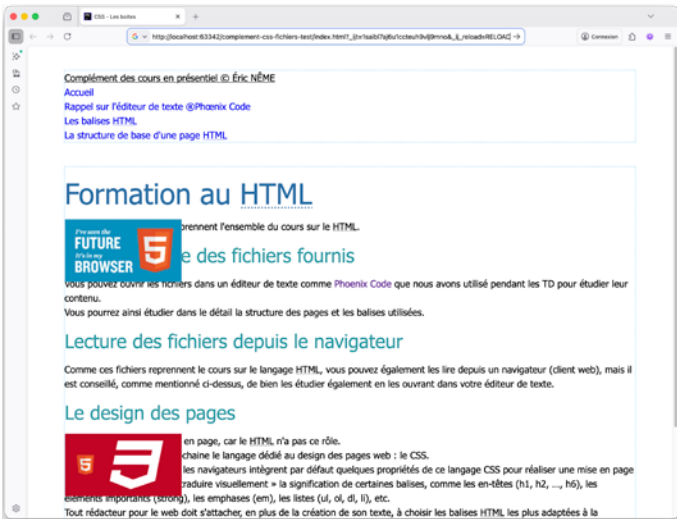
Nous pouvons reprendre tout ce que nous venons de dire de la valeur absolute pour expliquer le comportement d'une boite qui a la propriété `position` : `fixed` à l'exception des points suivants :

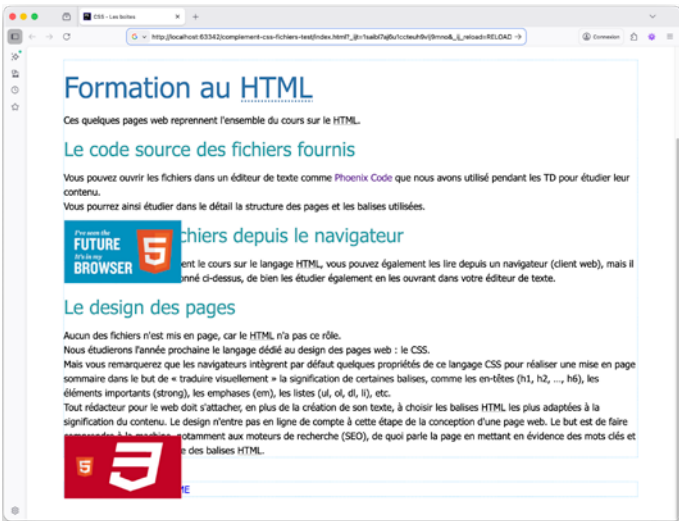
- La boite qui servira de référence pour le placement est toujours la boite `<body>`, c'est à dire la fenêtre du navigateur.
- Si le contenu de la page est plus grand que la zone d'affichage de la fenêtre du navigateur, la boite restera fixe (d'où le nom de cette valeur) et donc toujours visible en cas de défilement du contenu. Ce n'est pas le cas des valeurs vues précédemment pour lesquelles les boites « suivent le mouvement » en cas de défilement jusqu'à disparaître si elles dépassent la zone d'affichage de la fenêtre du navigateur.

On peut l'utiliser, par exemple, pour fixer l'entête (ou le pied de page), ce qui permet de le laisser visible et donc toujours accessible (utile pour le menu de navigation) lorsque l'on fait défiler la page. Dans les exemples qui vont suivre nous interviendrons encore sur nos images.

› Testons la propriété `position` avec la valeur `fixed` :

```
main img {
    position : fixed;
}
```



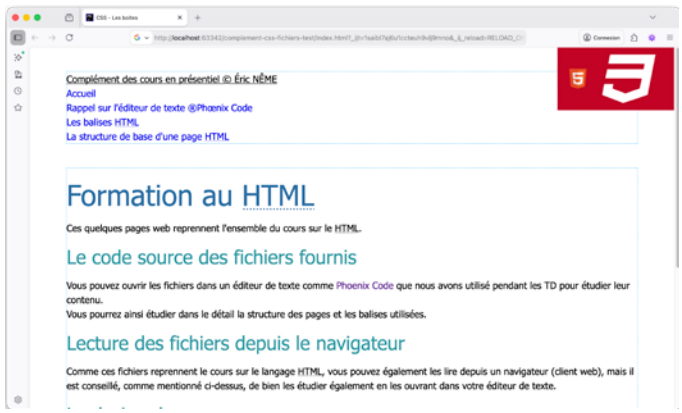


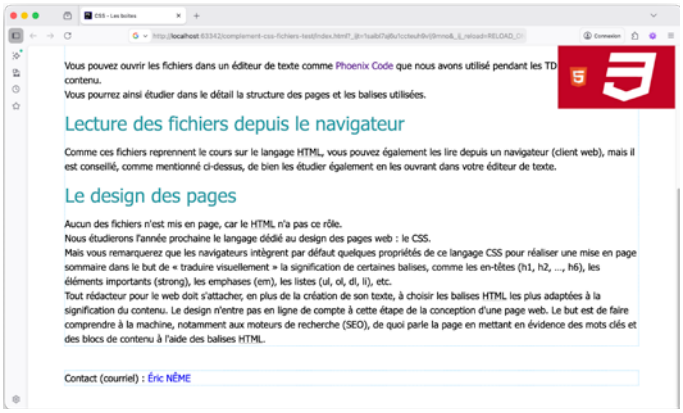
Les deux captures décran montrent la position de nos images en fonction du défilement du contenu. Pour celle de gauche, la barre de défilement est en haut, on voit donc la partie supérieur de la page, pour celle de droite la partie inférieure.

Dans les deux cas, les images sont fixes, elles restent à leur position initiale même en cas de défilement.

› Introduisons les propriétés `top` et `right` :

```
main img {
    position : fixed;
    top : 0;
    right : 0;
}
```





La boite `<body>`, c'est-à-dire la fenêtre du navigateur, est la boite référente pour le positionnement de nos deux images. Elles se placent donc contre le bord supérieur (`top : 0`) et de droite (`right : 0`) de la fenêtre. Même si un de leur ancêtre avait une propriété `position`, cela ne changerait rien à leur comportement : elles restent fixes et toujours visibles en cas de défilement de la page.

8.3.4 La valeur `sticky` de la propriété `position`

Cette valeur assez particulière permet des comportements d'affichage des boites intéressants que l'on rencontre assez souvent. Elle demande un peu plus d'explications pour comprendre son fonctionnement, mais les exemples qui vont suivre devraient pouvoir vous faciliter son utilisation.

La valeur sticky est utile dans le contexte suivant :

- La boite qui possède la propriété `position` : `sticky` dépend de l'ancêtre le plus proche qui a la propriété `position` avec une valeur autre que `static`. Si cet ancêtre n'existe pas, la boite référente est `<body>` c'est-à-dire la fenêtre du navigateur.
- C'est lors du défilement de la page et de la situation de cette boite et de l'ancêtre duquel elle dépend que les effets de la valeur `sticky` seront visibles : la boite se fige pour rester visible lorsqu'elle atteint un bord de la fenêtre. Elle restera dans cette position jusqu'à ce que la boite parente de laquelle elle dépend sorte de la zone d'affichage de la fenêtre du navigateur.

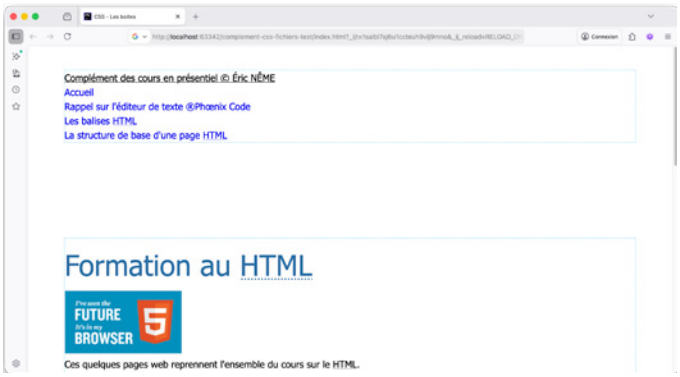
Comme cette valeur `sticky` dépend du défilement de la page, vous pouvez vous référer au schéma de la page 41 qui présente le contexte d'affichage d'une page web dans un navigateur pour mieux comprendre les exemples qui vont suivre.

Pour les besoins de la démonstration, nous allons augmenter la taille des marges externes supérieure et inférieure de la boite `<main>`. Cela permettra de mieux visualiser le comportement de cette valeur, qui nécessite un volume de contenu suffisant pour obtenir un défilement de la page plus ample.

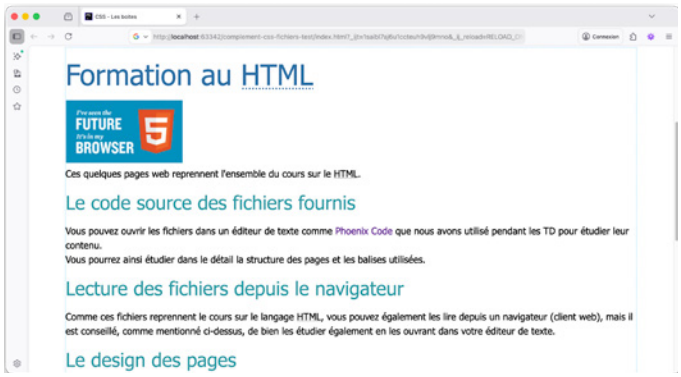
- Testons la propriété `position` avec la valeur `sticky` :

```
main {
    margin : 200px 0;
}

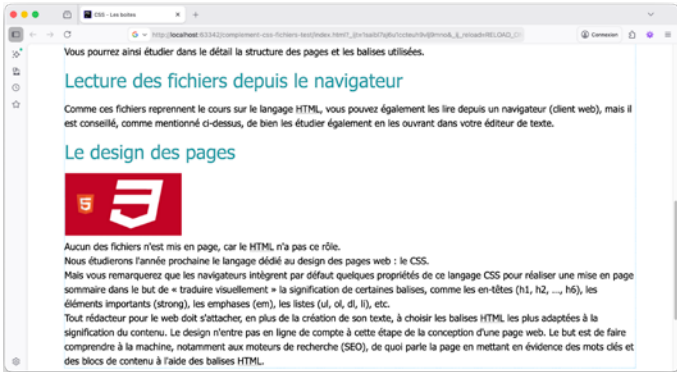
main img {
    position : sticky;
}
```



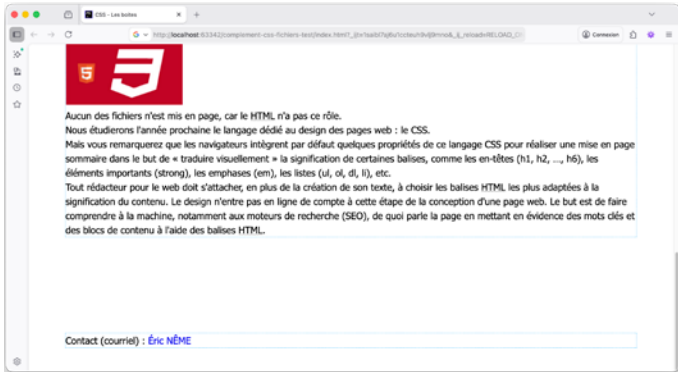
Capture 1 : affichage de la partie haute de la page. L'image «logo HTML 5» est dans sa position initiale.



Capture 2 : défilement du contenu vers le haut, l'image «logo HTML 5» suit le mouvement et reste dans sa position initiale.



Capture 3 : défilement du contenu vers le haut. L'image «logo HTML 5» dépasse le bord supérieur de la fenêtre et n'est plus visible. L'autre image «logo CSS 3» est à sa position initiale.

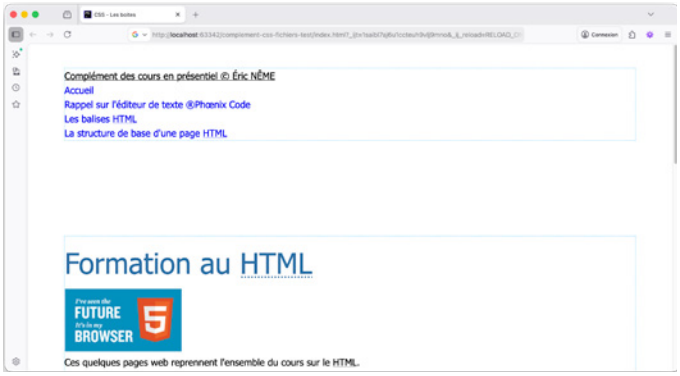


Capture 4 : défilement du contenu vers le haut. L'image «logo CSS 3» suit le mouvement et reste dans sa position initiale.

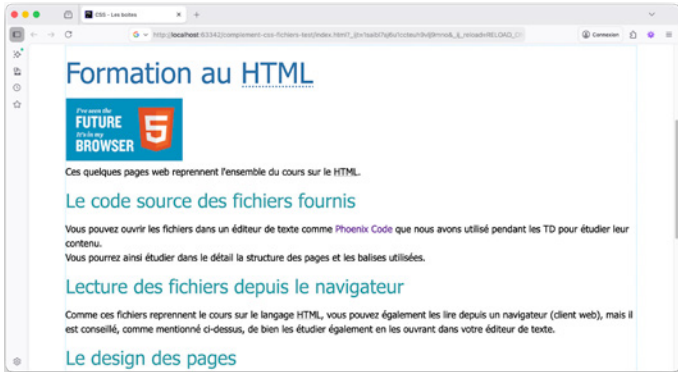
Les quatre captures d'écran montrent que la valeur `sticky` se comporte de la même manière que la valeur `relative` : les deux boites `<img>` restent à leur place initiale et les autres boites n'empiètent pas sur l'espace qu'elles occupent. Ce comportement est conservé, même lors du défilement.

› Introduisons les propriétés `top` et `left` :

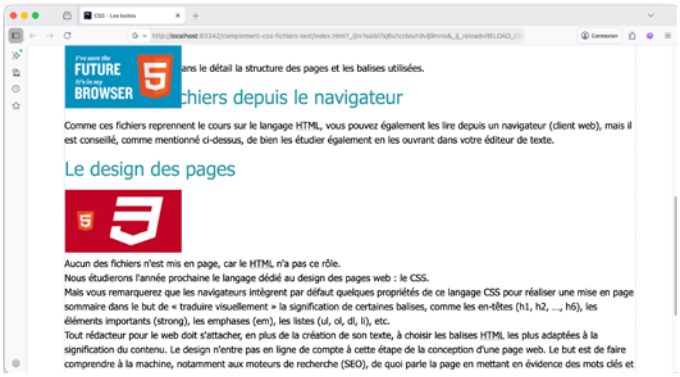
```
main img {
    position : sticky;
    top : 0;
    left : 0;
}
```



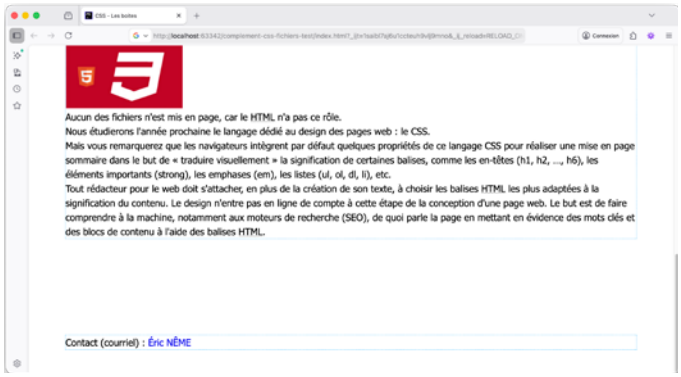
Capture 1 : affichage de la partie haute de la page. L'image «logo HTML 5» est dans sa position initiale.



Capture 2 : défilement du contenu vers le haut, l'image «logo HTML 5» n'a pas atteint le sommet de la fenêtre. Elle suit le mouvement et reste dans sa position initiale.



Capture 3 : défilement du contenu vers le haut. L'image «logo HTML 5» a atteint le sommet de la fenêtre. Elle est maintenant fixe et les boites suivantes lui passent dessous. L'autre image «logo CSS 3» continue de suivre le défilement.



Capture 4 : défilement du contenu vers le haut. C'est au tour de l'image «logo CSS 3» d'atteindre le bord supérieur de la fenêtre et de se figer. L'image «logo HTML 5» n'a pas changé de position, mais se retrouve maintenant sous l'image «logo CSS 3».

Je vous invite à lire les commentaires sous les quatre captures d'écran qui montrent différentes positions du contenu de la fenêtre et des boites `<img>` à certains moments du défilement.

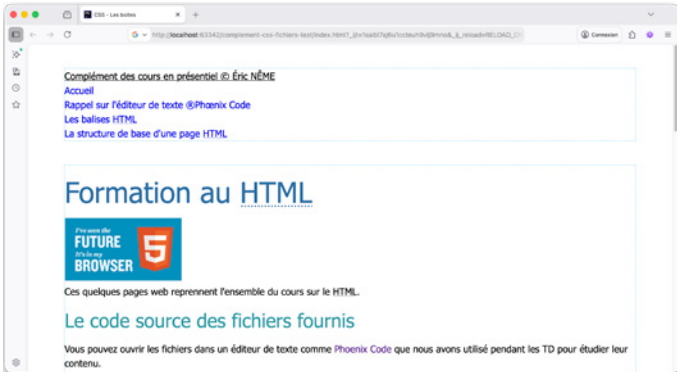
Pour compléter ces commentaires, nous pouvons rajouter les informations suivantes :

- La capture 1 montre que la valeur `sticky` se comporte comme la valeur `relative` : les deux boites <img> restent à leur place initiale et les autres boites n'empiètent pas sur l'espace qu'elles occupent. Ce comportement est conservé, même lors du défilement (cf. capture 2). C'est donc le même comportement que précédemment sans les propriétés `top` et `left`.
- Pour les captures 3 et 4 : dès qu'une boite `<img>` atteint le bord supérieur de la fenêtre, elle se comporte comme si elle avait la valeur `fixed`. La boite se fige et les autres boites empiètent sur son espace comme si elle n'était plus présente.
- Important : lorsque l'on fait défiler le contenu de la page vers le bas pour retrouver la partie supérieure et qu'une boite `<img>` retrouve sa position initiale, elle reprend son comportement normal et suit le mouvement.

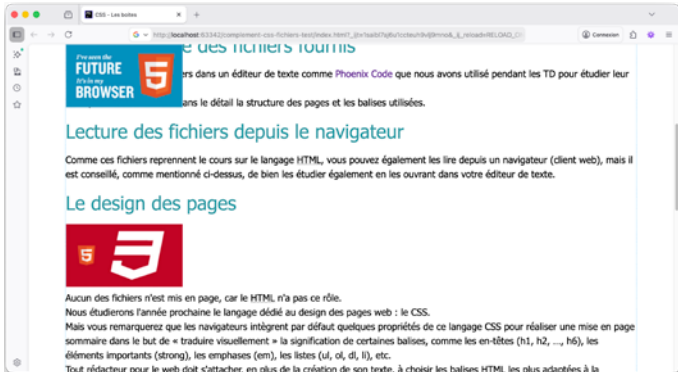
Je vous invite à faire le test sur votre navigateur pour bien comprendre ce mécanisme.

- Rajoutons maintenant la propriété `position : relative` à la boite `<main>`. Rappelez-vous que la valeur `relative` ne modifie pas la position initiale ni le comportement de la boite `<main>`. Nous l'introduisons pour que les deux boites `<img>` puissent y référer leur positionnement. Pour la démonstration nous rajoutons une très grande marge externe au bas de la boite `<main>`.

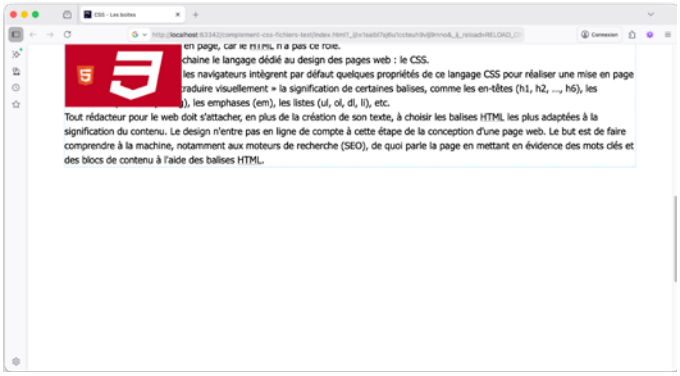
```
main {
    margin-bottom : 1000px;
    position : relative;
}
main img {
    position : sticky;
    top : 0;
    left : 0;
}
```



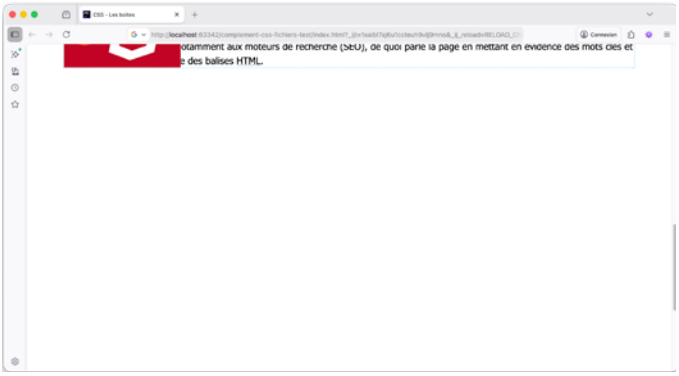
Capture 1 : affichage de la partie haute de la page. L'image «logo HTML 5» est dans sa position initiale.



Capture 2 : défilement du contenu vers le haut. L'image «logo HTML 5» a atteint le sommet de la fenêtre. Elle est maintenant fixe et les boites suivantes lui passent dessous. La boite <main> qui sert de référence commence à sortir la fenêtre. L'autre image «logo CSS 3» continue de suivre le défilement.



Capture 3 : défilement du contenu vers le haut. C'est au tour de l'image «logo CSS 3» d'atteindre le bord supérieur de la fenêtre et de se figer. L'image «logo HTML 5» n'a pas changé de position, mais se retrouve sous l'image «logo CSS 3». Le bas de la boite <main> se rapproche du bord supérieur de la fenêtre.



Capture 4 : défilement du contenu vers le haut. Les boites <img> étant liées à la boite <main> et l'espace entre le bas de la boite <main> et le bord supérieur de la fenêtre étant plus petit que la hauteur des images, elles sont libérées de leur figement. Elles vont disparaitre progressivement en suivant le bord inférieur de la boite <main>.

Je vous invite à lire les commentaires sous les quatre captures d'écran qui montrent différentes positions du contenu de la fenêtre et des boites `<img>` à certains moments du défilement.

Je vous invite à faire le test sur votre navigateur pour bien comprendre ce mécanisme.

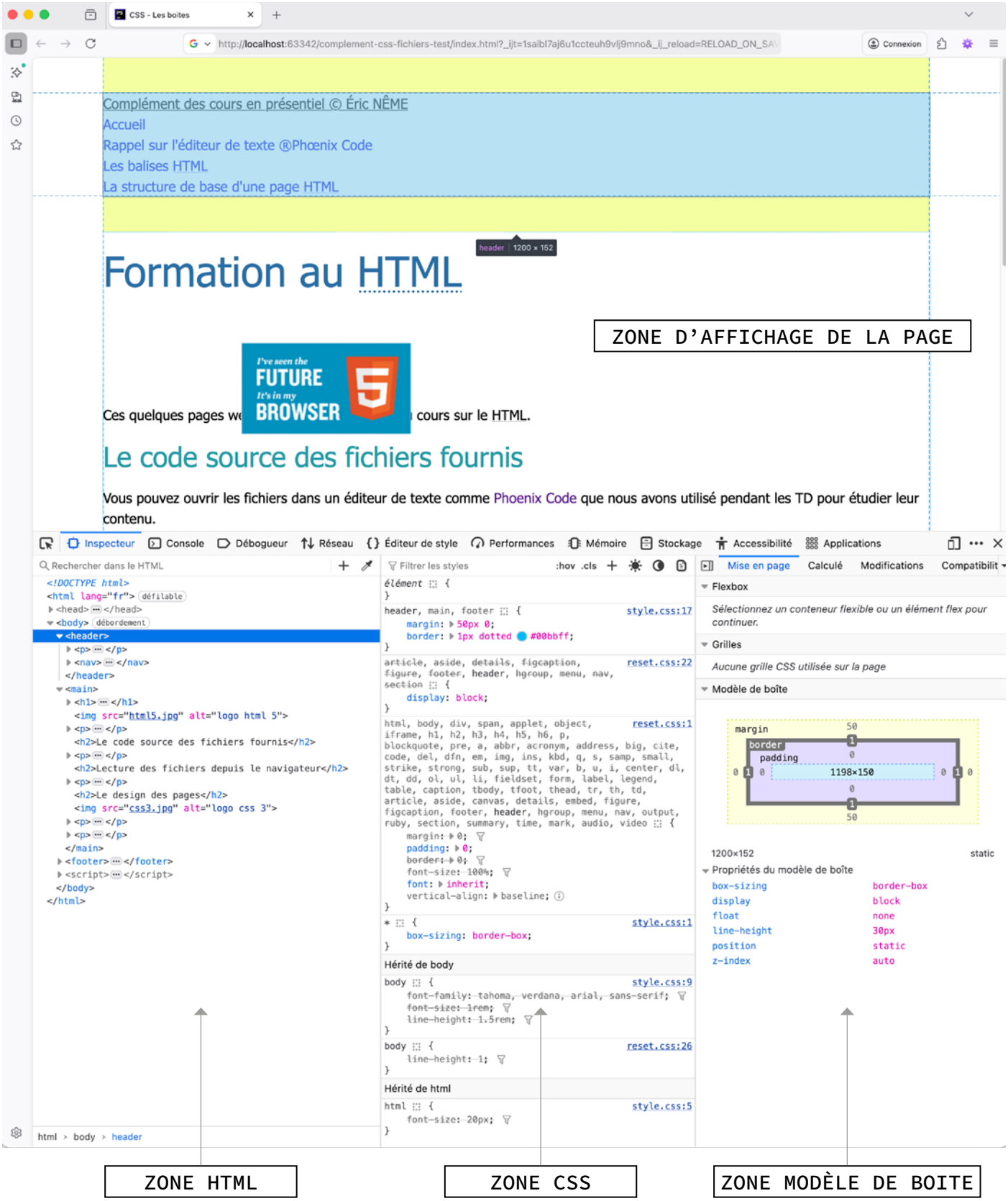
9 - L' outil de développement (l'inspecteur)

Tous les navigateurs modernes intègrent un outil de développement très utile. Grâce à lui, toutes les pages web que vous visualisez dans le navigateur peuvent être analysées, voire modifiées pour réaliser des tests. Vous pouvez vous référer à la documentation Mdn - Outil développement pour avoir plus de détail sur son activation et son utilisation.

Pour aller à l'essentiel, voici quelques informations concernant cet outil :

- Lorsqu'une page web est affichée dans le navigateur, vous pouvez afficher l'outil par un «clic droit» depuis n'importe quel endroit sur la page et en choisissant «inspecter».
- La fenêtre du navigateur se partage en deux parties : la zone d'affichage de la page et la zone de l'outil de développement.
- L'outil de développement se décompose lui-même en trois parties :
  - La partie de gauche qui montre le HTML.
  - La partie centrale qui montre les styles CSS.
  - La partie de droite qui montre sous forme d'un schéma le modèle de boite CSS.
- Beaucoup d'autres options d'affichages sont disponibles dans cet outil correspondant à un grand nombre d'informations utiles. Je vous invite à les découvrir par vous-même en consultant la documentation et les nombreux tutoriels que vous trouverez sur le web.

Voici un exemple de présentation de cet outil :



Avec les quelques informations complémentaires qui vont suivre sur l'outil de développement, vous devriez pouvoir mieux comprendre le comportement des éléments des pages sur lesquelles vous allez travailler et faire des tests pour apporter quelques améliorations avant de les intégrer définitivement dans vos feuilles de style.

- Dans la zone HTML, lorsque l'on survole une balise avec la souris, l'élément est signalé dans la zone d'affichage : il est entouré de repères et, s'il est affecté par des propriétés du modèle de boite CSS, celles-ci sont visuellement indiquées à l'aide d'un code couleur que l'on retrouve dans la zone de modèle de boite.
- Si l'on clique sur une balise HTML de la zone HTML, les propriétés CSS qui l'affectent sont listées en détail dans la zone CSS. Pour cet élément, il est mentioné :
  - Dans quel fichier CSS et à quelle ligne il a été repérée par le navigateur.
  - Les propriétés qui le concernent.
  - De quelles propriétés il hérite et quel est l'ancêtre qui les lui a transférées.
  - Les propriétés héritées qui ne lui sont pas appliquées apparaissent en grisé barré.
- Dans la zone modèle de boite, un schéma représentatif de la boite montre ses propriétés fondamentales et leurs valeurs précises. Certaines de ces valeurs peuvent être calculées par le navigateur, dans ce cas, vous aurez accès au résultat de son calcul. De plus, sous ce schéma, vous trouverez la synthèse de ces propriétés fondamentales, mais aussi de celles qui permettent de la positionner.

10 - Conclusion

Ce complément de cours a présenté en détail le concept de boite, de modèle de boite CSS et de placement des boites. Bien qu'il n'épuise pas toutes les propriétés de style, il est un premier volet incontournable pour la mise en page d'un fichier HTML. C'est donc une base suffisante pour présenter de manière harmonieuse et ergonomique le contenu d'une page et le hiérarchiser.

Néanmoins, il présente une lacune que j'ai mentionnée à plusieurs reprises, mais qui aurait demandé un développement trop long pour la combler. Il s'agit des deux propriétés `display : grid` et `display : flex` qu'il faudrait étudier par vous-même en utilisant les références que vous trouverez dans la webographie ci-après.

Enfin, je vous invite à vous exercer en testant les propriétés étudiées. Je vous transmets une archive ZIP qui contient le fichier HTML, les images et les deux fichiers CSS utilisés comme support dans ce document. Pour vos tests, n'hésitez pas à les modifier en supprimant, rajoutant et changeant tout ce que vous voulez.

11 - Webographie

11.1 Tutos vidéo

Je vous recommande vivement de suivre ces formations gratuites et notamment les vidéos de mise en pratique.

- Sur le HTML : https://grafikart.fr/formations/html
- Sur le CSS : https://grafikart.fr/formations/css
- Desgin pattern en CSS : https://grafikart.fr/formations/css-design-pattern

11.2 Le HTML

- Site Mozilla (MDN) : https://developer.mozilla.org/fr/docs/Web/HTML
- Les balises HTML (MDN) : https://developer.mozilla.org/fr/docs/Web/HTML/Reference/Elements
- W3 schools : https://www.w3schools.com/html/default.asp

11.3 Le CSS

- Site Mozilla (MDN) : https://developer.mozilla.org/fr/docs/Web/CSS
- Les propriétés CSS (MDN) : https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Properties
- W3 schools : https://www.w3schools.com/css/default.asp
- CSS Tricks : https://css-tricks.com
- Jeu pour maîtriser la propriété `display` : `grid` : https://cssgridgarden.com/#fr
- Jeu pour maîtriser la propriété display `display` : `flex` : https://flexboxfroggy.com/#fr