Ouvrir le Box Model

Nous nous sommes familiarisés avec HTML et CSS, nous savons à quoi cela ressemble et nous maîtrisons les bases. Maintenant nous allons aller un peu plus loin et regarder exactement comment sont affichés les éléments sur une page et comment ils sont dimensionnés.
Dans le processus nous allons discuter de ce qui est appelé le modèle de boîte et comment il fonctionne avec HTML et CSS. Nous allons également examiner quelques nouvelles propriétés CSS et utiliser certaines des valeurs de longueur que nous avons abordés dans la leçon précédente. C'est parti !

Comment les éléments sont affichés?

Avant de plonger dans le modèle de boîte, essayons de comprendre comment les éléments sont affichés. Dans l'un des cours précédent nous avons découvert la différence entre les éléments de niveau bloc et les éléments de niveau ligne. Pour résumer rapidement, les éléments de niveau bloc occupent toute la largeur disponible, quelle que soit leur contenu, et ils commencent toujours sur une nouvelle ligne. Les éléments de niveau de ligne occupent seulement la largeur de leur contenu et vont s'aligner sur la même ligne, l'un après l'autre. Les éléments de niveau bloc sont généralement utilisés pour de gros morceaux de contenu, tels que les titres et pour les éléments de structure. Éléments de niveau ligne sont généralement utilisés pour de petits morceaux de contenu, par exemple pour mettre en gras ou en italique quelques mots choisis.

Display

La façon dont les éléments sont affichés, éléments de niveau bloc, éléments en ligne, ou autre, est déterminée par la propriété display. Chaque élément a évidemment une valeur par défaut pour la propriété display. Cependant, comme avec toutes les autres valeurs de propriété, cette valeur peut être écrasée. Il y a plusieurs valeurs pour la propriété `display,` mais les plus courantes sont `block`, `inline`, `inline-block`,et `none`.

Nous pouvons changer la valeur d'un élément `display.` Par exemple une valeur `block` fera que cet élément passera du niveau en ligne au niveau bloc.

```
p{
  display: block;
}
```

Une valeur d'affichage `inline` fera cet élément un élément de niveau en ligne.

```css
p {
  display: inline;
}
```

Les choses deviennent intéressantes avec la valeur `inline-block`. Cette valeur permettra à un élément de se comporter comme un élément de niveau bloc, acceptant toutes les propriétés du modèle de boîte (que nous aborderons bientôt). Toutefois, l'élément sera affiché en ligne avec les autres éléments, et il ne commencera pas sur une nouvelle ligne par défaut.

```css
p {
  display: inline-block;
}
```

L’espace entre les éléments Inline-Block

Une distinction importante avec les éléments inline-block est qu'ils ne sont pas toujours accolés ou affichés directement l’un après l'autre. Habituellement un petit espace existe entre deux éléments inline-block. Cet espace peut être ennuyeux, mais est tout à fait normal. Nous expliquerons la raison pour laquelle cet espace existe et comment le supprimer dans la suite du cours

Enfin, si on utilise une valeur `none` pour la propriété `display` on va complètement cacher l’élément et rendre cet élément invisible dans le flux de la page, de fait faire comme s’il n'existait pas. Tous les éléments imbriqués dans cet élément seront également cachés.

```css
div {
  display: none;
}
```

Savoir comment les éléments sont affichés et comment modifier la valeur `display` est assez important, car l'affichage d'un élément a une implications sur la façon dont le modèle de boîte est rendu. Nous ne manquerons pas d'examiner ces différentes implications et comment ils peuvent affecter la présentation d'une page.

Quel est le modèle de boîte?

Selon le concept du modèle de boîte, chaque élément d'une page est dans une boîte rectangulaire et peut avoir une largeur, une hauteur, des marges internes,des bordures et des marges externes.

Cela vaut peine de le répéter :

chaque élément d'une page est une boîte rectangulaire .

Figure 1

Lorsque nous examinons chaque élément individuellement, nous pouvons voir comment ils sont tous rectangulaires, peu importe leurs formes présentées

Chaque élément sur chaque page se conforme au modèle de boîte, il est donc extrêmement important. Jetons un coup d'oeil, ainsi que quelques nouvelles propriétés CSS, afin de mieux comprendre ce que nous travaillons avec.

Travailler avec le modèle de boîte

Chaque élément est une boîte rectangulaire, et il y a plusieurs propriétés qui déterminent la taille de cette boîte. Le noyau de la boîte est définie par la largeur et la hauteur d'un élément, qui peut être déterminée par la propriété `display` par le contenu de l'élément, ou en spécifiant les propriétés width et `height`. La marge interne `padding` puis les bordures `border` vont élargir les dimensions de la boîte vers l'extérieur de la largeur et de la hauteur de l'élément. Enfin, les marges, propriété `margin` seront après la bordure.
Chaque partie du modèle de boîte correspond à une propriété CSS: width, height, padding, `border`, et `margin`..

Regardons ces propriétés dans du code:

```
div {
  border: 6px solid #949599;
  height: 100px;
  margin: 20px;
  padding: 20px;
  width: 400px;
}
```

Selon le modèle de boîte, la largeur totale d'un élément peut être calculé selon la formule suivante:

```
margin-right + border-right + padding-right + largeur +
padding-left + border-left + margin-left
```

En comparaison, selon le modèle de boîte, la hauteur totale d'un élément peut être calculé selon la formule suivante:

```
margin-top + border-top + padding-top + hauteur + padding-bottom + border-bottom + margin-bottom
```

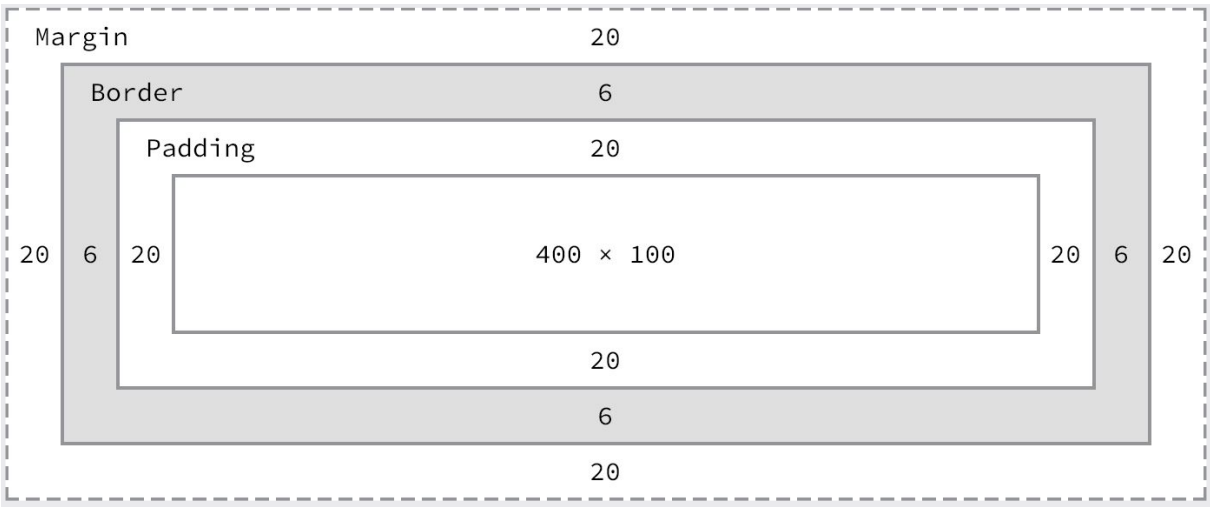

Figure 2
Le modèle de boîte comprends les dimensions suivantes: hauteur, largeur, marges internes, bordures et marges.

En utilisant les formules, nous pouvons trouver la hauteur totale et largeur de notre code d'exemple

- Largeur: `492px = 20px + 6px + 20px + 400px + 20px + 6px + 20px`
- Hauteur: `192px= 20px + 6px + 20px + 100px + 20px + 6px + 20px`

Le modèle de boîte est sans aucun doute une des parties les plus déroutante du HTML / CSS. Nous avons fixé une valeur de `400` pixels à la propriété `width`, mais la largeur réelle de notre élément est de `492 pixels`.
Par défaut le modèle de boîte est additif; donc de déterminer la taille réelle d'une boîte nous devons tenir compte des marges internes,bordures et marges pour les quatre côtés de la boîte. Notre largeur comprend non seulement la valeur de propriété `width,` mais aussi la taille du padding gauche et droite, des bordures gauche et droite, et les marges gauche et droite.
Jusqu'à présent beaucoup de ces propriétés pourraient ne pas faire sens, on est d'accord. Pour clarifier les choses, nous allons jeter un coup d'oeil à tous les propriété -`width`, `height`, `padding`,`border` et `margin`- qui participent à la formation du modèle de boîte.

Largeur et Hauteur

Chaque élément à une largeur et une hauteur par défaut. Cette largeur et hauteur peuvent être non précisée mais les navigateurs, par défaut, rendront tous les éléments avec une taille. Selon la façon dont un élément est affiché, la largeur et la

hauteur par défaut peuvent être adéquates. Si un élément est la clé de la mise en forme d'une page, il peut exiger de spécifier les valeur des propriétés `width` et `height`.

Largeur

La largeur par défaut d'un élément dépend de sa valeur d'affichage. Les éléments de niveau bloc ont une largeur par défaut de 100% et consomme la totalité de l'espace horizontal disponible. Les éléments inline et inline-block se dilatent et se contractent horizontalement pour accueillir leur contenu. Les éléments de niveau inline ne peuvent pas avoir une taille fixe, ainsi que des `width` et `height` qui ne concernent que les éléments non inline.
Pour définir une largeur spécifique pour un élément non-inline, utilisez la propriété width:

```
div{
  width: 400px;
}
```

Hauteur

La hauteur par défaut d'un élément est déterminée par son contenu. Un élément se dilate et se contracte verticalement au besoin pour tenir son contenu. Pour définir une hauteur spécifique pour un élément non-inline, utilisez la propriété `height`:

```
div{
  height: 100px;
}
```

Dimensionnement des éléments Inline

Gardez à l'esprit que les éléments de niveau inline n'acceptent pas les propriétés `width` et `height`.

Margin & padding

Selon l'élément, les navigateurs peuvent appliquer des marges internes et externes à un élément par défaut pour aider à la lisibilité et à la clarté. Nous allons généralement voir cela avec les éléments de base pour le texte. Évidemment les marges externes par défaut et internes peuvent varier pour un élément d'un navigateur à un autre. Dans le premier cours nous avons parlé de l'aide d'une réinitialisation CSS pour homogénéiser et consolider l'ensemble de ces valeurs par défaut à zéro. Cela nous permet de travailler avec des fondations solides et de préciser nos propres valeurs.

Margin

La propriété `margin` nous permet de définir la quantité d'espace qui entoure un élément. Marges pour un élément se situent en dehors de toute bordure et sont complètement transparentes, pas de couleur. Les marges peuvent être utilisées pour aider à positionner un éléments dans un endroit particulier sur une page ou de fournir une marge de manœuvre,gardant tous autres éléments une ont une distance de sécurité.
Voici la propriété `margin` en action:

```
div {
  margin: 20px;
}
```

Une bizarrerie avec la propriété `margin` est que les marges verticales, `top` et `bottom`, ne sont pas acceptés par les éléments inline. Ces marges verticales sont cependant acceptées par les éléments de niveau block et inline-block.

padding

La `padding` est une propriété très similaire à la propriété `margin` cependant, elle ne va impacter que la marge intérieure de la bordure d'un élément, un élément à toujours une bordure même si elle n'est pas forcément visible. La propriété `padding` est utilisée pour fournir un espacement à l'intérieur un élément. Voici le code:

```
div {
  padding: 20px;
}
```

La propriété `padding`, contrairement à la propriété `margin`, fonctionne verticalement sur des éléments de niveau en ligne. Ce `padding` vertical peut se fondre dans la ligne en dessus ou au dessous de l'élément donné, mais il sera affiché.

Margin & padding pour les éléments Inline

Les éléments inline sont affectés un peu différemment des éléments block et inline-block en ce qui concerne les margin et les padding.

- `Margin` ne fonctionnent que horizontalement -`left` et `right`- sur une ligne.
- `Padding` fonctionne sur les quatre côtés d'un élément inline, sachant que, dans l'axe vertical le `padding -top` et `bottom-` peut déborder sur les lignes au dessus et au dessous un élément.

Les `margin` et `padding` fonctionnent normalement avec les éléments block et inline-block.

Déclarations des margin & padding

En CSS, il y a plus d'une façon de déclarer les valeurs de certaines propriétés. Nous pouvons utiliser la version longue, énumérant chacune des propriétés et valeurs les une après l'autre, ou chaque propriété aura sa propre valeur. Nous pouvons aussi utiliser la version raccourcie, condensée, énumérant plusieurs valeurs associées à une seule propriété. Toutes les propriétés ne possèdent pas une alternative courte, nous devons donc nous assurer que nous utilisons la structure de propriété/valeur correcte.
Les propriétés `margin` et `padding` peuvent être déclarées des deux façons, sous la forme détaillée ou la forme abrégée. Lorsque vous utilisez la forme abrégée de `margin` pour définir la même valeur aux quatres côtés d'un élément, nous spécifions une seule valeur:

```
div{
  margin: 20px;
}
```

Pour définir une valeur pour le `top` et `bottom` et une autre valeur pour les `left` et `right` côtés d'un élément, vous pouvez ne spécifier que deux valeurs: `top` et `bottom` d'abord, puis `left` et `right` ensuite. Ici nous mettons des marges de `10` pixels sur le `haut` et `bas` d'un `<div>` et des marges de `20 pixels` sur la `gauche` et droite:

```
div{
  margin: 10px 20px;
}
```

Pour définir des valeurs uniques pour les quatre côtés d'un élément, spécifiez ces valeurs dans l'ordre de `top`, `right`, `bottom`, et `left`, en se déplaçant dans le sens des aiguilles d'une montre. Ici Nous mettons des marges de `10 pixels` sur le `dessus` d'un , `20` pixels sur la droite, `0` pixels sur le dessous et `15 pixels` sur la gauche.

```
div{
  margin: 10px 20px 0 15px;
}
```

Utiliser les propriétés uniques `margin` ou `padding` avec un certain nombre de valeurs, est considéré comme un raccourci. Nous pouvons définir la valeur d'un côté à la fois en utilisant une propriétés uniques. Chaque nom de propriété (dans le cas de margin ou padding) est alors suivi d'un tiret et du nom du côté de la boîte à laquelle la valeur doit être appliquée: `top`, `right`, `bottom` ou `left`. Par exemple, la propriété `padding-left` acceptera un seule valeur et fixera la largeur interne gauche de cet élément et la propriété `margin-top` qui acceptera une seule valeur, fixera la marge supérieure pour cet élément.

Par exemple:

```
div{
  margin-top: 10px;
  padding-left: 6px;
}
```

Quand on veut identifier une seule `marge`, interne ou externe, il est préférable d'utiliser les propriétés détaillées. Cela maintient notre code explicite et nous aide à éviter toute confusion. Par exemple, ne nous voulons vraiment régler que le haut de l'élément pour avoir une marge de `0` pixels, ou si ne nous voulons vraiment régler la marge `inférieure` à `10` pixels. Lorsque vous traitez avec trois ou plusieurs valeurs, cependant, la version condensée est très utile.

Couleurs des marges

Les propriétés `margin` et `padding` sont complètement transparentes et n'acceptent pas de valeurs de couleur. Puisqu'elles sont transparentes, les marges montrent les couleurs d'arrière plan des éléments relatifs. Pour les `margin` , nous voyons la couleur de fond de l'élément parent, et pour `padding` , nous voyons la couleur de fond de l'élément auquel il est appliqué.

Les Bordures

Elles se situent entre le `margin` et `padding`, rendant graphiquement un filet autour d'un élément. La propriété `border` requiert trois valeurs: `width`, `style` et `color.` `La` La forme raccourcie de la propriété `border` sont indiqués dans cette ordre -largeur -`width-`, style et couleur -`color-`.Dans la version complète, ces trois valeurs peuvent être divisés en propriétés : `border-width`, `border-style` et `border-couleur`.
Ces propriétés sont utiles pour modifier ou écraser, une valeur unique la bordure.
La `width` et `color` des bordures peuvent être définies en utilisant les unités CSS de longueur et couleur, vues dans la leçon précédente.
les bordures peuvent avoir différentes apparences. Les valeurs de style les plus courantes sont `solid`, `double`, `dotted`, `dashed,` et `none` ..mais il y a plusieurs autres.
Voici le code pour une bordure large de 6 pixel, pleine et de couleur grise qui entourera les quatres côtés d'une `<div>`:

```
div {
  border: 6px solid #949599
}
```

Exemple de bordures

HTML

```
<div class="border-solid">2px <br> solid</div>
<div class="border-double">6px <br> double</div>
<div class="border-dashed">8px <br> dashed</div>
<div class="border-dotted">4px <br> dotted</div>
```

CSS

```
div{
  background: #eaeaed;
  color: #666;
  display: inline-block;
  height: 70px;
  margin: 0 14px;
  padding-top: 20px;
  text-align: center;
  width: 90px;
}
.border-solid {
  border: 2px solid #9799a7;
}

.border-double {
  border: 6px double #9799a7;
}
.border-dashed {
  border: 8px dashed #9799a7;
}
.border-dotted {
  border: 4px dotted #9799a7;
}
```

Résultat

Gérer les tous les côtés des bordures

Comme avec les propriétés `margin` et padding, les bordures peuvent être placés sur un côté à la fois s d'un élément. Cela nécessite alors d'utiliser des nouvelles propriétés: `border-top`, `border-right`, `border-bottom` et `border-left`. Les valeurs de ces propriétés sont les mêmes que celle de la propriété `border` seule: `width`, `style` et `color`. Si nous voulons, nous pouvons faire une qu'une bordure n'apparaisse qu'en dessous d'un élément:

```
div{
  border-bottom: 6px solid #949599
}
```

En outre, les styles des différents côtés de bordures peuvent être contrôlés à un niveau encore plus fin. Par exemple, si l'on ne veut modifier que la largeur de la bordure inférieure nous pouvons utiliser le code suivant:

```
div{
  border-bottom-width: 12px;
}
```

Ces propriétés très spécifiques de bordure comprennent une série de mots clés séparés par des traits d'union à partir de du nom de la propriété, `border`, suivie par le côté sélectionné, `top right`, `bottom`, ou `left` et puis en fonction de la propriété désirée `width`, `style` ou `color`.

Border radius

Si nous faisons le tour de toutes les propriétés différentes pour les bordures, nous devons examiner la propriété `border-radius` qui nous permet d'arrondir les angles d'un élément.
La propriété `border-radius` accepte des unités de longueur, y compris pourcentages et pixels, pour déterminer le rayon par lequel les coins d'un élément doivent être arrondies. Une seule valeur arrondit les quatre coins d'un élément, deux valeurs arrondissent dans cet ordre les coins haut gauche -`top left-` et bas droit -`bottom right-` et les coins haut droite -`top right-` bas gauche -`bottom left-` et enfin quatre valeurs pour arrondir les coins en haut à gauche -`top left-`, en haut à droite -`top right-`, en bas à droite -`bottom right-` et en bas à gauche -`bottom left-` dans cet ordre.
Rappelez vous que l'ordre dans lequel les valeurs sont appliquées à la propriété `border-radius` (cela est aussi le cas pour les propriétés `margin` et `padding`), est celui dans le sens des aiguilles d'une montre à partir du coin supérieur gauche d'un élément.

```
div{
  border-radius: 5px
}
```

Exemple de border-radius

HTML

```html
<div class="border-rounded">5px </div>
<div class="border-cercle">50% </div>
<div class="border-ovale">15px 75px</div>
```

CSS

```css
div{
  background: #eaeaed;
  color: #666;
  display: inline-block;
  height: 70px;
  margin: 0 14px;
  padding-top: 20px;
  text-align: center;
  width: 90px;
}
.border-rounded {
  border-radius: 5px;
}

.border-cercle {
  border-radius: 50%;
}
.border-ovale {
  border-radius: 5px 15px 25px 50px;
}
```

Résultat

La propriété `border-radius` peut également être éclatée en plusieurs propriétés qui nous permettent de changer les rayons des coins individuels d'un élément. Ces propriétés commencent avec `border`, continuer avec emplacement de l'angle vertical `( top` ou `bottom )` et emplacement horizontal du coin `( left` ou `right )`, puis terminer par rayon. Par exemple, pour modifier le rayon d'un haut à droite d'une `<div>`, peut être utilisée la propriété `border-top-right-radius`.

```
div{
  border-top-right-radius: 5px
}
```

Box Sizing

Jusqu'à présent le modèle de boîte se faisait par addition de valeurs. Si vous définissez la `width` d'un élément à `400` pixels, puis ajouter `20` pixels de `padding` et une `border` de `10` pixels sur tous les côtés, la largeur de l'élément réel devient `460` pixels. Rappelez vous que nous devons ajouter les propriétés `width`, `padding` et `border` valeurs ensemble pour obtenir la largeur réelle d'un élément.
Le modèle de boîte peut toutefois être modifié pour supporter différents calculs. CSS3 introduit la propriété `box-sizing,` qui nous permet de changer le fonctionnement du modèle de boîte et donc comment la taille d'un élément est calculée. La propriété accepte trois valeurs -`content-box`, `padding-box` et `border-box`- chacune ayant un impact légèrement différent sur la façon dont la taille de la boîte est calculée.

Content box

La valeur `content-box` est la valeur par défaut qui correspond au modèle de boîte tel que l'on le connaît. Si nous précisons la propriété `box-sizing,` avec cette valeur par défaut elle s'appliquera à tous éléments et donc la La taille d'un élément sera l'addition des propriétés `width` et `height,` puis la valeur des propriétés `padding`, `border,` ou `margin` y seront ajoutés.

```
div{
  -webkit-box-sizing: content-box;
  -moz-box-sizing: content-box;
  box-sizing: content-box;
}
```

Propriétés spécifiques et valeurs spécifiques des navigateurs web

A quoi correspondent tous ces préfixes que l'on voit ci-dessus sur la propriété `box-sizing` ?
Comme en CSS3 a été introduit,navigateurs ont progressivement commencé à soutenir des propriétés et des valeurs différentes, y compris pour la propriété `box-sizing,` en utilisant des "préfixes fournisseurs". Comme seules certaines parties de la spécification CSS3 sont finalisées et que les nouvelles versions du navigateur sont réalisées plus fréquemment, ces préfixes fournisseurs deviennent de moins en moins pertinente. Dans quelques temps les préfixes des fournisseurs ne seront probablement que de lointains souvenir; cependant, ces préfixes continuent à fournir une aide pour supporter certains anciens navigateurs qui les avaient mises à profit.

Le préfixes fournisseurs peuvent être vus sur les propriétés et les valeurs, en fonction de la spécification CSS. Ici ils sont présentés sur la propriété `box-sizing`.
Les éditeurs de navigateurs étaient libres de choisir quand utiliser un préfixe et quand ne pas le faire. Ainsi, certaines propriétés et valeurs exigent des préfixes pour certains fournisseurs de navigateur mais pas pour autres.
A l'avenir, quand une propriété ou une valeur aura besoin d'un préfixe de fournisseur, le préfixe ne sera utilisé que dans l'introduction de cette propriété ou valeur (dans l'intérêt de gardant notre code simple et concis). Ne pas oublier d'ajouter les préfixes fournisseurs nécessaires lorsque vous êtes train écrire le code.
Pour référence,préfixes des fournisseurs les plus courants:

- Mozilla Firefox: `-moz-`
- Microsoft Internet Explorer: `-ms-`
- Webkit(Google Chrome et Apple Safari): `-webkit-`

Padding-box

La valeur `padding-box` modifie le modèle de boîte en incluant toutes les valeurs de propriété de padding au sein de la largeur et de la hauteur d'un élément. Lors de l'utilisation de la valeur `padding-box,` si un élément a une `largeur` de `400` pixels et un `padding` de `20` pixels de chaque côté, la largeur réelle restera `400` pixels. Toute augmentation des valeurs de padding, fera que la taille du contenu dans un élément se réduira proportionnellement.
Si nous ajoutons une bordure ou une marge,ces valeurs seront ajoutées aux largeurs ou hauteur pour calculer la taille totale de la boîte. Par exemple, si on ajoute une `border` de `10` pixels et un `padding` de `20` pixels de chaque côté de l'élément d'une `width` de `400` pixels, la largeur réelle deviendra `420` pixels.

```
div{
  box-sizing: padding-box;
}
```

Border-box

Enfin, la valeur `border-box` modifie le modèle de boîte de telle sorte que toutes les bordures ou padding seront inclus dans la largeur et hauteur d'un élément. Lorsque vous utilisez la valeur `border-box,` si un élément a une `width` de `400` pixels, un `padding` de `20` pixels, et une `border` de `10` pixels de chaque côté, la largeur réelle restera `400` pixels.
Si on ajoute une `margin`, ces valeurs devront être ajoutés pour calculer la taille totale de la boîte. Peu importe quelle valeur de `box-sizing` est utilisée, toute les valeurs `de margin` devront être ajoutés pour calculer la taille totale de l'élément.

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

Figure 3
Les différentes valeurs de box-sizing feront que la largeur d'un élément—sa boîte—aura une surface calculée différente.

Choisir la bonne valeur de box-sizing

Généralement, la meilleure valeur de `box-sizing` à utiliser est `border-box`.
La valeur `border-box` rend nos calculs beaucoup, beaucoup plus facile. Si nous voulons qu'un élément soit de `400` pixels large, il est, et il restera à `400` pixels de large et ce peu importe les valeurs de padding ou de bordures que nous y ajouterons.
De plus, on peut facilement mélanger les valeurs de longueur. Disons que nous voulons notre boîte soit de large de `40%`. Ajouter un `padding` de `20` pixels et une `border` de `10` pixels de chaque côté d'un élément ne posera pas de problème, et nous pouvons toujours garantir que la largeur réelle de notre boîte restera `40%`.
Le seul inconvénient à l'utilisation la propriété `box-sizing` est que dans le cadre de la spécification CSS3, il n'est pas pris en charge dans tous les navigateurs, il manque en particulier de support dans les navigateurs les plus anciens. Heureusement cela devient de moins en moins pertinent au fur et à mesure que les nouveaux navigateurs sortent.

Les outils de développement

La plupart des navigateurs ont ce qu'on appelle les "*Developer Tools*", en français "Outils pour développeurs" .Ces outils nous permettent d'inspecter un élément sur une page, voir où cet élément se positionne dans le document HTML, et voir quelles sont les propriétés et valeurs CSS qui lui sont appliquées. La plupart de ces outils comprennent également un diagramme de modèle de boîte pour montrer la taille calculée d'un élément.

Pour voir les outils de développement de Google Chrome, faites clic-droit "Inspecter cet élément" ou dans le barre de menu, choisissez l'entrée "Plus d'outils" et puis "Outils de développement". Cela ouvre un panneau, par défaut au bas de la fenêtre du navigateur, qui offre une poignée d'outils pour inspecter notre code.

En cliquant sur la “flèche” dans la barre d’outil , cela nous permet de passer au dessus des éléments de la page et de cliquer différents éléments sur la page pour examiner les informations données sur eux dans le panneau “développeur”.
Après avoir sélectionné un élément, nous allons voir une poignée d'onglets sur le côté droit du panneau. La sélection de l'onglet "Computed" va nous montrer une ventilation du modèle de boîte pour notre élément sélectionné dans le navigateur.
Jouer avec les outils de développement, que ce soit dans Google Chrome, Mozilla Firefox, Apple Safari, ou autres il y a beaucoup à apprendre à regarder le code réalisé. Inspecter le code des autres sites pour voir comment ils sont construits, est intéressant pour s’inspirer ou pour apprendre.

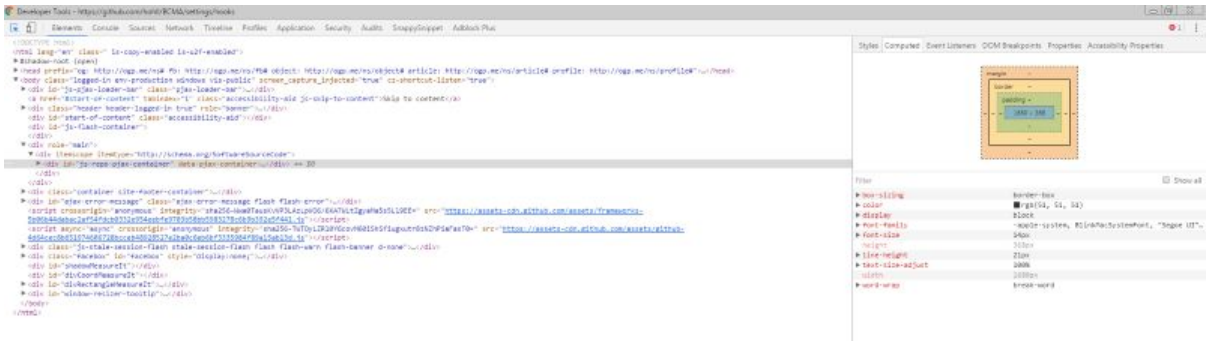

Figure 4
Le Google Chrome Developer Tools, qui nous aident à inspecter le code HTML et CSS sur une page

Dans la pratique

Revenons à notre site pour lui ajouter un peu de mise en forme
Commençons en ajustant la taille de notre boîte en utilisant la version `border-box` du modèle de boîte, qui fera que l’ajustement de la taille de tous nos éléments sera beaucoup plus facile. Dans notre fichier `style.css` nous allons ajouter un commentaire pour identifier le code de ce qui deviendra notre grille qui va nous aider à déterminer la mise en page de notre site Web. De là, nous pouvons utiliser le sélecteur universel, `*`,ainsi que les pseudo-éléments universels, *:`before` et `*:after`, pour sélectionner tous les élément imaginable et changer le `box-sizing` à `border-box`. Rappelez vous, nous devons inclure tous les préfixes des fournisseurs nécessaires à la propriété `box-sizing`, car elle est une propriété relativement nouvelle.

```
/ *
  ========================================
  Grille
  ========================================
* /
*, *:before, *:after{
  -webkit-box-sizing: border-box;
     -moz-box-sizing: border-box;
          box-sizing: border-box;
}
```

Ensuite nous allons voulons créer une classe qui servira de conteneur pour nos éléments. Nous pouvons utiliser cette classe `container` sur différents éléments pour définir une, `largeur` commune, centrer les éléments sur la page et appliquer un `padding` horizontal commun.
Juste en dessous de notre ensemble de règles de sélections universelles, nous allons créer un sélecteur avec une classe `container`.Dans ce sélecteur nous allons définir notre `width` à `960` pixels, notre padding droite et gauche à `30` pixels,nos marges supérieure et inférieure à `0` et nos marges gauche et droite à `auto`.
Définir une largeur indique au navigateur la largeur de tout élément avec la classe `container`. En utilisant `auto` comme valeur pour les marges de gauche et de droite permet au navigateur de comprendre automatiquement que les marges seront égales à gauche et à droite pour l'élément, centrant ainsi le contenu sur la page. Enfin, le padding à gauche et droite assure que notre contenu ne sera pas directement coller sur le bord de l'élément et fournit un peu d'aération pour le contenu.

```
.container {
  margin: 0 auto;
  padding-left: 30px;
  padding-right: 30px;
  width: 960px;
}
```

Maintenant que nous avons une classe `container` , nous allons l'appliquer dans les éléments `<header>` et `<footer>` de chaque page, y compris le fichier index.html…

```
<header class="container">...</header>
...
<footer class="container">...</footer>
```

Pendant que y nous sommes, nous allons aller de l'avant et centrer le reste du contenu de nos pages. Sur la page d'accueil, notre fichier `index.html,` ajoutons la classe de `container` pour chaque élément `<section>` sur la page, un pour notre section de présentation (la section qui introduit notre cours) et une pour notre section d'appel.

```
<section class="container">...</section>
```

En outre, nous allons envelopper tous les `<h1>` éléments sur chaque page avec une `<section>` élément avec la classe `container`

```
<section class="container">
  <h1>...</h1>
</section>
```

Nous reviendrons ajuster ces éléments et ces classes plus tard, mais pour l'instant nous nous avançons dans la bonne direction.
Maintenant que l'ensemble de notre contenu est centré, nous allons créer un certain espacement vertical entre les éléments. Pour commencer nous allons placer une marge inférieure de `22` pixel sur des éléments titres et paragraphes. Nous allons placer et commenter ces styles typographiques ci-dessous nos styles de grille.

```
/ *
  ======================== ================
  Typographie
  ================================ ========
* /
h1, h2, h3, h4, h5, p {
  margin-bottom: 22px;
}
```

Nous avons volontairement sauté l'éléments `<h6>` parce nous ne l'utiliserons pas pour le moment.
Faisons aussi la création d'un bordure et de quelques coins arrondis. Nous allons commencer par placer un bouton dans l'élément `<section>` sur notre page d'accueil, situé juste en dessous de l'entête.
Auparavant nous avons ajouté un élément `<a>` dans cette `<section>`. Ajoutons les classes de `btn` et `btn-alt` à cette ancre.

```
<a class="btn btn-alt">...</a>
```

Maintenant nous allons créer quelques styles pour ces classes au sein de notre CSS. Ci dessous notre règle de typographie définies, nous allons créer une nouvelle section du fichier CSS pour les boutons.
Pour commencer nous allons ajouter la classe `btn` et appliquer des styles communs qui peuvent être partagées entre tous les boutons. Nous voulons que tous nos boutons aient un `border-radius de 5` pixel. Ils doivent être présentés comme des éléments `inline-block` afin que nous puissions ajouter du padding autour des quatre côtés sans problème. Nous supprimons toute marge.

```
/ *
  ============================ ============
  Boutons
  ====================================== ==
* /
.btn {
  border-radius: 5px;
  display: inline-block;
  margin: 0;
}
```

Nous voulons également inclure les styles spécifiques à ce bouton, que nous ferons en utilisant la classe `btn-alt.` Ici nous allons ajouter une bordure de `1`-pixel, solide et grise avec `10` pixels de padding en haut et en bas de la touche et `30` pixels de padding à gauche et droite du bouton.

```
.btn-alt {
  border: 1px solid #dfe2e5;
  padding: 10px 30px;
}
```

L'utilisation de deux classes `btn` et `btn-alt` sur le même élément `<a>` permet à ces styles de se superposer sur un seul élément.
Parce que nous travaillons sur la page d'accueil, ajoutons aussi un peu de `padding` à l'élément `<section>` qui contient notre élément `<a>` avec les classes de `btn` et `btn-alt`. Nous allons faire cela en ajoutant une classe `leader` à l'élément <section> , au côté de la valeur d'attribut de classe `container`, comme ce sera le premier de notre site Web.

```
<section class="container leader">
  ...
</section>
```

Ensuite nous allons créer une nouvelle règle au sein de notre fichier CSS pour la classe `leader` qui appliquera un `padding` des quatre côtés de l'élément sélectionné par la classe. Le padding sera en haut de 22 pixels, à droite de 80 pixels, en dessous de 66 pixels et à gauche de 80 pixels.

```
/ *
  ======================================== =
  Accueil
  ========================================
* /
.leader {
  padding: 22px 80px 66px 80px;
}
```

Le sélecteur universel

Dans la première étape de cet exercice nous avons utilisé le sélecteur universel "*". En CSS l'astérisque, `*,` est le sélecteur universel, qui sélectionne tous les éléments. Plutôt que d'énumérer tous les éléments imaginables, nous pouvons utiliser l'astérisque comme un fourre-tout pour sélectionner tous les éléments pour nous. Les pseudo-éléments `:before` et `:after` qui sont également mentionnés dans cette étape sont des éléments qui peuvent être générés dynamiquement avec CSS. On ne va pas faire utilisation ces éléments au sein de notre projet; Cependant, lorsque

utilisez le sélecteur universel il est une bonne pratique d'inclure également ces pseudo-éléments dans le cas où ils devraient apparaître.

Résumé

Apprendre toutes les différentes parties du modèle de boîte est pas mince exploit. Ces concepts, bien que brièvement présenté, prennent un peu de temps à être maîtrisés pleinement.
En bref, dans cette leçon nous avons parlé de ce qui suit:

- Comment les différents éléments sont affichés
- Ce qu’est le modèle de boîte et pourquoi il est important
- Comment changer la taille, y compris la hauteur et largeur, des éléments
- Comment ajouter les margin, padding, et borders aux éléments
- Comment changer la boîte, le dimensionnement des éléments et les effets de cela a sur le modèle de boîte

Maintenant que nous avons une meilleure compréhension de la façon dont les éléments sont affichés, on peut comprendre comment positionner ces éléments.