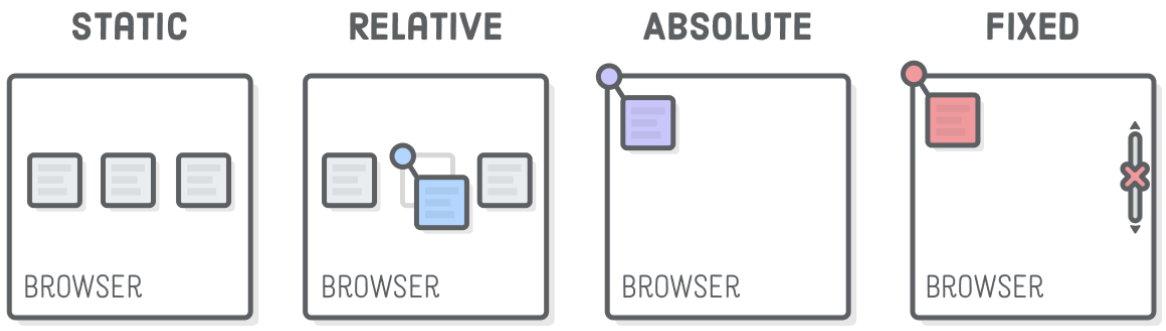
Positionnement avancé

Le "positionnement statique" se réfère au flux normal de la page. Le modèle de boîte CSS, les float, flexbox ou grid fonctionnent tous dans ce flux «static», mais ce n'est pas le seul système de positionnement disponible en CSS.



Les trois autres types de positionnement sont «relative», «absolute», et «fixed». Chacun d'entre eux vous permettent de positionner "manuellement" des éléments à l'aide des coordonnées spécifiques. Au lieu de dire "mets cette case dans le centre de son conteneur," le positionnement vous permet de dire des choses comme "Mettez cette case 20 pixels au-dessus et 50 pixels vers la droite de l'origine de son parent."

La grande majorité des éléments d’une page web doivent être positionnés en fonction du flux statique de la page. Les autres systèmes de positionnement entrent en jeu quand vous voulez faire des choses plus avancées comme modifier la position d'un élément particulier ou animer un composant d'interface utilisateur sans modifier les éléments environnants.

Nous allons commencer par l'examen des positionnements relatifs, absolu, et fixe, puis nous appliquerons tout ce que nous avons appris à un menu déroulant.

installer

Commencez par créer un nouveau dossier dans VS Code appelé “position” et ajouter les fichiers téléchargés ici http://b.link/css50.

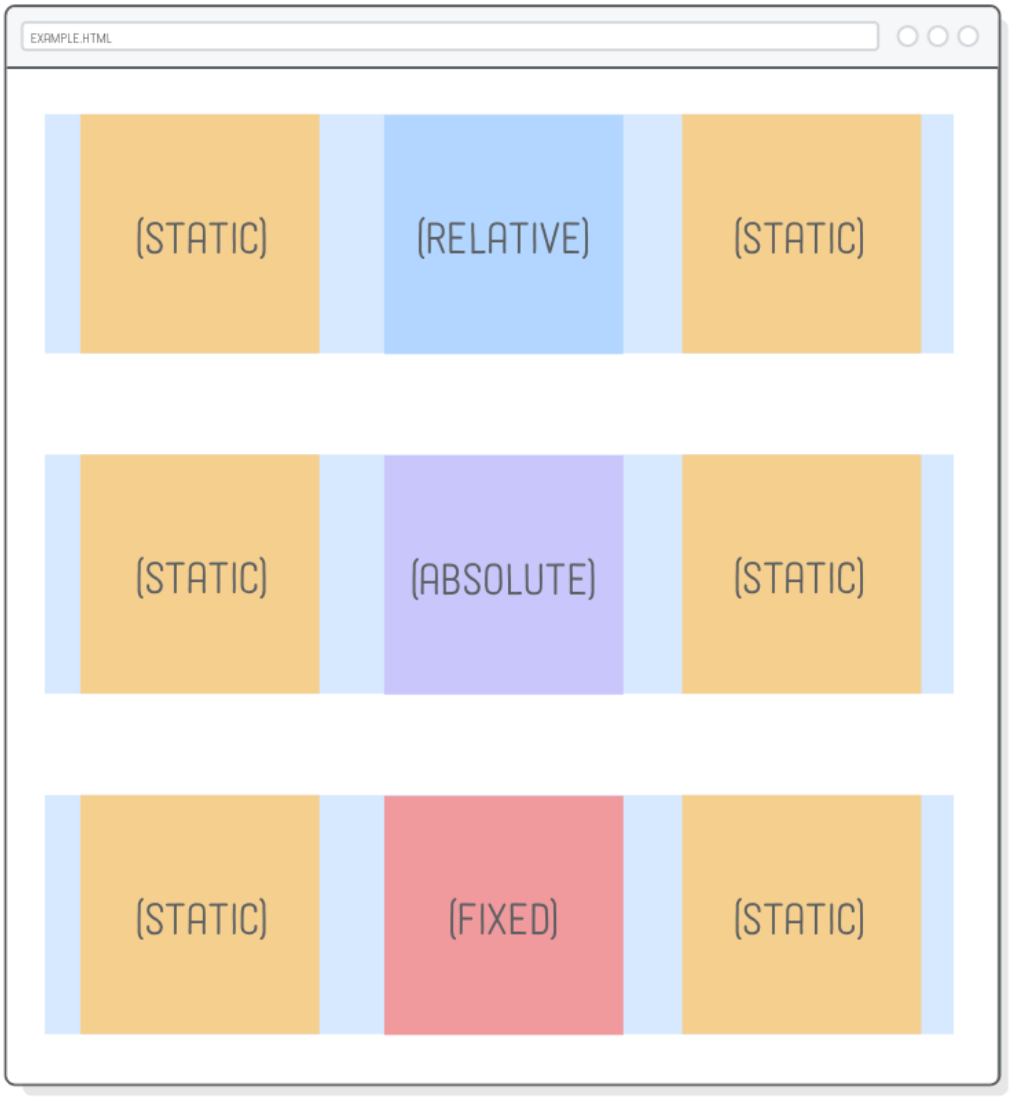
Nous avons trois exemples, tous avec la même structure de HTML. On constatera que la modification du comportement de positionnement à l'intérieur de chacun d'eux aura des effets radicalement différents.

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Positioning Is Easy!</title>
    <link href="styles.css" rel="stylesheet">
  </head>
  <body>
    <section class="container">
      <div class="example relative">
        <div class="item"><img src="images/static.svg" ></div>
        <div class="item item-relative"><img src="images/relative.svg" ></div>
        <div class="item"><img src="images/static.svg" ></div>
      </div>
    </section>
    <section class="container">
      <div class="example absolute">
        <div class="item"><img src="images/static.svg" ></div>
        <div class="item item-absolute"><img src="images/absolute.svg" ></div>
        <div class="item"><img src="images/static.svg" ></div>
      </div>
    </section>
    <section class="container">
      <div class="example fixed">
        <div class="item"><img src="images/static.svg" ></div>
        <div class="item item-fixed"><img src="images/fixed.svg" ></div>
        <div class="item"><img src="images/static.svg" ></div>
      </div>
    </section>
  </body>
```

Cette page repose sur des images et sur des styles contenus dans le fichier styles.css :

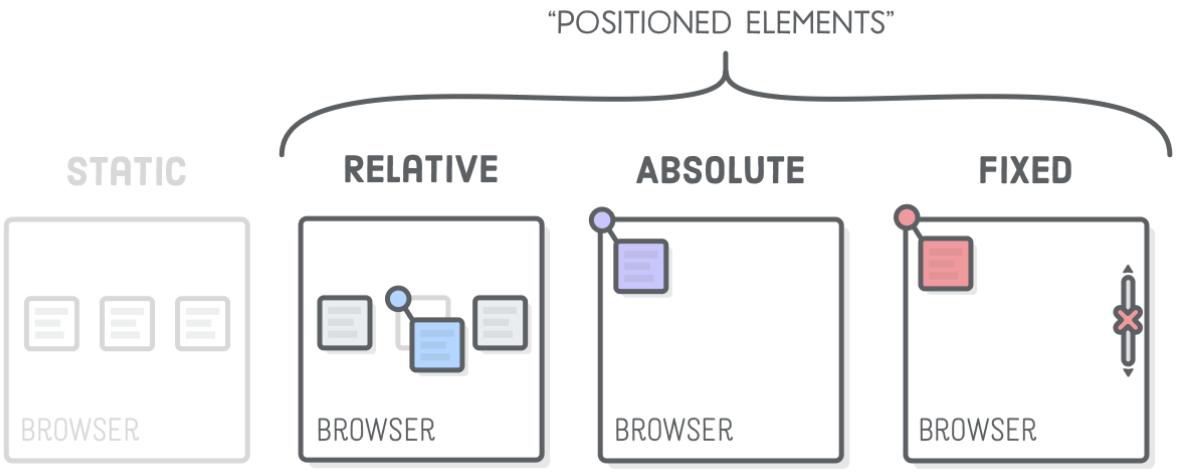
```css
* {  margin: 0;  padding: 0;  box-sizing: border-box; }
body {  height: 1200px;  }
.container {
  display: flex;
  justify-content: center;
}
.example {
  display: flex;
  justify-content: space-around;
  width: 800px;
  margin: 50px 0;
  background-color: #D6E9FE;
}
.item img {
  display: block;
}
```

Rien de nouveau ici, juste quelques techniques familières flexbox pour créer une grille d'articles. La seule chose bizarre est la hauteur explicite de l'élément <body>, qui nous permettra de faire défiler la page vers le haut et vers le bas pour montrer différents comportements de positionnement.



Eléments positionnés

La propriété `position` vous permet de modifier le schéma de positionnement d'un élément particulier. Sa valeur par défaut est static. Lorsque la propriété de position d'un élément n'a pas pour valeur static, il est appelée «élément positionné».



Il est possible de mélanger différents systèmes de positionnement. Encore une fois, la plupart de vos pages doivent être `static`, mais il est fréquent de trouver des éléments relativement et absolument positionnés à l'intérieur d'autres éléments qui font partie du flux normal de la page.

Positionnement relatif

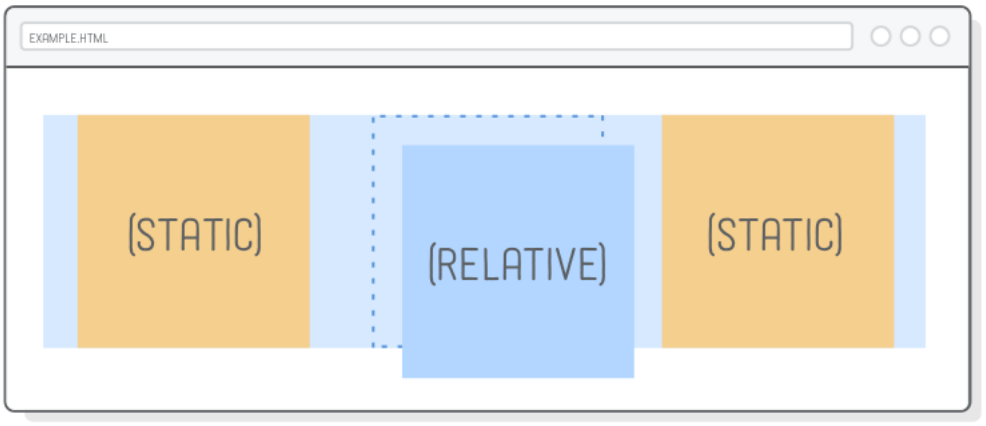
Le «positionnement relatif» déplace les éléments autour de l'endroit où ils apparaîtraient normalement dans le flux de la page. Cela est utile pour pousser les boîtes juste un peu hors du flux normal.

RELATIVE POSITIONING

Transformez l'élément .item-relatif dans le html en un élément positionné relativement. Ajoutez la règle suivante à styles.css:

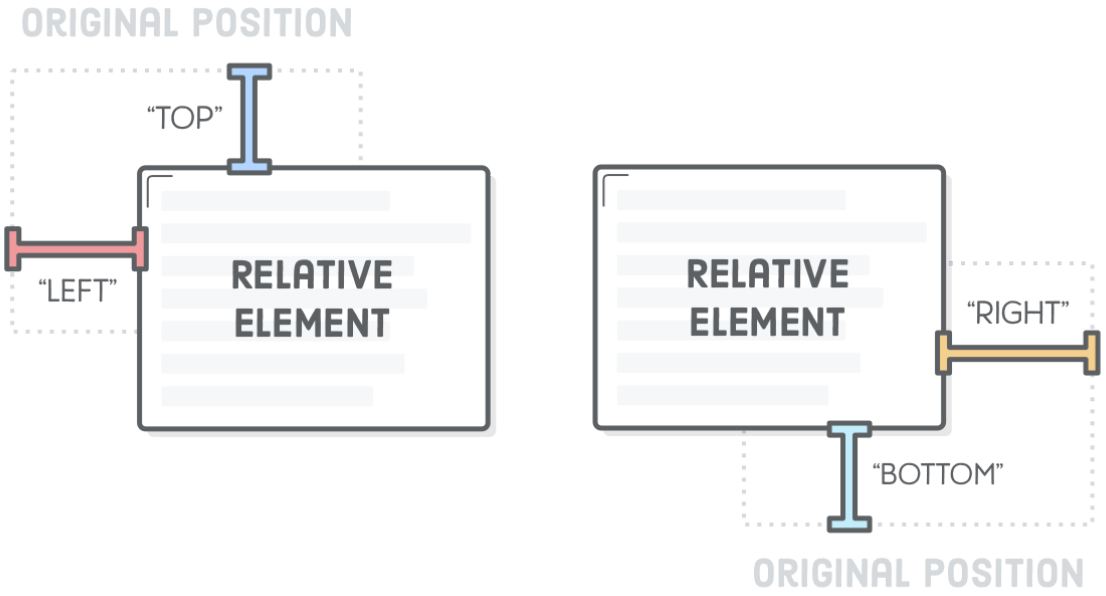
```
.item-relative {
  position: relative;
  top: 30px;
  left: 30px;
}
```

La ligne position: relative; en fait un élément positionné, et les propriétés top et left vous permettent de définir dans quelle mesure il est décalé par rapport à sa position static. Ceci est un peu comme la fixation d'un (x , y) pour coordonnées de l'élément.



Le positionnement relatif fonctionne de façon similaire à des marges, avec une différence très importante: ni les éléments qui l'entourent ou les élément parent ne sont affectés par les valeurs top et left. Tout le reste se comporte comme si .item-relative était dans sa position initiale. C'est comme si les décalages étaient appliqués après le navigateur est terminé la mise en page.

Les propriétés top et left se mesurent respectivement à partir des bords supérieur et gauche de la boîte d'origine. Nous pouvons décaler par rapport aux autres bords avec les propriétés bottom et right.



Par exemple, ce qui suit poussera la boîte dans la direction opposée:

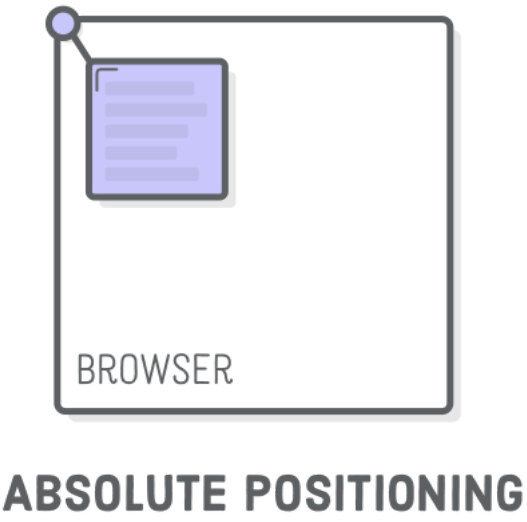
```
.item-relative {
  position: relative;
  bottom: 30px;
  right: 30px;
}
```

Notez que ces propriétés acceptent des valeurs négatives, ce qui signifie qu'il ya deux façons de spécifier le même décalage. Nous pourrions tout aussi bien utilisé `top: -30px;` à la place de la `bottom: 30px;` dans la déclaration ci-dessus.

Positionnement absolu

"Le positionnement absolu" est tout comme le positionnement relatif, mais le décalage est par rapport à la fenêtre du navigateur au lieu de la position initiale de l'élément. Comme il n'y a plus aucune relation avec le flux

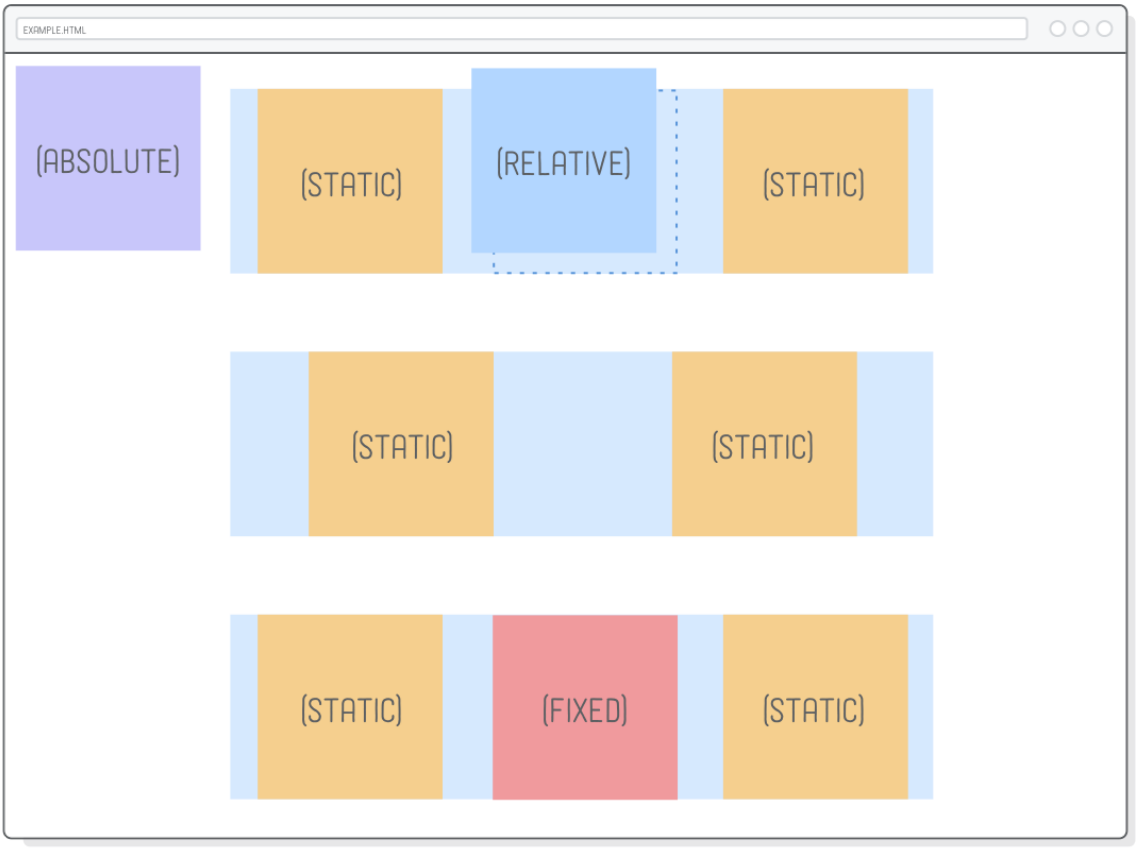
statique de la page, considérez ceci comme la manière la plus précise pour positionner un élément.



Jetons un coup d'oeil en ajoutant la règle suivante à notre feuille de style:

```
.item-absolute {
  position: absolute;
  top: 10px;
  left: 10px;
}
```

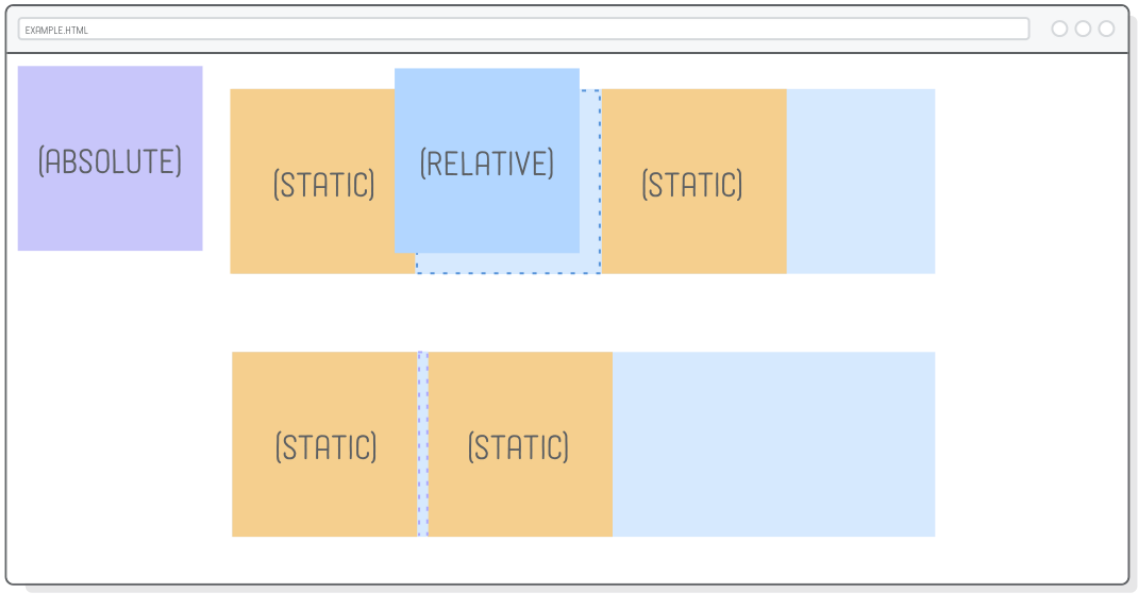
Notre structure HTML est exactement la même que dans l'exemple précédent, mais cela va coller l'image pourpre dans le coin supérieur gauche de la fenêtre du navigateur. Vous pouvez également essayer de définir une valeur pour bottom ou right pour obtenir une idée plus claire de ce qui se passe.



L'autre effet intéressant de absolute est qu'il supprime complètement un élément du flux normal de la page. Cela est plus facile de voir avec des éléments alignés à gauche, donc nous allons changer temporairement la propriété justify-content dans notre règle .example:

```
.example {
  display: flex;
  justify-content: flex-start;  /* Mettre à jour */
  /* ... */
}
```

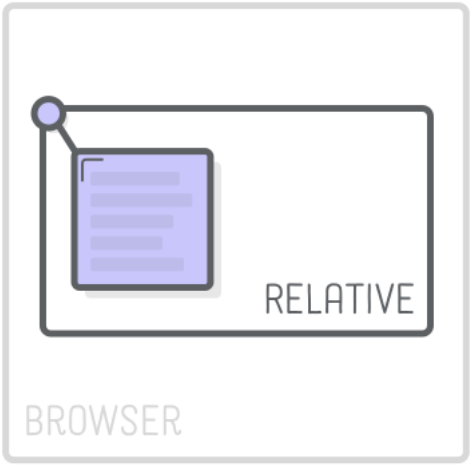
Dans notre exemple de positionnement relatif (la première ligne), il y a encore un espace où l'élément positionné utilisé pourrait être, mais avec un positionnement absolu, cet espace a disparu. Il est comme si .item-absolute n'existe même pas pour son parent et les éléments environnants. Assurez-vous de changer le justify-content de retour à space-around avant de passer à la suite.



Ce comportement est pas vraiment utile la plupart du temps parce que cela signifierait que tout sur votre page doit être placé de façon absolue, sans quoi nous obtiendrions des chevauchements imprévisibles des éléments static avec des éléments absolute. Alors, à quoi sert la position absolute ?

Positionnement absolu relativement

Le positionnement absolu devient beaucoup plus pratique quand elle est relative à un autre élément qui est dans le flux de la page. Heureusement, il y a un moyen de changer le système de coordonnées d'un élément en position absolue.

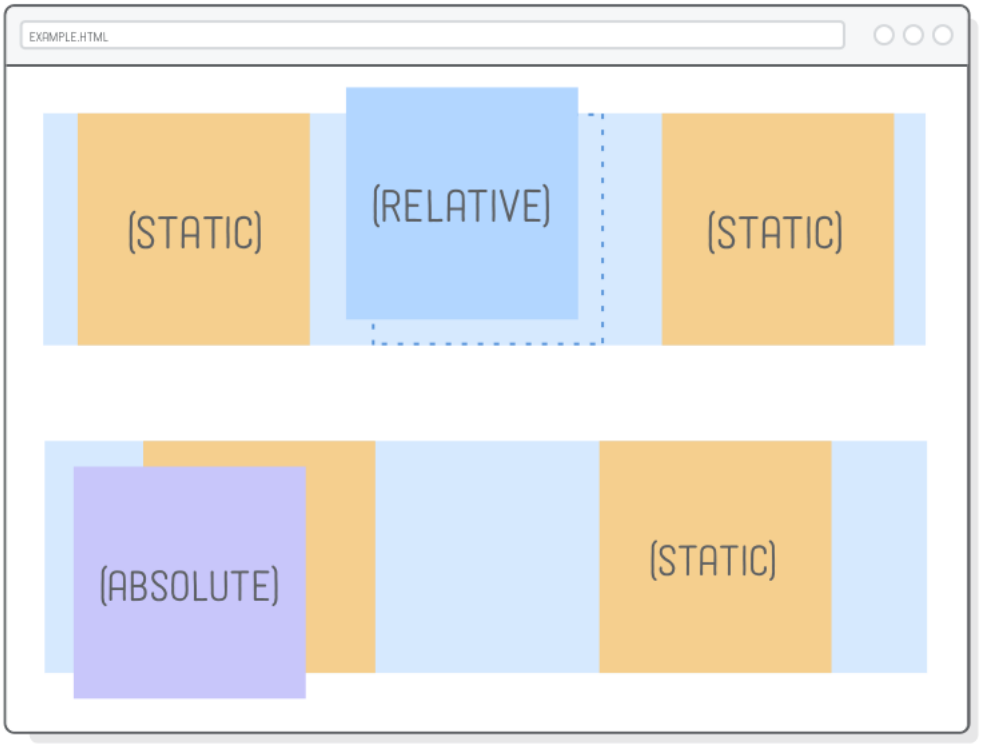


RELATIVELY ABSOLUTE POSITIONING

Les coordonnées des éléments absolus sont toujours par rapport au conteneur le plus proche qui est un élément positionné. Cela restera positionné par rapport au navigateur seuelemnt si aucun de ses parents n’est un élément positionné. Donc, si nous changeons l'élément parent de `.item-absolute` en position relative, il devrait apparaître dans le coin supérieur gauche de cet élément au lieu de la fenêtre du navigateur.

```
.absolute {
  position: relative;
}
```

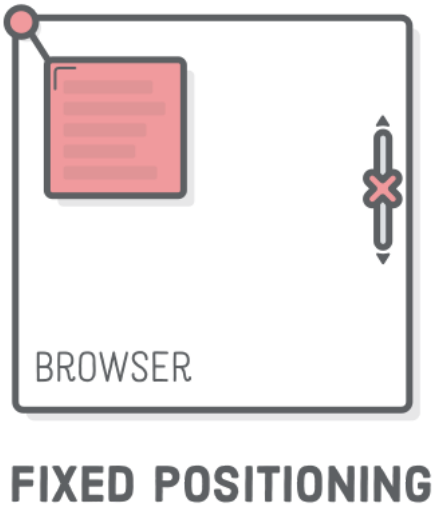
Le div `.absolute` est disposé dans le flux normal de la page, et nous pouvons manuellement déplacer notre `.item-absolute` partout où nous en avons besoin. C'est très bien, car si nous voulons modifier le flux normal du conteneur, disons, pour une disposition mobile, tous les éléments positionnés se déplaceront automatiquement avec elle.



Remarquez comment nous ne spécifions pas de coordonnées de décalage pour `.absolute`. Nous utilisons le positionnement relatif dans le seul but de laisser notre élément absolu revenir dans le flux normal de la page. C'est ainsi que nous combinons en toute sécurité un positionnement `absolute` avec un positionnement `static`.

Positionnement fixe

"Positionnement fixe" a beaucoup en commun avec le positionnement absolu: il est très “manuel”, l'élément est retiré de l'écoulement normal de la page, et le système de coordonnées est par rapport à la fenêtre du navigateur. La principale différence est que les éléments `fixed` ne défilent pas avec le reste de la page.



FIXED POSITIONING

Allez-y et mettre à jour notre troisième exemple pour utiliser le positionnement fixed:

```
.item-fixed {
  position: fixed;
  bottom: 0;
  right: 0;
}
```

Cela placera l'image rouge dans le coin inférieur droit de l'écran. Essayez de faire défiler la page, et vous découvrirez qu'il ne se déplace pas avec le reste des éléments de la page, tandis que l'image pourpre en position absolue fait.

Cela vous permet de créer par exemple des barres de navigation qui restent toujours à l'écran.

Eléments positionnés pour animation

Ceci est un peu hors de portée, puisque ce tutoriel est sur HTML et CSS, pas JavaScript. Cependant, l'animation est l'un des cas d'utilisation principal pour le positionnement relatif et absolu, donc nous allons jeter un coup d'oeil un peu dans le futur en animant un de nos éléments.

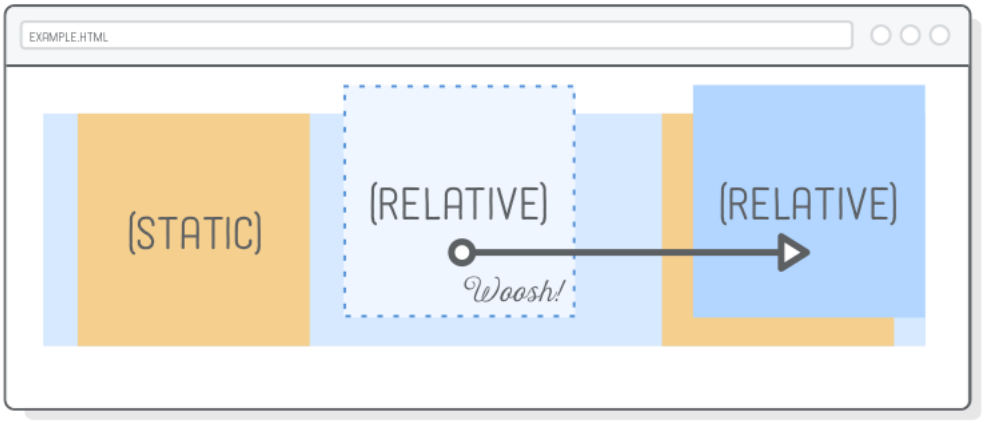
Ces systèmes de positionnement avancés permettent à JavaScript de déplacer des éléments tout en évitant toute interaction avec les éléments environnants. Dans votre fichier position.html juste avant l'élément <body>, l'élément <script> contient tous le code et nous allons supprimer les "/*" juste après la balise <script> et le "*/" avant la balise </script>.

```
<script>
  var left = 0;

  function frame() {
    var element = document.querySelector('.item-relative');
    left += 2;
    element.style.left = left + 'px';
    if (left >= 300) {
      clearInterval(id)
    }
  }

  var id = setInterval(frame, 10)
</script>
```

Ce code JavaScript crée une animation simple qui met à jour en permanence la propriété left de la classe .item-relative. Lorsque vous rechargez la page, vous devriez voir l'image float bleu sur le bord droit de son conteneur.



C'est un exemple assez rudimentaire, mais vous pouvez espérer voir comment il est applicable aux sympathiques animations d'UI -pour User Interface-. Si vous essayez d'obtenir le même effet en manipulant les propriétés de marge ou de padding, vous déplacerez les cases positionnées statiquement et / ou aussi l'élément contenant .example.

Eléments positionnés pour menus

Voila toutes les techniques de positionnement. Faisons quelque chose d'avancé avec elles ! Le reste de ce chapitre applique nos nouvelles compétences pour réaliser un menu de navigation so cool avec un menu déroulant interactif pour l' un de ses liens. Nous allons construire cette page à partir de zéro.

Le positionnement fixe nous permettra de mettre le menu en haut de la page, et le positionnement relatif nous servire juste “d’ancre” pour le menu déroulant qui sera lui positionné en absolu. Nous aurons également l'occasion de parler des meilleures pratiques du menu de navigation et de voir quelques applications pratiques des pseudo-classes.

Pour commencer, nous avons besoin d'une nouvelle page web appelée menu.html qui a un entête et un menu simple:

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Menu !</title>
    <link href="styles.css" rel="stylesheet">
  </head>
  <body>
    <header class="header">
      <div class="logo"><img src="images/awesome-logo.svg"></div>
      <nav class="menu">
       <ul class="menu">
        <li class="dropdown"><span>Menu &#9662;</span></li>
        <li><a href="#">Blog</a></li>
        <li><a href="#">Souscrire</a></li>
        <li><a href="#">A propos</a></li>
       </ul>
      </nav>
    </header>
  </body>
</html>
```

Les menus de navigation devraient presque toujours être marqués comme une liste <ul> au lieu d'un groupe d'éléments . Ces sémantiques rendent la navigation de votre site beaucoup plus accessible aux moteurs de recherche. Notez également comment nous nous préparons pour notre menu déroulant en ajoutant un attribut de classe au premier <li> dans la

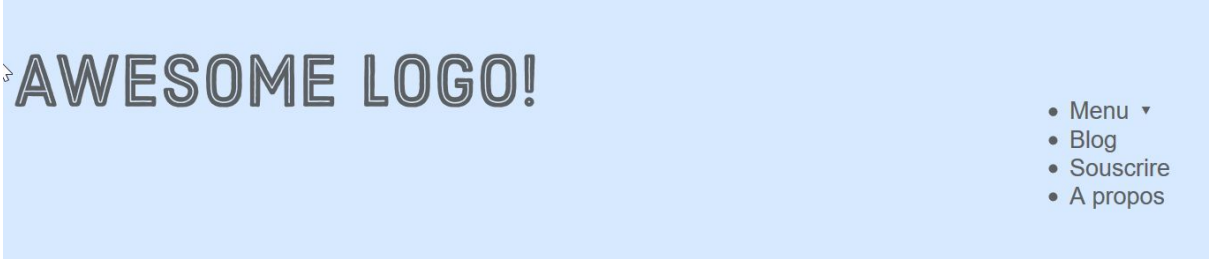
liste. Ce nous permettra de différencier l'étiquette du sous-menu qu'elle révèle.

Ensuite, nous avons besoin de modifier notre feuille de style:

```
body {
  height: 1200px;
  font-size: 18px;
  font-family: sans-serif;
  color: #5D6063;
}
a:link,a:visited {
  color: #5D6063;
  text-decoration: none;
}
a:hover {
  text-decoration: underline;
}

.header {
  position: fixed;
  display: flex;
  justify-content: space-between;
  width: 100%;
  padding: 50px;
  background: #D6E9FE;
}
```

Tout cela devrait être familier, mais il faut noter la position fixed de la .header, ce qui maintient notre menu de navigation en haut de tout contenu qui irait dans la page.



Menu en ligne

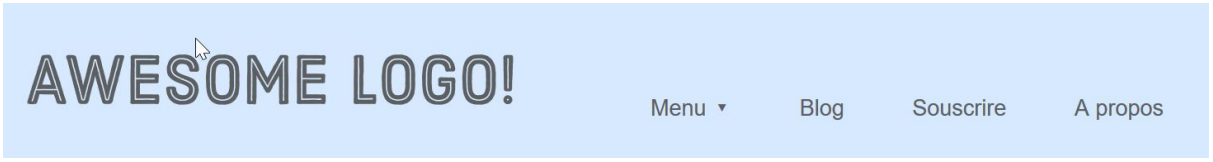
En dépit d' être balisé comme des listes non ordonnées, les menus de navigation pour la plupart des sites Web ne ressemblent pas réellement à une liste. Nous pouvons résoudre ce problème en créant les zones de liste en ligne à la place des zones de bloc via la propriété display. Ajoutez ce qui suit à style.css:

```
.menu {
  margin-top: 15px;
}

.menu > li {
  display: inline;
  margin-right: 50px;
}

.menu > li:last-of-type {
  margin-right: 0;
}
```

Nous devons utiliser les sélecteurs d'enfants ici au lieu de sélecteurs descendants parce que nous voulons que sélectionner les éléments <li> qui sont directement à l'intérieur du .menu. Cela va devenir important une fois que nous ajouterons notre sous-menu, qui a ses propres éléments <li> et que nous ne voulons pas le style hérités de cette règle. On ajoute également des marges à tous les éléments de la liste, mais on les suppriment du dernier élément <li> en utilisant la pseudo-classe :last-of-type. Cette technique est assez classique pour créer des marges entre les éléments.



Sous menus

Notre sous-menu va ressembler au menu de niveau supérieur, sauf que tout sera imbriqué dans un élément de liste.

Modifiez l'élément .menu pour qu'il corresponde à ce qui suit, en veillant à ce que la liste entière .features-menu soit enveloppée dans le premier <li> de l'élément .menu.

```
<ul class='menu'>
  <li class='dropdown'><span>Menu &#9662;</span>
    <ul class='features-menu'>  <!-- Début du sous menu -->
      <li><a href='#'>Accueil</a></li>
      <li><a href='#'>Info</a></li>
      <li><a href='#'>Contact</a></li>

    </ul>                          <!-- Fin du sous menu -->
  </li>
  <li><a href='#'>Blog</a></li>
  <li><a href='#'>Souscrire</a></li>
  <li><a href='#'>À propos</a></li>
</ul>
```

En ce qui concerne le CSS, nous traiterons de la partie déroulante interactive plus tard. Ajoutez quelques styles simples afin que nous puissions voir la boîte que nous essayons de positionner:

```
.features-menu {
  display: flex;
  flex-direction: column;
  background: #B2D6FF;
  border-radius: 5px;
  padding-top: 60px;
}

.features-menu li {
  list-style: none;
  border-bottom: 1px solid #FFF;

  padding: 0 40px 10px 20px;
  margin: 10px;
}

.features-menu li:last-of-type {
```

```
    border-bottom: none;
  }
```

Le sous-menu lui-même est correctement stylé, mais il apparaît à la mauvaise place et pousse le reste de nos éléments de menu de niveau supérieur. Cela devrait être prévu car il est toujours positionné statiquement, ce qui signifie qu'il interagit toujours avec son parent et les éléments environnants.

Pour créer la disposition souhaitée, nous avons besoin de faire appel à nos nouvelles compétences de positionnement CSS.

sous menus absolute relativement

Nous voulons que nos autres éléments de menu de niveau supérieur s'affichent comme avant l'ajout du sous-menu, comme si le sous-menu n'était pas là. Attendez une seconde ... c'est le comportement exact des éléments absolument positionnés. Ajoutez quelques lignes à la règle .features-menu:

```
  .features-menu {
    display: flex;
    flex-direction: column;
    background: #B2D6FF;
    border-radius: 5px;
```

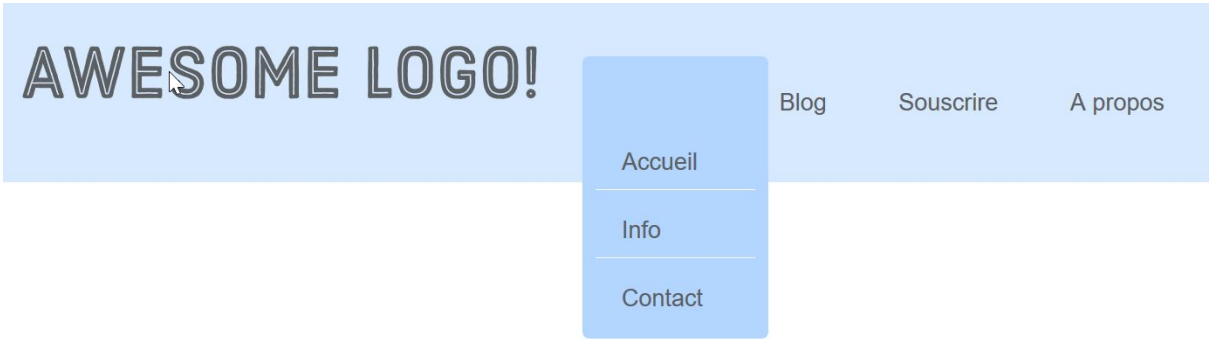
```
    padding-top: 60px;
    position: absolute;     /* Magie */
    top: -25px;
    left: -30px;
  }
```

Le sous-menu ne fait plus partie du flux statique de la page, donc nos éléments de menu de niveau supérieur sont de retour à la normale. Toutefois, le sous-menu doit apparaître sous l'étiquette des fonctionnalités - pas dans le coin de la fenêtre du navigateur. Quelle coïncidence ... nous venons d'apprendre comment faire ça !

Le sous-menu réside dans <li class='dropdown'>. Le transformer en un élément positionné devrait changer le système de coordonnées utilisé par notre positionnement absolu .features-menu:

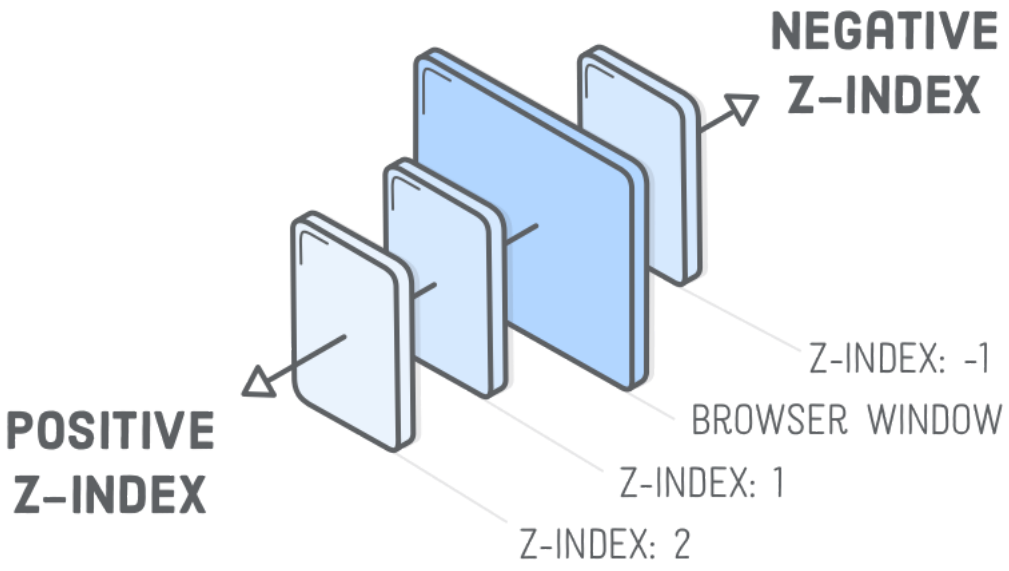
```
.dropdown {
  position: relative;
}
```

Ok, problème suivant. Notre sous-menu est au bon endroit, mais maintenant il recouvre le libellé d'entête de notre menu.



Z-index

Nous avons déja eu à traiter des problèmes de «profondeur» avant, mais révisons un peu. Jusqu'à présent, tous nos éléments HTML étaient affichés au-dessus ou au-dessous les uns des autres de manière intuitive. La propriété z-index vous permet de contrôler la profondeur des éléments sur la page. Si vous pensez de votre écran comme un espace 3D, les valeurs négatives z-index vont plus loin dans la page, et les positives sortent de la page.



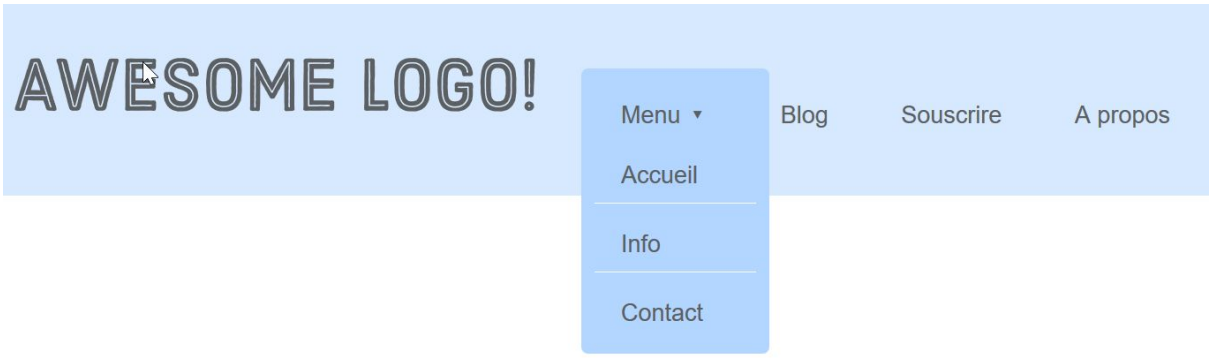
En d'autres termes, l'élément .features-menu doit avoir un z-index plus faible que le libellé d'entête. La valeur par défaut de z-index est 0, donc nous allons mettre aux deux éléments des valeurs supérieures. Nous avons inclu le libellé de l'entête dans un , nous permettant ainsi de le styler via un sélecteur enfant:

```
.dropdown > span {
  z-index: 2;
  position: relative;  /* Important! */
  cursor: pointer;
}

.features-menu {
```

```
    /* ... */
    z-index: 1;
  }
```

Le libellé de l'entête doit maintenant apparaître au-dessus du sous-menu. Prenez note de cette ligne position: relative;. Elle est **nécessaire** parce que seuls les éléments positionnés peuvent sont sensible à leur propriété z-index. C'est facile à oublier, alors notez le bien pour la prochaine fois que vous aurez des problèmes de profondeur et que vos règles CSS ne semblent pas avoir d'effet.



Nous avons ajouté la propriété cursor pour faire ressembler la souris à une main, donc faire l'effet d'un lien, lorsque l'utilisateur survole l'entrée de menu.

les pseudo-classes pour des menus déroulants

Bien! Sous-menu fait! Notre tâche finale est de le cacher jusqu'à ce que l'utilisateur passe dessus. Rappelez-vous que la pseudo classe :hover… Nous pouvons utiliser cela pour transformer notre sous-menu en un menu déroulant interactif.

Tout d'abord, nous devons modifier notre règle de menu existante .features pour afficher uniquement le sous-menu lorsque l'utilisateur passe dessus en ajoutant un sélecteur descendant hover. Mettez à jour le sélecteur de menu .features pour qu'il corresponde à ce qui suit:

```
.dropdown:hover .features-menu {
  display: flex;
  flex-direction: column;
  background: #B2D6FF;
  /* ... */
}
```

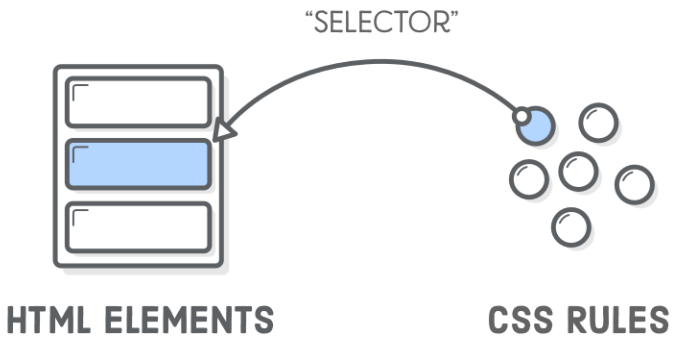
Ensuite, nous avons besoin de se cacher d'abord le sous-menu en utilisant la propriété display. Ajouter une nouvelle règle à la css:

```
.features-menu {              /* Ajouter cette règle */
  display: none;
}
```

Le réglage de display à none rend fait disparaître notre élément. En remplaçant cette valeur par flex dans la règle :hover, nous demandons au navigateur de réafficher le menu .features. Cette combinaison intelligente de sélecteurs descendants et de pseudo-classes nous permet de cacher ou de montrer un élément.

Sélecteurs complexes

Les sélecteurs sont l'un des, sinon, la partie la plus importante de la CSS. Ils façonnent la cascade et déterminent comment les styles doivent être appliqués à des éléments sur une page.



Jusqu'à récemment, CSS n'avait jamais vraiment bougé sur les sélecteurs. De temps en temps il y avait des mises à jour incrémentielles au sein de la spécification des sélecteurs, mais jamais des améliorations majeures. Heureusement, une plus grande attention a été accordée aux sélecteurs, un nouveau regard à été porté sur la façon de sélectionner différents types d'éléments ou des éléments dans différents états d'utilisation.

CSS3 a apporté de nouveaux sélecteurs, l'ouverture d'un tout nouveau monde de possibilités et des améliorations aux pratiques existantes. Nous allons voir ces sélecteurs, anciens et nouveaux, et nous allons discuter de la meilleure façon de les mettre à profit.

installer

Commencez par créer un nouveau dossier appelé select et ajouter les fichiers styles.css et selecteur.html.

Les sélecteurs communs

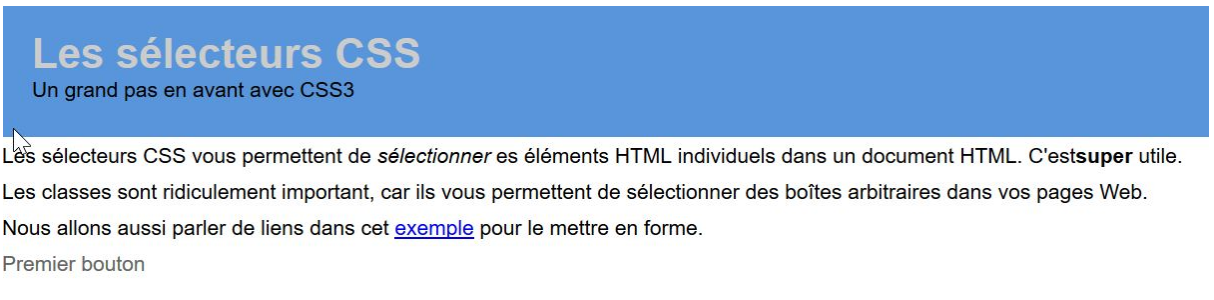
Avant de plonger trop profondément dans certains des sélecteurs plus complexes, nous allons jeter un coup d'œil sur quelques-uns des sélecteurs les plus fréquemment utilisé aujourd'hui. Ces sélecteurs comprennent le type, la classe et l'identifiant.

Le sélecteur de *type* identifie un élément en fonction de son type, spécifiquement comment cet élément est déclaré dans le HTML. Le sélecteur de *classe* identifie un élément basé sur sa valeur d'attribut de classe, qui peut être réutilisé sur plusieurs éléments si nécessaire pour aider à partager des styles populaires. Enfin, le sélecteur *id* identifie un élément en fonction de sa valeur d'attribut id, qui est unique et ne doit être utilisé qu'une seule fois par page.
Pour exemple créer le code HTML suivant.

```
<body>
  <header id="main">
    <h1>Les sélecteurs</h1>
    <p class="subtitle">Un grand pas en avant avec CSS3</p>
  </header>
</body>
```

Puis modifier le CSS

```
h1{   color: #000; }
#main{  color: #fff; background-color: #5995DA;   padding: 20px; }
.subtitle {   color: #ccc; }
```



Sélecteurs d'enfants

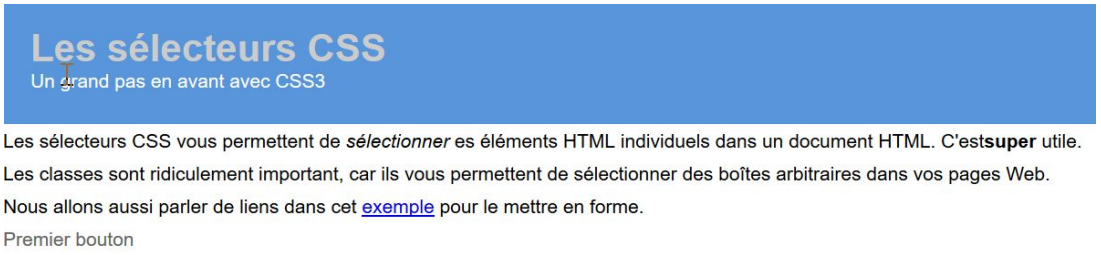
Les sélecteurs d'enfants fournissent un moyen de sélectionner les éléments imbriqués dans un autre, les rendant ainsi enfants de leur élément parent. Ces sélections peuvent être faites de deux façons différentes, en utilisant soit les sélecteurs descendants ou les sélecteurs enfants directs.

Sélecteur Descendant

Le plus commun des sélecteurs de l'enfant est le sélecteur descendant, ce qui correspond à chaque élément qui suit un ancêtre identifié. L'élément descendant n'a pas à venir directement après l'élément ancêtre dans l'arborescence du document, comme une relation parent-enfant, mais peut se trouver partout dans l'élément ancêtre. Les sélecteurs descendants sont créés par un espace entre des éléments dans un sélecteur.

```
header p{   color: #DEDEDE; }
```

Le sélecteur `header p` est un sélecteur descendant sélectionnant uniquement les `p` qui se situent à l'intérieur d'un `header`.
Indication, peut importe où se trouve l'élément `p`, tant qu'il se situe dans l'élément `header`, il sera toujours sélectionné. En outre, tout `p` extérieur de l'élément `header` n'a pas été sélectionné.



Sélecteur direct de l'enfant

Parfois les sélecteurs descendants vont un peu trop loin, plus que prévu. Parfois, seuls les descendants directs d'un élément parent doivent être sélectionnés et non pas tous les éléments imbriqués profondément à l'intérieur d'un ancêtre.

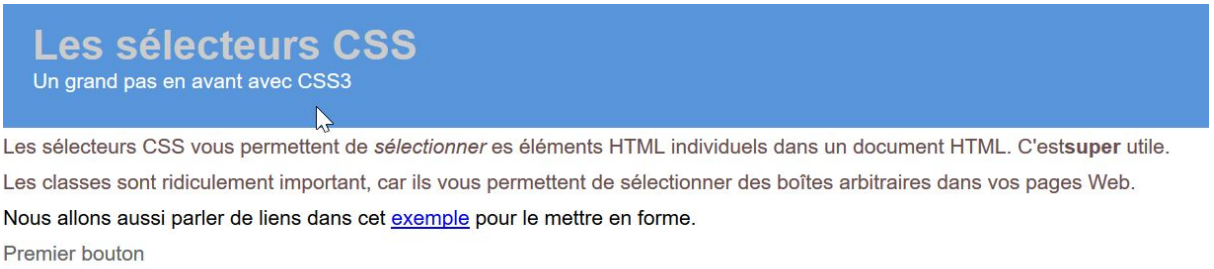
Dans ce cas, le sélecteur d'enfant direct peut être utilisé en plaçant un signe supérieur, >, entre l'élément parent et élément enfant dans le sélecteur.

```
<section>
 <p>...</p>
 <p>...</p>
 <div>
  <p>Nous allons aussi parler de liens dans cet
   <a href="https://google.com">exemple</a> pour le mettre en forme.
  </p>
 </div>
  <div>Bouton un</div>
</section>
```

Puis modifier le CSS

```
section > p{   color: #6c4e4e; }
```

Par exemple, section > p est un sélecteur d'enfant direct qui va seulement identifier les éléments p qui sont directement dans une section. Tout élément p placé à l'extérieur d'un élément section, ou imbriqué dans un autre élément autre que l'élément section ne sera pas sélectionné.



Sélecteurs de fratrie

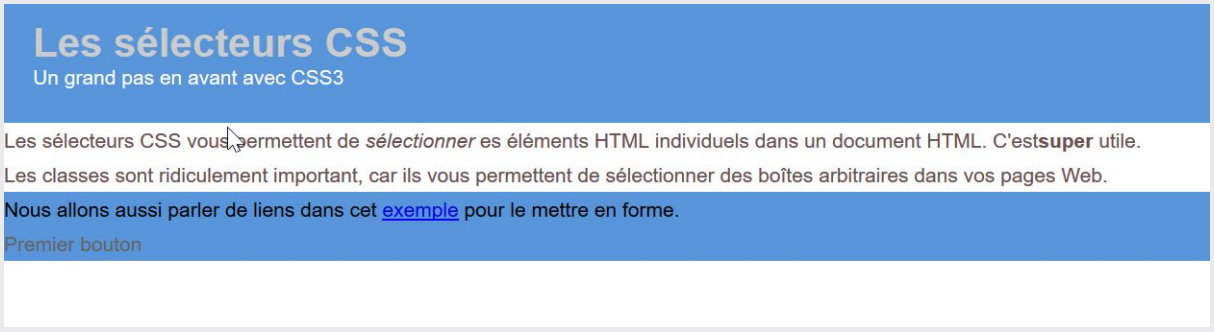
Savoir comment sélectionner les enfants d'un élément est largement bénéfique, et assez souvent vu. Cependant les éléments frères et sœurs, qui partagent un parent commun, peuvent également être sélectionnés. Ces sélections frères et sœurs peuvent être faites par le biais de la sélection générale de fratrie et par le sélecteurs d'enfants adjacents.

Sélecteur général de fratrie

Le sélecteur général de fratrie permet de sélectionner les éléments frères et sœurs, ceux qui partagent le même parent commun. Ils sont créés en utilisant le caractère tilde, `~`, entre deux éléments dans un sélecteur. Le premier élément identifie ce que le deuxième élément est un frère et les deux doivent partager le même parent.

Le sélecteur `p~div` est un sélecteur de fratrie général qui cherche des éléments `div` qui suivent, et partagent le même parent, de tous les éléments `p`. Pour qu'un élément `div` soit sélectionné, il doit venir après tout élément `p`.

Les div suivant le dernier paragraphe sont sélectionnés comme ils viennent après la position à l'intérieur de l'arbre du document et partagent le même parent que leur tête de frère.



Sélecteur de frère adjacent

Parfois, un peu plus de contrôle peut être souhaité, y compris la possibilité de sélectionner un élément frère qui suit directement après l'autre élément frère, c'est l'endroit où l'élément frère adjacent entre en jeu. Le sélecteur de

frère adjacent ne sélectionne des éléments frères qui sont directement après un autre élément frère. Au lieu d'utiliser le caractère tilde, comme avec les sélecteurs généraux de frères et sœurs, le sélecteur de frère adjacent utilise le caractère + entre les deux éléments dans un sélecteur. Encore une fois, le premier élément identifie ce que le second élément doit suivre immédiatement après l'élément, être un frère et les deux doivent partager le même parent.

Pour illustrer cela créons un menu de type "burger", que l'on va pouvoir ouvrir et fermer en cliquant sur une case à cocher transformée en bouton… Mais encore…

```
<input type="checkbox" id="toggle">
<label for="toggle">&#9776;</label>
<h1>Les sélecteurs</h1>
<nav>
  <ul>
    <li><a href="#">Accueil</a></li>
    <li><a href="#">Infos</a></li>
    <li><a href="#">Services</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</nav>
```

Puis modifier le CSS

```
input {                          /* Important */
  display: none;
}
label, ul {
  border: 1px solid #cecfd5;
  border-radius: 6px;
}
label {
  color: #0087cc;
  cursor: pointer;
  display: inline-block;
```

```
  font-size: 18px;
  padding: 5px 9px;
  float: left;
  margin-right: 15px;
}
label:hover {
  color: #ff7b29;
}
input:checked + label {     /* Important */
  box-shadow: inset 0 1px 2px rgba(0, 0, 0, 0.15);
  color: #9799a7;
}
nav {                                    /* Important */
  max-height: 0;
  overflow: hidden;
}
input:checked ~ nav {      /* Important */
  max-height: 500px;
}
ul {
  list-style: none;
  margin: 15px 0 0 0;
  padding: 0;
  width: 300px;
}
li {
  border-bottom: 1px solid #cecfd5;
}
li:last-child {
  border-bottom: 0;
}
a {
  color: #0087cc;
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
a:hover {
  color: #ff7b29;
}
```

Sélection par attribut

Maintenant les éléments peuvent être choisis en fonction de la présence d'un attribut et de ce que sa valeur peut contenir.

Sélection d'un attribut

Le premier sélecteur d'attribut identifie l'élément est basé sur si la présence de l'attribut ou non, indépendamment de toute valeur réelle. Pour sélectionner un élément en fonction de la présence d'un attribut ou pas il faut inclure simplement le nom de l'attribut entre crochets `[]`, dans un sélecteur. Les crochets peuvent ou ne peuvent pas suivre un qualificatif comme un type d'élément ou de classe, tout en fonction du niveau de spécificité souhaitée.

```
CSS         a[target] {...}

HTML      <a href="#" target="_blank">...</a>
```

La valeur d'attribut est égale à...

Pour identifier un élément avec une valeur d'attribut spécifique et exacte, le même sélecteur peut être utilisé, mais cette fois à l'intérieur des crochets qui suivent le nom de l'attribut, on va inclure la valeur correspondante souhaitée. À l'intérieur des crochets doit être inscrit le nom de l'attribut suivi d'un signe égal, `=`, des quotes, `""`, et à l'intérieur des quotes devrait être la valeur d'attribut correspondant désiré.

```
CSS         a[target="_blank"] {...}

HTML      <a href="#" target="_blank">...</a>
```

La valeur d'attribut contient...

Lorsque vous cherchez à trouver un élément basé sur une partie d'une valeur d'attribut, mais pas une correspondance exacte, le caractère astérisque, `*` peut être utilisé dans les crochets d'un sélecteur. L'astérisque

devrait être ajouté soit juste après le nom d'attribut, soit juste avant le signe égal. Cela signifie que la valeur à suivre doit apparaître ou être contenu, dans la valeur d'attribut.

```
CSS         a[href*="login"] {...}

HTML      <a href="/login.php" target="_blank">...</a>
```

La valeur d’attribut commence par...

En plus de choisir un élément basé sur une valeur d'attribut contenue dans une valeur déclarée, il est également possible de sélectionner un élément basé sur la valeur d'attribut qui “commence par”. L'utilisation d'un accent circonflexe, ^, dans les crochets d'un sélecteur entre le nom d'attribut et signe égal indique que la valeur d'attribut doit commencer par la valeur indiquée.

```
CSS     a[href^="https://"] {...}

HTML    <a href="https://mabanque.fr" target="_blank">...</a>
```

La valeur d’attribut termine par...

En face du commence avec le sélecteur, il y a aussi une extrémités avec sélecteur d'attribut. Au lieu d'utiliser l'accent circonflexe, les extrémités avec sélecteur d'attribut utilise le signe du dollar, $, dans les crochets d'un sélecteur entre le nom d'attribut et signe égal. En utilisant le signe dollar indique que la valeur de l'attribut doit se terminer par la valeur indiquée.

```
CSS         a[href$=".pdf"] {...}

HTML      <a href="/docs/menu.pdf" target="_blank">...</a>
```

Une valeur d’attribut parmi plusieurs...

Parfois , les valeurs d'attributs peuvent être espacées, dans lesquels un seul des mots doit être visé afin de faire une sélection. Dans ce cas , en utilisant le caractère tilde, ~, dans les crochets d'un sélecteur entre le nom d'attribut et signe égal indique une valeur d'attribut qui devrait être séparés par des espaces avec un seul mot correspondant à la valeur exacte indiquée.

```
CSS       a[rel~="tag"] {...}

HTML      <a href="#" rel="tag nofollow">...</a>
```

Une valeur d'attribut avec un tiret...

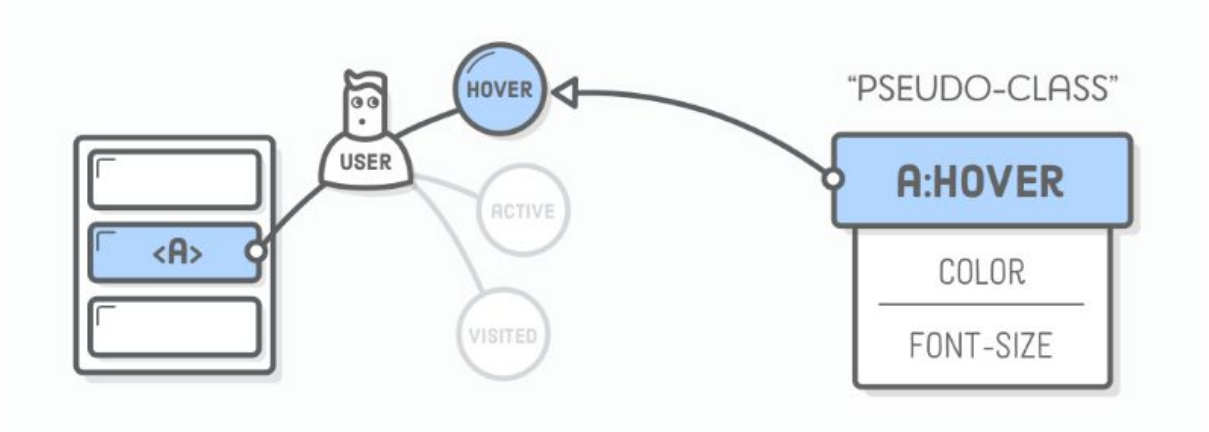
Quand une valeur d'attribut est séparée par un trait d'union, plutôt que séparés par des espaces, le caractère de la ligne verticale | , peut être utilisés dans les crochets d'un sélecteur entre le nom d'attribut et le signe égal. La ligne verticale indique que la valeur d'attribut peut être un trait d'union mais les mots séparés par le trait d'union doivent commencer par la valeur indiquée.

```
CSS       a[lang|="fr"] {...}

HTML      <a href="#" lang="fr-FR">...</a>
```

Pseudo-classes

Les pseudo-classes sont similaires à des classes régulières au format HTML mais elles ne sont pas directement écrites au sein du balisage, mais sont générées dynamiquement en raison des actions des utilisateurs ou de la structure du document. La pseudo-classe la plus courante, et celle que vous avez probablement vu auparavant, est :hover. Remarquez comment cette pseudo-classe commence par le caractère :.



Pseudo-classes

Les pseudo-classes commencent par deux points suivis du nom de la classe désirée. Les pseudo-classes de lien les plus courantes sont les suivantes:

- :link - Un lien que l'utilisateur n'a jamais visité.
- :visited - Un lien que l'utilisateur a déjà visité.
- :hover - Un lien avec la souris de l'utilisateur.
- :active - Un lien appuyé par une souris (ou un doigt).

Jetons un coup d'oeil à tous ces en ajoutant les règles suivantes à notre feuille de style CSS :

```
a:link {
  color: blue;
  text-decoration: none;
}
a:visited {
  color: purple;
}
a:hover {
  color: aqua;
  text-decoration: underline;
}
a:active {
  color: red;
}
```

Si vous n'avez jamais été à la page d'accueil du site, vous devriez voir un lien bleu. Sinon, vous verrez un lien violet. Lorsque vous survolez le lien, il devient bleu aqua, et quand vous poussez vers le bas sur elle, il deviendra rouge.

Action de l'utilisateur et pseudo-classes

Sur la base d'une action de l'utilisateur, différentes pseudo-classes peuvent être appliquées dynamiquement à un élément, dont les pseudo-classes :hover, :active et :focus. La pseudo-classe :hover est appliquée à un élément lorsqu'un utilisateur déplace son curseur sur l'élément, c'est le plus couramment utilisé avec les éléments d'ancrage. La pseudo-classe :active est appliquée à un élément lorsqu'un utilisateur sélectionne un élément, par exemple en cliquant sur un élément. Enfin, la pseudo-classe :focus est appliquée à un élément quand un utilisateur arrive sur un élément dans la page, souvent en utilisant le clavier pour tabuler d'un élément à un autre.

État d'interface utilisateur pseudo-classes

Il y a aussi des pseudo-classes générées autour de l'état des éléments de l'interface utilisateur, en particulier au sein des éléments de formulaire. Ces états de l'élément d'interface utilisateur comprennent des pseudo-classes :enabled, :disabled, :checked et :indeterminate.

La pseudo-classe :enabled sélectionne une entrée qui est dans l'état actif par défaut et disponible pour une utilisation, alors que la pseudo-classe :disabled sélectionne une entrée qui a l'attribut disabled. De nombreux navigateurs par défaut grisent les entrées désactivées pour informer les utilisateurs que l'entrée n'est pas disponible, mais ces styles peuvent être ajustés comme souhaité avec la pseudo-classe :disabled.

Les deux derniers interface utilisateur Etat élément pseudo-classes de :checked et :indeterminate tournent autour des éléments d'entrée checkbox et boutons radio. La pseudo-classe :checked sélectionne les cases à cocher ou des boutons radio qui sont, comme vous pouvez vous attendre, cochés. Quand un bouton de case à cocher ou la radio n'a été ni sélectionné ou est non sélectionné , il est dans un état indéterminé, à partir de laquelle la pseudo-classe :indeterminate peut être utilisée pour cibler ces éléments.

Pseudo-classes de structure et de position

Il ya aussi un tas d'autres pseudo-classes qui fournissent des informations supplémentaires sur l'environnement d'un élément. Par exemple, la pseudo-classe :last-of-type sélectionne l'élément final d'un type particulier dans son élément parent. Cela nous donne une alternative aux sélecteurs de classe pour sélectionner des éléments spécifiques.

:first-child,:last-child, & :only-child

La pseudo-classe :first-child choisira un élément si elle est le premier enfant dans son parent, alors que la pseudo-classe :last-child choisira un élément si elle est le dernier élément dans son parent. Ces pseudo-classes sélectionnent les premiers ou les derniers éléments d'une liste et ainsi de suite. En outre, le :only-child sélectionnera un élément si elle est le seul élément au sein d'un parent. Alternativement, la pseudo-classe :only-child peut être écrite comme :first-child:last-child cependant :only-child détient une spécificité inférieure.
Ici le sélecteur li:first-child identifie le premier élément de la liste dans une liste, tandis que le sélecteur li:last-child identifie le dernier élément de la liste dans une liste, donc les lignes 2 et 10 sont sélectionnés. Le sélecteur div:only-child est à la recherche d'une div qui est l'enfant unique d'un élément parent, sans autres frères et sœurs. Dans ce cas, la ligne 4 est choisi car il est la seule division au sein de l'élément de liste spécifique.

CSS

```
1 li:first-child {...}
2 li:last-child {...}
3 div:only-child {...}
```

HTML

```
1 <ul>
2   <li>Cet élément sera sélectionné</li>
3   <li>
4     <div>Cette div sera sélectionné</div>
5   </li>
6   <li>
7     <div>...</div>
8     <div>...</div>
```

```
  </li>
  <li>Cet élément sera sélectionné</li>
</ul>
```

:first-of-type, :last-of-type, & :only-of-type

Trouver les premiers, dernier, et seuls enfants d'un parent, est assez utile, et souvent nécessaire. Cependant, parfois, vous voulez sélectionner le premier, le dernier ou enfant unique d'un type spécifique d'élément. Par exemple, si vous voulez seulement sélectionner le premier ou le dernier paragraphe dans un article, ou la seule image dans un article. Heureusement c'est là que pseudo-sélecteurs :first-of-type, :last-of-type et les :only-of-type viennent en place.
La pseudo-classe :first-of-type choisira le premier élément de ce type au sein d'un parent, alors que la pseudo-classe :last-of-type choisira le dernier élément de ce type au sein d'un parent. La pseudo-classe :only-of-type choisira un élément si elle est la seule de ce type au sein d'un parent. Dans l'exemple ci-dessous la pseudo-classes p:first-of-type et p:last-of-type sélectionner les premiers et derniers paragraphes au sein de l'article, respectivement, peu importe si elles sont en fait les premiers ou les derniers enfants dans l'article. Les lignes 3 et 6 sont choisis réflexe ces sélecteurs. Le sélecteur img:only-of-type identifie l'image sur la ligne 5 car elle est la seule image à apparaître dans l'article, donc également sélectionné.

CSS

```
p:first-of-type {...}
p:last-of-type {...}
img:only-of-type {...}
```

HTML

```
<article>
  <h1>...</h1>
  <p>Ce paragraphe sera sélectionné</p>
  <p>...</p>
  <img src="#"><!-- Cette image sera sélectionnée -->
  <p>Ce paragraphe sera sélectionné</p>
  <h6>...</h6>

```

```
</article>
```

Enfin , il y a quelques pseudo-classes qui sélectionnent des éléments basés sur un nombre ou une expression algébrique. Ces pseudo-classes comprennent :nth-child(n), :nth-last-child(n), :nth-of-type(n) et :nth-last-of-type(n). Tous ces pseudo-classes uniques sont préfixés avec nth et acceptent pour argument un nombre ou une expression à l'intérieur de la parenthèse, indiquée par le caractère n.

Le nombre ou l'expression qui se trouve dans la parenthèse détermine exactement quel élément, ou éléments, doivent être sélectionnés. L'utilisation d'un nombre définitif comptera les éléments individuels du début ou de la fin de l'arborescence des documents, puis sélectionnera un élément, tandis que l'utilisation d'une expression comptera de nombreux éléments du début ou de la fin de l'arborescence des documents et les sélectionnera en groupes ou en multiples.

```
1  li:nth-child(2n) {
2    padding: 5px;
3    background-color: #333;
4    color: #fff;
5    margin: 10px 0;
6  }
```

La notation n dans l'équation ci-dessus représente le nombre séquentiel (0, 1, 2, 3, etc.). Dans l'exemple ci-dessus, 2 sera multiplié par n, ce qui donne les résultats suivants :

2(0) = 0 (ne sélectionne rien)
2(1) = 2 (sélectionne le 2ème élément enfant)
2(2) = 4 (sélectionne le 4e élément enfant)
2(3) = 6 (sélectionne le 6ème élément enfant)
... etc
Pour voir comment cela fonctionne dans une équation différente, vous pouvez utiliser cet outil, appelé le “:nth testeur”.

Utiliser des nombres et des expressions de pseudo-classe

Comme mentionné précédemment, l'utilisation de nombres directement dans une pseudo-classe comptera à partir du début ou de la fin de l'arborescence des documents et sélectionnera un élément en conséquence. Par exemple, le sélecteur `li:nth-child (4)` sélectionnera le quatrième élément de liste dans une liste. Le comptage débute avec le premier élément de la liste et augmente d'un point pour chaque élément de la liste jusqu'à ce que finalement, le quatrième élément soit sélectionné. Lorsqu'il utilise un numéro, il doit être un nombre positif.

Les expressions pour les pseudo-classes tombent dans le format de `a`, `a + b`, `an-b`, `n + b`, `-n + b` et `-an + b`. La même expression peut être traduite et lue comme `(a × n) ± b`. La variable `a` représente le multiplicateur dans lequel les éléments seront comptabilisés alors que la variable `b` représente l'endroit où le comptage commencera ou se produira.

Par exemple, le sélecteur `li:nth-child (3n)` identifie chaque troisième élément de liste dans une liste. En utilisant l'expression cela équivaut à 3 × 0, 3 × 1, 3 × 2, et ainsi de suite. Comme vous pouvez le voir, les résultats de cette expression conduisent au troisième, au sixième et à chaque élément, un multiple de trois étant sélectionné.

En outre, les valeurs de mot clé impair et pair peuvent être utilisées. Comme prévu, ils sélectionnent respectivement des éléments impairs ou pairs. Si les valeurs des mots clés ne sont pas attrayantes, l'expression de `2n+1` sélectionnera tous les éléments impairs tandis que l'expression de `2n` sélectionnera tous les éléments pairs.

L'utilisation du sélecteur `li:nth-child (4n + 7)` identifiera chaque élément de la liste à partir du septième élément de la liste. Encore une fois, en utilisant l'expression cela équivaut à (4 × 0) +7, (4 × 1) +7, (4 × 2) +7 et ainsi de suite. Les résultats de cette expression conduisant au septième, au onzième, au quinzième et à chaque élément un multiple de quatre ici à être sélectionné.

En utilisant l'argument `n` sans être préfixé par un nombre, la variable `a` est interprétée comme 1. Avec le sélecteur `li:nth-child (n + 5),` chaque élément de la liste sera sélectionné à partir du cinquième élément de la liste, laissant les quatre premiers Liste des éléments non sélectionnés. Dans l'expression, il se décompose comme (1 × 0) +5, (1 × 1) +5, (1 × 2) +5 et ainsi de suite.

Pour rendre les choses un peu plus complexes on peut utiliser des numéros négatifs. Par exemple, le sélecteur `li:nth-child (6n-4)` commencera à compter chaque sixième élément de liste commençant à quatre négatifs, en sélectionnant les deuxième, huitième et quatorzième éléments de liste et ainsi de suite. Le même sélecteur, `li:nth-child (6n-4)`, pourrait également être écrit comme `li:nth-child(6n + 2)`, sans l'utilisation d'une variable `b` négative.

Une variable négative, ou un argument `n` négatif, doit être suivie d'une variable `b` positive. Lorsqu'il est précédé d'un argument négatif ou d'un argument n négatif, la variable `b` identifie le niveau de comptage atteint. Par exemple, le sélecteur `li:nth-child(-3n+12)` sélectionnera chaque troisième élément de liste dans les douze premiers éléments de la liste. Le sélecteur `li:nth-child(-n+9)` sélectionne les neuf premiers éléments de la liste dans une liste car l'argument n, sans aucune variable déclarée, est par défaut -1.

:nth-child (n) & :nth-last-child (n)

Avec une compréhension générale de la façon dont les nombres pseudo-classe et les expressions fonctionnent , nous allons jeter un regard sur les pseudo-classes réelles dans lesquelles ces chiffres et expressions peuvent être utilisés, dont le premier étant les pseudo-classes `:nth-child(n)` et `:nth-last-child(n)`. Ces pseudo-classes fonctionnent un peu comme les pseudo-classes `:first-child` et `:last-child` en ce qu'ils regardent, et comptent, tous les éléments dans un parent et seulement sélectionner l'élément spécifiquement identifié. Les `:nth-child(n)` œuvres du début de l'arbre du document, tandis que les `:nth-last-child(n)` œuvres de la fin de l'arbre du document.

Utilisation de la pseudo-classe :nth-child(n), regardons le sélecteur li:nth-child(3n). Le sélecteur ici identifiera chaque troisième élément de liste, donc les lignes 4 et 7 sont choisis.

CSS

```
li:nth-child(3n) {...}
```

HTML

```
<ul>
  <li>...</li>
  <li>...</li>
  <li>Cet élément sera sélectionné</li>
  <li>...</li>
  <li>...</li>
  <li>Cet élément sera sélectionné</li>
</ul>
```

Utilisation d'une expression différente dans le :nth-child(n) donnera une autre sélection. Le li:nth-child(2n+3) sélecteur, par exemple, permettra d'identifier chaque seconde élément de la liste à partir de la troisième, puis en avant. Par conséquent, les lignes de la liste des objets 4 et 6 sont sélectionnés.

CSS

```
li:nth-child(2n+3) {...}
```

HTML

```
<ul>
  <li>...</li>
  <li>...</li>
  <li>Cet élément sera sélectionné</li>
  <li>...</li>
  <li>Cet élément sera sélectionné</li>
  <li>...</li>
</ul>
```

Changer à nouveau l'expression, cette fois avec une valeur négative, donne une nouvelle sélection. Ici, le li:nth-child(-n+4) sélecteur est

d'identifier les quatre principaux éléments de la liste, en laissant le reste de la liste des éléments non sélectionnés, donc les lignes 2 à 5 sont choisis.

CSS

```
li:nth-child(-n+4) {...}
```

HTML

```
<ul>
  <li>Cet élément sera sélectionné</li>
  <li>Cet élément sera sélectionné</li>
  <li>Cet élément sera sélectionné</li>
  <li>Cet élément sera sélectionné</li>
  <li>...</li>
  <li>...</li>
</ul>
```

Ajout d'un entier négatif avant l'argument de change à nouveau la sélection. Ici, le `li:nth-child(-2n+5)` sélecteur identifie chaque seconde élément de liste dans les cinq premiers éléments de la liste commençant par la première liste élément, donc les éléments de liste sur les lignes 2, 4 et 6 sont choisis.

CSS

```
li:nth-child(-2n+5) {...}
```

HTML

```
<ul>
  <li>Cet élément sera sélectionné</li>
  <li>...</li>
  <li>Cet élément sera sélectionné</li>
  <li>...</li>
  <li>Cet élément sera sélectionné</li>
  <li>...</li>
</ul>
```

Modification de la `:nth-child(n)` pseudo-classe à la `:nth-last-child(n)` pseudo-classe commute le sens de comptage, avec comptage à partir de la fin de l'arborescence du document en utilisant la `:nth-last-child(n)` pseudo-classe. Le `li:nth-last-child(3n+2)` sélecteur, par exemple, permettra

d'identifier chaque article troisième liste à partir de la deuxième à la dernière élément dans une liste, se déplaçant vers le début de la liste. Voici la liste des articles sur les lignes 3 et 6 sont sélectionnés.

CSS

```
li:nth-last-child(3n+2) {...}
```

HTML

```
<ul>
  <li>...</li>
  <li>Cet élément sera sélectionné</li>
  <li>...</li>
  <li>...</li>
  <li>Cet élément sera sélectionné</li>
  <li>...</li>
</ul>
```

:nth-of-type (n) & :nth-last-of-type (n)

Les pseudo-classes :nth-of-type(n) et :nth-last-of-type(n) sont très similaires à celle de la :nth-child(n) et de :nth-last-child(n), mais au lieu de compter tous les éléments au sein d'un parent les pseudo-classes :nth-of-type(n) et :nth-last-of-type(n) ne comptent que des éléments de leur propre type. Par exemple, lors du comptage des paragraphes dans un article, les pseudo-classes :nth-of-type(n) et :nth-last-of-type(n) vont sauter toutes les rubriques, les div, ou des éléments divers qui ne sont pas des paragraphes, alors que le :nth-child(n) et :nth-last-child(n) compteraient tous les éléments, peu importe son type, et sélectionner seulement ceux qui correspondent à cet élément dans le sélecteur indiqué. En outre, tous les mêmes possibilités d'expression utilisés dans les :nth-child(n) et :nth-last-child(n) pseudo-classes sont également disponibles dans les :nth-of-type(n) et :nth-last-of-type(n) pseudo-classes.
Utilisation de la pseudo-classe :nth-of-type(n) dans le sélecteur p:nth-of-type(3n) , nous sommes en mesure d'identifier chaque troisième alinéa dans un parent, indépendamment des autres éléments frères au sein de la société mère. Voici les paragraphes sur les lignes 5 et 9 sont sélectionnés.

CSS

```
p:nth-of-type(3n) {...}
```

HTML

```
<article>
  <h1>...</h1>
  <p>...</p>
  <p>...</p>
  <p>Ce paragraphe sera sélectionné</p>
  <h2>...</h2>
  <p>...</p>
  <p>...</p>
  <p>Ce paragraphe sera sélectionné</p>
</article>
```

Comme avec les :nth-child(n) et :nth-last-child(n), la principale différence entre la :nth-of-type(n) et :nth-last-of-type(n) est que la :nth-of-type(n) compte des éléments depuis le début de l'arbre du document et de la :nth-last-of-type(n) compte des éléments de la fin de l'arbre du document. Utilisation de la :nth-last-of-type(n) nous pouvons écrire le sélecteur p:nth-last-of-type(2n+1) qui identifie chaque deuxième alinéa de l'extrémité d'un élément parent en commençant par le dernier paragraphe. Voici les paragraphes sur les lignes 4, 7 et 9 sont choisis.

CSS

```
p:nth-last-of-type(2n+1) {...}
```

HTML

```
<article>
  <h1>...</h1>
  <p>...</p>
  <p>Ce paragraphe sera sélectionné</p>
  <p>...</p>
  <h2>...</h2>
  <p>Ce paragraphe sera sélectionné</p>
  <p>...</p>
  <p>Ce paragraphe sera sélectionné</p>
</article>
```

Target

La pseudo-classe :target est utilisée pour des éléments de style lorsque la valeur de l'attribut ID d'un élément correspond à celui de l'identifiant de fragment d'URI. L'identificateur de fragment dans une URI peut être reconnu par le caractère de hachage, #, et ce qui le suit directement. L'URL http://example.com/index.html#hello comprend l'identifiant de fragment de hello. Lorsque cet identifiant correspond à la valeur de l'attribut ID d'un élément sur la page, <section id="hello"> par exemple, cet élément peut être identifié et stylisé en utilisant la pseudo-classe :target. Les fragment d'identifiants sont les plus couramment observés sur une page des liens, ou un lien vers une autre partie de la même page.
En regardant le code ci-dessous, si un utilisateur visite une page avec l'identificateur de fragment URI #hello la section avec cette même valeur d'attribut ID serait stylisée en conséquence en utilisant la pseudo-classe :target. Si l'identificateur de fragment URI change, et correspond à la valeur d'attribut ID d'une autre section, cette nouvelle section peut être stylisée en utilisant le même sélecteur et pseudo-classe qu'avant.

CSS

```
1 section:target {...}
```

HTML

```
1 <section id="hello">...</section>
```

Empty

La pseudo-classe :empty permet de sélectionner des éléments qui ne contiennent pas d'enfants ou de nœuds de texte.

L'utilisation de la pseudo-classe div:empty identifiera les div sans enfants ou nœuds de texte. Ci-dessous, les div sur les lignes 2 et 3 sont choisis car ils sont complètement vides. Même si la deuxième div contient un commentaire, il est pas considéré comme un enfant, laissant ainsi la div vide. La première division contient du texte, la quatrième division contient un espace de texte vide, et la dernière division contient un strong élément enfant, donc ils sont tous exclus et ne sont pas sélectionnés.

CSS

```
div:empty {...}
```

HTML

```
<div>Hello</div>
<div><!-- Beintôt --></div><!-- Cette div sera
sélectionné -->
<div></div><!-- Cette div sera sélectionné -->
<div> </div>
<div><strong></strong></div>
```

Not

La pseudo-classe :not(x) est une pseudo-classe qui prend un argument qui est filtré à partir de la sélection à faire. Le sélecteur p:not(.intro) utilise la négation pour identifier chaque élément de paragraphe sans la classe "intro". L'élément de paragraphe est identifié au début de la sélection suivie par la pseudo-classe :not(x). A l'intérieur des parenthèses se trouve le sélecteur de négation, la classe "intro" dans ce cas.
Ci-dessous, à la fois le div:not(.awesome) et :not(div) utilisent la pseudo-classe :not(x). Le div:not(.awesome) identifie une div sans la classe "awesome", tandis que le :not(div) identifie tout élément qui est pas une div. En conséquence, la div de la ligne 1 est sélectionnée, ainsi que les deux sections sur les lignes 3 et 4, ainsi ils sont marqués en gras. Le seul élément sélectionné est la division de la classe "awesome", comme il est en dehors des deux pseudo-classes not.

CSS

```
div:not(.awesome) {...}
:not(div) {...}
```

HTML

```
<div>Cette div sera sélectionné</div>
<div class="awesome">...</div>
<section>Cette section sera sélectionné</section>
<section class="awesome">Cette section sera
sélectionné</section>
```

Exemple de pseudo-classes

HTML

```html
<table>
  <thead>
    <tr>
      <th>Number</th>
      <th>Player</th>
      <th>Position</th>
      <th>Height</th>
      <th>Weight</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>8</td>
      <td>Marco Belinelli</td>
      <td>G</td>
      <td>6-5</td>
      <td>195</td>
    </tr>
    <tr>
      <td>5</td>
      <td>Carlos Boozer</td>
      <td>F</td>
      <td>6-9</td>
      <td>266</td>
    </tr>
    ...
  </tbody>
</table>
```

CSS

```css
table {
  border-collapse: separate;
  border-spacing: 0;
  width: 100%;
}
th,
td {
  padding: 6px 15px;
```

```
}
th {
  background: #42444e;
  color: #fff;
  text-align: left;
}
tr:first-child th:first-child {
  border-top-left-radius: 6px;
}
tr:first-child th:last-child {
  border-top-right-radius: 6px;
}
td {
  border-right: 1px solid #c6c9cc;
  border-bottom: 1px solid #c6c9cc;
}
td:first-child {
  border-left: 1px solid #c6c9cc;
}
tr:nth-child(even) td {
  background: #eaeaed;
}
tr:last-child td:first-child {
  border-bottom-left-radius: 6px;
}
tr:last-child td:last-child {
  border-bottom-right-radius: 6px;
}
```

Number	Player	Position	Height	Weight
8	Marco Belinelli	G	6-5	195
5	Carlos Boozer	F	6-9	266
21	Jimmy Butler	G-F	6-7	220
9	Luol Deng	F	6-9	220
22	Taj Gibson	F	6-9	225
32	Richard Hamilton	G	6-7	193
12	Kirk Hinrich	G	6-4	190
48	Nazr Mohammed	C	6-10	250
13	Joakim Noah	C	6-11	232
77	Vladimir Radmanovic	F	6-10	235
2	Nate Robinson	G	5-9	180
1	Derrick Rose	G	6-3	190
25	Marquis Teague	G	6-2	190

Les pseudo-éléments

Les pseudo-éléments sont des éléments dynamiques qui n'existent pas dans l'arborescence des documents et lorsqu'ils sont utilisés dans les sélecteurs, ces pseudo-éléments permettent de styliser des parties uniques de la page. Un point important à noter, un seul pseudo-élément peut être utilisé dans un sélecteur à un moment donné.

Les pseudo-éléments textuels

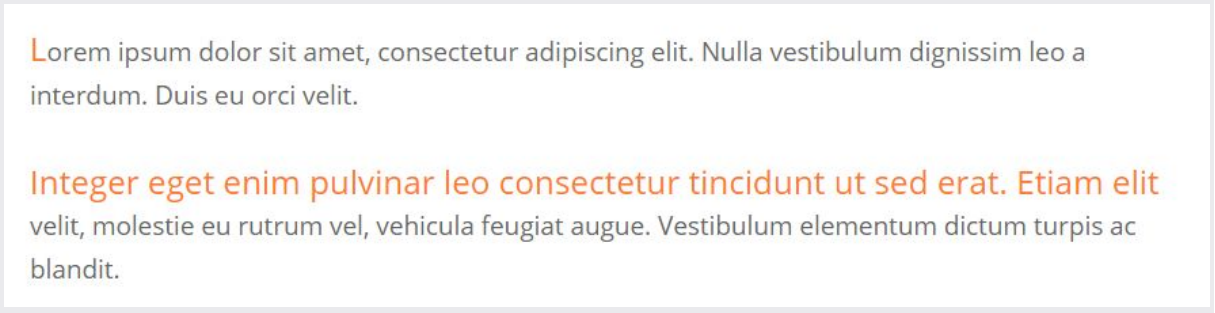
Les premiers pseudo-éléments à avoir été créés étaient les pseudo-éléments textuels :first-letter et :first-line. Le pseudo-élément de première lettre identifie la première lettre de texte dans un élément, alors que le pseudo-élément de première ligne identifie la première ligne de texte d'un élément.
Dans la démonstration ci-dessous la première lettre du paragraphe par la classe .alpha est stylé avec une police de grande taille et de couleur orange, comme la première ligne du paragraphe par la classe .bravo. Ces sélections sont faites par l' utilisation respectivement pseudo-éléments textuelles des :first-letter et :first-line.

CSS

```css
.alpha:first-letter,
.bravo:first-line {
  color: #ff7b29;
  font-size: 18px;
}
```

HTML

```html
<p class="alpha">Lorem ipsum dolor...</p>
<p class="bravo">Integer eget enim...</p>
```



Les pseudo-éléments générés

Les pseudo-éléments :before et :after créer de nouveaux pseudo-éléments de niveau ligne juste à l'intérieur de l'élément sélectionné. Le plus souvent, ces pseudo-éléments sont utilisés en conjonction avec la propriété content pour ajouter des informations insignifiantes à une page. Une autre utilisation de ces pseudo-éléments peut être d'ajouter des composants d'interface utilisateur à la page sans avoir à encombrer le document avec des éléments qui ne font pas sens, non sémantique.
Le pseudo-élément :before crée un pseudo-élément avant l'élément sélectionné, alors que le pseudo-élément :after crée un pseudo-élément après l'élément sélectionné. Ces pseudo-éléments sont imbriqués dans l'élément sélectionné, et non en dehors de celui-ci. En dessous de la :after pseudo-élément est utilisé pour afficher la href valeur de l'attribut des liens d'ancrage entre parenthèses après les liens réels. L'information ici est utile, mais finalement pas nécessaire si un navigateur supporte pas ces pseudo-éléments.

CSS

```css
a:after {
  color: #9799a7;
```

```
  content: " (" attr(href) ")";
  font-size: 11px;
}
```

HTML

```
<a href="http://google.com/">Search the Web</a>
```

Rechercher sur le web (http://google.com/)

Pseudo-élément fragment

Le pseudo-élément ::selection identifie une partie du document qui a été sélectionné ou mis en surbrillance par un utilisateur. La sélection peut alors être stylisée, mais uniquement en utilisant les propriétés color, background, background-color et text-shadow. Il est intéressant de noter que la propriété background-image est ignorée.

Simple : contre double ::

Le pseudo-élément de fragment a été ajouté avec CSS3 et dans la tentative de différencier les pseudo-classes de pseudo-éléments les double -points ont été ajoutées pour les pseudo-éléments. Heureusement , la plupart des navigateurs supportent les deux valeurs pour les pseudo-éléments. Seul le pseudo-élément ::selection doit toujours commencer par un double deux-points.

Lors de la sélection tout le texte au sein de la démonstration ci-dessous l'arrière-plan apparaît orange et les ombres de texte seront supprimées grâce au pseudo-élément de fragment ::selection.

```
::selection {
  background: #ff7b29;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla vestibulum dignissim leo a interdum. Duis eu orci velit. In urna quam, eleifend in pulvinar vitae, commodo vitae quam. Praesent vel magna nibh. Nullam lectus nibh, pellentesque ac convallis vulputate, ornare vel libero.

Exemple de pseudo-éléments

HTML

```html
<a class="arrow" href="#">Lire la suite</a>
```

CSS

```css
.arrow {
  background: #2db34a;
  color: #fff;
  display: inline-block;
  height: 30px;
  line-height: 30px;
  padding: 0 12px;
  position: relative;
  text-decoration: none;
}
.arrow:before,.arrow:after {
  content: "";
  height: 0;
  position: absolute;
  width: 0;
}
.arrow:before {
  border-bottom: 15px solid #2db34a;
  border-left: 15px solid transparent;
  border-top: 15px solid #2db34a;
  left: -15px;
}
.arrow:after {
  border-bottom: 15px solid transparent;
  border-left: 15px solid #2db34a;
  border-top: 15px solid transparent;
  right: -15px;
}
.arrow:hover {
  background: #ff7b29;
}
.arrow:hover:before {
  border-bottom: 15px solid #ff7b29;
  border-top: 15px solid #ff7b29;
}
```

```
35 .arrow:hover:after {
36   border-left: 15px solid #ff7b29;
37 }
38
```