Transitions & Animations

Une des évolutions de CSS3 était la capacité d'écrire des comportements pour des transitions et des animations. Les développeurs front-end ont demandé la possibilité de concevoir ces interactions au sein de HTML et CSS, sans utiliser JavaScript ou Flash, depuis des années. Maintenant, leur souhait est devenu réalité.

Avec les transitions CSS3 vous avez la possibilité de modifier l'apparence et le comportement d'un élément à chaque fois qu'un changement d'état se produit, par exemple quand la souris est placée au-dessus, le focus est mis dessus, est active ou ciblée.

Les animations CSS3 permettent de changer l'aspect et le comportement d'un élément en plusieurs images clés. Les transitions fournissent un changement d'un état à un autre, alors que les animations peuvent définir plusieurs points de transition sur les différentes images clés.

Les démos pour ce chapitre sont accessibles à cette url:
https://codepen.io/collection/lLwmD/

Transitions

Comme mentionné précédemment, pour qu'une transition ait lieu, un élément doit avoir un changement d'état, et les styles différents doivent être identifiés pour chaque état. La meilleure façon pour déterminer les styles pour différents états est d'utiliser les pseudo-classes `:hover`, `:focus`, `:active` et `:target`.

Il y a quatre propriétés liées à la transition au total `transition-property`, `transition-duration`, `transition-timing-function` et `transition-delay`. Tous ces éléments ne sont pas nécessaires pour construire une transition, les trois premiers sont les plus souvent utilisés.

Dans l'exemple ci-dessous la boîte va changer sa couleur `background` sur un laps de temp d' `1` seconde en mode `linear`.

```css
.box {
  background: #2db34a;
  transition-property: background;
  transition-duration: 1s;
  transition-timing-function: linear;
}
.box:hover {
  background: #ff7b29;
}
```

Préfixes navigateurs

Le code ci-dessus, comme le reste des exemples de code dans cette leçon, ne sont pas préfixés. Ceci est intentionnel afin de maintenir les extraits de code simples et compréhensibles. Pour un meilleur support de tous les navigateurs, vous devez utiliser des préfixes.

Pour information, la version préfixé du code ci-dessus ressemblerait à ce qui suit.

```css
.box {
    background: #2db34a;
    -webkit-transition-property: background;
       -moz-transition-property: background;
         -o-transition-property: background;
            transition-property: background;
    -webkit-transition-duration: 1s;
       -moz-transition-duration: 1s;
         -o-transition-duration: 1s;
            transition-duration: 1s;
    -webkit-transition-timing-function: linear;
       -moz-transition-timing-function: linear;
         -o-transition-timing-function: linear;
            transition-timing-function: linear;
}
.box:hover {
  background: #ff7b29;
}
```

Propriété de transition

La propriété `transition-property` détermine exactement quelles propriétés seront modifiées conjointement avec les autres propriétés de transition. Par défaut, toutes les propriétés dans différents états d'un élément seront modifiées en cas de changement. Cependant, seules les propriétés identifiées dans la valeur `transition-property` seront affectées par les transitions.
Dans l'exemple ci-dessus, la propriété `background` est identifiée dans la valeur `transition-property`. Ici la propriété `background` est la seule propriété qui va changer au cours de la durée de `1` seconde dans un mode `linear`. Toutes les autres propriétés incluses lors du changement de l'état d'un élément, mais non inclus dans la valeur `transition-property`, ne recevront pas les comportements de transition fixés par les propriétés `transition-duration` ou `transition-timing-function`.

Si plusieurs propriétés doivent être affectées par la transition elles peuvent être séparées par des virgules dans la valeur `transition-property`. En outre, la valeur de mot-clé `all` peut être utilisée pour que la transition affecte toutes les propriétés d'un élément.

```
.box {
    background: #2db34a;
    border-radius: 6px
    transition-property: background, border-radius;
    transition-duration: 1s;
    transition-timing-function: linear;
  }
  .box:hover {
    background: #ff7b29;
    border-radius: 50%;
  }
```

Propriétés de transition

Il est important de noter que toutes les propriétés ne peuvent être transformées, seules les propriétés ayant un point de repère identifiable. Les couleurs, les tailles de police et les mêmes peuvent être transférés d'une valeur à l'autre car ils ont des valeurs reconnaissables entre elles. La propriété `display`, par exemple, peut ne pas être transférée car elle

n'a aucun point médian. Les propriétés de transition les plus populaires sont les suivantes.

- `background-color`
- `background-position`
- `border-color`
- `border-width`
- `border-spacing`
- `bottom`
- `clip`
- `color`
- `crop`
- `font-size`
- `font-weight`
- `height`
- `left`
- `letter-spacing`
- `line-height`
- `margin`
- `max-height`
- `max-width`
- `min-height`
- `min-width`
- `opacity`
- `outline-color`
- `outline-offset`
- `outline-width`
- `padding`
- `right`
- `text-indent`
- `text-shadow`
- `top`
- `vertical-align`
- `visibility`
- `width`
- `word-spacing`
- `z-index`

Durée de la transition

La durée pendant laquelle une transition a lieu est réglée à l'aide de la propriété `transition-duration`. La valeur de cette propriété peut être définie à l'aide des valeurs de timing général, y compris les secondes (`s`)

et millisecondes (`ms`). Ces valeurs de synchronisation peuvent également apparaître dans des mesures fractionnaires, `.2s` par exemple.
Lors de la transition des propriétés multiples, vous pouvez définir plusieurs durées, une pour chaque propriété. Comme avec la valeur de la propriété `transition-property`, plusieurs durées peuvent être déclarées à l'aide des valeurs séparées par des virgules. L'ordre de ces valeurs lors de l'identification des propriétés individuelles et la durée importe. Par exemple, la première propriété identifiée dans la propriété `transition-property` correspondra avec celle la première fois identifiée dans la propriété `transition-duration`, et ainsi de suite.
Si plusieurs propriétés sont en transition avec une seule valeur de durée déclarée, cette valeur sera la durée de toutes les propriétés de transition.

```
.box {
  background: #2db34a;
  border-radius: 6px;
  transition-property: background, border-radius;
  transition-duration: .2s, 1s;
  transition-timing-function: linear;
}
.box:hover {
  background: #ff7b29;
  border-radius: 50%;
}
```

Période de transition

La propriété `transition-timing-function` est utilisée pour régler la vitesse à laquelle une transition se fait. Connaissant la durée de la propriété `transition-duration` d'une transition, on peut avoir plusieurs vitesses dans un même durée. Quelques-unes des valeurs de mots-clés les plus populaires pour la propriété `transition-timing-function sont linear`, `ease-in`, `ease-out` et `ease-in-out`.
La valeur clé `linear` identifie une transition se déplaçant à une vitesse constante d'un état à un autre. La valeur `ease-in` identifie une transition qui commence lentement et accélère tout au long de la transition, alors que la valeur `ease-out` identifie une transition qui commence rapidement et ralentit tout au long de la transition. La valeur `ease-in-out` identifie une transition qui commence lentement, accélère au milieu, puis ralentit à nouveau avant la fin.

Chaque fonction de synchronisation correspond à une courbe cubique-Bézier, qui peut être réglée précisément en utilisant la fonction `cubic-bezier(x1, y1, x2, y2)`. On peut aussi utiliser les valeurs `step-start`, `step-stop` et la fonction `steps(nombre_etapes, direction)`.

Lors de la transition des propriétés multiples, vous pouvez identifier de multiples fonctions de synchronisation. Ces valeurs de synchronisation, comme avec d'autres valeurs de propriétés de transition, peuvent être déclarés comme valeurs séparées par des virgules.

```
.box {
  background: #2db34a;
  border-radius: 6px;
  transition-property: background, border-radius;
  transition-duration: .2s, 1s;
  transition-timing-function: linear, ease-in;
}
.box:hover {
  background: #ff7b29;
  border-radius: 50%;
}
```

Retard de transition

En plus de déclarer la propriété de transition, la durée et la fonction de synchronisation, vous pouvez également définir un délai à la propriété `transition-delay`. Le retard définit une valeur de temps, secondes ou millisecondes, qui détermine la durée de la transition doit être au point mort avant d' exécuter. Comme avec toutes les autres propriétés de transition, à retarder de nombreuses transitions, chaque retard peut être déclarée comme valeurs séparées par des virgules.

```css
.box {
  background: #2db34a;
  border-radius: 6px
  transition-property: background, border-radius;
  transition-duration: .2s, 1s;
  transition-timing-function: linear, ease-in;
  transition-delay: 0s, 1s;
}
.box:hover {
  background: #ff7b29;
  border-radius: 50%;
}

```

Transitions raccourcies

Déclarer chaque propriété de transition individuellement peut devenir très fastidieux, en particulier avec des préfixes fournisseurs. Heureusement , il existe un raccourci, `transition,` capable de supporter toutes ces propriétés et des valeurs différentes. En utilisant la propriété `transition` seule, vous pouvez définir chaque valeur de transition dans l'ordre `transition-property`, `transition-duration`, `transition-timing-function` et enfin `transition-delay`. Il ne faut pas utiliser des virgules avec ces valeurs.

Exemples

```css
.box {
  background: #2db34a;
  border-radius: 6px;
  transition: background .2s linear, border-radius 1s
ease-in 1s;
}
.box:hover {
  color: #ff7b29;
  border-radius: 50%;
}
```

Exemple de bouton de transition

HTML

```html
<button>Awesome Button</button>
```

CSS

```css
button {
  border: 0;
  background: #0087cc;
  border-radius: 4px;
  box-shadow: 0 5px 0 #006599;
  color: #fff;
  cursor: pointer;
  font: inherit;
  margin: 0;
  outline: 0;
  padding: 12px 20px;
  transition: all .1s linear;
}
button:active {
  box-shadow: 0 2px 0 #006599;
  transform: translateY(3px);
}

```

Exemple de Flip Card

HTML

```html
<div class="card-container">
  <div class="card">
    <div class="side">...</div>
    <div class="side back">...</div>
  </div>
</div>
```

CSS

```css
.card-container {
  height: 150px;
  perspective: 600;
  position: relative;
  width: 150px;
}
.card {
  height: 100%;
  position: absolute;
  transform-style: preserve-3d;
  transition: all 1s ease-in-out;
  width: 100%;
}
.card:hover {
  transform: rotateY(180deg);
}
.card .side {
  backface-visibility: hidden;
  height: 100%;
  position: absolute;
  width: 100%;
}
.card .back {
  transform: rotateY(180deg);
}
```

Animations

Les transitions font un excellent travail pour construire des interactions visuelles d'un état à l'autre et sont parfaites pour ces types de changements d'état unique. Cependant, lorsqu'un contrôle supplémentaire est nécessaire, les transitions doivent avoir plusieurs états. En retour, c'est là que les animations sont utiles.

Animations et images clés

Pour définir plusieurs points au cours de laquelle un élément doit subir une transition, utilisez la règle `@keyframes`. La règle `@keyframes` inclut le

nom de l'animation, les points d'arrêt d'animation et les propriétés destinées à être animés.

```
@keyframes slide {
  0% {
    left: 0;
    top: 0;
  }
  50% {
    left: 244px;
    top: 100px;
  }
  100% {
    left: 488px;
    top: 0;
  }
}

```

Préfixes pour la règle Keyframe

La règle `@keyframes` doit être préfixée, tout comme tous les autres propriétés `transition` et `animation`. Les préfixes fournisseurs pour la `@keyframes` ressemblent à ce qui suit:

- `@-moz-keyframes`
- `@-o-keyframes`
- `@-webkit-keyframes`

L'animation ci-dessus est nommé `slide`, qui est déclarée juste après l'ouverture de la règle `@keyframes`. Les différents points d'arrêt d'images clés sont définies à l'aide des pourcentages, à partir de `0%` et jusqu'à `100%` avec un point d'arrêt intermédiaire à `50%`. Les mots-clés `from` et `to` pourraient être utilisés à la place `0%` et `100%` si on le souhaite. D'autres points d'arrêt, ici `50%`, peuvent également être indiqués. Les propriétés de l'élément qui sera animé sont répertoriés à l'intérieur de chacun des points d'arrêt, `left` et `top` dans l'exemple ci-dessus.
Il est important de noter que, comme pour les transitions, des propriétés individuelles peuvent être animées. Voyez comment vous pouvez déplacer un élément de haut en bas par exemple. Essayer d'animer à partir `top: 0;` de `bottom: 0;` ne fonctionnera pas, car les animations ne peuvent

appliquer une transition qu'au sein d'une seule propriété, pas d'une propriété à l'autre. Dans ce cas, l'élément devra être animé de `top:0 à top: 100%;`.

Allez sur le démo "Animation Keyframe" et passez la souris sur la balle ci-dessous pour voir l'animation en action.

Nom d'animation

Une fois les images clés pour une animation ont été déclarés , ils doivent être affectés à un élément. Pour ce faire, la propriété `animation-name` est utilisée avec le nom d'animation qui est celui défini comme la valeur de la règle `@keyframes`. La déclaration `animation-name` corresponf à l'élément sur lequel l'animation sera appliquée.

```
.stage:hover .ball {
  animation-name: slide;
}
```

Utiliser la propriété `animation-name` seule ne suffit pas cependant. Vous devez également déclarer une propriété `animation-duration` et préciser la valeur afin que le navigateur puisse savoir combien de temps une animation devrait prendre pour se terminer.

Durée de l'animation, synchronisation et délai

Une fois que vous avez déclaré la propriété `animation-name` sur un élément, des animations se comportent comme des transitions. Ils comprennent une durée, la fonction de synchronisation, et le délai si on le souhaite. Pour commencer, les animations ont besoin d' une durée déclarée en utilisant la propriété `animation-duration`. Comme pour les transitions, la durée peut être réglée en quelques secondes ou en millisecondes.

```
.stage:hover .ball {
  animation-name: slide;
  animation-duration: 2s;
}
```

Une fonction de synchronisation et de délai peuvent être déclarée à l'aide respectivement des propriétés `animation-timing-function` et `animation-delay`.

Les valeurs de ces propriétés et se comportent comme avec les transitions.

```
.stage:hover .ball {
  animation-name: slide;
  animation-duration: 2s;
  animation-timing-function: ease-in-out;
  animation-delay: .5s;
}
```

L'animation ci-dessous devrait faire rebondir la balle `-.ball-` une fois en se déplaçant vers la gauche, mais seulement lorsque l'on survole la scène `-.stage-`.

HTML

```
<div class="stage">
  <figure class="ball"></figure>
</div>
```

CSS

```
@keyframes slide {
  0% {left: 0;top: 0; }
  50% { left: 244px;top: 100px; }
  100% {left: 488px; top: 0; }
}
.stage {
  height: 150px; position: relative;
  background: #eaeaed;
}
.ball {
    height: 50px; position: absolute;width: 50px;
    background: #2db34a; border-radius: 50%;
}
.stage:hover .ball {
  animation-name: slide;
  animation-duration: 2s;
  animation-timing-function: ease-in-out;
  animation-delay: .5s;
}
```

Passez la souris sur la balle ci-dessous pour voir l'animation en action.

Personnalisation des animations

Les animations offrent également la possibilité de personnaliser le comportement d'un élément, y compris la possibilité de déclarer le nombre de fois qu'une animation fonctionne, ainsi que la direction dans laquelle une animation sera faite.

Itération des animations

Par défaut, les animations font leur cycle une fois du début à la fin, puis s'arrêtent. Pour avoir une animation qui se répète plusieurs fois on peut utiliser la propriété `animation-iteration-count`. Les valeurs de `animation-iteration-count` comprennent un entier ou le mot-clé `infinite`. En utilisant un entier, on répétera l'animation autant de fois que spécifié, alors que le mot-clé `infinite` répétera indéfiniment l'animation.

```
.stage:hover .ball {
  animation-name: slide;
  animation-duration: 2s;
  animation-timing-function: ease-in-out;
  animation-delay: .5s;
  animation-iteration-count: infinite;
}
```

Direction de l'animation

En plus d'être en mesure de définir le nombre de fois qu'une animation se répète, vous pouvez également déclarer la direction d'une animation complète en utilisant la propriété `animation-direction`. Les valeurs de `animation-direction` comprennent `normal`, `reverse`, `alternate` et `alternate-reverse`.
La valeur `normal` joue une animation comme prévu du début à la fin. La valeur `reverse` va jouer l'animation exactement à l'opposé de celles se trouvant dans les règles `@keyframes`, en partant donc de `100%` et en remontant à `0%`.
La valeur `alternate` va jouer une animation en avant puis en arrière. Dans les images clés qui comprend en cours d'exécution avant de `0%` la

`100%` puis vers l'arrière à partir `100%` de `0%`. L'utilisation de la propriété `animation-iteration-count` peut limiter le nombre de fois qu'une animation tourne à la fois vers l'avant et en arrière. Le décompte commence à `1` l'exécution d'une animation avant de `0%` à `100%`, puis ajoute `1 à` l'exécution d'une animation en arrière par rapport `100%` à `0%`. La combinaison fera donc un total de `2` itérations. La valeur `alternate` des fonctions aussi, inverse de synchronisation lors de la lecture en sens inverse. Si une animation utilise la valeur `ease-in` allant de `0%` la `100%`, il utilise alors la valeur `ease-out` de `100%`la `0%`.
Enfin, la valeur `alternate-reverse` combine à la fois les valeurs `alternate` et les `reverse`, en cours d'exécution d'une animation en arrière puis vers l' avant. La valeur `alternate-reverse` commence à `100%` vers `0%` et puis va revenir à `100%` à nouveau.

```
.stage:hover .ball {
  animation-name: slide;
  animation-duration: 2s;
  animation-timing-function: ease-in-out;
  animation-delay: .5s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
```

Etat d'une animation

La propriété `animation-play-state` permet de définir si l'animation est jouée ou en pause en utilisant respectivement les valeurs de mot clé `running` et `paused`. Lorsque vous jouez une animation mise en pause, elle reprendra l'exécution de son état actuel plutôt que de commencer dès le début à nouveau.

Dans l'exemple ci-dessous la propriété `animation-play-state` est définie `paused` au moment de la classe `stage` est active, soit en cliquant dessus. Remarquez comment l'animation se met en pause temporairement jusqu'à ce que vous relâchiez le bouton de la souris.

```
.stage:hover .ball {
  animation-name: slide;
  animation-duration: 2s;
  animation-timing-function: ease-in-out;
  animation-delay: .5s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
.stage:active .ball {
  animation-play-state: paused;
}

```

Animation Fill Mode

La propriété `animation-fill-mode` identifie la façon dont un élément doit être styler avant, après, ou avant et après qu'une animation est été exécutée. La propriété `animation-fill-mode` accepte quatre valeurs clés, à savoir `none`, `forwards`, `backwards` et `both`.

La valeur `none` n'appliquera aucun styles aux éléments avant ou après qu'une animation aura été exécutée.

La valeur `forwards` gardera les styles déclarés dans la dernière image clé spécifiée `@keyframe`. Ces styles peuvent toutefois être affectés par les valeurs `animation-direction` et les `animation-iteration-count`.

La valeur `backwards` appliquera les styles dans la première image clé spécifiée dès comme étant identifié, avant que l'animation a été exécutée. Cela ne comprend l'application de ces styles pendant tout le temps qui peut être réglé dans un délai d'animation. La valeur `backwards` peut également être affectée par la valeur de la propriété `animation-direction`.

Enfin, la valeur `both` L'animation respectera aussi les règles qui s'appliquent à `forwards` et `backwards`, entraînant ainsi l'extension des propriétés de l'animation dans les deux directions.

HTML

```html
<p>Déplacez votre souris sur la boîte grise.</p>
<div class="demo">
  <div class="grows">La boîte grandit</div>
  <div class="growsandstays">La boîte grandit et s'arrête</div>
</div>
```

```css
.demo {
  border-top: 100px solid #ccc;
  height: 300px;
  font-family: sans-serif;
}
@keyframes grow {
    0% { font-size: 0 }
    100% { font-size: 40px }
}
@-webkit-keyframes grow {
    0% { font-size: 0 }
    100% { font-size: 40px }
}
.demo:hover .grows {
    animation-name: grow;
    animation-duration: 3s;
    -webkit-animation-name: grow;
    -webkit-animation-duration: 3s;
}
.demo:hover .growsandstays {
    animation-name: grow;
    animation-duration: 3s;
    animation-fill-mode: forwards;
    -webkit-animation-name: grow;
    -webkit-animation-duration: 3s;
    -webkit-animation-fill-mode: forwards;
}
```

Raccourcis et animation

Heureusement les animations , tout comme les transitions, peuvent être écrites dans un format raccourci. Ceci est possible avec la propriété `animation`, plutôt que de déclarer plusieurs propriété `animation`....
L'ordre des valeurs dans la propriété `animation` doit être le suivant: `animation-name`, `animation-duration`, `animation-timing-function`, `animation-delay`, `animation-iteration-count`, `animation-direction`, `animation-fill-mode` et enfin `animation-play-state`.

```
.stage:hover .ball {
  animation: slide 2s ease-in-out .5s infinite alternate;
}
.stage:active .ball {
  animation-play-state: paused;
}
```

Ressources et liens

- Introduction aux animations via La Cascade
- Le Guide CSS Animation: principes et exemples par Smashing Magazine
- L' utilisation des animations CSS via Mozilla Developer Network

https://webdesign.tutsplus.com/articles/15-inspiring-examples-of-css-animation-on-codepen--cms-23937