Cours d'intégration

3W Academy

CSS 7. Transformations et animations

7. Transformations et animations

7.1. Présentation

Ce chapitre rassemble 3 techniques issues de CSS 3 dédiées au positionnement et à l'animation des éléments HTML: les transformations, les transitions et les animations.

Les transformations permettent de modifier l'emplacement des éléments HTML sans perturber le flux normal.

Les transitions permettent d'appliquer une durée entre deux changement d'état d'un élément pour une plusieurs propriétés CSS, y compris les transformations.

Les animations permettent d'animer un élément suivant un scénario pré établi.

Dans les exemples ci dessous, nous utiliserons les codes HTML et CSS suivant comme base.

Le code HTML

```
<div class="test"></div>
```

Le code CSS

```
.test {
    width: 400px;
    height: 200px;
    margin: 100px auto 0;
    background-color: #fc3;
}
```

Le document contient à présent un rectangle jaune centré.

7.2. Les transformations

7.2.1. Déclaration

Il existe 2 propriétés majeures pour définir les transformations:

La propriété ***transform-origin*** spécifie le point d'origine sur lequel s'applique les transformations, par défaut, le coin supérieur gauche de l'élément.

La propriété ***transform*** décrit la transformation à appliquer à l'élément via une liste de transformations séparées par des espaces appliques les unes après les autres.

Attention à bien utiliser les préfixes: -moz-, -webkit-

7.2.2. Un exemple

```
.test {
    transform-origin: center center;
    transform: scale(2) rotate(45deg);
}
```

Ici, notre rectangle jaune sera 2 fois plus grand et sera pivoté d'un angle de 45 degrés par rapport à son centre (suivant la propriété ***transform-origin***).

7.2.3. Les différentes transformations (2D)

- **scale:** pour un effet de zoom, par exemple: scale(2), scale(2,0.5)
- **scaleX**: seulement pour un zoom horizontal: scaleX(2), l'élément sera déformé
- **scaleY**: seulement pour un zoom vertical: scaleY(2), l'élément sera déformé
- **rotate**: par exemple rotate(45deg), rotate(0.5turn)
- **translate**: pour déplacer l'élément: translate(20px, 25%)
- **translateX**: seulement à l'horizontal: translateX(20px)
- **translateY**: seulement à la verticale: translateY(20px)
- **skew**: pour incliner l'élément: skew(20deg, 30deg)
- **skewX**: seulement à l'horizontal: skewX(20deg)
- **skewY**: seulement à la verticale:skewY(20deg)
- **matrix**: pour effectuer une transformation plus complexe

7.3. Les transitions

Afin de créer une animation entre deux état d'un élément plutôt que d'avoir un changement brusque, on peut utiliser les transitions. Les différents états peuvent êtres liés à une pseudo classe (:hover, :active) ou associé à un comportement en javascript (non décrit ici).

Les transitions permettent de déterminer la durée de l'animation, le délai initial et la fonction utilisée lors de cette animation pour la ou les propriétés CSS à animer.

7.3.1. Un exemple

Par exemple, si l'on souhaite que notre rectangle jaune devienne rouge lors d'un survol de la souris (:hover) mais avec une animation entre le passage du jaune au rouge d'une durée de 1 seconde:

```
.test {
    transition-property: background-color;
    transition-duration: 1s;
}

.text:hover {
    background-color: red;
}
```

On peut aussi utiliser la propriété raccourcie ***transition***:

```
.test {
    transition: background-color 1s;
}

.text:hover {
    background-color: red;
}
```

Il est important de déclarer la transtion dans l'état initial (et pas dans l'état :hover) pou que l'effet fonctionne en retour lorsque l'on retire le curseur du rectangle.

7.3.2. Transition sur plusieurs propriétés CSS

On peut par exemple utiliser le mot clé all au lieu de la propriété

```
.test {
    transition: all 1s;
}

.text:hover {
    background-color: red;
    transform: scale(2);
}
```

Ici, le rectangle sera aussi 2 fois plus grand.

On peut aussi déclarer plusieurs transitions pour chacune des propriétés (séparées par des virgules):

```
.test {
    transition: background-color 1s, transform 0.5s;
}

.text:hover {
    background-color: red;
    transform: scale(2);
}
```

7.3.3. La fonction de transition

Il s'agit de la courbe d'accélération utilisée pour calculer les valeurs intermédiaire de la propriété transformée en fonction du temps. La valeur par défaut est ***ease***. Les différentes fonctions sont:

- ease
- ease-in
- ease-out
- ease-in-out
- linear
- step-start
- step-end
- cubic-bezier(): par exemple cubic-bezier(0.1, 0.7, 1.0, 0.1)

Pour personnaliser la fonction cubic-bezier: http://cubic-bezier.com

```
.test {
    transition-property: background-color;
    transition-duration: 1s;
```

```
    transition-timing-function: linear;
}

.text:hover {
    background-color: red;
}
```

Ou avec la propriété raccourcie ***transition***:

```
.test {
    transition: background-color 1s linear;
}

.text:hover {
    background-color: red;
}
```

7.3.4. Le délai initial

L'ajout d'un délai initial permet de déclencher l'animation avec un certain retard.

```
.test {
    transition-property: background-color;
    transition-duration: 1s;
    transition-delay: 0.5s;
}

.text:hover {
    background-color: red;
}
```

Ou avec la propriété raccourcie ***transition***:

```
.test {
    transition: background-color 1s 0.5s;
}

.text:hover {
    background-color: red;
}
```

7.3.5. Un exemple complet

```
.test {
    transition: all 1s linear 0.5s;
}

.text:hover {
    background-color: red;
    transform: scale(2);
}
```

7.4. Les animations

Les animations en CSS nécessitent deux déclarations distinctes:

- la déclaration de l'animation comprenant les différentes caractéristiques de celle-ci comme la durée et éventuellement une fonction, un délai initial, un compteur, une direction, etc…
- la déclaration des différentes étapes de l'animation.

7.4.1. La déclaration de l'animation

On peut utiliser les différentes propriétés:

```
.test {
    animation-duration: 2s;
    animation-name: slideLeft;
}
```

Ou la propriété raccourcie:

```
.test {
    animation: slideLeft 2s;
}
```

Les différentes propriétés sont:

- **animation-delay**: le délai initial en secondes
- **animation-direction**: pour indiquer si l'animation doit faire des aller-retour (alternate) doit être jouée normalement (normal) ou inversée (reverse) ou les deux (aleternate-reverse)

- **animation-duration**: la durée de l'animation
- **animation-iteration-count**: pour indiquer le nombre de fois que l'animation est jouée ou utiliser le mot-clé infinite afin de répéter indéfiniment l'animation
- **animation-name**: pour indiquer le nom des étapes (@keyframes) correspondant
- **animation-timing-function**: pour indiquer le nom de la fonction de courbe d'accélération utilisée (fonctions identiques aux transitions)

7.4.2. la déclaration des étapes

Les étapes utilisées par l'animation sont déclarées via le règle ***@keyframes*** suivi du nom de l'animation défini lors de la déclaration de l'animation (ici slideLeft).

Attention de définir un nom d'animation qui comme un nom de variable ne peut contenir que des lettres et des chiffres et doit commencer par une lettre. On peut utiliser le tiret.

À l'intérieur de la règle ***@keyframes*** on va déclarer les différentes étapes soit à l'aide de pourcentages, soit, à l'aide des mots clé from (0%) et to (100%)

```
@keyframes slideLeft {
    from {
        transform: translateX(50%);
    }
    to {
        transform: translateX(0);
    }
}
```

Équivalent à:

```
@keyframes slideLeft {
    0% {
        transform: translateX(50%);
    }
    100% {
        transform: translateX(0);
    }
}
```