webサイトにおけるアニメーション

代表的なものとしては以下の種類がある。 授業内ではCSSアニメーションを取り扱う。

- CSSでのアニメーション
- JSを用いたアニメーション
- Canvas要素を用いたアニメーション(JS)

CSSアニメーションの使い所

- 動作が軽い
- 複雑な動きはむかない

ざっくり以上のような特徴がある。よく見る具体的な使い所は、

- カーソルを乗せたとき(hover)ゆっくり文字色等が変わる
- クリックした時にメニューが展開する

などがある。後述になるが、アニメーションの発火タイミングの制御として極簡単にJS,JQなどにお手伝いして貰うことが多い。

CSSアニメーション

CSSアニメーションは以下の２つがある。
違いに関しては実際に書いてしまうのが一番わかりやすい。

- animation
- transition

1 ：transition

- 以下表のプロパティを用いて動き方を指定する
- アニメーションが開始するタイミングに関しては指定出来ないため**擬似クラスやJS**などで制御する
- 擬似クラスではなくアニメーションを定義するクラス本体に記述するのが一般的(後述)

プロパティ	意味	値
transition-propaty	プロパティを選択	all,opacity等
transition-duration	アニメーションの総時間	100ms,1s等
transition-timing-function	アニメーションの加速度変化	ease,linear等
transition-delay	アニメーションの遅延	100ms,1s等
transition	上記を半角スペース区切りで一括指定	例)all 1s ease 50ms

【transitionで動きをつける流れ(例)】

1. まずデフォルトの状態(CSS)を書く

```
.box {
    width: 100px;
    height: 100px;
    border: 1px solid black;
}
```

2. アニメーションが始まるきっかけを考える
   よく使うのは以下の擬似クラス

```
.box:hover {
    /* カーソルがboxに重なったらという意味の擬似クラス */
}
.box:active {
    /* boxをクリックしたら */
}
```

3. 変化後の状態を上記で定義したクラスに記述する

```
/* 今回はhoverを採用 */
.box:hover {
    width: 200px;
}
/* heightとか変化しない部分については継承されるので記述しなくて良い */
```

4. 1から3の状態に変化する間の動きをtransitionで記述する、終わり

```
.box {
    (略)
    transition: width 1s linear 0.1s;
    /*
    widthを1秒かけてlinearという動きで,カーソルが重なってから0.1秒後にアニメーションする
     */
}
```

- 複数のプロパティに別々のアニメーションを定義する時はカンマで区切って書く

```
/* 例えば上記の例に加えheightも変化している場合 */
.box {
    transition: width 1s linear 0.1s, height 2s ease 0.5s;
}

/* widthもheightも同じ動きでいい場合はpropatyをallにすれば良い */
.box {
    transition: all 1s linear 0.1s;
}
```

【transitionの注意点】

- 何かきっかけ(hoverなど)がないとアニメーションが開始しない
- アニメーションのループが出来ない
- transitionを記述するのは元のクラスの方
- アニメーション出来ないプロパティもある

> 【擬似クラスなどにtransitionを書いた場合】
> 擬似クラスなど、変化後の状態を定義したクラスにtransitionを記述すると、アニメーションが逆再生されなくなります。上記で言えば:hoverクラスに書いてしまった時です。カーソルが外れた時はbox:hoverではなくboxになるわけですが、この時にtransitionが記述されていない状態になってしまうためです。

2：animation

- 以下表のプロパティを用いて動き方を指定する
- アニメーション開始のタイミングは指定してもしなくても良い→常に動いているようなアニメーションが実装できる
- keyframeというものを使う
- 繰り返しができる

プロパティ	意味	値
animation-name	アニメーションの名前を定義	(任意の名前をつける)
animation-duration	アニメーションの総時間	100ms,1s等
animation-timing-function	アニメーションの加速度変化	ease,linear等
animation-delay	アニメーションの遅延	100ms,1s等
animation-iteration-count	繰り返し回数	1,2,infinite(無限ループ)等
animation-direction	再生方向(逆再生とか)	normal,alternate等
animation-fill-mode	開始前・終了後のスタイルを指定	forwards,backwards等
animation-play-state	再生・停止	running,paused
animation	上記を半角スペース区切りで一括指定	(略)

【transitionで動きをつける流れ(例)】

1. クラスに各種animationプロパティを記述

```
.box {
    width: 100px;
    height: 100px;
    opacity: 1;
    animation: wi-op-anime 1s ease 0s infinite alternate;
    /* 不要なプロパティは省略可 */
}
```

2. keyframeでどのプロパティをどう変化させるのか書く

- @keyframeの後にanimetionの名前を書く
- 開始を0%,終了を100%として、変化を書いていく

```
@keyframe wi-op-anime{/* 上記で決めた名前 */
    0% {
        width: 100px;
        opacity: 1;
    }
    50% {
```

```
        width: 43px;
        opacity: 0.2;
    }
    100% {
        width: 88px;
        opacity: 1;
    }
}
```

【animationの注意点】

- 複数適用させる時は下記の通りカンマで区切る

```
@keyframe anime1 {
    /* (略) */
}
@keyframe anime2 {
    /* (略) */
}
.box {
    animation: anime1 1s (略), anime2 (略);

    /*
    animation-name: anime1;
    animation-name: anime2;
    のように書くと、anime2で更新されてしまうので、基本はanimationの形で書きましょう。
     */
}
```

transform

- 要素を回転・拡大・移動などができるプロパティ
- 直接アニメーションとは関係ないがtransition,animationと併せてよく使われる

```
/* 例 */
/* 要素(.box)をX軸方向に100pxずらす */
.box {
    transform: translateX(100px);
}
```

値	意味	記述例	備考
translate	指定した方向(X,Y,Z)へ移動	translateY(-20px)	マイナス値,%等も可
rotate	指定した方向(X,Y,Z)へ回転	rotateZ(45deg)	-
scale	指定した方向(X,Y,Z)へ伸縮	scaleX(100px)	-
skew	指定した方向(X,Y)へ傾斜	skew(30deg)	Zはありません

【transform-origin】

- 要素の基点を設定できる
- rotateZと併用するとわかりやすい

```
.box {
    transform: rotateZ(180deg);
    transform-origin: 50% 50%;
    /* transform-origin: x方向 y方向; */
    /* %の場合は要素のwidthやheightが基準 */
    /* これで要素の中心を基点として回転するようになる */
}
```

【transformの注意点】

- 複数指定するときは半角スペースでかく

```
.box {
    transform: translateX(30px) rotateZ(90deg);
}

/* 下記のように書くと、一番下のrotateZのみ適用されてしまう */
.box {
    transform: translateX(30px);
    transform: translateY(42px);
    transform: scale(22px);
```

```
    transform: rotateZ(90deg);
}
```

他にも色々と指定が出来るので調べたり、実際に動かしたりしよう！

JSでアニメーションを発火する

【アニメーションを開始するタイミング】
transitionの場合は何かしらアニメーションを発火するきっかけ・イベントが必要。
hover等の擬似クラス以外では、JSでの制御がよくある。

【概要】

1. クラスの付け外しを行うことで状態変化をさせる
2. 下記のようにcssに記述

```
/* 例 */
.element {
    background-color: blue;
    transition: (略);
}

.element.red {
    /* ↑半角スペースを入れない！ */
    /* elementとred２つのクラスを同時に持っている要素という意味 */
    background-color: red;
}
```

3. JS側で任意のタイミングでredクラスを付けたり外したりすることでアニメーションが実現される

```
// 分かりやすさの為にjQueryで書いてます
$(function(){
    $('.element').on('click', function(){
        $(this).toggleClass('red');
    })
})

.elementが→『$('element')』
クリックされる度→『.on('click'function(){}』
.elementに→『$(this)』
.redというクラスを付け外しする→『.toggleClass('red')』

// ざっくりこのような流れ。
```