Animace

Animace nejspíše znáte. Pro náš předmět si je definujeme tak, že se jedná o zdánlivé pohybování objektu na monitoru. Monitory neumí tvořit však plynulý pohyb a proto musíme používat různé okliky, abychom je viděli. Jednotlivé animace se skládají z snímku - rámců či frames.

Pro to, aby byl pohyb plynulý, musíme ošálit oko. To se dělá tím, že použijeme správný počet FPS, který ošálí oko tak, že si bude myslet, že se jedná o plynulý pohyb. Oko lze ošálit už okolo 24 snímku za sekundu, obyčejně se používá 28-33, případně více.

Definovat 24 či více snímku je však těžké a proto se používá interpolace. Interpolace je způsob jak vyplnit nějaké nedefinované místo obsahem, který získáme z výpočtu předchozího a následujícího snímku. Tím pádem třeba definujeme dva snímky - na začátku a na konci, po 30s. Zbytek vyplníme dopočtem takových hodnot, aby se např. prvek posunoval postupně.

Na WBA ukážeme tři typy animací:

- přechodové animace (CSS)
- *pravá* animace (CSS)
- pokročilé animace (JS)

Přechodové animace

Přechodové animace definují to, jak se prvku v CSS změní vlastnosti, když jej změníme. Pod změnou si můžeme přestavit to, že se mu zničeho nic zaktivují nějaké pravidla, třídy nebo např. že na něm najedeme myší. Přechodovými animacemi určíme, jak se tyto vlastnosti z pravidel spojí do výsledného požadovaného stylu.

Tyto animace určujeme vlastností `transition`, která bere tři hodnoty - jaké vlastnosti se pro danou vlastnost budou kontrolovat (nebo všechny pomocí `all`), jak dlouho bude přechod trvat a jakou přechodovou funkci použijeme.

Např.:

```
.box {
    width: 200px;
    height: 200px;
    background-color: #a782ac;
}

.box:hover {
    background-color: #92a8ca;
    border-radius: 50px;
```

```
    transition: all 0.5s ease; /* všechny vlastnosti se za 0.5s pomocí "ease" přetransformu
}
```

Výše uvedený příklad funguje pouze při najetí myší, při odstranění myši z prvku je přechod neexistují. Je to tím, že vlastně při přechodu ihned pravidlo `.box:hover` přestane platit a tím pádem nemáme definovaný transition. Proto jej musíme vložit i (nebo jen) do pravidla `.box`.

V `transition` můžeme určovat i více vlastností a jejich speciální přechody:

```
transition: width 3s ease, height 3s ease, background-color 10s linear, border-radius 2s ea
```

Přechodové funkce nám určuje, jak interpolace mezi stavy bude vypadat. Existuje mnoho typů, naleznete je ZDE.

Je důležité si uvědomit, že interpolace nefunguje mezi všemi typy vlastností. Existují totiž vlastnosti s výčtem hodnot či jen s hodnotami, které nelze rozdělit. Jedná se např. od `display`, `visibility` a další. Tyto vlastnosti se proto zaktivují okamžitě.

Pravé animace

Pravé animace jsou animace, kde již přesně definujeme jednotlivé rámce animace. Určujeme tedy v jakém momentu bude mít prvek jaké vlastnosti. Animace definujeme v pravidlu `@keyframes` a můžeme ji používat na více prvků.

Animace zapínáme pomocí vlastnosti `animation`, kde určujeme její jméno, délku, přechodovou funkci, počet iterací a další. Velmi často se používá vlastnost `animation-play-state`, kterou určujeme, zda animace běží či nikoliv. Můžeme ji tedy v průběhu zastavit a poté volně navázat.

Animace tedy definujeme např. následovně:

```
.box {
    width: 200px;
    height: 200px;
    background-color: #a782ac;

    animation-name: boxAnimation;
    animation-timing-function: ease;
    animation-duration: 3s;
    animation-play-state: running;
    animation-iteration-count: infinite;
}
```

```
@keyframes boxAnimation {
    from { /* Ekvivalentní 0% */
        width: 100px;
        height: 100px;
        background-color: #a782ac;
    }

    50% {
        width: 300px;
        height: 300px;

        border-radius: 50%;

        background-color: #8884cc;
    }

    to {
        width: 100px;
        height: 100px;
        background-color: #a782ac;
    }
}
```

Problémové bývá to, když začátek a konec, buď 0, from a nebo 100, to nemá stejné hodnoty. Když tyto hodnoty nejsou stejné, tak se animace zasekává právě na těchto rámcích. Stejně jako u ostatních animací se ostatní rámce mezi těmito hodnotami vyplní pomocí interpolace.

Pokročilé animace

Velmi se podobá pravým animacím, rozdíl je v tom, že neurčujeme kdy se daný rámec zobrazí a vše píšeme uvnitř JS. Výhody pokročilých animacích jsou takové, že je můžeme programovat, programování taktéž znamená, že správně můžeme zachytit taktéž konec animace a nebo ji třeba i předčasně ukončit.

Celé animace probíhají pomocí volání metody `.animate` na Elementu z DOMu, který chceme animovat:

```
const box = document.getElementsByClassName("box")[0];

const animation = box.animate([
    {
```

```
        backgroundColor: "#a782ac",
        transform: "scale(1.0)"
    },
    {
        backgroundColor: "#aabbcc",
        transform: "scale(1.5)"
    },
    {
        backgroundColor: "#a7bb11",
        transform: "scale(1.7)"
    }
], {
    duration: 1500, // 1.5s,
    easing: "ease",
    iterations: 3,
});

animation.addEventListener('finish', function () {
    box.style.transform = 'scale(1.7)';
    box.style.backgroundColor = '#a7bb55';
    box.style.borderRadius = '10px';
});
```