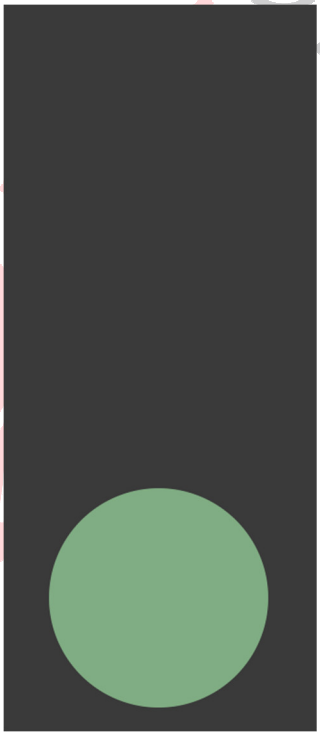
Osnove JavaScript animacije

Animacija se unutar web aplikacija može kreirati korišćenjem CSS i JavaScript jezika. Prethodna lekcija ilustrovala je osnovne načine za kreiranje animacije korišćenjem CSS jezika. Imali ste prilike da vidite da je reč o pristupima koji omogućavaju da se animacija kreira na vrlo jednostavan način. Ipak, takvi pristupi su limitirani u pogledu dinamičnosti i mogućnosti kontrolisanja, pa se stoga primenjuju isključivo za postizanje animacije elemenata grafičkog korisničkog okruženja. Ukoliko je potrebno kreirati napredniju, dinamičku animaciju ili igru, pribegava se korišćenju JavaScript jezika. Mi ćemo se u nastavku ovoga kursa isključivo baviti animacijom koja se postiže upotrebom JavaScript jezika, a lekcija koja je pred Vama poslužiće kao osnova za ulazak u svet JavaScript animacije.

Prvi primer JavaScript animacije

U ovoj lekciji će odmah biti prikazan prvi primer animacije koja se postiže korišćenjem JavaScript jezika. Primer će nam pomoći da u nastavku uvidimo sve najznačajnije osobine procesa kreiranja JavaScript animacija.

Primer će izgledati kao na animaciji 3.1.



Animacija 3.1 - Efekat prvog primera animacije kreirane JavaScript jezikom

Primer podrazumeva animiranje jednog `div` HTML elementa:

```
<div id="my-div"></div>
```

Ovakav `div` element i `body` unutar koga se on nalazi, stilizovani su na sledeći način:

```
body {
    height: 100vh;
    margin: 0px;
    background-color: #3a3a3a;
}

#my-div {
    background-color: #80ad83;
    border-radius: 100%;
    width: 240px;
    height: 240px;
    margin: 0 auto;
}
```

HTML i CSS kodom obavljeno je kreiranje jednog `div` elementa koji unutar web pregledača stvara prikaz jednog kruga, baš kao što je to prikazano animacijom 3.1. Kako bi takav krug oživeo i počeo da se kreće gore-dole, potrebno je definisati sledeći JavaScript kod:

```
let body = document.getElementById("body");
let myDiv = document.getElementById("my-div");

let direction = +1;
let speed = 8;

function update() {

    bodyBounds = body.getBoundingClientRect();
    myDivBounds = myDiv.getBoundingClientRect();

    myDiv.style.transform = "translateY(" + (myDivBounds.top + direction * speed) + "px)";

    if (myDivBounds.bottom > bodyBounds.bottom - 2*speed) {
        direction = -1;
    } else if (myDivBounds.top < bodyBounds.top + 2*speed) {
        direction = 1;
    }
}

setInterval(update, 17);
```

Unutar prikazanog JavaScript koda, prvo se dolazi do reference na DOM objekte koji predstavljaju `body` element i `div` element koji je potrebno animirati. Zatim se definišu dve promenljive (`direction` i `speed`) koje će odrediti osobine animacije, odnosno usmerenje kretanja kruga koji se animira i brzinu animacije.

Funkcija `update()` koristi se za ažuriranje osobina kruga. Takva funkcija se poziva na svakih 17 milisekundi, što je postignuto korišćenjem `setInterval()` funkcije. Tako će na svakih 17 milisekundi, krug biti pomeren za nekoliko piksela dole ili gore, u zavisnosti od usmerenja animacije.

Pomeranje kruga se obavlja korišćenjem CSS transformacija. Koristi se *translateY* transformacija, koja omogućava pomeranje elementa po Y osi (vertikalnoj osi) koordinatnog sistema web dokumenta. Vrednost `transform` CSS svojstva formira se u zavisnosti od unapred utvrđene brzine animacije (promenljiva `speed`) i položaja unutar dokumenta na kome se nalazi krug koji se animira. Naime, animacija se prvo obavlja tako što se krug spušta do dna stranice, a zatim se animacija obavlja u suprotnom smeru, sve do vrha dokumenta, kada opet dolazi do promene usmerenja.

Usmerenje animacije definiše se promenljivom `direction`. Kada ova promenljiva ima pozitivnu vrednost (1), animacija podrazumeva spuštanje kruga ka dnu stranice. Kada promenljiva `direction` dobije negativnu vrednost (-1), krug menja smer kretanja.

Vrednost svojstva `direction` formira se u zavisnosti od položaja kruga u odnosu na okvire vidnog polja web pregledača. Pozicija nekog elementa u odnosu na vidno polje (*viewport, engl.*) dobija se korišćenjem metode `getBoundingClientRect()`.

Metoda getBoundingClientRect()

Metoda `getBoundingClientRect()` omogućava dobijanje pozicije nekog HTML elementa, relativno vidnom polju. Pored pozicije, ova metoda omogućava dobijanje i visine i širine koju element zauzima na stranici. Kao svoju vrednost, metoda `getBoundingClientRect()` vraća objekat `DOMRect`, koji poseduje sledeća svojstva:

- `x` - x koordinata referentne tačke elementa (udaljenost leve ivice elementa, od leve ivice vidnog polja)
- `y` - y koordinata referentne tačke elementa (udaljenost gornje ivice elementa, od gornje ivice vidnog polja)
- `top` - udaljenost gornje ivice elementa, od gornje ivice vidnog polja; identična vrednost kao i svojstvo `y`
- `bottom` - udaljenost donje ivice elementa, od gornje ivice vidnog polja; zbir vrednost svojstava `y` i `height`
- `left` - udaljenost leve ivice elementa, od leve ivice vidnog polja; identična vrednost kao i `x`
- `right` - udaljenost desne ivice elementa, od leve ivice vidnog polja; zbir vrednosti svojstava `x` i `width`
- `width` - širina nevidljivog pravougaonika unutar koga se nalazi element
- `height` - visina nevidljivog pravougaonika unutar koga se nalazi element

U prikazanom primeru metoda `getBoundingClientRect()` koristi se kako bi se došlo do položaja kruga koji se animira i kako bi se na kraju, utvrdilo kada je animacija potrebno da promeni smer.

Tajming funkcije

U upravo prikazanom primeru, čiji je cilj bio da pokaže osnovni princip na kome se zasniva JavaScript animacija, mogli ste videti da je osnovni motor animacije metoda `setInterval()`. Ona omogućava da se funkcija - `update()` izvršava na svakih 17 milisekundi i da se pomeranjem `div` elementa u svakoj iteraciji, stvori utisak kretanja. `setInterval()` je jedna od tajming funkcija koje postoje unutar `window` objekta web pregledača, a koje čine osnovu svake JavaScript animacije. U nastavku ove lekcije upoznaćemo se sa osnovnim osobinama svih takvih funkcija.

Unutar `window` objekta nalaze se sledeće tajming funkcije:

- `setTimeout()` – izvršava neku funkcionalnost jednom, nakon što protekne određena količina vremena
- `setInterval()` – izvršava određenu funkcionalnost iznova i iznova, sa određenim fiksnim razmacima između svakog izvršavanja
- `requestAnimationFrame()` - poziva prosleđenu funkciju neposredno pre nego što je web pregledač spreman da obavi ponovno crtanje korisničkog okruženja

Tri upravo navedene funkcije, u nastavku lekcije biće razmotrene iz ugla kreiranja animacije korišćenjem JavaScript jezika.

setTimeout()

`setTimeout()` je funkcija koja omogućava da se određeni kod izvrši jedanput u budućnosti, nakon određenog vremenskog perioda. Sintaksa ove funkcije je sledeća:

```
setTimeout(function, delay, arg);
```

Može se videti da funkcija `setTimeout()` može da prihvati sledeće parametre:

- `func` – funkcija koja će se izvršiti nakon nekog vremenskog perioda; ovo je obavezni parametar
- `delay` – vremenski period nakon koga će prosleđena funkcija biti aktivirana; vrednost se izražava u milisekundama; ovaj parametar nije obavezan i ukoliko se ne prosledi, za njegovu vrednost se uzima 0
- `arg` – proizvoljan broj parametara koji će biti prosleđeni `func` funkciji

Funkcija `setTimeout()` se za potrebe kreiranja JavaScript animacije može upotrebiti u kombinaciji sa rekurzijom. Tako se već viđeni primer može transformisati ovako:

```
function update() {

//frame creation and drawing logic

    setTimeout(update, 17);
}

setTimeout(update, 17);
```

S obzirom da metoda `setTimeout()` poziva prosleđenu funkciju samo jednom, nakon određenog vremenskog perioda, na kraj funkcije `update()` postavljen je poziv `setTimeout()` metode. Na ovaj način dobija se identičan efekat kao i u prvom primeru, ali je realizacija nešto drugačija.

setInterval()

Još jedna tajming funkcija koju web pregledači izlažu na korišćenje kodu koji mi samostalno pišemo jeste i funkcija `setInterval()`. Za razliku od `setTimeout()` funkcije, ova funkcija omogućava da se logika jedne funkcije izvršava iznova i iznova, sa jednakim, unapred definisanim razmacima između izvršavanja. Stoga je ona znatno pogodnija za postizanje animacije, s obzirom da nas oslobađa potrebe za kreiranjem rekurzije.

Metoda `setInterval()` koristi se na identičan način kao i već prikazana `setTimeout()` metoda:

```
setInterval(func, delay, arg);
```

Može se videti da je sintaksa metode `setInterval()` identična sintaksi `setTimeout()` metode. Ona može da prihvati sledeće parametre:

- `func` – funkcija koja će se izvršavati iznova i iznova
- `delay` – vremenski razmak između dva susedna poziva `func` funkcije; vrednost se izražava u milisekundama; u ovom slučaju je reč o obaveznom parametru; ipak većina web pregledača kao minimalni razmak između dva izvršavanja definiše vrednost od 4ms
- `arg` – proizvoljan broj parametara koji će biti prosleđeni `func` funkciji

Funkcija `setInterval()` je na početku ove lekcije iskorišćena za realizaciju uvodnog primera animacije, s obzirom da omogućava da se ciklično izvršavanje nekog koda obavi na nešto jednostavniji i kompaktniji način, bez potrebe za rekurzijom.

Otkazivanje tajmera i intervala

Otkazivanje logike koja se prosleđuje `setInterval()` i `setTimeout()` metodama može se obaviti korišćenjem sledećih metoda:

- `clearTimeout()` - za otkazivanje izvršavanja logike koja je prosleđena `setTimeout()` metodi
- `clearInterval()` - za otkazivanje izvršavanja logike koja je prosleđena `setInterval()` metodi

Metode `setInterval()` i `setTimeout()` kao svoju povratnu vrednost emituju podatak koji se odnosi na identifikator tajmera, odnosno intervala. Takav identifikator se koristi za otkazivanje logike koja je prosleđena na izvršavanje.

Primer otkazivanja tajmera:

```
let timeoutId = setTimeout(update, 17);
clearTimeout(timeoutId);
```

Primer otkazivanja intervala:

```
let intervalId = setInterval(update, 17);
clearInterval(intervalId);
```

Funkcionalnost koja se od strane JavaScript izvršnog okruženja koristi za generisanje identifikatora tajmera i intervala je zajednička. Drugim rečima, ne može se dogoditi da interval i tajmer poseduju identičan identifikator. Stoga, bez obzira da li je reč o intervalu ili tajmeru, Vi možete iskoristiti bilo koju od ove dve funkcije za otkazivanje. Ipak, zbog preglednosti i boljeg razumevanja koda koji se piše, savetuje se korišćenje funkcije odgovarajućeg naziva.

U nastavku će logika za otkazivanje tajmera i intervala biti iskorišćena za dopunu već kreiranog primera, sa početka ove lekcije. Naime, biće omogućeno stopiranje i ponovno nastavljanje animacije, klikom bilo gde unutar površine web stranice. Kompletna JavaScript logika će izgledati ovako:

```
        let intervalId;

        let body = document.getElementById("body");
        let myDiv = document.getElementById("my-div");
        let direction = +1;
        let speed = 8;
        function update() {
            bodyBounds = body.getBoundingClientRect();
            myDivBounds = myDiv.getBoundingClientRect();
            myDiv.style.transform   =   "translateY("   +   (myDivBounds.top   +
direction * speed) + "px)";

            if (myDivBounds.bottom > bodyBounds.bottom - 2 * speed) {
                direction = -1;
            } else if (myDivBounds.top < bodyBounds.top + 2 * speed) {
                direction = 1;
            }
        }
        intervalId = setInterval(update, 17);
        body.onclick = function () {
            if (intervalId) {
                clearInterval(intervalId);
                intervalId = undefined;
            } else {
                intervalId = setInterval(update, 17);
            }
        }
```

Primer je sada proširen jednom promenljivom `intervalId`, koja se popunjava vrednošću koja predstavlja identifikator intervala koji se kreira korišćenjem `setInterval()` metode. Definisana je i logika koja će se aktivirati prilikom klika na `body` element. U takvim situacijama, proveravaće se vrednost `intervalId` promenljive i ukoliko je različita od `undefined`, obavlja će se zaustavljanje intervala, što za efekat ima pauziranje animacije. Animacija će biti zaustavljena u onom stanju u kome se nalazi prilikom klika.

Prilikom zaustavljanja animacije, vrednost promenljive `intervalId` se postavlja na `undefined`, pa se tako prilikom sledećeg klika izvršava logika koja je definisana unutar `else` uslovnog bloka. Takva logika će obaviti kreiranje novog intervala, što će za posledicu imati da se animiranje nastavi tamo gde je prilikom prethodnog klika zaustavljeno.

Na ovaj način kreirana je logika koja će klikom bilo gde unutar HTML dokumenta, naizmenično zaustavljati i pokretati animaciju.

Vreme definisano `delay` parametrom nije zagarantovano

U prethodnim redovima ste mogli da vidite da funkcije `setTimeout()` i `setInterval()`, kao jedan od svojih parametara prihvataju vreme izraženo u milisekundama, nakon koga će biti pozvana prosleđena callback funkcija. Ipak, vrlo je bitno znati da vreme koje se definiše takvim `delay` parametrom, nije zagarantovano.

Callback funkcije koje se prosleđuju `setTimeout()` i `setInterval()` metodama izvršavaju se asinhrono, tek kada se stek poziva oslobodi. Stoga, ukoliko se nakon poziva ovih metoda nalazi neka vremenski zahtevna operacija, period koji će proći do početka izvršavanja funkcije može biti i duži od onoga koji smo mi kao parametar naveli. Ipak, bitno je znatni da period nikada ne može biti kraći od definisanog.

requestAnimationFrame()

Funkcije `setTimeout()` i `setInterval()` mogu se okarakterisati kao funkcije opšte namene koje je moguće koristiti za odloženo i ciklično izvršavanje nekog koda. Iako su, kao što ste mogli videti, sasvim upotrebljive za postizanje animacije, web pregledači nama na korišćenje izlažu još jednu metodu koja ja znatno prikladnija za ažuriranje osobina elemenata koji se animiraju. Reč je o metodi `requestAnimationFrame()`.

Metoda `requestAnimationFrame()` može da prihvati samo jedan parametar:

```
requestAnimationFrame(callback);
```

Parametar koji `requestAnimationFrame()` metoda prihvata, odnosi se na funkciju koju je potrebno izvršiti neposredno pre narednog osvežavanja korisničkog okruženja od strane web pregledača. Funkcija koja se prosleđuje `requestAnimationFrame()` metodi treba da sadrži logiku koja će ažurirati vizualne karakteristike elemenata koji se animiraju. Upravo takva je i funkcija `update()` iz prethodnog primera, pa je tako već viđeni primer moguće modifikovati na sledeći način:

```
        let requestId;

        let body = document.getElementById("body");
        let myDiv = document.getElementById("my-div");

        let direction = +1;
        let speed = 8;

        function update() {

            bodyBounds = body.getBoundingClientRect();
            myDivBounds = myDiv.getBoundingClientRect();
            myDiv.style.transform   =   "translateY("   +   (myDivBounds.top   +
direction * speed) + "px)";
            if (myDivBounds.bottom > bodyBounds.bottom - 2 * speed) {
                direction = -1;
            } else if (myDivBounds.top < bodyBounds.top + 2 * speed) {
                direction = 1;
            }

            requestId = requestAnimationFrame(update);
        }

        requestId = requestAnimationFrame(update);
        body.onclick = function () {
            if (requestId) {
                cancelAnimationFrame(requestId);
                requestId = undefined;
            } else {
                requestId = requestAnimationFrame(update);
            }

        }
```

U odnosu na prethodni primer, sada su pozivi metode `setInterval()`, zamenjeni pozivima metode `requestAnimationFrame()`. Iz primera se može zaključiti nekoliko veoma važnih osobina kreiranja animacije upotrebom metode `requestAnimationFrame()`:

- metoda `requestAnimationFrame()` prihvata samo jedan parametar koji se odnosi na callback funkciju koja je potrebno da ažurira osobine elemenata koji se animiraju
- prilikom korišćenja metode `requestAnimationFrame()` ne definiše se vreme, odnosno zadrška (*delay, engl.*), zato što će web pregledač samostalno da obavi pozivanje prosleđene metode, neposredno pre narednog osvežavanja korisničkog okruženja
- metoda `requestAnimationFrame()` kao svoju povratnu vrednost emituje kod koji jednoznačno određuje zahtev
- zahtev za izvršavanjem callback funkcije moguće je otkazati upotrebom metode `cancelAnimationFrame();` njoj se tom prilikom prosleđuje identifikator zahteva
- pozivanje metode `requestAnimationFrame()` kreira zahtev za izvršavanjem određene logike samo neposredno pre prvog sledećeg osvežavanja; stoga je za kontinuirano animiranje neophodno samostalno, uvek iznova pozivati metodu `requestAnimationFrame()`, iz callback funkcije; to je u primeru obavljeno na kraju metode `update()`

Iz upravo prikazanog primera mogli ste videti da je osnovna osobina `requestAnimationFrame()` metode, odsustvo parametra za definisanje vremena. Jednostavno web pregledač će samostalno u određenom vremenskom trenutku da pozove funkciju koja se prosleđuje `requestAnimationFrame()` metodi. Na taj način se logika kojom se kontroliše frekvencija izvršavanja funkcije za animiranje, prepušta web pregledaču. Moderni web pregledači frekvenciju osvežavanja korisničkog okruženja vezuju za frekvenciju osvežavanja displeja korisničkog uređaja, pa se tako osigurava tečna i glatka animacija.

Pitanje

Koja tajming funkcija omogućava da se pozivanje metode za ažuriranje animacije, obavi neposredno pre narednog osvežavanja korisničkog okruženja:

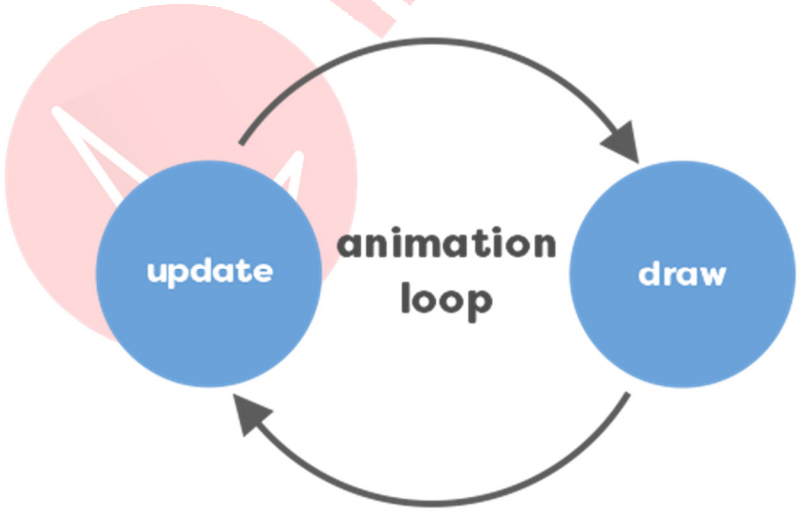
a) `setTimeout()`
b) `setInterval()`
c) `requestAnimationFrame()`

Objašnjenje:

Metoda `requestAnimationFrame()` poziva prosleđenu funkciju neposredno pre nego što je web pregledač spreman da obavi ponovno crtanje korisničkog okruženja.

Anatomija JavaScript animacije

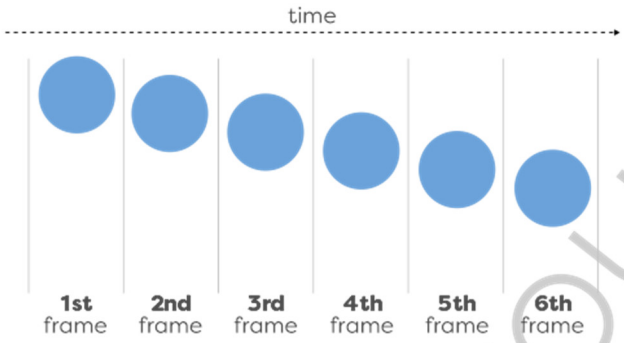
U prethodnim redovima prikazan je vrlo jednostavan primer animacije koja je postignuta korišćenjem JavaScript jezika. Mogli ste videti da se animacija zasniva na konstantnom ažuriranju vizuelnih osobina elementa koji se animira. Drugim rečima, jedna funkcija poziva se približno 60 puta u jednoj sekundi i unutar nje se obavlja ažuriranje vizuelne prezentacije elementa. Sve to je ilustrovano slikom 3.1.



Slika 3.1 - Glavna petlja animacije

Slika 3.1 ilustruje način na koji funkcioniše animacija kreirana u ovoj lekciji. Cikličnim ponavljanjem logike za ažuriranje grafičkih elemenata, stvara se takozvana **glavna petlja animacije** (*main animation loop, engl.*).

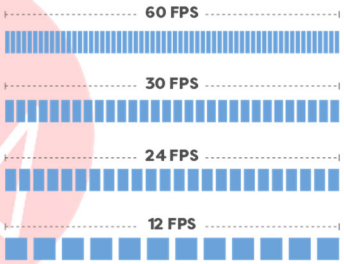
Svakom iteracijom unutar petlje animacije, dobija se jedan po jedan **kadar animacije** (*animation frame, engl.*). Tako je animacija sačinjena iz mnoštva kadrova koji se pred očima korisnika brzo smenjuju i stvaraju privid kretanja (slika 3.2).



Slika 3.2 - Kadrovi animacije

Nešto ranije je rečeno da web pregledači nastoje da korisničko okruženje web sajtova osvežavaju onoliko puta u sekundi, koliko je displej uređaja sposoban da prikaže. Većina displeja može da prikaže 60 kadrova u sekundi, što praktično znači da će u takvoj situaciji animacija biti ažurirana 60 puta u jednoj sekundi.

Brzina smenjivanja kadrova drugačije se naziva **frame rate**, frekvencija osvežavanja ili broj kadrova po sekundi (*frames per second - FPS, engl.*). Primeri različite frekvencije osvežavanja, ilustrovani su slikom 3.3.



Slika 3.3 - Broj kadrova u sekundi

Upravo spomenuti pojmovi (*main loop, frames, frames per second*) čine osnovu svake animacije. Ipak, u ovoj lekciji prikazan je najjednostavniji način za kreiranje glavne petlje, bez bilo kakvog zalaženja u specifičnosti takvog postupka. Takođe, animacija je realizovana nad HTML elementom, dok se u praksi JavaScript primarno koristi za animiranje grafike koja se crta unutar jednog posebnog HTML elementa - `canvas`. Stoga će priča o animaciji, biti nastavljena nakon što se u narednim lekcijama upoznamo sa crtanjem grafike unutar `canvas` elementa.

Rezime

- kreiranje naprednije, dinamičke animacije ili igre, podrazumeva korišćenje JavaScript jezika
- metoda `getBoundingClientRect()` omogućava dobijanje pozicije nekog HTML elementa, relativno vidnom polju
- JavaScript animacija postiže se kontinuiranim ponavljanjem logike za ažuriranje vizuelnih karakteristika elementa
- kontinuirano ponavljanje logike za ažuriranje vizuelnih karakteristika elementa, može se obaviti korišćenjem tajming funkcija
- unutar `window` objekta nalazi se nekoliko tajming funkcija: `setTimeout()`, `setInterval()` i `requestAnimationFrame()`
- `setTimeout()` izvršava neku funkcionalnost jednom, nakon što protekne određena količina vremena
- `setInterval()` izvršava određenu funkcionalnost iznova i iznova, sa određenim fiksnim razmacima između svakog izvršavanja
- `requestAnimationFrame()` poziva prosleđenu funkciju neposredno pre nego što je web pregledač spreman da obavi ponovno crtanje korisničkog okruženja
- cikličnim ponavljanjem logike za ažuriranje grafičkih elemenata, stvara se takozvana glavna petlja animacije (*main animation loop, engl.*)
- svakom iteracijom unutar petlje animacije, dobija se jedan po jedan kadar animacije (*animation frame, engl.*).
- brzina smenjivanja kadrova drugačije se naziva frame rate, frekvencija osvežavanja ili broj kadrova po sekundi (*frames per second - FPS, engl.*)