

CSS tranzicija i animacija

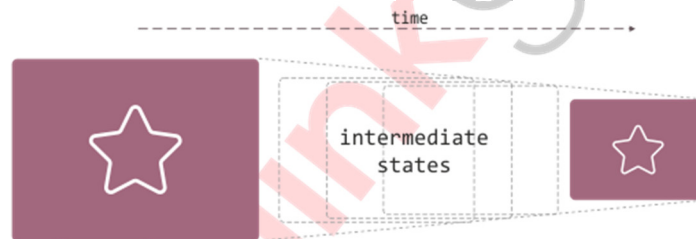
Najjednostavniji način za kreiranje animacije unutar web pregledača, podrazumeva korišćenje CSS jezika. CSS jezik poseduje dva skupa funkcionalnosti pomoću kojih je moguće definisati animaciju koja će se obaviti nad elementima grafičkog korisničkog okruženja web aplikacija. Takvi skupovi funkcionalnosti slikovito se nazivaju:

- CSS tranzicije (*CSS transitions, engl.*)
- CSS animacije (*CSS animations, engl.*)

Oba CSS sistema za animiranje omogućavaju identično - postepenu promenu vizuelnih osobina elemenata tokom određenog vremenskog perioda, čime se stvara privid animacije. Ipak, *CSS animations* sistem omogućava znatno preciznije kontrolisanje animacije. O takvoj, ali i ostalim osobinama CSS sistema za animiranje, imaćete prilike da čitate u lekciji koja je pred Vama.

Šta su CSS tranzicije?

Postepeni prelazak elementa iz jednog oblika u drugi, tokom određenog vremenskog perioda naziva se tranzicija. Tako CSS tranzicije omogućavaju da je jedan element pređe iz jednog u drugo stanje, ali postepeno, tokom određenog vremenskog perioda (slika 2.1).



Slika 2.1 – CSS tranzicija

Slika 2.1 ilustruje način na koji se obavlja jedna CSS tranzicija. Na levoj strani slike 2.1, prikazan je početni izgled elementa, a na desnoj završni. Veličina elementa se postepeno, tokom određenog vremenskog intervala menja, pri čemu sam web pregledač brine o adekvatnom prikazu svih međustanja, koja su na slici 2.1 ilustrovana isprekidanim linijama između početnog i završnog prikaza.

Kako se definišu CSS tranzicije?

CSS tranzicije se definišu korišćenjem svojstva **transition**. Prilikom definisanja tranzicije, obavezno je definisati dva podatka:

- naziv svojstva čija vrednost će se postepeno menjati
- dužina trajanja promene

Sledeći primer će ilustrovati definisanje jedne tranzicije. Dokument treba da poseduje jedan `div` element:

```
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</div>
```

Ovakav `div` element može biti stilizovan na sledeći način:

```
div {  
  background-color: #A2687E;  
  color: white;  
  padding: 16px;  
  margin: 32px auto;  
  font-family: sans-serif;  
  font-size: 28px;  
  width: 250px;  
  height: 180px;  
  border-radius: 8px;  
  transition: width 1s;  
}
```

Unutar prikazanog CSS opisa, bitno je primetiti poslednju deklaraciju:

```
transition: width 1s;
```

Na ovaj način, definisana je CSS tranzicija koja govori da je vrednost svojstva `width` potrebno promeniti postepeno i da će takva promena trajati jednu sekundu.

Ipak, definisanjem ovakve CSS deklaracije nad nekim HTML elementom, na stranici se automatski neće dogoditi nikakva promena. Da bi se efekat tranzicije video, neophodno je da dođe do promene vrednosti svojstva definisanog tranzicijom (u prikazanom primeru to je svojstvo `width`).

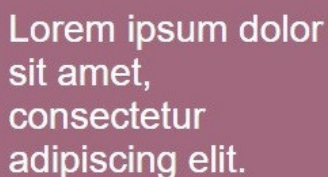
Tako se može zaključiti da je iniciranje CSS tranzicije moguće obaviti na dva načina:

- programabilnom intervencijom nad stilizacijom, korišćenjem JavaScript jezika
- promenom vrednosti CSS svojstava upotrebom pseudo klasa, kada se element nađe u nekom specifičnom stanju

Jedan od načina za iniciranje tranzicije može da izgleda ovako:

```
div:hover {  
  width: 320px;  
}
```

Na ovaj način, prelaskom pokazivača miša preko `div` elementa, započeće animiranje takvog elementa (animacija 2.1).



Lorem ipsum dolor
sit amet,
consectetur
adipiscing elit.

Animacija 2.1 – Tranzicija širine div elementa koja se aktivira prelaskom strelice miša

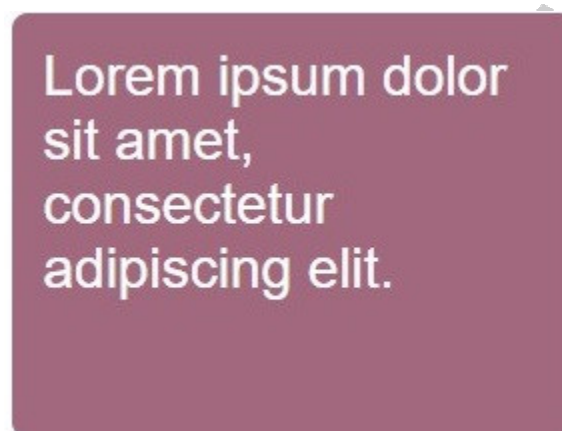
Moguće je u istom trenutku definisati tranziciju većeg broja CSS svojstva:

```
transition: width 1s, height 1.5s;
```

Ovoga puta, tranzicija je definisana za `width` i `height` svojstva. Kako bi se obe tranzicije pokrenule, neophodno je promeniti vrednosti oba CSS svojstva:

```
div:hover {  
    width: 320px;  
    height: 220px;  
}
```

Sada će biti dobijen efekat kao na animaciji 2.2.



Animacija 2.2 – Tranzicija širine i visine div elementa koja se aktivira prelaskom strelice miša

CSS svojstva nad kojima se može upotrebiti tranzicija

CSS tranzicije se mogu primeniti nad gotovo svim CSS svojstvima. Spisak takvih CSS svojstava je zaista dug, ali bitno je reći da se postepena promena vrednosti nekog CSS svojstva, može postići samo nad onim svojstvima kod kojih tako nešto ima smisla. Na primer, tranziciju nije moguće definisati nad svojstvom `font-family`.

Još jedna zanimljivost odnosi se na nemogućnost definisanja tranzicija nad svojstvom `display`. Ova činjenica veoma često zbunjuje početnike koji pokušavaju da fade-in ili fade-out efekte postignu korišćenjem tranzicije i promene vrednosti `display` svojstva sa `block` na `none`. Da bi se spomenuti efekti postigli, neophodno je koristiti neka druga svojstva kao što su `opacity` ili `visibility`.

Tajming funkcije tranzicija

Brzina kojom se promena vrednosti nekog svojstva obavlja tokom vremena, naziva se tajming tranzicije, odnosno tajming funkcija. Kontrolise se korišćenjem svojstva **transition-timing-function**. Ovo svojstvo može imati vrednosti prikazane tabelom 2.1.

Vrednost	Opis
ease	spor početak, zatim ubrzanje, spor kraj; ovo je podrazumevana vrednost
linear	jednaka brzina tokom celog trajanja
ease-in	spor početak
ease-out	spor kraj
ease-in-out	spor početak i spor kraj
cubic-bezier(n,n,n,n)	proizvoljno definisanje dinamike izvršavanja navođenjem parametara za kreiranje <u>beziјerove krive</u>

Tabela 2.1 – Vrednosti transition-timing-function svojstva

Korišćenje svojstva `transition-timing-function` ilustrovano je primerom:

```
transition: width 2s;  
transition-timing-function: ease-in;
```

Na ovaj način, definisanjem tajming funkcije je rečeno da će tranzicija započeti nešto sporije, a da će zatim kako se bliži kraju, sve više ubrzavati.

Odloženo izvršavanje tranzicija

Efekat tranzicije je moguće odložiti korišćenjem svojstva **transition-delay**. Na taj način je moguće definisati vremenski interval koji je potrebno da protekne od aktiviranja tranzicije do njenog početka. Na primer, umesto da tranzicija započne odmah kada dođe do promene vrednosti svojstva za koje je tranzicija definisana, tranzicija će biti odložena za vreme definisano svojstvom `transition-delay`.

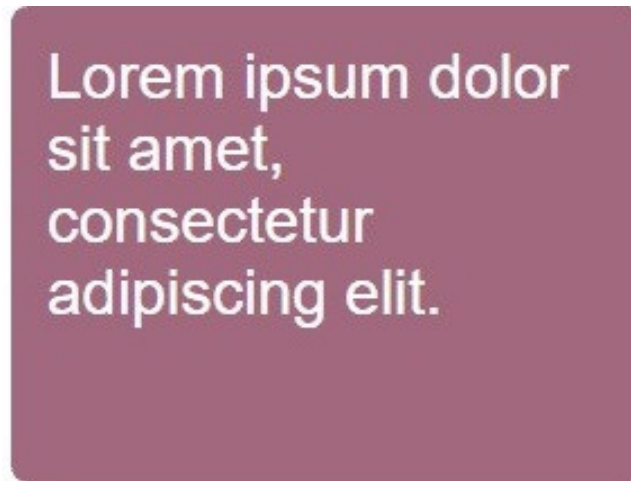
```
div {  
    transition: width 2s;  
    transition-timing-function: ease;  
    transition-delay: 1s;  
}
```

Na ovaj način, CSS tranzicija će započeti dve sekunde nakon promene vrednosti `width` svojstva `div` elementa.

Iniciranje CSS tranzicija korišćenjem JavaScript-a

Još jedan način za iniciranje CSS tranzicija koji se aktivno koristi prilikom razvoja web sajtova i aplikacija, jeste upotreba JavaScript jezika i DOM API-a. Ipak, iako je vrednost CSS svojstava moguće direktno menjati korišćenjem `style` svojstva koje poseduju svi DOM objekti, u praksi se najčešće pribegava intervenciji nad skupom klasa jednog HTML elementa. Naredni primer će između ostalog, ilustrovati i iniciranje CSS tranzicije korišćenjem JavaScript jezika.

Efekat koji će biti dobijen primerom, ilustrovan je animacijom 2.3.



Animacija 2.3 – Efekat primera čija realizacija će biti prikazana u nastavku

Tranzicija se obavlja nad već viđenim `div` elementom:

```
<div id="my-div">Lorem ipsum dolor sit amet, consectetur adipiscing elit.</div>
```

Ovakav `div` element, inicijalno je stilizovan na sledeći način:

```
div {  
  background-color: #A2687E;  
  color: white;  
  padding: 16px;  
  margin: 32px auto;  
  font-family: sans-serif;  
  font-size: 28px;  
  width: 250px;  
  height: 180px;  
  border-radius: 8px;  
  text-decoration: line-through transparent;  
}
```

Mi želimo da klikom na ovakav `div` element izvršimo tranziciju nad bojom pozadine i bojom i dekoracijom teksta. Stoga je unutar CSS deklaracije za stilizovanje ovakvog elementa potrebno dodati sledeći CSS opis:

```
div {
  background-color: #A2687E;
  color: white;
  padding: 16px;
  margin: 32px auto;
  font-family: sans-serif;
  font-size: 28px;
  width: 250px;
  height: 180px;
  border-radius: 8px;
  text-decoration: line-through transparent;

  transition: text-decoration 0.5s, background-color 0.5s, color
0.5s;
}
```

Bitno je da primetite da se nad `text-decoration` svojstvom ne može direktno definisati tranzicija. Stoga se tranzicija obavlja nad bojom dekoracije koja se postavlja nad tekstem (*line-through*).

Tranzicije će biti inicirane dodavanjem jedne posebne klase nad ovakvim elementom:

```
.deleted {
  background-color: #474747;
  color: rgb(148, 148, 148);
  text-decoration-color: white;
}
```

Svakim klikom na `div` element, korišćenjem JavaScript jezika, obavlja se ciklično dodavanje i uklanjanje klase `.deleted`:

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

myDiv.onclick = function() {
  if(myDiv.classList.contains('deleted')){
    myDiv.classList.remove("deleted");
  } else {
    myDiv.classList.add("deleted");
  }
}
```

Ovakvim kodom, prvo se dolazi do reference na DOM objekat koji predstavlja `div` element. Zatim se definiše funkcija koja se aktivira klikom na `div` element. Unutar funkcije, proverava se da li `div` element poseduje klasu `.deleted`. Ukoliko je ne poseduje, obavlja se dodavanje takve klase, a kada ona postoji na elementu, obavlja se njeno uklanjanje.

Primer - Upotreba CSS tranzicija za otvaranje/zatvaranje popup prozora

Upravo prikazane tehnike za definisanje CSS tranzicija, sumiraćemo na jednom realnom primeru. Korišćenjem CSS tranzicija biće dobijen efekat prikazan animacijom 2.4.



Animacija 2.4 - Efekat otvaranja i zatvaranja popup prozora koji je realizovan korišćenjem CSS tranzicija

Kompletan kod primera, možete da preuzmete sa sledećeg linka:

`example-css-transitions-popup.rar`

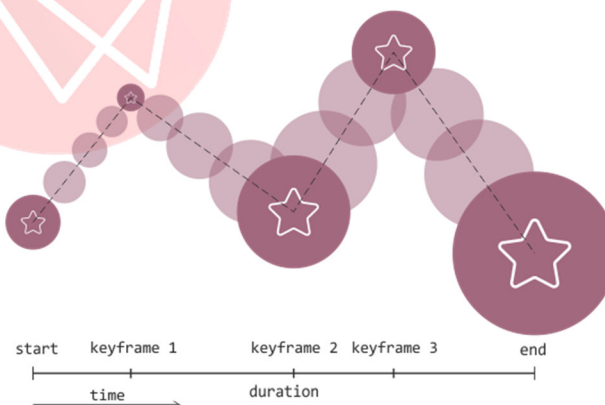
Način na koji je realizovan ovaj primer objašnjen je u video lekciji.

CSS animacije

U prethodnim redovima ste mogli da vidite da sistem CSS tranzicija omogućava definisanje samo dve ključne tačke animacije - početnu i završnu. Naime, početna ključna tačka jeste početna vrednost CSS svojstva koje se animira, dok je završna tačka vrednost koju svojstvo dobija prilikom iniciranja tranzicije. Broj ključnih tačaka kojima se određuje jedna animacija, osnovna je razlika između CSS tranzicija i CSS animacija. Naime, CSS animacije omogućavaju definisanje proizvoljnog broja kontrolnih tačaka animacije. Pored toga, još jedna razlika između CSS tranzicija i CSS animacija ogleda se u načinu na koji se inicira početak animacije. Sve osobine CSS animacija biće ilustrovane u nastavku ove lekcije.

Kako se definišu CSS animacije?

CSS animacije definišu se veoma slično već ilustrovanim CSS tranzicijama. Osnovna razlika jeste neophodnost definisanja ključnih kadrova. Drugim rečima, CSS animacije su određene skupom ključnih kadrova (*keyframes*, *engl.*). Ključni kadrovi definišu vizualne osobine elementa u određenim trenucima animacije (slika 2.2).



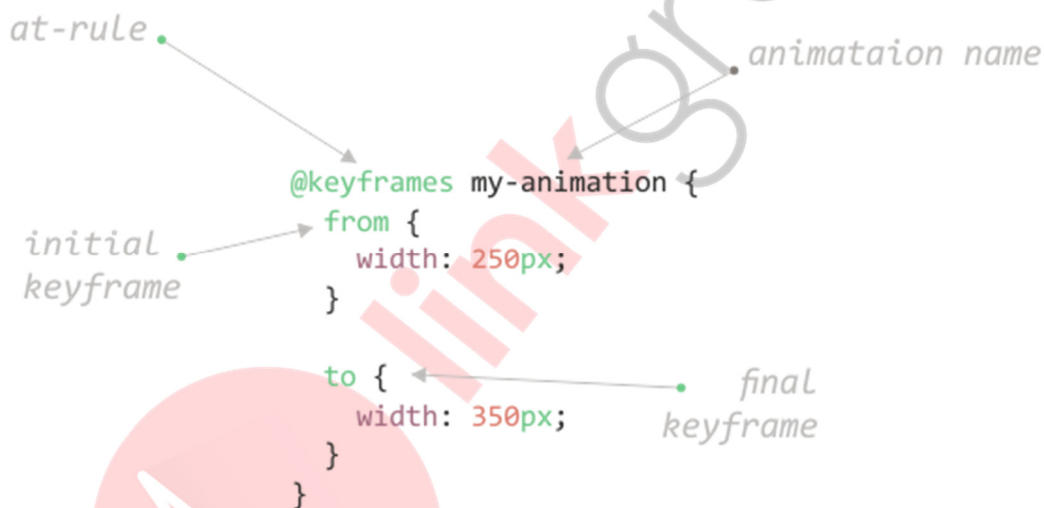
Slika 2.2 – CSS animacija

Na slici 2.2 se može videti tok jedne CSS animacije. Animacija ima svoj početak i kraj (*start* i *end*) i ukupno tri ključna kadra (*keyframe 1*, *keyframe 2*, *keyframe 3*).

Ključni kadrovi CSS animacije definišu se korišćenjem `@keyframes` at-rule izjave:

```
@keyframes my-animation {  
  from {  
    width: 250px;  
  }  
  
  to {  
    width: 350px;  
  }  
}
```

Upravo je ilustrovan blok CSS koda, kreiran korišćenjem `@keyframes` izjave, kojim se definišu ključni kadrovi jedne jednostavne animacije. Blok unutar sebe sadrži dva pod-bloka, kojim se definišu početni i završni kadar animacije (slika 2.3).



Slika 2.3 – Definisanje ključnih kadrova animacije

Na ovaj način, stvorena je jedna veoma jednostavna animacija koja je po osobinama identična transformacijama koje su prikazane u prvom delu ove lekcije. Jednostavno, na ovaj način širina elementa nad kojim će ovakva animacija biti primenjena, biće animirana sa 250px na 350px.

Definisanje ključnih kadrova samo po sebi ne proizvodi nikakav rezultat, već je animaciju neophodno dodeliti elementu koji će biti animiran:

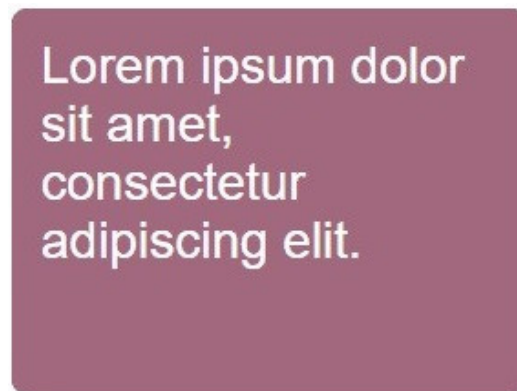
```
.box {  
  animation-name: my-animation;  
}
```


CSS animacija se elementu dodaje korišćenjem svojstva **animation-name**. Pored dodeljivanja animacije, neophodno je definisati i njeno trajanje:

```
.box {  
  animation-name: my-animation;  
  animation-duration: 3s;  
}
```

Dužina trajanja animacije definiše se korišćenjem svojstva **animation-duration**.

Nakon definisanja naziva i dužine trajanja, moći će se videti efekat kreirane animacije (animacija 2.5).



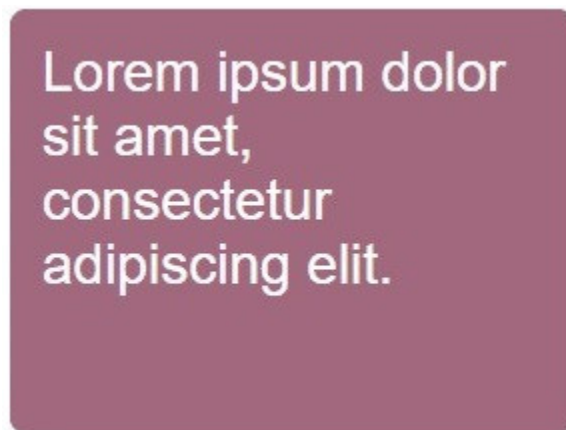
Animacija 2.5 – Efekat CSS animacije

Kroz upravo kreiranu animaciju nisu se mogle uvideti sve prednosti CSS animacija u odnosu na tranzicije. Osim mogućnosti pokretanja animacije odmah nakon učitavanja stranice, sve ostalo je identično kao i kod tranzicija.

Prava prednost CSS animacija odnosi se na mogućnost definisanja proizvoljnog broja ključnih kadrova. U prethodnom primeru njih je bilo dva, i oni su bili obeleženi ključnim rečima `from` i `to`. Ove ključne reči samo su drugačiji način za izražavanje vrednosti 0% i 100%. Stoga se lako može zaključiti da je drugi način za definisanje ključnih tačaka korišćenje procentualnih vrednosti:

```
@keyframes my-animation {  
  0% {  
    width: 250px;  
  }  
  33% {  
    width: 300px;  
  }  
  66% {  
    width: 325px;  
  }  
  100% {  
    width: 350px;  
  }  
}
```

Animacija je sada definisana korišćenjem četiri ključna kadra, koja se karakterišu procentualnim vrednostima. Stoga će se vrednost menjati u 3 koraka (s obzirom da se izuzima početni kadar). Kako sve to izgleda, ilustruje animacija 2.6.



Animacija 2.6 – Efekat CSS animacije (2)

CSS animacije omogućavaju i animiranje većeg broja svojstava:

```
@keyframes my-animation {  
  0% {  
    transform: translateX(0px) rotate(0deg);  
    width: 250px;  
    height: 180px;  
  }  
  
  33% {  
    transform: translateX(0px) rotate(360deg);  
    width: 250px;  
    height: 180px;  
  }  
  
  66% {  
    transform: translate(300px, 50px) rotate(360deg);  
    width: 100px;  
    height: 75px;  
  }  
  
  100% {  
    transform: translateX(0px) rotate(0deg);  
    width: 250px;  
    height: 180px;  
  }  
}
```

Ovoga puta je prikazana nešto kompleksnija animacija, koja podrazumeva animiranje nekoliko različitih svojstava. U početnom kadru su definisane inicijalne vrednosti. To su vrednosti `transform`, `width` i `height` svojstava. Animiranje se sprovodi u 3 koraka:

- u prvom koraku se obavlja rotacija za 360 stepeni
- u drugom koraku se element pomera za 300px u desno i smanjuje se njegova veličina promenom vrednosti `width` i `height` svojstava
- u trećem i poslednjem koraku, vrednosti svojstava koja su animirana vraćaju se na početne

Efekat ovoga primera je ilustrovan animacijom 2.7.



Animacija 2.7 – Efekat CSS animacije (3)

Konfigurisanje CSS animacija

Sve do sada, bavili smo se različitim pristupima za definisanje ključnih kadrova animacije, čime se definiše njen izgled. Ipak, ostale njene osobine, definišu se na samom elementu koji se animira, baš kao što je to bio slučaj sa svojstvima `animation-name` i `animation-duration`, koja su već iskorišćena u dosadašnjem toku lekcije. Kompletan spisak svojstava za konfigurisanje CSS animacija prikazan je tabelom 2.2.

Svojstvo	Opis
<code>animation-name</code>	definiše naziv animacije koja je definisana ključnim kadrovima korišćenjem <code>@keyframes</code> <code>at-rule</code> konstrukcije; obavezno svojstvo
<code>animation-duration</code>	definiše trajanje animacije, odnosno koliko će biti potrebno vremena da prođe od početka do kraja animacije; izražava se u sekundama (s); ukoliko se ovo svojstvo izostavi, animacija se neće izvršiti
<code>animation-delay</code>	definiše vreme koje je potrebno da protekne od učitavanja elementa, pa do početka animacije
<code>animation-iteration-count</code>	definiše broj ponavljanja animacije
<code>animation-direction</code>	definiše usmerenje animacije, odnosno da li će se animacija izvršavati u napred (od početka do kraja), unazad (od kraja do početka) ili će naizmenično menjati usmerenje

animation-timing-function	definiše krivu izvršavanja animacije, kojom se kontroliše njena brzina u različitim delovima
animation-fill-mode	definiše vizualne osobine elementa pre i nakon završetka animacije; s obzirom da animacija trajno ne menja osobine elementa, ovim svojstvom je moguće uticati na vizualne karakteristike elementa kada se animacija ne izvršava
animation-play-state	omogućava zaustavljanje i ponovno nastavljanje animacije

Tabela 2.2 – Svojstva za konfigurisanje CSS animacija

Zadržka animacije

Vreme koje je potrebno da protekne od učitavanja elementa pa do početka animacije, može se definisati svojstvom **animation-delay**:

```
animation-delay: 2s;
```

Vrednost animation-delay svojstva se izražava u sekundama.

Broj ponavljanja animacije

Inicijalno, CSS animacija se izvršava jednom. Na broj ponavljanja animacije se može uticati svojstvom **animation-iteration-count**:

```
animation-iteration-count: 2;
```

Na ovaj način, animacija će se izvršiti dva puta.

Ukoliko je animaciju potrebno ponavljati beskonačan broj puta, dovoljno je napisati:

```
animation-iteration-count: infinite;
```

Usmerenje animacije

Podrazumevano, animacija se izvršava u napred, odnosno od prvog pa do poslednje definisanog ključnog kadra. Na takvu osobinu je moguće uticati korišćenjem svojstva **animation-direction**. Vrednosti koje ovo svojstvo može imati, ilustrovane su tabelom 2.3.

Vrednost	Opis
normal	animacija se izvršava u napred; ovo je podrazumevana vrednost
reverse	animacija se izvršava u nazad, odnosno od poslednjeg do prvog ključnog kadra
alternate	animacija se izvršava naizmenično, prvo u napred, a zatim u nazad
alternate-reverse	animacija se izvršava naizmenično, prvo u nazad, a zatim u napred

Tabela 2.3 – Vrednosti svojstva animation-direction

Primer upotrebe svojstva animation-direction:

```
animation-direction: reverse;
```

Tajming funkcija animacije

Baš kao što je to bio slučaj i kod tranzicija i CSS animacije omogućavaju definisanje tajming funkcije, koja utiče na brzinu izvršavanja animacije u njenim različitim delovima. Tajming funkcija se definiše korišćenjem svojstva **animation-timing-function** i može imati vrednosti ilustrovane tabelom 2.4.

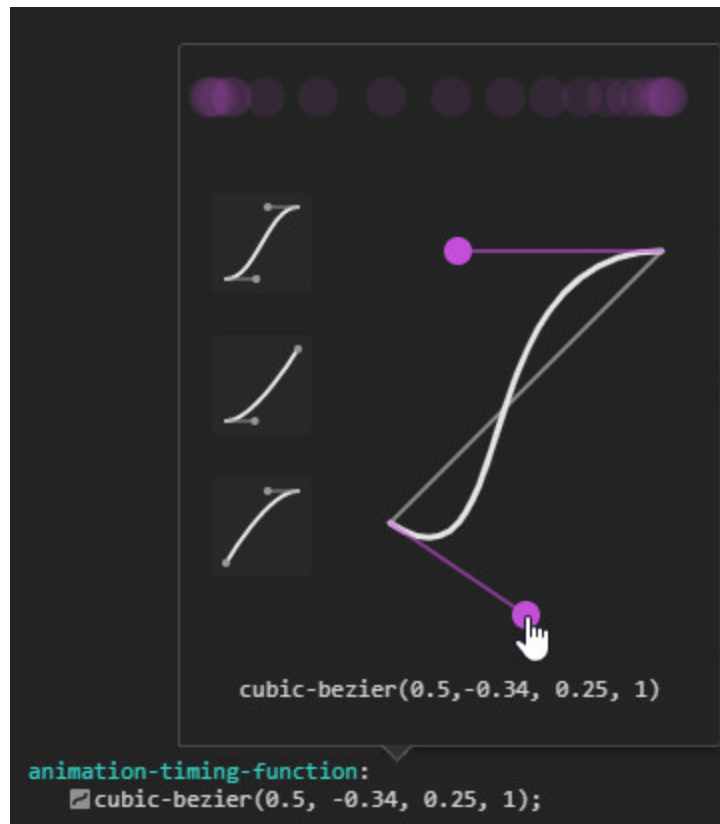
Vrednost	Opis
ease	spor početak, zatim ubrzanje, spor kraj; ovo je podrazumevana vrednost
linear	jednaka brzina tokom celog trajanja
ease-in	spor početak
ease-out	spor kraj
ease-in-out	spor početak i spor kraj
cubic-bezier(n,n,n,n)	proizvoljno definisanje dinamike izvršavanja navođenjem parametara za kreiranje bezijerove krive

Tabela 2.4 – Vrednosti svojstva animation-timing-function

Primer definisanja vrednosti svojstva animation-timing-function:

```
animation-timing-function: cubic-bezier(0.5, -0.34, 0.25, 1);
```

U prikazanoj liniji CSS koda, za definisanje vrednosti svojstva animation-timing-function iskorišćena je jedna od CSS funkcija - cubic-bezier(). Korišćenjem ove funkcije omogućeno je definisanje osobina bezijerove krive, na osnovu koje će biti određena brzina animacije u različitim trenucima. Na svu sreću, web developer-i ne moraju samostalno da sastavljaju ovakve funkcije, s obzirom da većina modernih web pregledača poseduje neku vrstu olakšice za njihovo definisanje (slika 2.4).



Slika 2.4 – Google Chrome panel za definisanje osobina tajming funkcije

Osobine elementa pre i posle animacije

Do sada ste mogli da vidite da CSS animacija ni na koji način ne utiče na vizualne osobine elementa pre, ali i nakon završetka animacije. Drugim rečima, nakon završetka animacije, element se vraća u svoj prvobitni vizuelni oblik, koji je definisan CSS deklaracijama koje su nad takvim elementom primenjene. Kako bi se uticalo na vizualne osobine elementa koji se animira, pre i nakon animacije, potrebno je koristiti svojstvo **animation-fill-mode**. Ovo svojstvo može imati vrednosti ilustrovane tabelom 2.5.

Vrednost	Opis
none	animacija neće uticati na osobine elementa koji se animira; ovo je podrazumevana vrednost
forwards	nad elementom će ostati primenjene osobine koje su definisane poslednjim ključnim kadrom animacije
backwards	nad elementom će ostati primenjene osobine koje su definisane prvim ključnim kadrom animacije
both	element će moći da dobije osobine prvog, ali i poslednjeg ključnog kadra, naravno sve u zavisnosti od usmerenja animacije

Tabela 2.5 – Vrednosti svojstva animation-fill-mode

Svojstvo `animation-fill-mode` neophodno je razmatrati u kombinaciji sa svojstvima kojima se utiče na usmerenje animacije i broj njenih ponavljanja (`animation-direction` i `animation-iteration-count`). Jednostavno, ova dva svojstva će odrediti kojim kadrom će animacija započeti i završiti se. Na primer, ukoliko se animacija izvršava u napred i to samo jednom, onda vrednost `backwards`, svojstvo `animation-fill-mode` neće imati nikakav efekat. Sa druge strane, u identičnoj situaciji, vrednosti `forwards` i `both` učiniće da stilizacija poslednjeg ključnog kadra ostane primenjena na elementu i nakon završetka animacije.

Primer definisanja vrednosti svojstva `animation-fill-mode` može izgledati ovako:

```
animation-fill-mode: forwards;
```

Pauziranje i nastavljavanje animacije

CSS animaciju je moguće pauzirati, čime se postiže njeno zaustavljanje u trenutku u kome se našla tokom izvršavanja. Za obavljanje takvog posla, koristi se svojstvo **`animation-play-state`**, koje može imati dve vrednosti (tabela 2.6).

Vrednost	Opis
paused	vrednost koja pauzira izvršavanje animacije
running	vrednost koja nastavlja izvršavanje animacije

Tabela 2.6 – Vrednosti svojstva `animation-play-state`

Upotrebna vrednost `animation-play-state` svojstva vrlo je ograničena ukoliko se posmatra isključivo iz ugla CSS jezika. Jednostavno, kako bi se animacija pauzirala u nekom trenutku njenog izvršavanja, neophodno je obaviti dinamičko postavljanje vrednosti svojstva `animation-play-state` na `paused`. CSS poseduje veoma ograničen skup scenarija koji bi tako nešto omogućili. Vrhunac može biti upotreba nekih pseudo-klasa:

```
.box {  
    animation-fill-mode: forwards;  
}  
  
.box:hover {  
    animation-play-state: paused;  
}
```

Ovim primerom postignut je efekat pauziranja animacije prilikom prelaska strelice miša preko elementa koji se animira.

Objedinjeno definisanje osobina animacija

Za kraj ove lekcije, biće prikazano svojstvo koje se prilikom konfigurisanja CSS animacija možda i najviše koristi. Reč je o svojstvu koje omogućava objedinjeno definisanje svih osobina animacije prikazanih tabelom 2.2. Reč je o svojstvu **`animation`**:

```
animation: animation-name animation-duration animation-timing-function  
animation-delay animation-iteration-count animation-direction animation-  
fill-mode animation-play-state;
```

Korišćenjem svojstva `animation`, sve osobine animacije koje su do sada definisane pojedinačnim svojstvima, sada se mogu objediniti na sledeći način:

```
animation: my-animation 3s ease 2s 2 reverse forwards running;
```

Pitanje

Ukoliko se postepena promena vizuelnih osobina nekog HTML elementa, obavlja automatski prilikom promene vrednosti nekog CSS svojstva, reč je o:

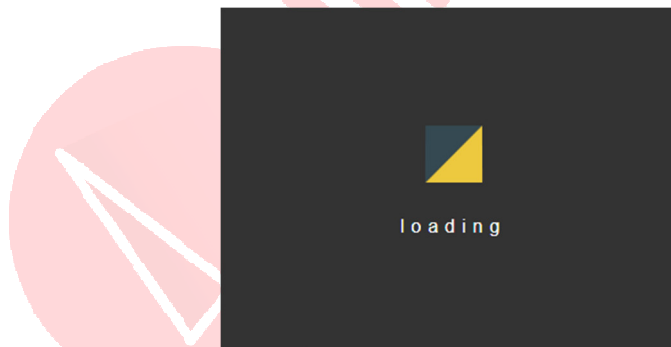
- a) tranziciji
- b) animaciji
- c) balansiranju
- d) centriranju

Objašnjenje:

CSS tranzicije su pojam koji označava funkcionalnost CSS jezika, koja omogućava da se vrednost nekog CSS svojstva postepeno menja tokom nekog vremenskog perioda. Tranzicije se automatski aktiviraju promenom vrednosti CSS svojstva.

Primer - Realizacija loader-a korišćenjem CSS animacija

Neki od postulata rada sa CSS animacijama, biće iskorišćene za kreiranje jednog loader-a. Reč je o animaciji koja može biti prikazana dok traje neka vremenski zahtevna operacija ili učitavanje stranice. Primerom će biti dobijen efekat kao na animaciji 2.8.



Animacija 2.8 - Loader kreiranim korišćenjem CSS animacija

Kompletan kod primera, možete da preuzmete sa sledećeg linka:

`example-css-animations-loader.rar`

Način na koji je realizovan ovaj primer objašnjen je u video lekciji.

Rezime

- CSS poseduje dva sistema za postizanje postepene promene osobina HTML elemenata tokom određenog vremenskog perioda – CSS tranzicije i CSS animacije
- tranzicija predstavlja glatku i postepenu promenu vrednosti nekog CSS svojstva
- tranzicije se definišu korišćenjem svojstva `transition`
- prilikom definisanja tranzicije, obavezno je definisati naziv svojstva čija vrednost će se postepeno menjati i dužinu trajanja promene
- brzina kojom se promena vrednosti nekog svojstva obavlja tokom vremena, naziva se tajming tranzicije, odnosno tajming funkcija, a u CSS-u se kontroliše korišćenjem svojstva `transition-timing-function`
- efekat tranzicije je moguće odložiti, korišćenjem svojstva `transition-delay`
- CSS animacije odnose se na zaseban sistem CSS jezika, koji obezbeđuje znatno veću slobodu prilikom definisanja animacija
- CSS animacije su određene skupom ključnih kadrova (*keyframes, engl.*)
- ključni kadrovi CSS animacija, definišu se korišćenjem `@keyframes` izjave
- CSS animacija se elementu dodaje korišćenjem svojstva `animation-name`
- dužina trajanja animacije definiše se korišćenjem svojstva `animation-duration`
- vreme koje je potrebno da protekne od učitavanja elementa pa do početka animacije, može se definisati svojstvom `animation-delay`
- broj ponavljanja animacije definiše se svojstvom `animation-iteration-count`
- usmerenje animacije moguće je definisati korišćenjem svojstva `animation-direction`
- tajming funkcija animacije definiše se korišćenjem svojstva `animation-timing-function`
- za uticanje na vizualne osobine elementa koji se animira, pre i nakon animacije, potrebno je koristiti svojstvo `animation-fill-mode`
- CSS animaciju je moguće pauzirati, korišćenjem svojstva `animation-play-state`
- svojstvo koje omogućava objedinjeno definisanje svih osobina animacije odjednom je `animation`

