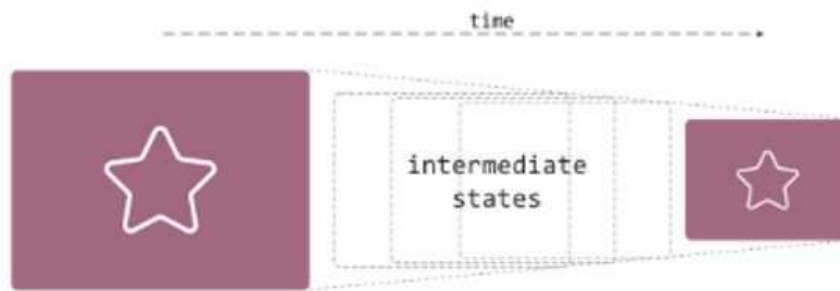


# CSS tranzicije i animacija

U prethodnim lekcijama ovog modula ilustrovano je postizanje 2D i 3D transformacija nad HTML elementima. Kao što ste imali prilike da vidite, reč je o veoma moćnim funkcionalnostima CSS jezika, koje developerima obezbeđuju neslućene mogućnosti prilikom izgradnje korisničkog interfejsa. Ipak, moć CSS jezika se ovde ne završava. Pored transformacija, CSS obezbeđuje i funkcionalnosti za lako definisanje tranzicija i animacija. Lekcija koja je pred vama biće posvećena ovim pojmovima.

## Šta su CSS tranzicije?

U prethodnim lekcijama je rečeno da se pojam transformacije odnosi na prelazak elementa iz jednog oblika u drugi. Ukoliko se takva promena obavi glatko i postepeno, tokom određenog vremenskog perioda, govori se o tranziciji. Ipak, tranzicije nisu ograničene na rad sa transformacijama, već omogućavaju glatku promenu vrednosti gotovo svih CSS svojstava (slika 7.1).



Slika 7.1. CSS tranzicija

Slika 7.1. ilustruje način na koji se obavlja jedna CSS tranzicija. Na levoj strani slike je prikazan početni izgled elementa, a na desnoj završni. Veličina elementa se postepeno, tokom određenog vremenskog intervala, menja, pri čemu sam browser brine o adekvatnom prikazu svih međustanja, koja su na slici 7.1. ilustrovana isprekidanim linijama između početnog i završnog prikaza.

## Kako se definišu CSS tranzicije?

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

- naziv svojstva čija vrednost će se postepeno menjati;
- dužinu trajanja promene.

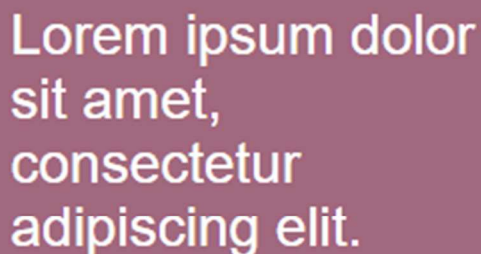
Sledeći primer ilustruje definisanje jedne tranzicije:

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

Ovako definisana tranzicija 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. Drugim rečima, da bi se efekat tranzicije video, neophodno je da dođe do promene vrednosti svojstva definisanog tranzicijom (u prikazanom primeru to je svojstvo `width`).

Promena vrednosti nekog CSS svojstva može biti inicirana na različite načine. Neki od tih načina su izvan oblasti koje pokriva ovaj kurs (prevashodno se misli na promenu vrednosti CSS svojstava korišćenjem JavaScript jezika). Ipak, i CSS jezik obezbeđuje određene pristupe koji mogu inicirati promenu vrednosti nekog svojstva. Jedan od takvih pristupa već je ilustrovan u prethodnim lekcijama ovog kursa, kada je bilo reči o stilizovanju specifičnih stanja u kojima se mogu naći HTML elementi. Tako se promena širine `div` elementa može obaviti prilikom prelaska strelice miša preko njega (animacija 7.1).



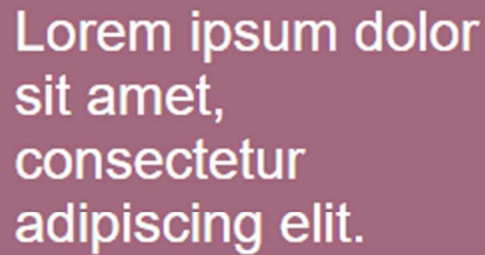
Lorem ipsum dolor  
sit amet,  
consectetur  
adipiscing elit.

*Animacija 7.1. Tranzicija širine `div` elementa koja se aktivira prelaskom strelice miša*

Moguće je u istom trenutku definisati efekat tranzicije za više od jednog CSS svojstva:

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

Ovoga puta, tranzicija je definisana za `width` i `height` svojstva (animacija 7.2).



Lorem ipsum dolor  
sit amet,  
consectetur  
adipiscing elit.

*Animacija 7.2. Tranzicija širine i visine div elementa koja se aktivira prelaskom strelice miša*

Unutar radnog okruženja nalazi se jednostavan primer animacije koja se aktivira prilikom prelaska mišem preko box elementa. Pokušajte da umesto width i height svojstva animirate visibility ili opacity svojstva kako bi postigli fade-in i fade-out efekte opisane u prilogu ispod.

#### **Radno okruženje**

##### **HTML fajl:**

```
<div class="box">Lore ipsum dolor sit amet, consectetur, adipiscing elit.</div>
```

##### **CSS fajl:**

```
.box {  
    background: coral;  
    padding: 10px;  
    color: white;  
    width: 100px;  
    height: 100px;  
    transition: width 1s, height 1.5s;  
}  
.box:hover {  
    width: 200px;  
    height: 150px;  
}
```

#### **CSS svojstva nad kojima se može upotrebiti tranzicija**

Nešto ranije u ovoj lekciji rečeno je da se tranzicije mogu primeniti nad gotovo svim HTML elementima. Zapravo, stvarnost je takva 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 jeste to da se tranzicija ne može definisati za svojstvo `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`.

Tranziciju je moguće aktivirati i za sva svojstva nekog HTML elementa. U takvim situacijama se navodi vrednost `all`.

## 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 7.1.

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

Tabela 7.1. Vrednosti `transition-timing-function` svojstva

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

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

## 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;  
}
```

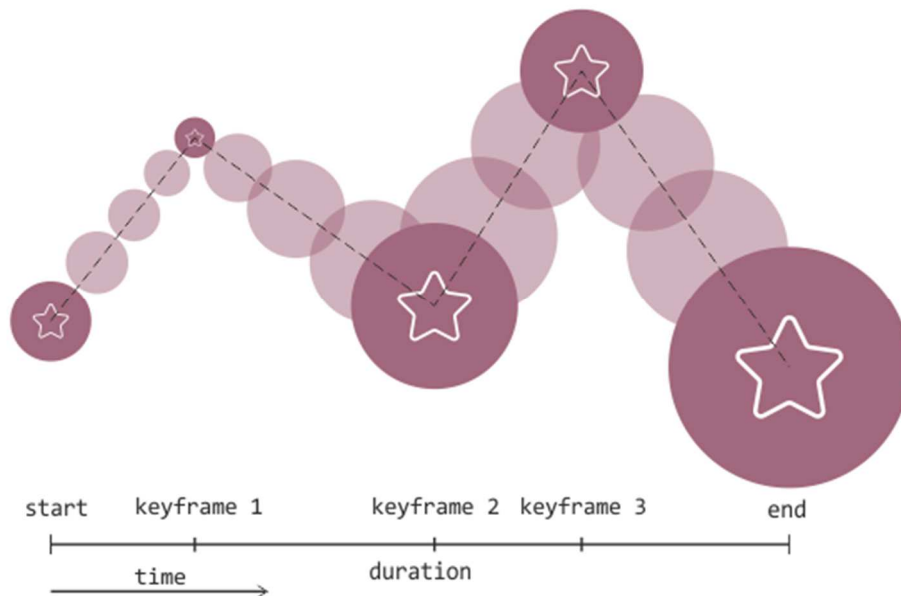
## Šta su CSS animacije?

Iako su upravo prikazane CSS tranzicije jedan od načina na koje se mogu stvoriti animacije na web sajtovima, pojam animacija u CSS-u ima posebno značenje. Tako se CSS animacije odnose na zaseban sistem CSS jezika, koji obezbeđuje znatno veću slobodu prilikom definisanja animacija.

Osnovna razlika između CSS tranzicija i CSS animacija ogleda se u načinu na koji se inicira njihov početak, kao i u načinu na koji se definišu osobine animacije. CSS animacije omogućavaju mnogo fleksibilniji sistem za kontrolu animacije.

## Kako se definišu CSS animacije?

CSS animacije definišu se veoma slično već ilustrovanim CSS tranzicijama. Osnovna razlika jeste u neophodnosti definisanja ključnih kadrova. Drugim rečima, CSS animacije su određene skupom ključnih kadrova (*keyframes*). Ključni kadrovi definišu vizualne osobine elementa u određenim trenucima animacije (slika 7.2).



Slika 7.2. CSS animacija

Na slici 7.2. 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*). Kreiranje ključnih kadrova CSS animacije postiže se upotrebom jednog specifičnog CSS jezičkog elementa. On se naziva **at-rule**.

### At-rule

At-rule je, pored CSS opisa, osnovni gradivni blok CSS jezika, odnosno osnovni tip CSS izjave. Tako at-rule CSS izjave dodatno konfiguriraju ponašanje CSS jezika.

At-rule započinje karakterom @, nakon koga se navode identifikator i jedno ili više pravila:  
`@IDENTIFIER (RULE);`

Ukoliko postoji veći broj pravila, ona se smeštaju unutar zagrada.

Postoji veliki broj *at-rule* izjava, koje proizvode različite efekte. Na primer, kako bi se definisao tip karaktera koji se koristi unutar CSS fajla, može se iskoristiti at-rule `@charset`:

```
@charset "utf-8";
```

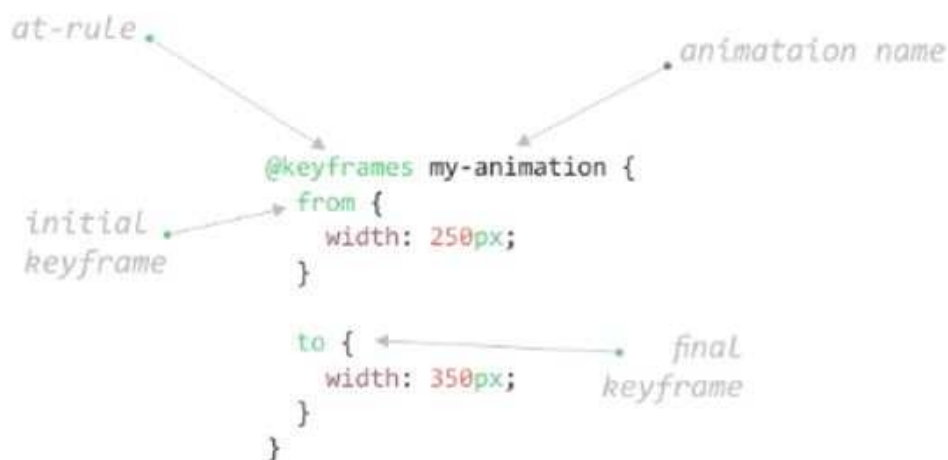
At-rule se između ostalog koristi i za definisanje eksternih fontova – `@font-face`.

U kontekstu ove lekcije, posebno zanimljiva at-rule izjava jeste i ona koja omogućava definisanje ključnih kadrova animacije – `@keyframes`.

Kao što ste upravo mogli da pročitate, 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, kojima se definišu početni i završni kadar animacije (slika 7.3).



Slika 7.3. Definisanje ključnih kadrova animacije

Na prikazani 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:

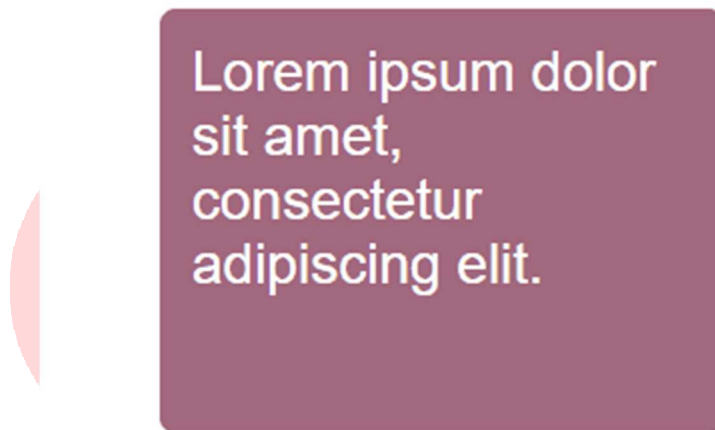
```
.box {  
    animation-name: my-animation;  
}
```

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 7.3).



*Animacija 7.3. Efekat CSS animacije*

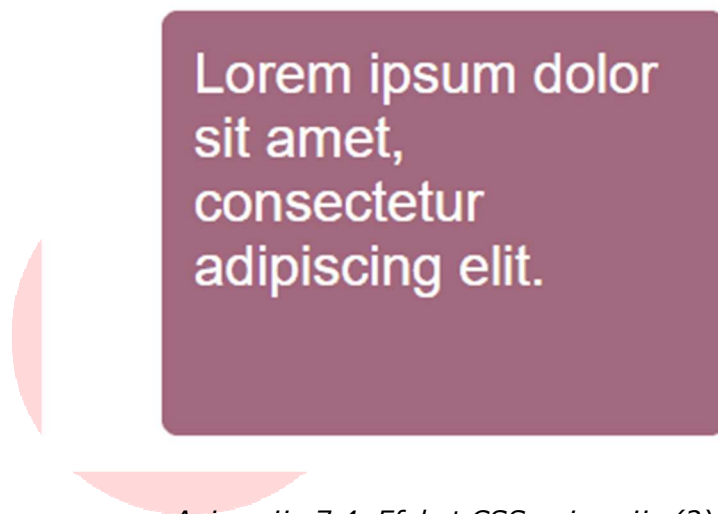
Kroz upravo kreiranu animaciju nisu se mogle uvideti sve prednosti CSS animacija u odnosu na tranzicije. Osim mogućnosti za pokretanje animacije odmah nakon učitavanja stranice, sve ostalo je identično kao i kod tranzicija.

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 zapravo menjati u tri koraka (pošto se početni kadar izuzima). Kako sve to izgleda – ilustruje animacija 7.4.



*Animacija 7.4. 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, kakve element poseduje u svom izvornom obliku. Animiranje se sprovodi u tri koraka:

- u prvom koraku se obavlja rotacija za 360 stepeni;
- u drugom koraku se element pomera za 300px udesno 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 7.5.





### *Animacija 7.5. Efekat CSS animacije (3)*

Unutar radnog okruženja prikazan je primer sa kompleksnijom animacijom. Pokušajte da u animaciju uključite i rotaciju zvezde oko Y ose.

#### **Radno okruženje**

##### **HTML fajl:**

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

##### **CSS fajl:**

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

@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;
    }
}

```

## 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 7.2.

Svojstvo	Opis
<code>animation-name</code>	definiše naziv animacije koja je definisana ključnim kadrovima korišćenjem <code>@keyframes</code> at-rule konstrukcije; obavezno svojstvo
<code>animation-duration</code>	definiše trajanje animacije, odnosno koliko će vremena proći 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 unapred (od početka do kraja), unazad (od kraja do početka) ili će naizmenično menjati usmerenje
<code>animation-timing-function</code>	definiše krivu izvršavanja animacije, kojom se kontroliše njena brzina u različitim delovima
<code>animation-fill-mode</code>	definiše vizualne osobine elementa pre završetka i nakon završetka animacije; s obzirom na to da animacija ne menja trajno osobine elementa, ovim svojstvom je moguće uticati na vizualne karakteristike elementa kada se animacija ne izvršava
<code>animation-play-state</code>	omogućava zaustavljanje i ponovno nastavljanje animacije

Tabela 7.2. Svojstva za konfigurisanje CSS animacija

### Zadrška 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 unapred, odnosno od prvog pa do poslednjeg definisanog ključnog kadra. Na takvu osobinu je moguće uticati korišćenjem svojstva `animation-direction`. Vrednosti koje ovo svojstvo može imati ilustrovane su tabelom 7.3.

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

Tabela 7.3. 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, 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 7.4.

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

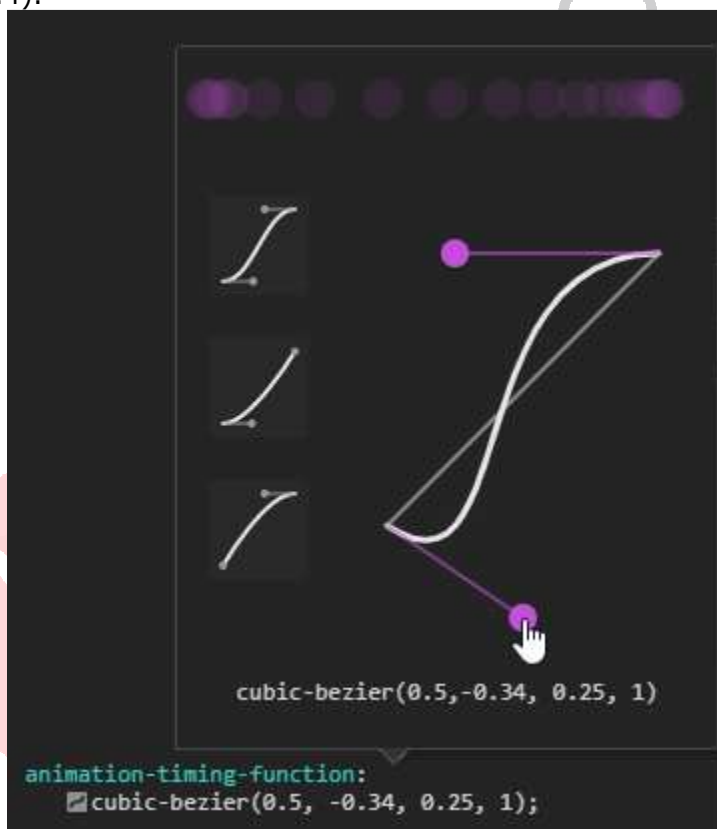
ease-in-out	spor početak i spor kraj
cubic-bezier(n,n,n,n)	proizvoljno definisanje dinamike izvršavanja navođenjem parametara za kreiranje Bezjeove krive

*Tabela 7.4. 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 Bezjeove krive na osnovu koje će biti određena brzina animacije u različitim trenucima. Na svu sreću, web developeri ne moraju samostalno da sastavljaju ovakve funkcije, pošto većina modernih web browsera poseduje neku vrstu olakšice za njihovo definisanje (slika 7.4).



*Slika 7.4. 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 ni nakon završetka animacije. Drugim rečima, nakon završetka animacije, element se vraća u svoj prvobitni vizualni 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 7.5.

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 7.5. 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 unapred i to samo jednom, onda vrednost `backwards` svojstva `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 7.6).

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

Tabela 7.6. Vrednosti svojstva `animation-play-state`

Upotrebnost vrednosti `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 pseudoklasa predstavljanih u prethodnim lekcijama ovog kursa:

```
.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 7.2. 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 objединити na sledeći način:

```

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

```

### Pitanje

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

- **tranziciji**
- animaciji
- balansiranju
- 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 1 – Carousel animacija

Prvi primer u kome će biti ilustrovane neke od tehnika prikazanih u ovoj lekciji predstavlja nadogradnju primera iz prethodne lekcije. Reč je o carousel primeru koji je iskorišćen za demonstraciju mogućnosti 3D transformacija. Sada će već kreirane 3D transformacije biti i animirane:

```

@keyframes carousel-animation {
    0% {

```

```

        transform: translateZ(-300px) rotate3d(0, 1, 0, 0deg);
    }

    45% {
        transform: translateZ(-300px) rotate3d(0, 1, 0, 160deg);
    }

    55% {
        transform: translateZ(-300px) rotate3d(0, 1, 0, 160deg);
    }

    100% {
        transform: translateZ(-300px) rotate3d(0, 1, 0, 360deg);
    }
}

```

Kod ilustruje animaciju koja je definisana korišćenjem četiri ključna kadra. U svakom od kadrova animira se rotacija po  $y$ -osi. Ovako kreirana animacija na elementu se konfiguriše na sledeći način:

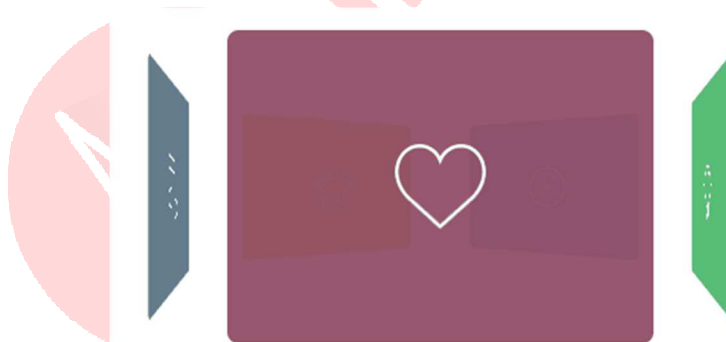
```

#tile-container {
    ...

    animation-name: carousel-animation;
    animation-duration: 4s;
    animation-iteration-count: infinite;
    animation-delay: 1s;
}

```

Efekat koji se dobija ilustruje animacija 7.6.



*Animacija 7.6. Carousel animacija*

Unutar radnog okruženja moguće je pokrenuti gore navedeni primer i testirati kod bez ikonica. Pokušajte da izmenite CSS kod za animaciju kako bi se slajder pomerao za po jednu pločicu.



## Radno okruženje

### HTML fajl:

```
<div id="main-container">
  <div id="tile-container">
    <div class="tile" id="tile-1"></div>
    <div class="tile" id="tile-2"></div>
    <div class="tile" id="tile-3"></div>
    <div class="tile" id="tile-4"></div>
    <div class="tile" id="tile-5"></div>
  </div>
</div>
```

### CSS fajl:

```
#main-container {
  perspective: 400px;
  position: relative;
  width: 300px;
  height: 220px;
  margin: 80px auto;
}

@keyframes carousel-animation {
  0% {
    transform: translateZ(-300px) rotate3d(0, 1, 0, 0deg);
  }
  45% {
    transform: translateZ(-300px) rotate3d(0, 1, 0, 160deg);
  }
  55% {
    transform: translateZ(-300px) rotate3d(0, 1, 0, 160deg);
  }
  100% {
    transform: translateZ(-300px) rotate3d(0, 1, 0, 360deg);
  }
}

#tile-container {
  transform: translateZ(-300px) rotate3d(0, 1, 0, 0deg);
  transform-style: preserve-3d;
  animation-name: carousel-animation;
  animation-duration: 4s;
  animation-iteration-count: infinite;
  animation-delay: 1s;
}

.tile {
  width: 300px;
  height: 220px;
  border-radius: 8px;
  position: absolute;
  margin: 0 auto;
```

```

    background-repeat: no-repeat;
    background-position: center;
    background-size: 64px;
    opacity: 0.95;
}

#tile-1 {
    background-color: #914E67;
    transform: rotateY(0deg) translateZ(300px);
}

#tile-2 {
    background-color: #4FBA6F;
    transform: rotateY(72deg) translateZ(300px);
}

#tile-3 {
    background-color: #5192D2;
    transform: rotateY(144deg) translateZ(300px);
}

#tile-4 {
    background-color: #E8BF1C;
    transform: rotateY(216deg) translateZ(300px);
}

#tile-5 {
    background-color: #5D7586;
    transform: rotateY(288deg) translateZ(300px);
}

```

## Primer 2 – Animacija scroll down strelice

Drugi primer u ovoj lekciji ilustriraće animiranje scroll down strelice koja je kreirana u jednoj od prethodnih lekcija. Za animiranje, biće korišćena sledeća animacija:

```

@keyframes bounce {
    0%,
    20%,
    50%,
    80%,
    100% {
        transform: translateY(0);
    }

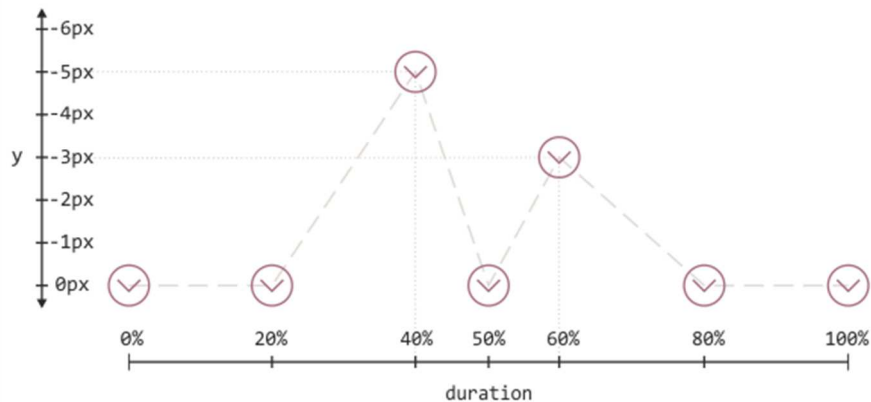
    40% {
        transform: translateY(-5px);
    }

    60% {
        transform: translateY(-3px);
    }
}

```

Definisana animacija poseduje sedam ključnih kadrova. Pet ključnih kadrova poseduje identične osobine, te su zbog toga oni objedinjeni unutar jednog bloka CSS koda. Preostala dva ključna kadra poseduju svoje specifične blokove.

Animacija funkcioniše pomeranjem elementa po y-osi. Pomeranje se vrši korišćenjem 2D transformacija, odnosno funkcijom `translateY()`. Kompletna logika prikazane animacije ilustrovana je dijagramom na slici 7.5.



Slika 7.5. Dijagram bounce animacije

Iz dijagrama se može zaključiti sledeće:

- prvih 20% trajanja animacije, element miruje;
- od 20 do 40%, element se podiže za 5px animiranjem transformacije `translateY()`;
- od 40 do 50%, element se vraća u svoj prvobitni položaj;
- od 50 do 60%, element se podiže za 3px;
- od 60 do 80%, element se vraća u prvobitni položaj;
- poslednjih 20%, element miruje.

Ovako kreirana animacija elementu se dodeljuje na sledeći način:

```
animation: bounce 2s infinite running;
```

Sve ovo će stvoriti efekat prikazan animacijom 7.7.



Animacija 7.7. Animacija scroll down strelice

Na kraju, može se otići i korak dalje i dodatno poboljšati korisnički ugođaj pauziranjem animacije kada se strelica miša nalazi iznad elementa:

```
#scroll-down-arrow:hover {  
    animation-play-state: paused;  
}
```

## 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*).
- Ključni kadrovi CSS animacija definišu se korišćenjem `@keyframes` at-rule 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 funkcije utiče na brzinu izvršavanja animacije u njenim različitim delovima, a 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, čime se postiže njeno zaustavljanje u trenutku u kome se našla tokom izvršavanja, korišćenjem svojstva `animation-play-state`.
- Svojstvo koje omogućava objedinjeno definisanje svih osobina animacije odjednom je `animation`.