CSS funkcije i promenljive

Osnovna namena CSS jezika jeste stilizovanje HTML dokumenata. Zbog toga se CSS drugačije naziva jezikom za stilizovanje. Takođe, CSS je jedan od primera kompjuterskih jezika koji se ne mogu nazvati programskim. Jednostavno, CSS-u nedostaju različiti, veoma značajni jezički elementi koji programskim jezicima omogućavaju formulisanje programske logike. Programska logika obezbeđuje da se postupak za rešavanje nekog problema iz realnog života pretoči u funkcionalan programski kod. Takvi poslovi se u programskim jezicima obavljaju korišćenjem različitih pojmova – tipovi, strukture, nizovi, operatori, petlje, grananja itd. Ovih pojmova u CSS jeziku nema, pa se upravo zbog toga i kaže da CSS nije programski jezik.

Ipak, sve ovo ne znači da CSS ne poseduje makar neke elemente koji oponašaju osobine programskih jezika. Tako dolazimo i do pojmova kojima će biti posvećena ova lekcija. Reč je o proizvoljnim svojstvima i funkcijama CSS jezika.

Šta su proizvoljna CSS svojstva (varijable)?

CSS varijable, CSS promenljive ili CSS proizvoljna svojstva (*custom properties*) nazivi su koji se koriste da označe specijalne entitete CSS jezika koji autoru dozvoljavaju definisanje vrednosti koje se mogu koristiti na proizvoljnom broju mesta unutar CSS stilizacije. To praktično znači da se jednom definisana vrednost može iskoristiti na različitim mestima unutar dokumenta.

CSS proizvoljna svojstva se definišu na poseban način, navođenjem dva karaktera crtica (--) ispred naziva svojstva:

```
--primary-color: blue;
```

Upravo prikazana linija CSS koda ilustruje kreiranje jednog proizvoljnog CSS svojstva sa nazivom `--primary-color` i vrednošću `blue`.

Zbog čega su nam potrebna proizvoljna CSS svojstva?

Korišćenjem proizvoljnih CSS svojstava rešavaju se dva veoma značajna problema prilikom pisanja CSS koda:

- smanjivanje ponavljanja koda;
- poboljšanje razumljivosti i čitljivosti.

CSS varijable omogućavaju smanjenje ponavljanja koda koji se piše. Naime, u dosta situacija se može dogoditi da se određena vrednost ponavlja na više mesta unutar dokumenta. Umesto definisanja takve vrednosti na svakom pojedinačnom mestu, CSS varijable omogućavaju da bude dovoljno definisati je samo jednom.

Takođe, vrednosti nekih CSS svojstava mogu biti prilično nerazumljive ljudima. Ovde se pre svega misli na definisanje boja u heksadecimalnoj notaciji:

```
background-color: #914E67;
```

Ovo je primer definisanja pozadine upotrebom boje u heksadecimalnom obliku. Uvidom u ovakvu vrednost veoma je teško zaključiti o kojoj boji je reč. Ipak, kada bi se vrednost ovakvog svojstva definisala korišćenjem proizvoljnog CSS svojstva, kod bi bio znatno čitljiviji:

```
background-color: var(--primary-color);
```

Osnovni primer korišćenja proizvoljnih CSS svojstava

Najjednostavniji slučaj korišćenja proizvoljnih CSS svojstava biće ilustrovan korišćenjem sledeće HTML strukture:

```
<div id="main-container">
      <div id="box1">Element 1</div>
      <div id="box2">Element 2</div>
      <div id="box3">Element 3</div>
      <div id="box4">Element 4</div>
      <div id="box5">Element 5</div>
</div>
```

Ovakvi HTML elementi inicijalno mogu biti stilizovani na sledeći način:

```
body {
      font-family: sans-serif;
      font-size: 28px;
}


#box1 {
      background-color: #914E67;
      color: white;
}

#box2 {
      color: #914E67;
}

#box3 {
      background-color: #914E67;
      color: white;
}

#box5 {
      border: 4px solid #914E67;
}
```

Bitno je primetiti da se jedna ista boja (#914E67) koristi kao vrednost nekoliko svojstava na različitim mestima – dva puta kao vrednost background-color i po jednom kao vrednost svojstava color i border. Ukoliko dođe do potrebe za izmenom ovakve boje, bilo bi neophodno pronaći sva mesta na kojima je naša boja upotrebljena i obaviti izmenu. Stoga je mnogo lakše boju definisati na samo jednom mestu, a zatim je referencirati na svim mestima na kojima se javi potreba za njenim korišćenjem. Tako nešto se korišćenjem proizvoljnih CSS svojstava rešava na veoma lak način. Boja koja će se više puta koristiti unutar dokumenta definiše se kao vrednost proizvoljnog CSS svojstva na sledeći način:

```
:root {
      --primary-dark-color: #914E67;
}
```

Primer ilustruje kreiranje jednog proizvoljnog CSS svojstva sa nazivom --primary-dark-color. Proizvoljna CSS svojstva je neophodno smestiti unutar tela nekog CSS opisa. Za formiranje takvog CSS opisa u primeru je iskorišćen jedan poseban selektor (pseudoklasa).

Pseudoklasa :root

Selektor :root predstavlja pseudoklasu koja se može koristiti za selektovanje korenog elementa unutar dokumenta. U slučaju HTML dokumenata, takav element uvek je element html. Ipak, CSS nije ograničen na stilizovanje isključivo HTML dokumenata, već je njega moguće koristiti i za stilizovanje SVG ili XML dokumenata. Tako će se u takvim tipovima dokumenata selektor :root odnositi na neke druge elemente.

Unutar HTML dokumenata, u najvećem broju slučajeva potpuno je svejedno da li se koristi :root ili html selektor, s obzirom na to da se u oba slučaja selektuje html element. Ipak, potrebno je znati da selektor :root poseduje viši prioritet, pa će u slučaju postojanja oba selektora biti primenjena stilizacija definisana opisom sa :root selektorom.

S obzirom na to da je u prikazanom primeru proizvoljno CSS svojstvo definisano unutar tela opisa sa :root selektorom, ono će biti primenljivo nad svim elementima na stranici. Kako bi se njegova vrednost iskoristila, dovoljno je upotrebiti CSS funkciju **var()**:

Radno okruženje

HTML fajl:

```
<div id="main-container">
      <div id="box1">Element 1</div>
      <div id="box2">Element 2</div>
      <div id="box3">Element 3</div>
      <div id="box4">Element 4</div>
      <div id="box5">Element 5</div>
</div>
```

CSS fajl:

```
:root {
      --primary-dark-color: #914E67;
}
```

```
body {
      font-family: sans-serif;
      font-size: 28px;
}

#box1 {
      background-color: var(--primary-dark-color);
      color: white;
}

#box2 {
      color: var(--primary-dark-color);
}
#box3 {
      background-color: var(--primary-dark-color);
      color: white;
}

#box5 {
      border: 4px solid var(--primary-dark-color);
}
```

Sada je za definisanje vrednosti različitih svojstava koja zahtevaju boju #914E67 upotrebljena konstrukcija koja podrazumeva upotrebu CSS funkcije **var()**. Kao parametar ove funkcije navodi se naziv proizvoljnog CSS svojstva. Na taj način svojstvo dobija vrednost CSS varijable.

Pitanje

Proizvoljna CSS svojstva kreiraju se korišćenjem karaktera:

- -
- **(--)**
- !!
- $

Objašnjenje:

CSS proizvoljna svojstva se definišu na poseban način, navođenjem dva karaktera crtica (--) ispred naziva svojstva.

Nasleđivanje CSS svojstava

Iz uvodnog primera korišćenja proizvoljnih CSS svojstava (varijabli), mogla se uvideti jedna njihova veoma značajna osobina – proizvoljna CSS svojstva se nasleđuju niz strukturu HTML elemenata. Upravo zbog toga je proizvoljno CSS svojstvo definisano na korenom elementu dokumenta moglo biti iskorišćeno na bilo kom elementu potomku. Iz ovoga proizilazi da obrnut slučaj nije moguć – nije moguće koristiti vrednost nekog proizvoljnog CSS svojstva koje je u HTML strukturi definisano ispod elementa na kojem se primenjuje.

Osobine nasleđivanja proizvoljnih CSS svojstava biće ilustrovane na sledećem primeru:

Radno okruženje

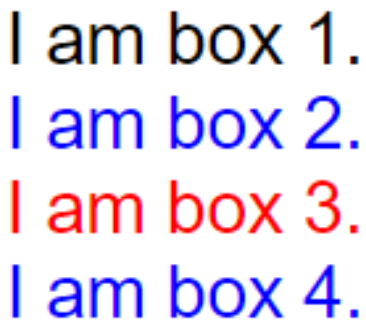
HTML fajl:

```
<div id="box-1">
      I am box 1.
      <div id="box-2">
         I am box 2.
         <div id="box-3">I am box 3.</div>
         <div id="box-4">I am box 4.</div>
      </div>
</div>
```

CSS fajl:

```
body{
      font-family: sans-serif;
      font-size: 22px;
}

#box-2 {
      --text-color: blue;
}

#box-3 {
      --text-color: red;
}

#box-1 {
      color: var(--text-color);
}

#box-2 {
      color: var(--text-color);
}

#box-3 {
      color: var(--text-color);
}

#box-4 {
      color: var(--text-color);
}
```

Unutar radnog okruženja HTML struktura primera se sastoji iz četiri `div` elementa. Boja teksta unutar ovih elemenata postavljena je korišćenjem proizvoljnog CSS svojstva `--text-color`. Vrednost ovoga svojstva definisana je na dva različita mesta, i to na elementima `box-2` i `box-3`. Na ovaj način biće dobijen efekat kao na slici 23.1.



Slika 23.1. Efekat nasleđivanja proizvoljnih CSS svojstava

Unutar radnog okruženja i na slici 23.1. mogu se videti osobine nasleđivanja proizvoljnih CSS svojstava. Element `box-1` ne dobija boju teksta definisanu `--text-color` svojstvom. Ovo je potpuno očekivano, s obzirom na to da je svojstvo `--text-color` definisano na elementima koji se u HTML strukturi nalaze ispod `box-1` elementa.

Element `box-2` dobija plavu boju teksta, s obzirom na to da je na njemu definisano `--text-color` svojstvo sa vrednošću `blue`.

Element `box-3` ima crvenu boju teksta, zato što je na njemu obavljeno redefinisanje vrednosti svojstva `--text-color` na `red`.

Na kraju, element `box-4` vrednost svojstva `--text-color` nasleđuje od svog direktnog roditeljskog elementa `box-2`, pa je zbog toga boja njegovog teksta plava.

Pokušajte da unutar radnog okruženja izmenite CSS kod kako bi testirali nasleđivanje nekog drugog svojstva, na primer: font-size.

Podrazumevane vrednosti CSS svojstava

U nekim situacijama, proizvoljno CSS svojstvo može biti nedostupno, baš kao što je to bio slučaj u prethodnom primeru na elementu `box-1`. Takođe, vrednost proizvoljnog svojstva može biti nevalidna, odnosno neprimenljiva na određenom svojstvu. Tako nešto bi se dogodilo kada bi svojstvo `--text-color` iz prethodnog primera, na primer, imalo vrednost `10px`.

Kako bi se prevazišli problemi koji mogu nastati iz dve upravo opisane situacije, unutar funkcije `var()` moguće je definisati podrazumevanu vrednost koja će biti korišćena kada CSS svojstvo nije dostupno ili je njegova vrednosti nevalidna:

```
#box-1 {
      color: var(--text-color, gray);
}
```

Unutar prikazanog primera, CSS funkcija `var()` sada prihvata dva parametra. Prvi parametar se odnosi na proizvoljno CSS svojstvo, dok se drugim parametrom definiše podrazumevana vrednost koja će biti iskorišćena ukoliko je svojstvo `--text-color` nedostupno ili nevalidno.

Šta su CSS funkcije?

CSS funkcije su specijalni elementi CSS jezika koji se mogu koristiti za definisanje vrednosti različitih CSS svojstava. U prethodnim redovima ovog kursa već je iskorišćena jedna takva funkcija. Reč je o CSS funkciji `var()`. Pored ove funkcije, u dosadašnjem toku ovoga kursa obrađene su i razne druge funkcije, kao što su `blur()`, `hsl()`, `linear-gradient()`, `rgb()`, `rotate()`, `url()`... Sve ove funkcije poseduju svoje specifične namene, koje su usko vezane za određene oblasti CSS jezika. Upravo zbog toga su one obrađene u onim delovima kursa u kojima je bilo reči o srodnim pojmovima. Ipak, dvema funkcijama do sada nije posvećena posebna pažnja, pa će one biti obrađene u nastavku ove lekcije. Reč je o funkcijama `calc()` i `attr()`.

calc()

CSS funkcija `calc()` omogućava obavljanje jednostavnih proračuna korišćenjem osnovnih računskih operacija prilikom definisanja vrednosti nekog CSS svojstva. Tako funkcija `calc()` zapravo omogućava da se za vrednost nekog svojstva postavi matematički izraz, a ne konkretna vrednost. S obzirom na to da rukuje numeričkim vrednostima, funkciju `calc()` je moguće koristiti za definisanje vrednosti svojstava kojima se izražava visina, širina, učestalost (broj ponavljanja), ugao, vreme i slično.

Jednostavan primer korišćenja funkcije `calc()` može da izgleda ovako:

```
width: calc(100% - 60px);
```

U primeru je funkcija `calc()` iskorišćena za definisanje vrednosti širine (*width*). Širina elementa sa ovakvom CSS deklaracijom će nakon proračuna biti za 60px manja od širine roditeljskog elementa.

U prikazanom primeru unutar funkcije `calc()` iskorišćena je računska operacija oduzimanja. Pored ove operacije, prilikom formulisanja izraza unutar `calc()` funkcije moguće je koristiti i druge operatore. Kompletan spisak operatora je sledeći:

- sabiranje (+)
- oduzimanje (-)
- množenje (*)
- deljenje (/)

Napomena

Veoma je bitno obratiti pažnju na to da je operatore sabiranja (+) i oduzimanja (-) neophodno odvojiti razmacima od ostatka izraza. Tako nešto nije obavezno kada je reč o operatorima množenja (*) i deljenja (/), ali se svakako preporučuje zbog preglednosti.

Iz uvodnog primera korišćenja funkcije `calc()` može se videti još jedna njena važna osobina: ona dozvoljava korišćenje računskih operacija nad vrednostima koje su izražene različitim jedinicama. Tako je u primeru vrednost u pikselima oduzeta od vrednosti u procentima.

Postoje brojne situacije u kojima funkcija `calc()` može biti i više nego korisna. U nastavku će biti predstavljene dve takve situacije.

Primer 1 – Korišćenje funkcije calc() za pozicioniranje pozadinske slike

Poziciju pozadinskih slika moguće je definisati korišćenjem CSS svojstva `background-position`. Ovo svojstvo prihvata dve vrednosti, koje definišu poziciju slike po `x` i po `y` osi. U zavisnosti od tipa vrednosti, razlikuju se dva ponašanja:

- kada se vrednosti definišu korišćenjem procenata, one su relativne kompletnom elementu; na primer, vrednost od `0%` po y osi označava poravnanje gornje ivice slike sa gornjom ivicom elementa; vrednost od `100%` označava poravnanje donje ivice slike sa donjom ivicom elementa; tako vrednost od `50%` proizvodi poravnanje po `x` ili y osama elementa;
- kada se vrednosti definišu korišćenjem apsolutnih jedinica, kao što su `px`, `cm`, `pt` i slično – tada vrednosti označavaju udaljenost gornje ivice slike od gornje ivice elementa i udaljenost leve ivice slike od leve ivice elementa.

Uzimajući u obzir ovo što je rečeno, ukoliko je pozadinsku sliku potrebno pozicionirati 20px od gornje i leve ivice, tako nešto bi moglo da se obavi na veoma jednostavan način:

```
background-position: 20px 20px;
```

Na ovaj način, pozadinska slika će biti pozicionirana 20px od leve i gornje ivice. Sada se postavlja pitanje: *šta ukoliko je potrebno uraditi ovo isto, ali u odnosu na donju i desnu ivicu?* U takvoj situaciji, neophodno je iskoristiti CSS funkciju `calc()`:

```
background-position: calc(100% - 20px) calc(100% - 20px);
```

Unutar upravo prikazane linije koda, pozicioniranje pozadinske slike obavljano je korišćenjem `calc()` funkcije. Kako bi se slika pozicionirala tako da od donje i desne ivice elementa bude udaljena po 20px, definisan je sledeći izraz:

```
calc(100% - 20px)
```

Vrednosti od `100%` bi inače poravnale ivice pozadinske slike sa ivicama elementa. Oduzimanjem `20px` od vrednosti od `100%`, postiže se distanciranje slike od donje i desne ivice.

Radno okruženje

HTML fajl:

```
<div class="box1"></div>
```

CSS fajl:

```
.box1 {
    width: 200px;
    height: 200px;
    border: 2px solid black;
    background-image:   url('https://images.pexels.com/photos/461077/pexels-
photo-461077.jpeg?auto=compress&cs=tinysrgb&dpr=2&h=650&w=940');
    background-size: cover;
    background-repeat: no-repeat;
    background-position: calc(100% - 20px) calc(100% - 20px);

}
```

Pokušajte da unutar radnog okruženja izmenite CSS kod, odnosno da upotrebite funkciju calc kod height i width svojstava.

Primer 2 – Korišćenje funkcije calc() za definisanje fleksibilne veličine teksta

U jednoj od prethodnih lekcija u kojima je bilo reči o kreiranju responsive teksta, spomenut je pristup koji podrazumeva kombinovanje viewport jedinica i `calc()` CSS funkcije. Sada je došao trenutak da se upoznamo i sa takvim pristupom za kreiranje responsive teksta:

```
font-size: calc(13px + (21 - 13) * ((100vw - 320px) / (1600 - 320)));
```

Upravo je prikazan primer definisanja vrednosti svojstva `font-size`, i to korišćenjem funkcije `calc()`. Matematika unutar funkcije `calc()` nešto je komplikovanija nego do sada i zahteva podrobniju analizu (slika 23.2).

Slika 23.2. Struktura izraza za definisanje prilagodljive veličine fonta

Sa slike 23.2. može se videti logika koja se primenjuje prilikom definisanja izraza za računanje veličine teksta. Računanje se obavlja relativno u odnosu na širinu vidnog polja i upravo zbog toga se unutar izraza pojavljuje vrednosti od `100vw`. Preostale numeričke vrednosti unutar izraza nisu izabrane slučajno. One se odnose na sledeće:

- minimalna veličina fonta – 13px;
- minimalna širina vidnog polja – 320px;
- maksimalna veličina fonta – 21px;
- maksimalna širina vidnog polja – 1600px.

Osnovni razlog kompleksnosti upravo prikazanog izraza jeste logika koja omogućava da veličina teksta pri širini vidnog polja od 320px bude 13px, a pri širini vidnog polja od 1600px – 21px. U to se možemo i uveriti na sledeći način:

```
13 + (21 - 13) * ((320-320) / (1600 - 320)) =
13 + 8 * (0 / 1280) =
13 + 8 * 0 =
13
```

Prikazana računica oslikava proces dolaska do vrednosti kada je širina vidnog polja 320px. Kada je širina vidnog polja 1600px, računica je sledeća:

```
13 + (21 - 13) * ((1600-320) / (1600 - 320)) =
13 + 8 * (1280 / 1280) =
13 + 8 * 1 =
13 + 8 =
21
```

Sada je potpuno jasno da će pri širini vidnog polja od 1600px veličina fonta biti 21px, a pri širini vidnog polja od 320px – 13px. Za sve širine vidnog polja između 320px i 1600px, veličina fonta će se proporcionalno menjati.

Kako bi upravo prikazani pristup za postizanje responsive teksta bio potpun, neophodno je pokriti još jedan scenario, odnosno situacije u kojima je vidno polje uže od 320px ili šire od 1600px. Tako nešto se vrlo jednostavno može rešiti korišćenjem medija upita:

```
@media screen and (max-width: 319px) {
            #main-container {
            font-size: 13px;
      }
}

@media screen and (min-width: 1600px) {
      #main-container {
      font-size: 21px;
}
```

attr()

Još jedna CSS funkcija koja će biti obrađena u ovoj lekciji jeste funkcija `attr()`. CSS funkcija `attr()` koristi se za čitanje vrednosti nekog atributa na selektovanom elementu, koja se zatim može koristiti unutar stilizacije.

Veoma često se `attr()`CSS funkcija koristi za postavljanje vrednosti CSS svojstva `content` koje se može koristiti na pseudoelementima:

```
content: attr(data-descr);
```

Upravo prikazana linija CSS koda ilustruje čitanje vrednosti atributa `data-descr` i njeno postavljanje za vrednost svojstva `content`.

Primer 3 – Korišćenje attr() funkcije u kombinaciji sa data atributima za kreiranje tooltip-a

Kao što je već rečeno, CSS funkcija `attr()` veoma se često koristi za čitanje vrednosti proizvoljnih HTML data atributa koji po pravilu započinju prefiksom `data-`. Upravo jedan takav primer biće ilustrovan u nastavku ove lekcije.

Unutar radnog okruženja HTML struktura primera sastojaće se iz jednog paragrafa. Može se primetiti da se unutar paragrafa nalazi jedan span element, koji obuhvata reč *maximus*. Span element poseduje i jedan specijalan *data atribut*. Vrednost ovog atributa biće iskorišćena kao tekst za tooltip. Tooltip treba da se pojavi prilikom prelaska strelice miša preko span elementa.

Radno okruženje

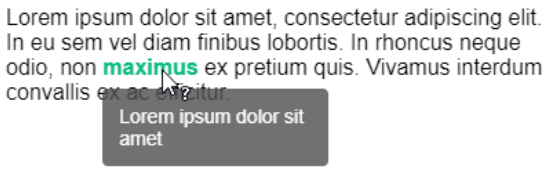
HTML fajl:

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. In eu sem vel diam finibus lobortis. In rhoncus neque odio, non <span data-descr="Lorem ipsum dolor sit amet">maximus</span> ex pretium quis. Vivamus interdum convallis ex ac efficitur.</p>
```

CSS fajl:

```
body {
      width: 300px;
}
p {
      font-family: sans-serif;
}

span[data-descr] {
      position: relative;
      font-weight: bold;
      color: rgb(12, 187, 129);
      cursor: help;
}

span[data-descr]:hover::after {
      content: attr(data-descr);
      position: absolute;
      left: 0;
      top: 24px;
      min-width: 140px;
      border-radius: 4px;
      background-color: rgba(77, 77, 77, 0.8);
      padding: 12px;
      color: white;
      font-size: 14px;
      font-weight: normal;
      z-index: 1;
}
```

Unutar CSS-a prvo su stilizovani svi `span` elementi sa atributom `data-descr`, a zatim je `::after` pseudoelement dodat na `span` samo u specijalnoj situaciji, odnosno kada dođe do hovera. Na taj način, tooltip, koji je definisan uz pomoć `::after` pseudo elementa, pojaviće se samo u situacijama kada se strelica miša nađe iznad `span` elementa koji ima atribut `data-descr`. Slika 23.3. ilustruje efekat prikazanog koda za kreiranje tooltipa.



Slika 23.3. Prikaz tooltipa koji je dobijen korišćenjem pseudoelementa, data atributa i attr() CSS funkcije

Za realizaciju upravo prikazanog primera iskorišćena je i funkcija `attr()`. Ona omogućava prikaz teksta unutar tooltipa, koji dolazi iz same HTML strukture, odnosno sa atributa `data-descr`.

Pokušajte da primer unutar radnog okruženja proširite sa više tooltipova različitih boja.

Rezime

- CSS varijable, CSS promenljive ili CSS proizvoljna svojstva (*custom properties*) nazivi su koji se koriste da označe specijalne entitete CSS jezika, koji autoru dozvoljavaju definisanje vrednosti koje se mogu koristiti na proizvoljnom broju mesta unutar CSS stilizacije.
- CSS proizvoljna svojstva se definišu navođenjem dva karaktera crtica (--) ispred naziva svojstva.
- Proizvoljna CSS svojstva smanjuju potrebu za ponavljanjem koda i poboljšavaju razumljivost i čitljivost.
- Selektor `:root` predstavlja jednu od pseudoklasa, koja se može koristiti za selektovanje korenog elementa unutar dokumenta.
- Korišćenje vrednosti proizvoljnih CSS svojstava obavlja se upotrebom CSS funkcije `var()`.
- Proizvoljna CSS svojstva se nasleđuju niz strukturu HTML elemenata.
- Proizvoljno CSS svojstvo može imati podrazumevanu vrednost, koja se definiše kao drugi parametar CSS funkcije `var()`.
- CSS funkcije su specijalni elementi CSS jezika koji se mogu koristiti za definisanje vrednosti različitih CSS svojstava.
- CSS funkcija `calc()` omogućava obavljanje jednostavnih proračuna korišćenjem osnovnih računskih operacija prilikom definisanja vrednosti nekog CSS svojstva.
- CSS funkcija `attr()`, koristi se za čitanje vrednosti nekog atributa na selektovanom elementu, koja se zatim može koristiti unutar stilizacije.