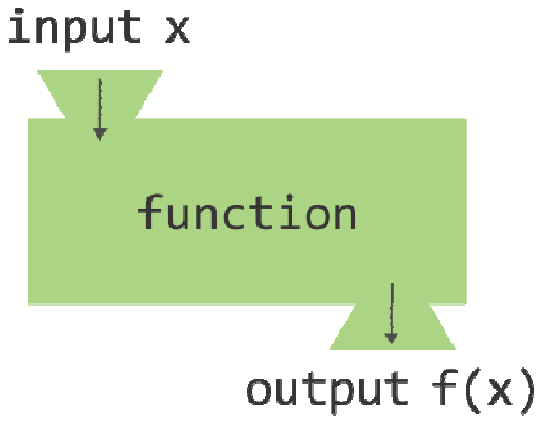
Funkcije

Kada je u prethodnim lekcijama bilo reči o blokovima koda, spomenut je pojam funkcija. Funkcije su osnovni gradivni blok gotovo svakog programskog jezika, te se stoga programiranje ne može zamisliti bez njihove upotrebe. Naravno, ni JavaScript nije izuzetak, te će stoga lekcija pred vama biti posvećena pojmu JavaScript funkcija.

Šta su funkcije?

Funkcija je specijalni blok koda, zadužen za izvršenje određene logike. Pri tome, funkcija prilikom izvršavanja takve logike može da koristi određene vrednosti, koje se nazivaju ulazni parametri, a može i emitovati svoj rezultat kao povratnu vrednost (slika 7.1).



Slika 7.1. Funkcija

Na slici 7.1. slovom x je označen ulazni parametar, odnosno vrednost koju funkcija dobija na obradu. Nakon završene obrade, funkcija emituje rezultat izvršavanja. Ipak, nije obavezno da funkcija ima ulazne i izlazne parametre.

Sa slike 7.1. se još može videti da je funkcija jedna zaokružena celina. To je vrlo važna osobina funkcije iz koje proizlaze brojne prednosti funkcionalnog programiranja.

Najznačajnije prednosti su:

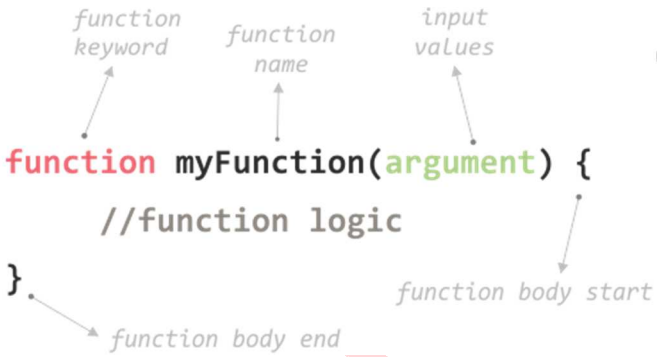
- enkapsulacija,
- smanjenje redundantnosti koda,
- brzina kodiranja,
- portabilnost.

Funkcija je zaokružena celina u potpunosti sposobna da izvrši jedan deo posla, zato što sadrži sve potrebne elemente za takvo nešto. Grupisanje svih potrebnih elemenata u jednu celinu drugačije se naziva **enkapsulacija**.

Takođe, prilikom pisanja koda često se javlja potreba da se određeni deo koda izvrši nekoliko puta na različitim mestima, odnosno u različitim situacijama. Da funkcije ne postoje, za rešavanje spomenutog problema bilo bi neophodno jedan isti kôd kopirati na sva ona mesta na kojima se javi potreba za određenom funkcionalnošću. Na taj način bi se znatno povećala **redundantnost koda**, smanjila preglednost i otežalo održavanje. Funkcije ovakve probleme elegantno rešavaju omogućavanjem da se jednom definisana logika upotrebi onoliko puta koliko je to potrebno.

Definisanje funkcija

U programskom jeziku JavaScript funkcija se definiše korišćenjem ključne reči `function`. Nakon ove ključne reči navodi se naziv funkcije, eventualni parametri i blok koda, oivičen uglastim zagradama (slika 7.2).



Slika 7.2. Sintaksa funkcije u JavaScriptu

Funkcija sa slike ima naziv `myFunction`. Za imenovanje funkcija važe sva pravila kao i za definisanje bilo kojeg drugog identifikatora:

- nazivi mogu biti sastavljeni iz slova, brojeva i karaktera donja crta i dolar,
- naziv ne sme započeti brojem, odnosno mora započeti slovom, donjom crtom (_) ili karakterom dolar ($),
- nazivi su osetljivi na mala i velika slova,
- za definisanje naziva koristi se UNICODE set karaktera,
- ključne reči ne mogu biti nazivi funkcija.

Zbog čega su nam potrebne funkcije?

Zamislite sada situaciju po kojoj je potrebno u nekoj aplikaciji računati površinu pravougaonika. Površina pravougaonika se računa tako što se pomnože dužine njegovih stranica. JavaScript kôd koji bi takvo nešto obavljao mogao bi da izgleda ovako:

```
var a = 5;
var b = 10;

var rect_area = a*b;
```

Prikazani kôd je potpuno funkcionalan, odnosno, bez problema računa površinu pravougaonika sa dužinama stranica 5 i 10. Ali šta će biti ukoliko je potrebno napraviti univerzalnu funkcionalnost, koja bi omogućila računanje površine pravougaonika bilo kojih veličina stranica? Da ne postoje funkcije, bilo bi neophodno uvek iznova ponavljati već prikazanu logiku za računanje površine, ali sa drugačijim vrednostima promenljivih. Funkcionalno programiranje omogućava da se upravo opisana logika enkapsulira unutar jedne funkcije:

```
function calculateRectArea(a, b){
      return a*b;
}
```

Kreirana funkcija je nazvana `calculateRectArea`, prihvata dva parametra, obavlja računanje površine i emituje rezultat kao povratnu vrednost.

Na ovaj način ne samo da je izvršena enkapsulacija logike za računanje površine pravougaonika, već je funkcionalnost generalizovana, pa je tako primenljiva na pravougaonicima svih veličina.

Pozivanje funkcije

Definisanje funkcije, samo po sebi, ne pokreće izvršavanje njene logike. Kako bi se logika funkcije uposlila, nju je potrebno pozvati. Sledeća linija koda ilustruje pozivanje nešto ranije kreirane funkcije:

```
calculateRectArea(5, 10);
```

Na ovaj način poziva se funkcija sa nazivom `calculateRectArea` i njoj se prosleđuju ulazni parametri `5` i `10`. Tako se aktivira logika funkcije, koja je definisana unutar njenog tela.

Funkcija obavlja posao definisan njenom logikom i vraća vrednost izvršavanja. Ukoliko se obavi testiranje prikazanog primera, sve ovo što je navedeno ostaće nevidljivo za posmatrača, odnosno povratna vrednost funkcije nigde neće biti vidljiva. To je potpuno logično, s obzirom na to da u prikazanoj liniji u kojoj se poziva funkcija nije precizirano šta treba uraditi sa rezultatom koji funkcija isporučuje. Stoga će primer biti modifikovan na sledeći način:

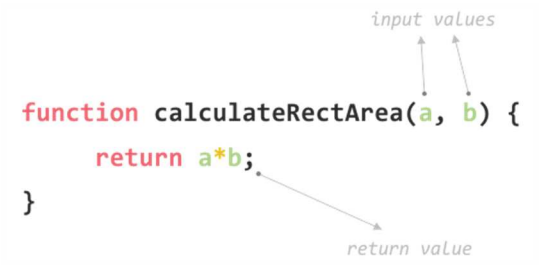
```
function calculateRectArea(a, b){
   -  return a*b;
}

var result = calculateRectArea(5, 10);
document.write(result);
```

Sada je rezultat funkcije smešten unutar promenljive sa nazivom `result`. Da je to stvarno tako, potvrđuje ispis koji proizvodi prikazani kôd:

```
50
```

Ulazni parametri i povratna vrednost

Funkcija može imati ulazne parametre (argumente) i povratnu vrednost, ali to nije obavezno. Prethodni primer, u kome je prikazana funkcija za računanje površine pravougaonika, ilustrovao je upotrebu ulaznih parametara i povratne vrednosti.



Slika 7.3. Ulazni i izlazni parametri funkcije

Na slici 7.3. jasno su obeleženi ulazni i izlazni parametri. Ulazni parametri su a i b, dok je izlazni parametar njihov proizvod. Naredba kojom se definiše povratna vrednost funkcije započinje ključnom rečju `return`.

Potpuno je legitimno imati funkcije bez ikakvih parametara. Drugim rečima, funkcija ne mora imati ni ulazne, a ni izlazne parametre. Nekoliko takvih varijacija biće prikazano u nastavku.

Primer funkcije sa jednim ulaznim parametrom, bez povratne vrednosti:

```
function sayHello(name){
      console.log("Hello " + name);
}
```

Funkcija `sayHello()` prihvata jedan parametar, ali nema izlaznih parametara, zato što unutar njenog tela nije upotrebljena ključna reč `return`. Ova funkcija se može pozvati na sledeći način:

```
sayHello("John");
```

Unutar konzole se dobija:

```
Hello John
```

Primer funkcije bez ulaznih parametara i bez povratne vrednosti

```
function sayHelloWorld() {
      console.log("Hello World");
}
```

Funkcija `sayHelloWorld` ne prihvata nijedan parametar. Takođe ona nema ni povratnih vrednosti. Svakim pozivom ove funkcije unutar konzole se ispisuje tekst *Hello World*:

```
sayHelloWorld(); //Hello World
```

Pitanje

Da li je neophodno da funkcija vraća neku vrednost?

- Jeste
- **Nije**

Objašnjenje:

Nije neophodno da funkcija vraća neku vrednost, baš kao što nije neophodno ni da prihvata ulazne parametre.

Podrazumevana vrednost ulaznih parametara

Prilikom pozivanja funkcija koje poseduju ulazne parametre ne postoji obaveza da se takvi parametri uistinu i definišu. Na primer, pogledajte sledeću funkciju:

```
function calculateRectArea(a, b) {
	console.log(a);
	console.log(b);
	return a * b;
}
```

Reč je o funkciji koja je već viđena u ovoj lekciji. Ona prihvata dva ulazna parametra. Ipak, mi je možemo pozvati na sledeći način:

```
calculateRectArea();
```

Funkcija je sada pozvana bez definisanja vrednosti koje će predstavljati njene ulazne parametre. Ovo je potpuno legitiman scenario, što znači da ne proizvodi nikakvu grešku. Ipak, postavlja se pitanje šta će se dogoditi unutar funkcije sa vrednostima ulaznih parametara.

Ulazni parametri funkcija poseduju podrazumevanu vrednost. Reč je o vrednosti `undefined`. Upravo zbog toga će poziv `calculateRectArea()` funkcije, bez prosleđivanja parametara, da proizvede sledeći rezultat:

```
Undefined

undefined
```

Obratite pažnju na to da su unutar funkcije `calculateRectArea()` postavljene dve naredbe za ispis vrednosti ulaznih parametara. Kada se parametri ne proslede, takve dve naredbe unutar konzole emituju vrednosti `undefined`.

Podrazumevane vrednosti ulaznih parametara JavaScript funkcija moguće je i ručno definisati. Dovoljno je u potpisu funkcije, nakon svakog parametra, definisati i njegovu podrazumevanu vrednost:

```
function calculateRectArea(a = 0, b = 0) {
	console.log(a);
	console.log(b);
	return a * b;
}
```

Sada su nakon ulaznih parametara definisane i njihove podrazumevane vrednosti, baš kao da se obavlja inicijalizacija promenljivih. Nakon pozivanja ovakve funkcije bez navođenja vrednosti parametara, unutar konzole će biti dobijeno:

```
0

0
```

Umesto `undefined`, ulazni parametri sada imaju podrazumevane vrednosti 0.

Varijabilni broj ulaznih parametara

Već je rečeno da JavaScript funkcije mogu imati proizvoljan broj ulaznih parametara. Ipak, korišćenjem dosadašnjih pristupa prilikom deklaracije funkcije bilo je neophodno definisati fiksni broj parametara. Drugim rečima, funkciji koja prihvata dva parametra nije bilo moguće proslediti tri ili četiri parametra. Ipak, JavaScript omogućava da funkcija bude u stanju da prihvati promenljivi (varijabilni) broj parametara:

```
function myFunction(...arguments) {
      console.log(arguments);
}
```

Upravo je prikazana funkcija `myFunction()` koja može da prihvati varijabilni broj parametara. To je postignuto dodavanjem tri tačke (...) ispred naziva ulaznog parametra (`arguments`). Unutar tela funkcije obavlja se ispisivanje prosleđenih vrednosti:

```
myFunction(1, 2, 3, 4);
```

Funkciji su prosleđena četiri parametra, pa će efekat ovakvog koda biti:

```
[1, 2, 3, 4]
```

Bitno je znati da na ovaj način JavaScript sve parametre smešta unutar jednog niza.

Ugrađene JavaScript funkcije

Iako to možda i niste znali, u dosadašnjem toku ovoga kursa korišćeno je nekoliko različitih funkcija. Pri tome se pod pojmom *korišćenja* misli na njihovo pozivanje, s obzirom na to da mi samostalno nismo kreirali takve funkcije. Naime, reč je o funkcijama koje su automatski dostupne unutar JavaScript jezika. Stoga se one često nazivaju ugrađene (engl. *built-in functions*).

Za ispis poruka unutar konzole u prethodnim lekcijama veoma često je korišćena funkcija `log()`, objekta `console`:

```
console.log("Hello World");
```

Funkcija `log()` je namenjena za ispis poruke unutar konzole web pregledača. Prihvata jedan ulazni parametar i nema izlaznih parametara.

U prethodnim lekcijama korišćena je i jedna ugrađena funkcija koja se nalazi unutar objekta `document`. Reč je o funkciji `write()`, koja je korišćena za ispis teksta unutar HTML dokumenta:

```
document.write("Hello World");
```

Funkcija `write()` može da prihvati jedan parametar ili više parametara i nema povratne vrednosti.

Funkcije `log()` i `write()` samo su neke od mnogih funkcija koje su automatski dostupne programerima prilikom pisanja JavaScript koda. Sve one su podeljene u nekoliko kategorija u zavisnosti od njihove namene i biće predstavljene u narednim lekcijama kada se javi potreba za takvim nečim.

Primer – Kreiranje funkcije za konverziju temperature izražene u farenhajtima u celzijuse

U nastavku će biti prikazan primer kreiranja jedne funkcije za obavljanje konverzije temperature izražene u farenhajtima u celzijuse. Formula po kojoj se takva konverzija obavlja izgleda ovako:

```
T(c) = (T(f)- 32) * 5/9;
```

Funkcija će izgledati ovako:

```
function f2c(f) {
      let c = (f - 32) * 5 / 9;
      return c;
}
```

Funkcija za konverziju nosi naziv `f2c` i prihvata jedan ulazni parametar koji je imenovan nazivom `f`. Unutar funkcije obavlja se računanje na osnovu nešto ranije prikazane formule. Rezultat se smešta unutar lokalne promenjive `c`, a zatim se emituje takva promenljiva kao povratna vrednost. Ovakva funkcija se može pozvati i njen rezultat prikazati unutar HTML dokumenta, na sledeći način:

```
document.write(f2c(150).toFixed(2));
```

Funkcija `f2c()` se poziva na mestu na kome se navodi parametar metode `write()`. Tako će povratna vrednost funkcije `f2c()` biti odštampana na HTML stranici. Funkciji `f2c()` se prosleđuje vrednost `150`, a zatim se nad povratnom vrednošću ove funkcije poziva i metoda `toFixed()`, čime se dobijeni rezultat zaokružuje na dve decimale. U ovakvoj situaciji unutar HTML dokumenta se dobija sledeći ispis:

```
65.56
```

Rezime

- Funkcija je specijalni blok koda zadužen za izvršenje određene logike.
- Funkcija prilikom izvršenja logike može da koristi određene vrednosti, koje se nazivaju ulazni parametri, a može i emitovati svoj rezultat kao povratnu vrednost.
- U programskom jeziku JavaScript funkcija se definiše korišćenjem ključne reči function.
- Logika funkcije se definiše unutar njenog tela.
- Kako bi se logika funkcije uposlila, nju je potrebno pozvati.
- Funkcija može imati ulazne parametre i povratnu vrednost, ali to nije obavezno.
- Povratna vrednost funkcije se emituje ključnom rečju return.
- Podrazumevane vrednosti ulaznih parametara JavaScript funkcija moguće je ručno definisati navođenjem vrednosti nakon svakog parametra.
- JavaScript funkcije mogu da prihvataju varijabilni broj parametara, navođenjem karaktera tri tačke pre naziva parametra.
- JavaScript poseduje veliki broj ugrađenih funkcija, koje su automatski dostupne za korišćenje prilikom pisanja koda.