JS FUNKCIJE

(još malo)

Novi Sad, 2019

Miroslav Zarić

DEFINICIJA FUNKCIJE

- JavaScript funkcije se definišu korišćenjem ključne reči ***function***.

- Za definiciju funkcije može se koristiti konstrukt tipa deklaracije funkcije ili funkcionalni izraz.

DEKLARACIJA FUNKCIJE

function ***nazivFunkcije***(parametar) {

  // telo funkcije

}

- Iza deklaracije funkcije se obično ne stavlja **;** jer se separator koristi za odvajanje izvršivih izraza.

- Deklarisana funkcija se ne izvršava odmah - mora se pozvati negde iz ostatka koda npr:

var x = ***nazivFunkcije***(*"tekst za parametar"*);

FUNKCIONALNI IZRAZ

- Funkcija se može definisati i putem izraza (*function expression*)

```
var x = function (a, b) {return a + b};
```

- Kako ovde nema naziva funkcije u definiciji - kako se ova funkcija posle poziva?

```
var y = x(10, 27);
```

FUNKCIONALNI IZRAZ

- Često se na ovaj način definiše funkcija umesto putem deklaracije

- Varijabla praktično može biti ***tipa*** funkcije.
  A funkcija za JS nije ništa drugo nego objekat.

```
var mojaFunkcija = function(a, b){return a * b};

var x = mojaFunkcija(4, 3);
```

FUNCTION HOISTING (PROMOCIJA FUNKCIJA)

- JavaScipt dozvoljava da se varijabla, pa tako i funkcija deklariše **nakon** što je već negde upotrebljena. Ova osobina se na engleskom naziva *hoisting* (uzdizanje, promocija)

- *Hoisting* je JavaScript podrazumevano ponašanje - automatski pri interpretiranju koda "podiže" sve deklaracije na početak trenutnog opsega važenja (početak bloka, funkcije ili script fajla)

- Ovo podizanje na vrh opsega važi SAMO za deklaracije ne i za izraze inicijalizacije!!! **Kao posledica ovoga funkcije definisane izrazima se ne "podižu".**

FUNCTION HOISTING (PRIMER)

Ovo može:

```
var y = saberi(10,18);

function saberi(a,b){
   return a+b;
}
```

Ovo ne može:

```
var y = saberi(10,18);


var saberi=function(a,b){return a+b;};
```

SAMOPOZIVAJUĆE FUNKCIJE

- Funkcionalni izraz se može napisati tako da funkcija bude samopozivajuća, tj. da odmah nakon definicije sledi njen poziv.

- Ovo je moguće samo kada se za definiciju funkcije koristi izraz. Nije moguće kada se koristi deklaracija funkcije.

- Ovo se često može videti u kodu koji koristi neki od JS Frameworka ili biblioteka (jQuery, Vue, …)

SAMOPOZIVAJUĆE FUNKCIJE (PRIMER)

```
(function () {
    var x = "Hello!!";
    alert(x);
})();


(function(message) {
    alert(message);
})('Moja poruka');
```

PARAMETRI FUNKCIJE

- Funkcije mogu, ali ne moraju imati parametre
- ***Parametri*** su nazivi zadati u definiciji funkcije
    - U JavaScript-u se ne definiše tip parametra
    - JavaScript ne radi proveru tipa prosleđenih argumenata
    - JavaScript funkcije NE proveravaju broj primljenih argumenata niti da li se on slaže sa brojem parametara navedenim u definiciji
- ***Argumenti*** su stvarne vrednosti koje se u momentu pozivanja funkcije prosleđuju funkciji.
- Argumenti se prosleđuju **po vrednosti.**
- Zbog načina kako JS radi sa objektima, **ako je argument objekat** on se prosleđuje **po referenci**.

PODRAZUMEVANA VREDNOST PARAMETARA

- Ukoliko se funkcija pozove sa nedovoljnim brojem argumenata (manjim od broja parametara) sve nedostajuće vrednosti će biti postavljene na specijalnu vrednost ***undefined***
- Ovo je ponekad OK, ali je često dobro proveriti u kodu

```
function mojaFunkcija(a, b) {
   if (b === undefined) {
      b = 0;
   }
   return a*b;
}
```

PODRAZUMEVANA VREDNOST PARAMETARA

- Od verzije ES6 (2015. god) dozvoljeno je zadati podrazumevanu vrednost parametra pri deklaraciji funkcije

```
function mojaFunkcija(a=1, b=0) {
   return a*b;
}
```