
Napredni JavaScript
(napredni koncepti)

Napredni JavaScript

Ključne riječi
closures, scope, “use strict”, OOP, patterns, ES6, API

Cjeline

Uvod u napredne koncepte

Funkcije - napredni koncepti

Vježba 1

JavaScript OOP - 1.dio

JavaScript OOP - 2.dio

Vježba 2

Obrasci programskog koda

ES6 standard - 1.dio

ES6 standard - 2.dio

Rad s API-jima

Vježba 3

Animacije u JavaScriptu

Literatura

1. JavaScript: The Definitive Guide
2. JavaScript Patterns: Build Better Applications with Coding and Design Patterns

Online literatura

http://developer.mozilla.com
https://www.w3schools.com/js/

Osnove programiranja

Uvod u napredne koncepte

Povezana literatura

JavaScript: The Definitive Guide

- Part I. Core JavaScript, 3. Types, values and variables

https://www.w3schools.com/js/js_type_conversion.asp

Eksplicitna konverzija podataka

Par je metoda koje možemo koristiti kad želimo konvertirati vrijednost:

1. Upotrijebiti **Number, String, Boolean i Object** funkcije
2. **toString** metoda za konverziju bilo koje vrijednosti u string
3. **parseInt** i **parseFloat** parsiraju vrijednost i izvlače broj

Hoisting (izvlačenje)

Hoisting je premještanje svih deklaracija na vrh trenutnog opsega (na vrh trenutne skripte ili trenutne funkcije).

Varijable je uvijek dobro deklarirati na početku funkcije!

Objekt

Osnovni oblik podataka JavaScripta-a, zbirka svojstava od kojih svako ima ime (ključ) i vrijednost.

```
{ name: 'Golden retriever' }

function bark() {}

['John', 'Jane', 'Joe']
```

I objekti i funkcije i nizovi su objektni tip podatka i mogu imati metode i svojstva!

Opseg (scope)

Opseg određuje dostupnost varijabli, objekata i funkcija iz različitih dijelova koda.

JavaScript koristi **leksički opseg** implementiran **funkcijama.**

Postoje 2 tipa dosega opsega u Javascriptu: **lokalni** i **globalni**.

```
var x = 2;

function() {
    var x = 1;
}

console.log(x);
```

Globalne i lokalne varijable

```
var x = 2;

function() {
    x = 1;
}

console.log(x);
```

Koju će vrijednost ispisati console.log?

U slučaju da je varijabla deklarirana unutar funkcije, ona je **lokalna,** *osim ako se izostavi ključna riječ var*.

U slučaju da je varijabla deklarirana van funkcije, ona je **globalna**.

Namespace

Problem s globalnim varijablama je taj što ih dijele svi kodovi u vašoj JavaScript aplikaciji ili web stranici.

Oni žive u istom **globalnom prostoru imena** i uvijek postoji mogućnost sudara imenovanja - kada dva odvojena dijela aplikacije definiraju globalne varijable s istim nazivom, ali s različitim svrhama.

this

Ključna riječ **this** odnosi se na objekt konteksta izvođenja.

1. Van objekta ili u funkciji se odnosi na *globalni objekt*
2. U metodi objekta se to odnosi na sam objekt

   … o drugim kontekstima pričat ćemo na Naprednom Javascript dijelu

```
var person = {
     name: 'John',
     speak: function(word) {
          return this.name;
     }
};
```

Napredni JavaScript

Funkcije - napredni koncepti

Povezana literatura

JavaScript: The Definitive Guide

- 8. Functions

JavaScript Patterns

- 4. Functions

Načini korištenja funkcije

U JavaScriptu funkcije su objekti prve klase, što znači se se tretiraju kao bilo koja druga varijabla i da mogu imati vlastita svojstva i metode.

Funkcije se mogu:

- dodijeliti varijabli
- proslijediti drugoj funkciji kao argument
- vratiti iz druge funkcije (koristeći return)

Funkcija koja vraća drugu funkciju zove se HOF (Higher-Order Function)

Načini korištenja funkcije

Kao varijabla

```
function vratiNesto(x) {
          return x;
}

var y = vratiNesto();
```

```
function vratiNesto(x) {
          return x;
}

var y = vratiNesto;

y(“Hello world”);
```

Načini korištenja funkcije

Kao argument druge funkcije

```
function vratiNesto(x) {
          var y = x(1);
          return y;
}

function vratiMene(z){
          return z/2;
};

var y = vratiNesto(vratiMene);
```

Načini korištenja funkcije

Kao vraćena vrijednost

```
function vratiNesto(x) {
        return function() {
                return x;
    };
}

var y = vratiNesto(true);
```

```
function vratiNesto(x) {
        return function() {
                return x;
    };
}

var y = vratiNesto(true)();
```

Anonimna funkcija

```
(function vratiNesto() {
        var x = 1;
        return x;
})
```

Anonimna funkcija je funkcija koja nema ime. Može biti izravno dodijeljena varijabli, ili obuhvaćena zagradama (npr. da ne zagađuje globani scope).

```
(function vratiNesto() {
        var x = 1;
        return x;
})()
```

Immediately Invoked Function Expression (IIFE)
tj.
Samopozivajuća anonimna funkcija

Closure

U računalnoj znanosti, closure je kombinacija objekta funkcije i dosega (scopea) unutar kojeg su varijable funkcije razriješene.

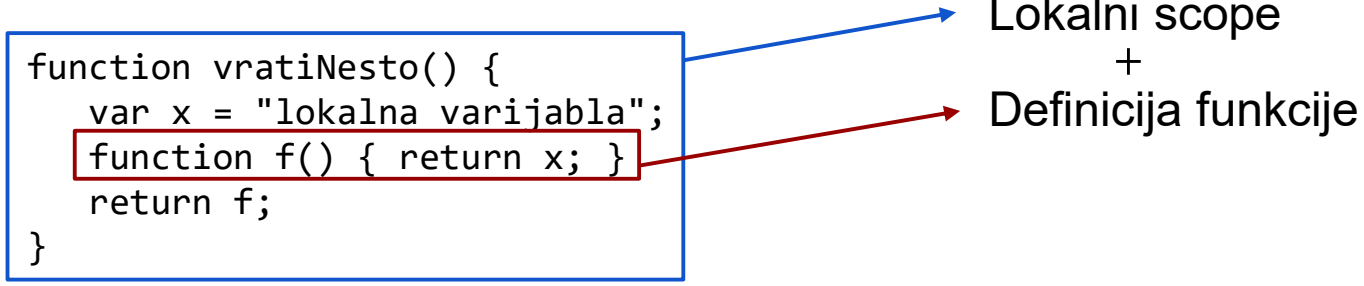
Sve funkcije u JavaScriptu implementiraju **closure** tehniku**.**
Doseg (scope) varijabla funkcije je onaj unutar kojeg je funkcija definirana, a ne unutar kojeg je pozvana!

Closure

```
var x = "globalna varijabla";

function vratiNesto() {
    var x = "lokalna varijabla";
    function f() { return x; }
    return f;
}

var y = vratiNesto()();
```



Funkcionalno programiranje

Funkcionalno programiranje je jedna od programskih paradigmi, uz npr. objektno-orijentirano programiranje. Više o tome u React modulu!

> Funkcionalno programiranje je programska paradigma koji računanje smatra kao evaluaciju matematičkih funkcija i izbjegava promjenu stanja i promjenjivih podataka.