



JavaScript

Treći dio



Pregled

- Scope
- Hoisting
- Stringovi i njihovi metodi
- Brojevi i njihovi metodi
- Nizovi i njihovi metodi
- Math
- Zadaci



Obnavljanje

- Coercion, koja dva tipa imamo i čemu služi
- Falsy i Truthy vrijednosti
- Razlika između `==` i `===`
- Šta smo rekli, koja je razlika između function expression i function statement
- Razlika između argumenta i parametra funkcije
- Metodi kod objekata



Scope

- Blok je grupa od 0 ili više naredni
- Može biti if statement, loop, a može biti i blok { code }
- Scope je opseg važenja varijable, a zavisi od toga u kom se bloku nalazi varijabla
- Sad ćemo detaljnije da vidimo razliku između **var**, **const** i **let**
- Varijable deklarisanе sa **var** imaju function scope, što znači da promjenljive deklarisanе sa var imaju opseg važenja unutar cijele funkcije
- Varijable deklarisanе sa **let** ili **const** imaju blok scope, tj. postoje u bloku u kom su deklarisanе
- Pogledajmo primjere (code-1)
- Ako deklarirate promjenljivu sa const, toj promjenljivoj ne može biti dodijeljena nova vrijednost ali ako je tip promjenljive niz ili objekata šta onda?
 - **const** ne dozvoljava promjenu reference za nizove i objekte!
 - Radićemo šta znači čuvanje podataka po vrijednost, a šta po referenci
- Koristite **const** kad je god moguće, **let** ako želite novo dodjeljivanje
- Pogledajmo još primjera (code-2)

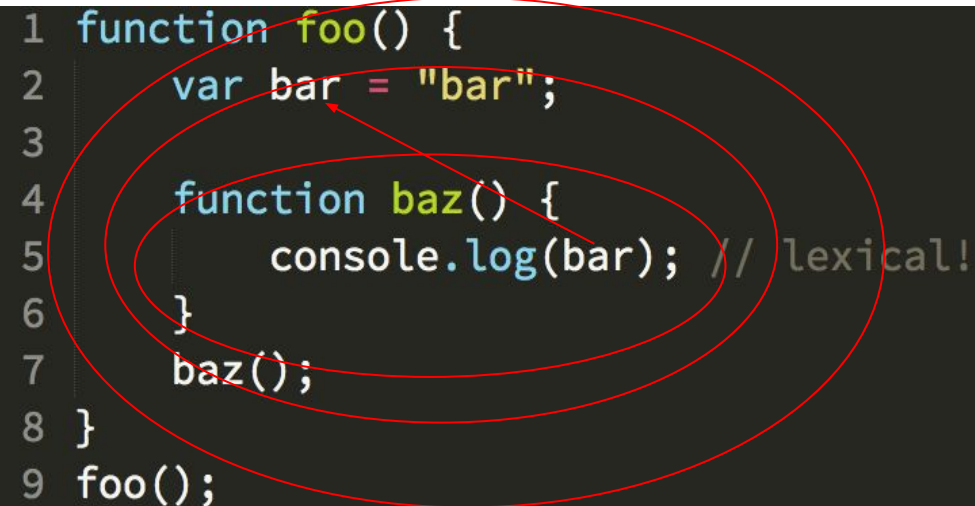


Scope, named function expression

- Imenovanje funkcija koje se dodjeljuju varijabli
- Ima niz prednosti:
 - Self-reference (za rekurzivne pozive) - radićemo kratko i rekurziju
 - Lakše za debugovanje stack trace-a (stack trace je niz pozivanja funkcija)
 - Bolje za dokumentovanje koda
- Varijable koje čuvaju funkcije najčešće se definišu sa **const**

Scope, lexical

- Lexical scope



```
1 function foo() {  
2     var bar = "bar";  
3  
4     function baz() {  
5         console.log(bar); // lexical!  
6     }  
7     baz();  
8 }  
9 foo();
```

The diagram illustrates lexical scope resolution. It features three nested red ellipses on a dark background. The outermost ellipse encompasses the entire code block. The middle ellipse encompasses the function `foo()` and its body, including the `var bar = "bar";` declaration. The innermost ellipse encompasses the function `baz()` and its body. A red arrow originates from the `bar` variable in the `console.log(bar);` statement within the `baz()` function and points to the `var bar = "bar";` declaration within the `foo()` function, demonstrating that `baz()` looks up the variable `bar` in the scope of its caller, `foo()`.



Scope, function scoping i IIFE

- Function scoping - scope (opseg važenja) na nivou funkcije
- Pogledati primjer (code-3)
- IIFE (immediately-invoked function expression) su funkcije koje se automatski pozivaju
- Jedna od glavnih primjena je **razbijanje koda na module (Module pattern)** i **skrivanje globalnih varijabli**
- Sintaksa: **(function name(params) {code goes here})(args);**
- Ime ovih funkcija je opciono



Hoisting

- Pomjeranje svih deklaracija (varijable i funkcije) na vrh scope-a (na vrh trenutne skripte ili funkcije)
- Pogledajmo par primjera (code-5)
- Nije isto za let/const i var !



Stringovi i njihovi metodi

- Slično kao i kod drugih programskih jezika
- Najjednostavnije ih je kreirati sa “string value” ili ‘string value’ , tj. jednostrukim ili dvostrukim navodnicima
- Interesantni primjeri
 - “string ‘hello’ ”
 - “string \“hello\””
 - Kako biste napisali string koji ima u sebi \
- **U JSu stringovi su immutable, tj. string nije moguće izmijeniti, npr. pretvori 5. slovo u A, itd.**
- Kad kažemo **mutable** mislimo na vrijednosti koje su promjenljive, a **immutable** su fiksirane vrijednosti, tj. nije ih moguće izmijeniti



Stringovi i njihovi metodi, nastavak - 1

- Neke od korisnih metoda za stringove su:
 - **str.length** - vraće dužinu stringa (broj karaktera)
 - **str.indexOf("value")** - vraće index prvog pojavljivanja vrijednosti value u stringu str
 - Ako ne nađe vraća -1
 - **str.lastIndexOf("value")** - vraće index zadnjeg pojavljivanja vrijednosti value u stringu str
 - Ako ne nađe vraća -1
 - I **indexOf** i **lastIndexOf** imaju opcioni drugi parametar koji služe za definisanje startne pozicije za search
 - **str.indexOf("value", 5)** - vraće index prvog pojavljivanja vrijednosti value u string str počevši od pozicije 5
 - **str.search("value")** - isto kao i **indexOf**, pa zašto onda još jedan metod
 - **search()** podržava naprednu pretragu za regularnim izrazima, ali ne podržava drugi argument za definisanje startne pozicije za pretragu



Stringovi i njihovi metodi, nastavak - 2

- Postoje tri načina izvlačenja dijelova stringa
 - **str.slice** (startIndex, endIndex) - izvlači dio string počevši od index startIndex(uključujući) sve do indexa endIndex (ne uključujući endIndex)
 - Mogu da se definišu i negativne vrijednosti za računanje od kraja stringa
 - Ako se preskoči druga vrijednost onda se izvlači dio stringa od zadatog indexa, pa do kraja stringa
 - I ova vrijednost može biti negativna (kreće se od kraja)
 - **str.substring** (startIndex, endIndex) - isto kao slice samo što nisu podržane negativne vrijednosti
 - **str.substr** (startIndex, length) - isto kao slice osim što drugi argument predstavlja koliko karaktera uzimamo nakon startnog indeksa
- String metod **replace()** obavlja replace dijela stringa
 - **str.replace** ("Original", "Replace")
 - Može da koristi i regularne izraze



Stringovi i njihovi metodi, nastavak-3

- Moguće je uvećati/umanjiti slova stringa
 - `str.toUpperCase()` - konvertuje string u velika slova
 - `str.toLowerCase()` - konvertuje string u mala slova
- Za spajanje stringova koristi se `concat()`
 - `str1.concat("concat value", str2)`
- Za uklanjanje bjelina (whitespaces)
 - `str.trim()` - oklanja sve tabove, spaces lijevo i desno od stringa ali ne između riječi u stringu
- Tri su metoda za pristupanje karakterima stringa
 - `str.charAt(index)` - vraće karakter iz stringa sa pozicije index
 - `str.charCodeAt(index)` - vraće unicode(utf-16) karaktera iz stringa sa pozicije index
 - `str[index]` - isto kao `charAt`

```
var str = "HELLO WORLD";
str[0] = "A";// Gives no error, but does not work
str[0];// returns H
```



Stringovi i njihovi metodi, nastavak-4

- Takođe, string je moguće razbiti na više cjelina i to na osnovu splitera
 - **str.split(splitter)** - razbija string na više riječi ako je splitter npr. space, splitter može da bude bila šta što razdvaja stringove
- Kroz praksu ćete zapamtiti ove metode, ima ih još, možete da dodate vaše ali nije preporučljivo jer pretraživač dodaje svoje metode za stringove



Brojevi i njihovi metodi

- Kod JSa svi brojevi se čuvaju kao 64-bit Floating point
 - `var x = 3.14` // broj sa decimalom
 - `var y = 3` // broj bez decimala
 - `var z = 123e5` // 123^{10}
 - `var w = 123e-5` // 123^{-5}
- Nije baš savršeno precizan rad sa brojevima
 - `var x = 0.2 + 0.1` // x je 0.30000000000000004
 - Pa kako da se riješi ovaj problem
 - `var x = (0.2 * 10 + 0.1 * 10) / 10; // x will be 0.3`
- Napomena: Kao što smo vijeli + operator može da služi za sabiranje ili spajanje stringova!



Brojevi i njihovi metodi, nastavak - 1

- Moguće je pozivate metode i nad brojevima. Zašto?
 - JavaScript posmatra sve kao objekat, pa i brojeve
- Neki od korisnih metoda:
 - **number.toString()** - vraće broj kao string
 - **number.toExponential(arg)** - vraće string sa zaokruživanjem i eksponencijalnim zapisom
 - Parametar arg je opcionalan i ako se ne definiše JS neće raditi zaokruživanje
 - Primjer:

```
var x = 9.656;  
x.toExponential(2);    // returns 9.66e+0  
x.toExponential(4);    // returns 9.6560e+0  
x.toExponential(6);    // returns 9.656000e+0
```



Brojevi i njihovi metodi, nastavak - 2

- Još korisnih metoda:

- **number.toFixed(arg)** - vraće string zaokruženog broja na određeni broj decimala, ako se ne definiše arg default je 0

- Primjer

```
var x = 9.656;  
x.toFixed(0);           // returns 10  
x.toFixed(2);           // returns 9.66  
x.toFixed(4);           // returns 9.6560  
x.toFixed(6);           // returns 9.656000
```

- **number.toPrecision(arg)** - vraće string zaokruženog na definisanu dužinu, default je isti taj broj

- Primjer

```
var x = 9.656;  
x.toPrecision();        // returns 9.656  
x.toPrecision(2);       // returns 9.7  
x.toPrecision(4);       // returns 9.656  
x.toPrecision(6);       // returns 9.65600
```




Brojevi i njihovi metodi, nastavak - 3

- Još metoda

- `(num).valueOf()` - vraće broj, zagrede nisu potrebne ako je num varijabla
- Svi tipovi podataka imaju `valueOf` i `toString()`
- Kod brojeva, `valueOf()` metod služi za konverziju iz broja koji je objekat u primitivnu vrijednost number
- Primjer

```
var x = 123;
x.valueOf();           // returns 123 from variable x
(123).valueOf();       // returns 123 from literal 123
(100 + 23).valueOf();  // returns 123 from expression 100 + 23
```



Brojevi i njihovi metodi, nastavak -3

- Koriste se 3 metoda koji konvertuju vrijednost (najčešće string) u broj
 - **Number** (arg) - vraće broj
 - **parseFloat** (arg) - vraće float number
 - **parseInt** (arg) - vraće cio broj
 - Ova tri metoda su globalni JS metodi (dio window objekta)

```
Number(true);           // returns 1
Number(false);          // returns 0
Number("10");           // returns 10
Number(" 10");          // returns 10
Number("10 ");          // returns 10
Number(" 10 ");         // returns 10
Number("10.33");        // returns 10.33
Number("10,33");        // returns NaN
Number("10 33");        // returns NaN
Number("John");         // returns NaN

parseInt("10");          // returns 10
parseInt("10.33");       // returns 10
parseInt("10 20 30");    // returns 10
parseInt("10 years");    // returns 10
parseInt("years 10");    // returns NaN

parseFloat("10.33");     // returns 10.33
```



Brojevi i njihova svojstva (properties)

- Već smo pomenuli neka
 - `Number.MAX_VALUE` - vraće najveći mogući broj koji JS može da sačuva
 - `Number.MIN_VALUE` - ... najmanji
 - `Number.POSITIVE_INFINITY` - predstavlja Infinity
 - `Number.NEGATIVE_INFINITY` - predstavlja -Infinity
 - `Number.NaN` - Predstavlja Not-a-Number vrijednosti



Nizovi i njihovi metodi

- Kao što smo i pomenuli najbolje ih je kreirati sa []
 - `var cars = ["Volvo", "BMW", "Fiat"]`
- Kod JS nizovi počinju sa 0
 - `cars[0]` -> "Volvo"
- Kod JS, elementi niza mogu da budu svi tipovi podataka, pa čak i funkcije (jer su u suštini one samo objekti kao i sve ostalo u JSu).
- Elementi niza mogu da budu različiti tipovi podataka
 - ```
var person = ["John", "Doe", 46];

var person = [];
person["firstName"] = "John";
person["lastName"] = "Doe";
person["age"] = 46;
var x = person.length; // person.length will return 0
var y = person[0]; // person[0] will return undefined
```



# Nizovi i njihovi metodi, nastavak - 1

- Pomenućemo neke metode nizova koji se najviše koriste
  - **arr.toString()** - konvertuje niz u string koji sadrži elemente niza koji su razdvojeni zarezima
  - Primjer

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
```

```
fruits.toString(); // Banana,Orange,Apple,Mango
```

- **arr.join (arg)** - isto kao i prethodni metod samo što ovdje možete definisati separator

```
var fruits = ["1", "2", "3", "4"];
fruits.join(" * "); // 1 * 2 * 3 * 4
```



## Nizovi i njihovi metodi, nastavak - 2

- Vrlo korisni metodi su oni za dodavanje i brisanje elemenata iz niza
  - `arr.push(elem)` - dodaje element na kraj niza
  - `arr.pop()` - briše zadnji element niza
  - `arr.unshift()` - dodaje element na početak niza
  - `arr.shift()` - briše prvi element niza
- Kao i kod većina drugih programskih jezika moguće je da promijenite element niza jednostavno
  - Primjer

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[0] = "Kiwi"; // Changes the first element of fruits to "Kiwi"
```

- Moguće je brisanje elementa na sledeći način (ne preporučuje se, koristite `pop()` i `shift()`)

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
delete fruits[0]; // Changes the first element in fruits to undefined
```



## Nizovi i njihovi metodi, nastavak - 3

- Pogledajmo još neke korisne metode sa nizovima
  - `arr.splice (startPosition, length, elements)` - služi za dodavanje ili brisanje elemenata iz niza
  - Primjer dodavanja:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(2, 0, "Lemon", "Kiwi");
// Banana, Orange, Lemon, Kiwi, Apple, Mango
```
  - Primjer brisanja

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(0, 1); // Removes the first element of fruits
```



## Nizovi i njihovi metodi, nastavak - 3

- Spajanje i izdvajanje nizova

- `arr.concat(arr1, arr2,...)` - spaja nizove u jedan novi niz
- Uvijek vraća novi niz, ne mijenja postojeće
- Primjer

```
var arr1 = ["Cecilie", "Lone"];
var arr2 = ["Emil", "Tobias", "Linus"];
var arr3 = ["Robin", "Morgan"];
var myChildren = arr1.concat(arr2, arr3);
// Concatenates arr1 with arr2 and arr3
```

- `arr.slice(arg)` - izdvaja elemente niza u neki novi niz

```
var fruits = ["Banana", "Orange", "Lemon", "Apple"];
var citrus = fruits.slice(2); // ["Lemon", "Apple"]
var citrus = fruits.slice(1, 3); // ["Orange", "Lemon"]
```





## Nizovi i njihovi metodi, nastavak - 4

- Za nalaženje indeksa elementa u nizu (ako ga ima, ako nema -1) koriste se
  - `arr.indexOf(elem)` - u nizu arr nađi poziciju člana elem
  - `arr.lastIndexOf(elem)` - u nizu arr nađi zadnju poziciju člana elem
  - I prvi i drugi metod mogu da imaju opcioni argument kojim se definiše početna pozicija za pretraživanje
- O ovim metodama ćemo govoriti kad budemo radili funkcionalno programiranje
  - `arr.forEach()`
  - `arr.sort()`
  - `arr.filter()`
  - `arr.map()`
  - `arr.reduce()`
  - `arr.every()`
  - `arr.some()`



# Math metodi

- Omogućava obavljanje matmatičkih operacija

|                          |                                                                                     |
|--------------------------|-------------------------------------------------------------------------------------|
| <code>abs(x)</code>      | Returns the absolute value of x                                                     |
| <code>acos(x)</code>     | Returns the arccosine of x, in radians                                              |
| <code>asin(x)</code>     | Returns the arcsine of x, in radians                                                |
| <code>atan(x)</code>     | Returns the arctangent of x as a numeric value between $-\pi/2$ and $\pi/2$ radians |
| <code>atan2(y, x)</code> | Returns the arctangent of the quotient of its arguments                             |
| <code>ceil(x)</code>     | Returns the value of x rounded up to its nearest integer                            |
| <code>cos(x)</code>      | Returns the cosine of x (x is in radians)                                           |
| <code>exp(x)</code>      | Returns the value of $E^x$                                                          |
| <code>floor(x)</code>    | Returns the value of x rounded down to its nearest integer                          |



# Math metodi, nastavak

|                                   |                                                            |
|-----------------------------------|------------------------------------------------------------|
| <code>floor(x)</code>             | Returns the value of x rounded down to its nearest integer |
| <code>log(x)</code>               | Returns the natural logarithm (base E) of x                |
| <code>max(x, y, z, ..., n)</code> | Returns the number with the highest value                  |
| <code>min(x, y, z, ..., n)</code> | Returns the number with the lowest value                   |
| <code>pow(x, y)</code>            | Returns the value of x to the power of y                   |
| <code>random()</code>             | Returns a random number between 0 and 1                    |
| <code>round(x)</code>             | Returns the value of x rounded to its nearest integer      |
| <code>sin(x)</code>               | Returns the sine of x (x is in radians)                    |
| <code>sqrt(x)</code>              | Returns the square root of x                               |
| <code>tan(x)</code>               | Returns the tangent of an angle                            |



# Math svojstva

- Math sadrži 8 matematičkih konstanti

```
Math.E // returns Euler's number
Math.PI // returns PI
Math.SQRT2 // returns the square root of 2
Math.SQRT1_2 // returns the square root of 1/2
Math.LN2 // returns the natural logarithm of 2
Math.LN10 // returns the natural logarithm of 10
Math.LOG2E // returns base 2 logarithm of E
Math.LOG10E // returns base 10 logarithm of E
```



# Zadaci

- Napisati funkciju koja provjerava da li se string (prvi argument funkcije) završava sa target stringom (drugi argument funkcije)
- Napisati funkciju koja skraćuje string (prvi argument) do unijete dužine (drugi argument). Na kraj stringa dodati ...
- Caesars Cipher
  - Pomjeranje slova za 13 pozicija
  - `str.charCodeAt()` i `String.fromCharCode("num")`
- Vježbajte što više!
- Dosta interesantnih zadataka:
  - [freeCodeCamp](#)
  - [codeWars](#)
  - [HackerRank](#)