



Prirodoslovno-matematički fakultet
Matematički odsjek
Sveučilište u Zagrebu

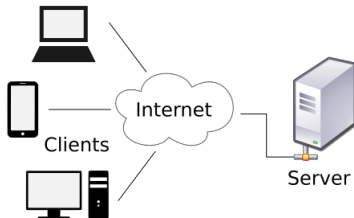
RAČUNARSKI PRAKTIKUM II

Predavanje 08 - Uvod u JavaScript

7. svibnja 2018.

Sastavio: Zvonimir Bujanović





Aktivne web-stranice = skriptiranje na strani klijenta:

- Server pošalje klijentu tražene HTML, CSS i ostale datoteke. Te datoteke sadrže i neki program.
- Web-browser na klijentu prikaže "downloadane" datoteke, te izvršava program unutar web-browsera na klijentskom računalu.
- Zbog interakcije programa s korisnikom, korisnik ima dojam da je web-stranica "aktivna".

Skriptiranje na strani klijenta

Osnovna varijanta:

- Program se u potpunosti izvršava na klijentu.
- Program nakon “downloadanja” nema više nikakvu interakciju sa serverom.

Složenija varijanta:

- Program na osnovu korisničkih akcija (klik/unos teksta) ima daljnju kombinaciju sa serverom.
- Obično se ta komunikacija “skriva” od korisnika (nema ponovnog učitavanja stranice, nego se odvija u pozadini).

Tipični programski jezici za klijentsko skriptiranje:

- **JavaScript**, ActionScript (Adobe Flash), Java applet

Uoči: **svaki klijent** treba imati podršku za programski jezik!

- Svi moderni browseri podržavaju JavaScript *out-of-the-box*.
- Adobe Flash i Javu najčešće treba manualno instalirati.

Programski jezik JavaScript

- Višeplatformski, objektno orijentirani skriptni jezik.
- Jezgra jezika (sintaksa, tipovi, ključne riječi, operatori...) je tzv. **ECMAScript**, standard ECMA-262
 - Aktualna verzija je ECMAScript 2017.
 - Na **ovom** linku možete vidjeti podršku u browserima.
 - Jezik se vrlo brzo razvija i ima **ekstremno puno** biblioteka.
- Jezgra jezika je proširena objektima koji pružaju kontrolu nad:
 - web-browserom – **Browser Object Model (BOM)**,
 - učitanim HTML dokumentom – **Document Object Model (DOM)**.
- Na primjer:
 - Aplikacija može dodavati ili micati elemente na web-stranici, mijenjati njihov stil.
 - Aplikacija može odgovarati na događaje poput klika mišem, unosa podataka u formu ili navigacije unutar web-stranice.

Prva implementacija: Netscape Navigator, 1996. godine.
Dolaskom HTML5, JavaScript doživljava veliku ekspanziju.

Knjige:

- N. Zakas – *Professional JavaScript for Web Developers*, 2012.
- E. Freeman, E. Robson – *Head First HTML5 Programming*, 2011.
- E. Freeman, E. Robson – *Head First JavaScript Programming*, 2014.

Web-resursi:

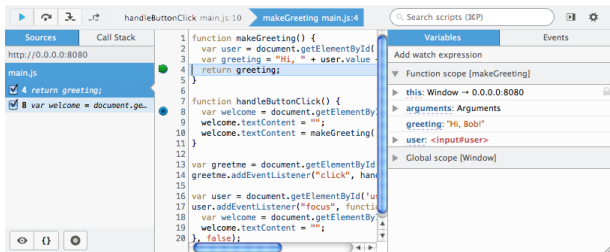
- w3schools – JavaScript Tutorial
- Mozilla Developer Network – JavaScript
- Eloquent JavaScript
- Speaking JavaScript
- HTML Dog – JavaScript

Mozilla Firefox

- Editor: Tools → Web Developer → Scratchpad
- Debugger: Tools → Web Developer → Debugger

Alternative:

- Aptana Studio
- NetBeans
- SublimeText + JavaScript Linter
- Atom + ternjs
- WebStorm
- Visual Studio Code



Uključivanje JavaScript koda u HTML dokumente

Direktno unutar HTML koda:

```
1 <script type="text/javascript">
2 function sayHi() {
3     alert("Hi!");
4 }
5 </script>
```

Link iz HTML dokumenta na eksterni file:

```
1 <script type="text/javascript" src="example.js"></script>
```

- U HTML5 nije nužan atribut `type="text/javascript"`.
- Eksterni file se može nalaziti na drugoj domeni.
- Ranije je bilo uobičajeno stavljati `<script>` unutar `<head>`.
- Novija praksa je stavljati `<script>` na kraj `<body>`, kad se cijela web-stranica već učita i prikaže (no postoje iznimke).
- Kod se izvršava redom kojim je naveden/uključen u HTML-u.

Osnovna sintaksa

Sintaksa vrlo slična C-u:

- case-sensitive
- komentari `//... i /* ... */`
- naredbe završavaju sa `;`
- blokovi naredbi unutar `{...}`

JavaScript je *duck*, *slabo* i *dinamički* tipiziran jezik.

Varijable se deklariraju ključnom riječi `var`:

```
1 var a;  
2 console.log("The value of a is " + a); // "...undefined"  
3 console.log("The value of b is " + b); // ReferenceError exception
```

- Doseg varijable je funkcijski ili globalni (ne postoji blokovski!)
- Varijable mogu mijenjati tip tijekom izvršavanja (nije preporučeno).
- `typeof (var)` vraća string s tipom varijable `var`.

Tipovi podataka

Postoji 6 *primitivnih* tipova podataka:

`undefined`, `null`, `Boolean`, `Number`, `String`, `Symbol`.

Primitivni tipovi se alociraju na stogu.

- `undefined`

- sadrži samo jednu vrijednost, `undefined`
- to je tip deklarirane neinicijalizirane varijable

```
1 var message;  
2 console.log( message == undefined ); // "true"  
3 console.log( typeof( message ) ); // "undefined"
```

- `null`

- sadrži samo jednu vrijednost, `null`
- koristimo ga kad ne postoji smisleni objekt koji bismo pridružili varijabli

```
1 var obj = null;  
2 console.log( typeof( obj ) ); // "object"
```

- Boolean

```
1 var found = true, lost = false;
```

Drugi tipovi se mogu automatski konvertirati u Boolean:

```
1 var hello = "hello";  
2 if( hello ) console.log( "String hello je true" );
```

- Number

Svi brojevi (i cijeli i realni) se reprezentiraju po IEEE-754, tj. kao C-ov double.

```
1 var a = 5, b = 3.14, c = 1.5e2, d = 012, e = 0x1f;
```

Konverzije iz drugih tipova su jednostavne (vidi i `parseInt`):

```
1 var x = true, y = "3.124";  
2 var x_num = Number( x ), y_num = Number( y );
```

• String

- String-konstante mogu biti unutar '...' ili "...".
- Specijalni znakovi: \n, \t, \", \unnnn.
- Pristup znakovima unutar stringa pomoću [...].
- Znakove unutar stringa ne možemo mijenjati (*immutable*).
- Usporedba po abecedi sa <.
- Konkatencija sa +.
- Jedino svojstvo klase String: length.
- Metode klase String: indexOf, match, substr, ...
- Drugi tipovi se u string konvertiraju metodom toString.

```
1 var str = "hello ", x = 3.14;
2 str = str + "world";
3 console.log( str ); // "hello world"
4 console.log( str.length ); // 11
5 console.log( str.substr( 1, 4 ) ); // "ello"
6 str = str + x.toString(); // ili str=str+String(x);
7 console.log( str ); // "hello world3.14"
```

- Object

Najjednostavniji tip koji se čuva na heap-u (*reference type*). Predstavlja kolekciju podataka i funkcija (slično klasi).

```
1 var osoba = new Object(); // osoba je "pointer"!
2 osoba.ime = "Pero"; osoba.starost = 20;
3
4 var x = osoba;
5 x.ime = "Ana"; console.log( osoba.ime ); // "Ana"
```

Često se koristi ovakav ekvivalentan zapis:

```
1 var osoba = {
2     ime: "Pero",
3     starost: 20
4 };
```

Pazi:

```
1 var x = 5, y = new Number( 5 );
2 console.log(typeof(x) + " " + typeof(y)); // "number object"
```

- Funkcije

- Sintaksa relativno slična C/PHP-u:

```
1 function square(number) { return number * number; }
2 var x = square( 5 ); // x = 25
3 var f = square;
4 console.log( f(3) ); // 9
```

- Nema overloadanja.
- Vrlo se često koriste anonimne funkcije (lambda izrazi):

```
1 var f = function(num) { return num * num; };
2 var kvadrat = ( x => { return x*x; } );
3 var zbroj = ( (x, y) => x+y ); // "arrow function"
```

- Funkcije mogu primiti po volji mnogo parametara → polje arguments

```
1 function f() { console.log( arguments[1] ); }
2 f( 3, "nesto", 123 ); // "nesto"
```

- Argumenti se prenose po vrijednosti (pazi, Object je "pointer").

- Array

- Niz podataka ne nužno istog tipa!
- Pristup elementima sa [...].
- Indexi su nenegativni cijeli brojevi.
- Automasko produljivanje/skraćivanje po potrebi.
- Svojstvo `length`; nije read-only(!)
- Metode: `push`, `pop`, `reverse`, `sort`, `indexOf`, `map`, `forEach` ...

```
1  var x = new Array();
2  x.push( 7 ); x.push( 3 ); x[2] = 5; // x = [7, 3, 5]
3  x.sort( function(a, b) {
4      return (a<b) ? -1 : ((a>b) ? 1 : 0);
5  } );
6  console.log( x.toString() ); // "3,5,7"
7  x.length = 5; // x = [3, 5, 7, undefined, undefined]
8
9  var y = ["red", "green", "blue"];
```

Operatori, kontrola toka, petlje

- Operatori kao u C-u; iznimka `==` i `===`.
- `a==b` radi konverziju ako `a` i `b` nisu istog tipa.
- `a===b` vraća `false` ako `a` i `b` nisu istog tipa.
- Slično `!=` i `!==`.
- `if`, `switch`, `while`, `do...while`, `for`, `break`, `continue`, `throw/try/catch` identični kao u C-u.
- `for ... in`
Iteracija po svim svojstvima objekta.

```
1  var x = {  
2      ime: "Pero",  
3      starost: 20  
4  };  
5  for( var i in x )  
6      console.log( i + "->" + x[i] );  
7  // "ime->Pero", "starost->20"
```

Zadatak 1

Napišite program koji na web-stranicu ispisuje prvih 100 prirodnih brojeva.

Ispišite te brojeve i u konzolu.

Što se dogodi ako probamo ispisati prvih 100000 brojeva?

Upute:

- Poziv `document.write(str)`; će na to mjestu u HTML-u ispisati string `str`.
- Poziv `console.log(str)`; ispisuje string `str` u konzolu. Ovo je korisno za debugiranje.
- Ako `x` nije string, onda `document.write(x)`; automatski poziva `x.toString()` prije ispisa.

Zadatak 2

Napišite program koji na web-stranicu ispisuje tablicu množenja za prvih N prirodnih brojeva. Broj N učitajte pomoću "skočnog okvira" (engl. *popup*).

Uputa:

- Postoje 3 vrste "skočnog okvira":
 - `alert("Poruka");`
Okvir u kojem piše "Poruka". Jedino možemo kliknuti na OK.
 - `var ret = prompt("Što želiš?");`
Okvir u kojem piše "Što želiš?". String koji utipkamo će biti spremljen u varijablu `ret`.
 - `var ret = confirm("Da ili ne?");`
Okvir u kojem piše "Da ili ne?". Možemo kliknuti na "OK" ili "Cancel"; boolean će biti spremljen u varijablu `ret`.

Pomoću ugrađenog objekta `Date`, dohvatite trenutno vrijeme (na kljetnovom računalu!). Ispišite trenutni datum (s mjesecima na hrvatskom!) i trenutno vrijeme, te:

- Ako je trenutno manje od 12h, ispišite poruku "Dobro jutro!".
- Ako je između 12h i 18h, ispišite "Dobar dan!".
- Ako je više od 18h, ispišite "Dobra večer!".

Uputa:

- `var t = new Date();`
U varijablu `t` sprema trenutno vrijeme. Nakon toga je dostupan cijeli niz funkcija poput `t.getDate()`, `t.getMinutes()`.

Koristeći “skočni okvir”, učitajte neki string, te ispišite sve njegove podstringove.

Na primjer, za učitani string “abc”, treba ispisati:

- “”, “a”, “b”, “c”, “ab”, “ac”, “bc”, “abc”

(ne nužno tim redom).

- U JavaScriptu, regularne izraze možemo deklarirati na dva ekvivalentna načina:
 - `var re = /ab+c/;`
 - `var re = new RegExp("ab+c");`
- U klasi `RegExp` postoje metode:
 - `exec`, `test`, `match`, `search`, `replace`, `split`.

```
1 var str = "110010001";
2 var re = /^(0|(1[01]*))$/;
3
4 if( re.test( str ) )
5     document.write( str + " je binarni broj " );
6 else
7     document.write( str + " nije binarni broj " );
```

Koristeći "skočni okvir", učitavajte datum sve dok se ispravno ne unese u formatu `dd/mm/yyyy`. Nakon toga, ispišite koji je to dan u tjednu.

Uputa:

- Dan u tjednu možete dohvatiti pomoću metode `getDay()` klase `Date`. Datum koji odgovara učitanoj možete napraviti ovako (oprez, za siječanj je `month=0!`):

```
var d = new Date( year, month, day );
```
- Dan, mjesec i godinu iz učitanoj stringa možete dohvatiti i pomoću metode `exec` klase `RegExp`.

Objektno orijentirano programiranje

Objekti mogu imati i svojstva i metode kao članove:

```
1 var osoba = {  
2     ime: "Pero",  
3     starost: 20,  
4  
5     kaziBok: function() { console.log("Bok!"); }  
6 };  
7 osoba.prezime = "Perić"; osoba.kaziBok();
```

Kako napraviti puno objekata istog tipa?

JavaScript:

- Tek u ECMAScript 2015 su uvedeni klasični elementi OOP poput klasa, konstruktora, destuktora, nasljeđivanja i slično.
- JavaScript koristi tzv. OOP baziran na **prototipima**.
- OOP elementi se ostvaruju procesom "dekoriranja", odnosno proširivanja postojećih objekata koji služe kao prototipi.
- Ovo je tzv. *bezklasno* ili *prototype-oriented* programiranje.

Objektno orijentirano programiranje: Constructor pattern

Stvaranje puno objekata istog tipa → **Constructor** pattern:

```
1 function Osoba( ime, starost ) {  
2     // this se odnosi na objekt koji se konstruira sa new.  
3     this.ime = ime;  
4     this.starost = starost;  
5     this.kaziBok = kaziBok;  
6 }  
7  
8 function kaziBok() { console.log( "Bok!" ); }  
9  
10 var pero = new Osoba( "Pero", 20 );  
11 var ana = new Osoba( "Ana", 19 );  
12  
13 console.log( pero instanceof Osoba ); // "true"  
14 console.log( ana instanceof Object ); // "true"  
15 console.log( pero.kaziBok === ana.kaziBok ); // "true"
```

Problem ovog pristupa je "zagađivanje" koda globalnim funkcijama.

Objektno orijentirano programiranje: prototype

Svaka funkcija u JavaScriptu ima svojstvo `prototype`.
Ono propisuje kako se kreira objekt kad se pozove `new` na toj funkciji.

```
1 function Osoba() {}  
2  
3 Osoba.prototype.ime = "Pero";  
4 Osoba.prototype.starost = 20;  
5 Osoba.prototype.kaziBok = function() { console.log("Bok!"); };  
6  
7 var pero = new Osoba(), ana = new Osoba();  
8 ana.ime = "Ana";  
9  
10 console.log( [pero.ime, ana.ime] ); // "Pero, Ana"  
11 console.log( pero instanceof Osoba ); // "true"  
12 console.log( ana instanceof Object ); // "true"  
13 console.log( ana.kaziBok === pero.kaziBok ); // true
```

Svojstvo `prototype` sadrži podsvojstvo `constructor`.
Po defaultu, `prototype.constructor` sadrži "pointer" na samu funkciju (`Osoba`).

Objektno orijentirano programiranje: Constructor + prototype

Rješenje problema "zagađivanja" globalnog dosega daje kombinacija Constructor + prototype.

```
1 function Osoba(ime, starost) {  
2     this.ime = ime;  
3     this.starost = starost;  
4 }  
5  
6 Osoba.prototype = {  
7     constructor: Osoba, // treba jer {} kreira NOVI objekt  
8     kaziBok: function() { console.log( "Bok!" ); }  
9 };  
10  
11 var pero = new Osoba( "Pero", 20 ),  
12     ana  = new Osoba( "Ana", 19 );  
13  
14 console.log( [pero.ime, ana.ime] ); // "Pero, Ana"  
15 console.log( pero instanceof Osoba ); // "true"  
16 console.log( ana instanceof Object ); // "true"  
17 console.log( ana.kaziBok === pero.kaziBok ); // true
```

Objektno orjentirano programiranje: dinamički prototip

Alternativno, konstruktor možemo napisati ovako:

```
1  function Osoba(ime, starost) {
2      this.ime = ime;
3      this.starost = starost;
4
5      if( typeof(this.kaziBok) !== "function" )
6          Osoba.prototype.kaziBok = function() {
7              console.log( "Bok!" );
8          };
9  }
10
11  var pero = new Osoba( "Pero", 20 ),
12      ana  = new Osoba( "Ana", 19 );
13
14  console.log( [pero.ime, ana.ime] ); // "Pero, Ana"
15  console.log( pero instanceof Osoba ); // "true"
16  console.log( ana instanceof Object ); // "true"
17  console.log( ana.kaziBok === pero.kaziBok ); // true
18
19  pero.kaziBok();
```

Objektno orijentirano programiranje: nasljeđivanje

Donji kod implementira tzv. *parazitsko kombinirano* nasljeđivanje. Za detalje o OOP u JavaScriptu vidi i [Mozilla Developer Network](#).

```
1 // .. definiramo tip Osoba kao prije
2 function Student( ime, starost, JMBAG ) {
3     Osoba.call( this, ime, starost ); // konstr. bazne klase
4     this.JMBAG = JMBAG;              // dodaj novo svojstvo
5 }
6
7 Student.prototype = Object.create(Osoba.prototype);
8 Student.prototype.constructor = Student;
9
10 Student.prototype.dajJMBAG = function() {
11     return this.JMBAG; // nova metoda u klasi Student
12 }
13
14 var pero = new Student( "Pero", 20, "12345" );
15 pero.kaziBok();
16 console.log( pero.dajJMBAG() );
```

Klasičnija notacija OOP u JavaScriptu

ECMAScript 2015 (FF 45+, CH 42+) uvodi **standardniju OOP notaciju**.
Ovo je samo *syntactic sugar* za OOP baziran na prototipovima!

```
1 class Osoba {
2     constructor(ime) { this.ime = ime; }
3     kaziBok() { console.log( "Bok!" ); }
4 }
5
6 class Student extends Osoba {
7     constructor(ime, JMBAG) { super(ime); this.JMBAG = JMBAG; }
8     dajJMBAG() { return this.JMBAG; }
9 }
10
11 var pero = new Student("Pero", "12345"), ana = new Osoba("Ana");
12 pero.kaziBok(); console.log( pero.dajJMBAG() );
13
14 console.log( pero instanceof Object ); // "true"
15 console.log( pero instanceof Osoba ); // "true"
16 console.log( pero instanceof Student ); // "true"
17 console.log( ana instanceof Student ); // "false"
```

- Napravite hijerarhiju klasa `Cetverokut`, `Pravokutnik`, `Kvadrat`.
- Za svaki lik može se izračunati opseg.
- Za pravokutnike i kvadrate može se izračunati površina.
- Napravite i konverziju svakog od likova u string tako da predefinirate metodu `toString`:

```
1 Kvadrat.prototype.toString = function()
2 {
3     return "Kvadrat sa stranicom " + this.a;
4 }
5
6 // Sada (uz definiciju konstruktora) radi:
7 var K = new Kvadrat( 5 );
8 console.log( "K je " + K );
```

Closure

- **Closure** je funkcija koja koristi "nezavisnu" varijablu (onu koja nije definirana unutar te funkcije).
- U donjem primjeru, funkcije `povecaj`, `smanji` i `getX` su *closure*.
- Ovakvi trikovi se često koriste u JS; treba biti oprezan jer mogu imati neobične nuspojave.

```
1 var brojac = function() {  
2     var x = 0;  
3  
4     return {  
5         povecaj: function() { ++x; },  
6         smanji: function() { --x; },  
7         getX: function() { return x; }  
8     };  
9 }  
10  
11 var A = brojac(), B = brojac();  
12 console.log( A.getX() + " " + B.getX() ); // 0 0  
13 A.povecaj(); A.povecaj(); B.smanji();  
14 console.log( A.getX() + " " + B.getX() ); // 2 -1
```