



Интернет програмирање предавање 09

Проф. др Мирослав Лутовац
mlutovac@viser.edu.rs



JavaScript JSON

- JSON je format za memorisanje i prenos podataka
- JSON se često koristi kada se podaci šalju sa servera na veb stranu
- JSON je skraćenica od **J**ava**S**cript **O**bject **N**otation
- JSON lagani format za razmenu podataka
- JSON je jezički nezavisan
 - Sintaksa je izvedena iz JS objektne notacije,
 - ali to je samo tekst
 - Kod za čitanje i generisanje JSON podataka može da bude napisan u bilo kom programskom jeziku
- JSON je jednostavan razumljiv sam po sebi



JavaScript JSON

- Primer JSON sintakse koja definiše objekat zaposleni: niz kao zapis 3 zaposlena (objekta)

```
{
```

JSON primer

```
"zaposleni": [
```

```
  {"ime": "Marko", "prezime": "Jokic"},
```

```
  {"ime": "Ana", "prezime": "Jovin"},
```

```
  {"ime": "Milan", "prezime": "Kokov"}],
```

```
]
```

```
}
```



JavaScript JSON

- JSON format evaluiira u JavaScript objektima
- JSON format sintaksički je identičan kodu za kreiranje JS objekta
- Zbog ove sličnosti, JS program može jednostavno da konvertuje JSON podatke u obične JS objekte
- JSON sintaksna pravila
 - Podaci su parovi ime/vrednost
 - Podaci su razdvojeni zarezima
 - Vitičaste zagrade ograđuju objekte
 - Uglaste zagrade ograđuju nizove

JSON sintaksa



JavaScript JSON

- JSON se pišu kao parovi ime/vrednost, isto kao JavaScript object osobine
- ime/vrednost par se sastoji od imenovanih polja (pod dvostrukim navodnicima), iza kojih sledi dvotačka, pa zatim vrednost
- JSON imena zahtevaju dvostruke navodnike, JS imena ne zahtevaju

`"ime" : "Marko"`

JSON podaci kao
parovi ime i vrednost



JavaScript JSON

- JSON se pišu unutar vitičastih zagrada
- Kao i kod JS, objekti mogu da sadrže višestruke parove ime/vrednost

```
{ "ime" : "Marko" , "prezime" : "Jokic" }
```

JSON objekti



JavaScript JSON

- JSON nizovi se pišu u okviru uglastih zagrada
- kao i kod JS, niz može da sadrži objekte

```
"zaposleni": [
```

```
  { "ime": "Marko", "prezime": "Jokic" },
```

```
  { "ime": "Ana", "prezime": "Jovin" },
```

```
  { "ime": "Milan", "prezime": "Kokov" } ],
```

```
]
```

JSON nizovi

- ✓ objekat "zaposleni" je niz; sadrži 3 objekta
- ✓ svaki objekat je zapis osobe (ime i prezime)



JavaScript JSON

- Uobičajeno se JSON koristi da se pročitaju podaci sa veb servera, i ti podaci prikažu na veb strani
- Neka su pročitani podaci u vidu stringa kao ulaz
- Prvo se kreira JS string koji sadrži JSON sintaksu

```
var text = '{ "zaposleni" : [' +  
  '{ "ime":"Marko" , "prezime":"Jokic" },' +  
  '{ "ime":"Ana" , "prezime":"Jovin" },' +  
  '{ "ime":"Milan" , "prezime":"Kokov" } ]}';
```

1. Konverzija JSON teksta u JS objekat



JavaScript JSON

- Upotrebljava se JS ugrađena funkcija **JSON.parse()** da se konvertuje string u JS objekat

```
var obj = JSON.parse(text) ;
```

2. Konverzija JSON teksta u JS objekat



JavaScript JSON

- Upotrebljava se novi JS objekat na veb strani

```
<p id="demo"></p>
```

```
<script>
```

```
document.getElementById("demo").innerHTML =
```

```
obj.employees[1].firstName + " " + obj.employees[1].lastName;
```

```
</script>
```

3. Konverzija JSON teksta u JS objekat

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <h4>Pravljenje objekta u JS iz JSON
5 <p id="demo"></p>
6 <script>
7 var tekst = '{"zaposleni":[' +
8 '{"ime":"Marko","prezime":"Jokic" },
9 '{"ime":"Ana","prezime":"Jovin" },' +
10 '{"ime":"Milan","prezime":"Kokov" ]}]';
11 objekt = JSON.parse(tekst);
12 var ukupno = objekt.zaposleni.length;
13 var zaPrikazTekst = "Novih zaposlenih ima " +
14 ukupno + " .To su:<br>";
15 for (i = 0; i < ukupno; i++) {
16 zaPrikazTekst += objekt.zaposleni[i].ime + " " +
17 objekt.zaposleni[i].prezime + "<br>";
18 }
19 document.getElementById("demo").innerHTML = zaPrikazTekst;
20 </script>
21 </body>
22 </html>
```

Pravljenje objekta u JS iz JSON stringa

Novih zaposlenih ima 3 .To su:

Marko Jokic

Ana Jovin

Milan Kokov



Ulazni tekst pretvoren u niz-objekat,
for petlja za pravljenje stringa za prikaz



JavaScript objekti

- U JS sve su objekti osim primitivnih vrednosti
 - Logičke vrednosti ako su definisani sa new
 - Brojevi ako su definisani sa new
 - Stringovi ako su definisani sa new
 - Datumi i vremena
 - Mats
 - Regularni izrazi
 - Nizovi
 - Funkcije
 - Objekti



JavaScript objekti

- **JS primitivne vrednosti**

- Primitivne vrednosti su vrednosti koje nemaju osobine ili metode
- Primitivne vrste podataka su one koje imaju primitivne vrednosti (ne mogu se promeniti, uvek su to što jesu)
 - ✓ string ("Zdravo" je uvek "Zdravo")
 - ✓ broj (3.14 je uvek 3.14)
 - ✓ logička vrednost (true je uvek true, false je uvek false)
 - ✓ null (null je uvek null)
 - ✓ undefined (undefined je uvek undefined)



JavaScript objekti

- U JS promenljive su objekti koji sadrže promenljive
- Objekti mogu sadržati više promenljivih vrednosti se zapisuju kao parovi razdvojeni dvotačkom :
imePromenljive : vrednostPromenljive
- JS objekat je kolekcija imenovanih vrednosti

```
var osobaBroj01 = "Marko Markov";
```

```
var osoba = {ime:"Marko", prezime:"Markov",  
             starost:50, bojaOciju:"plava"};
```



JavaScript objekti

- Osobine objekata
- Imenovane vrednosti u JS se nazivaju osobine

Osobina	vrednost
----------------	-----------------

ime	Mark
-----	------

prezime	Markov
---------	--------

starost	50
---------	----

bojaOciju	plava
-----------	-------



JavaScript objekti

- Objekti se zapisuju kao parovi ime i vrednost slično kao u drugim programskim jezicima
 - asocijativni niz u PHP
 - rečnik u Python
 - heš tabele u C
 - heš mape u Java
 - heševi Ruby i Perl



JavaScript objekti

- **Metode objekata**
- Metode su aktivnosti koje mogu da se izvrše na objektima
- Osobine objekata mogu da budu primitivne vrednosti, objekti ili funkcije
- Metoda objekta je osobina objekta koja sadrži definiciju funkcije
- JS objekti su kontejneri za imenovane vrednosti i pozivane osobine ili metode



JavaScript objekti

- Primer osobine i metoda

Osobina

vrednost

ime	Mark
prezime	Markov
starost	50
bojaOciju	plava
imeFunkcije	<code>function() {return this.ime + " " + this.prezime;}</code>



JavaScript objekti

- **Kreiranje JS objekta**
- U JS može da se definiše i kreira sopstveni objekat
 - Definiše i kreira jedan objekat, korišćenjem literalnih objekata
 - Definiše i kreira jedan objekt, koristeći new
 - Definiše konstruktor objekat, a zatim kreiraju objekti konstruktor tipa



JavaScript objekti

- **Kreiranje objekta korišćenjem literalnih objekata**
- Ovo je najjednostavniji način kreiranja JS objekta
- Na ovaj način, definiše se i kreira se objekat kao jedna naredba
- Literalni objekat je lista parova ime:vrednost u okviru vitičastih zagrada (primer sa 4 osobine)

```
var osoba = {ime:"Marko", prezime:"Markov",  
             starost:50, bojaOciiju:"plava"};
```



JavaScript objekti

- **Kreiranje objekta korišćenjem literalnih objekata**
- Praznine i prelazak u novi red nemaju uticaj
- Može biti u više linija

```
var osoba = {  
    ime: "Marko",  
    prezime: "Markov",  
    starost: 50,  
    bojaOciju: "plava"  
};
```



JavaScript objekti

- **Kreiranje objekta korišćenjem new**
- Primer novog JS objekta sa 4 osobine

```
var osoba = new Object();  
  
osoba.ime = "Marko";  
  
osoba.prezime = "Markov";  
  
osoba.starost = 50;  
  
osoba.bojaOciju = "plava";
```



JavaScript objekti

- **Mutiranje JS objekata**
- Objekti mutiraju
- Objektima se pristupa po referenci, a ne po vrednosti
- Ako je objekat **osoba**, dodela drugoj promenljivoj neće napraviti kopiju objekta **osoba**

```
var x = osoba;
```

- Oba objekta (**x** i **osoba**) ukazuju na isti objekat
- Promena u objektu **x** dovodi do promene i u objektu **osoba**

JS promenljive ne mutiraju
Samo JS objekti mutiraju



Pristup osobinama objekata

- Može se pristupiti osobinama na više načina:

`imeObjekta.osobinaObjekta`

`imeObjekta["osobinaObjekta"]`

`x="osobinaObjekta"; imeObjekta[x]`

`osoba.ime`

`osoba.["ime"]`

`x="ime"; osoba[x]`

D:\aLutovac\Internet programiranje\primeri JS\pJSubject01.html - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins

pJSubject01.html

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p>pristup osobinama objekta</p>
5 <p> <b>imeObjekta.osobina</b> </p>
6 <p id="demo1"></p>
7 <p> isto se dobija i sa
8 <b>imeObjekta["osobina"]</b></p>
9 <p id="demo2"></p>
10 <script>
11 var osoba = {
12     ime:"Marko", prezime:"Markov",
13     starost:56, bojaOciju:"plava"
14 };
15 document.getElementById("demo1").innerHTML =
16 "<b>" + osoba.ime + "</b> ima <b>" +
17 osoba.starost + "</b> godina.";
18 document.getElementById("demo2").innerHTML =
19 "<b>" + osoba["ime"] + "</b> ima <b>" +
20 osoba["starost"] + "</b> godina.";
21 </script>
22 </body>
23 </html>
```

JavaScript Properties x pJSubject01.html x

file:///D:/aLutovac/Internet%20programiranje/pri...

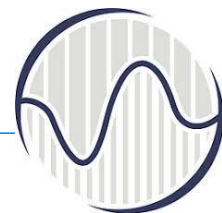
pristup osobinama objekta

imeObjekta.osobina

Marko ima **56** godina.

isto se dobija i sa **imeObjekta["osobina"]**

Marko ima **56** godina.



Dva pristupa osobinama objekta

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p id="demo"></p>
5 <script>
6   var tekst = "";
7   var osoba = {
8     ime:"Marko", prezime:"Markov",
9     starost:56, bojaOciiju:"plava"
10  };
11   var i;
12   for (i in osoba) {
13     tekst += osoba[i] + " ";
14   }
15   document.getElementById("demo")
16 </script>
17 </body>
18 </html>
```

Marko Markov 56 plava

JS objekat u petlji



Blok koda u petlji
izvršava se za svaku
osobinu objekta po
jednom



Pristup osobinama objekata

- Dodavanje osobine je moguće ako već postoji objekat, tako što se dodeljuje nova vrednost novoj osobini objekta
- Ne mogu se koristiti rezervisana imena, za osobine ili metode
- Brisanjem se briše i osobina i vrednost osobine
- Nakon brisanje, ne može se koristiti
- Brisanje ne važi za promenljive i funkcije

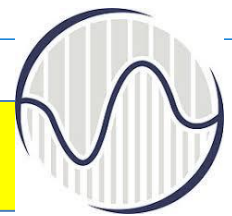
```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p id="demo"></p>
5 <script>
6 var tekst = "";
7 var osoba = {
8     ime:"Marko", prezime:"Markov"
9     starost:56, bojaOciju:"plava"
10 };
11 osoba.pismo = "latinica";
12 delete osoba.starost;
13 var i;
14 for (i in osoba) {
15     tekst += osoba[i] + " ";
16 }
17 document.getElementById("demo").innerHTML = tekst;
18 </script>
19 </body>
20 </html>
21
```

Marko Markov56plava

Marko Markov plavalatinica

JS dodavanje osobine

JS brisanje osobine





Atributi osobina objekata

- Sve osobine imaju ime, i njima se dodeljuju vrednosti
- Vrednost je jedan od attribute osobina
- Ostali atributi su :
enumerable, configurable, writable
- Ovi atributi određuju kako se osobini može pristupiti (čitljiva, može da se upiše)
- u JS, svi atributi mogu da se čitaju, ali samo vrednost može da se menja (ako je writable)



JavaScript metode

- JS metode su akcije koje se mogu izvršiti na objektima
- JS metoda je osobina koja sadrži definiciju funkcije
- Deo koji se zove **this**, je objekat koji *poseduje* JS kod
- Vrednost od **this**, u funkciji, je objekat koji *poseduje* funkciju
- **this** je ključna reč. Ne može se promeniti njena vrednost

Osobina	vrednost
ime	Mark
prezime	Markov
starost	50
bojaOciju	plava
imeFunkcije	function() {return this .ime + " " + this .prezime;}



Pristup metodama objekata

- imeObjekta.imeMetode ()

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p>metoda je funkcija
5 kao vrednost osobine</p>
6 <p id="demo"></p>
7 <script>
8 var osoba = {
9     ime: "Milan",
10    prezime : "Markovic",
11    id : 5678,
12    starost:56, bojaOciiju:"plava",
13    imePrezime : function() {
14        return this.ime + " " + this.prezime;
15    }
16 };
17 document.getElementById("demo").innerHTML = osoba.imePrezime();
18 </script>
19 </body>
20 </html>
```

metoda je funkcija kao vrednost osobine

Milan Markovic



sa ()

metoda je funkcija kao vrednost osobine

```
function () { return this.ime + " " + this.prezime; }
```

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p>metoda je funkcija
5 kao vrednost osobine</p>
6 <p id="demo"></p>
7 <script>
8 var osoba = {
9     ime: "Milan",
10    prezime : "Markovic",
11    id : 5678,
12    starost:56, bojaOciju:"plava",
13    imePrezime : function() {
14        return this.ime + " " + this.prezime;
15    }
16 };
17 document.getElementById("demo").innerHTML = osoba.imePrezime;
18 </script>
19 </body>
20 </html>
```



bez ()



Korišćenje ugrađenih metoda

- toUpperCase() metoda za objekte tipa string, da se konvertuje tekst u velika slova

```
var message = „Zdravo svima!";
```

```
var x = message.toUpperCase();
```

- za x se dobija

ZDRAVO SVIMA!



Dodavanje nove metode

- toUpperCase() metoda za objekte tipa string, da se konvertuje tekst u velika slova

```
osoba.imePrezime = function () {  
    return this.ime + " " + this.prezime;  
};
```

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p>metoda je funkcija
5 kao vrednost osobine</p>
6 <p id="demo"></p>
7 <script>
8 var osoba = {
9     ime: "Milan",
10    prezime : "Markovic",
11    id : 5678,
12    starost: 56,
13 };
14 osoba.imePrezime = function() {
15     return this.ime + " " + this.prezime;
16 };
17 document.getElementById("demo").innerHTML = osoba.imePrezime();
18 </script>
19 </body>
20 </html>
```

metoda je funkcija kao vrednost osobine

Milan Markovic

umesto : sada je =





Konstruktor

- Standardni način da se napravi objektni tip jeste da se koristi konstruktor funkcije
- Ovo je potrebno da bi se kreirao veći broj objekata istog tipa

```
osoba.imePrezime = function () {  
    return this.ime + " " + this.prezime;  
};
```

D:\aLutovac\Internet pro

JavaScript Constructors x pJSubject08.html x

File Edit Search View E

file:///D:/aLutovac/Internet%20programiranje/primeri%20JS/pJSubject08.html

pJSubject08.html x

1 <!DOCTYPE

2 <html>

3 <body>

4 <p id="demo"></p>

5 <script>

6 function osoba(ime1, last, godina, oke) {

7 this.ime = ime1;

8 this.prezime = last;

9 this.god = godina;

10 this.bojaOciju = oke;

11 }

12 var mojOtac = new osoba("Marko", "Markov", 51, "plave");

13 var mojaMajka = new osoba("Ana", "Lukic", 43, "zelene");

14 document.getElementById("demo").innerHTML =

15 "Moj otac ima " + mojOtac.god +

16 ". Moja majka ima " + mojaMajka.god + ".";

17 </script>

18 </body>

19 </html>

Moj otac ima 51. Moja majka ima 43.



<

Hyper Text Markup Language file

length : 476 lines : 20

Ln : 20 Col : 1 Sel : 0 | 0

38



Ugrađeni konstruktori

```
var x1 = new Object();    // A new Object object
var x2 = new String();    // A new String object
var x3 = new Number();    // A new Number object
var x4 = new Boolean();   // A new Boolean object
var x5 = new Array();     // A new Array object
var x6 = new RegExp();    // A new RegExp object
var x7 = new Function();  // A new Function object
var x8 = new Date();      // A new Date object
```



pJSubject09.html

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p id="demo"></p>
5 <script>
6 var x1 = new Object(); // A new Object object
7 var x2 = new String(); // A new String object
8 var x3 = new Number(); // A new Number object
9 var x4 = new Boolean(); // A new Boolean object
10 var x5 = new Array(); // A new Array object
11 var x6 = new RegExp(); // A new RegExp object
12 var x7 = new Function(); // A new Function object
13 var x8 = new Date(); // A new Date object
14 document.getElementById("demo").innerHTML =
15 "x1: " + typeof x1 + "<br>" +
16 "x2: " + typeof x2 + "<br>" +
17 "x3: " + typeof x3 + "<br>" +
18 "x4: " + typeof x4 + "<br>" +
19 "x5: " + typeof x5 + "<br>" +
20 "x6: " + typeof x6 + "<br>" +
21 "x7: " + typeof x7 + "<br>" +
22 "x8: " + typeof x8 + "<br>";
23 </script>
24 </body>
25 </html>
```

pJSubject09.html

JavaScript C

file:///D:/aLutovac/Inter

x1: object
x2: object
x3: object
x4: object
x5: object
x6: object
x7: function
x8: object





Konstruktor

- Math je globalni objekt
- Reč **new** ne može da se koristi uz **Math**
- JS ima objektne verzije primitivnih String, Number i Boolean
- Nema potrebe kreirati složene objekte; primitivne vrednosti se izvršavaju brže
- Nema potrebe koristiti new Array(). Bolje je []
- Nema potrebe koristiti new RegExp()
- Nema potrebe koristiti new Function()
- Nema potrebe koristiti new Object()



Umesto ugrađenih konstruktora

```
var x1 = {};           // new object
var x2 = "";           // new primitive string
var x3 = 0;            // new primitive number
var x4 = false;        // new primitive boolean
var x5 = [];           // new array object
var x6 = /()/          // new regexp object
var x7 = function(){}; // new function object
```

**Jednostavnije je;
ne koristiti new**



Prototip objekata

- Svaki JS objekat ima prototip
- Prototip je takođe objekat
- Svi JS objekti nasleđuju svoje osobine i metode od prototipova
- Object.prototype je na vrhu lanca prototipova
- Svi JS objekti (Date, Array, RegExp, Function,) nasleđuju od Object.prototype



Kreiranje prototipa

- Standardni način kreiranja prototipa objekta jeste da se koristi konstruktor funkcija

```
function Person(first, last, age, eyecolor)
{
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eyecolor;
}
```



Kreiranje prototipa

- Sa konstruktor funkcijom, koristi se new da se kreira novi objekat iz istog prototipa

```
var myFather = new Person("John", "Doe", 50,  
"blue");
```

```
var myMother = new Person("Sally", "Rally",  
48, "green");
```

Dobra je praksa da se imenuje konstruktor funkcija sa velikim početnim slovom



Dodavanje osobine i metode objektu

- Dodavanje osobine i metode
 - Već postojećem objektu
 - Svim objektima određenog tipa
 - Prototipu objekta
- Dodela vrednosti je najjednostavniji način dodele osobine

```
var myFather = new Person("John", "Doe", 50, "blue");
```



```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p id="demo"></p>
5 <script>
6 function osoba(ime1, last, godina, oke) {
7     this.ime = ime1;
8     this.prezime = last;
9     this.god = godina;
10    this.bojaOciju = oke;
11 }
12 var mojOtac = new osoba("Marko", "Markov", 51, "plave");
13 var mojaMajka = new osoba("Ana", "Lukic", 43, "zelene");
14 mojOtac.nacionalnost = "Srbin";
15
16 document.getElementById("demo").innerHTML =
17 "Moj otac je " + mojOtac.nacionalnost + ".";
18 </script>
19 </body>
20 </html>
```

Moj otac je Srbin.



```
1 <!DOCTYPE html>
2 <html>
3 <body>
4   <p id="demo"></p>
5   <script>
6     function osoba(ime1, last, godina, oke) {
7       this.ime = ime1;
8       this.prezime = last;
9       this.god = godina;
10      this.bojaOciiju = oke;
11    }
12    var mojOtac = new osoba("Marko", "Markov", 51, "plave");
13    var mojaMajka = new osoba("Ana", "Lukic", 43, "zelene");
14    mojOtac.nacionalnost = "Srbin";
15
16    document.getElementById("demo").innerHTML =
17    "Moja majka je " + mojaMajka.nacionalnost + ".";
18  </script>
19 </body>
20 </html>
```

Moja majka je undefined.





Dodavanje metode objektu

- Dodela vrednosti je najjednostavniji način dodele osobine

```
myFather.name = function () {  
    return this.firstName + " " + this.lastName;  
};
```

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4   <p id="demo"></p>
5   <script>
6     function osoba(ime1, last, godina, oke) {
7       this.ime = ime1;
8       this.prezime = last;
9       this.god = godina;
10      this.bojaOciiju = oke;
11    }
12    var mojOtac = new osoba("Marko", "Markov", 51, "plave");
13    var mojaMajka = new osoba("Ana", "Lukic", 43, "zelene");
14    mojOtac.imePrezime = function() {
15      return this.ime + " " + this.prezime;
16    };
17    document.getElementById("demo").innerHTML =
18    "Moj otac se zove " + mojOtac.imePrezime() + ".";
19  </script>
20 </body>
21 </html>
```

Moj otac se zove Marko Markov.



```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p id="demo"></p>
5 <script>
6 function osoba(imel, last, godina, oke) {
7     this.ime = imel;
8     this.prezime = last;
9     this.god = godina;
10    this.bojaOciiju = oke;
11 }
12 var mojOtac = new osoba("Marko", "Markov", 51, "plave"),
13 var mojaMajka = new osoba("Ana", "Lukic", 43, "zelene");
14 mojOtac.imePrezime = function() {
15     return this.ime + " " + this.prezime;
16 };
17 document.getElementById("demo").innerHTML =
18 "Moj otac se zove " + mojOtac.imePrezime + ".";
19 </script>
20 </body>
21 </html>
```

Moj otac se zove function () { return
this.ime + " " + this.prezime; }.





Dodavanje osobine objektu

- Ne može se dodati osobina prototipu kao objektu, zato što prototip nije objekat, dobiće se greška ("undefined")
- Nova osobina se dodaje prototipu preko konstruktora

```
Person.nationality = "English";
```

```
function Person(first, last, age, eyecolor) {  
    this.firstName = first;  
    this.lastName = last;  
    this.age = age;  
    this.eyeColor = eyecolor;  
    this.nationality = "English";  
}
```

Prototip osobina može imati prototip vrednost kao predefinisanu vrednost (nije argument funkcije)

Moj otac je Srbin, a moja majka je Srbin.

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p id="demo"></p>
5 <script>
6 function osoba(imel, last, godina, oke) {
7     this.ime = imel;
8     this.prezime = last;
9     this.god = godina;
10    this.bojaOciiju = oke;
11    this.nacionalnost = "Srbin";
12 }
13 var mojOtac = new osoba("Marko", "Markov", 51, "plave");
14 var mojaMajka = new osoba("Ana", "Lukic", 43, "zelene");
15
16 document.getElementById("demo").innerHTML =
17 "Moj otac je " + mojOtac.nacionalnost +
18 ", a moja majka je " + mojaMajka.nacionalnost + ".";
19 </script>
20 </body>
21 </html>
22
```



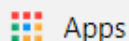


Dodavanje metode objektu

- Ne može se dodati metoda prototipu kao objektu, zato što prototip nije objekat, dobiće se greška
- Nova metoda se dodaje prototipu preko konstruktora

```
function Person(first, last, age, eyecolor) {  
    this.firstName = first;  
    this.lastName = last;  
    this.age = age;  
    this.eyeColor = eyecolor;  
    this.name = function() {return  
+ this.lastName;};  
}
```

Prototip može imati
prototip vrednost kao
predefinisanu vrednost
(nije argument funkcije)



Moj otac se zove Marko Markov, a majka se zove Ana Lukic.

```
1 <!DOCTYPE
2 <html>
3 <body>
4 <p id="demo">
5 <script>
6 function osoba(ime1, last, godina, oke) {
7     this.ime = ime1;
8     this.prezime = last;
9     this.god = godina;
10    this.bojaOciju = oke;
11    this.imePrezime = function() {
12        return this.ime + " " + this.prezime;
13    };
14 }
15 var mojOtac = new osoba("Marko", "Markov", 51, "plave");
16 var mojaMajka = new osoba("Ana", "Lukic", 43, "zelene");
17 document.getElementById("demo").innerHTML =
18 "Moj otac se zove " + mojOtac.imePrezime() +
19 ", a majka se zove " + mojaMajka.imePrezime() + ".";
20 </script>
21 </body>
22 </html>
```





Dodavanje metode unutar konstruktora

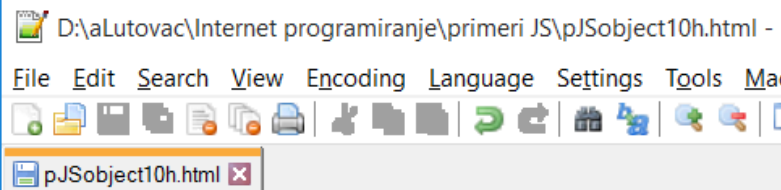
- Dodavanje metoda objektu radi se unutar konstruktor funkcije

```
function Person(firstName, lastName, age, eyeColor) {  
    this.firstName = firstName;  
    this.lastName = lastName;  
    this.age = age;  
    this.eyeColor = eyeColor;  
    this.changeName = function (name) {  
        this.lastName = name;  
    };  
}
```

```
myMother.changeName ("Doe") ;
```

Moj otac se preziva Markov
, a majka se preziva Markov





```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p id="demo"></p>
5 <script>
6 function osoba(ime1,ime2,godina,oke) {
7     this.ime = ime1;
8     this.prezime = ime2;
9     this.god = godina;
10    this.bojaOciju = oke;
11    this.novoIme = function (novo) {
12        this.prezime = novo;
13    }
14 }
15 var mojOtac = new osoba("Marko", "Markov", 51, "plave");
16 var mojaMajka = new osoba("Ana","Lukic",43,"zelene");
17 //mojaMajka.novoIme("Markov");
18 mojaMajka.novoIme(mojOtac.prezime);
19 document.getElementById("demo").innerHTML =
20 "Moj otac se preziva " + mojOtac.prezime +
21 "<br>, a majka se preziva " + mojaMajka.prezime;
22 </script>
23 </body>
24 </html>
```

Moj otac se preziva Markov
, a majka se preziva Markov

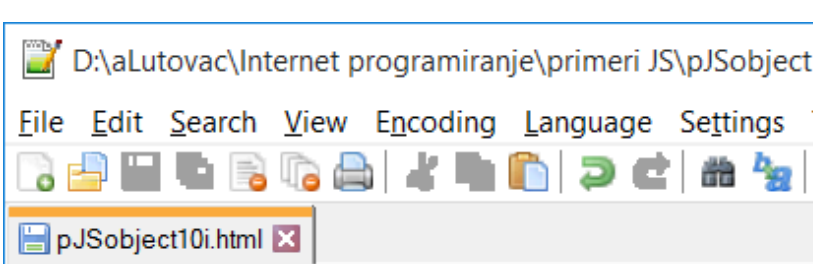




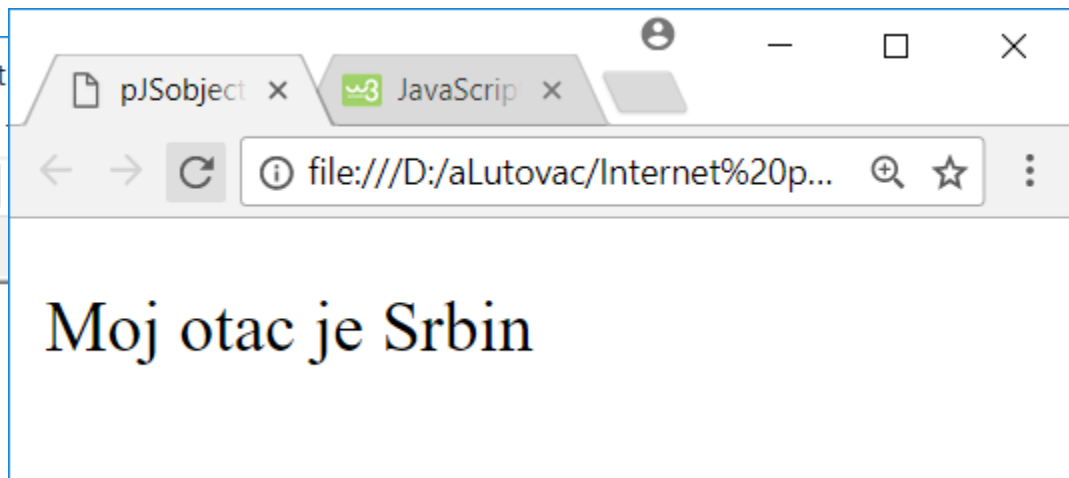
Dodavanje osobine korišćenjem osobine prototajpa

- Primer

```
function Person(first, last, age, eyecolor) {  
    this.firstName = first;  
    this.lastName = last;  
    this.age = age;  
    this.eyeColor = eyecolor;  
}  
  
Person.prototype.nationality = "English";
```



```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p id="demo"></p>
5 <script>
6 function Osoba(ime1,ime2,godina,oke) {
7     this.ime = ime1;
8     this.prezime = ime2;
9     this.god = godina;
10    this.bojaOciiju = oke;
11 }
12 Osoba.prototype.nacionalnost = "Srbin";
13 var mojOtac = new Osoba("Marko", "Markov", 51, "plave");
14 document.getElementById("demo").innerHTML =
15 "Moj otac je " + mojOtac.nacionalnost;
16 </script>
17 </body>
18 </html>
```





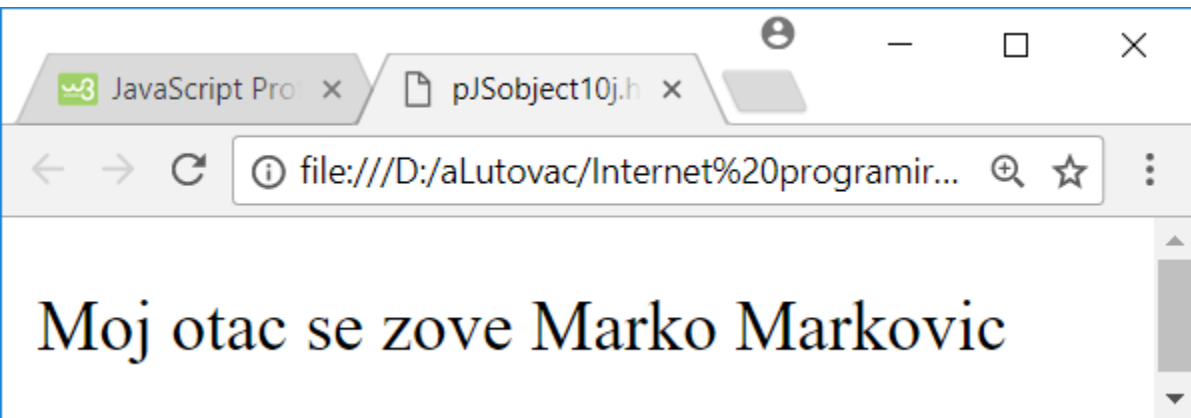
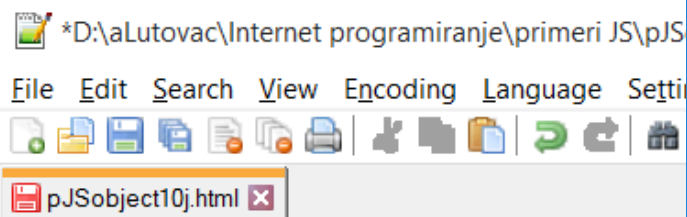
Dodavanje metode korišćenjem osobine prototajpa

- Primer

```
function Person(first, last, age, eyecolor) {  
    this.firstName = first;  
    this.lastName = last;  
    this.age = age;  
    this.eyeColor = eyecolor;  
}
```

Mogu se modifikovati
sopstvene metode,
a nikako standardnih JS objekata

```
Person.prototype.name = function() {  
    return this.firstName + " " + this.lastName;  
};
```



```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p id="demo"></p>
5 <script>
6 function Osoba(ime1, ime2, godina, oke) {
7     this.ime = ime1;
8     this.prezime = ime2;
9     this.god = godina;
10    this.bojaOciju = oke;
11 }
12 Osoba.prototype.imePrezime = function() {
13     return this.ime + " " + this.prezime
14 };
15 var mojOtac = new Osoba("Marko", "Markovic", 51, "plave");
16 document.getElementById("demo").innerHTML =
17 "Moj otac se zove " + mojOtac.imePrezime();
18 </script>
19 </body>
20 </html>
```

