Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko

JavaScript

JavaScript

- JavaScript kot programski jezik
  - sintaksa, funkcije, operatorji, stavki, objekti
  - polja, regularni izrazi
  - integracija z brskalnikom
- komunikacija z vsebino dokumenta (DOM)
  - objekti brskalnika, naslavljanje
  - dogodki, registracija dogodkov
  - programsko spreminjanje dokumenta
- dinamične spletne strani
  - premikanje elementov, vidnost, barve, pisave, prekrivanje
  - animacija

JavaScript

- LiveScript, JavaScript (1995: Netscape & Sun Microsystems), ECMA-262 standard (pozna 90. leta)

- razdeli se lahko v tri dele:
  - **jedro – osrednji del: operatorji, izrazi, stavki, podprogrami**
  - **stran odjemalca: objekti za podporo nadzora odjemalca in komunikacijo z uporabnikom**
  - stran strežnika: objekti uporabni pri spletnem strežniku, npr. podpora komunikaciji s podatkovnimi bazami

- interpretirana koda na strani odjemalca:
  - zaporedje JavaScript kode imenujemo skripta
  - dokument HTML lahko vsebuje poljubno število skript

JavaScript

- lastnosti:
  - dinamično tipiziran – tip spremenljivke se določa med izvajanjem
  - drugačno obravnavanje objektov kot Java (spreminjanje med izvajanjem)
  - interakcija z uporabnikom je enostavna
  - pomemben za Ajax
  - omogoča dinamično interakcijo programerja z dokumentom HTML preko modela **Document Object Model** (DOM)
  - občutljiv na male/velike črke
- ni objektno usmerjen jezik, nima razredov (objekti so tudi modeli)
- osnovni objekt je `Object`
  - vsebuje metode, nima podatkovnih lastnosti
  - prototipno dedovanje
- osnovni (primitivni) tipi: `Number`, `String`, `Boolean`, `Undefined` in `Null`
- komentarji: `//` (enovrstični) in `/* .. */` (večvrstični)

Vključevanje JS kode v HTML

- uporabljamo značko <script>, vstavljamo lahko:
  - direktno v HTML dokument:

```
<script type = "text/javaScript">
  -- programska koda --
</script>
```

  - ali vključitev zunanje izvorne datoteke

```
<script type = "text/javaScript" src = "mojaSkripta.js">
</script>
```

- koda je lahko vključena v glavo in v telo dokumentov HTML
- zunanje datoteke so lahko dosegljive na oddaljenih računalnikih

Primer vključene kode

```
<!DOCTYPE html>

<!-- Fig. 6.1: welcome.html -->
<!-- Displaying a line of text. -->
<html>
   <head>
      <meta charset = "utf-8">
      <title>A First Program in JavaScript</title>
      <script type = "text/javascript">

         document.writeln(
            "<h1>Welcome to JavaScript Programming!</h1>" );

      </script>
   </head><body></body>
</html>
```

- značka <script>
- tip MIME
- objekt document
- izpis kode HTML
- podpičje!!! (JavaScript interpreter poskuša implicitno postaviti podpičja a ne uspe vedno)
- priporočljivo je napisati celoten stavek v eni vrstici

Izključevanje JavaScript kode

- z uporabo značke `<script>` lahko vključimo kodo
- nekateri uporabniki imajo lahko onemogočeno izvajanje JavaScript programov ali pa imajo brskalnike, ki ne podpirajo JavaScript-a
- lahko se uporabi značka <noscript>, ki omogoča definiranje alternativne vsebine za uporabnike

```
<script type = "text/javaScript">
  document.write("<h2>It appears that JavaScript works</h2>")
</script>
<noscript>
  <h2>Your browser does not support JavaScript!!!</h2>
</noscript>
```

- značka <noscript> se lahko pojavi v telesu (HTML 4.01) ali v telesu oziroma glavi dokumenta (HTML5)
- starejši brskalniki ne razumejo JavaScripta, zato je potrebno dele kode pred njimi skrit → `<!--  -- JavaScript code --  //-->`

Izključevanje JavaScript kode

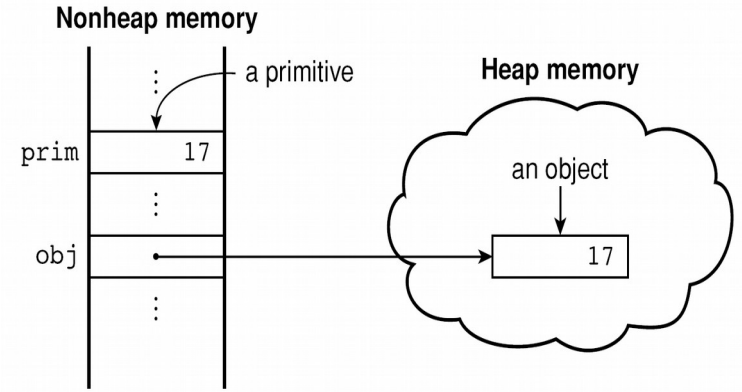
```
<!DOCTYPE HTML>
<html>
 <head>
   <title>Testing of tinfoil hats</title>
   <style type="text/css"> h1 {color: blue;} h2 {color: green; font-size: 0.8em;}
   </style>
   <noscript>
     <style type="text/css">        h1 {color: orange;} h2 {color: red; font-size: 2.5em;}
     </style>
   </noscript>
 </head>
 <body>
   <h1>This is an exampel of changing the document
       appearance according to the availability of JavaScript!
   </h1>
  <script>document.write("<h2>It appears that the JavaScript works</h2>");
  </script>
  <noscript>
     <h2>I am sorry, but your browser does not allow for JavaScript</h2>
  </noscript>
  <hr>
 </body>
</html>
```

Tipizacija jezika

- **dinamično tipiziran jezik**: vsaka spremenljivka lahko hrani poljuben tip podatka (osnovni tip ali referenco na objekt)
- **šibko tipiziran jezik**: podpira implicitne pretvorbe tipov pri aritmetičnih operacijah
- **eksplicitna** (`var`) ali **implicitna** (prireditev vrednosti) deklaracija spremenljivk

```
var sum = 0,
    today = "Monday",
    flag = false;
```

- `typeof(spremenljivka)` vrne tip spremenljivke
- osnovni in kompleksni tipi (`Number`, `String` in `Boolean` so tako osnovni kot tudi kompleksni tipi)



Ključne besede

- JavaScript vsebuje več rezerviranih besed
- JavaScript standard definirane še dodatne ključne besede, ki pa niso (trenutno) še v uporabi (ES6, 2015, uvaja: class, implements, static, extends, super => podobno konstruktom temelječim na razredih, NI ISTO, prototipno dedovanje)
  - sintaktični sladkorček na obstoječo prototipno dedovanje

JavaScript reserved keywords

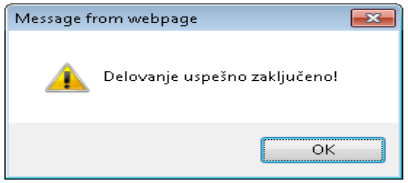
break	case	catch	continue	default
delete	do	else	false	finally
for	function	if	in	instanceof
new	null	return	switch	this
throw	true	try	typeof	var
void	while	with		

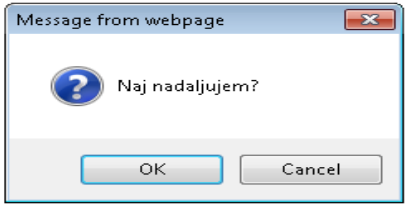
Keywords that are reserved but not used by JavaScript

class	const	enum	export	extends
implements	import	interface	let	package
private	protected	public	static	super
yield				

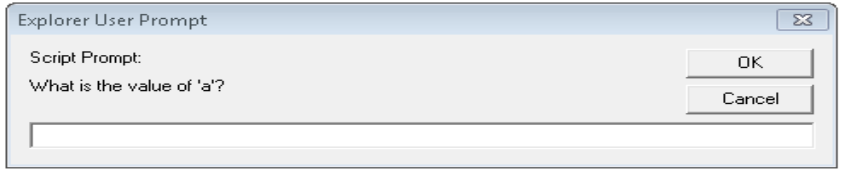
Pisanje v dokument, branje vnosa

- objekt `Window`, ki ima lastnost `document` (referenca na dokument)
  - `document.write('Sedaj pa smo nekaj izpisali<br />')`
- objekt `Window` ima metode za uporabo pogovornih oken:
  - `alert("Delovanje uspešno zaključeno! \n");`





  - `confirm("Naj nadaljujem?");`



  - `var a = prompt("What is the value of 'a'? \n", "");`
    - brano besedilo je String, potrebne pretvorbe (parseInt, parseFloat)

Primer izpisa

```
<!DOCTYPE HTML >
<html>
  <head> <title>Prvi JS program</title>
    <meta charset="utf-8" />
    <script type = "text/javascript">  alert("Prikaz opozorilnega okna");  </script>
  </head>
  <body>
    <script type = "text/javascript">
      document.write("<h1>Uspelo nam je prikazati prvo okno</h1>");
      var resp = confirm("Naj nadaljujem?");
      if (resp == true) {
        var ime = prompt("Vpisi ime", "Danko Bananko");
        document.write("<h2> Pozdravljen "+ ime + " </h2>"); }
    </script>
    <h2>Test klicanja oken</h2>
    <script type="text/javascript">  console.log('Pa se en primer izpisa'); </script>
    <hr>
    <script type="text/javascript">
      document.write("<h4>Sedaj pa bomo koncali s tem primerom</h4>");
    </script>
  </body>
</html>
```

Objekti, metode, lastnosti, operatorji

- **Math:** Math.floor(), round(), max(), min(), sin(), cos(), ...
- **Number:** Number.MAX_VALUE, MIN_VALUE, NaN, POSITIVE_INFINITY, NEGATIVE_INFINITY, PI, toString()
- **String:** String.length, charAt(), indexOf(), substring(), toLowerCase(), toUpperCase()
  - operator +: konkatenacija nizov
- **Date:** Date.toLocaleString(), getDate(), getMonth(), getFullYear(), getTime(), getHours(), getMinutes(), getSeconds(), getMilliseconds()

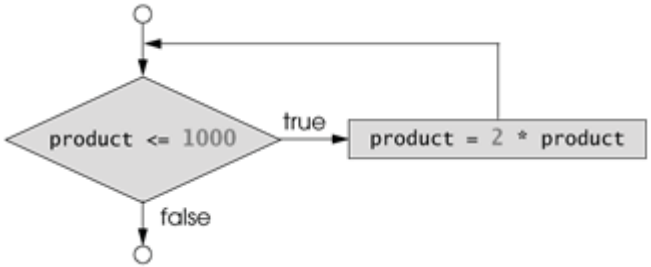
Operator	Associativity	Type
++ --	right to left	unary
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== != === !==	left to right	equality
?:	right to left	conditional
= += -= *= /= %=	right to left	assignment

Pretvorbe tipov

- implicitna pretvorba tipov:
  - JavaScript izvaja nekaj implicitnih vrst pretvorb (“coercion” – prisila)
  - vrednost enega tipa na mestu, kjer se rabi drug tip
  - “21. oktober ” + 2015 => “21. oktober 2015”
  - 2015 + “ oktober” => “2015 oktober”
  - 7 * “3” => 21
  - 7 + “3” => 73 (7 + 3 => 10)
- eksplicitna pretvorba tipov:
  - lahko se pretvorijo s konstruktorjem `String`:
    ```
    var str_val = String(value)
    ```
  - lahko se uporabi metoda `toString()`
    ```
    var num=6;   var str_val=num.toString(2);
    ```
  - lahko se uporabi konstruktor `Number`:
    ```
    var number = Number(aString)
    ```
  - omejitve: številu v nizu lahko sledi samo presledek

Kontrola toka programa

- `if (pogoj1) {...}`
  `[else if (pogoj2) {...} ]`
  `[else {...}]`
- `pogoj ? then-blok : else-blok`
- `switch (izraz) {`
  `case vrednost_1:`
  `// vrednost_1 stavki !!!! break;`
  `case vrednost_2:`
  `// vrednost_2 stavki !!!! break;`
  `…`
  `[default:`
  `// default stavki]`
  `}`
- `while (pogoj) ...`
- `do`
  `stavki`
  `while (pogoj)`
- `for (inicializacija; pogoj; sprememba stanja) {...}`
- `for (identifikator in objekt) {...}`



Funkcije

- podobne funkcijam v C in C++, sestavljene iz glave in telesa, ki opisuje akcije funkcije:
```
function ime_funkcije([formalni_parametri]) {
 -- telo funkcije --
}
```
- globalne spremenljivke
- lokalne le tiste, ki so eksplicitno deklarirane v funkciji (beseda var)
- vrednosti parametrov so podane v spremenljivki `arguments`
- funkcije so objekti, tako da se lahko spremenljivke, ki jih referencirajo, obravnavajo kot druge spremenljivke (parametri, prirejanje, elementi polj):
```
function fun(){ ... };
ref_fun = fun;
ref_fun();
```
- definicije funkcij ponavadi v glavi dokumenta HTML
- primitivni tipi se prenašajo po vrednosti (ustvari se lokalna kopija)
- kompleksni tipi (polja) se prenašajo po referenci

Funkcije

```
<!DOCTYPE HTML >
<html>
  <head> <title>Drugi JS program</title>
    <meta charset="utf-8" />
    <script type = "text/javascript">
      function prvaFunkcija(ime){
        console.log("Hello, " + ime )
      }
    </script>
  </head>
  <body>
    <h1> Test funkcij</h1>
    <input type="button" value="Pritisni me" onclick="drugaFunkcija()">
    <script type = "text/javascript">
    function drugaFunkcija() {
      alert("You successfully called first function");
      var ime = prompt("Enter your name");
      prvaFunkcija(ime);
    }
    </script> <hr>
  </body>
</html>
```

Izdelava in spreminjanje objektov

- izdelava objektov z ukazom new

```
var my_object = new Object();
```

- lastnosti je objektu možno dodati ali izbrisati **kadarkoli** med izvajanjem

```
var student = new Object();
    student.ime = “Danko";
    student.letnik = "prvi";
delete student.letnik
```

- objekt je možno kreirati tudi na krajši način:

```
var student = {ime: “Danko”, letnik: “prvi”};
```

Izdelava in spreminjanje objektov

- konstruktor **inicializira objekt**

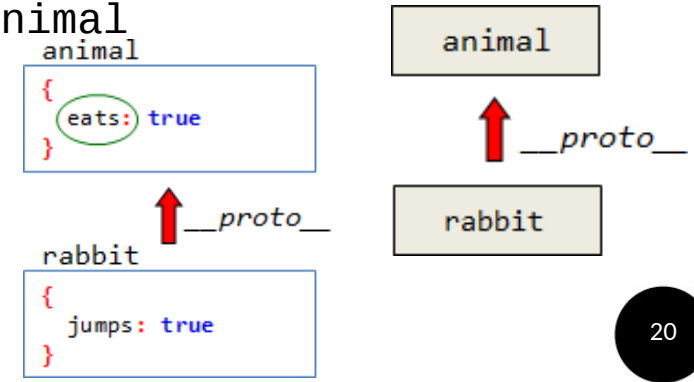
```
function student(ime1, letnik1, letoVpisa1) {
    this.ime = ime1;
    this.letnik = letnik1;
    this.letoVpisa = letoVpisa1;
}
var mojKolega = new student("David", "tretji", "2011");
```

- iteriranje preko lastnosti objekta
  - `for (var prop in mojKolega)`
    `document.write(mojKolega[prop] + "<br/>");`
- lastnosti objekta so lahko tudi funkcije

Prototipno dedovanje

- večina jezikov uporablja razrede in objekte:razredi dedujejo od ostalih razredov
- JavaScript nima razredov; objekt deduje od drugega objekta
- ko objekt `ChildObject` deduje od objekta `parentObject` v JS, to pomeni, da obstaja posebna lastnost `ChildObject.__proto__ = parentObject`
- te lastnosti se v nekaterih brskalnikih ne da nastaviti direktno (samo FF/Chrome dovoljujeta direkten dostop), potrebno je uporabiti kodo posebej za to
  `ChildObject.prototype = parentObject`
- če poskuša dostopati do latnosti v objektu `rabbit` in se je ne najde v `rabbit`, interpreter sledi povezavi `__proto__` in išče v objektu `animal`

```
var animal = {eats: true}
function Rabbit (name) {// => constructor
      this.name = name
      this.jumps = true
}
Rabbit.prototype = animal
// => set __proto__ = animal for all Rabbit objects
var rabbit = new Rabbit('John') // inheritance
alert(rabbit.eats) // true, rabbit.__proto__= animal
```



Prototipno dedovanje

- če pa se pri dostopu lastnost najde v `rabbit`, se povezavi v objekt `animal` ne sledi

```
var animal = {eats: true}
function Rabbit (eats) {this.eats = eats}
Rabbit.prototype = animal // => set __proto__ = animal for all Rabbit objects
var fedUpRabbit = new Rabbit(false) // inheritance
// sets the rabbit property eats to false shadows __proto__ property eats
alert(fedUpRabbit.eats) // displays false since rabbit.eats = false
```

- podobno se lahko lastnosti in funkcije v `animal` definirajo/dostopajo preko `__proto__`

```
alert(fedUpRabbit.__proto__.eats) // displays true,__proto__ property eats = true
```

- spreminjanje `prototype` ne vpliva na že kreirane objekte,
- spreminjanje parentObject vpliva na že kreirane objekte
- brskalniki podpirajo metodi `create` (za kreiranje in dedovanje) in `hasOwnProperty` (preverjanje, če metoda pripada objektu ali prototipu)

```
var animal = {eats: true}
rabbit = Object.create (animal) // inheritance
alert(rabbit.eats) // displays true
alert(rabbit.hasOwnProperty('eats')) // false, defined in prototype
```

Polja (arrays)

- polja so dinamične dolžine
- izdelava

```
var myList = new Array(24, "bread", true); // polje dolžine 3
var myList2 = [24, "bread", true]; // polje dolžine 3
var myList3 = new Array(24); // prazno polje dolžine 24
```

- elementi v polju so lahko različnih tipov
- dolžina polja: `myList.length`
- metode: `join()`, `reverse()`, `sort()`, `concat()`, `slice()`, `toString()`, `push()` in `pop()`; `shift()` in `unshift()`: sklada, kombinacija pa za vrsto
- prirejanje vrednosti novemu elementu ustvari ta element

```
myList3[1000]= 2222
```

- v funkcije se prenašajo po referenci
- sortiranje polj: leksikografsko / numerično

```
function num_ord(a,b){return b-a;}
num_list.sort(num_ord);
```

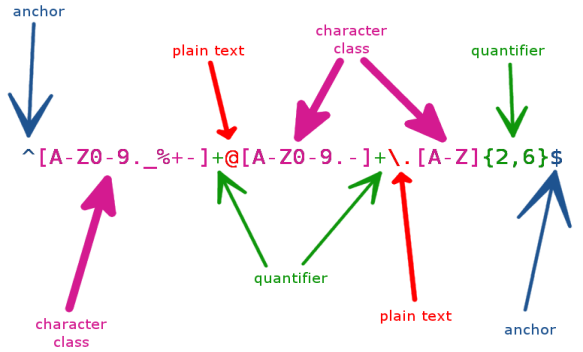
Polja objektov - primer

```
<!DOCTYPE HTML >
  <html> <head> <title>Drugi JS program</title> <meta charset="utf-8" />
    <script type = "text/javascript">
      function preberiKandidata(){
        var stev=prompt("Vpisi stevilo kandidatov", "1"); preberiKandidate(stev); }
      function preberiKandidate(stev){
        var objs = new Array(); console.log(stev);
        for (i = 0; i < stev; i++) {
          obj=new Object();
          obj.ime=prompt("Vpisi ime");
          obj.pri=prompt("Vpisi priimek");
          obj.str=prompt("Vpisi starost");
          objs.push(obj);
        }
        var json=new Object();
        json.kandidati=objs;
        console.log(JSON.stringify(json));
      }
    </script>
  </head>
  <body> <h1> Test funkcij</h1>
    <input type="button" value="Pritisni me" onclick="preberiKandidata()">
  </body> </html>
```

```
4
{"kandidati":[
  {"ime":"Danko",    "pri":"Bananko",  "str":"24"},
  {"ime":"Maruska", "pri":"Hruska",    "str":"24"},
  {"ime":"Anka",     "pri":"Zaspanka", "str":"25"},
  {"ime":"Janko",    "pri":"Bananko",  "str":"26"}
 ]
}
```

Regularni izrazi in preverjanje vzorcev

- uporabno pri preverjanju **pravilnosti strukture** vnosov v obrazce
- regularne izraze navedemo v poševnicah, npr. `/student/`
- posebni znaki:
  - . poljuben znak ,razen nove vrstice (`/d.n/` ustreza nizom *dan* in *drn*)
  - * nič ali več ponovitev (`/a*b/` ustreza nizom *b*, *ab*, *aab*, ...)
  - + ena ali več ponovitev (`/a+b/` ustreza nizom *ab*, *aab*, ...)
  - ? nič ali ena ponovitev (`/a?b/` ustreza nizoma *b*, *ab*)
  - {n} n ponovitev (`/a{3}b/` ustreza nizu *aaab*)
  - [abcd] razred znakov, eden od navedenih znakov (`/[abc]b/` ustreza nizom a*b*, *bb*, in *cb*)
  - [a-z] razpon znakov (`/[a-c]b/` ustreza nizom a*b*, *bb*, in *cb*)
  - [^0-9] ^ predstavlja negacijo, npr. vsi znaki razen števk (`/[^abc]b/` ustreza nizom *db*, *eb* ...)
  - /^xxx/ ^ v poševnicah predstavlja sidro - začetek niza
  - /xxx$/ $ v poševnicah predstavlja sidro - konec niza



Name	Equivalent Pattern	Matches
\d	[0-9]	A digit
\D	[^0-9]	Not a digit
\w	[A-Za-z_0-9]	A word character (alphanumeric)
\W	[^A-Za-z_0-9]	Not a word character
\s	[\r\t\n\f]	A white-space character
\S	[^ \r\t\n\f]	Not a white-space character

Regularni izrazi in preverjanje vzorcev

- modifikatorji niza
  - `/izraz/i` povzroči ignoriranje velikih/malih črk
  - `/izraz/g` povzroči globalno iskanje (vse pojavitve niza)
  - `/izraz/x` ignorira bele presledke (presledek, tabulator, nova vrstica, …) v izrazu
- funkcije
  - **`search(izraz)`** vrne lokacijo iskanega podniza

```
var str = "Some rabbits are rabid";
var pos = str.search(/rabb/);
document.write("'rabb' se pojavi na poziciji", pos);
```

  - **`replace(izraz, niz)`** zamenja podniz z drugim podnizom; `$1, $2, …` vrednosti zamenjanih podnizov

```
var str = "Some rabbits are rabid";
str.replace(/(rab)/g, "tim");
rezultat: str = "Some timbits are timid"; RegExp.$1 ima vrednost "rab"
```

  - **`match(izraz)`** vrne polje najdenih nizov

```
var str = "My 3 kings beat your 2 aces";
var matches = str.match(/[ab]/g);
rezultat: matches = ["b", "a", "a"]
```

  - **`split(parameter)`** razdeli niz glede na regularni izraz

Regularni izrazi in preverjanje vzorcev

```
<!DOCTYPE HTML>
<html> <head> <title>Regularni izrazi</title> <meta charset="utf-8"/>
  <script type="text/javascript">
    function preveriIP(){
      var uC="[A-Z\u010C\u0160\u017d\u0106\u0110]";
      var lC="[a-z\u010d\u0161\u017e\u0107\u0111]";
      var ipRE=new RegExp("^("+uC+lC+"+)( +"+uC+"\.)?( +"+uC+lC+"+){1,2}$");
      var imp=prompt("Vpisi ime in priimek, (ime [inic.] priimek [priimek])");
      var pos=imp.search(ipRE);
      if (pos < 0) {alert("Vnesel si napacno ime");}
      if (!ipRE.test(imp)) {alert("Vnesel si napacno ime-test");}
    }
    function preveriRD(){
      var rd=prompt("Vpisi datum rojstva, (dd mm llll )");
      var pos=rd.search(/^(\d{1,2})[ \-\s:\/]+(\d{1,2})[ \-\s:\/]+(\d{4})$/);
      if (pos < 0) {alert("Vnesel si napacen datum");return false;}
      var mtch=rd.match(/^(\d{1,2})[ \-\s:\/]+(\d{1,2})[ \-\s:\/]+(\d{4})$/);
      console.log(mtch);
      if (mtch[1]<1 || mtch[1]>31) {alert("Vnesel si neobstojec dan: " + mtch[1]);}
      if (mtch[2]<1 || mtch[2]>12) {alert("Vnesel si neobstojec mesec: " + mtch[2]);}  }
 </script> </head>
 <body> <h1>Regularni izrazi</h1>
   <input type="button" value="Vpisi ime in priimek" onclick="preveriIP()" />
   <input type="button" value="Vpisi datum rojstva" onclick="preveriRD()" />
 </body> </html>
```

Razhroščevanje programske kode

- prikaz s strani brskalnika
  - IE:
    Tools/Developer Tools/Console
  - FF:
    Tools/Web Developer/Inspector/Console
  - Chrome:
    Tools/JavaScript Console

- razhroščevalnik (debugger):
  - IE:
    Tools/Developer Tools/Debugger
  - FF:
    Tools/Web Developer/Inspector/Debugger
  - Chrome:
    Tools/JavaScript Console/Sources