# STANDARDNI KORISNIČKI INTERFEJSI

Predavanje broj: 09

Nastavna jedinica: JavaScript

Nastavne teme:

Scope. Događaj programskog dugmeta. String i prelom linije koda. nalaženje stringa u stringu. Ekstrahovanje dela stringa. Metode stringa. JavaScript brojevi. Globalne metode. Math. Datum. Nizovi (sortiranje, kretanje kroz niz, kreiranje). Metode niza. Bitwise operacije. Labela. Svojstvo constructor. Konverzije. Regularni izrazi. Upravljanje greškama. Use strict. Validacija.

Predavač: prof. dr Perica S. Štrbac, dipl. ing.

Literatura:

J. D. Gauchat, "Integrisane tehnologije za izradu WEB strana", Mikroknjiga, Beograd, 2014.

W3C Tutorials, Internet, 2014

# scope

- JavaScript ima lokalne i globalne promenljive.
  - Lokalna promenljiva se vidi samo u okviru funkcije u kojoj je deklarisana.
  - U funkciji **var** ispred promenljive određuje da je ista lokalna (koristi se i **let** gde se uvažava područje važenja).

```
<!DOCTYPE html>
<html><body>
<p id="demo"></p>
<script>
var carFord = "Ford";          //globalna
myFunction();
document.getElementById("demo").innerHTML =
                                  "Volvo je tipa " + typeof carName;
document.getElementById("demo").innerHTML =
document.getElementById("demo").innerHTML +"<br> Ford je tipa " +
typeof carFord +"<br> BMW je tipa " + typeof carBMW;
//moze i krace
function myFunction() {
    var carName = "Volvo";                    //lokalna
    carBMW = "BMW";                           //g l o b a l n a
}
</script></body></html>
```

Volvo je tipa undefined
Ford je tipa string
BMW je tipa string

# scope

- JavaScript globalne promenljive postoje sve vreme od trenutka deklaracije
  - Globalne promenljive se brišu kada se zatvori stranica.
  - U JavaScriptu globalni opseg je kompletno JavaScript okruženje.
- JavaScript lokalne promenljive se brišu po povratku iz funkcije u kojoj su kreirane.
- Argumenti funkcije predstavljaju lokalne promenljive unutar te funkcije.
- U HTMLu globalni opseg je window objekat
  - Sve globalne promenljive pripadaju window objektu.

```
<!DOCTYPE html>
<html><body>
<p>In HTML, all global variables will become window variables.</p>
<p id="demo"></p>
<script>
myFunction();
function myFunction() {    carName = "Volvo";}
document.getElementById("demo").innerHTML =
"I can display " + window.carName;
</script>
</body></html>
```

# Preimenovanje natpisa programskog dugmeta

```
<!DOCTYPE html>
<html><body>
<button onclick="this.innerHTML=Date()">The time is?</button>
</body></html>
```

- Lista nekih HTML događaja:
  - onchange
    - HTML element je promenjen
  - onclick
    - korisnik je kliknuo na HTML element
  - onmouseover
    - korisnik pomera miš iznad HTML elementa
  - onmouseout
    - korisnik pomeranjem miša izlazi van prostora HTML elementa
  - onkeydown
    - korisnik je pritisnuo taster na tastaturi
  - onload
    - browser je učitao celu stranicu

# string i prelom linije koda

- Dužina stringa:

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
```

- Prelom linije koda unutar stringa:

```
document.getElementById("demo").innerHTML = "Hello \
   Dolly!";
```

sigurnije rešenje je

```
document.getElementById("demo").innerHTML = "Hello " +
   "Dolly!";
```

- NE sme se uraditi sledeće:

```
document.getElementById("demo").innerHTML = \
   "Hello Dolly!";
```

- NE kreiratu stringove kao objekte jer dolazi do usporavanja izvršavanja. Mala napomena:

```
var x = "John";
var y = new String("John");
```

navedeni stringovi su jednaki ali nisu identični.

# Nalaženje stringa u stringu

```
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate");
//vraca prvu poziciju pojavljivanja, 7

var str = "Please locate where 'locate' occurs!";
var pos = str.lastIndexOf("locate");
//vraca poslednju poziciju pojavljivanja, 21
```

- Obe prethodne metode mogu imati i drugi paramatar koji govori odakle (od kog indeksa) počinje pretraga. Obe metode će vratiti vrednost -1 ako ne nađu traženi string.
- Metoda **search** radi isto što i **indexOf** ali je moćnija jer može da koristi i regularne izraze.
- Metode za ekstrahovanje dela stringa:
  - slice (start, end)
  - substring (start, end)
  - substr (start, length)

# Ekstrahovanje dela stringa

```
var str = "Apple, Banana, Kiwi";
var res = str.slice(7,13);
//Banana
```

- Ako je parameter negativan onda se pozicija računa od kraja stringa.
  - ekstrahovanje dela stringa od pozicije -12 do pozicije -6 isključno:

```
var str = "Apple, Banana, Kiwi";
var res = str.slice(-12,-6);
//Banana
```

- Bez drugog parametra metoda slice vraća ostatak stringa:

```
var res = str.slice(7);
//Banana, Kiwi
```

isto vredi i za negativni indeks:

```
var res = str.slice(-12);
//Banana, Kiwi
```

- Metoda **substring()** je slična metodi slice(), osim što NE prihvata negativne indekse:

```
var str = "Apple, Banana, Kiwi";
var res = str.substring(7,13);    //Banana
```

# metode string-a

- Metoda **substr()** je sličan metodi slice(). Razlika je što se drugi parametar odnosi na dužinu ekstrahovanog dela stringa.

```
var str = "Apple, Banana, Kiwi";
var res = str.substr(7,6);
//Banana
```

- Metoda **replace()** zamenjuje specificirani string drugim stringom:

```
str = "Please visit Microsoft!";
var n = str.replace("Microsoft","Google");
```

- Konvertovanje teksta u velika slova vrši se pomoću metode **toUpperCase()**:

```
var text1 = "Hello World!";       // Hello World!
var text2 = text1.toUpperCase();  // HELLO WORLD!
```

- Konvertovanje teksta u mala slova vrši se pomoću metode **toLowerCase()**:

```
var text1 = "Hello World!";       // Hello World!
var text2 = text1.toLowerCase();  // hello world!
```

# metode stringa

- Metoda **concat()** spaja dva ili više stringova:

```
var text1 = "Hello";
var text2 = "World";
text3 = text1.concat(" ",text2);
```

  - Ova metoda može se koristiti umesto operatora plus.
  - Sledeće linije koda imaju isto dejstvo:

```
var text = "Hello" + " " + "World!";
var text = "Hello".concat(" ","World!");
```

- Dve su metode za ekstrakciju karaktera iz stringa:
  - charAt(position) , vraća kakrakter na specificiranom indeksu stringa

```
var str = "HELLO WORLD";
str.charAt(0);            // returns H
```

  - charCodeAt(position) ,vraća unicode karaktera na specificiranom indeksu stringa:

```
var str = "HELLO WORLD";
str.charCodeAt(0);         // returns 72
```

# metode stringa

- Pristup stringu kao nizu je nesiguran (*unsafe*)

  NE raditi sledeće:

```
var str = "HELLO WORLD";
str[0];                    // returns H
```

  - nesigurno je i nepredvidivo
  - str[0] = "H" neće dati grešku ali i ne radi

- Ako se baš želi raditi sa nizom onda bi trebalo konvertovati string u niz.
  - String se može konvertovati u niz korišćenjem metode **split()**:

```
var txt = "a,b,c,d,e";          // String
arr1 = txt.split(",");          // Split on commas
arr2 = txt.split(" ");          // Split on spaces
arr3 = txt.split("|");          // Split on pipe
```

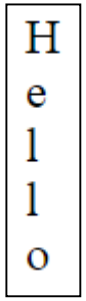
- Ako se ne navede separator onda će ceo string biti u rezultatnom nizu smešten na poziciju niz[0].

# metode stringa

- Ako je separator "", rezultatani niz biće niz karaktera datog stringa:

```
var txt = "Hello";          // String
txt.split("");              // Split in characters
```

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
var str = "Hello";
var arr = str.split("");
var text = "";
var i;
for (i = 0; i < arr.length; i++) {
    text += arr[i] + "<br>"
}
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```



# JavaScript brojevi

- Realni brojevi su u formatu IEEE754 (1 predznak, 11 karakteristika, 52 mantisa/frakcija).
- Preciznost integer-a je na 15 cifara:

```
var x = 999999999999999;   // x =   999999999999999
var y = 9999999999999999;  // y = 10000000000000000
```

- Paziti na nepreciznost float-a:

```
var x = 0.2 + 0.1;     // x = 0.30000000000000004
```

rešenje problema

```
var x = (0.2 * 10 + 0.1 * 10) / 10;      // x = 0.3
```

- Korišćenje metode toString(baza) za prikaz brojeva u datoj bazi:

```
var myNumber = 128;
myNumber.toString(16);      // returns 80
myNumber.toString(8);       // returns 200
myNumber.toString(2);       // returns 10000000
```

# JavaScript brojevi

- JavaScript vraća Infinity ako je rezultat broj veći od najvećeg dozvoljenog:

```
var myNumber = 2;
while (myNumber != Infinity) {  // Execute until Infinity
    myNumber = myNumber * myNumber;
}
```

deljenje sa nulom vraća Infinity

```
var x =  2 / 0;          // x = Infinity
var y = -2 / 0;          // y = -Infinity
```

- Infiniti je tipa broj:

```
    typeof Infinity;     // number
```

- NaN znači nije broj (Not a Number):

```
var x = 100 / "Apple";  // x = NaN (Not a Number)
isNaN(x);               // true
typeof NaN;             // number
```

- Paziti na korišćenje NaN:

```
var x = NaN;
var y = 5;
var z = x + y;          // z = NaN
```

# JavaScript brojevi

- Brojevi mogu biti objekti (NE treba to koristiti):

```
var x = 123;
var y = new Number(123);
// typeof x returns number
// typeof y returns object
// (x == y) is true
// (x === y) is false
```

- JavaScript objekti se ne mogu porediti po stanju:

```
var x = new Number(500);
var y = new Number(500);
// (x == y) is false,
```

- Vrednosti dostupne preko **Number.vrednost:**
  - MAX_VALUE najveći broj u JavaScript-u (1.7976931348623157e+308)
  - MIN_VALUE najmanji broj u JavaScript-u (5e-324)
  - NEGATIVE_INFINITY negativno beskonačno (kod overflow-a) (-Infinity)
  - POSITIVE_INFINITY beskonačno (kod overflow-a) (Infinity)
  - NaN nije broj (NaN)
- Paziti na: `var x = 6; var y = x.MAX_VALUE;  // y = undefined`

# Globalne metode

- Globalne metode:
  - Number() vraca broj od konvertovanog argumenta.
  - parseFloat() parsira argument i vraca floating point broj
  - parseInt() parsira argument i vraca integer broj
- Metode brojeva:
  - toString() vraća broj kao string
  - toExponential() vraća string kao zaokružen broj sa eksponencijalnom notacijom
  - toFixed() vraća string kao broj sa specificiranim brojem decimala
  - toPrecision() vraća string kao broj specificirane dužine (broja cifara)
  - valueOf() vraća broj

```
var x = 123;
x.toString();              // 123 iz x
(123).toString();          // 123 iz literala 123
(100 + 23).toString();     // 123 iz izraza 100 + 23

                  // setiti se toString(2) → binarni prikaz
```

# Globalne metode

```
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
```

  - Ako se izostavi parametar nema zaokruživanja broja

- Podesno za baratanje novcem:

```
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
```

- Preciznost do na broj cifara:

```
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
```

# Globalne metode

- Konvezija promenljive u broj:
  - Number()
  - parseInt()
  - parseFloat()
- Primeri:

```
x = true;         Number(x); // 1
x = false;        Number(x); // 0
x = new Date();   Number(x); // milisekunde od 1.1.1970.
                             // 1404568027739

x = "10";         Number(x); // 10
x = "10 20";      Number(x); // NaN


parseInt("10");           // 10
parseInt("10.33");        // 10
parseInt("10 20 30");     // 10
parseInt("10 years");     // 10
parseInt("years 10");     // NaN
parseInt('FF',16);      // 255,  mora znati bazu
```

# Globalne metode

```
parseFloat("10");          // 10
parseFloat("10.33");       // 10.33
parseFloat("10 20 30");    // 10
parseFloat("10 years");    // 10
parseFloat("years 10");    // NaN


var x = 123.456;
x.valueOf();               // 123.456 iz x
(123.456).valueOf();       // 123.456 iz literala 123.456
(100+23+0.456).valueOf();   // 123.456 iz izraza 100+23+0.456
```

- Objekat Math:

```
Math.random();                      // slučajan broj [0..1)
Math.min(0, 150, 30, 20, -8);       // -8
Math.max(0, 150, 30, 20, -8);       // 150
Math.round(4.7);                    // 5
Math.round(4.4);                    // 4
Math.ceil(4.4);                     // 5
Math.floor(4.7);                    // 4
Math.floor(Math.random() * 11);     // slučajan broj [0..11)
```

# Math

```
Math.E;          // Euler-ov broj
Math.PI          // PI
Math.SQRT2       // kvadratni koren od 2
Math.SQRT1_2     // kvadratni koren od 1/2
Math.LN2         // prirodni logaritam od 2
Math.LN10        // prirodni logaritam od 10
Math.LOG2E       // logaritam od E po bazi 2
Math.LOG10E      // logaritam od E po bazi 10
```

`abs(x)`, apsolutno x; `acos(x)`, arkuskosinus od x u radijanima;
`asin(x)`, arkussinus od x u radijanima;
`atan(x)`, arkustangens od x u opsegu -PI/2 i PI/2 radijana;
`atan2(y,x)`, arkustangens preko koeficijenta argumenata;
`ceil(x)`, najbliži integer iznad ili jednak x;
`cos(x)`, kosinus od x (x je u radijanima);
`exp(x)`, $E^x$; `floor(x)`, nabliži integer ispod ili jednak x;
`log(x)`, prirodni logaritam od x; `log10(x)`, logaritam od x po bazi 10;
`max(x,y,...,n)`, `min(x,y,...,n)`, vraća max/min od datih elemenata;
`pow(x,y)`, $x^y$; `round(x)`, zaokruživanje na najbliži integer;
`sin(x)`, sinus od x (x je u radijanima); `sqrt(x)` , kvardratni koren od x;
`degrees(radijani)` → u stepene, `radians(stepeni)` → u radiajane
`tan(x)`, tangens od x (x je u radijanima);

# Datum

- Kreiranje datuma:

```
new Date();
new Date(milliseconds);
new Date(dateString);
new Date(year, month, day, hours, minutes, seconds, milliseconds);
```

- Kreiranje tekućeg datuma i vremena :

```
<script>
var d = new Date();
document.getElementById("demo").innerHTML = d;
</script>
```

- Kreiranje specificiranog datuma i vremena :

```
<script>
var d = new Date("October 13, 2014 11:13:00");
document.getElementById("demo").innerHTML = d;
</script>
```

- Kod new Date(*number*), kreira objekat datum koji odgovara datumu udaljenom za *number* milisekundi od 01.01.1970.

```
var d = new Date(86400000);
```

# Datum

```
<script>
var d = new Date(99,5,24,11,33,30,0);
document.getElementById("demo").innerHTML = d;
</script>

<script>
var d = new Date(99,5,24);
document.getElementById("demo").innerHTML = d;
</script>
```

- Automatska konverzija u string ili eksplicitnim korišćenjem metode toString():

```
<p id="demo"></p>
<script>
d = new Date();
document.getElementById("demo").innerHTML = d;
</script>
```

isto kao:

```
<p id="demo"></p>
<script>
d = new Date();
document.getElementById("demo").innerHTML = d.toString();
</script>
```

# Datum

```
<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.toUTCString();
</script>
```

- Izlaz: Sun, 19 Apr 2015 22:39:07 GMT

```
<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.toDateString();
</script>
```

- Izlaz: Sun Apr 19 2015

- JavaScript stringovi datuma:
  - ISO Dates
    - `var d = new Date("2015-03-25");`
    - `var d = new Date("2015-03");`
    - `var d = new Date("2015-03-25T12:00:00");`
  - Long Dates
    - `var d = new Date("Mar 25 2015");`

# Datum

- `var d = new Date("25 Mar 2015");`
- `var d = new Date("2015 Mar 25");`
- `var d = new Date("January 25 2015");`
- `var d = new Date("Jan 25 2015");`
- case insensitive, ignorišu se zarezi
  - `var d = new Date("2015, JANUARY, 25");`

– Short Dates

- `var d = new Date("03/25/2015");`
- `var d = new Date("03-25-2015");`
- `var d = new Date("2015/03/25");`

- Pun format:

```
var d = new Date("Wed Mar 25 2015 09:56:24 GMT+0100 (W. Europe
Standard Time)");
```

– `izlaz:` Wed Mar 25 2015 09:56:24 GMT+0100 (Central Europe Standard Time)

```
var d = new Date("Fri Mar 25 2015 09:56:24 GMT+0100 (Tokyo
Time)");
```

– `izlaz:` **Wed** Mar 25 2015 09:56:24 GMT+0100 (Central Europe Standard Time)

# Metode datum-a

| Metoda | Opis |
|---|---|
| `getDate()` | Dan kao broj (1-31) |
| `getDay()` | Dan u nedelji kao broj (0-6), ponedeljak = 1 |
| `getFullYear()` | Godina, 4 cifre (yyyy) |
| `getHours()` | Sat (0-23) |
| `getMilliseconds()` | Milisekunde (0-999) |
| `getMinutes()` | Minuti (0-59) |
| `getMonth()` | Meseci (0-11) |
| `getSeconds()` | Sekunde (0-59) |
| `getTime()` | Vreme u milisekundama od 1.1.970. |

# Metode datum-a

| Metoda | Opis |
|---|---|
| `setDate()` | Postavi dan kao broj (1-31)<br>`var d = new Date();`<br>`d.setDate(d.getDate() + 50);` |
| `setFullYear()` | Postavi godinu |
| `setHours()` | Postavi sate (0-23) |
| `setMilliseconds()` | Postavi milisekunde (0-999) |
| `setMinutes()` | Postavi minute (0-59) |
| `setMonth()` | Postavi mesec (0-11) |
| `setSeconds()` | Postavi sekunde (0-59) |
| `setTime()` | Postavi vreme u milisekundama od 1.1.970. |

# Date.parse

- Metoda **parse** vraća broj milisekundi od nultog datuma (1.1.1970):

```
<script>
var msec = Date.parse("March 21, 2012");
document.getElementById("demo").innerHTML = msec;
</script>
```

- Može se kreirati i datum prosleđujući mu kao parametar broj milisekundi (od 1.1.1970):

```
<script>
var msec = Date.parse("March 21, 2012");var d = new Date(msec);
document.getElementById("demo").innerHTML = d;
</script>
```

- Poređenje datuma:

```
<!DOCTYPE html>
<html><body><p id="demo"></p><script>
var today, someday, text;    today = new Date();
someday = new Date();someday.setFullYear(2100, 0, 14);
if (someday > today) text = "Today is before January 14, 2100.";
else                 text = "Today is after January 14, 2100.";
document.getElementById("demo").innerHTML = text;
</script></body></html>
```

# Niz: sort, length, dodavanje elementa

- Niz se može sortirati u alfabet poretku metodom sort:

```
<!DOCTYPE html>
<html><body>
<p id="demo"></p>
<script>
var person = ["John", "Doe", 46];
person.sort();
document.getElementById("demo").innerHTML = person;
</script></body></html>
```

46,Doe,John

- Svojstvo length se odnosi na broj elemenata niza:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.length;   // 4
```

- Dodavanje elementa u niz:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[fruits.length] = "Lemon";//dodavanje elementa na kraj
```

- Paziti na stvaranje nedefinisane rupe u nizu:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[10] = "Lemon";   // paziti na ovo
```

# Kretanje kroz niz

- Kretanje kroz niz:

```
var index;
var fruits = ["Banana", "Orange", "Apple", "Mango"];
for (index = 0; index < fruits.length; index++) {
    text += fruits[index];
}
```

- JavaScript NE podržava asocijativni niz, te ako koristite asocijativan pristup JavaScript će redefinisati niz u standardni objekat pri čemu će metode i svojstva ovog "niza" dati nedefinisane ili nekorektne rezultate:

```
var person = [];
person[0] = "John";
person[1] = "Doe";
var x = person.length;  // vraca 2
var y = person[0];      // ok
```

```
var person = [];
person["firstName"] = "John";
person["lastName"] = "Doe";
var x = person.length;          // vraca 0
var y = person[0];              // undefined
```

# Kreiranje niza

- Preporuke su:

```
var points = new Array();          // NOK
var points = [];                   // OK
```

```
var points = new Array(40, 100, 1, 5, 25, 10);  // NOK
var points = [40, 100, 1, 5, 25, 10];           // OK
```

- Paziti na sledeće:

```
var points = new Array(40, 100);  // kreira niz od 2 elementa
                                  // (40 i 100)
var points = new Array(40); // kreira niz od 40 nedefinisanih
                            // elemenata
```

- Provera da li je nešto niz:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
typeof fruits; //typeof ne vredi jer kaze da je u pitanju object
```

koristiti sledeće:

```
function isArray(myArray) {
    return myArray.constructor.toString().indexOf("Array")>-1;
}
```

ako je u pitanju niz onda je u stringu "function Array() { [native code] }" indeks 9.

# Metode niza

- Vraćanje vrednosti odnosno stringa niza:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.valueOf();
ili
document.getElementById("demo").innerHTML = fruits.toString();
```

- Spajanje elemenata niza u string sa datim separatorom:

```
<p id="demo"></p>
<script>
var fruits = ["Banana", "Orange","Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.join(" * ");
</script>
// izlaz: Banana * Orange * Apple * Mango
```

- Uklanjanje i dodavanje elementa niza:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var novaduzinaniza= fruits.push("Kiwi");
// dodaje novi element na kraj niza i vraca novu duzinu niza
var izbacenje=fruits.pop(); // vraca izbaceni element sa kraja
                            // niza
```

# Metode niza

- Metoda **shift** izbacuje prvi element niza, a ostale šiftuje za jedno mesto ulevo:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var izbacenje = fruits.shift();  // ostaje: Orange,Apple,Mango
```

suprotno od shift radi metoda **unshift**

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var novaduzinaniza = fruits.unshift("Lemon");
// novi niz je: Lemon,Banana,Orange,Apple,Mango
```

- Promena elementa niza:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[0] = "Kiwi";
```

- Brisanje elementa niza:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
delete fruits[0]; // nulti element niza je undefined
```

- Metoda **splice** umeće nove elemente niza i/ili uklanja postojeće elemente niza:

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

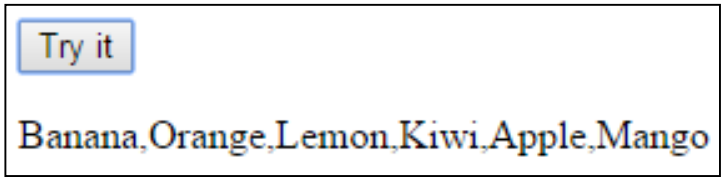
# Metode niza

- Splice parametri u primeru:

```
fruits.splice(2, 0, "Lemon", "Kiwi");
```

  - parametar 2 određuje indeks dodavanja novih elemenata
  - parametar 0 određuje koliko će elemenata od date pozicije biti uklonjeno
  - "Lemon", "Kiwi" su elementi koji će biti dodati

```
<!DOCTYPE html>
<html><body>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>


<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits;
function myFunction() {
    fruits.splice(2, 0, "Lemon", "Kiwi");
    document.getElementById("demo").innerHTML = fruits;
}
</script>
</body></html>
```

# Metode niza

- Uklanjanje elementa metodom splice:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(0, 1);  // uklanja element ciji je indeks 0
```

- Sortiranje i postavljanje obrnutog resleda niza:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.sort();       // sortiranje
fruits.reverse();    // obrtanje redosleda elemenata niza
```

- Sortiranje niza čiji su elementi brojevi korišćenjem funkcije poređenja:

```
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return a-b}); //rastuci redosled
// sada points[0] ima najmanju vrednost
points.sort(function(a, b){return b-a}); //opadajuci redosled
// sada points[0] ima najvecu vrednost
```

- Konkatenacija nizova:

```
var myGirls = ["Cecilie", "Lone"];
var myBoys  = ["Emil", "Tobias","Linus"];
var myChildren = myGirls.concat(myBoys);
```

# Metode niza

- Konkatenacija više nizova:

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

- Korišćenje slice za kreiranje novog niza:

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

***napomena:*** ide se do isključno drugog navedenog indeksa

  - ako se ne navede drugi argument podrazumevano se da se ide do kraja niza:

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

# Boolean

- Paziti na sledeće:

```
var x = 0;
Boolean(x);        // false
var x = -0;
Boolean(x);        // false
var x = "";
Boolean(x);        // false
var x;
Boolean(x);        // false
var x = null;
Boolean(x);        // false
var x = 10 / "H";
Boolean(x);        // false, NaN
```

- Ternarni operator

```
var odgovor = (godine >= 18) ?
                    "ima pravo da glasa"
                    :
                    "nema pravo da glasa";
```

# Probajte različite prekidače

- U sledećem primeru probajte različite vrednosti switch izraza

```
<!DOCTYPE html>
<html><body>
<p id="demo"></p>
<script>
var text="";
switch ("string")
{
    case "string":
    case 2:
    case 3:
    default:      text = "Prvi blok, default"; break;
    case 4:
    case "nesto": text = "Drugi blok";         break;
    case 0:
    case 6+6:
        text = "Treci blok";        break;
}
document.getElementById("demo").innerHTML = text;
</script>
</body></html>
```

# Labela

- Izlazak iz bloka označenog labelom

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
var cars = ["BMW", "Volvo", "Saab", "Ford"];
var text = "";
list: {
    text += cars[0] + "<br>";
    text += cars[1] + "<br>";
    text += cars[2] + "<br>";
    break list;
    text += cars[3] + "<br>";
    text += cars[4] + "<br>";
    text += cars[5] + "<br>";
}
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

# Svojstvo constructor

- Svojstvo constructor vraća konstruktorsku funkciju za sve JavaScript promenljive:

```
"John".constructor  //   function String()  { [native code] }
(3.14).constructor  //   function Number()  { [native code] }
false.constructor   //   function Boolean() { [native code] }
[1,2,3,4].constructor // function Array()   { [native code] }
{name:'John', age:34}.constructor
                      // function Object()  { [native code] }
new Date().constructor
                      // function Date()    { [native code] }
function(){}.constructor
                     //  function Function(){ [native code] }
```

- Na primer: ako se želi provera da li je dati objekat datum:

```
function isDate(myDate) {
    return myDate.constructor.toString().indexOf("Date") > -1;
}
```

# Konverzije

- Broj u string, efekat kao i kod metode toString:

```
String(x)          // string od x
String(123)        // string iz broja literala 123
String(100 + 23)   // string iz broja kao rezultata izraza
String(false)      // "false"
String(true)       // "true"
```

- String u broj:

```
Number("3.14")     // 3.14
Number(" ")        // 0
Number("")         // 0
Number("99 88")    // NaN
Number(false)      // 0
Number(true)       // 1
```

za datum:

```
d = new Date(); Number(d);
//ili
d = new Date(); var broj = d.getTime();
```

# Automatske konverzije

- Primeri automatskih konverzija:

```
5 + null     // 5          null se konvertuje u 0
"5" + null   // "5null"    null se konvertuje u "null"
"5" + 1      // "51"       1 se konvertuje u "1"
"5" - 1      // 4          "5" se konvertuje u 5
```

- Kada je potrebno izvršavaju se automatske konverzije u string metodom toString().

| Original | Konverzija u broj | Konverzija u String | Konverzija u Boolean |
|---|---|---|---|
| false | 0 | "false" | false |
| true | 1 | "true" | true |
| 0 | 0 | "0" | false |
| 1 | 1 | "1" | true |
| "0" | 0 | "0" | true |
| "1" | 1 | "1" | true |
| NaN | NaN | "NaN" | false |

# Automatske konverzije

| Original | Konverzija u broj | Konverzija u String | Konverzija u Boolean |
|---|---|---|---|
| Infinity | Infinity | "Infinity" | true |
| -Infinity | -Infinity | "-Infinity" | true |
| "" | **0** | "" | **false** |
| "20" | 20 | "20" | true |
| "twenty" | NaN | "twenty" | true |
| [ ] | **0** | "" | true |
| [20] | **20** | "20" | true |
| [10,20] | NaN | "10,20" | true |
| ["twenty"] | NaN | "twenty" | true |
| ["ten","twenty"] | NaN | "ten,twenty" | true |
| function(){} | NaN | "function(){}" | true |
| { } | NaN | "[object Object]" | true |
| null | **0** | "null" | **false** |
| undefined | NaN | "undefined" | **false** |

# Regularni izrazi

- Regularni izraz je sekvenca karaktera koja čini uzorak pretraživanja (**search pattern**)

  */pattern/modifiers*;

  primer: `var patt = /abc/i;`

  uzorak je abc, pri čemu modifikator je case-**i**nsensitive

- Konkretno:

```
var str = "Gde je X a gde Y?";
var n = str.search(/x/i); //7
```

  ako bi se koristio string umesto regularnog izraza:

```
var n = str.search("x"); //nije case-insensitive
```

- Korišćenje replace sa regularnim izrazom:

```
var str = "Visit Porsche center!";
var res = str.replace(/porsche/i, "Aero");
```

  ako bi se koristio string umesto regularnog izraza:

```
var res = str.replace("Porsche", "Aero");
```

# Regularni izrazi

- Modifikatori:
  - m izvršava mulitlinijsko poklapanje
  - i case-insensitive poklapanje
  - g globalno pretraživanje (nalazi sva poklapanja)
- Uzorci regularnih izraza:
  - [abc] nađi bilo koji od navedenih karaktera
  - [^abc] nađi karakter koji nije naveden
  - [0-9] nađi bilo koju navedenu cifru
  - [^0-9] nađi što nije navedeno
  - (x|y) nađi neku od alternativa odvojenih sa |
- Metakarakteri:
  - . nađi pojedinačni karakter osim znaka za novu liniju ili kraj linije
  - \w nađi slovo
  - \W nađi što nije slovo
  - \d nađi cifru
  - \D nađi što nije cifra

# Regularni izrazi

- \s nađi belinu
- \S nađi što nije belina
- \b nađi poklapanje na početku ili kraju reči
- \B nađi što nema poklapanje na početku ili kraju reči
- \0 nađi NUL karakter
- \t nađi znak tab
- \r nađi znak carriage- return
- \n nađi znak new-line
- \v nađi znak vertical-tab
- \f nađi znak form- feed
- \xxx nađi znak specificiran oktalnim brojem
- \xdd nađi znak specificiran heksadecimalnim brojem
- \uxxxx nađi specificirani Unicode karakter

- Kvantifikatori:
  - n? poklapanje sa stringom koji sadrži 0 ili jedan n
  - n* poklapanje sa stringom koji sadrži 0, 1 ili više pojavljivanja n
  - n+ poklapanje sa stringom koji sadrži bar jedan n

# Regularni izrazi

- n{X} pronalazi string koji sadrži sekvencu od X n-ova
- n{X,Y} pronalazi string koji sadrži sekvencu od X do Y n-ova
- n{X,} pronalazi string koji sadrži sekvencu od bar X n-ova
- ^n pronalazi string koji ima n na početku
- n$ pronalazi string koji ima n na kraju
- ?=n pronalazi string iza koga sledi n
- ?!n pronalazi string iza koga ne sledi n

- Svojstva objekta RegExp:
  - constructor vraća funkciju koja kreira prototip RegExp objekta
    - function RegExp() { [native code] }
  - multiline proverava da li postavljen "m" modifikator
  - ignoreCase proverava da li postavljen "i" modifikator
  - global proverava da li postavljen "g" modifikator
  - lastIndex specificira indeks od koga počinje poklapanje
  - source vraća tekst RegExp uzorka

# Regularni izrazi

- Metode RegExp objekta:
  - `compile()` zastarelo od verzije1.5, kompajlira regularni izraz
  - `exec()` testira poklapanje u stringu (vraća prvo poklapanje)

```
var str = "The best things in life are free";
var patt = new RegExp("e");
var res = patt.exec(str); //e
```

  - `test()` testira poklapanje u stringu (vraća true ili false)

```
var str = "The best things in life are free";
var patt = new RegExp("e");
var res = patt.test(str); //true
```

  - `toString()` vraća string uzorka

```
var patt = new RegExp("Hello World", "g");
var res = patt.toString();
izlaz je /Hello World/g
```

# Regularni izrazi

- Pokazano je da metod test testira da li postoji traženo poklapanje:

```
var patt = /e/;
var odgovor = patt.test("The best things in life are free!");
//true
```

može se primeniti i skraćeno pisanje:

```
/e/.test("The best things in life are free!");
```

- Metoda exec() je RegExp metoda koja pretražuje string za dati patern i vraća nađeni tekst.
  - Ako ne nađe poklapanje vraća null.

Npr. skraćeno pisanje, pretražuje se da li string sadrži karakter e

```
/e/.exec("The best things in life are free!");
```

Rezultat će biti:

e

# Upravljanje greškama, izuzecima

- JavaScript koristi paradigmu **try** (pokušaj da izvršiš kod), **catch** (obradi nastalu grešku) izbačenu klauzulom **throw** i nakon bloka try-catch izvrši **finally** blok.

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
try {
    adddlert("Welcome guest!");
}
catch(err) {
    document.getElementById("demo").innerHTML = err.message;
}
</script>
</body>
</html>
```

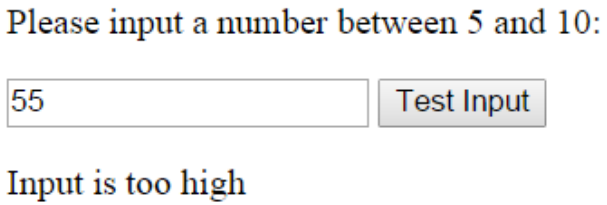
adddlert is not defined

- JavaScript baca (generiše) izuzetak datog tipa klauzulom throw:

```
throw "Too big";    // throw a text
throw 500;          // throw a number
```

# Validacija unosa

```
<!DOCTYPE html>
<html><body>
<p>Please input a number between 5 and 10:</p>
<input id="demo" type="number" min="5" max="10" step="1">
<button type="button" onclick="myFunction()">Test Input</button>
<p id="message"></p>
<script>
function myFunction() {
    var message, x;
    message = document.getElementById("message");
    message.innerHTML = "";
    x = document.getElementById("demo").value;
    try {
        if(x == "")     throw "empty";
        if(isNaN(x))    throw "not a number";
        x = Number(x);
        if(x < 5)       throw "too low";
        if(x > 10)      throw "too high";
    }
    catch(err) { message.innerHTML = "Input is " + err;    }
}</script></body></html>
```



# Finally

- Primer finally bloka, vezan za prethodni primer:

```
function myFunction() {
    var message, x;
    message = document.getElementById("message");
    message.innerHTML = "";
    x = document.getElementById("demo").value;
    try {
        x = Number(x);
        if(x == "") throw "is empty";
        if(isNaN(x)) throw "is not a number";
        if(x > 10) throw "is too high";
        if(x < 5) throw "is too low";
    }
    catch(err) {
        message.innerHTML = "Error: " + err + ".";
    }
    finally {
        document.getElementById("demo").value = "";
    }
}
```

# console

- JavaScript koristi metodu console.log za debagovanje.
- Built-in debugger u browseru aktivira se sa F12.
  - U debager meniju aktivirati tab Console

```
<!DOCTYPE html>
<html><body><h1>Ispisi 11 u Console</h1>
<script>
a = 5;
b = 6;
c = a + b;
console.log(c);
</script></body></html>
```

- Zaustavljanje izvršavanja JavaScript-a korišćenjem klauzule debugger:

```
<!DOCTYPE html>
<html><head></head><body><p id="demo"></p>
<script>
var x = 15 * 5;
debugger;
document.getElementById("demo").innerHTML = x;
</script></body></html>
```

# "use strict"

- Postavljanjem strikt moda nije dozvoljeno korišćenje nedeklarisanih promenljivih.
  - globalna deklaracija

```
"use strict";
x = 3.14;       // error
myFunction();   // error

function myFunction() {
    x = 3.14;
}
```

  - lokalna deklaracija

```
x = 3.14;
myFunction();   // error

function myFunction() {
   "use strict";
    x = 3.14;
}
```

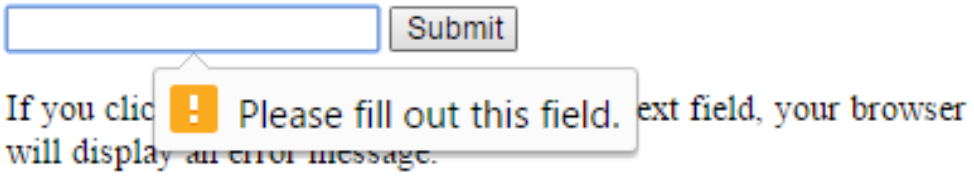
# Validacija

```
<!DOCTYPE html>
<html><head>
<script>
function validateForm() {
    var x = document.forms["myForm"]["fname"].value;
    if (x == null || x == "") {
        alert("Name must be filled out");
        return false;
    }
    return true;
}
</script>
</head><body>

<form name="myForm" action="demo_form.asp"
      onsubmit="return validateForm()" method="post">
Name: <input type="text" name="fname">
      <input type="submit" value="Submit">
</form>

</body></html>
```

# Validacija

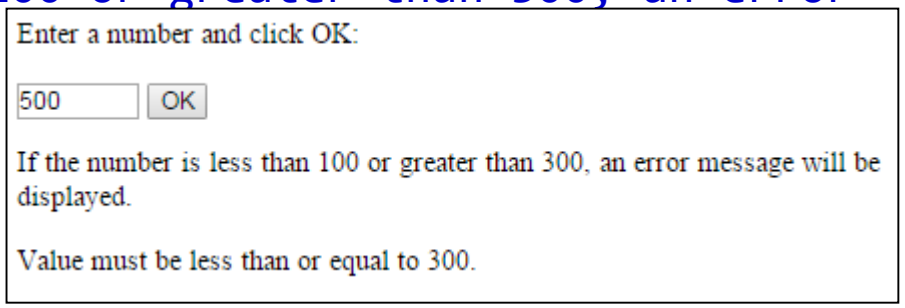
- Automatska validacija (rađeno ranije)

```
<!DOCTYPE html>
<html>
<body>

<form action="demo_form.asp" method="post">
  <input type="text" name="fname" required>
  <input type="submit" value="Submit">
</form>

<p>If you click submit, without filling out the text field,
your browser will display an error message.</p>

</body>
</html>
```



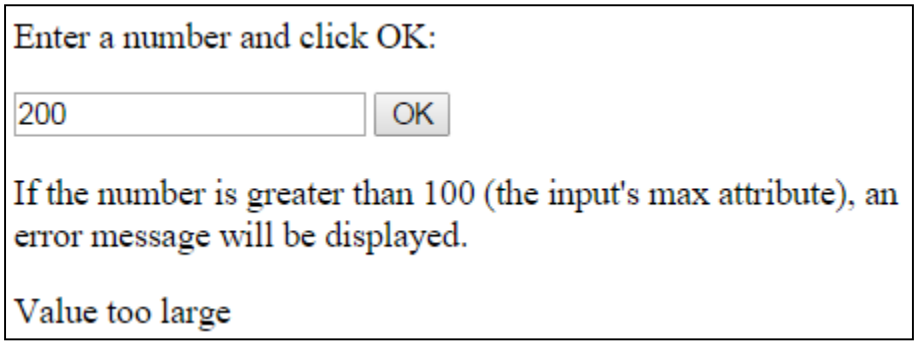
# Validacija

```
<!DOCTYPE html>
<html><body>
<p>Enter a number and click OK:</p>
<input id="id1" type="number" min="100" max="300">
<button onclick="myFunction()">OK</button>

<p>If the number is less than 100 or greater than 300, an error
message will be displayed.</p>

<p id="demo"></p>
<script>
function myFunction() {
    var inpObj = document.getElementById("id1");
    if (inpObj.checkValidity() == false) {
        document.getElementById("demo").innerHTML =
                                        inpObj.validationMessage;
    } else {
        document.getElementById("demo").innerHTML = "Input OK";
    }
}
</script></body></html>
```



# Validacija

```
<!DOCTYPE html>
<html><body>
<p>Enter a number and click OK:</p>
<input id="id1" type="number" max="100">
<button onclick="myFunction()">OK</button>
<p>If the number is greater than 100 (the input's max attribute),
an error message will be displayed.</p>
<p id="demo"></p>


<script>
function myFunction() {
    var txt = "";
    if (document.getElementById("id1").validity.rangeOverflow) {
        txt = "Value too large";
    } else {
        txt = "Input OK";
    }
    document.getElementById("demo").innerHTML = txt;
}
</script></body></html>
```



# Validacija

- Svojstvo willValidate pokazuje da li je element kandidat za validaciju:

```
<div                id="one"></div>
<input type="text" id="two" />
<input type="text" id="three" disabled />
<script>
    document.getElementById(  'one').willValidate;   //undefined
    document.getElementById(  'two').willValidate;   //true
    document.getElementById('three').willValidate;   //false
</script>
```

- Postavljanje provere validnosti:

```
<input id="foo" />
<input id="bar" />
<script>
    document.getElementById('foo').validity.customError; //false

    document.getElementById('bar').setCustomValidity('Invalid');
    document.getElementById('bar').validity.customError; //true
</script>
```

# Validacija

- Provera odstupanja uzorka, patternMismatch:

```
<input id="foo" pattern="[0-9]{4}" value="1234" />
<input id="bar" pattern="[0-9]{4}" value="ABCD" />
<script>
  document.getElementById('foo').validity.patternMismatch; //false
  document.getElementById('bar').validity.patternMismatch; //true
</script>
```

- Provera prekoračenja opsega preko gornje granice, rangeOverflow:

```
<input id="foo" type="number" max="2" value="1" />
<input id="bar" type="number" max="2" value="3" />
<script>
    document.getElementById('foo').validity.rangeOverflow; //false
    document.getElementById('bar').validity.rangeOverflow; //true
</script>
```

- Provera prekoračenja opsega ispod donje granice rangeUnderflow:

```
<input id="foo" type="number" min="2" value="3" />
<input id="bar" type="number" min="2" value="1" />
<script>
  document.getElementById('foo').validity.rangeUnderflow; //false
  document.getElementById('bar').validity.rangeUnderflow; //true
</script>
```

# Validacija

- Provera validnosti koraka, stepMismatch:

```
<input id="foo" type="number" step="2" value="4" />
<input id="bar" type="number" step="2" value="3" />
<script>
    document.getElementById('foo').validity.stepMismatch; //false
    document.getElementById('bar').validity.stepMismatch; //true
</script>
```

- Postavljanje poruke ne-validnosti, validationMessage:

```
<input type="text" id="foo" required />
<script>
    var strvm = document.getElementById('foo').validationMessage;
    //strvm = 'Please fill out this field.'
</script>
```

# Validacija

- Validacija vrednosti prema datom tipu ulaza, typeMismatch:

```
<input id="foo" type="url" value="http://foo.com" />
<input id="bar" type="url" value="foo" />

<input id="foo2" type="email" value="foo@foo.com" />
<input id="bar2" type="email" value="bar" />


<script>
    document.getElementById('foo').validity.typeMismatch; //false
    document.getElementById('bar').validity.typeMismatch; //true

    document.getElementById('foo2').validity.typeMismatch; //false
    document.getElementById('bar2').validity.typeMismatch; //true
</script>
```

# Validacija

- Validacija vrednosti kod atributa required, valueMissing:

```
<input id="foo" type="text" required value="foo" />
<input id="bar" type="text" required value="" />
<script>
    document.getElementById('foo').validity.valueMissing; //false
    document.getElementById('bar').validity.valueMissing; //true
</script>
```

- Validacija da li je sve što se validira ispravno, valid:

```
<input id="valid-1" type="text" required value="foo" />
<input id="valid-2" type="text" required value="" />
<script>
    document.getElementById('valid-1').validity.valid; //true
    document.getElementById('valid-2').validity.valid; //false
</script>
```

# Validacija

- Provera validnosti forme, checkValidity:

```
<form id="form-1">
    <input id="input-1" type="text" required />
</form>
<form id="form-2">
    <input id="input-2" type="text" />
</form>
<script>
    document.getElementById('form-1').checkValidity();  //false
    document.getElementById('input-1').checkValidity(); //false

    document.getElementById('form-2').checkValidity();  //true
    document.getElementById('input-2').checkValidity(); //true
</script>
```

- Postavljanje funkcije koja obrađuje događaj (EventListener-a):

```
document.getElementById('input-1').addEventListener('change',
function(event) {
 if (event.target.validity.valid) {//Field contains valid data. }
 else                             {//Field contains invalid data.}
}, false);
```

# Validacija

- Moguće je postaviti i korisnički tekst poruke validacije sa setCustomValidity():
- Primer provere potvrde lozinke:

```
if (document.getElementById('password1').value !=
    document.getElementById('password2').value)
{
document.getElementById('password1').setCustomValidity('Passwords
must match.');
}
else {
    document.getElementById('password1').setCustomValidity('');
}
```