

**ИНТЕРНЕТ ПРОГРАМИРАЊЕ  
ПРЕДАВАЊА – ПРЕЗЕНТАЦИЈА – 4  
ЈАВА СКРИПТ**

- 01-JAVA SCRIPT
- 02-SCRIPT TAG
- 03-POZIVANJE JAVA SCRIPTA
- 04-JAVA SCRIPT ISKAZI
- 05-SINTAKSA
- 06-JAVA SCRIPT–OSOBINE
- 07-JAVA SCRIPT-SINTAKSA
- 08-LET I CONST
- 09-JAVA SCRIPT MOGUCNOSTI
- 10-KOMENTARI
- 11-OPERATORI
- 12-ARITMETICKI OPERATORI
- 13-OPERATORI POREDJENJA
- 14-LOGICKI OPERATORI
- 15-OPERATORI NA NIVOU BITA
- 16-TERNARNI OPERATOR
- 17-TYPE OPERATORI
- 18-TIPOVI PODATAKA
- 19-KONTROLA TOKA
- 20-HTML DOM
- 21-DOCUMENT OBJECT
- 22-COOKIE
- 23-NEDOSTACI COOKIE-A
- 24-BOM
- 25-WINDOW OBJECT
- 26-POPUP BOX-OVI
- 27-RAD SA TIMING DOGADJAJIMA
- 28-SCREEN OBJEKAT
- 29-WINDOW OBJEKAT
- 30-LOCATION OBJEKAT
- 31-HISTORY OBJEKAT
- 32-STRING
- 33-NUMBER
- 34-ARRAY OBJEKAT
- 35-DATE OBJEKAT
- 36-MATH OBJEKAT
- 37-BOOLEAN OBJEKAT
- 38-PRIMJERI
- 39-FUNKCIJE
- 40-PRIMJERI FUNKCIJA
- 41-ARROW/ANONYMUS/LAMBDA FUNKCIJE/IZRAZI
- 42-OBJEKTI
- 43-DOGADJAJI
- 44-PRIMJERI ZA OBJEKTE I DOGADJAJE
- 45-STRICT MODE
- 46-HOISTING
- 47-JSON
- 48-KLASE

49-RAD SA FORMAMA  
50-RAD SA GRESKAMA  
51-DEBUGGING  
52-DOBRE PREPORUKE U PRAKSI

FIRE BUG  
WEB DEVELOPER  
DEVELOPER TOOLS

## JAVA SCRIPT

**obično se koristi za:**

- dinamičke promjene sadržaja HTML strana
- manipulaciju slikama
- validaciju formi

**besplatan**

**platformski neutralan**

**sintaksa slična Java programskom jeziku**

**nema tipove podataka**

- kod deklaracije promenljivih se ne navodi tip (dinamički se dodjeljuje)

**ugrađene funkcije**

**JavaScript je OO jezik**

**ugrađeni objekti**

- String, Number, Boolean, Array, Object, Null, Undefined, Date, Math, RegExp, ...

**kreiranje vlastitih objekata**

**sistem događaja**

## SCRIPT TAG

tag **<script>** specificira Script kod koji se pokreće direktno u browser-u

browser sve između tagova **<script>** i **</script>** smatra elementima skripta

tag **script** se može javiti bilo gdje u HTML dokumentu, u head-u, u body-ju ili i u heād-u i u body-ju

funkcije definisane u tagu **<script>** u zaglavlju dokumenta, mogu se pozivati bilo gdje u dokumentu

JavaScript programski kod može se naći i u eksternom dokumentu (apsolutne i relativne adrese) – prednosti:

- razdvajanje HTML i JS koda
- jadnostavnije održavanje
- keširanje

moguće je jednoj HTML stranici dodijeliti više JS datoteka

```
<script src="file1.js"></script>
<script src="file2.js"></script>
```

## POZIVANJE JAVA SCRIPTA

▶ kao reakcija na neki događaj

- unutar **<script>** taga bilo gdje unutar HTML dokumenta
- unutar **<body>** taga
- unutar **<head>** taga
  - ako koristimo JavaScript funkciju, nju moramo da definišemo unutar **<head>** taga da bismo mogli da je pozivamo iz bilo kog JavaScript koda

▶ kao adresu unutar **<a>** taga:

```
<a href="javascript:funkcija('parametar');"> klikni</a>
```

▶ u eksternoj datoteci

```
<head>
<script src="javascript.js"></script>
</head>
```

## JAVASCRIPT ISKAZI

JS iskazi sastoje se od vrijednosti, operatora, izraza, ključnih riječi i komentara

JS koristi Unicode karakter set

JS iskazi se razdvajaju znakom ;

- ovo nije obavezno, ali se preporučuje

JS ignoriše whitespace-ove

- koristiti ih za fino formatiranje koda

blokovi koda označeni vitičastim zagradama

## SINTAKSA

### **promjenljive**

- mogu biti:
  - globalne (izvan funkcije, sve f-je je vide)
  - lokalne (unutar funkcija)
- nemaju tip
- ključna riječ var deklariše lokalnu promjenljivu
  - nije obavezna – ako se varijabla deklariše bez ključne riječi var onda će imati global scope (postaće property window objekta)
- ključna riječ const deklariše konstantu
- ključna riječ let deklariše promjenljivu sa ograničenim opsegom vidljivosti
- dodjela vrijednosti – operator “=”
- radi sa dinamičkim tipovima – ista promjenljiva se može koristiti da čuva vrijednosti različitih tipova

### **operatori**

- slični operatorima programskog jezika Java/C++

### **kontrola toka**

- if, for, while,...

## JAVASCRIPT OSNOVNE OSOBINE

**JavaScript je jednostavniji programski jezik**

- ne zahtjeva se, i nije dozvoljeno, deklarisati tip podataka promjenljive

**JavaScript interpreter:**

- automatski prepoznaje koji tip podataka je smješten u okviru promjenljive
- dinamički dodeljuje tip podatka promjenljivoj

**JavaScript je case sensitive jezik**

```
var x;           // undefined
var x = 5;       // number
var x = "x";     // string
```

## OSNOVI SINTAKSE JAVASCRIPTA – STILIZOVANJE

**pravila i konvencije pri imenovanju promjenljivih (identifikatora):**

- identifikator mora počinjati slovom, znakom dolar (\$), ili donjom crtom (\_)
- u okviru imena se mogu koristiti brojevi, ali ne kao prvi karakter
- ne mogu se koristiti prazna mjesta u okviru imena
- ključne riječi ne mogu da se koriste kao imena promjenljivih (rezervisane riječi):
  - var, return, switch, break, continue, if, else, debugger, do, while, for, function, try, catch, ...
- nazivi su *case sensitive*
  - notacija – preporuka je Lower Camel Case
    - firstName, secondName

## LET I CONST

- let i const obezbjeđuju blok opseg vidljivosti
- promjenljive deklarisane van funkcija imaju globalni opseg vidljivosti
  - i var i let i const
- promjenljive deklarisane unutar funkcija imaju opseg vidljivosti funkcije
  - i var i let i const
- var deklarisana promjenljiva ne može imati blok opseg vidljivosti – let može
- u HTML-u, globalni opseg je window objekat
  - var deklarisana promjenljiva pripada window objektu, a let deklarisana promjenljiva ne pripada
- const deklarisana promjenljiva se ponaša slično kao i let deklarisana promjenljiva, s tim što joj se vrijednost ne može više puta dodijeliti
- const deklarisanoj promjenljivoj vrijednost mora biti dodijeljena prilikom inicijalizacije
- const definiše konstantnu vrijednost reference
  - primitivne vrijednosti ne mogu biti promijenjene, ali vrijednost atributa objekta mogu
  - elementi niza se takođe mogu mijenjati

```
{  
  let x = 2;  
}  
// x can NOT be used here
```

```
{  
  var x = 2;  
}  
// x CAN be used here
```

## JAVA SCRIPT MOGUCNOSTI

- Može dinamički ugraditi sadržaj u HTML stranicu
  - document.getElementById("test").innerHTML = "JavaScript";
- Može mijenjati vrijednost atributa HTML tagova
- Može mijenjati CSS stilove
- Može sakriti/prikazati HTML elemente
- Može se koristiti za validaciju unesenih podataka
- Može se koristiti za detekciju web čitača klijenta
- Može se koristiti za kreiranje kolačića
- Zvanično ime ECMAScript
- ECMA-262 je zvanični JavaScript standard (1997. godina)
- Razvija se i danas – ECMAScript 10 (ECMAScript 2019)

## KOMENTARI

za jednolinijski komentar – „//”

// komentar u jednoj liniji ...

za komentar više redova – „/\*” za početak bloka pod komentаром i „\*/” za kraj bloka pod komentаром:

```
/*
komentar u više redova...
*/
```

## OPERATORI

Aritmetički  
Na nivou bita  
Relacionalni  
Logički

### ARITMETICKI OPERATORI

- + sabiranje
- + = sabiranje i dodjela
- oduzimanje
- = oduzimanje i dodjela
- \* množenje
- \* = množenje i dodjela
- / dijeljenje
- / = dijeljenje i dodjela
- % moduo
- % = moduo i dodjela
- \*\* eksponent
- \*\* = eksponent i dodjela
- ++ inkrement
- dekrement

## ARITMETICKI OPERATORI KOD STRINGOVA

-  
+ sabiranje  
+= sabiranje i dodjela  
- oduzimanje  
-- oduzimanje i dodjela  
\* množenje  
\*= množenje i dodjela  
/ dijeljenje  
/= dijeljenje i dodjela  
% moduo  
%= moduo i dodjela  
\*\* eksponent  
\*\*= eksponent i dodjela  
++ inkrement  
-- dekrement

operator + kod stringova  
◦ dva stringa – konkatenacija  
◦ string i broj – string  
operator +=

## OPERATORI POREDJENJA

jednakost (**==**) rezultat je true ako su operandi jednaki  
nejednakost (**!=**) rezultat je true ako su operandi različiti

veće (**>**) rezultat je true ako je lijevi operand veći od desnog operanda  
veće ili jednako (**>=**) rezultat je true ako je lijevi operand veći ili jednak desnom operandu

manje (**<**) rezultat je true ako je lijevi operand manji od desnog  
operanda

manje ili jednako (**<=**) rezultat je true ako je lijevi operand manji ili  
jednak desnom operandu

jednako bez konverzije tipova (**====**) rezultat je true ako su operandi  
jednaki bez konverzije podataka – jednakost vrijednosti i jednakost tipa  
različito bez konverzije tipova (**!==**) rezultat je true ako su operandi  
različiti bez konverzije podataka – vrijednost ili tip nisu jednaki

kod poređenja stringova i brojeva, string se uvijek konvertuje u broj (ako  
je to moguće), pa se onda radi poređenje

- ako je u pitanju prazan string, on se konvertuje u **0**
- ako je u pitanju string koji se ne može konvertovati u broj (a nije ni prazan), onda se  
konvertuje u **Nan** – tada je rezultat svakog poređenja **false**

## LOGICKI OPERATORI

I (**&&**) – **expr1 && expr2** – rezultat je true, jedino ako su oba operanda true, u ostalim slučajevima rezultat je false

ILI (**||**) – **expr1 || expr2** – rezultat je false, jedino ako su oba operanda false, u ostalim slučajevima rezultat je true

NE (!) – **!expr** – komplement vrijednosti operanada. Ako je operand true, rezultat je false, ako je operand false, rezultat je true

## OPERATORI ZA RAD NA NIVOU BITA

logičko I (AND) – **a & b** – rezultat je 1, jedino ako su oba bita 1, u ostalim slučajevima rezultat je 0

logičko ILI (OR) – **a | b** – rezultat je 0, jedino ako su oba bita 0, u ostalim slučajevima rezultat je 1

logičko eksluzivno ILI (XOR) – **a ^ b** – rezultat je 1, ako biti imaju različite vrijednosti, u slučaju da imaju iste vrijednosti, rezultat je 0

logičko NE (NOT) – **~ a** – komplementira bitove operanda a  
pomjeranje uljevo – **a << b** – pomjera binarni sadržaj operanda a za b mjesta uljevo. Prazna mjesta popunjava sa vrijednošću 0  
pomjeranje udesno sa znakom – **a >> b** – pomjera binarni sadržaj operanda a za b mjesta udesno. Prazna mjesta popunjavaju se sa vrijednošću najstarijeg bita.

pomjeranje udesno sa nulama – **a >>> b** – sadržaj operanda a pomjera se za b mjesta udesno. Prazna mjesta popunjavaju se vrijednošću 0.

## TERNARNI OPERATOR

**expression ? statement1 : statement2**

izraz *expression* je bilo koji izraz čiji rezultat je vrijednost logičkog tipa

ako je rezultat izraza true, onda se izvršava *statement1*, u suprotnom *statement2*

## TYPE OPERATORI

**typeof** – vraća tip promjenljive

**instanceof** – vraća true ako je objekat instanca nekog tipa

### **atribut constructor**

- vraća konstruktor funkciju za sve JavaScript promjenljive
  - "abc".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]}
  - {a:'abc', d:"def"}.constructor // function Object() {[native code]}
  - new Date().constructor // function Date() {[native code]}
  - function () {}.constructor // function Function() {[native code]}

### **konverzije tipova**

- automatske
- pomoću funkcija
  - String(), Number()...

## PRETVORBA/KONVERZIJA TIPOVA

Originalna vrijednost	Konverzija		
	u Number	u String	u Boolean
FALSE	0	"false"	FALSE
TRUE	1	"true"	TRUE
0	0	"0"	FALSE
1	1	"1"	TRUE
"0"	0	"0"	TRUE
"000"	0	"000"	TRUE
"1"	1	"1"	TRUE
NaN	NaN	"NaN"	FALSE
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(){} {} null undefined	NaN	"function(){}" "[object Object]" "null" "undefined"	TRUE

## TIPOVI PODATAKA

JS radi sa dinamičkim tipovima – ista promjenljiva se može koristiti tako da čuva vrijednosti različitih tipova stringovi – sekvenca karaktera između jednostrukih ili dvostrukih znakova navoda

- tekst="neki tekst";
- tekst='neki tekst';

brojevi – sa ili bez decimala

boolean – true ili false

nizovi – uglaste zagrade

objekti – vitičaste zagrade

- var student = {firstName:"Marko", secondName:"Markovic", age:20};

primitivni tipovi (može ih vratiti typeof operator):

- string, number, boolean, undefined, bigint, simbol
- wrapper-i za primitivne tipove: String, Number, BigInt, Boolean, Symbol

složeni tipovi

funkcije i objekti

## KONTROLA TOKA

if

if else

else if

switch

- za poređenje koristi ===
- brake, default

petlje

◦ while

◦ do while

◦ for

◦ for in

◦ for of

break

continue

with

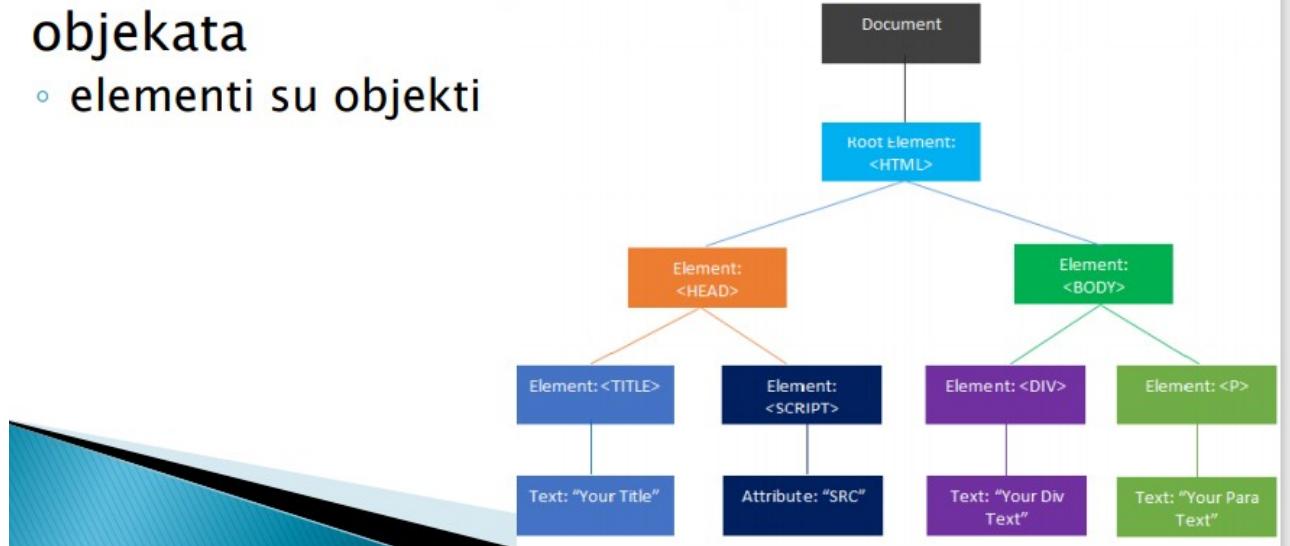
## HTML DOM

### DOM – Document Object Model

kada se stranica učita, web čitač kreira njen  
DOM model

HTML DOM model predstavljen je kao stablo  
objekata

- elementi su objekti



### JavaScript može:

- mijenjati sve HTML elemente na stranici
- mijenjati sve HTML atribute na stranici
- mijenjati sve CSS stilove na stranici
- ukloniti postojeće HTML elemente i atribute
- dodati nove HTML elemente i atribute
- reagovati na sve postojeće HTML događaje
- kreirati nove HTML događaje na stranici

## DOCUMENT OBJEKAT

**predstavlja tekuću web stranicu**  
**metode za pronalaženje HTML elemenata**

- `document.getElementById(id)`
- `document.getElementsByTagName(name)`
- `document.getElementsByClassName(name)`

**izmjene elementa**

- `element.innerHTML`
- `element.getAttribute`
- `element.style.property`
- `element.setAttribute(attribute, value)`

**dodavanje i brisanje elemenata**

- `document.createElement(element)`
- `document.removeChild(element)`
- `document.appendChild(element)`
- `document.replaceChild(new, old)`
- `document.write(text)`

**dodavanje *event handler-a***

`element.onclick = function(){code}`

**metode:**

- `open()` – otvara tok podataka za write metodu
- `close()` – zatvara tok podataka
- `write()` – ispisuje tekst na ekran
- `document.addEventListener()`, `document.adoptNode(node)`,  
`document.createAttribute()`, `document.createComment()`,  
`document.createDocumentFragment()`, `document.createElement()`,  
`document.createTextNode()`, `document.getElementById()`,  
`document.getElementsByName()`, `document.getElementsByTagName()`,  
`document.getElementsByClassName()`, `document.importNode()`, `document.normalize()`,  
`document.normalizeDocument()`, `document.querySelector()`,  
`document.querySelectorAll()`, `document.removeEventListener()`,  
`document.renameNode()`, `document.writeln()`

**atributi:**

- `forms` – niz svih formi u dokumentu
- `links` – niz svih linkova u dokumentu
- `applets` – niz svih apleta u dokumentu
- `title` – sadrzaj title taga
- `URL` – kompletan URL dokumenta
- `document.anchors`, `document.baseURI`, `document.body`, `document.cookie`,  
`document.doctype`, `document.documentElement`, `document.documentElementMode`,  
`document.documentElementURI`, `document.domain`, `document.domConfig`,  
`document.embeds`, `document.head`, `document.images`, `document.implementation`,  
`document.inputEncoding`, `document.lastModified`,  
`document.readyState`, `document.referrer`, `document.scripts`,  
`document.strictErrorChecking`

## COOKIE

Cookie je tekstualni fajl koji se može zapamtiti na računaru klijenta

Format koji cookie fajl mora da zadovolji je:

```
ime=vrijednost  
[;EXPIRES=datum]  
[;DOMAIN=imeDomena]  
[;PATH=putanja]  
[;SECURE]
```

ime – ime koje definiše upisani cookie

vrijednost – je informacija koja je potrebno zapamtiti  
datum – je datum koji definiše do kada cookie ostaje  
upisan na klijentskoj mašini

imeDomena – definiše jedini domen sa kojeg cookie  
može da se pročita i sa kojeg može da mu se mijenja  
vrijednost

putanja – definiše jedinu putanju sa koje cookie  
može da se pročita i sa koje može da mu se mijenja  
vrijednost

SECURE – upis i čitanje cookie se izvršava preko  
sigurnih linija

opcije EXPIRES, DOMAIN, PATH, SECURE su opcione i  
nije bitan redoslijed u kojem se pojavljuju

cookie može biti kreiran, pročitan i obrisan  
korišćenjem JavaScript-a

dostupni putem document.cookie

primjer

- `document.cookie = 'cookie1=testcookie;  
expires=Sat, 30 Nov 2019 20:47:11 UTC; path=/'`
- čitanje: `cookie1=testcookie`
  
- `document.cookie = 'cookie2=testcookie2;  
expires=Sat, 30 Nov 2019 20:47:11 UTC; path=/'`
- čitanje: `cookie1=testcookie; cookie2=testcookie2`

## NEDOSTATCI COOKIE-A

nepouzdana identifikacija – cookie ne identificuje osobu, već kombinaciju korisničkog naloga, računara i web čitača  
**cookie hijacking**

- krađa cookie-a:
  - presretanjem saobraćaja
  - cross-site scripting

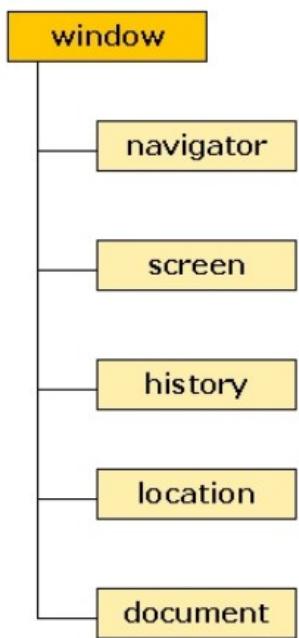
**cookie poisoning**

nekonzistentno stanje na klijentu i serveru

životni vijek cookie-a

## BOM

**BOM – Browser Object Model**  
omogućava JavaScript kodu da komunicira sa web čitačem



## WINDOW OBJECT

omogućava manipulaciju prozorima  
sadrži informacije o tekućem prozoru  
sadrži i document objekat  
svi JS globalni objekti, sve globalne metode i globalne promjenljive postaju članovi window objekta

- globalne promjenljive postaju atributi window objekta
- globalne funkcije postaju metode window objekta

Neke od metoda:

- alert(), atob(), blur(), btoa(), clearInterval(), clearTimeout(), close(), confirm(), createPopup(), focus(), moveBy(), moveTo(), open(), print(), prompt(), resizeBy(), resizeTo(), scroll(), scrollBy(), scrollTo(), setInterval(), setTimeout(), stop()

atributi:

- history – istorija odlazaka na stranice,
- document – tekući HTML dokument,
- frames – niz svih frejmova u prozoru,
- location – kompletan URL tekuće stranice,
- screen – predstavlja ekran objekat povezan sa prozorom,
- closed, defaultStatus, innerHeight, innerWidth, length, name, navigator, opener, outerHeight, outerWidth, pageXOffset, pageYOffset, parent

## POP-UP BOX-OVI

### Alert Box

- alert("sometext");

### Confirm Box

- confirm("sometext");

### Prompt Box

- prompt("sometext", "defaultvalue");

```
let x = 1
console.log("x:", x)
```

```

<!DOCTYPE html>
<html>
<body>

<h1>The dialog element</h1>

<p>This is some text.</p>
<p>This is some text.</p>
<dialog open>This is an open dialog window</dialog>
<p>This is some text.</p>
<p>This is some text.</p>
<p><b>Note:</b> The dialog tag is not supported in Safari and Edge (prior version 79).</p>
</body>
</html>

```

## The dialog element

This is some text.

This is some text.

This is some text.

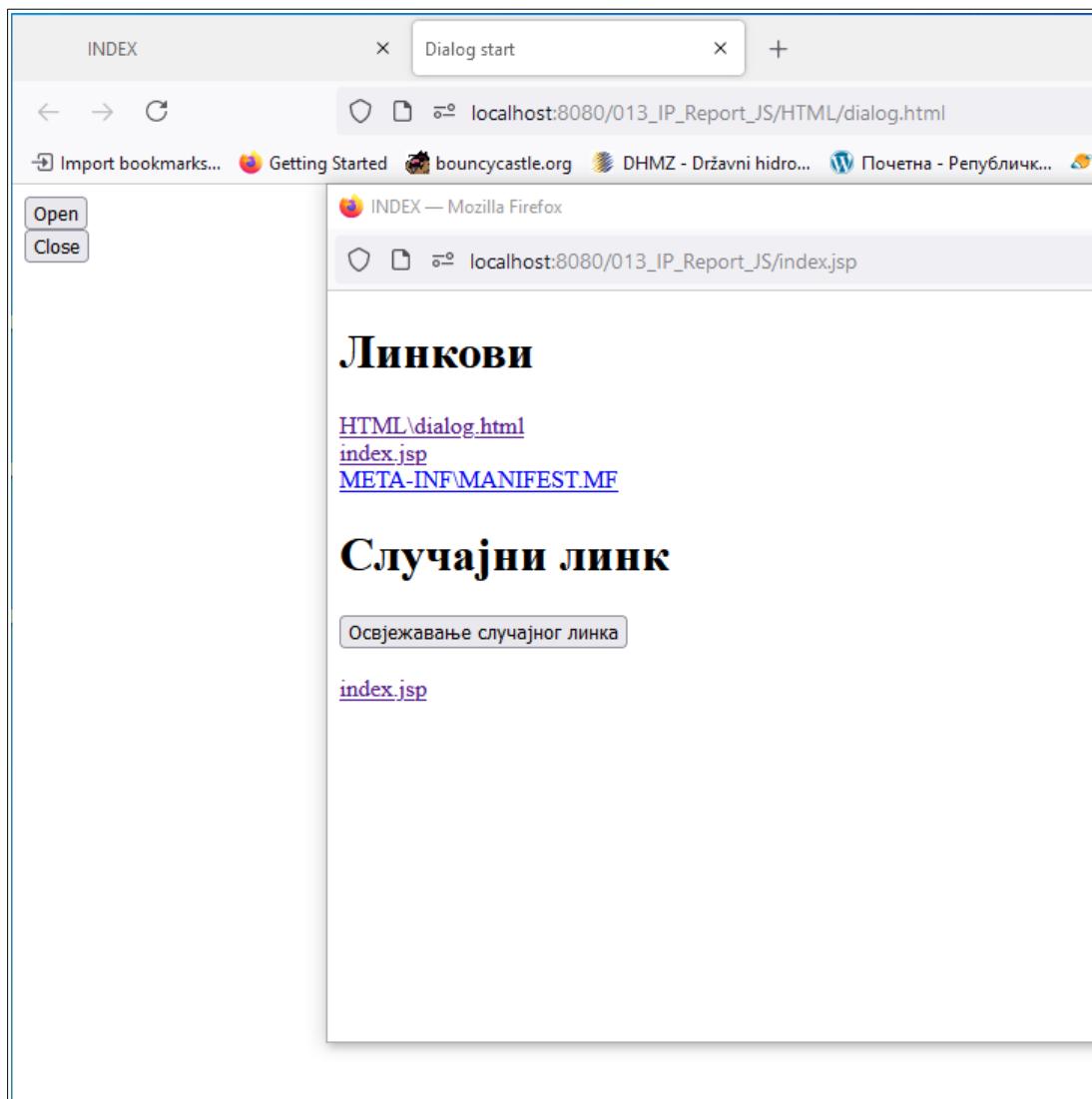
This is some text.

This is an open dialog window

**Note:** The dialog tag is not supported in Safari and Edge (prior version 79).

The screenshot shows a web browser window with the URL <https://www.resultati.com/tenis/atp-singl/wimbledon/>. The page is titled "ATP Wimbledon" and features a sidebar with navigation links for US Open, Wimbledon, Australian Open, French Open, US Open, and Wimbledon. Under "Moje ekipa", there are links for ATP Singlovi, WTA Singlovi, ATP Singlovi Race, WTA Singlovi Race, ATP Parovi, WTA Parovi, ATP Parovi Race, and WTA Parovi Race. The main content area shows the "Zadnji rezultati" (Recent Results) for the ATP - SINGL: Wimbledon (Ujedinjeno Kraljevstvo), trava. The final match listed is between Djokovic N. (Srb) and Berrettini M. (Ita) on 11.07. 15:10, with Djokovic winning 3-1. Below this, the semifinal and quarter-final results are shown.

This screenshot shows the same ATP Wimbledon results page as the previous one, but it has been zoomed in on the final match between Djokovic and Berrettini. The right side of the screen displays a detailed view of the match, including player profiles, scores (3-1), and match statistics. The sidebar and other content on the left remain the same.



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Dialog start</title>
6     <script type='text/javascript'>
7       var win;
8       function openD(){
9         try{
10           if(typeof(win)=='undefined' || win==null)
11             win = window.open("http://localhost:8080/013_IP_Report_JS/index.jsp", "dlg", "width=700,height=500", "pizza", 6.98);
12           }catch(err){
13             win=null;
14           }
15       }
16       function closeD(){
17         try{
18           if(typeof(win)!='undefined' && win!=null)
19             win.close();
20           win = null;
21           }catch(err){
22             alert(err);
23           }
24         }
25     </script>
26   </head>
27   <body>
28     <button onclick='openD()'>Open</button><br>
29     <button onclick='closeD()'>Close</button>
30   </body>
31 </html>
```



```
<head>
<base href="https://etf.unibl.org/" />
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<meta name="description" content="Електротехнички факултет, Faculty of Electrical Engineering" />
<meta name="generator" content="Joomla! - Open Source Content Management" />
<title>Почетна</title>
```

```
<link href="https://etf.unibl.org/index.php/sr-RS/" rel="alternate" hreflang="sr-RS" />
<link href="https://etf.unibl.org/index.php/en/" rel="alternate" hreflang="en-GB" />
<link href="/favicon.ico" rel="shortcut icon" type="image/vnd.microsoft.icon" />
<link href="/media/com_attachments/css/attachments_hide.css" rel="stylesheet" type="text/css" />
<link href="/media/com_attachments/css/attachments_list.css" rel="stylesheet" type="text/css" />
<link href="/media/jui/css/bootstrap.min.css" rel="stylesheet" type="text/css" />
<link href="/media/jui/css/bootstrap-responsive.css" rel="stylesheet" type="text/css" />
<link href="/templates/etf_template/css/style.css" rel="stylesheet" type="text/css" />
```

## РАД СА ТАЈМИНГ/ВРЕМЕНСКИМ ДОГАЂАЈИМА

### metode window objekta

#### **setTimeout(function, milliseconds)**

- izvršava funkciju nakon specificiranog vremenskog perioda

#### **setInterval(function, milliseconds)**

- kao i setTimeout, samo što se periodično ponavlja

#### **clearTimeout()**

#### **clearInterval()**

### pokretanje

- var t=setTimeout("alertMsg()",3000);

### zaustavljanje

- clearTimeout(t);

## SCREEN ОБЈЕКАТ

sadrži informacije o ekranu  
atributi:

- screen.width
- screen.height
- screen.availWidth
- screen.availHeight
- screen.colorDepth
- screen.pixelDepth

## LOCATION ОБЈЕКАТ

reprezentuje URL učitane stranice

- location = “<http://www.google.com>”

sadrži informacije o tekućem dokumentu

metode:

- assign() – učitavanje novog dokumenta
- reload() – ponovno učitavanje tekućeg dokumenta
- replace() – učitava novi dokument i briše URL iz istorije (back() ne radi)

atributi:

- href – URL tekuće stranice
  - location.href=“<http://www.google.com>”
- protocol – protokol iz URL-a
- host – adresa servera iz URL-a + port iz URL-a
- port – port iz URL-a
- pathname – putanja do resursa
- search – parametri forme
- hash – vraća anchor dio (#) URL-a
- hostname – adresa servera iz URL-a
- origin – protokol iz URL-a + adresa servera iz URL-a + port iz URL-a

### HISTORY ОБЈЕКАТ

omogućava pristup ranije posjećenim stranicama

sadrži listu adresa posjećenih stranica  
metode:

- back() – učitava prethodnu stranicu iz liste
- forward() – učitava sljedeću stranicu iz liste
- go() – učitava zadatu adresu iz liste

atributi:

- length – broj stavki u history listi

### NAVIGATOR ОБЈЕКАТ

sadrži informacije o web čitaču korisnika

atributi:

- navigator.appName
- navigator.appCodeName
- navigator.platform
- navigator.appVersion
- navigator.cookieEnabled
- navigator.userAgent
- navigator.platform
- navigator.language
- navigator.onLine

## STRING

string predstavlja proizvoljan niz karaktera između dvostrukih ("neki tekst") ili jednostrukih ('neki tekst') znakova navoda

- var s= new String("tekst"); // typeof -> object
- var s= "tekst"; // typeof -> string
- funkcija String("tekst") // typeof -> string
- radi konverziju tipa (u string iz object)

atributi:

- *length* - dužina stringa

operator +

- x="5"+5; – x sadrži string vrijednost

metode i atributi su dostupni primitivnom tipu, jer ih u trenutku izvršavanja metoda ili pristupa atributima JS tretira kao objekte

*substring()* – vraća dio stringa

*substr()* – drugi parametar je dužina stringa

*slice()* – prihvata negativne indeksne vrijednosti (indeksne pozicije se računaju od kraja stringa)

*split()* – vraća niz stringova kao rezultat "razdvajanja" stringa

*indexOf()*, *lastIndexOf()* – vraća poziciju nekog podstringa

*charAt()* – vraća karakter sa zadate pozicije

*search()* – vraća poziciju zadatog stringa

*replace()* – mijenja dio stringa

*concat()* – konkatenacija

*trim()* – uklanjanje whitespace-a sa početka/kraja stringa

*toUpperCase()*, *toLowerCase()*

*padStart()*, *padEnd()* – padding na početak/kraj stringa

## NUMBER

brojevi mogu biti i objekti

nije potrebno kreirati objekte – potencijalno komplikuje kod (poređenje objekata, vrijednosti,...) i usporava izvršavanje koda

metode:

- `toString()`
- `toExponential()`
- `toFixed()`
- `toPrecision()`
- `valueOf()`

konverzija u broj

- `Number()`
- `parseFloat()`
- `parseInt()`

atributi

- `MAX_VALUE`
- `MIN_VALUE`
- `NEGATIVE_INFINITY`
- `NaN`
- `POSITIVE_INFINITY`

## ARRAY OBJEKAT

```
var cars=new Array();
cars[0]="Audi";
cars[1]="Volvo";
cars[2]="BMW";
```

```
var cars=new Array("Audi","Volvo","BMW");
```

```
var cars=["Audi","Volvo","BMW"];
```

elementi niza mogu biti i objekti, funkcije, kao i nizovi

atribut: *length*

metode:

- `push()`, `pop()`, `shift()`, `unshift()`, `isArray()`, `toString()`, `join()`,  
`splice()`, `concat()`, `slice()`, `sort()`, `reverse()`, `forEach()`, `map()`,  
`filter()`, `reduce()`, `reduceRight()`, `every()`, `some()`, `indexOf()`,  
`lastIndexOf()`, `find()`, `findIndex()`

izbjegavati korištenje operatora new

## DATE OBJECT

omogućava rad sa datumom i vremenom podrazumijevano, JavaScript koristi vremensku zonu web čitača i prikazuje datum kao *full text string*

Tue Dec 01 2020 08:22:42 GMT+0100 (Central European Standard Time)

### **kreiranje Date objekta**

- new Date() // trenutni datum i vrijeme
- new Date(milliseconds) // milisekunde od 01.01.1970. (UTC)
- new Date(dateString)
- new Date(year, month, day, hours, minutes, seconds, milliseconds)

### **metode:**

- **toString()**  
// Tue, 01 Dec 2020 07:27:15 GMT
- **toUTCString()**  
// Tue Dec 01 2020
- **toDateString()**  
// 2020-12-01T07:26:21.504Z
- **toISOString()**

### **get metode:**

- **getDate()**
- **getDay()**
- **getFullYear()**
- **getHours()**
- **getMilliseconds()**
- **getMinutes()**
- **getMonth()**
- **getSeconds()**
- **getTime()**
- **Date.now()**

### **set metode**

- **setDate()**
- **setFullYear()**
- **setHours()**
- **setMilliseconds()**
- **setMinutes()**
- **setMonth()**
- **setSeconds()**
- **setTime()**

## MATH OBJEKAT

omogućava matematičke operacije nad brojevima  
atributi

- E
- LN2
- LN10
- LOG2E
- LOG10E
- PI
- SQRT1\_2
- SQRT2

metode

- abs(x), acos(x), acosh(x), asin(x), asinh(x), atan(x), atan2(y, x), atanh(x), cbrt(x), ceil(x), cos(x), cosh(x), exp(x), floor(x), log(x), max(x, y, z, ..., n), min(x, y, z, ..., n), pow(x, y), random(), round(x), sin(x), sinh(x), sqrt(x), tan(x), tanh(x), trunc(x)

## BOOLEAN OBJEKAT

Boolean predstavlja dvije vrijednosti: "true" ili "false"  
podrazumijevano, Boolean je primitivni tip

- var b = true;

Boolean() – funkcija – izračunava da li je izraz istinit

- var b = Boolean(10>5)

Boolean može biti i objekat

- var b = new Boolean(true)

ako Boolean nema inicijalnu vrijednost ili ako je proslijedjena  
vrijednost jedna od sljedećih:

- 0
- -0
- null
- ""
- false
- undefined
- NaN
- objekat ima vrijednost false – za bilo koju drugu vrijednost objekat ima  
vrijednost true (čak i ako je inicijalizovan vrijednošću "false" – string)

## REGEXP OBJEKAT

- ▶ pretraga teksta i zamjena teksta
- ▶ String metode: search i replace koriste regularne izraze

- ▶ **sintaksa**

- var patt=new RegExp(pattern,modifiers);  
ili  
var patt=/pattern/modifiers;

- ▶ **primjer**

```
<html>
<body>
```

```
<script type="text/javascript">
    var string = "Ovo je Testna recenica...";
    var pattern = /test/i;
    var string2 = "Ovo je test recenica... Ovo je druga test recenica...";
    var pattern2 = /test/g;
    document.write(string.match(pattern));
    document.write("<br><br>");
    document.write(string2.match(pattern2));
</script>
```

```
</body>
</html>
```

rezultat:

Test  
test,test

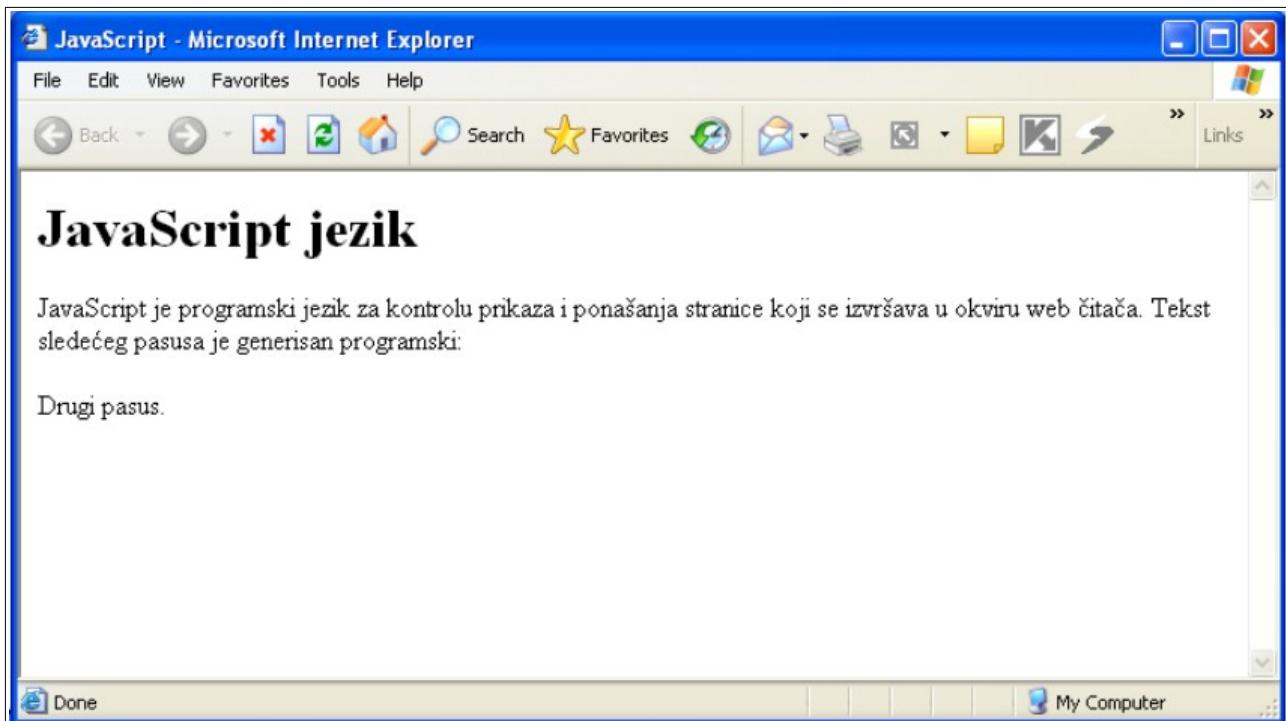
## PRIMJER JS

```
<html>
  <head>
    <title>JavaScript</title>
  </head>
  <body>
    <h1>JavaScript jezik</h1>

    <p>
      JavaScript je programski jezik za kontrolu prikaza i
      ponašanja stranice
      koji se izvršava u okviru web browser-a. Tekst
      sljedećeg pasusa je generisan programski:
    </p>

    <p>
      <script language="JavaScript">
        document.write("Drugi pasus.");
      </script>
    </p>
  </body>
</html>
```

SAX, BOM, XOM, DOM



## FUNKCIJE

### sintaksa

```
function f(arg1, arg2) {  
...  
return vrijednost;  
}
```

može biti pozvana eksplisitno iz JS koda ili kada se desi odgovarajući događaj (npr. klik na dugme)  
pozivanje funkcije – sa malim zagradama “()”

- ako se navede ime funkcije, ali bez malih zagrada, onda se referencira objekat funkcije, a ne rezultat njenog izvršavanja

može se dodijeliti promjenljivoj

- `x = f(10, 20);`

promjenljive deklarisane u tijelu funkcije (uz pomoć ključne riječi var) su lokalne promjenljive

### sistemske funkcije:

- `isNaN()` – vraća true ako prosliđeni string nije broj
- `eval()` – interpretira prosliđeni string kao JavaScript kod
- `parseInt()` – parsira string u integer
- `parseFloat()` – parsira string u float
- `alert()` – ispis poruke u MessageBox-u
- `escape()`, `unescape()` – kodira/dekodira URL-ove (npr. zamjenjuje razmak simbolima %20 i sl.)

## PRIMJER ZA FUNKCIJE

```
<html>
<head>
    <title>JavaScript</title>
    <script language="JavaScript">
        function ispisi() {
            document.write("Drugi pasus, ali iz funkcije.");
        }
    </script>
</head>
<body>
    <h1>JavaScript funkcije</h1>

    <p>
        Tekst sljedeceg pasusa je generisan pozivom funkcije koju smo
        mi napisali:
    </p>

    <p>
        <script language="JavaScript">
            ispisi();
        </script>
    </p>
</body>
</html>
```



ARROW ANONYMYS (BEZIMENE) LAMBDA FUNKCIJE U STILU OBJEKATA  
(I IME FUNKCIJE JE REFERENCA NA OBJEKAT ILI SE MOZE TAKO TRETIJATI)

## kraći zapis funkcija

```
arrow = function() {  
    return "Test";  
}  
  
arrow = () => {  
    return "Test";  
}  
  
arrow = () => "Test";
```

ako postoje parametri, navode se u zagradama  
ako postoji samo jedan parametar, zgrade se mogu ukloniti

ako postoji this – odnosi na objekat koji definiše arrow funkciju

POZIVI SU POSEBNI OPERATORI NAD OBJEKTOM FUNKCIJE ILI SE BAR TAKO MOGU TRETIJATI (DA NISTA MEHANIZMI NAD OBJEKTIMA FUNKCIJA)

## OBJEKTI

objekti su kontejneri za property-je i metode

- prvi način:

```
var osobaObj=new Object();
osobaObj.firstName="Marko";
osobaObj.lastName="Markovic";
osobaObj.age=25;
```

- drugi način:

```
var osobaObj={
    firstName:"Marko",
    lastName:"Markovic",
    age:50
};
```

pristup property-ju objekta

- `osobaObj.firstName`
- `osobaObj["firstName"]`

metode

```
var osobaObj={
    firstName: "Marko",
    lastName: "Markovic",
    age:50,
    name: function() {
        return this.firstName + " " + this.lastName;
    }
};
```

pristup metodi objekta

- `osobaObj.name()`
- `osobaObj.name` – vraća definiciju funkcije  
`function() { return this.firstName + " " + this.lastName; }`

ne preporučuje se deklaracija String, Number i Boolean kao objekata

- `var s = new String();`
- `var n = new Number();`
- `var b = new Boolean();`

## primjer funkcije koja kreira objekte

```
function Person(fn,ln,age)
{
    this.firstName=fn;
    this.lastName=ln;
    this.age=age;
}
```

## korištenje

- var osobaObj = new Person("First", "Last", 20);

**null**

osobaObj = null; // vrijednost je null, ali je tip i dalje objekat

**undefined**

osobaObj = undefined; // vrijednost je nedefinisana, tip je nedefinisani

## DOGADJAJI

događaji su akcije koje mogu biti detektovane JavaScript-om

- okončanje učitavanja stranice, klik na dugme, ...

događaji se registruju i obrađuju *event handler-ima*

*event handler* atributi – dodaju se HTML elementima:

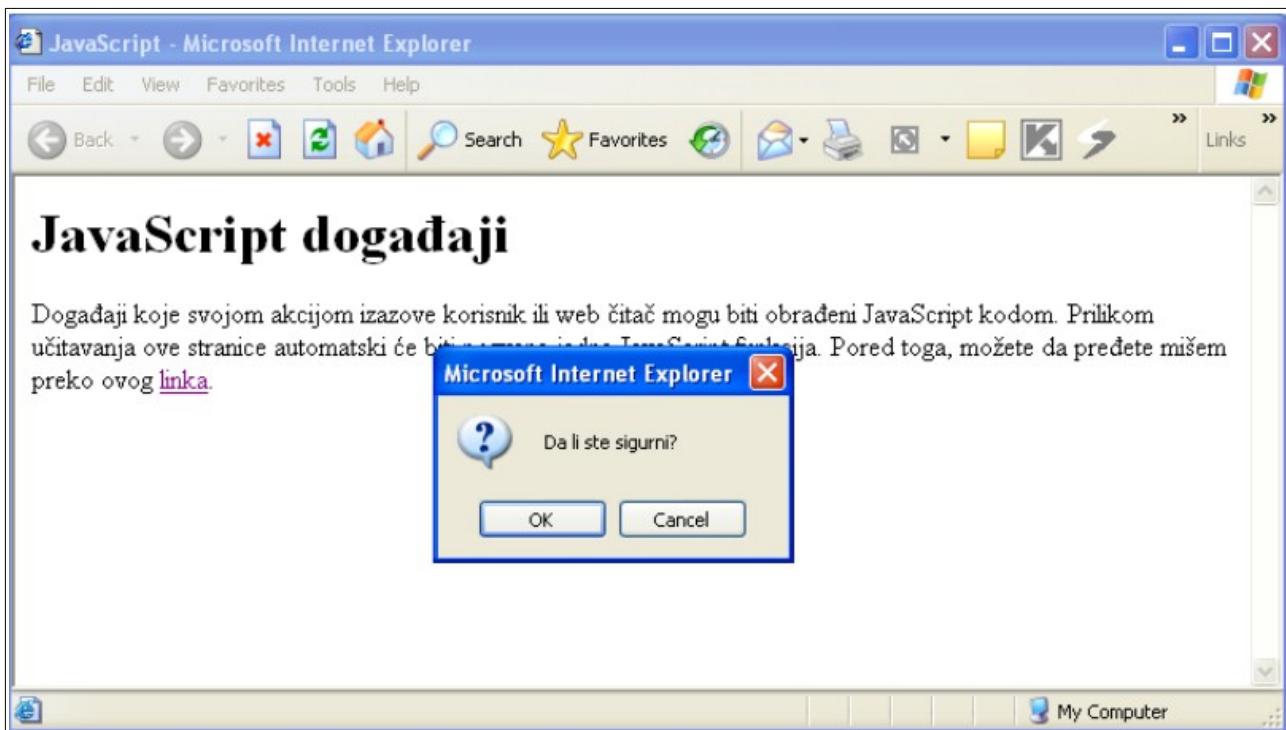
- *onChange*,
- *onClick*,
- *onDbClick*,
- *onDragDrop*,
- *onFocus*,
- *onKeyDown*, *onKeyUp*, *onKeyPress*,
- *onLoad*,
- *onUnload*,
- *onMouseDown*, *onMouseUp*, *onMouseMove*, *onMouseOver*, *onMouseOut*,
- *onSubmit*,
- *onSelect*,
- *onReset*,
- *onError*,
- *onAbort*,
- *onBlur*

## PRIMJER ZA DOGADJAJE

```
<html>
  <head>
    <title>JavaScript</title>
    <script language="JavaScript">
      function mis() {
        confirm("Da li ste sigurni?");
      }

      function greeting() {
        alert("Dobrodosli na ovu stranicu");
      }
    </script>
  </head>
  <body onLoad="greeting()">
    <h1>JavaScript dogadjaji</h1>
    <p>
      Dogadjaji koje svojom akcijom izazove korisnik ili web browser mogu biti obradjeni JavaScript kodom. Prilikom ucitavanja ove stranice automatski ce biti pozvana jedna JavaScript funkcija. Pored toga, mozete da predjete misem preko ovog <a href="primjer03.html" onMouseOver="mis()">linka</a>.
    </p>
  </body>
</html>
```





#### STRICT MODE

direktiva – kod treba da bude izvršen u „strict“ modu

“use strict“ se mora naći na početku JS koda  
varijable moraju biti deklarisane, objekti moraju  
biti deklarisani, brisanje varijable nije dozvoljeno,  
brisanje funkcije nije dozvoljeno, itd.

bez “strict” moda nedeklarisane varijable  
automatski postaju globalne – u “strict” modu to  
nije moguće

nije dozvoljena upotreba rezervisanih riječi:

- implements, interface, let, package, private, protected, public, static, yield

“use strict”  
“UTF-8”

## HOISTING

pomjeranje deklaracija na vrh JS koda  
promjenljive mogu biti deklarisane i nakon što su  
korištene

promjenljive deklarisane pomoću let i const se  
pomjeraju na vrh, ali se ne inicijalizuju

- korištenje promjenljive prije nego što je deklarisana  
pomoću let i const dovodi do greške (ReferenceError)  
deklaracija i inicijalizacija (istovremeno) se ne  
pomjera na vrh

preporuka je deklarisati promjenljive na početku  
opsega

## JSON-JAVA SCRIPT SERIJALIZACIJA-JAVASCRIPT OBJECT NOTATION

### **sintaksa**

- name-value parovi
- odvojeni zarezima
- uglaste zagrade sadrže elemente niza
- vitičaste zagrade sadrže objekte

### **konverzija JSON teksta u JS objekat**

- ugrađena JS funkcija JSON.parse()

### **JS objekat u JSON tekst**

- ugrađena JS funkcija JSON.stringify()

## KLASE

ključna riječ class

uvijek se definiše metoda constructor()

- poziva se automatski prilikom kreiranja novog objekta
- koristi se da inicijalizuje atribute objekta

```
class Person {  
    constructor(name, yearBorn) {  
        this.name = name;  
        this.yearBorn = yearBorn;  
  
    }  
    age() {  
        let date = new Date();  
        return date.getFullYear() - this.yearBorn;  
    }  
}  
  
let person = new Person("Marko Markovic", 2000);
```

klasa može da sadrži metode

~~instanciranje objekata – ključna riječ new~~

## RAD SA FORMAMA

reprezentovane objektom

metode:

- *submit()* – šalje podatke iz forme na odredište definisano **action** atributom taga **form**
- *reset()* – simulira pritisak na Reset dugme forme

atributi:

- *elements* – niz elemenata forme. Svaki element ima **value** atribut za pristup sadržaju,
- *length* – broj elemenata na formi.
- *action* – sadržaj action atributa.
- *acceptCharset*, *encoding*, *enctype*, *method*, *name*, *target*

# validacija HTML formi pomoću JS

```
function validateForm() {  
    var x = document.forms["form1"]["firstName"].value;  
    if (x == "") {  
        alert("First name is empty!");  
        return false;  
    }  
}  
  
<form name="form1" action="#" onsubmit="return validateForm()" method="post">  
    Name: <input type="text" name="firstName">  
    <input type="submit" value="Submit">  
</form>
```

## automatska HTML validacija – atribut required

- disabled, max, min, pattern, required, type

```
<form name="form1" action="#" method="post">  
    Name: <input type="text" name="firstName" required>  
    <input type="submit" value="Submit">  
</form>
```

**required** – da postoji vrednost, ako je tekst u pitanju

- Forms API
- metode:
  - checkValidity() – vraća true ako input element sadrži validne podatke
  - setCustomValidity() – postavlja validationMessage property input elementa
- property:
  - validity – sadrži boolean property-je koji se odnose na validnost input elementa
    - customError, patternMismatch, rangeOverflow, rangeUnderflow, stepMismatch, tooLong, typeMismatch, valueMissing, valid
  - validationMessage – sadrži poruku koju će browser prikazati ako je validnost false
  - willValidate – označava da li će input element viti validiran

## RAD SA GRESKAMA

ključne riječi: try, catch, throw i finally  
try-catch blokovi

```
try {           // kod koji može baciti grešku
}             catch(err) {
               // obrada greške...
}
finally {
               // kod koji se uvijek izvršava
}
```

primjer

```
try {
      alerttt("Welcome guest!");
}
catch(err)
{
      txt="funkcija alerttt ne postoji.\n";
      alert(txt);
}
```

## bacanje izuzetka – ključna riječ throw

```
<html>
  <body>
    <script type="text/javascript">
      var x=prompt("Unesite broj veci od 0 i manji od 10:","");
      try
      {
        if(x>10)
          throw "Err1";
        else if(x<0)
          throw "Err2";
        else if(isNaN(x))
          throw "Err3";
      }
      catch(er)
      {
        if(er=="Err1")
          alert("Greska! Unijeli ste broj veci od 10!");
        if(er=="Err2")
          alert("Greska! Unijeli ste broj manji od 0!");
        if(er=="Err3")
          alert("Greska! Niste unijeli broj!");
      }
    </script>
  </body>
</html>
```

## DEBUGGING

- ▶ `console.log()`
- ▶ `debugger`

```
var x = 10;  
var y = 20;  
var z = x * y;  
console.log(z);
```

```
var x = 10 * 50;  
debugger;  
document.getElementById("test").innerHTML = x;  
x = 20 * 50;  
document.getElementById("test").innerHTML = x;
```

- ▶ Developer Tools / Web Developer

### DOBRA PRAKSA

izbjegavati korištenje new operatora  
izbjegavati korištenje globalnih promjenljivih  
deklarisati lokalne promjenljive (kako ne bi  
postale globalne)  
deklaracije pozicionirati na vrhu skripte ili  
funkcije  
vršiti inicijalizacija promjenljivih  
izbjegavati korištenje String, Number,  
Boolean, i dr. objekata – koristiti primitivne  
tipove

## VALIDACIONES - UVID

# Constraint Validation DOM Methods

Property	Description
checkValidity()	Returns true if an input element contains valid data.
setCustomValidity()	Sets the validationMessage property of an input element.

---

## KLIJENTSKO UCITAVANJE KODA – NE PREPORUCUJE SE

Evaluate/Execute JavaScript code/expressions:

```
var x = 10;
var y = 20;
var a = eval("x * y") + "<br>";
var b = eval("2 + 2") + "<br>";
var c = eval("x + 17") + "<br>";

var res = a + b + c;
```