

# JavaScript

Internet programiranje  
IV godina, ETF Banjaluka

# JavaScript

- ▶ obično se koristi za:
  - manipulaciju slikama
  - validaciju formi
  - dinamičke promjene sadržaja HTML strana
- ▶ besplatan
- ▶ platformski neutralan
- ▶ sintaksa slična Java programskom jeziku
- ▶ nema tipove podataka
  - kod deklaracije promenljivih se ne stavlja tip
- ▶ ugrađene funkcije
- ▶ JavaScript je OO jezik
- ▶ ugrađeni objekti
  - String, Number, Boolean, Array, Object, Null, Undefined, Date, Math, RegExp
- ▶ kreiranje vlastitih objekata
- ▶ sistem događaja

# Sintaksa

- ▶ promjenljive
  - mogu biti:
    - globalne (izvan funkcije, sve f-je je vide)
    - lokalne (unutar funkcija)
  - nemaju tip
  - ključna reč *var* deklarira lokalnu promjenljivu
    - nije obavezna
  - nizovi:
    - upotreba ključne riječi *new*.  
`var mojNiz=new Array(10);`
  - stringovi:  
`tekst="neki tekst";`  
`tekst='neki tekst';`
- ▶ operatori
  - slični operatorima programskog jezika Java/C++
- ▶ kontrola toka
  - if, for, while,...

# JavaScript

```
var x;           // Undefined  
var x = 5;       // Number  
var x = "x";     // String  
x = undefined;  // i vrijednost i tip su sada nedefinisani
```

- ▶ JavaScript je jednostavniji programski jezik
  - ne zahtjeva se, i nije dozvoljeno, deklarirati 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

# Sintaksa

- ▶ 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
  - ne mogu se koristiti rezervisane riječi za identifikatore
  - nazivi su *case sensitive*

# SCRIPT tag

- ▶ tag **<script>** specificira Script kod koji se pokreće direktno u browser-u
- ▶ ako atribut **language** ima vrijednost “JavaScript”, tada se radi o JavaScript programskom jeziku
- ▶ 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 head-u i u body-ju
- ▶ funkcije definisane u tagu **<script>** u zaglavlju dokumenta, mogu se pozivati bilo gdje u dokumentu

# JavaScript

- ▶ 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)

# JavaScript

- ▶ 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
- ▶ ključne riječi ne mogu da se koriste kao imena promjenljivih (rezervisane riječi) – primjeri:
  - var, return, switch, break, continue, if, else, ...



# Komentari

- ▶ za jednolinijski komentar – „//”  
// komentar u jednoj liniji ...
- ▶ za komentar više redova – „/\*” za početak bloka pod komentarom i „\*/” za kraj bloka pod komentarom:

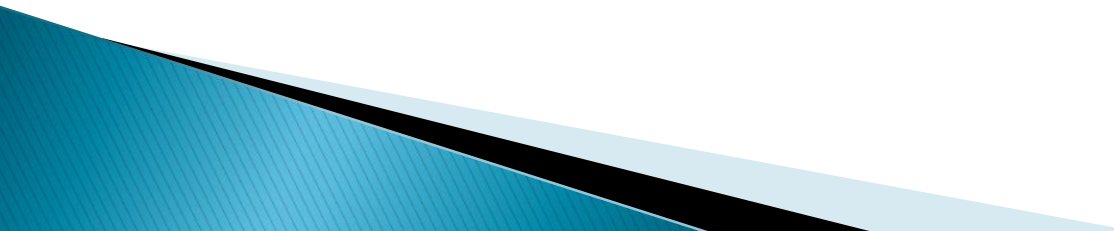
```
/*
```

```
komentar u više redova...
```

```
*/
```



# Operatori

- ▶ Aritmetički
  - ▶ Na nivou bita
  - ▶ Relacionalni
  - ▶ Logički
- 

# Aritmetički operatori

- ▶ + sabiranje
- ▶ += sabiranje i dodjela
- ▶ - oduzimanje
- ▶ -= oduzimanje i dodjela
- ▶ \* množenje
- ▶ \*= množenje i dodjela
- ▶ / dijeljenje
- ▶ /= dijeljenje i dodjela
- ▶ % moduo
- ▶ %= moduo i dodjela
- ▶ ++ inkrement
- ▶ -- dekrement
  
- ▶ operator + kod stringova
  - dva stringa – konkatencija
  - string i broj – string

# Operatori 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 ekskluzivno ILI (XOR) –  $a \wedge b$  – rezultat je 1, ako biti imaju različite vrijednosti, u slučaju da imaju iste vrijednosti, rezultat je 0
- ▶ logičko NE (NOT) –  $\sim a$  – komplementira bitove operanda a
- ▶ pomjeranje ulijevo –  $a \ll b$  – pomjera binarni sadržaj operanda a za b mjesta ulijevo. Prazna mjesta popunjava sa vrijednošću 0
- ▶ pomjeranje udesno sa znakom –  $a \gg b$  – pomjera binarni sadržaj operanda a za b mjesta udesno. Prazna mjesta popunjavaju se sa vrijednošću najstarijeg bita.
- ▶ pomjeranje udesno sa nulama –  $a \ggg b$  – sadržaj operanda a pomjera se za b mjesta udesno. Prazna mjesta popunjavaju se vrijednošću 0.

# Logički 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 poređenja

- ▶ 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

# Ternarni operator

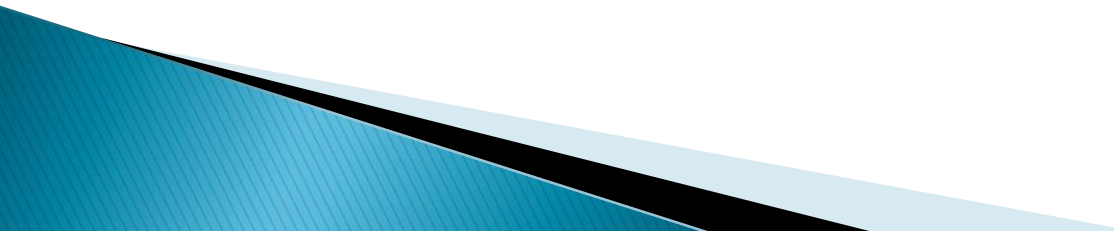
- ▶ `expression ? statement1 : statement2`
- ▶ izraz *expression* je bilo koji izraz čiji rezultat je vrijednost logičkog tipa
- ▶ ako je rezultat izraza `true`, onda se izvršava *statement1*, u suprotnom *statement2*

# Type operatori

- ▶ `typeof` – vraća tip promjenljive
- ▶ `instanceof` – vraća `true` ako je objekat instanca nekog tipa



# Kontrola toka

- ▶ if
  - ▶ if-else
  - ▶ switch
  - ▶ while
  - ▶ do while
  - ▶ for
  - ▶ for in
  - ▶ break
  - ▶ continue
  - ▶ with
- 

# Specijalni karakteri

- ▶ \' jednostruki navodnik
  - ▶ \" dvostruki navodnik
  - ▶ \& ampersand
  - ▶ \\ backslash
  - ▶ \n nova linija
  - ▶ \r carriage return
  - ▶ \t tab
  - ▶ \b backspace
  - ▶ \f form feed
- 

# Pozivanje JavaScript-a

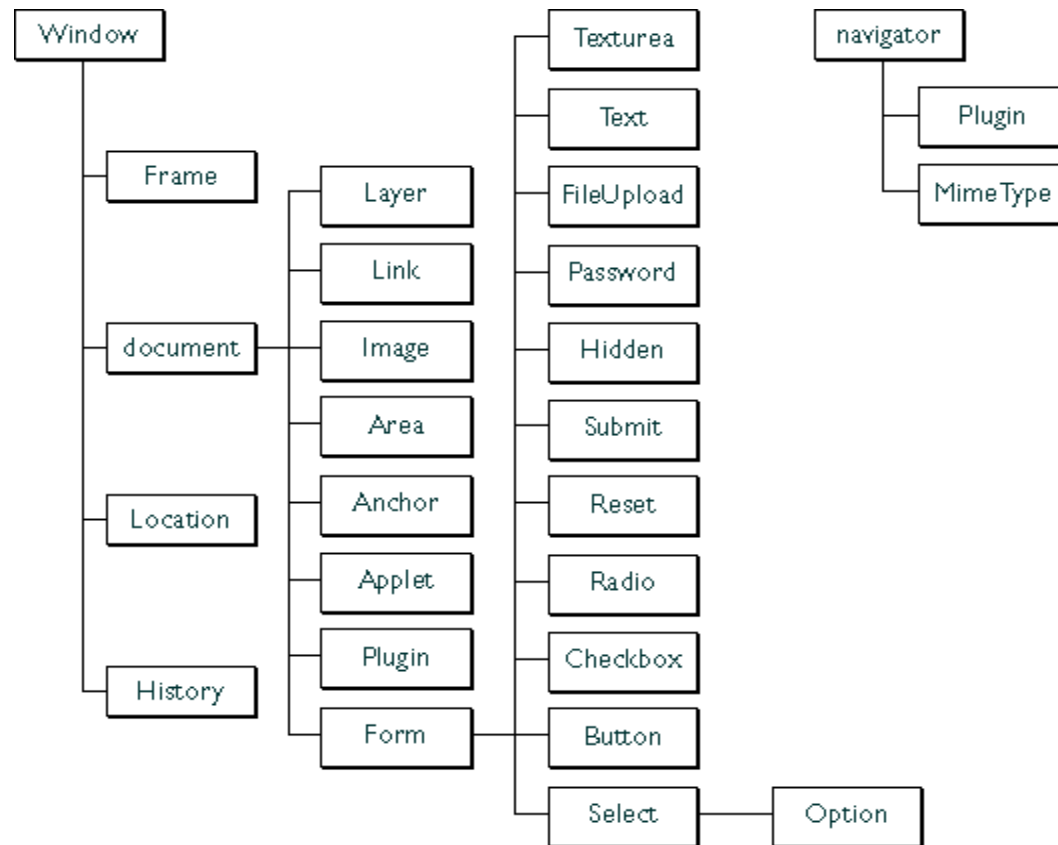
- ▶ kao reakcija na neki događaj
  - unutar `<script>` taga bilo gdje unutar HTML dokumenta
  - unutar `<body>` taga
  - unutar `<head>` taga
    - ako koristimo JavaScript funkciju, nju moramo da definišemo unutar `<head>` taga da bismo mogli da je pozivamo iz bilo kog JavaScript koda
- ▶ kao adresu unutar `<a>` taga:

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

- ▶ u eksternoj datoteci

```
<head>  
<script type="text/javascript"  
rc="javascript.js"></script>  
</head>
```

# Hijerarhija objekata



# Window objekt

- ▶ omogućava manipulaciju prozorima
- ▶ sadrži informacije o tekućem prozoru
- ▶ 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.
  - *closed, defaultStatus, innerHeight, innerWidth, length, name, navigator, opener, outerHeight, outerWidth, pageXOffset, pageYOffset, parent, screen, screenLeft, screenTop, screenX, screenY, self, status, top*

# Location objekt

- ▶ reprezentuje URL stranice koja je učitana u navigator  
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* – pun URL do 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 objekt

- ▶ 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

# Document objekt

- ▶ omogućava ispis HTML-a na ekran
- ▶ sadrži informacije o tekućem dokumentu
- ▶ 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.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* – sadržaj **title** taga
  - *URL* – kompletan URL dokumenta
  - *document.anchors*, *document.baseURI*, *document.body*, *document.cookie*, *document.doctype*, *document.documentElement*, *document.documentMode*, *document.documentURI*, *document.domain*, *document.domConfig*, *document.embeds*, *document.head*, *document.images*, *document.implementation*, *document.inputEncoding*, *document.lastModified*, *document.readyState*, *document.referrer*, *document.scripts*, *document.strictErrorChecking*



# String

- ▶ string predstavlja proizvoljan niz karaktera između dvostrukih (“neki tekst”) ili jednostrukih (‘neki tekst’) znakova navoda
  - `var s = new String("tekst");`
  - `var s = "tekst";`
- ▶ metode:
  - *substring()* – vraća dio 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
- ▶ atributi:
  - *length* – dužina stringa
- ▶ operator +
  - `x="5"+5;` – x sadrži string vrijednost

# Number

- ▶ brojevi mogu biti sa i bez decimala
- ▶ cjelobrojni brojevi se mogu koristiti sa brojnom osnovom 10, 2, 8 ili 16
- ▶ podržani su i racionalni brojevi – 3.14, 314E-2 ili 314e-2
- ▶ brojevi su uvijek 64-bitni floating point
- ▶ NaN – Not a Number – rezervisana riječ koja označava vrijednost koja nije broj
- ▶ Infinity (ili -Infinity) – veći/manji od maksimalne/minimalne vrijednosti za broj
- ▶ `typeof Infinity; // number`

# Number

- ▶ brojevi mogu biti i objekti
  - `var n = new Number(123456);` // 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`

# Date objekt

- ▶ `new Date()` // trenutni datum i vrijeme
- ▶ `new Date(milliseconds)` // miliskunde od 01.01.1970.
- ▶ `new Date(dateString)`
- ▶ `new Date(year, month, day, hours, minutes, seconds, milliseconds)`
- ▶ metode:
  - `toString()`
  - `toUTCString()`
  - `toDateString()`

# Date objekt

- ▶ metode:
  - getDate()
  - getDay()
  - getFullYear()
  - getHours()
  - getMilliseconds()
  - getMinutes()
  - getMonth()
  - getSeconds()
  - getTime()

# Array objekt

- ▶ `var cars=new Array();`     `//` nema potrebe za ovakvim kreiranjem nizova  
`cars[0]="Audi";`  
`cars[1]="Volvo";`  
`cars[2]="BMW";`
- ▶ `var cars=new Array("Audi","Volvo","BMW");`
- ▶ `var cars=["Audi","Volvo","BMW"];`
- ▶ atribut: *length*
- ▶ metode:
  - *push(), pop(), shift(), unshift(), isArray(), toString(), join(), splice(), concat(), slice(), sort(), reverse()*

# Boolean objekt

- ▶ Boolean objekt predstavlja dvije vrijednosti: "true" ili "false"
- ▶ ako Boolean objekt nema inicijalnu vrijednost, ili ako je proslijeđena 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)

# Math objekt

## ▶ property

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

## ▶ metode

- `abs(x)`, `acos(x)`, `asin(x)`, `atan(x)`, `atan2(y,x)`, `ceil(x)`, `cos(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)`, `sqrt(x)`, `tan(x)`



# RegExp objekt

## ▶ 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

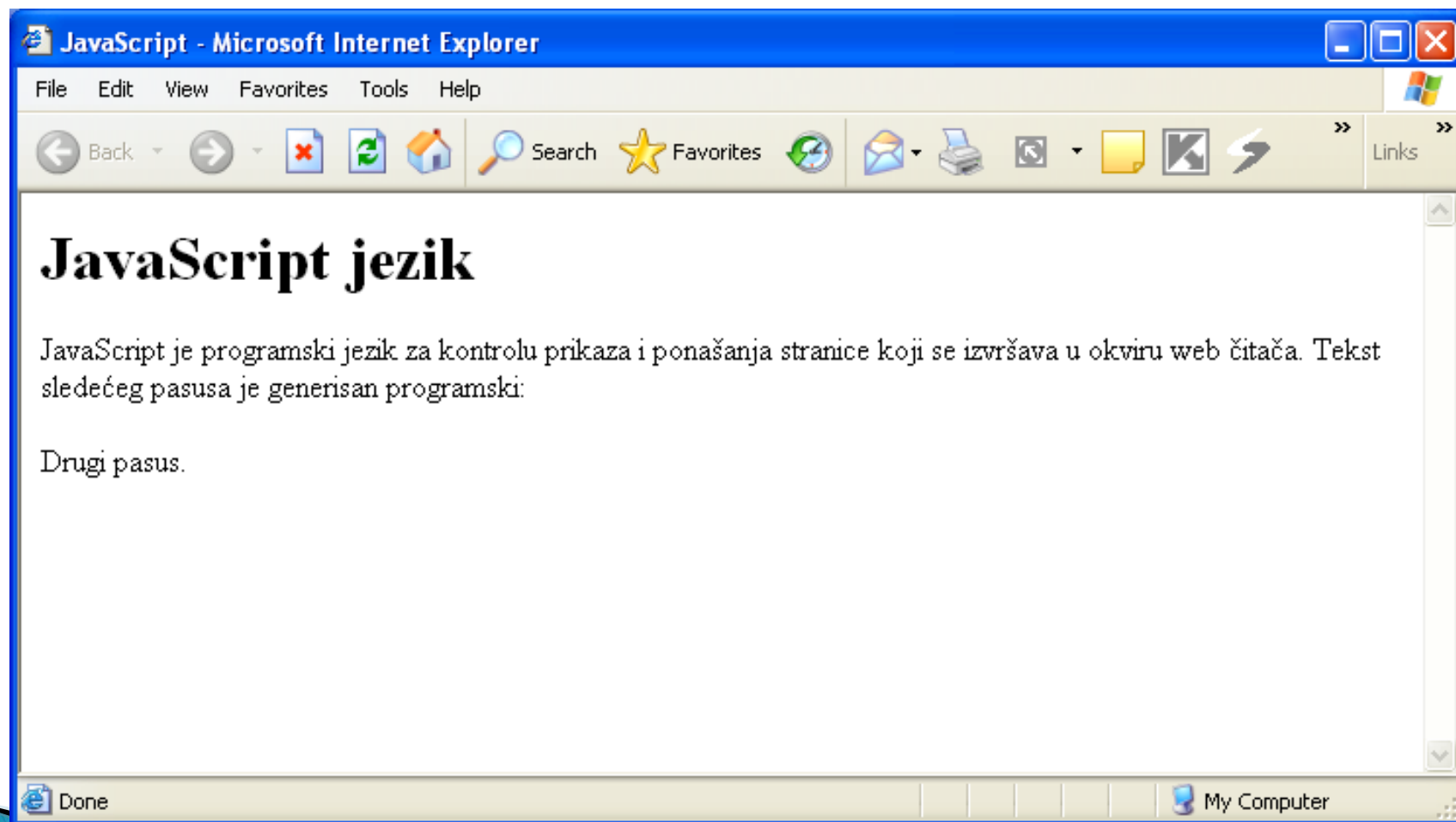
# Primjer

```
<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>
```

# Primjer



# Funkcije

- ▶ definicija funkcija unutar <head> taga:

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

- ▶ poziv funkcije iz tijela HTML dokumenta (unutar <body> taga)
- ▶ sistemske funkcije:
  - *isNaN()* – vraća true ako prosljeđeni string nije broj
  - *eval()* – interpretira prosljeđeni 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

```
<html>
  <head>
    <title>JavaScript</title>
    <script language="JavaScript">
      function ispis() {
        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">
        ispis();
      </script>
    </p>
  </body>
</html>
```

# Primjer



# Događaji

- ▶ događaji su akcije koje mogu biti detektovane JavaScript-om
- ▶ događaji se registruju i obrađuju *event handler*-ima
- ▶ *event handler*-i:
  - *onBlur*,
  - *onClick*,
  - *onChange*,
  - *onDbClick*,
  - *onDragDrop*,
  - *onFocus*,
  - *onKeyDown*, *onKeyUp*, *onKeyPress*,
  - *onLoad*,
  - *onUnload*,
  - *onMouseDown*, *onMouseUp*, *onMouseMove*, *onMouseOver*, *onMouseOut*,
  - *onSubmit*,
  - *onSelect*,
  - *onReset*,
  - *onError*,
  - *onAbort*

# Primjer

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

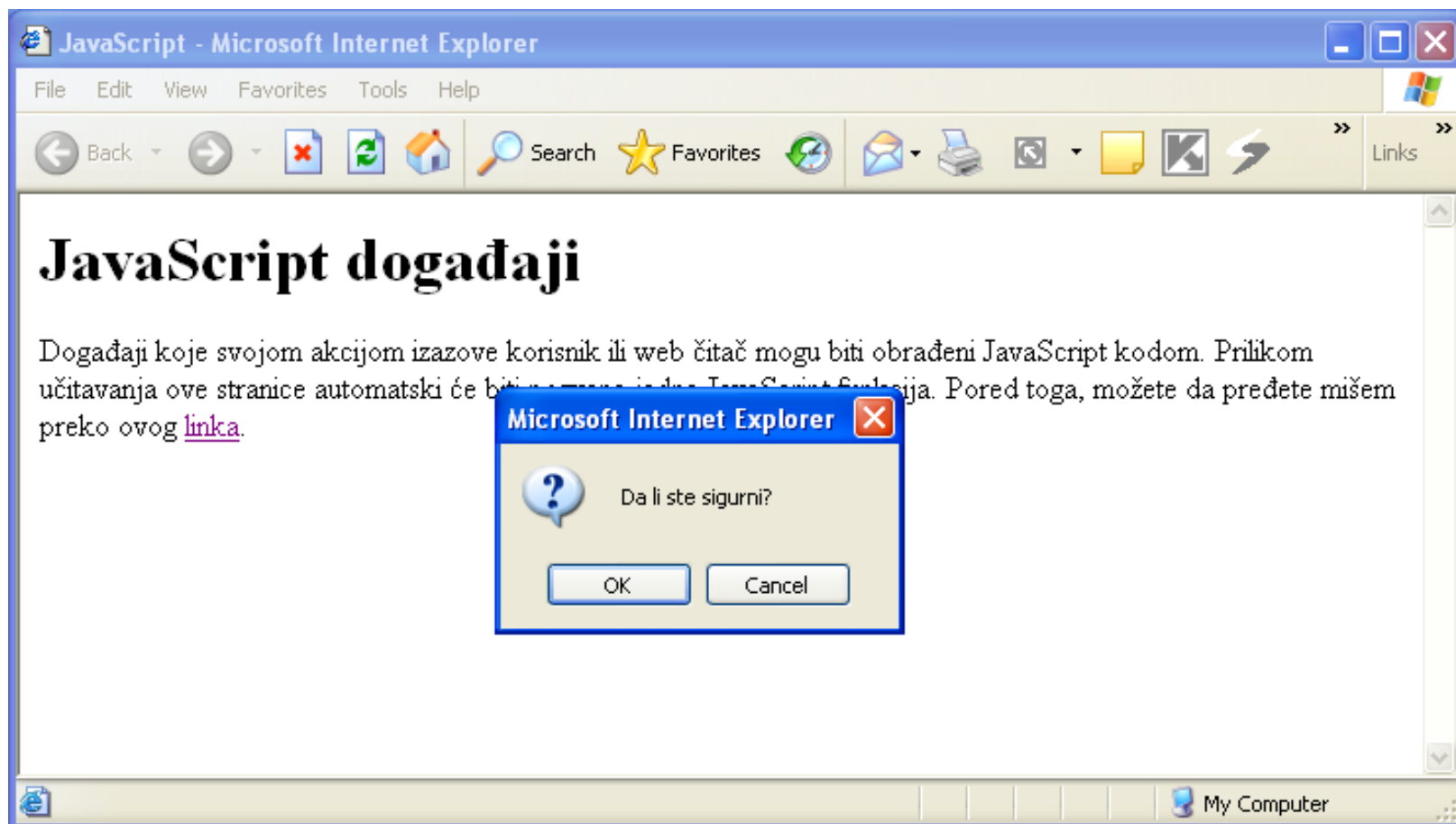
      function greeting() {
        alert("Dobro dosli na ovu stranicu");
      }
    </script>
  </head>
  <body onLoad="greeting()">
    <h1>JavaScript dogadjaji</h1>
    <p>
      Dogadjaji koje svojom akcijom izazove korisnik ili web browser
      mogu biti obradjeni JavaScript kodom. Prilikom ucitavanja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



# Primjer



# Primjer



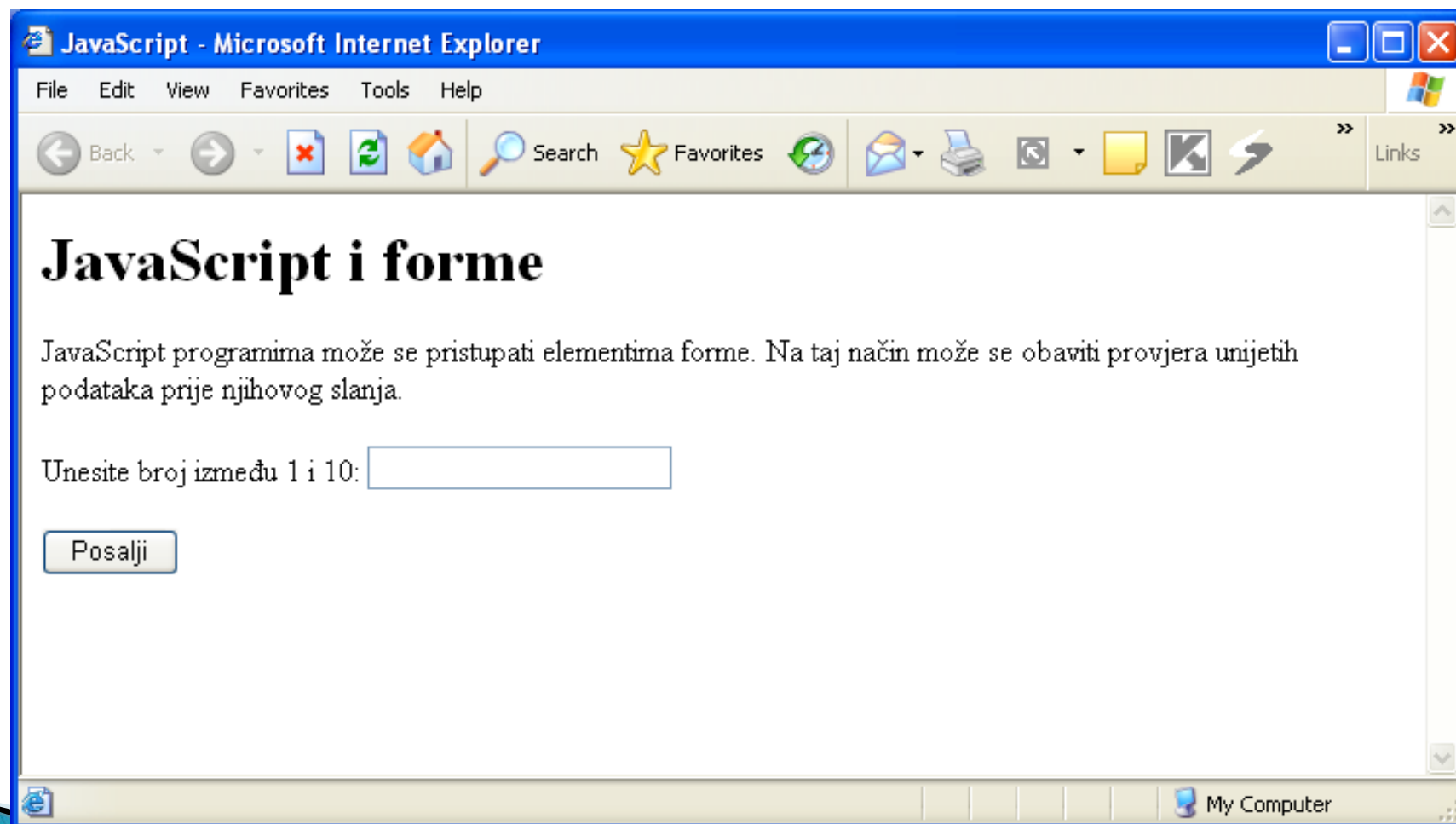
# Forme

- ▶ reprezentovane Form 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`

# Primjer

```
<html>
  <head>
    <title>JavaScript</title>
    <script language="JavaScript">
      function provjera() {
        vrijednost = document.forms['forma'].polje1.value;
        if (isNaN(vrijednost)) {
          alert("Niste unijeli broj");
          return false;
        } else if (vrijednost >= 1 && vrijednost <= 10) {
          return true;
        } else {
          alert("Niste unijeli broj u opsegu od 1 do 10");
          return false;
        }
      }
    </script>
  </head>
  <body>
    <h1>JavaScript i forme</h1>
    <p>
      JavaScript programima može se pristupati elementima forme. Na taj način može se obaviti
      provjera unijetih podataka prije njihovog slanja.
    </p>
    <form name="forma" action="primjer04b.html" onSubmit="return provjera()">
      <p>
        Unesite broj između 1 i 10:
        <input type="text" name="polje1"> <br><br>
        <input type="submit" name="polje2" value=" Posalji ">
      </p>
    </form>
    primjer04
  </body>
</html>
```

# Primjer



# Primjer

```
<html>

  <head>
    <title>JavaScript</title>
  </head>

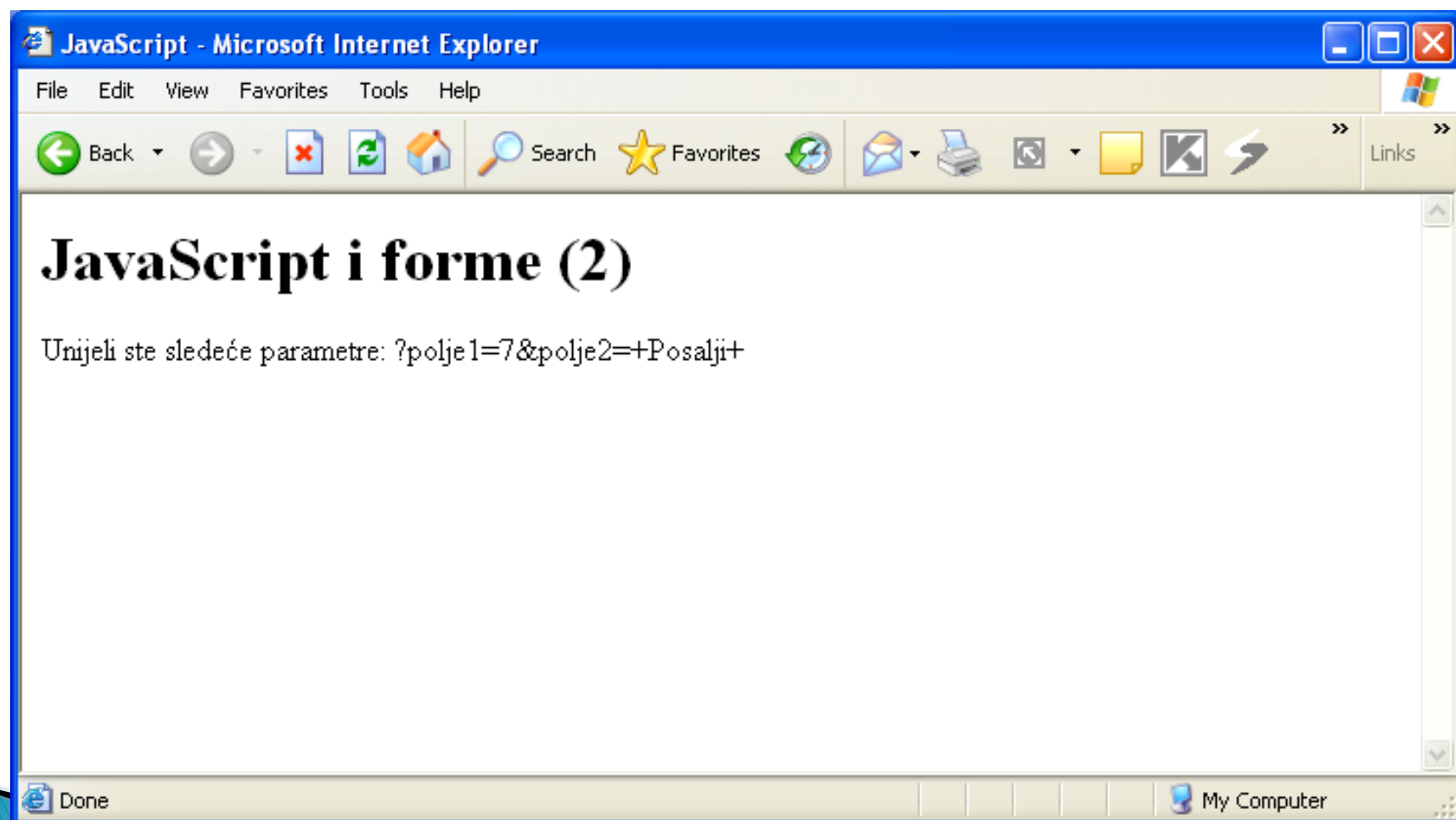
  <body>
    <h1>JavaScript i forme (2)</h1>

    <p>
      Unijeli ste sljedece parametre:
      <script language="JavaScript">
        document.write(window.location.search);
      </script>
    </p>

  </body>

</html>
```

# Primjer



# Cookie

- ▶ Cookie je tekstualni fajl koji se može zapamtiti na računaru klijenta
- ▶ Format koji cookie fajl mora da zadovolji je:

`ime=vrijednost`

`[;EXPIRES=datum]`

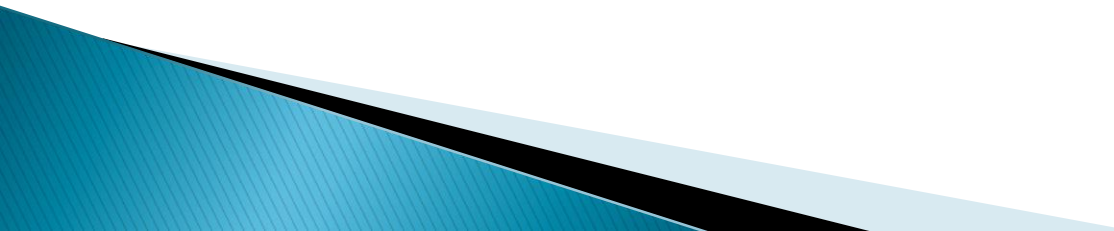
`[;DOMAIN=imeDomena]`

`[;PATH=putanja]`

`[;SECURE]`



# Cookie

- ▶ 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

- ▶ cookie može biti kreiran, pročitan i obrisano 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`

# Cookie – nedostaci

- ▶ nepouzdana identifikacija – cookie ne identifikuje 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

# Web čitači koji ne podržavaju JS

- ▶ prikazuju JavaScript kao sadržaj stranice
- ▶ rješenje:
  - HTML komentar oko JavaScript koda

```
<script type="text/javascript">  
<!--  
document.write("tekst!");  
//-->  
</script>
```

# Popup box-ovi

- ▶ Alert Box
  - `alert("sometext");`
- ▶ Confirm Box
  - `confirm("sometext");`
- ▶ Prompt Box
  - `prompt("sometext","defaultvalue");`

# Rad s *timing* događajima

- ▶ setTimeout()
- ▶ clearTimeout()

- ▶ pokretanje

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

- ▶ zaustavljanje

```
clearTimeout(t);
```



# Rad s vlastitim objektima

- ▶ kreiranje objekta i dodavanje property-ja

prvi način:

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

drugi način:

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

- ▶ primjer funkcije koja kreira objekte

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

- ▶ null

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

- ▶ undefined

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

# JavaScript ispisi

- ▶ `innerHTML` – ispis u HTML element
- ▶ `document.write()` – ispis u HTML stranicu
- ▶ `window.alert()` – ispis u alert box
- ▶ `console.log()` – ispis u konzolu browser-a



# Debugging

- ▶ `console.log()`
- ▶ `debugger`

```
<script>  
var x = 10 * 50;  
debugger;  
document.getElementById("demo").innerHTML = x;  
x = 20 * 50;  
</script>
```

# use strict

- ▶ direktiva – kod treba da bude izvršen u „strict“ modu
- ▶ “use strict” se mora naći na početku JS koda
- ▶ varijable moraju biti deklarisanе, objekti moraju biti deklarisani, brisanje varijable nije dozvoljeno, brisanje funkcije nije dozvoljeno, itd.
- ▶ nije dozvoljena upotreba rezervisanih riječi:
  - implements, interface, let, package, private, protected, public, static, yield

# JSON

- ▶ `var text = '{ "students" : [' +  
 '{ "fn":"Marko" , "ln":"Markovic" },' +  
 '{ "fn":"Ana" , "ln":"Anic" },' +  
 '{ "fn":"Petar" , "ln":"Petrovic" } ]}';`
- ▶ `var obj = JSON.parse(text);`

# Rad s greškama

- ▶ try-catch blokovi

- ▶ try

```
{
  // kod koji može baciti grešku
}
catch(err)
{
  // obrada greške...
}
```

- ▶ primjer

- ▶ try

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

# Rad s greškama

- ▶ 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>
```

# Rad s formama

## ► validacija

```
function validateForm()
{
  var x=document.forms["myForm"]["fname"].value;
  if (x==null || x=="")
  {
    alert("First name must be filled out");
    return false;
  }
}
```

# Rad s formama

- ▶ 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 biti validiran