ПРОГРАМИРАЊЕ ВЕБ АПЛИКАЦИЈА

JavaScript

JAVASCRIPT

- Uvod u JS
  - Šta je JS
  - Čemu služi JS
  - Gde se pojavljuje
  - Kako se implementira
  - Osnovne funkcije
- DOM
  - Navigacija po DOM dokumentu
  - Manipulacija DOM objektima
  - Stilizacija DOM objekata
- Forme i kontrole
  - Događaji
  - validacija

ŠTA JE JS

- Programski jezik
- Derivat C programskog jezika
- Multifunkcionalan je
- Najčešće se koristi u izradi klijentskih delova aplikacija
- Nije isto što i Java
- Dijalekti (način pisanja) se razlikuje u odnosu na pretraživač
- Poslednja verzija je ECMA6 Script

OSOBINE

- Osobine
  - Objektno zasnovan (ne orjentisan) – koristi hijerarhiju ugrađenih objekata sa definisanim metodama i osobinama, ali ne podržava koncepte objekto orjentisanih jezika
  - Platformski neutralan – kod se izvršava u okviru veb čitača, bez obzira na hardversko i softversko okruženje, veličina programa je obično mala
  - Modularno programiranje – za čuvanje i izvršavanje koda mogu se koristiti posebni dokumenti sa **.js** ekstenzijom
  - Integrisanost sa HTML-om – u okviru jedne HTML stranice moguće je, na proizvoljan način, kombinovati JS i HTML kod. Takođe je, iz JS, moguće generisati HTML kod

HTML KAO OGRANIČAVAJUĆI FAKTOR

- HTML je platformski neutralan jezik koji omogućava efikasno prezentovanje željenog sadržaja na klijentskoj strani
- Glavni nedostatak HTML-a je u tome što ne podržava neposrednu komunikaciju sa korisnikomu smislu mogućnosti unosa podataka od strane korisnika i dinamičku obradu unetih podataka
- HTML predstavlja ograničavajući faktor u aplikacijama koje zahtevaju dinamičku obradu unetih podataka, pa je korišćenje JS u ovakvim slučajevima neophodno.

JAVASCRIPT NA HTML STRANI

- Ukoliko ne koristimo JS provera podataka bi se događala na sledeći način
  - Korisnik unese podatak
  - Podatak odlazi do servera
  - Server prihvata podatak i obrađuje ga
  - Server šalje odgovor klijentu da li je podatak ispravan ili ne
- U ovakvim slučajevima svaki unešen podatak mora biti obrađen od strane servera za šta je potrebna stalna komunikacija klijent-server
- Ako koristimo JS za proveru podataka
  - Korisnik unese podatak
  - Podatak se odmah proverava na klijentskoj mašini pre slanja na server
  - Ukoliko podatak nije dobar JS obaveštava korisnika bez da je zahtev otišao do servera

ISTORIJA JS-A

- Decembar 1995 – Netscape i Sun su predstavili jezik JavaScript, originalno nazvan LiveScript. Kod napisan u ovom jeziku se mogao izvršavati na veb čitaču **Netscape Navigator**
- **Microsoft** je na osnovu javne dokumentacije implementirao svoju, skoro identičnu, verziju jezika koju je nazvao JScript za svoj **Internet Explorer**
- Udruženje **ECMA** (European Computer Manufacturer Association) je usvojilo standard za ovaj jezik i nazvalo ga ECMAScript 1.0
- Danas je aktivan ECMAScript 6

JS DANAS

- JS je, tokom svog postojanja, pretrpeo značajan evolutivni pomak. Iako i danas ima istu namenu (uvođenje dinamike na strani klijenta) sada su njegove mogućnosti daleko veće a upotreba daleko učestalija
- U ovom trenutku nemoguće je zamisliti veb aplikaciju bez JS
- Postoje framework rešenja bazirana samo na JS
- JS je našao svoje mesto i u kontekstu serverskih tehnologija (node.js, spark .....)
- Postoje aplikacije bazirane samo na JS (unity)

ALATI ZA RUKOVANJE

- Bilo koji tekst editor
- Bilo koji alat za rukovanje HTML-om (dreamweaver, front page ...)
- Konzola u pretraživaču (Debug bar, Firebug.....)
- Notepad++, PHPStrom, WebStorm, Visual Studio Code .....

GDE SE POJAVLJUJE JS

- U HTML dokumentima
- U vidljivom i nevidljivom delu strane (head, body)
- U zasebnom dokumentu kao biblioteka
- U zasebnom dokumentu na mreži

KAKO SE IMPLEMENTIRA

- Direktnim unosom JS koda na HTML strani (u okviru **script** taga)

```
<script>
    //Ovde ide JS kod
</script>
```

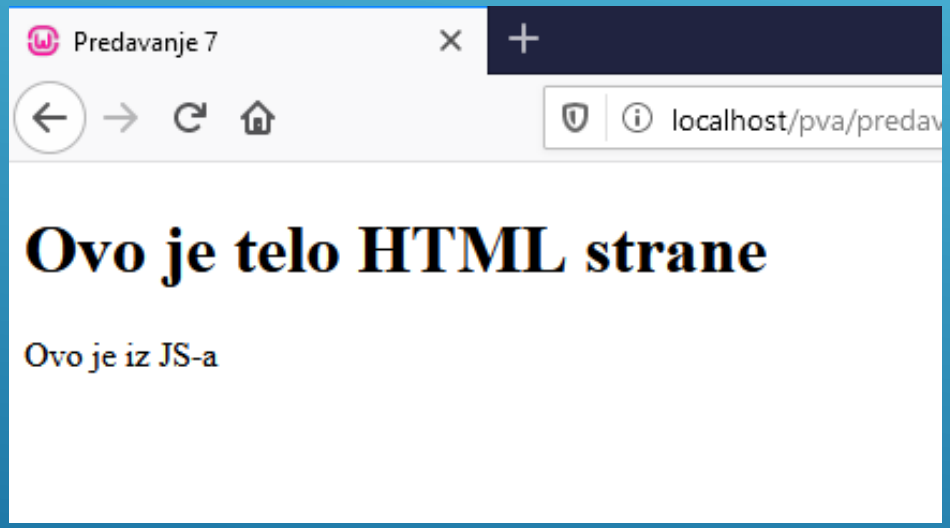
- Dodavanjem reference na eksterni fajl (u okviru **script** taga, atribut **src**)

```
<script src="putanja/do/script.js" type="text/javascript"></script>
```

IMPLEMENTACIJA JS U OKVIRU HTML

- Direktni kod unosa na HTML stranu

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
</head>
<body>
<h1>Ovo je telo HTML strane</h1>
<script>
    document.write("Ovo je iz JS-a");
</script>
</body>
</html>
```

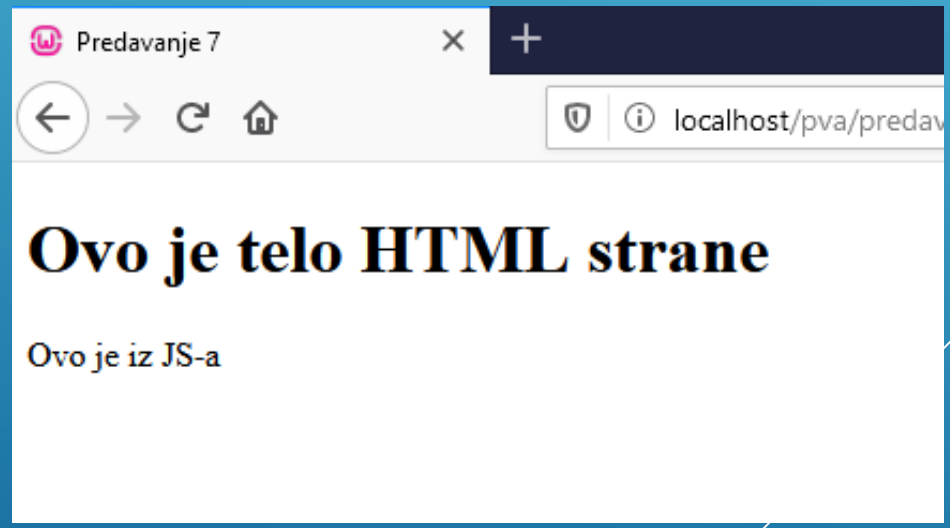


IMPLEMENTACIJA JS U OKVIRU POSEBNE DATOTEKE

- U datoteci script.js se malazi JS kod

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
</head>
<body>
<h1>Ovo je telo HTML strane</h1>
<script src="script.js"></script>
</body>
</html>
```

```
//script.js
document.write("Ovo je iz JS-a");
```



ISPISIVANJE TEKSTA

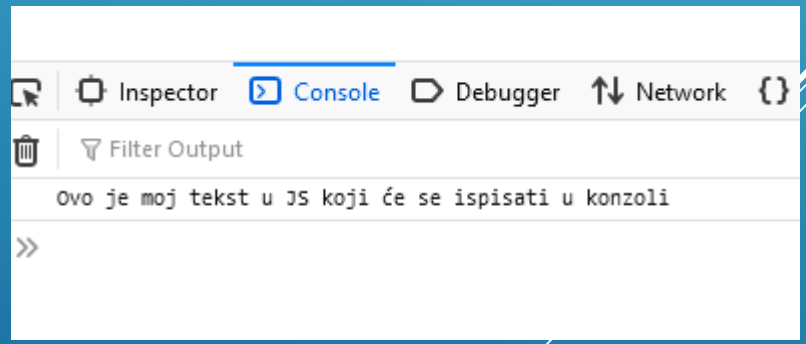
- Ispisivanje teksta je neophodno u svakom programskom jeziku, pa tako i u JS. Možemo ispisati poruku korisniku, proveriti vrednost promenljive, kreirati objekat .....

```
<script>
    let tekst="Ovo je moj tekst u JS";
    document.write(tekst);
</script>
```

ISPISIVANJE U KONZOLI

- Ako samo želimo da proverimo vrednost neke promenljive, nema potrebe da to ispisujemo na stranici. Možemo evidentirati u konzoli.

```
<script>
    let tekst="Ovo je moj tekst u JS koji će se ispisati u konzoli";
    console.log(tekst);
</script>
```



KOMENTARI

- Postoji mogućnost komentarisanja jednog reda i mogućnost komentarisanja više redova
  - Jedan red - //
  - Više redova - /*.....*/

PROMENLJIVE

- Imena promenljivih (varijable, identifikatori) u JS su osetljivi na mala i velika slova (Case Sensitive)
- Postoje pravila za imenovanje promenljivih
  - Ime promenljive mora počinjati slovom, simbolom „$“, ili donjom crtom, sve ostalo je nedozvoljeno
  - U okviru imena se, pored velikih i malih slova engleske abecede, mogu koristiti i brojevi
  - U okviru imena ne sme biti ni jedan specijalan simbol, osim donje crte (space, navodnici, operatori.....)
  - Rezervisane reči se ne mogu koristiti kao imena promenljivih (if, for, function .....)

```
<script>
    let mojaPromenljiva=5;
</script>
```

TIPOVI PODATAKA

- JS nije strogo tipiziran jezik, jedna promenljiva može u sebi imati više tipova podataka tokom izvršavanja skripta
- JS interpreter automatski prepoznaje tip podatka na osnovu njegove vrednosti
- Od ECMAScript 6 mogu se definisati tipovi podataka, u zavisnosti od potrebe
- Od tipova podataka JS podržava:
  - Celi brojevi
  - Realni brojevi
  - Stringovi
  - Logički tip

OPERATORI

- Operatori predstavljaju specijalne karaktere koji definišu koju operaciju treba izvršiti nad operandima. Operandi mogu biti promenljive, izrazi i konstante
- Operatori se mogu klasifikovati u sledeće grupe
  - Aritmetički operatori
  - Operatori na nivou bita
  - Logički operatori
  - Operatori poređenja

ARITMETIČKI OPERATORI

Operator	Opis	Operator	opis
+	Sabiranje	+=	Sabiranje i dodela
-	Oduzimanje	-=	Oduzimanje i dodela
*	Množenje	*=	Množenje i dodela
/	Deljenje	/=	Deljenje i dodela
%	Moduo	%=	Moduo i dodela
++	inkrement	--	dekrement

OPERATORI NA NIVOU BITA

Operator	Izraz	Opis
Logičko I (AND)	a&b	Rezultat 1 ako su oba bita 1
Logičko ILI (OR)	a | b	Rezultat je 0 ako su oba bita 0
Logičko ekskluzivno ILI (XOR)	a^b	Rezultat je 1 ako su bitovi različiti
Logičko NE (NOT)	~a	Komplementira bitove operanda a
Pomeranje ulevo	a<<2	Pomeriće operand a u levo za dva mesta. Prazna mesta popunjava 0
Pomeranje udesno	a>>2	Pomeriće operand a u desno za dva mesta. Prazna mesta popunjava MSB bitom

LOGIČKI OPERATORI

Operator	Izraz	Opis
I (&&)	exp1&&exp2	Rezultat je TRUE ako su oba operanda TRUE
ILI (||)	exp1||exp2	Rezultat je FALSE ako su oba operanda FALSE
NE (!)	!expr1	Rezultat je komplement expr1

OPERATORI POREĐENJA

Operator	Izraz	Opis
Jednakost (==)	x==y	Rezultat je TRUE ako x i y imaju iste vrednosti
Nejednakost (!=)	x!=y	Rezultat je TRUE ako x i y nemaju iste vrednosti
Veće (>)	x>y	Rezultat je TRUE ako je x veće od y
Manje (<)	x<y	Rezultat je TRUE ako je x manje od y
Manje ili jednako (<=)	x<=y	Rezultat je TRUE ako je x manje ili jednako y
Veće ili jednako (>=)	x>=y	Rezultat je TRUE ako je x veće ili jednako y
Identičnost (===)	x===y	Rezultat je TRUE ako x i y imaju iste vrednosti i ako su istog tipa
Neidentičnost (!==)	x!==y	Rezultat je TRUE ako x i y nemaju iste vrednosti ili ako nisu istog tipa

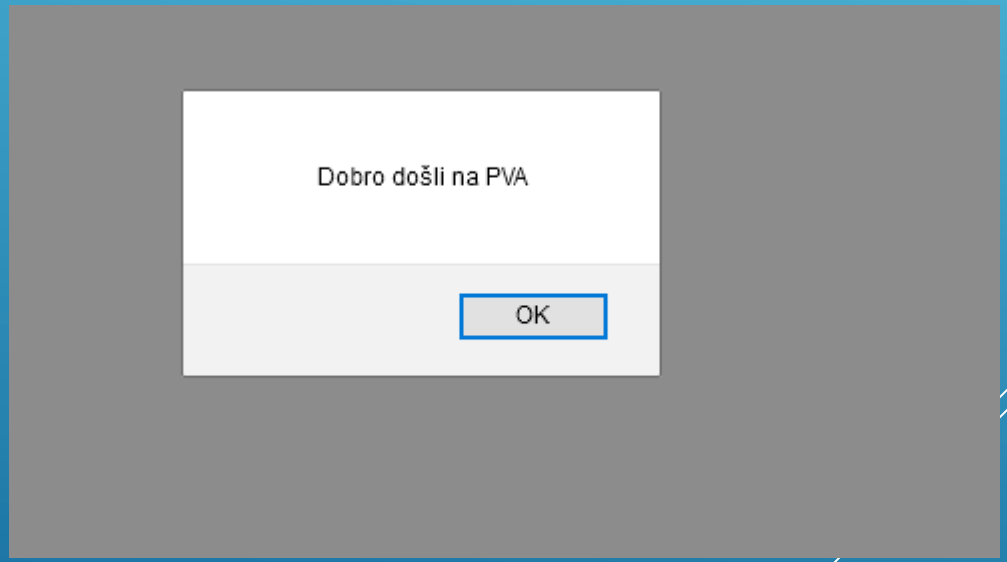
UGRAĐENE FUNKCIJE

- JS ima svoje ugrađene funkcije.
- Neke od najkorišćenijih su:
  - alert()
  - confirm()
  - prompt()

ALERT()

- alert() funkcija služi da se korisnik obavesti o nekom koraku.
- To je modalni prozor i rad se ne može nastaviti dok korisnik ne pročita obaveštenje
- Funkcija ne vraća ništa

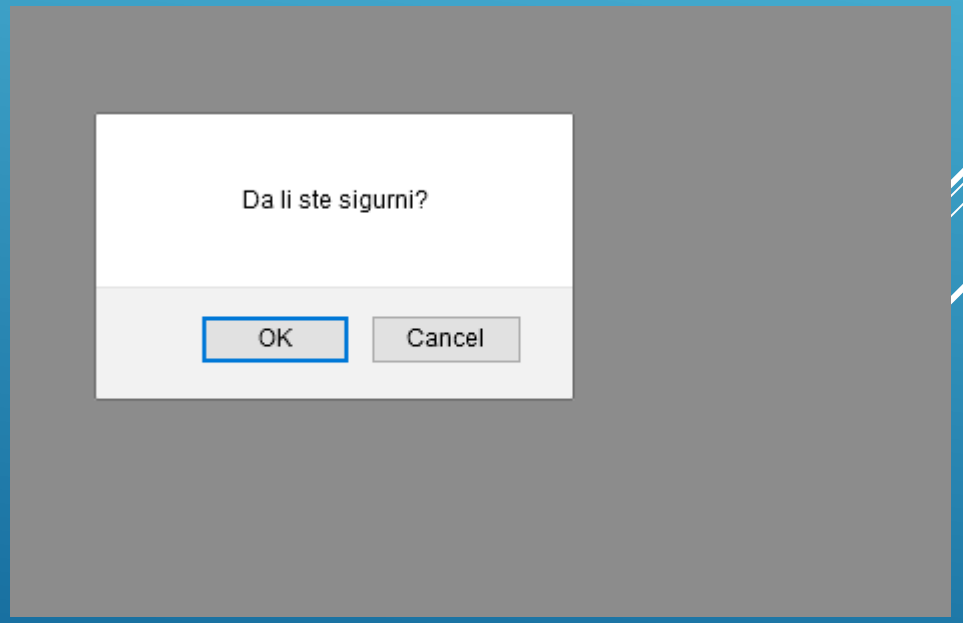
```
<script>
    alert("Dobro došli na PVA");
</script>
```



CONFIRM()

- Confirm funkcija služi za dobijanje potvrde od strane korisnika za neku akciju.
- Obavezna je kada se briše neki podatak
- Povratna informacija je TRUE ako je korisnik kliknuo na OK/YES ili FALSE ako je korisnik kliknuo na NO/CANCEL

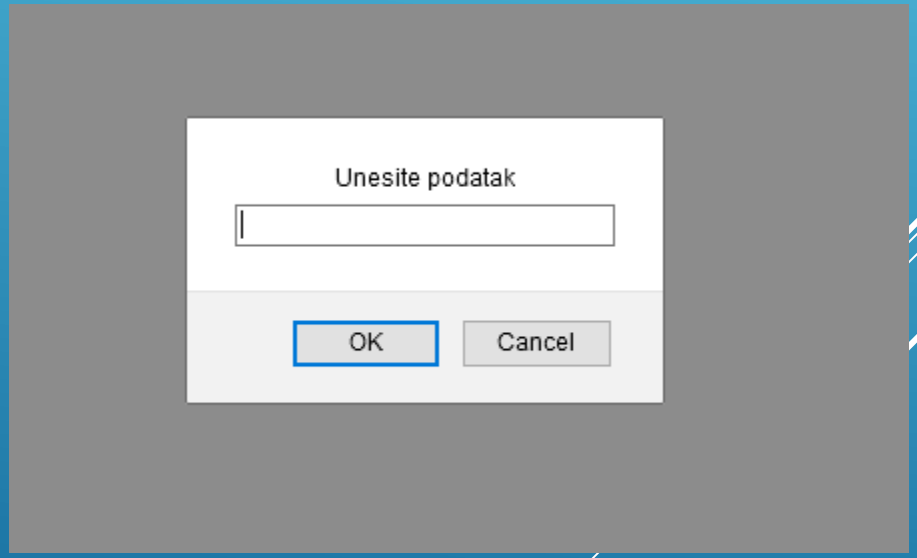
```
<script>
    let a=confirm("Da li ste sigurni?");
    if(a) alert("Vidim da ste sigurni!!!");
    else alert("Dvoumite se");
</script>
```



PROMPT()

- prompt() funkcija služi da dobijemo informaciju od korisnika koja je neophodna za nastavak rada programa.
- Povratne informacije su string, ako je korisnik kliknuo na OK, ili NULL podatak, ako je kliknuo na CANCEL

```
<script>
    let a=prompt("Unesite podatak");
    if(a==null) alert("Izabrali ste CANCEL");
    else alert(a);
</script>
```



KORISNIČKE FUNKCIJE

- U js se primenom rezervisane reči **function** deklariše funkcija sa zadatim parametrimakoji mogu biti numeričkog tipa, stringovi ili objekti.
- Opšti oblik je:

```
<script>
    function imeFunkcije([parametar1], [parametar2], ......)
    {
        //telo fukcije
    }
</script>
```

KORISNIČKE FUNKCIJE

- Korisničke funkcije se, po ulaznim parametrima, dele na :
  - Bez ulaznih parametara
  - Sa obaveznim ulaznim parametrima
  - Sa opcionim ulaznim parametrima
- Po izlazu
  - Bez izlaza – funkcija ne vraća ništa glavnom programu
  - Sa izlazom - funkcija vraća podatak/podatke glavnom programu

KORISNIČKE FUNKCIJE – PODELA PO ULAZU

- Bez ulaznih parametara

```
<script>
    function bezUlaznihParametara()
    {
        alert("ovo je funkcija bez ulaznih parametara");
    }
    bezUlaznihParametara();
</script>
```

- Sa obaveznim ulaznim parametrima

```
<script>
    function obavezniUlazniParametri(a, b)
    {
        alert("Uneti parametri su: " + a + " " + b);
    }
    obavezniUlazniParametri(5,3);
</script>
```

- Sa opcionim ulaznim parametrima

```
<script>
    function opcioniUlazniParametri(a, b=0)
    {
        alert("Uneti parametri su: " + a + " " + b);
    }
    opcioniUlazniParametri(5);
</script>
```

KORISNIČKE FUNKCIJE – PODELA PO IZLAZU

- Bez izlaza

```
<script>
    function bezIzlaza(a, b)
    {
        alert("Uneti parametri su: " + a + " " + b);
    }
    bezIzlaza(5, 3);
</script>
```

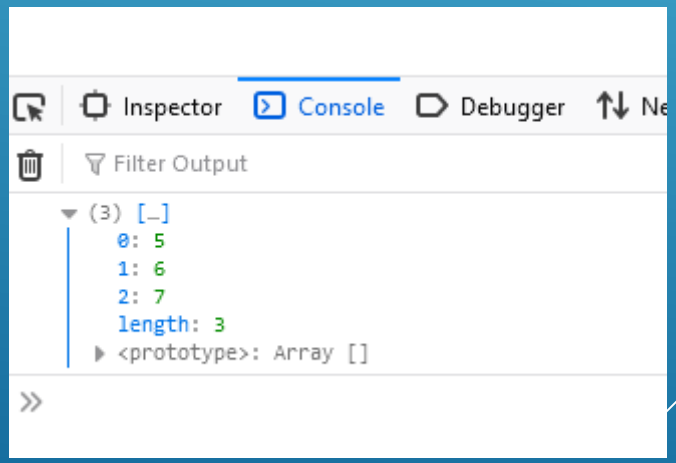
- Sa izlazom

```
<script>
    function saIzlazom(a, b)
    {
        let zbir=a+b;
        return zbir;
    }
    let a=saIzlazom(5, 3);
</script>
```

FUNKCIJE SA IZLAZOM - RETURN

- Rezervisana reč **return** označava izlazak iz funkcije i vraćanje podatka koji se nalazi iza nje
- U korisničkoj funkciji može biti više **return** funkcija, ali se iz funkcije izlazi kad se naiđe na prvu
- Podatak može biti celobrojna vrednost, realna vrednost, string, niz ili objekat

```
<script>
    function nizKaoIzlaz()
    {
        let zbir=[5, 6, 7];
        return zbir;
    }
    let a=nizKaoIzlaz();
    console.log(a);
</script>
```



KONTROLA TOKA

- Za kontrolu toka programa možemo koristiti:
  - **IF** strukturu
  - **SWITCH** strukturu
  - Petlje

IF NAREDBA

- IF naredba se koristi za donošenje odluka na osnovu kojih će se neki delovi koda izvršiti, odnosno neki delovi koda se neće izvršiti, u zavisnosti od navedenog uslova
- Osnovna sintaksa:

```
If(uslov)
{
    bloknaredbi1
}
else
{
    bloknaredbi2
}
```

SWITCH NAREDBA

- SWITCH naredba se koristi za kontrolu toka i menja više IF naredbi
- Za razliku od IF naredbe gde se proverava uslov koji može biti promenljiva ili izraz u SWITCH naredbi se proverava samo vrednost promenljive
- Sintaksa je:

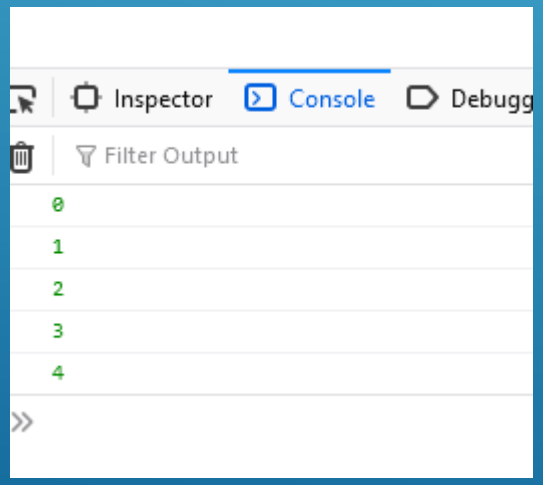
```
<script>
    let a=prompt("Unesite vrednost");
    switch(a)
    {
        case "1":
            alert("Uneli ste 1");
            break;
        case "5":
            alert("Uneli ste 5");
            break;
        case "pera":
            alert("Uneli ste 'pera'");
            break;
        default:
            alert("Niste uneli ništa od očekivanog");
    }
</script>
```

PETLJE

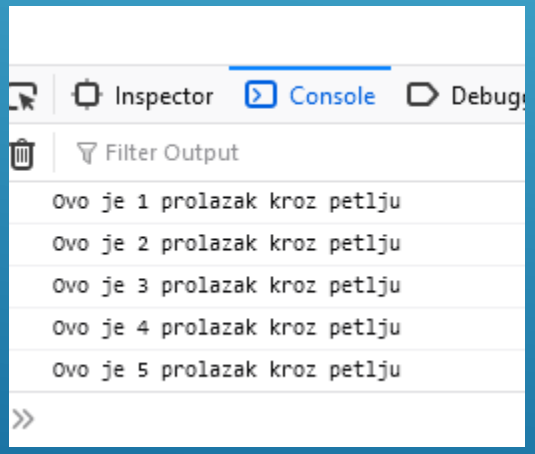
- Petlje služe za više ponavljanja jednog bloka naredbi
- U JS postoji nekoliko tipova petlji
  - FOR()
  - WHILE()
  - DO....WHILE()
  - FOR...IN – čita indekse niza
  - FOR...OF – čita vrednosti niza

PETLJE

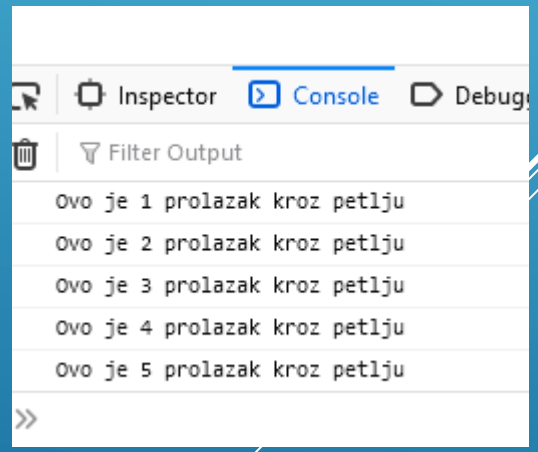
```
<script>
for(let i=0;i<5;i++)
    console.log(i);
</script>
```




```
<script>
let i=0;
while(i<5)
    console.log("Ovo je " + ((i++)+1) +
" prolazak kroz petlju");
</script>
```




```
<script>
let i=0;
do{
    console.log("Ovo je "+ ((i++)+1) + "
prolazak kroz petlju");
}while(i<5);
</script>
```



NIZOVI

- Nizovi u JS su promenljive koje mogu imati više vrednosti u jednom trenutku
- Svaka vrednost u nizu se naziva član niza
- Svaki član niza ima svoj ključ/indeks koji određuje njegovo mesto u nizu
- Indeksi u JS programskom jeziku počinju od 0
- JS ima samo numeričke nizove (indeksi su brojevi)
- Asocijativni nizovi kao takvi ne postoje, ali se mogu zameniti neimenovanim objektima

DEKLARACIJA NIZA

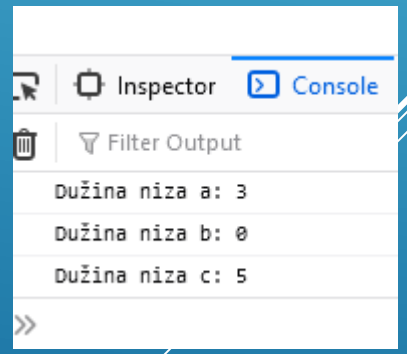
- Niz se u JS deklariše sa rezervisanom reči **Array()** (postoji i skraćeni oblik bez rezervisane reči)
- Možemo definisati prazan niz, pa mu naknadno dodavati članove a možemo dodeliti članove nizu prilikom definicije

```
<script>
    //definicije dva prazna niza
    let a=new Array();
    let b=[];
    //dodela vrednosti nizu a
    a[0]=1;
    a[1]="pera";
    a.push(5.6);
    //Definicija niza i dodela vrednosti
    let c=new Array(5, 5.8, "Laza");
</script>
```

DUŽINA NIZA

- Dužinu niza možemo dobiti svojstvom **length**

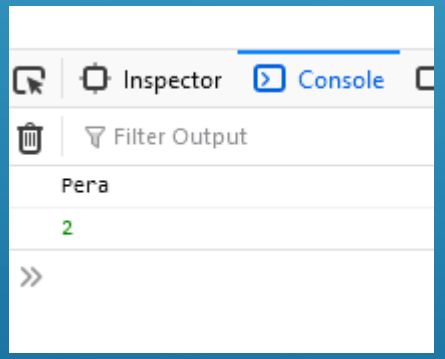
```
<script>
    //definicije dva prazna niza
    let a=new Array();
    let b=[];
    //dodela vrednosti nizu a
    a[0]=1;
    a[1]="pera";
    a.push(5.6);
    //Definicija niza i dodela vrednosti
    let c=new Array(5, 5.8, "Laza", true, "Mile");
    console.log("Dužina niza a: " + a.length);
    console.log("Dužina niza b: " + b.length);
    console.log("Dužina niza c: " + c.length);
</script>
```



ASOCIJATIVNI NIZOVI

- Ako želimo da radimo sa asocijativnim nizovima moramo koristiti neimenovane objekte
- Definicija takvih nizova je kao definicija objekata
- Pristup elementima niza možemo uraditi na dva načina: sa uglastim zagradama, kao i numerički nizovi i sa tačkom, kao svojstvu objekta

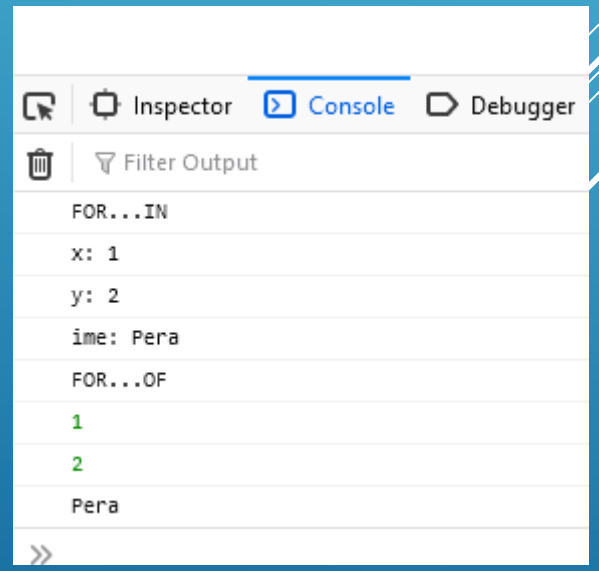
```
<script>
    let a={x:1, y:2, ime:"Pera"};
    console.log(a.ime);
    console.log(a['y']);
</script>
```



PETLJE ZA RAD SA NIZOVIMA

- Petlje FOR...IN i FOR...OF služe za rad sa nizovima
- FOR...IN prolazi kroz indekse niza – koristi se za asocijativne nizove
- FOR...OF prolazi kroz vrednosti niza – koristi se za numeričke nizove

```
<script>
    let a={x:1, y:2, ime:"Pera"};
    console.log("FOR...IN");
    for(indeks in a)
        console.log(indeks + ": " + a[indeks]);
    a=new Array(1, 2, "Pera");
    console.log("FOR...OF");
    for(vrednost of a)
        console.log(vrednost);
</script>
```

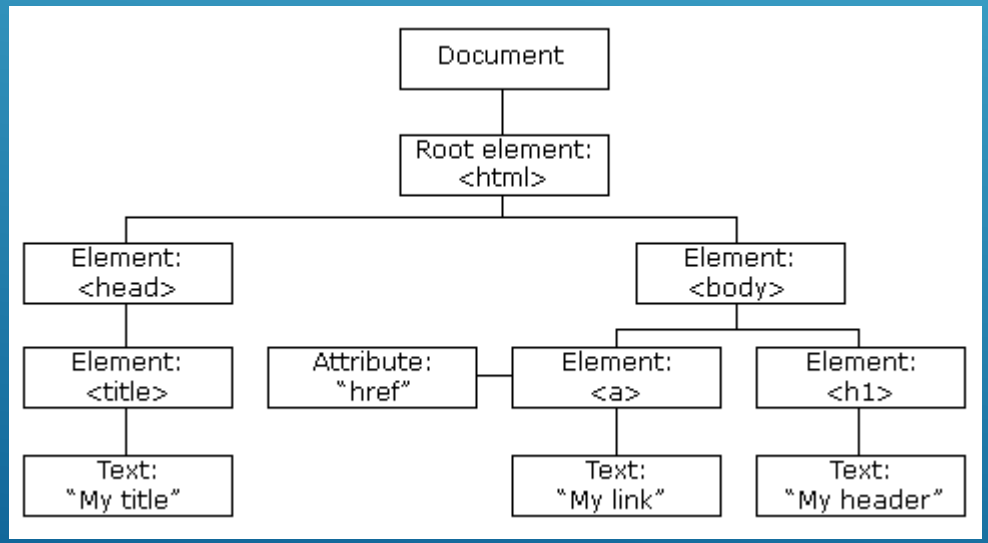


DOCUMENT OBJECT MODEL - DOM

- XML dokument pretvoren u objektni model
- Aktivira se NAKON parsiranja strane
- U JS kompletna struktura se nalazi u objektu **document**

STRUKTURA DOM-A

- Sadrži nodove, podnodove (decu nodove) i atribute
- Svaki nod može sadržati atribute
- Neki nodovi mogu imati registrovane **event**-e (događaje)



RUKOVANJE DOM-OM

- Za rukovanje DOM-om postoje:
  - DOM metode
  - Navigacija u DOM-u
  - Manipulacija DOM-om

DOM METODE

- DOM metode su komande kojima možemo vršiti manipulaciju nad različitim članovima DOM objekta
  - createElement('elementName')
  - createTextNode('value')
  - appendChild(element) .....

NAVIGACIJA PO ELEMENTU

- Referenca po ID-ju elementa

document.getElementById('elementId')

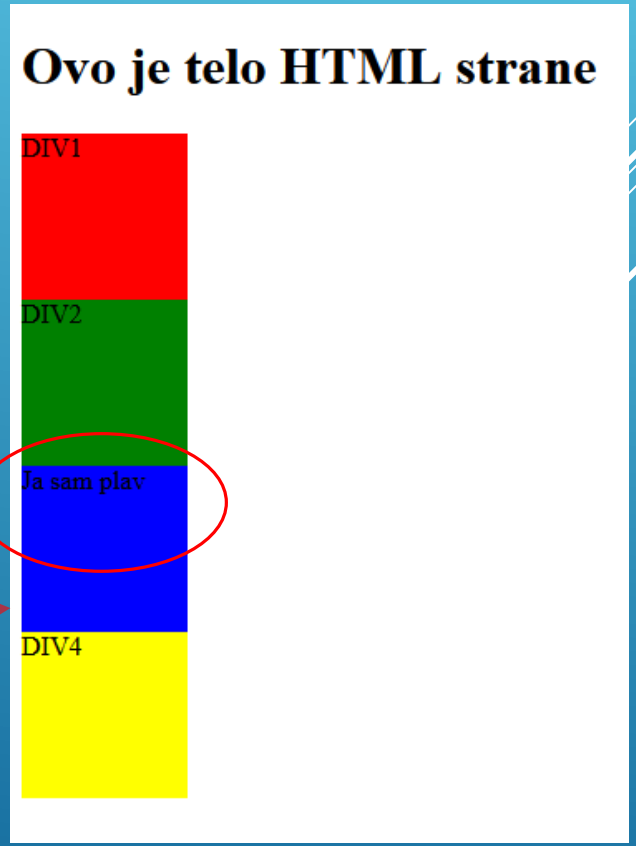
- Metodu getElementById koristimo da bi se referencirali na elemenat čiji id nam je poznat
- ID mora biti jedinstven na toj HTML stranici

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
</head>
<body>
<h1>Ovo je telo HTML strane</h1>
<p id="paragraf">Lorem ipsum ......</p>
</body>
</html>
<script>
    let p=document.getElementById("paragraf");
</script>
```

NAVIGACIJA PO ELEMENTU

- Metodom getElementById() se referenciramo na element koji nam je od važnosti
- Nakon toga možemo menjati svojstva elementa

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
</head>
<body>
<h1>Ovo je telo HTML strane</h1>
<div id="prvi" style="background-color:red; width:100px;height:100px">DIV1</div>
<div id="drugi" style="background-color:green; width:100px;height:100px">DIV2</div>
<div id="treci" style="background-color:blue; width:100px;height:100px">DIV3</div>
<div id="cetvrti" style="background-color:yellow; width:100px;height:100px">DIV4</div>
</body>
</html>
<script>
    let d=document.getElementById("treci");
    d.innerHTML="Ja sam plav";
</script>
```



NAVIGACIJA PO ELEMENTU

- Kada radimo navigaciju po elementu OBAVEZNO je da se script tag nalazi nakon html taga
- Prvo se iscrtava stranica, pa tek nakon toga pali DOM

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
</head>
<body>
<h1>Ovo je telo HTML strane</h1>
<div id="prvi" style="background-color:red;
width:100px;height:100px">DIV1</div>
<div id="drugi" style="background-color:green;
width:100px;height:100px">DIV2</div>
<div id="treci" style="background-color:blue;
width:100px;height:100px">DIV3</div>
<div id="cetvrti" style="background-color:yellow;
width:100px;height:100px">DIV4</div>
</body>
</html>
<script>
    let d=document.getElementById("treci");
    treci.innerHTML="Ja sam plav";
</script>
```

```
<script>
    let d=document.getElementById("treci");
    treci.innerHTML="Ja sam plav";
</script>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
</head>
<body>
<h1>Ovo je telo HTML strane</h1>
<div id="prvi" style="background-color:red;
width:100px;height:100px">DIV1</div>
<div id="drugi" style="background-color:green;
width:100px;height:100px">DIV2</div>
<div id="treci" style="background-color:blue;
width:100px;height:100px">DIV3</div>
<div id="cetvrti" style="background-color:yellow;
width:100px;height:100px">DIV4</div>
</body>
</html>
```

NAVIGACIJA PO ELEMENTU

- Ukoliko želimo da dohvatimo više elemenata moramo koristiti druge metode
  - getElementsByClassName('imeklase')
  - getElementsByTagName('imetaga')
  - getElementsByName('imeelemenata')
- Sve ove metode vraćaju niz elemenata, čak i ako nađu samo jedan
- Ako želimo da radimo sa ovim metodama, moramo rezultat tretirati kao niz

NAVIGACIJA PO ELEMENTU – PO IMENU KLASE

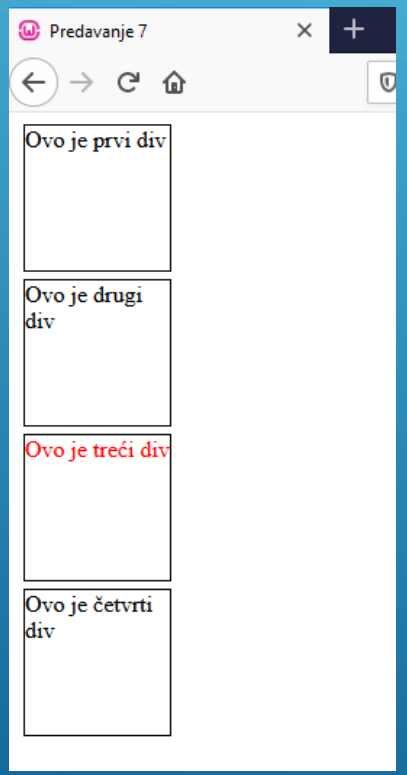
```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
    <style>
        .test{
            width: 100px;
            height: 100px;
            border:1px solid black;
            margin:5px;
        }
    </style>
</head>
<body>
<div class="test">Ovo je prvi div</div>
<div class="test">Ovo je drugi div</div>
<div class="test">Ovo je treći div</div>
<div class="test">Ovo je četvrti div</div>
</body>
</html>
```

```
<script>
    let e=document.getElementsByClassName('test');
    e[2].style.color="red";
</script>
```

NAVIGACIJA PO ELEMENTU – PO IMENU

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
    <style>
        .test{
            width: 100px;
            height: 100px;
            border:1px solid black;
            margin:5px;
        }
    </style>
</head>
<body>
<div class="test" name="ime">Ovo je prvi div</div>
<div class="test">Ovo je drugi div</div>
<div class="test" name="ime">Ovo je treći div</div>
<div class="test" name="ime">Ovo je četvrti div</div>
</body>
</html>
```

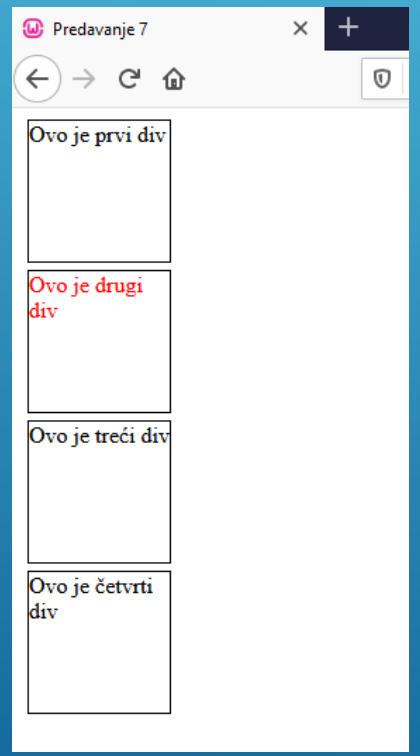
```
<script>
    let e=document.getElementsByName('ime');
    e[1].style.color="red";
</script>
```



NAVIGACIJA PO ELEMENTU – PO IMENU TAGA

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
    <style>
        .test{
            width: 100px;
            height: 100px;
            border:1px solid black;
            margin:5px;
        }
    </style>
</head>
<body>
<div class="test" name="ime">Ovo je prvi div</div>
<div class="test">Ovo je drugi div</div>
<div class="test" name="ime">Ovo je treći div</div>
<div class="test" name="ime">Ovo je četvrti div</div>
</body>
</html>
```

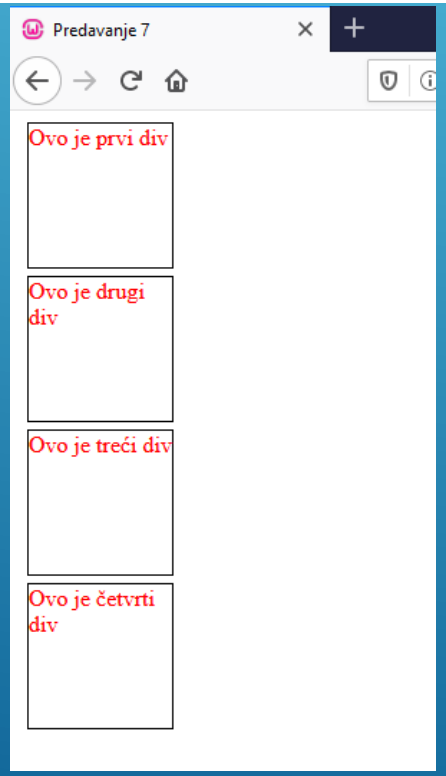
```
<script>
    let e=document.getElementsByTagName('div');
    e[1].style.color="red";
</script>
```



NAVIGACIJA PO ELEMENTU – PROLAZAK KROZ SVE ELEMENTE

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
    <style>
        .test{
            width: 100px;
            height: 100px;
            border:1px solid black;
            margin:5px;
        }
    </style>
</head>
<body>
<div class="test" name="ime">Ovo je prvi div</div>
<div class="test">Ovo je drugi div</div>
<div class="test" name="ime">Ovo je treći div</div>
<div class="test" name="ime">Ovo je četvrti div</div>
</body>
</html>
```

```
<script>
    let e=document.getElementsByTagName('div');
    for(let i=0;i<e.length;i++)
        e[i].style.color='red';
</script>
```



NAVIGACIJA PO ELEMENTU - ALTERNATIVA

- Pored navedenih metoda koje postoje u svim verzijama JS-a, od verzije ECMAScript 6 postoje i alternativni načini referenciranja na elemente
- Koriste se CSS selektori
  - #id – referenca na element sa id-jem *id*
  - .imeKlase – referenca na elemente koji imaju klasu *imeKlase*
  - imeTaga – referenca na elemente čiji je tag *imeTaga*
  - .....
- Metode su **querySelector()** i **querySelectorAll()**

NAVIGACIJA PO ELEMENTU - PRIMERI

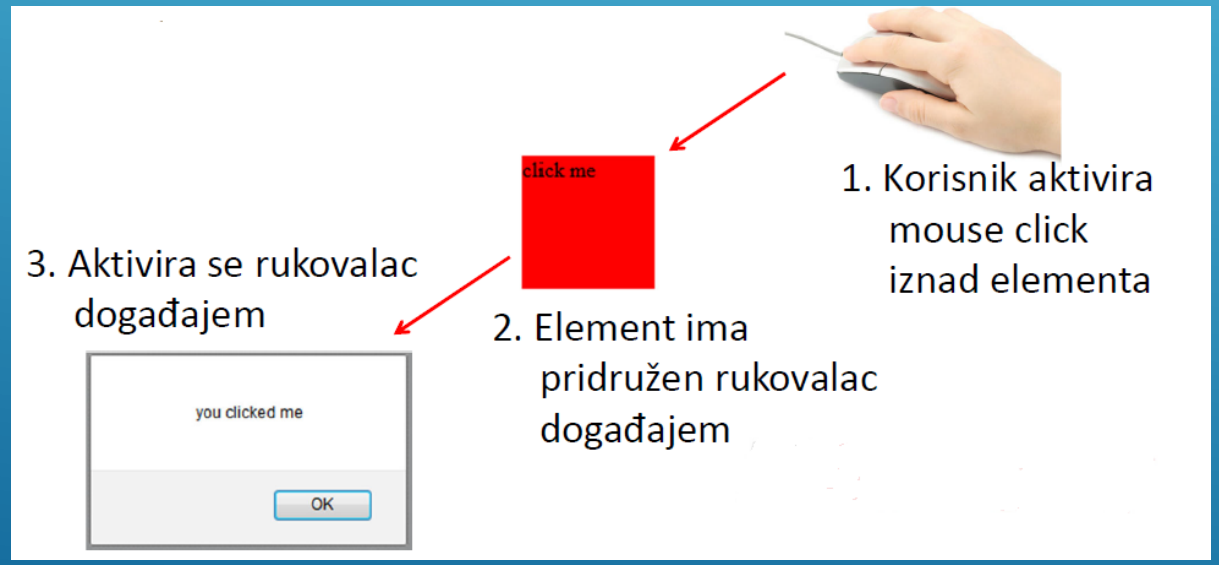
- document.querySelector('#treci') – document.getElementById('treci')
- document.querySelectorAll('.test') – document.getElementsByClassName('test')
- document.querySelectorAll('div') – document.getElementsByTagName('div')
- Pored navedenih korišćenjkem ove dve funkcije se možemo referencirati na elemente koristeći sve CSS selektore
  - document.querySelectorAll("a[target]") – referenca na sve **a** elemente sa atributom **target**
  - document.querySelectorAll("div > p") – referenca na sve p elemente koji su deca div elementa
  - document.querySelectorAll("h2, div, span") – referenca na sve h2, div i span elemente
  - .....

HTML DOGAĐAJI

- Svi ili gotovo svi elementi u HTML-u u stanju su da reaguju na neke događaje. Ovi događaji su najčešće vezani za korisnički interfejs i tiču se miša ili tastature, ali takođe, i nekih elementarnih pojava na strani (skrolovanje, fokus, napuštanje elementa .....)
- Događaje možemo uhvatiti kombinacijom stributa nad elementom čiji nas događaji interesuju i koda koji će taj događaj obraditi

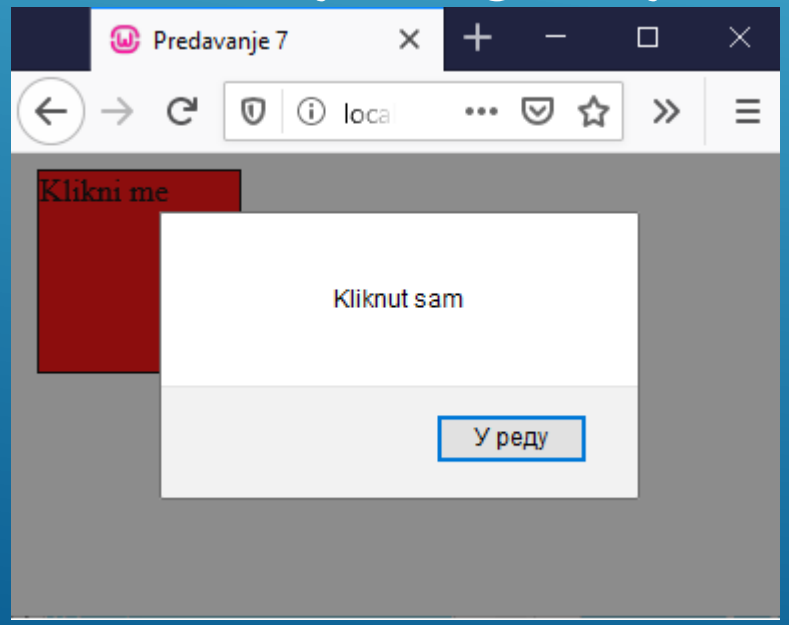
HTML DOGAĐAJI

- Kod koji se aktivira kao rezultat događaja se naziva rukovalac događajem



HTML DOGAĐAJI

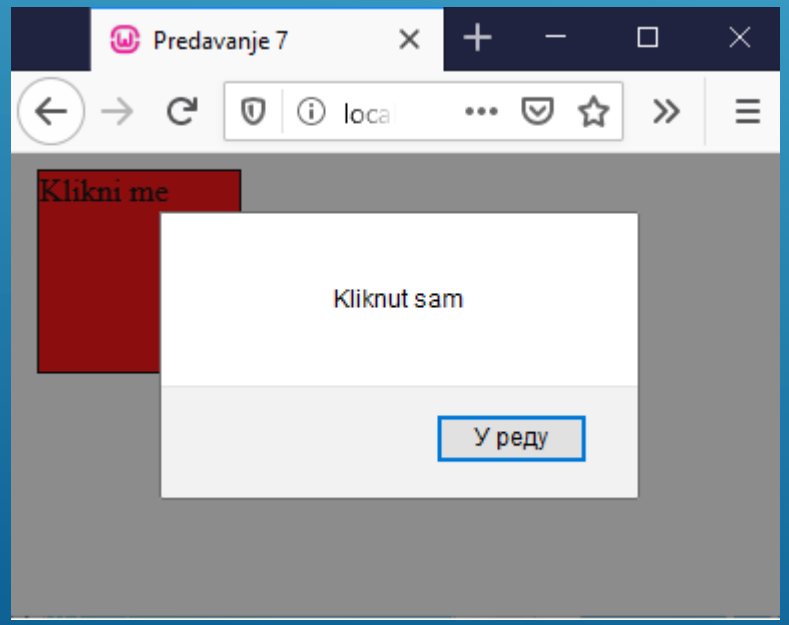
- Rukovaoci događajem mogu biti definisani deklarativno, unutar HTML taga, pomoću HTML atributa
- U tom slučaju HTML atribut sadrži JS kod koji će se izvršiti prilikom aktivacije događaja




```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
    <style>
        .test{
            background-color: red;
            width: 100px;
            height: 100px;
            border:1px solid black;
            margin:5px;
        }
    </style>
</head>
<body>
<div class="test" name="ime" onclick="alert('Kliknut sam')">Klikni me</div>
</body>
</html>
```

HTML DOGAĐAJI

- Najčešće u atributu elementa stoji ime funkcije koja je definisana u script tagu




```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
    <style>
        .test{
            background-color: red;
            width: 100px;
            height: 100px;
            border:1px solid black;
            margin:5px;
        }
    </style>
</head>
<body>
<div class="test" name="ime" onclick="klik();">Klikni me</div>
</body>
</html>
<script>
function klik()
{
    alert("Kliknut sam");
}
</script>
```

HTML ДОГАЂАЈИ

- Postoji mnoštvo događaja koje možemo uhvatiti, ali njihov tačan broj se ne može utvrditi jer ne postoje svi događaji na svim pretraživačima
- Ipak, većina standardnih događaja postoji i obrađuje se na isti ili sličan način
- Najčešći događaji:
  - Događaji miša (onclick, onmouseover, onmouseenter, onmouseup.....)
  - Događaji tastature (onkeyup, onkeydown, onkeypress)
  - Događaji dokumenta i prozora (onload, onresize ....)
  - Događаji kontrola (onblur, onfocus, onchange ....)

HTML ДОГАЂАЈИ - ONCLICK

- Na ovom primeru možemo videti da se klikom na jedan div, njegov sadržaj kopira u drugi div

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
    <style>
        .test{
            background-color: red;
            width: 100px;
            height: 100px;
            border:1px solid black;
            margin:5px;
        }
    </style>
</head>
<body>
<div class="test" id="div1" onclick="klik()">Klikni
me</div>
<div class="test" id="div2"></div>
</body>
</html>
<script>
function klik()
{
    let div1=document.querySelector("#div1");
    let div2=document.querySelector("#div2");
    div2.innerHTML=div1.innerHTML;
}
</script>
```

HTML FORME

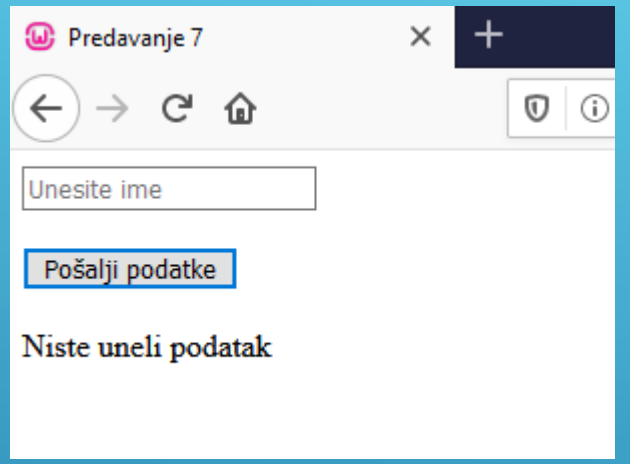
- Tag **form** obezbeđuje interaktivnost korisnika sa stranicom
- Služi za unos podataka od strane korisnika
- Ima dva obavezna atributa: **action** i **method**
  - action – govori gde će podaci koje je korisnik uneo biti poslati
  - method – govori na koji način će podaci biti poslati (GET, POST)

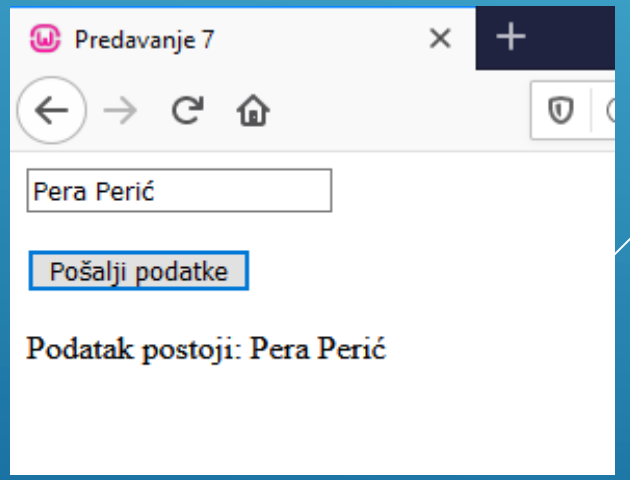
PROVERA PODATAKA NA FORMI

- Sve podatke koje korisnik šalje treba proveriti u cilju zaštite aplikacije
- PHP skripte koje obrađuju podatke MORAJU imati proveru
- JS nam omogućava da se podaci provere na klijentskoj strani, pre nego što uopšte budu poslati serveru
- 99% grešaka pri unosu podataka su slučajni, ali treba da se čuvamo onih 1%
- JS će zaustaviti 99% slučajnih grešaka pre nego što odu na server
- PHP (na serverskoj strani) će zaustaviti one preostale

PROVERA PODATAKA NA FORMI - INPUT

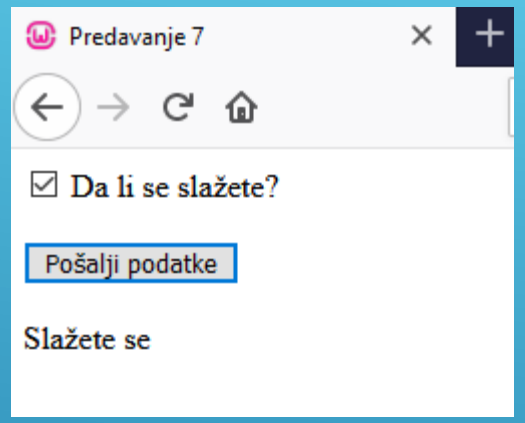
```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
</head>
<body>
<form action="index.php" method="post" name="forma" id="forma">
    <input type="text" id="ime" name="ime" placeholder="Unesite ime" />
    <br><br>
    <input type="button" onclick="proveri()" value="Pošalji podatke"/>
</form>
<br>
<div id="odgovor"></div>
</body>
</html>
<script>
function proveri()
{
    let input=document.querySelector("#ime");
    let odgovor=document.querySelector("#odgovor");
    if(input.value=="") odgovor.innerHTML="Niste uneli podatak";
    else odgovor.innerHTML="Podatak postoji: " + input.value;
}
</script>
```

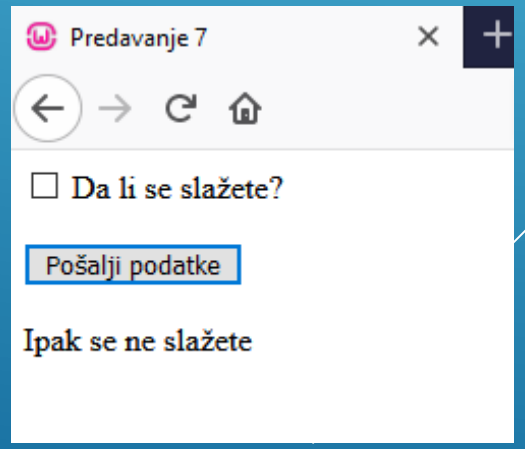




PROVERA PODATAKA NA FORMI - INPUT

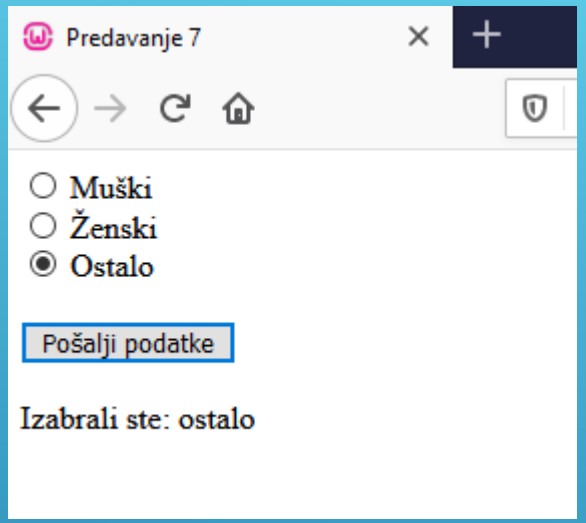
```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
</head>
<body>
<form action="index.php" method="post" name="forma" id="forma">
    <input type="checkbox" id="opcija" name="opcija" value="1"/> Da li se slažete?
    <br><br>
    <input type="button" onclick="proveri()" value="Pošalji podatke"/>
</form>
<br>
<div id="odgovor"></div>
</body>
</html>
<script>
function proveri()
{
    let input=document.querySelector("#opcija");
    let odgovor=document.querySelector("#odgovor");
    if(input.checked) odgovor.innerHTML="Slažete se";
    else odgovor.innerHTML="Ipak se ne slažete";
}
</script>
```





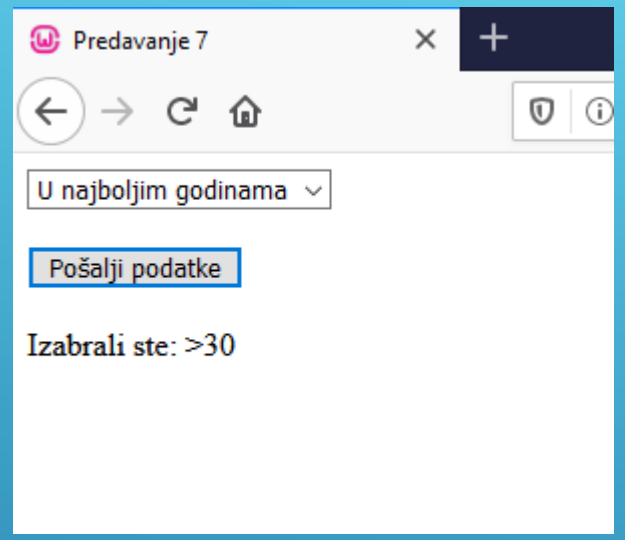
PROVERA PODATAKA NA FORMI - INPUT

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
</head>
<body>
<form action="index.php" method="post" name="forma" id="forma">
    <input type="radio" name="grupa" id="radio1" value="muski" checked> Muški<br>
    <input type="radio" name="grupa" id="radio2" value="zenski"> Ženski<br>
    <input type="radio" name="grupa" id="radio3" value="ostalo"> Ostalo<br><br>
    <input type="button" onclick="proveri()" value="Pošalji podatke"/>
</form>
<br>
<div id="odgovor"></div>
</body>
</html>
<script>
function proveri()
{
    let input=document.getElementsByName("grupa");
    let odgovor=document.querySelector("#odgovor");
    for(let i=0;i<input.length;i++)
        if(input[i].checked) odgovor.innerHTML="Izabrali ste: " + input[i].value;
    /*
    let input=document.querySelector('input[name="grupa"]:checked');
    let odgovor=document.querySelector("#odgovor");
    odgovor.innerHTML="Izabrali ste: " + input.value;
    */
}
</script>
```



PROVERA PODATAKA NA FORMI - SELECT

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
</head>
<body>
<form action="index.php" method="post" name="forma" id="forma">
    <select name="godine" id="godine">
        <option value="0" selected>--Izaberite godine--</option>
        <option value="<18">Maloletni ste</option>
        <option value="18-25">Neiskusni ste</option>
        <option value=">30">U najboljim godinama</option>
    </select><br><br>
    <input type="button" onclick="proveri()" value="Pošalji podatke"/>
</form>
<br>
<div id="odgovor"></div>
</body>
</html>
<script>
function proveri()
{
    let input=document.querySelector("#godine");
    let odgovor=document.querySelector("#odgovor");
    if(input.value=="0") odgovor.innerHTML="Niste izabrali godine";
    else odgovor.innerHTML="Izabrali ste: " + input.value;
}
</script>
```



PROVERA PODATAKA NA FORMI - PRIJAVA

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predavanje 7</title>
</head>
<body>
<form action="index.php" method="post" name="forma" id="forma">
    <input type="text" name="korime" id="korime" placeholder="Unesite korisničko ime"/>
    <br><br>
    <input type="password" name="lozinka" id="lozinka" placeholder="Unesite lozinku"/>
    <br><br>
    <input type="button" onclick="proveri()" value="Pošalji podatke"/>
</form>
<br>
<div id="odgovor"></div>
</body>
</html>
```

```
<script>
function proveri()
{
    let korime=document.querySelector("#korime");
    let lozinka=document.querySelector("#lozinka");
    let forma=document.querySelector("#forma");
    let odgovor=document.querySelector("#odgovor");
    if(korime.value=="" || lozinka.value=="")
    {
        odgovor.innerHTML="Niste uneli sve podatke";
        return false;
    }
    if(korime.value.indexOf("=")>-1 || lozinka.value.indexOf("=")>-1)
    {
        odgovor.innerHTML="Podaci sadrže nedozvoljeni znak '='";
        return false;
    }
    forma.submit();
}
</script>
```