



## IT255 - VEB SISTEMI 1

### Web korisnički interfejs – HTML5 i Javascript jezik

#### Lekcija 03

PRIRUČNIK ZA STUDENTE

# IT255 - VEB SISTEMI 1

## Lekcija 03

### *WEB KORISNIČKI INTERFEJS – HTML5 I JAVASCRIPT JEZIK*

- ✓ Web korisnički interfejs – HTML5 i Javascript jezik
- ✓ Poglavlje 1: Osnove JavaScript jezika
- ✓ Poglavlje 2: Elementi JavaScript jezika
- ✓ Poglavlje 3: HTML i JavaScript - Vežba 3
- ✓ Poglavlje 4: Domaći zadatak 3
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## UVOD - JAVASCRIPT

*Programiranje u JavaScript programskom jeziku ograničiti samo na one oblasti za koje je ovaj jezik namenjen a to je dinamička izmena teksta, reagovanje na događaje korisnika, validacija...*

Osnovni problem rada sa HTML-om je nemogućnost interakcije sa korisnikom. HTML strane su statične i nemaju interakciju. Ovo je veliki nedostatak standardnih HTML strana. JavaScript je standardizovan programski jezik koji se koristi u pregledačima.

JavaScript programski jezik nadomešćuje nedostatke standardnog HTML-a obezbeđujući elementarnu interakciju sa korisnikom. Ovaj programski jezik je modularnog tipa pa samim tim nije za komplikovanje zadatke. Iako JavaScript ima svoje objekte ovo nije objektni programski jezik. Ovaj programski jezik je takođe i interpreterski što znači da njegov kod pregledač interpretira dok se stranica prikazuje. Odnosno ne kompajlira se već se sam kod nalazi u tekstualnom obliku kad dođe do korisnika pa se implementira od strane veb pregledača u tom trenutku.

Trebalo bi programiranje u ovom programskom jeziku ograničiti samo na one oblasti za koje je ovaj jezik namenjen, a to je dinamička izmena teksta, reagovanje na događaje korisnika, validacija formi, kreiranje vizuelno atraktivnih dinamičkih efekata, a može da se koristi i za specifične situacije kada želimo da pristupimo određenim mogućnostima koje ima samo pojedinačni pregledač.

Koristi se za:

- *dinamičku izmenu teksta*
- *reagovanje na događaje korisnika*
- *validaciju formi na klijentskoj strani*
- *dinamičke vizuelne efekte*
- *detekciju korisnikovog Browser-a*

## ▼ Poglavlje 1

# Osnove JavaScript jezika

## KRATAK ISTORIJAT JEZIKA JAVASCRIPT

*JavaScript programeru omogućava mnogo veću funkcionalnost na klijentskoj strani.*

Nedostatak HTML strana je nemogućnost dinamičke obrade unetih podataka od strane korisnika. Zato se došlo do zaključka da HTML postaje ograničavajući faktor i da je potrebna nova tehnologija za realizaciju dinamičkih delova aplikacije.

Prvi pokušaj je bio pomoću serverskih komponenti, od kojih je najpopularnija bila CGI (Common Gateway Interface). Ipak, problem je predstavljala česta klijent-server komunikacija. Sve akcije se obavljaju na serverskoj strani.

Decembra 1995. godine, Netscape i Sun predstavili su jezik JavaScript 1.0, originalno nazvan LiveScript. Ovaj jezik je omogućio ne samo formatiranje podataka na klijentskoj strani, već i obradu i dinamičko izvršavanje stranica. Treba napomenuti da je implementiran deo jezika koji se izvršavao i na serverskoj strani, čime je omogućio da se ista tehnologija koristi na obe strane aplikacije, ali ovaj deo JavaScript jezika nije dostigao veću popularnost i neće se razmatrati.

Sledeći korak u popularnosti JavaScript jezika je bila Microsoft-ova implementacija u okviru čitača Internet Explorer verzije 3, pri čemu je ova verzija od strane Microsoft-a nazvana JScript. JScript je bio baziran na javnoj dokumentaciji Netscape-a bio je skoro identičan JavaScript jeziku.

ECMA JavaScript verzija postao Netscape-ova implementacija ovog standarda, a JScript Microsoft-ova. I danas obe verzije standarda su identične u preko 95% slučajeva.

JavaScript je:

- *u osnovi objektno baziran,*
- *platformski neutralan,*
- *višekorisnički jezik.*

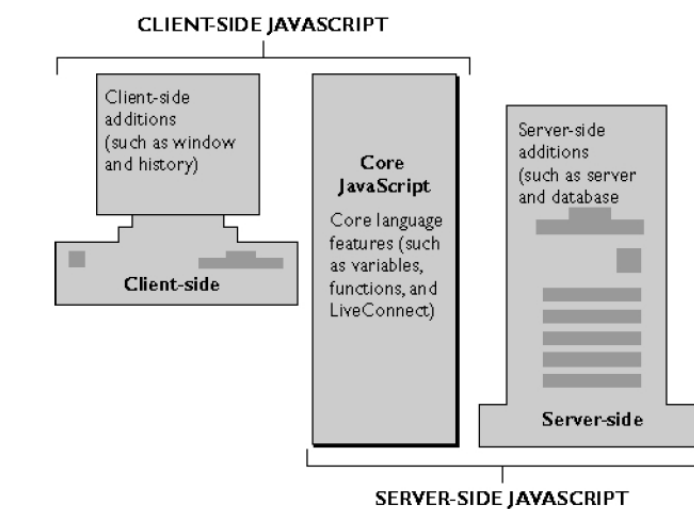
JavaScript programeru omogućava mnogo veću funkcionalnost na klijentskoj strani. Svi koncepti objektno orijentisanih jezika nisu realizovani u ovom jeziku, jer je veoma limitiran rad sa nasleđivanjem, važenjem i funkcionalnošću samih objekata. Sa druge strane postoje hijerarhija ugrađenih objekata i oni se mogu koristiti, sa već definisanim metodama i osobinama (property).

Ovakvim pristupom dobijeno je na jednostavnosti samog jezika, a pomoću ugrađenih objekata nije izgubljena potrebna funkcionalnost.

# JAVASCRIPT KAO KLIJENT TEHNOLOGIJA

*Web programeri pišu client - side skripte u jezicima poput JavaScript-a (Client - side JavaScript).*

**Client - side scripting** tehnologije se obično odnose na grupu računarskih programa na vebu koji se izvršavaju na strani klijenta, tj. od strane veb čitača na klijentskim računarima, umesto na strani servera, od strane aplikacija veb servera (Slika 1). Ovaj vid računarskog programiranja je bitan deo DHTML (*Dynamic HTML*) koncepta, zato što omogućava da veb strane budu skriptovane. Na taj način, veb strane postaju dinamičke i menjaju svoj sadržaj u zavisnosti od akcija korisnika, uslova u okruženju (kao npr. doba dana) i ostalih varijabli.



Slika 1.1 Opis klijent-server primene JavaScript jezika

Web programeri pišu *client - side* skripte u jezicima poput **JavaScript**-a (*Client - side JavaScript*). Skripte klijentske strane su uglavnom „prikačene u okviru HTML (*HyperText Markup Language*) dokumenta, ali takođe mogu biti ubačene u zaseban fajl, koji je referenciran u HTML dokumentu. Odmah po zahtevu za određenom stranom koja sadrži klijentski skript, neophodni fajlovi se šalju na klijentski računar od strane veb servera. Nakon toga, klijentski čitač izvršava skriptu i prikazuje dokument (veb stranicu), sa svim vidljivim izlazima iz klijentske skripte.

Skripte klijentske strane takođe mogu sadržati instrukcije koje veb čitač treba da sledi ako korisnik interaguje sa stranicom na određeni način, npr. ako klikne na odgovarajuće dugme na dokumentu. Takve instrukcije se mogu izvršavati bez komunikacije sa veb serverom.

Ako korisnici pogledaju izvorni kod veb strane koja sadrži klijentsku skriptu, mogu videti i kod te skripte. Mnogi veb programeri uče kako da pišu klijentske skripte istražujući izvorne kodove dokumenata drugih autora. S druge strane, kod serverskih skripti, korisnici ne mogu videti izvorni kod, jer se on izvršava na veb serveru i odmah se nakon toga prebacuje u HTML. Korisnici nisu čak ni svesni da je izvršena serverska skripta kada istražuju neki HTML dokument.

# UPOTREBA SKRIPTI KLIJENTSKE STRANE

*Skriptni jezici koji su podržani od strane širokog varijeteta veb čitača nisu uvek implementirani na isti način na svim čitačima i operativnim sistemima.*

Skripte klijentske strane imaju veći pristup informacijama i funkcijama koje se nalaze na klijentskim računarima, dok skripte serverske strane imaju veći pristup istim na serveru. Skripte klijentske strane ne zahtevaju dodatno instaliran softver na serveru i tako postaju popularne autorima veb stranica koji imaju malo administrativnih ovlašćenja na serveru. Međutim, skripte klijentske strane zahtevaju da klijentski brauzer „razume” skriptni jezik u kome su pisane. Baš zbog toga je krajnje nepraktično pisanje skripti u jezicima koji nisu podržani od strane većine veb čitača. Skriptovi klijentske strane se upotrebljavaju:

- *Da bi se dobili podaci koji su na ekranu klijenta ili u browseru klijenta*
- *Za online računarske igre*
- *Za personalizaciju prikaza (bez reloadovanje Web strane)*
- *Za predprocesiranje formi*

Skriptovi serverske strane se upotrebljavaju:

- *Za personalizaciju brauzera*
- *Za procesiranje (obradu) formi*
- *Za izgradnju i prikazivanje veb strana sa informacijama iz baza podataka.*

Postoji i treća tehnologija skriptnih jezika koja koristi i serverske i klijentske tehnologije. To je *udaljeno skriptovanje (remote scripting)* koje koristi najbolje od oba, brzinu klijentskih skripti sa fleksibilnošću serverskog koda. Međutim, ni udaljeno skriptovanje nije bez problema zbog načina na koji radi.

Udaljeno skriptovanje, funkcioniše tako što dozvoljava skriptama klijentske strane pristup funkcijama koje se nalaze na strani servera, a koje server obrađuje.

## KARAKTERISTIKE JAVASCRIPT JEZIKA

*JavaScript je platformski neutralan jezik, kao i HTML, što znači da bi njegov kôd (ako je pisan po standardu) trebalo da se izvršava u okviru čitača klijenta.*

### Karakteristike JavaScript jezika

*skripta* – lista naredbi koja se može izvršiti bez intervencije korisnika

*JavaScript* - najpoznatiji programski jezik za pisanje skripti (skriptni jezik); skripte se najčešće ugrađuju direktno u HTML dokument - JavaScript uvodi interaktivnost u HTML dokumente; besplatan je i podržan u najčešće korišćenim Internet čitačima.

## Platformski nezavisan jezik

Platformski neutralan jezik, kao i HTML, što znači da bi njegov kod (ako je pisan po standardu) trebalo da se izvršava u okviru čitača klijenta, bez obzira koja je hardverska mašina ili softversko okruženje u pitanju.

Veličina programa pisanih u ovom jeziku dovoljno je mala da može da se izvršava i na mašinama sa lošijim performansama.

## JavaScript i HTML

Još jedna od prednosti JavaScript jezika je njegova integrisanost sa HTML-om. U okviru jedne stranice je moguće je na proizvoljan način kombinovati JavaScript i HTML k&ocirc;d. Takođe iz JavaScript-a moguće je generisati sam HTML kod, u zavisnosti od određene akcije korisnika.

### Šta sve možemo s JavaScriptom?

- JavaScript je skriptni jezik koji HTML dizajnerima (koji nisu programeri) daje mogućnost pisanja jednostavnih programa i njihovog ugrađivanja direktno u HTML dokument
- JavaScript omogućuje dodavanje dinamičkog teksta HTML dokumentu (npr. "<h1>" + name + "</h1>" - u HTML dokumentu ispisuje vrednost varijable name kao zaglavlje tipa <h1>)
- JavaScript može reagovati na događaje - može se podesiti tako da je izvršavanje skripte povezano s nekim događajem (npr. učitavanjem stranice - onLoad, klikom miša - onClick)

## ŠTA SVE MOŽEMO S JAVASCRIPTOM?

*JavaScript je skriptni jezik koji HTML dizajnerima (koji nisu programeri) daje mogućnost pisanja jednostavnih programa i njihovog ugrađivanja direktno u HTML dokument.*

JavaScript daje mogućnost pisanja jednostavnih programa i njihovog ugrađivanja direktno u HTML dokument:

- JavaScript može čitati i menjati sadržaj HTML elementa
- JavaScript može prepoznavati web čitač krajnjeg korisnika - tako omogućuje da taj krajnji korisnik vidi dokument kreiran upravo za web čitač koji on koristi.

### Kako se uključuje programski kod?

Programski kod ovog jezika se može uključiti u okviru HTML stranice na dva načina:

Prvi je direktnim pisanjem koda u okviru stranice.

```
<SCRIPT LANGUAGE="JavaScript"> ...neki JavaScript kod... </SCRIPT>
```

Nije neophodno da se navodi atribut language= "JavaScript", jer on ima podrazumevanu vrednost JavaScript. Skripta pisana jezikom JavaScript se u HTML dokument upisuje između tagova <script> i </script> ovako:

```
<html>
  <body>
    <script type="text/javascript">
      ...
    </script>
  </body>
</html>
```

### **<script type="text/javascript">**

- označava početak JavaScript-a
- atribut type s vrednošću "text/javascript" definiše korišćeni skriptni jezik

Primer:

```
<html>
<body>
<script type="text/javascript"> document.write("Hello World!");
</script>
</body>
</html>
```

### **Prikaz HTML teksta**

HTML tekst se prikazuje pomoću JavaScript koda na stranici korišćenjem metoda `document.write("neki tekst koji se prikazuje");` Argument ovog metoda je string koji može biti proizvoljan HTML kod. Na primer:

```
<SCRIPT LANGUAGE="JavaScript">

document.write("<B>Prvi red</B><BR><I>Drugi red</I>")

</SCRIPT>
```

## **NAREDBA DOCUMENT.WRITE()**

*Web nivo se sastoji od komponenti koje obrađuju interakciju između klijenata i nivoa poslovne logike.*

`document.write()`

- JavaScript naredba za ispisivanje definisanog sadržaja
- skripta iz ovog primera u pripadajućem HTML dokumentu ispisuje string, npr. "Hello World!", i to tačno na mestu na kojem je skripta uključena u HTML dokument
- [http://www.w3schools.com/js/js\\_howto.asp](http://www.w3schools.com/js/js_howto.asp)

Čitači koji ne podržavaju JavaScript tretiraju ga kao obični tekst – k&ocirc; JavaScript-a prikazuju kao običan tekstualni sadržaj HTML dokumenta. Da bismo to izbegli dovoljno je JavaScript napisati kao HTML komentar:



```
<html>
<body>
<script type="text/javascript">
<!--
document.write("Hello World!");
//-->
</script>
</body>
</html>
```

Kose crte “//” su simbol za JavaScript komentar – one ovde sprečavaju izvršavanje tag-a “-->”

### Komentari u programiranju JavaScript-ama

Za komentar jedne linije koda se koristi oznaka //, na primer:

```
// komentar u jednoj liniji
```

Za komentarisanje više redova koriste se oznake

```
/* za početak bloka pod komentarom i
```

oznake \*/ za kraj bloka pod komentarom. Primer:

```
/* ovo je komentar
```

```
u više linija */
```

Više primera se može naći na: [http://www.w3schools.com/js/tryit.asp?filename=tryjs\\_text](http://www.w3schools.com/js/tryit.asp?filename=tryjs_text)

## UKLJUČIVANJE JAVASCRIPT-A U HTML DOKUMENTU

*Drugi način uključivanja JavaScript-i je poziv js dokumenta. U okviru taga se definiše spoljašnji dokument u okviru atributa src.*

### Gde uključiti JavaScript u HTML dokumentu?

JavaScript uključen u telo HTML dokumenta izvršava se automatski prilikom učitavanja tog HTML dokumenta u web čitač ako želimo da se JavaScript izvršava nakon učitavanja HTML dokumenta u web čitaču ili da je izvršavanje JavaScripta povezano s nekim događajem (npr. klikom miša, klikom na neko dugme i sl.) koristimo funkcije – JavaScript s definicijom funkcije na standardan način smeštamo u zaglavlje HTML dokumenta, a funkcije pozivamo u telu dokumenta; na taj način razdvajamo definicije JavaScript funkcija od sadržaja HTML dokumenta.

**Spoljašnji JavaScript kod** – omogućuje korišćenje/pozivanje iste skripte u više HTML dokumenata

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

Više primera može se naći na veb stranici: [http://www.w3schools.com/js/js\\_where.asp](http://www.w3schools.com/js/js_where.asp)

Na primer:

```
document.write("Ovo je JavaScript eksterni fajl!");
```

Odvajanje linija koda

Podrazumevani separator je novi red.

Nije greška ako se koristi simbol ";".

Jedini izuzetak, kada se obavezno mora koristiti tačka-zarez je ako se navodi više naredbi u istom redu.

Tada se svaka pojedinačna naredba mora odvojiti sa tačkom-zarez.

## ▼ Poglavlje 2

# Elementi JavaScript jezika

## JAVASCRIPT – VARIJABLE

*Nije neophodno deklarirati promenljivu pre prve dodele vrednosti (automatski će se izvršiti deklarisanje), ali je i predeklarisanje je dozvoljeno.*

Nazivi promenljivih - Imena promenljivih mogu da sadrže brojeve i slova engleske abecede, ali prvi znak mora da bude slovo engleske abecede ili simbol “\_” , pored toga:

- Ne mogu se koristiti prazna mesta u okviru imena.
- Ne mogu se koristiti rezervisane reči kao imena promenljivih.

**Promenljiva ili varijabla** – opisno rečeno je memorijska lokacija, “spremnik” za određenu informaciju/podatak ili vrednost.

Koristi se za deklarisanje promenljive (deklaracija je kreiranje promenljive, a definicija znači i inicijalizaciju - postavljanje početne vrednosti), čija sintaksa je:

*var imePromenljive;*

Opciono moguće je izvršiti i njenu inicijalizaciju.

```
var imePromenljive = vrednost;

var ime promenljive1 = vrednost1, ime promenljive2 = vrednost2,
...;

var evro; //deklaracija promenljiva

var dinar = 95; //definicija int promenljive
```

Nije neophodno deklarirati promenljivu pre prve dodele vrednosti (automatski će se izvršiti deklarisanje). Predeklarisanje je dozvoljeno. Više primera je na veb stranicama:

[http://www.w3schools.com/js/tryit.asp?filename=tryjs\\_variable](http://www.w3schools.com/js/tryit.asp?filename=tryjs_variable)

Deklaracija varijabli u JavaScriptu se vrši na sledeći način:

*var x;*

Sledeće četiri sekvence imaju isti efekat:

```
var x; x=8;
```

```
x=8; var x;
```

```
var x=8;
```

```
x=8;
```

## TIPOVI PODATAKA

*Tip podataka definiše i vrste operacija koje se mogu izvršiti sa tom promenljivom.*

Drugi način deklarisanja istih varijabli:

```
var x=5; var ime="Iva";
```

na ovaj su način deklarisane varijable nazvane x i ime; pri tome varijabli x pridružena je numerička vrednost 5, a varijabli ime vrednost "Iva" koja je tipa string.

Više primera ima na: [http://www.w3schools.com/js/js\\_variables.asp](http://www.w3schools.com/js/js_variables.asp)

### Case sensitive

JavaScript je *case sensitive* jezik, što znači da se velika i mala slova razlikuju, pa je promenljiva *Aaa* različita promenljiva od promenljive *AAA*.

Takođe se ključne reči (for, if, else, class, int,...) ne mogu koristiti u imenu promenljivih.

### Tipovi podataka

Informacija koja se sadrži u promenljivoj određuje tip varijable.

Postoje: celobrojni brojevi, racionalni brojevi, stringovi (niz karaktera), logički tip (true/false).

Tip podataka definiše i vrste operacija koje se mogu izvršiti sa tom promenljivom.

### Celobrojni brojevi

Mogu se koristiti sa brojnom osnovom 10, sa osnovom 8 i osnovom 16.

Uobičajena je predstava pomoću osnove 10. Ovakvi brojevi imaju cifre od 0 - 9, s tim da početna cifra ne sme biti 0.

Brojevi prikazani u oktalnom brojnom sistemu sa osnovom 8 moraju počinjati sa cifrom 0, a ostale cifre su od 0 - 7.

Brojevi prikazani u heksadecimalnom brojnom sistemu sa osnovom 16 moraju počinjati sa 0x ili 0X, a ostale cifre su od 0 - 15, s tim da se cifre 10 - 15 prikazuju slovima A - F.

### Racionalni brojevi

Mogu se prikazati na dva načina:

- pomoću decimalne tačke, na primer 3.14

- pomoću eksponencijalne prezentacije, na primer 314E-2 ili 314e-2

## OSTALI TIPOVI PODATAKA

*Velika količina obrade zahteva se u ranim fazama odvija u obliku (stil softverske arhitekture) proširive cevne obrade.*

### String

String predstavlja proizvoljan niz karaktera između navodnika ("neki tekst") ili između apostrofa ('neki tekst'). U stringovima se mogu koristiti i specijalni karakteri.

### Specijalni karakteri

\b = jedno mesto levo (backspace)

\f = jedan red nadole (form feed)

\n = početak novog reda (new line character)

\r = return (carriage return)

\t = tabulator (tab)

### Konverzija u string - primer

```
<script>
x=2+4;
document.write(x); document.write("<br/>");
x="2"+"4";
document.write(x); document.write("<br/>");
x=2+"4";
document.write(x); document.write("<br/>");
x="2"+4;
document.write(x); document.write("<br/>");
</script>
```

Rešenje nakon **izvršenja tog koda je:**

```
6
24
24
24
```

### Zaključak:

Integer se uvek konvertuje u string pri spajanju sa stringom.

### Logički tip

Logički tip podataka obuhvata dve vrednosti true (tačno) i false (netačno).

Prilikom rada ako je potrebno može se izvršiti konverzija logičke vrednosti `true` u broj `1` i vrednosti `false` u broj `0`.

### Konverzije podataka

JavaScript je jezik koji automatski izvršava promenu jednog tipa u drugi, jer se dozvoljava da promenljiva ima različite tipove podataka u različito vreme izvršavanja programa.

## JAVASCRIPT OPERATORI

*Operatori su specijalni karakteri, koji definišu operaciju koja treba da se izvrši nad operandima, koji mogu biti promenljive, izrazi ili konstante.*

Primer promene jednog tipa u drugi u Javascript jeziku:

```
a = 5; //a je sada celobrojni podatak
```

```
b = 8; //b je sada celobrojni podatak
```

```
b = "broj " + a; //b je sada string podatak, zato što se na string "broj" nadovezuje ceo broj, pa se dobija string!
```

### Null vrednost

Vrednost `null` je:

- tip podataka/vrednost koja se može dodeliti promenljivoj
- promenljiva koja nema vrednost
- dodeljena promenljivoj kada želimo da definišemo da promenljiva ne sadrži nikakav podatak

### Operatori

Operatori su specijalni karakteri, koji definišu operaciju koja treba da se izvrši nad operandima, koji mogu biti promenljive, izrazi ili konstante.

#### Aritmetički operatori

Koriste se za matematičke operacije. Ukoliko je jedan od operandi tipa `String` za sve operatore, osim za sabiranje, pokušaće se da se izvede konverzija `String` u broj i da se tako izvrši definisana operacija. Ako se ne uspe kao rezultat se dobija specijalna vrednost *NaN* (*Not A Number*).

Izuzetak kod sabiranja: podatak koji nije tipa `String` konvertuje se u `String` i izvršava se sabiranje dva `String`. Primer je:

```
a=24; b = "broj " + a; //dobija se da je b: broj 24
```

Pregled aritmetičkih operatora je prikazan na Slici 1.

Operator	Opis	Operator	Opis
+	sabiranje	+=	sabiranje dodela
-	oduzimanje	-=	oduzimanje dodela
*	množenje	*=	množenje dodela
/	deljenje	/=	deljenje dodela
%	moduo	%=	moduo dodela
++	inkrement (x=x+1)	--	dekrement (x=x-1)

Slika 2.1 Aritmetički operatori - pregled

## OPERATORI NA NIVOU BITA

*Operatori na nivou bita obavljaju operacije nad celobrojnim brojevima, i to dužine 32 bita.*

Operatori iz ove grupe obavljaju operacije nad celobrojnim brojevima, i to dužine 32 bita. Ukoliko neki od operandi nije celobrojni broj dužine 32 bita, pokušaće se izvršiti konverzija u traženi tip, pa tek onda primeniti operacija.

Primeri su:

13 & 8 daje 8 (1101 & 1000 = 1000)

13 | 8 daje 13 (1101 | 1000 = 1101)

13 ^ 8 daje 5 (1101 ^ 1000 = 0101)

Ostali operatori na nivou bita su dati na Slici 2.

Operator	Opis
Logičko I (and)	a & b Rezultat je 1, samo ako su oba bita 1.
Logičko ILI (or)	a   b Rezultat je 0, samo ako su oba bita 0.
Logičko ekskluzivno ILI (xor)	a ^ b Rezultat je 1, samo ako je jedan bit 1, a drugi 0.
Logičko NE (not)	~ a Komplementira bit 0->1, 1->0.
Pomeranje ulevo	a << b Pomera binarni sadržaj operanda a za b mesta ulevo. Prazna mesta popunjava nulama.
Pomeranje udesno sa znakom	a >> b Pomera binarni sadržaj operanda a za b mesta udesno. Prazna mesta popunjava vrednošću najstarijeg bita.
Pomeranje udesno sa nulama	a >>> b Pomera binarni sadržaj operanda a za b mesta udesno. Prazna mesta popunjava sa vrednošću 0.

Slika 2.2 Tabela operatori na nivou bita

**Logički operatori** Imaju vrednosti:

- true
- false

Ovi operatori imaju veliku primenu u okviru kontrolama toka. Primer upotrebe navedenih operatora je:

```
a = true;
b = false;
c = a || b;
d = a && b;
f = (!a && b) || (a && !b);
g = !a;
document.write( " a = " + a + "<BR>" );
document.write ( " b = " + b + "<BR> " );
document.write ( " c = " + c + "<BR> " );
document.write ( " d = " + d + "<BR> " );
document.write ( " f = " + f + "<BR> " );
document.write ( " g = " + g );
```

Ostali logički operatori su dati na Slici 3.

Operator		Opis
I (&&)	izraz1 && izraz2	Rezultat je TRUE, jedino ako su oba izraza TRUE, u ostalim slučajevima FALSE.
ILI (  )	izraz1    izraz2	Rezultat je TRUE, ako je bar jedan izraz TRUE, ako su oba FALSE, rezultat je FALSE.
NE (!)	! izraz	Rezultat daje komplement: ako je izraz TRUE rezultat je FALSE i obrnuto.

Slika 2.3 Logički operatori - pregled

## OPERATORI POREĐENJA

*Operatori poređenja obavljaju poređenje dve vrednosti i kao rezultat vraćaju vrednost logičkog tipa true ili false.*

### Operatori poređenja

Obavljaju poređenje dve vrednosti i kao rezultat vraćaju vrednost logičkog tipa true ili false. Svaki dozvoljeni tip podataka, celobrojan, racionalni, karakter, String i logički tip može se upoređivati koristeći operatore == i !=. Samo numerički tipovi koriste ostale operatore.

Tabela operatora za poređenje je prikazana na Slici 4.

Operator	Upotreba	Opis
Jednakost	x == y	Rezultat je TRUE, ako su operandi x i y jednaki
Nejednakost	x != y	Rezultat je TRUE, ako su operandi x i y različiti
Veće	x > y	Rezultat je TRUE, ako je x veće od y
Veće ili jednako	x >= y	Rezultat je TRUE, ako je x veće ili jednako 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
Jednakost (bez konverzije tipova)	x === y	Rezultat je TRUE, ako su x i y jednaki, ali bez konverzije tipova (moraju biti istog tipa!)
Različito (bez konverzije tipova)	x !== y	Rezultat je TRUE, ako su x i y različiti, ali bez konverzije tipova



Slika 2.4 Tabela operatora za poređenje

**Razlika između == i ===**

Operatori == i != obavljaју potrebnu konverziju podataka pre poređenja, ukoliko su operandi različitog tipa.

Znači za ove operatore vrednosti 5 (integer) i "5" (string) su iste, pa će posle njihovog poređenja rezultat sa operatorom == biti TRUE, a sa operatorom != FALSE.

S druge strane operatori === i !== ne obavljaју potrebnu konverziju podataka pre poređenja, ukoliko su operandi različitog tipa. Znači za ove operatore vrednosti 5 (ceo broj) i "5" (string) su različite, pa će posle njihovog poređenja rezultat sa operatorom === biti FALSE, a sa operatorom !== TRUE.

Primer koji to ilustruje je:

```
a = 4;
b = 1;
c = a < b;
d = a == b;

document.write( " c = " + c + "<BR>" );

document.write ( " d = " + d );

Rezultat izvršavanja prethodnog primera je

c = false
d = false
```

## KONSTRUKCIJE NAREDBI ZA KONTROLU TOKA IZVRŠAVANJA PROGRAMA

*Konstrukcije naredbi za kontrolu toka izvršavanja programa u JavaScript jeziku su slične kao i u ostalim programskim jezicima.*

**Naredba if - else**

Ova konstrukcija omogućava izvršenje određenog bloka instrukcija ako je uslov konstrukcije ispunjen. Opšti oblik konstrukcije je:

```
if (boolean_izraz) blok1;
[else blok2;]
```

svaki od blokova, bilo u if ili u else delu može biti nova if-else konstrukcija. Primer upotrebe ove konstrukcije je:

```
if (x == 8) {
  y=x;
} else {
  z=x;
  y=y*x
}
```

### Naredba if - then - else

Forma ovog operatora je:

expression ? statement1 : statement2

gde je izraz expression bilo koji izraz čiji rezultat je vrednost logičkog tipa (na primer: a>b).

Ako je rezultat izraza true, onda se izvršava statement1, u suprotnom statement2. Primer:

(x%2==0) ? document.write("paran broj") : document.write("neparan broj");

### Izgled složene if - else konstrukcije:

```
if (mesec == 1)
  ime_meseca = "Januar"
else if (mesec == 2)
  ime_meseca = "Februar"
else if (mesec == 3)
  ime_meseca = "Mart"
else if (mesec == 4)
  ime_meseca = "Maj"
else
  ....
else if (mesec == 12)
  ime_meseca = "Decembar"
```

Naredba switch se koristi na sledeći način:

```
switch(mesec) {
  case 1: ime_meseca = "Januar"; break;
  case 3: ime_meseca = " Mart"; break;
  case 5: ime_meseca = "Maj"; break;
  case 7: ime_meseca = "Jul"; break;
  case 8: ime_meseca = "Avgust"; break;
  case 10: ime_meseca = "Oktobar"; break;
  case 12: ime_meseca = "Decembar"; break;
  case 4: ime_meseca = " April "; break;
  case 6: ime_meseca = "Jun"; break;
  case 9: ime_meseca = "Septembar"; break;
  case 11: ime_meseca = "Novembar"; break;
  case 2: ime_meseca = " Februar ";
  default: ime_meseca = " Nije naveden mesec";
```

```
}
```

## IZVRŠAVANJE WHILE PETLJE U JAVASCRIPT JEZIKU

*U JavaScript jeziku petlja se koristi za deo programa koji se izvršava nijednom, jednom ili više puta.*

Ukoliko se vrednost izraza mesec, u prethodnom prikeru, ne nalazi medju vrednostima case 1,..., N, tada se izvršava blok naredbi default;

### Naredba while petlja

while petlja funkcioniše na taj način što se blok instrukcija unutar nje ponovljeno izvršava sve dok je uslov za ostanak u petlji, koji se nalazi na ulasku u petlju, ispunjen. Opšti oblik petlje izgleda ovako:

```
while(uslov_ostanka) {  
    telo_petlje;  
}  
Jednostavan primer:  
i=1  
while(i<=10){  
    document.writeln(i);  
    i=i+1;  
}
```

### Izvršavanje while petlje

Nakon izvršavanja ovog primera dobiće se prikazani brojevi od 1 do 10. Treba napomenuti da će se u slučaju da uslov petlje nije ispunjen kada se prvi put ispituje uslov petlje, telo petlje neće izvršiti nijednom.

### Izvršavanje while petlje

Nakon izvršavanja ovog primera dobiće se prikazani brojevi od 1 do 10. Treba napomenuti da će se u slučaju da uslov petlje nije ispunjen kada se prvi put ispituje uslov petlje, telo petlje neće izvršiti nijednom. Dakle, ovo je petlja koja se izvršava nijednom, jednom ili više puta.

### Izvršavanje do - while petlja

Za razliku od prethodne petlje koja je imala uslov na svom početku, do-while petlja ima uslov na kraju. Prema tome, telo petlje će se sigurno izvršiti bar jednom kao u sledećem primeru.

```
do {  
    telo_petlje  
    [iteracija]  
} while (uslov);  
i=1  
do {
```

```
document.writeln(i);  
i++; //i=i+1  
} while(i<=10)
```

## IZVRŠAVANJE FOR PETLJE I BREAK NAREDBA

*Naredba return se koristi za povratak iz funkcije na mesto poziva.*

### Izvršavanje for petlje

Opšti oblik for petlje izgleda ovako:

```
for( inicijalizacija; uslov; iteracija){  
  telo_petlje;  
}  
for(i=0; i<10; i++){  
  document.writeln(i);  
}
```

Promenljiva i je privremena promenljiva, a blok u kome je definisana je blok u kome se nalazi for petlja.

### break naredba

BREAK se koristi za skok na kraj bloka koji je označen labelom uz break ili na kraj bloka u kome se break nalazi, ako break stoji bez labele. Labele, pomoću kojih se označavaju blokovi, se formiraju kao i svi ostali identifikatori s tim što iza njih mora stajati dvotačka (:). Na primer, sledeći k&ocirc;d:

```
a: {  
  b: {  
    c: {  
      document.writeln("pre break-a"); //ovo se izvorsava!  
      break b;  
      document.writeln("ovo nece biti prikazano"); //ovo se ne izvorsava!  
    }  
  } // ovde izlazi iz bloka kada uradi break b!  
  document.writeln("posle break-a"); //ovo se izvorsava!  
}
```

### Naredba return

Naredba return se koristi za povratak iz funkcije na mesto poziva. Ukoliko funkcija vraća neku vrednost tada return mora slediti izraz čiji je tip kompatibilan sa povratnim tipom funkcije. U suprotnom return izjava može stajati sama, kao u sledećem primeru:

```
function kvadratBroja( x ){  
  return x * x;  
}
```

```
x = kvadratBroja(5);  
  
/* poziv funkcije */  
  
document.write("Kvadrat od 5 je " + x);
```

Kao rezultat poziva funkcije dobija se:

Kvadrat od 5 je 25

## NAREDBA CONTINUE

*Prelaz na sledeću iteraciju petlje, a da se deo koda pre njenog kraja ne izvrši postiže se upotrebom naredbe continue.*

### Naredba continue

Prelaz na sledeću iteraciju petlje, a da se deo koda pre njenog kraja ne izvrši. Za takve situacije se koristi continue.

```
for( i=0; i<10; i++) {  
    document.write(i+ " ");  
  
    if (i%2 == 0) continue; /*kada je broj paran  
    preskace sve naredbe  
    do kraja petlje */  
  
    document.writeln("  
    ");  
}
```

Zahvaljujući continue naredbi nakon izvršavanja ovog primera dobija se:

```
0 1  
  
2 3  
  
4 5  
  
6 7  
  
8
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 3

# HTML i JavaScript - Vežba 3

## REZIME VAŽNIH HTML ELEMENATA.

*Pre prelaska na izučavanje okvira neophodno je napraviti pregled najvažnijih HTML elemenata.*

Pre prelaska na izučavanje okvira, čije se stranice baziraju na HTML standardima, neophodno je napraviti kraći pregled najvažnijih HTML elemenata, neophodnih za dalji rad, kroz teme:

1. Liste i rad sa listama;
2. Rad sa linkovima i tabelama;
3. HTML5 - okviri i forme

## ▼ 3.1 Primer 1 - rad sa listama

### NUMERISANE LISTE

*Dodavanjem atribut taga mogu se praviti različiti izgledi numerisanih lista.*

Numerisane liste su liste kod kojih se ispred stavki liste prikazuju redni brojevi ili slova. Za definisanje liste ove vrste koristi se tag <OL> Stavke </OL>, dok se za definisanje stavki koristi tag <LI>.

Opšti oblik definisanja numerisane liste je:

<OL>

<LI> prva stavka

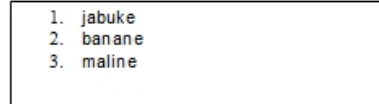
....

<LI> poslednja stavka

</OL>

Primer koda

```
<OL>
<LI> jabuke
<LI> banane
<LI> maline
</OL>
```



Slika 3.1.1 Izgled HTML dokumenta sa numerisanom listom

Brojevi ili slova koji se prikazuju ispred stavki numerisane liste mogu se definisati pomoću atributa `type`.

Atribut `type` može imati sledeće vrednosti:

**A**, što odgovara velikim slovima

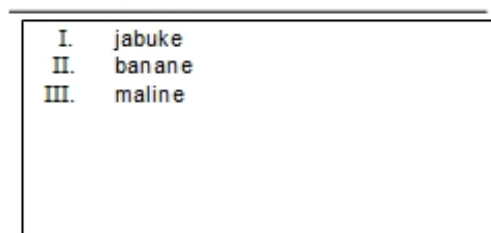
**a**, što odgovara malim slovima

**I**, što odgovara velikim rimskim brojevima

**i**, što odgovara malim rimskim brojevima

Primer koda:

```
<OL type="I">
<LI> jabuke
<LI> banane
<LI> maline
</OL>
```



Slika 3.1.2 Izgled HTML dokumenta sa atribut numerisanom listom

## NEUREĐENE LISTE

*Svaka stavka ovakve liste se sastoji iz dva dela: termina koji hoćemo da definisemo i njegove definicije.*

Svaka stavka ovakve liste se sastoji iz dva dela:

termina koji hoćemo da definisemo i njegove definicije.

- Termini zapocinju `<dt>` tagom, a njihove definicije `<dd>` tagom.
- Termini se poravnavaju uz levu marginu, a njihove definicije se pojavljuju u novom redu i uvučene su za određen broj mesta.

Opšta struktura je:

```
<DL>
```

```
<DT> odrednica 1 <DD> opis odrednice 1
```

```
<DT> odrednica 2 <DD> opis odrednice 2
```

```
.....
```

```
</DL>
```

Primer za ovu vrstu listi je:

Prolece: traje od 21. marta do 21. juna. Leto: traje od 21. juna do 21. septembra. Jesen: traje od 21. septembra do 21. decembra. Zima: traje od 21. decembra do 21. marta.

HTML kod za prethodni primer je:

```
<dl>

<dt> Prolece:

<dd> traje od 21. marta do 21. juna.

<dt> Leto:

<dd> traje od 21. juna do 21. septembra.

<dt> Jesen:

<dd> traje od 21. septembra do 21. decembra.

<dt> Zima:

<dd> traje od 21. decembra do 21. marta.

</dl>
```

## LISTA UNUTAR LISTE

*HTML jezik dozvoljava liste unutar listi upotrebom taga <LI> i &am*

Lista unutar liste

Unutar neke liste može se definisati nova lista. Na primer:

```
<UL>

<LI> A few New England states:

<UL>

<LI> Vermont

<LI> New Hampshire

<LI> Maine
```



```
</UL>

<LI> Two Midwestern states:

<UL>

<LI> Michigan

<LI> Indiana

</UL>

</UL>
```

Rezultat prethodnog primera:

- A few New England states:

- o *Vermont*

- o *New Hampshire*

- o *Maine*

- *Two Midwestern states:*

- o *Michigan*

- o *Indiana*

## ▼ 3.2 Primer 2 - LINKOVI I TABELE

### LINKOVI ILI HIPERVEZE

*Linkovi ili hiperveze su veze koje se uspostavljaju između HTML stranica.*

Linkovi ili hiperveze omogućavaju jednostavan prelazak sa jednog mesta na drugo u okviru iste stranice, ili prelazak na sasvim novu stranicu. Mesto odakle se prelazi naziva se polazna pozicija, a mesto na koje se prelazi, krajnja pozicija.

Postoje sledeće vrste linkova:

- krajnja pozicija se nalazi na istoj stranici kao i polazna pozicija
- krajnja pozicija se nalazi na drugoj stranici
- druga stranica je u okviru iste aplikacije i nalazi se na istom serveru kao i stranica sa polaznom pozicijom
- druga stranica ne pripada istoj aplikaciji i nalazi se na drugom serveru

## USPOSTAVLJANJE LINKA

*Koristi se poseban tag za postavljanje linka na HTML stranici*

Tag za definisanje linka je: `<A atribut> ... </A>`

*Slučaj1:* krajnja pozicija je na istoj stranici kao i polazna

### Uspostavljanje linka:

definisati ime krajnje pozicije -

`<A NAME="ime_krajnje_pozicije"> tekst na krajnjoj poziciji </A>`

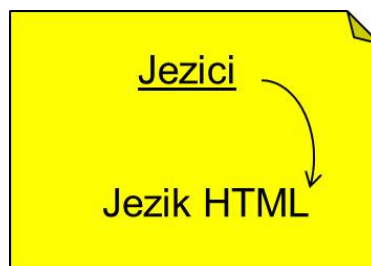
na polaznoj poziciji definisati referencu na krajnju poziciju -

`<A HREF="#ime_krajnje_pozicije"> tekst na polaznoj poziciji </A>`

### Primer:

`<A href="#htmlJezik"> Jezici </A>`

`<A name="htmlJezik"> Jezik HTML </A>`



Slika 3.2.1 Link u okviru iste strane HTML dokumenta

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## IZRADA TABELA U HTML DOKUMENTU

*Tabela je i u HTML-u dvodimenziona matrica čiji se elementi nazivaju ćelije (engl. cell).*

Ćelija može sadržavati raznovrsne informacije: brojeve, tekst, boje, liste, hiper-veze, slike, itd. Tabela se sastavlja tako što se opisuju redom njene vrste (redovi, engl. row) i sadržaj svake ćelije u redu.

Tabela se opisuje uz pomoć složenog taga `TABLE` koji može sadržavati više atributa:

- `BORDER` koji opisuje širinu spoljašnjeg okvira tabele;
- `CELLSPACING` koji opisuje širinu linije koja razdvaja dve ćelije;
- `CELLPADDING` koji opisuje prostor oko sadržaja ćelije;

- **WIDTH** koji opisuje ukupnu širinu tabele.

Nadnaslov tabele se može zadati tagom **CAPTION** koja se ispisi iznad tabele i može imati atribut **ALIGN**:

- za vertikalno poravnavanje: **TOP, MIDDLE, BOTTOM**
- za horizontalno poravnavanje: **LEFT, CENTER, RIGHT**

Svaki red u tabeli se opisuje između tagova **<TR>** i **</TR>** (engl. *table row*). ITakođe, tag TR može imati attribute:

- za horizontalno poravnavanje, atribut **ALIGN** sa vrednostima: **LEFT, CENTER, RIGHT**
- za vertikalno poravnavanje, atribut **VALIGN** sa vrednostima: **TOP, MIDDLE, BOTTOM**

Pojedinačna ćelija se opisuje između tagova **<TD>** i **</TD>**. Tag TD, pored atributa **ALIGN** i **VALIGN**, može imati i attribute:

- za horizontalno spajanje ćelija: **ROWSPAN** (spaja ćelije iste vrste) i
- za vertikalno spajanje ćelija: **COLSPAN** (spaja ćelije iste kolone) .

Tag **<TH>** ima ista svojstva kao tag **<TD>** s tom razlikom što obezbeđuje da sadržaj ćelije bude automatski centriran i boldovan. Tabela ne mora da sadrži **<TH>** tag, ali mora da sadrži bar jedan **<TD>** tag, u koji se smešta sadržaj tabele.

## IZGLED HTML KODA ZA IZRADU TABELE

### Tagovi **TABLE** i **CAPTION** - primena

Opšta struktura jedne tabele je sledeća:

```
TABLE>
<!-- pocetak definicije tabele -->

<CAPTION> sadrzaj naslova tabele </CAPTION>
<!-- definicija naslova-->

<TR>
<!-- start definicije headera -->
<TH> sadržaj prve ćelija headera </TH>
<TH> sadržaj poslednje ćelije headera </TH>
</TR>
<!-- kraj definicije headera -->

<TR>
<!-- start prvog reda -->
<TD> sadržaj prve ćelije prvog reda </TD>
<TD> sadržaj poslednje ćelije prvog reda </TD>
</TR>
<!-- kraj prvog reda -->
```

```
<TR>
<!-- start poslednjeg reda -->
<TD> sadržaj prve ćelije poslednjeg reda </TD>
<TD> sadržaj poslednje ćelije poslednjeg reda </TD>
</TR>
<!-- kraj poslednjeg reda -->

</TABLE>
<!-- kraj definicije tabele-->
```

Treba naglasiti da ako se tekstu van tabele upotrebom `<font>` taga dodeli neki font različit od default fonta, u Internet Exploreru će se ovaj tag odnositi i na tekst u ćelijama tabele. U Netscape Navigatoru će tekst u tabeli biti prikazan u default fontu. Da bi i Netscape Navigator tekst u tabeli prikazao u željenom fontu treba u svaku ćeliju posebno da se ubaci `<font>` tag.

Ako se u kodu tabele ne navede određeni atribut tabele nemaju graničnu liniju, `border`. Da bi se dodelila tabeli granična linija odgovarajuće debljine, u `<table>` tag se stavlja atribut `border`, a vrednost debljine linije se zadaje u pikselima. Sledeći primer pokazuje upotrebu atributa `border`:

Ime:	Prezime:	Zvanje:
Bosko	Nikolic	Predavac

Slika 3.2.2 Izgled tabele u HTML dokumentu

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PROMENA DIMENZIJE TABELE U HTML DOKUMENTU

*Treba napomenuti da tabela uopšte ne mora da bude uniformno formatirana, tj. da se svakoj ćeliji može dodeliti neko drugo svojstvo pomoću atributa koji se definišu unutar &*

Koćim se dobija prethodni primer tabele je sledeći:

```
<table border="2">
<tr>
<th><font face="Verdana,, size="2">Ime:</font></th>
<th><font face="Verdana" size="2">Prezime:</font></th>
<th><font face="Verdana" size="2">Zvanje:</font></th>
</tr>
<tr>
<td><font face="Verdana" size="2">Bosko</font></td>
<td><font face="Verdana" size="2">Nikolic</font></td>
<td><font face="Verdana" size="2">Predavac</font></td>
</tr>
</table>
```

Osnovna razlika između teksta koji se nalazi između `<th>` i `<td>` tagova je u tome što je tekst u prvom slučaju boldovan i centriran unutar ćelije, a u drugom slučaju tekst nije podebljan i poravnat je uz levu ivicu ćelije. Ivice tabele se priljubljuju uz sadržaj ćelija maksimalno koliko je moguće.

Širina kolona je određena prvom ćelijom u svakoj koloni. Naravno, ovim se ne iscrpljuje mogućnost formatiranja tabele. U tu svrhu se koriste razni atributi.

Da bi tabela imala odgovarajuće dimenzije koriste se atributi *height* i *width*. Vrednost ovih atributa se kao i kod `<img>` taga može zadati ili u pikselima ili u procentima veličine prozora HTML browsera. Preporučuje se drugi način definicije, jer onda izgled stranice ne zavisi od rezolucije ekrana. Sledeći primer predstavlja tabelu sa širinom od 80% i visinom od 30%:

Ime :	Prezime :	Zvanje :
Bosko	Nikolic	Predavac

Slika 3.2.3 Izgled skalirane (umanjene) visine i širine tabele u HTML-u

Kod kojim se dobija prethodni primer tabele sa Slike 2 je sledeći:

```
<table border="2" height="30%" width="80%">
<tr>
<th><font face="Verdana" size="2">Ime :</font></th>
<th><font face="Verdana" size="2">Prezime :</font></th>
<th><font face="Verdana" size="2">Zvanje :</font></th>
</tr>
<tr>
<td><font face="Verdana" size="2">Bosko</font></td>
<td><font face="Verdana" size="2">Nikolic</font></td>
<td><font face="Verdana" size="2">Predavac</font></td>
</tr>
</table>
```

## FORMATIRANJE ĆELIJA TABELE U HTML DOKUMENTU

*Pomoću **cellspacing** atributa se može odrediti rastojanje između pojedinih ćelija tabele.*

Da bi kolone bile jednake širine trebalo bi da u odgovarajuće `<th>` ili `<td>` tagove ubaciti

atribute `width` sa željenom širinom. U slučaju tabele iz prethodnih primera to znači da treba u svaki `<th>` tag definisati atribut `width` sa vrednošću 33%. Tako, na primer, se može realizovati tabela koja će se protezati preko cele širine stranice, a svaka kolona će zauzimati tačno trećinu širine tabele (Slika 3):

Ime :	Prezime :	Zvanje :
Bosko	Nikolic	Predavac

Slika 3.2.4 Izgled definisane visine i širine tabele u HTML dokumentu

HTML kod za prethodni primer je:

```
HTML kôd za prethodni primer je:
<table border="2" width="100%">
<tr>
<th width="33%"><font face="Verdana" size="2">Ime :</font></th>
<th width="33%"><font face="Verdana" size="2">Prezime :</font></th>
<th width="33%"><font face="Verdana" size="2">Zvanje :</font></th>
</tr>
<tr>
<td><font face="Verdana" size="2">Bosko</font></td>
<td><font face="Verdana" size="2">Nikolic</font></td>
<td><font face="Verdana" size="2">Predavac</font></td>
</tr>
</table>
```

Dva atributa koja imaju veliku primenu su i *cellpadding* i *cellspacing*. Pomoću cellpadding atributa definiše se rastojanje između sadržaja ćelije i njene granične linije. Vrednost ovog atributa se zadaje u pikselima, ako se ne navede nijedna vrednost podrazumeva se da 1.

Pomoću cellspacing atributa se može odrediti rastojanje između pojedinih ćelija tabele, tj. debljina linije između ćelija. Vrednost ovog atributa se zadaje, takođe, u pikselima ako se ne navede nijedna vrednost podrazumeva se da 1. Ako se u prethodnoj tabeli definiše vrednost cellpadding atributa od 30 piksela, a vrednost cellspacing atributa od 10 piksela dobija se sledeća tabela:

Ime :	Prezime :	Zvanje :
Bosko	Nikolic	Predavac

Slika 3.2.5 Primena cellpadding i cellspacing atributa tabele u HTML dokumentu

## UPOTREBA ATRIBUTA COLSPAN I ROWSPA

*HTML dopušta mogućnost da se pojedine ćelije tabele protežu duž više redova ili kolona tabele.*

HTML kod ove tabele sa Slike 17 glasi:

```
<table border="2" cellpadding="30" cellspacing="10">
<tr>
<th width="33%"><font face="Verdana" size="2">Ime :</font></th>
<th width="33%"><font face="Verdana" size="2">Prezime :</font></th>
<th width="33%"><font face="Verdana" size="2">Zvanje :</font></th>
</tr>
<tr>
```

```
<td><font face="Verdana" size="2">Bosko</font></td>
<td><font face="Verdana" size="2">Nikolic</font></td>
<td><font face="Verdana" size="2">Predavac</font></td>
</tr>
</table>
```

HTML dopušta mogućnost da se pojedine ćelije tabele protežu duž više redova ili kolona tabele.

Dani u nedelji :					
	ponedeljak	utorak	sreda	cetvrtak	petak
c a s o v i	1. srpski	istorija	fizicko	hemija	srpski
	2. matematika	srpski	fizicko	hemija	srpski
	3. fizicko	matematika	istorija	biologija	engleski
	4. fizicko	matematika	fizika	informatika	engleski
	5. fizika	razredni	geografija	informatika	matematika

Slika 3.2.6 Primena colspan i rowspan atributa tabele u HTML dokumentu

Ovakav efekat se može postići pomoću atributa *colspan* i *rowspan*, koji se ubacuju u `<td>` ili `<th>` tag one ćelije koja se želi posebno da formatira. Vrednost ovih atributa se zadaje brojem kolona ili redova tabele duž kojih treba da se prostire data ćelija. U sledećem primeru je prikazana upotreba ovih atributa

HTML kod tabele iz Slike 5 je:

```
<table border="2" width="100%">
<tr>
<th rowspan="2" colspan="2"></th>
<th colspan="5"> Dani u nedelji :</th>
</tr>
<tr>
<th width="20%">ponedeljak</th>
<th width="20%">utorak</th>
<th width="20%">sreda</th>
<th width="20%">cetvrtak</th>
<th width="20%">petak</th>
</tr>
<tr>
<th rowspan="5">c<p>a<p>s<p>o<p>v<p>i</th>
<th>1.</th>
<td>srpski</td>
<td>istorija</td>
<td>fizicko</td>
<td>hemija</td>
<td>srpski</td>
</tr>
<tr>
<th>2.</th>
<td>matematika</td>
<td>srpski</td>
```

```
<td>fizicko</td>  
<td>hemija</td>
```

## ▼ 3.3 Primer 3 - OKVIRI I NOVI HTML5 STANDARD

### NAMENA OKVIRA U HTML DOKUMENTU

*Pregledač, ukoliko je sposoban za to, interpretira frejmove kao podelu tekućeg prozora na više nezavisnih podprozora od koji svaki sadrži adresirani dokumenat.*

Browser, ukoliko je sposoban za to, interpretira frejmove (okvire) kao podelu tekućeg prozora na više nezavisnih potprozora od koji svaki sadrži adresirani dokumenat. Osnovni tag je složeni tag `<FRAMESET>`. Ovaj tag zamenjuje tag `BODY` u `HTML`-dokumentu.

Tekst dokumenta koji se rastavlja na frejmove sadrži isključivo informacije namenjene pregledaču koji poziva adresirana dokumenta i postavlja ih u odgovarajuće frejmove.

Tag `<FRAMESET>` ima attribute:

- `COLS` za vertikalnu podelu prozora i
- `ROWS` za horizontalnu podelu prozora navigatora.

Adresa dokumenta se navodi u okviru taga `FRAME` čiji su atributi

- `SRC`, preko koje se zadaje adresa dokumenta koji će biti prikazan u zoni tog taga i
- `MARGINWIDTH` i `MARGINHEIGHT`

Tag `NOFRAME` sadrži poruku za browser koji nije u stanju da interpretira frejmove. Opšta struktura `HTML` stranice sa frejmovima je:

```
<HTML> <HEAD> </HEAD>  
  
<FRAMESET> ... </FRAMESET> </HTML>
```

Primer `HTML` koda stranice koja koristi frejmove i njen izgled je dat na sledećoj slici:





Slika 3.3.1 Primer HTML koda stranice koja koristi frejmove

Tagovi koji su do sada razmatrani omogućavaju samo da se oformi hipertekstualni dokument koji će se razgledati browserom. Ali pregledač može i da prenese podatke ka web-serveru da bi se ti podaci tamo obradili. Ova mogućnost se ostvaruje preko koncepta formi (engl. *form*).

## FORME I ELEMENTI FORME

*Forma dopušta da se uspostavi komunikacija između korisnika i servera: korisnik popunjava formu i šalje je ka serveru.*

Polazeći od podataka iz forme vrši se odgovarajuća obrada na serveru, a o rezultatima korisnik eventualno biva obavešten. Obaveštavanje se ostvaruje tako što server generiše dokument u HTML-u koristeći se informacijama iz formi i vraća ga korisniku na elektronski način. Osnovna razlika između razgledanja neke strane i formi se ogleda u tome što je za "običnu" hipertekstuelnu stranu dovoljno "kliknuti" na hiper-vezu sa adresom x da bi se prešlo sa tekućeg na dokument na adresi x dok se kod formi uspostavlja veza sa adresom x nekog programa na serveru. Preko formi mogu se realizovati različite aplikacije kao što su:

- upit nad bazama podataka ili na pretraživačima;
- identifikacija pristupa određenom servisu servera;
- elektronska trgovina (prijem narudžbine i regulisanje načina plaćanja);
- obaveštavanje o promenljivim podacima (npr. vreme polaska aviona određenog datuma, i sl.)

Forma se implementira preko taga `<FORM>` čiji je opšti oblik: `<FORM> ... </FORM>`

Tag `<FORM>` sadrži dva atributa:

- atribut ACTION koji sadrži adresu (URL) programa na serveru;
- atribut METHOD kojim je opisana metoda prenosa argumenata programa. Ovaj atribut može imati vrednosti GET ili POST. Češće se koristi metoda POST. Metoda GET ima ograničen broj parametara: mora važiti da je dužina URL + dužina parametara < 1KB

## FORME I AKCIJE

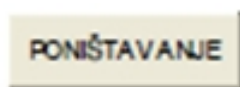
*Akcije sa navedene forme sa servera će se pozvati stranicu i realizovati navedenu metodu*

Primer HTML koda pomoću koga se definiše forma je:

```
<form action="Primer.html" method="post"> . . . </form>
```

Kada se pozove akcije sa navedene forme sa servera će se pozvati stranica Primer.html, i to pomoću post metoda. U okviru taga `<FORM>` mogu se navesti i drugi tagovi koji opisuju, na primer, izgled polja za unos podataka u formu, izgled polja za potvrdu, i sl. Tagovi `<FORM>` se ne mogu umetati jedan u drugi. Zajednički atributi različitih tagova u okviru forme su name, kojim se definiše ime promenljive preko koje će biti izvršena dodela vrednosti, i value, koja predstavlja ili izabranu vrednost u formi ili tekst koji će biti prikazan. Moguće je upotrebljavati sledeće tagove u okviru forme:

- `<INPUT>`: za unos podataka sa atributom `TYPE` koji opisuje prirodu podataka koji se unose. Vrednost ovog atributa može biti: `SUBMIT`, koja opisuje dugme čijim se pritiskom odašilje sadržaj popunjene forme ka serveru: `<INPUT TYPE="SUBMIT" NAME="SLANJE">` i `RESET`, koja postavlja sve vrednosti u formi na predefinisane vrednosti:



```
<INPUT TYPE="SUBMIT" NAME="PONISTAVANJE">
```

Slika 3.3.2 Primer RESET atributa u HTML kodu za frejmove

## NOVI TAGOVI I ATRIBUTI U IZRADI FORME

*Moguće je upotrebljavati `<SELECT>`, `<OPTION>`, i `<TEXTAREA>`*

Moguće je upotrebljavati sledeće tagove u okviru forme:

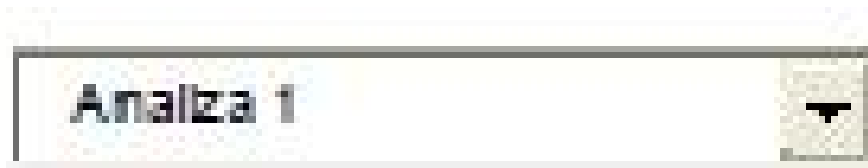
- `HIDDEN`, koje definiše nevidljivo ulazno polje čija će vrednost biti poslata sa drugim vrednostima iz forme kada se ovaj pošalje ka serveru kao, na primer, sledeći forma:

```
<FORM>  
<input type=hidden name=ident value="08100-0EM-38069">  
</FORM>
```

Tagovi `<SELECT>` i `<OPTION>` - polje `<SELECT>` opisuje izbor između mogućih vrednosti navedenih u okviru taga `<OPTION>`. Tag `<OPTION>` sa atributom `SELECTED` se uzima kao predefinisana vrednost. Tag `<SELECT>` se realizuje kao padajući meni sa vrednostima zadatim u okviru taga `<OPTION>`. Atributi uz tag `<SELECT>` su:

- `SIZE` kojim je opisan broj elementa u opcija koje će biti prikazane;

- **MULTIPLE** kojim se omogućava izbor višestruke opcija.



Slika 3.3.3 Upotreba , , i tagova u okviru forme

Tag `<TEXTAREA>` prikazuje tekst u odvojenom tekst prozoru sa scrollbarovima. Dimenzije prozora su određene atributima `<rows>` (broj redova teksta) i `<cols>` (broj kolona). Atribut `name` dodeljuje simboličko ime području u kome se nalazi tekst. Opšti oblik ovog taga je

`<TEXTAREA name=ime rows=n cols=m> ... neki tekst ... </TEXTAREA>`

## NOVI HTML5 STANDARD

*Novi HTML5 jezik nudi znatno bogatiju paletu tagova i njihovih atributa.*

HTML5 je novi Web standard. U okviru ovog novog standarda dolazi čitav skup novih tagova koji bi trebali da obezbede nove mogućnosti, uklone ili bar umanje potrebu za spoljnim dodacima poput Flash-a, da bolje upravljaju greškama, da odgovarajućim novim tagovima umanje potrebu za skriptingom, i olakšava uređenje strana pomoću CSS-a (pogotovu prilagođeno CSS3), takođe treba da obezbedi nezavisnost od strane platforme i uređaja.

Posebno interesantne opcije novog HTML5 standarda su: `canvas` element koji predstavlja platno za crtanje i crteže, ugrađen multimedijalni plejer koji olakšava puštanje video i audio materijala, mogućnost pamćenja podataka u lokalnim resursima, novi elementi koji su usmereni na sadržaj poput `<article>`, `<footer>`, `<header>`, `<nav>`, `<section>`. Novi standard uvodi i bolju podršku formularima pa novi tagovi olakšavaju unos podataka sa html strana. Novi tagovi koji su od velike koristi u formularima su:

- `<calendar>`,
- `<date>`,
- `<time>`,
- `<email>`,
- `<url>` i
- `<search>`.

### Layout

Osnovna prednost HTML5 u odnosu na predhodnu verziju je olakšano definisanje elemenata na strani pomoću novih tagova `<article>`, `<footer>`, `<header>`, `<nav>`, `<section>`. Zahvaljujući njima na različiti način se kreiraju web strane nego pre pojave HTML5 standarda.

Kada se kreira web strana u HTML5 prvo se napravi kostur strane koji sadrži samo osnovne elemente definisane standardizovanim tagovima.

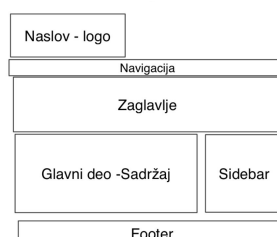
Prilikom kreiranja HTML5 strane najjednostavnije je početi od osnovnih tagova koji definišu osnovnu strukturu strane.

```
<!doctype html>
<html>
<head>
<title>Page title</title>
</head>
  <body>
<header>
<h1>Page title</h1>
</header>
<nav>
<!-- Navigation -->
</nav>
<section id="intro">
<!-- Introduction -->
</section>
<section>
<!-- Main content area -->
</section>
<aside>
<!-- Sidebar -->
</aside>
<footer>
<!-- Footer -->
</footer>
  </body>
</html>
```

## ČITLJIVOST HTML5 KODA

*Novine HTML koda su da umesto gomile div-ova u ovakvoj postavci HTML-a imamo jasno izdvojene celine koje samim nazivom taga definišu šta je gde.*

Jednostavni HTML5 kôd, koji je prethodno pokazan, će da kreira osnovnu strukturu kao što je pokazano na slici.



Slika 3.3.4 Jednostavni HTML5 kod osnovne strukture dokumenta

Ovakva postavka značajno uvećava čitljivost HTML koda, jer umesto gomile div-ova u ovakvoj postavci HTML-a imamo jasno izdvojene celine koje samim nazivom taga definišu šta je gde. Takođe će kasnije biti mnogo jednostavnije praviti CSS jer će moći da se poziva na ove tagove a ne na određene class i id attribute. Naravno class i id-ovi nam ostaju za odvajanje pod delova i delova koji su pod istim tagovima.

**Navigacija** - kompletna se nalazi pod `nav` tagom, na ovaj način je jasno odvojena navigacija od drugih elemenata HTML-a.

```
<nav> <ul> <li><a href="#">Blog</a></li>
<li><a href="#">0 nama</a></li>
<li><a href="#">Članci</a></li>
<li><a href="#">Kontakt</a></li>
<li class="subscribe"><a href="#">Pratite nas</a></li>
</ul> </nav>
```

**Zaglavlje** - je deo sajta se nalazi obično uvodna priča o sajtu i o onome što je glavna tema sajta. Ovde može da se stave i neke slike ili citati. Šta god bi bilo prikladno za upoznavanje korisnika sa sajtom.

```
<section id="intro"> <header>

<h2>Da li vam se dopada HTML5?</h2>

</header><p> Na ovom delu sajta se nalazi obično uvodna priča o sajtu i o onome što
je glavna tema sajta. Ovde može da se stave i neke slike ili citati. Šta god bi
bilo prikladno za poznavanje korisnika sa sajtom.</p> </section>
```

## GLAVNI SADRŽAJ HTML5 DOKUMENTA

*Najčešće je sadržaj spakovan u jedan ili više article tagova.*

Glavni sadržaj sajta u HTML-u sadrži u sebi sve elemente koje želimo da budu u glavnom delu sajta, to je ono što je centar pažnje posetioca sajta.

```
<section>

<article class="blogPost">

<header>

<h2>Ovde ide naslov bloga</h2>

<p>Postavljeno: <time datetime="2012-10-08T23:31:45+01:00">08
Oktobar 2012</time> postavio: <a href="#">Student</a> -
<a href="#comments">3 comments</a></p>
```

```
</header>

<p>U ovom delu se nalazi tekst bloga, to je u slučaju ovog sajta.

U nekom drugom slučaju bi ovom delu bile druge informacije koje

predstavljaju osnovni motiv sajta.</p>

</article>

</section>
```

## GLAVNI SADRŽAJ HTML5 KODA - NASTAVAK

*U okviru sekcije može da se nađu razni HTML elementi.*

Najčešće je sadržaj spakovan u jedan ili više [article](#) tagova. Ovaj tag kreira jednu instancu artikla, koji može u okviru sebe da ima složenu strukturu.

Ovo je zgodno i kasnije za kreiranje CSS-a. Svakom tagu možemo dodati id ili class da bi ga odvojili od drugih istih tagova ako mu je kasnije kroz CSS potrebna diferencijacija. [Header](#) tag služi da bi se odvojio sadržaj artikla (bloga u ovom slučaju), od zaglavlja tog artikla (bloga). U ovom haderu se nalazi naslov okružen H2 tagom, ali i spcifični podaci vezani za definisanje vremena. HTML5 ima secijalni tag koji olakšava definisanje vremenskog podatka to je tag [time](#). Atribut [datetime](#) treba da bude u formi godina-mesec-danTsati:minuti:sec+pomeraj u odnosu na GMT. Komentari na blog bi mogli da se smeste u zasebnu sekciju:

```
<section id="comments">

<header> <h3>Komentari</h3>

</header> <article> <header>

<a href="#">Vladimir Milićević</a> na dan <time datetime="2019-

10-09T22:05:05+01:00">09 Oktobar 2012</time> </header>

<p>Ovo je jako dobra lekcija i sve je lako da se nauči što je

osnovni cilj učenja na našem fakultetu. Iz tog razloga svima

preporučujem da ovo prate i vredno uče. </p> </article>
```

**<header>**

**<a href="#">Marko Rajević</a> na dan **<time**  
datetime="2012-10-09T22:06:06+01:00">09 Oktobar 2012**</time>****

**</header>**

**<p>**HTML5 je veoma jednostavan, mnogo jednostavniji od predhodnih

verzija koje su bile dosta kofuzne i čiji kod je bio jako težak za čitanje i praćenje. </p>

</article>

</section>

## NOVINE HTML5 JEZIKA U KREIRANJU FORMI

*HTML5 donosi čitav niz novih mogućnosti kada je u pitanju rad sa formama.*

PHP i mnogi drugi mehanizmi kreiranja web aplikacija koriste standardnu html formu za komunikaciju sa korisnicima. Da bi ove forme bile funkcionalne ranije se koristile razne dodatne biblioteke i JS, ali sa HTML5 za većinu njih nema potrebe jer su unapređenja koja su uneta u HTML5 takva da rešavaju većinu problema zbog koji su standardne html forme bile od slabe korisiti. Sada bi trebalo da se programeri fokusiraju pre svega na korišćenju standardnih HTML5 formi pa tek ako nešto ne može pomoću njih da se reši da traže alternativne načine. Posebo treba istaći par sitnica koje ranije nisu postojale a rešavaju veliki broj problema, kao na primer atribut required koji obezbeđuje da nije moguće poslati rezultate forme bez popunjenog polja koji ima ovaj atribut. Neka česta polja u formam zaslužuju posebnu validaciju to se pre svega odnosi na polja za unos emaila i web strane pa zato HTML5 ima posebne tagove za polja koja bi primala ovakve podatke. Ova polja vrše validaciju ovih složenih podataka po pravilima za krecanje ovih podataka, i time oslobađaju programera proveravanja validnosti ovih podataka prilikom obrade rezultata sa forme. Da bi neko polje postalo polje za unos emaila ili url-a neophodno je samo da u input tagu se doda type email ili url.

```
<form action="#" method="post">

<h3>Dajte komentar</h3>

<p> <label for="name">Ime</label>

<input name="name" id="name" type="text" required /> </p>

<p> <label for="email">E-mail</label>

<input name="email" id="email" type="email" required /> </p>

<p> <label for="website">Website</label>

<input name="website" id="website" type="url" /> </p> <p>

<label for="comment">Komentar</label>

<textarea name="comment" id="comment" required></textarea> </p>

<p><input type="submit" value="Postavi komentar" /></p> </form>
```

## NOVINE HTML5 JEZIKA U UNOSU OPCIJA, BROJEVA, DATUMA....

*Osim osnovnih oblika još dosta toga postoji u HTML5 što olakšava posao programerima prilikom pravljenja složenih formi.*

### Lista opcija

Osim ovih osnovnih oblika još dosta toga postoji u HTML5 što olakšava posao programerima prilikom pravljenja složenih formi. Na primer jedna od čestih situacija je kada je potrebno napraviti listu mogućih opcija, i da korisnik prilikom upisa dobije jednu od ponuđenih.

```
<input list="cars"/> <datalist id="cars"> <option value="BMW"/> <option value="Ford"/>  
<option value="Volvo"/> </datalist>
```

### Unos broja

Problem sa unosom broja je u tome što je ranije bilo potrebno da se broj unosi kao tekst pa da se kasnije kovertuje u broj. Umesto toga treba koristiti komponentu za unos celobrojnog broja.

```
<input type="number" step="1" min="-5" max="10" value="0" />
```

### Unos datuma

Unos datuma je jedan od kritičnijih koraka u unosu podatka. Korisnici lako pogreše očekivani format unosa datuma što za posledicu ima pogrešno uneti datum. Da bi se to izbeglo koristi se obično kalendar. Sam HTML5 ima posebnu komponentu za to input tag tipa date.

```
<input type="date" min="2012-08-14" max="2012-12-21" value="2012-08-24"/>
```

### Unos podatka iz opsega

Prilikom unosa podatka obično je neophodno postaviti minimalnu imaksimalnu vrednost unetog podatka. Kada se ovo radi onda je bolje da prikaz ne bude obično polje već slider sa mogućnošću postavljanja položaja u neki od vrednosti iz opsega. Evo kako se to radi u HTML5.

```
<input type="range" min="0" max="50" value="10" />
```

Prilikom unosa podatka obično je neophodno postaviti minimalnu imaksimalnu vrednost unetog podatka. Kada se ovo radi onda je bolje da prikaz ne bude obično polje već slider sa mogućnošću postavljanja položaja u neki od vrednosti iz opsega. Evo kako se to radi u HTML5.

```
<input type="range" min="0" max="50" value="10" />
```

## NOVINE U UNOSU SLIKE, BROJA TELEFONA, BOJE ....

*Stavljanje slika na stranu je uobičajeno, međutim moguće je razlikovati dve vrste slika, to su slike koje su stavljene na stranu kao deo dizajna strane i slike koje su tu kao deo sadržaja.*

**Unos telefona** - Vrlo čest podatak je unos telefona, međutim telefon može da bude zahtevan u raznim formatima. HTML5 omogućava da se to reši koristeći regular expressions.

```
<input type="tel" placeholder="(555) 555-5555" pattern="^\d{3}[-\s]\d{3}[-\s]\d{4}.*?$" />
```

**Unos boje** - može da bude problematičan jer korisnik mora da unosi brojeve umesto da odabere samo boju iz palete. Ovo je rešeno u HTML5 tako što se korisniku ponudi paleta iz koje može da odabere boju koju želi da odabere.



```
<input type="color" placeholder="#A00" />
```

**Polje za pretragu** - Česta komponenta na HTML stranama je polje za pretragu. Evo kako može ovo polje da se kreira pomoću HTML5 tagova.

```
<input type="search" results="10" placeholder="Search..." />
```

**Sidebar i footer** - Sidebar i footer je jednostavno kreirati, svodi se na isti princip kao i kod dosadašnjih elemenata. Dakle treba dodati u okviru aside taga odnosno footer taga odgovarajuće sekcije, article, slike, linkove, etc... i kreirati željenu strukturu ovih delova.

**Slike** - Stavljanje slika na stranu je uobičajeno, međutim moguće je razlikovati dve vrste slika, to su slike koje su stavljene na stranu kao deo dizajna strane i slike koje su tu kao deo sadržaja. Slike koje su deo sadržaja obično imaju i tekst ispod slike. HTML5 omogućava da se slike koje su deo sadržaja unose na drugačiji način od slika koje predstavljaju samo dizajn. Ovo olakšava i kasnije uređenje ovih elemenata pomoću CSS.

```
<figure>  <figcaption>Chart 1.1</figcaption> </figure>
```

## NOVINE U UNOSU MULTIMEDIJA U HTML5 DOKUMENT

*Video je do sad uglavnom bio predstavljan pomoću flash-a, ali mnogi uređaji pre svega telefoni ne podržavaju flash, pa je zato neophodno da se puštanje videa ubaci u sam HTML dokument.*

### Detalji

Jedna od mogućnosti koje često trebaju je da se na klik miša pokaže više informacija o nečemu. Ovo je moguće u HTML5 uraditi na sledeći način:

```
<details>

<summary>HTML 5</summary>

Ovo će se pokazati kada se klikne na opciju HTML5.

</details>
```

### Video

Video je do sad uglavnom bio predstavljan pomoću flash-a, ali mnogi uređaji pre svega telefoni ne podržavaju flash, pa je zato neophodno da se puštanje videa ubaci u sam HTML. HTML5 podržava dva formata zapisa mp4 ili ogg. Chrome browser koji je dostupan za sve platforme podržava oba formata dok neki drugi browseri podržavaju samo jedan od ova dva formata. Na primer Safari 3 i IE9 podržavaju samo mp4 dok Firefox 3.5 i Opera 10.5 podržavaju samo ogg.

Kada se postavlja video na strani trebalo bi obezbediti oba formata kako bi se pokrio što veći broj browsera. Evo primera kako treba postaviti video na strani.

```
<video width="320" height="240" controls>
<source src="movie.ogg" type="video/ogg" />
<source src="movie.mp4" type="video/mp4" />
Vaš browser ne podražava video tagove.
</video>
```

## NOVINE U UNOSU VIDEO U HTML5 DOKUMENT

*Video je do sad uglavnom bio predstavljjan pomoću flash-a, ali mnogi uređaji pre svega telefoni ne podržavaju flash, pa je zato neophodno da se puštanje videa ubaci u sam HTML.*

### Video

Video je do sad uglavnom bio predstavljjan pomoću flash-a, ali mnogi uređaji pre svega telefoni ne podržavaju flash, pa je zato neophodno da se puštanje videa ubaci u sam HTML. HTML5 podržava dva formata zapisa mp4 ili ogg. Chrome browser koji je dostupan za sve platforme podržava oba formata dok neki drugi browseri podržavaju samo jedan od ova dva formata. Na primer Safari 3 i IE9 podržavaju samo mp4 dok Firefox 3.5 i Opera 10.5 podržavaju samo ogg.

Kada se postavlja video na strani trebalo bi obezbediti oba formata kako bi se pokrio što veći broj browsera. Evo primera kako treba postaviti video na strani.

```
<video width="320" height="240" controls>
<source src="movie.ogg" type="video/ogg" />
<source src="movie.mp4" type="video/mp4" />
Vaš browser ne podražava video tagove.
</video>
```

## AUDIO TAGOVI U HTML5 DOKUMENTU

*Audio tag je sličan video tagu, opcije su iste kao i za video osim što ne postoje opcije za širinu i visinu.*

### Audio

Audio tag je sličan video tagu. Kada je reč o video tagu onda imamo tri moguća formata zvuka to su wav, mp3 i ogg. Kao i u slučaju video formata nemaju svi browser-i podršku za sve formate, ogg i mp3 su podržani od Chrome browser-a dok na primer ogg nije podržan od

strane Safari 3 a mp3 nije podržan od strane Opere 10.5 i Firefox-a 3.5. Kao i u slučaju video fajla moguće je postaviti više formata na stranu pa u zavisnosti od browsera odgovarajući će biti pušten. Evo primera koda:

`<audio controls>`

`<source src="song.ogg" type="audio/ogg" />`

`<source src="song.mp3" type="audio/mpeg" />`

Vaš browser ne podražava audio tagove.

`</audio>`

Opcija	Vrednost	Opis
autoplay	autoplay	Počinje puštanje videa odmah čim je dostupan
controls	controls	Kontrole za video poput play tastera i dokle je stigao sa puštanjem da budu vidljive.
height	pixels	Visina videa koji se pušta
loop	loop	Ako postoji ovaj parametar onda će video da se pušta is početka kada se završi.
preload	preload	Ovaj parametar obezbeđuje da se video učita odmah nakon što se strana učita a ne tek kada se pusti video. Biće ignorisan ako je postoji autoplay parametar.
src	url	Putanja do video fajla.
width	pixels	Definisanje širine puštenog videa u pikselima.

Slika 3.3.5 Tagovi za multimediju u HTML5 dokumentu

## SKRIPTOVI ZA PRAVLJENJE GRADIJENTA I UČITAVANJE SLIKA U HTML5 DOKUMENTU

*U slučaju HTML5 gradijent i učitavanje slika se vrši pomoću JS.*

### Pravljenje gradijenta :

```
<script type="text/javascript">
var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
var grd=cxt.createLinearGradient(0,0,175,50);
grd.addColorStop(0,"#FF0000");
grd.addColorStop(1,"#00FF00");
cxt.fillStyle=grd;
cxt.fillRect(0,0,175,50); </script>
```

### Učitavanje slike :

```
<script type="text/javascript">

var c=document.getElementById("myCanvas");

var cxt=c.getContext("2d");

var img=new Image()

img.src="img_flwr.png"

cxt.drawImage(img,0,0); </script>
```

### Svg format slike

Jedna od mogućnosti je kreiranje i baratanje sa [svg](#) formatom slika.

```
<svg>

<circle id="myCircle" class="important" cx="50%" cy="50%" r="100"

fill="#AA0"

onmousedown="alert('hello');"/>

</svg>
```

## STANDARDNI ATRIBUTI

*U predhodnoj verziji HTML-a bilo je samo par standardnih atributa poput id, class, accesskey, lang itd.. ali u novoj verziji HTML5 postoji čitav niz novih atributa koji obezbeđuju nove funkcije*

Standardni atributi su opcije koje mogu da se dodaju na bilo koji tag. Standardni atribut u HTML5 koji je vezan za drag'n'drop mogućnosti je atribut [draggable](#) koji može da ima vrednosti true, false i auto.

Atribut hidden postavlja taj element kao nevidljiv. Subject na primer povezuje element sa nekim elementom preko njegovog id-a dakle vrednost ovog atributa je id nekog drugog elementa. Nova mogućnost je i uključivanja spellcheck mogućnosti na sadržajem nekog elementa, za to se koristi standardni atribut [spellcheck](#) koji može da ima vrednost true ili false. Mogućnost davanja dodatnog kontekstnog menija se postiže standardnim atributom contextmenu koji ima vrednost menu\_id. Postoji čak i mogućnost dodavanja sopstvenih novih atributa samo treba da počnu sa data- i onda ide naziv koji autor odabere.

### Događaji

Događaji su akcije koje mogu da se izvedu nad nekim od elementima. HTML5 uvodi veliki broj novih akcije što obezbeđuje delako veće mogućnosti u kombinaciji sa JS. Na primer nad prozorom je bilo samo tri akcije, onload, onblur i on focus. Sada ih ima daleko više.

Formularima je ukinuta akcija *onreset* ali su dodato još par novih akcija kao na sledećoj slici:

Akcije	Vrednost	Opis
oncontextmenu	script	Pokreće skript kada se promeni kontekstni meni
onformchange	script	Pokreće skript kada se promeni sadržaj formulara.
onforminput	script	Pokreće skript kada korisnik počen unos u formular.
oninput	script	Pokreće skript kada element ima ulaz.
oninvalid	script	Pokreće skript kada je podatak neispravno unet.

Akcije	Vrednost	Opis
onafterprint	script	Pokreće skript nakon što se štampa završi
onbeforeprint	script	Pokreće skript pre puštanja na štampu
onbeforeunload	script	Pokreće skript pre nego što učitava stranu
onblur	script	Pokreće skript kada prozor nema više fokus (postoji od ranije)
onerror	script	Pokreće skript kada se pojavi greška
onfocus	script	Pokreće skript kada prozor dobije fokus (postoji od ranije)
onhaschange	script	Pokreće skript kada se dokument promeni
onload	script	Pokreće skript kada se strana učitava (postoji od ranije).
onmessage	script	Pokreće skript kada se aktivira pruka
onoffline	script	Pokreće skript kada dokument nije više online
online	script	Pokreće skript kada se nađe online
onpagehide	script	Pokreće skript kada se strana sakrije
onpageshow	script	Pokreće skript kada strana postane vidljiva
onpopstate	script	Pokreće skript kada se promeni istorija strane
onredo	script	Pokreće skript kada se pokrene naredba redo
onresize	script	Pokreće skript kada prozor promeni veličinu
onstorage	script	Pokreće skript kada se dokument učitava

Slika 3.3.6 Akcije u HTML5 dokumentu pomoću Javascript-a

## AKCIJE VEZANE ZA RAD SAM MIŠEM

*Akcije vezane za rad sam mišem su obogaćene sa dve akcije vezane za točkić miša i scroll *onmousewheel* i *onscroll*.*

Akcije vezane za rad sam mišem su obogaćene sa dve akcije vezane za točkić miša i scroll *onmousewheel* i *onscroll*. Takođe su dodate akcije za rad sa drag'n'drop mehanizmom *ondrag*, *ondragend*, *ondragenter*, *ondragleave*, *ondragover*, *ondragstart* i *ondrop*.

Akcije vezane za nove mogućnosti multimedijalnog sadržaja:

Akcija	Vrednost	Opis
onabort	script	Pokreće skript kada se odustane od događaja.
oncanplay	script	Pokreće skript kada multimedijalni sadržaj može da se pusti, ali u nekim slučajevima zaustavlja baferovanje sadržaja.
oncanplaythrough	script	Pokreće skript kada sadržaj može da se pusti ceo do kraja bez baferovanja.
ondurationchange	script	Pokreće skript ako se promeni dužina sadržaja.
onemptied	script	Pokreće skript kada sadržaj odjednom nestane usled greške u mreži, ili učitavanju ili iz bilo kog drugog razloga.
onended	script	Pokreće skript kada puštanje sadržaja dođe do kraja.
onerror	script	Pokreće skript kada dođe do greške u učitavanju.
onloadeddata	script	Pokreće skript kada se sadržaj učitava.
onloadedmetadata	script	Pokreće skript kada se podaci o sadržaju učitaju.
onloadstart	script	Pokreće skript kada browser počne sa učitavanjem sadržaja.
onpause	script	Pokreće skript kada je puštanje sadržaja pauzirano.
onplay	script	Pokreće skript kada se pritisne dugme play.
onplaying	script	Pokreće skript kada se počne sa puštanjem sadržaja.
onprogress	script	Pokreće skript kada se učitava sadržaj.
onratechange	script	Pokreće skript kada se promeni brzina puštanja.
onreadystatechange	script	Pokreće skript kada je promenjeno stanje „Ready-state“
onseeked	script	Pokreće skript kada prestane da se traži sadržaj, odnosno kada je opcija seeking false.
onseeking	script	Pokreće skript kada počne da traži sadržaj odnosno kada je opcija seeking true.
onstalled	script	Pokreće skript kada postoji greška u dovlačenju sadržaja (stalled)
onsuspend	script	Pokreće skript kada je dovlačenje sadržaja zaustavljeno pre nego što je celokupan sadržaj dovučen.
ontimeupdate	script	Pokreće skript kada se promeni pozicija puštanja u sadržaju.
onvolumechange	script	Pokreće skript kada se promeni jačina zvuka ili se postavi na mute.
onwaiting	script	Pokreće skript kada se zaustavi puštanje sadržaja ali se očekuje nastavak puštanja.

Slika 3.3.7 Akcije u HTML5 dokumentu vezane za rad sam mišem

## ▼ 3.4 Individualne vežbe 3

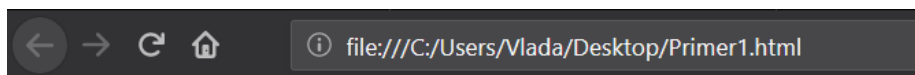
### ZADATAK 1

#### *Samostalno vežbanje dodavanja JS koda u HTML - tabele.*

Dat je sledeći primer, koji primenom JS omogućava prikaz tablice množenja u HTML dokumentu. Na početku se preko dijaloga zadaju dimenzije tablice, nakon čega se ona prikazuje u HTML stranici (slika 1).

```
<html>
<head>
  <title>Tablica množenja</title>
  <script type="text/javascript">
    var rows = prompt("Koliko vrsta ima tablica množenja?");
    var cols = prompt("Koliko kolona ima tablica množenja?");
    if(rows == "" || rows == null)
      rows = 10;
    if(cols == "" || cols == null)
      cols = 10;
    createTable(rows, cols);
    function createTable(rows, cols)
    {
      var j=1;
      var output = "<table border='1' width='500' cellpadding='5'>";
      for(i=1;i<=rows;i++)
```

```
{
  output = output + "<tr>";
  while(j<=cols)
  {
    output = output + "<td>" + i*j + "</td>";
    j = j+1;
  }
  output = output + "</tr>";
  j = 1;
}
output = output + "</table>";
document.write(output);
}
</script>
</head>
<body>
</body>
</html>
```



1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Slika 3.4.1 Tablica množenja u HTML dokumentu

### Zadatak:

1. Samostalno implementirajte pokazani primer;
2. Omogućite na sličan način prikazivanje korena i kvadrata prvih 10 brojeva u novoj tabeli.
3. dodajte link za odlazak na veb sajt našeg univerziteta.

## ZADATAK 2

*Samostalno vežbanje dodavanja JS koda u HTML - forme.*

Sledećim primerom je omogućen unos korisničkih podataka putem forme i validacija polja email (slika 2):

```
<html>
<head>
  <title>Form Validation</title>
  <script type="text/javascript">
    var divs = new Array();
    divs[0] = "errFirst";
    divs[1] = "errLast";
    divs[2] = "errEmail";
    divs[3] = "errUid";
    divs[4] = "errPassword";
    divs[5] = "errConfirm";
    function validate()
    {
      var inputs = new Array();
      inputs[0] = document.getElementById('first').value;
      inputs[1] = document.getElementById('last').value;
      inputs[2] = document.getElementById('email').value;
      inputs[3] = document.getElementById('uid').value;
      inputs[4] = document.getElementById('password').value;
      inputs[5] = document.getElementById('confirm').value;
      var errors = new Array();
      errors[0] = "<span style='color:red'>Please enter your first name!</span>";
      errors[1] = "<span style='color:red'>Please enter your last name!</span>";
      errors[2] = "<span style='color:red'>Please enter your email!</span>";
      errors[3] = "<span style='color:red'>Please enter your user id!</span>";
      errors[4] = "<span style='color:red'>Please enter your password!</span>";
      errors[5] = "<span style='color:red'>Please confirm your password!</span>";
      for (i in inputs)
      {
        var errMessage = errors[i];
        var div = divs[i];
        if (inputs[i] == "")
          document.getElementById(div).innerHTML = errMessage;
        else if (i==2)
        {
          var atpos=inputs[i].indexOf("@");
          var dotpos=inputs[i].lastIndexOf(".");
          if (atpos<1 || dotpos<atpos+2 || dotpos+2>=inputs[i].length)
            document.getElementById('errEmail').innerHTML = "<span style='color:red'>Enter a valid email address!</span>";
          else
            document.getElementById(div).innerHTML = "OK!";
        }
        else if (i==5)
        {
          var first = document.getElementById('password').value;
          var second = document.getElementById('confirm').value;
          if (second != first)
            document.getElementById('errConfirm').innerHTML = "<span style='color:red'>Your passwords don't match!</span>";
        }
      }
    }
  </script>
</head>
<body>
  <div>
    <input type="text" id="first" value="First Name" />
    <input type="text" id="last" value="Last Name" />
    <input type="text" id="email" value="Email" />
    <input type="text" id="uid" value="User ID" />
    <input type="password" id="password" value="Password" />
    <input type="password" id="confirm" value="Confirm Password" />
    <input type="button" value="Validate" />
  </div>
</body>
</html>
```



```

        else
            document.getElementById(div).innerHTML = "OK!";
        }
        else
            document.getElementById(div).innerHTML = "OK!";
    }
}

function finalValidate()
{
    var count = 0;
    for(i=0;i<6;i++)
    {
        var div = divs[i];
        if(document.getElementById(div).innerHTML == "OK!")
            count = count + 1;
    }
    if(count == 6)
        document.getElementById("errFinal").innerHTML = "All the data you
entered is correct!!!";
    }
</script>
</head>
<body>
    <table id="table1">
        <tr>
            <td>First Name:</td>
            <td><input type="text" id="first" onkeyup="validate();" /></td>
            <td><div id="errFirst"></div></td>
        </tr>
        <tr>
            <td>Last Name:</td>
            <td><input type="text" id="last" onkeyup="validate();" /></td>
            <td><div id="errLast"></div></td>
        </tr>
        <tr>
            <td>Email:</td>
            <td><input type="text" id="email" onkeyup="validate();" /></td>
            <td><div id="errEmail"></div></td>
        </tr>
        <tr>
            <td>User Id:</td>
            <td><input type="text" id="uid" onkeyup="validate();" /></td>
            <td><div id="errUid"></div></td>
        </tr>
        <tr>
            <td>Password:</td>
            <td><input type="password" id="password" onkeyup="validate();" /></td>
            <td><div id="errPassword"></div></td>
        </tr>
        <tr>
            <td>Confirm Password:</td>
            <td><input type="password" id="confirm" onkeyup="validate();" /></td>
            <td><div id="errConfirm"></div></td>
        </tr>
    </table>

```

```

</tr>
<tr>
  <td><input type="button" id="create" value="Create"
onclick="validate();finalValidate();" /></td>
  <td><div id="errFinal"></div></td>
</tr>
</table>
</body>
</html>

```

First Name:	<input type="text" value="Vladimir"/>	OK!
Last Name:	<input type="text" value="Milićević"/>	OK!
Email:	<input type="text" value="vladam@gmail.com"/>	OK!
User Id:	<input type="text" value="aaa"/>	OK!
Password:	<input type="password" value="••••"/>	OK!
Confirm Password:	<input type="password" value="••••"/>	OK!

All the data you entered is correct!!!

Slika 3.4.2 Primer JS forme

### Zadatak:

1. Samostalno implementirajte pokazani primer;
2. Kreirajte vlastitu formu za unos i proveru unetih podataka o automobilu sa poljima: marka, tip, snaga, kubikaža, boja;
3. snaga i kubikaža ne smeju da budu negativne vrednosti.

pomoćni link: <https://stackoverflow.com/questions/3571717/javascript-negative-number>

## ▼ Poglavlje 4

### Domaći zadatak 3

#### DOMAĆI ZADATAK BROJ 3

*Cilj domaćeg zadatka je da student provežba naučeno na vežbama*

Za domaći zadatak broj 4 potrebno je da student uzme svoj rad iz DZ2 i da postupi po sledećim zahtevima:

- osnov za izradu DZ su primeri urađeni na individualnim vežbama.
- kreirajte HTML dokument sa jednom tabelom ili formom i primetite stil po izboru;
- dodajte JavaScript kod koji manipuliše podacima iz tabele ili forme;
- omogućiti prikazivanje rezultata manipulacije podacima primenom JavaScript-a.

Domaći zadatak postaviti na [GitHub](#) repozitorijum i poslati na link na mejl predmetnom asistentu sa naslovom *IT255 - DZ03*.

## ▼ Zaključak

### REZIME LEKCIJE 3

*Lekcija je napravila kratak uvod u skript jezik JavaScript kao osnove za izučavanje Angular okivra.*

Lekcija je napravila kratak uvod u skript jezik *JavaScript* koji predstavlja osnovu za izučavanje buduće problematike vezane za razvoj veb aplikacija primenom radnog okvira *Angular*. Izlaganje lekcije je teklo analizom i diskusijom vezanom za sledeće teme:

- pregled kratkog istorijata i primene JS jezika;
- upoznavanje sa osnovnim elementima JS jezika;
- obnavljanje važnih HTML elemenata za budući rad: rad sa listama, linkovima, tabelama i formama;

Posebno, pokazne i individualne vežbe su rezervisane za demonstraciju integrisanja JS koda i HTML dokumenta.

### LITERATURA ZA LEKCIJU 03

*U izradi ove lekcije korišćena je navedena literatura.*

#### Obavezna literatura:

1. Learning Web Design, Fourth Edition, by Jennifer Niederst Robbins, Copyright &copy; 2012 Littlechair, Inc.
2. HTML5 and CSS3 Responsive Web Design Cookbook, Benjamin LaGrone, Copyright &copy; 2013 Packt Publishing.

#### Dopunska literatura:

1. Thomas Yager, Windows 2000 – Razvoj Web aplikacija, CET Computer Equipment and Trade, Beograd, 2001.
2. Mick P. Couper, Designing Effective Web Surveys, CAMBRIDGE UNIVERSITY PRESS, &copy; Mick P. Couper 2008.

#### Veb lokacije:

1. <http://taligarsiel.com/Projects/howbrowserswork1.htm>
2. <http://www.w3schools.com/>
3. <https://www.packtpub.com/>

