

# Uvod u Veb i Internet tehnologije

## JavaScript

Filip Marić  
Vesna Marinković

# Šta se sve može uraditi korišćenjem JavaScript-a?

- JavaScript može da izmeni sadržaj HTML-a (01\_js.html)
- JavaScript može da izmeni vrednost atributa HTML-a (02\_js.html)
- JavaScript može da izmeni stil HTML elementa (03\_js.html)
- JavaScript može da sakrije HTML element i ponovo ga prikaže (04\_js.html)

# Gde se piše JavaScript kôd?

- JavaScript kôd se zadaje kao sadržaj elementa `script`
- JavaScript funkcija se izvršava kada se pozove, a poziva se kada se desi neki događaj
- U okviru HTML dokumenta može se javiti proizvoljan broj skriptova
- Skript se može navesti:
  - u telu dokumenta – mora se navesti nakon elementa koji se menja (05\_js.html)
  - u zaglavlju dokumenta (06\_js.html)
  - u spoljašnjoj datoteci sa ekstenzijom `.js` (06\_js.html i skript.js)
  - kao vrednost nekog atributa koji služe da opišu reakcije na događaje, npr `onclick`, `onkeydown`

# Gde se mogu prikazati podaci korišćenjem JavaScript-a?

- JavaScript može prikazati podatke na različite načine:
  - u okviru HTML elementa, korišćenjem svojstva `innerHTML`, za pristup HTML elementu može se iskoristiti metod `document.getElementById(id)` (07\_js.html)
  - pisanjem na HTML izlaz, pozivom metoda `document.write()` (*napomena*: ako se ovaj metod pozove nakon što je ceo HTML dokument već učitao, obrišaće se sav postojeći HTML kôd) (08\_js.html i 09\_js.html)
  - pisanjem u prozor za obaveštenja pozivom metoda `window.alert()` (10\_js.html)
  - pisanjem u konzolu veb pregledača pozivom metoda `console.log()` (11\_js.html)

# Tipovi podataka i vrste operatora

- Tri primitivna tipa podataka:
  - brojevni tip (jedan tip za celobrojne i realne brojeve),
  - niske karaktera (predstavljene klasom String)
  - logički tip (predstavljen klasom Boolean: vrednosti true i false)
- Podržani operatori:
  - operatori konverzije: String, Boolean, parseInt, parseFloat
  - aritmetički operatori: +, -, \*, / (ne postoji celobrojno deljenje), %, ++, -- (prefiksni i postfiksni)
  - relacioni operatori: <, <=, >, >=, !=, == (jednaki po vrednosti), === (jednaki i po vrednosti i po tipu, nisu dozvoljene implicitne konverzije)
  - logički operatori: &&, ||, !
  - bitski operatori: &, |, ~, ^, <<, >>, >>> (upražnjena mesta se uvek popunjavaju nulama)
  - operatori dodele: =, +=, -=, ...
- Primeri: 12\_js.html

# Promenljive, naredbe, kontrolne strukture

- Deklaracija promenljive je oblika: `var ime_prom;`
- JavaScript je **dinamički tipiziran jezik** – jedna ista promenljiva može imati različit tip tokom izvršavanja programa
- Grananje je oblika:  
`if (uslov) then blok naredbi [else blok naredbi]`
- Podržan je i operator `?:`
- Višestruko grananje je oblika:  
`switch (vrednost) {case vr1: blok naredbi; break; ...}`
- Postoji podrška za `for` petlju, `while` petlju i `do while` petlju (sintaksa je C-ovska)
- Primeri: 13\_js.html

# Funkcije

- Definicija funkcije je oblika:  
`function ime_fje(lista parametara){ naredbe }`
- U promenljivoj je moguće sačuvati funkcijski izraz; na ovaj način dobijamo **anonimne funkcije** (ne navodi im se ime)
- Moguće je koristiti notaciju lambda izraza za zapis funkcije:  
`var f = (x=>3*x)`
- Funkcija može biti parametar druge funkcije:  
`function funkcijaFunkcije(f,y) return f(y)`
- Kao argument funkcije može se zadati anonimna funkcija:  
`console.log(funkcijaNiza(x=>3*x,[1,2,3,4,5]))`
- Primeri: 14\_js.html

# Funkcije – zatvorenje

- Funkcije mogu biti ugnježdene: tada je iz unutrašnje funkcije moguće pristupiti lokalnim promenljivim spoljašnje funkcije
- Funkcija nakon vraćanja vrednosti i dalje ima pristup lokalnim promenljivim funkcije koja ju je vratila – ova pojava naziva se [zatvorenje](#)
- Primeri: 21\_js.html



# Rad sa stringovima

- String se može zadati pod dvostrukim ili jednostrukim navodnicima  
`var ime='Marko', prezime="Petrovic";`
- `s.length` – vraća dužinu niske `s`
- `s.charAt(i)` – vraća karakter na  $i$ -toj poziciji u stringu `s` (indeksi idu od 0)
- `s1.concat(s2)` – nadovezuje nisku `s2` na nisku `s1`, može se koristiti i `s1 + s2`
- `s1.indexOf(s2)` ili `s1.indexOf(s2,k)` – prvo pojavljivanje niske `s2` u niski `s1` (počev od pozicije  $k$ )
- `s1.lastIndexOf(s2)` ili `s1.lastIndexOf(s2,k)` – poslednje pojavljivanje niske `s2` u niski `s1` (počev od pozicije  $k$ )
- `s.substr(k)` ili `s.substr(k,l)` – izdvaja podnisku niske `s` počev od pozicije  $k$  do kraja (odnosno do pozicije  $l$ )

# Rad sa stringovima

- `s.toLowerCase()` i `s.toUpperCase()` – niska `s` se prevodi u mala, odnosno velika slova
- `s.split(c)` ili `s.split(c,k)` – vrši se podela niske `s` u odnosu na separator `c` i vraćaju se sve (ili najviše `k`) dobijenih podniski
- `s.charCodeAt(i)` – vraća UNICODE kôd datog karaktera
- `String.fromCharCode(c)` – vraća karakter na osnovu datog UNICODE kôda `c`
- `s.substring(i,j)` i `s.slice(i,j)` – izdvaja deo niske `s` između pozicije `i` i pozicije `j` (ako `j` nije navedeno izdvaja se do kraja)  
mogu se koristiti i negativne vrednosti indeksa, tada se broji od 0 od kraja niske (ali se ne može koristiti 0)
- Primeri: 15\_js.html i 16\_js.html

# Rad sa stringovima – specijalni karakteri

- Za zadavanje rezervisanih karaktera koristi se prekidački simbol '\\':
  - \' – apostrof
  - \" – dvostruki navodnici
  - \\ – obrnuta kosa crta
- Linija kôda koja sadrži string može se prelomiti preko reda korišćenjem znaka '\\n' ili konkatencijom stringova (bolji način)
- Primeri: 16\_js.html

# Rad sa matematičkim funkcijama

- Postoji podrška za osam matematičkih konstanti:  
`E`, `PI`, `LN2`, `LN10`, `LOG2E`, `LOG10E`, `SQRT2`, `SQRT1_2`
- Postoji podrška za matematičke funkcije:
  - `abs(x)` – apsolutna vrednost
  - `sin(x)`, `cos(x)`, `tan(x)` – trigonometrijske funkcije
  - `asin(x)`, `acos(x)`, `atan(x)` – inverzne trigonometrijske funkcije
  - `round(x)`, `ceil(x)`, `floor(x)` – zaokruživanje na najbliži, prvi veći i prvi manji ceo broj
  - `exp(x)`, `log(x)`, `pow(x)` – eksponencijalna, logaritamska i stepena funkcija
  - `sqrt(x)` – kvadratni koren broja
  - `min(x1,...,xn)`, `max(x1,...,xn)` – najmanji i najveći od nekoliko brojeva
  - `random()` – slučajan broj iz intervala `[0,1]`
- Primeri: `17_js.html`

# Nizovi

- Nizovi se predstavljaju tipom `Array`
- Svojstvo `length` sadrži broj elemenata niza
- Metode za rad sa nizovima:
  - `a1.concat(a2,...,an)` – na niz `a1` nadovezuju se nizovi `a2, ..., an`
  - `a.join(c)` – od niza `a` kreira nisku koja se sastoji od elemenata niza `a` međusobno razdvojenih karakterom `c`
  - `a.push(x)` – umeće element `x` na kraj niza `a`
  - `a.pop()` – uklanja element sa kraja niza `a`
  - `a.unshift(x)` – umeće element `x` na početak niza `a`
  - `a.shift()` – uklanja element sa početka niza `a`
  - `a.reverse()` – obrće elemente niza
  - `a.splice(i,br,e1,...,en)` – uklanja `br` elemenata niza `a` počev od pozicije `i` a od indeksa `a` smešta elemente `e1, ..., en` (`n` ne mora biti jednako `i`)
  - `a.slice(poc,kraj)` – izdvaja deo niza između indeksa `poc` i `kraj` (ili do kraja niza ako nije dat indeks `kraj`)

# Nizovi

- Prilikom dodele `a = b` vrši se **plitko kopiranje** nizova – ove dve promenljive predstavljaju referencu na isti objekat
- Ako želimo da se iskopira samo vrednost možemo uraditi `a = b.slice()`;
- Metod **`s.map(f)`** – pravi se novi niz čije se vrednosti dobijaju kada se na članove niza `s` primeni funkcija `f`: može biti ugrađena funkcija (`Math.sqrt`) ili korisnički definisana
- Metod **`s.sort()`** – sortira elemente niza `s` rastuće, razmatrajući elemente niza kao stringove; može se proslediti funkcija poređenja **`s.sort(f)`**
- Primeri: `18_js.html`

# Objekti

- **Objekti** se sastoje od više vrednosti koje mogu biti različitog tipa
- Promenljivoj se može dodeliti više vrednosti na sledeći način:

```
var student = {
    ime: "Ana",
    prezime: "Petrović",
    brojIndeksa: "56/2015",
    prosek: 9.21
};
```

- Vrednosti **svojstva** objekta se zadaju u obliku: naziv:vrednost
- Svojstvima se pristupa na jedan od dva načina:

- nazivObjekta.nazivSvojstva
- nazivObjekta["nazivSvojstva"]

- **Metoda** se zadaje kao svojstvo koje sadrži definiciju funkcije

```
punoImeIPrezime: function(){
    return this.ime + " " + this.prezime;
}
```

- **this** unutar objekta označava sâm taj objekat

# Datum i vreme

- Vremenske odrednice se mogu predstaviti objektom tipa `Date`
- Datum se konstruiše pozivom `new Date()` – ako se ne prosledi parametar, pravi se objekat koji predstavlja trenutno vreme na klijentskoj mašini
- Vremenske odrednice se mogu predstaviti i brojem milisekundi koje su protekle od ponoći 1.1.1970.
- Postoji konstruktor sa 7 parametara: godina, mesec, dan, sat, minut, sekund, milisekund; ako se neki parametar izostavi podrazumeva se 0
- Prilikom konstruisanja datuma, može se zadati i niska
- Moguće je porediti dva datuma operatorima `<` i `>`
- Metode za izdvajanje komponenti: `getFullYear()`, `getMonth()`, `getDate()`, `getHours()`, `getMinutes()`, `getSeconds()`, `getTime()`,...
- Metode za postavljanje komponenti: `setFullYear()`, `setMonth()`, `setDate()`, `setHours()`, `setMinutes()`, `setSeconds()`, `setTime()`,...
- Primeri: `20_js.html`, `digitalni-sat.html`



# JSON

- **JSON** (*JavaScript Object Notation*) je format za čuvanje i transport podataka
- Pravila:
  - podaci se čuvaju u obliku parova “naziv”: “vrednost” – ne mogu se koristiti jednostruki navodnici
  - podaci se međusobno razdvajaju zapetom
  - za čuvanje objekata koriste se vitičaste zagrade
  - za čuvanje nizova koriste se uglaste zagrade
- Ako je u promenljivoj tekst smešten JavaScript string zapisan u JSON sintaksi, onda se on može konvertovati u JavaScript objekat korišćenjem funkcije **JSON.parse()**
- Primeri: 22\_js.html

# Formulari

- U HTML-u postoji podrška za pravljenje **formulara** u koje posetioci veb-sajtova mogu da unose podatke
- Za obrađivanje podataka iz formulara koriste se skript-jezici
- Osnovni elementi:
  - **form** – predstavlja formular
  - **input** – opisuje veći broj kontrola, atribut **name** služi za imenovanje kontrole, **value** sadrži njenu vrednost, atribut **type** može imati vrednosti:
    - **text** – polje za unos teksta
    - **password** – polje za unos lozinke
    - **radio** – radio-dugme (sva dugmad iz iste grupe imaju isto ime)
    - **checkbox** – polje za štrikliranje
    - **button** – obično dugme
    - **submit** – dugme čijim se aktiviranjem prikupljeni podaci šalju na server

# Formulari

- Osnovni elementi (nastavak):
  - `select` – predstavlja padajuću listu
    - atributom `size` postavlja se broj opcija koje se vide;
    - pojedinačne stavke u listi se zadaju elementom `option` koji ima atribute `name` i `value` (važne prilikom slanja podataka)
  - `textarea` – polje za unos teksta u više redova; atributi `name`, `rows`, `cols`
  - `label` – natpis čiji je sadržaj vidljiv na ekranu; atribut `for` može da sadrži identifikator kontrole sa kojom se uspostavlja veza

# Primer formulara (1. deo)

```
<form>
  <label for="ime">Ime i prezime: </label>
  <input type="text" id="ime" name="ime" />
  <br />

  <label for="ime">Lozinka: </label>
  <input type="password" id="lozinka" name="lozinka" />
  <br />

  <label for="adresa">Adresa: </label>
  <textarea id="adresa" name="adresa"></textarea>
  <br />

  <label for="vrsta">Vrsta pice: </label>
  <select id="vrsta">
    <option value="kapričoza">Kapričoza</option>
    <option value="margarita">Margarita</option>
    <option value="vegetarijana">Vegetarijana</option>
  </select>
  <br />
```

## Primer formulara (2. deo)

```
<input type="radio" name="velicina" id="velika" value="velika"
      checked />
<label for="velika">Velika</label> <br/>
<input type="radio" name="velicina" id="srednja" value="srednja" />
<label for="srednja">Srednja</label> <br/>
<input type="radio" name="velicina" id="mala" value="mala" />
<label for="mala">Mala</label> <br/>

<input type="checkbox" name="kecap" id="kecap" />
<label for="kecap">Kečap</label>
<input type="checkbox" name="origano" id="origano" />
<label for="origano">Origano</label>
<br />

<input type="button" value="Poništi"
      onclick="ponisti()" />

<input type="submit" value="Naruči"
      onclick="naruciPicu()" />
</form>
```

# Izgled formulara

Ime i prezime:

Lozinka:

Adresa:

Vrsta pice:

☒ Velika  
☐ Srednja  
☐ Mala

☒ Kečap ☐ Origano

- Primer: 23\_js.html

# Objektni model dokumenta

- **Objektni model dokumenta** (*Document Object Model, DOM*) je interfejs koji omogućava skriptovima da dinamički pristupe i izmene sadržaj, strukturu ili stil veb dokumenta
- Elementi dokumenta predstavljaju se **objektima** koji imaju svoja **svojstva** i **metode**; promenom vrednosti svojstava i pozivima metoda menja se veb dokument
- DOM je nezavisan od jezika iz kojeg se koristi i od platforme na kojoj se koristi

# Istorijat DOM-a

- Nastao je u vreme ratova pregledača
- 1996. Netscape u okviru njihovog pregledača ugrađuje podršku za jezik JavaScript; Microsoft u IE 3.0 ugrađuje podršku za jezik JScript
- Definiše “DOM Level 0” odnosno “Legacy DOM” koji omogućava pristup samo nekim elementima HTML dokumenta
- 1997. sa novijim verzijama pregledača javlja se bolja podrška za dinamički HTML i DOM se proširuje; svaka kompanija vrši nezavisno proširenje i ove verzije su poznate pod nazivom “Intermediate DOM”
- Krajem 1990-tih pod okriljem W3C započinje standardizacija klijentskih skript jezika i DOM-a; razvijen standard ECMAScript i standardna verzija DOM-a poznata kao DOM Level 1 kojom se definiše kompletan model za HTML i XML dokumente
- 2000. godine DOM Level 2, 2004. godine DOM Level 3, 2014. godine DOM4



# Struktura DOM

- Svakom delu dokumenta pridružen je zaseban DOM objekat
- Svojstvima objekta pristupa se sa `objekat.svojstvo`, a metodama sa `objekat.metod(parametri)`
- Svaki DOM objekat ima svoj tip (kojim su određena njegova svojstva i metode)
- Tipovi su određeni interfejsima, a interfejsi su organizovani u hijerarhiju nasleđivanja
- Npr. svi DOM objekti su čvorovi DOM stabla i implementiraju interfejs `Node`; objekti koji odgovaraju elementima dokumenta implementiraju interfejs `Element` koji nasleđuje interfejs `Node`

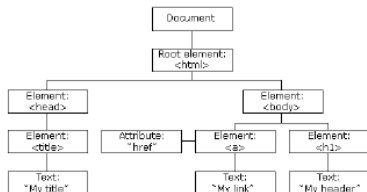
# DOM stablo

- DOM objekti su međusobno povezani i čine strukturu stabla
- Svaki objekat predstavlja jedan čvor stabla; postoje različite vrste čvorova:
  - čvor dokumenta
  - čvor elementa
  - čvor teksta
  - čvor atributa
  - čvor komentara

```

<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <a href="page.html">My link</a>
    <h1>My header</h1>
  </body>
</html>

```



# DOM stablo: interfejs Node

- Svojstva kojima se pristupa osnovnim podacima o čvoru:
  - `nodeType` – tip čvora (1 - element, 2 - atribut, 3 - tekst, 8 - komentar, 9 - dokument)
  - `nodeName` – ime čvora, nepromenljivo svojstvo (ime elementa, ime atributa, `#text`, `#comment`, `#document`)
  - `nodeValue` – vrednost čvora (za attribute – vrednost atributa, za tekst i komentare – sam tekst, za element i dokument vraća `NULL`)
  - `innerHTML` – za svaki čvor sadrži HTML kôd koji ga opisuje (uključujući i njegove naslednike)
- Celom dokumentu odgovara čvor kome se može pristupiti sa `document`, npr. korenom čvoru dokumenta može se pristupiti sa `document.documentElement`

# DOM stablo: interfejs Document

- Objekt koji predstavlja dokument, pored interfejsa Node nasleđuje i interfejs Document
- Metode za lociranje čvorova bez direktne navigacije kroz DOM stablo
  - `getElementById(id)` – locira se čvor sa datim identifikatorom
  - `getElementsByTagName(tag)` – vraća se lista svih objekata (čvorova) koji odgovaraju datom elementu
- Pogledati detalje o DOM svojstvima i metodama iz skripte prof. Filipa Marića
- Primeri: 24\_js.html