

# HTML5 i JavaScript za video igre

## DOM

Predavač:  
doc. dr Jelena Graovac

# Uvod

- **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

# Raspoređivačke mašine

- DOM predstavlja sliku HTML/XML dokumenta u obliku stabla čiji su čvorovi DOM objekti
- Svaki pregledač mora da sadrži **rašćlanjivač** ili **parser** koji čita tekst datoteke i predstavlja ga u obliku DOM-a
- Ove komponente pregledača nazivaju se **raspoređivačke mašine** (*layout engine*)
- Danas najpoznatije:
  - Trident/MSHTML koji koristi Internet Explorer
  - Presto koji koristi Opera
  - Webkit koji koriste Safari, Google Chrome, Google Android...
  - Gecko koji koriste Mozilla alati (Firefox,...)

# 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`

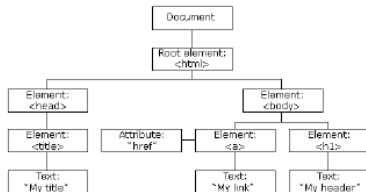
# DOM stablo

- 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

- Čvorovi u stablu su u odnosima: roditelj, dete, brat
- U stablu postoji jedinstveni čvor koji se označava kao **koren**
- Svaki čvor može da ima proizvoljan broj dece; lista dece svakog čvora je uređena
- List je čvor bez dece
- Jedan čvor je brat drugog čvora ako imaju istog roditelja

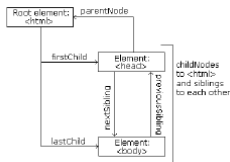
# DOM stablo: interfejs Node

- 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`
- Korenom čvoru dokumenta može se pristupiti sa `document.documentElement`



# DOM stablo: interfejs Node

- Svojstva za navigaciju kroz DOM stablo:
  - `firstChild` – prvo dete čvora, za list vraća NULL
  - `lastChild` – poslednje dete čvora, za list vraća NULL
  - `childNodes` – niz čvorova dece datog čvora
  - `parentNode` – roditeljski čvor, u slučaju korena vraća NULL
  - `nextSibling` – naredni brat čvora, za poslednji čvor u listi dece vraća NULL
  - `previousSibling` – prethodni brat čvora, za prvi čvor u listi dece vraća NULL
  - `attributes` – niz čvorova atributa čvora
  - `ownerDocument` – jedinstveni čvor dokumenta (svi čvorovi su sadržani u ovom čvoru)



# DOM stablo: interfejs Node

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

- Primer: tekstu My header moguće je pristupiti na sledeći način:

```
document.documentElement.firstChild.nextSibling
    .childNodes[1].firstChild.nodeValue
```

# DOM stablo: interfejs Node

- Metode za manipulaciju stablom:
  - `var c = document.createElement("p")` – kreira novi čvor (element `p`) ali ga ne dodaje u stablo
  - `node.appendChild(c)` – dodaje novo dete `c` čvoru `node` (na kraj)
  - `node.removeChild(c)` – uklanja dete `c` čvora `node`
  - `node.insertBefore(nc, rc)` – ubacuje novo dete `nc` pre postojećeg deteta `rc`
  - `node.insertAfter(nc, rc)` – ubacuje novo dete `nc` posle postojećeg deteta `rc`

# DOM stablo: interfejs Document

- Objekat koji predstavlja dokument, pored interfejsa Node nasleđuje i interfejs Document
- Ovaj interfejs sadrži metode za lociranje čvorova bez direktne navigacije kroz DOM stablo:
  - `document.body` – body element
  - `document.getElementById(id)` – locira se čvor sa datim identifikatorom
  - `document.getElementsByTagName(tag)` – vraća se lista svih čvorova koji odgovaraju datom elementu
  - `document.getElementsByClassName(cls)` – vraća se lista svih čvorova koji pripadaju datoj klasi
  - `document.querySelector(sel)` – vraća se prvi čvor koji zadovoljava dati selektor
  - `document.querySelectorAll(sel)` – vraća se lista svih čvorova koji zadovoljavaju selektor
- Ovo je najčešći način na koji se pristupa elementima u stablu

# Promene stilova elemenata

- Svaki čvor koji odgovara elementu ima svojstvo `style`
- Svojstvo `style` sadrži svojstva koja odgovaraju CSS svojstvima
  - `color`
  - `backgroundColor`
  - `display`
  - `borderWidthTop`
  - ...

```
var el = getElementById("mydiv");  
el.style.color = "red";  
el.style.borderWidth = "2px";
```

# Događaji

- **Događaji** (engl. **events**) su mehanizam koji omogućava interakciju veb strane sa korisnikom
- Najčešće nastaju kao reakcija na odgovarajuću radnju korisnika (klik i prelaz mišem, pritisak tastera na tastaturi)
- Postoje i događaji koji se dešavaju nevezano za ponašanje korisnika (npr. učitavanje strane ili nekog elementa na strani je takođe događaj)
- Najzad, događaji se mogu emitovati i ručno iz JavaScript programa, kreiranjem odgovarajućeg objekta događaja (**Event** konstruktor) i pozivanjem metode **el.dispatchEvent(event)** za odgovarajući element
- Element koji emituje događaj zove se ciljani (engl. **target**) element

# Tipovi događaja

- Svaki događaj ima svoj tip koji određuje njegovo značenje
- Tipovi događaja su zadati niskama koje sadrže ime tipa (npr. `click`, `load`, `mouseover`, `keypress` i sl.)
- Interno, događaji su predstavljeni objektima koji se kreiraju `Event` konstruktorom sa argumentom koji predstavlja naziv tipa događaja
- Nazivu tipa događaja za objekat događaja `e` se može pristupiti preko svojstva `e.type`
- Postoje i izvedeni konstruktori `MouseEvent`, `KeyboardEvent` i sl. koji kreiraju objekte događaja sa dodatnim svojstvima koje bliže određuju odgovarajuće događaje (npr. koordinate miša, kôd pritisnutog tastera)
- Događaj se uvek vezuje za neki element – ciljni element kome se može pristupiti pomoću `e.target` (npr. element na koji je kliknuto mišem, element koji je u fokusu prilikom pritiska tastera na tastaturi, i sl.)
- Tipično, veb pregledač automatski kreira odgovarajući objekat i emituje događaj kao reakciju na neko dešavanje ili akciju korisnika

# Obrada događaja

- Događaji se obrađuju tako što se definišu posebne funkcije koje služe kao **upravljači događajima** (engl. **event handlers**)
- Za svaki element mogu se postaviti različiti upravljači za različite tipove događaja
- Kada se događaj desi, tada se najpre poziva upravljač za odgovarajući tip događaja za element koji emituje događaj (ciljni element)
- Nakon toga se događaj u talasu prenosi kroz pretke ciljnog elementa i za svaki od njih se u redosledu ka korenu pozivaju odgovarajući upravljači događajem tog tipa, ako postoje
- Upravljač događaja kao argument ima objekat događaja
- Unutar upravljača događaja ključna reč **this** se odnosi na element za koji se upravljač izvršava
- Događaji se tipično obradjuju na nivou elementa koji je izazvao događaj, ili na nivou **document** čvora



# Registrowanje upravljača događajima

- Prvi način: pomoću HTML atributa:

```
<span onclick="this.style.color = 'red';"> Klikni me </span>
```

- Drugi način: pomoću `onclick` svojstva DOM čvora:

```
<span id="s"> Klikni i mene </span>  
<script type="text/JavaScript">  
  var elem = document.getElementById("s");  
  elem.onclick = function (e) { this.style.color = 'red'; };  
</script>
```

- Treći način: pomoću `addEventListener` metode DOM čvora (preporučljivo):

```
<span id="s"> Klikni i mene </span>  
<script type="text/JavaScript">  
  var elem = document.getElementById("s");  
  elem.addEventListener("click",  
    function (e) { this.style.color = 'red'; });  
</script>
```

## Neki značajni tipovi događaja

- **load**: emituje se kada se završi sa učitavanjem dokumenta ili elementa
- **submit** i **reset**: vezani za forme (vidi dole)
- **resize**: promena dimenzija prozora ili elementa
- **scroll**: skrolovanje dokumenta ili elementa
- **keydown**: taster pritisnut dok je element u fokusu
- **keypress**: taster emitovao štampajući karakter
- **keyup**: taster otpušten
- **mouseover**: miš je postavljen iznad elementa
- **mousedown**: levi taster miša je pritisnut dok je iznad datog elementa
- **mouseup**: levi taster miša je otpušten
- **click**: klik na nekom elementu
- **dblclick**: dvostruki klik na nekom elementu
- **contextmenu**: desni klik na nekom elementu
- ...

# Formulari

- U HTML-u postoji podrška za pravljenje **formulara** u koje posetioci veb-sajtova mogu da unose podatke
- Formular se sastoji iz **kontrola** – elemenata koji omogućavaju unos vrednosti ili izbor opcija
- Osnovni elementi:
  - **form** – predstavlja formular, atribut **method** određuje metod HTTP zahteva prilikom slanja upita koji je prikuljen formularom (GET ili POST), a atribut **action** određuje URL na koji se šalje upit
  - **input** – predstavlja veći broj različitih vrsta kontrola; atribut **name** predstavlja ime (ključ) parametra upita, **value** sadrži pridruženu vrednost, a atribut **type** predstavlja tip kontrole i može imati vrednosti:
    - **text** – polje za unos teksta (atribut **value** je početna vrednost)
    - **password** – polje za unos lozinke
    - **radio** – radio-dugme (sva dugmad iz iste grupe imaju istu vrednost atributa **name**)
    - **checkbox** – polje za štrikliranje (atribut **checked** ako je podrazumevano štrikliran)
    - **button** – obično dugme (koje podrazumevano ne radi ništa)
    - **submit** – dugme čijim se aktiviranjem prikupljeni podaci šalju na server
    - **reset** – dugme koje resetuje formu
    - **hidden** – sakriveno polje sa fiksiranom vrednošću
    - **date** – izbor datuma
    - **email** – dodatna validacija ispravnosti e-mail adrese

# Formulari

- Pored kontrola koja se mogu kreirati `input` elementom, postoje i drugi elementi koji mogu kreirati kontrole u formularu:
  - `select` – predstavlja padajuću listu:
    - atribut `name` određuje ime parametra upita koji je pridružen kontroli
    - atributom `size` postavlja se broj opcija koje se vide (podrazumevano 1)
    - pojedinačne stavke u listi se zadaju elementom `option` koji ima atribut `value` koji predstavlja vrednost pridruženu parametru
  - `textarea` – polje za unos teksta u više redova; atributi `name`, `rows`, `cols`
  - `label` – tekstualni opis pridružen kontroli; atribut `for` sadrži `id` kontrole koja dobija fokus klikom na labelu
  - `button` – kreira dugme koje može imati proizvoljan sadržaj. Atribut `type`: `button`, `submit`, `reset`

# 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" name="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>
```

# Izgleđ formulara

Ime i prezime:

Lozinka:

Adresa:

Vrsta pice:  ▾

Velika

Srednja

Mala

Kečap  Origano