Univerzitet u Sarajevu
Elektrotehnički fakultet Sarajevo
Odsjek za računarstvo i informatiku
Predmet: Web tehnologije

Tutorijal br. 4:
JavaScript (2)

Uvod u tutorijal

Browser Object Model (BOM) - ponavljanje

- http://www.w3schools.com/js/js_window.asp
- https://developer.mozilla.org/en-US/docs/Web/API/Window
- https://developer.mozilla.org/en-US/docs/Web/API/Location
- https://developer.mozilla.org/en-US/docs/Web/API/Navigator
- https://developer.mozilla.org/en-US/docs/Web/API/History

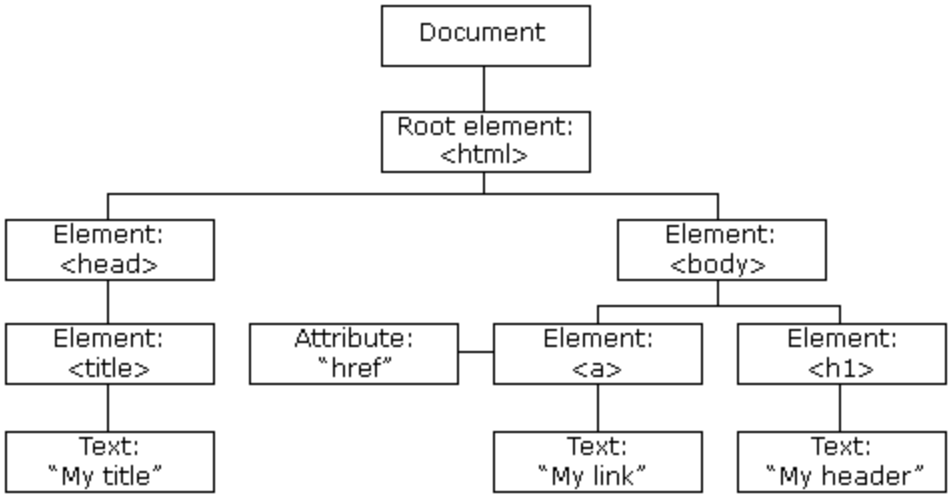
JavaScript nudi niz objekata za manipulaciju samim web preglednikom i okruženjem.

- **Window** - nudi metode za upravljanje samim prozorom web preglednika. Primjeri:
```
window.open("http://etf.ba"); // Otvara link u novom prozoru
window.resize(200,100); // Resize prozora
```
- **Location** - trenutna lokacija (URL). Primjeri:
```
alert(location.href); // Trenutni URL
location.replace("http://etf.ba"); // Redirekcija
```
- **Navigator** - informacije o web pregledniku i sistemu. Primjeri:
```
alert(navigator.appName); // Naziv web preglednika npr. Firefox
```
- **History** - rad sa historijom preglednika. Primjer:`window.history.back(); // Povratak na prethodnu stranicu`

Document Object Model (DOM) - ponavljanje

- http://www.w3schools.com/js/js_htmldom.asp
- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

DOM omogućuje manipulisanje sadržajem web stranice iz JavaScripta. Pomoću DOMa možete: dodavati, mijenjati, brisati sve HTML elemente, njihove atribute, događaje (events) i stil (CSS). Svaki tag predstavljen je instancom objekta HTMLElement iz koje je izveden niz objekata za rad sa specifičnim tagovima. Npr. Table objekat daje metode za pristup redovima i kolonama, dodavanje reda/kolone i slično. Globalni objekat **document** (tipa Document) omogućuje kretanje kroz stablo tagova (primjer stabla na slici ispod).



Specifični HTML element koji vas interesuje možete pronaći metodama:

- **getElementById** - vraća objekat sa datim atributom ID. U validnom HTML dokumentu ID mora biti jedinstven za svaki tag, u suprotnom ponašanje je nedefinisano. Primjer:

```
var tabela = document.getElementById("tabela");
var red = tabela.insertRow(0);
var celija = red.insertCell(0);
celija.innerHTML = "Zdravo!";
```

- **getElementsByName** - vraća niz elemenata sa datim atributom NAME. Za razliku od ID, ovaj atribut je legalan samo na poljima formi i ne mora biti jedinstven (obratiti pažnju na množinu: element**s**). Primjer:

```
var input = document.getElementsByName("adresa")[0]; // Prvo polje sa imenom "adresa"
```

- **getElementsByTagName** - vraća niz elemenata sa datim imenom taga. Primjer:

```
var prvi_paragraf = document.getElementsByTagName("p")[0];
```

- **getElementsByClassName** - niz elemenata prema vrijednosti atributa CLASS.
- **document.forms** - vraća niz Form objekata koji odgovaraju formama na stranici. Primjeri:

```
var email = document.forms["forma"]["email"].value;
var email2 = document.forms[0]["email"].value; // Ako je prva "forma"
var email3 = document.forms[0].email.value;
```

- Kretanje kroz stablo koristeći metode objekta Node koje nasljeđuje i objekat HTMLElement. Primjer:

```
var element = document.lastChild.getElementByTagName("p")[0].childNodes[1].
innerHTML;
```

Objašnjenje: posljednje dijete document-a (body), prvi paragraf (p-tag), njegovo drugo dijete, unutrašnji HTML. Pristup CSSu se vrši kroz svojstvo (property) **style** ali treba obratiti pažnju na tipove podataka i na to da se neki atributi ne zovu isto. Primjer:

```
document.getElementById("celija").style.backgroundColor = "red";
```

HTML tagovi DIV i SPAN koji nemaju nikakav vizuelni efekat posebno se često koriste kako bi se nekom dijelu stranice pridružio ID radi manipulacija preko DOMa.

- **document.querySelector** parametar ove metode je CSS selektor string pomoću kojeg možete pretraživati dokument po imenu taga, klasi ili IDu a vraća se prvi element koji odgovara upitu. Klasa počinje znakom tačka a ID znakom hash (#). Primjeri:
    - *document.querySelector("button")* - vraća prvi HTML element sa tagom <button>
    - *document.querySelector(".mojaklasa")* - vraća prvi HTML element čiji je class atribut "mojaklasa"
    - *document.querySelector("#dugme")* - vraća HTML element čiji je id "dugme"
    - Parametri se mogu i kombinovati npr. *document.querySelector("p.description")* - prvi HTML element sa tagom <p> i klasom "description"
    - Znakom razmak možete se kretati kroz hijerarhiju tagova. Npr: *document.querySelector("#content img")* - vraća prvi HTML element čiji je tag <img> unutar elementa čiji je id "content"
    - Znakovima uglaste zagrade i dvotačka mogu se definisati očekivane vrijednosti atributa. Primjer:
        - *document.querySelector('#formular input[type="radio"]:checked')*
        - vraća prvi element čiji je tag <input>, atribut "type" jednak "radio" a vrijednost mu je "checked", unutar elementa čiji je id "formular".
    - *document.querySelectorAll* vraća niz svih elemenata koji zadovoljavaju uslove selektora npr. *document.querySelector("img")* vraća sve slike u dokumentu.

jQuery

Biblioteka jQuery nudi nešto kraći i jednostavniji zapis određenih JavaScript konstrukcija i bolju podršku za starije web preglednike (mada za Internet Explorer 6-8 morate koristiti jQuery 1.x). Recimo, querySelector možete u jQuery-ju postići naredbom $( ) npr:

```
$('#content img').style.width="100px";
```

Da biste koristili jQuery biblioteku morate je uključiti npr.

```
<script
src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></scrip
t>
```

Ovo je primjer uključivanja biblioteke sa Google sajta. Ako ne želite da Google ima evidenciju ko sve pristupa vašem sajtu, možete jQuery preuzeti i koristiti lokalni URL u vašoj skripti. Da biste kroz jQuery definisali funkciju koja se izvršava odmah nakon učitavanja stranice možete pisati ovako:

```
$(function() { alert('Ucitano'); } );
```

Primjer

Dat je sljedeći HTML kod:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
  <META http-equiv="Content-Type" content="text/html; charset=utf-8">
  <TITLE>Tutorijal 4, Uvod</TITLE>
  <SCRIPT src="t4uvod.js"></SCRIPT>
</HEAD>
<BODY>
  <div id="mojelement">Ovo je moj element :))</div>
  <br><br><br>
  <a href="http://www.etf.ba" onclick="return pritisnut()">Pritisni me</a>
  <br><br><br>
  <input type="button" id="izmjena" onmouseover="return klikDugme(this)"
value="Sakrij">
</BODY>
</HTML>
```

Fajl **t4uvod.js** glasi ovako:

```
window.onload = function () {
  document.getElementById('mojelement').style.fontWeight="bold";
  // postavljamo css stil font-weight na bold
}

function pritisnut() {
  var x = document.getElementById('izmjena');
  x.value="Prikazi";     // pristup atributu elementa
  document.getElementById('mojelement').style.display="none";
  return false;    // osigurava da link ne bude izvrsen
}
function klikDugme(referenca) {      // pristup elementu preko this
  if (referenca.value==="Sakrij") {
    // postavi CSS property display na vrijednost "none"
    document.getElementById('mojelement').style.display="none";
    referenca.value="Prikazi";
    return false;
  }

  referenca.value="Sakrij";
  // Pristup DIV elementu mojelement preko DOM stabla
  var z = document.body.children[0];
  // ili s obzirom da je DOM stablo prvi element je DIV ispod body taga
  z = document.body.children.mojelement;
  // moze i ovako ili document.body.children['mojelement']
  z = document['body']['children']['mojelement'];
  // moze i ovako pa cak i window['document'] :)
  z.style.display="block"; // prikazi ga ili z['style']['display'] = ...
}
```

Pokrenite primjer dat iznad, isprobajte različite opcije i analizirajte koji dio radi šta i zašto.

Zadatak 1.

Korištenjem DHTML-a (JavaScript, CSS, DOM) napraviti interaktivno stablo predmeta. Nakon otvaranja stranice prikazuje se tekst:

+ Prva godina

+ Druga godina

Kada se klikne na znak plus ili tekst "Prva godina" ili "Druga godina" treba se "raširiti" taj dio stabla, a znak plus se treba pretvoriti u minus, tako da izgleda npr. ovako:

- Prva godina
    - IM1
    - IM2
    - OE
    - EES
    - ….

+ Druga godina

Ponovnim klikom na isto mjesto treba se taj dio stabla ponovo skupiti. Pri tome se stranica ne treba osvježavati!

Savjet
Koristite CSS svojstvo **display** kako biste prikazali (display:block) odnosno sakrili (display:none) sloj (DIV) sa sadržajem podstabla.

Tranzicije
Da bi se stablo otvaralo postepeno umjesto odjednom, možete koristiti CSS3 tranzicije, npr. u stil sloja dodajte:

```
transition: height 2s;
```

Nedostatak CSS3 tranzicija je loša podrška u browserima. Da bi tranzicija bila podržana u Chrome <26.0 i Safari <6.1 morate dodati:

```
-webkit-transition: height 2s;
```

a u Firefoxu <16.0:

```
-moz-transition: height 2s;
```

Za IE <10.0 i vrlo stare browsere možete simulirati tranziciju pomoću JavaScript koda sličnog kao u sljedećem zadatku. Alternativa je jQuery .effect() metoda.

Zadatak 2.

Napraviti lopticu-skočicu koja se prikazuje preko web stranice i odbija se od rubove ekrana kao na ovom primjeru: link

Savjeti

- Nađite sliku lopte na Google Images, postavite je na stranicu i podesite apsolutno pozicioniranje.
- Koristite JavaScript funkciju setInterval da ažurirate poziciju lopte (CSS atributi top i left) svakih 0.1 sekundu.
- Pojednostavimo problem tako što ćemo reći: lopta se može kretati u jednom od četiri pravca: gore-lijevo (top i left se smanjuju), gore-desno (top se smanjuje, left se povećava itd.), dolje-lijevo i dolje-desno.
- Kada lopta "udari" u gornji rub pravac se mijenja iz gore-lijevo u dolje-lijevo, a iz gore-desno u dolje-desno. Kada "udari" u desni rub pravac se mijenja iz gore-desno u gore-lijevo, a iz dolje-desno u dolje-lijevo. Slično za donji i lijevi rub.
- Za gornji i lijevi rub znamo da je koordinata 0, za donji i desni koristite svojstva innerWidth i innerHeight objekta Window i ne zaboravite da oduzmete širinu i visinu lopte.

Physics engine

Da bi odbijanje lopte bilo realnije (npr. za igrice) trebate koristiti neki physics engine npr. PhysicsJS.

Regularni izrazi

- http://regexone.com/ - online interaktivni tutorijal
- http://regex101.com/ - automatski tumač i tester
- https://developer.mozilla.org/en/docs/Web/JavaScript/Guide/Regular_Expressions

Regularni izraz (regular expression, regex) je izuzetno moćan alat za provjeru poklapanja nekog stringa sa šablonom. Najviše se koriste za validaciju, izdvajanje stringova na osnovu uzorka (npr. izdvajanje email adrese ili broja telefona iz nekog teksta) i slično. Postoji više regex dijalekata a danas je preovladao dijalekt koji potiče iz programskog jezika Perl (Perl-kompatibilni regularni izrazi, PCRE).

U JavaScriptu regexi se mogu koristiti u funkcijama **search** i **replace** koje su metode objekta String.

Slova i brojevi unutar regexa odgovaraju istim slovima i brojevima u stringu. No određeni znakovi imaju posebna značenja, pa da bismo u stringu pronašli taj znak moramo ispred njega staviti backslash. Primjer: znak tačka označava bilo koji karakter, a ako želimo da pronađemo baš znak tačku u stringu moramo pisati \.

Primjeri:

`var re=/\w+ada/ig;` `console.log(tekst.match(re));`	Vraća sve riječi u tekstu koje završavaju slovima "ada" npr. zgrada, vlada, Kanada. Modifikator **i** osigurava da se zanemaruju velika i mala slova, a **g** da će biti vraćeni svi rezultati. **w** je neki znak koji je dio riječi, + znači da takvih znakova treba biti 1 ili više.
`var re=/(www\.[\w\.\/]+)/;` `alert(tekst.replace(re, " http://$1"));`	Zamjenjuje u tekstu linkove oblika www.nesto sa http://www.nesto Dio u običnim zagradama će biti ubačen umjesto $1 u zamjeni. U uglastim zagradama su navedeni karakteri koji su validni za link: \w (slova i cifre), tačka, slash (treba dodati još karaktera...)
`var re=/(www\.[\w\.\/]+)/;` `alert(tekst.replace(re, " <a href=\"http://$1\">$1</a>"));`	Isto, samo što se sada tekst oblika www.nesto zamijenjuje HTTP linkom (A tag).
`var re=/(\<student\s+ime\=\")(\w+)("\s+prezime\=\")(\w+)(\"\>)/g;` `alert(tekst.replace(re,"$1Drugo$3drugi$5"));`	Ako imamo xml sa tagovima: <student ime=”Neko” prezime=”neki”> i trebamo zamijeniti imena sa Drugo i prezimena drugi <student ime=”Drugo” prezime=”drugi”>

Zadatak 3.

Napravite jednostavan HTML dokument sa TEXTAREA poljem u koje možete ukucati proizvoljan tekst. Zatim napišite JS funkciju koja iz unesenog HTML koda briše sve tagove i ostavlja samo tekst koristeći regex. (HTML tag je niz teksta između znakova < i >). HTML entitete zamijenite njihovom vizuelnom reprezentacijom: & sa &,   sa razmakom, " sa navodnicima.

Zadatak 4.

Napišite JS koji uneseni programski kôd u programskom jeziku C modifikuje tako što sve for petlje zamjenjuje while petljama koristeći regex. Pretpostavite da je tijelo petlje uvijek zatvoreno vitičastim zagradama. Primjer:

```
#include <stdio.h>
int main() {
   int i,x;
   scanf("%d", &x);
   for (i=1; i<=10; i++) {
      printf("%d %d\n", i, pow(x,i));
   }
```

```
    return 0;
}
```

Treba postati:

```
#include <stdio.h>
int main() {
    int i,x;
    scanf("%d", &x);
    i=1;
    while (i<=10) {
        printf("%d %d\n", i, pow(x,i));
        i++;
    }
    return 0;
}
```

Zadatak 5.

Apache access log koristi sljedeći format (Combined log):

```
77.78.252.2 - - [21/Mar/2014:15:51:58 +0100] "GET /css/zamger.css HTTP/1.1" 200 1465 "https://proba/" "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko"
77.78.252.2 - - [21/Mar/2014:15:51:58 +0100] "GET /images/16x16/zad_bug.png HTTP/1.1" 200 1107 "https://proba/" "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko"
77.78.252.2 - - [21/Mar/2014:15:51:58 +0100] "GET /css/print.css HTTP/1.1" 200 1295 "https://proba/" "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko"
77.78.252.2 - - [21/Mar/2014:15:51:58 +0100] "GET /images/etf-50x50.png HTTP/1.1" 200 2062 "https://proba/" "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko"
77.78.252.2 - - [21/Mar/2014:15:51:58 +0100] "GET /images/16x16/dokumentacija.png HTTP/1.1" 200 725 "https://proba/" "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko"
77.78.252.2 - - [21/Mar/2014:15:51:58 +0100] "GET /js/stablo.js HTTP/1.1" 200 316 "https://proba/" "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko"
77.78.252.2 - - [21/Mar/2014:15:51:58 +0100] "GET /images/32x32/dokumentacija.png HTTP/1.1" 200 2717 "https://proba/" "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko"
77.78.252.2 - - [21/Mar/2014:15:51:58 +0100] "GET /favicon.ico HTTP/1.1" 404 1150 "-" "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko"
```

Napišite JavaScript funkciju koja u logu (unesenom u TEXTAREA) koristeći regexe pronalazi sve pristupe koji su rezultirali kôdom 404 (nepostojeća datoteka), a zatim u prostor ispod

TEXTAREA upisuje tabelu sa dvije kolone. U prvoj koloni se treba nalaziti datum i vrijeme u formatu: "21.3.2014. 15:51:58". U drugoj koloni se treba nalaziti lokacija koja je izazvala tu grešku npr. /favicon.ico.

Validacija forme

Dat je sljedeći HTML kôd koji predstavlja formu:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
  <META http-equiv="Content-Type" content="text/html; charset=utf-8">
  <TITLE>Tutorijal 4, Uvod 2</TITLE>
</HEAD>
<BODY>
  <div id="greska" style="color:red; font-weight:bold"></div>
  <form name="mojaforma" id="mfid" action="http://google.com" method="GET">
    Ime: <input type="text" size=100 maxlength=100 id="ime" name="ime">
<br>
    Telefon: <input type="text" size=20 maxlength=20 id="telefon"
name="telefon"> <br>
    Password: <input type="password" name="password"> <br>
    Tip: <input type="text" disabled=true name="tip" value="Student"> <br>
    Godina:<select name="godina">
           <option value="1">Prva</option>
           <option value="2" SELECTED>Druga</option>
           </select><br>
    <input type="checkbox" name="sretan" value="sretan" checked>
Sretan?<br>
    <input type="button" name="dummy" value="Obicni Button">
    <input type="hidden" name="q" value="Web Tehnologije site:etf.unsa.ba">
    <input type="submit" name="posalji" value="Posalji nacin 1">
    <input type="button" name="stela" id="stela" value="Posalji preko
stele">
  </form>
  <SCRIPT src="t4primjer2.js"></SCRIPT>
</BODY>
</HTML>
```

Fajl t4primjer2.js je dat na sljedeći način:

```
function provjeriFormu() {
      var greska = document.getElementById('greska');
                   // ili document.body.children[0]
      greska.innerHTML=""; // ocistimo prethodne greske

      var forma=document.getElementById('mfid');
      if (forma.ime.value.length>10 || forma.ime.value.length<3) {
            greska.innerHTML+="Predugo/prekratko ime<br>";
            return false;
      }

      // Ovaj regex ukucajte u regex101.com za tumacenje:
```

```
	var telefonRegEx = /^\(?(\d{3})\)?[-]?(\d{3})[-]?(\d{3})$/;
	if (!telefonRegEx.test(forma['telefon'].value)) {
		greska.innerHTML+="Telefon format: (061)-123-345 ili
061-123-456 ili 061123456<br>";
		return false;
	}

	if (!forma.sretan.checked) return false;
	var sel=forma.godina.options[forma.godina.selectedIndex].text;
		// vazan je i .value
	if (sel!=="Druga") return false;

	return true;
}

document.getElementById("mfid").addEventListener( "submit",
provjeriFormu, false );

document.getElementById("stela").addEventListener( "click", function(ev) {
	document.forms.mojaforma.submit();
}, false);
```

Testirati i analizirati kod.

Zadatak 6.

1) Dodati novo polje “prezime” i napraviti validaciju koja provjerava da li je prezime uneseno ili nije.

2) Dodati novo polje “email” i napraviti validaciju istog prema pravilu: svaki email mora sadržavati karaktere @ i . , mora postojati najmanje jedan karakter prije simbola @ kao i iza njega. Dio domene mora imati bar dva slova.

(Ovo nije 100% korektna validacija emaila ali je dovoljna za ovaj tutorijal. Koga zanima ispravna validacija email-a prema standardu neka pogleda:

http://www.ex-parrot.com/pdw/Mail-RFC822-Address.html tj.
http://www.ietf.org/rfc/rfc0822.txt)

3) Dodati novo polje “potvrdi password” tipa “password” te uraditi kros-validaciju (da li su jednaki password polja sa “potvrdi password” poljem).

4) Korištenjem DHTML-a (JavaScript, CSS, DOM) napraviti da pozadina polja pocrveni ukoliko unos nije validan a pozeleni ukoliko je validan nakon submita.

5) Korištenjem DOM ***onblur*** eventa za polja napraviti da se polja validiraju nakon gubitka fokusa (da pocrveni ako nije ispravno i da se ispiše greska).

Za najbolje prakse za validaciju formi pogledati:
http://www.smashingmagazine.com/2009/07/07/web-form-validation-best-practices-and-tutorials

Zadaci za dodatnu vježbu (uključujući i dio kod kojeg piše "Napredne web tehnologije"):
https://docs.google.com/document/d/19PU8Ld2RDnfGJj2CM-WT3bgUS5hh27NaEwF5ivnv_IY/edit