ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

Pregled mogucnosti jQuery alata u veb dizajnu

DIPLOMSKI RAD

Kandidat:
Dunjić Stefan, 07/201

Mentor:
Dr Zoran Čiča

Beograd, januar 2014.

SADRŽAJ

SPISAK KORIŠĆENIH SKRAĆENICA

Skraćenica	Pun naziv
DOM	Document Object Model
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
AJAX	Asynchronous JavaScript And XML
URL	Uniform Resource Locator
XML	Extensible Markup Language
JSON	JavaScript Object Notation
jQuery UI	jQuery User Interface
CDN	Google Content Distribution Network

1. UVOD

jQuery je JavaScript biblioteka specijalno dizajnirana da pojednostavi izvršavanje skripti na korisničkoj strani. DŽon Reisig kreirao je jQuery 2005. godine sa ciljem da napravi sažete JavaScript selektore. Selektori su metode za nalaženje DOM elemenata, kao što su na primer: element.getElementById(id), element.getElementsByTagName(tagname)…). Stalnim unapređivanjem biblioteke, sadašnji jQuery predstavlja ultimativan alat za pravljenje skripti koje će biti kompatibilne sa svim Internet brauzerima, i novim i starim verzijama, pri čemu se naročito ističe kompatibilnost sa starijim verzijama Internet Explorer-a.

Pored kompatibilnosti sa različitim Internet brauzerima, glavna prednost jQuery biblioteke je njena efikasnost u pogledu kompleksnosti koda u odnosu na JavaScript. Zbog toga je sasvim opravdan moto jQuery biblioteke "write less, do more", jer za određen kod u JavaScript-u, njegov ekvivalent u jQuery-ju može biti napisan u manjem broju linija.

Cilj ovog rada je da predstavi mogućnosti i način upotrebe jQuery biblioteke. U toku rada će biti predstavljeni primeri, koji će slikovito objasniti funkcionalnosti jQuery-ja. Za potpuno razumevanje jQuery-ja, potrebno je imati osnovno znanje HTML-a i CSS-a.

Rad je organizovan u deset poglavlja. Prvo poglavlje predstavlja uvod u kom se daju osnovne napomene o jQuery biblioteci. U drugom poglavlju je opisan postupak instalacije i način uključivanja jQuery biblioteke. U okviru trećeg poglavlja su opisani selektori i filteri na osnovu kojih jQuery nalazi i uzima sadržaj HTML elemenata. Četvrto poglavlje se bavi manipulacijom sadržaja koji uzimamo sa selektorima. U petom poglavlju je objašnjeno postizanje interakcije sa korisnikom tako što se za određene događaje koje korisnik inicira vezuje određeni deo jQuery koda. U šestom poglavlju su predstavljeni osnovni efekti i animacije, na primer, usporeno otvaranje menija uz određenu animaciju. U okviru sedmog, osmog i devetog poglavlja je predstavljeno osnovno korišćenje jQuery UI dodatka (*plug-in*), validacija formi dodatkom za validaciju (*validation plug-in*) i korišćenje AJAX-a u okviru jQuery-ja. U desetom, poslednjem poglavlju osvrnućemo se na prednosti i mane korišćenja jQuery-ja.

2. INSTALACIJA I KORIŠĆENJE JQUERY-JA

jQuery biblioteka se konstantno unapređuje, samim tim postoji više verzija i podverzija ove biblioteke koje su na raspolaganju. U zavisnosti koje brauzere je potrebno da aplikacija podržava, najbitnije izabrati verziju v1 ili verziju v2 (verzija v2 ne podržava Internet Explorer verzije 6, 7 i 8). Verzija v2 zbog nepodržavanja starijih brauzera zauzima oko 20% manje prostora na disku i samim tim stranica se brže učitava. Pored verzije bitno je odabrati produkcijsku verziju (*minified*) ili razvojnu verziju. Razvojna verzija zauzima značajno više mesta na disku, ali omogućava lakše pronalaženje grešaka u toku razvoja (*debugging*), dok se produkcijska verzija postavlja na sajt.

Postoji više načina za uključivanje jQuery biblioteke unutar sajta. Dva najpopularnija načina su: preuzimanje biblioteke sa Interneta i postavljanje na server, a drugi način je uključivanje putem Gugl CDN-a (*Google Content Distribution Network*). Postavljanje biblioteke na server je standardan način za uključivanje jQuery biblioteke na sajt, ali u poslednje vreme sve značajnije postaje i uključivanje preko Gugl CDN-a koje ima prednost po pitanju performansi, jer ukoliko je korisnik posetio neki drugi sajt koji je na isti način (sa istim URL-om) uključio jQuery biblioteku na svoj sajt, korisnik je već preuzeo biblioteku i nema potrebe za njenim ponovnim preuzimanjem, zbog čega se vreme učitavanje sajta se smanjuje.

Ukoliko se odlučimo da biblioteku postavimo na server, potrebno je da biblioteku uključimo sa odgovarajućim <*script*> elementom koji treba da ima podatke o lokaciji biblioteke. Zbog preglednosti praksa je da se skripte uključuju unutar <head> elementa, iako to nije neophodno da bi biblioteka radila pravilno.

```
<head>
      <title>Naziv sajta</title>
      <script type="text/javascript" src="jquery-1.10-min.js"></script>
</head>
```

Ako biblioteku uključujemo preko Gugl CDN-a, postupak je isti, samo što umesto lokacije biblioteke na serveru, moramo uneti URL na kome se biblioteka nalazi.

```
<head>
      <title>Naziv sajta</title>
      <script
src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
</head>
```

U drugom primeru izostavili smo unutar <script> elementa tip skripte, zbog toga što je u HTML5 osnovno podešavanje da je tip skripte "text/javascript", pa nije neophodno eksplicitno navesti tip skripte. Spisak svih biblioteka koje Gugl podržava da budu uključene na ovaj način nalazi se na sajtu https://developers.google.com/speed/libraries/devguide.

Nakon uključivanja, biblioteku je potrebno koristiti unutar <script> elementa kao i svaki drugi JavaScript kod. Pristup biblioteci se vrši pozivom funkcije jQuery (isti naziv kao i biblioteka). Zbog toga se izbegava konflikt sa drugim bibliotekama koje imaju sopstvene funkcije. jQuery funkcija ima alijas $ za pristupanje biblioteci kako bi bilo izbegnuto pisanje jQuery prilikom svakog poziva biblioteci. Svaka komanda se sastoji iz četiri dela, od alijasa za pozivanje funkcije, selektora, akcije i parametara. Prvo se koristi selektor kako bi odabrali određene elemente na sajtu. Posle toga

biramo akciju koja se primenjuje na elemente koje smo selektovali. Na kraju prosleđujemo parametre koji se odnose na akciju koja se izvršava.

Selektor	Akcija	Parametar
jQuery(‘p’)	.css	(‘color’, ‘red’);
$(‘p’)	.css	(‘color’, ‘red’);

U ovom primeru vidimo selektovanje preko jQuery funkcije u prvom redu i korišćenje alijasa u drugom redu. Prvo su odabrani svi paragrafe (‘p’), potom je odabrana “css” akcija i dodeljena osobina, u ovom slučaju crvena boja.

Važno je napomenuti i da jQuery biblioteka, kao i JavaScript, razlikuje upotrebu malih i velikih slova (*case sensitive*).

3. SELEKTORI I FILTERI

3.1 Selektori

Selektori su metode za nalaženje DOM elemenata. U jQuery biblioteci selektori služe za određivanje elemenata na stranici nad kojima se izvode naprednije operacije. Da bi osigurali da su svi elementi na sajtu učitani na početku skripte, mora se naglasiti da se biblioteka aktivira tek po učitanoj stranici. Unutar jQuery biblioteke to je organizovano sa jednom linijom koda "$(document).ready()". Pošto to samo osigurava da početak rada biblioteke bude na potpuno učitanoj stranici, potrebno je ubaciti i funkciju koju želimo da bude izvršena. Iako će strana raditi i ako početnu liniju deklarišemo više od jednom, praksa je da se deklariše samo jednom unutar jedne stranice zbog preglednosti. Izgled početne linije skripte:

```
$(document).ready(function() {
      // kod koji želimo da se izvrši
});
```

U nastavku ovog rada, da bi izbegli ponavljanje, svaki kod biće unutar ovog dela koda, ukoliko se drugačije ne naglasi.

Selektori se oslanjaju na znanje CSS-a, zato je preporučljivo da se poznaju osnove CSS-a. Osnovni selektor jQuery biblioteke selektuje elemente po HTML tagu. Primeri:

```
$('ovde treba upisati selektore')
```

`$('p')` – selektuje sve paragrafe na stranici

`$('div')` – selektuje sve div elemente na stranici

`$('h1')` – selektuje sve h1 naslove na stranici

Ukoliko želimo da selektujemo određenu klasu elemenata potrebno je selektor napisati sa tačkom na početku:

```
$('.ime klase')
```

`$('.podnaslov')` - selektuje sve elemente klase "podnaslov" na stranici.

Selekciju je moguće uraditi i preko id-a elementa. Tada je sintaksa koju jQuery koristi sledeća:

```
$('#id elementa')
```

`$('#form')` – selektuje jedinstveni element (id elementi su jedinstveni na stranici) na stranici sa form id-em

Naprednije selektovanje se obavlja kombinovanjem osnovnih načina selektovanja:

`$('.data a')` – selektuje sve linkove koji se nalaze unutar data klase

Kombinovanje selektora nije ograničeno na samo dva kombinovanja, već možemo imati komplikovaniji selektor:

`$('form div.data p span')` – selektuje sve span elemente koji se nalaze unutar paragrafa, koji se nalazi unutar div elementa klase data i koji se nalazi unutar forme.

3.2 Filtri

Selektore koje smo opisali u prošlom poglavlju su najefikasniji kada tačno znamo koji element želimo da selektujemo na stranici. U određenim situacijama, kada znamo samo relativno mesto na stranici u odnosu na neke elemente, koje možemo da selektujemo preko običnih selektora, potrebno je koristiti filtre kao dopunu običnim selektorima kako bi se tačno označio element koji želimo da selektujemo. Filtri se koriste tako što se u postojeće selektore dodaje reč koja će bliže opisati selektor.

```
$('selektor :filtar')
```

Najkorišćeniji filtri su sledeći:

- :first – filtrira samo prvi element koji zadovoljava kriterijum selektovanja
- :last – filtrira samo poslednji element koji zadovoljava kriterijum selektovanja
- :eq(n) – filtrira samo n-ti element koji zadovoljava kriterijum selektovanja, indeks n se upisuje unutar zagrada
- :gt(n) – filtrira elemente čiji indeksi su su veći od n-tog, indeks n se prosleđuje unutar zagrada
- :lt(n) – isto kao :gt(), samo indekse koji su manji od n-tog
- :animated – sve elemente koji su u datom trenutku u procesu animacije
- :focus – sve elemente koji se u datom trenutku koriste

`$('parent > child')` – filtar koji funkcioniše prema hijerarhiji u zavisnosti kako je HTML organizovan, koristi se u slučajevima kada znamo da tačno odredimo "roditeljski" element, koji sadrži element "deteta" koji želimo da selektujemo

Primer korišćenja filtara:

`$('.podnaslov > p:first')` – izdvajamo prvi paragraf koji se nalazi unutar elementa klase podnaslov

Važno je napomenuti da za sve filtre koji rade sa indeksima, brojač počinje od nule. Ukoliko stavimo da je indeks negativan, brojač kreće od poslednjeg elementa unazad. Na primer, ukoliko je filtar :lt(-4) on će zanemariti poslednja četiri elementa.

4. MANIPULACIJA SADRŽAJA STRANE

Do ovog poglavlja jQuery nije imao nikakvu akciju, jer kao što smo na početku objasnili da se sintaksa jQuery biblioteke sastoji iz selektora, akcije i parametara. Za početak pokazaćemo kako sa jQuery bibliotekom možemo pristupiti CSS osobinama HTML elemenata. Sintaksa za pristup CSS osobinama je sledeća:

```
$('selektor').css('parametar', 'vrednost');
```

Primer:

```
$(document).ready(function() {
      $('.podnaslov p:first').css('font-size','140%');
      //prvi paragraf unutar elementa sa klasom podnaslov će dobiti CSS
      //osobinu font-size sa vrednošću od 140%
});
```

Ukoliko želimo da prosledimo više osobina odjednom koje želimo da promenimo, ispravno je kopirati liniju iz prethodnog primera i promeniti u željene osobine i respektivne vrednosti. U tom slučaju redovi će se izvršavati jedan za drugim i na kraju ćemo dobiti željeni rezultat. Međutim, kod neće biti dovoljno čitak, te se u takvim situacijama CSS akciji prosleđuje objekat koji sadrži sve osobine i vrednosti koje želimo da promenimo. Tako umesto:

```
$('.podnaslov p:first').css('font-size','140%');
$('.podnaslov p:first').css('color', 'red');
```

možemo napisati sledeći kod:

```
$('.podnaslov p:first').css({
      'font-size' : '140%',
      'color' : 'red'
});
```

Važno je napomenuti da se objekat prosleđuje unutar {} zagrada, a da se osobine unutar njega odvajaju sa zarezom. Isto tako razlika sa standardnim CSS-om je da se CSS osobine i vrednosti prosleđuju unutar navodnika.

Iako je moguće ovako menjati CSS, ovakva upotreba se izbegava, jer stilove treba menjati unutar CSS fajla, a ne na samom elementu, što je upravo ovde slučaj, sem kada je promena zaista mala. U slučaju potrebe većih izmena stilova na elementu, u praksi je najbolje napraviti unutar CSS fajla klasu koja će sadržati te stilove, a onda samo unutar jQuery biblioteke staviti klasu na element koji želimo da ima željene stilove ili u suprotnom obrisati tu klasu. Da bi uspešno dodali klasu na HTML element putem jQuery potrebno je da nam sintaksa bude sledeća:

```
$("selektor").addClass("klasa koju dodajemo");
```

Primer:

```
$("h3").addClass("podnaslov");
```

Analogno možemo i skinuti klasu sa elementa. Sintaksa u tom slučaju je sledeća:

```
$("h3").removeClass("podnaslov");
```

Do sada smo samo selektovali i menjali postojeće elemente, međutim jQuery ima mogućnost dodavanja i brisanja HTML elemenata, isto tako i potpuno menjanje HTML sadržaja određenog elementa. jQuery ima veliki broj mogućnosti prilikom dodavanja elemenata u zavisnosti gde želimo da dodamo element u odnosu na selektovani element.

Ukoliko želimo da element koji dodajemo sadrži neki postojeći element (ubacujemo element oko drugog elementa), onda koristimo opciju .wrap(), kojoj prosledjujemo element koji želimo da dodamo, a postojeći element stavljamo da je selektor. U situaciji kada imamo

```
<body>
	<p>Tekst</p>
	<p>Tekst2</p>
</body>
```

Možemo paragrafe ubaciti u div element klase kontejner:

```
$("p").wrap("<div class='kontejner'></div>");
```

Posle izvršenja ove linije koda, imaćemo:

```
<body>
	<div class="kontejner">
		<p>Tekst</p>
	</div>
	<div class="kontejner">
		<p>Tekst2</p>
	</div>
</body>
```

Svaki nađeni element se smešta u unutar elementa koji prosledimo. Ukoliko pak želimo da sve elemente ubacimo u jedan element, umesto da svaki element ubacujemo pojedinačno u elemente, koristićemo metodu wrapAll umesto wrap. Važno je napomenuti da samo elementi koji zadovoljavaju selektor će biti ubačeni u element (ukoliko se neki element nalazi između elemenata koji su selektovani, taj element neće biti premešten). Isto tako ukoliko element u koji hoćemo da smestimo selektovane elemente, već postoji na stranici, selektovani elementi neće biti premešteni u taj element, već će se napraviti novi element.

Moguće je ubaciti element posle drugog elementa. Za tu svrhu se koristi metoda .after(). Obe ove metode ubacuju element tako što ne modifikuju element koji je selektovan, već samo dodaju pored tog elementa izabrani element. Ukoliko želimo da menjamo HTML strukturu elementa koji smo selektovali potrebno je koristiti metode prepend() ili append() u zavisnosti da li želimo element ili eksplicitno HTML da dodamo na početku ili na kraju selektovanog elementa, respektivno. Ukoliko na HTML kod iz prethodnog primera izvršimo sledeći kod:

```
$("kontejner").prepend("<p>Tekst pre</p>");
$("kontejner").append("<p>Tekst posle</p>");
```

Imaćemo sledeći HTML kod:

```
<body>
	<div class="kontejner">
		<p>Tekst pre</p>
		<p>Tekst</p>
		<p>Tekst posle</p>
	</div>
	<div class="kontejner">
		<p>Tekst pre</p>
		<p>Tekst2</p>
		<p>Tekst posle</p>
	</div>
</body>
```

Pomenute metode ne menjaju postojeći sadržaj elementa, već samo dodaju određene elemente unutar postojećeg. Ipak, u određenim situacijama želimo da kompletno zamenimo sadržaj HTML elementa. U tu svrhu koristimo metodu html(). Ako želimo samo promeniti sadržaj teksta unutar HTML elementa možemo koristiti i metodu text(). Ipak, moramo biti pažljivi, jer ukoliko unutar text() metode imamo neki HTML tag koji treba da se parsira (na primer) on neće biti parsiran, već će zaista na ispisu tog elementa na sajtu stajati . U situaciji kada imamo sledeći HTML kod:

```
<div class=”kontejner”>
	<div class=”prvi”></div>
	<div class=”drugi”></div>
</div>
```

Možemo HTML kod izmeniti na sledeći način:

```
$(".prvi" ).text("Novi tekstualni sadržaj");
$(".drugi").html("<p>Novi sadržaj</ p>");
```

Imamo sledeći HTML kod:

```
<div class=”kontejner”>
	<div class=”prvi”>Novi tekstualni sadržaj</div>
	<div class=”drugi”>
		<p>Novi sadržaj</ p>
	</div>
</div>
```

Isto tako je moguće i obrisati ili isprazniti element. Ukoliko želimo da obrišemo ceo element onda koristimo metodu remove(), koja briše i selektovani element. Ipak, ukoliko hoćemo da obrišemo samo njegove potomke, a da nam ostane selektovani element koristićemo metodu empty(). U sledećem primeru pokazaćemo razliku između ove dve metode na HTML kod iz prošlog primera:

Ukoliko bi izvršili:

```
$(".drugi").remove();
```

Dobili bi:

```
<div class=”kontejner”>
	<div class=”prvi”>Novi tekstualni sadržaj</div>
</div>
```

Ipak sa metodom empty():

```
$(".drugi").empty();
```

Imali bi sledeći HTML kod:

```
<div class=”kontejner”>
	<div class=”prvi”>Novi tekstualni sadržaj</div>
	<div class=”drugi”></div>
</div>
```

5. DOGAĐAJI

Događaji su akcije koje se dešavaju na sajtu, bilo da su posledica delovanja korisnika ili samog brauzera. Kada se neka akcija dogodi, kažemo da se određeni događaj "okinuo", a kada se pri tome izvršava kod koji je vezan za taj događaj, onda možemo da kažemo da smo "uhvatili" događaj.

Svakog trenutka na sajtu postoje hiljade događaja koji se dešavaju: kada korisnik pomeri kursor na ekranu, kada klikne, kada se prozor raširi ili skupi…

Događaje delimo na četiri vrste:

- događaji povezani sa mišem
- događaji povezani sa tastaturom
- događaji unutar forme
- događaji koji su povezani sa stranicom ili prozorom

U okviru ovog tutorijala već smo spomenuli jedan događaj, $(document).ready koji se "okida" kada se učita strana. Isto tako važan događaj je kada korisnik klikne na neki element na stranici. Njega koristimo tako što nam selektor označava element za koji želimo da vežemo događaj, a za akciju biramo click() metodu u čiji argument stavljamo funkciju koju želimo da se pozove kada se klik na taj element dogodi.

```
$('#dugme').click(function() {
      //kod koji se izvršava
});
```

Kada korisnik klikne na element čiji id je dugme, kod će se izvršiti. Na isti način koristi se i metoda dblclick() koja izvršava kod ukoliko se na element dva puta klikne.

U prošlom primeru imali smo da klik "hvatamo" na dugme koje je jednoznačno određeno na strani (određeno sa id-em elementa), pa samim tim znamo na koji smo element kliknuli. Kada "uhvatimo" događaj često želimo nešto uraditi sa elementom koji je "okinuo" događaj. Za tu svrhu nam služi referenca this. Na primer, kada želimo da kliknemo na određeni paragraf i da dodamo klasu samo na paragraf na koji smo kliknuli:

```
$('p').click(function() {
      $(this).addClass('crvena');    //samo paragraf na koji smo kliknuli
                                     // će dobiti klasu crvena, ostali
                                     // paragrafi će biti nepromenjeni
});
```

Unutar događaja koji su povezani sa mišem, postoje još nekoliko metoda:

- mouseenter() – "okida" se kada miš uđe u prostor elementa
- mouseleave() – "okida" se kada miš izađe iz prostora elementa
- hover() – metoda je kombinacija mouseenter() i mouseleave(), samim tim ima dva argumenta, funkciju za mouseenter() i funkciju za mouseleave()

```
$("p").hover(function(){
      // kod kada ulazi u prostor elementa
  },
  function(){
      // kod kada izlazi iz prostora elementa
});
```

Događaji koji su povezani sa tastaturom se koriste na isti način kao i oni koji su povezani sa mišem. Treba izdvojiti dve metode:

- keydown() – "okida" se kada korisnik prvi put pritisne dugme na tastaturi
- keyup() – "okida" se kada korisnik pusti pritisnuto dugme na tastaturi

Kada su u pitanju događaji koji su u vezi sa formom, mogu se primeniti na <input> , <select> ili <textarea> elemente. Mogu služiti kao pomoć korisniku prilikom popunjavanja forme, kako bi korisnik bio obavešten za bitne stvari vezane za formu ili klijentsku validaciju forme. U ovoj vrsti događaja izdvajaju se tri metode:

- focus() – "okida" se kada je određen element forme u fokusu
- blur() – "okida" se kada određen element forme izgubi fokus
- change() – "okida" se kada određen element forme promeni vrednost

Metode focus() i blur() se često koriste u kombinaciji, jer su komplementarne. Na primer, kada želimo da naglasimo element koji je u fokusu na formi, promenićemo mu boju pozadine, kako bi se izdvojio od ostalih <input> elemenata. Međutim, posle toga je potrebno da boja pozadine bude bela, kako se element ne bi više izdvajao ukoliko nije u fokusu:

```
$("input").focus(function(){
    $(this).css("background-color","#cccccc");
});
$("input").blur(function(){
      $(this).css("background-color","#ffffff");
});
```

Za događaje koji su povezani sa stranicom ili prozorom, pored već pomenute ready() metode, postoji i load() metoda koja se izvršava posle učitavanja određenog elementa. Ovoj metodi je moguće proslediti bilo koji element koji je povezan sa URL-om: slike, skripte, okviri ili window objekat. Bitno je naglasiti da se ova metoda ne koristi na ovaj način posle 1.8 verzije jQuery biblioteke.

```
$(".okvir").load(function() {
      // kod koji će se izvršiti kada se elementi sa klasom okvir učitaju
      // na veb stranu
});
```

U svim dosadašnjim primerima smo elementima dodeljivali određene događaje, međutim, kada su jednom dodeljeni, oni bi se "okidali" prilikom svakog tog događaja. Kako bi uklonili dodeljene događaje, potrebno je da koristimo unbind() metodu, koja zajedno sa bind() metodom čini srž dodeljivanja događaja biblioteke, jer bind() metoda je zapravo već implementirana u sve događaje koje smo spomenuli u ovom poglavlju. Samim tim i korišćenje bind/unbind metoda je gotovo isto. Primer kada dodeljujemo događaj elementu sa bind() metodom:

```
$('.slika').bind({
	hover: function(e) {
		// deo koda koji će se izvršiti u slučaju da
		// korisnik kursorom prelazi preko elementa
	},
	click: function(e) {
		// deo koda koji će se izvršiti u slučaju da
		// korisnik klikne na element
	}
});
```

6. ANIMACIJE I EFEKTI

Sakrivanje i pojavljivanje elemenata na stranici je uobičajen zadatak za JavaScript. Kao posledica jednostavnosti selektovanja elemenata i posedovanja rukovodilaca animacije (*animation helpers*), jQuery biblioteka može poslužiti za pravljenje jednostavnih animacija i efekata. U tom segmentu biblioteka se posebno ističe, jer njena jednostavnost dolazi do izražaja, stoga je moguće lakše napraviti željeni efekat i lakše razumeti deo koda sa efektom i animacijom.

Kod animacija i efekata važno je napomenuti da svaka funkcija vezana za efekte i animacije ima opcione parametre u sebi (ukoliko ih izostavimo biće izvršena osnovna podešavanja). Parametri su brzina animacije i funkcija povratnog poziva (*callback function*), koja se poziva nakon što se izabrana animacija završi. Na ovaj način moguće je postići napredne efekte na stranici. Brzina animacije je vreme za koje će animacija biti završena. Kao argument moguće je proslediti reč, koja će odrediti brzinu animacije ("slow", "normal" ili "fast") ili broj koji će biti trajanje animacije u milisekundama.

Prilikom sakrivanja i pojavljivanja elemenata, elementi se zapravo ne brišu, odnosno ne dodaju na stranu. Element postoji u DOM-u, samo se njegova CSS osobina "display" menja na "none" ili "block".

Osnovne metode za sakrivanje i prikazivanje elemenata su:

- show() – prikazuje sakriveni element, a ukoliko je element već vidljiv nema nikakav uticaj
- hide() – sakriva element koji se trenutno vidi na stranici, a nema uticaja ako je element već nevidljiv
- toggle() – sakriva element ako se vidi, a pokazuje ako se ne vidi

Ukoliko ne prosledimo parametre za brzinu efekta, element će se odmah pojaviti ili nestati. Primer:

```
$(".sakriveniTekst").show("slow");
```

Elementi sa klasom "sakriveniTekst" polako će se prikazati.

```
$(".sakrijTekst").hide(2000);
```

Elementi sa klasom "sakrijTekst" će se sakriti u trajanju od 2000 ms.

Za vizuelno lepše prikazivanje i sakrivanje elemenata koriste se metode koje menjaju prozirnost (*opacity*) elementa kroz vreme. Metode za menjanje prozirnosti elemenata poseduju parametar za brzinu i koriste se na isti način kao i osnovne metode i one su:

- fadeIn() – element prvo zauzima prostor na strani (ostali elementi se pomeraju, ukoliko je takav raspored elemenata), pa onda postepeno element postaje sve više vidljiv
- fadeOut() – element sve više bledi, dok ne postane nevidljiv, a potom se prostor koji je zauzimao dodeljuje drugim elementima, iako ostaje vidljiv u DOM-u
- fadeTo() – ova metoda je različita od ostalih metoda. Ona menja prozirnost elementa do određene vrednosti bez obzira da li je element vidljiv ili nevidljiv. Parametri za ovu metodu su brzina i prozirnost i oba parametra su obavezna.

Primer:

```
$('.podnaslov').fadeIn(2000);
```
- za 2 sekunde prozirnost elementa biće 100%

```
$('.podnaslov').fadeTo(2000,0.75);
```
- za 2 sekunde prozirnost elementa biće 75%

Postoji još jedna grupa metoda koja je slična "fade" metodama, a to su "slide" metode, koje umesto menjanja prozirnosti elementa, one vizuelno element pomeraju na veb stranu ili van veb strane. Metode su slideDown() i slideUp(). Prva ubacuje element na stranu i polako povećava visinu elementa sve do 100%, dok druga sklanja element tako što smanjuje visinu elementa do 0%, a zatim menja CSS osobinu "display" na "none". Kao i ranije pomenutim metodama, metodama slideDown() i slideUp() je možemo proslediti argument za brzinu animacije (reč ili trajanje u ms).

```
$( ".podnaslov" ).slideUp();   //visina elemenata sa klasom podnaslov će se
                               //smanjivati i elementi će smanjivanjem
                               //visine na 0% biti sakriveni
```

Sem ovih metoda koje su uključene unutar jQuery biblioteke, moguće je koristiti i metodu animate() pomoću koje možemo napraviti neke animacije koje su prilagođene našim potrebama. Metoda animate() koristi CSS osobine koje smo spomenuli u prethodnim poglavljima. Ona animira element od postojećih CSS osobina do onih koje dodelimo, tako da ne pravi naglu promenu, već se ta promena dešava postepeno u vremenskom trajanju koje odredimo sa parametrom za brzinu. Primer:

```
$('.podnaslov').animate(
      {
            left: '300px',
            fontSize: '16px'
      },
      2000
);
```

Klasa podnaslov će dobiti ove CSS osobine na kraju animacije koja će trajati dve sekunde.

7. jQUERY UI DODATAK

jQuery UI biblioteka je kolekcija naprednih jQuery dodataka, efekata i interakcija. Ona sadrži selektore za datum, umetke (*tabs*), harmonika meni(*accordions*), okvire za dijalog (*dialog boxes*)…

Uključivanje jQuery UI biblioteke unutar sajta je komplikovanije u odnosu na uključivanje same jQuery biblioteke. Razlog toga je sama veličina biblioteke. Razvojna verzija biblioteke je trenutno 430KB, dok je verzija za produkciju 230KB, a to je i pored današnje povećane brzine interneta, još uvek velika veličina za samo jedan deo sajta, ako se uzmu u obzir današnji standardi, gde je prosečna veličina cele stranice sa svim slikama, CSS, JS, HTML… oko 1 MB. Zbog toga pre nego što preuzmemo jQuery UI biblioteku sa ovog sajta http://jqueryui.com/download/, potrebno je da imamo u vidu koje funkcionalnosti biblioteke želimo da koristimo i da u zavisnosti od toga selektujemo te funkcionalnosti pre preuzimanja biblioteke. Ukoliko nećemo koristiti puno funkcionalnosti biblioteke, sama njena veličina može biti i do 40KB.

Potrebno je izabrati i temu koja omogućava da potrebne elemente biblioteke prilagodimo samom dizajnu sajta koji već postoji. Proces biranja teme se sastoji od biranja boje, ikonice i mnogo drugih podešavanja za jQuery UI elemente. Kada je sav proces selekcije završen, potrebno je preuzet sadržaj uključiti unutar sajta, na isti način kao kada smo uključivali samu jQuery biblioteku. jQuery UI biblioteka podrazumeva da je jQuery već sadržan unutar sajta, jer bez nje neće raditi. Zbog toga prilikom preuzimanja jQuery UI biblioteke, zajedno sa njom se preuzima i osnovni jQuery, ali isto tako i CSS fajl koji sadrži stilove za temu koju smo izabrali. CSS fajl po izboru možemo uključiti samostalno ili dodati ga u neki od postojećih fajlova koji sadrže stilove za sajt. jQuery UI biblioteka u sebi sadrži mnogo funkcionalnosti. U ovom poglavlju biće opisano par najčešće korišćenih funkcionalnosti.

Jedan od elemenata koji se najčešće pojavljuju na sajtovima je element za odabir datuma. Pravilno pravljenje elementa za odabir datuma može vremenski puno da košta. Sa jQuery UI bibliotekom programiranje elementa za biranje datuma se svodi na jednu liniju koda ukoliko želimo osnovno biranje datuma na engleskom jeziku. Biblioteka podržava mnogobrojna prilagođavanja, pa je tako moguće namestiti i lokalnu verziju datuma (ime dana u nedelji i meseca), da li će se prikazivati imena dana u nedelji ili ne, da li će se prikazivati dani iz prethodnog ili budućeg meseca u istom okviru ili ograničiti opseg u kome će biti moguće odabrati datum. Primer:

```
$(function() {
      $("#odabir_datuma").datepicker();
});
```

Element sa id-em “odabir_datuma” će postati element za biranje datuma. Ukoliko je taj element datum će biti vidljiv sve vreme. U praksi se najčešće odabir datuma “vezuje” za <input> element. Tada je odabir datuma nevidljiv sve dok polje za unos datuma nije u fokusu.

Modalni dijalozi su obaveštenja koja iskaču iz strane i moraju se prihvatiti kako bi korisnik nastavio sa korišćenjem strane. Oni se koriste samo kada je to neophodno, jer korisnici ne žele stalno prekidanje u radu. Zbog sve češće upotrebe AJAX-a (AJAX će biti opisan u pretposlednjem poglavlju) ovakav način obaveštavanja korisnika se izbegava. Međutim i dalje je prisutan u velikoj meri. jQuery UI omogućava jednostavno prilagođavanje ovih dijaloga, kako bi dijalozi bili prilagođeni dizajnu našeg sajta i izbegli osnovan izgled modalnih dijaloga.

Primer:

```
$(function() {
	$( "#dijalog" ).dialog();	//sadržaj modalnog dijalog biće sadržaj
					// elementa sa id-em dijalog, a stil samog
					// dijaloga zavisi od teme koje smo odabrali
					// ili nekog prilagođenog CSS fajla koji smo
					//sami napravili
});
```

Modalni dijalozi se ubacuju unutar sakrivenih elemenata. Naslov elementa biće naslov modalnog dijaloga, a sadržaj elementa biće sadržaj modalnog dijaloga.

Opis kontrole (*tooltip*) je element koji se pojavljuje kada se kursorom pređe preko kontrole. Opis se ispisuje u element koji se pojavljuje pored elementa preko koga se pređe. U sadržaj opisa elementa ulazi naziv elementa (title) preko koga se prelazi. Internet pretraživači podržavaju ovakav element i bez jQuery UI, međutim tada ne postoji mogućnost da se takav element stilizuje. Potrebna je samo jedna linija koda kako bi jQuery UI biblioteka preuzela prikaz opisa kontrole na strani:

```
$(function() {
	$(document).tooltip();	//omogućava opciju da se prikazuje opis
				//kontrole na celoj veb stranici, stil opisa
				//kontrole zavisi od teme koje smo odabrali
				//ili nekog prilagođenog CSS fajla koji smo
				//sami napravili
});
```

Harmonika (*accordion*) je element čiji sadržaj je podeljen u logičke sekcije. Sadržaj sekcije se otvara kada se klikne na određenu sekciju. Svaka sekcija mora da ima zaglavlje i sadržaj. Zaglavlje se vidi i kada je sekcija zatvorena. Klikom na zaglavlje, otvara se sadržaj te sekcije. Harmonika element zahteva poseban raspored HTML elemenata. Potrebno je da ceo sadržaj bude unutar jednog elementa, zaglavlja da budu unutar <h3> oznaka i sadržaj sekcije da bude unutar elementa koji ide posle zaglavlja:

```
<div id="accordion">
	<h3>Sekcija 1</h3>
	<div>sadržaj prve sekcije</div>
	<h3>Sekcija 2</h3>
	<div>sadržaj druge sekcije</div>
</div>
```

Samo uključivanje jQuery UI harmonike posle toga je jednostavno, izvršava se u jednoj liniji, ali je potrebno ispuniti zahtevan raspored HTML elementa koji je selektovan:

```
$(function() {
	$("#accordion").accordion();	// harmonika će biti napravljena od unutar
					// elementa sa id-em "accordion"
});
```

Za razliku od harmonike kod koje je prikaz zaglavlja sekcija vertikalan, umeci (*tabs*) imaju prikaz zaglavlja sekcija koji je horizontalan. HTML kod se razlikuje u odnosu na harmoniku, jer je

potrebno napraviti listu bez redosleda (*unordered list*) za navigaciju između sekcija (u ovom slučaju umetaka):

```
<div id="umeci">
	<ul>
		<li><a href="#umetak1"> Zaglavlje prvog umetka</a></li>
		<li><a href="#umetak2">Zaglavlje drugog umetka</a></li>
	</ul>
	<div id="umetak1">sadržaj prvog umetka </div>
	<div id="umetak2">sadržaj drugog umetka </div>
</div>
```

Uključivanje umetaka sa jQuery UI bibliotekom je isto kao i za harmoniku, u jednoj liniji, ali je potrebno ispuniti zahtevan raspored HTML elemenata:

```
$(function() {
	$("#umeci").tabs();
});
```

8. jQUERY VALIDACIJA FORMI

U okviru ovog poglavlja biće opisana klijentska validacija. Važno je napomenuti da i pored aktivne klijentske validacije, uvek je potrebno uraditi serversku validaciju, jer je moguće da podaci stignu sa neželjene strane (da naša strana bude hakovana).

Programiranje sopstvene validacije formi može da bude komplikovano. Pogotovu ako se radi o email adresi, telefonu ili broju kreditne kartice. Tada je potrebno znati regularne izraze (*regular expressions*) kako bi napravili pravilnu validaciju. U tome može pomoći jQuery dodatak za validaciju (*jQuery Validation Plugin*). Takođe je koristan dodatak ukoliko se radi validacija u toku samog unosa podataka.

jQuery dodatak za validaciju se preuzima sa adrese http://jqueryvalidation.org/. Pored osnovno jQuery biblioteke potrebno je uključiti i jquery.validate.js skriptu unutar sajta i ukoliko je potrebno uključiti i skriptu za dodatni jezik (osnovni jezik je engleski). Unutar <head> HTML elementa je potrebno uključiti skriptu na sajt:

```
<script src="jquery.validate.js"></script>
```

Validacija forme se poziva sa validate() metodom. Metodi prosleđujemo imena polja nad kojima želimo da izvršimo validaciju, pravila koje želimo da polja zadovolje prilikom validacije i tekst poruke u slučaju greške, tj. da polje ne zadovoljava pravilo za validaciju. U dodatak su uključena najčešće korišćena pravila, kao što su: required, email, url, date, number, creditcard, equalTo, minlength, maxlength, min, max, range… Pored predefinisanih pravila, moguće je definisati i nova pravila.

Ukoliko validacija na nekom polju nije uspešna, jQuery dodatak za validaciju će dodati labelu pored tog polja sa klasom “error”, koju ukoliko želimo možemo posebno da stilizujemo. U osnovnom podešavanju tekst greške biće ispisan crvenim slovima i lako je uočljiv. Poruke za grešku u osnovnom podešavanju su na engleskom jeziku, stoga je potrebno poruke za grešku prilagoditi jeziku na kojem je veb strana. Moguće je prilagoditi poruku za grešku za svaku formu ponaosob, ali praktikuje se da ukoliko imamo više formi na sajtu koristimo jedinstvene poruke za isti tip greške, a time ćemo umanjiti i redundantnost koda. Tada uključujemo i skriptu koja u sebi sadrži poruke za grešku na jeziku koji želimo da prikažemo poruku. U tom fajlu (fajl se nalazi u direktorijumu *localization*, a ime fajla sadrži ime jezika koji uključujemo) možemo direktno menjati tekst poruke za grešku koji će tada biti isti za sve forme koje imamo na sajtu. Važno je napomenuti da ukoliko prilikom poziva metode za validaciju deklarišemo tekst poruke za grešku, taj tekst će zameniti osnovni tekst poruke za grešku (bilo da je on na engleskom jeziku ili učitan preko dodatnog fajla za nov jezik).

HTML kod forme za validaciju:

```
<form id="forma" method="post" action="">
      <input id="ime" name="ime" type="text" />
      <input id="prezime" name="prezime" type="text" />
      <input id="username" name="username" type="text" />
      <input id="password" name="password" type="password" />
      <input id="confirm_password" name="confirm_password" type="password" />
      <input id="email" name="email" type="email" />
```

```
	<input type="checkbox" class="checkbox" id="saglasnost" name="saglasnost"
/>
	<input class="submit" type="submit" value="Pošalji"/>
</form>
```

jQuery kod za validaciju forme:

```
$("#forma").validate({
	rules: {
		ime: "required", // da bi forma bila validna ime mora biti popunjeno
		prezime: "required",
		username: { //osim teksta, za pravilo validacije moguće je
				//proslediti objekat
			required: true,
			minlength: 5   //potrebno je da username ima 5 karaktera
		},
		password: {
			required: true,
			minlength: 8
		},
		confirm_password: {
			required: true,
			minlength: 8,
			equalTo: "#password"    //potrebno je da polje bude identično
						//polju "password"
		},
		email: {
			required: true,
			email: true
		},
		saglasnost: "required"  //u slučaju polja za potvrdu, kada imamo
					//"required" pravilo onda to polje mora biti
					//potvrđeno
	},
	messages: { //deklarišemo tekst greške za određene tipove greške
			//na definisanim poljima
		ime: "Molim Vas unesite vaše ime",
		prezime: "Molim Vas unesite vaše prezime",
		username: {
			required: "Molim Vas unesite korisničko ime ",
			minlength: "Vaše korisničko ime mora da ima bar 5 karaktera"
		},
		password: {
			required: "Molim Vas unesite lozinku",
			minlength: "Vaša lozinka mora da ima bar 8 karaktera"
		},
		confirm_password: {
			required: "Molim Vas unesite lozinku",
			minlength: "Vaša lozinku mora da ima bar 8 karaktera",
			equalTo: "Molim Vas unesite lozinku koja se poklapa sa
prethodnom lozinkom"
		},
		email: "Molim Vas unesite pravilnu email adresu",
		saglasnost: "Molim Vas potvrdite da ste saglasni sa uslovima
korišćenja"
	}
});
```

Ako se validacija izvrši na ovaj način, onda će u slučaju greške biti ispisane poruke za grešku koje smo ovde definisali, a ne preko dodatnog fajla. Ukoliko pak želimo da imamo poruke za validaciju iz posebnog fajla za jezik koji koristimo, messages objekat ne prosleđujemo kao argument validate metode.

9. jQUERY SA AJAX-OM

AJAX je skraćenica za Asinhroni JavaScript i XML. Korišćenjem AJAX-a, Internet aplikacija može da šalje i prima podatke sa servera asinhrono (u pozadini) bez smetnji na postojećoj strani (nema potrebe za osvežavanjem strane). Možemo ažurirati delove stranice bez prekidanja korisnika. Zbog ovoga korisnik može dobiti utisak da je odziv sajta mnogo bolji i utisak je približniji desktop aplikacijama.

Podaci se preuzimaju preko XMLHttpRequest objekta. Iako sadrži XML u imenu, upotreba XML-a nije neophodna, već se često podaci razmenjuju preko JSON-a (*JavaScript Object Notation*) ili samog HTML. Pošto Internet pretraživač pokreće AJAX poziv, postoji mala razlika u pokretanju AJAX poziva između različitih Internet pretraživača. Upravo iz ovog razloga jQuery je idealno rešenje, jer je biblioteka kompatibilna za različite internet pretraživače, a AJAX poziv će biti identičan za sve Internet pretraživače. jQuery će nam biti od pomoći i pri manipulaciji sadržaja strane (u četvrtom poglavlju ovog tutorijala smo videli manipulaciju sa DOM-om), jer je u većini slučajeva potrebno da sa odgovorom koji dobijemo od servera ažuriramo deo strane.

jQuery poseduje više različitih AJAX metoda i one se razlikuju po tipu podataka koji se šalju ili primaju (JSON ili HTML) i po načinu slanja podataka (POST ili GET). Ipak sve ove metode su skraćenice jedne metoda koja obuhvata sve pomenute načine, a to je metoda ajax().

Najjednostavnija AJAX metoda je load(). Ona učitava HTML sadržaj fajla sa servera unutar elementa za koji se poziva. Prilikom učitavanja sadržaja fajla koristi se metoda innerHTML() kako bi se sadržaj ubacio u odgovarajući element, pa zbog toga postoji mogućnost da učitan sadržaj bude različit od onog sadržaja kada bi fajl bio pozvan samostalno. Ograničenje load() metode je da fajl koji učitavamo mora biti na istom domenu kao i internet strana koja ga poziva, jer u određenim slučajevima Internet pretraživači će blokirati ovaj AJAX zahtev.

```
$( "#rezultat" ).load( "fajl_koji_želimo_da_učitamo.html", "podaci_koje_
šaljemo");
```

Podaci koje šaljemo su opcioni parametar load() metode. Mogu biti u formi stringa, tada će podaci biti poslani kao da smo izvršili GET zahtev. Ukoliko pošaljemo objekat umesto stringa kao podatak koji šaljemo, POST zahtev će biti izvršen.

```
$( "#rezultat" ).load( "fajl_koji_želimo_da¬_učitamo.html", function(data,
status, response) {
        //kod koji će se izvršiti kada je učitavanje HTML sadžaja završeno
});
```

Funkcija povratnog poziva je isto opcioni parametar.

Još dve metode su skraćenice metode ajax(), a to su get() i post(). Obe metode imaju iste parametre u pozivu, jedina razlika je u HTTP zahtevu, koji je u slučaju get() metode GET zahtev, dok u slučaju post() je POST zahtev. Pozivanje metoda se vrši na sledeći način:

$.get(url, data, callback, dataType);

$.post(url, data, callback, dataType);

URL – je lokacija fajla koji želimo da učitamo

data – podaci koje želimo da prosledimo serveru

callback – je funkcija koja se "okida" samo ukoliko je zahtev bio uspešan i biće joj prosleđen odgovor servera (response) i status zahteva

dataType – određujemo tačan tip podataka koji će biti prosleđeni callback funkciji, ukoliko se izostavi jQuery će sam odrediti tip podataka

Svi parametri sem URL, su opcioni.

Mana ovog pristupa, do verzije 1.5 jQuery biblioteke, je bila da ukoliko bi došlo do nekakve greške u slanju podataka, ni korisnik na sajtu, a ni u samom kodu ne bismo znali da je do greške došlo, već se samo kod ne bi izvršio. Zbog toga su uvedene tri metode, koje moraju da se koriste odmah posle AJAX zahteva, jer su povezane. Povezane metode su done(), fail() i always(). Metoda done() se izvršava kada se AJAX poziv uspešno završi, za razliku od fail() metode koja se izvršava u slučaju neke greške. Always() metoda se uvek izvršava bez obzira da li se poziv završio uspešno ili ne. Sve tri metode dobijaju podatke sa servera, poruku statusa (success, error ili timeout) i XMLHTTPRequest objekat. Primer:

```
$.get("fajl_koji_želimo_da¬_učitamo.html")
.done(function(data, status, xhr){
      // kod koji se izvršava ukoliko se poziv završio uspešno
})
.fail(function(data, status, xhr){
      // kod koji se izvršava ukoliko je došlo do neke greške
})
.always(function(data, status, xhr){
      // kod koji se izvršava bilo da je poziv uspešno završen
      //  ili je došlo do greške
});
```

Metoda ajax(), koja obuhvata sve pomenute metode (load, get i post), ima drugačiju sintaksu. Potrebno je naglasiti tip HTTP zahteva. Takođe deo za uspešan poziv i greške prilikom slanja je različit. Primer za ajax() metodu:

```
$.ajax({
      type: "POST",      // tip HTTP zahteva
      url: "fajl_koji_želimo_da_učitamo.html", //adresa fajla (HTML ekstenzija
                                               //je primer)
      data: {
            ime_promenljive: vrednost_promenljive
            },    // podaci koje želimo da prosledimo fajlu
      error: function() {
                  // deo koda koji se izvršava u slučaju greške
      },
      success: function() {
                  // deo koda koji se izvršava ukoliko je AJAX poziv uspešan
      },
      complete: function() {
                  // deo koda koji se izvršava bez obzira da li
                  // je poziv bio uspešan ili ne
      }
});
```

Generalno pravilo je da se prvo isprogramira kod za slučaj greške, pa tek onda za slučaj kada je poziv uspešan. Takvim pristupom se izbegava da se možda zaboravi programiranje za slučaj greške, ali isto tako ima se uvid u neke manje očigledne greške sa ostatkom koda.

```
$.ajax({
	type: "POST", // tip HTTP zahteva
	url: "ajaxfile.php", // adresa fajla kome želimo da prosledimo podatke
	data: { podatak1: 1, podatak2: 2 }, // šaljemo promenljive podatak1 i
                                        // podatak2 sa vrednostima 1 i 2
                                        // respektivno
	error: function() {
		$('#status').text('Greška prilikom slanja! Pokušajte
ponovo.').slideDown('slow');
		// unutar elementa sa id-em status upisujemo poruku kojom ćemo
		// obavestiti korisnika da se greška desila, element ćemo polako
		// prikazati
	},
	success: function() {
		$('#status').text('Uspešno ste poslali!');
		// unutar elementa sa id-em status upisujemo poruku da smo uspešno
		// poslali podatke
	},
	complete: function() {
		setTimeout(function() {
			$('#status').slideUp('slow');
		}, 3000);
		// 3 sekunde po završetku odgovora (bio uspešan ili ne) element sa
		// porukom o statusu će se polako smanjiti i nestati
	}
});
```

10. ZAKLJUČAK

jQuery biblioteka se trenutno nalazi na 65% aktivnih sajtova. Samim tim predstavlja najpopularniju JavaScript biblioteku. Podrška za različite Internet pretraživače predstavlja glavnu prednost biblioteke, zbog toga je popularna među programerima.

Biblioteka poseduje mnogobrojne dodatke koji se svakodnevno razvijaju i omogućavaju još lakše korišćenje svih funkcionalnosti koje jQuery pruža. Jednostavnost AJAX poziva je svakako još jedna od velikih prednosti biblioteke.

Zbog popularnosti mobilnih uređaja, trenutno je aktuelna jQuery Mobile verzija koja se svakodnevno unapređuje i optimizuje za rad sa uređajima sa ekranima na dodir. Svakako da će ovo biti pravac razvijanja jQuery biblioteke, kao i mogućnost nepodržavanja starijih verzija Internet Explorer-a, kako bi se povećale performanse i smanjila veličina biblioteke.

jQuery trenutno predstavlja osnovu Internet programiranja i svakako da poznavanje ove biblioteke može olakšati svakodnevno programiranje. Jedina mana jQuery biblioteke je što se oslanja na prethodno poznavanje CSS-a, HTML-a i JavaScript-a.

LITERATURA:

[1] http://www.w3schools.com/

[2] http://jquery.com/

[3] http://jqueryui.com/

[4] http://jqueryvalidation.org/

[5] https://developers.google.com/speed/libraries/devguide

[6] Earle Castledine & Craig Sharkie “Novice to ninja”, 2012

[7] David Sawyer McFarland “JavaScript & jQuery: the missing manual”, 2012

[8] Bear Bibeault & Yehuda Katz “jQuery in Action”, 2010