JavaScript jQuery

Autori:
Milan Segedinac
Goran Savić

1. jQuery

JavaScript biblioteka jQuery[1] pojednostavljuje pristup elementima veb stranice, izmenu sadržaja i izgleda (css stilova) veb stranice, interakciju sa korisnikom i animacije, kao i komunikaciju sa back end delom aplikacije bez ponovnog učitavanja veb stranice. Da blismo mogli da koristimo ovu biblioteku, kao i svaki drugi JavaScript script, trebamo da je preuzmemo i uvezemo u HTML stranici pomoću `<script src="jquery-3.2.1.js"></script>`. Biblioteka jQuery uvodi jednu funkciju koja se zove baš `jQuery`, ali se najčešće koristi njena alias `$`. Ta funkcija kao parametar prima selektor po kom se preuzima element ili kolekcija elemenata iz stranice, a vraća objekat nad kojim mogu da se pozivaju odgovarajuće metode. Sintaksa korišćenja jQuery je data listingom ispod.

```
$(slektor).akcija();
```

Listing – jQuery sintaksa

Pogledajmo na listingu ispod kako se zadaju akcije pomoću jQuery.

```
<html>

<head>
	<script src="jquery-3.2.1.js">
	</script>
</head>
<body>
	<h2>jQuery sintaksa</h2>
	<p>Ovo je prvi paragraf.</p>
	<p>Ovo je drugi paragraf.</p>
	<button>Sakrij paragrafe</button>
	<script>
	$("button").click(function(){
		$("p").hide();
	});
	</script>
</body>
</html>
```

Listing – korišćenje jQuery

Na listingu vidimo da smo u zaglavlju HTML stranice uvezli jQuery biblioteku. Na samom kraju tela HTML stranice nalazi se `script` element u kom smo selektovali sve elemente tipa `button` pomoću `$("button")`. U slučaju posmatrane stranice vraćena kolekcija sadrži samo jedan element. Nad vraćenom kolekcijom izvršili smo metodu `click`, čime smo na događaj klika na svaki od elemenata kolekcije vezali obradu događaja. Obradu događaja smo opisali callback funkcijom, koja je u posmatranom slučaju zadata neimenovanim funkcijskim izrazom koji je napisan direktno kao argument metode click. U

[1] https://jquery.com/

telu callback funkcije ponovo smo iskoristili jQuery da selektujemo sve elemente tipa paragraf. Nad selektovanom kolekcijom izvršili smo metodu `hide()`, kojom se sakrivaju elementi tako što im se postavi css stil `display:none`. Implementirana funkcionalnost omogućuje da, kada kliknemo na dugme `Sakrij paragrafe` u veb stranici, paragrafi budu sakriveni.

Šta bi se desilo da smo skript u kome smo koristili jQuery postavili u zaglavlje ili na početak tela HTML stranice? Izvršio bi se kod za selekciju dugmeta nad do sada učitanim delo DOM stabla. Pošto još uvek ne bi bilo učitano dugme vratila bi se prazna kolekcija. Zatim bi se nad tom praznom kolekcijom definisao događaj klika. Nakon skripta bi se učitao i ostatak HTML stranice i konstruisao ostatak DOM stabla, ali na dugme ne bi bio postavljen događaj. Zbot toga jQuery omogućuje da se sačeka sa učitavanjem čitave stranice, što je prikazano listingom ispod.

```
<html>

<head>
      <meta charset="UTF-8">
      <script src="jquery-3.2.1.js">
      </script>
      <script>
      $(document).ready(function(){
            $("button").click(function(){
                  $("p").hide();
            });
      });
      </script>
</head>

<body>
      <h2>jQuery sintaksa</h2>
      <h2>jQuery sintaksa</h2>
      <p>Ovo je prvi paragraf.</p>
      <p>Ovo je drugi paragraf.</p>
      <button>Sakrij paragrafe</button>
</body>
</html>
```

Listing - `$(document).ready`

Navedeni kod funkcionalno je identičan kodu iz prethodnog primer - takođe će se sakriti svi paragrafi kada se klikne na dugme `Sakrij paragrafe`. Međutim, dodavanje reakcije na klik događaj smo odložili do momenta kada je dokument učitan tako što smo to postavljanje obrade događaja upakovali u callback funkciju koju smo prosledili metodi `$(document).ready`. Obratite pažnju da je ponovo u pitanju jQuery poziv: selektovan je element `document` i obrađen njegov događaj `ready` slanjem callback funkcije.

Selektori i događaji

Sleketori jQuery biblioteke neznatno se razlikuju od css selektora. Navešćemo najkorišćenije selektore:

Selektor	Primer	Objašnjenje
*	`$('*');`	Svi elementi na strani
#id	`$('#id_1');`	Element sa zadatim id-om
.class	`$('.testKlasa')`	Svi elementi zadate klase
Element	`$('p')`	Svi elementi zadatog tipa. Na primer svi paragrafi
a, b, c. … n	`$('th, td, .class, #id')`	Višestruka selekcija. Svi elementi koje selektuje makar jedan od selektora odvojenih zarezom.
parent child	`$('li a')`	Svi elementi tipa child koji se nalaze unutar elementa tipa parent.
:first	`$('ul li:first');`	Prvi element
:last	`$('ul li:last');`	Poslednji element
:first-child	`$('ul li:first-child');`	Prvi child element
:last-child	`$('ul li:last-child');`	Poslednji child element
:not(a)	`$ ('input:not(:checked)');`	Svi elementi na koje se ne odnosi selektor
:odd	`$('ul li:odd');`	Neparni elementi
:even	`$('ul li:even');`	Parni elementi
:parent	`$('li:parent');`	Parent element za selektovani element
[attribute]	`$('[href]');`	Elementi sa zadatim atributom
[attribute=value]	`$('[rel=external]');`	Elemenit čiji zadati atribut ima zadatu vrednost

Tabela – jQuery selektori

Neki od najčašće postavljanih događaja su:

- Događaji miša
  - `click` - klik miša
  - `dblclick` - dvoklik miša
  - `mouseenter` - aktivira sa prilikom ulaska pokazivača miša u oblast elementa
  - `mouseleave` - aktivira se kada pokazivač miša napust oblast elementa
  - `hover` - aktivira se dok je pokazivač miša iznad elementa
- Događaji tatature
  - `keypress` - pritisak dugmeta tastature

- `keydown` - na spuštanje dugmeta
- `keyup` - na podizanje dugmeta

- Događaji forme
  - `submit` - slanje podataka forme na server
  - `change` - izmena sadržaja polja za unos
  - `focus` - kada kurzor uđe u polje za unos
  - `blur` - kada kurzor napusti polje za unos
- Događaji dokumenta
  - `load` - kada se učita dokument
  - `resize` - promena veličine prozora
  - `scorll` - skrolovanje dokumenta
  - `unload` - kada se dokument promeni

Efekti

Najčešće se koriste efekti prikazivanja i skrivanja elemenata. Neki od često korišćenih efekata su.

- `hide` - skrivanje elementa postavljanjem stila `display: none`.
- `show` - prikazivanje skrivenog elementa uklanjnjem stila `display: none`.
- `toggle` - ukoliko je element skriven, prikazaće se. Ukoliko je prikazan, sakriće se
- `slideUp` - skrivanje uz efekat slajdovanja
- `slideDown` - otkrivanje uz efekat slajdovanja
- `slideToggle` - toggle uz efekat slajdovanja

Efekti u biblioteci jQuery dovode nas pred još jednu veoma važnu osobinu programskog jezika JavaScript. Prilikom izvšavanja hide efekta ima određeno vreme trajanja. Na primer, pozvavši `hide(5000)` nad nekim elementom postavili smo da skrivanje elementa traje 5000 milisekundi, odnosno 5 sekundi. Listingom ispod dat je primer takvog skrivanja elementa.

```
$("p").hide(5000);

alert("hello");
```

Listing - produženo skrivanje elementa

Mogli bismo očekivati da će navedeni kod rezultovati skrivanjem paragrafa u trajanju od 5 sekundi, nakon kog će uslediti poruka `'hello'`. Međutim, kada izvršimo navedeni kod videćemo drugačije ponašanje: poruka `'hello'` će biti odmah prikazana! Zašto?

Izvršavanje JavaScript koda je *neblokirajuće*. To znači da će početi da se izvršava skrivanje paragrafa i odmah nakon toga, pre nego što se završi skrivanje paragrafa, preći na sledeću liniju koda - `alert('hello')`. Ovo svojstvo programskog jezika JavaScript taj jezik čini posebno pogodnim za rad u veb okruženju, u kom bi aplikacije bile praktično neupotrbljive kada bi svaka funkcija morala da se završi da bi se prešlo na izvršavanje sledeće funkcije. Zamislimo samo kako bi izgledalo preuzimanje velikog broja resursa kada bi se za svaki poslao zahtev serveru, sačekalo da pristigne odgovor sa servera i tek tada prešlo na slanje sledećeg zahteva. Mnogo je efikasnije ako možemo istovremeno da pošaljemo sve zahteve, a da ih obrađujemo kako koji pristigne.

Međutim, nekada nam je baš potrebno da sačekamo da se jedna operacija završi da bismo prešli na neku drugu operaciju. Biblioteka jQuery omogućuje takvo izvršavanje kroz mehanizam callback funkcija (potsetimo se da je callback funkcija koja se drugoj funkciji šalje kao argument de bi se izvršila nakon nekog događaja). Već smo koristili taj mehanizam da zadamo programski kod koji će se izvršiti *tek nakon što* se učita dokument, ili *tek nakon što* se klikne na dugme. Taj mehanizam mogli bismo da iskoristimo i da obezbedimo da se alert prikaže *tek nakon što* se element sakrije, kao što je prikazano listingom ispod.

```
$("p").hide(500,function(){

     alert("hello");
});
```

Listing – callback funkcija i odloženo izvršavanje

Vidimo da smo `alert('hello')` „upakovali“ u neimenovani funkcijski izraz koji smo kao callback poslali funkciji `hide` na mestu drugog parametra.

Manipulacija DOM stablom

Osnovna funkcionalnost manipulisanja elementima veb stranice je prezimanje i postavljanje vrednosti i ona je podržana jQuery bibliotekom. Najvažnije funkcije koje su za to namenjene su:

- Funkcije za preuzimanje vrednosti:
  - `text()` - preuzimanje tekstualnog sadržaja elementa,
  - `html()` - preuzimanje html sadržaja elementa,
  - `value()` - preuzumanje vrednosti elementa,
  - `attr(nazivAtributa)` - preuzimanje vrednosti atributa sa zadatim nazivom
- Funkcije za postavljanje vrednosti
  - `text(novaVrednost)` - postavljanje tekstualnog sadržaja elementa,

  - `html(novaVrednost)` - postavljanje html sadržaja elementa,
  - `value(novaVrednost)` - postavljanje vrednosti elementa,
  - `attr(nazivAtributa, novaVrednost)` - postavljanje vrednosti za atribut sa zadatim nazivom

Svaka od navedenih funkcija može da primi callback funkciju kao dodatni parametar. Navedena callback funkcija izvršiće se nakon što se završi postavljanje ili preuzimanje vrednosti.

Dodavanje elemenata u stablo realizuje se pomoću sledećih funkcija:

- `append()` - dodavanje sadržaja na kraj elementa
- `prepend()` - dodavanje sadržaja na početak elementa
- `after()` - dodavanje sadržaja nakon zadatog elementa
- `before()` - dodavanje sadržaja pre zadatog elementa

Uklanjanje elemenata iz stabla ostvaruje se pomoću sledećih funkcija:

- `remove()` - uklanja element sa svim njegovim child elementima
- `empty()` - uklanja sve child elemente za selektovani element

Pomoću biblioteke jQuery moguće je dinamički menjaniti i CSS stilove u aplikaciji. Za to se koriste sledeće funkcije:

- `addClass()` - postavljanje klase za selektovani element
- `removeClass()` - uklanjanje klase za selektovani element
- `toggleClass()` - postavljanje/uklanjanje klase za selektovani element
- `css("propertyname","value")` - postavljanje vrednosti CSS svojstva za selektovani element

Biblioteka takođe ima i adekvatne funkcije za prolazak kroz DOM stablo:

- Penjanje uz stablo
  - `parent()` - direktni prethodnik
  - `parents()` - svi prethodnici
  - `parentsUntil()` - svi prethodnici do prethodnika selektovanog zadatim kriterijumom
- Silazak niz stablo
  - `children()` - svi sledbenici
  - `find()` - svi sledbenici selektovani zadatim kriterijumom

- Horizontalno kretanje kroz stablo
  - `siblings()` - svi čvorovi na istom nivou
  - `next()` - prvi sledeći čvor na istom nivou
  - `nextAll()` - svi sledeći čvorovi na istom nivou
  - `nextUntil()` - svi sledeći čvorovi na istom nivou do čvora selektovanog zadatim kriterijumom
  - `prev()` - prethodni čvor na istom nivou
  - `prevAll()` - svi prethodni čvorovi na istom nivou
  - `revUntil()` - svi prethodni čvorovi na istom nivou do čvora selektovanog zadatim kriterijumom
- Filtriranje stabla
  - `first()` - slektovanje prvog elementa
  - `last()` - slektovanje poslednjeg elementa
  - `eq()` - selektovanje elementa na zadatoj poziciji
  - `filter()` - izdvajanje svih elemenata koji zadovoljavju zadati kriterijum
  - `not()` - izdvajanje svih elemenata koji ne zadovoljavaju zadati kriterijum

AJAX

AJAX označava asinhroni JavaScript i XML i ovaj pristup se često koristi u veb aplikacijama jer omogućuje da se, umesto da se za svaku promenu na stranici učitava cela stranice, učitaju samo podaci koji su potrebni. Pri tome je potrebno omogućiti:

1. Preuzimanje podataka sa back end dela aplikacije bez ponovnog učitavanja stranice, što se postiže pomoću *XMLHttpRequest*
2. Da se podaci pošalju i prime nezavisno od njihovog formatiranja za šta se koristi XML, a još češće JSON format podataka
3. Da se učitani podaci ugrade u strukturu stranice, što se postiže JavaScript manipulacijom DOM stablom
4. Da se, po potrebi, izmene CSS stilovi stranice, što se postiže JavaScript manipulacijom CSS stilovima

Kao što vidimo AJAX nije neka nova tehnologija, već je skup postojećih tehnologija koje se koriste zajedno radi ostvarivanja jedinstvenih ciljeva, pre svega dobrog korisničkog iskustva u radu sa veb aplikacijom.

Biblioteka jQuery omgućuje AJAX pozive pomoću `.ajax` funkcije koja kao parametar prima konfiguracioni objekat. Mogućnosti ove metode su jako velike i čitava specifikacija

dostupna je na linku http://api.jquery.com/jquery.ajax/. Mi ćemo pogledati kako izgleda primer konfiguracionog objekta.

```
$.ajax({

  method: "POST",
  url: "api/articles",
   data: { title: "Novi clanak", description: "Opis", text: "Tekst
novog clanka"}
})
  .done(function( msg ) {
    alert( "Sacuvani su podaci: " + msg );
  });
```

Listing – AJAX poziv

Navedenim konfiguracionim objektom specificirali smo da će biti pozvana POST metoda, da će url biti `"api/articles"` i prosledili smo podatke koji će biti poslati u telu zahteva. Metoda `done` nam omogućuje da prosledimo callback funkciju koja će se izvršiti nakon što pristigne odgovor.

Biblioteka jQuery pojednostavljuje AJAX funkcionalnosti kroz tri funkcije:

- `$(selector).load(URL,data,callback);` - Učitavanje sadržaja sa servera i smeštanje tog sadržaja u selektovani element.
- `$.get(URL,callback);` - zahtev resurs sa servera
- `$.post(URL,data,callback);` - slanje resursa na server