Serverske veb tehnologije
- JQuery -

Dragan Ivanović

Katedra za informatiku, Fakultet Tehničkih Nauka, Novi Sad

2023.

Osnovni pojmovi

- http://jquery.com/
- JavaScript biblioteka
- Omogućava lakše pisanje JavaScript koda bazirano na CSS selektorima
- Obezbeđuje wrapper sintaksu za često korišćene animacije i manipulacije HTML strukturom i CSS stilovima

Instalacija

- Instalira se ukljucivanjem jQuery datoteke u HTML dokument
- Moguće je skinuti lokalnu kopiju ili koristiti neku od dostupnih online verzija (Google ili Microsoft CDN)

```
1 <script src="https://ajax.googleapis.com/
    ajax/libs/jquery/1.11.3/jquery.min.js">
```

Upotreba

- Preporuka je da se dodatni jQuery kod piše u posebnoj JavaScript datoteci
- Sve jQuery datoteke koje se izvršavaju na određenoj HTML stranici, potrebno je takođe uključiti pomoću script taga
- Obavezno NAKON uključivanja same jQuery biblioteke
- Preporuka je da se sav jQuery kod piše u okviru ready događaja HTML dokumenta

```
1 $(document).ready(function(e) {
2     ...
3 });
```

- Example01 - link

Selektori

- JQuery manipulacija HTML elementima se bazira na selektorima i akcijama
- Selektori označavaju skup HTML elemenata na koje se odnosi akcija
- JQuery selektori predstavljaju CSS selektore obuhvaćene znakovima "$("")" – na primer, $("div")
- Akcije predstavljaju konkretne metode manipilacije elementima
- Opšti oblik – $(selektor).akcija();

show()/hide()

- JQuery funkcije *show()* i *hide()* omogućavaju vizuelno sakrivanje i prikazivanje HTML elemenata
- Sakrivenim elementima se postavlja CSS pravilo diplay: none;
- Moguće je definisati brzinu animacije navodeći numeričku vrednost koja predstavlja broj milisekundi ili reči **slow** i **fast**

```
1 $("p").hide("fast");
2 $("h1.red").hide(400);
3 $("#container").hide("slow");
```

toggle()

- *toggle()* funkcija radi sakrivanje ili prikazivanje elemenata u zavisnosti od njihovog trenutnog stanja
- Vidljive elemente sakriva, a sakrivene prikazuje
- Moguće definisanje brzine parametrizovanjem
- Example02 - link

fadeIn()/fadeOut()

- JQuery *fadeIn()* i *fadeOut()* funkcije vrše sakrivanje odnosno prikazivanje elemenata pomoću fade efekta
- U suštini, menja se providnost elementa
- *fadeToggle()* - naizmenično in i out
- *fadeTo()* - menja providnost do zadate vrednosti
- Example03 - link

```
1 $("p").fadeTo(1000, 0.5);
```

slideUp()/slideDown()

- *slideUp()* i *slideDown()* sakrivaju odnosno prikazuju HTML elemente smanjenjem odnosno povećanjem visine
- Ovime se dobija efekat "razmotavanja", ondosno "skupljanja" elementa
- *slideToggle()* radi naizmenično up i down
- Example04 - link

Osnove

- JQuery omogućava reakciju na događaje direktno iz JavaScript koda
- Podržani su handleri na sve događaje koji su inače dostupni u JavaScriptu
- Opšta sintaksa:

```
1  $("selektor").dogadjaj(function(e)
       {
2    // reakcija na dogadjaj
3  });
```

- Parametar funkcije koja se izvršava kada se desi događaj omogućava dobijanje dodatnih informacija o događaju kao i manipulaciju predefinisanim "ponašanjem" u slučaju događaja
- Potpuna lista događaja se može naći na linku - link

Optimizacija

- U okviru reakcije na događaj, moguće je element za koji je vezan događaj selektovati posebnim selektorom: $(this).
- Primer:

```
1  $("#nekiElement").click(function(e
       ) {
2    $(this).hide();
3  });
```

- Ovime se olakšava pisanje koda i optimizuje skripta, s obzirom da se selekcija ne izvršava ponovo za isti element

Događaji miša

- Najčešće korišćeni događaji miša:
  1. *click*
  2. *dblClick*
  3. *mouseenter*
  4. *mouseleave*
- Example05 - link

Događaji miša

- Događaje mouseenter i mouseleave je moguće kombinovati handlerom hover
- Primer:

```
1 $("#nekiElement").hover(function(e) {
2   // Ovo se izvrsava na mouseenter
3 }, function(e) {
4   // A ovo na mouseleave
5 });
```

Događaji tastature

- jQuery obezbeđuje reakcije na sledeće događaje tastature:
  1. *keypress* – događaj pritiska tastera, ne hvata tastere koji se "ne vide" (npr. Shift)
  2. *keydown* – pritisak tastera bez otpuštanja, reaguje na sve tastere
  3. *keyup* – događaj otpuštanja tastera
- Događaji tastature će se izvršiti samo ako je element koji reaguje u fokusu - ako želimo uvek da ih hvatamo, najčešće ih vezujemo za $(document) selektorom.

Događaji tastature

- Nakon hvatanja pritiska na neki od tastera, iz event objekta je moguće dobiti kod tastera koji je pritisnut

```
1 $(document).keydown(function(e) {
2   var code = e.which;
3 });
```

- Na osnovu tog koda, moguće je dobiti karakter po Unicode tabeli:

```
1 var char = String.fromCharCode(e.which
    );
```

- Example06 - link

Događaji na formama

1. *submit* – slanje forme (enter ili klik na submit dugme)
2. *change* – promena value atributa elementa (samo za <input>, <select> i <textarea> elemente)
3. *focus* – dešava se kada element dobije fokus
4. *blur* – dešava se kada element izgubi fokus

Event objekat

- JavaScript Event objekat se prosleđuje funkcijama koje reaguju na događaje
- Sadrži različite informacije vezane za događaj koji se desio
- Više informacija - link

Event objekat

- Korisni atributi i metode:
- *type*: tip (naziv) događaja koji se desio.
- *target*: DOM element nad kojim se desio događaj
- *pageX*, *pageY*: koordinate na kojima se desio događaj
- *which*: taster koji je izazvao događaj
- *preventDefault()*: sprečava dešavanje događaja
- *stopPropagation()*: sprečava propagaciju na roditeljske elemente (*event bubbling*)
- Example07 - link

trigger() funkcija

- Umetno izazivanje događaja nad nekim elementom
- Kao parametar prima ime događaja

```
1 $("#addButton").trigger("click");
```

on() event handler

- Dodeljivanje event handler funkcija elementima se dešava kada se dokument učita
- Elementi kreirani posle toga, neće reagovati na pridružene događaje, iako su obuhvaćeni selektom
- Kako bi ovo izbegli, sve navedene događaje je moguće vezati za elemente pomoću *on()* funkcije

```
1 $(document).on("click", "#link", function(
      e) {
2     ...
3 });
```

Osnove CSS manipulacije

- Jquery omogućava nekoliko metoda za rukovanje CSS stilovima elemenata:
  - *addClass()*: Dodaje navedene CSS klase elementu
  - *removeClass()*: Uklanja navedene CSS klase
  - *css()*: Direktno postavlja navedena CSS pravila

addClass()

- Dodaje CSS klase HTML elementima

```
1 $("#main_heading").addClass("red");
```

- Ukoliko želimo da elementu odjednom pridružimo više CSS klasa, imena klasa se navode razvojena razmakom:

```
1 $("#main_heading").addClass("red
      selected");
```

removeClass()

- Uklanja CSS klase sa HTML elemenata

```
1 $("#main_heading").removeClass("red");
```

- Metode addClass i removeClass se mogu kombinovati pomoću metode toggleClass koja će dodati klase ako ih element ne poseduje, u suprotnom će klase ukloniti sa HTML elementa
- Example08 - link

css()

- Služi za očitavanje ili podešavanje pojedinačnih CSS pravila HTML elementima
- Ukoliko navedemo samo ime CSS atributa, bez vrednosti, funkcija ispisuje trenutnu vrednost tog atributa

```
1 $("div").css("visibility");
```

css()

- Ukoliko navedemo i imena atributa i vrednosti za njih, stil elementa će biti izmenjen u skladu s tim
- Ako želimo da postavimo samo jedno pravilo za HTML element:

```
1 $("#neki_element").css("fontweight", "
    bold");
```

- Sintaksa za više pravila:

```
1 $("#neki_element").css({
2   "color": "#000",
3   "fontsize": "12px"
4 });
```

- Example09 - link

Manipulisanje HTML strukturom

- JQuery poseduje ugrađene funkcije koje omogućavaju laku i brzu manipulaciju DOM strukturom HTML elementa
- Ove manipulacije uključuju:
  - Očitavanje i menjanje različitog sadržaja HTML strukture (tekstualni sadržaj, HTML kod, vrednosti atributa, itd.
  - Dinamičko kreiranje i uklanjanje elemenata
  - Prolazak kroz DOM stablo u proizvoljnom smeru

Očitavanje vrednosti DOM elemenata

- *html()*: vraća HTML sadržaj selektovanog elementa
- *text()*: vraća tekstualni sadržaj selektovanog elementa (bez HTML tagova)
- *attr()*: vraća vrednost navedenog atributa selektovanog elementa
- *val()*: vraća vrednost selektovanog elementa u formi
- *each()*: iterira kroz selektovane elemente

html()

- Ukoliko se pozove bez parametara, funkcija vraća HTML sadržaj selektovanih elemenata

```
1 $("div.mainContainer").html();
```

- Ukoliko se pozove sa parametrom, taj parametar se postavlja kao sadržaj selektovanih elemenata (prethodni sadržaj se prepisuje)

```
1 $("p").html("Ovo je <b>novi</b>
    sadrzaj.");
```

text()

- Ukoliko se pozove bez parametara, funkcija vraća tekstualni sadržaj selektovanih elemenata (samo ono što se vidi na ekranu)

```
1 $("div.mainContainer").text();
```

- Ukoliko se pozove sa parametrom, taj parametar se postavlja kao sadržaj selektovanih elemenata (prethodni sadržaj se prepisuje)

```
1 $("p").text("Ovo je novi sadrzaj.");
```

attr()

- Ukoliko se pozove samo sa imenom atributa kao parametrom, funkcija vraća vrednost tog atributa

```
1 $("img").attr("src");
```

- Ako se pozove sa imenom i vrednošću funkcija postavlja navedeni atribut na navedenu vrednost.
- Ukoliko su selektovani elementi već poseduju navedeni atribut, prošla vrednost se prepisuje novom

```
1 $(".gLink").attr("href", "htttp://www.
    google.com");
```

val()

- Ako se pozove bez parametara, funkcija vraća vrednost samo prvog selektovanog elementa

```
1 $("input").val()
```

- Ako se pozove sa parametrom, postavlja *value* za sve elemente obuhvaćene selektorom

```
1 $("input.req").val("Ovo polje je
      obavezno.");
```

each()

- Funkcija iterira kroz selektovane elemente pristupajući jednom pojedom u petlji
- U svakoj iteraciji, selektor $(this) se odnosi na trenutno selektovani element

```
1 $("h2").each(function() {
2   alert($(this).text());
3 });
```

- Example10 - link

Dinamičko kreiranje elemenata

- HTML elementi se uz pomoć *jQuery* biblioteke mogu kreirati na sledeći način:

```
1 var newH1 = $("<h1></h1>");
```

- Ovo će kreirati novi prazan h1 element
- Isti efekat postižemo i *createElement* funkcijom document objekta:

```
1 var newH1 = $(document.createElement("
    h1"));
```

- Ovako kreiranim elemenima moguće je dodati sadržaj putem text() i html() funkcija ili dodeliti druge dinamički kreirane elemente.

Dinamičko kreiranje elemenata

- Elementi kreirani na neki od prikazanih načina još uvek nisu deo HTML dokumenta
- Dodavanje kreiranih elemenata u neki od postojećih može se izvršiti uz pomoć neke od sledećih funkcija:
  - *append()*: dodaje sadržaj na kraj selektovanih elemenata
  - *prepend()*: dodaje sadržaj na početak selektovanih elemenata
  - *after()*: dodaje sadržaj nakon selektovanih elemenata
  - *before()*: dodaje sadržaj ispred selektovanih elemenata

Dinamičko kreiranje elemenata

- Sve funkcije za dodavanje sadržaja mogu odjednom da dodaju jedan ili više elemenata

```
1 var newHeading = $("<h1>jQuery intro</
    h1>");
2 var newSub = $("<h2>DOM manipulation</
    h2>");
3
4 $("body").append(newHeading, newSub);
```

Dinamičko kreiranje elemenata

- *empty()*: briše sav sadržaj iz selektovanih elemenata (ne i same elemente)

```
1 $("#div.footer").empty();
```

- *remove()*: briše selektovane elemente (i element i sadržaj)

```
1 $("#div.footer").remove();
```

Dinamičko kreiranje elemenata

- Funkcija remove može kao parametar da primi listu selektora koji služe za filtritanje elemenata koji treba da se obrišu
- U tom slučaju brišu se svi elementi obuvaćeni navedenim selektorima unutar selektovanog elementa

```
1 $("body").remove("h1, #mainHeading");
2 $("body").remove("h1, #mainHeading, .
      red");
```

- Example11 - link

Obilazak DOM stabla

- Pronalazak željenih elemenata u odnosu na selektovani element zahteva prolazak (delom) DOM stabla HTML elemenata
- U odnosu na selektovani element, ostali elementi pripadaju jednoj od sledećih kategorija:
  - Preci (Eng. *Ancestors*): Svi elementi kojima element pripada direktno ili indirektno. Direktan predak je roditelj (Eng. *Parent*)
  - Potomci (Eng. *Descendants*): Svi elementi unutar selektovanog. Direktni potomci su deca
  - Elementi na istom hijerarhijskom nivou (Eng. *Siblings*)

Obilazak DOM stabla na gore

- Obilazak elemenata predaka
- *parent()*: vraća element koji je direktan roditelj selektovanom elementu.

```
1 $("p").parent()
```

- *parents()*: vraća sve elemente koji su pretci selektovanog (sve do <html> elementa). Može kao parametar da prima selektor koji služi za filtriranje.

```
1 $("p").parents()
2 $("p").parents("div")
```

Obilazak DOM stabla na dole

- *childern()*: pronalazi i vraća sve elemente koji su direktni potomci selektovanog elementa. Može kao parametar da prima selektor koji služi za filtriranje

```
1 $("table").childern()
2 $("table").childern("tr:visible")
```

- *find()*: pronalazi sve elemente koji odgovaraju prosleđenom selektoru koji su naslednici selektovanog elementa

```
1 $("table").find("a")
```

Obilazak DOM stabla po istom nivou

- Za nalaženje elemenata na istom hijerarhijskom nivou koriste se sledeće funkcije:
  - *siblings()*
  - *next()*
  - *nextAll()*
  - *nextUntil()*
  - *prev()*
  - *prevAll()*
  - *prevUntil()*

Iteracija kroz selektovane elemente

- Za iteraciju kroz elemente obuhvaćene nekom od spomenutih funkcija može se korisiti funkcija *each*

```
1 $("tr").parents().each(function(e) {
2   $(this).css("fontweight", "bold");
3 });
```

- Example12 - link

animate()

- *animate()* funkcija omogućava animiranje dodatnih CSS promena koje nisu pokrivene osnovnim efektima
- Opšta sintaksa: animate(css, brzina)
- css – skup CSS pravila koja se animiraju, zadaju se kao prilikom korišćenja css funkcije
- Brzina: broj koji označava brzinu animacije u milisekundama (mogu se koristiti ključne reči "fast" i "slow")

```
1 $("#neki_element").animate({
2 width: "200px",
3 height: "200px"}, 500);
4 $(".mainTable").animate({
5 marginLeft: "+=20%"
6 }, "slow");
```

- Example13 - link

Kombinovanje efekata

- JavaScript kod se izvršava sekvencijalno, što znači da će izvršavanje da pređe na novu naredbu, bez obzira da li je efekat iz prethodne linije koda završio sa izvršavanjem
- Kako bi obezbedili da sledeći efekat počne tek nakon što prethodni završi, potrebno je animaciju koja sledi napisati posebnoj funkciji kao poslednji parametar prethodne animacije - *callback* funkcije

```
1 $("h1").fadeIn(500, function() {
2   $("h1").animate({fontSize: "2em"},
        200);
3 });
```

- Example14 - link

Zadatak

- Statičku HTML stranicu koja prikazuje listu nastavnika i detaljan prikaz nastavnika učiniti "dinamičkom" upotrebom JQuery-a (podaci o nastavnicima su hardkodirani u JavaScript-u)
- Example15 - link

$.ajax()

- REST servisima se može pristupiti upotrebom ajax() funkcije *jQuery*-ija
- *ajax* funkcija prima kao parametar objekat sa sledećim property-ijima
  - *url* - URL na koji se šalje ajax zahtev (URL RESTful servisa)
  - dataType - tip podatka koji se očekuje kao odgovor
  - cache - da li browser kešira zahtevane podatke (*true* ili *false*)
  - *data* - podaci koji se šalju u zahtevu
  - *type* - vrsta HTTP zahteva (GET, POST, DELETE, PUT)
  - *success* - JavaScript funkcija koja se izvršava ako je uspešno obrađen zahtev na serveru
  - *error* - JavaScript funkcija koja se izvršava ako je došlo do greške u obradi zahteva na serveru
  - kompletna lista - link i w3cschools

Primer

```
1  $.ajax({
2    url: 'http://localhost:3000/professor
       ',
3    dataType: 'json',
4    cache: false,
5    success: function (data) {
6      alert(data);
7    },
8    error: function (jqXHR, textStatus,
       errorThrown) {
9      alert(textStatus);
10   }
11 });
```

Primeri

- Primer rada sa REST servisima: Example16 - link
- Bootstrap-ovan prethodni primer: Example17 - link

Zadatak

- Statičku HTML stranicu koja prikazuje listu nastavnika i detaljan prikaz nastavnika učiniti dinamičkom upotrebom JQuery-a i REST API-ija putem kojeg se dobavljaju podaci iz *Back-end*-a.
- Primer *Back-end*-a je dostupan na adresi: http://minis.uns.ac.rs/filipparag
- API je opisan u dokumentu dostupnom na ovoj adresi: http://minis.uns.ac.rs/pedagoski/rest/DOCS.md