Prezentacija:

jQuery I

Uvod

- Jquery predstavlja javascript biblioteku koja je dizajnirana da olakša klijentsko pisanje HTML-a.
- Jquery je najpopularnija javascript biblioteka, koriste je čak u 60% najpopularnijih websajtova
- Koristi se za navigaciju, selekovanje DOM elemenata, pravljenje animacija i efekata, upravljanje događajima i pravljenje AJAX aplikacija
- Postoje dve verzije jQuery biblioteke:
  - JQuery 1 (1.11.3)
  - JQuery 2 (2.1.4) IE 6-8 nisu podržani

Šta je $?

- jQuery poseduje funkciju koja se koristi pri selekciji elemenata, korišćenjem CSS selektora
  - var stavkeListe = jQuery('li');
  - var stavkeListe = $('li');
- Ako pogledate izvorni kod jQuery biblioteke možete naći liniju:
  - // Expose jQuery to the global object
  - window.jQuery = window.$ = jQuery;

$(document).ready()

- Pre nego što možemo koristiti jQuery, moramo biti sigurni da je stanje stranice spremno za bilo koju DOM manipulaciju

```
$( document ).ready(function() {
  console.log( 'jQuery!' );
});
```

```
$(function() {
  console.log( 'jQuery!' );
});
```

Selektori

- Najjednostavnije stvari koje možemo raditi sa jQuery bibliotekom jeste selekcija elemenata
- Selekcija se radi jednostavnim korišćenjem CSS selektora, kao i dodatnih specijalnih jquery selektora
- Primeri:
  - $('#navigacija'); // selektuje element gde je ID 'navigacija'
  - $('li'); // selektuje sve stavke liste na strani
  - $('ul li'); // selektuje sve stavke neuređene liste na strani
  - $('.osoba'); // selektuje sve elemente sa klasom 'osoba'

Selektori

- O mogućim selektorima možete pročitati na jQuery websajtu : https://api.jquery.com/category/selectors
- $('.myclass')
- $('li:first')
- $('li:odd')
- $('li:has(a)')
- $('a:visible')
- $('[class^="my"]')
- $('a[title$="blog"]')
- $(':text')

- Testiranje selektora:
- http://codylindley.com/jqueryselectors/

Pravljenje novih elemenata

- Pored osnovnog pozivanja selektora $(), možete koristiti istu funkciju za pravljenje jQuery objekata
  - $(' '); // kreira novi element bez sadržaja
  - $(' Zdravo! '); // kreira novi element sa sadržajem
  - $(' Zdravo! '); // kreira novi sa sadržajem i klasom
- Možete inicijalizovati i neke od atributa na sledeći način:
  - $(' ', {
  - html: 'Zdravo!',
  - 'class': 'test'
  - });

Testiranje selekcije

- Nekada trebate izvršiti jQuery kod samo ako selekcija postoji

```
if ( $( '#ne-postojim' ) ) {
  // Netačno! Kod će se izvršiti!
}

if ( $( '#ne-postojim' ).length > 0 ) {
  // Tačno! Kod će se izvršiti ako postoji bar jedan element u selekciji
}

if ( $( '#ne-postojim' ).length ) {
      ….
}


$( 'li' ).eq( 0 ).is( '.special' ); // false
$( 'li' ).eq( 1 ).is( '.special' ); // true
```

Geteri i seteri

- Postoje više metoda koje možete pozivati nakom selekcije i one spadaju u dve kategorije: geteri (getters) i seteri (setters)
- Geteri dohvataju vrednost, dok seteri postavljaju vrednost
- Neki od primera su:
  - `$( 'li' ).html( '<span>Novi HTML</span>' );`
  - `$( 'li' ).html();`
  - `$( 'li' ).text( 'Novi Tekst' );`
  - `$( 'li' ).text();`

Implicitna ineracija

- Pored vrednosti možete proslediti funkciju seter metodi:

```
$( 'li' ).html(function( index, oldHtml ) {
  return oldHtml + '!!!'
});
```

Eksplicitna ineracija

- Nekada nećete moći da iskoristite iteraciju preko osnovnih jQuery metoda, tada ćete koristiti .each() metodu

```
$( 'li' ).each(function( index, elem ) {
  // this: trenutni DOM element
  // index: trenutni indeks elementa u selekciji
  // elem: trenutni DOM element (this)
  $( elem ).prepend( '<b>' + index + ': </b>' );
});
```

Lančanje

- Jedna od najboljih stvari u jQuery je mogućnost “lančanja“ metoda

```
$( 'li' )
  .click(function() {
    $( this ).addClass( 'clicked' );
  })
  .find( 'span' )
    .attr( 'title', 'Hover over me' );
```

Filterisanje selekcije

- Takože, možete filterisati postojeću selekciju na sledeće načine:

```
var elementi = $( 'li' );
// filterisati selekciju elemenata sa samo 'special' klasom
var special = elementi.filter( '.special' );
// filterisati selekciju elemenata bez 'special' klase
var notSpecial = elementi.not( '.special' );
// filterisati selekciju elementa samo onih koji sadrže span element
var hasSpans = elementi.has( 'span' );
```

Pretraga elemenata relativno selekciji

- var stavka = $('li').first(); // dohvatanje prve liste na stravni, postoji i .last()
- var braca = stavka.siblings(); // dohvatanje brace u listi
- var sledeci = stavka.next(); // dohvatanje seledeće stavke, postoji i .prev()
- var lista = stavka.parent(); // dohvatanje roditelja stavke
- var stavkeListe = lista.children(); // dohvatanje dece stavke
- var sveStavke = lista.find('li'); // dohvatanje stavki, uključujući ugnježdene
- var modul = stavka.parents('.module'); // dohvatanje svih predaka stavke sa klasom "module"
- var modul = stavka.closest('.module'); // dohvatanje najbližeg pretka stavke sa klasom "module"
- var dodatno = stavka.add('#lista li'); // dodavanje selekcije na trenutnu selekciju

Vraćanje originalnoj selekciji

- Da bi se vratili originalnoj selekciji koristi se metoda end() :

```
$( '#lista' ) .find('li') .addClass('special') // sada radimo sa stavkama liste
.end()
.addClass('super-special'); // sada radimo sa samom listom
```

- Metoda end() olakšava stvari, ali može dovesti do nepreglednog koda, zato se ne preporučuje njeno intezivno korišćenje.

```
var lista = $( '#lista' );
var stavke = lista.find('li');
stavke.addClass( 'special' );
lista.addClass( 'super-special' );
```

Menjanje atributa

- Postoji mnogo metoda koje se koriste za menjanje atributa, kao na primer:
  - $('li').addClass('hidden');
  - $('li').eq(1).removeClass('hidden');
  - $('li').eq(1).toggleClass('hidden');

Menjanje atributa

- Za menjanje stilova koristi se css() metoda:

```
var lista = $( '#lista' );
var width = Math.floor( lista.width() * 0.1 );

lista.find('li').each(function( index, elem ) {
  $( elem )( 'border',  '1px solid red' );
});
```

Menjanje atributa

- Ukoliko trebate izmeniti više css atributa odjednom koristite sledeći način:

```
$( 'li' ).eq( 1 ).css({
  'font-size': '20px',
  'padding-left': '20px'
});
```

Menjanje atributa

- Za postavljanje vrednosti tekst i select polja u formi koristite val() metodu:

```
$( 'input[type="text"]' ).val("nova vrednost");
$( 'select' ).val( '2' );
```

- Za čekbox polja koristite metodu prop() :

```
$( 'input[type="checkbox"]' ).prop( 'checked', true );
```

Menjanje atributa

- Kada treba izmeniti bilo koji drugi atribut, koristite attr() metodu:

```
$( 'a' ).attr( 'title', 'Click me!' );
```

- Kada postavljate vrednost atributa, kao argument možete proslediti i funkciju:

```
$( 'a' ).attr( 'href', function(index, value) {
  return value + '?special=true';
});
```

Dohvatanje podataka iz elemenata

- Za dohvatanje vrednosti podatka, koristimo geter metode koje smo već pomenuli:

```
var input = $( 'input[type="text"]' );
input.val( 'new value' );
input.val(); // vraća 'new value'
```

- Isto tako, css() metodu možemo koristiti za dohvatanje vrednosti css atrbuta:

```
var listItemColor = $( 'li' ).css( 'color' );
```

- Kada koristimo geter metode, manipuliše se uvek sa prvim elementom selekcije, osim kada je u pitanju text() metoda

Smeštanje elemenata unutar dokumenta

- Za smeštanje elementa unutar dokumenta koristi se više metoda:
  - .appendTo()
  - .append()
  - .insertAfter()
  - .after()

```
$("p").insertAfter("#paragraf");
$("#paragraf").after("p");
```

Kopiranje elemanta

- Možete napraviti kopiju elementa ili više elemenata koristeći metodu clone() :

```
var clones = $( 'li' ).clone();

clones.html(function( index, oldHtml ) {
  return oldHtml + '!!!';
});

$( '#lista' ).append( clones );
```

- Jquery neće dozvoliti da klonirate element sa ID-jem, zato je potrebno ukloniti id atribut pre kloniranja ukoliko postoji

Uklanjanje elemanta

- Za uklanjanje elementa se koristi više metoda:
  - .remove()
  - .detach()
  - .replaceWith()

- Za trajno uklanjanje elementa se koristi remove() metoda:

```
$( '#lista li' ).first().remove();
```

- Za privremeno uklanjanje elementa se koristi detach() metoda:

```
$( '#lista li' ).first().detach();
```

Uklanjanje elemanta

- Za menjanje elementa ili elemenata sa drugim elementom ili HTML-om koristi se .replaceWith() metoda:

- var replaced = $('#lista li')
  .first().replaceWith('
novo!
');

Događaji (Events)

- Neki od osnovnih događaja su:
  - .change()
  - .click()
  - .keydown()
  - .keypress()
  - .keyup()
  - .mouseover()
  - .mouseout()
  - .mouseenter()
  - .mouseleave()
  - .scroll()
  - .focus()
  - .blur()
  - .resize()

Događaji (Events)

- jQuery znatno olakšava rad sa događajima:

```
$( 'li' ).click(function( event ) {
  console.log( 'clicked', $( this ).text() );
});
```

- Takođe možete koristiti .on() metodu:

```
$( 'li' ).on( 'click', function( event ) {
  console.log( 'clicked', $( this ).text() );
});
```

- Za pokretanje (triggerovanje) nekog događaja programski koristite trigger() metodu:

```
$( 'li' ).trigger( 'click' );
$( 'li' ).click();
```

- Za uklanjanje događaja koristite off() metodu:
- $('li').off('click');

Namespaced Events

- Metoda “on“ dozvoljava kreiranje vaših spostvenih događaja

```
$( 'li' ).on( 'click.test', function() {
  console.log( 'stavka je kliknuta!' );
});

$( 'li' ).trigger( 'click.test' );
$( 'li' ).off( 'click.test' );
```

Povezivanje događaja

- Korišćenjem metode on() možete bind-ovati (povezati) više događaja istovremeno:

```
$( 'input[type="text"]' ).on('focus blur', function() {
  console.log( 'Korisnik je ušao ili izašao iz input polja' );
});
```

Prosleđivanje funkcije za upravljanje događajem

- Prosleđivanje funkcije za upravljanje događajem se radi na sledeći način:

```
var handleClick = function() {
  console.log( 'Nešto je kliknuto!' );
};

$( 'li' ).on( 'click', handleClick );
$( 'h1' ).on( 'click', handleClick );
```

Event objekat

- Bilo kada se izvrši neki događaj, event funkcija sadrži event argument
- Ovaj objekat sadrži dosta važnih atributa, koje su nam često potrebni I koje možemo iskoristiti unutar funkcije događaja

```
$( document ).on( 'click', function( event ) {
  console.log( event.type );    // tip događaja, npr. "click"
  console.log( event.which );   // šta je pritisnuto na tastaturi ili mišu
  console.log( event.target );  // originalni element
  console.log( event.pageX );   // pozicija miša po X osi
  console.log( event.pageY );   // pozicija miša po Y osi
});
```

Unutar funkcije događaja

- Kada se definiše event funkcija koja se koristi pri nekom događaju, funkcija dobija pristup “sirovom“ DOM elementu, kom možemo pristupiti sa this
- Ako želitmo raditi neke izmene this elementa, moramo ga obaviti sa $( )

```
$( 'input' ).on( 'keydown', function( event ) {
  // this: element koji je povezan sa događajem
  // event: event objekat

  // Promeni boju pozadine elementa na crvenu ako je ENTER pritisnut,
  // u suprotnom na zelenu boju
  $( this ).css( 'background', event.which === 13 ? 'red' : 'green' );
});
```

Preventing default action

- Često ćete želeti da unutar nekog događaja, sprečite podrazumevanu akciju

```
$( 'a' ).on( 'click', function( event ) {
  // Sprečite podrazumevanu akciju.
  event.preventDefault();
  // Logovanje...
  console.log( 'Klikunt link!' );
});
```

Event bubbling

- Zamislite sledeći kod:

```
<div id="div1">element 1<div id="div2">element 2</div></div>

$("#div1").click(function (event) {
    alert("kliknuli ste na div 1");
});

$("#div2").click(function (event) {
    alert("kliknuli ste na div 2");
});
```

- Korišćenjem metode stopPropagation() event objekta možemo prekinuti efekat koji se javlja

Event delegation

- Šta će se desiti ako se nekad izmeni element koji sadrži druge elemente koji imaju povezani događaj (event) ? Da li će događaj biti povezan?

```
$( '#lista li' ).on( 'click', function( event ) {
  console.log( this ); //loguje koja stavka je kliknuta
});
$( '<li>nova stavka!</li>' ).appendTo( '#lista' );
```

- Rešenje ovog problema:

```
$( '#lista' ).on( 'click', 'li', function( event ) {
  console.log( this ); //loguje koja stavka je kliknuta
});

$( '<li>nova stavka!</li>' ).appendTo( '#lista' );
```

PITANJA?