Prezentacija:

jQuery II

Efekti

- Jquery omogućava da se jednostavno prave animacije i efekti, korišćenjem već postojećih funkcija:
  - show() - prikazuje selekotvani element
  - hide() - sakriva selekotvani element
  - fadeIn() - animira transparentnost selektovanog elementa do 100%
  - fadeOut() - animira transparentnost selektovanog elementa do 0%
  - SlideDown() - prikazuje animaciju selektovanog elementa sa vertikalnom “slide“ animacijom
  - slideUp() - prikriva animaciju selektovanog elementa sa vertikalnom “slide“ animacijom

- $('.hidden').show();

Efekti

- Pored osnovnog pozivanja funkcija, postoje i “toggle“ funkcije koje se ponašaju kao “prekidači“ :
  - toggle()
  - fadeToggle()
  - slideToggle()

- $('.image').toggleFade();

Efekti

- Da bi postavili vremensko trajanje efekta, prosledite kao parametar funkciji broj u milisekundama ili već predefinisane konstante:
  - fast (200)
  - slow (600)
  - _default (400) → podrazumevano
- $('.block').fadeOut(2000);
- $('.block').fadeOut('slow');
- Ove konstante je moguće promeniti ili dodati nove na sledeći način:
  - jQuery.fx.speeds.fast = 50;
  - jQuery.fx.speeds.slower = 2000;
- $('.block').fadeOut('slower');

Efekti

- Često trebate da odradite neku akciju, tek kada se animacija završi, za taj slučaj koristite funkciju, kao drugi parametar:

- ```
  $( 'p.old' ).fadeOut( 300, function() {
    $( this ).remove();
  });
  ```
  ```

Efekti

- Ako postojeći efekti nisu dovoljni da zadovolje vaše potrebe, onda koristite animate() funkciju kako bi postigli razne efekte modifikujući css atribute:
-
```
$( 'p' ).animate({
    opacity: 0.25,
    fontSize: '12px'
  }, 300, function() {
    // izvršava kod nakon završene animacije
  }
);
```
- Prilikom poziva animate() funkcije, šalju se 3 moguća parametra:
  - objekat kohji definiše atribute za animaciju
  - dužina trajanja u milisekundama
  - callback funkcija koja se izvšava nakon završetka animacije
- Ukoliko je potrebo raditi animacije sa bojama, neophodno je uključiti: https://github.com/jquery/jquery-color/

Efekti

- Postoje dve važne metode za upravljanje animacijama:
  - stop() - prekidanje animacije
  - delay(milisec) → odlaganje animacije u milisekundama

- $('.animacija-u-toku').stop();
- $('p.demo').slideUp('slow').dalay(1000).slideDown('slow');

- Više o efektima: http://api.jquery.com/category/effects/

AJAX

- **AJAX** - "Asynchronous JavaScript and XML"
- Predstavlja način učitavanja podataka sa servera bez potrebe za ponovnim učitavanjem stranice
- Za to se koristi brauzerksa funkcionalnost **XMLHttpRequest (XHR)**
- Postoje nekoliko funkcija koje se koriste u ovu svrhu:
  - $.ajax()
  - $.get()
  - $.post()
  - $.getJSON()

$.ajax

```
// Napravite "callback" funkciju koja ce biti izvrsena na uspesan AJAX zahtev
var updatePage = function( resp ) {
  $( '#target').html( resp.people[0].name );
};
// ...kada AJAX zahtev ne uspe / se ne izvrsi
var printError = function( req, status, err ) {
  console.log( 'something went wrong', status, err );
};
// napravite objekat koji definise AJAX zahtev
var ajaxOptions = {
  url: '/data/people.json',
  dataType: 'json',
  success: updatePage,
  error: printError
};
// pokrenite AJAX zahtev
$.ajax(ajaxOptions);
```

$.ajax

- Drugi način pisanja ajax zahteva:

```
$.ajax( '/data/people.json', {
  type: 'GET',
  dataType: 'json',
  success: function( resp ) {
    console.log( resp.people );
  },
  error: function( req, status, err ) {
    console.log( 'something went wrong', status, err );
  }
});
```

XML je danas JSON

- **AJAX** je termin koji je stvoren 2005, opisivao komunikaciju sa serverom bez osvezavanja stranice
- Tada se na serverima koristio XML kao format za komunikaciju, medjutim danas, u svim modernim aplikacijama koristi se JSON
- JSON je stringovska reprezentacija podatka (objekta)

- Za parsiranje i pravljenje JSON-a koristite sledeće funkcije:
  - JSON.stringify()
  - JSON.parse()

JSON

```
{ "people" : [
  {
    "name" : "Ben",
    "url" : "http://benalman.com/",
    "bio" : "I create groovy websites, useful jQuery plugins, and play a mean funk bass. "
  },
  {
    "name" : "Jory",
    "url" : "http://joryburson.com",
    "bio" : "super-enthusiastic about open web education @bocoup"
  }
] }
```

Skraćeni ajax pozivi

- Ako pravite jednostavne zahteve, ili ako ne želimo da ispratimo sve moguće callback funkcije, možemo koristiti sledeće skraćene verzije:

```
$.get( '/podaci/strana.html', function( html ){
  $( '#sadrzaj' ).html( html );
});
```

```
$.post( '/contact/send',{email:“demo@gmail.com“,message: “Zdravo!“}, function( resp ) {
  console.log( resp );
});
```

Slanje podataka i rad sa formama

- Možemo slati naše podatke ajax zahtevom, postavljajući data parametar ajax konfiguracionog objekta, ili kao drugi argument u “skraćenim“ ajax funkcijama
- Ako se šalje GET zahtev, data parametri će biti dodati na kraj URL-a

  demo.php?field1name=field1value&field2name=field2value...

- Ako se šalje POST zahtev, data parametri će biti poslati kao form data podaci

Slanje podataka i rad sa formama

```
$( 'form' ).submit(function( event ) {
  event.preventDefault();
  var form = $( this );
  $.ajax({
    type: 'POST',
    url: '/data/save',
    data: form.serialize(),
    dataType: 'json',
    success: function( resp ) {
      console.log( resp );
    }
  });
});
```

jqXHR

- $.ajax() i ostale ajax funkcije uvek vraćaju **jqXHR** objekat **jQuery XML HTTP Request**
- Ovaj objekat sadrži mnogo moćnih metoda koje možemo koristiti, ali ga prethodno moramo smestiti u promenjivu kako bi ga koristili

```
var req = $.ajax({
  url: '/data/people.json',
  dataType: 'json'
});
```

- Korišćenjem metode abort() možemo prekinuti ajax zahtev

```
req.abort();
```

jqXHR

- Može se koristiti za callback funkcije zahteva, čak i kada se ajax zahtev završi
- then() metoda zahteva dva parametra, od kojih je samo prvi obavezan (success,error) I može se pozivati više puta

```
var success = function( resp ) {
  $( '#target' ).append(
    '<p>people: ' + resp.people.length + '</p>'
  );
};
var err = function( req, status, err ) {
  $( '#target' ).append( '<p>something went wrong</p>' );
};
req.then( success, err );
req.then(function() {
  $( '#target' ).append( '<p>it worked</p>' );
});
```

jqXHR

- Ako ne želimo da koristimo error i success callbakfunkcije istovremeno, možemo koristiti done() i fail() metodu
  - req.done(success);
  - req.fail(err);

- Ako želimo da napravimo callback funkciju koja se izvršava nezavisno da li je ajax zahtev uspeo ili ne, koristimo allways() metodu

```
req.always(function() {
  $( '#target' )
    .append( '<p>one way or another, it is done now</p>' );
});
```

JSONP

- Mnogi javaskript programeri budu iznenađeni kada prvi put koriste $.ajax() da bi dohvatili podatke sa drugog domena i njihov zahtev se nikad ne izvrši uspešno
- Zbog sigurnosnih razloga, ovo ponašanje je uključeno unutar samog brauzera i nije ga moguće modifikovati
- Da bi rešili ovaj problem, koristi se JSONP kao format
- JSONP se izvršava tako što se uključuje untar script taga

JSONP

```
callback({
        "sites":
        [
            {
                "siteName": "JQUERY4U",
                "domainName": "http://www.jquery4u.com",
                "description": "#1 jQuery Blog for your Daily News, Plugins, Tuts/Tips &amp;amp; Code Snippets."
            },
            {
                "siteName": "BLOGOOLA",
                "domainName": "http://www.blogoola.com",
                "description": "Expose your blog to millions and increase your audience."
            },
            {
                "siteName": "PHPSCRIPTS4U",
                "domainName": "http://www.phpscripts4u.com",
                "description": "The Blog of Enthusiastic PHP Scripters"
            }
        ]
    });
```

JSONP

- Da bi specificirali JSONP zahtev, možemo to uraditi na sledeći način:

```
$.ajax({
  url: '/data/search.jsonp',
  data: { q: 'a' },
  dataType: 'jsonp',
  success: function( resp ) {
    $( '#target' ).html( 'Results: ' + resp.results.length );
  }
});
```

- Moguće je napraviti JSONP zahtev pozivom getJSON metode, ali url mora da poseduje na kraju url-a callback=? ili slično

```
$.getJSON( '/data/search.jsonp?q=a&callback=?',
  function( resp ) {
    $( '#target' ).html( 'Results: ' + resp.results.length );
  }
);
```

PITANJA?