Prezentacija:

Backbone.js I

BACKBONE.JS

Sadržaj

- Uvod
- Modeli (Models)
  - Validacija
  - Nasledjivanje
  - …
- Kolekcije (Colections)
- Pogledi (View)

Uvod

- Backbone.js predstavlja JavaScript biblioteku sa REST JSON interfejsom i bazirana je na “model–view–presenter“ (MVP) aplikacijskom konceptu.
- Backbone je poznat po tome što je lagan i nezahtevan, zato što zahteva samo hjoš jednu JavaScript biblioteku, Underscore.js.
- Dizajniran je za kreiranje SPA (single-page applications), a kako bi imao različite podatke u web aplikaciji sinhronizovane sa serverom.
- Backbone.js je kreiran od strane **Jeremy Ashkenas**-a, koji je takodje poznat po tome što je kreirao **CoffeeScript**.

- Prva verzija: 13. Oktobar 2010.
- Poslednja verzija: 1.2.1 / 4. Jun 2015.
- Veličina: 7.3 KB

Uvod

- Airbnb
- Diaspora
- Digg
- Foursquare
- Grooveshark
- Hulu
- LinkedIn Mobile
- Pandora Radio
- Pinterest
- Sony Entertainment Network
- Soundcloud
- USA Today.com
- WordPress.com

Modeli / Models

- Modeli predstavljaju srce Javascript aplikacije, sadrže podatke kao I logiku koja ih okružuje: konverzije, validaciju, atribute,...

```
var Song = Backbone.Model.extend({
    initialize: function(){
        console.log(“A new song has been created“);
    }
});

var song = new Song();
```

Modeli / Models

```
var person = {};
person.name = “Marko“;
```

```
var Person = Backbone.Model.extend();
var person = new Person();
// za postavljanje vrednosti nekog atributa
person.set(“name“,“Marko);
```

Modeli / Models

```
var person = new Person();

person.set({
    name: “Marko“,
    age: 29
});


var person = new Person({
    name: “Marko“,
    age: 29
});
```

Modeli / Models

- Za dohvatanje odredjenog atributa iz modela:
  person.get(“name“);

- Za uklanjanje odredjenog atributa iz modela:
  person.unset(“title“);

- Za uklanjanje svih atributa iz modela:
  person.clear();

- Za proveru postojanosti nekog atributa:
  person.has(“name“);

Modeli / Models

- Za postavljanje podrazumevanih vrednosti u modelu:

```
var Song = Backbone.Model.extend({
    defaults: {
        genre: “Jazz“
    }
});
```

Modeli / Models

- Za validaciju modela, koristimo validate callback metodu:

```
var Song = Backbone.Model.extend({
    validate: function(attrs) {
        if(!attrs.title)
            return “Title is required“;
    }
});

var song = new Song();
console.log(song.isValid());

console.log(song.validationError);
```

Modeli / Models

- Nasledjivanje modela se radi na sledeci nacin:

```
var Animal = Backbone.Model.extend({
    walk: function() {
    console.log(“Animal is walking...“);
    }
});

var Dog = Animal.extend();

var dog = new Dog();
dog.walk();
```

Modeli / Models

```
var Animal = Backbone.Model.extend({
    walk: function() {
        console.log(“Animal is walking...“);
    }
});

var Dog = Animal.extend({
    walk: function(){
        alert(“Dog is walking“);
    }
});

var dog = new Dog();
dog.walk();
```

Modeli / Models

```
var Animal = Backbone.Model.extend({
    walk: function() {
            console.log(“Animal is walking...“);
    }
});

var Dog = Animal.extend({
    walk: function(){
            Animal.prototype.walk.apply(this);
            alert(“Dog is walking“);
    }
});

var dog = new Dog();
dog.walk();
```

Kolekcije / Collections

- Kolekcije pretstavljaju niz tj. Grupu inicijalizovanih modela:

```
var Song = Backbone.Model.extend();

var Songs = backbone.Collection.extend({
    model: Song
});

var songs = new Songs([
    new Song({title:“Title 1“}),
    new Song({title:“Title 2“}),
    new Song({title:“Title 3“}),
]);
```

Kolekcije / Collections

- Dodavanje pojedinacnih modela:
  songs.add(new Song({title:“Title 4“}));

- Dohvatanje modela na osnovu indeksa:
  songs.at(0);

- Dohvatanje modela na osnovu id/cid:
  songs.get(“c1“);

- Brisanje modela iz kolekcije:
  songs.remove(songs.at(0));
  songs.remove(songs.get(“c1“));

Pogledi / View

- Pogledi se koriste za:
  - Renderovanje sadržaja
  - Odgovaranje na određene DOM događaje (clicks, d&d,...)
  - Ponašaju se kao kontroleri (controllers) u MVC paradigmi
  - Poseduju svoj DOM element

Pogledi / View

```
var SongView = Backbone.extend.View({
    render: function(){
        this.$el.html(“Zdravo svima!“);
        return this;
    }
});

var songView = SongView({el:“#container“});
songView.render();
```

Pogledi / View

- Svaki put kada se kreira view objekat, on se kreira u memoriji, čak i kada nije direktno povezan sa nekim DOM elementom:

```
var songView = SongView();
songView.render();
```

- Dohvatanje sadržaja view objekta:
  $('#demo').html(songVIew.$el);

Pogledi / View

- Modifikovanje podrazumevanog div taga, kao I njegovih atributa se radi na sledeći način:

```
var SongView = Backbone.extend.View({
    tagName: “span“,
    className: “song“,
    id: “demo“,
    attributes: {
        “data-genre“: “Jazz“
    },
    render: function(){
            this.$el.html(“Zdravo svima!“);
        return this;
    }
});

var songView = SongView({el:“#container“});
songView.render();
```

Pogledi / View

- Također kod ispod možemo dodatno skratiti, s obzirom da možemo koristiti lančanje:

```
var songView = SongView();
songView.render();
$('#demo').html(songVIew.$el);
```

```
var songView = SongView();
$('#demo').html(songVIew.render().$el);
```

Pogledi / View

- Integracija modela sa pogledima se radi na sledeci nacin:

```
var SongView = Backbone.extend.View({
    render: function(){
        this.$el.html(this.model.get(“title“));
        return this;
    }
});

var song = new Song({title:“Title 1“});
var songView = SongView({el:“#container“,model:song});
songView.render();
```

Pogledi / View

- Integracija kolekcija sa pogledima se radi na sledeci nacin:

```
var SongsView = Backbone.extend.View({
    render: function(){
        var self = this;
        this.model.each(function(song){
            var songView = new SongView({model:song});
            self.$el.append(songView.render().$el);
        });
    }
});

var songsView = SongsView({el:“#container“,model:songs});
SongsView.render();
```

PITANJA?