Technische Universität München

Object Oriented Javascript

Javascript Technology Seminar 2014

Dario Banfi

Why do we need OO Javascript?

- Encapsulation
- Abstraction
- Polymorphism
- Modularity
- Inheritance

Is Javascript truly Object-Oriented?

Specification document for ECMAScript:

“an object-oriented programming language for performing computations and manipulating computational objects within a host environment.”

Kinds of Object-Oriented Paradigms

- Classical (or class-based) object-oriented languages

- Prototypal (or prototype-based) object-oriented languages

Javascript is prototype-based

In Javascript an Object is an aggregate of **key-value pairs**

The property name is a string and the property value can be any data type (including functions and other objects)

Declaring Objects

Using functions:

```
function Hostel(name) {
    this.name = name;
    this.rooms = [];
    this.getTotalPrice = function() {
        return this.rooms.length * 10;
    };
}
```

Every time a new Hostel Object is created, the method getTotalPrice() is recreated

Solution: declaring a prototype

Every function owns a prototype object from which other objects inherit properties

```
function Hostel(name) {
    this.name = name;
    this.rooms = [];
}

Hostel.prototype.getTotalPrice = function(){
    return this.rooms.length * 10;
};
```

Declaring Objects: continued

Using object literals:

```
var Hostel =  {
   rooms : [],
   getTotalPrice : function() {
        return this.rooms.length * 10;
    }
}
```

Singleton using a Function:

```
var Hostel = new function() {
    this.rooms = [];
    this.getTotalPrice = function () {
        return this.rooms.length * 10;
    };
}
```

Inheritance: prototypal inheritance

We set the prototype object as the parent class, so that the parent method will be called if not overridden

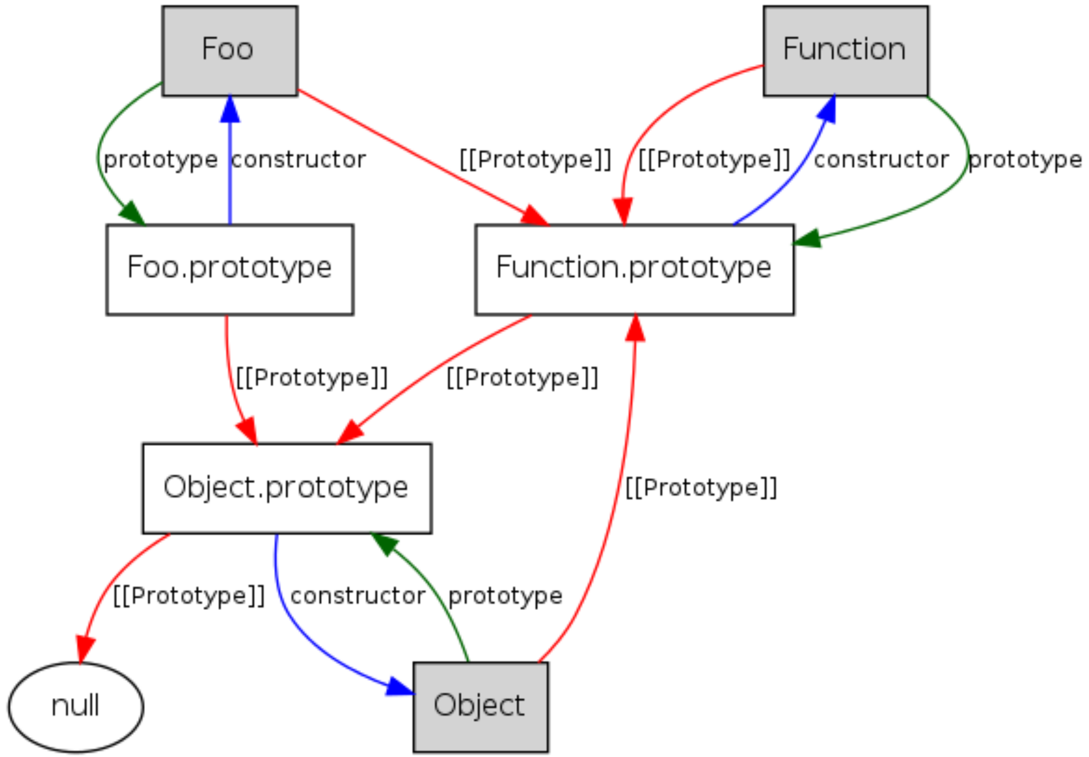
```
var room= { bed: true}

var luxury_room = { pool_access: true }

luxury_room.__proto__ = room

luxury_room.bed // true
```

Prototypal inheritance chain



Webapp - Hostel Manager

Reason: Manage occupancy of the rooms and assign client profiles to them

Webapp - Hostel Manager

- Configurable and reusable
- Simple interface and fast by being a **single-page webapp**
- Handling of local reservations and availability

FUTURE FEATURES: Connecting to existing reservation webservices

Frameworks considered

ease.js: “A classical Object-Oriented framework for JavaScript, intended to eliminate boilerplate code and “ease” the transition into JavaScript from other Object-Oriented languages”

ease.js

```
var Class = easejs.Class;
var Stack = Class( 'Stack',
{
    'private _stack': [],

    'public push': function( value ) {
        this._stack.push( value );
    },

    'public pop': function() {
        return this._stack.pop();
    },
} );
```

+

Classic object oriented approach
Interfaces
Classic inheritance
Abstract methods and classes
Access modifiers

–

Unnecessary and redundant for the size of my project

Libraries used

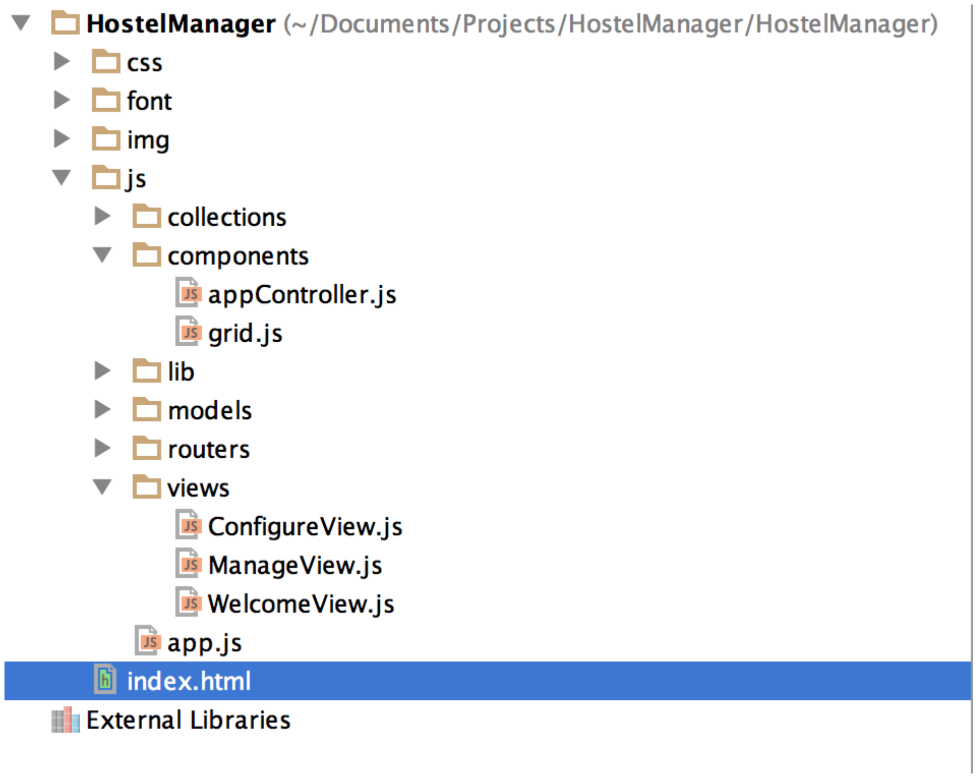
● jQuery	
● Bootstrap	
● Gridster	(Dynamic grid widget)
● Backbone.js	(MVP Javascript lib)
● Underscore	(Templating)
● Require.js	(Module loader)
● toastr	(Toast notifications)

DEMO TIME

General structure

The structure follows the Backbone.js convention:

Divided in views, models, collections and routers



```
▼ HostelManager (~/Documents/Projects/HostelManager/HostelManager)
  ▶ css
  ▶ font
  ▶ img
  ▼ js
    ▶ collections
    ▼ components
        appController.js
        grid.js
    ▶ lib
    ▶ models
    ▶ routers
    ▼ views
        ConfigureView.js
        ManageView.js
        WelcomeView.js
      app.js
    index.html
  External Libraries
```

Backbone.js

Javascript library with RESTful interface and based on MVP (Model View Presenter) paradigm.

★ “Competitors” : Angular.js, Ember.js

➔ Chosen over the others because of the relative simplicity and smaller learning curve

Backbone killer feature: REST integration

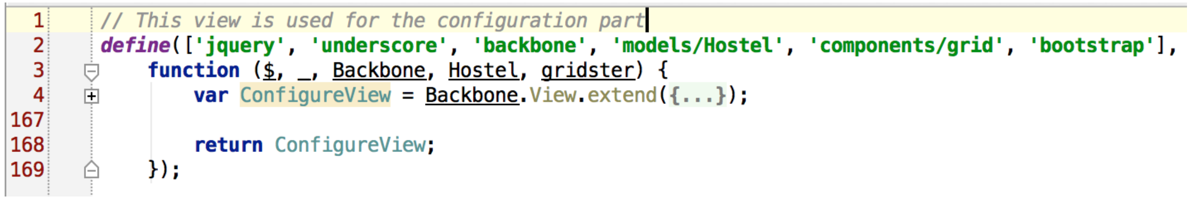
Backbone.js provides easy tools to keep your model in sync with a RESTful service

```
var Book = Backbone.Model.extend({
    defaults: {
        ID: "",
        BookName: ""
    },
    idAttribute: "ID",
    initialize: function () {
        });
    },
    urlRoot: 'http://localhost:8080/api/Books'
});
```

```
// Create operation
var book = new Book({ BookName: "Backbone Book 43" });
book.save({}, {
    success: function (model, respose, options) {
        console.log("The model has been saved to the
server");
    },
    error: function (model, xhr, options) {
        console.log("Something went wrong while saving
the model");
    }
});
```

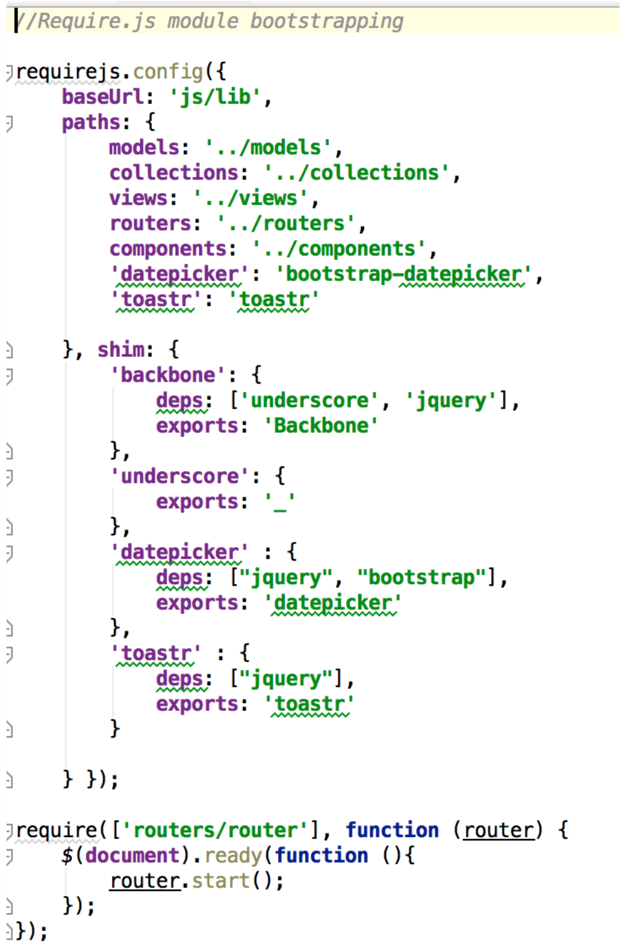
Require.js module loader

Require.js lets you split your code in modules which are loaded when needed.



```
// This view is used for the configuration part
define(['jquery', 'underscore', 'backbone', 'models/Hostel', 'components/grid', 'bootstrap'],
    function ($, _, Backbone, Hostel, gridster) {
        var ConfigureView = Backbone.View.extend({...});

        return ConfigureView;
    });
```



```
//Require.js module bootstrapping

requirejs.config({
    baseUrl: 'js/lib',
    paths: {
        models: '../models',
        collections: '../collections',
        views: '../views',
        routers: '../routers',
        components: '../components',
        'datepicker': 'bootstrap-datepicker',
        'toastr': 'toastr'

    }, shim: {
        'backbone': {
            deps: ['underscore', 'jquery'],
            exports: 'Backbone'
        },
        'underscore': {
            exports: '_'
        },
        'datepicker' : {
            deps: ["jquery", "bootstrap"],
            exports: 'datepicker'
        },
        'toastr' : {
            deps: ["jquery"],
            exports: 'toastr'
        }

    } });

require(['routers/router'], function (router) {
    $(document).ready(function (){
        router.start();
    });
});
```

Backbone.js + Require.js + Underscore.js

- - Learning curve
- - Libraries size (27 KB in total)

- + Code is more organized and well structured
- + They provide scalability
- + They make coding single-page apps much easier
- + Methods to interface with a REST backend

Code Highlights 1

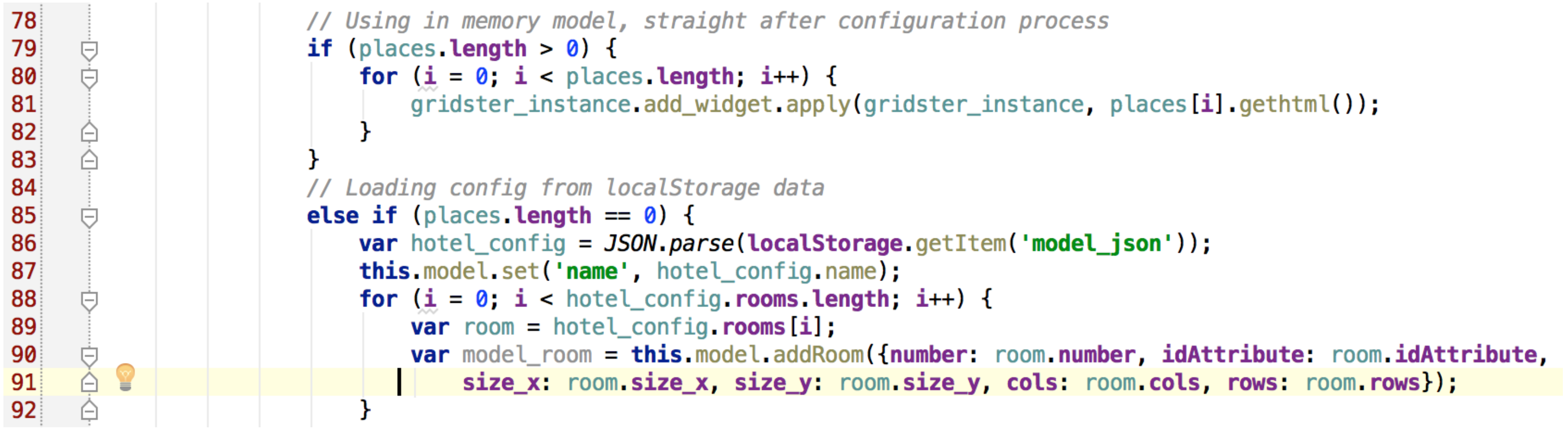
➔ No backend integration, purely javascript

The app configuration and state is kept through **HTML5 localStorage**: 5 MB of client-stored data which can be used as easily as:

```
localStorage.setItem('key', value);
localStorage.getItem('key');
```

Code Highlights 1

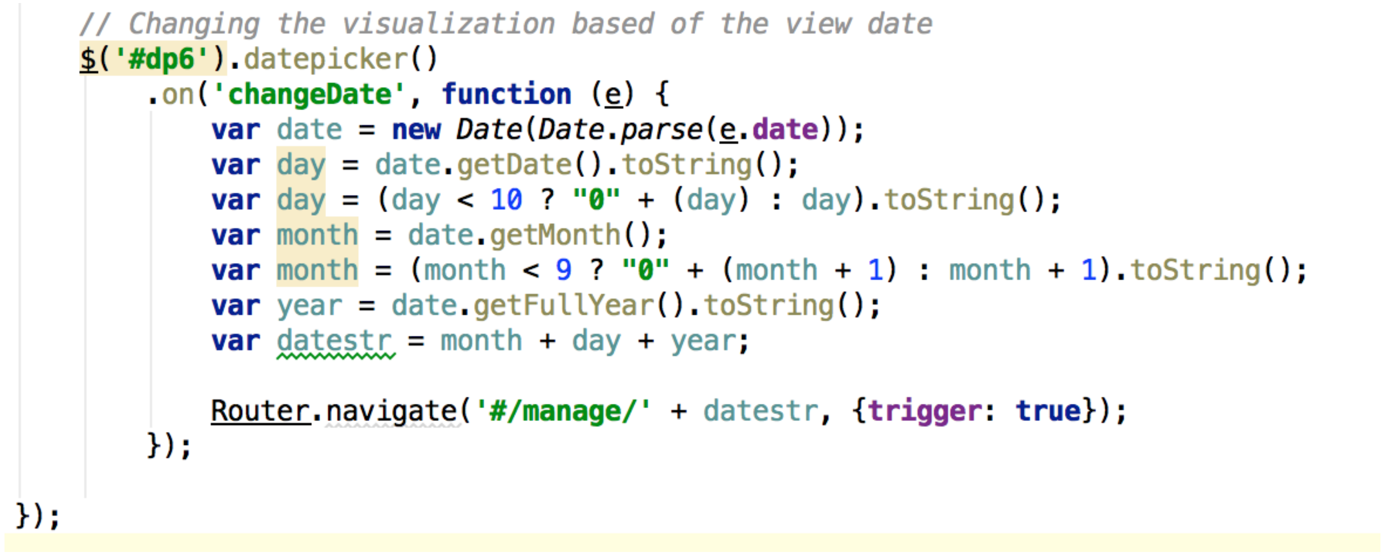
The room configuration is rebuild from a locally stored JSON model if the page is closed



```
// Using in memory model, straight after configuration process
if (places.length > 0) {
    for (i = 0; i < places.length; i++) {
        gridster_instance.add_widget.apply(gridster_instance, places[i].gethtml());
    }
}
// Loading config from localStorage data
else if (places.length == 0) {
    var hotel_config = JSON.parse(localStorage.getItem('model_json'));
    this.model.set('name', hotel_config.name);
    for (i = 0; i < hotel_config.rooms.length; i++) {
        var room = hotel_config.rooms[i];
        var model_room = this.model.addRoom({number: room.number, idAttribute: room.idAttribute,
            size_x: room.size_x, size_y: room.size_y, cols: room.cols, rows: room.rows});
    }
```

Code Highlights 2

Bootstrap-datepicker (based on jQuery datepicker) callbacks handling:



```
    // Changing the visualization based of the view date
    $('#dp6').datepicker()
        .on('changeDate', function (e) {
            var date = new Date(Date.parse(e.date));
            var day = date.getDate().toString();
            var day = (day < 10 ? "0" + (day) : day).toString();
            var month = date.getMonth();
            var month = (month < 9 ? "0" + (month + 1) : month + 1).toString();
            var year = date.getFullYear().toString();
            var datestr = month + day + year;

            Router.navigate('#/manage/' + datestr, {trigger: true});
        });

});
```

Conclusion

- The web app is covers the needed functionality of **local occupancy management**
- Javascript makes it more portable and easily accessible to mobile devices (mobile browser or phonegap)
- Possibility to complete it with external web services support and being sold as standalone product

Information

https://github.com/Ambigioz/HostelManager

dario.banfi@tum.de

Thanks for the attention

Questions?