by @littleiffel for @JSLuxembourg (http://www.json.lu)
April, 2012

Introduction to Backbone.js

about me

unifr.ch,liip.ch, eirenesuisse.ch, oashi.com, www.williambrownstreet.net

universitas friburgensis

2000-2008 Computer Science

L//P

2006-2008 ActionScript, Red5 Media Server

EIRENE Suisse

2008-2010 Volunteer

OAshi S.à r.l.

since 2010 Developer

Backbone.js

- **MVC Framework**

  “The goal of MVC is to simplify the architecture by decoupling models and views, and to make source code more flexible and maintainable.”

  http://www.wikipedia.org

- **Dependencies**

  underscore.js, jQuery/Zepto, json2.js

- **Lightweight**

  Only (5,6kb)

- **MIT software license**
- **Get Backbone and documentation**

  http://documentcloud.github.com/backbone

classical MVC

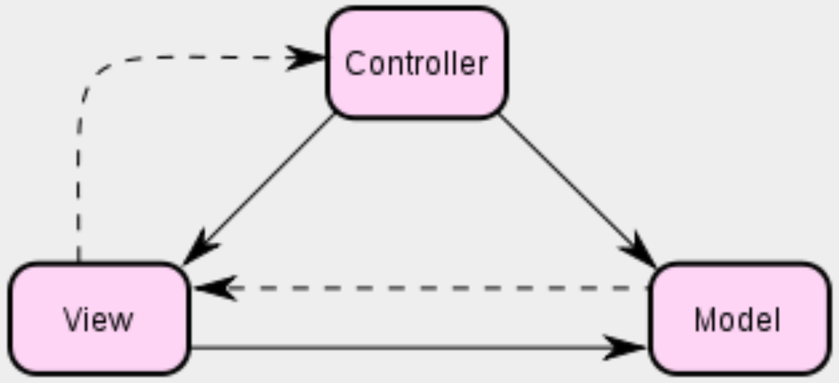
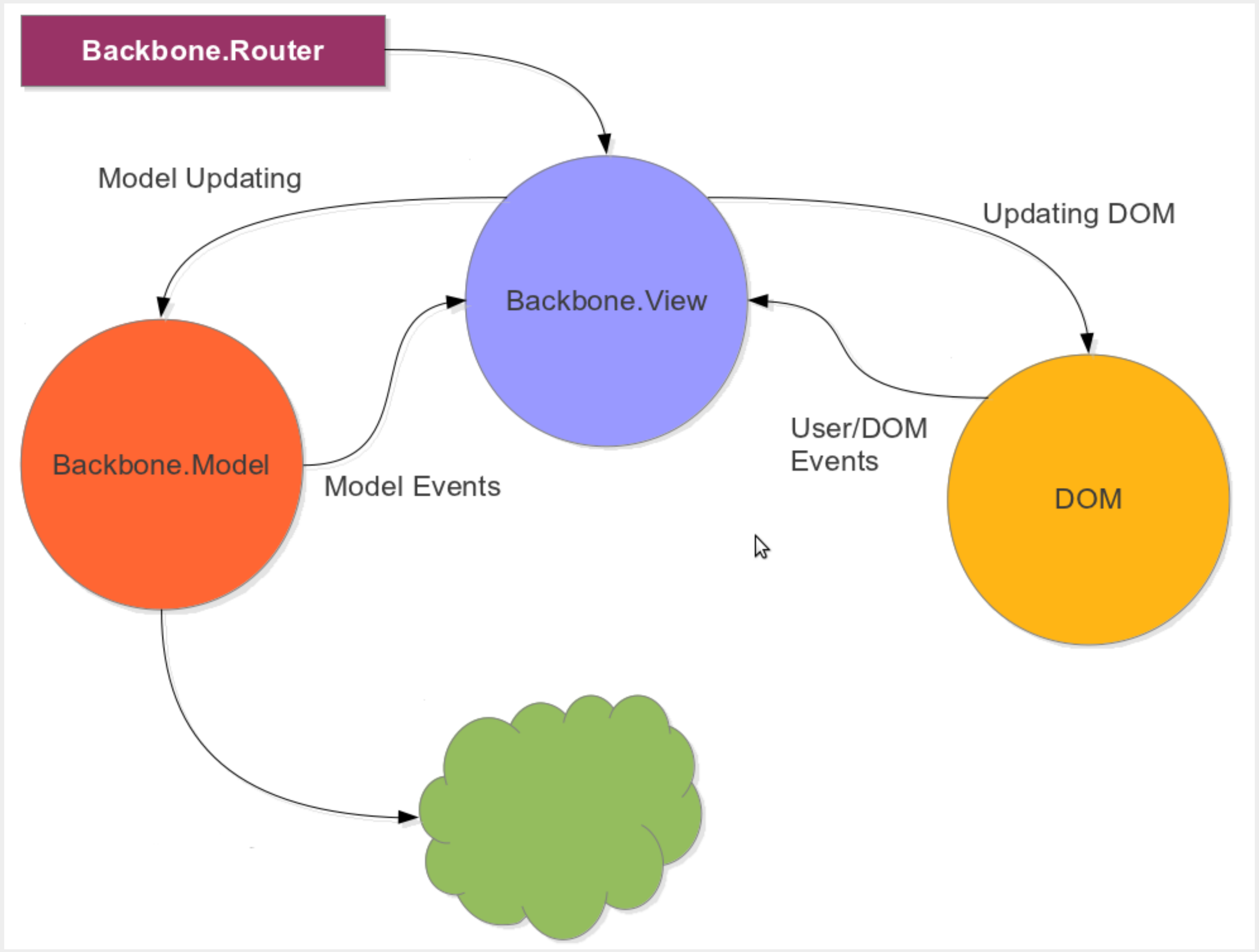


image source http://www.wikipedia.org

backbone.js - MVC



why use Backbone.js?

- **Create web/mobile apps that are strucutred and organized**
- **Move away from misusing jQuery to store everything in the DOM**
- **Move towards a Client/Server architecture**
- **Server sends data (as JSON) without the View (Html/CSS), Client is responsible for the View**
- **Many resources available, great community**
- **Many applications(trello.com, linkedin,...) in the wild**

```
CRUD operations to HTTP operations with REST

          Collection Operations
            HTTP/GET=Read
            HTTP/POST=Create

          Model Operations
```

all credits go to Jérôme Gravel-Niquet

Example with complete source code annotations here

Sample App (Todo App)

Backbone.Model

Backbone.Model

- **Single Data Entity**
- **Usually bound to a View**
- **Models use Backbone.Sync to persist to a storage**

  Either html5 localStorage or/and a remote Server

Clear Run

```javascript
// Simple Model Creation
var Todo = Backbone.Model.extend({
});
// Create an instance of the Model Todo
var myTodo = new Todo({
  title:'Wash ears',
  description:'Dont forget the ears'
});

myTodo.set({
  'title': 'Wash right and left ears'
}); // trigger change
console.log(myTodo.get('title'));
```

Backbone.Model

- **On save the object gets a unique id**
- **Storage of Model is independant of Model Implementation**

  either use url: for remote, or localStorage addon

Clear Run

```
// Model Creation with defaults
var Todo = Backbone.Model.extend({
  //urlRoot:'/api/todos', // Sync to remote server, remove backbone-localStorage.js
  //localStorage: new Store("todos"), // Sync to html5 localStorage
  defaults: {
        title:'new todo...',
        description:'missing...',
        done:false
  }
});

var myTodo = new Todo({});

console.log("Before save id:"+myTodo.get('id')); // Unique Id from Server
console.log("Before save cid:"+myTodo.cid); // Client side id
//myTodo.save(); // trigger sync

console.log(myTodo.get('title'));
console.log("After save id:"+myTodo.get('id'));
console.log("After save cid:"+myTodo.cid);
```

Sample App Todo.Model

```
// Our basic **Todo** model has `text`, `order`, and `done` attributes.
var Todo = Backbone.Model.extend({

 // Default attributes for a todo item.
 defaults: function() {
   return {
     done:  false,
     sortby: Todos.nextOrder() // Will see soon where Todos comes from
   };
 },

 // Toggle the `done` state of this todo item.
 toggle: function() {
   this.save({done: !this.get("done")});
 }
});

```

Backbone.Collection

Backbone.Collection

- **Backbone.Collection stores a set of models of the same typ**
- **Can have a REST point defined**

  url: '/todos'

  - create a new Model in the collection, calls HTTP POST /todos
  - delete a Model from the collection, calls HTTP DELETE /todos/model.id
  - edit a Model in the collection, calls HTTP PUT /todos/model.id
  - fetch all Models from the collection, calls HTTP GET /todos

```
// Collection Creation with model and url
var Todo = Backbone.Model.extend({}); // Model

var Todos = Backbone.Collection.extend({
  model: Todo,
  url: "/todos"
});

var todos = new Todos();
todos.fetch(); // Trigger reset Event

```

Clear Run

Backbone.Collection.create

- **Create New model and add to Collection**

```
// Collection Creation with model and url
var Todo = Backbone.Model.extend({}); // Model

var Todos = Backbone.Collection.extend({
  model: Todo,
  localStorage: new Store("todos")
});

var todos = new Todos();
todos.create({title:'Wash ears', done:false});
console.log(todos.length);

var myTodo = new Todo({title:'wash hands', done:false});
todos.add(myTodo);
console.log(todos.length);

```

Clear Run

Sample App Todos.Collection

```
// The collection of todos is backed by *localStorage* instead of a remote server.
var TodoList = Backbone.Collection.extend({

 // Reference to this collection's model.
 model: Todo,
 url: '/todos', // Use for remote server connection
 //localStorage: new Store("todos",) // Use for local html5 data storage

 // Filter down the list of all todo items that are finished.
 done: function() {
   return this.filter(function(todo){ return todo.get('done'); });
 },

 // Filter down the list to only todo items that are still not finished.
 remaining: function() {
   return this.without.apply(this, this.done());
 },

 // We keep the Todos in sequential order, despite being saved by unordered
 // GUID in the database. This generates the next order number for new items.
 nextOrder: function() {
   if (!this.length) return 1;
   return this.last().get('sortby') + 1;
 },

 // Todos are sorted by their original insertion order.
 comparator: function(todo) {
   return todo.get('sortby');
 }

});
```

Backbone.Events

Backbone.Events binding

- **Events are triggered on changes/destroy on Backbone Objects**
- **Events can be bound to Model/Collection (for example)**

Clear Run

```javascript
// Collection Creation with Event binding
var Todo = Backbone.Model.extend({
  initialize:function(){
   this.on('change', this.changeMe);
  },
  changeMe:function(){
    console.log('I have been changed');
  }
}); // Model
var Todos = Backbone.Collection.extend({
  model: Todo,
  localStorage: new Store("todos"),
  initialize: function(){
    this.on('add', this.addOne);
  },
  addOne: function(newTodo){
   console.log('Adding new Todo with title'+newTodo.get('title'));
  }
});
var todos = new Todos();
todos.create({title:'Wash ears', done:false});
console.log(todos.length);
var myTodo = new Todo({title:'wash hands', done:false});
todos.add(myTodo);
console.log(todos.length);

```

Backbone.Events Overview

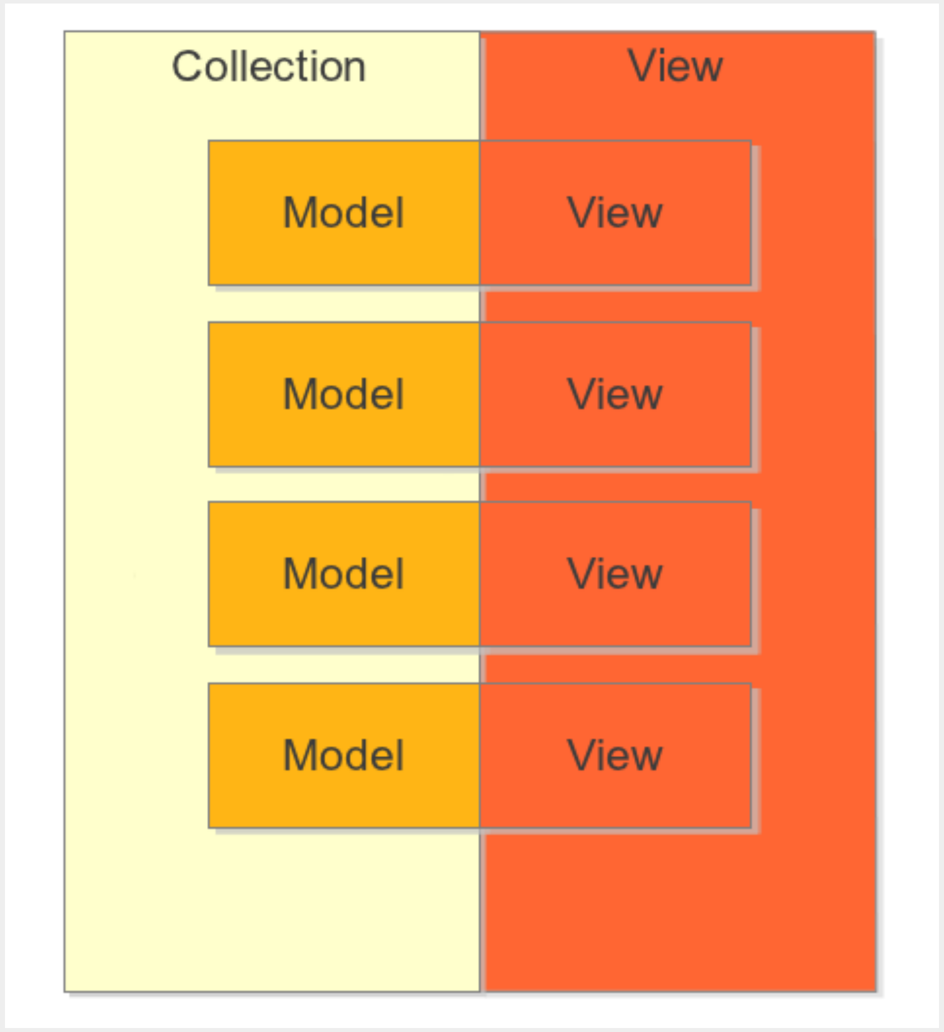
- **Events can be bound any object**

```
// Collection Creation with Event binding
todo.on('EVENT', function(){}, [context]); // Bind event to object
todo.on('change:done', function(model){/* code here */}, this);
todo.on('destroy', this.remove, this); //On deleting Object
todo.off('change'); // Remove Event Listener from object

todos.on('reset', this.addAll, this); // Reset is fired after fetch
todos.on('add', this.addone, this); // Reset is fired after fetch
todos.bind('all', this.render); // Bind is an alias for on, all listens on all events

```

Backbone.View

Backbone.View

- **Manipulates the DOM**
- **Delegates DOM Events**
- **Bound to either a Model or a Collection**



Backbone.View Example

- **All views have a DOM element at all times (the View.el property)**

  View.$el is a cached jQuery/Zepto object of the view's el element

```
// View for TodoCollection is a list ul
var TodosView = Backbone.View.extend({
  tagName: 'ul',
  className: 'todos-list',
  id: 'main-container',
});
var todoView = new TodosView();
console.log(todoView.el);

```

Backbone.View Example with Events

- **Constructor of View with Event binding and render method**

```
// View for TodoCollection is a list ul
var TodoView = Backbone.View.extend({
  //... is a list tag.
  tagName:  "li",

  // The DOM events specific to an item.
  events: {
    "click .check"              : "toggleDone" // Call function from view
  },

  // The TodoView listens for changes to its model, re-rendering.
  initialize: function() {
    this.model.bind('change', this.render, this);
    this.model.bind('destroy', this.remove, this);
  },

  // Re-render the contents of the todo item.
  render: function() {
    this.$el.text(this.model.get('title')); // Just render the title of the Todo
    return this;
  },

  toggleDone: function(){/*...*/}
});
var Todo = Backbone.Model.extend({});
var myTodo = new Todo({title:'Wash Ears'});
var todoView = new TodoView({model:myTodo});
console.log(todoView.el); // Log the View element
console.log(todoView.render()); //Log render out-put
```

Templates

- **Any template can be used**

  ...from underscore.js

- **Mustache, jquery.Tmpl()**

```
// Template with Mustache
 <script type="text/template" id="item-template">
   <div class="todo {{ done ? 'done' : '' }}">
     <div class="display">
       <input class="check" type="checkbox" {{ done ? 'checked="checked"' : '' }} />
       <div class="todo-text"></div>
       <span class="todo-destroy"></span>
     </div>
     <div class="edit">templ
       <input class="todo-input" type="text" value="" />
     </div>
   </div>
 </script>

```

Caching Template

- **The template instance can be cached in the View**

```
// Caching template instance in the View using underscore.js
var TodoView = Backbone.View.extend({

  // Cache the template function for a single item.
  template: _.template( $('#item-template').html() ),

  render: function() {
      this.$el.html(this.template(this.model.toJSON()));
   }

  /*....*/
});

```

Sample App Todo.Model

```javascript
var TodoView = Backbone.View.extend({
    //... is a list tag.
    tagName:  "li",
    // Cache the template function for a single item.
    template: _.template($('#item-template').html()),
    // The DOM events specific to an item.
    events: {
      "click .check"              : "toggleDone",
      "dblclick div.todo-text"    : "edit",
      "click span.todo-destroy"   : "clear",
      "keypress .todo-input"      : "updateOnEnter"
    },

    // The TodoView listens for changes to its model, re-rendering.
    initialize: function() {
      this.model.bind('change', this.render, this);
      this.model.bind('destroy', this.remove, this);
    },

    // Re-render the contents of the todo item.
    render: function() {
      $(this.el).html(this.template(this.model.toJSON()));
      this.setText();
      return this;
    },

    // To avoid XSS (not that it would be harmful in this particular app),
    // we use `jQuery.text` to set the contents of the todo item.
    setText: function() {},

    // Toggle the `"done"` state of the model.
    toggleDone: function() {},

    // Switch this view into `"editing"` mode, displaying the input field.
    edit: function() {},

    // Close the `"editing"` mode, saving changes to the todo.
    close: function() {},

    // If you hit `enter`, we're through editing the item.
    updateOnEnter: function(e) {},
```

Sample App Todos.Collection

```
// Our overall **AppView** is the top-level piece of UI.
  var AppView = Backbone.View.extend({
    // Instead of generating a new element, bind to the existing skeleton of
    // the App already present in the HTML.
    el: $("#todoapp"),

    // Our template for the line of statistics at the bottom of the app.
    statsTemplate: _.template($('#stats-template').html()),

    // Delegated events for creating new items, and clearing completed ones.
    events: {
      "keypress #new-todo":  "createOnEnter",
      "keyup #new-todo":     "showTooltip",
      "click .todo-clear a": "clearCompleted"
    },

    initialize: function() {
      this.input    = this.$("#new-todo");
      Todos.bind('add',   this.addOne, this);
      Todos.bind('reset', this.addAll, this);
      Todos.bind('all',   this.render, this);
      Todos.fetch();
    },

    render: function() {
      this.$('#todo-stats').html(this.statsTemplate({
        total:      Todos.length,
        done:       Todos.done().length,
        remaining:  Todos.remaining().length
      }));
    },

    // Add a single todo item to the list by creating a view for it, and
    // appending its element to the `<ul>`.
    addOne: function(todo) {
      var view = new TodoView({model: todo});
      $("#todo-list").append(view.render().el);
    },

    // Add all items in the **Todos** collection at once.
    addAll: function() {
```

Backbone.Router

Backbone.Router

- **Maps urls to functions**
- **Enables hashbang URLs**

  www.myapp.com/#!/todos/get/123, see twitter.com/#!/littleiffel
- **Enables Browser History/Bookmarking**

Backbone.Router Example

```
// View for TodoCollection is a list ul
var AppRouter = Backbone.Router.extend({
 routes: {
    "":      "index",
    "/add": "addTodo",
    "/show/:id": "showTodo",

 },
 index: function () { console.log("index"); },
 addTodo: function () { console.log("addTodo"); },
 showTodo: function(id){console.log("showing Todo id:"+id); }
});

```

Mobile App Creation

We almost got a mobile app with

- localStorage
- But....runs only Browser

- **Use localStorage**

  Store the application data in the borwser. Up to 5MB in a large "Cookie", check here

- **Cache Manifest**

  Store the application files (html/css/js) in Browser Cache to work in Offline Modus, Cache the application check here

What you call this a mobile app?

- Well, just use Titanium Appcelerator, PhoneGap, XUI, Cappucino,.....to convert THIS to a "native" app for iPhone, Android $ Co.
- Examples: LinkedIn iPhone, Android,...

Mobile HTML5 Offline Backbone Demo Todo App

Mobile HTML5 Offline Backbone Demo Todo App

- **Have a Look at the files:**
- index.html
- todos.js

So all is shiny with backbone.js?

- **There are many alternatives to backbone.js**

  Top 10 JS MVC Frameworks

- **Pros**

  extend, Huge Community, plenty of resources/tutorials, lightweight, underscore.js, many "real-world" examples

- **Cons**

  it's a framework -> either you like it or not, for the rest go find out for yourself...

Thanks for your patience and attention

Questions?

Resources

- Backbone.js
- Sample Todo App on Backbone.js
- Slides on HTML5 Moblie Apps from Any.Do developer
- A NICE presentation on Backbone.js
- Backbone.js Fundamentals Free EPub
- Backbone Boilerplate - Getting Started quickly with development
- Backbone Tutorials