BACKBONE.JS

javascript library

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

BackboneJS is a light weight JavaScript library that allows to develop and structure client side applications that run in a web browser. It offers MVC framework which abstracts data into models, DOM (Document Object Model) into views and bind these two using events.

This tutorial covers most of the topics required for a basic understanding of BackboneJS and to get a feel of how it works.

Audience

This tutorial is designed for software programmers who want to learn the basics of BackboneJS and its programming concepts in simple and easy ways. This tutorial will give you enough understanding on various components of BackboneJS with suitable examples.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of HTML, CSS, JavaScript, and Document Object Model (DOM). As we are going to develop a web-based applications using BackboneJS, it will be good if you have an understanding of how web-based applications work in general.

Copyright & Disclaimer

Table of Contents

1. BackboneJS – Overview

BackboneJS is a **lightweight JavaScript library** that allows to develop and structure the client side applications that run in a web browser. It offers MVC framework which abstracts data into models, DOM into views and bind these two using events.

History: BackboneJS was developed by Jeremy Ashkenas and was initially released on October 13th, 2010.

When to use Backbone

- Consider you are creating an application with numerous lines of code using JavaScript or jQuery. In this application, if you:
    - add or replace DOM elements to the application or
    - make some requests or
    - show animation in the application or
    - add more number of lines to your code,

  then your application might become complicated.
- If you want a better design with less code, then it is better to use the BackboneJS library that provides good functionality, is well organized and in a structured manner for developing your application.
- BackboneJS communicates via events; this ensures that you do not mess up the application. Your code will be cleaner, nicer and easy to maintain.

Features

The following are a list of features of BackboneJS:

- BackboneJS allows developing of applications and the frontend in a much easier way by using JavaScript functions.
- BackboneJS provides various building blocks such as models, views, events, routers and collections for assembling the client side web applications.
- When a model changes, it automatically updates the HTML of your application.
- BackboneJS is a simple library that helps in separating business and user interface logic.
- It is free and open source library and contains over 100 available extensions.
- It acts like a backbone for your project and helps to organize your code.

- It manages the data model which includes the user data and displays that data at the server side with the same format written at the client side.
- BackboneJS has a soft dependency with **jQuery** and a hard dependency with **Underscore.js**.
- It allows to create client side web applications or mobile applications in a well-structured and an organized format.

2. BackboneJS – Environment Setup

BackboneJS is very easy to setup and work. This chapter will discuss the download and setup of the **BackboneJS Library**.

BackboneJS can be used in the following two ways:

- Downloading UI library from its official website.
- Downloading UI library from CDNs

Downloading the UI library from its official website

When you open the link http://backbonejs.org/, you will get to see a screenshot as shown below:

As you can see, there are three options for download of this library:

- **Development Version** - Right click on this button and save as and you get the full source **JavaScript library**.
- **Production Version** - Right click on this button and save as and you get the **Backbone-min.js library** file which is packed and gzipped.
- **Edge Version** - Right click on this button and save as and you get an **unreleased version**, i.e., development is going on; hence you need to use it at your own risk.

Dependencies

BackboneJS depends on the following JavaScript files:

- **Underscore.js:** This is the only hard dependency which needs to be included. You can get it from here.

- **jQuery.js:** Include this file for RESTful persistence, history support via Backbone.Router and DOM manipulation with Backbone.View. You can get it from here.

- **json2.js:** Include this file for older Internet Explorer support. You can get it from here.

Download UI Library from CDNs

A CDN or **Content Delivery Network** is a network of servers designed to serve files to users. If you use a CDN link in your web page, it moves the responsibility of hosting files from your own servers to a series of external ones. This also offers an advantage that if the visitor to your webpage has already downloaded a copy of BackboneJS from the same CDN, it won't have to be re-downloaded.

As said above, BackboneJS has a dependency of the following JavaScript:

- jQuery
- Underscore

Hence CDN for all the above is as follows:

```
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.5.2/jquery.min.js"></script>

<script type="text/javascript"
src="https://ajax.cdnjs.com/ajax/libs/underscore.js/1.1.4/underscore-
min.js"></script>

<script type="text/javascript"
src="https://ajax.cdnjs.com/ajax/libs/backbone.js/0.3.3/backbone-
min.js"></script>
```

Note: We are using the CDN versions of the library throughout this tutorial.

Example

Let's create a simple example using BackboneJS.

```
<!DOCTYPE html>

<html>

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

  <title>Hello World using Backbone.js</title>
```

```
</head>
<body>
  <!-- ========= -->
  <!-- Your HTML -->
  <!-- ========= -->
  <div id="container">Loading...</div>
```

```
  <!-- ========= -->


  <!-- Libraries -->
  <!-- ========= -->
  <script src="https://code.jquery.com/jquery-2.1.3.min.js"
type="text/javascript"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/underscore.js/1.3.3/underscore-
min.js" type="text/javascript"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/backbone.js/0.9.2/backbone-
min.js" type="text/javascript"></script>



  <!-- ============== -->
  <!-- Javascript code -->
  <!-- ============== -->
  <script type="text/javascript">
    var AppView = Backbone.View.extend({
      // el - stands for element. Every view has an element associated with HTML
content, will be rendered.
      el: '#container',
      // It's the first function called when this view is instantiated.
      initialize: function(){
        this.render();
      },
      // $el - it's a cached jQuery object (el), in which you can use jQuery
functions to push content. Like the Hello TutorialsPoint in this case.
```

```
      render: function(){
        this.$el.html("Hello TutorialsPoint!!!");
      }
    });

    var appView = new AppView();
  </script>
</body>
</html>
```

The code comments are self-explanatory. A few more details are given below:

There's a html code at the start of *body* tag

```
<div id="container">Loading...</div>
```

This prints ***Loading...***

Next, we have added the following CDNs

```
<script src="https://code.jquery.com/jquery-2.1.3.min.js"
type="text/javascript"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/underscore.js/1.3.3/underscore-
min.js" type="text/javascript"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/backbone.js/0.9.2/backbone-
min.js" type="text/javascript"></script>
```

Next, we have the following script:

```
var AppView = Backbone.View.extend({
      // el - stands for element. Every view has an element associated with HTML
content, will be rendered.
      el: '#container',
      // It's the first function called when this view is instantiated.
      initialize: function(){
        this.render();
      },
      // $el - it's a cached jQuery object (el), in which you can use jQuery
functions to push content. Like the Hello World in this case.
```

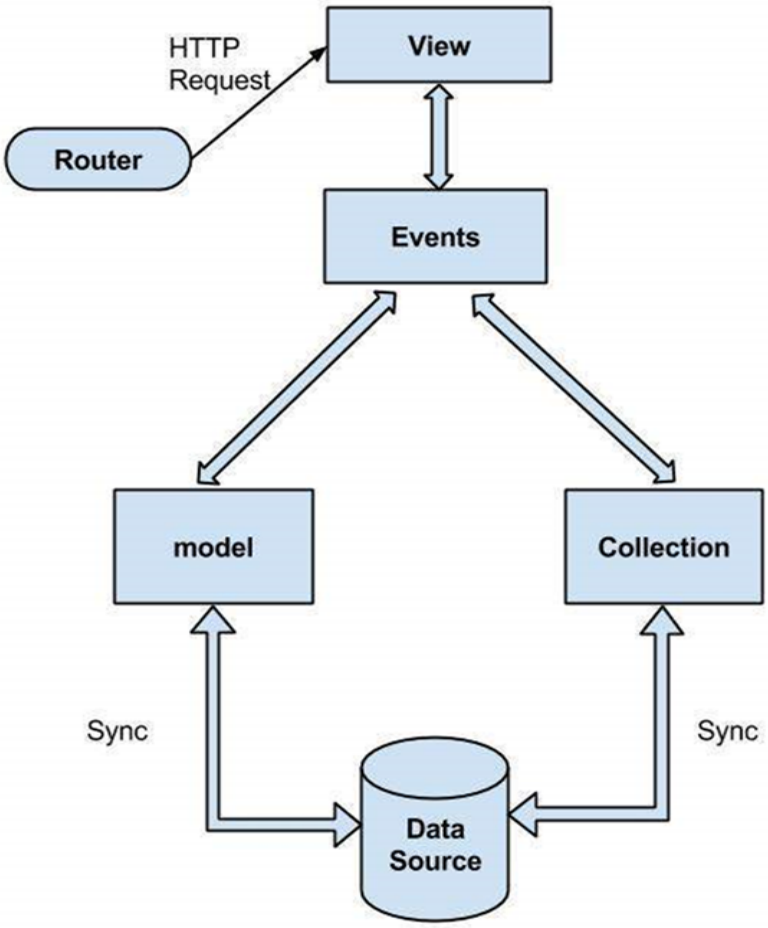
```
    render: function(){
      this.$el.html("<h1>Hello TutorialsPoint!!!</h1>");
    }
  });

  var appView = new AppView();
```

The comments are self-explanatory. In the last line, we are initializing ***new AppView()***. This will print the "Hello TutorialsPoint" in the **div** with **id="container"**

Save this page as **myFirstExample.html**. Open this in your browser and the screen will show the following text.

3. BackboneJS – Applications

The BackboneJS gives a structure to the web applications that allows to separate business logic and user interface logic. In this chapter, we are going to discuss the architectural style of the BackboneJS application for implementing user interfaces. The following diagram shows the architecture of BackboneJS :



The architecture of BackboneJS contains the following modules:

- HTTP Request
- Router
- View
- Events

- Model
- Collection
- Data Source

Let us now discuss all the modules in detail.

HTTP Request

The HTTP client sends a HTTP request to a server in the form of a request message where web browsers, search engines, etc., acts like HTTP clients. The user requests for a file such as documents, images, etc., using the HTTP request protocol. In the above diagram, you could see that the HTTP client uses the router to send the client request.

Router

It is used for routing the client side applications and connects them to actions and events using URL's. It is a URL representation of the application's objects. This URL is changed manually by the user. The URL is used by the backbone so that it can understand what application state to be sent or present to the user.

The router is a mechanism which can copy the URL's to reach the view. The Router is required when web applications provide linkable, bookmarkable, and shareable URL's for important locations in the app.

In the above architecture, the router sending an HTTP request to the View. It is a useful feature when an application needs routing capability.

View

BackboneJS views are responsible for how and what to display from our application and they don't contain HTML markup for the application. It specifies an idea behind the presentation of the model's data to the user. Views are used to reflect "how your data model looks like".

The view classes do not know anything about the HTML and CSS and each view can be updated independently when the model changes without reloading the whole page. It represents the logical chunk of the UI in the DOM.

As shown in the above architecture, the View represents the user interface which is responsible for displaying the response for the user request done by using the Router.

Events

Events are the main parts of any application. It binds the user's custom events to an application. They can be mixed into any object and are capable of binding and triggering custom events. You can bind the custom events by using the desired name of your choice.

Typically, events are handled synchronously with their program flow. In the above architecture, you could see when an event occurs, it represents the model's data by using the View.

Model

It is the heart of the JavaScript application that retrieves and populates the data. Models contain data of an application, logic of the data and represents the basic data object in the framework.

Models represents business entities with some business logic and business validations. They are mainly used for data storage and business logic. Models can be retrieved from and saved to data storage. A Model takes the HTTP request from the Events passed by the View using the Router and synchronizes the data from the database and sends the response back to the client.

Collection

A Collection is a set of models which binds events, when the model has been modified in the collection. The collection contains a list of models that can be processed in the loop and supports sorting and filtering. When creating a collection, we can define what type of model that collection is going to have along with the instance of properties. Any event triggered on a model will also trigger on the collection in the model.

It also takes the request from the view, bind events and synchronizes the data with the requested data and sends the response back to the HTTP client.

Data Source

It is the connection set up to a database from a server and contains the information which is requested from the client. The flow of the BackboneJS architecture can be described as shown in the following steps:

- A User requests for the data using the router, which routes the applications to the events using the URL's.
- The view represents the model's data to the user.
- The model and collection retrieves and populates the data from the database by binding custom events.

In the next chapter, we will understand the significance of Events in BackboneJS.

End of ebook preview

If you liked what you saw…

Buy it from our store @ **https://store.tutorialspoint.com**