Backbone Router

ROHAN RAJORE

Creating a Router

- Creating a router in Backbone follows the same patterns as all objects in the library, using the .extend method to define your custom router.

```
MyRouter = Backbone.Router.extend({
});
```

- Next you will need to match particular URL patterns with functions that get invoked when that URL fragment is navigated to. This is done by passing through a routes hash to your extend function.

```
MyRouter = Backbone.Router.extend({
routes: {'hello' : 'sayHello'},
});
```

Creating a Router

- The previous code sets up a route so that if the URL ends with #hello, a function named sayHello will be invoked. This function should be defined along with your router, like so:

```
MyRouter = Backbone.Router.extend({
    routes: {'hello' : 'sayHello'},
    sayHello: function(){
        console.log('Saying hello');
    }
});
```

Creating a Router

- To get the router listening for route changes, a new instance of the router will need to be created.

```
var router = new MyRouter();
```

- Finally, to monitor changes in the hash fragments and invoke the appropriate events, you will need to initialize Backbone.history.

```
Backbone.history.start();
```

Backbone.History

- Backbone.history is an abstraction provided by the library that will handle hashchange events or the HTML5 pushState. When a change in URL is observed, the appropriate route is matched with the callback that you have defined in your own router.
- As you saw in the example at the start of this section, Backbone.history.start() is called to start monitoring for change events. Without this, your routers will not run successfully.
- It’s worth paying attention to the Boolean value returned from the start() function; if no route matches the current URL, it will return false.