serious #**js**
to **infinity** and beyond!

backbone collections

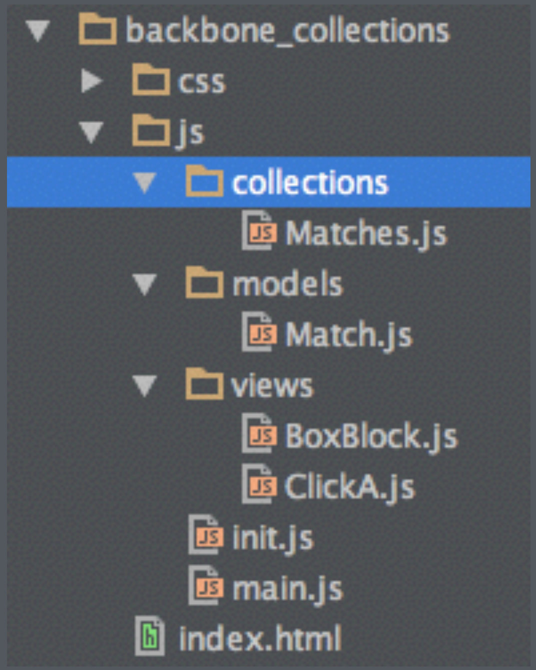
Backbone collections are ordered sets of models. You can bind "change" events to be notified when any model in the collection has been modified, listen for "add" and "remove" events, fetch the collection from the server, and use a full suite of Underscore.js methods.

```
//main.js
(function () {
    site.init = function () {
        var matches = new site.collections.Matches();
        new site.views.ClickA({el: "#clicker"});
        new site.views.BoxBlock({el: "#box", collection: matches});
    };

    site.$document.on('ready', site.init);
})();
```

#JS - JS STRUCTURE

code



backbone_collections
css
js
collections
Matches.js
models
Match.js
views
BoxBlock.js
ClickA.js
init.js
main.js
index.html

```
//Collection of maths pointing to an empty Model
site.collections.Matches = Backbone.Collection.extend({
    model: site.models.Match,
    url: 'http://docent.cmi.hr.nl/moora/imp03/api/wedstrijden'
});
```

#JS - BACKBONE COLLECTIONS

code

```
/**
 * Wrapper function to load the matches through the model
 */
loadMatches: function () {
    this.model.fetch({
        success: _.bind(this.loadMatchesSuccessHandler, this),
        error: _.bind(this.loadMatchesErrorHandler, this),
        data: {
            league: 'PrimeraDivision',
            club: 'Getafe'
        }
    });
},
```

#JS - BACKBONE MODEL FETCH

code

assignment

Bekijk hoe **collections** werken in Backbone op de website: http://backbonejs.org/#Collection

Zorg ervoor dat je de **model.fetch** ombouwt naar een **collection.fetch.** Maak in de success handler een variabele die alle models met thuiswedstrijden van de club uit de collection haalt. Je kunt hier de **filter** method voor gebruiken.

```
/**
 * Wrapper function to load the matches through the collection
 */
loadMatches: function () {
    this.collection.fetch({
        success: _.bind(this.loadMatchesSuccessHandler, this),
        error: _.bind(this.loadMatchesErrorHandler, this),
        data: {
            league: 'PrimeraDivision',
            club: 'Getafe'
        }
    });
},
```

#JS - BACKBONE COLLECTIONS

code

```
/**
 * @param collection
 * @param response
 * @param options
 */
loadMatchesSuccessHandler: function (collection, response, options) {
    console.log("SUCCESS");
    console.dir(collection);
    console.dir(collection.models);

    //Example of using models to filter data in new variable
    var homeMatches = collection.filter(function (match) {
        return match.get('homeClub') == 'Getafe';
    });

    console.dir(homeMatches);
},
```

#JS - BACKBONE COLLECTIONS

code

backbone router

Web applications often provide linkable, bookmarkable, shareable URLs for important locations in the app. Until recently, hash fragments (#page) were used to provide these permalinks, but with the arrival of the History API, it's now possible to use standard URLs (/page).

```
//Example router with custom routes
var Router = Backbone.Router.extend({
    routes: {
        'books/:book': ‘bookAction’,
        ‘test/:test/:more’: ‘testAction’ //Possible to create bigger urls
    },

    /**
     * Route callback
     *
     * @param book
     */
    bookAction: function (book) {
        //Do something with book.
    }
});
```

#JS - BACKBONE ROUTER

code

```
//Enables history for routing state
Backbone.history.start();

//Push state activates ‘normal’ urls, instead of # urls
Backbone.history.start({pushState: true);

//Pass root URL when working on deeper level
Backbone.history.start({pushState: true, root: ‘/some/url/’});
```

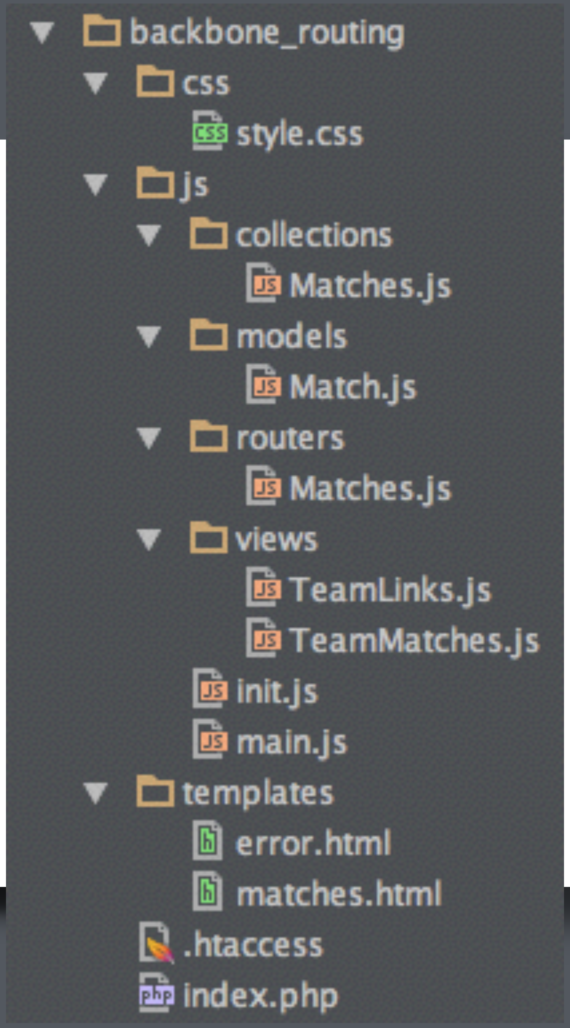
#JS - BACKBONE HISTORY

code

Note that using real URLs requires your web server to be able to correctly render those pages, so back-end changes are required as well. For example, if you have a route of /documents/100, your web server must be able to serve that page, if the browser visits that URL directly.

For full search-engine crawlability, it's best to have the server generate the complete HTML for the page ... but if it's a web application, just rendering the same content you would have for the root URL, and filling in the rest with Backbone Views and JavaScript works fine.

```
//init.js
(function () {
    window.site = {};
    site.$document = $(document);
    site.views = {};
    site.collections = {};
    site.models = {};
    site.routers = {};
    site.events = _.clone(Backbone.Events);
})();
```



```
backbone_routing
  css
    style.css
  js
    collections
      Matches.js
    models
      Match.js
    routers
      Matches.js
    views
      TeamLinks.js
      TeamMatches.js
    init.js
    main.js
  templates
    error.html
    matches.html
  .htaccess
  index.php
```

#JS - JS STRUCTURE

code

```
<!— index.php contents in <body> —>
<div id="container">
    <div id="team-links">
        <a href="#" data-club="Ajax" data-league="Eredivisie">Ajax</a>
        <a href="#" data-club="Getafe" data-
league="PrimeraDivision">Getafe</a>
        <a href="#" data-club="Liverpool" data-
league="PremierLeague">Liverpool</a>
    </div>

    <div id="team-matches">
        <span class="start-message">Choose a team on the left!</span>
    </div>
</div>
```

#JS - JS STRUCTURE

code

assignment

Maak een kleine applicatie met de volgende eisen:

- Routing op basis van **/matches/:league/:club**
- Haal **data** van het team op bij **klik** op een **link**
  - On click spreek je de router aan via **this.router.navigate**(url, {trigger: true, replace: true});
- **Toon matches** op scherm met clubs, stadion en plaats

Bonus:

- Gebruik templates via _**.template** voor matches & error
- Gebruik een **.htaccess** om **pushState** te kunnen gebruiken IPV # urls

```
<?php print “DEMO in phpStorm”; ?>
```

#JS - BACKBONE APPLICATION

code

huiswerk

- Oefenen met stof van Backbone lessen!
- **Werken aan eindopdracht**