**[BACKBONEJS – DÀNH CHO NGƯỜI MỚI BẮT ĐẦU](http://taichuong.com/index.php/2016/11/24/backbonejs-danh-cho-nguoi-moi-bat-dau/)**

BackboneJS là một thư viện Javascript, trong hàng đàn hàng đống các thư viện khác, nó được chú ý đặc biệt bởi cộng đồng phát triển web sử dụng Javascript bởi nó dễ sử dụng và rất dễ áp dụng cho các ứng dụng javascript.

Chú ý rằng BackboneJS không phải là một **framework**mà là một **library**. Sự khác nhau của hai khái niệm này là “who is control”. Sử dụng library, bạn là người điều khiển và sử dụng. Cơ mà với framework thì ngược lại, bạn phải apply code của bạn theo phương thức của framework. Libraries mang lại cho bạn sự linh động, do bạn chủ động dùng nó, trong khi đó frameworks rất cứng đầu nên bạn phải viết code theo frame work (bắt buộc luôn, bạn cứ code nhiều thì biết), cơ mà việc sử dụng framework sẽ tiết kiệm cho bạn rất nhiều code bởi vì framwork thường đi theo một hoặc một số kiểu ứng dụng nhất định, còn libraries thì bạn có thể dùng cho ứng dụng nào cũng được.

Thôi nào! Chúng ta không bàn bạc về **framework**và **library**nữa, đi vào BackboneJS thôi.

**Basic for anyone.**

Các ứng dụng nền web đã phát triền lâu lắm rồi, nhiều đến nỗi không nhớ nổi nữa. Thủa xa xưa thì web chỉ là **static HTML page**, nó bao gồm HTML, CSS, JS. Khi người dùng muốn thay đổi nội dung của web thì phải thay đổi code của 3 thằng mặt lợn bên trên. Về sau thì web 2.0 server side programming languages (PHP, Ruby, Java,…) được thêm vào để tạo ra HTML một cách động. “Động” có nghĩa là sử dụng dữ liệu được chứa trong database và phụ thuộc vào user input. Ví dụ như ngày xưa mà muốn in ra 1000 cái tên, tuổi, địa chỉ của 1000 em sinh viên thì phải viết code HTML cho 1000 thằng (ví dụ thôi nhé), còn bây giờ thì chỉ cần để thống số của 1000 thằng đó vào trong cơ sở dữ liệu, dùng truy vấn trên phía server rồi tạo ra html bằng server code (PHP, Java, C#.net, …) sau đó chuyển về cho người dùng.  
**Nhưng,** nếu web nó to, yêu cầu nhiều dữ liệu thì cái thằng server phải tạo ra nhiều file HTML cho client tương đương với mỗi request của user, kiểu này ngốn rất nhiều băng thông do phải thường xuyên liên lạc với server, cơ mà ở bên phía anh client cũng có cách để giảm thiểu cái việc này đó là dùng javascript ở client. Cơ mà nếu ông nào dùng Javascript thì biết, phần đa viết trong code một cách đếch có tổ chức gì. Tương đương với việc thằng nào đằng sau mà maintain lại thì đúng là thảm họa.  
**Nên**, cần một cái nền tảng hay thư viện gì gì đó để tổ chức lại cái đống javascript hỗn độn đó. Tất nhiên là bạn có thể làm bằng Javascript thuần và tự tạo một cách tổ chức cho riêng mình, cơ mà cái đó cần nhiều thời gian và kinh nghiệm để làm, very hard for beginners!.  
Kết quả, Backbone ra đời để giúp cho bạn tổ chức lại cái mớ lộn xộn javascript, đồng thời tăng tính chất maintainability (dễ đọc và dễ thay đổi code về sau) trong code của bạn. Trong thực tết thì có rất nhiều frameworks tương tự như EmberJS, AngularJS, … Tuy nhiên, backbone đã được gọi là library nên nó có nhiều cái tính linh động mà các frameworks khác không có được, đồng thời sau khi sử dụng Backbone bạn có thể có một cộng đồng hỗ trợ rất mạnh luôn ở bên bạn.

BackboneJS

Rồi rồi! Chém gió luyên thuyên một hồi rồi, chúng ta cũng nên bắt đầu thôi.

BackboneJS là một library phụ thuộc, nó sử dụng library khác, cụ thể như sau:

* [BackboneJS](http://backbonejs.org/)
* [Underscore](http://underscorejs.org/)
* [JQuery](http://jquery.com/)

Việc phụ thuộc vào thư viện khác cũng có cái lợi mà cũng có cái hại, nếu như trang của bạn có sử dụng JQuery thì bạn dùng thêm BackboneJS sẽ làm giảm tổng độ lớn của các thư viện trong web của bạn so với việc dùng Knockout, Angular hay Ember. Bởi vì BackboneJS có tổng kích cỡ nhỏ hơn mấy thằng kia nhiều. Hơn nữa underscore có nhiều hàm rất hữu dụng trong việc xử lý logic của các đối tượng trong javascript.

Nói qua về cấu trúc của Backbone: nó được tạo bởi một số các module như sau:

* Views: cái thằng này được coi là mặt tiền của library, nó sử dụng template của underscore để đưa ra HTML code, tương tự như việc render động dữ liệu trên server bằng PHP hay một số server code khác.
* Events: bao events của jquery, đồng thời nó có thêm một số phương pháp để sử dụng chung events qua các modules khác nhau
* Models: lưu giữ dữ liệu — chỉ nói thôi cũng hiểu rồi nhỉ. hehe. nhưng mà nó có thể exports và imports bằng JSON data. Đồng thời có thể đồng bộ giữ liệu với server thông qua AJAX method (cái này là wrap AJAX để cho mình dễ sử dụng)
* Collections: nó đơn giản là chứa nhiều model, đồng thời wrap các phương thức đồng bộ giữ liệu của model, Ví dụ thay bạn đồng bộ một chục cái model, bạn có thể sử dụng collections để đồng bộ cả cụm.
* Routers: thực ra mình hay viết app one page nên không hay sử dụng thằng này lắm, cơ mà thấy nhiều ví dụ về cái này. Nó dùng để router người dùng trên client, đồng thời sử dụng browser history để lưu lại các page để sau này quay lại.

**Shut up and show me the code!**

(thích nhất câu này ^.^)

OK! Ví dụ đầu tiên của chúng ta là một ứng dụng viết “Hello World” để làm quen với BackboneJS và ví dụ thứ hai là “To do” app, app này sẽ lưu lại các việc, và chúng ta có thể đánh dấu xem việc này đã xong chưa, thời hạn là bao giờ phải xong, … rồi, chúng ta sẽ code thôi.

**1.1. Hello world**

Để start thì bạn nên bắt download thư mục [html file](https://www.dropbox.com/s/589pgr9tlhhqgsz/backbone-tutorial.html?dl=0) thư mục này chứa library của Backbone, JQuery, Underscore và một file html cơ bản để bạn bắt đầu. Nếu bạn thấy file html này lạ quá thì bạn nên qua [www.w3schools.com](http://www.w3schools.com/html/default.asp) để đọc qua mấy cái thẻ tag đi rồi quay lại đây đọc tiếp cái này.

Sau khi download thì bạn sẽ thấy trong file html sẽ có thẻ:

[http://2.bp.blogspot.com/-4cK4gDnTOzA/VIXHllKvyZI/AAAAAAAAAao/qPn_yEnMSfU/s1600/img1.png](http://2.bp.blogspot.com/-4cK4gDnTOzA/VIXHllKvyZI/AAAAAAAAAao/qPn_yEnMSfU/s1600/img1.png)

Bạn sẽ thích cái thẻ này, vì nó sẽ bao toàn bộ app Hello world của bạn trong đó đó.

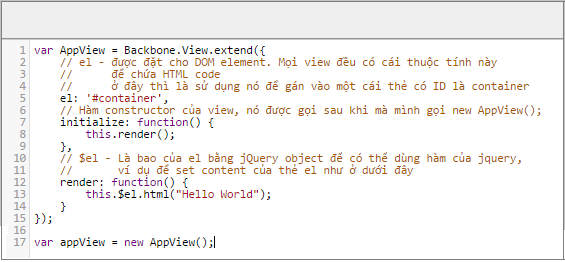
**1.2. Backbone’s Views**

Backbone’s views: trong backbone cũng có khái niệm MVC, nhưng mà cái vai trò của controller thì mỗi người một khác, có người viết controller ngay trong view, có người viết controller ra một module khác(cách này cần có một thời gian rồi thì mới ngộ ra được), cơ mà mỗi người một kiểu, kiểu nào cũng được, chúng ta sẽ đi vào kiểu thứ nhất để tiếp cận dễ dàng hơn.

Backbone sẽ dán các sự kiện như click, press key, mousedown, … của user vào views, đưa ra HTML bằng cách sử dụng template của underscore và tương tác với models – chứa dữ liệu.

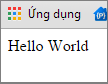
Dưới đây là ví dụ của Backbone.View: chú ý là đọc comment và chèn code vào **Javascript Block**trong HTML file mà bạn down load nhé. (thôi nghĩ lại rồi, các bạn viết code đi cho nhớ ^.^)

**Full code in here:**[I’m here](https://www.dropbox.com/s/9qfpc6k8r5ikxmd/backbone-tutorial.html?dl=0)

[](http://4.bp.blogspot.com/-U5MIl_qAYDY/VIXDF-XOw1I/AAAAAAAAAaA/0OQtJqCfX0g/s1600/img1.png)

**1.3. Test hello world**

Sau khi viết xong thì bạn mở file bằng browser. Và sẽ nhìn thấy cái chữ “Hello World” trên màn hình. Cơ mà nhìn xấu quá. Nhưng mà yên tâm là chỉ là bắt đầu thôi.

[](http://1.bp.blogspot.com/-2mKHFFZPI9k/VIXD_eMcYnI/AAAAAAAAAaI/C7Wm5r83hzU/s1600/img1.png)

**1.4. Backbone’s Templates**

Phần này là phần rất thú vị trong Backbone. Nó sử dụng Underscore template để tạo ra HTML code, một cách “động”. Như mình đã nói ở phần trước về tính “động” của server code khi tạo ra HTML trên server, các bạn có thể tưởng tượng underscore giống y như vậy, nó có thể chèn javascript code vào và chạy cùng với model để đưa ra kết quả là HTML code. Underscore khi được import vào HTML file nó có thể hiện là “\_”. Dấu gạch dưới (underscore).

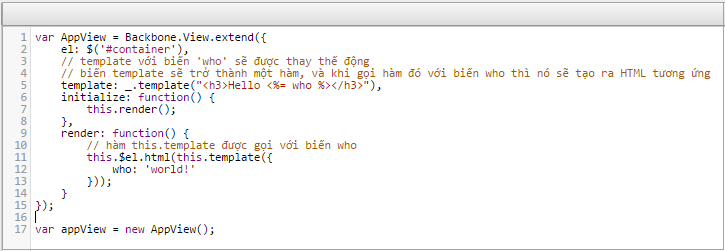
Cú pháp để tạo một template là:

\_.template(templateString, [data], [settings])

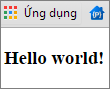
Với templateString bạn có thể sử dụng <%= dataVariable %> hoặc <%- dataVariable %> để chèn data động thông qua input. Hơn thế nữa, bạn có thể đặt javascript trong <% code %> để đặt logic cho phần tạo HTML

Rồi, chúng ta sẽ thay đổi code lúc trước một chút để tạo ra một app “Hello World” bằng template.

**Full code in here:**[I’m here](https://www.dropbox.com/s/l4zjgy0idkkx2nm/backbone-tutorial-2.html?dl=0)

[](http://4.bp.blogspot.com/-6mNiuHL99lw/VIXLZYfnf5I/AAAAAAAAAa0/xLwLd1x9ako/s1600/img1.png)

Giờ thì thử load lại xem nào!

[](http://1.bp.blogspot.com/-2Ptq4Xvyo-8/VIXGreCjN0I/AAAAAAAAAac/ZNdYQM9piSo/s1600/img1.png)

OK! Chúng ta đã thử với Backbone View. Giờ thì qua Model và Events xem thế nào nhé…

[**BACKBONE LÀ GÌ ?**](http://taichuong.com/index.php/2016/11/24/backbone-la-gi/)

Backbone là một javascript framework giúp cho việc code hiệu quả hơn, và tạo ra một ứng dụng js đẹp, tốt và có cấu trúc hơn. Nó cung cấp rất nhiều API giúp phát triển ứng dụng nhanh hơn.

Là lập trình viên, chúng ta đều hiểu rằng : Rồi một lúc nào đó, các yêu cầu càng ngày càng trở nên phức tạp hơn rất nhiều, việc code không còn đơn giản chỉ là sử dụng một vài dòng lệnh là đủ, Số lượng dòng code tăng cao, mối liên hệ giữa chúng cũng tăng, Việc code và quản lý code trở nên rất phức tạp.

Sự hiệu quả và ý nghĩa của lập trình hướng đối tượng, lập trình hướng khía cạnh cùng với các mô hình lập trình MVC, SOA, …, đã cho ta những cách lập trình hiệu quả, tốt hơn rất nhiều, cho phép ta dễ dàn quản lý, tạo được những sản phẩm phức tạp hơn, khả năng sử dụng lại ( reuse ) cao hơn, tăng tốc độ code, giảm thời gian, công sức cho một dự án phần mềm.

Những giai đoạn đầu tiên, chúng ta tập trung nhiều vào backend, vì vậy các framework dành cho backend ra đời trước, như Kohana, Zend hoặc gần đây nhất là laravel. Tuy nhiên, càng về sau, khi backend đã đạt tới một mức độ nhất định, và UX quyết định phần lớn doanh thu của website, chúng ta bắt đầu có những yêu cầu phức tạp hơn cho frontend.  Cũng như backend, Từ các nhu cầu như vậy, ta thiết kế ra các thư viện, framework như jQuery, Backbone và mới nhất là AngularJS của Google. Đó là điều tuyệt vời đối với những project lớn.

Và trong bài viết ngày hôm nay, ta sẽ cùng nhau tìm hiểu về BackbonJS. Ta sẽ trả lời các câu hỏi đại loại như :

1. Backbone là gì ?
2. Nó gồm những thành phần gì ?
3. Tại sao ta nên sử dụng nó ?

Có lẽ câu hỏi thứ 3 ta đã trả lời ở phần trên rồi. Đúng không nhỉ ?

Cách tốt nhất để bạn có thể học nó là xem trên trang chủ. Tại đây có các mô tả chi tiết và đúng nhất, đồng thời có các ví dụ, hướng dẫn rất “chuẩn”, được viết hoặc chọn lọc bởi chính tác giả, còn bài viết này … phải nói sao nhỉ … chỉ giống như một bài viết “được được” mà bạn có thể tham khảo qua.

**BACKBONE LÀ GÌ ?**

*BackboneJs* là một thư viện/framework hỗ trợ lập trình viên tổ chức code tốt hơn, lập trình với dữ liệu và các DOM nhanh chóng, tiện lợi và có tổ chức hơn, Nó cho phép viết một client-app có cấu trúc, đẹp và gọn hơn với khá nhiều API giúp việc lập trình web application trở nên nhanh chóng và dễ dàng.

Có thể bạn đã biết đến AJAX, và các mô hình đại loại có thể mô tả là “cập nhật dữ liệu mà không refresh trang” hay có cách nói dí dỏm là “web động mà không động” ( nó làm mình nhớ câu truyện “Đẹp mà không đẹp” quá ). Bạn có thể hình dung là chúng ta sẽ sử dụng javascript để trao đổi dữ liệu giữa client và server, chỉ dữ liệu thôi nhé. Theo chiều từ client-server, thì các đoạn script ở client sẽ thu thập thông tin từ các input và truyền về cho server, đồng thời thay đổi cấu trúc html trên trang hiện tại. Giờ là chiều từ server-client, Server sẽ tiếp nhận dữ liệu, xử lý và trả dữ liệu về cho client, script trên client sẽ nhận dữ liệu và tổ chức, fill vào các đoạn code html và hiển thị cho người dùng. Như vậy, toàn bộ hai quá trình đề không hề tải lại các tài nguyên như css, html và js, giúp giảm thời gian chờ của người dùng, thời gian đáp ứng của server và tăng độ “đẹp” của UX.

Backbone hỗ trợ ta làm điều này một cách dễ dàng và gọn nhẹn hơn. Nó cung cấp một vài khái niệm như : model dùng để map với data trao đổi giữa server-client, View là các đoạn hml dùng để show ra trang web, khung sường để hiển thị dữ liệu. Đồng thời cũng cho phép đồng bộ dữ liệu giữa clien và server. Rất nhanh chóng và tiện lợi. Vì nó đã viết sẵn các phương thức cần thiết, ta chỉ việc … ăn thôi.

Chúng ta có một vài “class” tiêu biểu như sau :

**Model** : Giống như linh hồn của app, model chứa dữ liệu, những dữ liệu này được dùng để trao đổi với server, được fill vào các View và show ra trang html, cũng như tính toán dữ liệu đề dựa trên nó.

**View** : Nếu model là linh hồn, thì view chính là thể xác, là phần da bọc bên ngoài, nó cho phép kết nối giữa model và html bên ngoài, fill dữ từ model vào các template, tạo thành code hml rồi show nó.

**Collection** : Có model thì cũng cần phải có cái gì đó để chứa và xử lý một danh sách các model này chứ đúng không, Collection được tạo ra là để làm điều đó.

**Event** : Backbone cũng là javascript, và nó hỗ trợ Event, cho phép bạn gọi thực hiện các thao tác trên model, view hoặc collection khi một sự kiện nào đó được gọi, bên cạnh các sự kiện bình thường như click, move,…, bạn cũng có thể tạo các event mới tùy ý.

**Sync** : Đây có lẽ là không thể thiếu đúng không, nó cho phép đồng bộ dữ liệu giữa client và server chỉ với phương thức save(). Ta chỉ việc khai báo endpoint, và gọi phương thức save() là được. Mà không cần phải sử dụng ajax ở khắp nơi nữa.

Và còn nhiều “class” khác mà bạn có thể tham khảo tại trang chủ, tin rằng bạn sẽ thích.

**CÁCH CHÈN BACKBONE VÀO TRANG HTML**

|  |  |
| --- | --- |
| 1  2  3  4 | <script src="lib/jquery-1.7.1.min.js"></script  <script src="lib/underscore-min.js"></script>  <script src="lib/backbone-min.js"></script>  <script src="js/myScript.js"></script> |

**BACKBONE.EVENTS**

Sử dụng javascript, nhất định bạn cũng đã biết ít nhiều về event, bạn có thể đọc lại tại đây ( [Javascript event](http://www.w3schools.com/js/js_events.asp" \t "_blank) ). Backbone cho phép bạn thêm một event bất kỳ và object bất kỳ, tùy chỉnh. Bạn có thể đọc thêm tại đây ( [Event](http://backbonejs.org/" \l "Events" \t "_blank) ).

Để xử lý được các event, một object cần có các phương thức thích hợp, để add các phương thức này vào object, bạn làm như sau :

|  |  |
| --- | --- |
| 1 | \_.extend(object, Backbone.Events); |

Giờ giả sử ta muốn handle sự kiện tên là “alert”, ta chỉ việc sử dụng phương thức on() như thường dùng trong jQuery. mà thực tế là vậy :

|  |  |
| --- | --- |
| 1  2  3 | object.on("alert", function(msg) {    alert("Triggered " + msg);  }); |

Sự kiện này không được gọi bởi hành động của user, không như khi click, mouseover, submit, …, vì vậy nó phải được gọi bằng trigger :

|  |  |
| --- | --- |
| 1 | object.trigger("alert", "an event"); |

**BACKBONE.MODEL**

Dữ liệu ở server là có cấu trúc, ta lưu trong database, và khi get nó ra hì a có các array, object. Và khi chuyển về client, theo cách thông thường, ta có thể chuyển qua dạng JSON, XML hoặc paintText. Tuy nhiên, khi đến client, nó không có một định dạng cố định, ngoài JSON, tuy nhiên, JSON thì lại không có nhiều phương thức hỗ trợ lắm. Vì vậy Backbone đã phát triển nó thành Model. Cho phép map các dữ liệu nhận được từ server, và sử dụng trong client như dữ liệu trên server.

Không chỉ như JSON, “Class” Model hỗ trợ khá nhiều phương thức hữu ích.

Để tạo một Model :

|  |  |
| --- | --- |
| 1  2  3 | var MyModel = Backbone.Model.extend({     //...  }); |

Để đặt các giá trị mặc định cho model, ta sử sẽ định nghĩa phương thức defaults(). Bạn cũng nên khai báo tất cả các property của model trong phương thức này, để tiện xác định tên các property :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var MyModel = Backbone.Model.extend({  defaults: {      "name":  "noname",      "age":     -1,      "edu":    ""    }  }); |

Để tạo một instance của model trên, ta gọi như cách tạo instance bình thường bằng toán tử new :

|  |  |
| --- | --- |
| 1 | var myModel = new MyModel(); |

Lúc này phương thức defaults được gọi, và đặt giá trị mặc định cho các property. Nếu muốn set lại các giá trị mới ta dùng phương thức set() :

|  |  |
| --- | --- |
| 1 | myModel.set("name", "New Name"); |

Cũng tương tự, để get giá trị, ta dùng get(“tên property”).

Bây giờ là phần thú vị, chúng ta sẽ handle sự kiện change giá trị của model :

|  |  |
| --- | --- |
| 1  2  3 | myModel.on("change:name", function(model, name){       alert(name);  }); |

Đây là phần mình rất thích, không còn phải làm như ngày xưa, mối khi change dữ liệu, cũng phải viết luôn code để cập nhật lại text ở code html ( show cho người dùng ), rồi khi có thay đổi, lại phải search từng dòng code để sửa, Còn bây giờ khác rồi, chỉ cần thay đổi giá trị của property, phương thức xử lý code html ( show cho người dùng ) sẽ tự động được gọi. Rất tiện. Nó rất giống với trong WPF, và android.

Phần thú vị và tiện lợi nhất của Backbone.Model là phần sync. Nó cho phép tự động đồng bộ dữ liệu giữa server và client, dựa vào phương thức ajax và endpoint url. Chúng ta chỉ cần gọi các phương thức, save, insert, delete … Chúng ta sẽ nói kỹ về Sync ở phần sau.

Để save model :

|  |  |
| --- | --- |
| 1 | myModel.save(); |

**BACKBONE.COLLECTION**

Collection là một tập hợp nhiều model, nó thêm cho object khá nhiều phương thức làm việc với collection. Tham số quan trọng và bắt buộc đối với collection là model.

|  |  |
| --- | --- |
| 1  2  3 | var myCollection = Backbone.Collection.extend({    model: MyModel  }); |

“Class” này cũng như model, cũng có các phương thức get, set, …, ngoài ra còn có phương thức where, dùng để truy vấn các phần tử,  Ví dụ ta muốn lấy một model có name là Được :

|  |  |
| --- | --- |
| 1 | var found = myCollection.where({name: "Được"}); |

Nó cũng hỗ trợ các phương thức của underscore.

**BACKBONE.SYNC**

Như đã nói trong các phần trước, đây là chức năng mà mình rất thích, nó cho phép đồng bộ dữ liệu trong model với data trên server.

Để có thể sync, ta không phải chỉ gọi save(), delete() là đủ, mà trong các property của modle, collection phải có property là url. đây là đường dẫn endpoint.

Bạn đọc thêm về sync tại : [Backbone Tut](http://backbonejs.org/docs/backbone.html)

**BACKBONE.VIEW**

Nó là View, nhưng bản chất và cách nó hoạt động thì lại giống với controler trong mô hình model-view-controler hơn. Trong nó có hai thành phần chính là data ( model hoặc collection ) và view ( template ). Template là code html. được đặt trong thẻ script.

Mỗi View không tồn tại một mình, nó cần được đính vào một element nào đó trong code html, Có nghĩa là trước khi nghĩ đến view, bạn cần phải xác định view đó nằm ở chỗ nào, sau đó tạo một element và cho nó một id, lấy id này khai báo trong view, Ví dụn trong code html của trang, có một element như sau :

|  |  |
| --- | --- |
| 1 | <div id="myViewEl"></div> |

Và trong js, tạo view như sau :

|  |  |
| --- | --- |
| 1  2  3  4 | var MyView = Backbone.View.extend({    el: 'myViewEL'  template : $('#templateId')  }); |

Giờ để sử dụng view này :

|  |  |
| --- | --- |
| 1 | new MyView() |

**TEMPLATE**

Đây chính là template của underscore. bạn có thể đọc tại đây ( [underscore template](http://underscorejs.org/#template) ). Bạn đặt các lệnh javascript trong thẻ <%  %> và để xuất giá trị các biến, bạn sử dụng  <%=myVar %>. Dưới đây là một ví dụ đơn giản.

|  |  |
| --- | --- |
| 1 | <script type="template" id="templateId"><div class="<%=mainClass%">><%=text%></div></template> |

Để truyền dữ liệu và biến template thành code html, ta sử làm như sau :

|  |  |
| --- | --- |
| 1 | var html = \_.template(this.template.html(), myModel.toJSON()) |

Bước tiếp theo là sử dụng code jquery để append html vào view el.

**KHAI BÁO APP**

Sẽ không có tác dụng gì, và code vẫn sẽ rất rối nếu chúng ta chỉ khai báo các biến rồi cho nó extend model hay view. Để tổ chức code tốt, và có tính global. có thể gọi và sử dụng ở những file khác nhau, ta cần phải khai báo một object toàn cục và xem nó có vai trò là App role :

|  |  |
| --- | --- |
| 1 | var MyApp = MyApp || {}; |

Đoạn code này bạn có thể đặt ở dòng đầu tiên của những file có sử dụng tới model, view trong app. Và tất nhiên rồi, ta không khai báo một model, view riêng lẽ, mà ta sẽ làm như sau :

|  |  |
| --- | --- |
| 1 | MyAPP.model.myModel = new MyModel(); |

**KẾT LUẬN**

Dù vậy, mô hình MVC của Backbone không phải là model – view – controler, Nó thiếu controler, vì vậy khi làm việc nhiều, bạn sẽ nhận thấy có sự nhập nhằn, vì các  thao tác làm trong controler giờ được trải rải rác vào model và view. Như trong ví dụ trên, bạn đã thấy chúng ta handle sự thay đổi của model ngay trong model, và trực tiếp thay đổi view. Điều đó cũng chưa thực sự đẹp lắm.

Trong tương lai, mình sẽ giới thiệu đến các bạn một framework đẹp hơn, được viết bởi google, nó là AngularJS, nếu bạn thích backbone thì nhất định bạn cũng sẽ rất thích AngularJS. hẹn gặp lại bạn ở bài viết này. Còn bây giờ, hãy bắt đầu làm những điều tuyệt vời với Backbone trước đã nhé.

[BACKBONE LÀ GÌ ?](http://taichuong.com/index.php/2016/11/24/backbone-la-gi/)

[**BACKBONE.JS FROM BEGINNER TO … (PART1)**](http://taichuong.com/index.php/2016/11/24/backbone-js-from-beginner-to-part1/)

[24/11/2016](http://taichuong.com/index.php/2016/11/24/backbone-js-from-beginner-to-part1/) [chuongnt](http://taichuong.com/index.php/author/chuongnt/)

**Introduction**

Bạn thực sự thấy mô hình này quen thuộc ?

Vậy còn đây thì sao ?

Nếu bạn từng đọc qua về Backbone.js thì hai mô hình này hẳn là rất quen thuộc. Tuy nhiên một lần nữa tôi cũng muốn giới thiệu lại về Backbone.js.

Để bắt đầu tìm hiểu Backbone.js tôi cần phải biết Backbone.js là gì. Ngay trong chính website của Backbone họ cũng khẳng định rằng Backbone.js không phải là một framework mà là một library của Javascript . Sự khác nhau giữa library và framework là gì? Sử dụng library bạn là người điều khiển, sử dụng framework bạn là người bị điều khiển. Như thế có nghĩa rằng bạn có thể nhúng Backbone vào các apps của bạn mà không phải gặp vấn đề gì.

Ok. Vậy đã một phần định nghĩa cho việc BackBone.js là gì. Vậy điều gì tiếp theo là gì. Thế mạnh của Backbone.js là gì. Và khi nào sử dụng nó ?

Chắc hẳn nếu bạn đã từng sử dụng Javascript, jQuery thì việc vướng mắc với một đống lộn xộn như jQuery selector, callbacks, và việc đồng bộ giữa HTML và UI là việc gây đau đầu nhất. Vì lý do vậy mà bạn có thể ngó ngàng tới Backbone.js như một option và một ứng cử viên sáng giá (đặc biệt single-page application).

Một điều khi bạn sử dụng Backbone.js rằng : mặc dù những khái niệm mà Backbone.js chỉ đề cập tới Model và View mà không hề đề cập tới controller trong mô hình MVC. Nhưng Backbone.js thực sự có nhiều khía cạnh của mô hình MVC. Về mô hình MVC trong Backbone sẽ được đề cập trong bài viết khác.

<h3><b>Backbone.js basic</b></h3>

Có quá nhiều điều về Backbone.js cần phải biết. Nhưng những định nghĩa cơ bản là điều không thể bỏ qua.

<h4>Models & Views</h4>

 Một lần nữa mô hình lại được lặp lại phải không. Thực sự không cần giải thích nhiều về mô hình hoạt động chung nữa.

Ngoài ra có một số điểm ở Model và View nên biết:

Models (trong bài này sẽ không đi chính vào Model trong Backbone.js)

* Hàm Initialization của model được gọi khi một model được khởi tạo
* Một số khái niệm cơ bản trong model : getter & setter, initialization, default values và Direct access ( như listen và change model, validation model )

Views

* Trong view bạn cần phải biết đến việc : tạo mới view và một khái niệm đặc biệt el (element)
* el : là một trong số central property của view. el chỉ một Dom element mà View nào cũng phải có. Có hai cách để tạo el trong View : sử dụng Dom element có sẵn(ex1) và tạo mới element(ex2)

Ex1:

<ul id="notes" class="container"></ul>

<script type="text/javascript">

var NotesView = Backbone.View.extend({

el: '#notes'

});

var notesView = new NotesView();

console.log(notesView.el);

</script>

Ex2:

<script type="text/javascript">

var ProductsView = Backbone.View.extend({

tagName: 'ul',

className: 'container',

id: 'products'

});

var productsView = new ProductsView();

console.log(productsView.el);

</script>

Không chỉ có vậy, View còn chứa đựng các hàm cũng như là các khái niệm khác mà cũng cần được chú ý như :

* setElement: nhằm mục đích thay đổi el khi cần thiết
* var note1 = $('<div>El1</div>');
* var note2 = $('<div>El3</div>');
* var NotesView = Backbone.View.extend({
* events: {
* click: function(e) {
* console.log(e)
* console.log(notesView.el === e.target);
* }
* }
* });
* var notesView = new NotesView({
* el: note1
* });
* notesView.setElement(note2);
* note1.trigger('click');

note2.trigger('click');

Render function: function này tuy không phải là một function mới. Tuy nhiên hãy xem ví dụ dưới đây:

* var app = {};
* var AppView = Backbone.View.extend({
* el: $('#container'),
* template: \_.template('<h1>Hello <%= name %></h1>'),
* initialize: function() {
* this.render();
* },
* render: function() {
* this.$el.html(this.template({name: 'world'}))
* }
* });
* app.views = new AppView();

 Note: *Chắc hẳn sẽ có một chút khó hiểu tại đây:*

\_.template('<h1>Hello <%= name %></h1>')

*Đây là một cú pháp được quy định bởi một thư viện được gọi là :*

<a href=”http://underscorejs.org/?utm\_source=adrianmejia.com”><i>underscore.js</i></a>

*Một chút giải thích trong function render: Bạn có thể dùng <%= %> hoặc <%- %> trong template. Thậm chí chèn javascript bằng cách dùng <% %>*

Giờ thì hãy chú ý vào render function. Như bạn biết, Backbone.js không sử dụng HTML markup. Thay vì đó nó đã dùng template để thay thế. Tại đây tôi sử dụng Underscore’s micro-template. Một lần nữa Backbone khẳng định mình không phải là framework. bạn có thể sử dụng lại các template của các framework js khác tại đây. Bây giờ bạn đã có một trang Hello world.

* The events hash: Backbone cho phép thêm các sự kiện vào nhưng el nhất đinh. và thường có cấu trúc một cặp key-value như ví dụ bên dưới.
* <script type="text/javascript">
* var note1 = $('<div class="toggle">El1</div>');
* var NotesView = Backbone.View.extend({
* el: note1,
* events: {
* 'click' : 'toggleCallback'//key-value
* },
* toggleCallback: function() {
* console.log('You\'ve click on .toggle ');
* }
* });
* var notesView = new NotesView();
* note1.trigger('click');

</script>

**Summary**

Như vậy chúng ta đã có một cái nhìn tổng quan về Backbone.js và có được cái nhìn chi tiết hơn của View trong Backbone.js. Chi tiết về các khái niệm : Model, Collection, Router sẽ được đề cập trong các bài viết sau.

Note : Các examples đã được đẩy lên Githup. Theo đường dẫn Backbone-Training/Lesson1. Link github: <https://github.com/emily0604/Backbone-Training>

Come and enjoy it!