AdVanced

The ng-repeat directive repeats a set of HTML, a given number of times.

The set of HTML will be repeated once per item in a collection.

The collection must be an array or an object.

The ng-show directive shows the specified HTML element if the expression evaluates to true, otherwise the HTML element is hidden.

10. Scope is the glue between application controller and the view. During the template [linking](https://docs.angularjs.org/guide/compiler) phase the [directives](https://docs.angularjs.org/api/ng/provider/$compileProvider#directive) set up [$watch](https://docs.angularjs.org/api/ng/type/$rootScope.Scope#$watch) expressions on the scope. The $watch allows the directives to be notified of property changes, which allows the directive to render the updated value to the DOM.

11. The initialize() method is called when a new instance of a model is created. Its use is optional; however you’ll see why it’s good practice to use it below.

**var** Todo = Backbone.Model.extend({

initialize: **function**(){

console.log('This model has been initialized.');

}

});

**var** myTodo = **new** Todo();

*// Logs: This model has been initialized.*

12. Backbone supports model validation through model.validate(), which allows checking the attribute values for a model prior to setting them. By default, validation occurs when the model is persisted using the save() method or when set() is called if {validate:true} is passed as an argument.

**var** Person = **new** Backbone.Model({name: 'Jeremy'});

*// Validate the model name*

Person.validate = **function**(attrs) {

**if** (!attrs.name) {

**return** 'I need your name';

}

};

*// Change the name*

Person.set({name: 'Samuel'});

console.log(Person.get('name'));

*// 'Samuel'*

*// Remove the name attribute, force validation*

Person.unset('name', {validate: **true**});

*// false*

Above, we also use the unset() method, which removes an attribute by deleting it from the internal model attributes hash.

Validation functions can be as simple or complex as necessary. If the attributes provided are valid, nothing should be returned from .validate(). If they are invalid, an error value should be returned instead.

13. **Fetching models from the server**

Collections.fetch() retrieves a set of models from the server in the form of a JSON array by sending an HTTP GET request to the URL specified by the collection’s url property (which may be a function). When this data is received, a set() will be executed to update the collection.

<https://addyosmani.com/backbone-fundamentals/>

# 14. **Backbone.sync**

It is a function that Backbone calls everytime to read or save the model to the server. It represents the state of the model.

Syntax

|  |
| --- |
| sync.(method, model, options) |

Parameters:

* *method:* It represents the CRUD operations such as create, read, update and delete.
* *model:* It includes the model to be saved.
* *options:* It fires success or error message depending on the method succeeded.

--------------------------------------------------------------------------------------------------------

15. Templates are essentially pieces of markup that you can use to create a bunch of different reusable copies of that markup (including all the styles and hierarchy that you want) but populating each component with different data.

By default, [**Backbone.js**](http://backbonejs.org/) uses the templating engine of [**Underscore.js**](http://underscorejs.org/) which makes sense because Backbone has a hard dependency on Underscore

17. **Model**

* 1. Manages the app data and state
  2. Not concerned with UI or presentation
  3. Often persists somewhere
  4. Same model should be reusable, unchanged in different interfaces

1. **View**  
   1. Present the Model to the user in an appropriate interface
   2. Allows user to manipulate data
   3. Does not store any data except to cache state
   4. Easily reusable & configurable to display different data
2. **Controller**  
   1. Intermediary between Model & View
   2. Updates the view when the model changes
   3. Updates the model when the user manipulates the view
   4. Typically where the app logic lives

**MVC - model view Controller**

18.  A view's render() function can be bound to a model's change() event, allowing the view to always be up to date without requiring a full page refresh.

19. el is just an identifier and it refers to an element, a DOM element, which is a convention in that library.

22. The emit() method provides a normalized mechanism for dispatching events. This method will dispatch an event using the native event facilities when available, and will emulate these facilities otherwise:

The signature of the method is:

| **Argument** | **Type** | **Description** |
| --- | --- | --- |
| target | Object | This is the target object (a DOM node or other event emitting object) that will be the source of the event. The target object may be a host object with its own event capabilities (like DOM elements or the window), or it may be a JavaScript object with an on() method. |
| type | String | This is the name of the event type to be dispatched (like select). This event may be a standard event (like click) or a custom event (like finished). |
| event | Object | This is an object with the properties of the event to be dispatched. Generally you should align your properties with W3C standards. Two properties are of particular importance:   * bubbles - This indicates that the event should bubble up, first firing on the target object, next on the target object’s parent (parentNode) and so on until it reaches the top of the DOM or bubbling is stopped. Bubbling is stopped when a listener calls event.stopPropagation(). * cancelable - This indicates that the event’s default action can be cancelled. The default action is cancelled by a listener by calling event.preventDefault(). The emit method does not perform any default action, it returns a value allowing the calling code to perform any default action. |

emit() returns the event object unless the event is cancelable and is cancelled by one of the listeners, in which case it will return false.

24. The dojoConfig object (known as djConfig prior to Dojo 1.6) is the primary mechanism for configuring Dojo in a web page or application. It is referenced by the module loader, as well as Dojo components with global options. It can further be used as a configuration point for custom applications, if desired.

27. There are three methods one can attach to the XHR Options object to determine what to do when the data comes back.

* load - executed when a successful Ajax call is complete. Is passed the data and an object of the XHR properties.
* error - executed when an Ajax call times out, or otherwise fails. Is passed the error and an object of the XHR properties.
* handle - combination of load and error callbacks, fired when either of the two conditions are met. In the success case, behaves just like load:, and in the failure case like error:

28. Dojo’s [event](https://dojotoolkit.org/reference-guide/1.7/quickstart/events.html#quickstart-events) system provides *dojo.disconnect* to cleanly remove event listeners created by [dojo.connect](https://dojotoolkit.org/reference-guide/1.7/dojo/connect.html" \l "dojo-connect). The more events in the page that exist, the more the browser (or otherwise) has to do.

34. Ext JS supports all major web browsers including:

* Internet Explorer 6+
* Firefox 3.6+ (PC, Mac)
* Safari 4+
* Chrome 10+
* Opera 11+ (PC, Mac)

## 58. Creating a New Application

The following command generates a new application with the namespace MyApp to "/path/to/www/myapp":

# Make sure the current working directory is the Sencha Touch SDK

cd /path/to/sencha-touch-sdk

sencha generate app MyApp /path/to/www/myapp

Or, you can specify the path to the SDK on the command line:

sencha -sdk /path/to/sencha-touch-sdk generate app MyApp /path/to/www/myapp