
ANGULARJS
by Google

About Me

- Chad Maughan
- @chadmaughan
- http://chadmaughan.com

Git

- Check your installation

```
$ git --version
git version 1.7.12.2
```

- Download http://git-scm.com/downloads
- Linux (Debian based)
  - `apt-get install git`
- OSX
  - `brew install git`
- Windows
  - Download, click stuff, confirm security exceptions, etc.

Node

- Check your installation

```
$ node --version
v0.8.11
```

- Download http://nodejs.org/download/
- Linux (Debian based)
  - `apt-get install node`
- OSX
  - `brew install node`
- Windows
  - download binaries, click more stuff, confirm, etc

Code

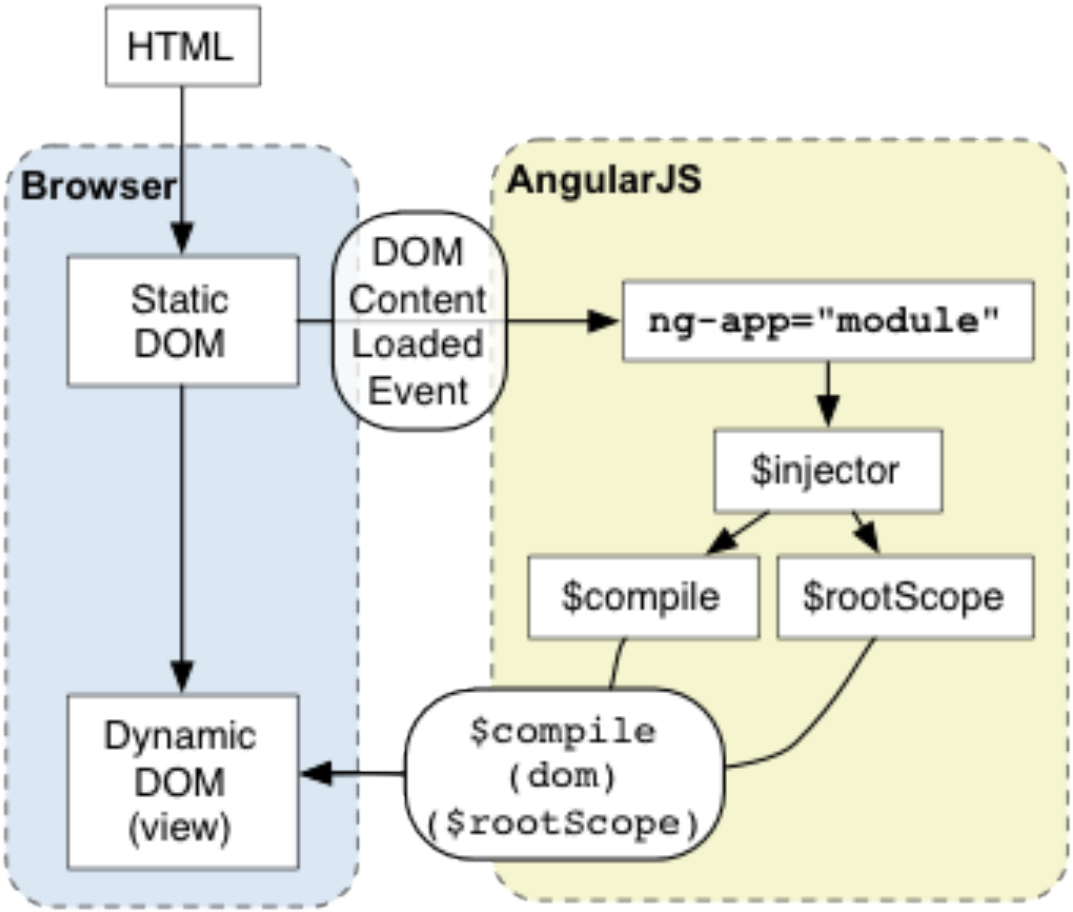
Download the sample app

```
$git clone
https://github.com/chadmaughan/
learn-angular
```

Differently “Open”

- Hosted separately at https://angularjs.org
  - (not https://developers.google.com)
- Code on Github
  - (not https://developers.google.com)

Overview


HTML
Browser
AngularJS
Static DOM
DOM Content Loaded Event
ng-app="module"
$injector
$compile
$rootScope
Dynamic DOM (view)
$compile (dom) ($rootScope)

Angular Namespace

To prevent accidental name collision, Angular prefixes names of objects which could potentially collide with $.

Please do not use the $ prefix in your code as it may accidentally collide with Angular code.

Directives

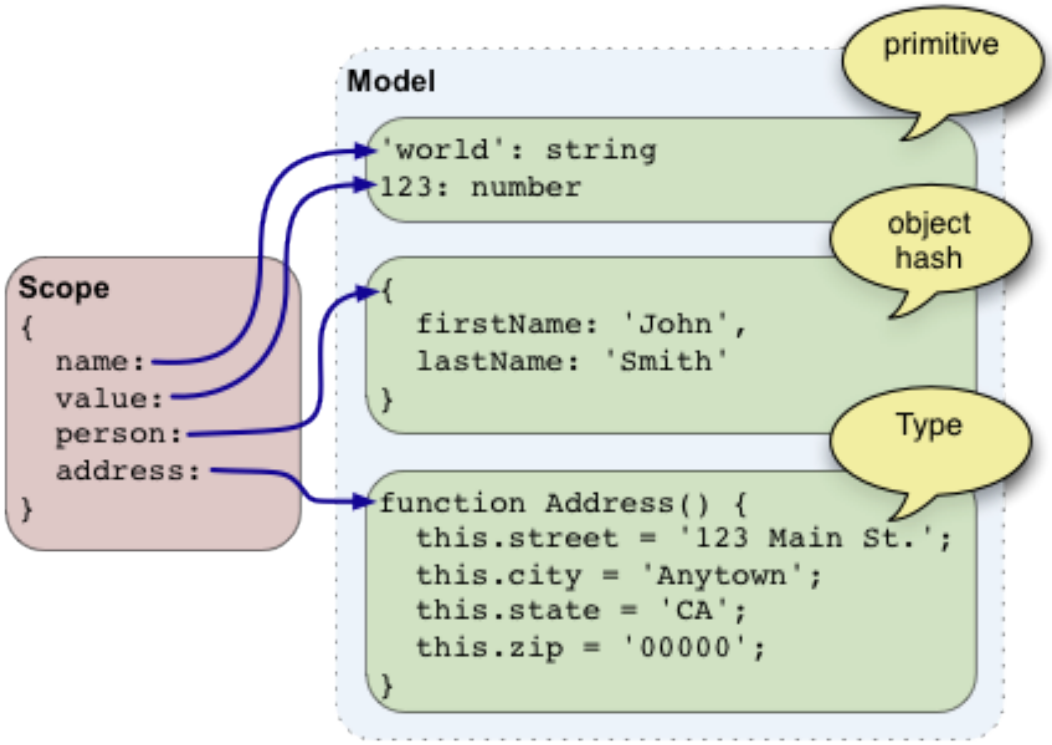
- Teach your HTML new tricks!
- Become reusable components
- Camel Case -> Snake Case

```
<div ng-controller="Ctrl1">
    Hello <input ng-model='name'>
    <span ng:bind="name"></span>
    <span ng_bind="name"></span>
    <span ng-bind="name"></span>
    <span data-ng-bind="name"></span>
    <span x-ng-bind="name"></span>
</div>
```

HTML Compiler

1. Parsed into DOM
   - needs to be valid HTML (unlike other templating solutions)
2. Traverses the DOM and matches directives
3. Link template with scope
   - Setup `watches`

Model (M of MVC)

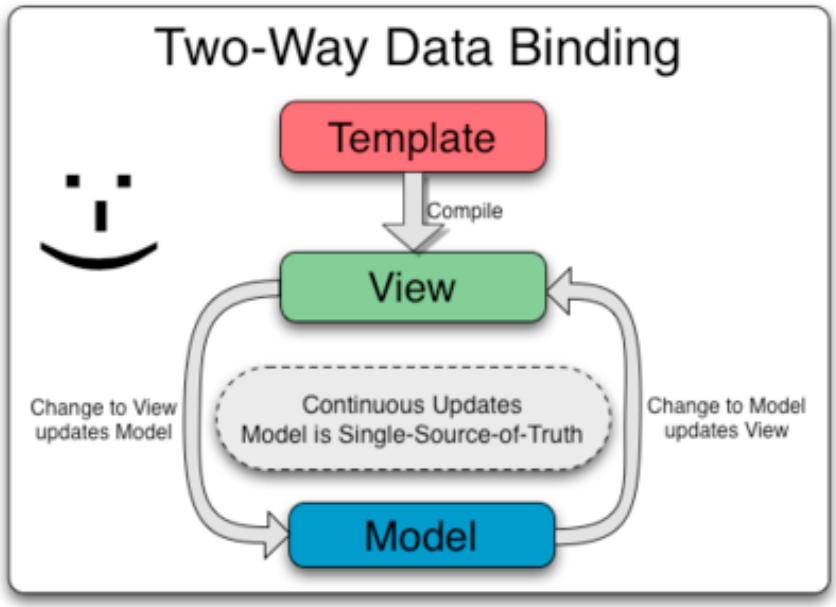


Scope

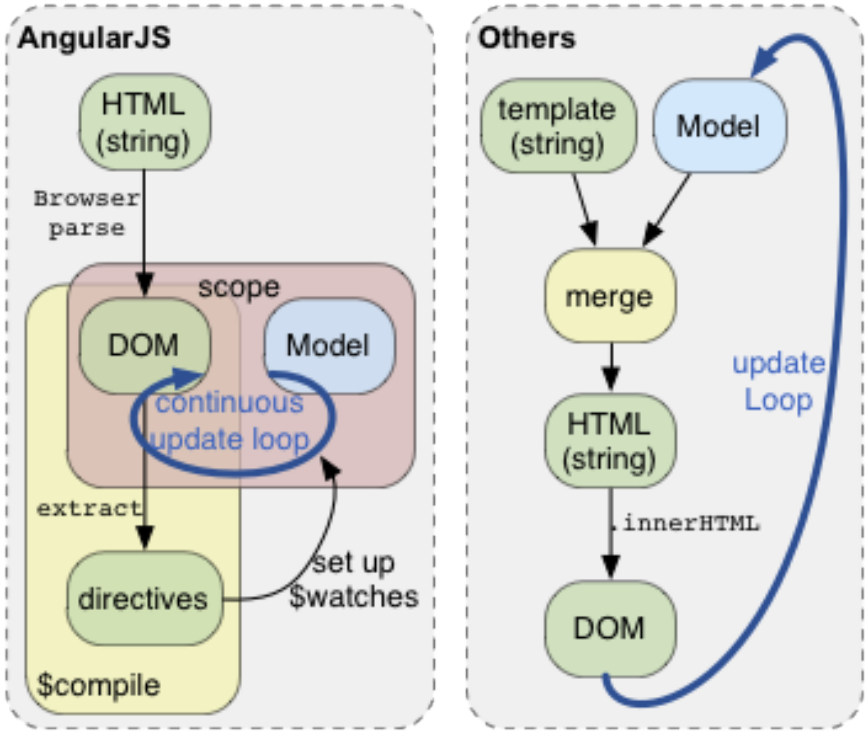
- Data model
- Glue between controllers and views
- Execution context for expressions
- Be nested to allow isolation of application components
- Provide watches for model mutation

Data Binding

- Model is “single source of truth”
- When model changes, view updates
- No DOM manipulation necessary



View (V of MVC)


AngularJS
HTML (string)
Browser parse
scope
DOM
Model
continuous update loop
extract
directives
set up $watches
$compile
Others
template (string)
Model
merge
HTML (string)
innerHTML
DOM
update Loop

Templates

- It is the static DOM, containing HTML, CSS, and angular-specific elements and angular-specific element attributes.
- Contains:
  - Directives
  - Markup (i.e. - {{ … }})
  - Filter
  - Form controls

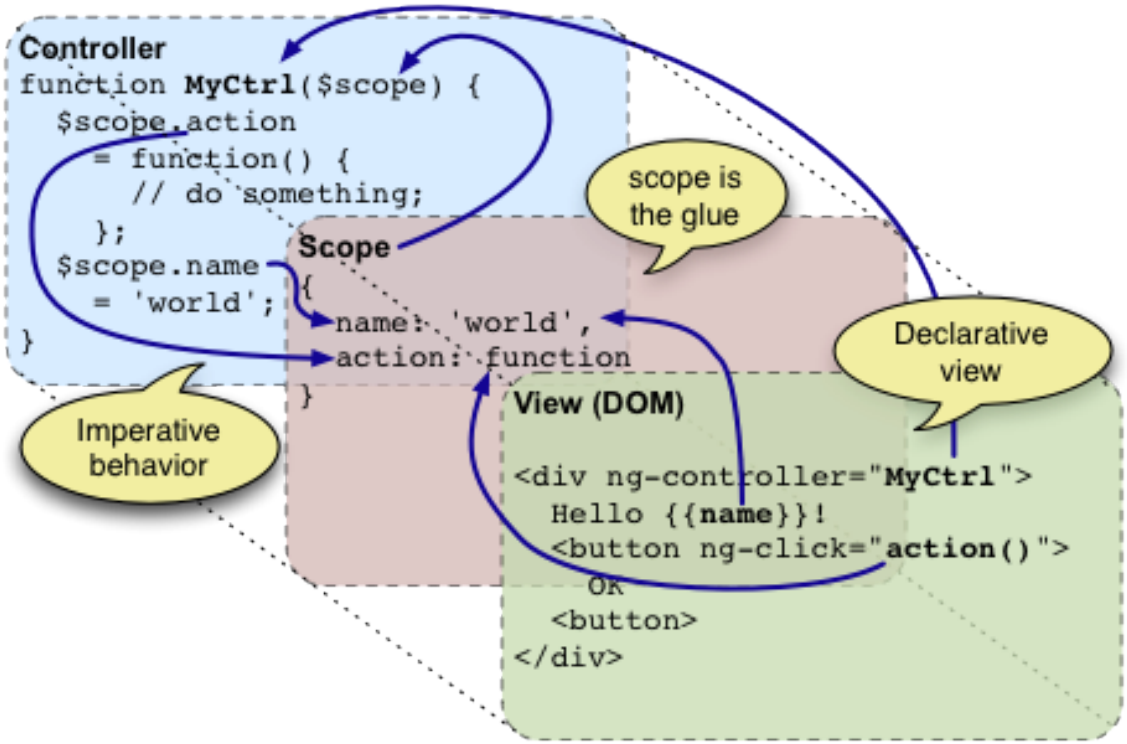
Expressions

- Processed by $parse
- Used with {{ expression }}
  - {{ 1 + 2 }}
  - {{ hello }}
- No control flow
- Not JS expression
  - i.e. doesn't use eval()

Filters

- Formats data for display to the user
- Chain filters
  - {{ name | uppercase }}
- Available filters
  - json, lowercase, uppercase, currency, number, date,

Controller (C of MVC)



Controllers

- Add in js/controllers.js
- Only business logic for associated view
- Keep small
  - Move complex stuff to services

Controller DON'TS

- Any kind of DOM manipulation
- Input formatting (use form controls instead)
- Output filtering (use filters instead)
- Run stateless or stateful code shared across controllers (use services instead)
- Instantiate or manage the life-cycle of other components (for example, to create service instances).

Services

- Instantiated “lazily”
- Are application singletons
- Add in js/services.js
- Register with `$provide`

```
$provide.factory('serviceId', function() {
   // return function that constructs
});
```

Dependency Injection

- Law of Demeter – Principle of Least Knowledge
- Three ways to get a dependency
  - The dependency can be created, typically using the new operator.
  - The dependency can be looked up by referring to a global variable.
  - The dependency can be passed in to where it is needed.

Injecting a Dependency

- Due to dynamic nature of JS, controllers explicitly identify dependencies

```
myCtrl.$inject = [‘$location’]

// typically done by bootstrapping
var injector = angular.injector(‘myMod’);
var greeter = injector.get('greeter');
```

Routes

- Maps URLs to controllers and views (partials)
- Examples:

```
$routeProvider.when('/states',
{templateUrl:'partials/states.html',
controller:StateListCtrl});

$routeProvider.otherwise({redirectTo:'/states'});
```

Unit Testing

- Dependency injection helps immensely
- Not required, but tutorial uses Jasmine BDD
- Example

```
it('should set value of order model', function() {
 expect(scope.order).toBe('age');
});
```

Questions

- Minification
- Pre-compilation of HTML partials
- Integration with other popular libraries
  - jQuery
  - Require.js
  - Stylus
- Other testing tools
  - Buster.js
- Touch devices

Seed App

- Application skeleton contains all libraries and gets you up to speed quickly
  - Download
    - `git clone` https://github.com/angular/angular-seed
  - Run
    - `node scripts/web-server.js`

Let’s Build

Download the sample app

```
$git clone
https://github.com/chadmaughan/
learn-angular
```