AngularJS

Unit Testing AngularJS Directives with Karma & Jasmine

Directives

- Directives are different from other components
  - they aren't used as objects in the JavaScript code
  - They are used in HTML templates of the application

- We write directives to perform DOM manipulations
  - we can't ignore them in unit tests as they play an important role
  - They directly affect the usability of the application

- Directives are the most important and most complex components in AngularJS

About Testing Directives

- Testing directives is tricky
  - They are not called like a function
  - They are declaratively applied on the HTML template
  - Their actions are executed when the template is compiled and a user interacts with the directive

- When performing unit tests
  - We need to automate the user actions
  - We need to manually compile the HTML in order to test the functionality of the directives

Set up Objects to test Directive

- Need to get references to objects needed before starting to test a directive

- The key object to create is **an element containing the directive** to be tested

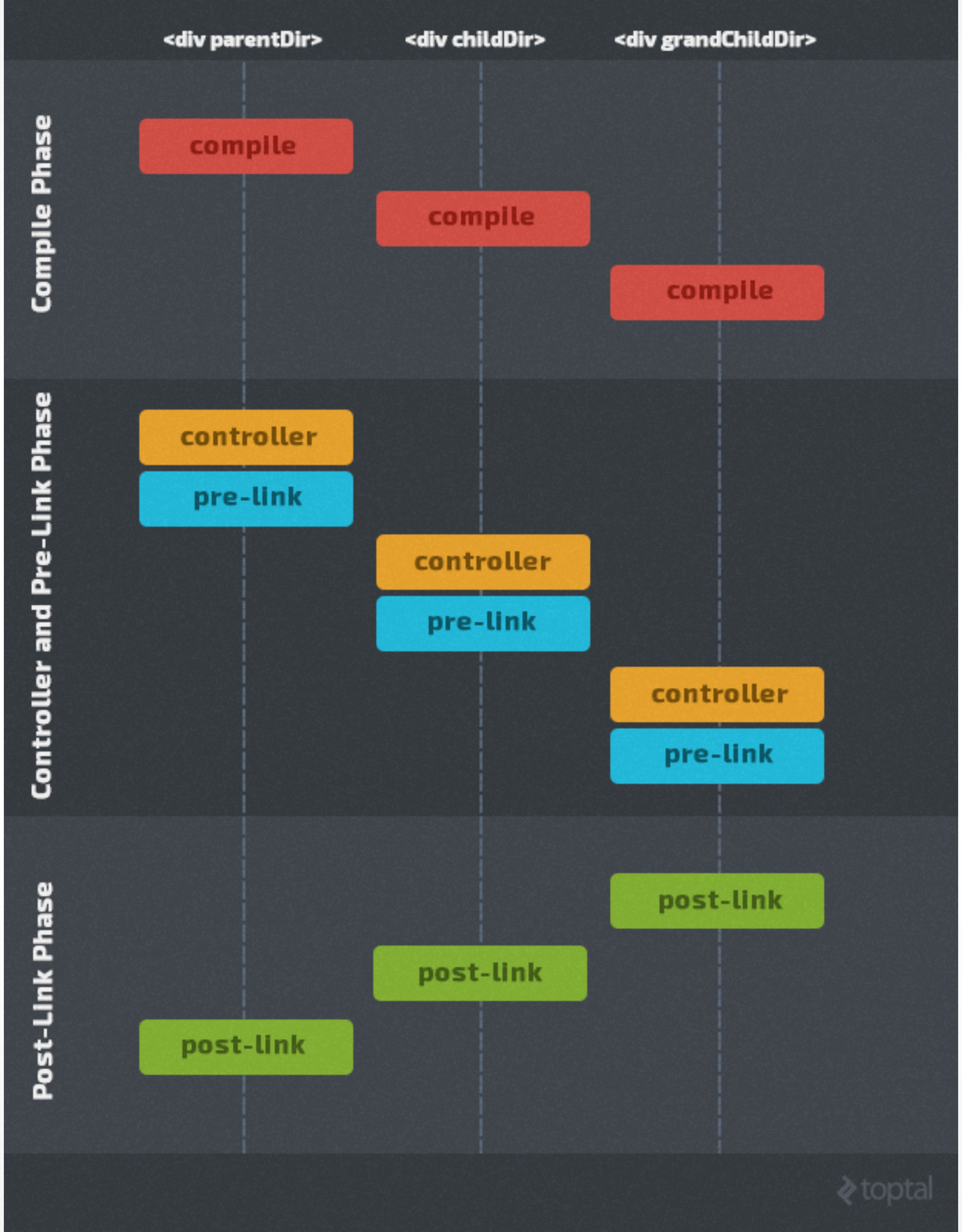
- We need to compile a piece of HTML **with the directive specified in** it to get the directive into action
    - On compilation of the HTML, the lifecycle of the directive is kicked in

Directive life cycle

- In a directive's life cycle, there are four distinct functions that can execute if they are defined
- Each enables the developer to control and customize the directive at different points of the life cycle.
- The **compile function** allows the directive to manipulate the DOM before it is compiled and linked thereby allowing it to add/remove/change directives, as well as, add/remove/change other DOM elements.

Directive life cycle (2)

- The **controller function** facilitates directive communication. Sibling and child directives can request the controller of their siblings and parents to communicate information.
- The **pre-link function** allows for private $scope manipulation before the post-link process begins.
- The **post-link method** is the primary workhorse method of the directive.


<div parentDir>
<div childDir>
<div grandChildDir>
Compile Phase
compile
compile
compile
Controller and Pre-Link Phase
controller
pre-link
controller
pre-link
controller
pre-link
Post-Link Phase
post-link
post-link
post-link
toptal

Getting Ready to test

- Compile HTML Template containing directive
  - Inject $compile service
  - Use $compile service to manually compile HTML
  - Compilation triggers directive life cycle

- Initiate next digest cycle
  - After next cycle, directive object would be in the same state as it appears on a page
  - Will want to initiate a digest cycle after making changes to the DOM or model

Things to test in Directive

- Testing Directive's Template
- Testing Link Function
  - DOM Manipulation
  - Watchers
  - DOM Events
- Testing Directive's Scope
- Testing Require
- Testing Replace

Setting up Karma

- install karma for development purposes

```
$ npm install --save-dev karma
```

- Install preprocessor to load html templates

```
$ npm install karma-ng-html2js-preprocessor --save-dev
```

- Install karma plug-ins to enable us to use Jasmine test framework and Google Chrome as target browser

```
$ npm install jasmine-core karma-jasmine
  karma-chrome-launcher --save-dev
```

Configuring Test Runner

- Create a configuration file for the karma settings

```
$ karma init karma.conf.js
```

- You will be asked several questions
- Accept the defaults to as many as you can
- **Answer NO for the RequireJS question**
- **Will fill in the source and test files section manually**
- The config file called *karma.conf.js* will be created
- Will use cofig file to run run tests from the terminal

angular-mocks to mock services

- Install angular-mocks to inject and mock Angular services into your unit tests

```
$ npm install angular-mocks --save-dev
```

Files section of config file

```
// list of files / patterns to load in the browser
files: [
        // Libraries
        'node_modules/jquery/dist/jquery.min.js',
        'node_modules/angular/angular.js',
        'node_modules/angular-mocks/angular-mocks.js',
        'node_modules/angular-route/angular-route.js',
        'node_modules/angular-resource/angular-resource.js',
        'node_modules/angular-messages/angular-messages.js',

        // Our app
        'src/app.js',
        'src/js/*.js',

        // tests
        'tests/*Spec.js',

        // Templates
        'views/*.html'
],
```

Preporcessor section of config file

```
preprocessors: {
     // Generate js files from html templates
     'views/*.html': 'ng-html2js'
},
```

Running unit tests

- Start test runner by issuing following command

```
$ karma start karma.conf.js
```

- Expect tests to fail (none written) & fix fixable errors
- Optimization: update the *package.json* manifest with *scripts* section to run karma

```
$ npm test
```

Scripts section of manifest

```
"scripts": {
     "test": "karma start karma.conf.js”
},
```

Testing AngularJS directives

- Create a test suite with **describe**.
  - The string parameter should include the name of the directive being tested.
  - The function parameter is the block of code that implements the suite
- Use **beforeEach** to load the module that contains the directive being tested and the HTML templates.
- Inject the **$compile** and **$rootScope** services in a **beforeEach** block
  - That allows you to create a new $scope and access the $compile service.
  - Compile an element containing the directive to be tested
  - Initiate the digest cycle of the $scope

Testing AngularJS directives (2)

- Now that everything is setup, we can ***spec*** out tests using the **it** function.
  - String parameter is title of spec or description of what the spec is testing
  - Function parameter is the spec or test.
- Test functionality of code that we write
  - See slide entitled "Things to test in Directive"
- Each test should have 1 or more expectations
  - Might be wise to follow this testing paradigm: setup → run code → assert

Examples

- Walk through process of creating and running directive tests for sample application.

Resources

- http://www.sitepoint.com/angular-testing-tips-testing-directives/
- https://docs.angularjs.org/guide/directive
- http://www.toptal.com/angular-js/angular-js-demystifying-directives
- https://docs.angularjs.org/api/ng/function/angular.element
- http://stackoverflow.com/questions/15753009/how-to-inject-a-service-in-a-directive-unit-test-in-angularjs
- https://github.com/vojtajina/ng-directive-testing