Angular 2 Interview Questions

Introduction

Reading interview questions is one of the great way to learn and brush up the left-over concepts even if you are not preparing for the interview. In this article, we tried to touch most of the important concepts of Angular 2. We have also provided the external links/references for further reading.

Disclaimer

Reading this article does not guarantee, in any way, that you will be able to clear the interview in angular 2. Our sole purpose is to get you a reference for last minute revision along with further reading.

If you feel that some more topic needs to be covered, please let us know. We will add those in the article.

Questions

Explain the life cycle hooks of Angular 2 application

Angular 2 component/directive has lifecycle events, managed by @angular/core. It creates the component, renders it, creates and renders its children, processes changes when its data-bound properties change, and then destroys it before removing its template from the DOM. Angular provides a set of lifecycle hooks(special events) which can be tapped into this lifecycle and perform operations when required. The constructor executes prior to all lifecycle events. Each interface has a single hook method prefixed with ng. For example, *ngOnint* interface has *Oninit* method that must be implemented in the component.

Some of the events are applicable for both component/directives while few are specific to components.

* **ngOnChanges**: Responds when angular sets its data-bound property which receives the current and previous object values.
* **ngOnInit**: Initializes the component/directive after first ngOnChange triggers. This is most frequently used method to retrieve the data for the template from a back-end service.
* **ngDoCheck**: Detect and act upon changes occuring outside Angular context. It is called when every change detection run.
* **ngOnDestroy**: Cleanup just before Angular destroys the directive/component. Unsubscribe observables and detach event handlers to avoid memory leaks.

**Component-specific hooks:**

* **ngAfterContentInit**: Component content has been initialized
* **ngAfterContentChecked**: After Angular checks the bindings of the external content that it projected     into its view.
* **ngAfterViewInit**: After Angular creates the component’s view.
* **ngAfterViewChecked**: After Angular checks the bindings of the component’s view.

What are the advantages of using Angular 2 over Angular 1?

1. Angular 2 is a platform not only a language:
2. Better Speed and Performance: *No $Scope in Angular 2, AOT*
3. Simpler Dependency Injection
4. Modular, cross platform
5. Benefits of ES6 and Typescript.
6. Flexible Routing with Lazy Loading Features
7. Easier to Learn

How routing works in Angular 2.

Routing is a mechanism which enables user to navigate between views/components. Angular 2 simplifies the routing and provide flexibility to configure and define at module level (Lazy loading).

The angular application has single instance of the Router service and whenever URL changes, corresponding Route is matched from the routing configuration array. On successful match, it applies redirects and the router builds a tree of ActivatedRoute objects and contains the current state of the router. Before redirection, the router will check whether new state is permitted by running guards ([CanActivate](https://blog.thoughtram.io/angular/2016/07/18/guards-in-angular-2.html)). Route Guards is simply an interface method that router runs to check the route authorization. After guard runs, it will resolve the route data and activate the router state by instantiation the required components into <router-outlet> </router-outlet>.

**Further Reading:**

<https://www.codeproject.com/Articles/1164813/Angular-Routing>  
<https://vsavkin.com/angular-2-router-d9e30599f9ea#.kt4z1v957>

What are Event Emitters and how it works in Angular 2?

Angular 2 doesn’t have bi-directional digest cycle, unlike angular 1. In angular 2, any change occurred in the component always gets propagated from the current component to all its children in hierarchy. If the change from one component needs to be reflected to any of its parent component in hierarchy, we can emit the event by using Event Emitter api.

In short, EventEmitter is class defined in @angular/core module which can be used by components and directives to emit custom events.

@output() somethingChanged = new EventEmitter();

We use somethingChanged.emit(value) method to emit the event. This is usually done in setter when the value is being changed in the class.

This event emit can be subscribed by any component of the module by using subscribe method.

myObj.somethingChanged.subscribe(val) => this.myLocalMethod(val));

**Further Reading:**

<http://stackoverflow.com/questions/36076700/what-is-the-proper-use-of-an-eventemitter>

<https://angular.io/docs/ts/latest/api/core/index/EventEmitter-class.html>

What is the use of codelyzer in angular 2 application.

All enterprise applications follows a set of coding conventions and guidelines to maintain code in better way. Codelyzer is an open source tool to run and check whether the pre-defined coding guidelines has been followed or not. Codelyzer does only static code analysis for angular and typescript project.

Codelyzer runs on top of tslint and its coding conventions are usually defined in tslint.json file. Codelyzer can be run via angular cli or npm directly. Editors like Visual Studio Code and Atom also supports codelyzer just by doing a basic settings.

To set up the codelyzer in Visual Studio code, we can go to File -> Preferences -> User Settings and add the path for tslint rules.

{

"tslint.rulesDirectory": "./node\_modules/codelyzer",

"typescript.tsdk": "node\_modules/typescript/lib"

}

To run from cli: ng lint.

To run from npm: npm run lint

**Further Reading:**

<https://github.com/mgechev/codelyzer>

<https://www.youtube.com/watch?v=bci-Z6nURgE>

What is lazy loading and How to enable lazy loading in angular 2?

Most of the enterprise application contains various modules for specific business cases. Bundling whole application code and loading will be huge performance impact at initial call. Lazy lading enables us to load only the module user is interacting and keep the rest to be loaded at runtime on demand.

Lazy loading speeds up the application initial load time by splitting the code into multiple bundles and loading them on demand.

Every Angular application must have one main module say AppModule. The code should be splitted into various child modules (NgModule) based on the application business case.

Plunkr Example: [Link](https://plnkr.co/edit/PNwh0Mn2ZJighpSoOTtw?p=info)

1. We don't require to import or declare lazily loading module in root module.
2. Add the route to top level routing (app.routing.ts) and set loadChildren. loadChildren takes absolute path from root folder followed by #{ModuleName}. RouterModule.forRoot() takes routes array and configures the router.
3. Import module specific routing in the child module.
4. In the child module routing, specify path as empty string ' ', the empty path. RouterModule.forChild again takes routes array for the child module components to load and configure router for child.
5. Then, export const routing: ModuleWithProviders **= RouterModule.forChild**(routes);

What are the security threats should we be aware of in angular 2 application?

Just like any other client side or web application, angular 2 application should also follow some of the basic guidelines to mitigate the security risks. Some of them are:

1. Avoid using/injecting dynamic Html content to your component.
2. If using external Html, that is coming from database or somewhere outside the application, sanitize it.
3. Try not to put external urls in the application unless it is trusted. Avoid url re-direction unless it is trusted.
4. Consider using AOT compilation or offline compilation.
5. Try to prevent XSRF attack by restricting the api and use of the app for known or secure environment/browsers.

**Further Reading:**

[https://angular.io/docs/ts/latest/guide/security.html#!#best-practices](https://angular.io/docs/ts/latest/guide/security.html#!)

How would you optimize the angular 2 application for better performance?

Well, optimization depends on the type and size of application and many other factors. But in general, I would consider the following points while optimizing the angular 2 app:

1. Consider AOT compilation.
2. Make sure the application is bundled, uglified, and tree shaking is done.
3. Make sure the application doesn’t have un-necessary import statements.
4. Make sure that any 3rd party library, which is not used, is removed from the application.
5. Have all dependencies and dev-dependencies are clearly separated.
6. I would consider lazy loading instead of fully bundled app if the app size is more.

**Further Reading:**

<https://medium.com/@areai51/the-4-stages-of-perf-tuning-for-your-angular2-app-922ce5c1b294#.pw4m2srmr>

<https://www.lucidchart.com/techblog/2016/05/04/angular-2-best-practices-change-detector-performance/>

How would you define custom Typings to avoid editor warnings?

Well, in most of the cases, the 3rd party library comes with its own .d.ts file for its type definition. In some cases, we need to extend the existing type by providing some more properties to it or if we need to define additional types to avoid Typescript warning.

If we need to extend the type definition for external library, as a good practice, we should not touch the node\_modules or existing typings folder. We can create a new folder, say “custom-typings” and keep all customized type definition in that.

To define typings for application (JavaScript/Typescript) objects, we should define interfaces and entity classes in models folder in the respective module of the application.

For those cases, we can define or extend the types by creating our own “.d.ts” file.

**Further Reading:**

<https://www.typescriptlang.org/docs/handbook/declaration-merging.html>

<https://typescript.codeplex.com/wikipage?title=Writing%20Definition%20%28.d.ts%29%20Files>

<http://stackoverflow.com/questions/32948271/extend-interface-defined-in-d-ts-file>

What is shadow DOM? How is it helping Angular 2 to perform better?

Shadow DOM is a part of the HTML spec which allows developers to encapsulate their HTML markup, CSS styles and JavaScript. Shadow DOM, along with a few other technologies, gives developers the ability to build their own 1st class tags, web components and APIs just like the <audio> tag. Collectively, these new tags and APIs are referred to as Web Components. Shadow DOM provides better separation of concern along with lesser conflict of styles and scripts with other HTML DOM elements.

Since shadow DOM are static in nature, it’s a good candidate to be cached as it is not accessible to developer. The cached DOM would be rendered faster in the browser providing better performance. Moreover, shadow DOM can be managed comparatively well while detecting the change in angular 2 application and re-paint of view can be managed efficiently.

**References/Further Reading:**

<https://developer.mozilla.org/en-US/docs/Web/Web_Components/Shadow_DOM>

<https://glazkov.com/2011/01/14/what-the-heck-is-shadow-dom/>

<https://code.tutsplus.com/tutorials/intro-to-shadow-dom--net-34966>

What is AOT compilation?

AOT compilation stands for Ahead Of Time compilation, in which the angular compiler compiles the angular components and templates to native JavaScript and HTML during the build time. The compiled Html and JavaScript is deployed to the web server so that the compilation and render time can be saved by the browser.

**Advantages**

1. Faster download: Since the app is already compiled, many of the angular compiler related libraries are not required to be bundled, the app bundle size get reduced. So, the app can be downloaded faster.
2. Lesser No. of Http Requests: If the app is not bundled to support lazy loading (or whatever reasons), for each associated html and css, there is a separate request goes to the server. The pre-compiled application in-lines all templates and styles with components, so the number of Http requests to the server would be lesser.
3. Faster Rendering: If the app is not AOT compiled, the compilation process happens in the browser once the application is fully loaded. This has a wait time for all necessary component to be downloaded, and then the time taken by the compiler to compile the app. With AOT compilation, this is optimized.
4. Detect error at build time: Since compilation happens beforehand, many compile time error can be detected, providing a better degree of stability of application.

**Disadvantages**

1. Works only with HTML and CSS, other file types need a previous build step
2. No watch mode yet, must be done manually (bin/ngc-watch.js) and compiles all the files
3. Need to maintain AOT version of bootstrap file (might not be required while using tools like cli)
4. Needs cleanup step before compiling

**References/Further Reading:**

<https://angular.io/docs/ts/latest/cookbook/aot-compiler.html>

What are the core differences between Observables and Promises?

*A nice answer taken from stack overflow:*

A Promise handles a **single event** when an async operation completes or fails.

Note: There are Promise libraries out there that support cancellation, but ES6 Promise doesn't so far.

An Observable is like a **Stream** (in many languages) and allows to pass zero or more events where the callback is called for each event. Often Observable is preferred over Promise because it provides the features of Promise and more. With Observable it doesn't matter if you want to handle 0, 1, or multiple events. You can utilize the same API in each case. Observable also has the advantage over Promise to be **cancelable**. If the result of an HTTP request to a server or some other expensive async operation isn't needed anymore, the Subscription of an Observable allows to cancel the subscription, while a Promise will eventually call the success or failed callback even when you don't need the notification or the result it provides anymore. Observable provides **operators** like map, forEach, reduce, ... similar to an array. There are also powerful operators like retry(), or replay(), ... that are often quite handy.

**Promises vs Observables**

* Promises:
  1. returns a single value
  2. not cancellable
* Observables:

1. works with multiple values over time
2. cancellable
3. supports map, filter, reduce and similar operators
4. proposed feature for ES 2016
5. use Reactive Extensions (RxJS)
6. an array whose items arrive asynchronously over time

**References/Further Readings:**

<http://stackoverflow.com/questions/36064303/what-are-the-differences-between-observables-and-promises-in-javascript>

Explain local reference variables, ViewChild, and ContentChild.

Local template variables in angular2 is used to refer HTML elements and use their properties to access siblings or children.

Let’s consider you have an input field named username.

<input type="text" required ... />

This HTMLInputField can be made available to the template using # symbol with a variable name say username.

 <input type="text" #username required ... />

Now, this HTMLInputElement can be accessed from anywhere in the current template for example, checking validation and showing appropriate message based on the validation rule. But, username HTML reference is not accessible in the component/directive.

To access this in the component, angular 2 provides @ViewChild decorator which accepts the local reference variable.

@ViewChild('username') username: HTMLInputElement;

ViewChild element can be read after the view is initialized (ngAfterViewInit).

ContentChild is used to query the reference of the DOM within ng-content. Content Child are set before the ngAfterContentInit lifecycle hook.

For example:

*// <code>app.component.ts</code>*

<my-component>

    <p #contentRef>{{test}}</p>

</ my-component >

*// MyComponent.component.ts*

@Component({

    selector: ‘my-component',

    template: `

    <ng-content></ng-content>

    <div> ContentChild Example </div>

})

export class LifecycleComponent implements ngAfterContentInit{

                @ContentChild(‘contentRef’)   childContent: HTMLElement;

ngAfterContentInit() {

              this.log('ngAfterContentInit');

console.log(this.childContent);

    }

}

**Further Reading:**

**Reading Parameters:**

**Path Variable Parameters (Required):**

**<a [routerLink]="['/products', 123, 'edit']">Add Product</a>**

**Sample url generated:** localhost:4200/products/123/edit

**Constructor(public route:ActivatedRoute){**

**}**

**Gives a snapshot of the parameter:**

**V2**: this.route.snapshot.params['id']

**V4+**: this.route.snapshot.paramMap.get(‘id’);

Trigger everytime that the params changes:

**V2**: this.route.params.subscribe(

params =>{

let id:number = +params['id'];

this.getProductMethod(id);

}

);

**V4+**:

this.route.paramMap.subscribe(

params =>{

let id:number = +params.get('id');

this.getProductMethod(id);

}

);

**Optional Parameters:**

<a [routerLink]="['/products', 0,'edit'**,{name:productName,code:productCode}]"**

>Add Product</a>

**Sample url generated:** localhost:4200/products/123/edit;name=productName;code=123456

**Gives a snapshot of the parameter:**

**V2**: this.route.snapshot.params['code']

**V4+**: this.route.snapshot.paramMap.get(‘code’);

Trigger everytime that the params changes:

**V2**: this.route.params.subscribe(

params =>{

let id:number = +params['code'];

this.getProductMethod(id);

}

);

**V4+**:

this.route.paramMap.subscribe(

params =>{

let id:number = +params.get('code');

this.getProductMethod(id);

}

);

**Query Parameters: these are mainly used when parameters are used across multiple routes.**

**Use in the template that calls the component**

<a [routerLink]="['/products', 0,'edit'**"**

**[queryParams]=” {name:productName,code:productCode}”**

**[preserveQueryParams]="true"> Angular2**

**queryParamsHandling="preserve">Angular4**

>Add Product</a>

this.router.nagigate([‘/products’],{queryParams:{fitlerBy:’er’,showImage:true}})

**Sample url generated:** localhost:4200/products/123/edit?name=productName;code=123456

**Gives a snapshot of the parameter:**

**V2**:

this.listFilter = this.route.snapshot.queryParams['filterBy'] || '';

this.showImage = (this.route.snapshot.queryParams['showImage'] === 'true');

**V4+**: ?

Trigger everytime that the params changes:

**V2**: this.route.params.subscribe(

params =>{

let id:number = +params['code'];

this.getProductMethod(id);

}

);

**V4+**:

this.route.paramMap.subscribe(

params =>{

let id:number = +params.get('code');

this.getProductMethod(id);

}

);

**this.route.snapshot.data:** used to read static data specified in the router definition or for data set from resolver. This value does not change for the life of the application

path:’’

,component:’’

,data:{pageTitle: ‘Product List’}

this.route.snapshot.data[‘pageTitle']

**this.route.data.subscribe :**

// Watch for changes to the resolve data

this.route.data.subscribe(data => {

this.onProductRetrieved(data['product']);

});

Retrieves data from snapshot:

this.route.snapshot.data[‘product']

Snapshot:

To read the parameters only once

Simple code

Observable:

If the parameter can change without navigating to another

component

**Providing Data With a Route:**

Route Parameters, Optional Route Parameters, Query Parameters, Route’s Data Property, Route Resolver, Angular Service

**Activating Route with Code**

Activating route with code is usually done to execute code before routing to a page. For instance, we may want to logout a user before redirecting to the login page.

Constructor(public router:Router){ }

Public logout():void{

//do some processing

This.router.navigate([‘/welcome’]);

}

This.router.navigateByUrl(‘/welcome’); this is used when we don’t want to maintain the previous url segment. For instance, if we navigate to a url /test, then navigate to /test2, the url will be /test/test2. But, navigatebyurl will not maintain /test and it will create a url as /test2. Navigatebyurl is not used often.

**Jasmine:**

It is the unit test framework that we use to test angular 2.

Jasmine uses four main functions for unit test:

Describe(): It is a containing function for a suit of tests

beforeEach(): It is used for common setup code for the test

it(): it() function creates a unit test.

Expect(): expect function is how we assert that the test passed.

**Karma:**

Karma is a command line test runner. It is the tool that actually causes the test to be executed.

**Isolated Vs Integrated Tests**

Isolated Tests test class only. It does not test templates.

Class is constructed in Test

Isolated tests are simple and require less code.

Isolated tests are best for services and pipes

Isolated tests are appropriate for components and directives

**Intregrated Tests**

Test class and template

Constructed by the framework

Integrated tests are complex

Integrated tests are mainly used for components and directives

**Install Karma**

Npm install –g karma-cli

Npm install karma or npm install [karma@1.7.0](mailto:karma@1.7.0) -D

Npm install karma-chrome-launcher –save-dev -D

Npm install karma-jasmine -D

Or

Npm install karma@latest karma-chrome-launcher@latest karma-jasmine@latest jasmine-core@latest @types/jasmine@latest –D

Get karma-test-shim.js for git repo and drop into the root of the project

<https://github.com/juliemr/ng2-test-seed>

**Start Karma:**

Karma start

**To test services:**

We create a unit test file spec. In the test file spec, we create the test and create instances of services that should be used as part of the test. We also create mock classes to pass to the constructor of the service classes if they depend on other classes. For instance, service classes that depend on http service. In this case the mock class will have dummy method for get/post or any other method.

**To Test Components:**

We do isolated tests with components. We don’t test the templates, but we test the type script component methods.

**Linting:** check the code standards and possible bugs.

[In this module, we're going to learn all about how to get our code ready to go to production.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=0) [The focus will be on the various things we can do to improve performance.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=5.5) [This includes standard things like reducing download code size and number of requests, and also some other](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=9) [things that will improve the performance of our code in the browser.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=14) [Let's take a look at our agenda. First, we're going to talk about linting.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=18) [Linting is the process of checking your code for typical type violations.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=23) [In many cases, these violations are simply stylistic. But in some cases, they can actually lead to](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=27) [insidious bugs in the right situations. Linting is normally used on teams in order to keep everybody's style](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=32) [somewhat the same. In our particular case, Jim has been doing a lot of coding without using excessive semicolons.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=38) [I, on the other hand, have used semicolons a lot more. In the section on linting, we're going to address that issue.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=43) [Now linting is not specifically an item that will improve performance like the other things we're going to](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=51) [talk about, but this is still a fairly appropriate place to talk about this topic.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=54) [Next, we'll talk about tuning our RxJS requests. This is going to help us reduce the number of requests that](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=58) [we're making. After that, we'll talk about enabling production mode.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=64) [This is a setting in Angular that will improve the performance of your code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=68) [Then we'll look at a very basic deployment strategy. After that, we'll talk about the Ahead-of-Time compiler](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=71) [and what it does to improve performance. Then we'll look at tree shaking and bundling,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=77) [techniques that reduce the size of code downloaded and the number of requests.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=81) [After that, we'll take a look at using Webpack. We've been using SystemJS this entire time, but Webpack is](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=86) [an alternative tool that you can use in order to handle your module loading and bundling.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=90) [Finally, we'll talk about optimistic bundle downloading. This is the process of taking lazy loaded bundles](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=95) [and actually optimistically downloading them even though the user hasn't gotten to that part of the application.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=102) [It's very important for you to know that the concepts of going to production are very complex, and many of](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=107) [the details are well beyond the scope of this course. As we talk about these concepts in this module,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=112) [we will simply talk about what is done and the tools that do them and look at the results of these tools.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=118) [This will give you a good introduction to the topics and let you understand what the moving pieces are,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=122) [but the details of how to actually run all the various processes and tools that can be done to your code](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=128) [when going to product are just too involved to fit into this course.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=133) [So this is not a module where we will focus on being able to follow along.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=136) [Instead, we will simply look at what the pieces do and the effects and benefits of each of those various options.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=0&mode=live&start=140)

[Learning the Concepts](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live)

[When preparing your code to go to production, you have three basic levels.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=1.733) [None of these are hard and fast rules, and oftentimes they can be mixed and matched.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=6.233) [But generally the things that you do will fall into these three areas, so we'll look at each of them.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=10.233) [The first is a minimal preparation. This is what you would do for a simple demo or a proof of concept](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=14.233) [deployment or possibly if you just had a simple site with minimal staff and usage and didn't care too much](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=20.233) [about the performance metrics of your site in production. This is definitely not a recommended setup.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=25.233) [With this level, you are basically just doing a copy and paste deployment of your code to production,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=31.233) [but you may take a few small steps. First, you will want to tune the RxJS requests that you are making.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=35.233) [As mentioned in our module on RxJS, the easy way is to just get all of it.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=41.233) [But this ends up being a lot of data and making a lot of requests, so tuning those requests is a simple way](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=47.233) [to prepare for production deployment. Second, you'll want to enable prod mode, which is short for](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=52.233) [production mode of course. We'll talk more about what prod mode is and what it does later on in this module.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=57.233) [That's the first level of preparation. The second level is to go with a full preparation for production.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=63.233) [This is more typical of what your average site will use. The effort for this level is significantly more](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=69.233) [than the previous level, but so also in the reduction of the number of requests made and the size of data](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=72.233) [sent over the wire. With this level, you would do most if not all of the following.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=78.233) [First, use the Ahead-of-Time compiler. This will remove a lot of the framework cost that has to be sent](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=82.233) [down to the browser. We'll discuss the AOT compiler more in a minute.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=88.233) [Second would be to bundle up your code. When we start, our code is all in separate files, and each of those](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=92.233) [files is a separate request for the browser. Bundling fixes that.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=97.233) [Third is to minify your code. This removes whitespace and does other things to reduce the size of your code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=101.233) [Finally is to tree shake your code and remove any pieces of it that aren't used.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=106.233) [This is a more recent advancement in JavaScript production preparation, and we'll look at it closer later on.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=111.233) [That completes the second level of production readiness. The third and final level is to use lazy loading](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=118.233) [so that you minimize the amount of code that has to be downloaded to start out your application.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=124.233) [In this level, you would add lazy loading to your code and optionally add optimistic downloading.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=128.233) [Now one thing to notice is that although these three levels generally have an order, they aren't strictly](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=133.233) [a progression. For example, you don't have to enable prod mode to use the Ahead-of-Time compiler,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=139.233) [and you can use lazy loaded modules even if you're not bundling. In fact, we have done none of these steps](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=143.233) [except we have our user module being lazy loaded. So in a sense, we skipped straight ahead to the end.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=148.233) [The truth of the matter is that you won't often use every technique listed here.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=154.233) [In fact, at the time of this recording, it is extremely difficult to do everything and would require](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=157.233) [quite a bit of work and understanding of the inner workings of various tools.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=164.233) [These are the different preparation levels and the various techniques used to prepare your code for production.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=167.233) [We'll look at all of these techniques in this course.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=172.233) [Before we get deeper into any of the production techniques, we first need to address a very important design question.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=175.233) [How many modules should I create? We've discussed this before, so we're simply going to revisit the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=182.233) [question here as it pertains to production deployment. The key here is the fact that modules are the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=187.233) [enabler for lazy loading. So for the maximum performance in a production environment, modules will be required.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=192.233) [This doesn't mean that you should be overly worried about this as you first build an application.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=199.233) [Since modules are easy to add, it's fine to delay the decision as to what pieces to turn into their own](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=203.233) [modules until later. Or use other criteria such as if you prefer to keep all features in their own module.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=208.233) [But just know that modules are a requirement for lazy loading.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=214.233) [We mentioned some fairly new concepts just a minute ago--bundling and tree shaking.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=219.233) [There're also related concepts of code splitting and minification.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=223.233) [Let's look at each of these concepts in turn.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=227.233) [Minification, which is often called uglification, is the process of removing extraneous characters from](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=230.233) [your source code that only exist for readability. Take the following service.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=235.233) [It has a lot of whitespace to make it very readable. After running it through a minifier, it would look like this.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=240.233) [The code is no less valid than it was before, but the whitespace has been removed.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=246.233) [This makes the code difficult to read, but that's okay for execution.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=252.233) [The browser doesn't need nicely formatted code to run it. There are further levels to minifiers and](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=255.233) [uglifiers, but this is the basic gist of what they do.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=259.233) [Imagine for a moment that you had a very simple JavaScript application that consisted of four files.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=263.233) [JavaScript doesn't inherently have to be split up into separate files.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=268.233) [It's a script language so dividing it up is purely a construct we do for our own benefit.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=271.233) [All of our JavaScript could be contained in a single file, and that is exactly what bundling is.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=277.233) [When you create a bundle, you take your JavaScript from a bunch of separate files and combine them all](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=282.233) [into one file. Of course, there're nuances to the process, but that's basically how it happens.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=286.233) [The benefit here is that your browser doesn't have to make hundreds of requests.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=292.233) [Instead, it just makes a single request for your JavaScript. This is usually a huge performance improvement](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=296.233) [for applications and often the biggest single improvement. With the recent advent of the HTTP/2 protocol,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=301.233) [we may in the future see the need for this to decrease. But at least for the time being, this is a very](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=308.233) [important step and one you won't want to skip.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=314.233) [Code splitting is the mechanism which allows for lazy loading in a bundled environment.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=316.233) [It's a variation of bundling. You still combine your JavaScript into bundles, but instead of one,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=320.233) [you create multiple bundles. Imagine a huge application with 5,000 JavaScript files.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=326.233) [That's way too many requests, so bundling it up is a great idea. But the problem is that that much](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=332.233) [JavaScript can actually be so large that the download file could take a long time to transmit even on](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=336.233) [high bandwidth clients. And, of course, high bandwidth is not something you want to count on.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=342.233) [So code splitting is the process of producing a few smaller bundles.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=347.233) [The number of bundles and sizes of each can vary greatly from system to system and involves a lot of](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=350.233) [balancing over what to give the users immediately and what to delay the download of.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=355.233) [In these cases, you still have a single main entry bundle. This is the bundle that's downloaded when a user](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=360.233) [hits your site. But as soon as a smaller bundle is downloaded, the application can begin to work for the user.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=366.233) [Then the other bundles are downloaded either just when they're needed, which can still introduce a small](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=371.233) [delay when they hit that part of your application, or in the case of optimistic downloading, they can be](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=377.233) [downloaded when the entry point bundles finish. That way, there's likely not any delay, the tradeoff being](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=381.233) [if the user never needs that part of the system, they still download it.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=387.233) [Our final concept is tree shaking. This is the most recent addition to code optimization for production in](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=392.233) [the JavaScript world. The basic concept is to remove code that isn't ever called in your system.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=397.233) [Imagine a system where you have a service called UserService with two methods--getUsers and getUser.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=401.233) [Now further imagine that in your system, this service is only used in one place and only the getUsers](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=409.233) [method is called, and the getUser is never called anywhere. You wrote it but never used it.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=415.233) [Well there's no reason for that method to even exist on the class anymore.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=421.233) [For all intents and purposes, you can remove it. But say you think that sometime in the future, it may get](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=424.233) [used or perhaps at some library that is used by multiple applications or in other applications it's used,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=429.233) [so you can't just delete the code. But when you deploy a production, it would be nice if that code didn't](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=435.233) [even make it into the bundle. That is tree shaking. Tree shaking can actually remove methods and](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=439.233) [properties that are never called.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=445.233) [With Angular, you will have two options for how to handle module loading and bundling.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=448.233) [These options are SystemJS or Webpack. We have used SystemJS in our application, but it's just as viable](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=452.233) [to use Webpack. The choice of which module loader to use will also dictate what you can do for minification,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=459.233) [bundling, code splitting, and tree shaking. If you're using SystemJS, then you will likely be using Rollup](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=465.233) [to handle bundling, minification, and tree shaking. Note that at the time of this recording, Rollup does not](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=470.233) [support code splitting. That is likely to change, so by the time you watch this, Rollup may support code splitting.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=477.233) [If you're using Webpack, then you will just use Webpack plugins to handle your minification, bundling,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=482.233) [code splitting, and tree shaking. In the end, either product will work just fine for getting your code](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=488.233) [to production, but each one has various strengths and weaknesses.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=492.233) [In general, you should use the tool you're most comfortable with. Although currently each tool has some](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=495.233) [pieces of going to production that they don't do well, that is likely to change in the near future as the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=502.233) [process of getting to production is an area of great focus for the Angular team.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=506.233) [We'll see some of these differences later on in this module.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=1&mode=live&start=510.233)

[Linting with TSLint](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live)

[We want to get our code linted, but since it's written in typescript, we can't use any of the typical](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=1.295) [JavaScript linters that are available because they won't lint typescript.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=5.795) [Instead, the tool we're going to use is called TSLint. We'll start off on the command line so we can](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=8.795) [install the correct npm module. I'm going to npm install tslint. I'm going to save this into my dev](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=14.795) [section of my package.json file as well.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=21.795) [Now that I've got TSLint installed, I can run it from the command line, although it's a little bit painful](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=25.795) [to do it if we just install it locally so you can always install it globally with the -g flag.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=30.795) [Since I've already got that done, I'm not going to execute this line of code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=38.795) [But what I do want to do is create an initial TSLint configuration file.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=41.795) [I can do that by running TSLint with the --init flag. Now if you haven't installed TSLint globally,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=46.795) [you'll execute it like this--node\_modules/.bin/tslint and then whatever command line flags you want to give it.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=54.795) [That has created a tsconfig.json file.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=63.795) [We can see that in our project right here. This file has a lot of pieces to it, but one of the very first](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=67.795) [things I want to do is take this jsRules node, which is the node that has the rules for JavaScript files,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=73.795) [and delete it. Since we're not worried about having TSLint lint our JavaScript, just our typescript,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=80.795) [we don't need this section. And it can actually be a little confusing.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=86.795) [So I'm going to delete the whole thing, and we're just going to be dealing with this rules section.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=90.795) [Now it's also really nice to have TSLint running in your editor.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=94.795) [In this case, I'm using Visual Studio code, and that's a very popular editor for working with Angular 2.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=99.795) [So I'll very quickly show you how to install TSLint to run inside of VS code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=103.795) [But most editors that you work with will support TSLint. You'll have to figure out how to integrate TSLint](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=109.795) [on your own if you're using an editor that isn't VS code. With VS code, I hit Ctrl+P.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=115.795) [On a Mac, that would be Cmd+P. And then I type in ext install tslint and then hit Enter.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=121.795) [I've already done this in my copy of Visual Studio code, so I won't execute this command.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=130.795) [Once you've got that installed or since we've just added our tsconfig.json file, we're going to want to](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=134.795) [restart our VS code. So I'll save that, close it down, and then open it back up.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=141.795) [And now we'll get linting inside of our editor. Let's open up our app.module, and we can see we're getting](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=148.795) [these green underlines that indicate some kind of linting error. In this case, the problem is the string rule.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=155.795) [By default, TSLint wants all strings to be delimited with double quotes and not single quotes.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=159.795) [So, of course, we could fix that ourselves by just changing these to double quotes.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=165.795) [We've still got a linting error, and that is a missing semicolon.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=171.795) [And, again, we can fix that ourselves. Or if we want, we can either click on this little lightbulb right](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=175.795) [here, and we get some options such as Fix the quote issue, and we can also do Fix the semicolon issue.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=181.795) [But in some cases, we can fix many issues at a time. We can also get that same menu by hitting Ctrl+.,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=189.795) [and so in this case, I'm going to say Fix all "semicolon" tslint warnings.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=195.795) [When I do that, it adds semicolons to all lines of typescript. Now not all of the typescript errors that](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=199.795) [you'll encounter can be fixed like this. But this is a simple way to fix one single file for one issues](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=205.795) [at a time depending on if that's an issue that supports an automatic fix.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=213.795) [In the other case with the quotes issue, if I hit Ctrl+P and I select Fix all "quotemark" tslint warnings,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=217.795) [you'll notice that nothing happens. That's because this particular issue at this particular time doesn't](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=226.795) [have an auto-fix. So that might be something we need to fix ourselves.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=231.795) [Now, personally, I'm not a big fan of using double quotes. I like single quotes.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=235.795) [So we're going to go into our rules file and make an adjustment. Down here I can find the quotemark rule](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=241.795) [and change it from double to single, hit Save. And now if I go back to my app.module, you'll notice I'm](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=248.795) [not getting errors anymore where I have single quotes, but I am where I have double quotes.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=253.795) [So I'll fix those and this one. And those issues are addressed. We can scroll down and see more issues](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=259.795) [such as here we've got the double quote issue. Here we've got a trailing whitespace issue.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=267.795) [I've got a single space. All these various issues are shown to us by TSLint.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=270.795) [Now it's nice that we get it for the file that we're working in. Unfortunately, we've got a lot of files](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=277.795) [in our project. What we really want to do is see all the errors across our entire project.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=281.795) [So that's where the command line comes in a lot handier. Let's save these changes and go out to the command line.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=287.795) [And we are going to run TSLint by typing tslint and passing in a string that contains the pattern match](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=293.795) [for which files we want to lint. In our case, we want to lint everything in the app directory,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=300.795) [and we want to lint across all subdirectories but just the typescript files.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=304.795) [So there's my command, and I'll hit Enter.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=309.795) [And we can see that it's listed out all the errors found in our project.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=311.795) [Most of these you'll see are missing semicolons or something to do with whitespace.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=316.795) [Now, again, I'm not too bothered by whitespace issues, so let's go back to our](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=321.795) [config file, and let's adjust the whitespace issues that we're having.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=325.795) [This whitespace rule right here, I want to change it from true to false.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=330.795) [And also up here the no-trailing-whitespace, I want to turn from true to false.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=334.795) [Now those rules won't be enforced, and if I go back,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=338.795) [and run TSLint again, most of the whitespace violations have disappeared.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=342.795) [I've still got the missing semicolon, and there're a few other rules that TSLint can do an automatic fix](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=347.795) [for us, and we can get that from the command line by adding the --fix flag.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=353.795) [So we'll run that. And it, again, reports out to us what the problem is.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=358.795) [In this case, it's still showing three missing semicolons. We'll have to see what that is.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=362.795) [We've got some string problems with using double quotes instead of single quotes.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=367.795) [That has to do with our event.service.ts. We can look at that.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=371.795) [And then a few other places where we have some issues with comments.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=376.795) [So let's start by going in and fixing the user.model file.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=382.795) [Open it up, and notice that we've got commas at the ends of these lines.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=387.795) [Even though it's reporting out to us a missing semicolon, the actual problem is that there're commas at the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=390.795) [end of the lines instead of semicolons. So I'll fix those and hit Save.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=396.795) [The next file we want to look at is this event.service.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=401.795) [We've got an issue with a string here. Let's change that to single quotes.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=405.795) [And one here. And now if we scroll down, we see that inside of the data that we used to have in memory,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=409.795) [We've got all of these double quote issues. Fortunately, we don't need this data anymore, so I'm going to](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=415.795) [delete the whole thing, and that should take care of all those errors.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=421.795) [I'll save that.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=425.795) [Let's go back and run TSLint again. Now we've got some issues inside our voter.service.spec files.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=426.795) [Let's fix that next. And, again, it's a matter of double quotes needing to be single quotes.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=431.795) [In this case, I'll do a search and replace. We'll find all the double quotes and replace them with single quotes.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=436.795) [And, of course, we want to make sure that we didn't introduce any syntax errors.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=442.795) [Then let's look at our navbar.component, which has another double quote issue.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=447.795) [And that's right here.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=452.795) [And we've got the routes file as well. This problem is that it doesn't expect there to be a space afterward](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=455.795) [when there's a colon. I'll fix that one,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=461.795) [run our linting one more time. Our app.module has a bunch of string issues.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=465.795) [Let's fix that. And, again, I'll just do a find and replace. And notice here we've got a couple of](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=471.795) [indicated errors about spaces after identifiers.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=479.795) [Next, we've got our simpleModal.component.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=483.795) [That's got an error right here.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=487.795) [Also our 404.component.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=489.795) [It wants there to be a space after the class identifier before the opening curly brace.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=492.795) [Fix that.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=497.795) [We have create-event,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=498.795) [which has a similar issue.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=500.795) [And we have the event-details/session-list test, the integrated test.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=502.795) [And the issue here is that it wants a space after a single line comment, one there and one right here.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=508.795) [Now let's run TSLint one more time. It should take off that fix flag, of course, when we run it.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=516.795) [And we see that we're getting no errors. So we've now fixed all of our linting errors.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=522.795) [Of course, when you make this many changes to an pap this big, you're going to have a pretty big check-in](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=525.795) [that just has a bunch of small changes but a lot of files touched.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=530.795) [But from here on out, we should be good. And, fortunately, our editor should keep us from making anymore](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=534.795) [linting errors. But we can't just rely on that. We do want linting if we're going to be using it on our](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=540.795) [project to be supported with something like continuous integration or set up with some kind of a gated](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=546.795) [check-in, something like that, it's really up to you. But linting can be a great way to make sure that](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=552.795) [the source code in your project all looks fairly uniform. And you don't get the issue that we had in our](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=557.795) [project where we've got a lot of lines with semicolons and a lot of lines without.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=562.795) [Instead, we've now got uniform, nice-looking code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=2&mode=live&start=567.795)

[Tuning Your rxJS Requests](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live)

[One of the things that's really easy to do to improve the performance of your application is to tune the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=0.756) [requests that you're getting for RxJS. Let's take a look at what the problem is so that we can understand](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=5.562) [why we want to fix it and see how big of a benefit it is when we do fix it.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=11.562) [In the browser with our application open, I'm going to open up the development tools and go to the Network tab.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=16.562) [Then I'm going to refresh the page so we can see all the network requests that we're making and how much](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=22.562) [data we're delivering over the wire. And in this case when we're all said and done, I've got 406 requests](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=26.562) [and 3.3 MB transferred. That is a lot of requests and quite a bit of data transmitted.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=33.562) [If we were to look through these requests, we're going to see something very interesting.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=38.562) [See these PairsObservable, NeverObservable, IntervalObservable, these are operators with RxJS.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=43.562) [Each one is a separate request and just a few bytes of data. But, overall, they add up to a lot of requests](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=51.562) [and, actually, a lot of data. We aren't using most of the stuff, so let's tune our project so it doesn't](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=58.562) [request anything we don't need.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=62.562) [Back to the code, what I want to do is do a search for RxJS/Rx. And you'll notice there're four places](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=65.562) [where that's found. Let's look at the first one. This is our VoterService spec file, and it's bringing in](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=73.562) [Observable when requesting Rx. In this case because it's a spec file, it actually doesn't matter.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=79.562) [We don't need to tune this up. It's just a test, so it's not going to be running on production.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=85.562) [But we'll fix the issue here anyway since it really doesn't hurt us.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=89.562) [I'm going to change this string from RxJS/Rx to RxJS/Observable. And I'll save that change.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=93.562) [We'll go to the next one and do the same exact thing. In each case, changing it to just Observable means](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=99.562) [that we're only requesting the Observable object and not everything inside the library.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=107.562) [Next, we're requesting both subject and Observable. In this case, once I change this to just Observable,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=112.562) [I'm going to get an error here because Subject is not found in this.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=118.562) [So I can create a second line and request Subject and remove Observable from this line and remove Subject](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=123.562) [from this line. But in the case of this file, Subject isn't even used anymore, so I can just remove it.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=132.562) [Save that change and go to the last one. And change this from Rx to just Observable.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=138.562) [Now having done that, you might think that we've fixed all of our issues, but in reality, we haven't.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=143.562) [We have new issues in that we are using operators that are not being requested.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=151.562) [And you can see in this very file, we've now got this error on the do method.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=155.562) [We've also got it on the of method. And a little bit farther down here, we've got it on the map method.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=161.562) [These different methods, again, are each their own request. We want these operators, but we don't want every operator.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=166.562) [Now we could just import those operators inside of this file to fix this issue.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=172.562) [But there's a much better way to fix this across all of our files, and that is to create a new RxJS extensions file.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=177.562) [So I'm going to go into the root of our application where the app.module is and add a new file called](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=185.562) [rxjs-extensions.ts. And now I'm going to import all of the operators that I'm using in my project or](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=193.562) [operators that I'm very likely to use. So in this case, we will import RxJS/add/observable/of.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=199.562) [And that's how we get the of operator. We'll also add throw. Notice this import looks a little bit](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=211.562) [different than our other imports. We don't have the destructing object right here and the from keyword.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=217.562) [This is how we import something that is used by a different object but still have it be available.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=224.562) [Next, we want to import some operators. I want the catch operator, so this is RxJS/add/operator/catch.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=230.562) [I also want the do operator and the map operator. Now there's one more operator that I think is fairly](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=242.562) [frequently used, which is filter. We're not actually using it in our project, but it is the kind of](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=248.562) [operator that you may want to include just because it's common enough that you may use it.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=255.562) [We also find over time that there're other operators that you frequently use, and they can be added to](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=260.562) [this file as well either when you start using them or proactively so that later on if you do use them,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=264.562) [you don't have to remember to come back to this file and add them.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=269.562) [Let's make a save, and we're going to go into our app.module right up at the very time.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=273.562) [And we are going to import that rxjs-extensions file. And since our app module's our starting point,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=278.562) [that's going to process that file, which will request all those operators that we have listed in the file.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=287.562) [And now if we go back to our auth.service, we're no longer getting errors on catch or do or map.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=292.562) [And now that we're good with everything, let's go back to the browser.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=300.562) [And we're going to refresh the page and see how the number of requests and size transferred has changed.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=303.562) [And there we go. We've gone from about 400 to about 120 and from almost 3-1/2 MB to just over 2.7 MB.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=310.562) [So there we go. With one fairly simple step, we have removed a ton of requests and a fair amount of data transferred.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=3&mode=live&start=319.562)

[Enabling Production Mode](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live)

[The next step we want to take is to enable prod mode or production mode in our Angular application.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=0.532) [Doing this is extremely simple, but let's first talk about what it is and what it does.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=6.532) [If we go back to the browser,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=11.532) [you may have noticed that we're always getting this message in the console that Angular 2 is running in](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=13.532) [development mode. Call enableProdMode to enable the production mode.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=16.532) [This is a simple reminder that you haven't turned on prod mode in the application.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=20.532) [While we're building the application development mode, that is fine.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=25.532) [We want development mode. The reason is that development mode runs an extra step through the Angular 2](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=28.532) [change detection process. When it's all done running change detection, it runs it one more time to make](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=34.532) [sure that nothing has changed. In the case of Angular 2, the purpose for this is because of the one-way data flow.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=40.532) [You should never be able to make a change to your state while running change detection.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=48.532) [If that ever happens, Angular 2 will warn you that you've done something really strange and funky,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=53.532) [and you need to fix it. So we want to be running in development mode while we're building our product](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=57.532) [so that as we write our code, if we were to accidentally cause this problem, we would get the warning.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=63.532) [But once we are ready to move to production, we want to turn on prod mode because that extra change](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=68.532) [detection cycle is an extra amount of work that we don't need in production.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=74.532) [So to turn that on, we'll go back](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=77.532) [to our main.ts file, and I need to import the enableProdMode function, and I do that by importing from](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=79.532) [@angular/core, and the enableProdMode function is imported just like that.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=88.532) [And now I can call it. And I want to call it just before I Bootstrap my module.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=94.532) [It's a simple function that takes in 0 parameters. And if I save and go back out to the browser and refresh,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=99.532) [we can see that our app is running, but we're no longer getting that message.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=106.532) [Now, unfortunately, what we've](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=109.532) [done here is turn on prod mode even though we're in development. We certainly don't want to be adding this](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=111.532) [line of code and then maybe commenting it out while we're doing development and then commenting when we](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=117.532) [go to production. Instead, you're going to want to do something to make sure that this line of code is](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=123.532) [only called when you're in production. There are various ways to do that.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=128.532) [Environment files that only show up on your production servers or maybe integrating adding this line of code](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=132.532) [during the CI process when you actually build for production. And how you're going to do it will really](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=138.532) [depend on your own build and your own system. So I'm not going to show how to actually do that.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=142.532) [So I'm not going to add any code that will do that. But just know that you don't want this line of code](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=149.032) [to be running in your application while you're in development.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=4&mode=live&start=152.532)

[Basic Copy Deployment](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=5&mode=live)

[We've made a few changes to our system. And for a test deployment or a simple site where performance isn't](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=5&mode=live&start=0.75) [so critical that we need to squeeze every millisecond of our application that we can, then we can do a](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=5&mode=live&start=6.506) [simple copy deployment. This might sound easy, but it can involve a few steps.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=5&mode=live&start=12.506) [Basically you're just copying all of your dev files up to your production environment as is.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=5&mode=live&start=17.506) [This sounds fine, but there're a few issues you want to watch out for.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=5&mode=live&start=21.506) [First off, your dev environment has a lot of files you may not need in production such as your test files,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=5&mode=live&start=24.506) [config files, source control files, etc. Now these files won't necessarily cause a problem when you deploy](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=5&mode=live&start=31.506) [them to production, but often it makes the process much simpler if you don't copy them.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=5&mode=live&start=37.506) [So in this case, often you'll want a script that only copies the files you want rather than manually](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=5&mode=live&start=42.506) [hitting Ctrl+C, Ctrl+V. You also need to worry about what you'll do about your node modules directory.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=5&mode=live&start=47.506) [Do you copy it as well or run an npm install with the production flag once you get your files up on your](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=5&mode=live&start=53.506) [production environment? You will have to figure these questions out.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=5&mode=live&start=59.906) [But, otherwise, you've got your application deployed, and you're good to go.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=5&mode=live&start=63.006)

[Ahead of Time Compiler Overview](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live)

[We're going to look at the Ahead-of-Time compiler in this section.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=1) [So let's talk about the benefits so that we know why we might want to use it, and then we'll talk about what it does.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=4.029) [First off, the Ahead-of-Time compiler or AOT compiler enables faster rendering.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=9.029) [As mentioned earlier, the Ahead-of-Time compiler produces a precompiled version of your application.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=16.029) [This isn't like a C++ compiler that produces some kind of exe file.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=21.029) [Instead, the Angular compiler just produces JavaScript from JavaScript, but it's derived from the components](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=25.029) [and templates that you write. And that code actually manipulates the browser.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=30.029) [Once this precompiled version is created, the browser runs this code to render the application much faster.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=36.029) [And you skip the whole step of compiling application. There are also fewer requests.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=41.029) [because the compiler inlines the HTML templates, it reduces the number of requests that you'll make.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=46.029) [In most cases, you're going to bundle anyway, so the number of requests will go down even more.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=52.029) [You also download less of the Angular framework. Since we're precompiling our application, we don't need the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=56.029) [parts of the Angular framework that handle complication. It turns out that this is a pretty hefty chunk](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=61.029) [of code, so you can really reduce your download size. Since the templates get compiled and turned into code,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=66.029) [certain errors will get caught during the precompilation process, which means that you'll know about them](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=71.029) [before they get to the browser. And, finally, since your templates are compiled, the AOT compiler removes](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=76.029) [the need to read risky, client-side HTML, and also does no JavaScript evaluation.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=83.029) [So there are fewer opportunities for injection attacks.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=89.029) [Now a big part of the compiler is compiling templates. When Angular runs in the browser, ultimately the code](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=92.029) [that you write needs to show something to the user. That means manipulating the DOM.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=97.029) [We author our templates in HTML to let Angular know what needs to be done.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=102.029) [We use bindings, events, directives, and components. Ultimately, this HTML is useless as it is.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=106.029) [It has to be turned into instructions for the Angular engine to actually manipulate the DOM and render something.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=111.029) [That process is the Angular compiler, which takes the HTML from your templates and turns it into JavaScript](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=118.029) [instructions that manipulate the DOM.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=123.029) [Here's an example from our own app. Here we've got the binding to session.name and an ngIf directive](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=126.029) [that will hide or show that tag based on a certain condition. Both of these pieces need to be compiled](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=132.029) [so that as the session name changes or as the number of voters in a session changes, the DOM can be](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=139.029) [manipulated to show the results. When we are in development, we use the Just-In-Time Compiler.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=143.029) [This is a compiler that runs in the browser and processes the HTML template to produce the necessary](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=148.029) [JavaScript to manipulate the DOM when certain things happen. The problem is that much of the code that is](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=152.029) [sent to the browser for Angular is actually just this compiler, so it takes extra time to download.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=158.029) [Furthermore, it takes time for the compiler to run and process the templates into code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=164.029) [This is something that doesn't need to happen at the last moment like this.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=169.029) [It can be done beforehand. And that's where our AOT compiler comes in.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=172.029) [The process of precompiling all the templates and components in a project takes enough time that it's not](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=176.029) [something we generally want to do in development. It just adds too much delay when we're changing code](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=181.029) [and trying to see it in the browser. In this case, the Just-In-Time compiler is just the right thing to use.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=185.029) [But generally when going to production, we want to use the Ahead-of-Time compiler.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=191.029) [This replaces our source code with compile code. As mentioned, it reduces the amount of code that we](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=195.029) [download, and it also speeds up rendering. Both of these things greatly improve the performance of our application.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=200.029) [Now before we actually show how to use the AOT compiler, there's a list of things that you need to know that](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=208.029) [you can't do in your code. Of course, this can be problematic since you can end up building a lot of your](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=212.029) [application that works just great in development, but when you try to use the AOT compiler and go to](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=217.029) [production, things don't work right. Also, this list is not exhaustive.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=221.029) [There are undoubtedly things you can do in dev mode that aren't on this list that the AOT compiler will not allow.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=226.029) [And it's also very likely that the list will change over time as new versions of Angular are released.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=232.029) [So don't consider it to be complete by any means. Some of the items on this list are things that we have](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=238.029) [done in our application, and I'll show you how to adjust them. First, in our templates, we can't do either](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=241.029) [of the following two things: Using form.controls.controlName or using control.errors?.someError.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=247.029) [We do both of these, and we'll see how to fix them in a minute. Also, you shouldn't use default exports](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=256.029) [in your module exports. We haven't done this so we don't have to worry about it.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=262.029) [You shouldn't use functions in your providers, routes, or declarations of a module.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=265.029) [Again, not something that we have to worry about in our specific application.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=269.029) [Any field that's used in a template, including inputs, has to be marked as public.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=274.029) [This is something that we didn't do, and we'll have to fix. And, finally, you can't use declare var the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=277.029) [way that we did when we wrapped jQuery and toastr to bring in global variables.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=284.029) [We'll have to fix that as well. Again, as we go through this section and look at these changes,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=289.029) [don't worry about trying to follow along. We're just skimming through what to do.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=294.029) [Actually, going through this process is far beyond the scope of this course.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=298.029) [One of the things that's important to understand is how your ES6 modules play into the process of compiling](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=303.029) [and bundling. When we look at how we configure the AOT compiler and later on when we configure Rollup,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=308.029) [we just give them a starting point, and then they walk through our code to figure out which files need to](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=314.029) [be included in our production build. This is nice. That way we don't have to specifically exclude our](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=318.029) [test files, our config files, or anything like that. The compiler will process just the code itself and](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=322.029) [see which files are actually used to know what to compile. Let's look at how that works in our application.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=329.029) [Our main entry point is main.ts. We can see that if we look in our](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=334.029) [index.html file. We are starting by hitting our main file. Of course, we are pointing at the js file since](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=338.029) [we are in the browser, but that's our main.ts file.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=345.029) [In turn, main.ts points to the app.module by importing it.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=348.029) [Again, the code shows that. Here we are importing our app.module and then using it down on line 6.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=352.029) [app.module in turn points to pretty much every other file in our application.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=360.029) [Again, we can see that](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=365.029) [in our code. Our app.module imports pretty much all of our components and services and then utilizes them](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=366.029) [in the module declaration.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=373.029) [Now the tree can get deeper like if we have a separate module, which we do, but this is the basics of it.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=376.029) [If we walk this tree for our whole application, we'll never see a connected test file or a config file or](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=382.029) [source control specific file. None of them will be imported so the compiler won't deal with them.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=388.029) [We can put hundreds of other files in our directories, but only the files referenced by our code will get](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=392.029) [compiled and bundled. So, therefore, when we are configuring the Ahead-of-Time compiler and Rollup,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=397.029) [we only need to give them entry points, and they'll walk the tree and handle the rest.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=403.029) [Next, we'll look at actually using the Ahead-of-Time compiler in our project.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=6&mode=live&start=408.029)

[Preparing for the AOT Compiler](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live)

[Let's look at actually using the AOT compiler. Now what I've done is I've prepared our application using](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=1) [the AOT compiler and Rollup, and we'll take a look at what I did in order to make that happen.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=6.853) [But, again, we're only looking at high-level items here. We're not looking at all the details.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=13.853) [The scope of doing that would be well beyond this course. A couple of things that are important that you](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=17.853) [need to know is you've got to install the compiler itself. So you do that with npm installing the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=23.853) [@angular/compiler-CLI and @angular/platform-server. These are the two node modules you need in order to](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=30.853) [run the AOT compiler. I'm not going to execute this code because I've already done it in the prepared](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=37.853) [version of our project. Now let's take a look at our code that's prepared to use the AOT.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=42.853) [Here's that project. Now there's a lot of things you have to do before you can run the AOT compiler,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=47.853) [and we're going to look at most of those pieces but, again, it really depends on your project, the code that](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=53.853) [you've written that'll determine what steps you have to go through in order to get your project running](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=57.853) [with AOT compiler. We're not going to be covering enough stuff to really figure all of that out,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=61.853) [but we are going to cover enough stuff that'll give you a solid idea of the different moving parts.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=66.853) [So one of the things the compiler has is its own typescript compiler built into it.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=72.853) [Therefore, we need a custom tsconfig in order to run the AOT compiler.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=75.853) [I've created this tsconfig-aot file right here, which is that custom file.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=81.853) [There's only a few pieces here that really matter. One is the module setting.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=86.853) [You've got to set this to es2015. The AOT compiler requires that the JavaScript that it reads is using](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=90.853) [es2015 modules and not something else like common js. Also, the files settings is important.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=97.853) [Here I've listed two files only--app.module.ts and main-aot.ts. We'll take a look at this file in just a second.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=103.853) [Finally, this last setting, the angularCompilerOptions. I've told it what directory to generate the output](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=111.853) [for, which is the AOT directory, and the skipMetadataEmit setting has to be set to true.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=119.853) [Next, let's look at the main.aot file that I created. This file is different from our main.ts.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=124.853) [Let's go back to the main.ts for reference. Here in our main.ts, we import platformBrowserDynamic,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=130.853) [and it is a function that has a bootstrapModule function. And we just need to import our AppModule and](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=137.853) [pass it in, whereas in the AOT version, we're not implementing platformBrowserDynamic anymore like we were here.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=142.853) [Notice that the difference of what's being imported is the word dynamic, and in main-aot, it's just platformBrowser.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=150.853) [That has a function called bootstrapModuleFactory, and we're passing in an AppModuleNgFactory.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=158.853) [This is because when our application actually runs, it's going to be using the files that have been produced](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=164.853) [through the AOT compiler. That produces an AppModuleNgFactory and various other files that all chain off of that.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=168.853) [You can actually go in and look at the files that the AOT compiler produces.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=176.853) [They're not really very readable, but they are somewhat interesting to look at.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=181.853) [And notice that this is coming from the AOT directory. Unlike with our main.ts file, we are bringing this in](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=186.853) [from our local app.module. Instead, we're digging down into the AOT directory and loading up one of the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=192.853) [files that's produced inside of there that's been produced, of course, by the AOT compiler.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=199.853) [Another thing that we had to do was adjust our index.html. You can see here we've got index-jit.html.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=203.853) [This is our old index file. The JIT means Just-In-Time compiler.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=211.853) [This is what it looked like before. Our new index.html file is different.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=216.853) [A couple of things that we've done is we've taken out a few scripts.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=222.853) [We used to have the reflect-metadata and the SystemJS files, of course, our SystemJS configuration.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=225.853) [Then we loaded everything up using SystemJS. In our new AOT version of the index.html file, we no longer](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=231.853) [have those files, and we're not loading up our script anymore through SystemJS.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=240.853) [We did add in this window.module. This is related to using relative file paths for your templates and](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=245.853) [stylesheets, and we'll talk about that in just a minute. But the code itself is actually loaded down here](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=252.853) [at the very bottom of the HTML, and we just point to the build.js file.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=257.853) [This is what's going to be produced by Rollup, and we'll see that in the next section.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=260.853) [It's very important that this script tag falls at the very end of your HTML.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=264.853) [It can't go any higher up. It's got to be the very last thing that's loaded up in your document.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=7&mode=live&start=269.353)

[Making Coding Fixes for the AOT Compiler](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live)

[Now in addition to this preparation, we also had to make a bunch of changes to our code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=1.71) [There's a fair amount of code that we wrote that just won't work with the AOT compiler.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=6.71) [One of the things is that we were using full paths for our templates and stylesheets in our components.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=9.71) [Let's go back to the original version of our application,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=16.71) [and let's look at our create-event.component right here. Notice that we've got a templateUrl.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=19.71) [That's pointing to the template for this component. And notice that the path to that is a fully qualified path.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=25.71) [It starts from the root of what our web server's going to be serving up and specifies the path all the way](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=33.71) [out to the HTML file. With the AOT compiler, you can't use these full paths.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=38.71) [You have to change them over to relative paths. So what we did in the same place is we just removed all of](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=42.71) [the prepending path information, and then we added the moduleId property and set it to module.id.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=51.71) [That is related to that module setting that we saw in the index file right here, this script where we set](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=58.71) [the window.module to aot. That enabled that for the AOT compiler.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=66.71) [This is not required if we're using relative paths without the AOT compiler.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=71.71) [But with the AOT compiler for the moment, this is required. So we set this to a relative path again.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=75.71) [All we do is strip off the directory information that we had and then add the module.id.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=82.71) [Now if you are using Webpack instead of SystemJS, you can actually specify it like this with the ./,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=87.71) [and that's how you specify your relative paths with Webpack. And in that case with Webpack,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=95.71) [you don't actually have to specify the moduleId setting here. So once we change that on every file inside](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=100.71) [of our application and note that it's not just templateUrl, if we're using a styleUrl, which we do use](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=106.71) [one of those in the Upvote component, here's the styleUrls, and again we've done the same thing,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=112.71) [we've stripped off the directory information. Once that's done in your application, the next thing that](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=119.71) [had to be fixed with our code specifically was how we were bringing in toastr and jQuery in our app.module.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=123.71) [Right up here at the very top, we were doing toastr and jQuery by just calling a declare var.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=129.71) [Let's go back to our original code so we can see that. Here we just did a declare let for toastr and jQuery.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=136.71) [But we can't do that with the compiler. It complains because it doesn't know about those variables.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=144.71) [Instead, we had to grab them off the window object. The compiler does know about the window object,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=148.71) [so it doesn't complain about that global variable. And then to access the toastr variable and the jQuery](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=153.71) [variable, we can't use the dot notation because typescript doesn't know about those properties.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=158.71) [Instead, we had to use the bracket notation. Most of the difficulties that we had were problems in our forms.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=164.71) [So let's go to the template for our create-event.component, and we'll start off with the original code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=170.71) [And see right here, we're grabbing newEventForm.controls.name, putting in a question mark so that it's](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=178.71) [nullable, and then looking at the invalid property. The problem is this name is not known by the compiler.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=184.71) [The controls property already exists, and it knows about that, but as far as the compiler's concerned,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=191.71) [it doesn't have a name property. There're a couple of ways to get around that.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=197.71) [One is there's a get function where we can pass in a string. Another one is to use bracket notation.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=201.71) [So let's look at what we had to do in order to make this work.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=206.71) [Notice that we've called newEventForm.form.get and then passed in the name property.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=209.71) [Whereas before we were calling it newEventForm.controls, now we're calling it newEventForm.form.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=215.71) [The this prefix isn't really required. We can take that off. So we have to change from .controls to .form](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=222.71) [and then call .get. We still use the question mark to make it nullable and then we can deal with asking for](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=228.71) [the invalid property or the touch property or any of those. Another thing we can't do is access the errors](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=233.71) [property to get to a specific error. We do that in our profile, so let's go to the profile.component.html.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=238.71) [Let's go back and look at the original code that we were doing.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=247.71) [Notice down here we have profileForm.controls.firstName, and it's got an errors property, which is a](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=251.71) [collection, and we're looking for the pattern property on that. This is the same exact problem we had with](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=256.71) [the controls collection. The properties on that just aren't known by the compiler.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=264.71) [So, I've got to access these in a different way. And how we did that was we called .get again in order to](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=266.71) [get the firstName, and then we call hasError and pass in which error we're looking for.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=275.71) [So, again, a change we've got to make in our project. Another thing you've got to do is make sure that](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=281.71) [when you're calling methods, you always call them with the right signature.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=286.71) [One place where that wasn't being done is when we were calling ngOnChanges.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=290.71) [Let's go back to our original code again. And this time, we're going to look at session-list.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=294.71) [And notice down here for ngOnChanges, I'm passing in no parameters.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=298.71) [Well, we implement the interface OnChanges, and the interface for that method actually receives a parameter.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=304.71) [So we had to change the signature of that in order to match up. What it receives is a changes object,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=310.71) [which is of type SimpleChanges. SimpleChanges was simply imported from @angular/core.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=318.71) [So I had to fix that interface as well in order for the AOT compiler to work.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=325.71) [Another thing that doesn't work with the AOT compiler is using properties inside of your templates that](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=331.71) [aren't marked as public. Now most things are marked public by default, such as all of these input variables here.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=337.71) [So unless you mark them private, you won't have a problem. But let's take a look at our nav.component.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=342.71) [Notice right here our AuthService is marked public. In our original code, in the nav.component,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=347.71) [when we brought in the AuthService, we marked it private. Now that's used in the template.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=355.71) [Here in the template, you can see that there're a few places where we check the auth component for testing](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=361.71) [to see if the user is authenticated or grabbing the currentUser. Since we're using it in the template,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=367.71) [it's got to be a public property. So, therefore, it's got to be marked public and not private.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=372.71) [And one more thing that you can do that the AOT compiler doesn't like is with ngModel variables,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=379.71) [we don't have to create a corresponding item in the component itself.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=385.71) [Let's take a look at our login component in our original code. We're going to look at the HTML.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=390.71) [And notice that the ngModel on this input box is set to userName, and the one on this input box is set to password.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=397.71) [If we go to the component itself, notice that the login.component class has no userName or password properties.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=404.71) [That doesn't fly with AOT compiler. So in our AOT code (let's go back to that and look at the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=412.71) [login.component), notice that we've created userName and password.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=420.71) [We've also created this mouseoverLogin. Let's look at what's going on in the template.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=425.71) [Notice that we're checking mouseoverLogin to see whether or not we display this error message.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=430.71) [And on the span that wraps around the Login button, we're actually setting mouseoverLogin to true and to false.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=437.71) [That's just fine with the Just-In-Time compiler, but with the AOT compiler, it needs to have that property](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=444.71) [exist on the component. So we had to create that property as well.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=8&mode=live&start=450.71)

[Running the AOT Compiler](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=9&mode=live)

[Now once you've fixed all the problems that the AOT doesn't like about your code, and, again, this is not](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=9&mode=live&start=1.75) [going to be an exhaustive list, writing your own application, you're probably going to discover other](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=9&mode=live&start=6.87) [things the AOT compiler doesn't like. Once you've got that all fixed, then you're ready to actually run](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=9&mode=live&start=11.87) [the AOT compiler. And you run the AOT compiler from the command line,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=9&mode=live&start=16.87) [and the way you do that is by calling ngc, which is, of course, underneath node\_modules/.bin/ngc,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=9&mode=live&start=20.87) [and then you give it a -p and your tsconfig file. Of course, you may want to create a script for this in](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=9&mode=live&start=29.87) [your package.json file so that it's easier to run with something like npm run aot.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=9&mode=live&start=35.87) [When you run the AOT compiler, if there's anything about your code it doesn't like, it will spit out an error.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=9&mode=live&start=42.87) [Some of the errors it will give you are easy to tell what they mean and what the problem is, some of them](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=9&mode=live&start=47.87) [not so much. But once you have all that fixed, then the AOT compiler will produce](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=9&mode=live&start=50.87) [the AOT directory, and that's right here. The output of this is going to be all of those module factories](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=9&mode=live&start=55.87) [and component factories that are produced by the AOT compiler. Once that's produced, we're now ready to](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=9&mode=live&start=62.87) [bundle it up with Rollup, which we'll do in the next section.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=9&mode=live&start=67.87)

[Tree Shaking and Bundling with Rollup](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live)

[With the AOT files produced, we can use Rollup to bundle, tree shake, and minify our code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=1) [Now as mentioned previously, at the current time, Rollup does not support code splitting.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=5.993) [That means lazy loaded modules don't work. So we've got to take one more step before we can use Rollup](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=11.993) [and remove our lazy loaded module. That's a relatively straightforward process to do.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=17.993) [In our app, the user.module was lazy loaded. This is the module right here.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=22.993) [Here're the routes to it. And if we go and look at our original code, what we care about is routes.ts](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=28.993) [inside of the root directory of our app. This is where we specify lazy loading by using this loadChildren](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=35.993) [property and just passing it in a string. That's what enables lazy loading.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=41.993) [We had to turn that off. So in order to turn that off in our code in the routes file, we actually imported](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=47.993) [the userRoutes. That wasn't being done before. Once we've imported the userRoutes, we simply mark them](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=54.993) [as children and pass them in to the route. You can see that I've commented out the loadChildren property.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=61.993) [And, instead, I'm using the children property and passing in those routes.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=67.993) [That will turn off the lazy loading of the module. And, therefore, we no longer need code splitting.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=71.993) [And Rollup will work just fine for us. So we go back out to the command line,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=75.993) [and we need to install Rollup and a couple of other libraries. So npm install rollup, and we also need](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=79.993) [rollup-plugin-node-resolve. This plugin is needed in order to make Rollup go through hour module tree](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=88.993) [that we saw in our previous section, to go through that correctly.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=95.993) [The details of why are beyond the scope of this course, but just know that this is a necessary plugin.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=99.993) [The next plugin we need is rollup-plugin-commonjs. The reason we need this plugin is because Rollup only](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=104.993) [works with es2015 modules that have the import and export statements that we've been using in all of our code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=113.993) [But there're lots of third-party libraries that aren't produced with es2015 modules.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=119.993) [For example, RxJS. RxJS is not produced using es2015 modules. It uses the node-style modules.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=125.993) [So this plugin allows Rollup to still process RxJS, which is absolutely necessary for our project because](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=132.993) [we use HTTP. And, finally, we need rollup-plugin-uglify. This is the minifier that we'll use to minify the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=138.993) [code and remove the whitespace like we showed before. Now I've already installed all of these, so I'm not](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=148.993) [going to run this command. But these are the pieces that you need to install in order to use Rollup.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=153.993) [The next step is to use a Rollup configuration file. So let's go back to our code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=158.993) [And let's look at the rollup.config.js file that I've created. You can see that I've imported Rollup](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=162.993) [itself plus all of the plugins that we've installed. And then I'm exporting a single object.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=169.993) [The properties of this object set the configuration settings. And let's look at those.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=174.993) [We've got the entry point, which is our main-aot file that we created.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=178.993) [We've got the destination of where the finished bundle file is going to be produced, and that's under the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=183.993) [dist folder in build.js. We can tell it whether or not we want to output a source map.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=188.993) [Format is iife. If you don't know what an iife is, it's a function that wraps around a small piece of code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=195.993) [It makes sure that the code doesn't conflict as we stick a whole bunch of JavaScript from different files](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=200.993) [into the same file. And then we have our plugin section where we configure each of the different plugins](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=204.993) [that we use. Commonjs and uglify, we simply call. But nodeResolve, we have to pass in jsnext: true](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=209.993) [and module: true. That's the configuration for our Rollup. And with that done, now all we need to do is call it from the command line.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=214.993) [Calling it is fairly easy. Put in node\_modules/.bin/rollup, and, of course, you could create a script](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=222.993) [inside of your package.json to make this a little bit easier. And we pass in a -c to tell it where the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=232.993) [configuration file is and pass in the name of our configuration file, which is rollup.config.js.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=238.993) [Once we run this, it's going to produce our build.js file.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=245.993) [I've already produced that, and we can see it here in the dist folder right here.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=249.993) [This is what it looks like. It's been completely minified, and so it's very difficult to read.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=254.993) [But we have the map file so that if we do need to debug, we can get to readable code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=259.993) [Now that's the last piece of our puzzle, and we've got our project ready to deploy to production.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=263.993) [At this point, it's a very simple copy and paste operation. We'll need the index file and the build.js](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=269.993) [file and any other files that are referenced by the index file. In our case, there're a few files that are](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=273.993) [referenced into the node\_modules. So we'll have to make sure that our production server has those](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=279.993) [node\_modules installed. With that, our application is deployed, and we're good to go.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=285.993) [Now with all these pieces in place, let's take a look at the change that we've made in the size of our](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=289.993) [download and the number of requests that we're making. Last time we ran this, we had about 200 requests](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=294.993) [and a little less than 3 MB of code. So let's go to the browser.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=300.993) [We'll open up our Network tab, and we'll refresh. And you can see that we're now only making 19 requests](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=305.993) [and transmitting only 1.1 MB of data. So that is a significant decrease to what we had way at the very](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=311.993) [beginning when we had around 3-1/2 MB and 400 requests. Now we're down to just this small amount.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=318.993) [In the next sections, we'll look at using Webpack to take our project to production.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=10&mode=live&start=325.993)

[Bundling with Webpack](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live)

[In this section, we're going to look at using Webpack to take our project to production.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=1) [Now, fortunately, this can actually be pretty easy because the CLI uses Webpack.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=4.667) [So if you're using the CLI for your project, you're already using Webpack.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=9.667) [In our case, we've been using SystemJS the whole time, so in order to switch it over to Webpack,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=13.667) [what I did was create a blank CLI project, copy our code into that, fiddle with it to get it to work,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=18.667) [and then we've got a running project underneath the CLI, which means that we've got Webpack running.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=24.667) [And it's going to do our production build for us. Now what's really nice about using this CLI is that taking](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=28.667) [your project to production is already built in to the CLI. It's as simple as starting up your project with](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=33.667) [a different command. So let's first go into that project. I'm going to go up one level.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=39.667) [I'm going to ng2-fundamentals-cli. You won't have this installed, of course, but we're going to look at it](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=47.667) [here and run it. Now to run the CLI in production mode, it's as simple as npm run start:prod.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=54.667) [That will start up a server in production mode, which means that we're going to be doing bundling,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=62.667) [we're going to have minification, we're going to have tree shaking, and, best of all, Webpack supports code](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=66.667) [splitting, so our lazy loaded module will still work. We've started up our project.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=73.667) [It is important to note that the CLI is not running our back-end server where our API is and the data is stored.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=79.667) [So I have to have that running in another command line window, and I've configured the CLI so that any](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=85.667) [requests going to /api will go to the web server instead of to the CLI server.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=89.667) [Now with that up and running, we're going to go to our web browser. And instead of localhost:8808, we're going to go to port 4200.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=96.667) [That's where the CLI by default serves up our code. And, of course, let's look at the network tabs.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=106.667) [Let's refresh. And we can see down here at the bottom, we're making 20 requests and transmitting 1.5 MB of data.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=113.667) [And what's really nice is that we are getting code splitting. So if we go to some component that's inside](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=121.667) [of our user module, we're going to download another bundle. So let's click into Login, and you can see that](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=124.667) [we've made another request. This long number.chunk.js, that is the bundle for the user module.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=131.667) [Now it's only 7-1/2 KB. So we haven't really saved a whole bunch of bytes on our initial download,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=140.667) [but if necessary we could put more code into the lazy loaded module.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=147.667) [Now one thing that's really important to notice is that even though we're actually transmitting more data](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=152.667) [than we were with Rollup, 1-1/2 MB instead of 1.1, right now the CLI is not able to use the Ahead-of-Time compiler.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=156.667) [That's still being worked on. So if the Ahead-of-Time compiler's working, that initial download size is](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=165.667) [going to go down again probably roughly to where it was at with Rollup.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=170.667) [So that's what it looks like using Webpack. And as you can see whether you use Webpack or you use Rollup,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=173.667) [we're getting roughly the same amount of space savings, but Webpack does have the benefit that it does](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=179.667) [support code splitting, although it's possible that that will change in Rollup and Rollup will start](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=186.75) [supporting code splitting in the future as well.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=11&mode=live&start=190.667)

[Optimistic Bundle Download](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live)

[Now the last thing I want to do is show how to turn on](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=1) [eager lazy loading of your modules. Remember how we didn't get this last chunk to download until we went](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=3) [in to the login page or some other page in the user module. Well we want to fix that.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=9.723) [We want to download the user module as soon as there's available bandwidth.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=14.723) [We still don't want to wait to render the application until it's downloaded.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=18.723) [We want it to render beforehand. But we want it to download as soon as there's available bandwidth.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=21.723) [If this is a particularly large module when we click the Login button, there could've been a delay for the user.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=27.723) [We don't want that to happen. To turn this on is a very simple process.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=32.723) [Let's go into our code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=35.723) [Now here I'm in the CLI version of our code, but this does work as well in the regular version of our code.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=38.723) [Whether we're going in production or we're in development mode, you can still do this.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=43.723) [What I want to do is I want to import a preloading strategy from the router.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=47.723) [So from the Angular router in addition to the router module, I'm also going to import the PreloadAllModules symbol.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=52.723) [Now, I'm going to go down to where we call the forRoot method on the RouterModule, and I'm going to add a](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=61.723) [second parameter to that call. This is an object. That object has a preloadingStrategy key, and the value](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=69.723) [of that key is going to be the PreloadAllModules identifier that we imported.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=76.723) [With that in place, if I save and go back to the browser,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=83.723) [let's go first to the events, and then we're going to refresh the page.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=88.723) [And notice we've got 21 requests made. If I click on the Login button, notice that the number of requests](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=94.723) [has not gone up at all. Instead, we can see right here that the chunk has already been downloaded.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=100.723) [That is the user module chunk. That was downloaded as soon as the app was ready to render and there was](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=107.723) [available bandwidth. Now in addition to the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=112.723) [PreloadAllModules strategy, we can actually create a custom strategy where we tell Angular to only download](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=114.723) [the modules that we want eagerly and other ones we want to wait until they're actually needed.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=121.223) [Doing that is beyond the scope of this course, but that is an option.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=12&mode=live&start=126.723)

[Summary](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live)

[In this module, we looked at a lot of different techniques to get your code ready for production.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=1) [We started off by looking at linting. This is not a requirement for getting your code up to production,](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=5.053) [but it is something that you want to do if you've got a fairly large project and multiple developers working on it.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=11.053) [You can certainly use it on small projects. But the bigger the project gets, the more important linting is.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=16.053) [We looked at how to tune our RxJS requests to cut down the number of requests we're making.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=21.053) [We looked at how to enable prod mode. We looked at how to use the Ahead-of-Time compiler.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=26.053) [We looked at bundling and tree shaking in order to reduce the number of requests we're making and the](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=31.053) [download size of our code. We looked at using Webpack instead of SystemJS.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=34.053) [And, finally, we looked at using optimistic bundle downloading in order to eagerly download our lazy loaded modules.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=39.053) [Lazy loaded modules allow us to minimize the initial download size, and using optimistic downloading means](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=46.053) [that we can avoid any delay when a user gets to the part of an application that isn't in that initial bundle.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=51.053) [We looked at all of these things from a very high level so that we understand the moving parts.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=57.053) [Getting them working with your project will require further research.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=62.053) [But now you know what the different pieces do and how they interact.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=64.053) [Finally, on behalf of myself and Jim Cooper, I want to thank you for watching our course.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=68.053) [We hope you got out of it as much education and enjoyment as we did.](https://app.pluralsight.com/player?course=angular-fundamentals&author=joe-eames&name=angular-fundamentals-m16&clip=13&mode=live&start=72.255)

**Performance:** we work on performance by making some changes to the project. We make sure that we don’t import the whole rxjs library, but that we import only the services that we need from rxjs. This prevent unneceray http calls being made by the rxjs service.

We always run tslint to make sure that the code is in compliance and has the right format.

Then, we build the project with ng build using the prod property to minify the code.

**Build Angular Project:** ng build–env=prod

**Build and minify Angular Project:** ng build –prod –env=prod

Npm install –g @angular/compiler-cli @angular/platform-server

We use webpack to package the code because we use angular cli. If somebody does not use the cli, then we may be using systemjs.

We also do AOT (ahead of time compiling) to do faster rendering, fewer requests, smaller angular framework download, detect template errors, and better security since the templates are compiled.

Development = Just In Time (JIT) compiler

Production = AOT compiler

Npm install –g @angular/compiler-cli @angular/platform-server

**Simple Copy Deployment:**

**AOT No-No’s:**

Templates should not have:

1 form.controls.controlName

2 control.error?.someError

3 default exports

4functions in providers, routes or declarations of a module

any field used in a template, including inputs, must be public

5 you can’t declare var for globals