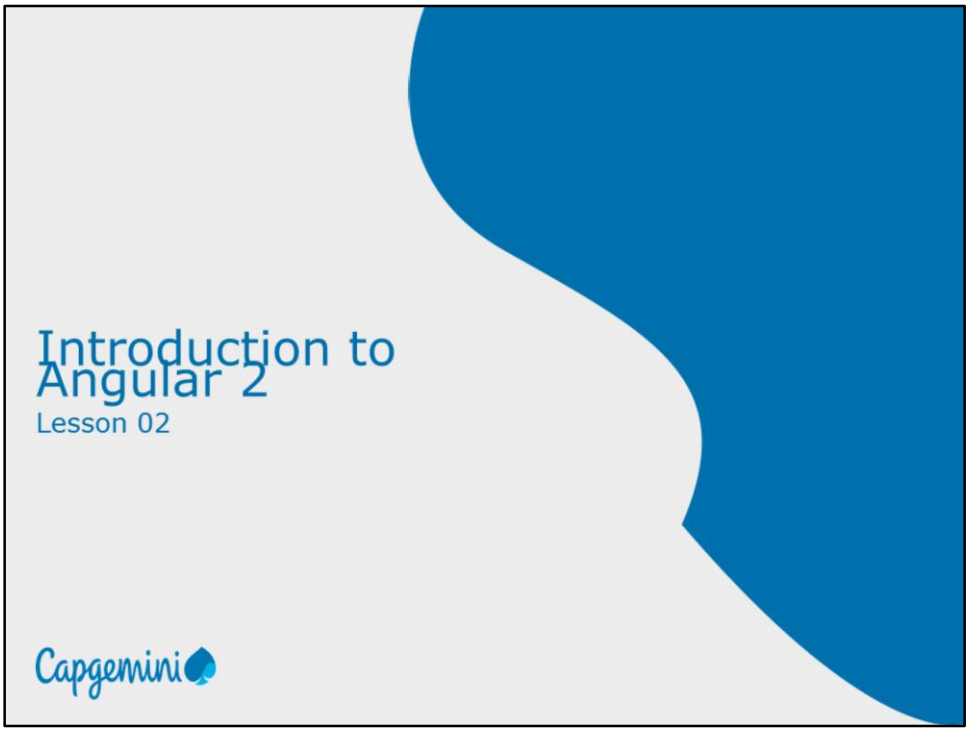
Instructor Notes:

Add instructor notes here.



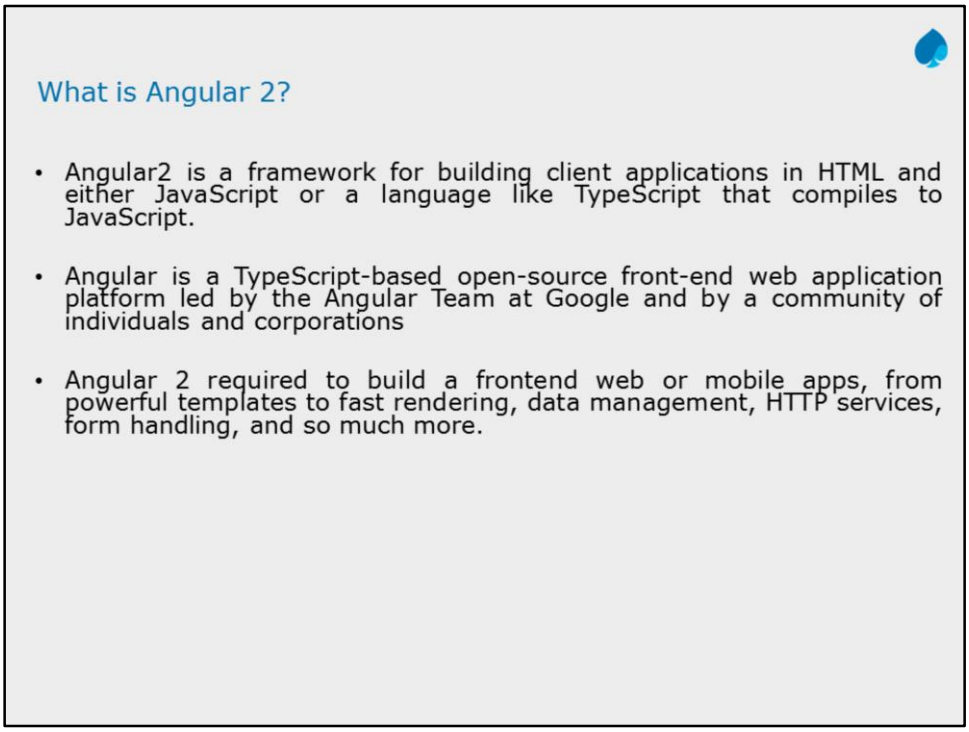
Instructor Notes:

Add instructor notes here.

Lesson Objectives

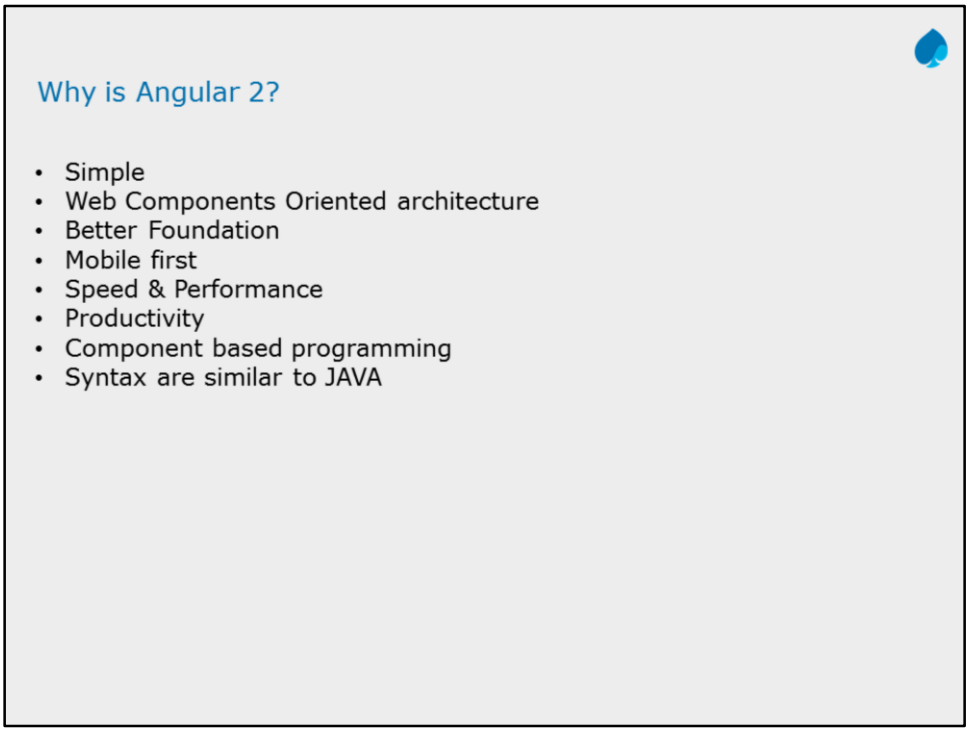
- Introduction of Angular2
- What is nodejs
- Building blocks of Angular2
- What is module-Root Module
- First Application with Angular2

Instructor Notes:



Angular is a framework for building client applications in HTML and either JavaScript or a language like TypeScript that compiles to JavaScript.
The framework consists of several libraries, some of them core and some optional.

Instructor Notes:



We can write Angular applications by composing HTML *templates* with Angularized markup, writing *component* classes to manage those templates, adding application logic in *services*, and boxing components and services in *modules*.

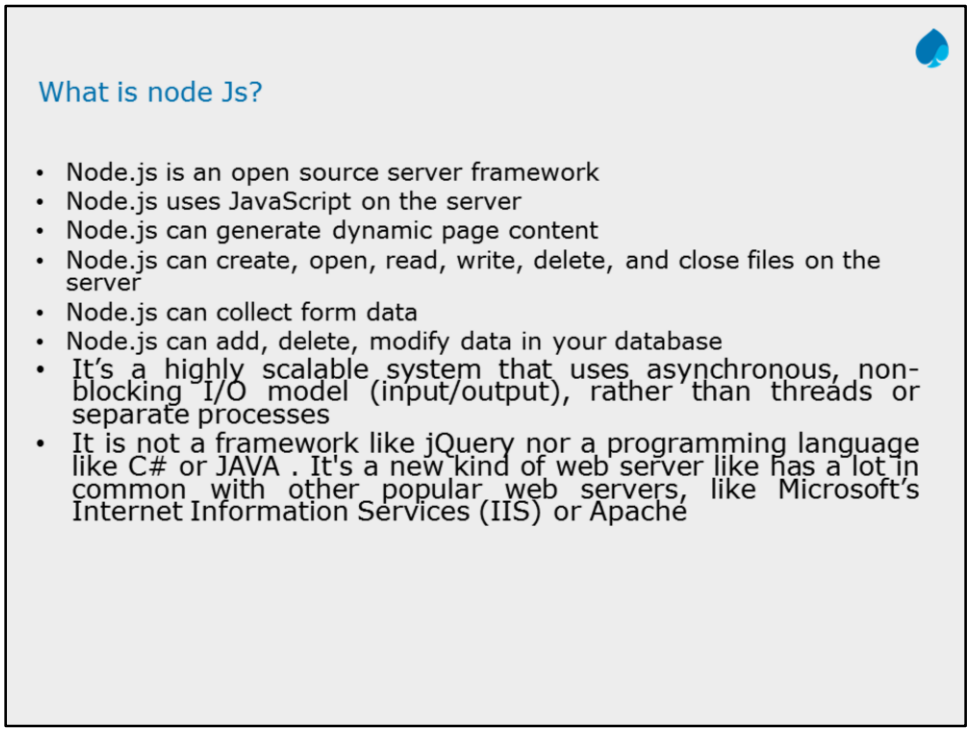
Angular makes HTML more expressive, It powers up HTML with features such as if conditions, for loops and local variables.

Angular has powerful data binding. We can easily display fields from our data model, track changes and process updates from the user.

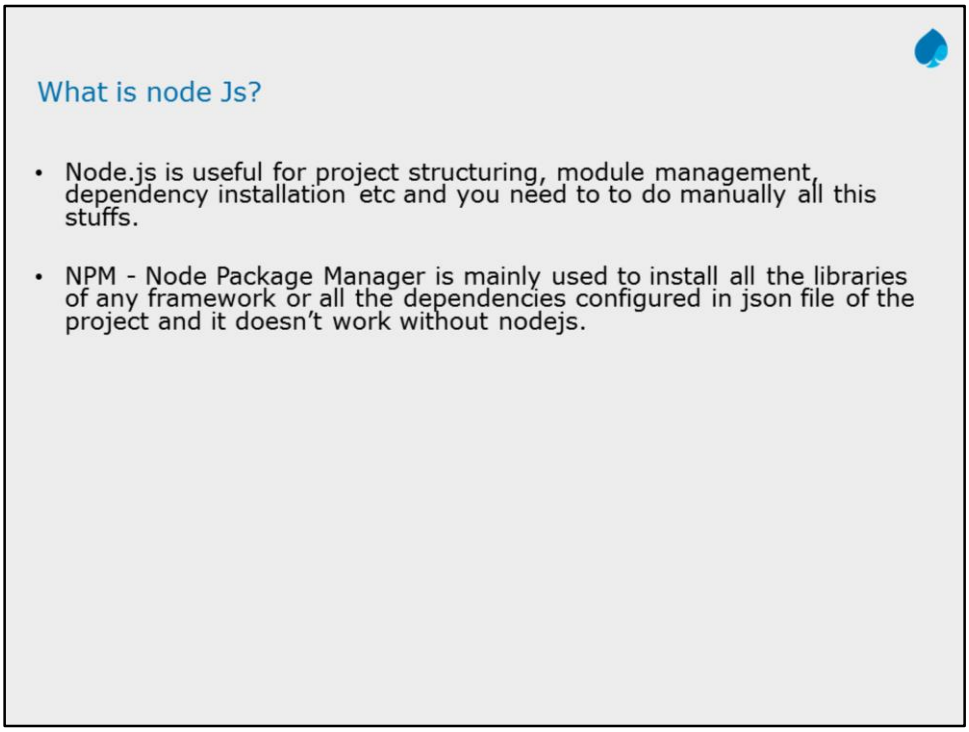
Angular promotes modularity by design so that the applications become a set of building blocks making it easier to create and reuse contents.

Angular has built-in support for communication with a backend service this makes it easy for Web applications to integrate with the backend service to GET and POST data or execute server side business logic.
Angular 2 was built for speed, It has faster initial loads faster change detection and improved rendering times.
Angular 2 is modern it takes advantage of features provided in the latest JavaScript standards such as classes, modules and decorators.
It leverages web component technologies for building reusable user interface widgets.
It supports both modern and legacy browsers like Chrome, Firefox and Internet Explorer back to IE 9.
It has a simplified API. It has fewer built-in directives to learn simple binding and a lower overall concept count.
It enhances productivity to improve day to day workflows

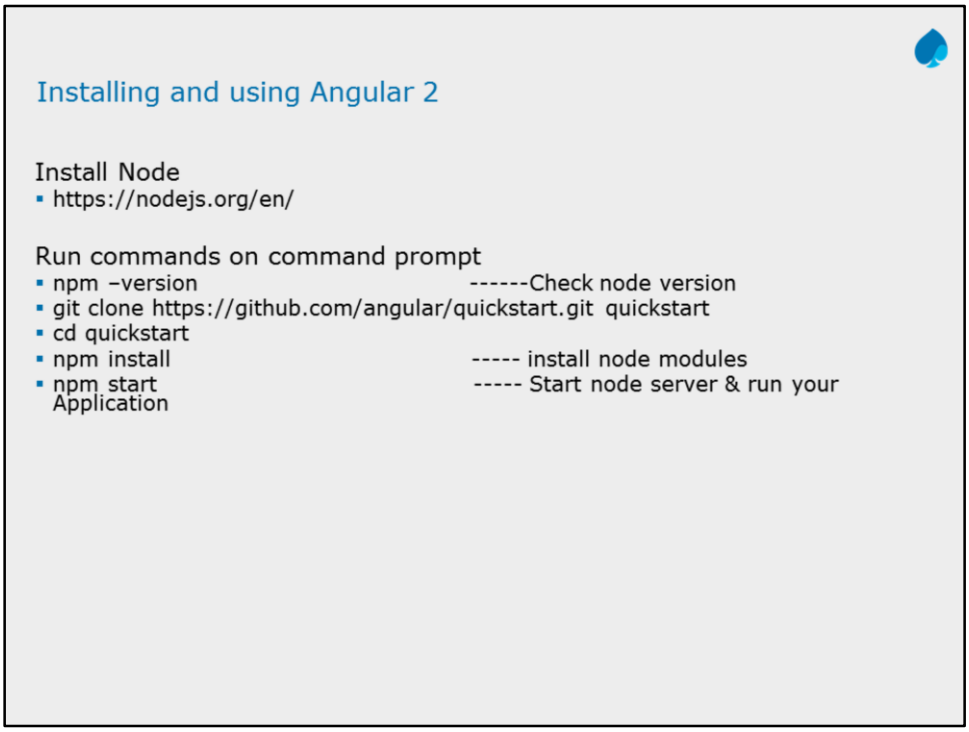
Instructor Notes:



Instructor Notes:

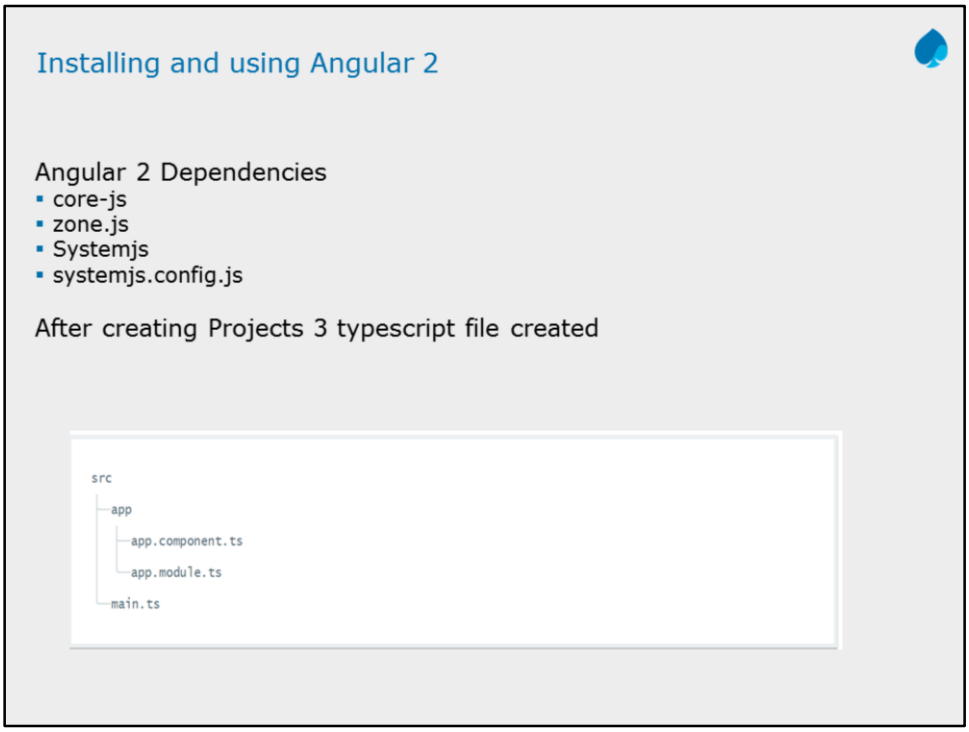


Instructor Notes:



If you have internet use git otherwise use basic demo
If you have internet use npm install -g @angular/cli & use ng serve

Instructor Notes:

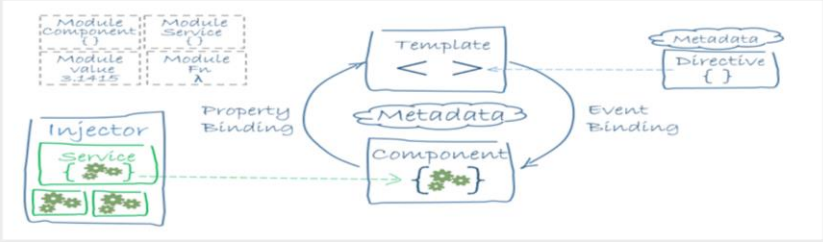


app/app.component.ts---→
Defines the same AppComponent as the one in the QuickStart playground. It is the root component of what will become a tree of nested components as the application evolves.
app/app.module.ts-→
Defines AppModule, the root module that tells Angular how to assemble the application. Right now it declares only the AppComponent. Soon there will be more components to declare.
main.ts--→
Compiles the application with the JIT compiler and bootstraps the application's main module (AppModule) to run in the browser. The JIT compiler is a reasonable choice during the development of most projects and it's the only viable choice for a sample running in a *live-coding* environment like Stackblitz. You'll learn about alternative compiling and deployment options later in the documentation.

Instructor Notes:

Blocks of Angular 2

The architecture diagram identifies the eight main building blocks of an Angular application-



- Modules
- Templates
- Data binding
- Services injection

Components
Metadata
Directives
Dependency

We can write Angular applications by composing HTML *templates* with Angularized markup, writing *component* classes to manage those templates, adding application logic in *services*, and boxing components and services in *modules*.
Then we launch the app by *bootstrapping* the *root module*. Angular takes over, presenting your application content in a browser and responding to user interactions according to the instructions we have provided.
Then we launch the app by *bootstrapping* the *root module*. Angular takes over, presenting our application content in a browser and responding to user interactions according to the instructions you've provided.

Module
Optional feature
Useful if you are using TypeScript which allows you to use interface or classes
export class AppComponent is like saying that this class is going to be public
Use relative file paths for importing modules
Component class is something you'd export from a module.

Component
Components controls Views
Logic to support the view can be inside a class
Angular creates/destroys components as user moves through UI

Template
A form of HTML that describes how to render the Component. It looks mostly like HTML syntax except if you add Angular keywords in them.

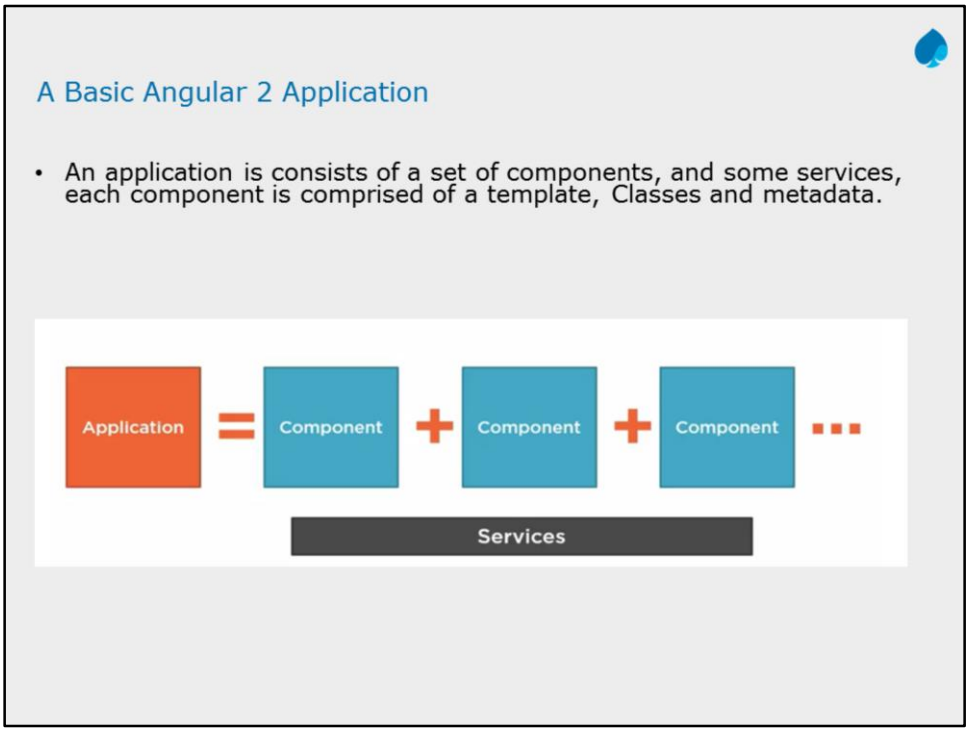
Metadata
Some ***@Component*** configuration options:
selector: css selector to be applied to that html element
templateUrl: address of the component itself
directives: array of components/directives that this component itself requires to function properly
providers: an array of *dependency injection providers* for *services*

Instructor Notes:

Data Binding
Following are the four possible ways of data binding:
** **{{hero.name}}** <hero-detail** [hero]="selectedHero"**></hero-detail> <div** (click)="selectHero(hero)"**> <input** [(ngModel)]="hero.name"**>**
The “interpolation” displays the component’s hero.name property value within thetags
The [hero] property binding passes the selectedHero from the parent HeroListComponent to the hero property of the child HeroDetailComponent
The (click) event binding calls the Component’s selectHero method when the user clicks on a hero’s name
Two way data binding combines property and event binding in a single notation using ngModel directive

Service
It can be any value, function or feature that works well.
Dependency Injection
A way to supply a new class instance with all the requirements. In TypeScript this can be achieved by providing everything inside the constructor.
An Injector maintains a list of service instances it has created previously so that it can reuse those if needed. The way it achieves this is by utilizing provider which is used within each Component
Directive
Class with directive metadata. Even Components are directives - directive with templates. Two other examples are:
Structural: They alter layout by adding, removing, and replacing elements in DOM
Attributes: Attribute directives alter the appearance or behavior of an existing element. In templates they look like regular HTML attributes, hence the name
Example:
The ngModel directive, which implements two-way data binding, is an example of an attribute directive.
<input [(ngModel)]="hero.name"**>**
Other examples: ngSwitch, ngStyle, ngClass

Instructor Notes:



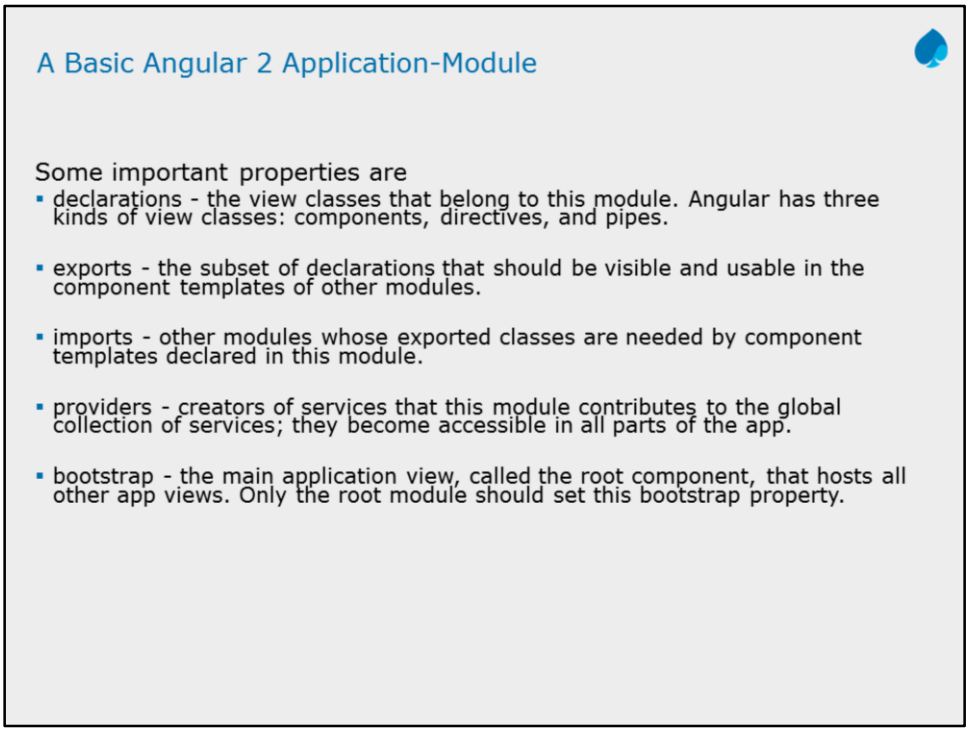
Angular Application is consists of set of components & some services .An Angular module, whether a *root* or *feature*, is a class with an @NgModule decorator.

Instructor Notes:

A Basic Angular 2 Application- Module

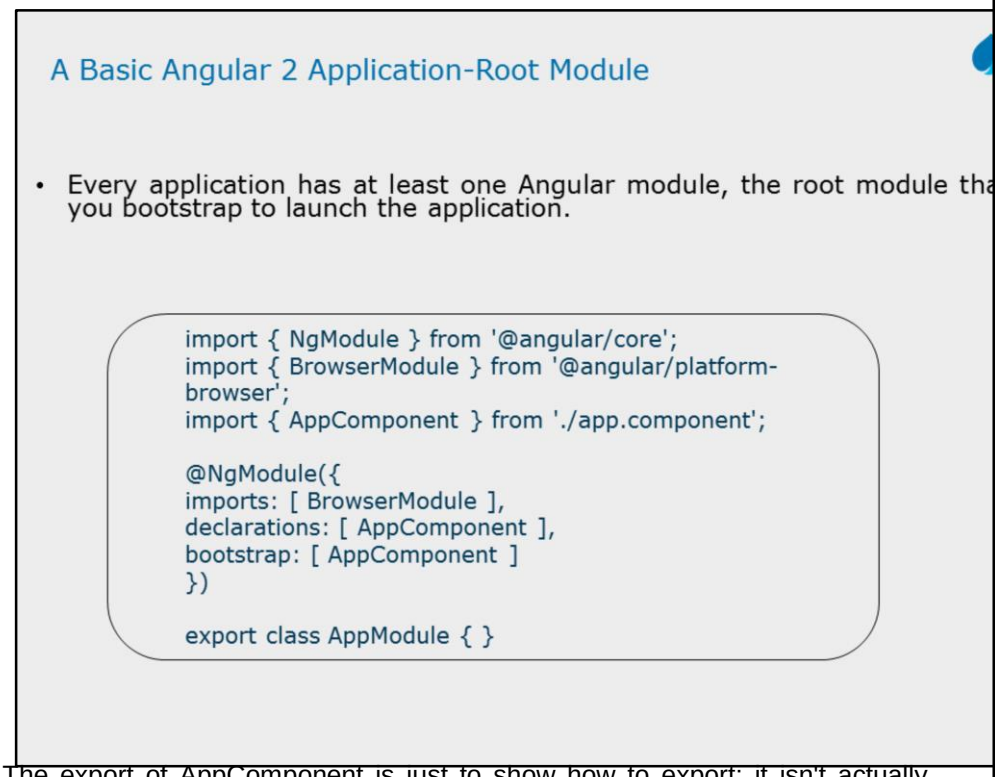
- Angular apps are modular and Angular has its own modularity system called *Angular modules* or *NgModules*.
- Every Angular app has at least one Angular module class, the root module, conventionally named AppModule
- An Angular module, whether a root or feature, is a class with an @NgModule decorator
- NgModule is a decorator function that takes a single metadata object whose properties describe the module.

Instructor Notes:



Instructor Notes:

Add instructor notes here.

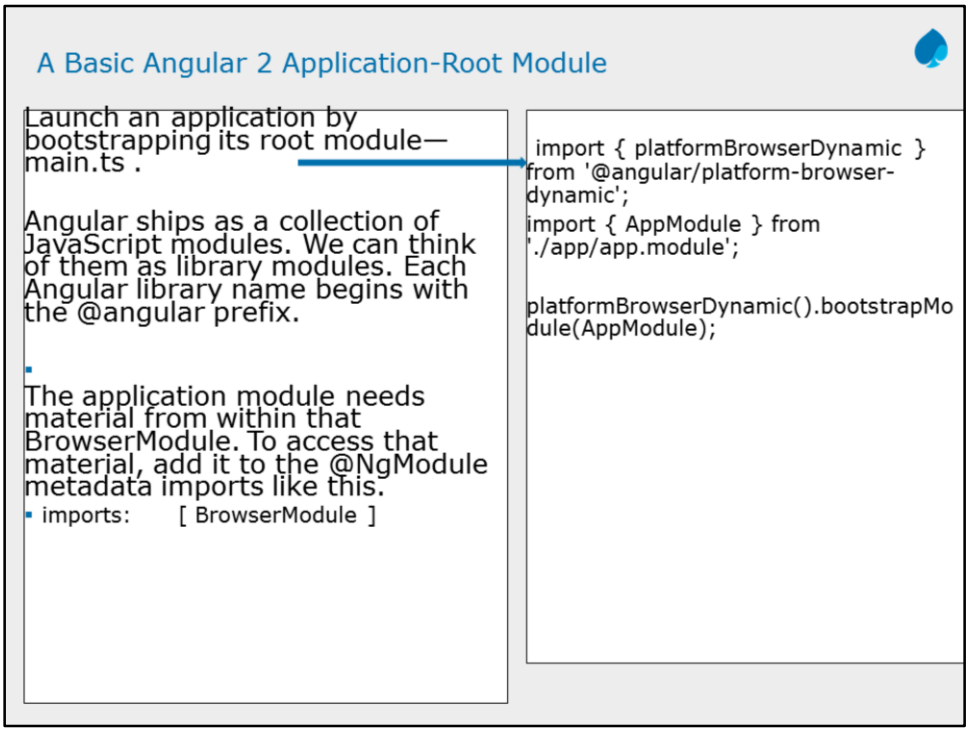


The export of AppComponent is just to show how to export; it isn't actually necessary in this example. A root module has no reason to *export* anything because other components don't need to *import* the root module.

Launch an application by *bootstrapping* its root module. During development you're likely to bootstrap the AppModule in a main.ts file like this one.

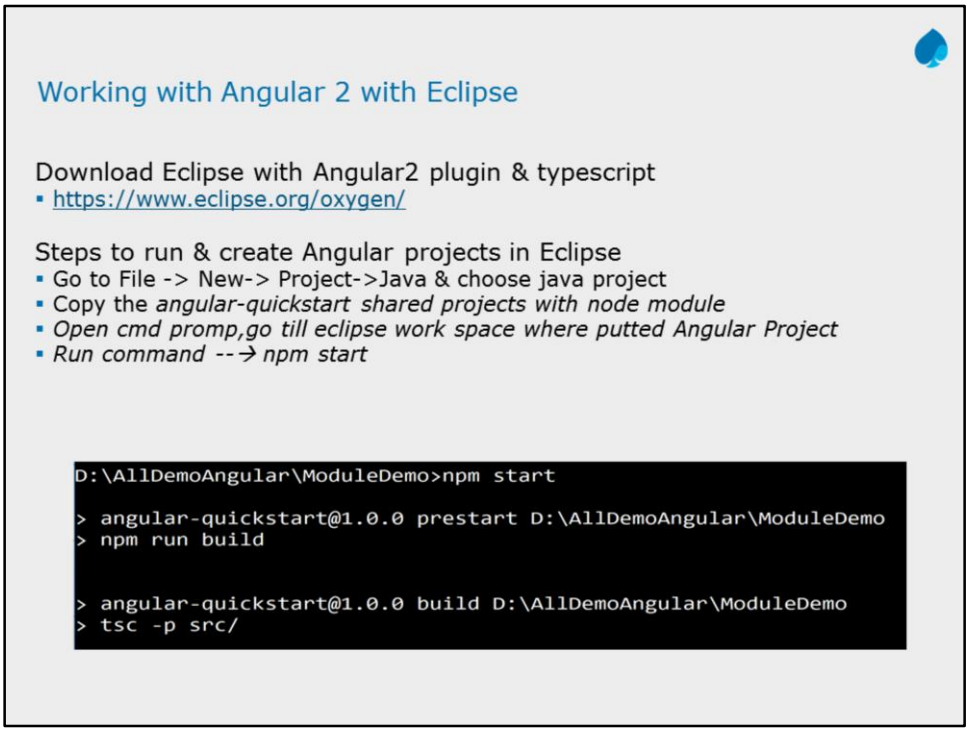
After the import statements, you come to a class adorned with the **@NgModule** *decorator*
The @NgModule decorator identifies AppModule as an Angular module class (also called an NgModule class). @NgModule takes a *metadata* object that tells Angular how to compile and launch the application.
imports — the BrowserModule that this and every application needs to run in a browser.
declarations — the application's lone component, which is also ...
bootstrap — the *root* component that Angular creates and inserts into the index.html host web page.

Instructor Notes:



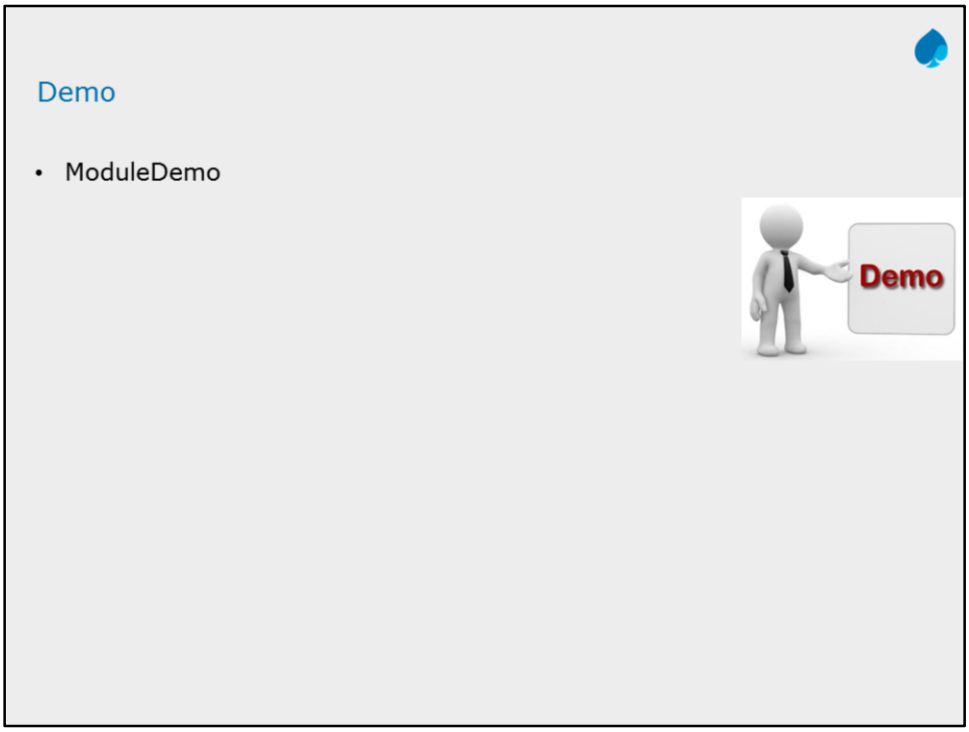
There are many ways to bootstrap an application. The variations depend upon how you want to compile the application and where you want to run it.
In the beginning, you will compile the application dynamically with the *Just-in-Time (JIT)* compiler and we run it in a browser.
The recommended place to bootstrap a JIT-compiled browser application is in a separate file in the src folder named src/main.ts
The *bootstrapping* process sets up the execution environment, digs the *root* AppComponent out of the module's bootstrap array, creates an instance of the component and inserts it within the element tag identified by the component's selector.
The AppComponent selector — here and in most documentation samples — is my-app so Angular looks for a <my-app> tag in the index.html
<my-app><!-- content managed by Angular --></my-app>

Instructor Notes:



Instructor Notes:

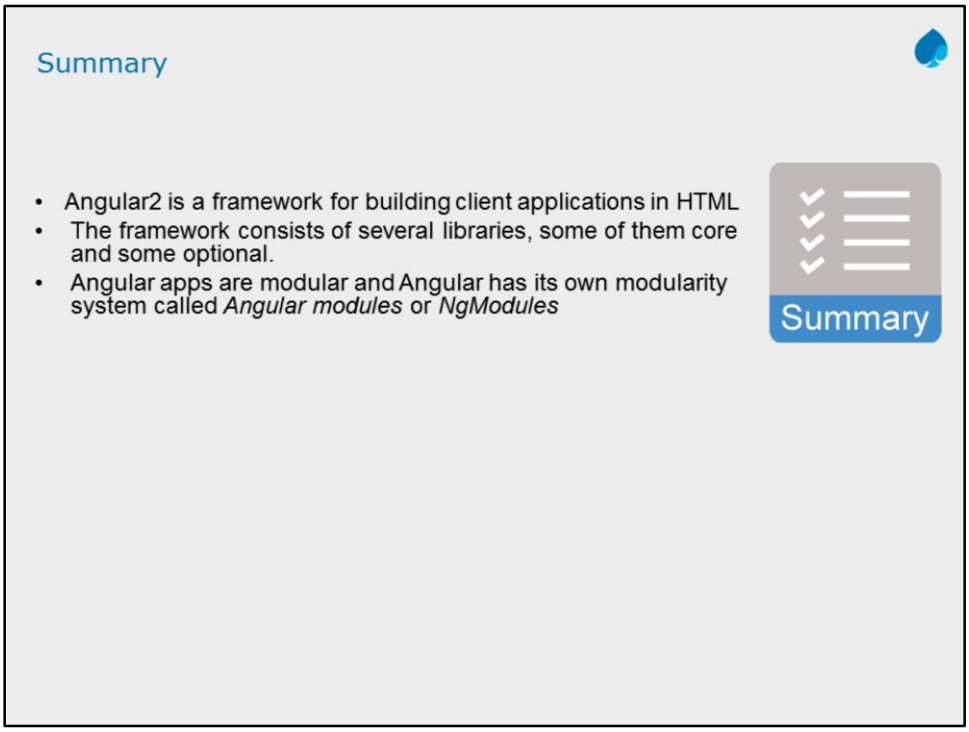
Add instructor notes here.



Add the notes here.

Instructor Notes:

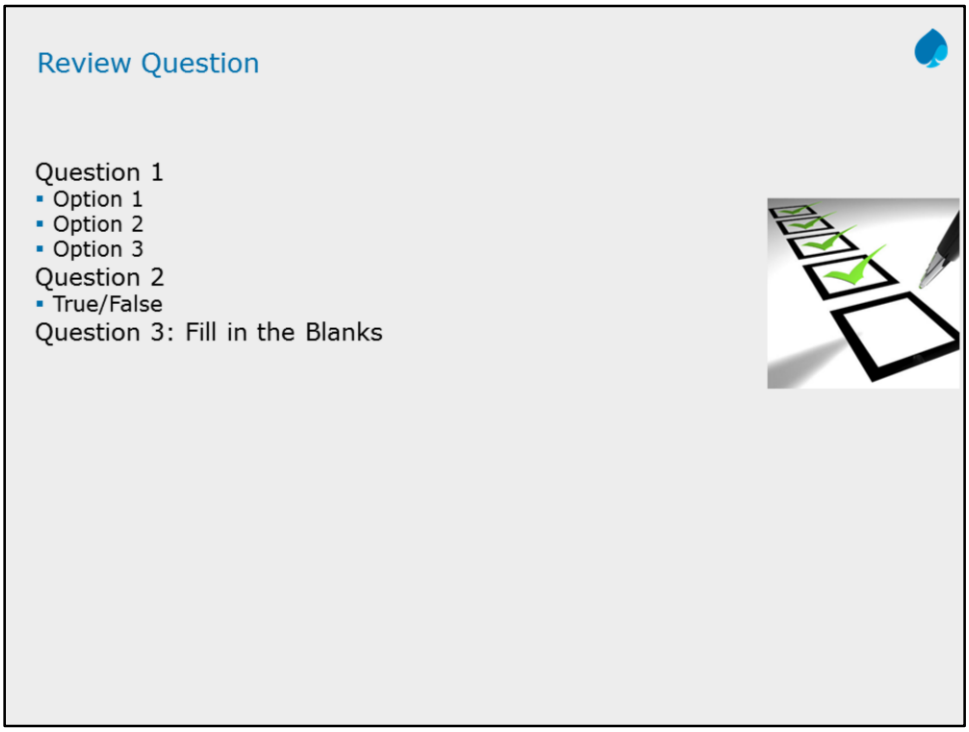
Add instructor notes here.



Add the notes here.

Instructor Notes:

Add instructor notes here.



Add the notes here.