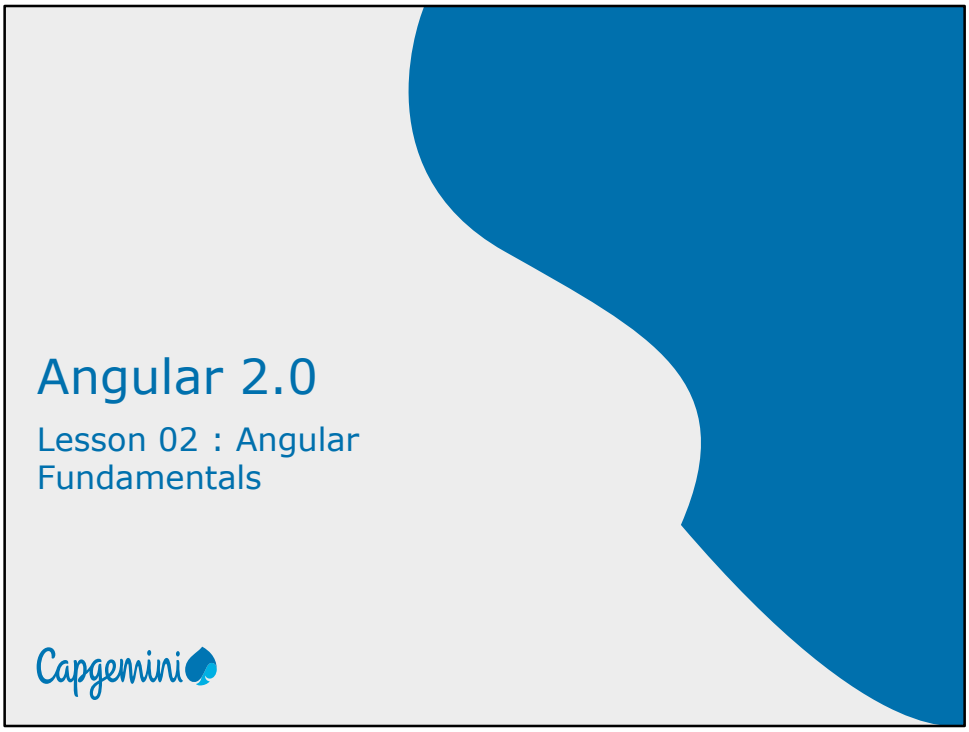
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.

Lesson Objectives

- Introduction of Angular2
- What is node JS?
- Building blocks of Angular2
- What is module-Root Module?
- First Application with Angular2

Instructor Notes:

What is Angular 2?

- Angular 2 is the next version of Google's massively popular MV* framework.
- Angular2 is a framework for building client applications in HTML and either JavaScript or a language like TypeScript that compiles to JavaScript.
- Angular is a TypeScript-based open-source front-end web application platform led by the Angular Team at Google and by a community of individuals and corporations.
- Angular 2 required to build a frontend web or mobile apps, from powerful templates to fast rendering, data management, HTTP services, form handling, and so much more.

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:

Why Angular 2?

- Simple
- Web Components Oriented architecture
- Better Foundation
- Mobile first
- Speed & Performance
- Productivity
- Component based programming
- Syntax are similar to JAVA

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:

Add instructor notes here.

Why Angular 2?

Instructor Notes:

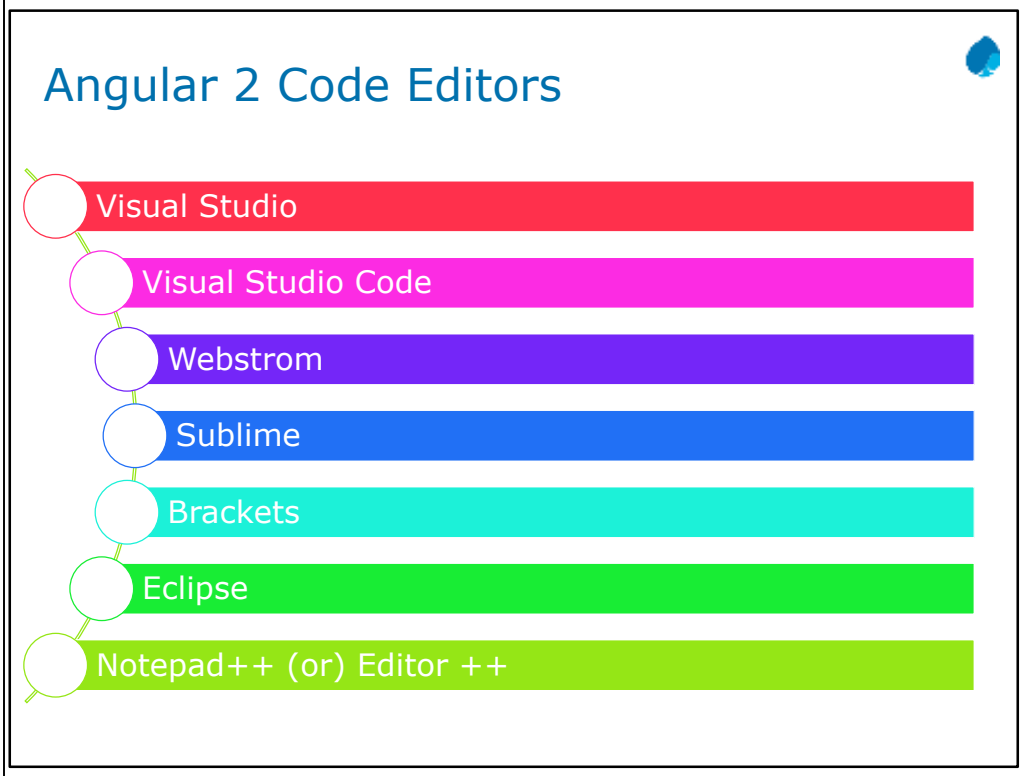
Add instructor notes here.

Why Angular 2?

- 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

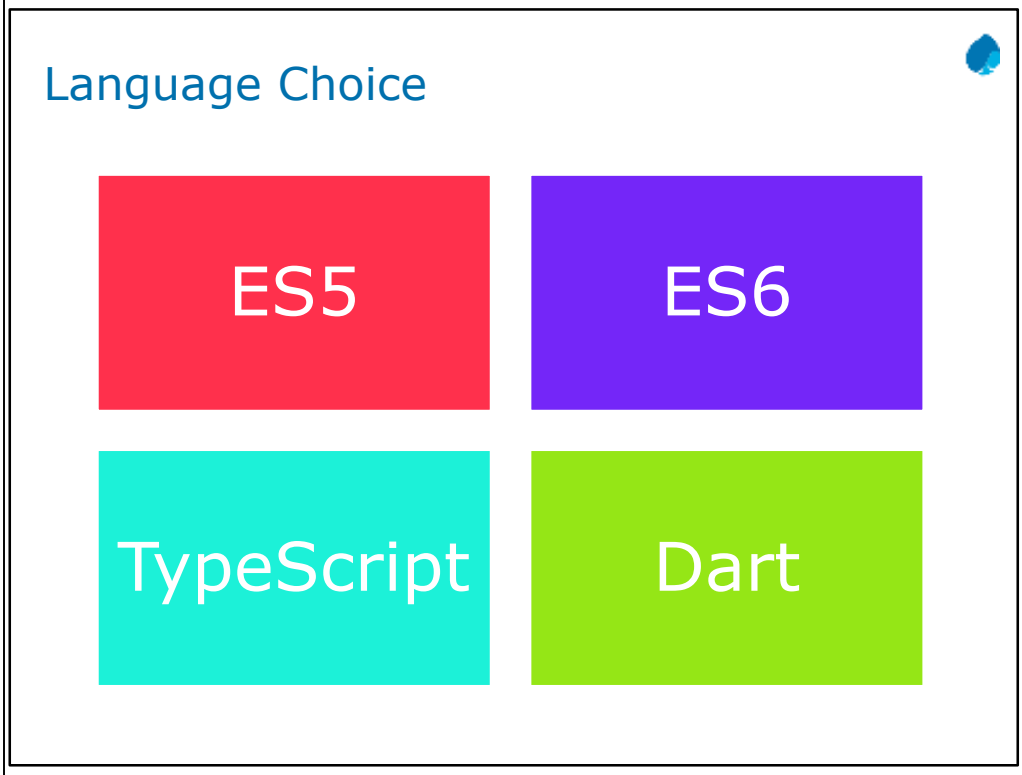
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.



ECMAScript, or ES:

•ES 3 is supported by older browsers.
•ES 5 is the version currently supported by most modern browsers.
•ES 6 specification was recently approved and renamed ES 2015(Most browsers don't yet support ES 2015).

TypeScript:

•TypeScript is the superset of JavaScript and must be transpiled.
•TypeScript has great tooling.
•Inline documentation.
•Syntax checking.
•Code navigation.
•Advanced refactorings (The Angular team itself takes advantage of these benefits and uses TypeScript to build Angular 2).

Dart

Dart is a non-JavaScript based object to built Angular2

Instructor Notes:

What is node Js?

- Node.js is an open source server framework
- Node.js uses JavaScript on the server
- Node.js can generate dynamic page content
- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data
- Node.js can add, delete, modify data in your database
- It’s a highly scalable system that uses asynchronous, non-blocking I/O model (input/output), rather than threads or separate processes
- It is not a framework like jQuery nor a programming language like C# or JAVA . It's a new kind of web server like has a lot in common with other popular web servers, like Microsoft’s Internet Information Services (IIS) or Apache

Instructor Notes:

What is node JS?

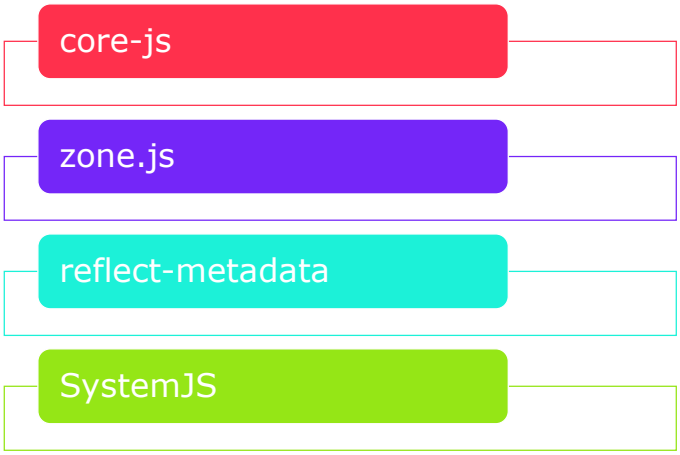
- Node.js is useful for project structuring, module management, dependency installation etc. and you need to do manually all this stuffs.
- NPM - Node Package Manager is mainly used to install all the libraries of any framework or all the dependencies configured in json file of the project and it doesn't work without node JS.

Instructor Notes:

Add instructor notes here.

Angular 2 Dependencies

- To run Angular 2, we depend on these four libraries:.



To include them, add the following inside your <head>

```
<script src="node_modules/core-js/client/shim.min.js"></script>
<script src="node_modules/zone.js/dist/zone.js"></script>
<script src="node_modules/reflect-metadata/Reflect.js"></script>
<script src="node_modules/systemjs/dist/system.src.js"></script>
```

Core-js
It provides shims so that legacy JavaScript engines behave as closely as possible to ECMAScript 6. It is not strictly needed for newer versions of Safari, Chrome, etc. but it is required for older versions of IE.

Zones
Zone.js is a library used by Angular, primarily for detecting changes to data

Reflect Metadata
Angular itself was written in Typescript, and Typescript provides annotations for adding metadata to code. The reflect-metadata package is a polyfill that lets us use this metadata.

SystemJS
SystemJS is a module loader. That is, it helps us create modules and resolve dependencies.

Instructor Notes:

Installing and using Angular 2

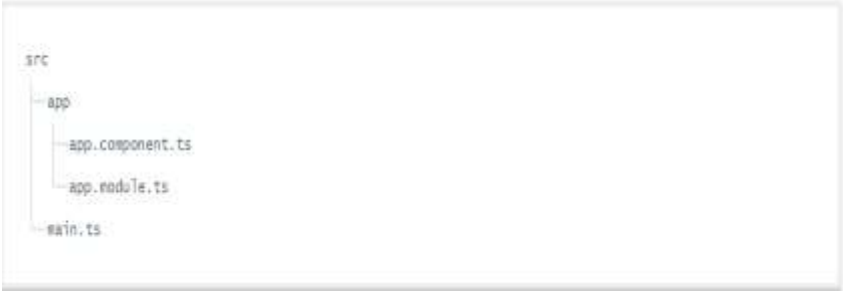
- Install Node
  - https://nodejs.org/en/
- Run commands on command prompt
  - npm –version --- Check node version
  - git clone https://github.com/angular/quickstart.git quickstart
  - cd quickstart
  - npm install --- install node modules
  - npm start ---Start node server & run your Application

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:

Installing and using Angular 2 (Contd…)

- Angular 2 Dependencies
  - core-js
  - zone.js
  - Systemjs
  - systemjs.config.js
- After creating Projects 3 typescript file created

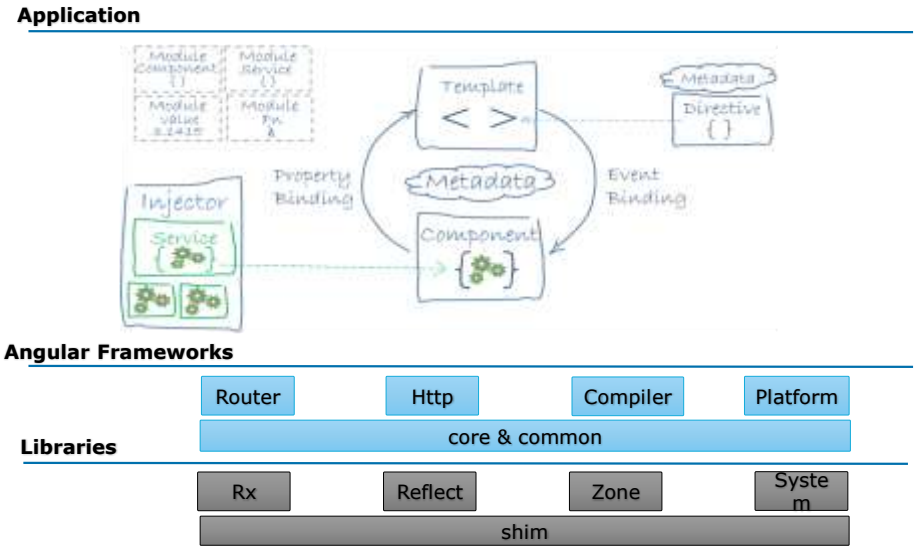


app/app.component.ts---→
Defines the same AppComponent as the one in the QuickStart playground. It is the root component of what will become a tree of nested components as the application evolves.
app/app.module.ts-→
Defines AppModule, the root module that tells Angular how to assemble the application. 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:

Add instructor notes here.

Building Blocks of an Angular 2



Instructor Notes:

Building Blocks of an Angular 2

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

•Modules	Components
•Templates	Metadata
•Data binding	Directives
•Services	Dependency injection

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.

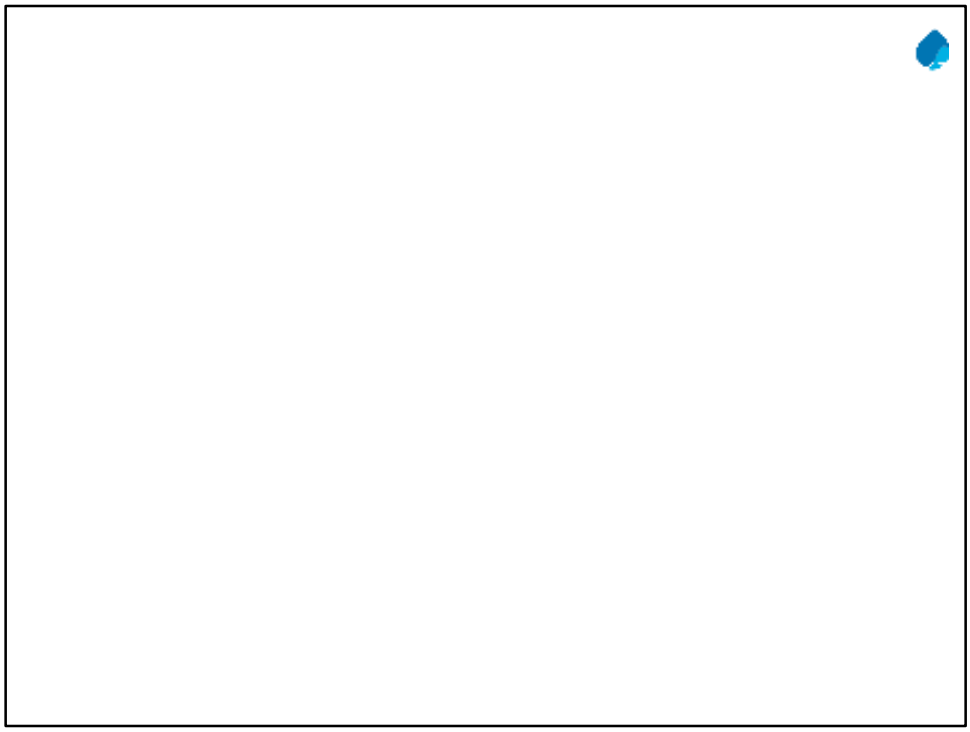
Module
Optional feature
Useful if you are using TypeScript which allows you to use interface or classes
export class AppComponent is like saying that this class is going to be public
Use relative file paths for importing modules
Component class is something you'd export from a module.

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:



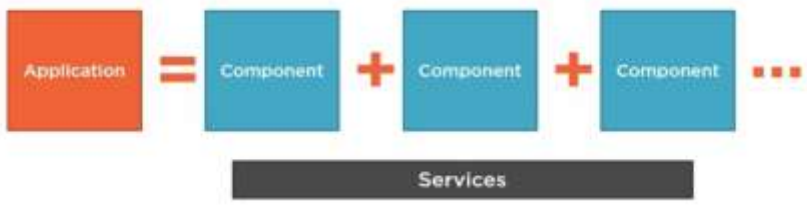
Data Binding
Following are the four possible ways of data binding:
** **{{hero.name}}** <hero-detail** [hero]="selectedHero"**></hero-detail> <div** (click)="selectHero(hero)"**> <input** [(ngModel)]="hero.name"**>**
The “interpolation” displays the component’s hero.name property value within thetags
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:

A Basic Angular 2 Application

- An application is consists of a set of components, and some services, each component is comprised of a template, Classes and metadata.



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.

An Angular 2 Application is nothing more than a tree of Components.

At the root of that tree, the top level Component is the application itself. And that's what the browser will render when "booting" (a.k.a bootstrapping) the app.

Angular 2 does not have a bootstrap directive (ng-app). We always launch the app in code by explicitly calling a bootstrap function and passing it the name of the application's module.

One of the great things about Components is that they're composable. This means that we can build up larger Components from smaller ones. The Application is simply a Component that renders other Components.

Because Components are structured in a parent/child tree, when each Component renders, it recursively renders its children Components.

Angular 2 standard module bootstrap is loaded from the angular2/platform/browser module. Using this module we have access to the bootstrap method, which we use right after the import statements to start our Angular 2 application by loading the App component via this method

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:

A Basic Angular 2 Application-Module

- Some important properties are
  - **declarations** - the view classes that belong to this module. Angular has three kinds of view classes: components, directives, and pipes.
  - **exports** - the subset of declarations that should be visible and usable in the component templates of other modules.
  - **imports** - other modules whose exported classes are needed by component templates declared in this module.
  - **providers** - creators of services that this module contributes to the global collection of services; they become accessible in all parts of the app.
  - **bootstrap** - the main application view, called the root component, that hosts all other app views. Only the root module should set this bootstrap property.

Instructor Notes:

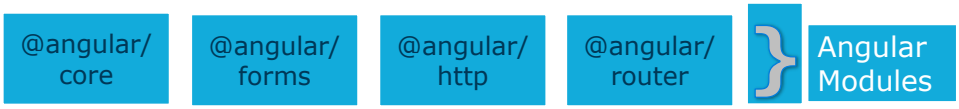
Add instructor notes here.

Importing Modules

- Module loader finds an external function or class using the *import* statement.
- *imports* statement allows us to use exported members from external modules. External modules can be a third party library or our own modules or from angular itself.
- If multiple members from the same module is needed, It can be listed in the import list separated by commas.

```
import { NgModule } from '@angular/core'
```

- Angular is a collection of library modules, each library is itself a module made up of several related feature modules.



Some modules are libraries of other modules. Angular itself ships as a collection of library modules.

angular2/core is the primary Angular library from which we get most of what we need.

For example, to import the Angular Component function from angular2/core

import {Component} from 'angular2/core';

import statement is part of ES 2015 and implemented in typescript . It is conceptually similar to the import statement in java or using statement in C#

import statement tells angular where to find the members that component needs from any external modules.

import statement requires the import keyword followed by the member name and module path, both are case sensitive.

The path to the module file must be enclosed in quotes, no need to specify the file extension.

Instructor Notes:

Add instructor notes here.

A Basic Angular 2 Application-Root Module

- Every application has at least one Angular module, the root module that you bootstrap to launch the application.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
imports: [ BrowserModule ],
declarations: [ AppComponent ],
bootstrap: [ AppComponent ]
})

export class AppModule { }
```

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:

Add instructor notes here.

Demo

- Module

Add the notes here.

Instructor Notes:

Add instructor notes here.

Summary

- Angular2 is a framework for building client applications in HTML.
- The framework consists of several libraries, some of them core and some optional.
- Angular apps are modular and Angular has its own modularity system called *Angular modules* or *NgModules.*

Add the notes here.