**Angular 2**



Note: Examples are written in Typescript (A superset of JavaScript)

<https://angular.io/docs/ts/latest/guide/learning-angular.html>

Angular 2 is made up of components.

**Component**: A combination of an HTML template and a component class that controls a portion of the screen.

|  |
| --- |
| import { Component } from '@angular/core';  @Component({  selector: 'my-app',  template: `<h1>Hello {{name}}</h1>`  })  export class AppComponent { name = 'Angular'; } |

@Component is a **decorator** function that takes a *metadata* object.

The selector property tells Angular to display the component inside the my-app tag in the index.html file.

The template property defines the message inside the header. {{name}} is an Angular interpolation binding expression. The value for name is replaced at runtime.

Angular Module:

A class that describes how the application parts fit together. Every application has at least one module, the Root Module that you bootstrap to launch the application.

@NgModule is another **decorator**. It identifies AppModule as an Angular module class. It takes a metadata object that tells Angular how to compile and launch the application.

|  |
| --- |
| import { NgModule } from '@angular/core';  import { BrowserModule } from '@angular/platform-browser';  // These import statements are JavaScript import statements give access to symbols in the specified file  @NgModule({  imports: [ BrowserModule ],  providers: [ Logger ],  declarations: [ AppComponent ],  exports: [ AppComponent ],  bootstrap: [ AppComponent ]  })  export class AppModule { } |

Imports Array:

Imports external Modules (Libraries). They are used to consolidate features that belong together into discrete units. Features like HTTP services are in the HttpModule, the router is in the RouterModule, etc.

BrowserModule is required in order for the application to execute in a browser. It is necessary to include it in the root AppModule’s imports array.

The imports array is exclusively in the @NgModule metadata object.

Providers Array:

Creators of services that this module contributes to the global collection of services. They become accessible in all parts of the app.

Exports Array:

The subset of declarations that should be visible and usable in the component templates of other modules. There is no point of the root Module exporting anything because other components don’t need it.

Declarations Array:

Tells Angular which components belong to the AppModule.

Every component must be declared in an NgModule class.

Other things that need to be declared in the declerations array are **Directives** and **Pipes**.

Bootstrap Array:

When the application is launched, the root AppMoudle is **bootstrapped**. The process creates the component listed in the bootstrap array and inserts each one into the browser DOM.

Each bootsrapped component is the base of its own tree of components that are all created. The root component can be called anything but most developers stick with AppComponent.

**Bootstrap Process:**

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.

The main.ts file describes the bootstrapping process. Example of bootstrapping a Just-In-Time (JIT) compiled browser application is:

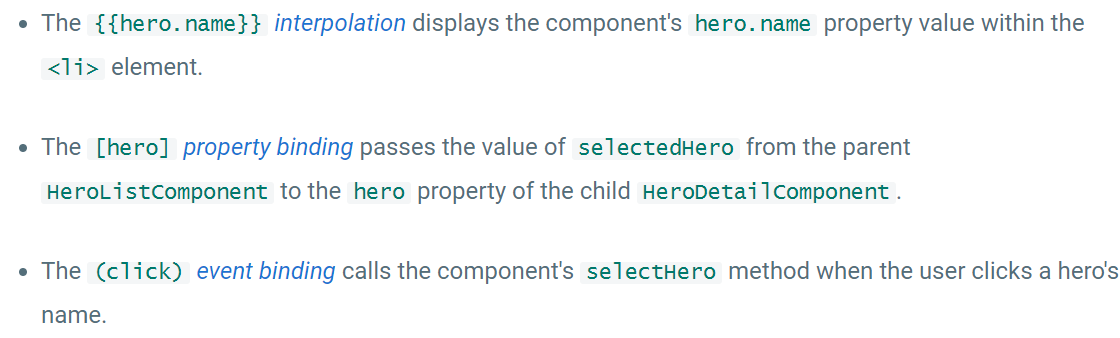
|  |
| --- |
| import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';  import { AppModule } from './app/app.module';  platformBrowserDynamic().bootstrapModule(AppModule); |

This code creates a browser platform for dynamic (JIT) compilation, and bootstraps the AppModule.

Can also statically bootstrap with Ahead-of-Time (AOT) compiler.

|  |
| --- |
| // The browser platform without a compiler  import { platformBrowser } from '@angular/platform-browser';  // The app module factory produced by the static offline compiler  import { AppModuleNgFactory } from './app/app.module.ngfactory';  // Launch with the app module factory.  platformBrowser().bootstrapModuleFactory(AppModuleNgFactory); |

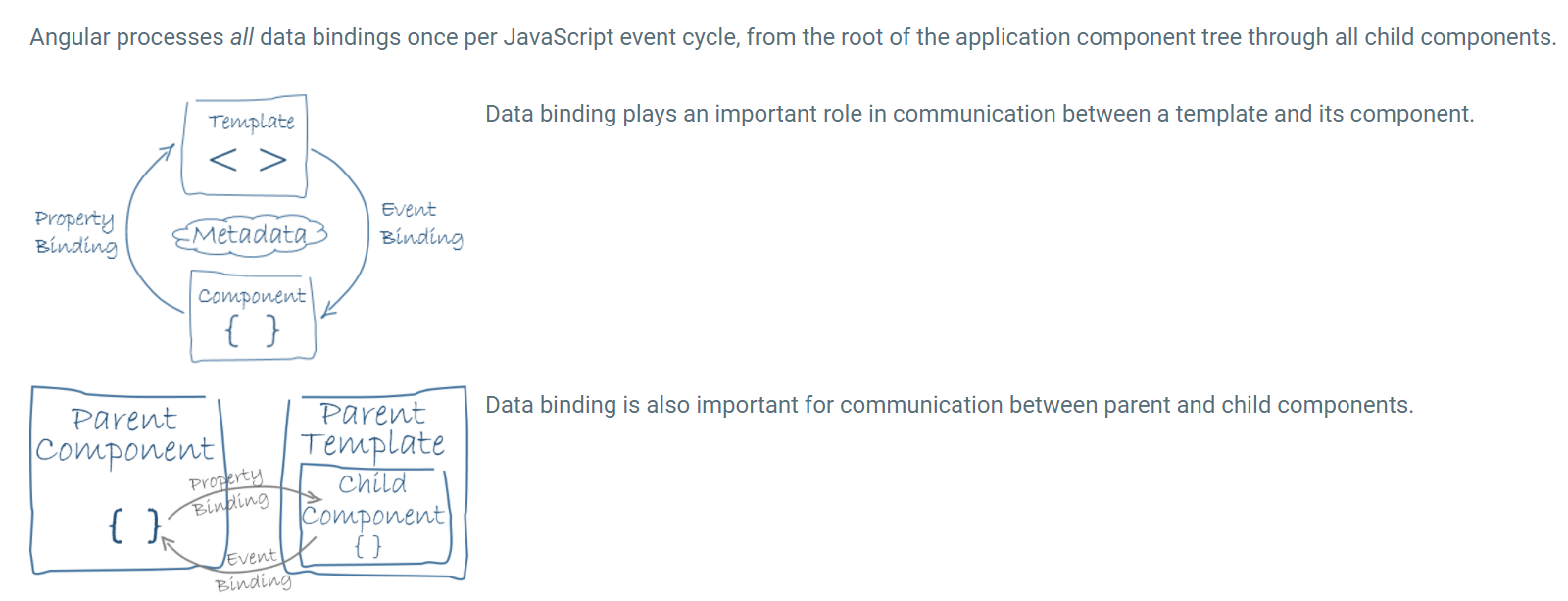
**Data Binding**



2 way binding combines property binding and event binding.

Eg.

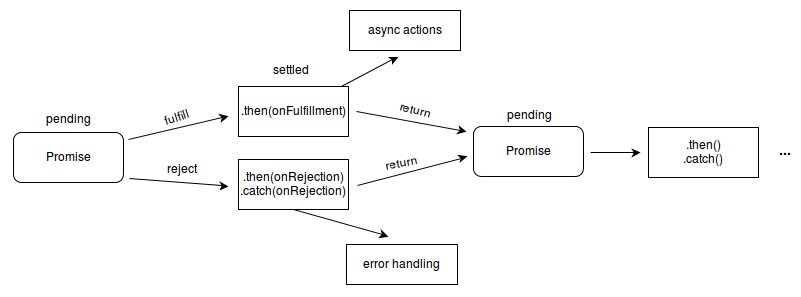
|  |
| --- |
| <input [(ngModel)]="hero.name"> |



Services:

Data is acquired from Angular Services, this way it can be made its own separate component and it can be used in many different locations.

Promise:



LifeCycle of a Component

