Angular: Getting Started

Steps to Create Angular application

Step 1 - Install the Angular CLI

Open a terminal and enter the following command to install the Angular CLI globally:

```
npm install -g @angular/cli
```

If you have the Angular CLI already installed globally then run below commands to update it to the latest version:

```
npm uninstall -g @angular/cli

npm cache clean --force

npm install -g @angular/cli@latest
```

Steps to Create Angular application continue...

Step 2 - Create a workspace and initial application

To create a new workspace and initial app project:

```
ng new firstApp
```

Step 3 - Create a workspace and initial application

```
cd firstApp
```

```
ng serve --open
```

The --open (or just -o) option automatically opens your browser to http://localhost:4200/

Important Files to understand

- **app/main.ts** - is the glue that combines the component and page together
  - The main entry point for your app. Compiles the application with the JIT compiler and bootstraps the application's root module (AppModule) to run in the browser.
- **app/app.module.ts** - the entry Angular Module to be bootstrapped
  - Defines the root module, named AppModule, that tells Angular how to assemble the application. Initially declares only the AppComponent. As you add more components to the app, they must be declared here.
- **app/app.component.ts** - this is where we define our root component
  - Defines the logic for the app's root component, named AppComponent. The view associated with this root component becomes the root of the view hierarchy as you add components and services to your app.
- **index.html** - this is the page the component will be rendered in
  - The main HTML page that is served when someone visits your site. The CLI automatically adds all JavaScript and CSS files when building your app, so you typically don't need to add any <script> or<link> tags here manually.

Building Blocks of Angular 7

- Modules
- Components
- Data Binding
- Templates
- Directives
- Services
- Dependency Injection
- Routing

Bootstrapping AppModule

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

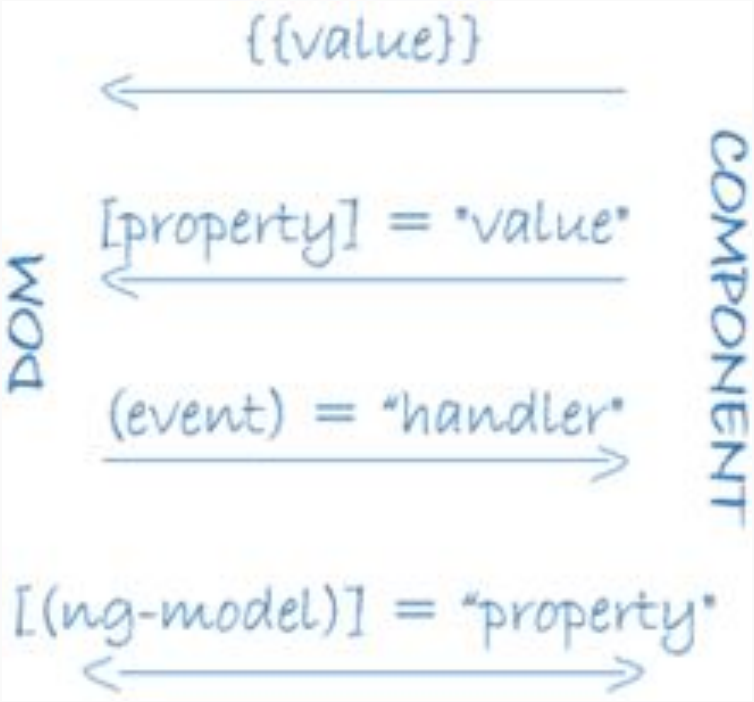
Understanding Modules

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

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Understanding Modules continue...

- **NgModule** is a decorator function that takes a single metadata object whose properties describe the module. The most 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.
- The application is launched by bootstrapping the root module.
- It is mostly done in main.ts and likely the name of root module is given as **AppModule.**

Data Binding




```
<h1>
  Welcome to {{ title }}!
</h1>
```

```
<img [src]="angularLogo"
     (click)="onLogoClick($event)"/>
<br>
<button [disabled]="buttonStatus"
        (click)="updateBtnStatus($event)">
            My Button
</button>
```

```
<input [(ngModel)]="productObj.prodName" />
<br>
<div>Product name: {{productObj.prodName}}</div>
```

Thank You!