# Introduction

## What is ANGULAR

AngularJS is a client side JavaScript MVC framework to develop a dynamic web application. AngularJS was originally started as a project in Google but now, it is open source framework.

## What is NODE JS

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world

## What is NPM

npm is the node package manager for JavaScript and the world’s largest software registry. Discover packages of reusable code — and assemble them in powerful new ways.

## What is ANGULAR/CLI

The Angular CLI is a tool to initialize, develop, scaffold and maintain Angular applications

## What is BOOTSTRAP

Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web

## TypeScript and JavaScript

TypeScript is a superset of JavaScript which primarily provides optional static typing, classes and interfaces. One of the big benefits is to enable IDEs to provide a richer environment for spotting common errors as you type the code.

For a large JavaScript project, adopting TypeScript might result in more robust software, while still being deployable where a regular JavaScript application would run.

# Prepare Environment

1. install nodejs at the latest version on <https://nodejs.org/en/>(if you already have it the good way to uninstall first)
2. open the command windows or power shell
3. update the npm

* npm install -g npm

1. install type script
   * npm install -g typescript
2. uninstall angular/cli (in case of you already have it) and then clear cache

* npm uninstall -g angular-cli @angular/cli
* npm cache clean -f

1. install angular/cli

* npm install -g @angular/cli

# Start Project

1. create new project

* ng new [app name]

1. install component (bootstrap)

* npm install --save [bootstrap]

1. set bootstrap css to app

* goto .angular-cli.json > at "styles":[] tag add "../node\_modules/bootstrap/dist/css/bootstrap.min.css",

1. but you also can install another library from npm

\* if you download (clone) project and want to initialize the project have to run

* npm install

# Angular CLI Command

1. run application

* ng serve [--open]

1. generating command

|  |  |
| --- | --- |
| **Scaffold** | **Usage** |
| Component | ng g component my-new-component |
| Directive | ng g directive my-new-directive |
| Pipe | ng g pipe my-new-pipe |
| Service | ng g service my-new-service |
| Class | ng g class my-new-class |
| Guard | ng g guard my-new-guard |
| Interface | ng g interface my-new-interface |
| Enum | ng g enum my-new-enum |
| Module | ng g module my-module |

1. build application

* ng build [--prod]

<https://cli.angular.io/>

# VS Code Editor Shotcut

1. create div with class container

* .container [tab]

1. create div with class row

* .row [tab]

1. create div with class col

* .col-xs-12 [tab]
* .col-md-4 [tab]

1. create button with class btn

* button.btn.btn-primary [tab]

1. create input text with class

* input.form-control [tab]

# START Angular

## Typescript Syntax

### Var, Let, const

The scope of **var** is global and in the function

The scope of **let** is inside { }

The scope of **const** is inside { } and also the value cannot be change

### Fat Arrow Functions

setTimeout(**function**() {

console.log("setTimeout called!");

}, 1000);

Can now be written as:

setTimeout(() => {

console.log("setTimeout called!")

}, 1000);

Arguments

**let** add = **function**(a,b) {

**return** a + b;

};

Can now be written as:

Copy**let** add = (a,b) => a + b;

### Object Destructuring

**const** obj = {first: 'Asim', last: 'Hussain', age: 39 };

We want to extract the first and last properties into local variables, we would have to write something like this:

**const** f = obj.first;

**const** l = obj.last;

console.log(f); *// Asim*

console.log(l); *// Hussain*

With destructing we can do so in one line, like so:

Copy**const** {first: f, last: l} = obj;

console.log(f); *// Asim*

console.log(l); *// Hussain*

### array destructuring

**const** arr = ['a', 'b'];

**const** [x, y] = arr;

console.log(x); *// a*

console.log(y); *// b*

### For & ForEach

**let** array = [1,2,3];

**for** (**let** i = 0; i < array.length; i++) {

console.log(array[i]);

}

we can also use the forEach method on the Array class, like so:

Copy**let** array = [1,2,3];

array.forEach(**function** (value) {

console.log(value);

});

### Promise API

It serves the same function as callbacks but has a nicer syntax and makes it easier to handle errors.

**let** error = **true**;

**function** doAsyncTask() {

**return** **new** Promise((resolve, reject) => {

setTimeout(() => {

**if** (error) {

reject('error'); *// pass values*

} **else** {

resolve('done'); *// pass values*

}

}, 1000);

});

}

doAsyncTask().then(

(val) => console.log(val),

(err) => console.error(err)

);

### Class

**class** Person {

**constructor**(public firstName: string, public lastName: string) { }

name() {

**return** `${**this**.firstName} ${**this**.lastName}`;

}

whoAreYou() {

**return** `Hi i'm ${**this**.name()}`;

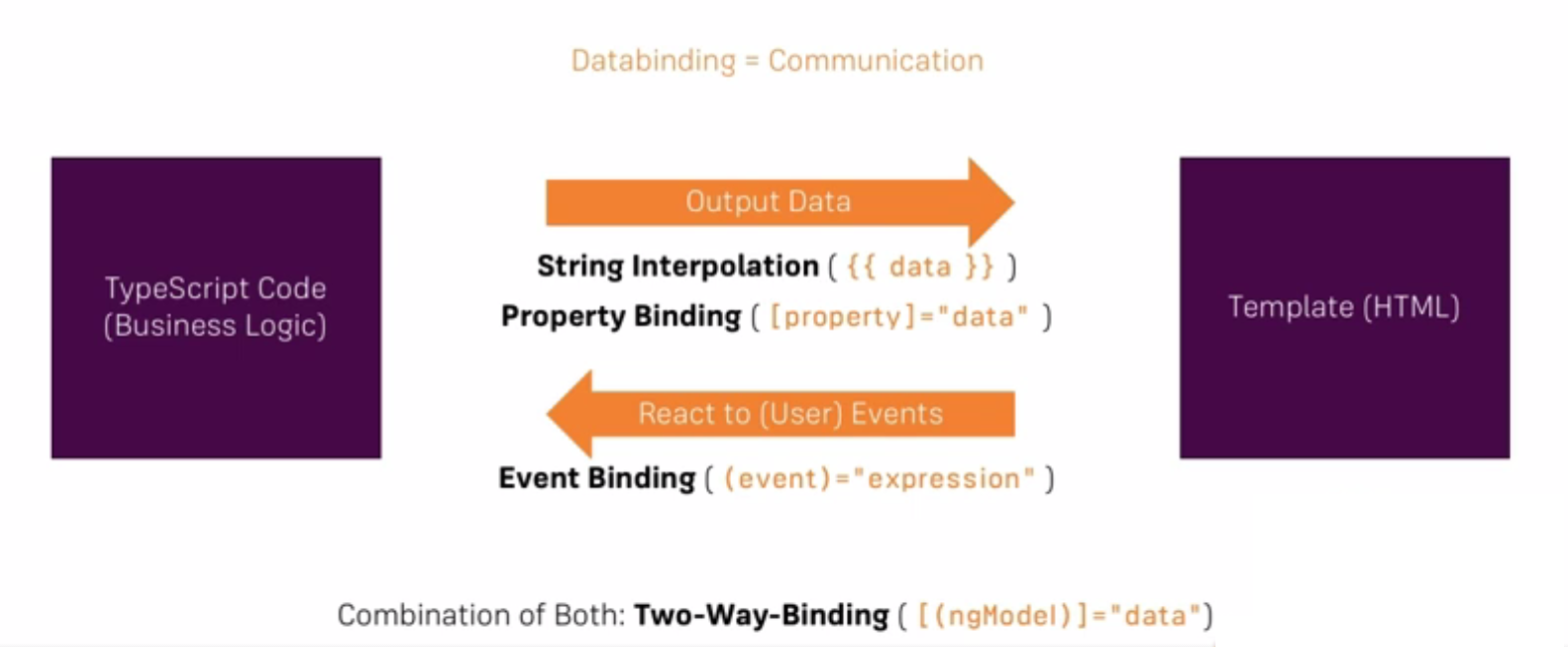
}

}

## Component

In Angular component are important!!

## Databinding



String interpolation

<**p**>{{information}}</**p**>

Property binding

<**p** [hide]=” hideInfo”>{{ information }}</**p**>

Event binding

<**p** [hide]=” hideInfo” (click)=”onClickP()”>{{information }}</**p**>

Two way binding

<**p** [hide]=”hideInfo” (click)=”onClickP()”>{{information }}</**p**>

<input type=”text” [(ngModel)]=” information”>

Template Local Variables

<**p** \*ngIf=” hideInfo; else showinfo” (click)=”onClickP()”>{{information }}</**p**>

<ng-template #showinfo> information is secret</ng-template>

<input type=”text” [(ngModel)]=” information”>

## Built-in Directives

There are three kinds of directives in Angular:

**Components**—directives with a template.

**Structural directives**—change the DOM layout by adding and removing DOM elements.

**Attribute directives**—change the appearance or behavior of an element, component, or another directive.

*Components* are the most common of the three directives. You saw a component for the first time in the QuickStart guide.

*Structural Directives* change the structure of the view. Two examples are NgFor and NgIf. Learn about them in the Structural Directives guide.

*Attribute directives* are used as attributes of elements. The built-in NgStyle directive in the Template Syntax guide, for example, can change several element styles at the same time.

### ngfor

<**div** class="card card-block"

\*ngFor="let joke of jokes; let idx = index ">

<**h4** class="card-title">{{joke.setup}}</**h4**>

<**p** class="card-text">{{joke.punchline}}</**p**>

</**div**>

### ngif

<**ul** \*ngFor="let person of people" \*ngIf="person.age < 30">

<**li**>{{ person.name }}</**li**>

</**ul**>

### ngswitch

<ul \*ngFor="let person of people"

[ngSwitch]="person.country">

<li \*ngSwitchCase="'UK'"

**class**="text-success">{{ person.name }} ({{ person.country }})

</li>

<li \*ngSwitchCase="'USA'"

**class**="text-primary">{{ person.name }} ({{ person.country }})

</li>

<li \*ngSwitchCase="'HK'"

**class**="text-danger">{{ person.name }} ({{ person.country }})

</li>

<li \*ngSwitchDefault

**class**="text-warning">{{ person.name }} ({{ person.country }})

</li>

</ul>

### ngstyle

<**div** [ngStyle]="{'background-color':'green'}"></<**div**>

ngStyle becomes much more useful when the value is dynamic

<**div** [ngStyle]="{'background-color':person.country === 'UK' ? 'green' : 'red' }"></<**div**>

### NgNonBindable

<**div**>

To render the name variable we use this syntax <**pre** ngNonBindable>{{ name }}</**pre**>

</**div**>

## Inputs

**import** { Input } from '@angular/core';

….

**class** JokeComponent {

**@Input**() joke: Joke;

}

## Outputs

**import** {Output, EventEmitter} from '@angular/core';

….

**class** JokeFormComponent {

**@Output**() jokeCreated = **new** EventEmitter<string>();

}

## Custom Directive

**import** { Directive } from '@angular/core';

….

**@Directive**({

selector:"[ccCardHover]"

})

**class** CardHoverDirective { }

to use that

<**div** class="card card-block" ccCardHover>...</**div**>

Or we can create directive like class

**import** { Directive } from '@angular/core';

…

**@Directive**({

selector:".ccCardHover"

})

**class** CardHoverDirective { }

to use that

<**div** class="card card-block ccCardHover">...</**div**>

### Create structure directive

We will try to create *unless* directive, this directive will do something opposite with ngIf

**Import { Directive, input, TemplateRef, ViewContainerRef} from ‘@angular/cor’;**

**@Directive**({

selector:"[appUnless] "

})

**export class** UnlessDirective {

@Input() set unless(condition: Boolean){

If(!condition) {

This.vcRef.createEmbeddedView(this.templateRef);

}else {

This.vcRef.clear();

}

}

}

## Service & Dependency Injection

*// added* if need to inject some service in this service

**import** { Injectable } from '@angular/core';

@Injectable() *// added*

**export** **class** FormService {

*// ...class code*

}

to use that

add to providers of AppModule and inject to constructor of component that need service

**constructor**(**private** formService: FormService) {

formService.

}

## Routing

**import** {Routes, RouterModule} from "@angular/router";

…

**const** routes: Routes = [

{ path: '', component: HomeComponent },

{ path: 'search', component: SearchComponent }

];

**@NgModule**({

imports: […,

RouterModule.forRoot(routes, {useHash: true})

],…

})

**class** AppModule { }

<a routerLink=”/page2”></a> 🡪 this will show the path from the beginning

<a routerLink=”page2”></a> 🡪 this will show the path from current path

<a routerLink=”../page2”></a> 🡪 this will up to 1 level and then show the path

----

Programming to

*….*

**import** { Router } from '@angular/router';

@Component({

selector: ‘app-home’,

….

})

**export** **class** SomeCompoent {

**onSomeAction**(){

// complex calculation

this.router.navigate([‘/somePage’]);

}

}

<https://codecraft.tv/courses/angular/quickstart/overview/>