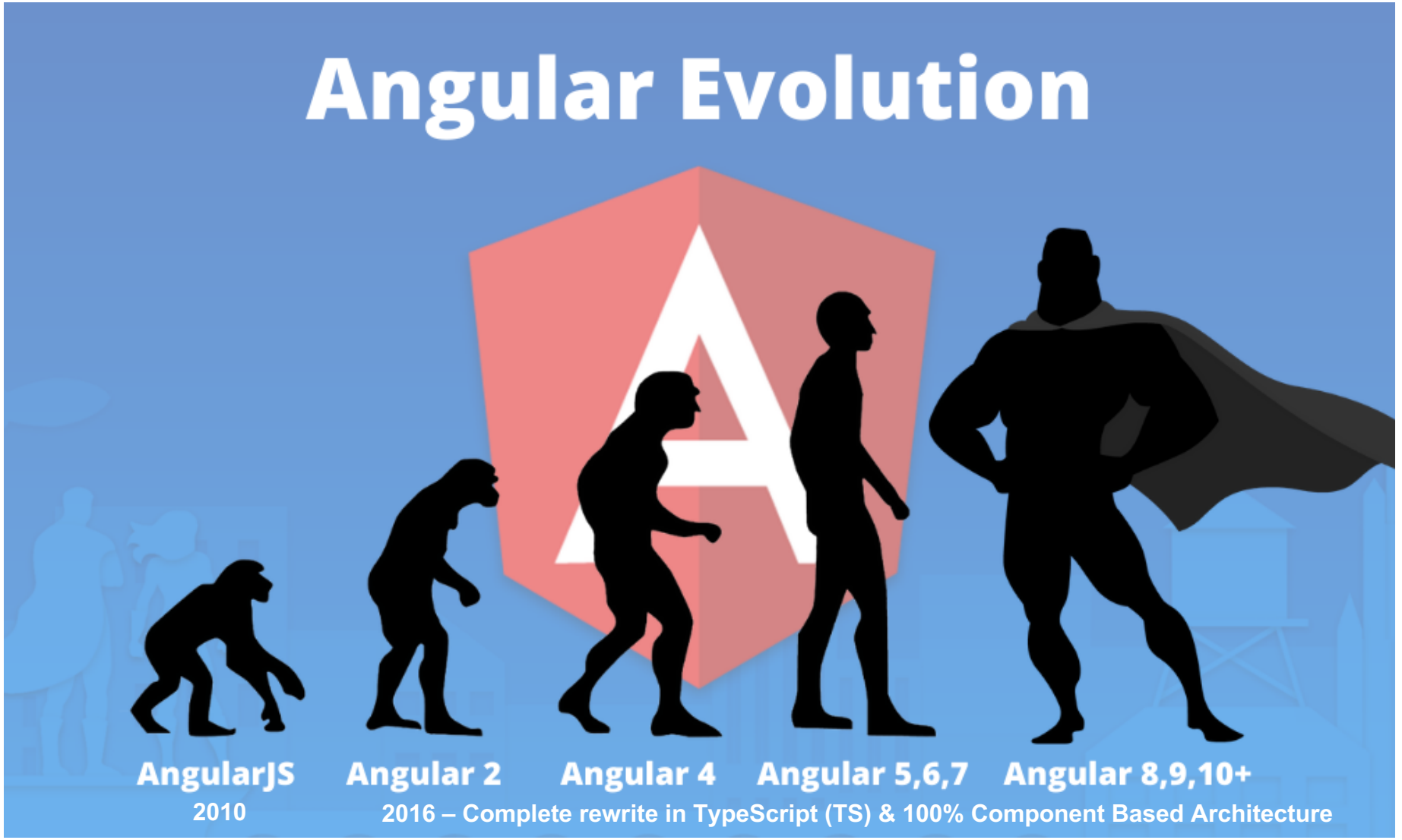
BridgeLabz

Employability Delivered

AngularJs

BridgeLabz
Employability Delivered

Angular Getting Started



BridgeLabz
Employability Delivered

BridgeLabz
Employability Delivered

Create a Helloworld Angular App to display “Hello from BridgeLabz”

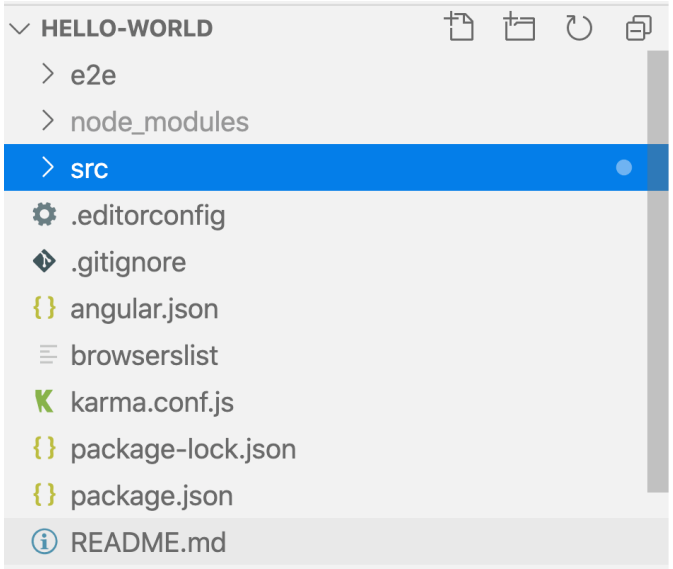
- To learn any new Language or any New Framework start with Helloworld
- This will ensure all the necessary components are involved
- A quick way to jumpstart ones learning

Angular Setup

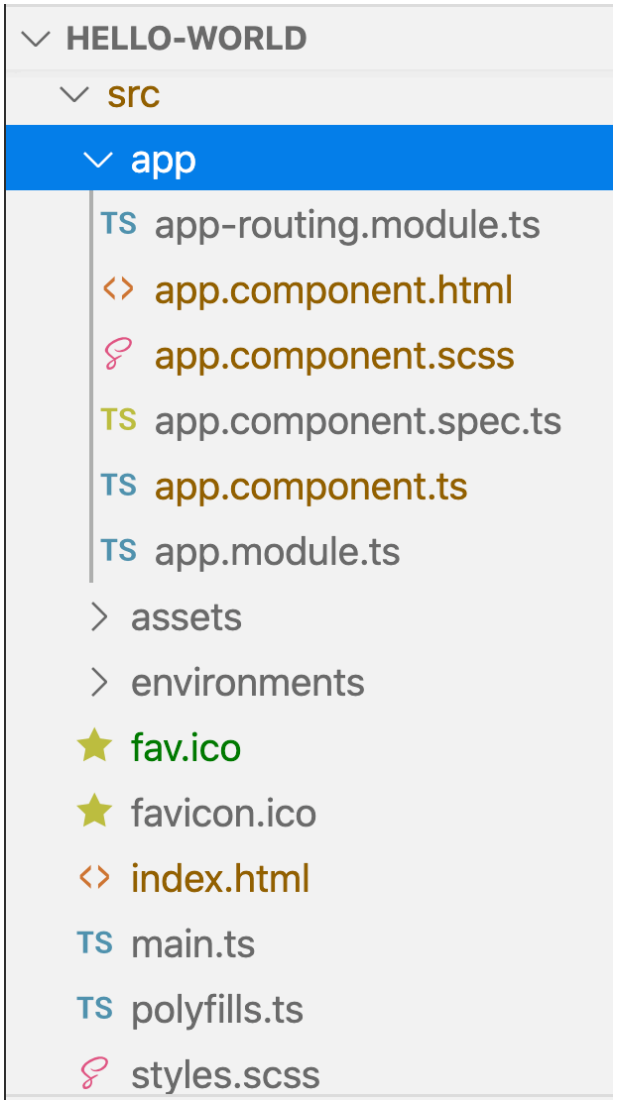
BridgeLabz
Employability Delivered

- Angular requirements: Install NodeJS and npm. Type npm –v to check the installation of Node Package Manager (npm)
- IDE – Visual Studio Code.
- Use NPM to install the Angular CLI in the Terminal. This is because Angular App are created using Angular CLI (Command Line Interface) – *npm install -g @angular/cli@9* to install version 9.
- Run ng – v on Terminal to check the installation.
- Create projects Directory and type *ng new Helloworld --style=scss --routing* Here
  - ng – This is how we call the Angular CLI.
  - new – This is the command issued to CLI to create new Project. Plenty of such commands can be issued.
  - Helloworld – Simply the name of the Project
  - Optional Flags – Next two are Optional Flags to generate project with SCSS (Superset of CSS) and routing.
- Launch and Run in VSCode –
  - To open in VSCode – *cd Helloworld* and *code .*
  - To run in Browser – ng serve -o The -o flag tells the CLI to launch your browser with the project.

Project Structure



node_modules consists of all the Libraries needed for Angular generated from package.json typically installed using `npm i` Typically this is auto installed if new is created using ng new. Also this directory is .gitignore because of File Size close to **400 - 500MB**



- **src –** All app development happens in this folder.
- **index.html and style.scss –** Plain HTML5 file. Starting point where global css is applied. Mainly has app-root
- **app-root –** placeholder to load App Components enabling **SPA – Single Page Application**
- **assets –** place for all assets.
- **app –** Most of time spent working on this folder. This mainly contains App Modules, App Routings and App Components.
- **App Component** is typically has the html and js code corresponding to every page.

Decode app.module.ts

src > app > TS app.module.ts > ...

```ts
// app.module.ts - Indicating this is a TypeScript file.
//                 Essentially Provides Strong Type Checking.

// Importing core modules of Angular
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';

// Whenever CLI is used to generate new components and services, it
// will automatically update this file to import and add them here.
import { AppComponent } from './app.component';

// Further using @NgModule decorator we provide additional metadata to
// specify the Components, the Services, the Imports, etc.. Further
// Decorators are used while processing, instantiation and at runtime
@NgModule({

  // Components are added here. Essentially the classes that has views,
  // they are Components, Directives and Pipes.
  declarations: [
    AppComponent
  ],

  // Various imports needed for Application can be added here.
  imports: [
    BrowserModule,
    AppRoutingModule
  ],

  // Typically Services like http-services are added to providers
  providers: [],

  // The root component which is the main view of the application. Only the
  // root module has this property and it indicates the component that's
  // gonna be bootstrapped.
  bootstrap: [AppComponent]
})
export class AppModule { }
```

BridgeLabz
Employability Delivered

Angular App Components

- Components are the most basic **building block** of an UI in Angular applications.
- A component controls one or more sections on the screen (what we call **views**).
- By Default on creating of new Project AppComponent is automatically created as the bootstrapped component.
- A component is self contained and represents a reusable piece of UI that is usually constituted by three important things:
  - A piece of **html** code that is known as the **view**
  - The **SCSS/CSS** the associated **CSS** rulesets for component
  - The **.spec.ts** file is for **testing** purposes.
  - The **.ts** file is the actual component file having **Application Logic**..

```
> src
  > app
    app.component.html
    app.component.scss (or .css)
    app.component.spec.ts
    app.component.ts
```

A **Component** is a class that **encapsulates** all available **data** and **interactions** to its corresponding **view** through an API of properties and methods architected by Angular.

Angular Components

BridgeLabz
Employability Delivered

src > app > TS app.component.ts > ...

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'Helloworld';
}
```

src > app > <> app.component.html > ...

```html
<div>
  <h1>Hello from BridgeLabz</h1>
</div>

<router-outlet></router-outlet>
```

src > app > app.component.scss > ...

```scss
.div {
    text-align: center;
    font-size: x-large;
}
```

- Start with the imports. Other Components can also be imported here.
- The **@Component decorator** is an object with associated property / value pairs that defines important stuff associated with this component.
- **selector: 'app-root'** – This is a unique identifier of this component that can be used in other areas of the app.
- **templateUrl & styleUrls** – identifies where this components html and CSS style is located.
- Using *ng generate component / ng g c* New Components can be created.

BridgeLabz
Employability Delivered

Angular Data Binding

BridgeLabz
Employability Delivered

To display “Hello from BridgeLabz” using One Way Data Bindings

- Modify App Component to demonstrate
- Interpolation using Template Expression

Interpolation Data Binding

BridgeLabz
Employability Delivered

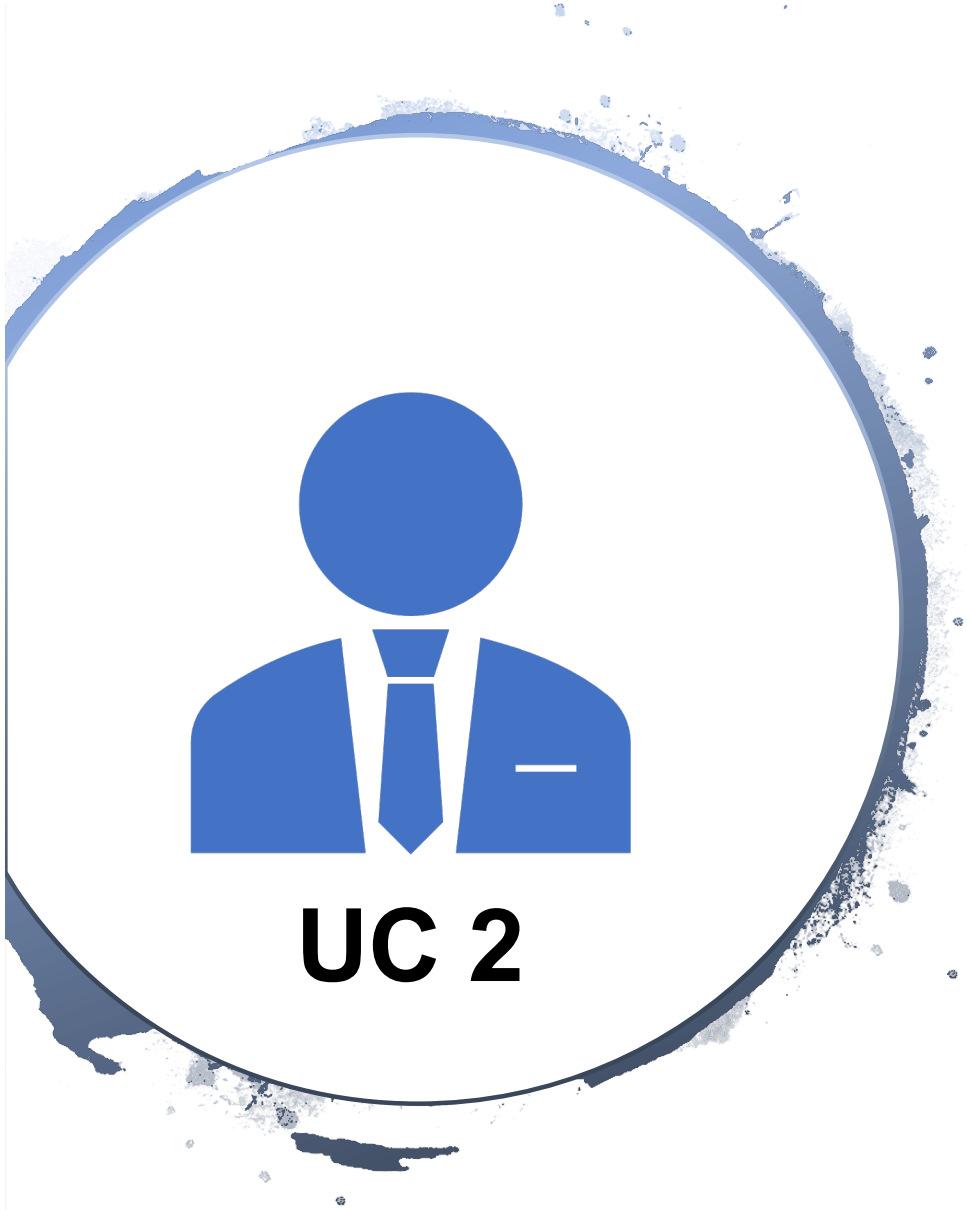
src > app > TS app.component.ts > ...

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'Helloworld';

  ngOnInit(): void {
    this.title = "Hello from BridgeLabz.";
  }
}
```

src > app > <> app.component.html > ...

```html
<div>
  <h1>{{this.title}}</h1>
</div>

<router-outlet></router-outlet>
```

- Finally the **logic** section of the Component file is modified to initialize **title** when **ngOnInit lifecycle event** is fired.
- There are around 8 Lifecycle Events fired right from when Component is initialized to any change till the Component is destroyed.
- Here **Interpolation** which is a **one-way databinding technique** is used to output the data from a TypeScript code to HTML template (view).
- It uses the **template expression** in double curly braces {{this.title}} to display the data i.e. the title from the component to the view.



BridgeLabz
Employability Delivered

To display BridgeLabz Logo using Property Binding Technique

- Modify App Component to display BridgeLabz Logo using Property Binding

Property Data Binding

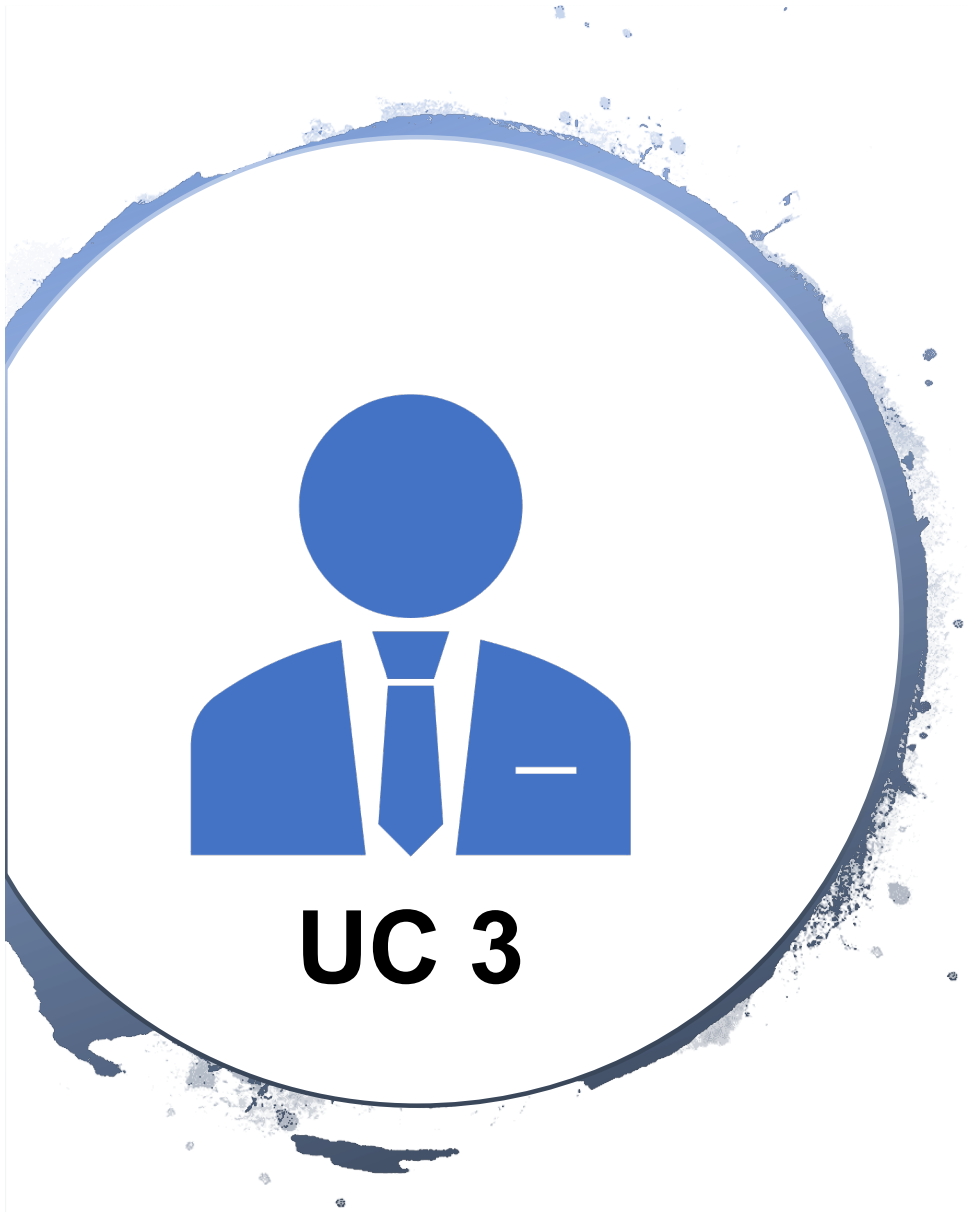
src > app > TS app.component.ts > ...

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'Helloworld';
  imgUrl="../assets/BL_logo_square_jpg.jpg";

  ngOnInit(): void {
    this.title = "Hello from BridgeLabz.";
  }
}
```

- Here imgUrl is linked to img element src property using Property Binding Technique
- Property Binding is also a one-way data binding technique.
- In property binding, we bind a property of a DOM element to a field which is a defined field in our component
- For e.g.

```html
<div>
  <h1>{{this.title}}</h1>
  <img [src]="imgUrl"
       alt="The BridgeLabz logo: a Bridge to Employment through a Lab Works."
  />
</div>

<router-outlet></router-outlet>
```

```css
img {
    display: block;
    margin: 0 auto;
    width: 25%;
}
```



BridgeLabz
Employability Delivered

Ability To launch BridgeLabz Site in a new Tab on clicking BridgeLabz Logo

- Use Event Binding Technique to bind Click Event to Launch BridgeLabz URL in a new Tab

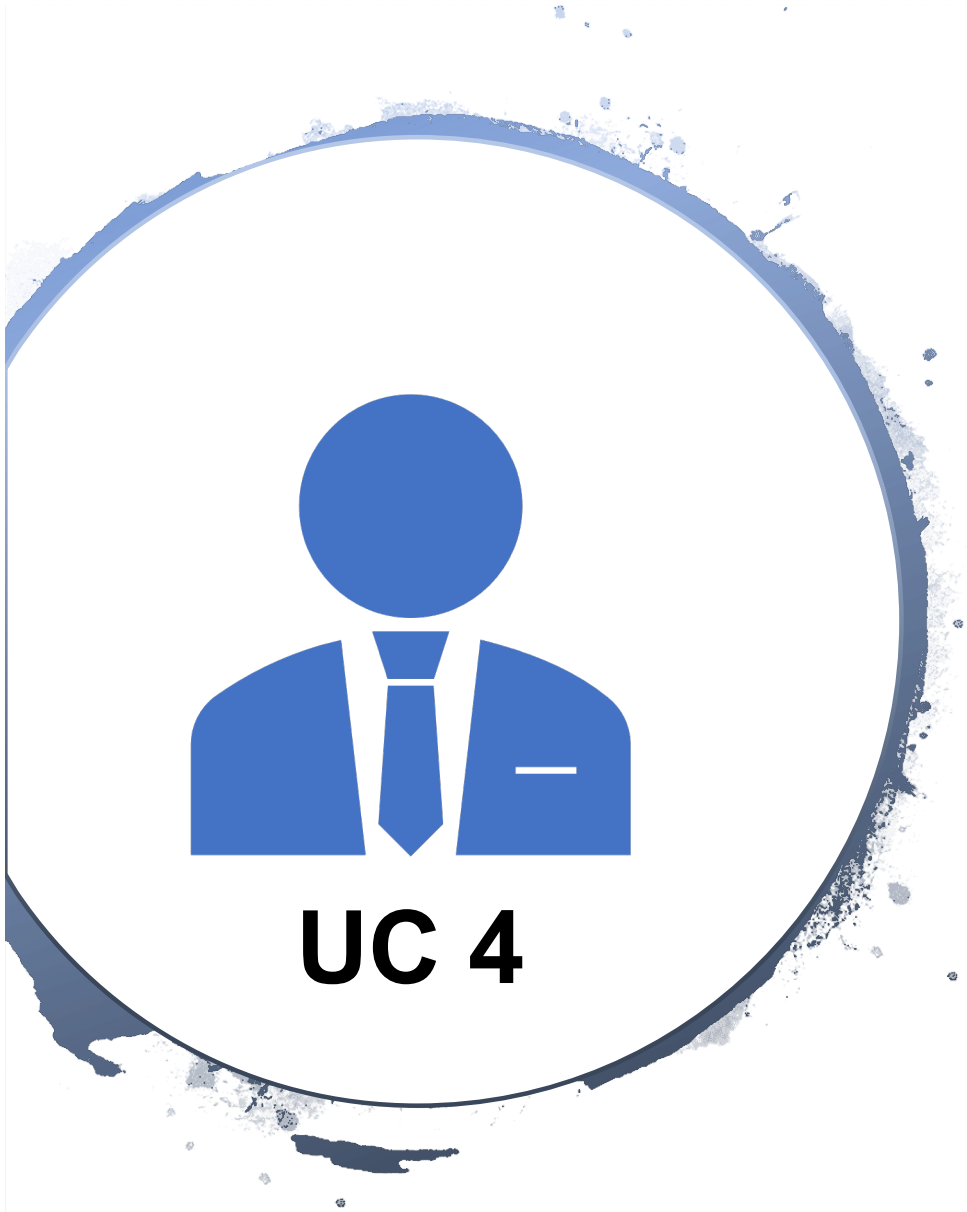
Event Binding


```ts
export class AppComponent {
  title = 'Helloworld';
  imgUrl="../assets/BL_logo_square_jpg.jpg";
  url = "https://www.bridgelabz.com";

  ngOnInit(): void {
    this.title = "Hello from BridgeLabz.";
  }

  onClick($event){
    console.log("Save button is clicked!", $event);
    window.open(this.url, "_blank");
  }
}
```

- In Angular, event binding is used to handle the events raised from the DOM like button click, mouse move etc.
- When the DOM event happens (eg. click, change, keyup), it calls the specified method in the component.

src > app > app.component.html > ...

```html
<div>
  <h1>{{this.title}}</h1>
  <img [src]="imgUrl" (click)="onClick($event)"
       alt="The BridgeLabz logo: a Bridge to Employment through a Lab Works."
  />
</div>

<router-outlet></router-outlet>
```



BridgeLabz
Employability Delivered

To display Hello from BridgeLabz with a user inputted name

- Make sure to import Form Module needed for Application
- Bind the Input Element to the userName of the App Component.
- Make sure it is two way binding so that the change in the value can be reflected in the Hello Message

Step 1: Import Form Module

src > app > TS app.module.ts > ...

```
1   import { BrowserModule } from '@angular/platform-browser';
2   import { NgModule } from '@angular/core';
3   import { AppRoutingModule } from './app-routing.module';
4   import { FormsModule } from '@angular/forms';
5
6   import { AppComponent } from './app.component';
7
8   @NgModule({
9     declarations: [
10      AppComponent
11    ],
12    imports: [
13      BrowserModule,
14      FormsModule,
15      AppRoutingModule
16    ],
17    providers: [],
18    bootstrap: [AppComponent]
19  })
20  export class AppModule { }
```

BridgeLabz
Employability Delivered

Importing Form Module allows us to use Angular Form Elements and bind them to the Model of the App Component

Step2: Specify userName as String Type in the App Component Model

```
export class AppComponent {
  title = 'Helloworld';
  imgUrl="../assets/BL_logo_square_jpg.jpg";
  url = "https://www.bridgelabz.com";
  userName: string = "";

  ngOnInit(): void {
    this.title = "Hello from BridgeLabz.";
  }

  onClick($event){
    console.log("Save button is clicked!", $event);
    window.open(this.url, "_blank");
  }
}
```

Note: Change in the userName reflected in the header Tag

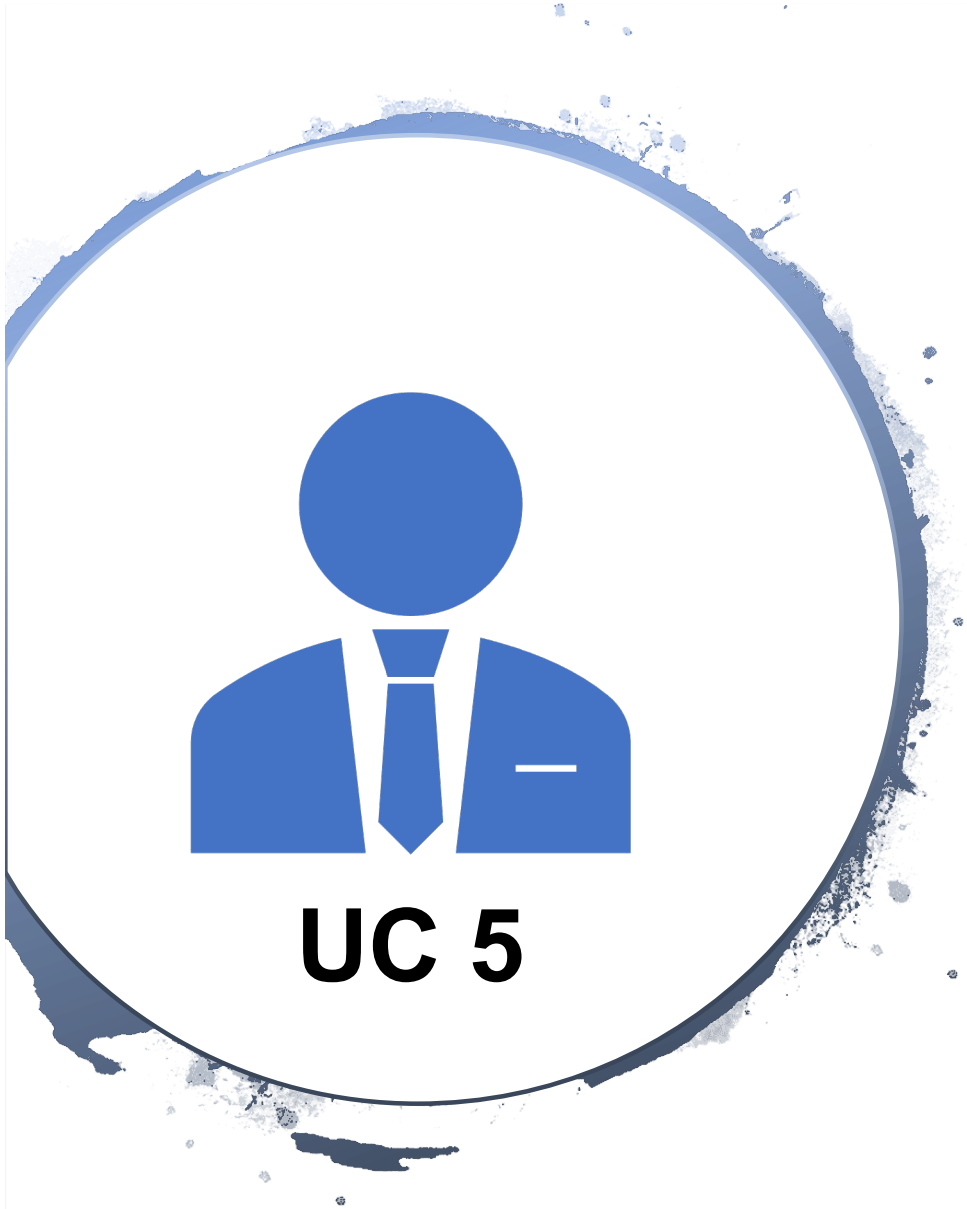
Step3: Bind the input Form Element using [(ngModel)] to userName

src > app > app.component.html > ...

```
1   <div>
2     <h1>Hello {{this.userName}} from BridgeLabz</h1>
3     <img [src]="imgUrl" (click)="onClick($event)"
4          alt="The BridgeLabz logo: a Bridge to Employment through a Lab Works."
5     />
6   </div>
7   <div>
8     <input [(ngModel)]="userName" /> <br/><br/>
9   </div>
10  <router-outlet></router-outlet>
```

Two Way Binding

- We have seen that in **one-way data binding** any change in the html template (view) will not be reflected in the App Component.
- To resolve this problem, Angular provides **two-way data binding**.
- The two-way binding has a feature to **update data** from component to view and vice-versa.
- In two-way databinding, **automatic synchronization** of data happens between the **Model and the View**. Here, change is reflected in both. Whenever you make changes in the Model, it will be reflected in the View and when you make changes in View, it will be reflected in Model.



BridgeLabz
Employability Delivered

To display Hello from BridgeLabz with a user inputted name

- Ensure the userName is the valid input consisting of Initial Caps and min 3 Letter.
- In Case Error display the error

Using Input Event Binding


```
export class AppComponent {
  title = 'Helloworld';
  imgUrl="../assets/BL_logo_square_jpg.jpg";
  url = "https://www.bridgelabz.com";
  userName: string = "";
  nameError: string = "";

  ngOnInit(): void {
    this.title = "Hello from BridgeLabz.";
  }

  onClick($event){
    console.log("Save button is clicked!", $event);
    window.open(this.url, "_blank");
  }

  onInput($event){
    console.log("Change Event Occurred!", $event.data);
    const nameRegex = RegExp('^[A-Z]{1}[a-zA-Z\\s]{2,}$');
    if (nameRegex.test(this.userName)) {
      this.nameError = "";
      return;
    }
    this.nameError = "Name is Incorrect!";
  }
}
```

```
<div>
  <h1>Hello {{this.userName}} from BridgeLabz</h1>
  <img [src]="imgUrl" (click)="onClick($event)"
       alt="The BridgeLabz logo: a Bridge to
            Employment through a Lab Works."
  />
</div>
<div>
  <input (input)="onInput($event)" [(ngModel)]="userName" />
  <span class = error-output>{{this.nameError}}</span>
</div>
<router-outlet></router-outlet>
```

```
.error-output {
    margin-left: 10px;
    font-size: 12px;
    font-style: italic;
    color: red;
}
```

Final Hello world App

BridgeLabz
Employability Delivered

Without Error

Hello Narayan from BridgeLabz

Narayan

Witt Error

Hello Na from BridgeLabz

Na *Name is Incorrect!*

References

BridgeLabz
Employability Delivered

- SASS vs SCSS
- Angular Life Cycle Hooks
- Angular Data Binding Techniques
- Angular Tutorial
- Angular Form Control
- Angular Routing Modules
- Sharing Data between Angular Components

BridgeLabz

Employability Delivered

Thank You