Angular Hands-on Workshop

Ervin Suhanko
Front End Software Engineer,
Team Lead & Mentor

Prerequisite

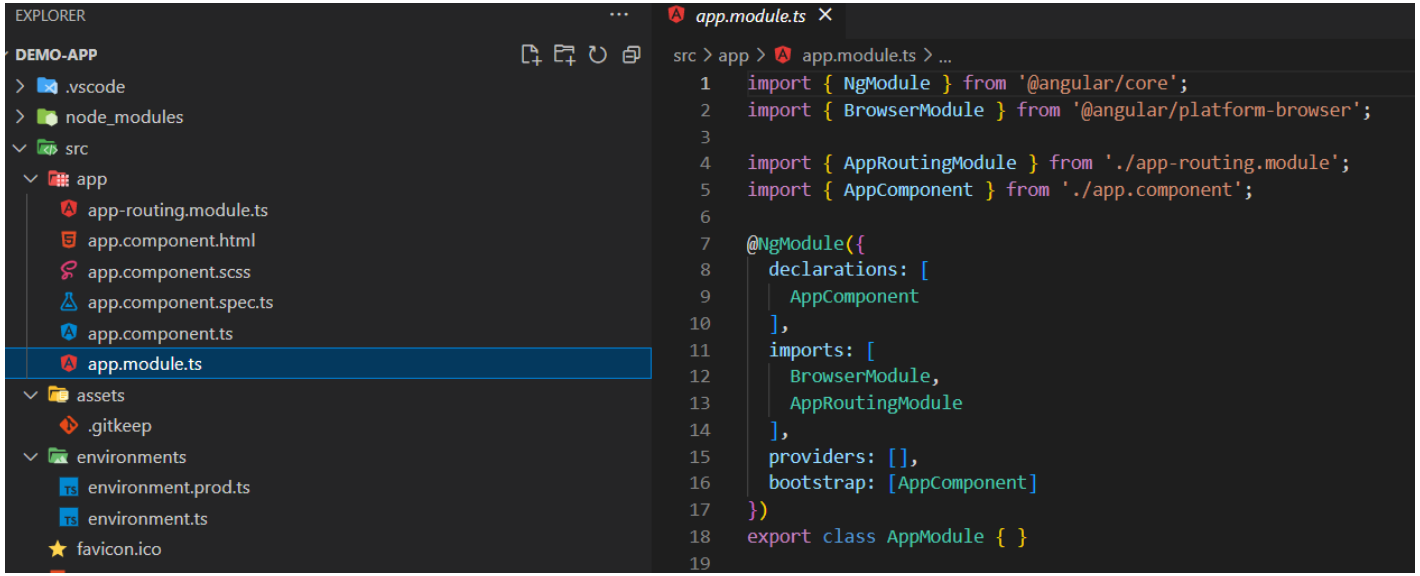
- Basic understanding of HTML, CSS, and JavaScript
- Basic understanding of Programming
- Familiarity with ES6 and Typescript is helpful

5th – 6th – 7th – 8th – 9th Sep'22 | 9:30 – 10:30 AM EST

Agenda

Day 1	Day 2	Day 3	Day 4	Day 5
Angular Introduction and Framework Overview	Modules	Routing	RxJS Essentials	Tips, Tricks and Best Practices
Angular CLI (Command-line Interface tool)	Components and Data flow	Pipes and Directives	Services and HTTP	
Angular Project Structure		Reactive Forms		
TypeScript Essentials				

Modules


EXPLORER
DEMO-APP
.vscode
node_modules
src
app
app-routing.module.ts
app.component.html
app.component.scss
app.component.spec.ts
app.component.ts
app.module.ts
assets
.gitkeep
environments
environment.prod.ts
environment.ts
favicon.ico
app.module.ts
src > app > app.module.ts > ...
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
@NgModule({
declarations: [
AppComponent
],
imports: [
BrowserModule,
AppRoutingModule
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule { }

What Is an Angular Module

- Its a class with an NgModule decorator

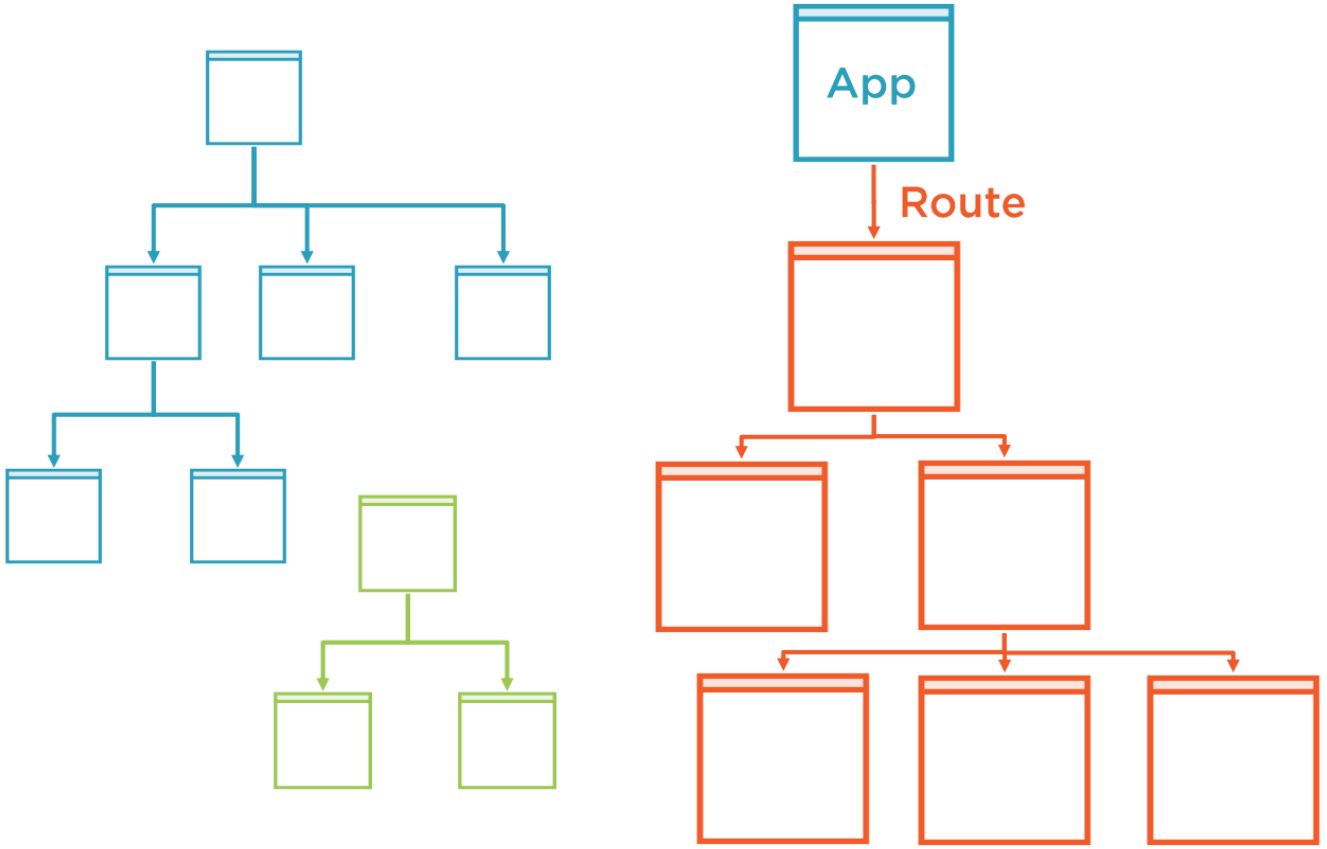
- Its purpose:
  - Organize the pieces of our application
  - Arrange them into blocks
  - Extend our application with capabilities from external libraries
  - Provide a template resolution environment
  - Aggregate and re-export

NgModule decorator

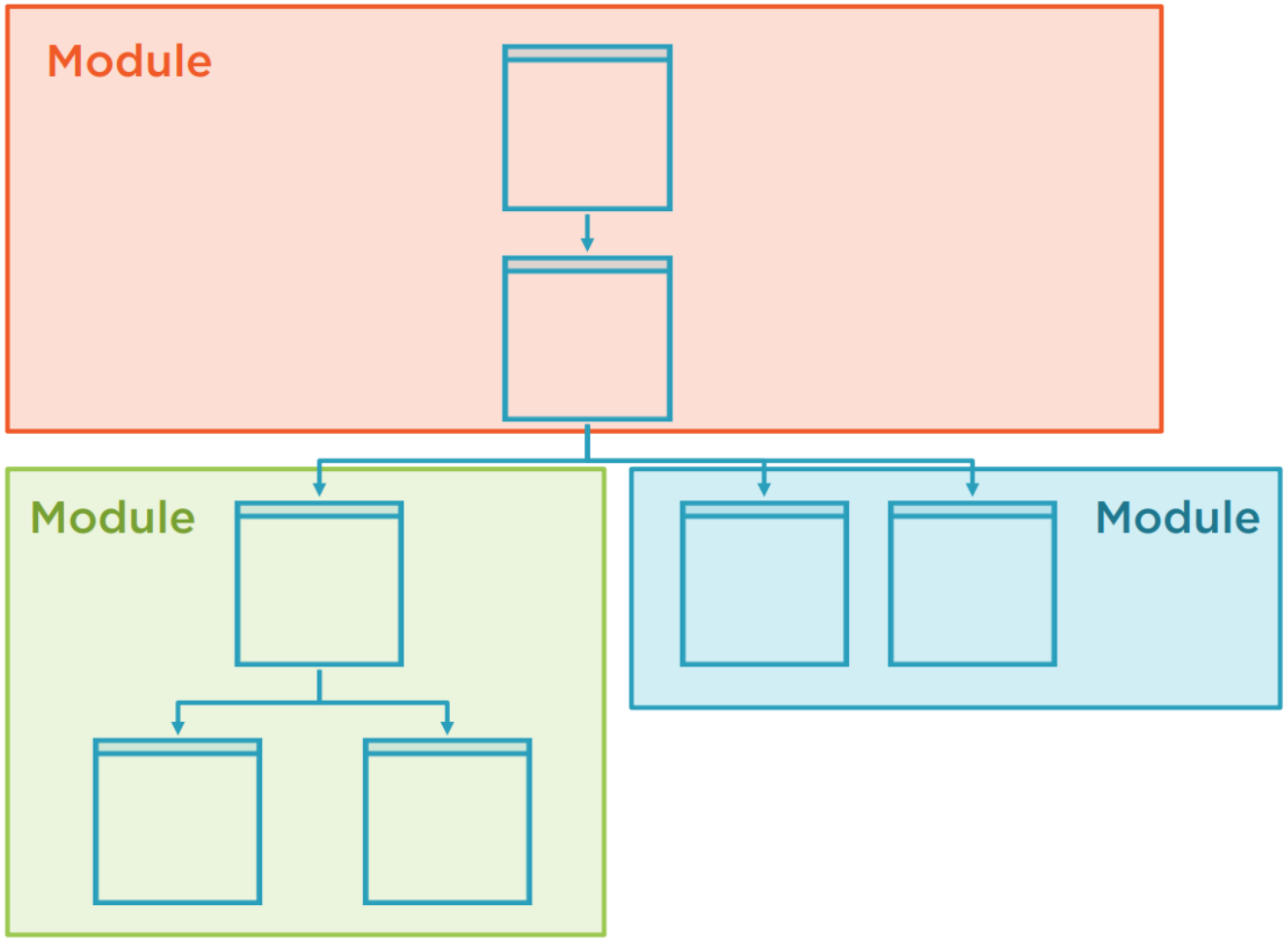
@NgModule

- Provides organization at a **framework** level
- **declarations** define view classes that are available to the module
- **imports** define a list of modules that the module needs
- **providers** define a list of services the module makes available
- **exports** define a list of modules the module makes available
- **bootstrap** defines the component that should be bootstrapped

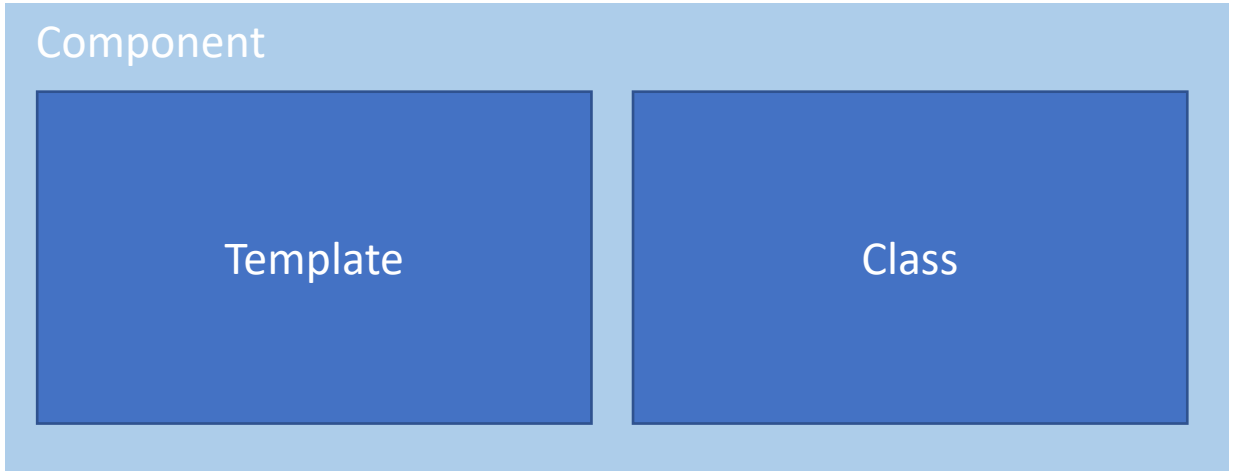
Angular Component Hierarchy



Angular Component Hierarchy

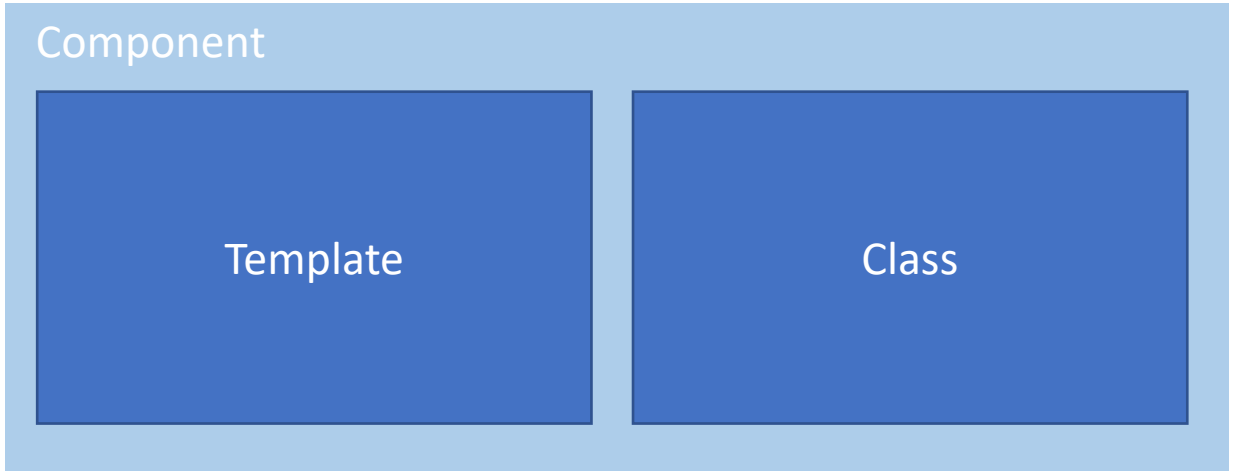


Components -Class-



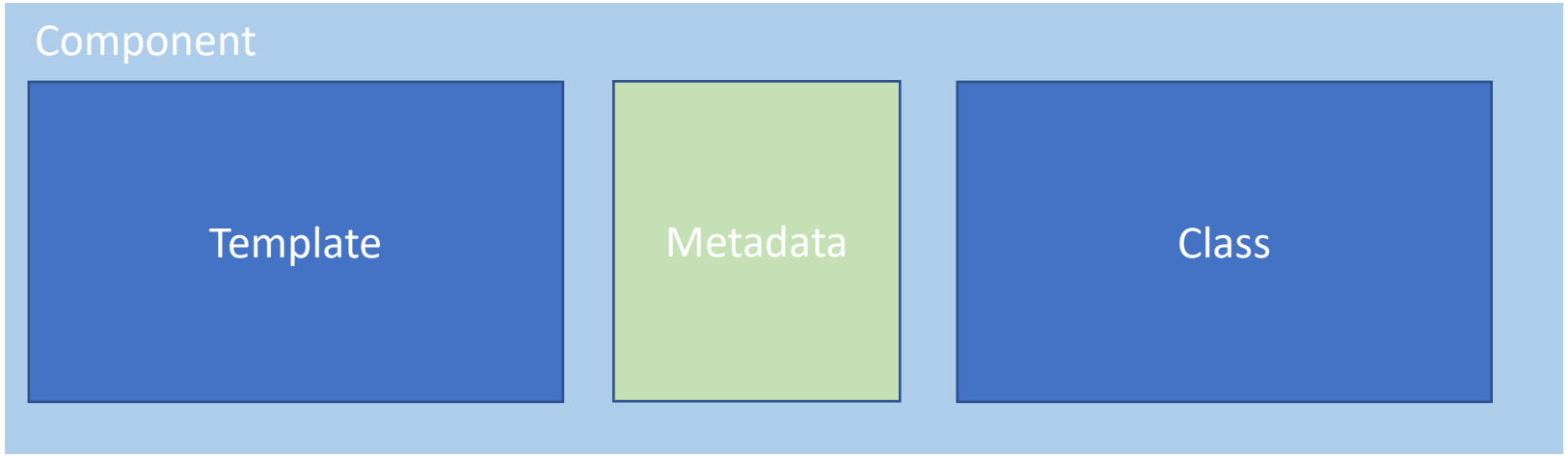
- Components are just ES6 classes
- Properties and methods of the component class are available to the template
- Providers (Services) are injected in the constructor
- The component lifecycle is exposed with hooks

Components -Template-



- A template is HTML that tells Angular how to render a component
- Templates include data bindings as well as other components and directives
- Angular leverages native DOM events and properties which dramatically reduces the need for a ton of built-in directives
- Angular leverages shadow DOM to do some really interesting things with view encapsulation

Components -Metadata-



- Metadata allows Angular to process a class
- We can attach metadata with TypeScript using decorators
- Decorators are just functions
- Most common is the @Component() decorator
- Takes a config option with the selector, templateUrl, styles, styleUrls, animations, etc

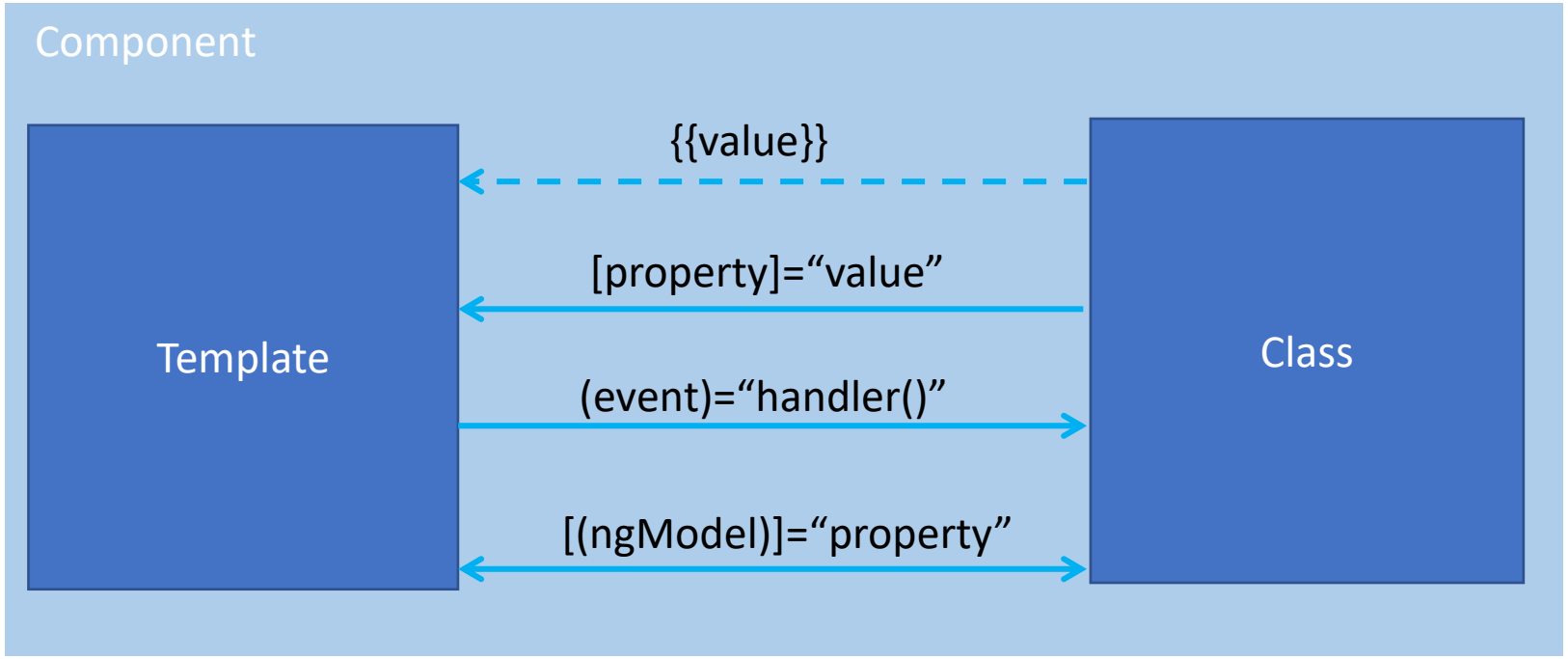
Demo

Creating a component

Data Binding

- Enables data to flow from the component to template and vice-versa
- Includes
  - interpolation
  - property binding,
  - event binding
  - two-way binding (property binding and event binding combined)
- The binding syntax has expanded but the result is a much smaller framework footprint

Data Binding



Data Binding

Interpolation

Interpolation refers to embedding expressions into marked up text. By default, interpolation uses the double curly braces {{ and }} as delimiters.

component1.component.html U

src > app > component1 > component1.component.html > h3

title Aa ab, .* No results

```
2
3
4
5  <h3>{{ courseName }}</h3>
6
```

component1.component.ts U

src > app > component1 > component1.component.ts > Component1Component

```
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-component1',
5    templateUrl: './component1.component.html',
6    styleUrls: ['./component1.component.scss']
7  })
8  export class Component1Component implements OnInit {
9
10   courseName = "Angular Hands-on Workshop";
11
12   constructor() { }
13
14   ngOnInit(): void {
15   }
16
17 }
18
```

Data Binding

Property Binding

Property binding in Angular helps you set values for properties of HTML elements or directives. Use property binding to do things such as toggle button functionality, set paths programmatically, and share values between components.

```
<h2>Property binding</h2>
<p><img alt="Property Bound item" [src]="itemImageUrl"> is the <i>property bound</i> image.</p>
```

Data Binding

Event Binding

Event binding lets you listen for and respond to user actions such as keystrokes, mouse movements, clicks, and touches.

```
<div class="group">
  <h3>Binding to a nested component</h3>
  <h4>Custom events with EventEmitter</h4>
  <app-item-detail (deleteRequest)="deleteItem($event)" [item]="currentItem"></app-item-detail>

  <h4>Click to see event target class:</h4>


<div class="parent-div" (click)="onClickMe($event)" clickable>Click me (parent)
  <div class="child-div">Click me too! (child) </div>
</div>

<h3>Saves only once:</h3>
<div (click)="onSave()" clickable>
  <button type="button" (click)="onSave($event)">Save, no propagation</button>
</div>

<h3>Saves twice:</h3>
<div (click)="onSave()" clickable>
  <button type="button" (click)="onSave()">Save with propagation</button>
</div>
```

Data Binding

Two-way binding

Two-way binding gives components in your application a way to share data. Use two-way binding to listen for events and update values simultaneously between parent and child components.

```
<h1 id="two-way">Two-way Binding</h1>
<div id="two-way-1">
  <app-sizer [(size)]="fontSizePx"></app-sizer>
  <div [style.font-size.px]="fontSizePx">Resizable Text</div>
  <label>FontSize (px): <input [(ngModel)]="fontSizePx"></label>
</div>
<br>
<div id="two-way-2">
  <h2>De-sugared two-way binding</h2>
  <app-sizer [size]="fontSizePx" (sizeChange)="fontSizePx=$event"></app-sizer>
</div>
```

Demo

Data Binding