



# ANGULAR 7

# Agenda

1. Introduction
2. Angular7 Directives - *ngIf
3. Angular7 Directives - *ngFor
4. Angular7 Directives - *ngStyle
5. Angular7 Directives - *ngClass
6. Angular7 Directives – Custom Directives
7. Angular7 Directives - *ngSwitch
8. Angular 7 Data Binding
9. Angular 7 Event Binding

# Angular7 Directives

# Directives

- **Directives** are instructions in the **DOM**.
- **Components** are also directives with a template.
- Apart from components, there are two more types of directives:

    1. Structural Directives
    2. Attribute Directives

# Structural Directives

- Structural Directives are responsible for the HTML layout. They shape or reshape the HTML view by simply adding or removing the elements in the DOM.
- These directives are the way to handle how the component or the element renders in a template.
- Basic structural directives available in Angular are:

    1. ***ngIf**
    2. ***ngFor**
    3. ***ngSwitch**

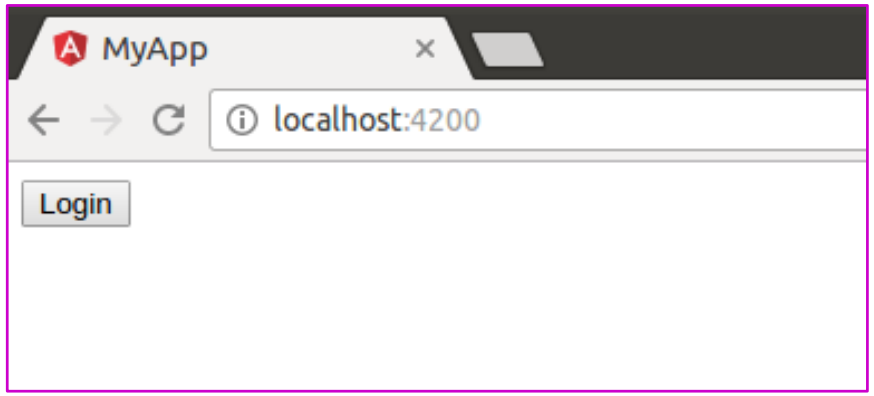
# *ngIf

Add or remove DOM elements based on conditions

```
export class AppComponent {

    userLoggedIn: boolean = false;

}
```

app.component.html

```
1   <button *ngIf="!userLoggedIn">Login</button>
2
3   <button *ngIf="userLoggedIn">Logout</button>
4
```



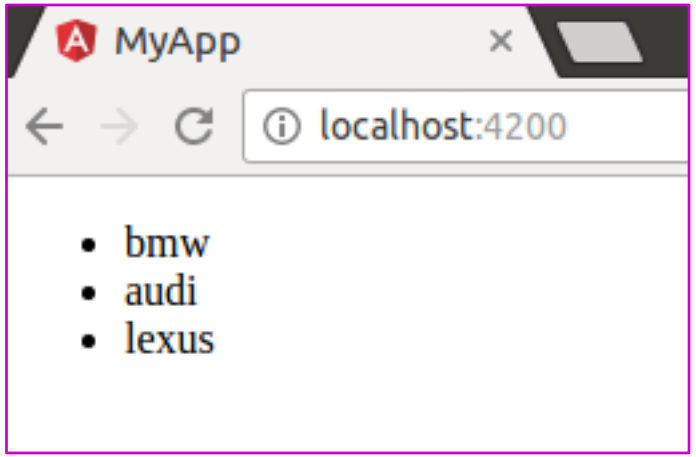
# *ngFor

Repeater directive - a way to present a list of items.

```
export class AppComponent {

  cars: Array<string> = ["bmw","audi","lexus"];

}
```

app.component.html

```
1   <ul>
2     <li *ngFor="let car of cars">
3       {{ car }}
4     </li>
5
6   </ul>
7
```



# *ngFor

Showing complex data - list of objects

car.ts

```
1   export class Car{
2
3       name:string;
4       speed: number;
5       color: string;
6   }
7
```

class Car

```
export class AppComponent {

  cars: Array<Car> = [
    {name:"bmw",color:"red",speed:200},
    {name:"audi",color:"blue",speed:300},
    {name:"lexus",color:"white",speed:100}
  ];

}
```

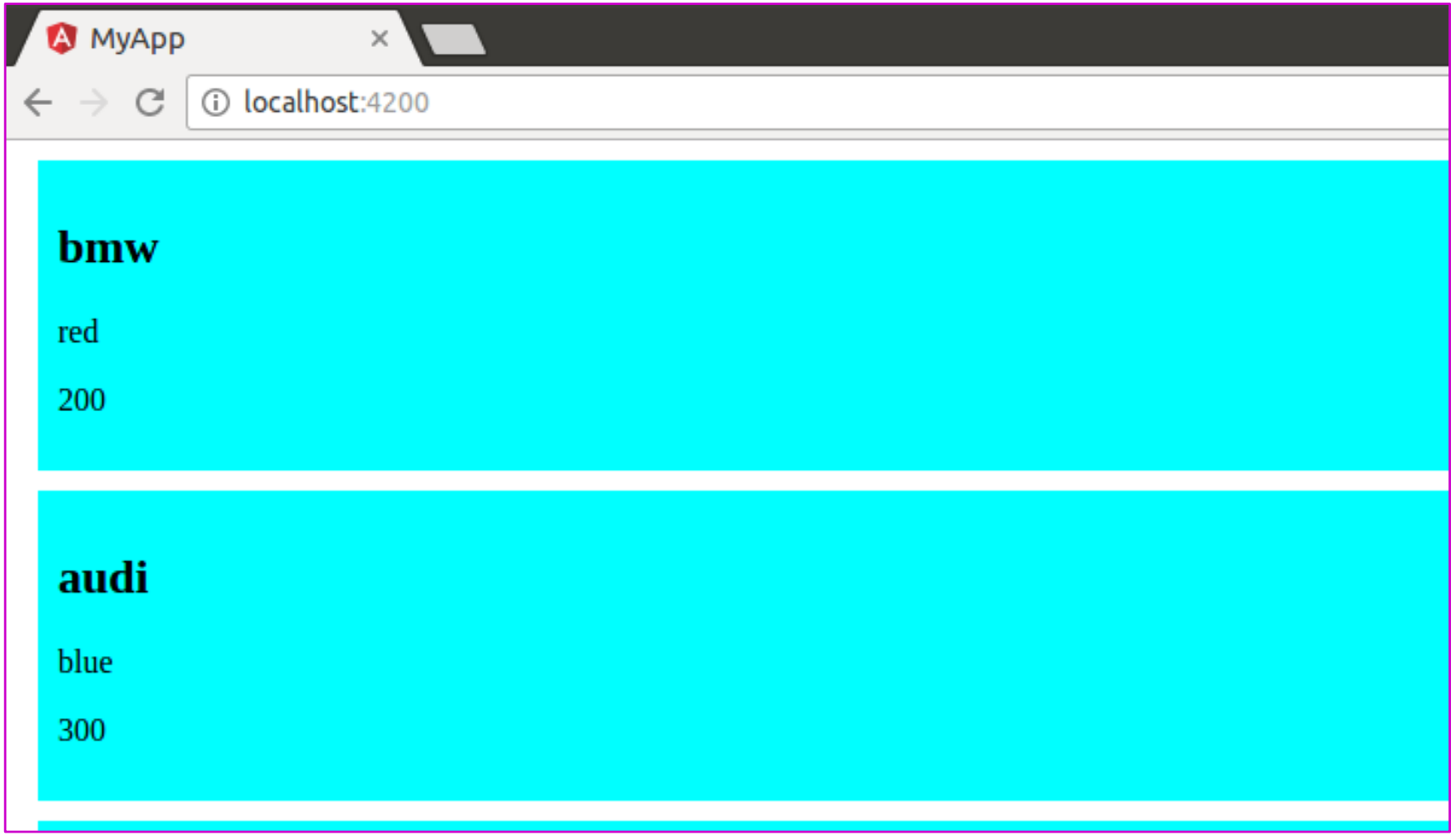
Array of Car objects

# *ngFor

app.component.html

```
1   <div
2     style="background: cyan;padding: 10px;margin: 10px"
3     *ngFor="let car of cars">
4
5     <h2>{{car.name}}</h2>
6     <p>{{car.color}}</p>
7     <p>{{car.speed}}</p>
8   </div>
```

# *ngFor



# Attribute Directives

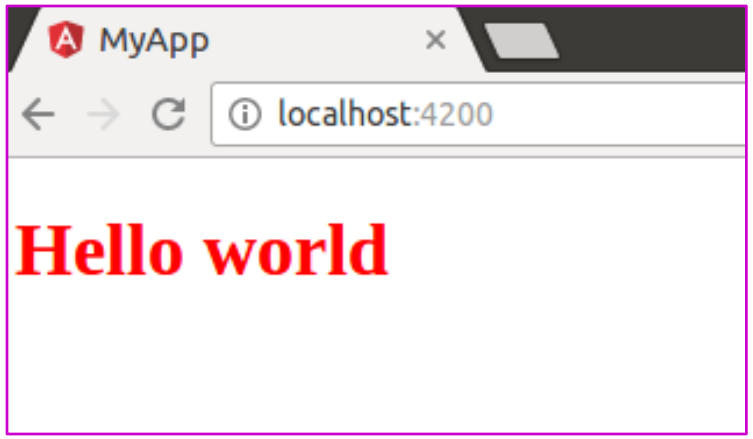
- Attribute Directives is a way to modify the appearance or behavior of an element or a component.
- There are two built-in Attribute Directives in Angular.
- ***ngStyle:** Angular provides a built-in NgStyle attribute to modify the element appearance and behavior.
- ***ngClass:** This attribute is used to change the class attribute of the element in DOM or the component to which it has been attached.

# *ngStyle

ngStyle allows to set inline styles dynamically to the elements

app.component.html

```
1   <h1 [ngStyle]="{color: getColor()}"> Hello world </h1>
2
```




```
export class AppComponent {

  colorFlag: boolean = true;

  getColor(){

    if(this.colorFlag){
      return "red";
    }else{
      return "blue";
    }

  }
```

# *ngClass

ngClass allows to set css class dynamically to the elements

app.component.css

```
1   .highlight{
2       color: red;
3       background-color: yellow;
4   }
```

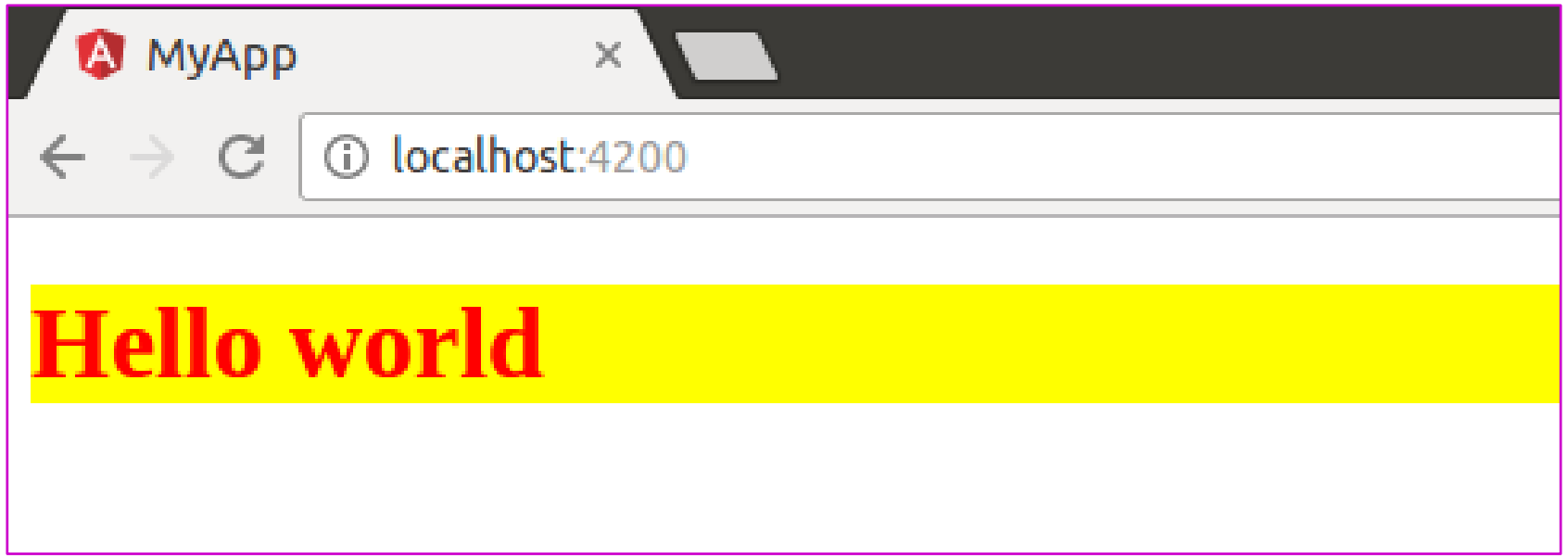
app.component.ts

```
export class AppComponent {

  toHighlight: boolean = true;

  highlightElement(){

    return this.toHighlight;

  }
}
```

app.component.html

```
1   <h1 [ngClass]="{highlight: highlightElement()}"> Hello world </h1>
2
```

# *ngClass



# Custom Directive

To create a custom attribute, run the following command:

**ng generate directive directive-name**

```
C:\Users\Admin\Desktop\Angular\demo>ng generate directive highlighter
CREATE src/app/highlighter.directive.spec.ts (244 bytes)
CREATE src/app/highlighter.directive.ts (151 bytes)
UPDATE src/app/app.module.ts (483 bytes)
```

Now let's write logic to highlight an element with yellow color on applying the above directive.

# Custom Directive

highlighter.directive.ts

```
1   import { Directive } from '@angular/core';
2
3   @Directive({
4     selector: '[appHighlighter]'
5   })
6   export class HighlighterDirective {
7
8     constructor() { }
9
10  }
11
```

import the "Directive" decorator from the angular/core module.

Name of directive

Decorator (meta data)

# Custom Directive

TS highlighter.directive.ts ×

```
1   import { Directive, ElementRef, Renderer2, OnInit } from '@angular/core';
2
3   @Directive({
4     selector: '[appHighlighter]'
5   })
6   export class HighlighterDirective implements OnInit {
7
8     constructor(private elRef: ElementRef, private renderer: Renderer2) { }
9
10    ngOnInit(){
11
12      this.renderer.setStyle(this.elRef.nativeElement,'background-color','yellow')
13
14    }
15
16  }
17
```

Style attribute to be changed

Attribute value

Reference of element affected by the directive

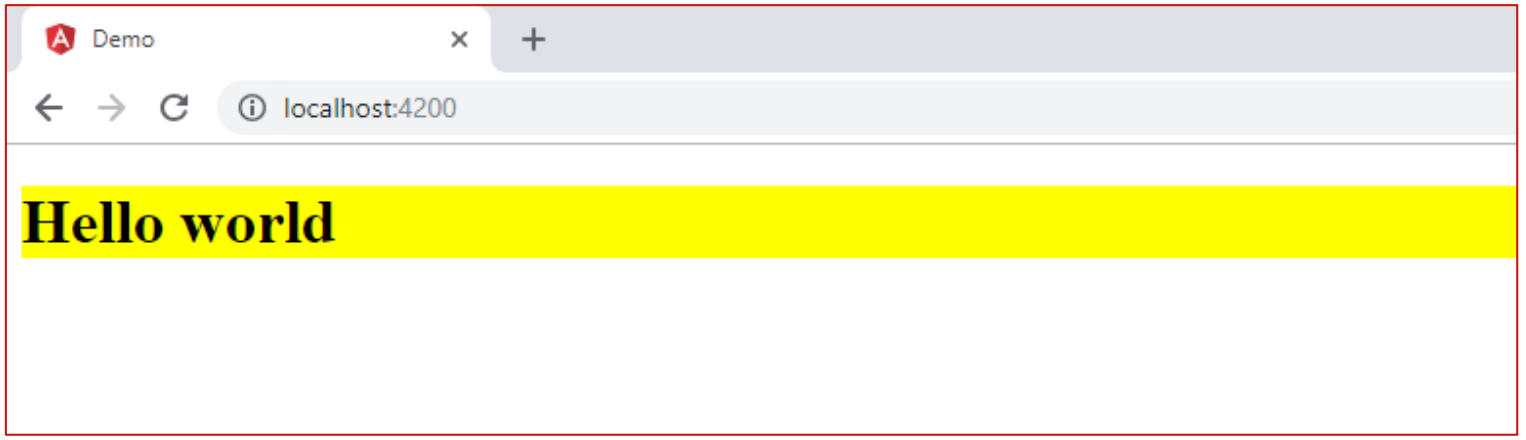
Renderer class is a built-in service that provides an abstraction for UI rendering manipulations

# Custom Directive

Applying the appHighlighter directive

app.component.html

```
1
2    <h1 appHighlighter> Hello world </h1>
3
4
```



# Custom Directive

**@HostListener** is used to apply directive only at specific event.

```
@Directive({
  selector: '[appHighlighter]'
})
export class HighlighterDirective {

  constructor(private elRef: ElementRef, private renderer: Renderer2) { }

  @HostListener('mouseenter') mouseover(){
   this.renderer.setStyle(this.elRef.nativeElement,'background-color','yellow')
   }

  @HostListener('mouseleave') mouseleave(){
   this.renderer.setStyle(this.elRef.nativeElement,'background-color','transparent')
   }


}
```

Applying directive only at mouse-enter event using @HostListener

Applying directive only at mouse-leave event using @HostListener

# Custom Directive

**@HostBinding** is used to define specific attribute of an element on which the directive should be applied.It is equivalent to renderer, but makes the code shorter and easy to use.

```
@HostBinding('style.backgroundColor') backgroundColor : string = 'transparent';

@HostListener('mouseenter') mouseover(){
    this.backgroundColor = 'yellow';
}
@HostListener('mouseleave') mouseleave(){
    this.backgroundColor = 'transparent';
}
```

Makes the code short and simple.

Defines the attribute to be changed.

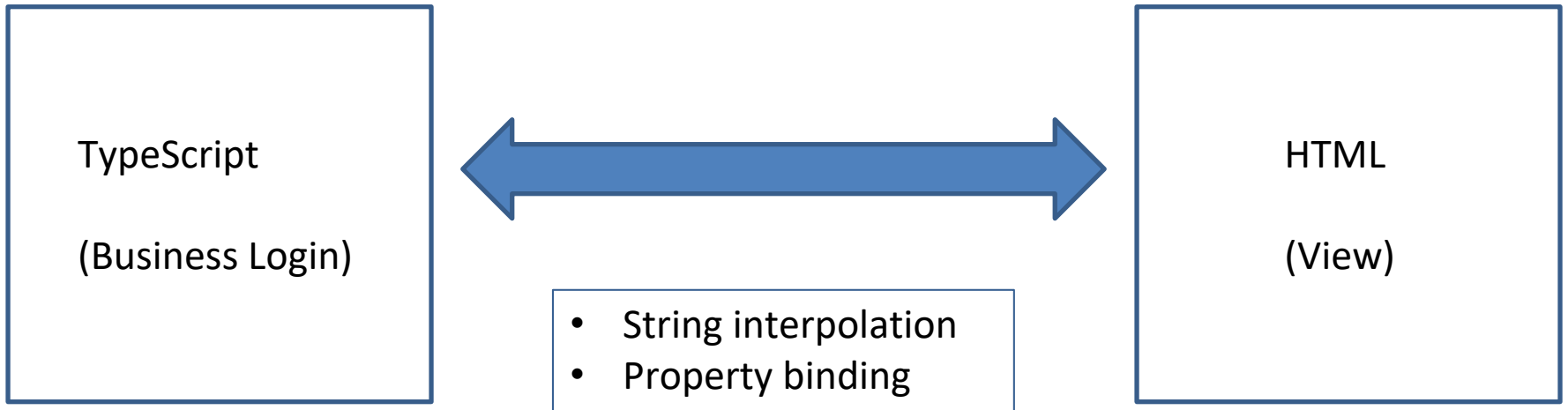
# *ngSwitch

- ngSwitch is basically comprised of two directives, an attribute directive and a structural directive. It's very similar to switch case statement in Javascript and other programming languages.
- The ngSwitch directive lets you hide/show HTML elements depending on an expression.
- We can also define a default section, by using the ng-switch default directive, to show a section if none of the other sections get a match.
- **ngSwitch, ngSwitchCase and ngSwitchDefault.**

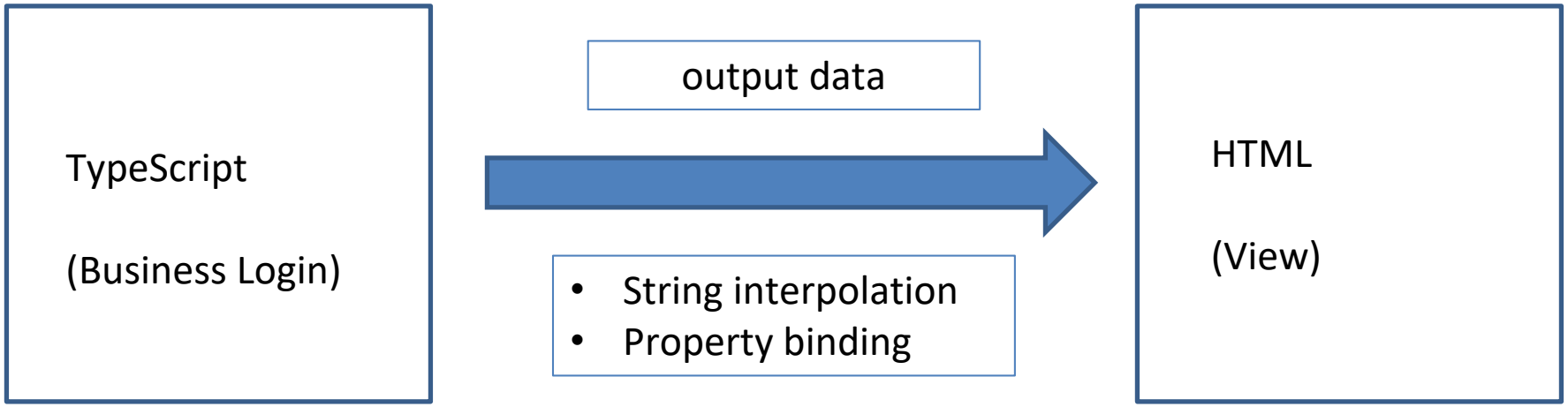
# Angular7 Data Binding

# Data Binding

Databinding is basically communication between our view(HTML) and business login (TypeScript)



# Data Binding

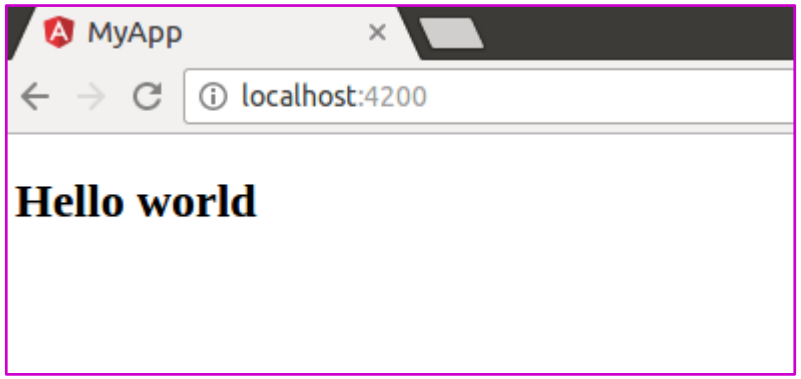


# String Interpolation

```
export class AppComponent {

  greeting = "Hello world";

}
```

app.component.html

```
1   <h2> {{greeting}} </h2>
2
3
```




Any data that is of type string, or which can be converted into string goes inside the curly brackets.

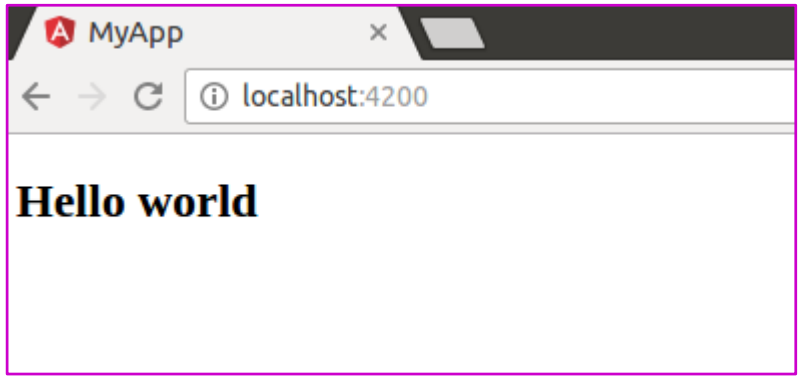
# String Interpolation

```
export class AppComponent {

  greeting = "Hello world";

  greet(){
    return this.greeting;
  }
}
```

app.component.html

```
1   <h2> {{ greet() }} </h2>
2
3
4
```

It can also have a method that returns string.



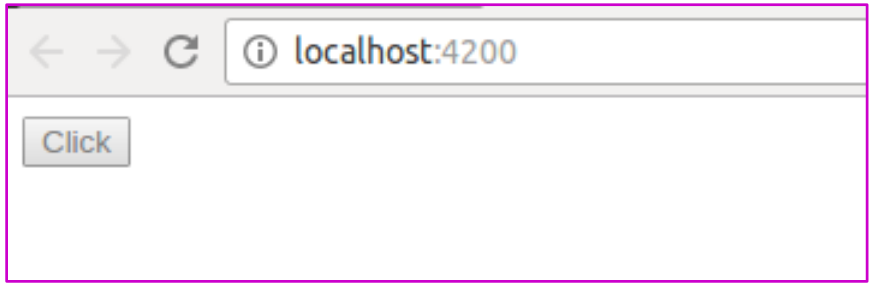
# Property Binding

```
export class AppComponent {

  isDisabled: boolean = true;

}
```

Properties are placed between [] brackets

app.component.html

```
1   <button [disabled]="isDisabled">Click</button>
2
3
```




Dynamically binding properties of html tag

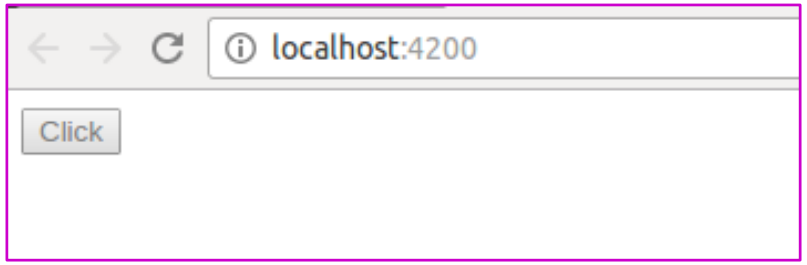
# Property Binding

```
export class AppComponent {

  isDisabled(){

    return true;

  }
```

Properties are placed between [] brackets

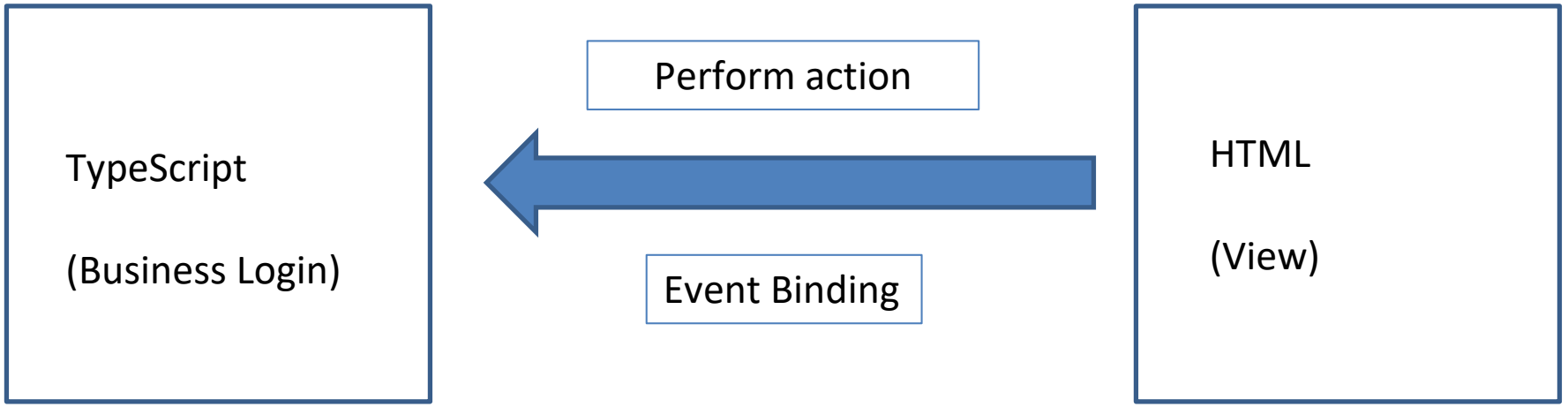
app.component.html

```
1   <button [disabled]="isDisabled()">Click</button>
2
3
4
```




Dynamically binding properties of html tag

# Event Binding

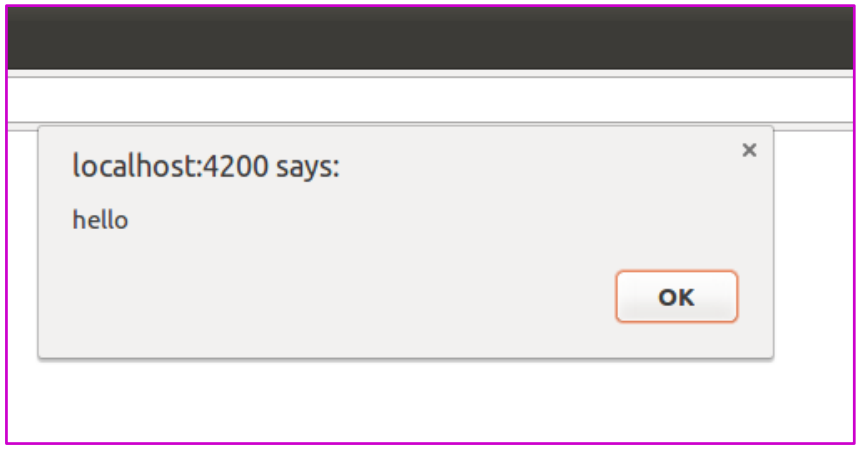


# Event Binding

Events are placed between () brackets

app.component.html

```
1   <button (click)="sayHello()">Click</button>
2
3
```



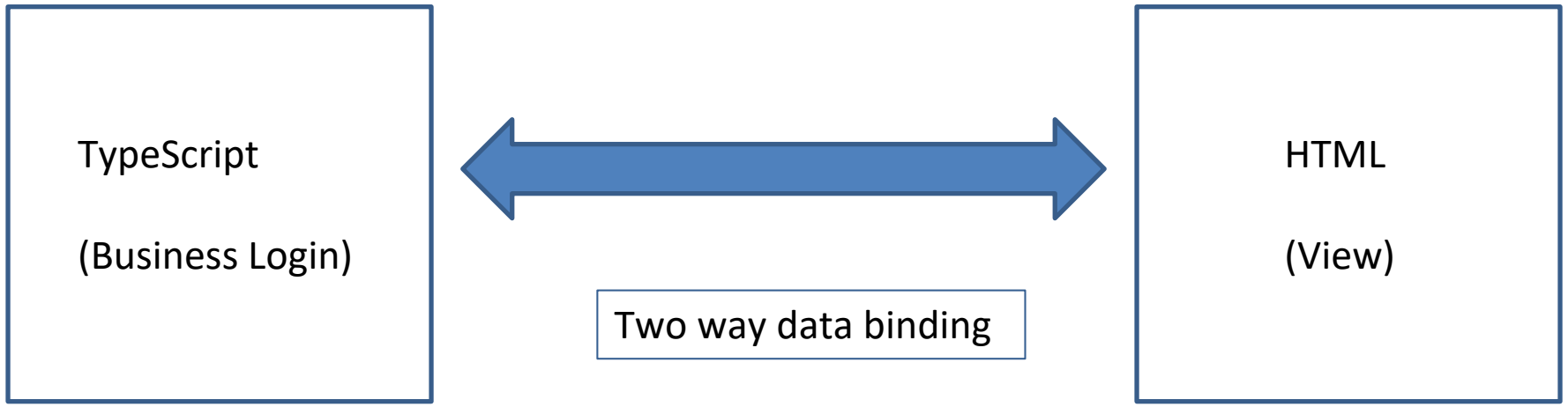

```
export class AppComponent {

  sayHello(){
    alert("hello");
  }

}
```

# Event Binding

## HTML events

- click
- mouseenter
- mousedown
- keyup
- keydown
- keypress
- drag
- drop
- submit
- scroll
- focus
- blur

# Two Way Data Binding



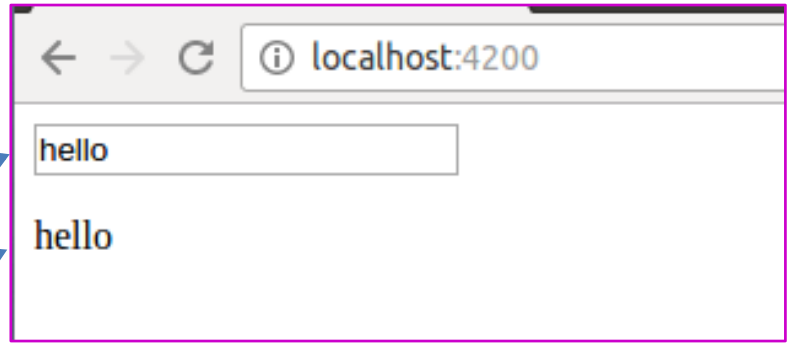
# Two Way Data Binding

app.component.html

```
1   <input [(ngModel)]="data">
2
3   <p>{{ data }}</p>
4
5
6
```

```
export class AppComponent {

    data = "hello";

}
```

Deprecated




Changing the value of data at <input>, changes the same value of data at <p> at same time.

greatlearning
Power Ahead

# Thank You