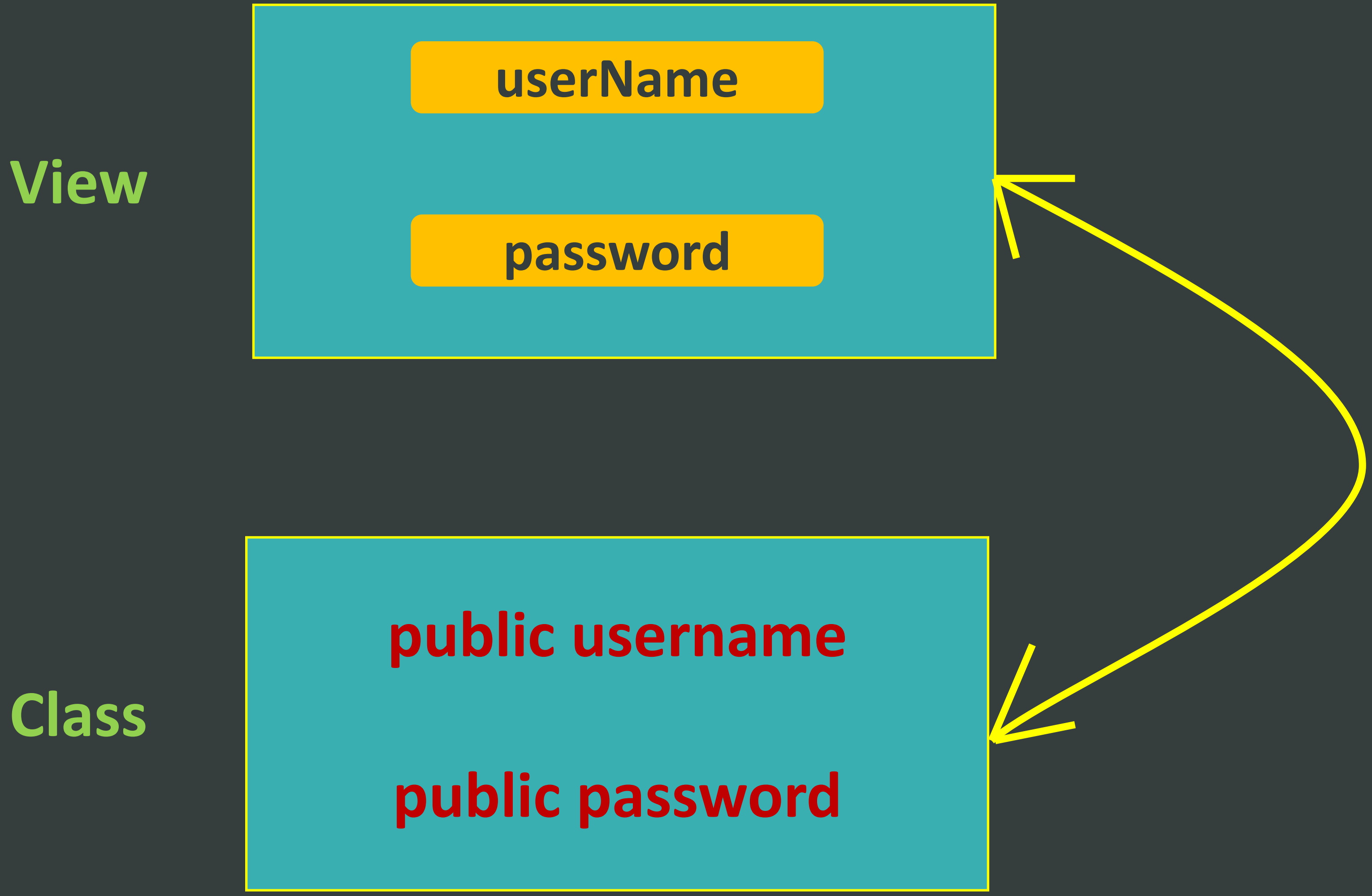
Angular (Part – 2)

PROF. P. M. JADAV
ASSOCIATE PROFESSOR
COMPUTER ENGINEERING DEPARTMENT
FACULTY OF TECHNOLOGY
DHARMSINH DESAI UNIVERSITY, NADIAD

Content

- Two-way Binding
- Pipes
- Structural Directives
- Custom Attributes

Two Way Binding



Two Way Binding ([()])

Banana in a Box

```
<input [(ngModel)]="username">

<p>Hello {{username}}!</p>
```

app.module.ts

```
import { FormsModule} from '@angular/forms';
@NgModule({
        …
        imports: [
                BrowserModule,
                FormsModule          //contains ngModel
        ],
        …
})
```

Two Way Binding (without ngModel)

Model to View
(property binding)

View to Model
(event binding)

```
<input
[value]="username"
(input)="username = $event.target.value"
>
        <p>Hello {{username}}!</p>
```

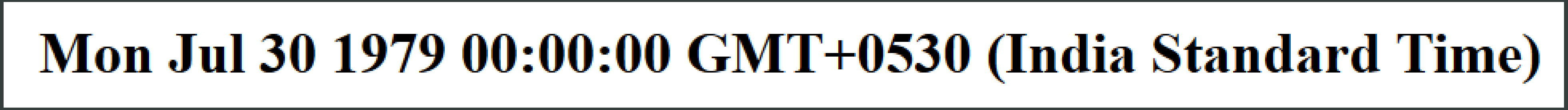
Pipes (|)

- Every application starts out with:
  - get data
  - transform them, and
  - show them to users
- A pipe takes in data as input and transforms it to a desired output

Pipes (|)

```
<h2> {{ birthday }} </h2>
```

```
public birthday : Date = new Date(1979, 6, 30);
//JavaScript counts months from 0 to 11
```

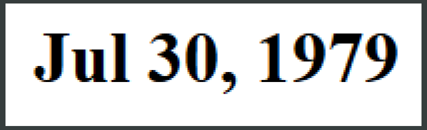


```
Mon Jul 30 1979 00:00:00 GMT+0530 (India Standard Time)
```

Pipes (|)

```
<h2> {{ birthday | date }} </h2>
```

```
public birthday : Date = new Date(1979, 6, 30);
//JavaScript counts months from 0 to 11
```

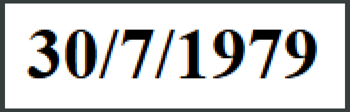


```
Jul 30, 1979
```

Pipes (|)

```
<h2> {{ birthday | date : 'd/M/y' }} </h2>
```

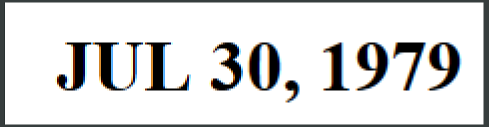
```
public birthday : Date = new Date(1979, 6, 30);
//JavaScript counts months from 0 to 11
```



```
30/7/1979
```

Chaining of Pipes (|)

```
<h2> {{ birthday | date | uppercase }} </h2>
```



JUL 30, 1979

Built-in Pipes (|)

- CurrencyPipe
- DecimalPipe
- LowerCasePipe
- PercentPipe
- AsyncPipe
- DatePipe
- JsonPipe
- UpperCasePipe
- SlicePipe

Pipes Examples (|)

{{ 1234.56 | currency : 'INR' }}	₹1,234.56
{{ 0.123456 | percent : '2.1-3' }}	12.346%
{{ [1, 2, 3, 4, 5, 6] | slice : 1 : 3 }}	2,3

Custom Pipes (|)

Run following angular-cli command in the component directory where you wish to use pipe

```
ng g p mypower
```

g – generate

p – pipe

mypower – name of the pipe

app.module.ts

```
…
import { MypowerPipe } from './test/mypower.pipe';
@NgModule({
        declarations: [
                AppComponent,   TestComponent,   MypowerPipe
        ],
        imports: [   BrowserModule,   FormsModule  ],
        providers: [],
        bootstrap: [AppComponent]
})
export class AppModule { }
```

mypower.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';
@Pipe({
        name: 'mypower'
})
export class MypowerPipe implements PipeTransform {
        transform(value: number, exponent?: string): number {
                let exp = parseFloat(exponent)
                return Math.pow(value, isNaN(exp) ? 1 : exp)
        }
}
```

```
<h3> {{ 2 | mypower : '6' }} </h3>
```

$\rightarrow$ 64

Directives

Directives is a class, which is declared as @Directive. We have 3 directives in Angular.

1. Component Directives: directives with a template (app-root). Component (subclass of Directive) is a directive with a view/template
2. Structural Directives: change the DOM layout by adding and removing DOM elements (e.g. *ngIf, *ngFor, *ngSwitch)
3. Attribute Directives: change the appearance or behaviour of an element, component, or another directive (built-in : ngStyle, ngClass, ngModel)

Structural Directives

- Add/remove HTML elements to/from DOM

- ngIf
- ngSwitch
- ngFor

*ngIf Directive

```
<h2 *ngIf = "true"> Good Morning</h2>
```

OR

```
<h2 *ngIf = "isMorning"> Good Morning</h2>
```

Class Property

*ngIf Directive

```
<p *ngIf="condition">
  I love Angular!
</p>
```

Equivalent

```
<ng-template [ngIf]="condition">
 <p>
    I love Angular!
 </p>
</ng-template>
```

ng-template

```
<ng-template>
  <h2> Good Morning!  </h2>
</ng-template>
```

- An ng-template is a composition of elements but Angular does not render it by default
- It is only defined in the source code
- In the browser's HTML source code it will appear inside comment (<!-- -->)

*ngIf with else

```
<h2 *ngIf = "isMorning; else elseBlock">
        Good Morning
</h2>
<ng-template #elseBlock>
  <h2>  Good day!   </h2>
</ng-template>
```

*ngIf with then and else

```
<div *ngIf = "isMorning; then thenBlock else elseBlock"> </div>


<ng-template #thenBlock>
  <h2>       Good Morning!            </h2>
</ng-template>
<ng-template #elseBlock>
  <h2>       Good day!            </h2>
</ng-template>
```

ng-container

- ng-container serves as a container for elements which can also accept structural directives
- It is not rendered to the DOM

ng-container

```
<ng-container *ngIf="store.products">
  <ng-container *ngIf="store.products.length > 0 else noProducts">
    <ul *ngFor="let product of store.products">
      <li>{{ product.name }}</li>
    </ul>
  </ng-container>
</ng-container>
<ng-template #noProducts>
  <p>There are no products in this store</p>
</ng-template>
```

*ngSwitch

color property

```
<div [ngSwitch]="color">
<h2 *ngSwitchCase="'red'">  You picked up red        </h2>
<h2 *ngSwitchCase="'green'">You picked up green    </h2>
<h2 *ngSwitchCase="'blue'">You picked up blue      </h2>
<h2 *ngSwitchDefault>Pick again                    </h2>
</div>
```

*ngFor

colors property

```
<div *ngFor = "let color of colors">
      <h2> {{ color }} </h2>
</div>
```

*ngFor

colors property

```
<div *ngFor = "let color of colors; index as i">
      <h2> {{ i }} {{ color }} </h2>
</div>
```

*ngFor

colors property

```
<div *ngFor = "let color of colors; first as f">
      <h2> {{ f }} {{ color }} </h2>
</div>
```

*ngFor

colors property

```
<div *ngFor = "let color of colors; last as l">
      <h2> {{ l }} {{ color }} </h2>
</div>
```

*ngFor

colors property

```
<div *ngFor = "let color of colors; odd as o">
      <h2> {{ o }} {{ color }} </h2>
</div>
```

*ngFor

colors property

```
<div *ngFor ="let color of colors; even as e">
    <h2> {{ e }} {{ color }} </h2>
</div>
```

Built-in Attribute Directives

- Attribute directives listen to and modify the behaviour of other
  - HTML elements,
  - attributes,
  - properties, and
  - Components
- They are usually applied to elements as if they were HTML attributes, hence the name.
- e.g. ngStyle, ngModel, ngClass

Creating Custom Attribute Directive

- Create the directive class file in a terminal window with this CLI command.

```
ng g d highlight
```

g – generate

d – directive

highlight – name of directive

app.module.ts

```
…

import { HighlightDirective } from './highlight.directive';
@NgModule({
        declarations: [ AppComponent, MypowerPipe, HighlightDirective],
        imports: [   BrowserModule,   FormsModule  ],
        providers: [],
        bootstrap: [AppComponent]
})
export class AppModule { }
```

highlight.directive.ts (generated code)

```
import { Directive } from '@angular/core';
@Directive({
		selector: '[appHighlight]'
})
export class HighlightDirective {
	constructor() { }
}
```

highlight.directive.ts (edited code)

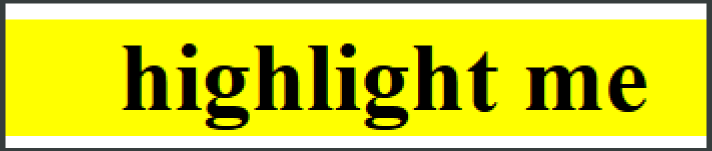
```
import { Directive, ElementRef } from '@angular/core';

@Directive({
        selector: '[appHighlight]'
})

export class HighlightDirective {
        constructor(el: ElementRef) {
                el.nativeElement.style.backgroundColor = 'yellow';
        }
}
```

test.component.ts

```
<h2 appHighlight>highlight me </h2>
```


highlight me

References

- https://angular.io/docs