# **ANGULAR**

* npm – node package manager
* npm install -g @angular/cli
* ng new my-dream-app
* package.json – all dependencies
* node\_modules – all dependencies installed

Component:

* always has template – html
* possibly css file
* typescript file – definitions of component
* it is a separated reusable part of the page,
* it has its own logic

Data binding:

* in component you have title = app
* in html template you have <h1> Hi this is my {{ title }} </h1>
* Result: Hi this is my app

App-root

* Your own selector defined in component as

selector: ‘app-root’

result : <app-root> </app-root>

Directive:

* ngModel

Everything starts in main.ts

main.ts > app.module.ts > app.component.ts > app.component.html

Angular is a JS framework changing your DOM (HTML) at runtime!

Decorater: @component - Enhance your classes, elements

Meta data for component @component ({ })

Module: Bundle different pieces – component - into packages

**CLI commands**

Vytvorenie componentu:

* ng generate component name\_of\_component = ng g c name\_of\_component

/ spec file is usually used for testing /

Do not forget to update app.module after creating component!

templateUrl: app.component.html

template: <h2> </h2>

styleUrls: [app.component.css]

styles: [ {h3{ color : blue }} ]

As for selector in @component there are some choices:

selector: ‘app-root’ – element

‘[app-root]’ - attribute

‘.app-root’ – class

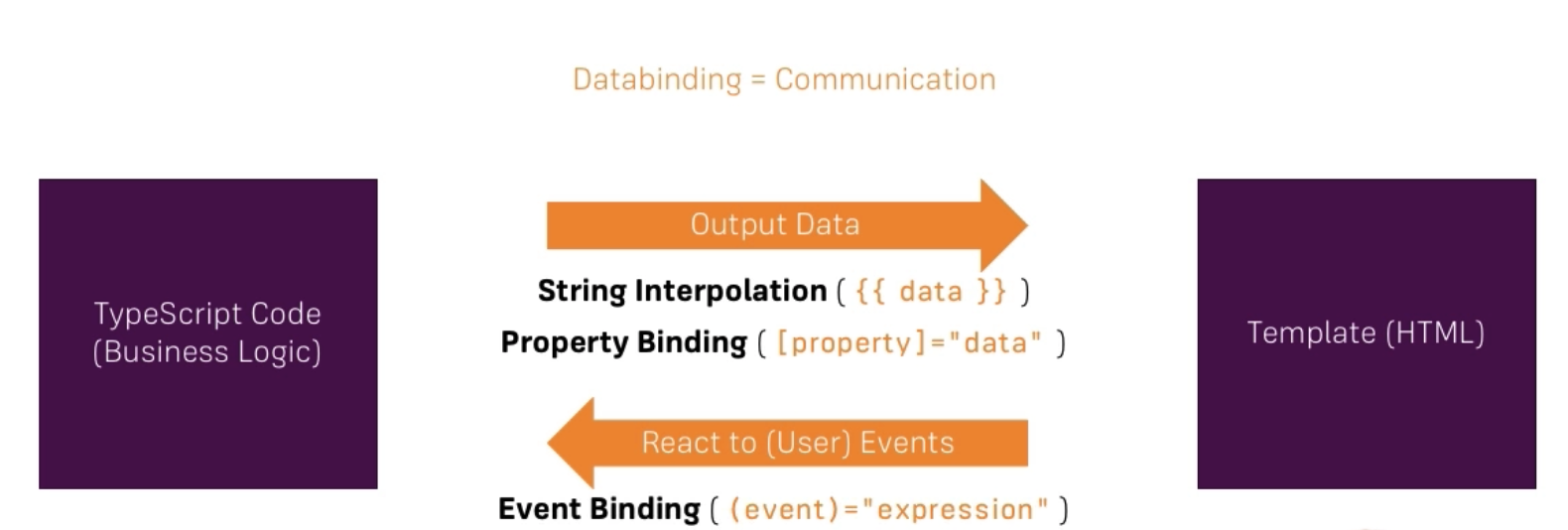
**Databinding** – communication between typescript code and template(html)

**Output Data**:

* string interpolation ( {{ data }} )
* property binding ( [property] = ”data” )

React to (user) Events **(Event binding)**

* (event) = “expression”





**String interpolation:**

You can use:

* {{ ‘server’ }} – normal string
* {{ serverId }} – data binding – defined in component.ts
* Has to resolve string in the end but number can be easily converted to a string so you can use number in {{ }}
* Methods

You cannot use:

* Multiple line expression
* For, if, …

**Property binding:**

**[disabled] =** we want to dynamically bind some property

**Why use angular?**

It is easy to interact with DOM to change something in runtime

**Property binding vs string interpolation:**

If you want to output something in your template print the test – string interpolation

If you want to change some property – property binding

**Event binding**

(click) = “method\_from\_component.ts()”

How do you know to which Properties or Events of HTML Elements you may bind? You can basically bind to all Properties and Events - a good idea is to console.log() the element you're interested in to see which properties and events it offers.

Important: For events, you don't bind to onclick but only to click (=> (click)).

The MDN (Mozilla Developer Network) offers nice lists of all properties and events of the element you're interested in. Googling for YOUR\_ELEMENT properties or YOUR\_ELEMENT events should yield nice results.

onUpdateServerName($event):

* $ = Data emitted with that event, capture the data

**Two way data binding:**

Important: For Two-Way-Binding (covered in the next lecture) to work, you need to enable the ngModel directive. This is done by adding the FormsModule to the imports[] array in the AppModule.

You then also need to add the import from @angular/forms in the app.module.ts file:

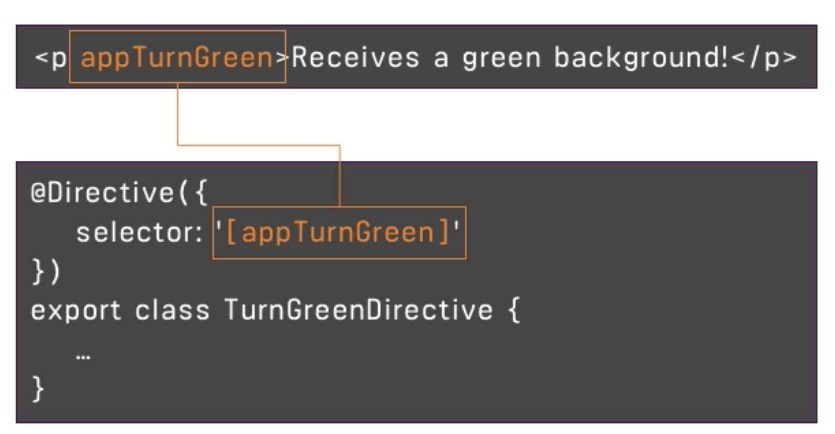
import { FormsModule } from '@angular/forms';

ngModel = directive

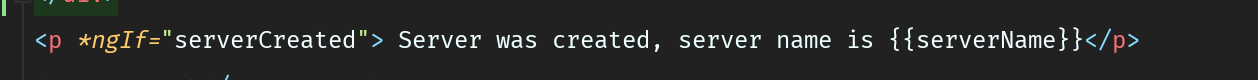
<input [(ngModel)] = “serverName”>

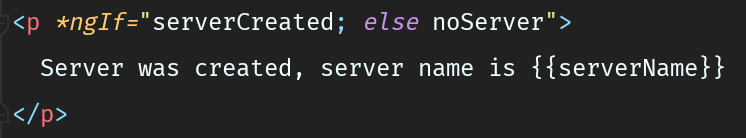
**Directives**

* Directives are instructions in the DOM
* Example: *<p appTurnGreen> Receives a green background! </p>*

**

**ngIf Output data conditionally** (ngIf is directive)

****



Styling element dynamically with **ngStyle**

***Unlike structural directives, attribute directives don’t add or remove elements. They only change the element they were placed on.***

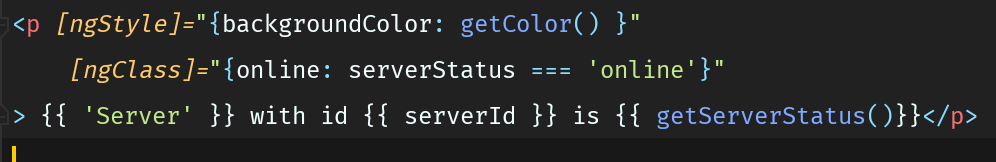


* We are binding to a property of the directive

You can use:

* <p [ngStyle]=”{ backgroundColor: red }”>
* <p [ngStyle]=”{ ‘background-color’: red”>

Applying CSS classes dynamically with ngClass



styles: [`

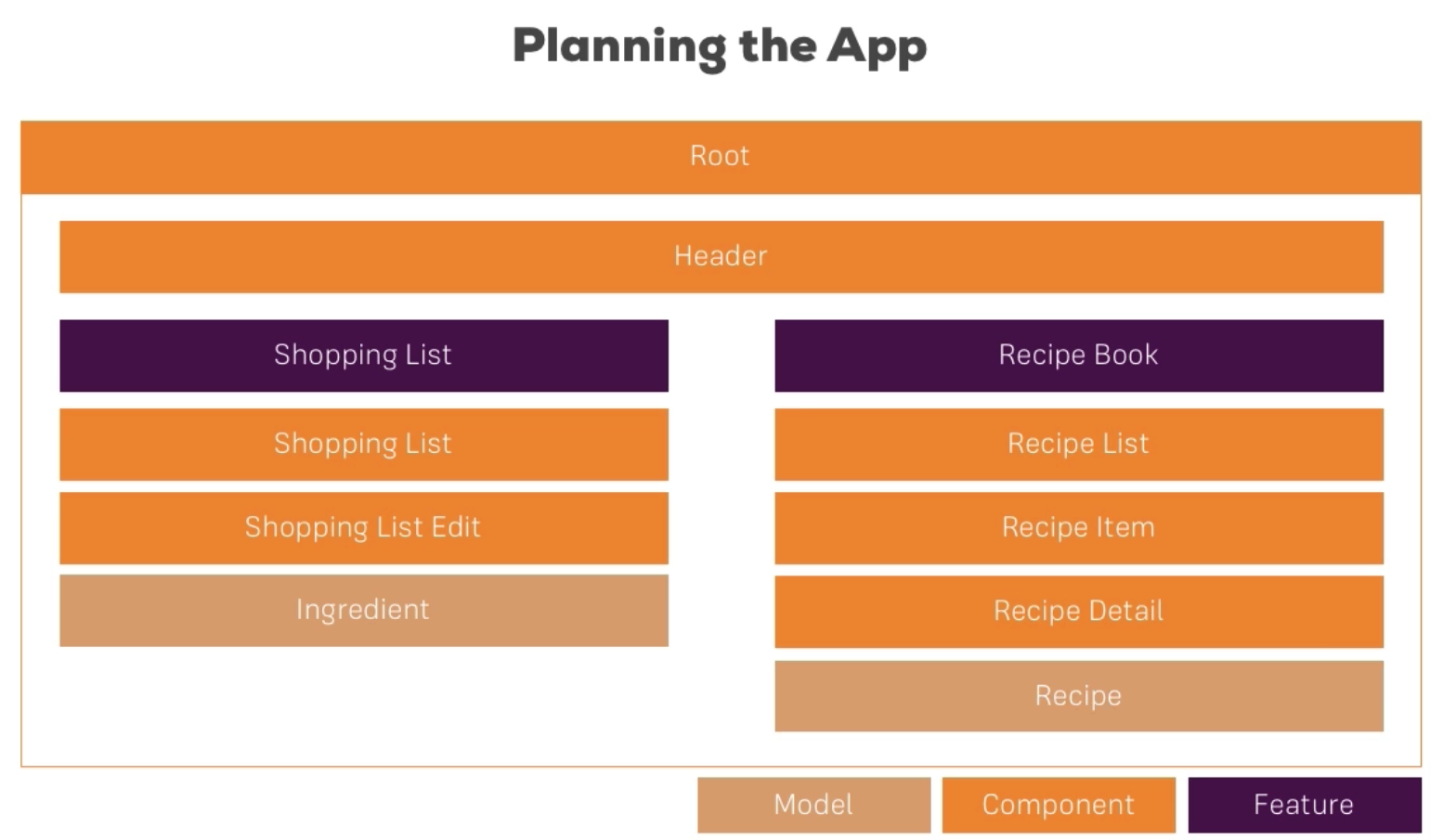
.online{

color: white;}`]}

How to make toggling paragraph - is displayed or not

<button  
 *class=*"btn btn-primary"  
 *(click)=*"displayed = !displayed"  
>  
 Display details  
</button>  
<p *\*ngIf=*"displayed"> Secret password = tuna</p>

Project:

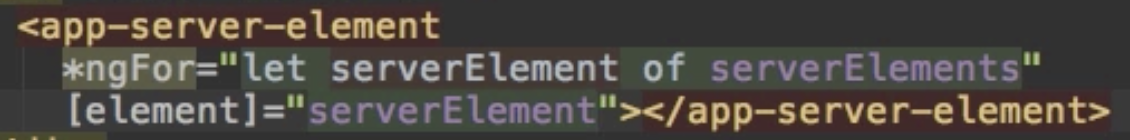


You can use:

src="{{recipe.imagePath}}" or [src]="recipe.imagePath"

\*ngFor=”let recipe of recipes”

**All properties of components are by default only accessible inside component not from outside. You have to be explicit which properties you want to expose**. You need to add decorator **@Input()**



Output nastavuješ vlastný event

A input keď si nastavuješ vlastnú property