



Home



Explore



Subscriptions



Library



History

All

Mixes

Music

Angular

Computer Science

Sushant Singh Rajput

Satsang

Cooking shows

Manoj Bajpayee

Madhuri I



Premium Quality, Luxury Laminates

800+ Designs | 100+ Textures | 1 New Design every 4 days

Ad Royale Touche

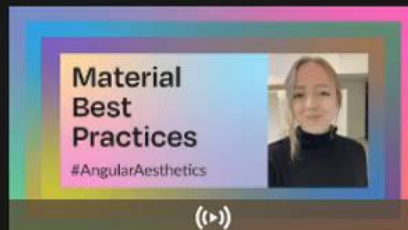


Manoj Bajpayee's BEST response to Karan Johar on...

BookMyTV

8.1M views • 4 years ago

2:02



Mix - Angular

More from this channel for you



Get Started with Angular Material |...

Angular

9.4K views • 2 months ago

7:11



Easy and Quick Quesadilla...

Desi Cooking & Travel Vlogs
42 views • 1 week ago

4:02



Aatma Ni Odakh | Malhar Thakar | Santvani Trivedi |...

Crystal Colors Event Studio
218K views • 5 days ago

4:43



Remove duplicates from array in Javascript |...

InterviewNest
117K views • 3 years ago

12:07



Moojibaba & Jai Sahaja! – Shankara Karunakara

Mooji Mala Music
1.4M views • 1 year ago

5:47

**Let us take a closer look at
some of the popular features
that make UX of front-end
apps richer.**



Home



Explore



Subscriptions



Library



History

Computer graphics and Illustration

Level-up your passion for art & illustration with this series that takes you through how to bring this hobby to life in the virtual world



Turn sketches into vector logos: Digitizing drawings...

Teaching Tech

117K views • 3 years ago



Comic Book Illustration: Oriana Colors | Picking a...

How to Draw Comics . NET

2.3K views • 1 year ago



Blender Beginner Tutorial - Part 1

Blender Guru

8.5M views • 1 year ago



Beginning Graphic Design: Typography

GCFLearnFree.org

2.7M views • 4 years ago



Combining Photography & Watercolour Illustrations w...

Demas Rusli

2.9K views • 1 year ago

Small Biz & Side Hustles

Aspiring and existing small business owners: broaden your skills and knowledge across a variety of domains related to growing your business



How To Start A Business In Australia - The 8 Steps

Jamie Xuereb

66K views • 3 years ago



How to take quality photos for your business listing

Google Small Business

20K views • 3 years ago



COVID-19 - How to prepare your Small Business for...

Australian Small Business and ...

1.7K views • 1 year ago



How to Hire the Right People When You're Starti...

GaryVee TV

37K views • 3 years ago

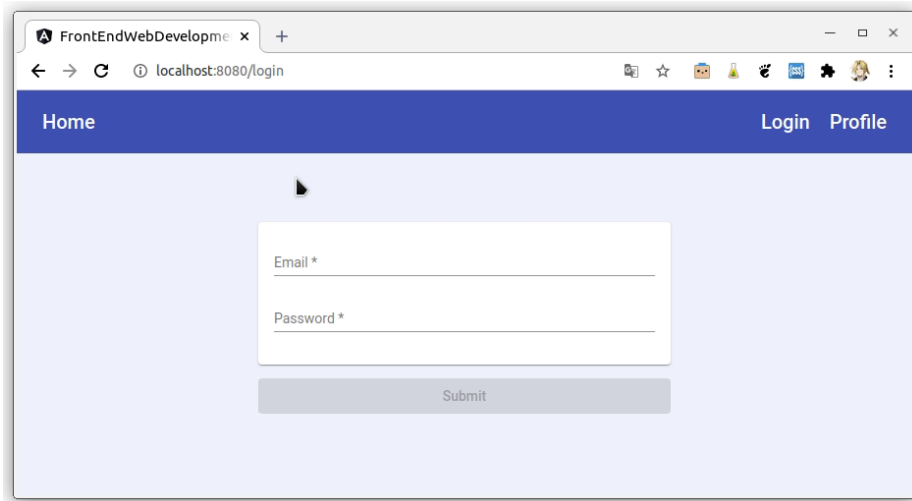


Legal Basics and Business Entity Formation: Crash...

CrashCourse

150K views • 1 year ago

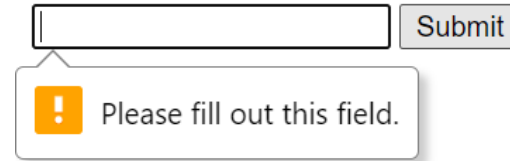




Source: <https://thomas-veillard.fr/wp-content/uploads/2021/02/submit-error-handler-in-tempalte.gif>

How are validations done smartly?

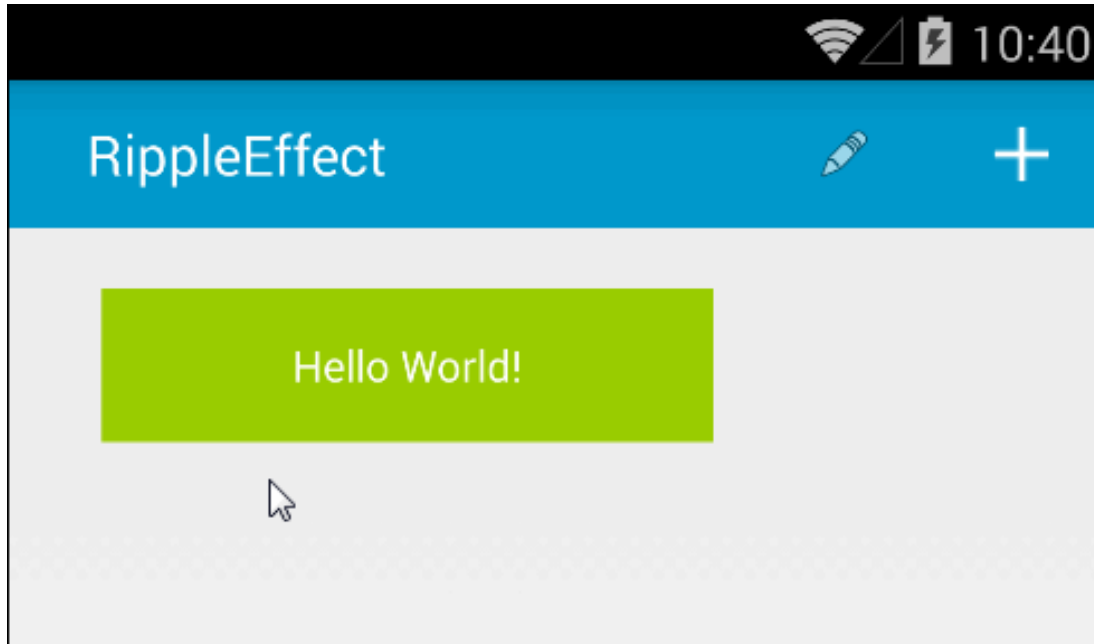
**Is the validation logic
rewritten every time it
is applied?**



A form with a text input field and a "Submit" button. Below the input field, a validation error message is displayed: "Please fill out this field." The message is enclosed in a white box with a yellow exclamation mark icon.

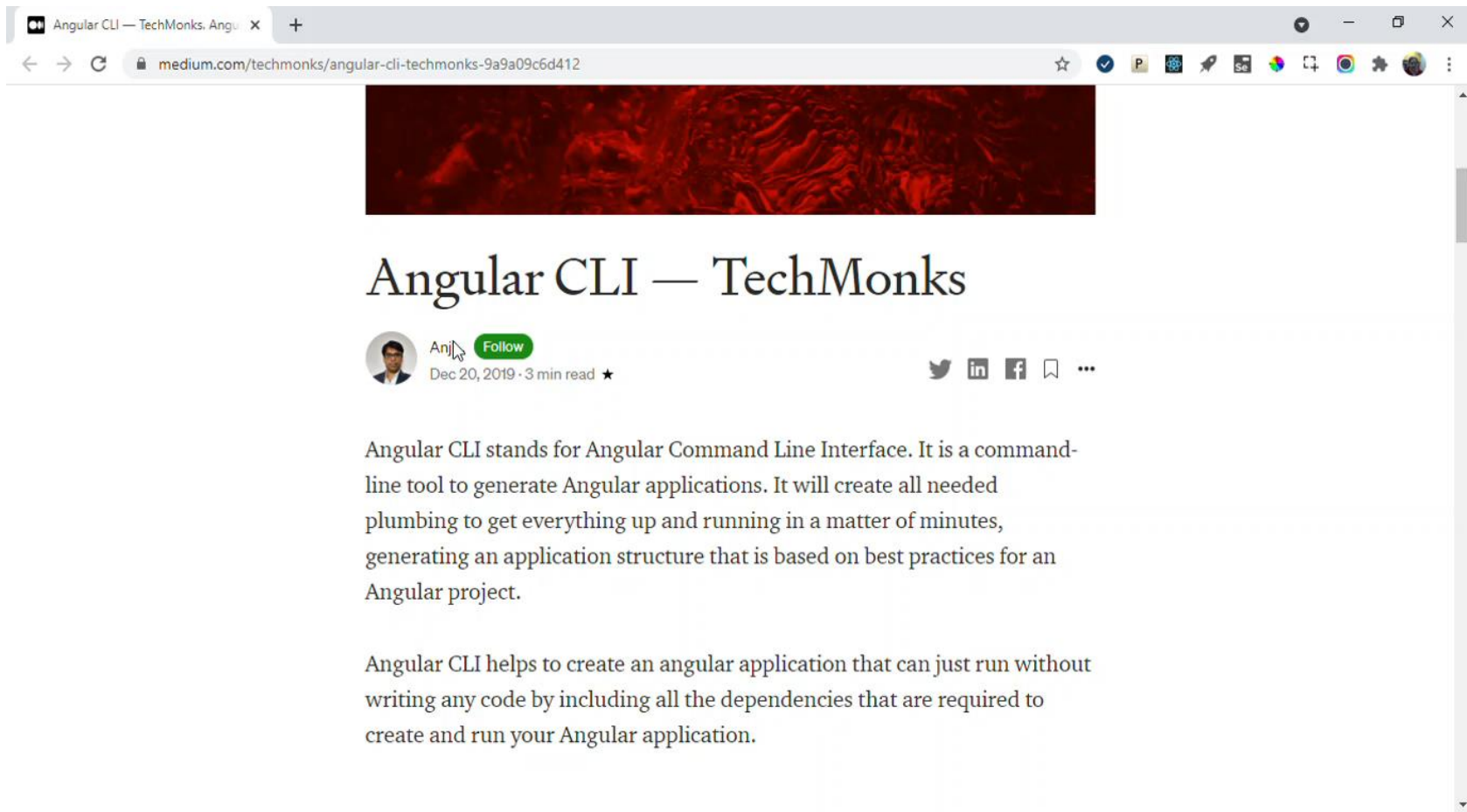
```
<form>  
  <input type="text" required=true>  
  
  <input type="submit">  
</form>
```

Adding Ripple Effect to Buttons



Can we add a ripple effect on multiple buttons by writing the logic only once?

**Have you ever noticed
how Medium.com
highlights text?**



The screenshot shows a web browser window with a single tab titled 'Angular CLI — TechMonks, Angu'. The address bar displays 'medium.com/techmonks/angular-cli-techmonks-9a9a09c6d412'. The browser's toolbar includes various extension icons. The article itself features a dark red, abstract, textured header image. Below the header, the title 'Angular CLI — TechMonks' is displayed in a large, black, serif font. The author's profile picture is a circular icon of a man with glasses. To the right of the profile picture is a green 'Follow' button. Below the profile picture, the text 'Dec 20, 2019 · 3 min read' is shown, followed by a star icon. To the right of the text are social media sharing icons for Twitter, LinkedIn, Facebook, and a bookmark icon, along with a three-dot menu icon. The main body of the article contains two paragraphs of text.

Angular CLI — TechMonks

Anj Follow
Dec 20, 2019 · 3 min read ★

Angular CLI stands for Angular Command Line Interface. It is a command-line tool to generate Angular applications. It will create all needed plumbing to get everything up and running in a matter of minutes, generating an application structure that is based on best practices for an Angular project.

Angular CLI helps to create an angular application that can just run without writing any code by including all the dependencies that are required to create and run your Angular application.

**Do you wish to create and apply
the same visual effects across
the different components in your
Angular app?**

Perform DOM Manipulation Using Angular Directives



Learning Objectives

- Perform DOM manipulation using Structural Directives
- Perform DOM manipulation using Attribute Directives



Can you recall the techniques that were used for performing visual activities in your Angular apps?

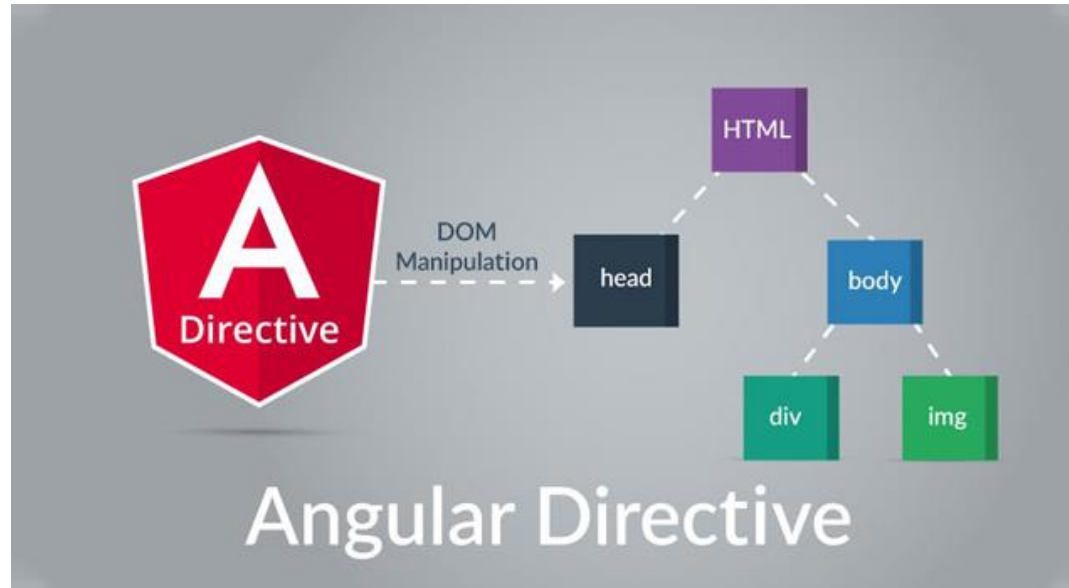
Recalling the Visual Activities

- How were the views controlled?
 - By components
- How were the contents on views rendered dynamically?
 - By `*ngFor`, `*ngIf`
- How were the template elements styled?
 - By `ngClass`, `ngStyle`



**Component, *ngFor, *ngIf,
ngClass, and ngStyle
are the directives that helped in
controlling the visual activities.**

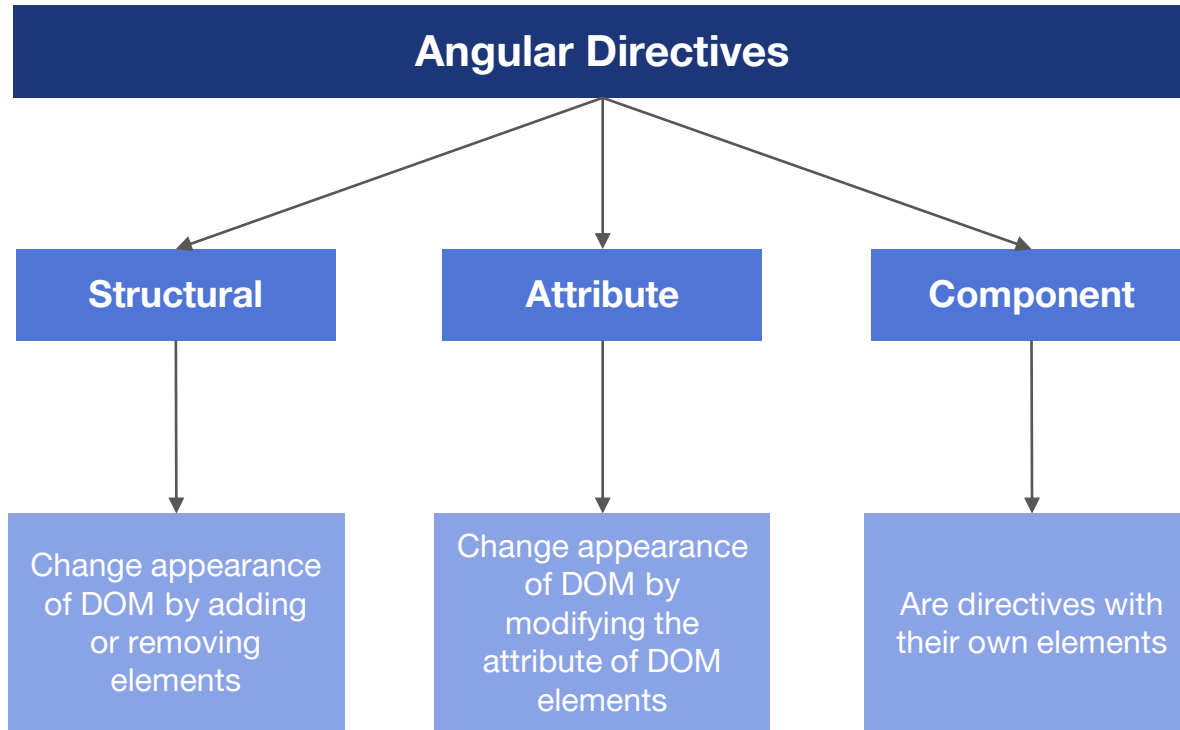
Role of Directives



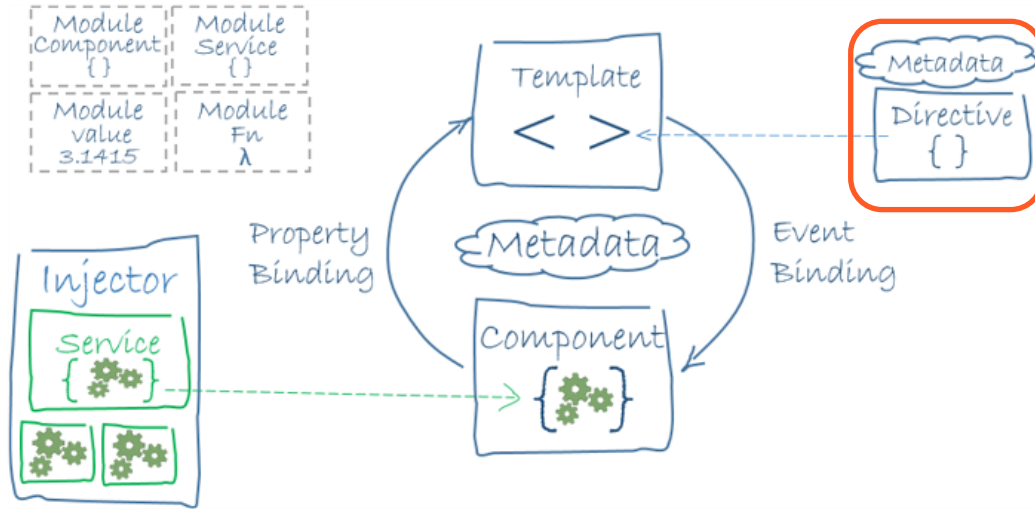
Source: https://miro.medium.com/max/640/1*G6-wF9ae7HiusRoTloCs-Q.png

Angular directives enhance the capability of the HTML elements by attaching custom behaviors to DOM.

Types of Directives



What Are Directives?



Source: <https://angular.io/generated/images/guide/architecture/overview2.png>

Directives are the classes that add additional behavior to the elements in your Angular applications.

- Directives are declarables.
- They must be declared by a NgModule to be usable in an app.

To define a directive, mark the class with the decorator and provide metadata.

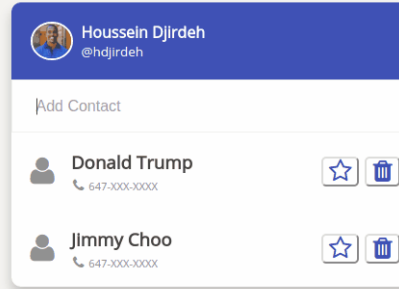
Need for Directives

- Angular templates are dynamic.
- When Angular renders templates, it transforms the DOM according to the instructions given by directives.

Need for Custom Directives

- Why do we create directives when we already have the built-in ones?
- Custom directives help to build reusable DOM manipulation logic.
- The reusable logic further helps in implementing DOM manipulation consistently across multiple components.

Role of Structural Directives in DOM Manipulation



Source: <https://creativetimblog.com/blog/wp-content/uploads/2020/03/contact-list-angular-components.gif>

- Structural directives are responsible for the HTML layout.
- They shape or reshape the DOM's structure, typically by adding, removing, and manipulating the host elements to which they are attached.

Role of Structural Directives in DOM Manipulation (contd.)

Source: <https://i0.wp.com/bearnithi.com/wp-content/uploads/2019/07/ezgif.com-video-to-gif-1.gif?resize=759%2C496>

- Structural directives are responsible for HTML layout.
- They shape or reshape the DOM's structure, typically by adding, removing, and manipulating the host elements to which they are attached.

Custom Structural Directives

Create a Custom Structural Directive - `*check`

Create a custom structural directive that functions like the built-in directive `*ngIf`.

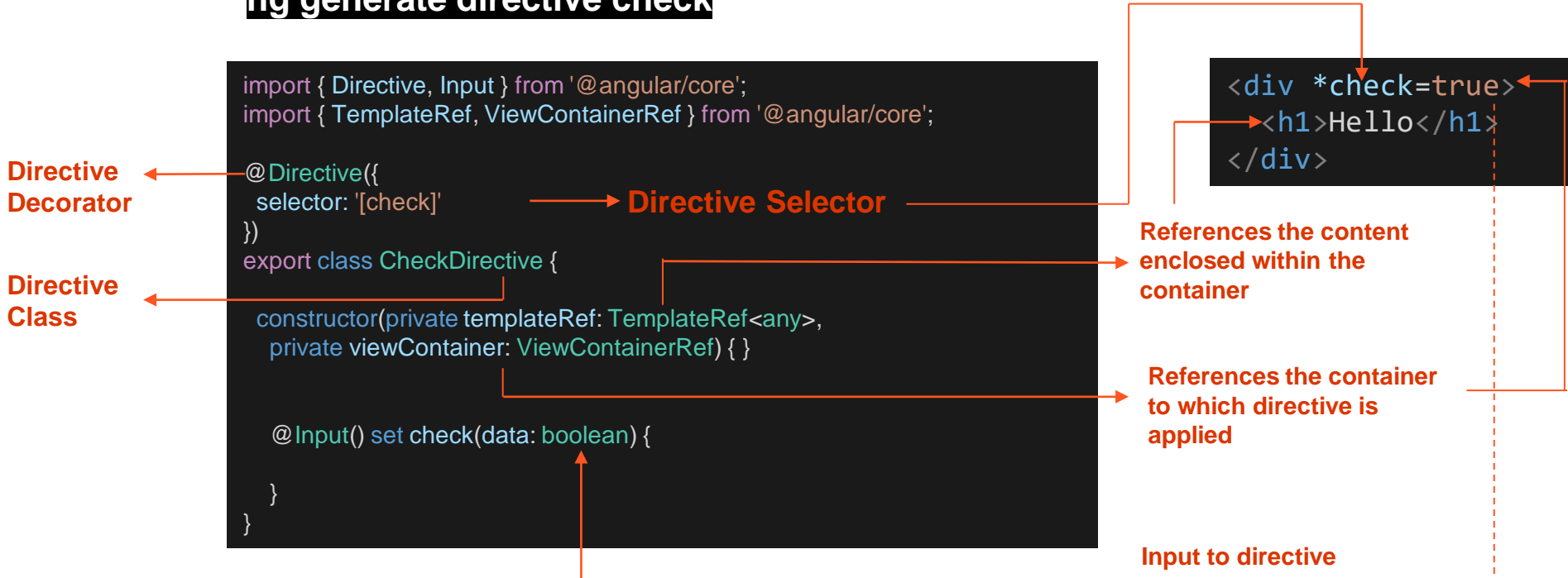
DEMO



Understanding Custom Structural Directive

- Directive is created by running the Angular CLI command

ng generate directive check



How Does a Structural Directive Work?

- The * notation in structural directive tells Angular that we have a structural directive and we will be manipulating the DOM.
- Behind the scenes, Angular converts every structural directive to ng-template syntax.
- The ng-template is a bunch of HTML tags enclosed in an HTML element <ng-template>.

```
<div *check=true>
  <h1>Hello</h1>
</div>
```

is equivalent to

```
<ng-template [check]=true>
  <div>
    <h1>Hello</h1>
  </div>
</ng-template>
```

How Does a Structural Directive Work? (contd.)

- The structural directive tells Angular to inject the `TemplateRef`.
 - `TemplateRef` is a class and a way to reference the `ng-template` in the component or directive class.
 - We can manipulate the template from the component code using `TemplateRef`.
- The template is inserted into the DOM when the condition is true by calling the method `createEmbeddedView()` of `ViewContainerRef`.
- The `clear()` removes the template from the DOM.
- The `ViewContainerRef` contains the reference to the host element that hosts the directive.

Quick Check

Is the following code valid for applying custom structural directive 'delay' on the div element?
`<div delay = 1000 >.... </delay>`

1. Yes
2. No



Quick Check: Solution

Is the following code valid for applying custom structural directive 'delay' on the div element?
`<div delay = 1000 >.... </delay>`

1. Yes
2. **No**

Explanation: Structural directives are applied using * notation

`<div *delay = 1000 >.... </delay>`



Quick Check

Identify the elements referenced by TemplateRef and ViewContainerRef in the following code.

```
<div *repeat="let item of itemList">

  <div>

    <div>{{item.name}}</div>

  </div>

</div>
```



Quick Check: Solution

Identify the elements referenced by TemplateRef and ViewContainerRef in the following code.

Referenced by
ViewContainerRef

Referenced by
TemplateRef

```
<div *repeat="let item of itemList">
  <div>
    <div>{{item.name}}</div>
  </div>
</div>
```



Attribute Directives

Role of Attribute Directives in DOM Manipulation

I love Angular

Attribute directives listen to and modify the behavior of other HTML elements, attributes, properties, and components.

Custom Attribute Directives

Creating Custom Attribute Directive - Focus

Create a custom attribute directive that changes the background color of the input element when it receives focus. The background color resets when the focus leaves the element.

DEMO



Understanding Custom Structural Directive

- The focus directive is created by running the Angular CLI command.

ng generate directive focus

Directive Decorator

Directive Class

```

import { Directive, Input } from '@angular/core';
import { HostListener, HostBinding } from '@angular/core';

@Directive({
  selector: '[appFocus]'
})
export class FocusDirective {

  @Input('appFocus') color: string="";
  @HostBinding('style.backgroundColor')
  bgColor: string = "";

  @HostListener('focus') onFocus(){
    this.bgColor = this.color;
  }

  @HostListener('blur') onBlur(){
    this.bgColor = "";
  }
}

```

Directive Selector

Binds bgColor with background-color style property

Listens to focus event and calls onFocus() event handler

Listens to blur event and calls onBlur() event handler

<input appFocus = "green" type="text">

@HostBinding and @HostListener

@HostBinding

- @HostBinding decorator allows directives to communicate with their host element by setting the properties of the host element.
- This decorator is used to bind a DOM property of the element to an instance variable in our custom Angular directive.

@HostListener

- @HostListener decorator allows interaction with the events on the host element.
- This decorator allows a directive to listen to events on its host element.
- When an event triggers, the directive calls the event handler decorated with @HostListener and modifies the value of the instance variable of the directive which further modifies the style property of the DOM element it is bound with.

Using ElementRef in Custom Attribute Directive

```
import { Directive, Input } from '@angular/core';
import { HostListener, ElementRef } from '@angular/core';
@Directive({
  selector: '[appFocus]'
})

export class FocusDirective {

  @Input('appFocus') color: string="";

  constructor(private elementRef : ElementRef) { }

  @HostListener('focus') onFocus(){
    this.elementRef.nativeElement.style.backgroundColor = this.color;
  }

  @HostListener('blur') onBlur(){
    this.elementRef.nativeElement.style.backgroundColor = null;
  }
}
```

ElementRef grants direct access to the host DOM element through its nativeElement property.

- It is injected through the constructor of the directive class.
- Use this API as the last resort when direct access to the DOM is needed.

Using Renderer2 in Custom Attribute Directive

```
import { Directive, Input } from '@angular/core';
import { HostListener, ElementRef, Renderer2 } from '@angular/core';
@Directive({
  selector: '[appFocus]'
})

export class FocusDirective {

  @Input('appFocus') color: string="";

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

  @HostListener('focus') onFocus(){
    this.renderer.setStyle(this.elementRef.nativeElement,'background-color',this.color);
  }

  @HostListener('blur') onBlur(){
    this.renderer.setStyle(this.elementRef.nativeElement,'background-color', null);
  }

}
```

- The Renderer2 class is an abstraction provided by Angular in the form of service.
- This service allows manipulating elements of the app without having to touch the DOM directly.
- This approach makes it easier to develop apps that can be rendered in environments that don't have DOM access.

Quick Check

ElementRef grants direct access to the host DOM element through its _____ property.

1. native
2. element
3. nativeElement
4. node



Quick Check: Solution

ElementRef grants direct access to the host DOM element through its _____ property.

1. native
2. element
3. **nativeElement**
4. node



Quick Check

Which Angular module is used to import the `@HostBinding` and `@HostListener` decorators?

1. common
2. cli
3. core
4. compiler



Quick Check: Solution

Which Angular module is used to import the `@HostBinding` and `@HostListener` decorators?

1. common
2. cli
3. **core**
4. compiler





Thank you!