■ College Answer Generator

Comprehensive Answers for Computer Science

Subject:	Computer Science
Answer Mode:	Exam Mode
Total Questions:	1
Generated On:	August 05, 2025 at 02:49 PM

Concise, exam-focused answers with key points and formal academic language.

Question 1

Q1: What are Angular Directives? Explain with types and examples.

Answer:

Angular Directives are a fundamental building block in the Angular framework, serving as special markers on a **DOM element** that tell Angular to attach a specific behavior to that element or to transform the DOM. Essentially, they extend HTML with new syntax and functionality, allowing developers to create **declarative templates** that dynamically modify the structure, appearance, or behavior of HTML elements based on application logic. All components in Angular are, in fact, directives, but with an associated template.

Directives operate by interacting with the *host element* they are applied to, enabling powerful DOM manipulation without direct imperative JavaScript calls, thereby promoting a more declarative and maintainable codebase. They are crucial for implementing reusable UI logic and managing element-level interactions.

Angular categorizes directives into three primary **types:**

```
1. **Component Directives:** * **Definition:** These are the most
common type of directives and are essentially directives *with a
template*. Every Angular application is built as a tree of
components. They are marked with the `@Component` decorator,
which extends the `@Directive` decorator with template-specific
options (like `templateUrl` or `template`). * **Purpose:** To
define a self-contained, reusable UI block with its own view
(template), logic (class), and styles. * **Example:** A
`ProductCardComponent` or `UserListComponent`. * **Conceptual
Code:** typescript import { Component } from '@angular/core';
```

```
@Component({ selector: 'app-product-card', // A directive
selector template: `... Product details ...`, styleUrls:
['./product-card.component.css'] }) export class
ProductCardComponent { // Component logic } * **Role:** While
technically directives, their primary role in Angular
architecture is to build the UI hierarchy.
```

```
2. **Structural Directives:** * **Definition:** These directives
are responsible for **manipulating the DOM layout** by adding or
removing elements from the DOM, or by creating and rendering
elements based on some condition or iteration. They are always
prefixed with an asterisk (`*`) in the template syntax, which is
*syntactic sugar* for a `` wrapper. * **Purpose:** To change the
structure of the view. * **Key Feature:** They directly interact
with the *template reference* (`TemplateRef`) and *view
container* (`ViewContainerRef`) to instantiate or destroy
embedded views. * **Examples:** * `*ngIf`: Conditionally adds or
removes an element from the DOM. html Content shown only if
isValid is true * `*ngFor`: Repeats a block of HTML for each item
in a collection. html {{ i }}: {{ item.name }} * `*ngSwitch`:
Renders one of several possible elements based on a match. html
Loading... Data loaded! Unknown status
```

```
3. **Attribute Directives:** * **Definition:** These directives
change the appearance or behavior of a DOM element, component, or
another directive without altering the DOM structure. They are
applied as attributes to elements. * **Purpose:** To modify
properties, styles, or classes of an element, or to add event
listeners. They are marked with the `@Directive` decorator. *
**Key Feature:** They often inject `ElementRef` to access the
underlying DOM element and `Renderer2` to safely manipulate the
DOM across different rendering environments (e.g., browser,
server-side rendering). * **Examples:** * `ngClass`: Adds or
removes CSS classes conditionally. html This text changes class.
* `ngStyle`: Sets inline CSS styles conditionally. html Stylized
content. * **Custom Attribute Directive (Conceptual):** To create
a custom attribute directive, you use the `@Directive` decorator.
typescript import { Directive, ElementRef, HostListener, Input,
Renderer2 } from '@angular/core';
```

```
@Directive({ selector: '[appHighlight]' // Applied as an
attribute: }) export class HighlightDirective {
@Input('appHighlight') highlightColor: string = 'yellow';
```

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

```
@HostListener('mouseenter') onMouseEnter() {
this.renderer.setStyle(this.el.nativeElement,
'background-color', this.highlightColor); }
```

```
@HostListener('mouseleave') onMouseLeave() {
this.renderer.removeStyle(this.el.nativeElement,
'background-color'); } } This `appHighlight` directive would
change the background color of the host element on mouse
enter/leave. The `ElementRef` provides direct access to the host
```

```
element, while `Renderer2` provides a cross-platform way to
modify its properties, crucial for avoiding direct DOM
manipulation in server-side rendering or web workers.
```

In summary, Angular directives provide a powerful, declarative mechanism for extending HTML and manipulating the DOM. Component directives define UI blocks, structural directives alter DOM structure, and attribute directives modify element appearance or behavior, all contributing to a modular, reusable, and maintainable application architecture.