Angular Interview Example

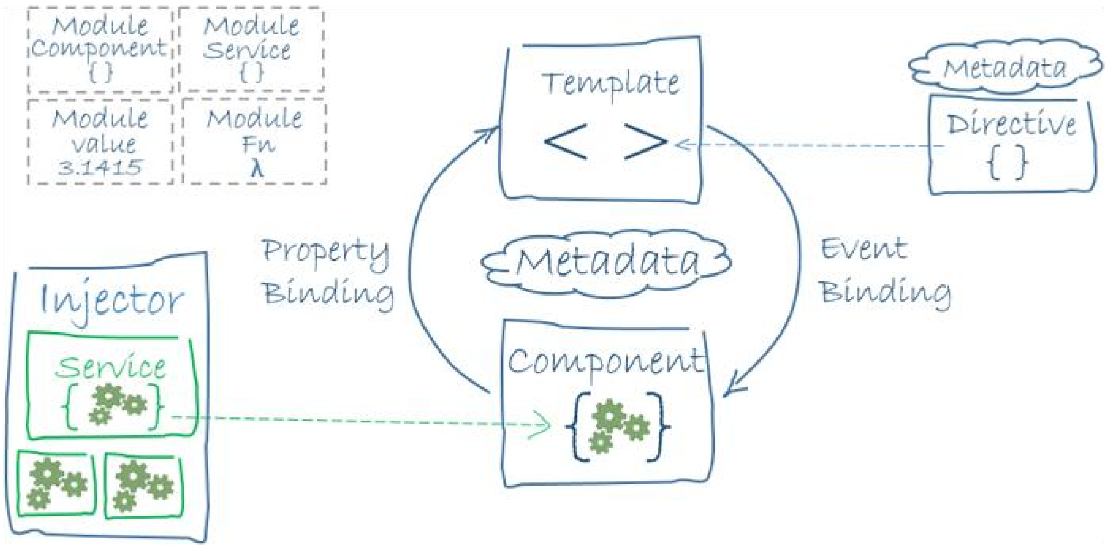
Q: What is the DOM?
A: DOM stands for Document Object Model.
It's basically a **tree-like structure** that browsers create to represent your HTML page.
so that **JavaScript** (and Angular, React, etc.) can **access**, **modify**, and **update** it.

Example:-
Now, with **JavaScript/Jquery** or Angular, you can do things like:
document.querySelector('h1').textContent = 'Hello from DOM!';

Q: What is the difference between AngularJS and Angular?
A: AngularJS (version 1.x) is a JavaScript framework. whereas Angular (version 2+) is a complete **rewrite of AngularJS using TypeScript.**

AngularJS	Angular
It is based on **MVC architecture**	** This is based on **Service/Controller**
It uses **JavaScript to build the application**	Uses **TypeScript to build the application**
Based on **controllers concept**	** This is a **component based** UI approach

Q: Write a pictorial diagram of Angular architecture?
The main building blocks of an Angular application are shown in the diagram below:-



Q:- What are the key compoenet of angualr?
A:
1. Modules (NgModules)
   Organize your application into cohesive blocks.
   Every Angular app has at least one root module (AppModule).
   Can import/export other modules.
   Declares components, directives, and pipes.

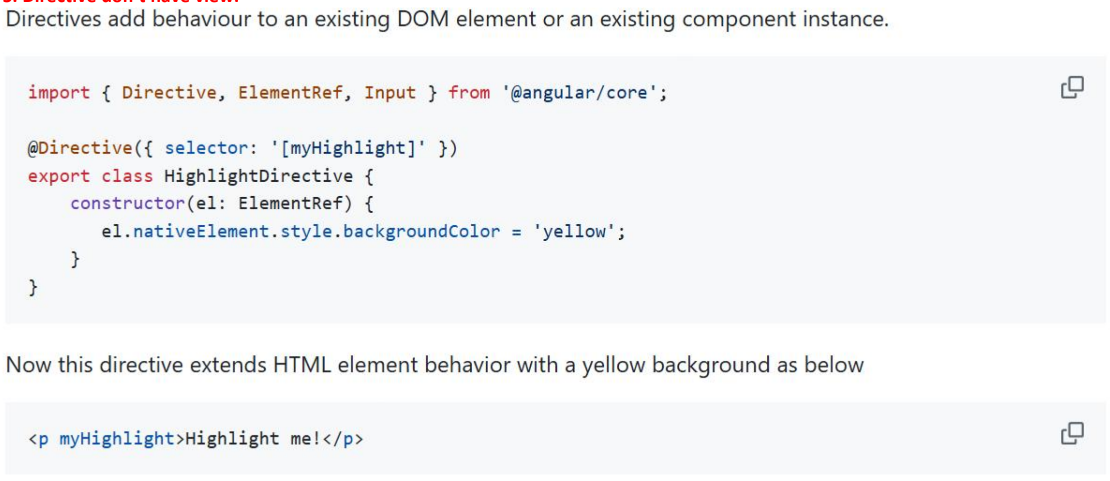
2. Components
   The core UI building blocks of Angular. [**templateUrl, selector, styleUrls**]

```
import { Component } from '@angular/core';

@Component ({
   selector: 'my-app',
   template: ` <div>
      <h1>{{title}}</h1>
      <div>Learn Angular6 with examples</div>
   </div> `,
})

export class AppComponent {
   title: string = 'Welcome to Angular world';
}
```

3. Templates
   HTML + Angular syntax.
   Used to define the view for a component.
   Supports binding and directives.

Angular Interview Example

4. Directives
    1. **Instructions in the DOM or change the behaviour.**
    2. **Structural directives: *ngIf, *ngFor (change DOM layout)**
    3. **Attribute directives: ngClass, ngStyle (change appearance/behavior)**
    4. **Many directive can be use per DOM element.**
    5. **Directive don't have view.**



Directives add behaviour to an existing DOM element or an existing component instance.

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

@Directive({ selector: '[myHighlight]' })
export class HighlightDirective {
    constructor(el: ElementRef) {
       el.nativeElement.style.backgroundColor = 'yellow';
    }
}
```

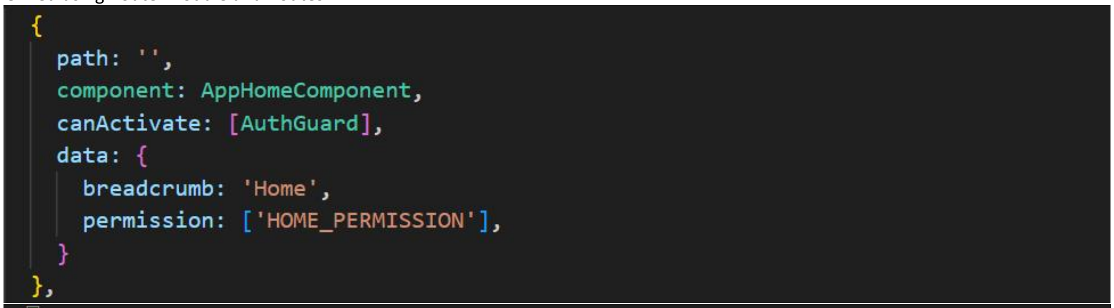
Now this directive extends HTML element behavior with a yellow background as below

```
<p myHighlight>Highlight me!</p>
```

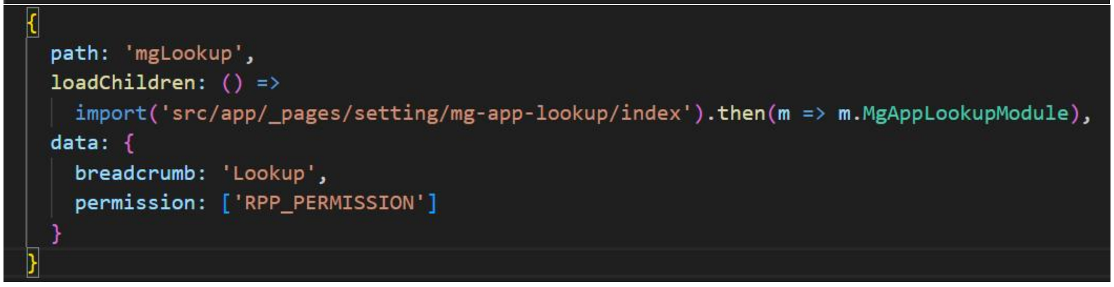
5. Services and Dependency Injection
   Services handle business logic, data access, etc.
   Angular's DI system allows injecting services into components or other services.

6. Pipes
   Pipes are simple functions that use template expressions to **accept data as input and transform it into a desired output**. .
   Examples: **date, uppercase, custom pipes.**

7. Routing
   Navigation system to switch between views/components.
   Defined using RouterModule and Routes.



```
{
  path: '',
  component: AppHomeComponent,
  canActivate: [AuthGuard],
  data: {
    breadcrumb: 'Home',
    permission: ['HOME_PERMISSION'],
  }
},
```



```
{
  path: 'mgLookup',
  loadChildren: () =>
    import('src/app/_pages/setting/mg-app-lookup/index').then(m => m.MgAppLookupModule),
  data: {
    breadcrumb: 'Lookup',
    permission: ['RPP_PERMISSION']
  }
}
```

8. Reactive Forms & Template-Driven Forms
   For building interactive forms with validation and data binding.

9. Observables (**RxJS**)
   Angular relies heavily on RxJS for reactive programming.
   Especially useful in HTTP requests and event handling.

9. HttpClient Module
   Used for communicating with backend APIs.

Angular Interview Example

Q: Can you give the stractue of Module?
A: In module mostly we use the arry of [**declarations , imports, providers, boostraps, export**]

```
@NgModule({
  declarations: [ /* Components, Directives, Pipes */ ],
  imports: [ /* Other Angular modules like CommonModule, FormsModule, etc. */ ],
  providers: [ /* Services and interceptors to be injected */ ],
  bootstrap: [ /* Root component to bootstrap (only in AppModule) */ ],
  exports: [ /* Components, Directives, Pipes you want to make available to other modules */ ]
})
export class SomeModule { }
```

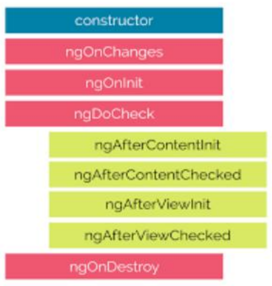
Q:- What are lifecycle hooks available?
A:- Angular application goes through an entire set of processes or has a lifecycle right from its initiation to the end of the application.

1. ngOnChanges: When the **value of a data bound property changes**, then this method is called.

2. ngOnInit:This is **called whenever the initialization of the directive/component** after Angular first displays the data-bound properties happens.

3. ngAfterViewInit: This is called in response after Angular initializes the component's views and child views.

4. ngOnDestroy: This is the **cleanup phase just before Angular destroys** the directive/component.



Q:- **Different b/w Promisise and Observable? [V.V IMP]**
A:- Here are the detai for Promise and Observable.

Feature	Promise	Observable
Emits	**One** value (single event)	**Multiple** values (stream of events)
Lazy vs Eager	**Eager** – executes immediately	**Lazy** – executes when subscribed
Cancellable	❌ Not cancellable	✅ Cancellable via `unsubscribe()`
Operators (map, filter)	❌ No built-in operators	✅ Rich set of operators via **RxJS**
Chaining	✅ With `.then()` and `.catch()`	✅ With RxJS operators (`.pipe()`)
Error Handling	`.catch()`	`.subscribe(error => ...)`
Multiple Subscribers	❌ No (unless wrapped again)	✅ Yes (hot/cold observables possible)
Use Case	One-time HTTP call	Streams, events, WebSockets, HTTP, etc.

Q:- What is ReJs can you explain about?. [V.V Important]
A: There are the list of concept [Creation, Transformation,Filering,Combination]

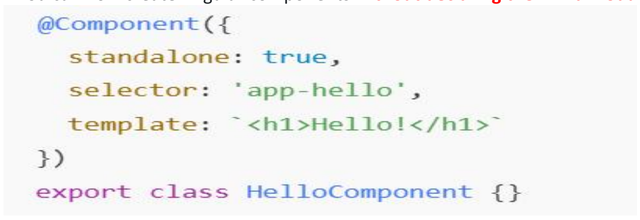
🔥 Concept	🎯 Purpose	🔧 Example Operator
Creation	Create observables	`of` , `from` , `interval`
Transformation	Change emitted values	`map` , `switchMap`
Filtering	Select specific values to emit	`filter` , `take` , `skip`
Combination	Merge or combine multiple observables	`combineLatest` , `merge`

Angular Interview Example

Q:- **What is a data binding?**
A:- Data binding is a core concept in Angular and allows defining communication between a component and the DOM, making it very easy to define interactive applications without worrying about pushing and pulling data.

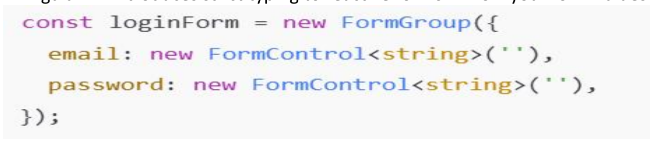
1. **From the Component to the DOM**:
   Interpolation: {{ value }}: Adds the value of a property from the component
   <li>Name: **{{ user.name }}**</li>
   <li>Address: {{ user.address }}</li>
2. **Property binding: [property]="value"**:
   The value is passed from the component to the specified property or simple HTML attribute
   <input type="email" **[value]="user.email"**>
3. **From the DOM to the Component**:
   Event binding: (event)="function": When a specific DOM event happens (eg.: click, change, keyup), call the specified method in the component
   <button **(click)="logout()"**></button>
4. **Two-way binding: Two-way data binding**:
   [(ngModel)]="value": Two-way data binding allows to have the data flow both ways. For example, in the below code snippet, both the email DOM input and component email property are in sync
   <input type="email" **[(ngModel)]="user.email"**>

Q: **What new feture in Angualr 14? [V.Imp Question]**
A:- Standalone, Typed Reactive Form, Inject() Function for DI
1. Standalone Components (Preview)
You can now create Angular components **without declaring them in a module**.



```
@Component({
  standalone: true,
  selector: 'app-hello',
  template: `<h1>Hello!</h1>`
})
export class HelloComponent {}
```

2. Inject() Function for DI
const http = inject(HttpClient);
No need constractor DI for service.
3. NgModel on Standalone FormControl
<input [formControl]="nameControl" [(ngModel)]="name">
4. Typed Reactive Forms
Angular 14 introduces strict typing to reactive forms — now your form values are type-safe! FormControl<**String**>



```
const loginForm = new FormGroup({
  email: new FormControl<string>(''),
  password: new FormControl<string>(''),
});
```

Q:- **What new in angluar 15? [Imp Question]**
1. Stable Standalone APIs: Fully module-free components & routing.
2. @Input({ required: true }): Required input enforcement at compile time.
3. Image Optimization with : New directive for image performance with built-in lazy loading and srcset

Q:- **What is redux and how its work? [Imp Question]**
A:- Redux is a JavaScript library used for managing the state of a web application, particularly when that state needs to be shared across multiple components or accessed globally.

Component ➡ dispatch(addUser({user}))
Effect listens to addUser ➡ calls API ➡ gets response
Effect dispatches addUserSuccess({newUser})
Reducer handles addUserSuccess ➡ adds newUser to the state

1. Demo Flow For Redcer With [Http Call]

```
Component
 |
 | dispatch()
 ↓
Action (e.g. loadUsers)
 |
 | (optional async API call)
 ↓
Effect (if any)
 |
 | → calls service (e.g. HTTP)
 ↓
Dispatch another Action (e.g. loadUsersSuccess)
 ↓
Reducer
 |
```

| → Updates Store (new state)
↓
Store
|
↓
Selector (in Component)

2.If no API call is needed (pure logic), you can **skip the Effect** and go:
Component ➡ Action ➡ Reducer ➡ Store

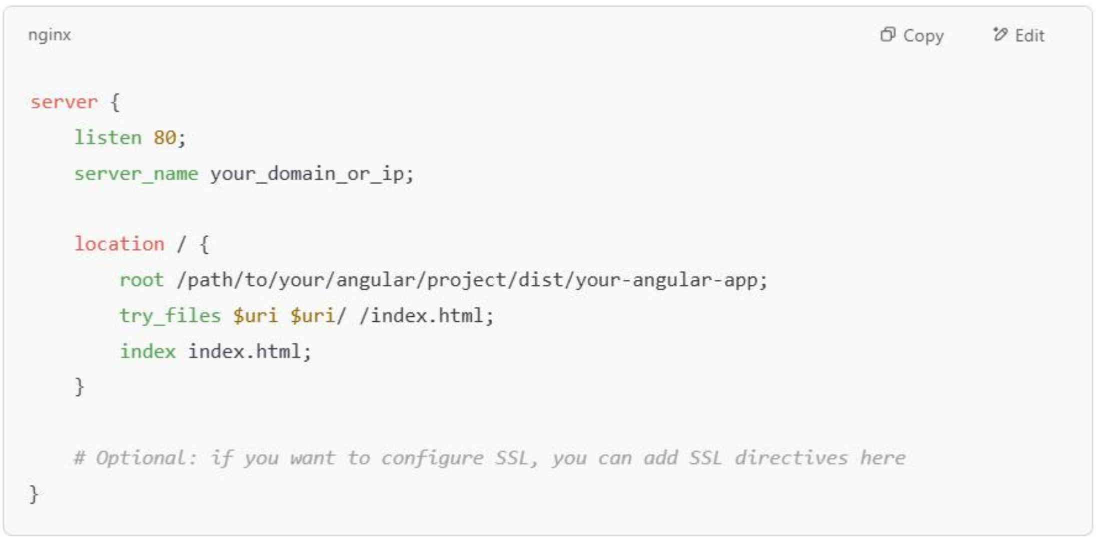
Q:- What are different types of compilation in Angular?
A:- Angular offers two ways to compile your application,
1. Just-in-Time (JIT) [Runtime]
2. Ahead-of-Time (AOT) [Compiler Time]

Q:- What is webpack?
A: Webpack is designed for **front-end JavaScript** projects, focusing on bundling and optimizing assets like JavaScript, CSS, and images.

Q:- what is the Style.Css.
A:- This is root level style which impact on whole application if we add want to add the root css.

Q:- How to deploy the application on server?
A:- we can deploy web application 3 way which is commonly use these days.
1) Deploy on S3
2) Deploy on Tomcat with Backend applicaito like MVC [add the index.html in web.xml] and few setting
3) Deploy on Ec2 instance usine NGINX server [docker or by menul both option work]

Q:- What is NGINX Server?
A: NGINX (pronounced "engine-x") is a high-performance web server that's also widely used as
1. Reverse proxy
2. Load balancer
1. HTTP cache
2. API gateway
Originally, it was designed to handle a large number of concurrent connections efficiently — which traditional web servers like Apache struggled with.



```nginx
server {
    listen 80;
    server_name your_domain_or_ip;

    location / {
        root /path/to/your/angular/project/dist/your-angular-app;
        try_files $uri $uri/ /index.html;
        index index.html;
    }

    # Optional: if you want to configure SSL, you can add SSL directives here
}
```

Q: New Build in control flow syntax [No More ngIf, ngFor]
A: Angular 17 introduce a new declarative control flow syntax using
@If, @for, and @switch.

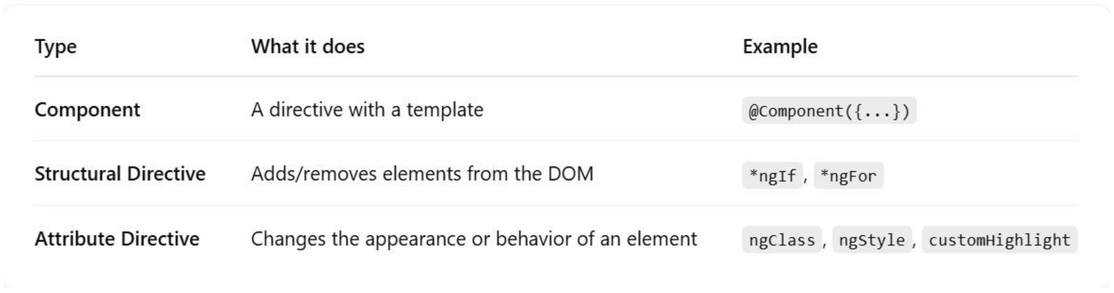
Q: Deferrable views for load the Lazy loading at the component level.
A: Lazy loading at the component template level! You can now defer part of your template rendering until a condition is met.
@defer (when isLoaded) {
<p>Data is loaded!</p>
} @placeholder {
<p>Loading...</p>
}

Q: What are signals?
A: A signal is a wrapper around a value that notifies interested consumers when that value changes.
Signals can contain any value, from primitives to complex data structures. **Signals may be either writable or read-only.**
const **count = signal(0)**; // Signals are getter **functions - calling** them reads their value.
console.log('The count is: ' + count());

Angular Interview Example

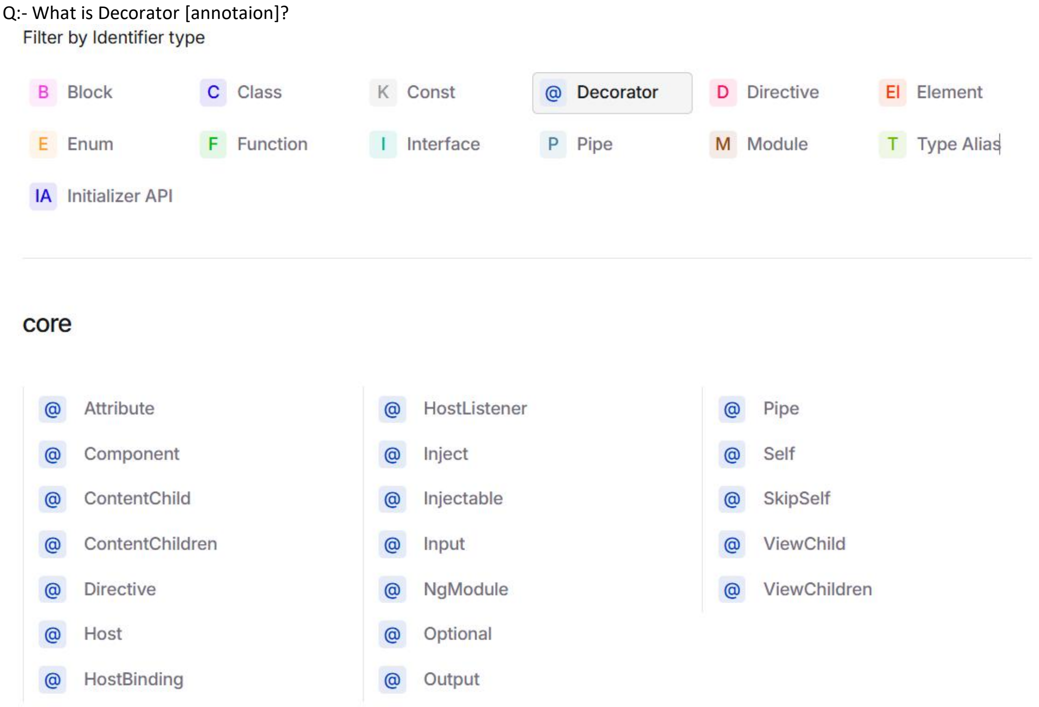
Q: **what is @ViewChild decorator? [V.V Imp]**
A: Accesses a child component or DOM element.

Q:- **Tyep of Directive? [V.V Imp]**



Type	What it does	Example
Component	A directive with a template	`@Component({...})`
Structural Directive	Adds/removes elements from the DOM	`*ngIf` , `*ngFor`
Attribute Directive	Changes the appearance or behavior of an element	`ngClass` , `ngStyle` , `customHighlight`

Q:- What is Decorator [annotaion]?



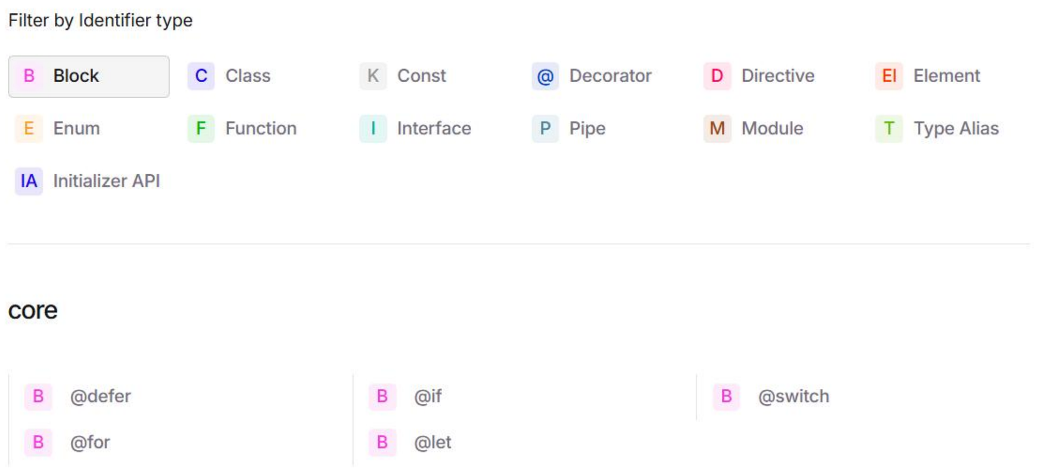
Filter by Identifier type

B Block	C Class	K Const	@ **Decorator**	D Directive	El Element
E Enum	F Function	I Interface	P Pipe	M Module	T Type Alias
IA Initializer API					

core

@ Attribute	@ HostListener	@ Pipe
@ Component	@ Inject	@ Self
@ ContentChild	@ Injectable	@ SkipSelf
@ ContentChildren	@ Input	@ ViewChild
@ Directive	@ NgModule	@ ViewChildren
@ Host	@ Optional	
@ HostBinding	@ Output	

Q:- What is Block?



Filter by Identifier type

B **Block**	C Class	K Const	@ Decorator	D Directive	El Element
E Enum	F Function	I Interface	P Pipe	M Module	T Type Alias
IA Initializer API					

core

B @defer	B @if	B @switch
B @for	B @let	