1. **Angular Lifecycle Hooks?**

|  |  |
| --- | --- |
| **Hook** | **Purpose and Timing** |
| ngOnChanges() | Respond when Angular (re)sets data-bound input properties. The method receives a [SimpleChanges](https://angular.io/api/core/SimpleChanges) object of current and previous property values.  Called before ngOnInit() and whenever one or more data-bound input properties change. |
| ngOnInit() | Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties.  Called *once*, after the *first* ngOnChanges(). |
| ngDoCheck() | Detect and act upon changes that Angular can't or won't detect on its own.  Called during every change detection run, immediately after ngOnChanges()and ngOnInit(). |
| ngAfterContentInit() | Respond after Angular projects external content into the component's view / the view that a directive is in.  Called *once* after the first ngDoCheck(). |
| [ngAfterContentChecked()](https://angular.io/api/core/AfterContentChecked#ngAfterContentChecked) | Respond after Angular checks the content projected into the directive/component.  Called after the ngAfterContentInit() and every subsequent ngDoCheck(). |
| ngAfterViewInit() | Respond after Angular initializes the component's views and child views / the view that a directive is in.  Called *once* after the first [ngAfterContentChecked()](https://angular.io/api/core/AfterContentChecked" \l "ngAfterContentChecked). |
| [ngAfterViewChecked()](https://angular.io/api/core/AfterViewChecked#ngAfterViewChecked) | Respond after Angular checks the component's views and child views / the view that a directive is in.  Called after the ngAfterViewInit and every subsequent [ngAfterContentChecked()](https://angular.io/api/core/AfterContentChecked" \l "ngAfterContentChecked). |
| ngOnDestroy() | Cleanup just before Angular destroys the directive/component. Unsubscribe Observables and detach event handlers to avoid memory leaks.  Called *just before* Angular destroys the directive/component. |

1. **HttpInterceptor**

Most interceptors will transform the outgoing request before passing it to the next interceptor in the chain, by calling next.handle(transformedReq).

interface HttpInterceptor {

intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>>

}

1. **Decorator**

**Class decorators**, e.g. **@Component**and *@NgModule*

**Property decorators** for properties inside classes, e.g. *@Input* and *@Output*

**Method decorators** for methods inside classes, e.g. *@HostListener*

**Parameter decorators** for parameters inside class constructors, *e.g. @Inject*

1. **Ahead-of-Time (AOT) compiler**

The Angular Ahead-of-Time (AOT) compiler converts your Angular HTML and TypeScript code into efficient JavaScript code during the build phase before the browser downloads and runs that code. Compiling your application during the build process provides a faster rendering in the browser.

Angular offers two ways to compile your application:

* **Just-in-Time (JIT),** which compiles your app in the browser at runtime.
* **Ahead-of-Time (AOT),** which compiles your app at build time.

JIT compilation is the default when you run the ng build (build only) or ng serve (build and serve locally) CLI commands:

**ng build**

**ng serve**

For AOT compilation, include the --aot option with the ng build or ng serve command:

**ng build --aot**

**ng serve –aot**

1. **Angular Lazy loading**

[**https://angular.io/guide/lazy-loading-ngmodules**](https://angular.io/guide/lazy-loading-ngmodules)

1. **Angular Storage?**

[**https://www.npmjs.com/package/angular-webstorage-service**](https://www.npmjs.com/package/angular-webstorage-service)