**Angular Developer Guide:**

***Standalone:*** it is a property in @component decorator, to tell application that this standalone component will declare its dependencies by its own instead of getting dependencies from ngModule and it can import another standalone component as dependencies using import array.

***Using ngModlue in Standalone component***:

If you want to access other components, directives or pipe some of these dependencies are not standalone component, but you can use it by import ngModule on standalone component.

***Using Standalone component in ngModule:***

You can import standalone component into ngModule by adding that in import array of ngModule.

***Bootstraping Angular Application:***

You can bootstrap an angular application using standalone component as a root component, by using function bootstrapApplication API we can achieve that.

***Configuring dependency Application***: To Bootstrap the application, we need to configure Angular’s Dependency Injection and provide configuration values and service to use throughout the application by **providers**

**ANGULAR APIs:**

***@angular/core***

***ngModule: Decorator***

*Ng-template:* ***Element****/****DIRECTIVE:*** It is an element/directive, that can refer a template in component, that will only render its content when any direct or indirectly instruct to do it.

It provides placeholder for content that can instantiate and rendered conditionally by other directives like ‘ngIf’, ‘ngFor’, ‘ngTemplateOutlet’.

*Ng-Conten****: Element/Directive****:* It is an element/directive, that is used to project the content in an particular place in template. It has ***select*** attribute to select particular selector from the template to show the content of it.

***@angular/forms***

*ControlValueAccessor*: It is an interface, acts like a bridge between angular forms API and native DOM element.

It has three method “writeValue”, “registerOnChange” and “registerOnTouch”

* writeValue set the value from parent component.
* registerOnChange is used to detect the changes in native DOM element to parent component.
* registerOnTouch is used to called when corresponding element get touched, mainly used for validation purpose.
* We should add *providers* and *implement* controlValueAccessor in the corresponding component, to above points to work correctly.

*When to use*

* When we use custom controls as component and that need to interact on angular forms at that time we need this controlValueAccessor
* Every native DOM element have their defaultValueAccessor, their value has been updated by formControl directives.

**Angular Component Lifecycle**:

***ngOnchanges***: Respond when angular set data bound input property changed. And this method receives ***simplechange*** object which has **currentValue** and **previousValue** property. This life cycle hook firstly called when component or directive .

***ngOnInit***: Called once, after first ngOnChanges(). Initialized angular component or directive after display data bound properties and set component or directive input propertires.

***ngDoCheck***: Called every time once change detection is run on every changes

***ngAfterContentInit***: Called when external content into the component’s view. Called once after ngDoCheck.

***ngAfterContentChanged***: Called when any change detection in content projected in angular component. Called every subsequent ngDoCheck().

***ngAfterViewInit***: Called after angular initialize the component’s view or its child views, or the view that contains directives.

***ngAfterViewChanged***: Called after angular checks the component’s view or its child view or view that contains directives. Called on every subsequent ngAfterContenChanged

***ngDestroy***: Cleanup just before angular destroy the directive or component. Unsubscribe observables and detach even handles to avoid memory leaks.