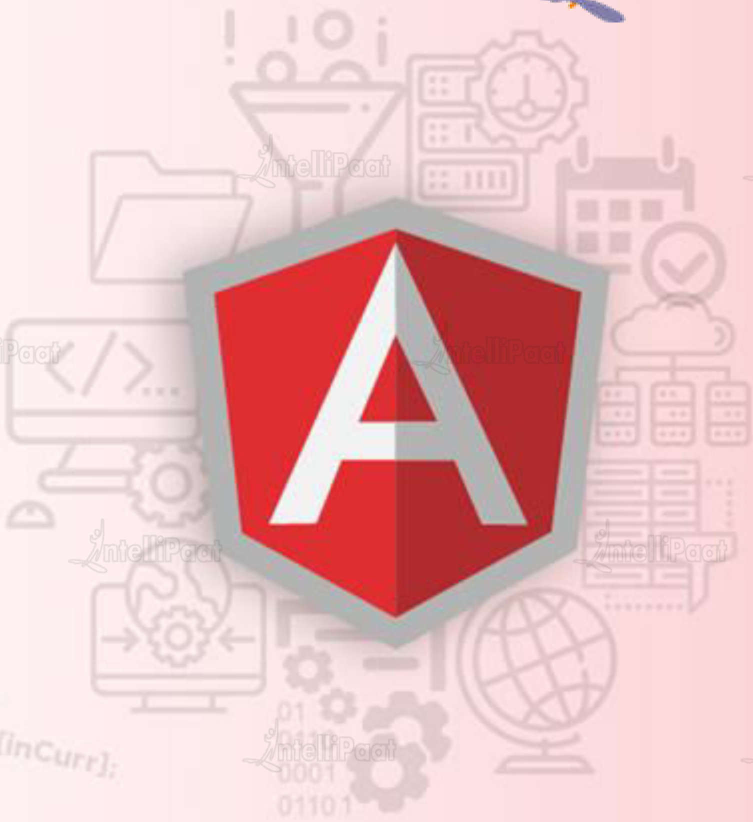
Component



Agenda

01 What are Components

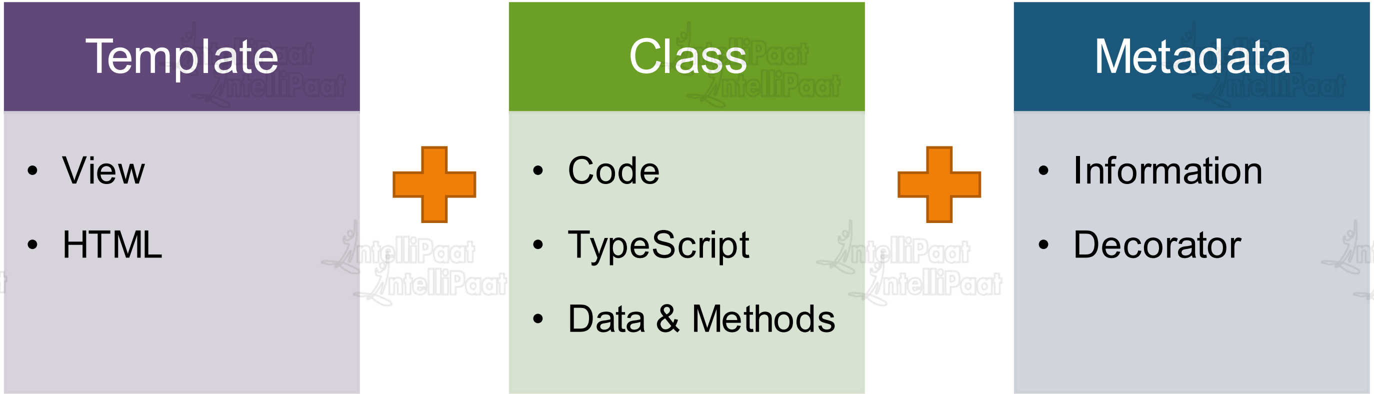
02 Nested Components

03 Component Interaction

04 Demo on Interaction

05 Component Lifecycle

Component



Template

- View
- HTML

Class

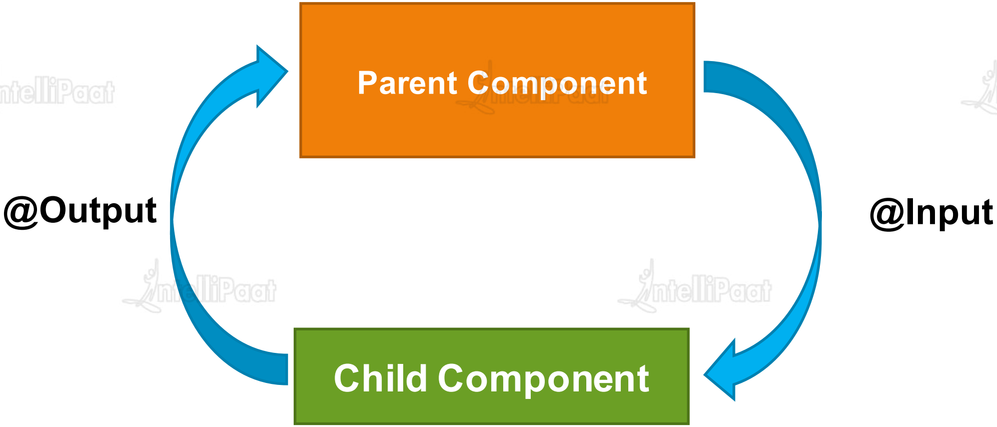
- Code
- TypeScript
- Data & Methods

Metadata

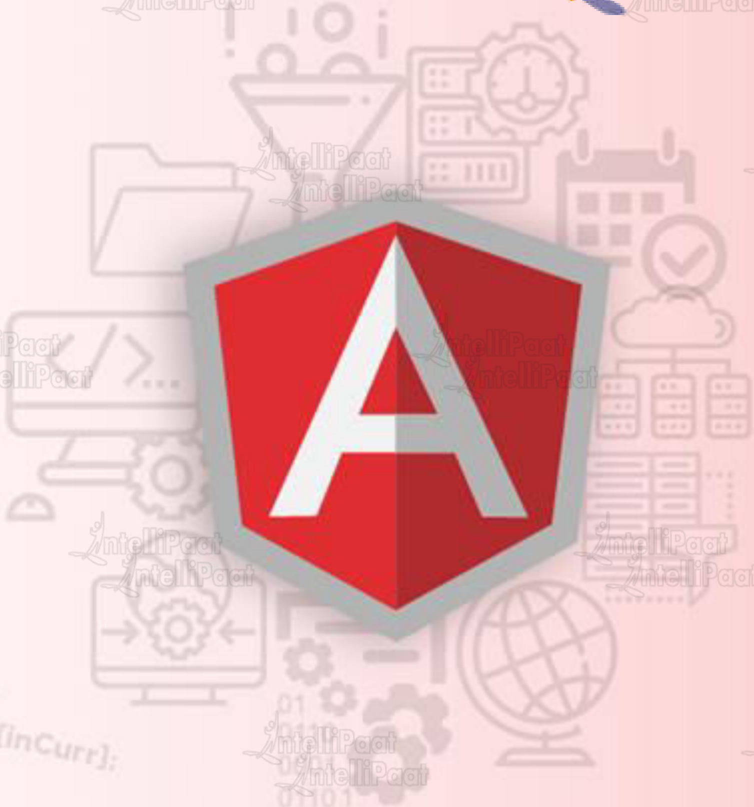
- Information
- Decorator

Component Interaction

Component Interaction

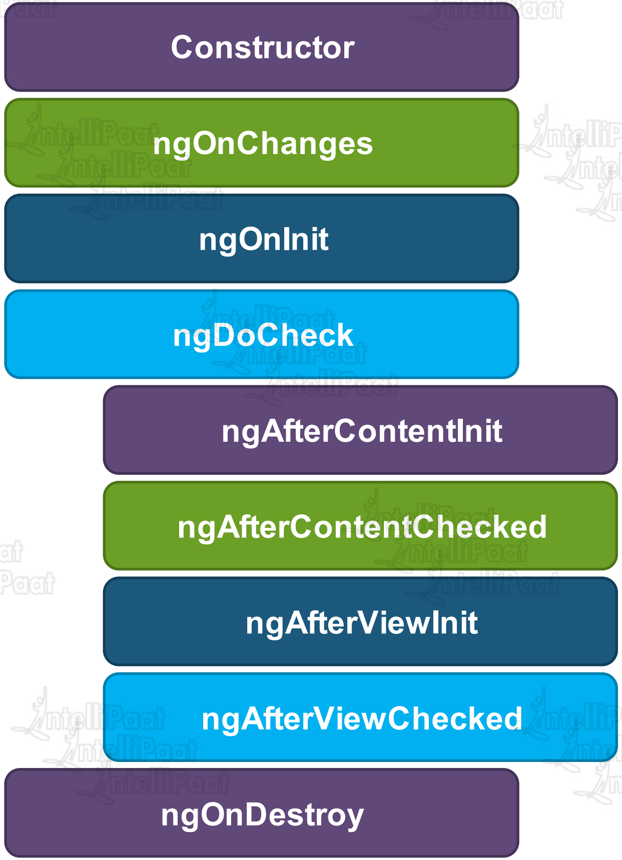


Component LifeCycle

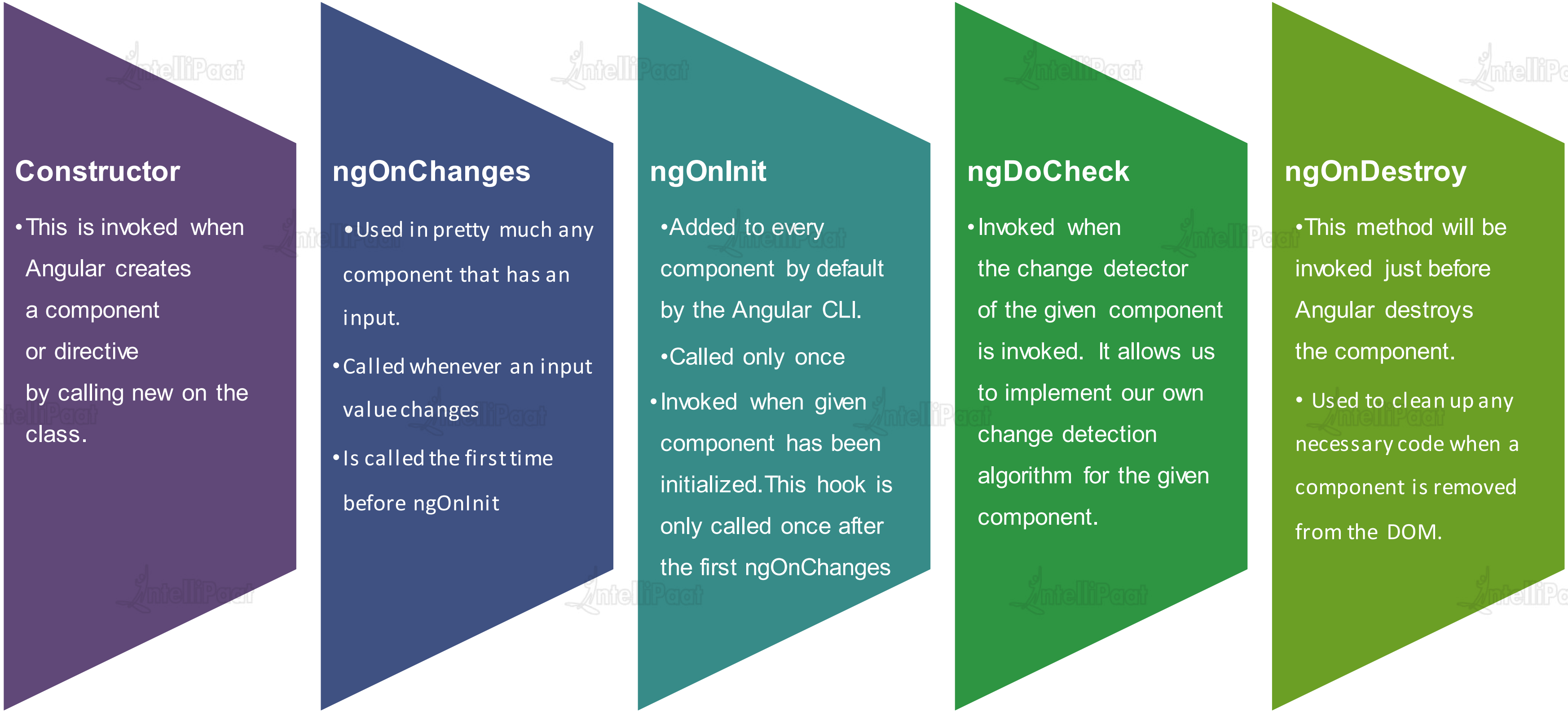


Component LifeCycle

- A component in Angular has a life-cycle, a number of different phases it goes through from birth to death and we call those methods *hooks*.
- The hooks are executed in this order.
- These phases are broadly split up into phases that are linked to the component itself and phases that are linked to the *children* of that component.
- Using life-cycle hooks we can fine tune the behavior of our components during creation, update and destruction.



Hooks for the Component



Constructor

- This is invoked when Angular creates a component or directive by calling new on the class.

ngOnChanges

- Used in pretty much any component that has an input.
- Called whenever an input value changes
- Is called the first time before ngOnInit

ngOnInit

- Added to every component by default by the Angular CLI.
- Called only once
- Invoked when given component has been initialized. This hook is only called once after the first ngOnChanges

ngDoCheck

- Invoked when the change detector of the given component is invoked. It allows us to implement our own change detection algorithm for the given component.

ngOnDestroy

- This method will be invoked just before Angular destroys the component.
- Used to clean up any necessary code when a component is removed from the DOM.

Hooks for the Component children

ngAfterContentInit

- Called only once after first ngDoCheck()
- Called after the first run through of initializing content

ngAfterContentChecked

- Called after every ngDoCheck()
- Waits till after ngAfterContentInit() on first run through

ngAfterViewInit

- Called after Angular initializes component and child component content.
- Called only once after view is fully initialized

ngAfterViewChecked

- Called after all the content is initialized and checked. (Component and child components).
- First call is after ngAfterViewInit()
- Called after every ngAfterContentChecked() call
- is completed

Demo

Summary

- Using life-cycle hooks we can fine tune the behaviour of our components during creation, update and destruction.
- We use the ngOnInit hook most often, this is where we place any initialisation logic for our component. It's preferred over initialising via the constructor since in the constructor we don't yet have access to the input properties whereas by the time ngOnInit is called they have been bound to and are available to use.
- ngOnChanges is the second most common hook, this is where we can find out details about which input properties have changed and how they have changed.
- The third most common hook is ngOnDestroy which is where we place any cleanup logic for our component.

☎ India : +91-7847955955

US : 1-800-216-8930 (TOLL FREE)

✉ sales@intellipaat.com

24X7 Chat with our Course Advisor