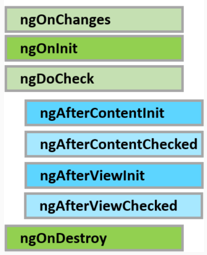
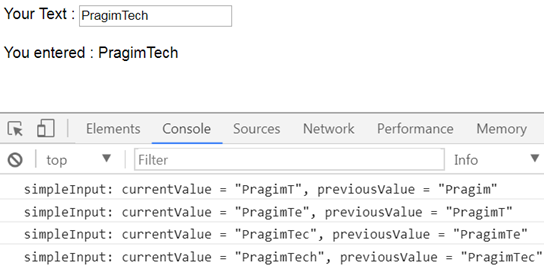
* + @Input and @Output

**A component has a lifecycle managed by Angular. Angular**

1. Creates the component
2. Renders the component
3. Creates and renders the component children
4. Checks when the component data-bound properties change, and
5. Destroys the component before removing it from the DOM
6. To tap into and react when these life cycle events occur, angular offers several lifecycle hooks as shown in the image below.    
      
     
   **The 3 most commonly used hooks are**

|  |  |
| --- | --- |
| **Life Cycle Hook** | **Purpose** |
| ngOnChanges | Executes, every time the value of an input property changes. The hook method receives a SimpleChanges object containing current and previous property values. This is called before ngOnInit |
| ngOnInit | Executes after the constructor and after ngOnChange hook for the first time. It is most commonly used for component initialisation and retrieving data from a database |
| ngOnDestroy | Executes just before angular destroys the component and generally used for performing cleanup |

1. **There are 3 simple steps to use the Life Cycle Hooks**  
   **Step 1 :** Import the Life Cycle Hook interface. For example, to use ngOnInit() life cycle hook, import OnInit interface.  
     
   import { OnInit } from '@angular/core';  
     
   **Step 2 :** Make the component class implement the Life Cycle Hook interface, using the implements keyword as shown below. This step is optional, but good to have so you will get editor support and flags errors at compile time if you incorrectly implement the interface method or make any typographical errors.  
     
   export class SimpleComponent implements OnInit { }  
     
   **Step 3 :** Write the implementation code for the life cycle interface method. Each interface has a single hook method whose name is the interface name prefixed with ng.  
     
   ngOnInit() {  
       console.log('OnInit Life Cycle Hook');  
   }  
     
   Let's understand ngOnChanges life cycle hook with a simple example. Here is what we want to do. As soon as the user starts typing into the text box, we want to capture the current and previous value and log it to the browser console as shown below. We can very easily achieve this by using the ngOnChanges life cycle hook.   
      
     
   ngOnChanges, is called every time the value of an input property of a component changes. So first let's create a SimpleComponent with an input property as shown below. We will continue with the example we worked with in our previous. Add a new folder in the App folder and name it Others. Add a new TypeScript file to this folder and name it - simple.component.ts. Copy and paste the following code which is commented and self explanatory.
2. // Step 1 : Import OnChanges and SimpleChanges
3. import { Component, Input, OnChanges, SimpleChanges } from '@angular/core';
4. // The selector "simple" will be used as the directive
5. // where we want to use this component. Notice we are
6. // also using the simpleInput property with interpolation
7. // to display the value it receives from the parent
8. // component
9. @Component({
10. selector: 'simple',
11. template: `You entered : {{simpleInput}}`
12. })
13. // Step 2 : Implement OnChanges Life Cycle Hook interface
14. export class SimpleComponent implements OnChanges {
15. // Input property. As and when this property changes
16. // ngOnChanges life cycle hook method is called
17. @Input() simpleInput: string;
18. // Step 3 : Implementation for the hook method
19. // This code logs the current and previous value
20. // to the console.
21. ngOnChanges(changes: SimpleChanges) {
22. for (let propertyName in changes) {
23. let change = changes[propertyName];
24. let current = JSON.stringify(change.currentValue);
25. let previous = JSON.stringify(change.previousValue);
26. console.log(propertyName + ': currentValue = '
27. + current + ', previousValue = ' + previous);
28. // The above line can be rewritten using
29. // placeholder syntax as shown below
30. // console.log(`${propertyName}: currentValue
31. // = ${current }, previousValue = ${previous }`);
32. }
33. }
34. }
35. Now copy and paste the following code in our root component - app.component.ts
36. import { Component } from '@angular/core';
37. // Notice we have placed the text box in this root component
38. // To keep the value in the textbox and the component property
39. // value "userText" in sync we are using 2 way data binding
40. // We have also bound userText property of this component
41. // to the input property of the SimpleComponent
42. @Component({
43. selector: 'my-app',
44. template: `Your Text : <input type='text' [(ngModel)]='userText'/>
45. <br/><br/>
46. <simple [simpleInput]='userText'></simple>
47. `
48. })
49. export class AppComponent {
50. userText: string = rashmi;
51. }
52. **Please note :** Do not forget to import and declare SimpleComponent in the root module - app.module.ts  
      
    At this point run the application and launch browser developer tools. As you start typing into the text box, the changes are logged to the console by the ngOnChanges life cycle hook method.  
      
    The steps for implementing the other component life cycle hook methods are very similar