

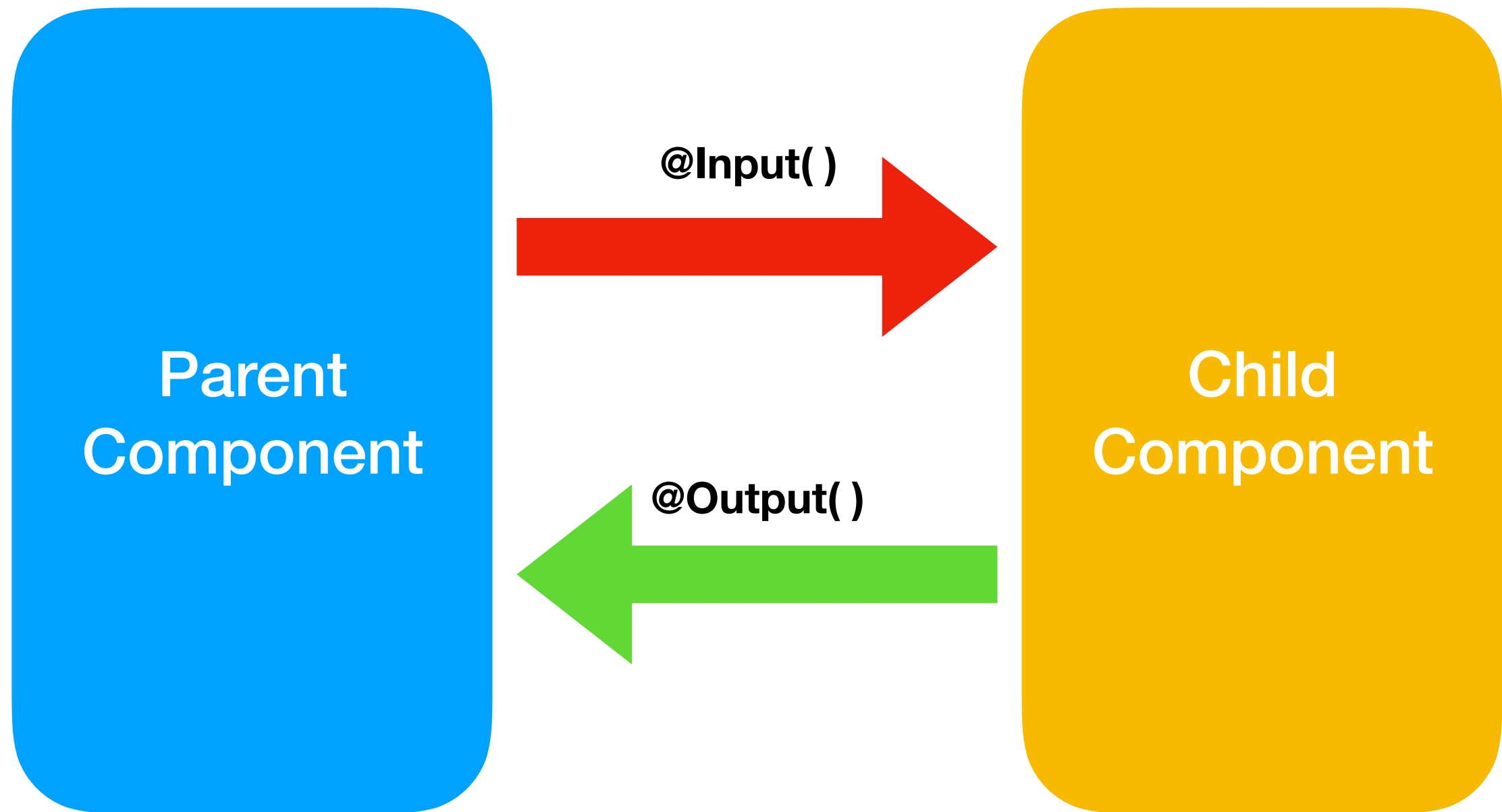
Component Communication & Lifecycle Hooks

523419

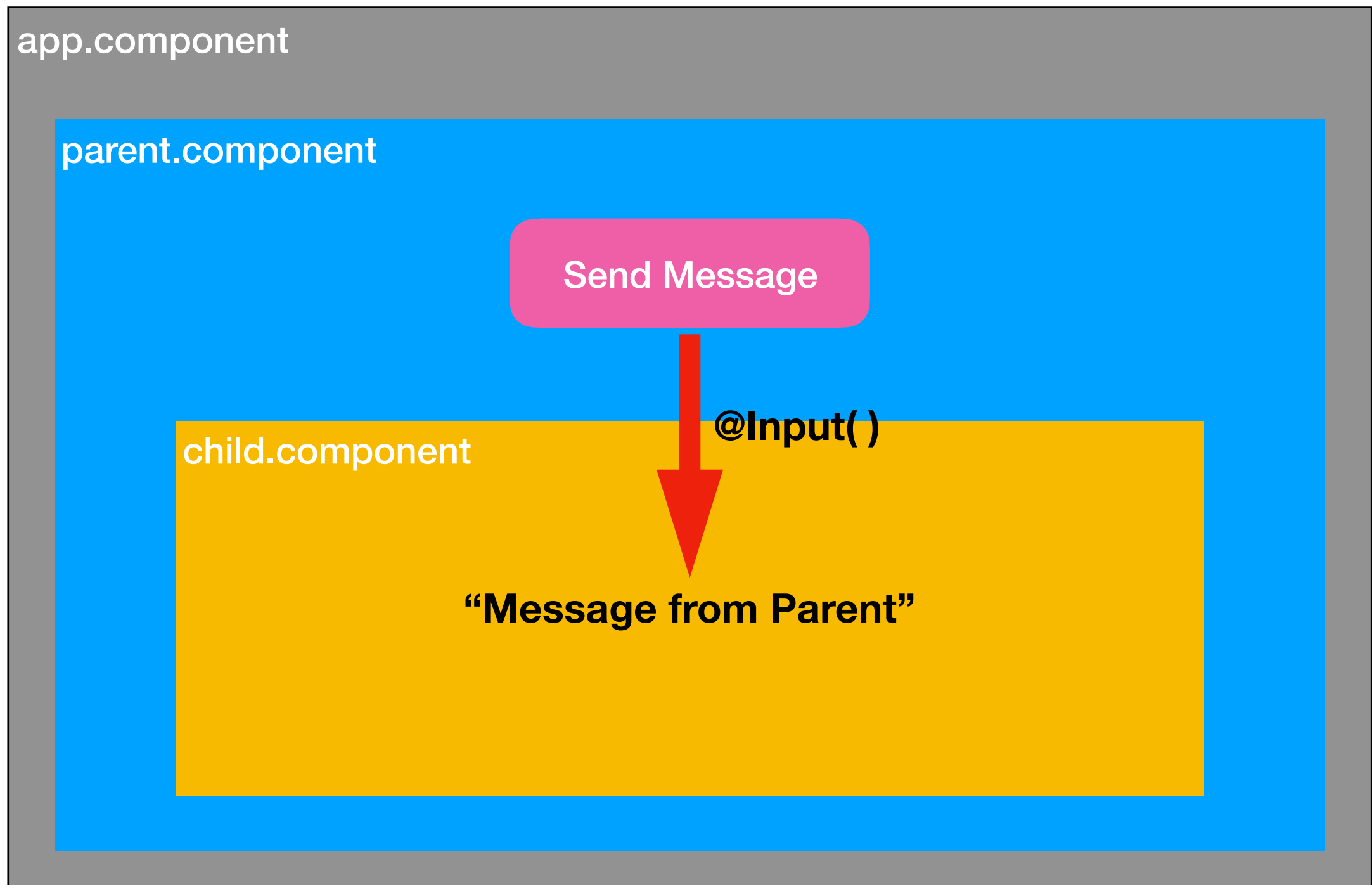
(Advanced Web Application Development)

Dr. Nuntawut Kaoungku
Assistant Professor of Computer Engineering

Component Communication



Parent to Child



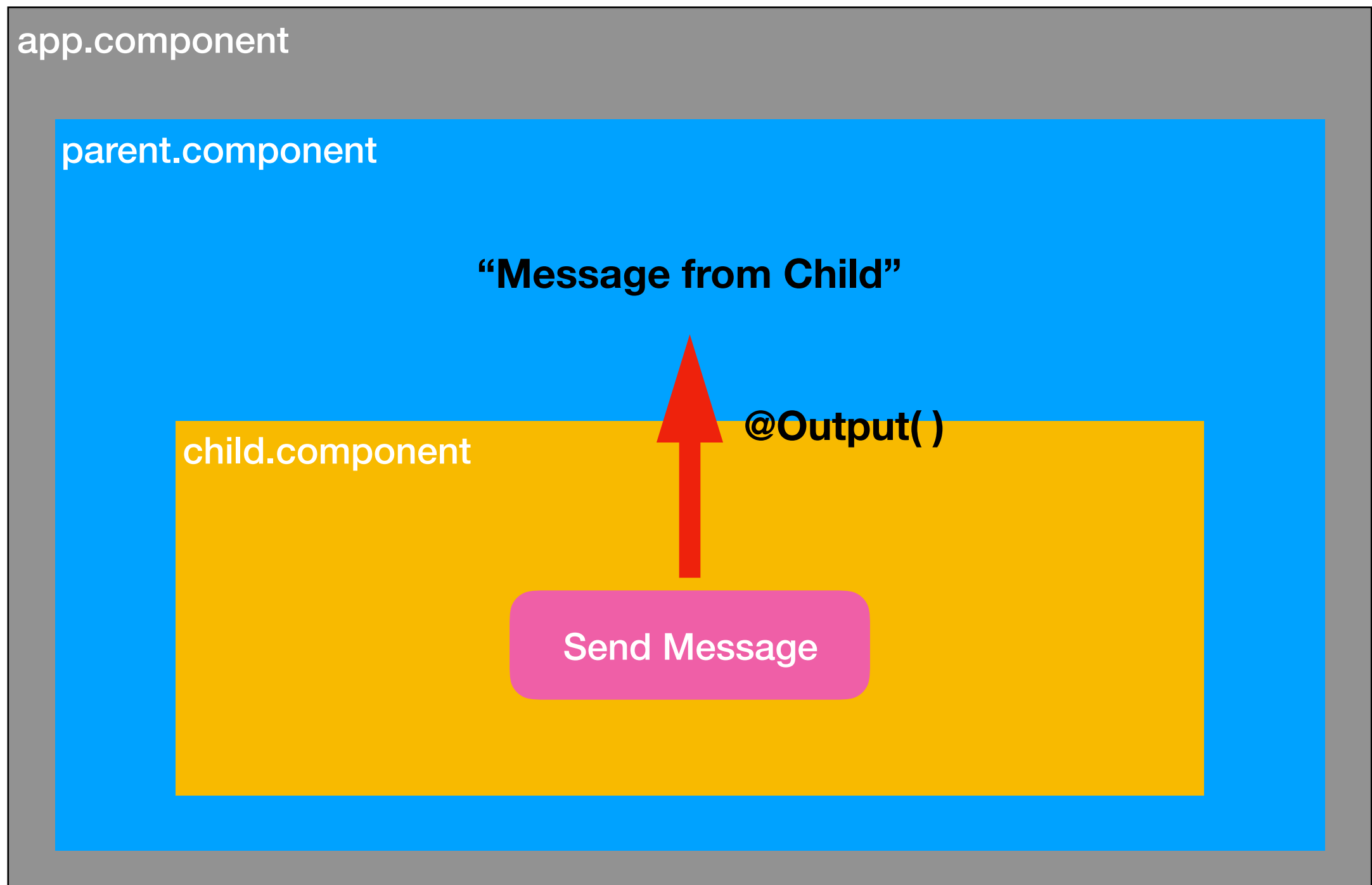
Parent to Child : Example

```
parent.component.html •
lab3-angular-app > src > app > components > parent > <> parent.component.html > ...
1  <div class="container">
2    <div class="card bg-primary text-white">
3      <div class="card-body">
4        <h4 class="card-title">Parent</h4>
5        <p class="card-text">
6          <button type="button" class="btn btn-light"
7            (click)="onClickParent()">Send to Child</button>
8          <app-child [parentMessage]="parentMessage"></app-child>
9        </p>
10     </div>
11   </div>
12 </div>
13
```

```
parent.component.ts •
lab3-angular-app > src > app > components > parent > TS parent.comp
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-parent',
5    templateUrl: './parent.component.html',
6    styleUrls: ['./parent.component.css']
7  })
8  export class ParentComponent implements OnInit {
9
10     parentMessage: number = 0;
11
12     constructor() { }
13
14     ngOnInit(): void {
15     }
16
17     onClickParent(){
18       this.parentMessage++;
19     }
20
21   }
22
```

```
child.component.ts •
lab3-angular-app > src > app > components > child > TS child.component.ts > ...
1  import { Component, OnInit, Input } from '@angular/core';
2
3  @Component({
4    selector: 'app-child',
5    templateUrl: './child.component.html',
6    styleUrls: ['./child.component.css']
7  })
8  export class ChildComponent implements OnInit {
9
10     @Input() parentMessage: number;
11
12     childMessage: number = 0;
13
14     constructor() { }
15
16     ngOnInit(): void {
17     }
18
19   }
```

Child to Parent



Child to Parent : Example

```
<> parent.component.html ×
lab3-angular-app > src > app > components > parent > <> parent.component.html > ...
1  <div class="container">
2    <div class="card bg-primary text-white">
3      <div class="card-body">
4        <h4 class="card-title">Parent</h4>
5        <p class="card-text">
6          Message from Child : {{childMessage}}
7          <app-child (messageEvent)="receiveMessage($event)"></app-child>
8        </p>
9      </div>
10   </div>
11 </div>
12
```

```
TS parent.component.ts •
lab3-angular-app > src > app > components > parent > TS parent.component.ts > ...
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-parent',
5    templateUrl: './parent.component.html',
6    styleUrls: ['./parent.component.css']
7  })
8  export class ParentComponent implements OnInit {
9
10     childMessage: number;
11
12     constructor() { }
13
14     ngOnInit(): void {
15     }
16
17     receiveMessage($event) {
18       this.childMessage = $event;
19     }
20
21   }
```

```
TS child.component.ts •
lab3-angular-app > src > app > components > child > TS child.component.ts > ...
1  import { Component, OnInit, Output, EventEmitter } from '@angular/core';
2
3  @Component({
4    selector: 'app-child',
5    templateUrl: './child.component.html',
6    styleUrls: ['./child.component.css']
7  })
8  export class ChildComponent implements OnInit {
9
10     childMessage: number = 0;
11
12     @Output() messageEvent = new EventEmitter<number>();
13
14     constructor() { }
15
16     ngOnInit(): void {
17     }
18
19     onClickChild(){
20       this.messageEvent.emit(this.childMessage++)
21     }
22
23   }
```

Child to Parent via @ViewChild

- ViewChild allows a one component to be injected into another, giving the parent access to its attributes and functions.
- One caveat, however, is that child won't be available until after the view has been initialized.
- This means we need to implement the AfterViewInit lifecycle hook to receive the data from the child.

ViewChild : Example

TS parent.component.ts •

```
lab3-angular-app > src > app > components > parent > TS parent.component.ts > ...
1  import { Component, OnInit, ViewChild } from '@angular/core';
2  import { ChildComponent } from '../child/child.component'
3
4  @Component({
5    selector: 'app-parent',
6    templateUrl: './parent.component.html',
7    styleUrls: ['./parent.component.css']
8  })
9  export class ParentComponent implements OnInit {
10
11    @ViewChild(ChildComponent)
12
13    childComponent: ChildComponent;
14
15    constructor() { }
16
17    ngOnInit(): void {
18    }
19
20    onClickViewChild(){
21      this.childComponent.onClickChild();
22    }
23
24  }
```

<> parent.component.html ×

```
lab3-angular-app > src > app > components > parent > <> parent.component.html > ...
1  <div class="container" style="margin-top:60px">
2    <div class="card bg-primary text-white">
3      <div class="card-body">
4        <h4 class="card-title">Parent</h4>
5        <p class="card-text">
6          <button type="button" class="btn bg-secondary"
7            (click)="onClickViewChild()">Access Property in Child</button>
8          <app-child></app-child>
9        </p>
10      </div>
11    </div>
12  </div>
13
14
```

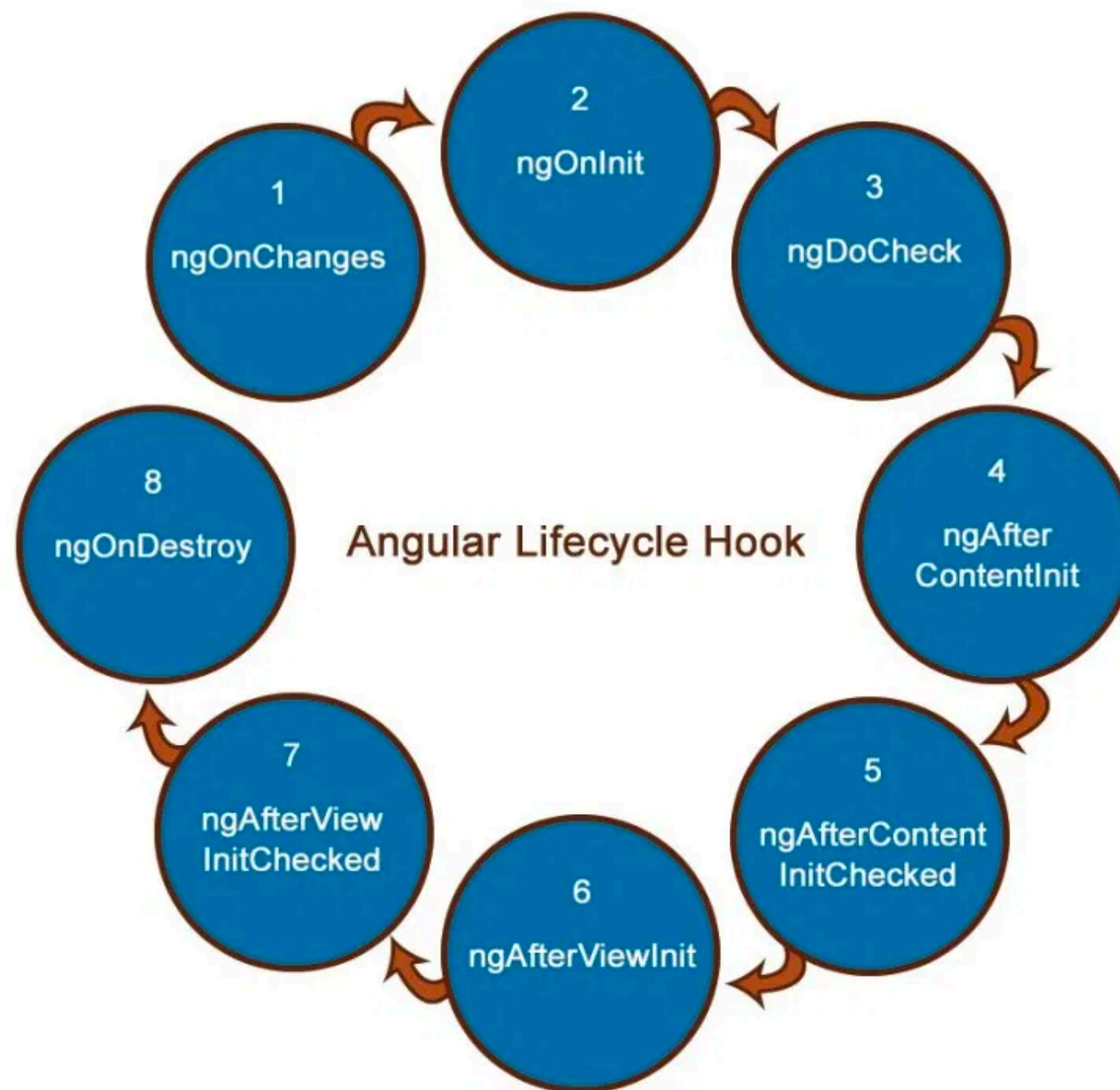
TS child.component.ts •

```
lab3-angular-app > src > app > components > child > TS child.component.ts > ...
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-child',
5    templateUrl: './child.component.html',
6    styleUrls: ['./child.component.css']
7  })
8  export class ChildComponent implements OnInit {
9
10    childMessage: number = 0;
11
12    constructor() { }
13    ngOnInit(): void {
14    }
15
16    onClickChild(){
17      this.childMessage++;
18    }
19
20  }
```


Lifecycle hooks

- A component has a lifecycle managed by Angular.
- Angular creates and renders components along with their children, checks when their data-bound properties change, and destroys them before removing them from the DOM.
- Angular offers lifecycle hooks that provide visibility into these key life moments and the ability to act when they occur.
- A directive has the same set of lifecycle hooks.

Lifecycle Sequence



ngOnChanges()

- Respond when Angular (re)sets **data-bound input properties**.
- The method receives a SimpleChanges object of current and previous property values.
- Called **before ngOnInit()** and whenever one or more data-bound input properties change.

```
ngOnChanges() {  
  console.log('ngOnChanges Work!');  
}
```

ngOnChanges() : Example

The screenshot illustrates the `ngOnChanges()` lifecycle hook in an Angular application. The application is running on `localhost:4200` and consists of two components: a **Parent** component (blue background) and a **Child** component (green background).

Parent Component:

- Contains a button labeled "Send to Child" (highlighted with a red circle).
- Contains a button labeled "Access Property in Child".
- Displays the message "Message from Child :".

Child Component:

- Displays the message "Message from Parent : 1" (highlighted with a red circle).
- Contains a button labeled "Click".

DevTools Console:

The console shows the sequence of lifecycle events. The `ngOnChanges()` event is highlighted with a red circle in two instances:

- First instance: `ngOnChanges Work! child.component.ts:29`
- Second instance: `ngOnChanges Work! child.component.ts:29`

Other lifecycle events shown in the console include `Constructor Work!`, `ngOnInit Work!`, `ngDoCheck Work!`, `ngAfterContentInit Work!`, `ngAfterContentChecked Work!`, `ngAfterViewInit Work!`, and `ngAfterViewChecked Work!`.

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().

```
ngOnInit() {  
  console.log('ngOnInit Work!');  
}
```

ngOnInit() : Example

The screenshot displays a web application running on `localhost:4200`. The application consists of two main components: a **Parent** component (blue background) and a **Child** component (green background).

Parent Component:

- Buttons: `Send to Child` and `Access Property in Child`.
- Text: `Message from Child :`

Child Component:

- Text: `Message from Parent : 0`
- Button: `Click`

DevTools Console:

The console shows the lifecycle events for both components. The `ngOnInit` event is highlighted with red circles for both the Parent and Child components.

| Event | File |
|--|-------------------------------------|
| Constructor Work! | <code>parent.component.ts:29</code> |
| Constructor Work! | <code>child.component.ts:26</code> |
| ngOnInit Work! | <code>parent.component.ts:35</code> |
| ngDoCheck Work! | <code>parent.component.ts:38</code> |
| ngAfterContentInit Work! | <code>parent.component.ts:41</code> |
| ngAfterContentChecked Work! | <code>parent.component.ts:44</code> |
| ngOnChanges Work! | <code>child.component.ts:29</code> |
| ngOnInit Work! | <code>child.component.ts:32</code> |
| ngDoCheck Work! | <code>child.component.ts:35</code> |
| ngAfterContentInit Work! | <code>child.component.ts:38</code> |
| ngAfterContentChecked Work! | <code>child.component.ts:41</code> |
| ngAfterViewInit Work! | <code>child.component.ts:44</code> |
| ngAfterViewChecked Work! | <code>child.component.ts:47</code> |
| ngAfterViewChecked Work! | <code>parent.component.ts:47</code> |
| ngAfterViewChecked Work! | <code>parent.component.ts:50</code> |
| Angular is running in the development mode. Call <code>enableProdMode()</code> to enable the production mode. | <code>core.js:40480</code> |
| ngDoCheck Work! | <code>parent.component.ts:38</code> |
| ngAfterContentChecked Work! | <code>parent.component.ts:44</code> |
| ngDoCheck Work! | <code>child.component.ts:35</code> |
| ngAfterContentChecked Work! | <code>child.component.ts:41</code> |
| ngAfterViewChecked Work! | <code>child.component.ts:47</code> |
| ngAfterViewChecked Work! | <code>parent.component.ts:50</code> |
| DevTools failed to parse SourceMap: <code>chrome-extension://gighmmpioibklfepjocnamgkbiglidom/include.postload.js.map</code> | |
| [WDS] Live Reloading enabled. | <code>client:52</code> |

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()`.

```
ngDoCheck() {  
  console.log('ngDoCheck Work!');  
}
```

ngDoCheck() : Example

The screenshot illustrates the `ngDoCheck()` lifecycle hook in an Angular application. The application is running in a browser at `localhost:4200` under the name `Lab2AngularApp`.

UI Components:

- Parent Component (Blue Box):** Contains two buttons: `Send to Child` (highlighted with a red circle) and `Access Property in Child` (highlighted with a yellow circle). Below them is a text field showing `Message from Child : 0` (highlighted with a yellow circle).
- Child Component (Green Box):** Contains a text field showing `Message from Parent : 1` (highlighted with a red circle) and a `Click` button.

DevTools Console:

The console shows the sequence of lifecycle hooks for both components. The `ngDoCheck` hook is highlighted in red in the console log, indicating when the change detection cycle is triggered.

Console Log (relevant entries):

```
ngDoCheck Work! parent.component.ts:38
ngAfterContentChecked Work! parent.component.ts:44
ngOnChanges Work! child.component.ts:29
ngDoCheck Work! child.component.ts:35
ngAfterContentChecked Work! child.component.ts:41
ngAfterViewChecked Work! child.component.ts:47
ngAfterViewInit Work! parent.component.ts:47
ngAfterViewChecked Work! parent.component.ts:50
[WDS] Live Reloading enabled. client:52
DevTools failed to parse SourceMap: chrome-extension://gighmmpioibklfepjocnamgkbbiglidom/
include.postload.js.map
ngDoCheck Work! parent.component.ts:38
ngAfterContentChecked Work! parent.component.ts:44
ngOnChanges Work! child.component.ts:29
ngDoCheck Work! child.component.ts:35
ngAfterContentChecked Work! child.component.ts:41
ngAfterViewChecked Work! child.component.ts:47
ngAfterViewChecked Work! parent.component.ts:50
ngDoCheck Work! parent.component.ts:38
ngAfterContentChecked Work! parent.component.ts:44
ngDoCheck Work! child.component.ts:35
ngAfterContentChecked Work! child.component.ts:41
ngAfterViewChecked Work! child.component.ts:47
ngAfterViewChecked Work! parent.component.ts:50
```

Arrows from the UI elements point to the corresponding log entries in the console:

- Red arrow from `Send to Child` button to the first `ngDoCheck` log entry.
- Yellow arrow from `Access Property in Child` button to the `ngOnChanges` log entry.
- Yellow arrow from `Message from Child : 0` text field to the `ngDoCheck` log entry.
- Red arrow from `Message from Parent : 1` text field to the `ngDoCheck` log entry.

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().

```
ngAfterContentInit() {  
  console.log('ngAfterContentInit Work!');  
}
```

ngAfterContentChecked()

- Respond after Angular checks the content projected into the directive/component.
- Called after the ngAfterContentInit() and every subsequent ngDoCheck().

```
ngAfterContentChecked() {  
  console.log('ngAfterContentChecked Work!');  
}
```

ngAfterContentChecked() : Example

The screenshot displays a web application with a parent component (blue box) and a child component (green box). The parent component has two buttons: "Send to Child" and "Access Property in Child". Below these buttons, it shows "Message from Child : 0". The child component shows "Message from Parent : 1" and a "Click" button.

The Chrome DevTools console on the right shows the lifecycle events for both components. The events for the child component are as follows:

| Event | Location |
|--|------------------------|
| ngAfterContentChecked Work! | parent.component.ts:44 |
| ngOnChanges Work! | child.component.ts:29 |
| ngOnInit Work! | child.component.ts:32 |
| ngDoCheck Work! | child.component.ts:35 |
| ngAfterContentInit Work! | child.component.ts:38 |
| ngAfterContentChecked Work! | child.component.ts:41 |
| ngAfterViewInit Work! | child.component.ts:44 |
| ngAfterViewChecked Work! | child.component.ts:47 |
| ngAfterViewInit Work! | parent.component.ts:47 |
| ngAfterViewChecked Work! | parent.component.ts:50 |
| Angular is running in the development mode. Call enableProdMode() to enable the production mode. | core.js:40480 |
| ngDoCheck Work! | parent.component.ts:38 |
| ngAfterContentChecked Work! | parent.component.ts:44 |
| ngDoCheck Work! | child.component.ts:35 |
| ngAfterContentChecked Work! | child.component.ts:41 |
| ngAfterViewChecked Work! | child.component.ts:47 |
| ngAfterViewChecked Work! | parent.component.ts:50 |
| [WDS] Live Reloading enabled. | client:52 |
| DevTools failed to parse SourceMap: chrome-extension://gighmmpioibklfepjocnamgkbbiglidom/include.postload.js.map | |
| ngDoCheck Work! | parent.component.ts:38 |
| ngAfterContentChecked Work! | parent.component.ts:44 |
| ngOnChanges Work! | child.component.ts:29 |
| ngDoCheck Work! | child.component.ts:35 |
| ngAfterContentChecked Work! | child.component.ts:41 |
| ngAfterViewChecked Work! | child.component.ts:47 |
| ngAfterViewChecked Work! | parent.component.ts:50 |
| ngDoCheck Work! | parent.component.ts:38 |
| ngAfterContentChecked Work! | parent.component.ts:44 |
| ngDoCheck Work! | child.component.ts:35 |
| ngAfterContentChecked Work! | child.component.ts:41 |
| ngAfterViewChecked Work! | child.component.ts:47 |
| ngAfterViewChecked Work! | parent.component.ts:50 |

The console also shows a warning message: "DevTools failed to parse SourceMap: chrome-extension://gighmmpioibklfepjocnamgkbbiglidom/include.postload.js.map".

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().

```
ngAfterViewInit() {  
  console.log('ngAfterViewInit Work!');  
}
```

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().

```
ngAfterViewChecked() {  
  console.log('ngAfterViewChecked Work!');  
}
```

ngAfterViewChecked() :

Example

The screenshot shows a web browser window with the title "Lab2AngularApp" and the address "localhost:4200". The browser displays a blue rectangular component labeled "Parent". Inside this component is a white input field containing the text "Test1234". Below the input field, the text "Message from Child : OK" is displayed. To the right of the browser window is the Chrome DevTools console, which shows a series of log messages. The messages are grouped by file and line number, with the file being "include.postload.js.map". The messages are as follows:

| Message | File | Line |
|-------------------------------|---------------------|------|
| [WDS] Live Reloading enabled. | client | 52 |
| T | parent.component.ts | 43 |
| ngDoCheck Work! | parent.component.ts | 44 |
| ngAfterContentChecked Work! | parent.component.ts | 53 |
| ngAfterViewChecked Work! | parent.component.ts | 61 |
| Te | parent.component.ts | 43 |
| ngDoCheck Work! | parent.component.ts | 44 |
| ngAfterContentChecked Work! | parent.component.ts | 53 |
| ngAfterViewChecked Work! | parent.component.ts | 61 |
| Tes | parent.component.ts | 43 |
| ngDoCheck Work! | parent.component.ts | 44 |
| ngAfterContentChecked Work! | parent.component.ts | 53 |
| ngAfterViewChecked Work! | parent.component.ts | 61 |
| Test | parent.component.ts | 43 |
| ngDoCheck Work! | parent.component.ts | 44 |
| ngAfterContentChecked Work! | parent.component.ts | 53 |
| ngAfterViewChecked Work! | parent.component.ts | 61 |
| Test1 | parent.component.ts | 43 |
| ngDoCheck Work! | parent.component.ts | 44 |
| ngAfterContentChecked Work! | parent.component.ts | 53 |
| ngAfterViewChecked Work! | parent.component.ts | 61 |
| Test12 | parent.component.ts | 43 |
| ngDoCheck Work! | parent.component.ts | 44 |
| ngAfterContentChecked Work! | parent.component.ts | 53 |
| ngAfterViewChecked Work! | parent.component.ts | 61 |
| Test123 | parent.component.ts | 43 |
| ngDoCheck Work! | parent.component.ts | 44 |
| ngAfterContentChecked Work! | parent.component.ts | 53 |
| ngAfterViewChecked Work! | parent.component.ts | 61 |
| Test1234 | parent.component.ts | 43 |
| ngDoCheck Work! | parent.component.ts | 44 |
| ngAfterContentChecked Work! | parent.component.ts | 53 |
| ngAfterViewChecked Work! | parent.component.ts | 61 |

```
ngDoCheck() {  
  console.log(this.textMessage);  
  console.log('ngDoCheck Work!');  
}  
  
ngAfterContentChecked() {  
  if(this.textMessage == "Test1234"){  
    this.status = "OK"  
  }else{  
    this.status = "Error"  
  }  
  console.log('ngAfterContentChecked Work!');  
}
```

ngDestroy()

- 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.Lifecycle sequence

```
ngOnDestroy() {  
  | console.log('ngOnDestroy Work!');  
  }  
}
```