Angular. Angular CLI. Components

Angular CLI

- The Angular CLI is a command line interface tool that can create a project, add files, and perform a variety of ongoing development tasks such as testing, bundling, and deployment.

Установка Angular CLI

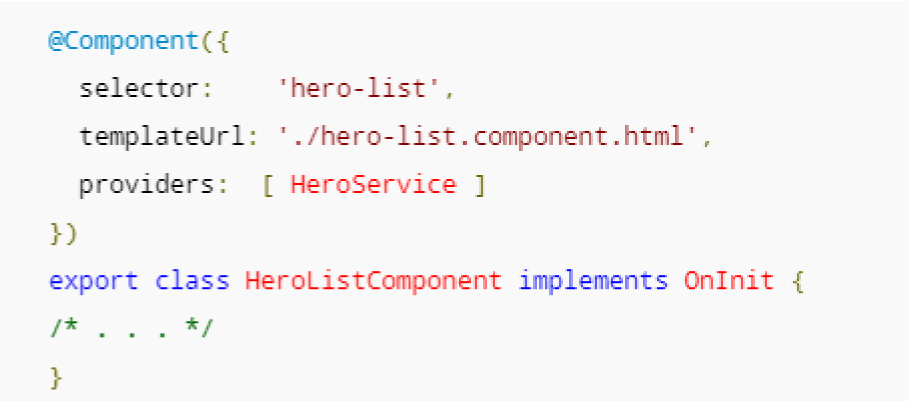
```
npm install -g @angular/cli
```

```
ng new my-app
```

```
ng serve --open
```

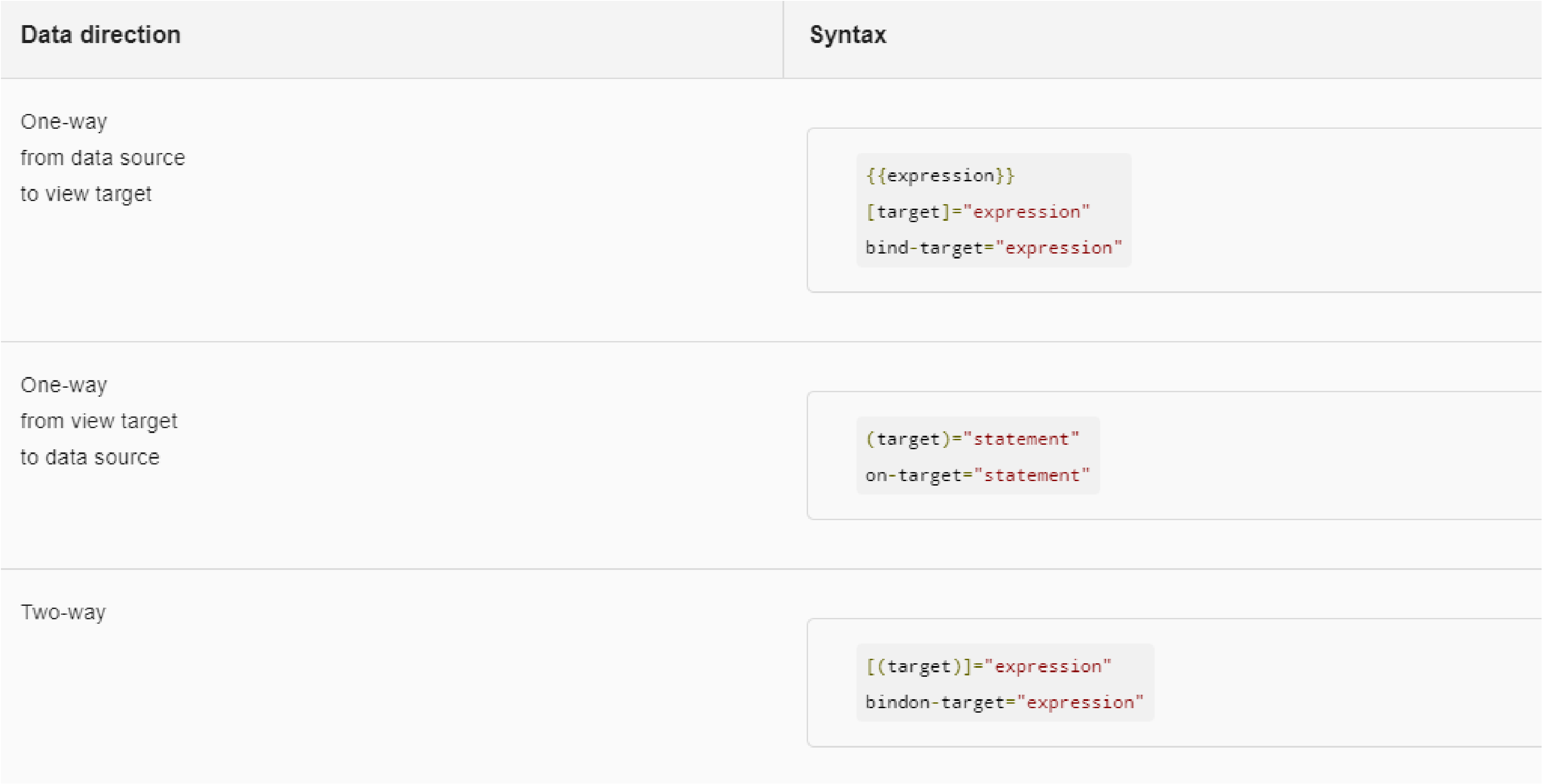
Компоненты

- Представление части отображения на странице.
- Взаимодействует с отображением.



```
@Component({
  selector:    'hero-list',
  templateUrl: './hero-list.component.html',
  providers:  [ HeroService ]
})
export class HeroListComponent implements OnInit {
/* . . . */
}
```

Template expressions



Data direction	Syntax

One-way
from data source
to view target
	{{expression}}
	[target]="expression"
	bind-target="expression"

One-way
from view target
to data source
	`(target)="statement"`
	`on-target="statement"`

Two-way	`[(target)]="expression"`
	`bindon-target="expression"`

Взаимодействие между компонентами

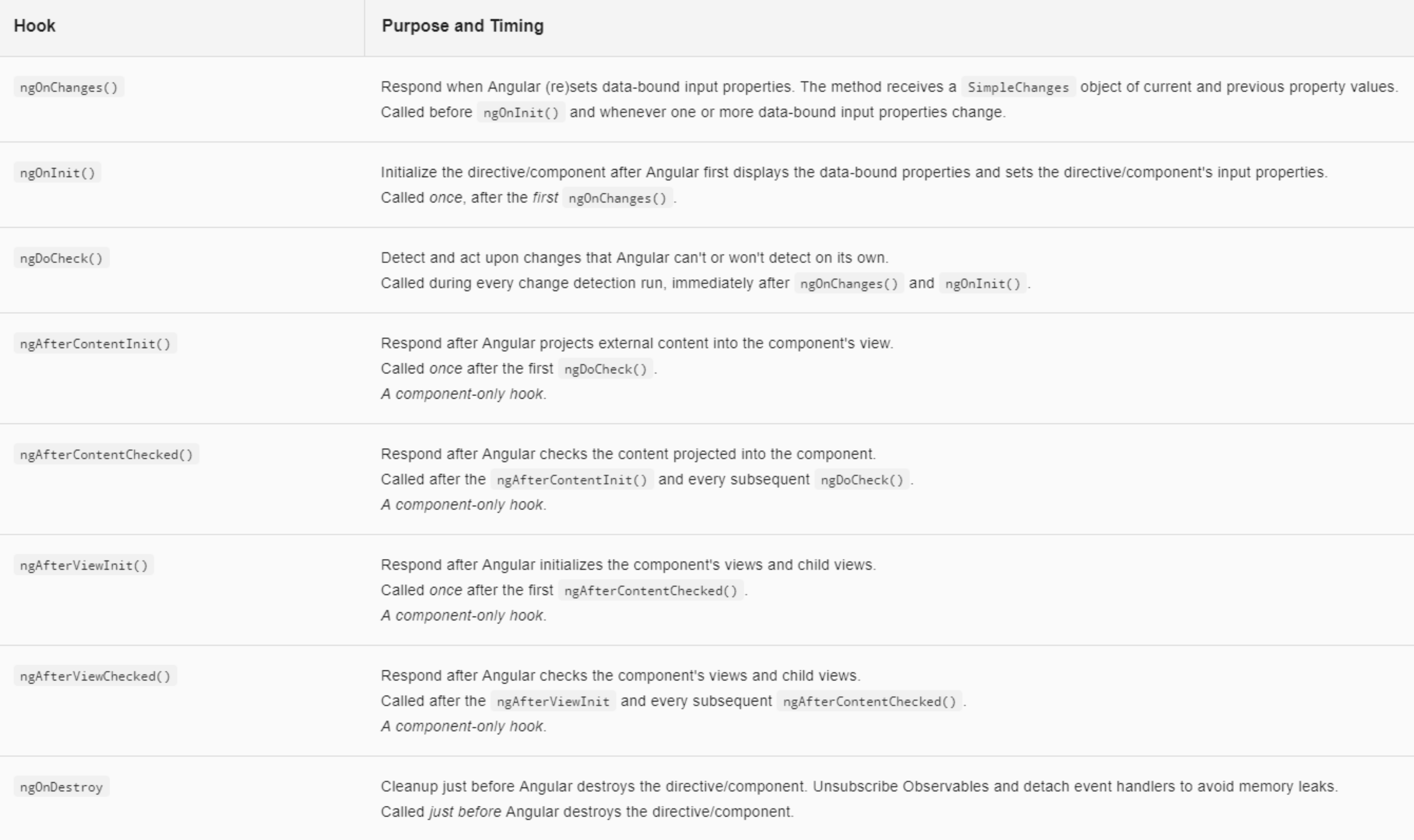
```
import { Component } from '@angular/core';

@Component({
    selector: 'my-app',
    template: `<child-comp [userName]="name" [userAge]="age"></child-comp>
                <input type="text" [(ngModel)]="name" />`
})
export class AppComponent {
    name: string = "Tom";
    age: number = 24;
}
```

```
import { Component, EventEmitter, Input, Output} from '@angular/core';

@Component({…})
class ChildComponent {
 …
 @Input() userName: string;
 @Input() userAge: string;
 …
}
```

Привязка к событиям дочернего компонента

```
import { Component, EventEmitter, Input, Output} from '@angular/core';

@Component({…})
class MyComponent {
 …
 @Output() onChanged = new EventEmitter<boolean>();
 …
}
```

Lifecycle Hooks



Hook	Purpose and Timing
`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.
`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. Called *once* after the first `ngDoCheck()` . *A component-only hook.*
`ngAfterContentChecked()`	Respond after Angular checks the content projected into the component. Called after the `ngAfterContentInit()` and every subsequent `ngDoCheck()` . *A component-only hook.*
`ngAfterViewInit()`	Respond after Angular initializes the component's views and child views. Called *once* after the first `ngAfterContentChecked()` . *A component-only hook.*
`ngAfterViewChecked()`	Respond after Angular checks the component's views and child views. Called after the `ngAfterViewInit` and every subsequent `ngAfterContentChecked()` . *A component-only hook.*
`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.

Ресурсы

- https://angular.io/
- https://metanit.com/web/angular2/

Q&A