

Attribute directives property bindings



Wat zijn directives

- Stukken HTML code met Angular logica
 - Koppelen van specifiek gedrag
 - Transformatie van een element
- Directives bestaan in 2 vormen in HTML templates:
 - Als attribuut van een element:

```
<div *ngIf="someVar">...</div>
```
 - Als elementnaam:

```
<app-myDirective></app-myDirective>
```
- Een deel directives zijn builtIn, andere directives zelf aanmaken

Soorten directives

Type	Omchrijving	Voorbeeld
Component	Een klasse met variabelen, methodes, een HTML template en een directive naam (selector tag).	<app-login></app-login> <app-contacts></app-contacts>
Attribuut directives	Wijzigt het element waarin de directive staat op een bepaalde manier.	ngStyle, ngClass, ...
Structurele directives	Aanbrengen van wijzigingen aan de DOM door elementen toe te voegen of te verwijderen.	ngIf, ngFor, ...

Stijlen toepassen m.b.v. [ngClass]

- ngClass: dynamisch toevoegen of verwijderen van een klasse aan een element

```
<div [ngClass] = "conditie">...</div>
```

- De conditie kan een value, expressie of functie zijn.

Stijlen toepassen m.b.v. [ngClass]

- De conditie van ngClass kan verschillende waardes krijgen:

Datatype	Voorbeeld	Omschrijving
String	“active bordered”	Alle klassen in de string worden gekoppeld aan het element. In dit voorbeeld zijn dit de klassen ‘active’ en ‘bordered’.
Array	[“active”, “bordered”]	Alle klassen in de array worden gekoppeld aan het element. In dit voorbeeld zijn dit de klassen ‘active’ en ‘bordered’.
Object	{active: isActive, bordered: hasBorder}	Voegt elke klasse toe, waarvan de waarde true is. In dit voorbeeld wordt de klasse ‘active’ toegevoegd als de variabele ‘isActive’ de waarde true heeft.

Stijlen toepassen m.b.v. [ngClass]

- Praktisch voorbeeld waarbij we starten van onderstaande component:

```
@Component({
  selector: 'app-styles',
  template: `<span>Test tekst</span>
    <br/> Voorzie rand: <input type="checkbox" [(ngModel)]="hasBorder"/><br/>
    Actief: <input type="checkbox" [(ngModel)]="isActive"/>`,
  styles: [`.active{color: red;}` .bordered{border: solid 1px black;}`]
})
export class StylesComponent {
  hasBorder: boolean = false;
  isActive: boolean = false;
}
```

- 2 CSS klasses voorzien: .active en .bordered
- De variabelen uit de StylesComponent klasse zijn gekoppeld aan de checkboxen via 2-way binding

Stijlen toepassen m.b.v. [ngClass]

- ngClass import toevoegen
- krijgt de ngClass directive met een object als value:
 - .bordered wordt toegevoegd als hasBordered = true
 - .active wordt toegevoegd als isActive = true;

```
<span [ngClass]="{bordered: hasBorder, active: isActive}">Test tekst</span>
<br/>
Voorzie rand: <input type="checkbox" [(ngModel)]="hasBorder"/><br/>
Actief: <input type="checkbox" [(ngModel)]="isActive"/>
```

Stijlen toepassen m.b.v. [ngClass]

```
import { Component, OnInit } from '@angular/core';
import { NgClass } from '@angular/common';

@Component({
  selector: 'app-styles',
  template: `<span
[ngClass]="{bordered: hasBorder, active: isActive}">Test tekst</span>
<br/> Voorzie rand: <input type="checkbox" [(ngModel)]="hasBorder"/><br/>
Actief: <input type="checkbox" [(ngModel)]="isActive"/>`,
  styles: [`.active{color: red;}` `.bordered{border: solid 1px black;}`]
})
export class StylesComponent {
  hasBorder: boolean = false;
  isActive: boolean = false;
}
```

Test tekst
Voorzie rand:
Actief:

Test tekst
Voorzie rand:
Actief:

Stijlen toepassen m.b.v. [ngStyle]

- **ngStyle**: Hiermee kunnen we CSS stijlen direct toevoegen aan een HTML element:

```
<div [ngStyle] = "styleExpressie">...</div>
```

- De conditie kan een Object, variabele of functie zijn.

Stijlen toepassen m.b.v. [ngStyle]

- De expressie van ngStyle kan verschillende waardes krijgen:

Type	Voorbeeld
Object	[ngStyle] = “{color: ‘blue’}”;
Variabele (uit de component)	[ngStyle] = “styles” // In de HTML styles: Object = { color: ‘blue’ } // In component klasse
Functie	[ngStyle] = “getStyles()” // In de HTML getStyles(){ // In de component klasse return {color: ‘blue’}; }

Stijlen toepassen m.b.v. [ngStyle]

- Praktisch voorbeeld waarbij we starten van onderstaande component:

```
@Component({
  selector: 'app-styles2',
  template: `<span>Tekst 1</span><br/>
    <span (click)="toggleStyles()">Klik mij</span>`,
})
export class Styles2Component {
  styles1: Object = { border: 'solid 1px black', color: 'red' };
  styles2: Object = { border: 'none', color: 'black' };
  styles: Object = this.styles1;

  toggleStyles() {
    if (this.styles = this.styles1) {
      this.styles = this.styles2;
    } else {
      this.styles = this.styles1;
    }
  }
}
```

- Styles zijn reeds voorzien als objecten in de component klasse
- toggleStyles() zorgt ervoor dat de variabele styles aangepast wordt

Stijlen toepassen m.b.v. [ngStyle]

- ngStyle import toevoegen
- krijgt de ngStyle directive met een object als value
- krijgt de ngStyle directive met een variable als value

```
<span [ngStyle]="{fontStyle:'italic', fontFamily:'sans-serif'}">Tekst 1</span>
<br/><span [ngStyle]="styles" (click)="toggleStyles()">Klik mij</span>
```

Stijlen toepassen m.b.v. [ngStyle]

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

@Component({
  selector: 'app-styles2',
  template: `<span>
    [ngStyle]="{fontStyle:'italic', fontFamily:'sans-serif'}">Tekst 1</span>
    <br/><span [ngStyle]="styles" (click)="toggleStyles()">Klik mij</span>`,
})
export class Styles2Component {
  styles1: Object = { border: 'solid 1px black', color: 'red' };
  styles2: Object = { border: 'none', color: 'black' };
  styles: Object = this.styles1;

  toggleStyles() {
    if (this.styles == this.styles1) {
      this.styles = this.styles2;
    } else {
      this.styles = this.styles1;
    }
  }
}
```

Tekst 1

Klik mij

Tekst 1

Klik mij

Element properties in Angular

- Binnen angular kan je eender welke DOM properties van elementen aanspreken

```
[prop_naam] = "variabele|expressie"
```

- De value kan volgende zaken bevatten:
 - Statische waardes
 - Variabelen
 - Expressies
 - Functies

Element properties in Angular

- Binding is dynamisch – properties veranderen als de variabele / statement verandert
- Voorbeeld:

```
[style.color] = “isRed ? ‘red’ : ‘blue’;”
[disabled] = “isDisabled”
[style.width] = “50px”
```

Zichtbaarheid van een element

- Gebruik maken van [hidden] property
- Syntax:

```
<div [hidden] = "conditie">...</div>
```

- De conditie kan een variabele of expressie zijn
- Voorbeeld:

```
<div [hidden] = "!showDiv">...</div>
```

Dynamische source van een afbeelding

- Gebruik maken van de [src] property
- Syntax:

```
<img [src] = "imageUrl" />
```

- De conditie kan een variabele of functie zijn
- [Src] kan hier dus ook variabel zijn, wat met “src=“ niet kan
- Voorbeeld:

```
<img [src] = "getImageUrl()" > ... </img>
```

Source van een afbeelding

```
@Component({
  selector: 'app-image',
  template: `<h3>Scr voorbeeld</h3>
<img [src]="imageUrl"/><br/>
<input type="button" value="wijzig" (click)="toggleImage()">`,
  styleUrls: ['./image.component.css']
})
export class ImageComponent {
  imageUrl = 'assets/badge1.png';
  index: number = 1;

  toggleImage() {
    this.index = this.index + 1;
    this.index = (this.index < 5) ? this.index : 1;
    this.imageUrl = 'assets/badge' + this.index + '.png';
  }
}
```

Scr voorbeeld



wijzig

Scr voorbeeld



wijzig

Dynamische hyperlinks

- Gebruik maken van de [href] property
- Syntax:

```
<a [href] = "hrefUrl" />
```

- De conditie kan een variabele of functie zijn
- Href kan hier dus ook variabel zijn, wat met “href=” niet kan
- Voorbeeld:

```
<a [href] = "getHrefUrl()" >...</div>
```

Structurele directives

- Structurele directives passen de DOM aan door:
 - Elementen toe te voegen
 - Elementen te verwijderen
- *ngIf
- *ngFor
- *ngSwitch

Structurele directives: *ngIf

- *ngIf
- Bepaalde DOM elementen toevoegen of verwijderen
- Meestal gekoppeld aan een boolean variabele, expressie of functie
- Syntax:

```
<div *ngIf="showMessage"> ... </div>
```

- Als showMessage true is, wordt de div toegevoegd aan de DOM
- Het *-teken geeft aan dat dit een verkorte schrijfwijze is

Structurele directives: *ngIf

- Enkele voorbeelden:

- //gekoppeld aan de variable isVisible van het type boolean in de component klasse

```
*ngIf="isVisible"
```

- Gekoppeld aan de variabele myVar van het type string.
ngIf verwacht een boolean

```
*ngIf="myVar == 'showit'"
```

- Gekoppeld aan een functie in de component klasse die een boolean retourneert:

```
*ngIf="showIt()"
```

Structurele directives: *ngFor

- *ngFor is een repeater directive
- Loopt door JavaScript arrays
- Wordt voornamelijk gebruikt om lijsten en tabellen op te stellen.
- Kan toegepast worden op elk element:
 - <div>
 - <p>
 - <app-contact>

Structurele directives: *ngFor

- Basis Syntax:

```
<li *ngFor="let pet of pets">  
{{pet}}  
</li>
```

- Data van de array uit de component klasse:

```
Pets: string[] = ["Cat", "Dog", "Turtle"];
```

- pets verwijst naar de array uit de klasse. Daarnaast wordt er een lokale variabele pet aangemaakt.

Structurele directives *ngFor

```
@Component({
  selector: 'app-ngfor',
  template: `
    <table>
      <tr *ngFor="let person of people">
        <td>{{person.name}}</td>
        <td>{{person.firstName}}</td>
        <td>{{person.email}}</td>
      </tr>
    </table>
  `,
  styles: [`table, td{ border: 1px solid black`]
})
export class NgforComponent {
  people: object = [
    {name: 'Swinnen', firstName: 'Dries', email: 'dries.swinnen@pxl.be'},
    {name: 'Doe', firstName: 'John', email: 'john.doe@gmail.com'},
    {name: 'Doe', firstName: 'Jane', email: 'jane.doe@gmail.com'},
    {name: 'Vader', firstName: 'Darth', email: 'anakin@theEmpire.com'}
  ];
}
```

Structurele directives: *ngFor

- *ngFor voorziet enkele loop-gerelateerde waardes die je kan gebruiken binnen de loop:
 - Index: array index van het huidige item
 - First: boolean value die true is als het element het eerste element uit de array is
 - Last: boolean value die true is als het element het laatste element uit de array is
 - Even: boolean value die true is al het element zijn positie in de array even is
 - Odd: boolean value die true is als het element zijn positie in de array odd is

Structurele directives: *ngFor

```
template: `<table>
  <tr *ngFor="let person of people; let i=index; let e=even;" [ngClass]="{evenrow:e,oddrow:!e}">
    <td>{{i}}</td>
    <td>{{person.name}}</td>
    <td>{{person.firstName}}</td>
    <td>{{person.email}}</td>
  </tr>
</table>
`,
styles: [`.evenrow{ background-color: yellow; }
         .oddrow{ background-color: orange;}`]
})
export class NgforComponent {
  people: object = [
    {name: 'Swinnen', firstName: 'Dries', email: 'dries.swinnen@pxl.be'},
    {name: 'Doe', firstName: 'John', email: 'john.doe@gmail.com'},
    {name: 'Doe', firstName: 'Jane', email: 'jane.doe@gmail.com'},
    {name: 'Vader', firstName: 'Darth', email: 'anakin@theEmpire.com'}
  ];
}
```

Structurele directives: *ngFor

- Angular bekijkt de achterliggende array om de view te updaten.
- Data die toegevoegd wordt aan de array, wordt ook automatisch aangepast in de view
- Data die verwijderd wordt uit de array, wordt ook automatisch aangepast in de view
- Items toevoegen en verwijderen aan een array, gaat met eenvoudige JavaScript methodes:

```
pets.push(petToAdd);  
pets.splice(indexToRemove,1);
```

*ngFor en @Input()

- Het is mogelijk om het element uit de array door te geven naar een @Input() variabele
- Dit is nodig als je bijvoorbeeld een *ngFor in een child component plaatst

```
<!--  
  parent html  
  contactList is an array of contacts  
-->  
<app-contact *ngFor="let contactItem of contactList;"  
             [contactChild]="contactItem">  
</app-contact>
```

```
// <app-contact> component.  
// Gets input from ngFor  
export class ContactComponent implements OnInit {  
  @Input() contactChild: Contact;  
  
  ngOnInit() { }  
}
```

Structurele directives: *ngSwitch

- *ngSwitch is een structurele directive
- Wordt gebruikt naast ngSwitchCase en ngSwitchDefault
- Toont één element uit een bepaalde set, en verbergt de andere
- Kan gebruikt worden om bijvoorbeeld tabs te genereren.

Structurele directives: *ngSwitch

```
@Component({
  selector: 'app-ngswitch',
  template: `
    <div [ngSwitch]="types[index]">
      <p *ngSwitchCase="'one'">Page 1</p>
      <p *ngSwitchCase="'two'">Page 2</p>
      <p *ngSwitchCase="'three'">Page 3</p>
      <p *ngSwitchDefault>Default</p>
    </div>
    <button (click)="next()">Volgende</button>`,
  styleUrls: ['./ngswitch.component.css']
})
export class NgswitchComponent {
  types: string[] = ['one', 'two', 'three'];
  index: number = 0;

  next(){
    this.index++;
    this.index = (this.index < 4) ? this.index : 0;
  }
}
```