Angular

VEERLE ONGENAE

Angular Overzicht

- Typescript

Typescript

- Superset Javascript
- Typering toegevoegd
  - Modules
  - Klassen
  - Types
  - Interfaces
  - …
- .ts
- Compileren naar javascript
  - tsc bestand.ts → bestand.js

```
function greeter(person: string) {
    return "Hello, " + person;
}
```

Typescript interface

```
interface Person {
    firstName: string;
    lastName: string;
}

function greeter(person: Person) {
    return "Hello, " + person.firstName + " "
                + person.lastName;
}

let user = { firstName: "Jane", lastName: "User" };

let text = greeter(user);
```

```
class Student {
    fullName: string;

    constructor(public firstName: string,
                public middleInitial: string,
                public lastName: string) {

        this.fullName = firstName + " " + middleInitial
                                          + " " + lastName;
    }
}

interface Person {
    firstName: string;
    lastName: string;
}

function greeter(person : Person) {
    return "Hello, " + person.firstName + " "
                + person.lastName;
}
```

automatisch aanmaken properties

```
let user = new Student("Jane", "M.", "User");
let text = greeter(user);
```


Angular - Overzicht

- Typescript
- Wat heb je nodig?

Node.js

- Server side javascript framework
- Gebouwd op de javascript runtime van Chrome
- NPM
  - Node Package Manager
  - Javascript packages installeren
    - Handmatig
    - Op basis van configuratiebestand package.json

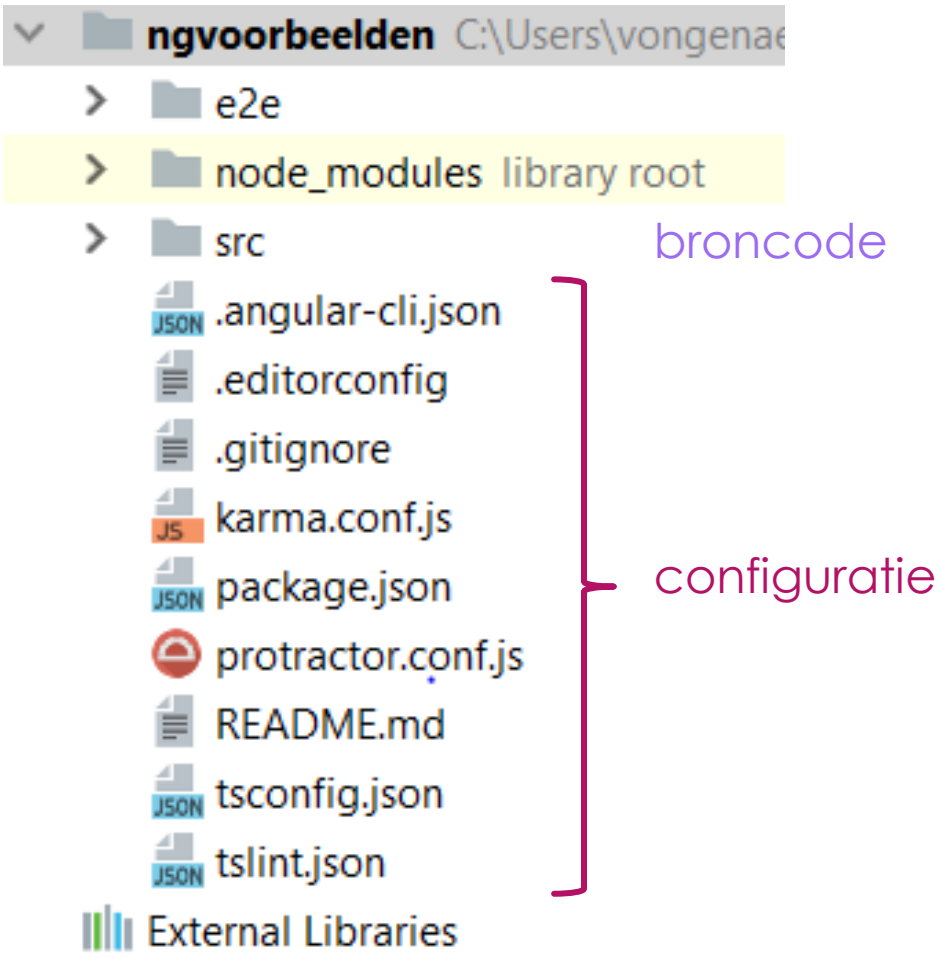
package.json

```
{
  "name" : "MyApp",
  "version" : "1.0.0",
  "dependencies" : {
      "sax" : "0.3.x",
      "nano" : "*",
      "request" : ">0.2.0"
  }
}
```

Angular CLI

- A command line interface for Angular
- Functionaliteit (zie ook https://github.com/angular/angular-cli)
  - Startproject genereren
  - Webserver starten om applicatie te testen
    - `ng server`
  - Angular componenten, … aanmaken
    - `ng generate component my-component`
  - …
- Installeren
  - `npm install -g @angular/cli`

Structuur project

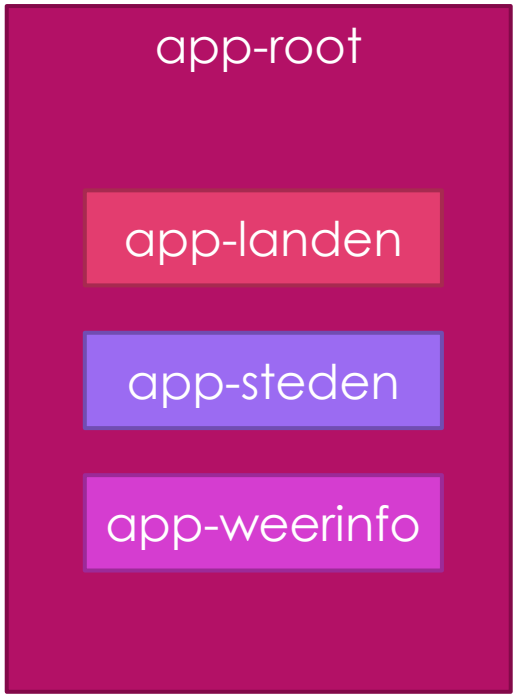


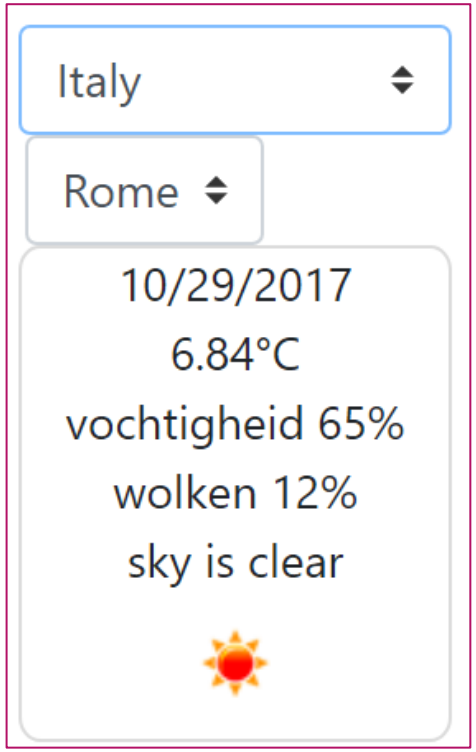
Angular - Overzicht

- Typescript
- Wat heb je nodig?
- Componenten

Wat is Angular?

- Basisidee: webpagina bestaat uit componenten (~**tag**)





Component

- Bouwsteen voor een webpagina

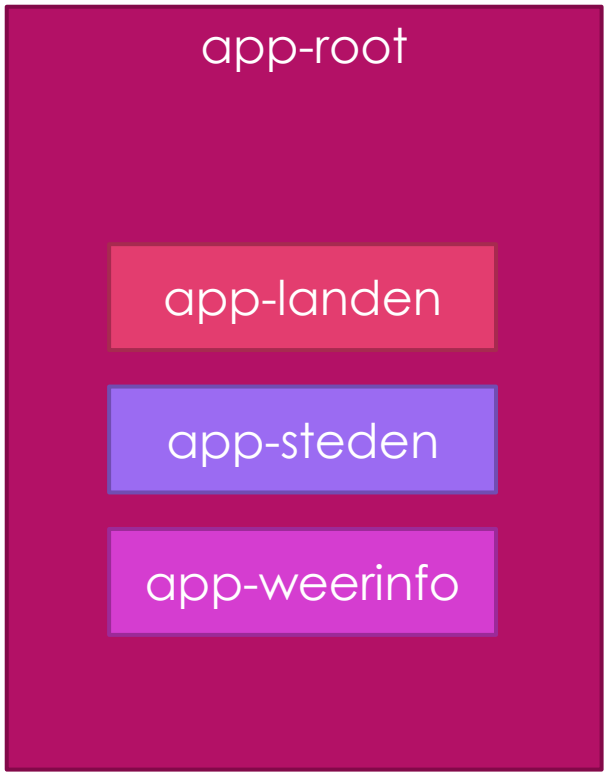
  ~ tag

index.html

```
<app-root></app-root>
```

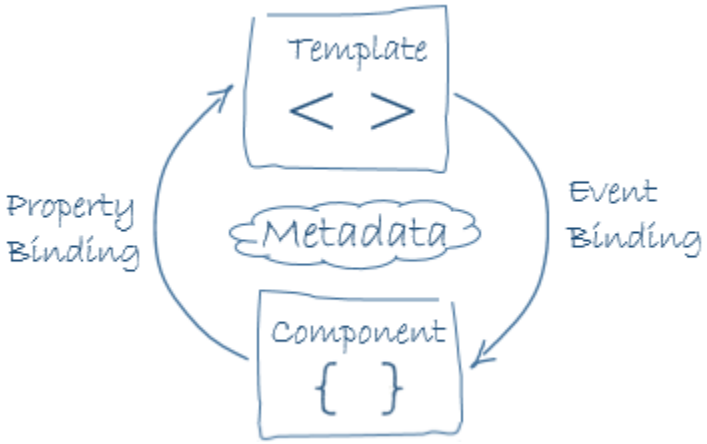
app.component.html

```
…
<app-landen …></app-landen>
<app-steden …></app-steden>
<app-weerinfo …></app-weerinfo>
…
```



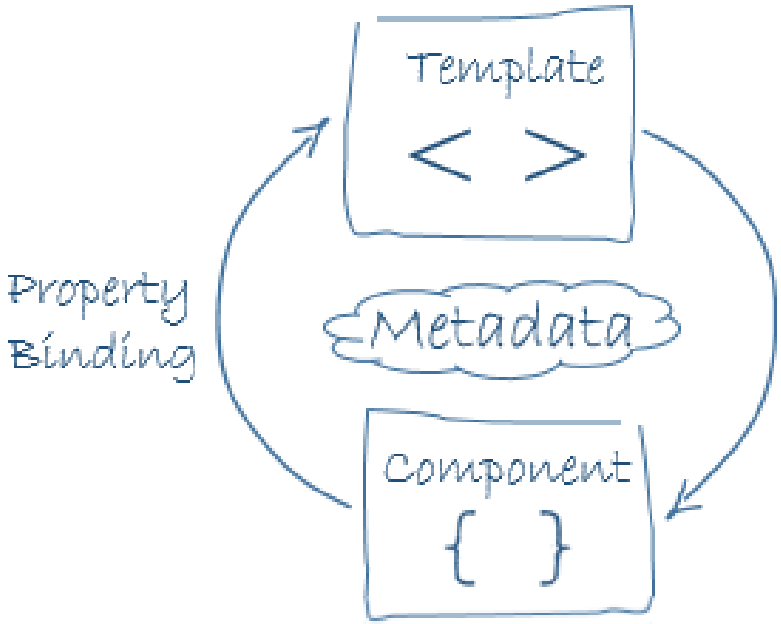
Component

- Bouwsteen voor een webpagina
- Bestaat uit
  - Klasse: functionaliteit
  - HTML-template (sjabloon): presentatie
  - Stylesheet: opmaak



Component

HTML

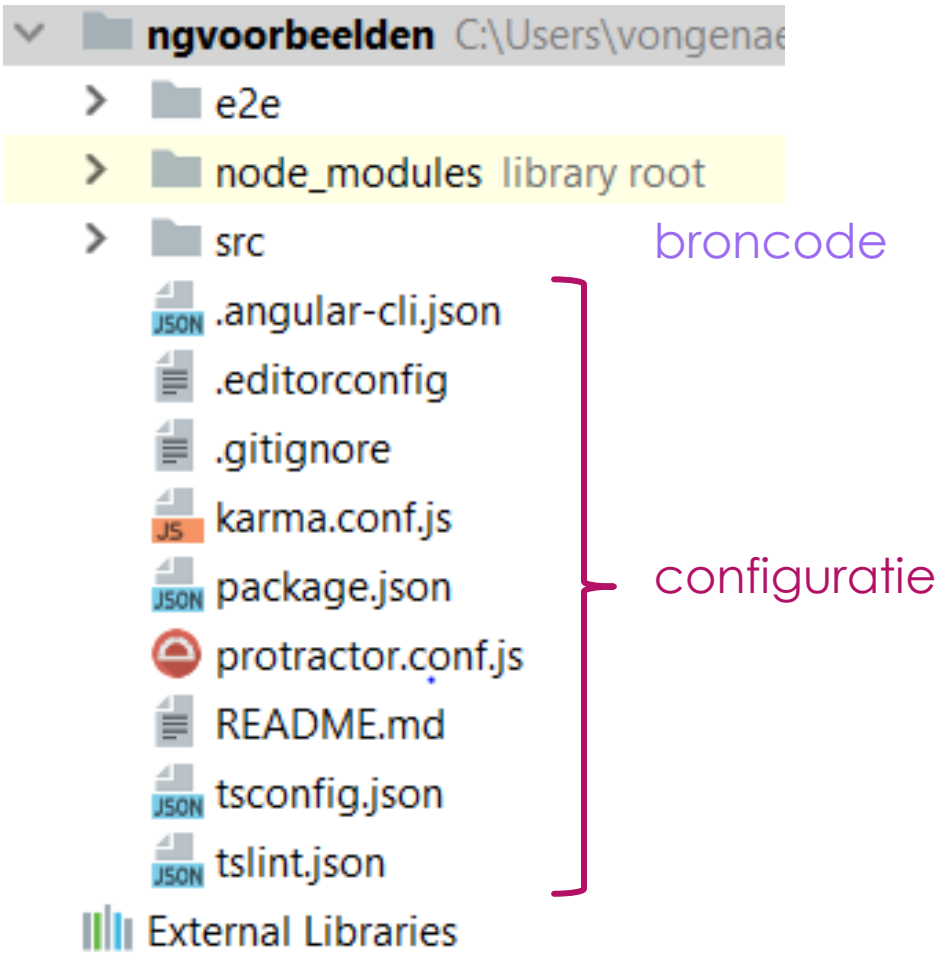


Attributen

Methodes

Klasse: attributen - methodes

Structuur project



Aanmaak Component

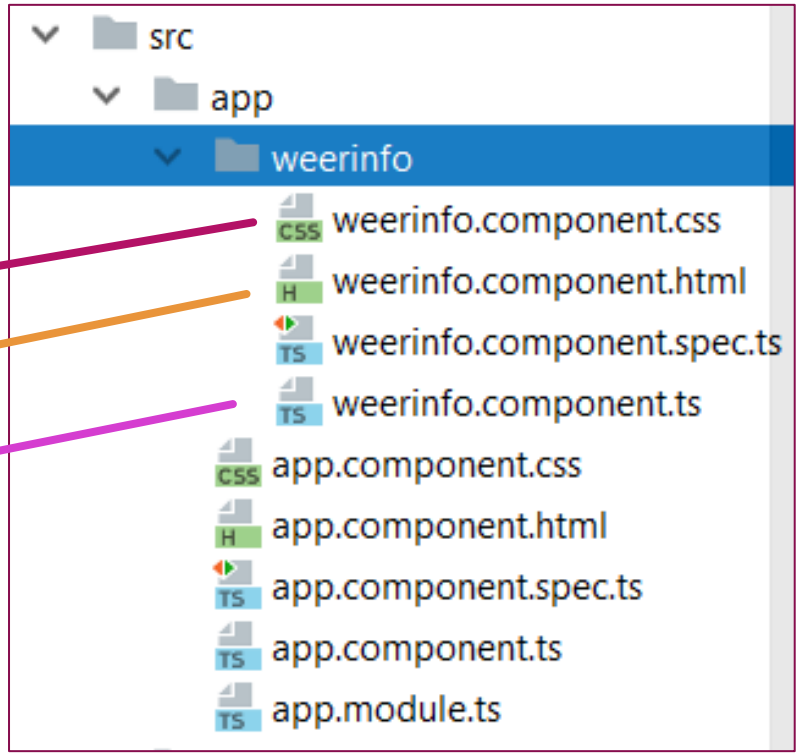
```
ng generate component my-component
```

```
ng g component my-component
```

stylesheet

template

klasse



Component – interpolation binding

HTML-template

```
<div class="weer">
  <div> {{datum}} </div>
  …
</div>
```

Klasse: attributen - methodes

```
@Component({
  selector: 'app-weerinfo',
  templateUrl: '….html',
  styleUrls: ['….css']
})
export class WeerinfoComponent {

  datum = '26/10/17’;
  …
}
```

Klasse component

weerinfo.component.ts


```
import {Component} from '@angular/core';
@Component({
  selector: 'app-weerinfo',
  templateUrl: './weerinfo.component.html',
  styleUrls: ['./weerinfo.component.css']
})
export class WeerinfoComponent {

  datum = '26/10/17';
  temperatuur = 16;
  vochtigheid = 90;
  bewolkt = 80;
  omschrijving = 'bewolkt';
  afbeelding = 'http://openweathermap.org/img/w/03n.png';
  constructor() {}
}
```

component decorator

tag

Template component

weerinfo.component.html

```
<div class="weer">
  <div> {{datum|date:'MM/dd/yyyy'}} </div>
  <div> {{temperatuur|number:'1.0-2'}}°C </div>
  <div> vochtigheid {{vochtigheid}}% </div>
  <div> wolken {{bewolkt}}% </div>
  <div> {{omschrijving}} </div>
  <div> <img src="{{afbeelding}}"> </div>
</div>
```

Interpolation binding → data tonen uit component

Stylesheet component

weerinfo.component.css

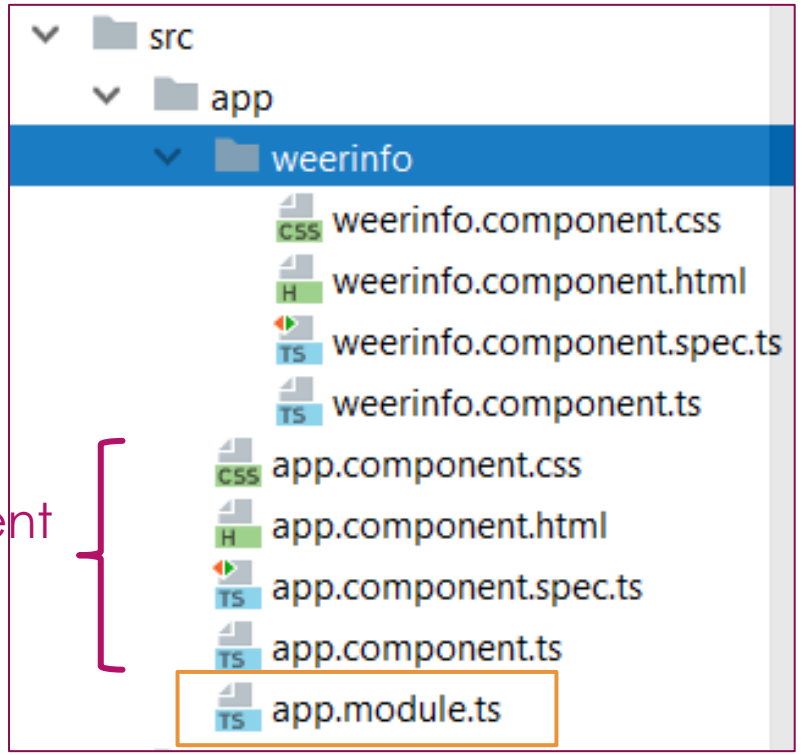
```
.weer {
  text-align: center;
  border: solid;
  border-color: gainsboro;
  border-width: 1px;
  border-radius: 5%;
}
```

Component gebruiken

```
<div class="container">
  …
  <div class="row">
    <div class="col-sm">
      …
    </div>
    <app-weerinfo class="col-sm-4">
    </app-weerinfo>
  </div>
</div>
```

app.component.html

root component
~<app-root>



root module

Component registreren

app.module.ts — root module

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { WeerinfoComponent } from './weerinfo/weerinfo.component';

@NgModule({
  declarations: [AppComponent, WeerinfoComponent],
  imports: [BrowserModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

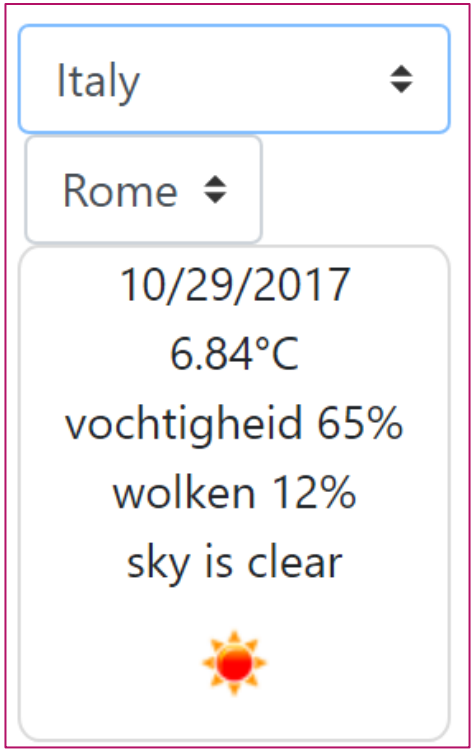
componenten
nodige functionaliteit
root component

index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Ngvoorbeelden</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet" href="https://.../bootstrap.min.css">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Angular - Overzicht

- Typescript
- Wat heb je nodig?
- Componenten
  - Structural directives: *ngFor en *ngIf
    - (her)vormen DOM-structuur
    - Toegepast op een host element
    - Beginnen met *

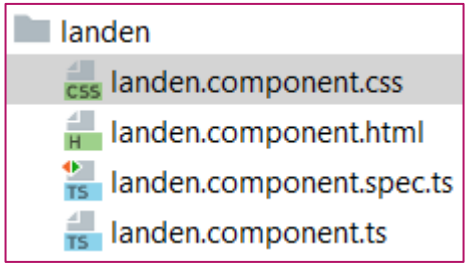


Klasse landencomponent

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

@Component({
  selector: 'app-landen',
  templateUrl: './landen.component.html',
  styleUrls: ['./landen.component.css']
})

export class LandenComponent implements OnInit {
  landen: string[];
  constructor() { }

  ngOnInit() {
    this.landen = ['Nederland', 'België'];
  }
}
```

Geen landen gevonden

Nederland ▾




```
<app-landen … >
</app-landen>
```

Template landencomponent

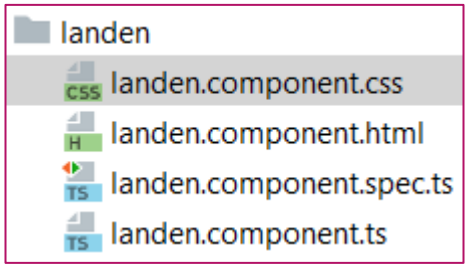
landen.component.html

```
<select
  *ngIf="landen !== undefined && landen.length != 0; else geenLanden">
  <option *ngFor="let land of landen">{{land}}</option>
</select>
<ng-template #geenLanden>
  <div>Geen landen gevonden</div>
</ng-template>
```

Host elementen: select, option

Geen landen gevonden

Nederland



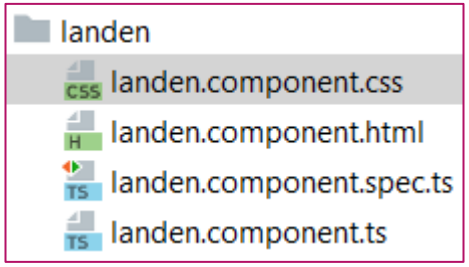
Alternatief

landen.component.html

```
<div *ngIf="landen !== undefined && landen.length != 0;
                          then wellanden else geenLanden"></div>
<ng-template #wellanden>
  <select>
    <option *ngFor="let land of landen">{{land}}</option>
  </select>
</ng-template>
<ng-template #geenLanden>
  <div>Geen landen gevonden</div>
</ng-template>
```

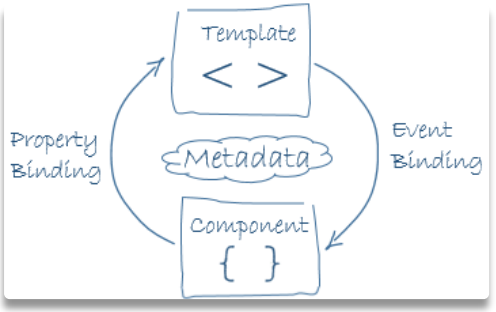
Geen landen gevonden

Nederland



Angular - Overzicht

- Typescript
- Wat heb je nodig?
- Componenten
  - Structural directives: *ngFor en *ngIf
  - One way Property binding
    - koppeling tussen
      - Eigenschap klasse
      - Attribuut element (_property DOM-element)



Component – oneway binding

HTML-template

```
<img [src]="afbeelding">
```

Klasse: attributen - methodes

```
@Component({
  selector: 'app-weerinfo',
  templateUrl: '….html',
  styleUrls: ['….css']
})
export class WeerinfoComponent {

  afbeelding = '...';

  …
}
```

Attributen tags ← Attributen klasse

Property Binding

- 3 mogelijkheden

weer-info.component.ts

```
export class WeerinfoComponent {
  afbeelding = '...';
}
```

weer-info.component.html

```
<img src="{{afbeelding}}">
```

```
<img [src]="afbeelding">
```

```
<img bind-src="afbeelding">
```

Voorbeeld

app.component.html

```
<button class="btn" [disabled]="nietgevonden">
  Toon weer
</button>
```

app.component.ts

```
…
export class AppComponent {
  nietgevonden = false;
  …
}
```

Class binding

template (.html)

```
<h1 [class]="titleClass">
  {{title}}
</h1>
```

klasse (.ts)

```
export class AppComponent {

  title = 'Hello!';
  titleClass = 'red-title';

}
```

Style binding

template (.html)

```
<button [style.color]="isSpecial ? 'red': 'green'">Red</button>
<button [style.background-color]="canSave ? 'cyan': 'grey’” >
  Save
</button>
```

klasse (.ts)

```
export class AppComponent {

  isSpecial = true;
  canSave = false;

}
```

NgClass

template (.html)

```
<h1 [ngClass]="titleClasses">
  {{title}}
</h1>
```

klasse (.ts)

```
export class AppComponent {

  titleClasses = {
    'red-title': true,
    'large-title': true
  }
}
```

NgStyle

template (.html)

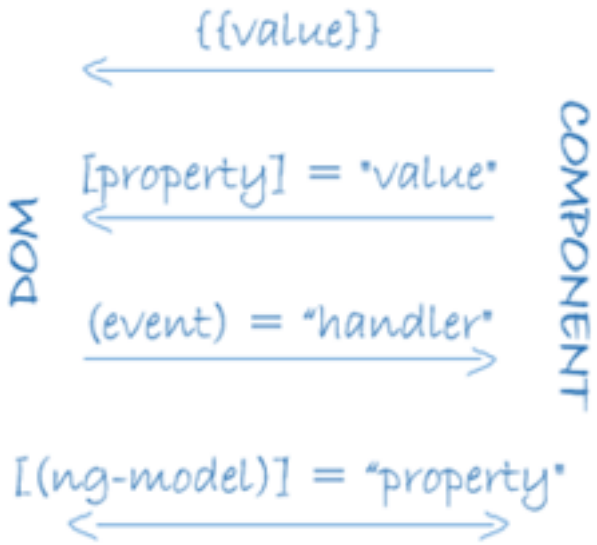
```
<h1 [ngStyle]="titleStyles">
  {{title}}
</h1>
```

klasse (.ts)

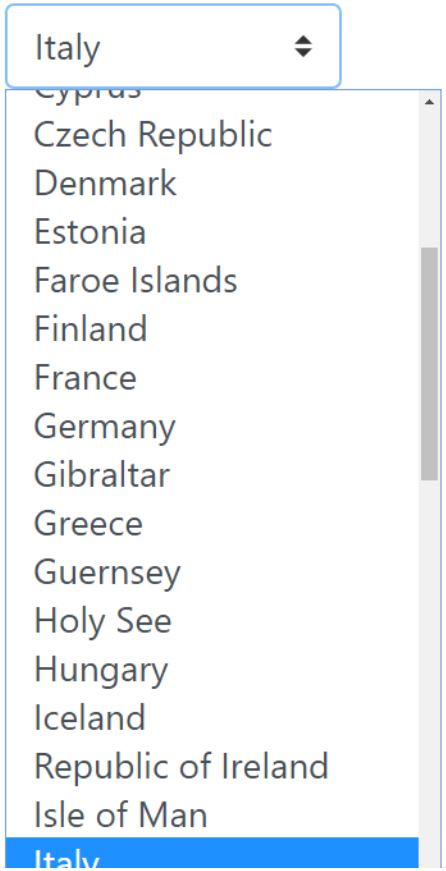
```
export class AppComponent {
  titleStyles = {
    'color': 'red',
    'font-size' : '4em'
  }
}
```

Angular - Overzicht

- Typescript
- Wat heb je nodig?
- Componenten
  - Structural directives: *ngFor en *ngIf
  - One way Property binding
  - Two-way binding
    - Attribuut template (.html) (~ eigenschap DOM-object)
    - Instantievariabele klasse (.ts)



Voorbeeld



landen.component.ts

```
export class LandenComponent implements OnInit
{
  selectedLand: Land | undefined;
  private _landen: Land[];
  get landen(): Land[] {return this._landen;}
}
```

landen.component.html

```
<select class="custom-select" [(ngModel)]="selectedLand"
(change)="veranderdLand()">
  <option *ngFor="let land of landen" [ngValue]="land">{{land.naam}}
  </option>
</select>
```

tweewegskoppeling

waarde voor optie → inhoud selectedLand

Component – tweewegsbinding

HTML-template

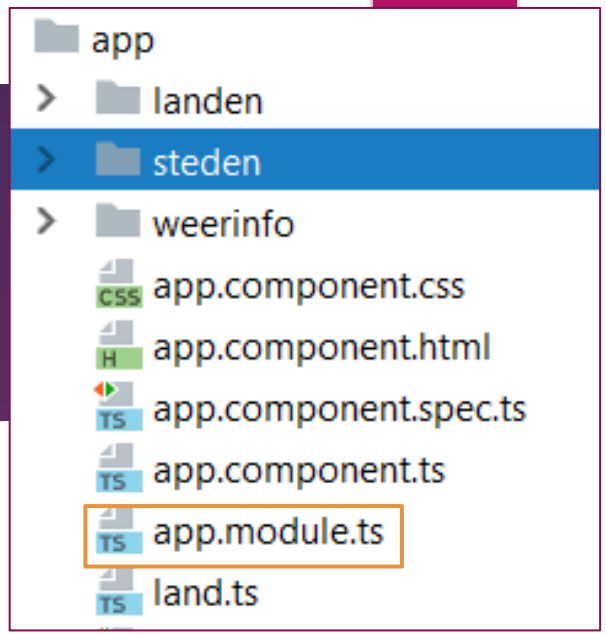
```
<select [(ngModel)]="selectedLand"
…">
  <option
    *ngFor="let land of landen“
    [ngValue]="land">
       {{land.naam}}
  </option>
</select>
```

Klasse: attributen - methodes

```
export class LandenComponent {

  selectedLand: Land;

  private _landen: Land[];

  get landen(): Land[] {
    return this._landen;
  }
}
```

Attributen tags ⟷ Attributen klasse

Tweewegsbinding



root module

- [(ngModel)]
  - Tweewegsbinding
  - Property/Field ↔ waarde HTML-component
- Behoort tot de module FormsModule
  - Importeren

app.module.ts

```
import {FormsModule } from '@angular/forms';
…
@NgModule({
  declarations: [AppComponent,   WeerinfoComponent,
    LandenComponent, StedenComponent],
  imports: [BrowserModule, FormsModule],
  providers: [],
  bootstrap: [AppComponent]
})
```

Angular - Overzicht

- Typescript
- Wat heb je nodig?
- Componenten
  - Structural directives: *ngFor en *ngIf
  - One way Property binding
  - Two-way binding
  - Events
    - Methode registreren bij event

Component – events

HTML-template

```
<select (change)="veranderdLand()"
… >
  …
</select>
```

Klasse: attributen - methodes

```
export class LandenComponent {

  veranderdLand() {
    …
  }
}
```

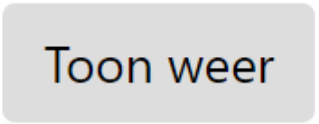
Event tag ⟷ Methode klasse

Events

app.component.html

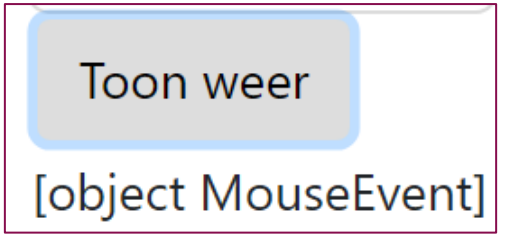
```
<button class="btn" (click)="toonClick($event)">
  Toon weer
</button>
```

event handler

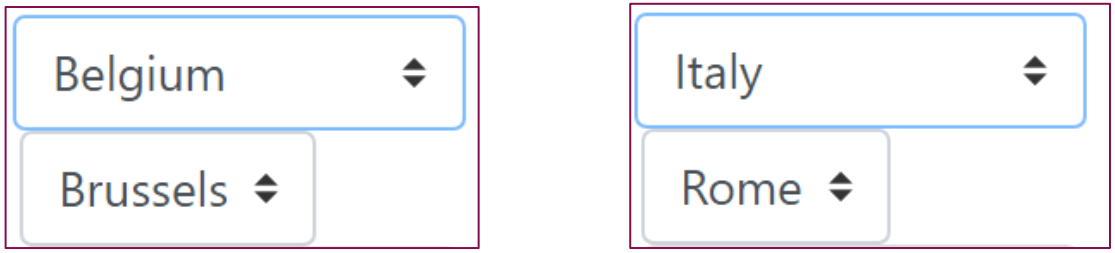


app.component.ts

```
…
toonClick(event) {
  this.tekst = event.toString();
}
```



Events



landen.component.html

event handler

```
<select class="custom-select" … (change)="veranderdLand()">
  …
</select>
```

```
<li *ngFor="let hero of heroes" (click)="onSelect(hero)">
  ...
</li>
```

lokale variabele

Binding types

Data direction	Syntax	Type

One-way
from data source
to view target
	`{{expression}}`
	`[target]="expression"`
	`bind-target="expression"`
		Interpolation
		Property
		Attribute
		Class
		Style

One-way from view target to data source	`(target)="statement"` `on-target="statement"`	Event
Two-way	`[(target)]="expression"` `bindon-target="expression"`	Two-way

https://angular.io/guide/template-syntax#binding-syntax-an-overview

Overzicht

- Typescript
- Wat heb je nodig?
- Componenten
  - Structural directives: *ngFor en *ngIf
  - One way Property binding
  - Two-way binding
  - Events
  - Attributen toevoegen

Component: @Input() – attributen toevoegen

Tag - component

```
<app-landen landen="…"> </app-landen>
```

Klasse - component

```
export class LandenComponent implements OnChanges{

  @Input()
  landen: Land[] = [];

  ngOnChanges(changes: SimpleChanges): void {
    if (this.landen !== [] && this.selectedLand == undefined ){
      this.selectedLand = this.landen[0];
    }
  }
}
```

@Input()

app.component.html

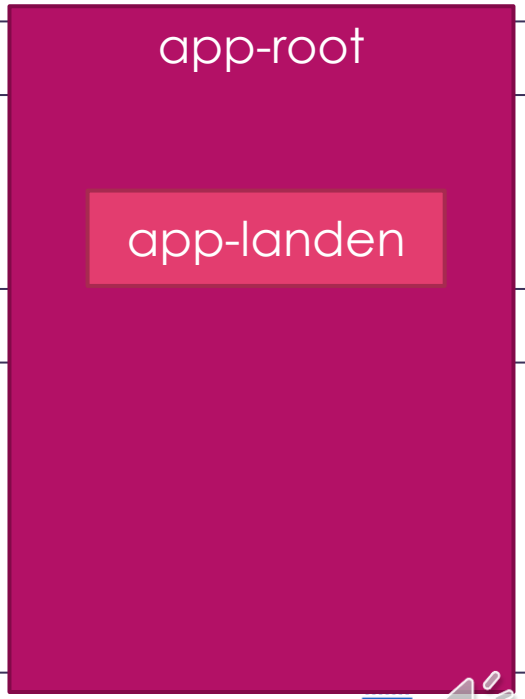
```
<app-landen [landen]="landen" (landChanged)="veranderLand($event)">
</app-landen>
```

app.component.ts

```
export class AppComponent {
  landen: Land[];
}
```

landen.component.ts

```
export class LandenComponent implements OnInit {
  private _landen: Land[];
  @Input()
  landen: Land[] = [];
}
```



Overzicht

- Typescript
- Wat heb je nodig?
- Componenten
  - Structural directives: *ngFor en *ngIf
  - One way Property binding
  - Two-way binding
  - Events
  - Attributen toevoegen
  - Events toevoegen

Component: @Output() – event toevoegen

Tag - component

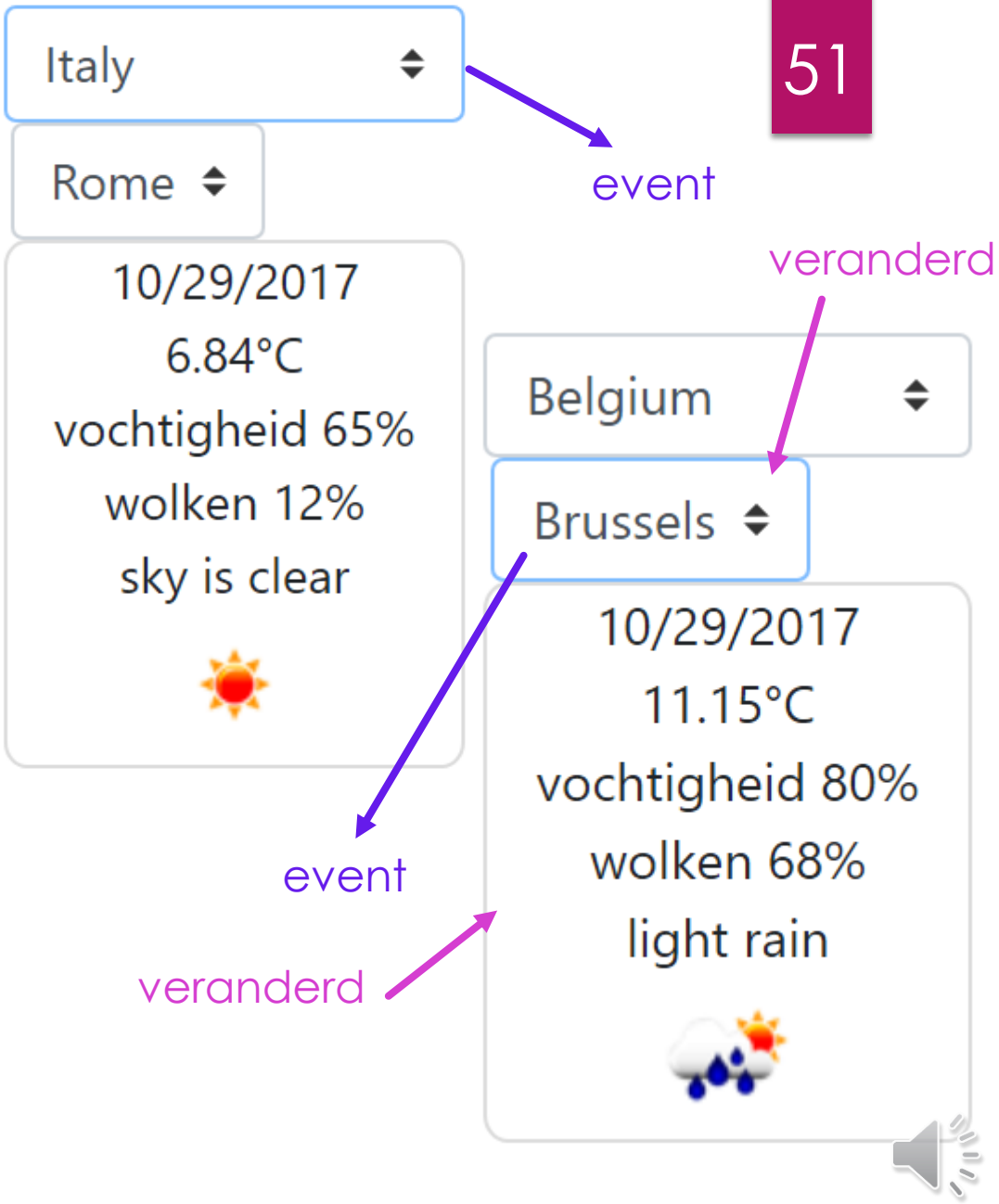
```
<app-landen (landChanged)="…"> </app-landen>
```

Klasse - component

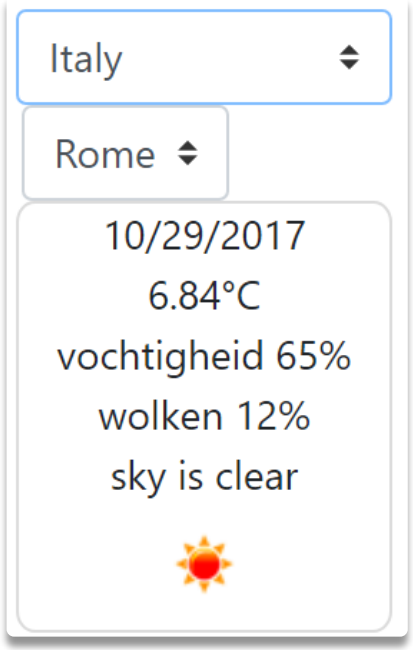
```
export class LandenComponent {

  selectedLand: Land;

  @Output() landChanged = new EventEmitter<Land>();
  veranderdLand() {
    this.landChanged.emit(this.selectedLand);
  }
}
```

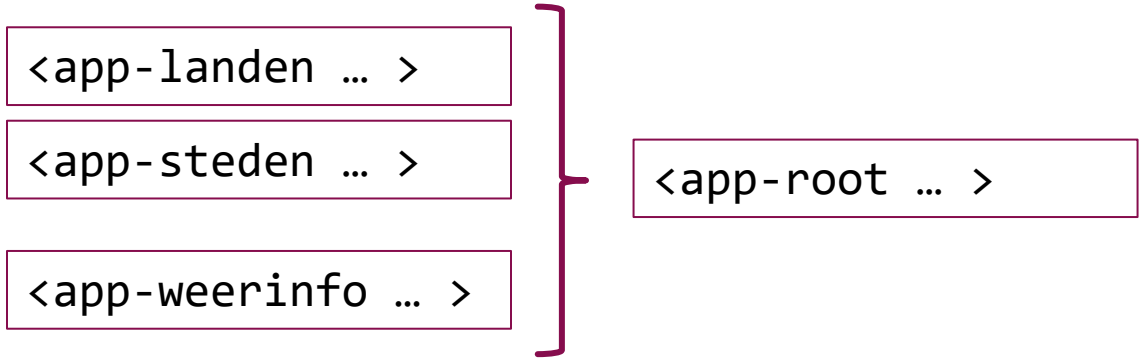
Zelf events definiëren

- Event Emitter in componentklasse
  - Declareren
  - Event uitsturen
- Registreren voor event



Zelf events definiëren





- Component waar event optreedt
  - Event voorzien
- Component die naar event luistert
  - Handler voorzien

Landen – event toevoegen

landen.component.html

```
<select [(ngModel)]="selectedLand" (change)="veranderdLand()">
  <option *ngFor="let land of landen" [ngValue]="land">
    {{land.naam}}
  </option>
</select>
```

landen.component.ts

```
import {…, Output, EventEmitter} from '@angular/core';
export class LandenComponent implements OnInit {
  selectedLand: Land;
  @Output() landChanged = new EventEmitter<Land>();
  veranderdLand() {
    this.landChanged.emit(this.selectedLand);
  }
```

Landen – handler toevoegen

landen.component.ts

```
import {…, Output, EventEmitter} from '@angular/core’;
export class LandenComponent implements OnInit {
  selectedLand: Land;
  @Output() landChanged = new EventEmitter<Land>();
  veranderdLand() {
    this.landChanged.emit(this.selectedLand);
  }
```

app.component.html

```
<app-landen [landen]="landen" (landChanged)="veranderLand($event)">
</app-landen>
```

Landen - handler

app.component.html

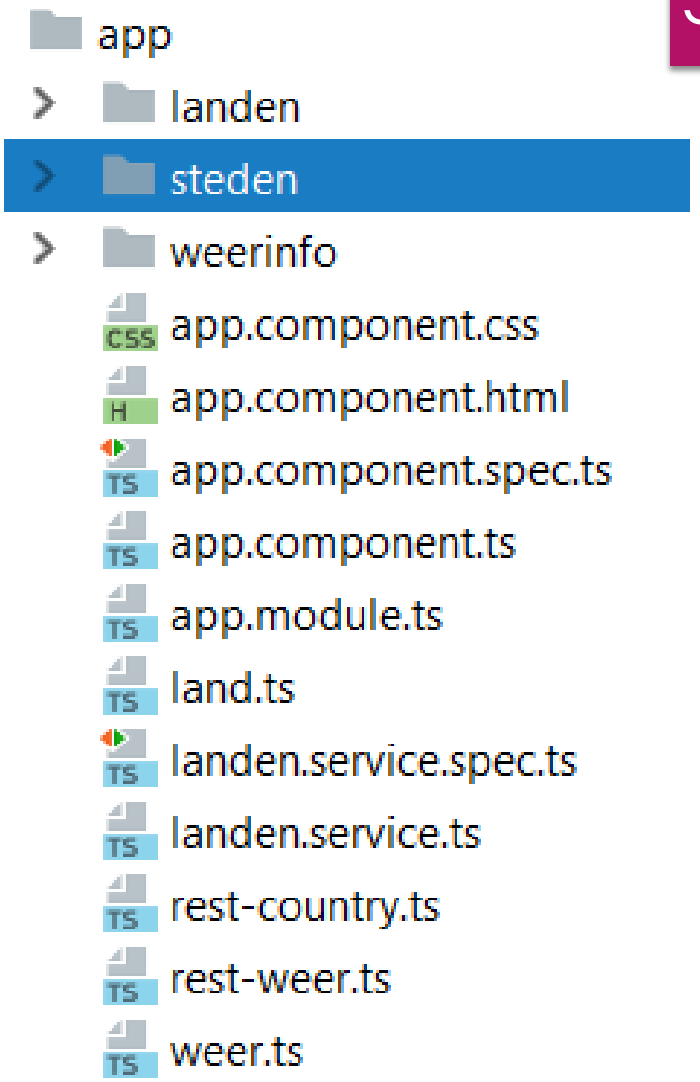
```
<app-landen [landen]="landen" (landChanged)="veranderLand($event)">
</app-landen>
```

app.component.ts

```
export class AppComponent {
  steden: string[];
  land: Land;
  constructor(private landenService: LandenService) {
    …
  }

  veranderLand(land: Land) {
    this.land = land;
    this.landenService.haalSteden(land)
      .then(steden => this.steden = steden);
  }
```

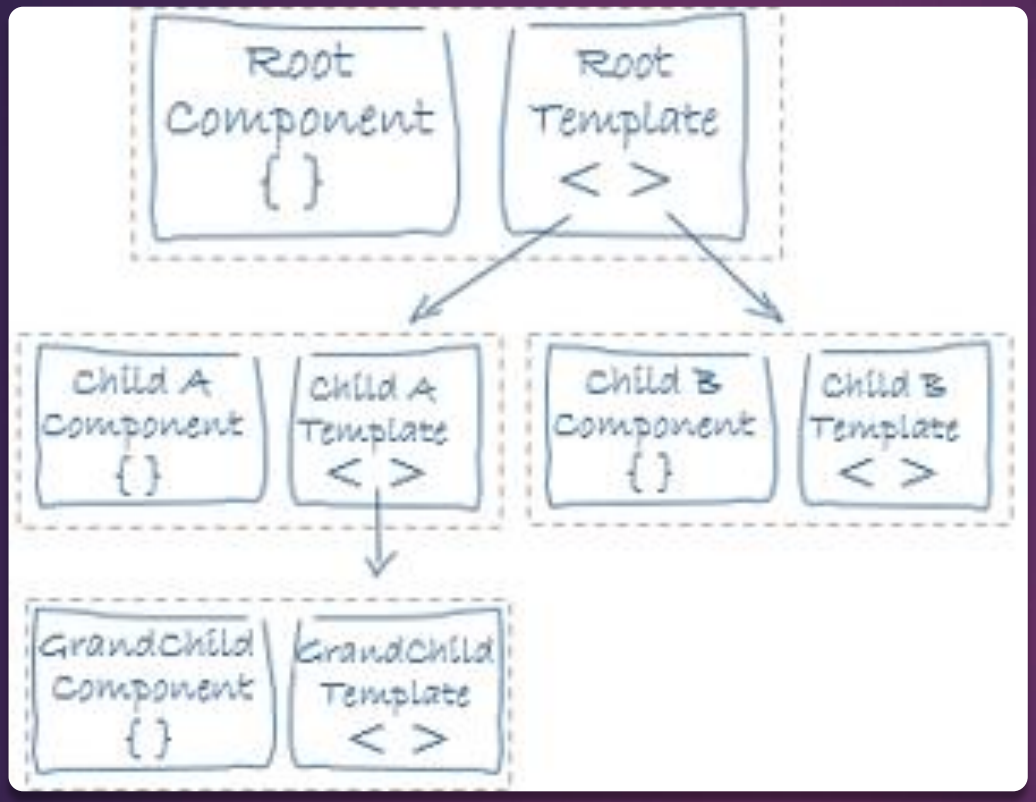
Steden – Weer

- Analoog
- Zie project



Overzicht

- Typescript
- Wat heb je nodig?
- Componenten
  - Structural directives: *ngFor en *ngIf
  - One way Property binding
  - Two-way binding
  - Events
  - Attributen toevoegen
  - Events toevoegen
  - Oudercomponenten

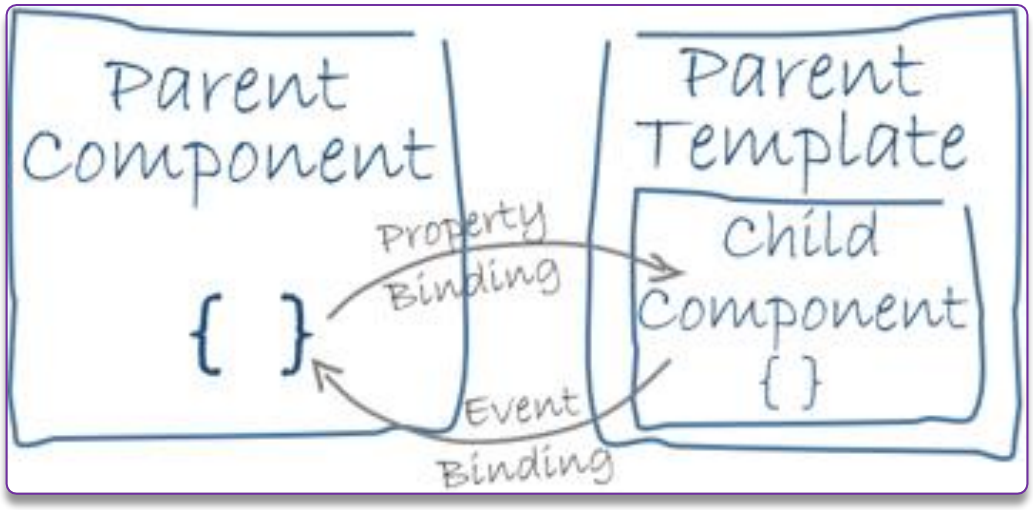


Componenten gebruiken in andere componenten

Component met attribuut en event gebruiken in andere component

Tag - component

```
<app-landen landen="…" (landChanged)="…"> </app-landen>
```



app-landen

app-root

Component met attribuut en event gebruiken in andere component

Tag - component

```
<app-landen landen="…" (landChanged)="…"> </app-landen>
```

HTML-template (app-root)

```
<app-landen [landen]="landen"
(landChanged)="veranderLand($event)"
>
</app-landen>
```

Klasse (app-root)

```
export class AppComponent {

  landen: Land[];

  veranderLand(land: Land) {
    …
  }
}
```

Event, attribuut tag ⟷ Attribuut, methode klasse

Overzicht

- Typescript
- Wat heb je nodig?
- Componenten
- Services

Service

- Biedt diensten aan
  - Data ophalen
  - Logging
  - Berekeningen
  - …
- Wordt automatisch geïnjecteerd in componenten
- Kan ook asynchroon

Service - overzicht

Klasse component

```
export class AppComponent
{
  constructor(private landenService: LandenService) {}

  …
}
```

Klasse service

```
@Injectable()
export class LandenService {

  haalLanden(): Land[] {…}

}
```

Kan geïnjecteerd worden in andere klassen

Service aanmaken en registreren

app.module.ts

```
import {LandenService} from './landen.service';
@NgModule({
  declarations: [AppComponent, WeerinfoComponent,
    LandenComponent, StedenComponent],
  imports: [BrowserModule, FormsModule],
  providers: [LandenService],
  bootstrap: [AppComponent]
})
```

landen.service.ts

```
import {Injectable} from '@angular/core';
@Injectable()
export class LandenService {

}
```

Kan geïnjecteerd worden in andere klassen

Service implementeren

landen.service.ts

```
import {Injectable} from '@angular/core’;
import {Land} from './land’;

@Injectable()
export class LandenService {

  haalLanden(): Land[] {
    let landen =
      [{code: 'BE', naam: 'België', hoofdstad: 'Brussel'},
       {code: 'NL', naam: 'Nederland', hoofdstad: 'Den Haag'}];
    return landen;
  }
}
```

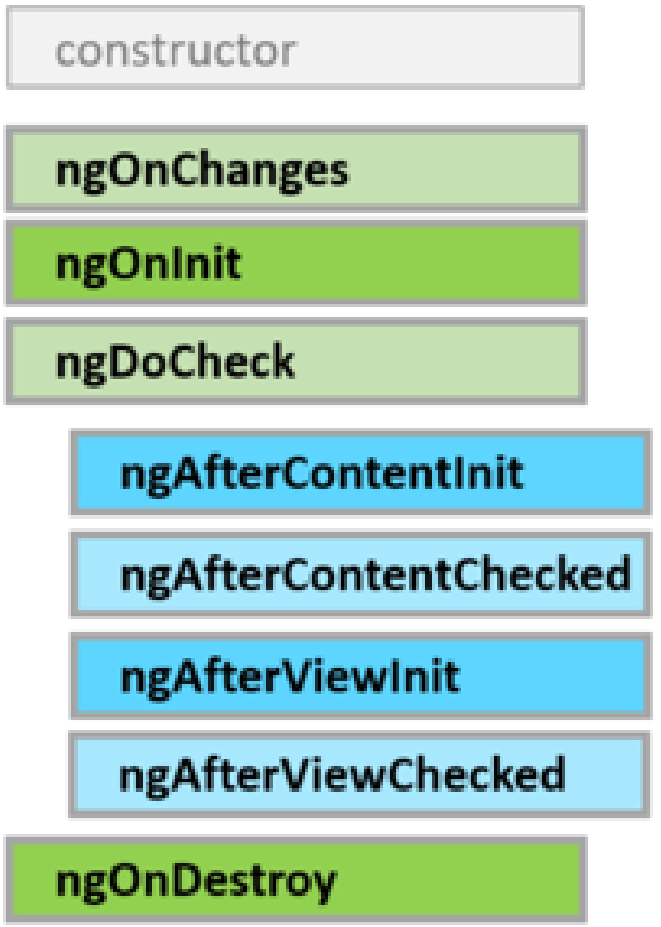
Service injecteren en gebruiken

app.component.ts

```
import {LandenService} from './landen.service';
import {Land} from './land';

export class AppComponent implements OnInit {
  landen: Land[];

  constructor(private landenService: LandenService) {}

  ngOnInit(): void {
    this.landen = this.landenService.haalLanden();
  }
}
```

Lifecycle Hooks

- Angular beheert levensloop component
- Hooks
  - Acties na uitvoeren van een bepaalde faze



https://angular.io/guide/lifecycle-hooks

Asynchrone service

landen.service.ts

```
import {Injectable} from '@angular/core’;
import {Land} from './land’;

@Injectable()
export class LandenService {

  haalLanden(): Promise<Land[]> {
    let landen =
      [{code: 'BE', naam: 'België', hoofdstad: 'Brussel'},
       {code: 'NL', naam: 'Nederland', hoofdstad: 'Den Haag'}];
    return Promise.resolve(landen);
  }
}
```

Asynchrone service gebruiken

app.component.ts

```
import {LandenService} from './landen.service';
import {Land} from './land';

export class AppComponent implements OnInit {
  landen: Land[];

  constructor(private landenService: LandenService) {}

  ngOnInit(): void {
    this.landenService.haalLanden()
      .then(landen => this.landen = landen);
  }
}
```

Angular - Overzicht

- Typescript
- Wat heb je nodig?
- Componenten
- Services
- HTTP-Services

HTTP-service

- Data van een server halen
  - AJAX-call
- Gebruikt HttpClientModule

Klasse service

```
@Injectable()
export class LandenService {

  constructor(private http: HttpClient) {
  }

}
```

Gebruik HTTP-module

app.module.ts

```
import {HttpClientModule} from '@angular/common/http';

@NgModule({
  declarations: [AppComponent, WeerinfoComponent,
    LandenComponent, StedenComponent],
  imports: [BrowserModule, FormsModule, HttpClientModule],
  providers: [LandenService],
  bootstrap: [AppComponent]
})
```

HTTP-service

landen.service.ts

```
import {HttpClient} from '@angular/common/http';
import {RestCountry} from './rest-country';

@Injectable()
export class LandenService {

  constructor(private http: HttpClient) {
  }

}
```

Interface REST-resultaat

rest-country.ts

```
export interface RestCountry {
  alpha2Code: string;
  name: string;
  capital: string;
}
```

https://restcountries.eu/rest/v1/region/europe.ts

[{"name":"Åland Islands","topLevelDomain":[".ax"],"alpha2Code":"AX","alpha3Code":"ALA","callingCodes":["358"],"capital":"Mariehamn","altSpellings":
["AX","Aaland","Aland","Ahvenanmaa"],"region":"Europe","subregion":"Northern Europe","population":28875,"latlng":
[60.116667,19.9],"demonym":"Ålandish","area":1580.0,"gini":null,"timezones":["UTC+02:00"],"borders":[],"nativeName":"Åland","numericCode":"248","currencies":
["EUR"],"languages":["sv"],"translations":{"de":"Åland","es":"Alandia","fr":"Åland","ja":"オーランド諸島","it":"Isole Aland"},"relevance":"0"},
{"name":"Albania","topLevelDomain":[".al"],"alpha2Code":"AL","alpha3Code":"ALB","callingCodes":["355"],"capital":"Tirana","altSpellings":
["AL","Shqipëri","Shqipëria","Shqipnia"],"region":"Europe","subregion":"Southern Europe","population":2893005,"latlng":
[41.0,20.0],"demonym":"Albanian","area":28748.0,"gini":34.5,"timezones":["UTC+01:00"],"borders":
["MNE","GRC","MKD","KOS"],"nativeName":"Shqipëria","numericCode":"008","currencies":["ALL"],"languages":["sq"],"translations":
{"de":"Albanien","es":"Albania","fr":"Albanie","ja":"アルバニア","it":"Albania"},"relevance":"0"},{"name":"Andorra","topLevelDomain":

HTTP-service: landen ophalen

landen.service.ts

```
get landen(): Observable<Land[]> {
    return this.http.get<RestCountry[]>(this.landenURL)
    .pipe(
      tap(_ => console.log('fetched landen')),
      map(items => items.map(
        item => new Land(item.alpha2Code, item.name,item.capital)))
    );}
```

GET

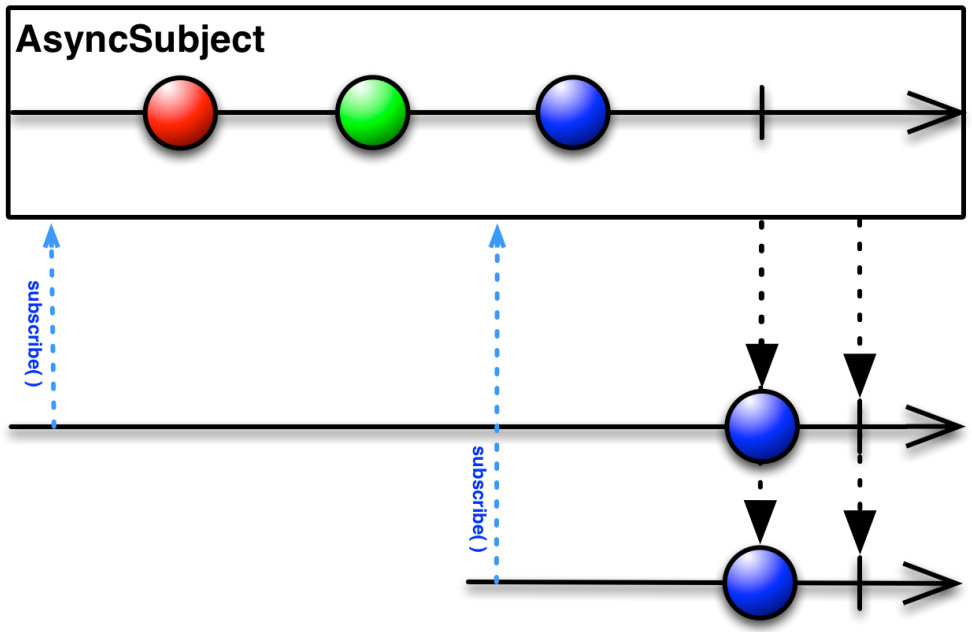
- http.get → Observable (meerdere asynchrone resultaten)
- RestCountry[] → Land[]

Resultaat methode-oproep

	Synchroon	Asynchroon
Eén resulaat	T	Promise<T>
Veel resultaten	T[]	Observable<T>

Observable

- Events
- Asynchrone methodes met meerdere resultaten



HTTP-service: landen ophalen

landen.service.ts

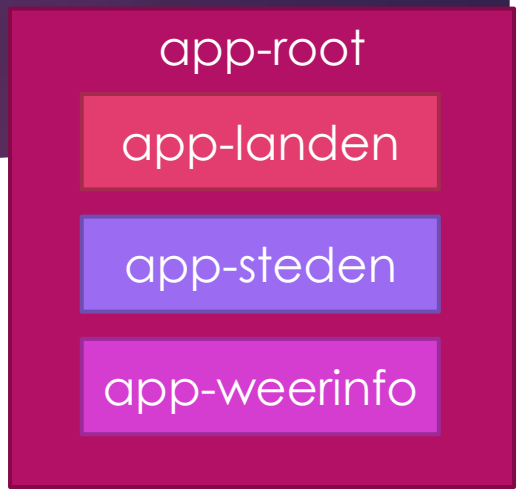
```
get landen(): Observable<Land[]> {
    return this.http.get<RestCountry[]>(this.landenURL)
    .pipe(
      tap(_ => console.log('fetched landen')),
      map(items => items.map(
        item => new Land(item.alpha2Code, item.name,item.capital)))
    );}
```

Asynchrone HTTP-service gebruiken

app.component.ts

```
ngOnInit(): void {
   this.landenService.landen.subscribe((landen) => {
     this.landen = landen;
     this.land = landen[0];
     this.landenService.haalSteden(this.land)
       .subscribe(steden => this.steden = steden);
  }
```

Weer tonen



- Steden instellen → stad geselecteerd
- Stad selecteren → weer aanpassen

app.component.html

```
<app-steden [steden]="steden" (stadChanged)="toonWeer($event)">
</app-steden>
```

app.component.ts

```
toonWeer(stad: string): void {
    this.landenService.haalWeer(stad)
        .subscribe(weer => this.weer = weer);
    this.stad = stad;
}
```

Overzicht

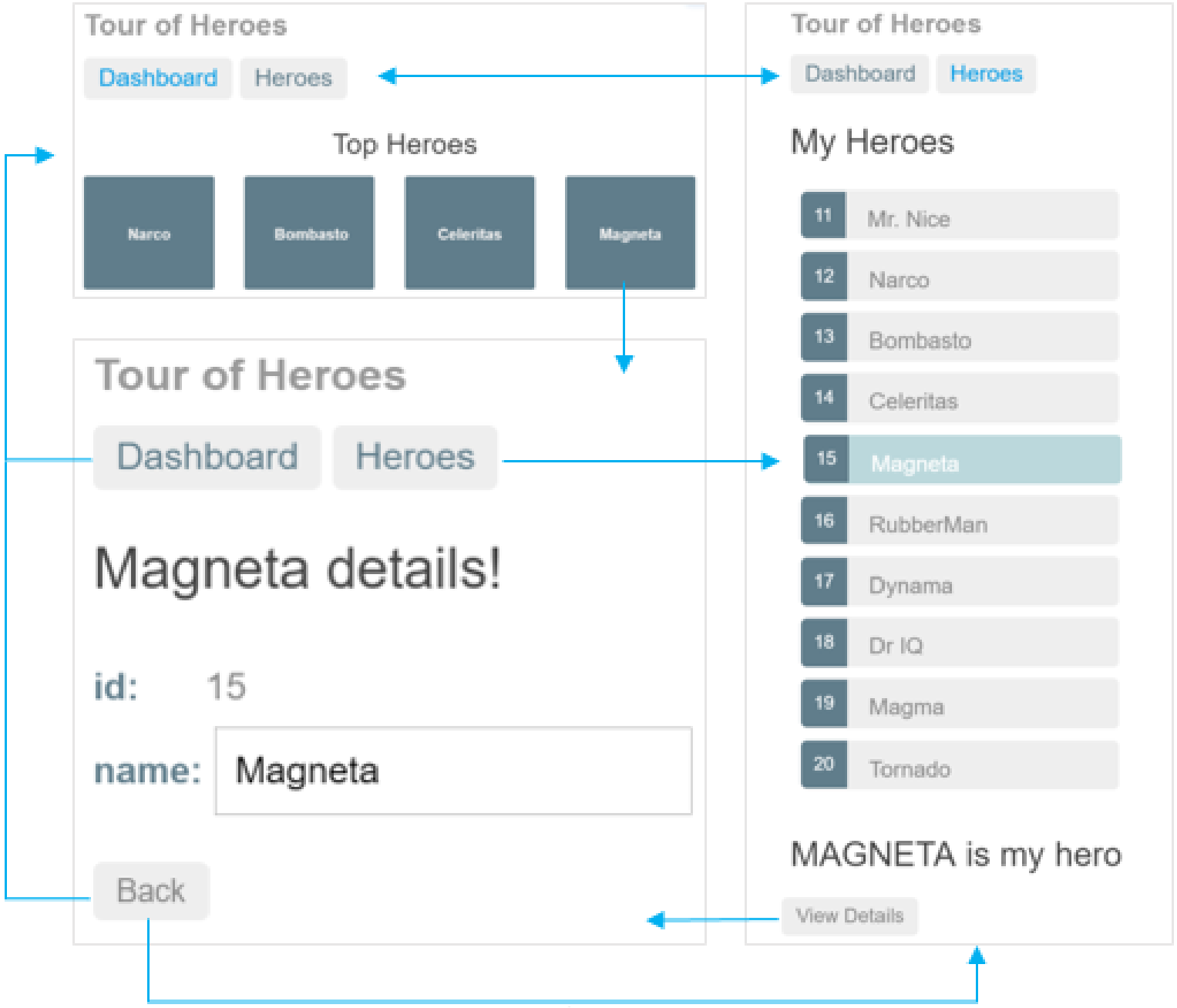
- Typescript
- Wat heb je nodig?
- Componenten
- Services
- HTTP-Services
- Configuratie

Configuratie – root module

```
import ...
@NgModule({
  declarations: [
    AppComponent,
    WeerinfoComponent,
    LandenComponent,
    StedenComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule
  ],
  providers: [LandenService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Overzicht

- Typescript
- Wat heb je nodig?
- Componenten
- Services
- HTTP-Services
- Configuratie
- Routing


Tour of Heroes
Dashboard
Heroes
Top Heroes
Narco
Bombasto
Celeritas
Magneta
Tour of Heroes
Dashboard
Heroes
My Heroes
11 Mr. Nice
12 Narco
13 Bombasto
14 Celeritas
15 Magneta
16 RubberMan
17 Dynama
18 Dr IQ
19 Magma
20 Tornado
MAGNETA is my hero
View Details
Tour of Heroes
Dashboard
Heroes
Magneta details!
id: 15
name: Magneta
Back

Basis-URL

- In index.html

```
<base href="/">
```

Routes configureren

- In aparte module
- Pad ↔ Component
- Route-configuratie inlezen
- Routes exporteren → beschikbaar voor de applicatie

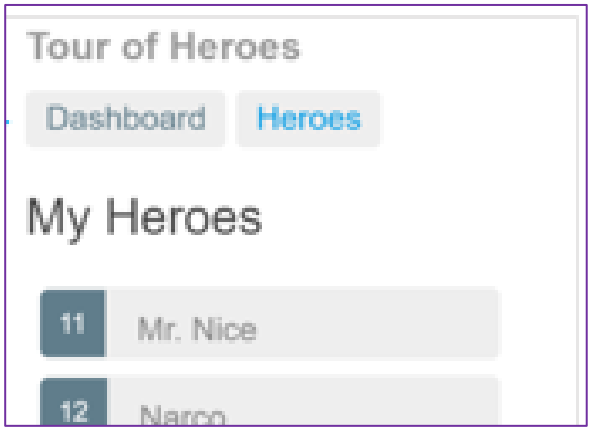
Routes configureren

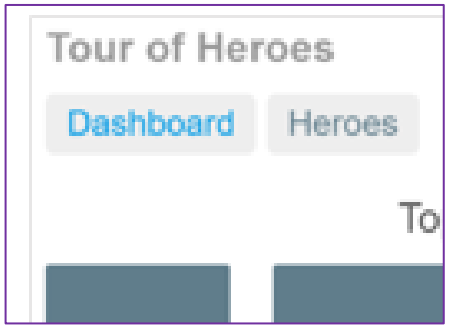
```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HeroesComponent } from './heroes/heroes.component';

const routes: Routes = [
  { path: 'heroes', component: HeroesComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

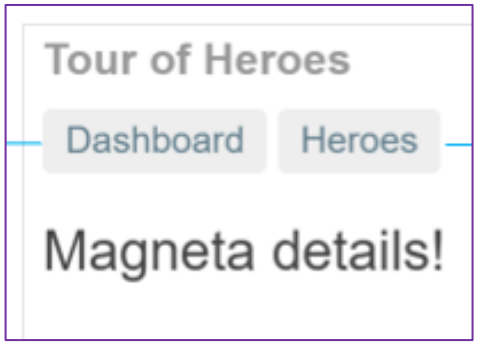
Waar component tonen?

- routerLink: naar waar?
- router-outlet: waar tonen?

```
<h1>{{title}}</h1>
<a routerLink="/heroes">Heroes</a>
<router-outlet></router-outlet>
```







Routes met parameter

- In module met routerconfiguratie
- Pad ↔ Component

```
{
  path: 'detail/:id',
  component: HeroDetailComponent
}
```

```
<a *ngFor="let hero of heroes"
  routerLink="/detail/{{hero.id}}">
```

Parameter identificeren (HeroDetailComponent)

```
constructor(
  private heroService: HeroService,
  private route: ActivatedRoute,
  private location: Location
) {}
```

serviceklasse

info route

info browser

route-info na init

parameter ophalen

```
ngOnInit(): void {
  const id = +this.route.snapshot.paramMap.get('id');
  this.heroService.getHero(id)
    .subscribe(hero => this.hero = hero);}
```

string → int

Terug naar vorige pagina (HeroDetailComponent)

```
constructor(
  private heroService: HeroService,
  private route: ActivatedRoute,
  private location: Location
) {}
```

serviceklasse

info route

info browser

```
goBack(): void {
  this.location.back();
}
```

"back" browser

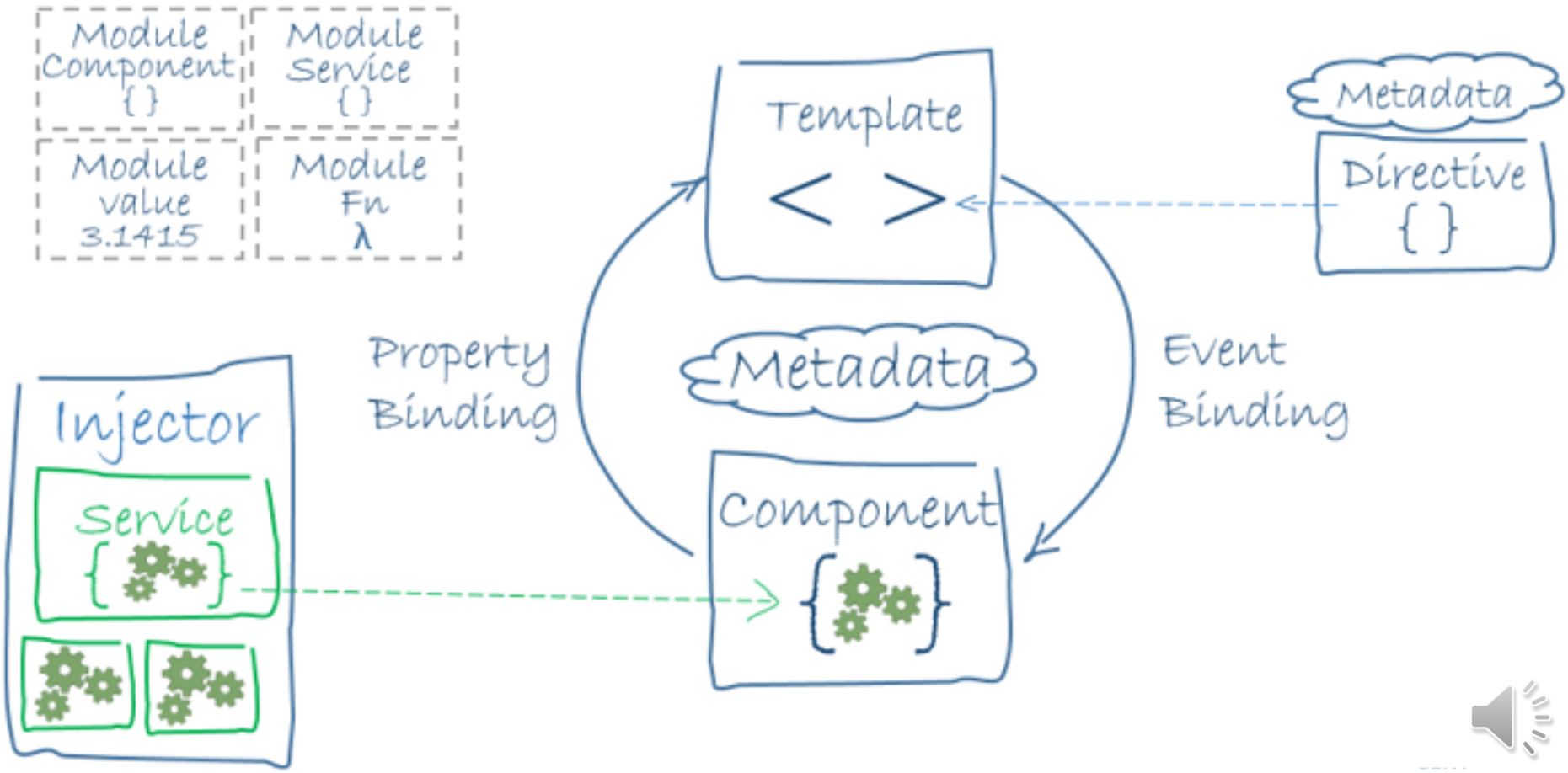
```
<button (click)="goBack()">
go back
</button>
```

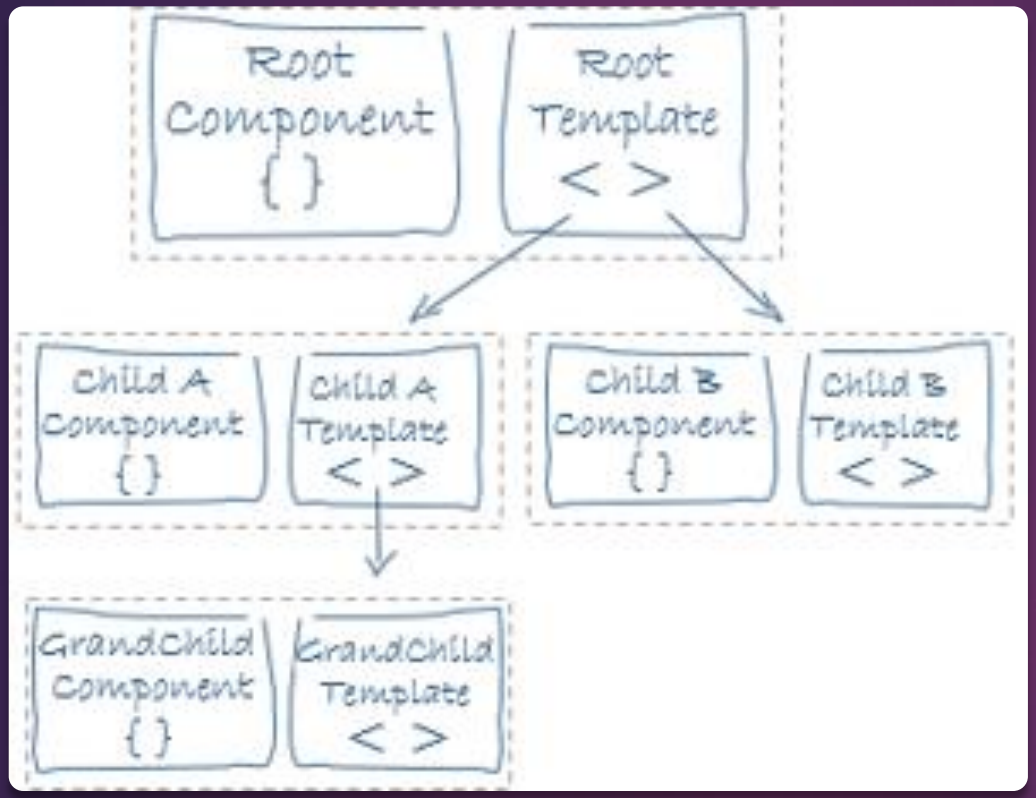
HTML-component

Overzicht

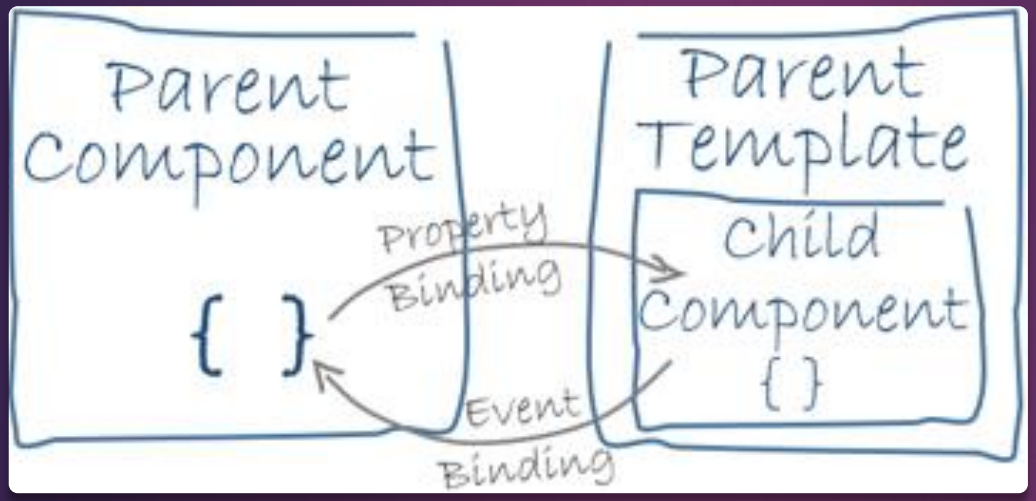
- Typescript
- Wat heb je nodig?
- Componenten
- Services
- HTTP-Services
- Configuratie
- Routing
- Architectuur

Architectuur



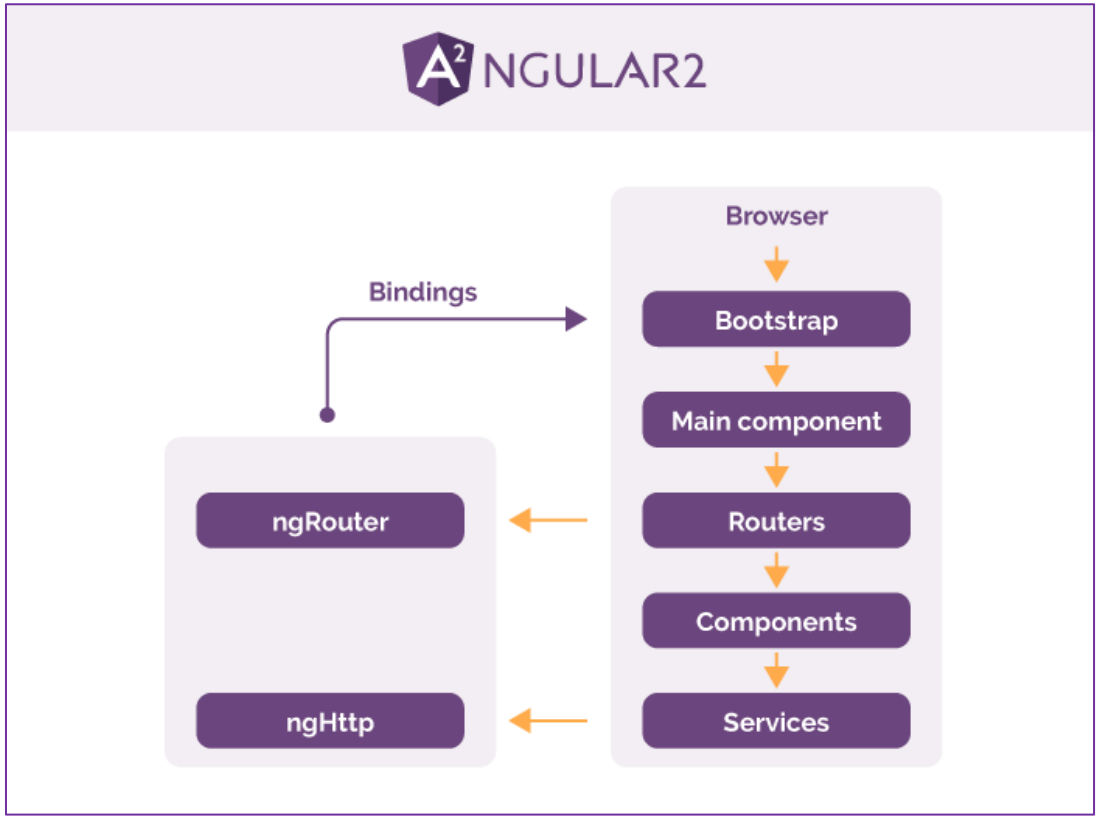


Kindcomponenten



Kindcomponenten

Overzicht Angular



https://rubygarage.org/blog/best-javascript-frameworks-for-front-end