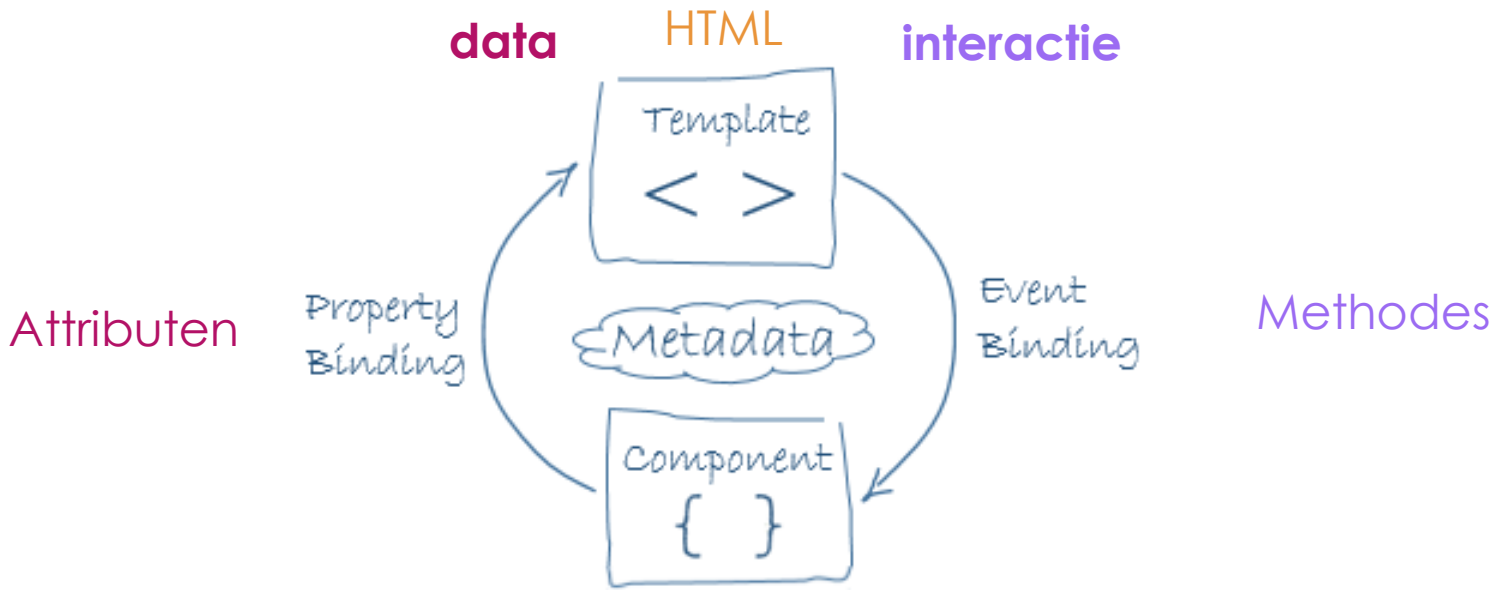
Op vraag van studenten: deze slides werden gebruikt in de gestreamde les van 15 maart 2022. Kleine wijzigingen t.o.v. de oorspronkelijke slides:

- de eerste drie slides zijn toegevoegd (herhaling van de vorige les)
- slides 46b, 46c en 60b zijn toegevoegd (herhalingen van voorgaande slides)
- aan slides 61 en 70 werd een kadertje toegevoegd

Angular

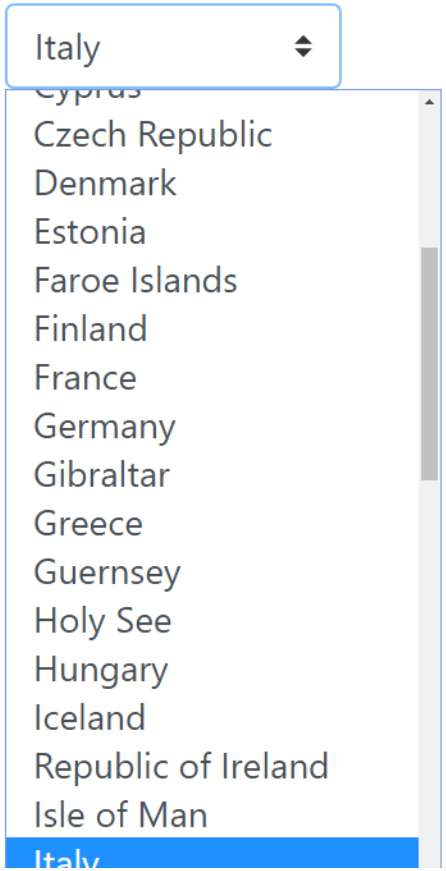
- Framework om vlotter webapplicaties te bouwen/onderhouden
- Gebruikt TypeScript
- EEN html-pagina met de look&feel van meerdere pagina's via
- Componenten
  - html-template presentatie
  - css-code opmaak
  - ts-klasse instantievariabelen die data bevatten
    methodes als event handlers

Wat toont/kan een webpagina?



Klasse: attributen - methodes

Voorbeeld



landen.component.ts

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

landen.component.html

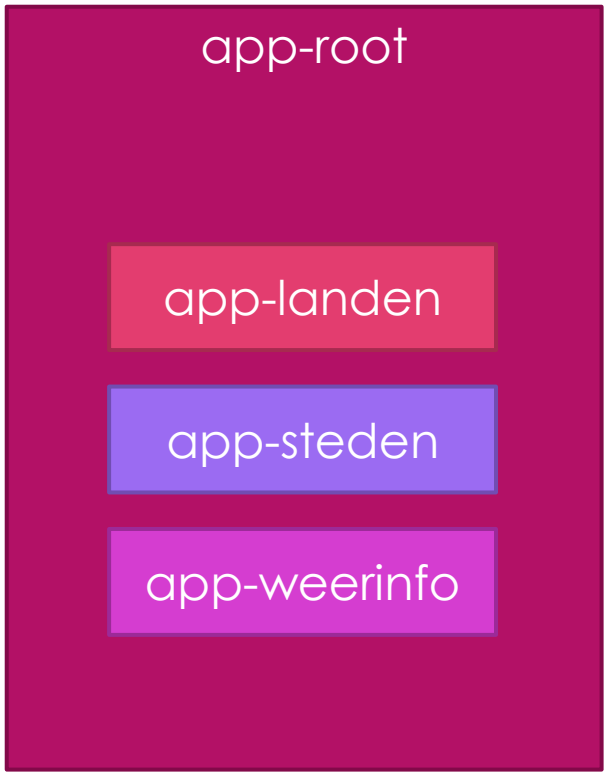
```
<select class="custom-select" [(ngModel)]="selectedLand"
(change)="veranderdLand()">

  <option *ngFor="let land of landen" [ngValue]="land">{{land.naam}}
  </option>
</select>
```

Overzicht

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

Data doorgeven

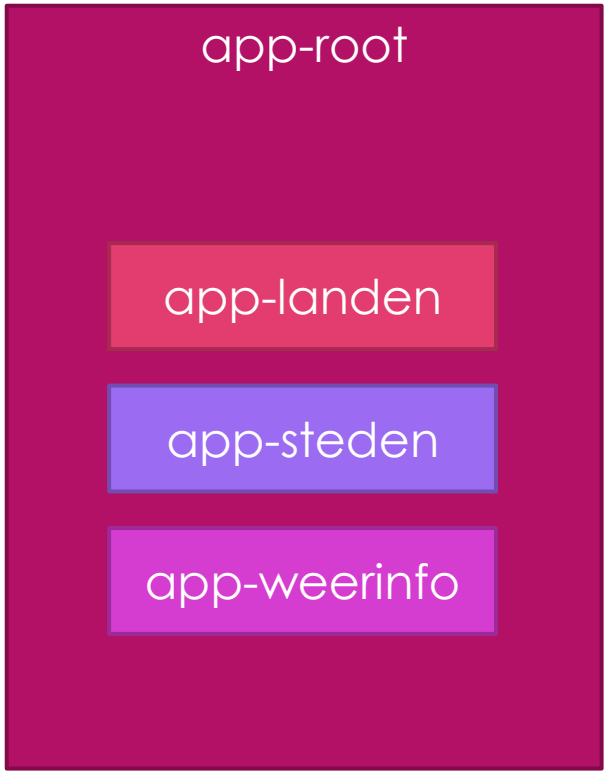
- Typescript
- Wat heb je nodig?
- Componenten
  - Structural directives: *ngFor en *ngIf
  - One way Property binding
  - Two-way binding
  - Events
  - Attributen toevoegen



Data doorgeven

app.component.html

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



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

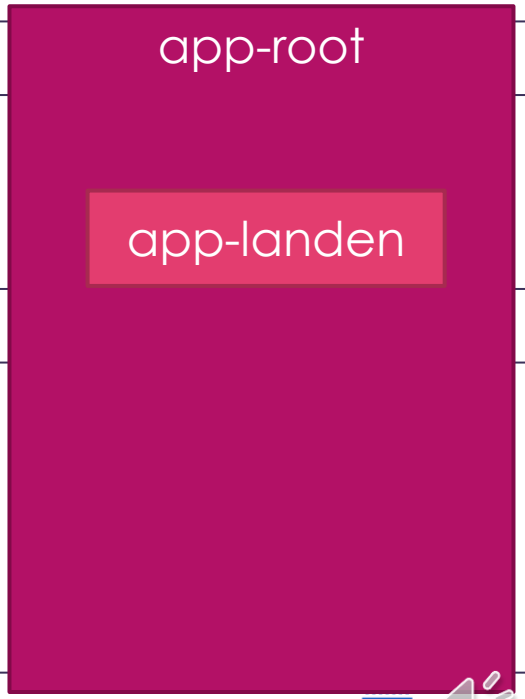
```
<app-landen [landen]="landen" (landChanged)="veranderLand($event)">
</app-landen>
```

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

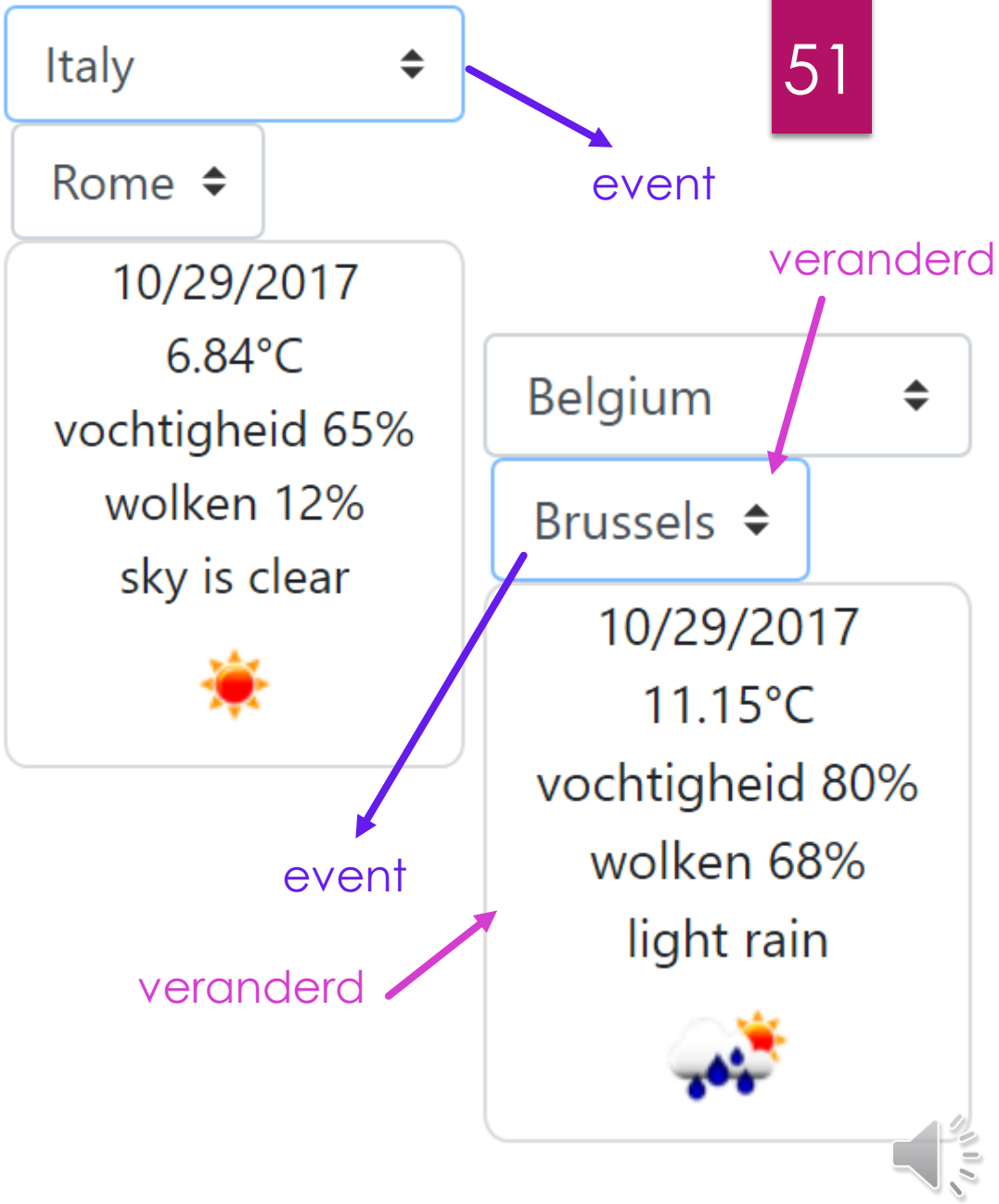
```
<app-landen (landChanged)="…"> </app-landen>
```

Klasse - component

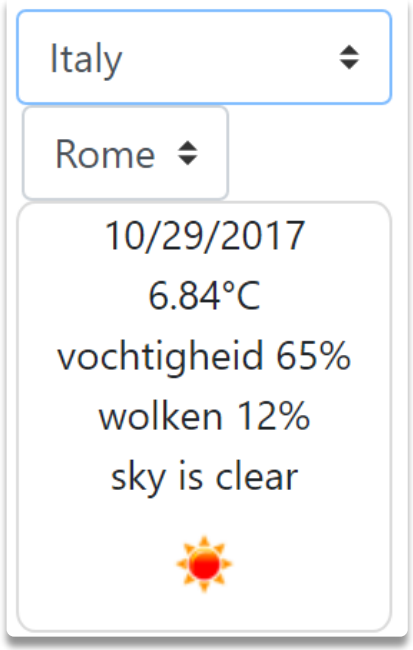
```
export class LandenComponent {

  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



<app-landen … >

<app-steden … >

<app-weerinfo … >

<app-root … >

- 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

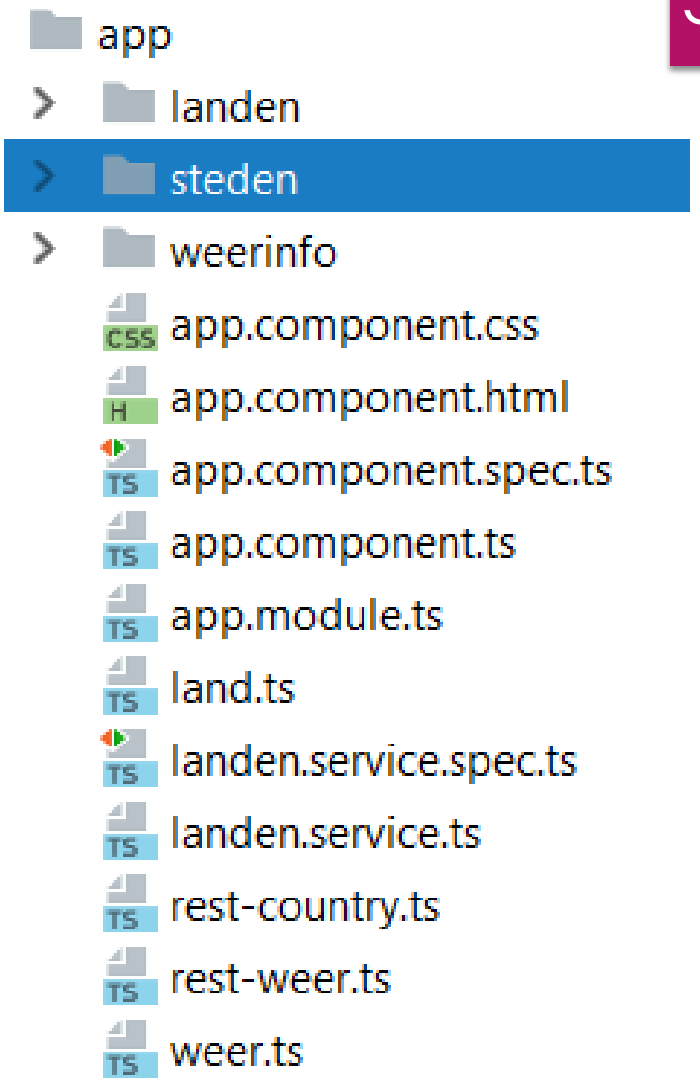
```
<app-landen [landen]="landen" (landChanged)="veranderLand($event)">
</app-landen>
```

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);
  }
```

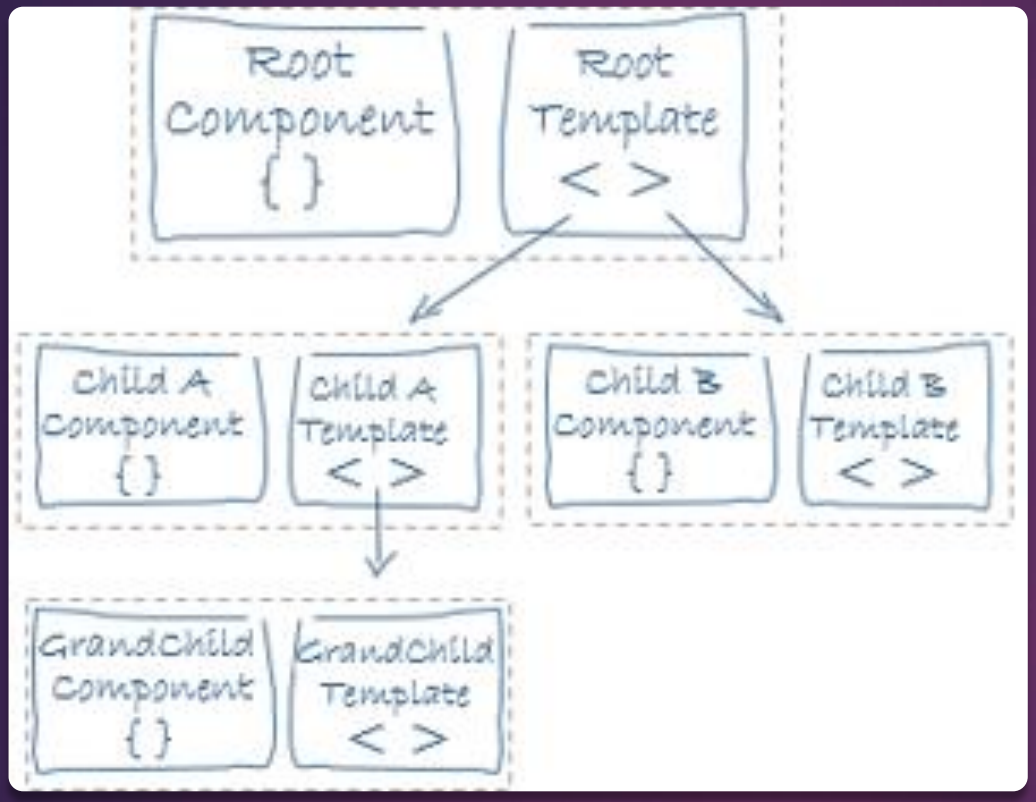
Steden – Weer

- 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

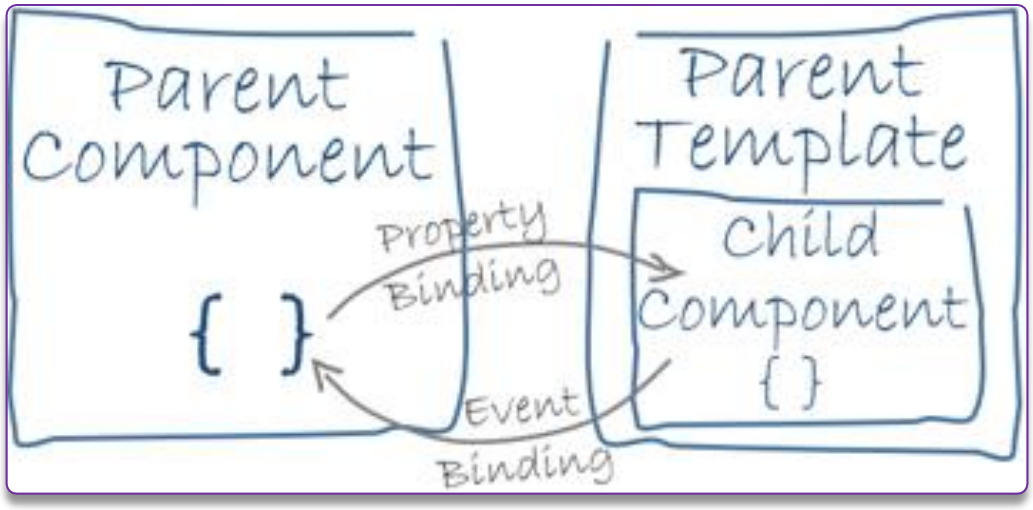


Componenten gebruiken in andere componenten

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
  - Structural directives: *ngFor en *ngIf
  - One way Property binding
  - Two-way binding
  - Events
  - Attributen toevoegen
  - Events toevoegen
  - Oudercomponenten

data en

interactie

uitwisselen tussen componenten

Overzicht

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

waar komen de data vandaan

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;
  }
}
```

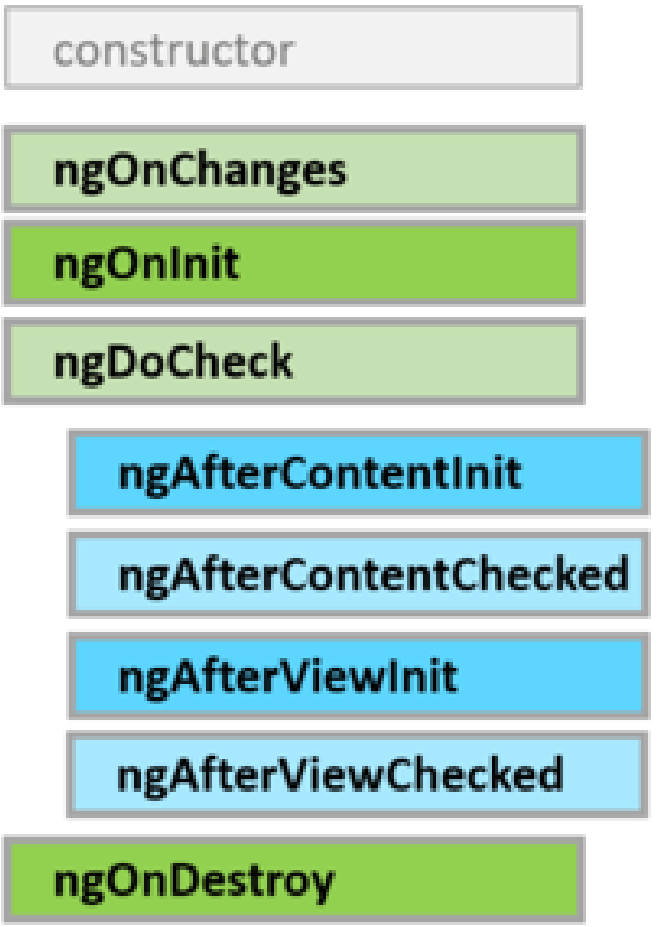
Service injecteren en gebruiken

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

waar komen de data vandaan

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

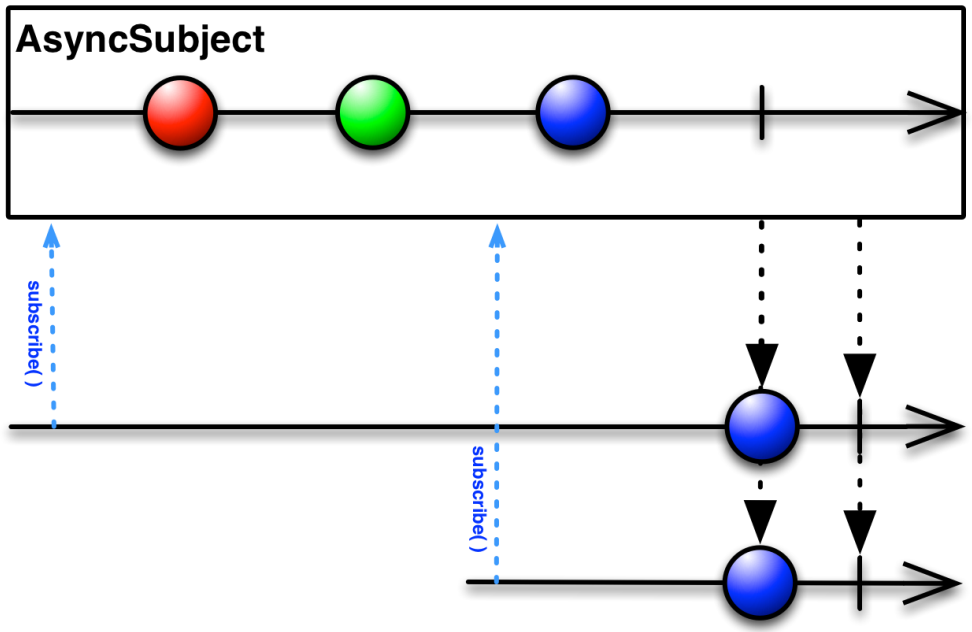
- http.get → Observable (meerdere asynchrone resultaten)
- RestCountry[] → Land[]

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

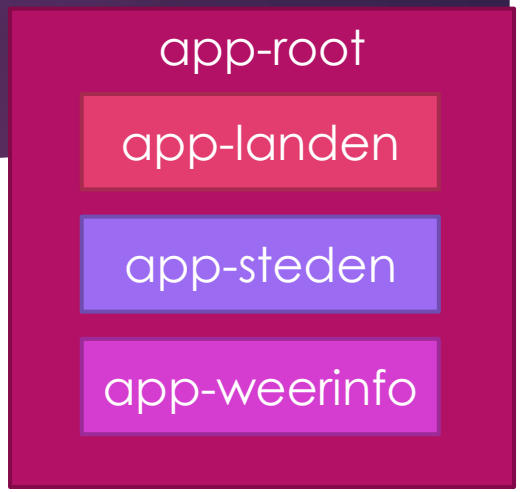
```
get landen(): Observable<Land[]> {
    return this.http.get<RestCountry[]>(this.landenURL)
    .pipe(
      tap(_ => console.log('fetched landen')),
      map(items => items.map(
        item => new Land(item.alpha2Code, item.name,item.capital)))
    );}
```

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

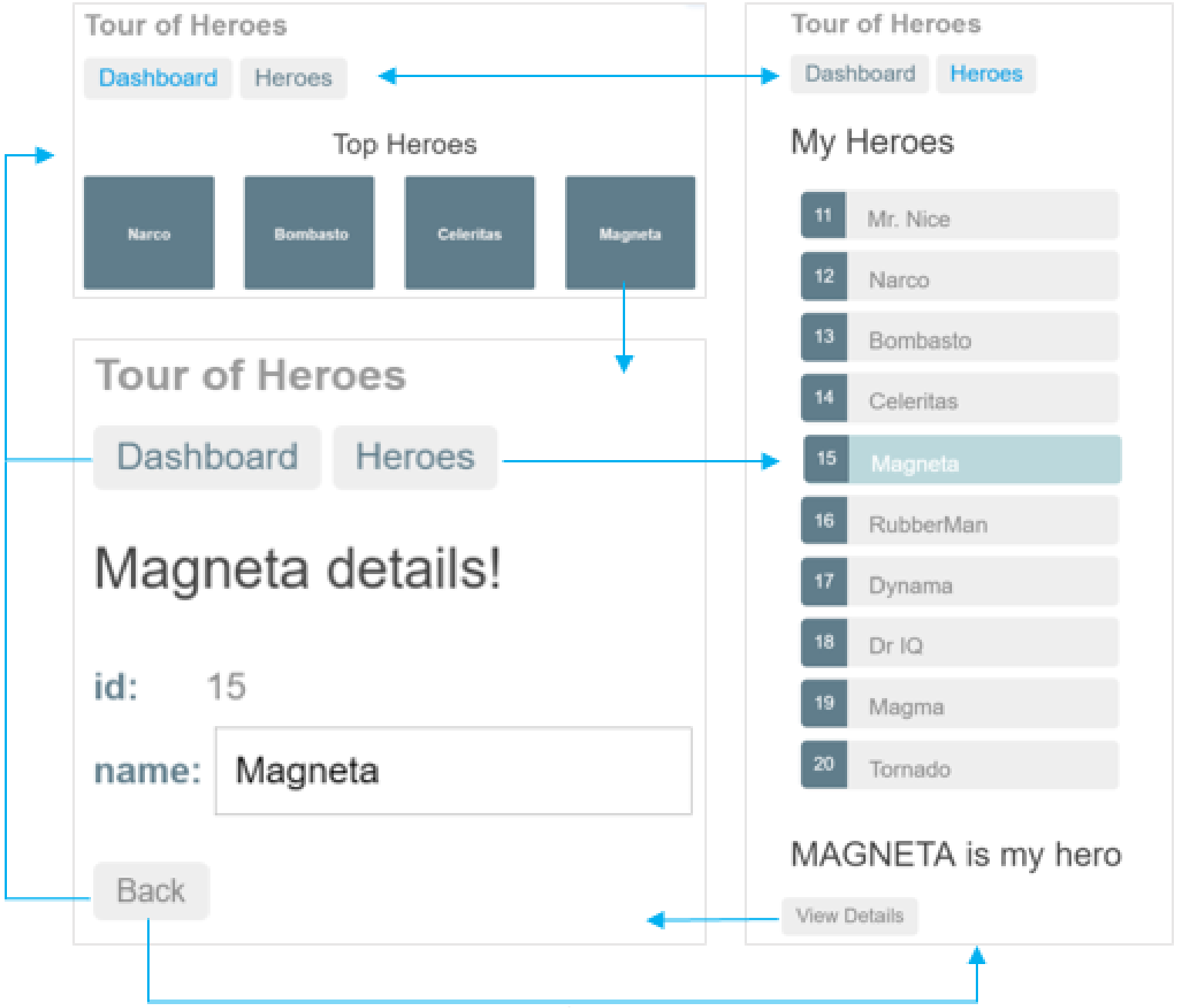
- Typescript
- Wat heb je nodig?
- Componenten
- Services
- HTTP-Services
- Configuratie

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

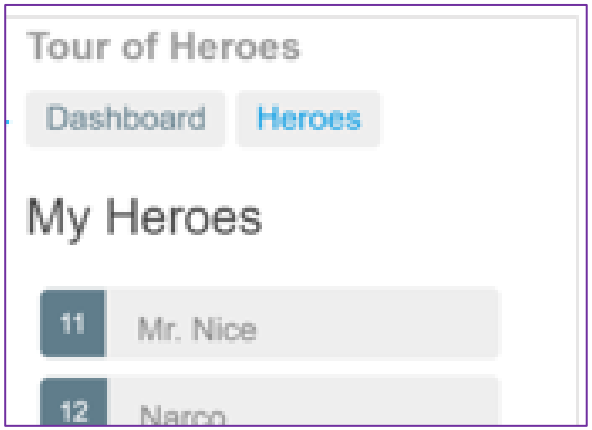
Routes configureren

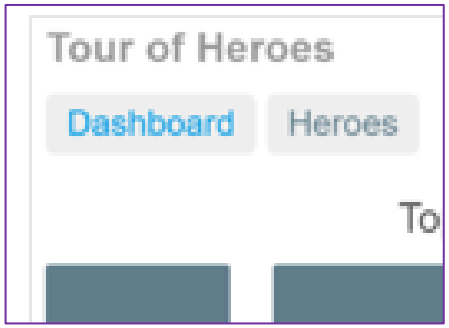
```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HeroesComponent } from './heroes/heroes.component';

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

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

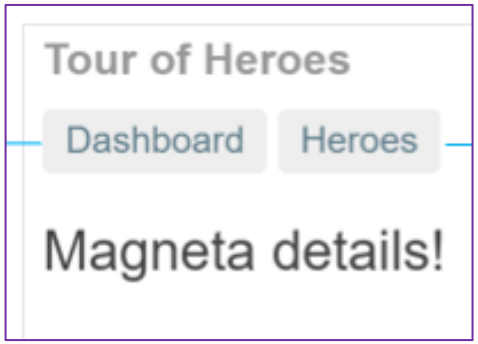
Waar component tonen?

- 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

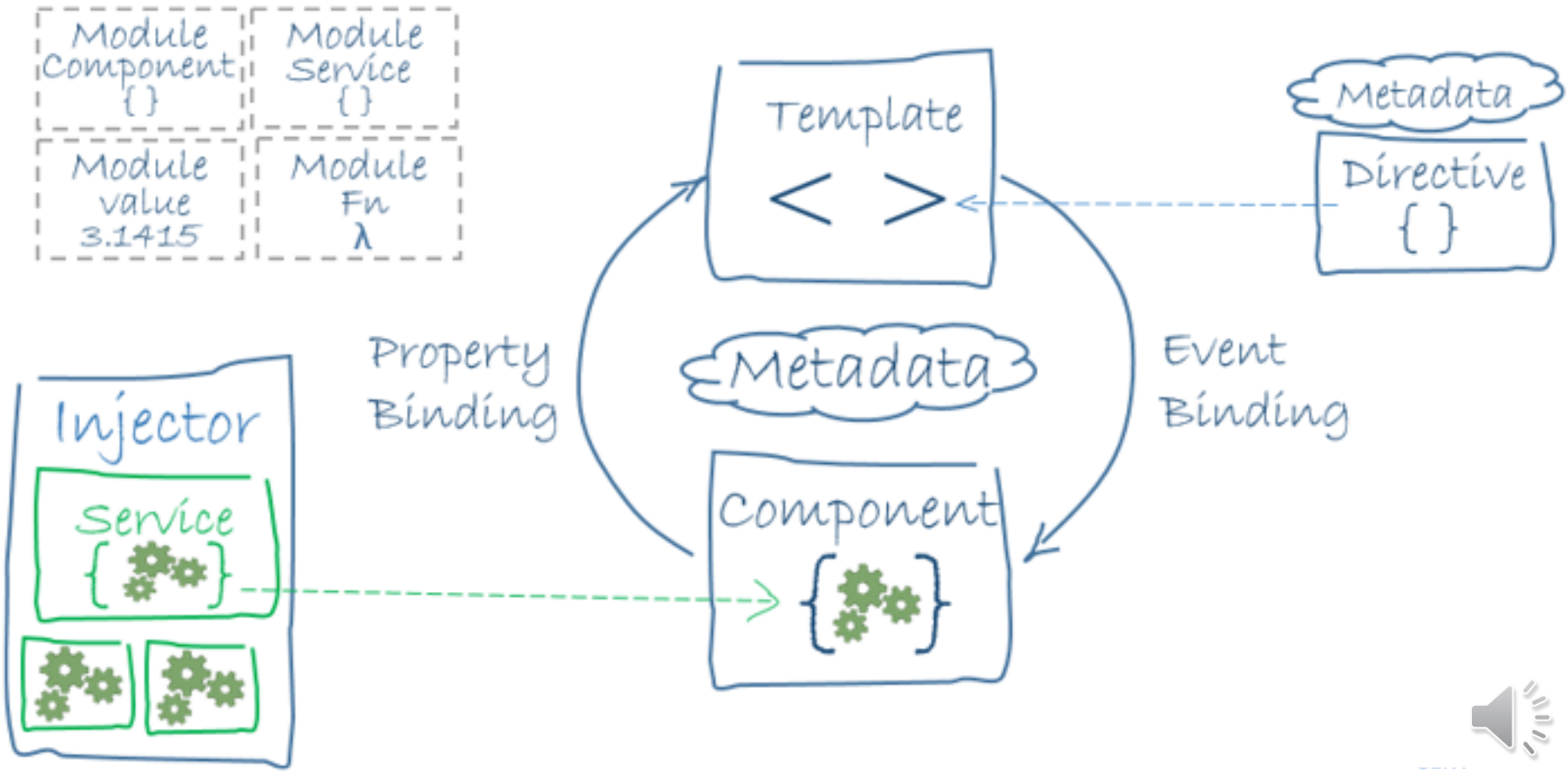
```
<button (click)="goBack()">
go back
</button>
```

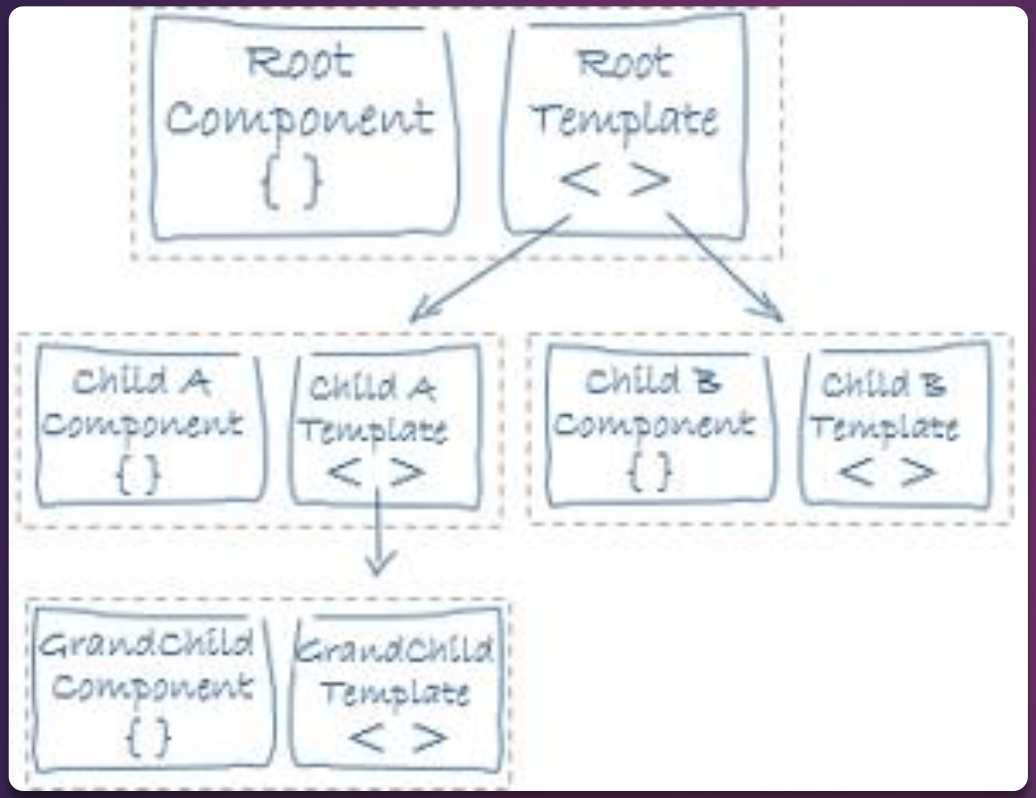
HTML-component

Overzicht

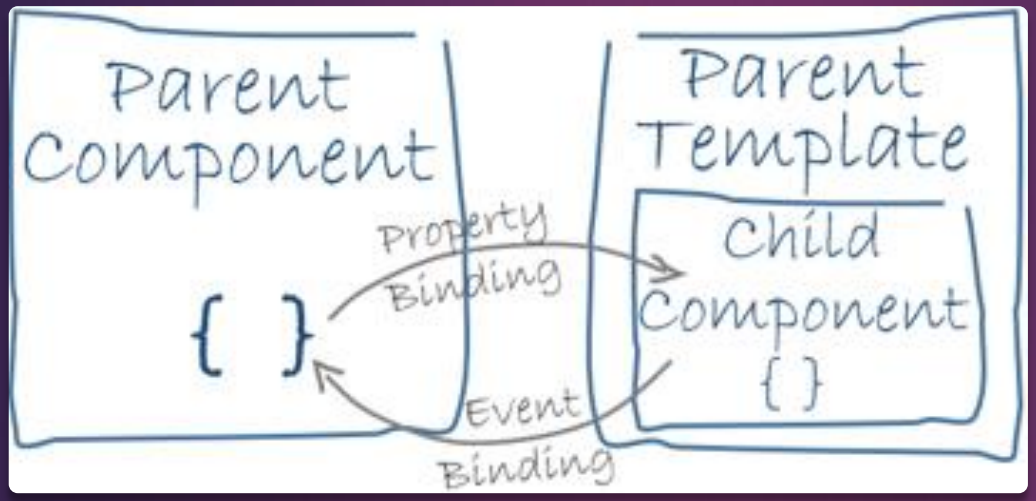
- Typescript
- Wat heb je nodig?
- Componenten
- Services
- HTTP-Services
- Configuratie
- Routing
- Architectuur

Architectuur



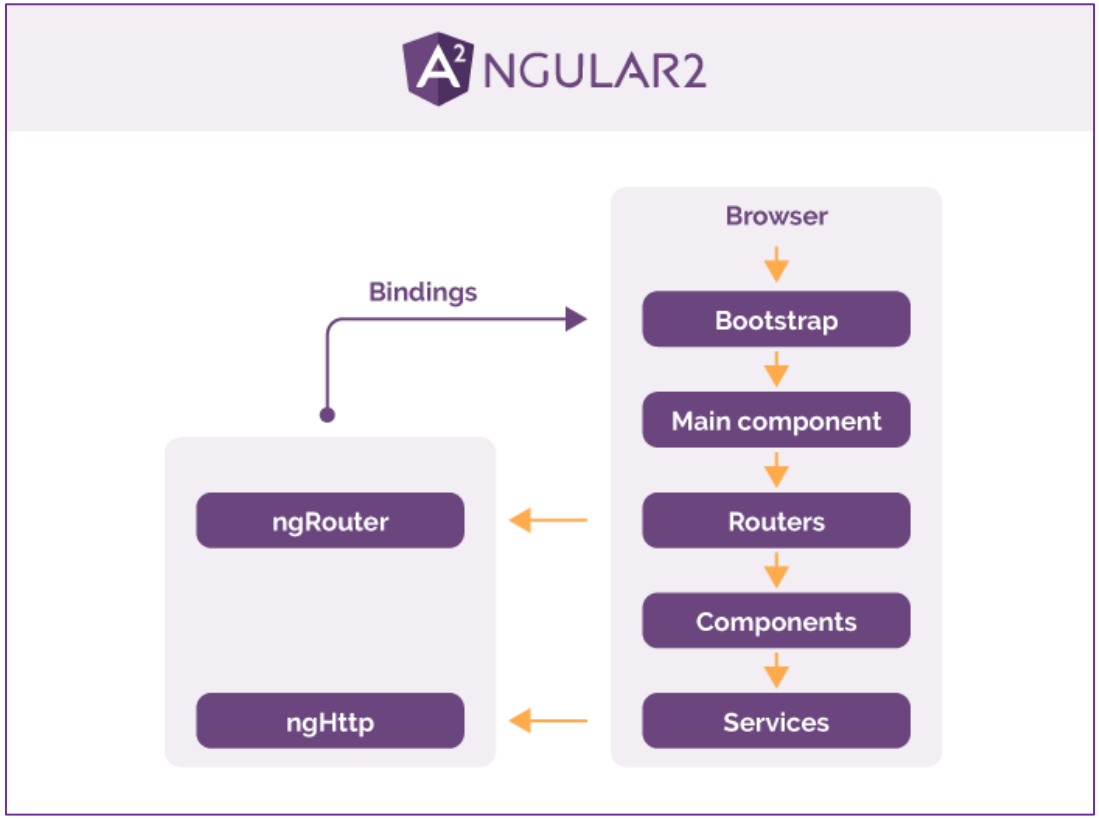


Kindcomponenten



Kindcomponenten

Overzicht Angular



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