

Components

Angular



Wat is een component

- Modules / custom objecten
- Implementatie van views met behulp van:
 - HTML templates
 - CSS stijlen
- Koppeling tussen view elementen en code


Wat is een component

- Components kunnen:
 - Andere components gebruiken
 - Code bevatten
 - Directives gebruiken
 - Services gebruiken

Wat is een component

- De voorbeeldcomponent heeft:
 - 2 input velden
 - Een output bericht
 - Een submit knop
- Om een component te gebruiken gebruiken we custom HTML tags:

```
<body>  
<app-login>Loading...</app-login>  
</body>
```



Inloggen

Naam:

Wachtwoord:

Welkom Dries!

Wat is een component

- Componenten in een groter geheel

Browser address bar: `ngrx.github.io/example-app/#/book/find`


Book Collection

Find a Book

Search for a book
angular

Ng-Book 2


The Complete Book on Angular 2



Ready to master Angular 2? What if you could master the entire framework - with solid foundations - in less time without beating your head against a wall? Imagine how quickly you could work if you knew the best practices and the best tools? Stop wastin...

Written By:
Nate Murray, Ari Lerner, Felipe Coury, and Carlos Taborda


AngularJS



Develop smaller, lighter web apps that are simple to create and easy to test, extend, and maintain as they grow. This hands-on guide introduces you to AngularJS, the open source JavaScript framework that uses Model-view-controller (MVC) architecture,...

Written By:
Brad Green and Shyam Seshadri


Professional AngularJS



A comprehensive guide to AngularJS, Google's open-source client-side framework for app development. Most of the existing guides to AngularJS struggle to provide simple and understandable explanations for more advanced concepts. As a result, some deve...

Written By:
Valeri Karpov and Diego Netto

Mastering Web Application Development with AngularJS




The book will be a step-by-step guide showing the readers how

`ngrx.github.io/example-app/#/book/eNExy_X1YyCc`


Eloquent JavaScript

A Modern Introduction to Programming



Provides information and examples on writing JavaScript code, covering such topics as syntax, control, data, regular

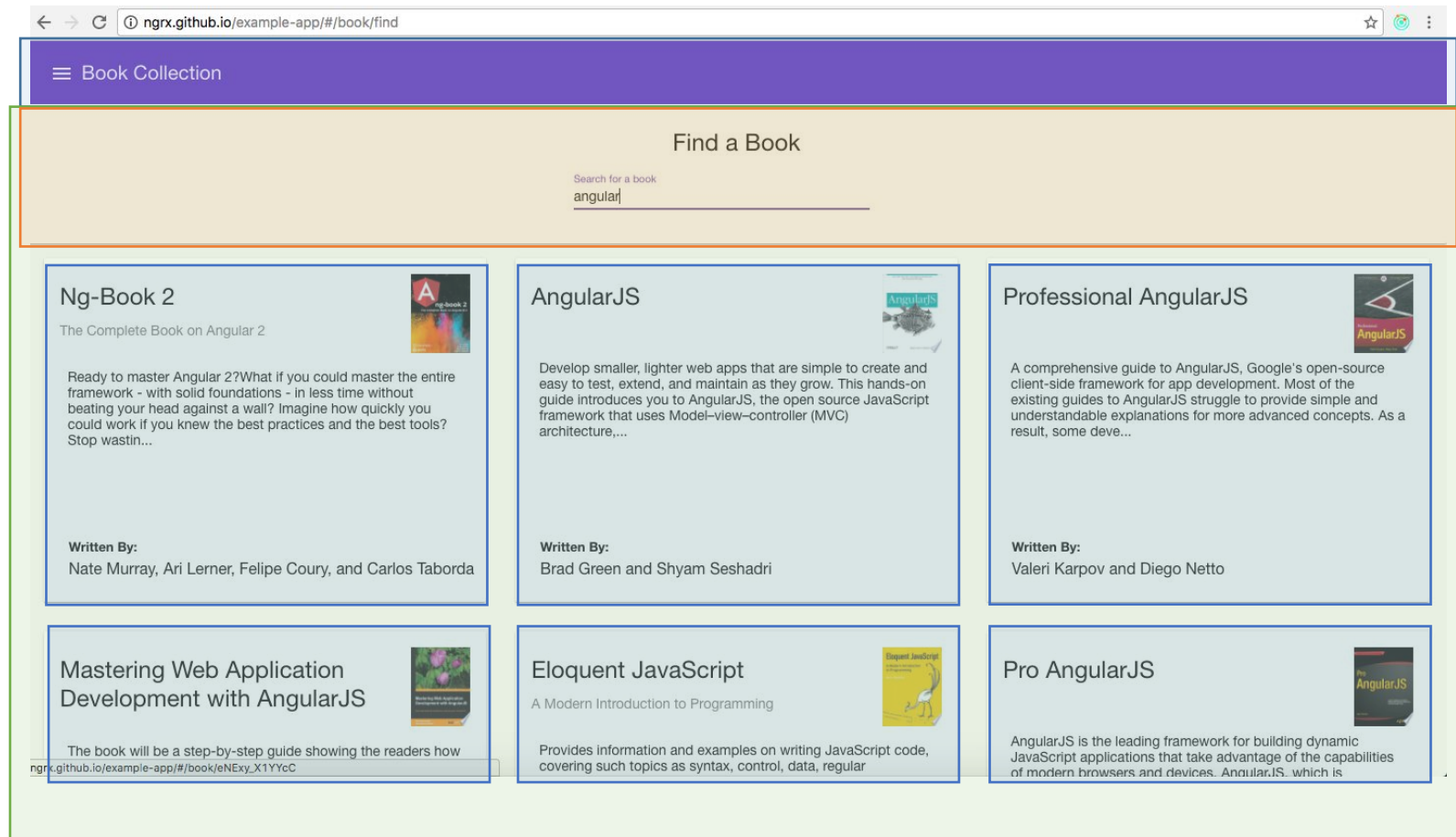
Pro AngularJS



AngularJS is the leading framework for building dynamic JavaScript applications that take advantage of the capabilities of modern browsers and devices. AngularJS, which is

Wat is een component

- Componenten in een groter geheel



Nav component

Search component

Booklist component

- > BookItem Component
- > BookItem Component
- > BookItem Component
- ...

Opbouw van een component

- Componenten hebben 3 secties:
 - Imports:
 - altijd minstens één import aanwezig.
 - Hier worden er toegevoegd als andere components / klassen / services gebruikt worden.
 - Annotatie: `@Component`
 - Geassocieerd met de klasse die eronder staat
 - Data in de annotatie wordt gebruikt om de view op te bouwen
 - JSON formaat
 - Component klasse
 - Export keyword zodat deze ook geïmporteerd kan worden in andere components

Opbouw van een component

```
TS app.component.ts x TS app.module.ts # app.component.css # styles.css
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-hallo',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'Hallo Web Expert!';
10  naam = 'Dries Swinnen';
11 }
```


@Component

- selector
- template
- styles
- templateUrl
- styleUrls
- input
- output
- providers
- ...

Components maken

- Handmatig aanmaken van de file
 - hallo.component.ts
- Voorzien van component import

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

- Voorzien van annotatie

```
@Component({  
  selector: 'app-hallo',  
  template: `<p>Hallo! dit is een test</p>`,  
  styles: [`p {color: blue;}`],  
})
```

Components maken

- Voorzien van component klasse

```
Export class HalloComponent {  
}
```

- **Toevoegen aan de declarations in de app.module.ts file**

```
import { HalloComponent } from './hallo.component';  
  
@NgModule({  
  declarations: [  
    AppComponent,  
    HalloComponent  
  ],  
})
```

- **Vervolgens kan je de selector <app-hallo> gebruiken in je project.**

Components maken

- hallo.component.ts

```
1  import { Component } from '@angular/core';
2
3  @Component({
4      selector: 'app-hallo',
5      template: `<p>Hallo! Dit is een test.` ,
6      styles: [`p {color: blue;}`],
7  })
8  export class HalloComponent {
9
10 }
11
```

Components maken

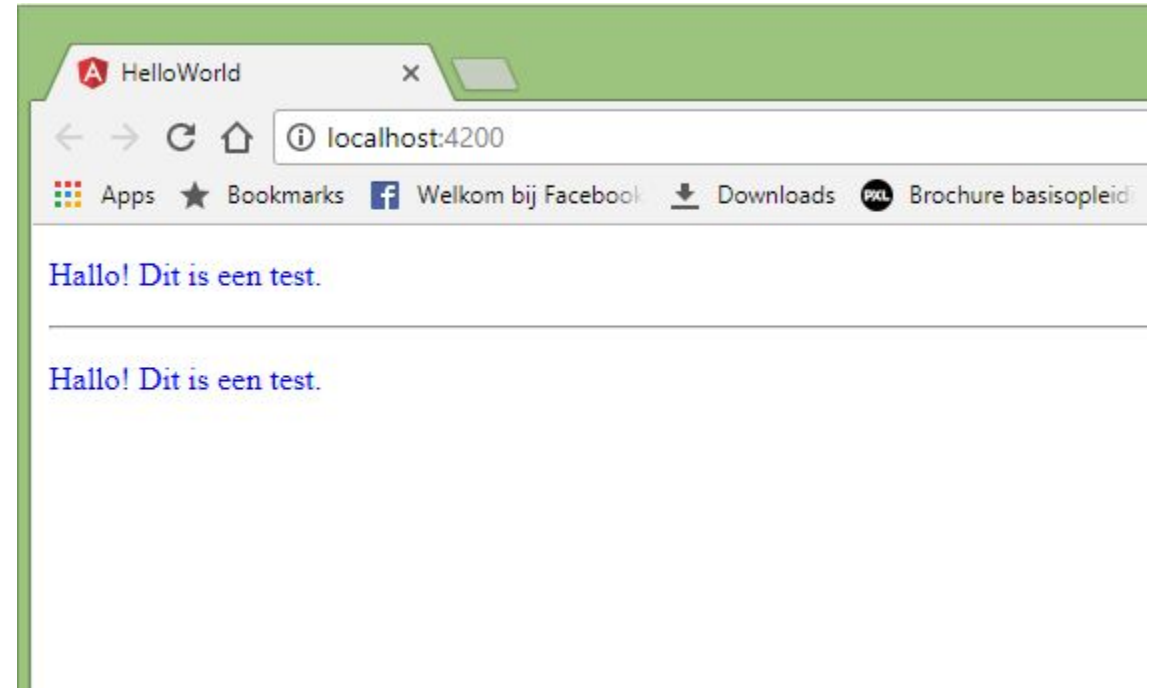
- app.module.ts

```
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3  import { AppComponent } from './app.component';
4  import { HalloComponent } from './hallo.component';
5
6
7  @NgModule({
8    declarations: [
9      AppComponent,
10     HalloComponent
11   ],
12   imports: [
13     BrowserModule
14   ],
15   providers: [],
16   bootstrap: [AppComponent]
17 })
18 export class AppModule { }
19
```

Components maken

- app.component.html

```
1 <app-hallo></app-hallo>  
2 <hr>  
3 <app-hallo></app-hallo>  
4  
5
```



Components maken

- AngularCLI

```
ng generate component path/to/component
```

Lifecycle hooks

- Components worden gemanaged door Angular zelf.
- Ze worden automatisch aangemaakt, gerenderd en vernietigd.
- Lifecycle hooks worden gebruikt om code uit te voeren op bepaalde tijdstippen:
 - `ngOnInit()`: Na het uitvoeren van de constructor van een component
 - `ngOnChanges()`: Na het aanpassen van een data-gebonden input veld
 - `ngOnDestroy()`: Bij het vernietigen van de component
 - ...

ngOnInit

- Meest gebruikte lifecycle hook
- Meestal om data op te halen die nodig is in de component
- Importeren uit @angular/core:

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

- OnInit interface implenteren op de component klasse:

```
export class halloComponent implements OnInit { ... }
```

- Methode met de naam ngOnInit() toevoegen aan de klasse

```
ngOnInit(){  
    ...  
}
```

ngOnInit

```
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-hallo',
5    template: `<p>Hallo! Dit is een test.` ,
6    styles: [`p {color: blue;}`],
7  })
8  export class HalloComponent implements OnInit {
9    constructor() {
10      console.log('uitvoeren constructor');
11    }
12    ngOnInit(): void {
13      console.log('initiatie van de component, na uitvoeren constructor');
14    }
15  }
16 }
```

Uitgewerkt voorbeeld: Login-component

- [CH3-voorbeeld1](#)
- Maak een nieuwe file aan login.component.ts met volgende inhoud:

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-login',
5    templateUrl: './login.component.html',
6    styles: [ ],
7  })
8  export class LoginComponent {
9    name: string;
10   password: string;
11   message: string = 'Please enter login details!';
12 }
13
```

Uitgewerkt voorbeeld: Login-component

- Maak een nieuw bestand login.comonent.html en voorzie volgende html code:

```
1  <h2>Please log-in</h2>
2  <table>
3    <tr>
4      <td>Name</td>
5      <td><input type="text" [(ngModel)]="name"/></td>
6    </tr>
7    <tr>
8      <td>Password</td>
9      <td><input type="password" [(ngModel)]="password"/></td>
10   </tr>
11 </table>
12 <p>{{ message }}</p>
13 <button (click)="verwerk()">Submit</button>
```

Uitgewerkt voorbeeld: Login-component

- Voeg de LoginComponent toe aan de declarations in de app.module.ts file:

```
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3  import { AppComponent } from './app.component';
4  import { LoginComponent } from './login.component';
5
6
7  @NgModule({
8    declarations: [
9      AppComponent,
10     LoginComponent
11   ],
```

- Voeg de <app-login></app-login> tag toe aan de app.component.html file

[(ngModel)]

- 2 way databinding
- Zorgt voor een verbinding tussen de inhoud van het input veld en een variabele in de component klasse
- Aanpassingen gebeuren automatisch in beide richtingen!
- **Inladen van de FormsModule in de app.module.ts file**

```
5 import { FormsModule } from '@angular/forms'; //voor gebruik ngModel
6
7 @NgModule({
8   declarations: [
9     AppComponent,
10    LoginComponent
11  ],
12   imports: [
13     BrowserModule,
14     FormsModule, //voor gebruik ngModel
15  ],
```

[(ngModel)]

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-login',
5   templateUrl: './login.component.html',
6   styles: [ ],
7 })
8 export class LoginComponent {
9   name: string;
10  password: string;
11  message: string = 'Please enter login details!';
12 }
13
```

```
1 <h2>Please log-in</h2>
2 <table>
3   <tr>
4     <td>Name</td>
5     <td><input type="text" [(ngModel)]="name"/></td>
6   </tr>
7   <tr>
8     <td>Password</td>
9     <td><input type="password" [(ngModel)]="password"/></td>
10  </tr>
11 </table>
12 <p>{{ message }}</p>
13 <button (click)="verwerk()">Submit</button>
```

- {{ ... }} wordt gebruikt als one way binding (model to view)

Event handling

- One way databinding van view naar model
- Events worden gekoppeld in de html code naar een methode in de component klasse

```
13 <button (click)="verwerk()">Submit</button>
```

- Verschillende events zoals: submit, click, dblclick, dragover, focus, blur, keydown, ...

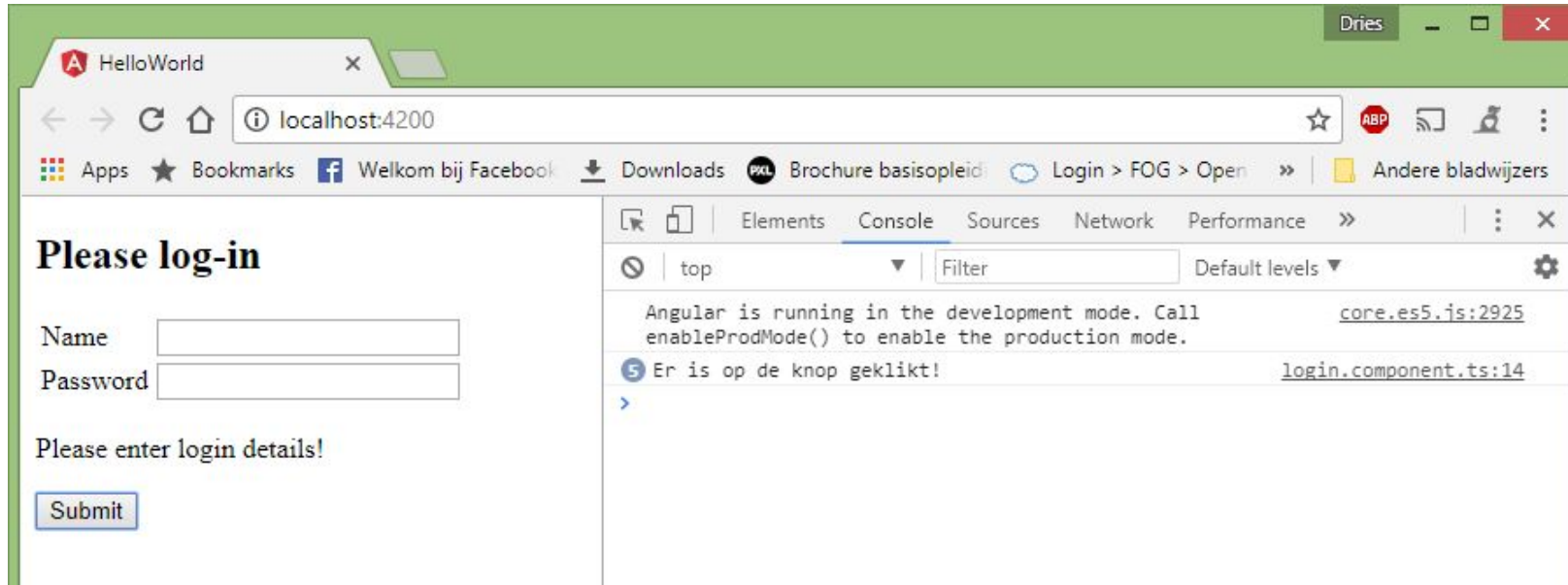
Event handling

- De event wordt altijd gekoppeld aan een methode in de component klasse:

```
8   export class LoginComponent {  
9       name: string;  
10      password: string;  
11      message: string = 'Please enter login details!';  
12  
13      verwerk(){  
14          console.log('Er is op de knop geklikt!');  
15      }  
16  }
```

- Verwerk() wordt opgeroepen bij het klikken op de button.

Event handling



Event handling & databinding

- In onderstaand voorbeeld wordt message aangepast.
- Name heeft automatisch de waarde van het input veld door `[(ngModel)]` (**2 way databinding**)
- Message wordt automatisch aangepast op de view door de verwijzing `{{ message }}` (**1 way databinding – model to view**)

```
8  export class LoginComponent {
9      name: string;
10     password: string;
11     message: string = 'Please enter login details!';
12
13     verwerk(){
14         this.message = this.name + ' has logged in!';
15     }
16 }
```

Please log-in

Name

Password

Dries has logged in!