# React vs Angular

In de volgende context wanneer er over Angular gesproken wordt er geen rekening gehouden met Angular1 maar wordt er Angular2/4 bedoeld. De verschillen tussen deze twee versies zijn zo groot dat Angular2 een geheel nieuw framework is t.o.v. Angular1. Angular1 vs Angular2 is al een geheel nieuwe vergijkende studie op zich.

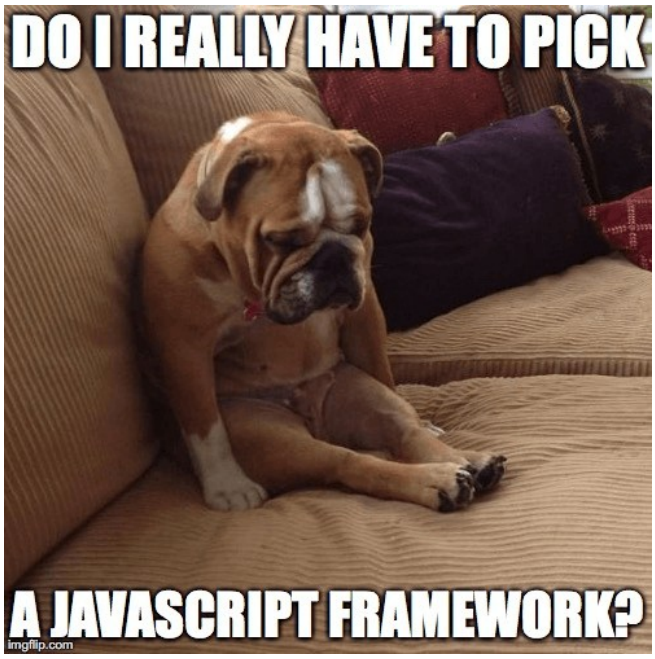
## Angular kort samengevat

Angular is een javascript **framework** ontwikkeld door Google. Het framework maakt gebruik van Javascript of **Typescript** (een superset van Javascript). Een applicatie bestaat uit verschillende **HTML-templates** met toevoeging van Angular markup. Die HTML-templates worden aangestuurd door **component klassen** die de templates beheren. Waarbij de logica wordt aangereikt door **Services**. Waarbij componenten en services in modules worden gewikkeld.

## React kort samengevat

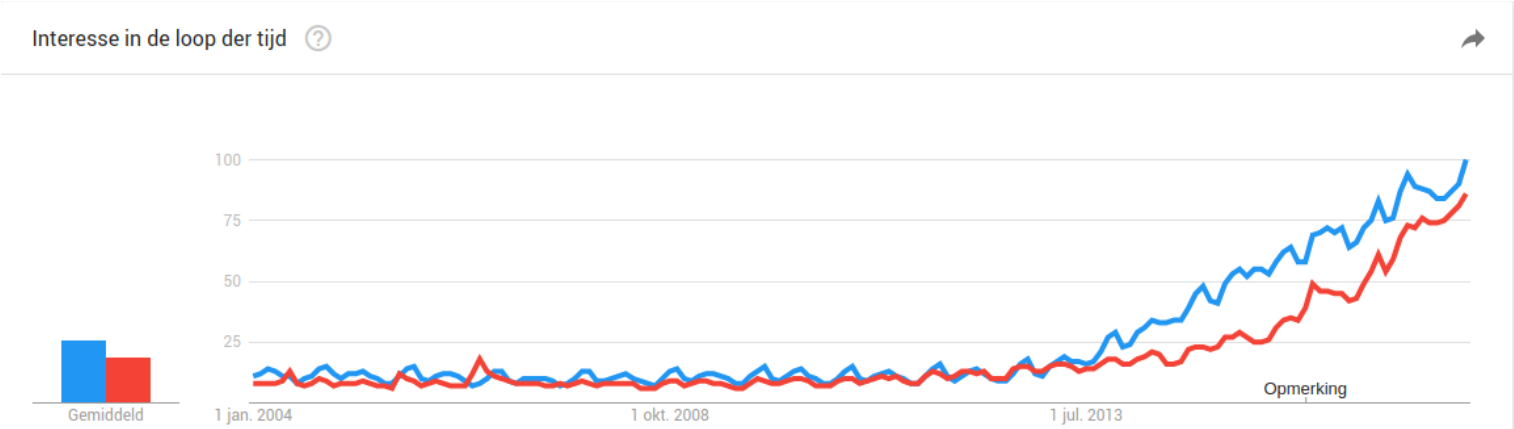
React is een javascript **library** ontwikkeld door Facebook. React maakt gebruik van **JSX** (een HTML XML syntax om de DOM te virtualiseren). React maakt gebruik van een zogenaamde '**Virtual DOM'** wat betekend dat React eerst begint met het renderen van alle component en bouwt een in-memory representatie van de DOM (een kopie als het ware). React kijkt dan of er veranderingen zijn en rendered alleen de elementen die veranderd zijn in de werkelijke DOM wat maakt dat het heel snel is aangezien DOM manipulatie rechtstreeks veel trager is  javascript manipulatie.

Hier een voorbeeld van React's viturtual DOM (gif bestand) : https://maerch.github.io/img/react-vdom/angular-react.gif

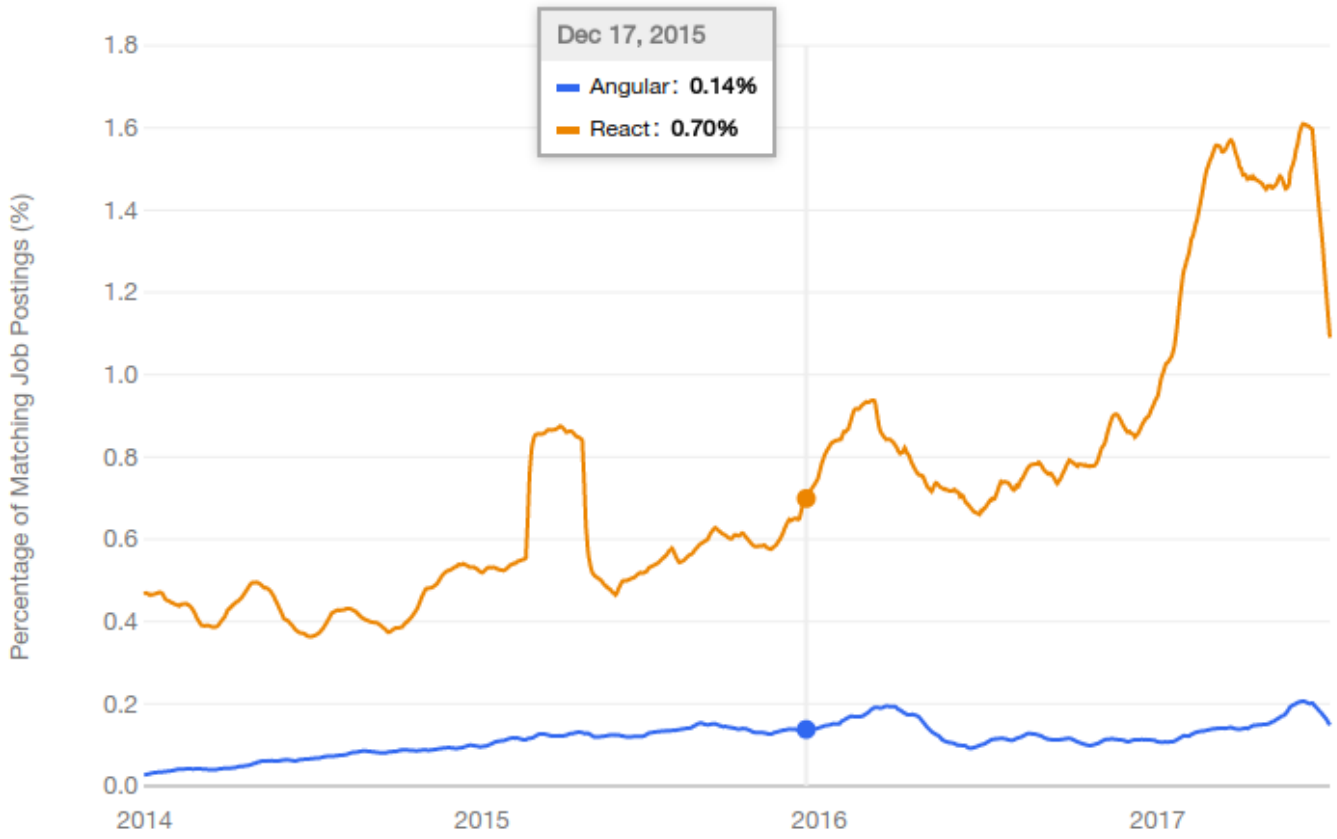


Statistieken

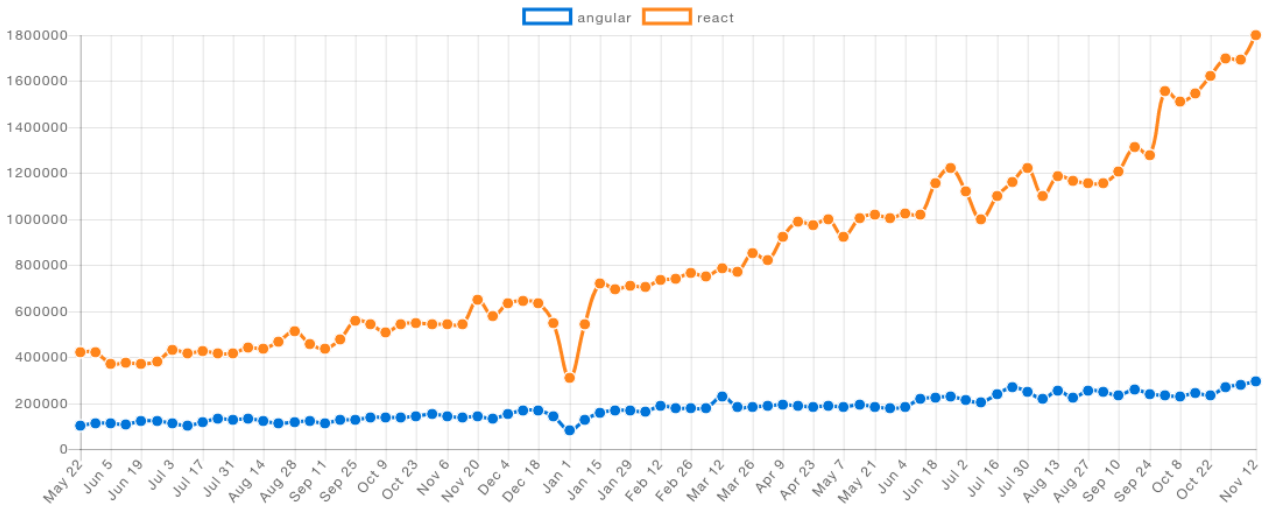
google trends (angular – blauw , React – rood):




Job trends (indeed.com) :



## NPM stats




### Github Stats

| | stars ☆ | forks 🍽 | issues ⚠ | updated 🛠 | created 🐣 |
|---|---|---|---|---|---|
| angular.js | 57526 | 28698 | 698 | Nov 15, 2017 | Jan 6, 2010 |
| react | 81195 | 15349 | 416 | Nov 15, 2017 | May 24, 2013 |

React is bezig met een opmars in het de wereld van front-end developing maar Angular blijft statistisch gezien de meest gebruikte. Alhoewel er duidelijk een verschil te zien is in de jobs die er gezocht worden waar React op het verlanglijstje staat. Een verklaring is dat de jobmarkt voor Angular verzadigd wordt aangezien er al veel mensen angular kunnen en de react op de achtergrond blijft.
Wat wel een feit blijft dat de populariteit blijft stijgen bij zowel React als Angular1

## Angular pro’s

**– Een out of the box werkend platform**
Angular twee brengt een all-in-one framework, geen noodig om aparte libraries te installeren angular heeft de nodige features al ingebakken om een webapplicatie op te zetten zonder het ongemak van configuratie.

**– Ahead Of Tim compilation**
Met ahead of time compilation wordt de HTML en Typescript eerst geconverteerd naar Javascript alvorens de browser de code download en runned. Dit zorgt voor een snellere rendering, kleinere file size en veiligere code aangezien de originele code niet wordt blootgesteld.

**– 1 codebase voor alle apparaten**
Door middel van NativeScript moet er geen aparte codebases meer gemaakt worden voor IOS, Android of web. Door aard van Angular is het mogelijk om met dezelfde code te ontwikkelen voor verschillende devices.

**– Typescript**
Dit is subjectief voor sommige zou dit eerder als nadeel gezien worden. Typescript werkt object georiënteerd waardoor code sneller en duidelijk wordt. Het kan zowel gebruikt worden voor client-side en server-side redering. ES6 features supported. En last but not least Static Type checking waarbij de compiler gaat kijken of de variabele types overeenkomen.

**– Component based**
Voor elk onderdeel van een app of site wordt een aparte component gemaakt zo wordt hergebruiken van componenten gemakkelijk alsook de code leesbaarder.

## Angular con’s

**– Moeilijker om te leren**
Om Angular onder de knie te hebben moet je eerst wat werk doen. Ik denk maar aan depedicy injection en module loading.

**– Niet veel vrijheid, je moet dingen the angular way doen**
Angular is gestructureerd en geeft je de tools die je nodig hebt er is niet veel vrijheid om dingen op je eigen manier te doen.

**– Gewone DOM**
Angular maakt gebruik van gew

## React pro’s

**– Efficiënt, mede door Virtual DOM**
Doordat alle comonenten in Javascript zitten is het gemakkelijk om code te lezen en dit maakt het efficiënt om alleen de veranderingen door te voeren die gebeurt i.p.v. de hele dom te rendere.

**– Component based**
Idem als bij Angular

**– Vrijheid**
Je hebt keuze welke libraries je gebruikt en hoe je alles doet. Dit maakt het ook ineens een nadeel omdat dit meer werk met zich meebrengt.

**– Eenvoudig**
De leercurve is kleiner dan bij Angular en voelt meer vertrouwd aan. Het gebruik van npm packages is ook eenvoudiger dan sommige Angular wrapper die moeten gemaakt worden om een library te laten werken.

**– Performance**
Door de virtual DOM is React zeer snel.

**– Javascript-centric**
React maakt gebruik van pure Javascript met HTML rendering (JSX) er is dus niets nieuws te leren;

## React con’s

**– Veel verschillende libraries nodig**
Het opstarten van een project kost meer tijd. Dit is ook een sterkte doordat het zo lightweight is. Maar alles moet van in het begin geconfigureerd worden.

**– Geen vaste structuur**
Elke developer heeft zijn manier om componenten te maken. Door de vrijheid die er is in React heeft iedereen een andere manier om een probleem te benaderen waardoor in een team aan projecten werken moeilijker is en code kan onduidelijk of niet performant worden. Het mist een vaste structuur.

**– Library keuze**
Er is zo een groot aanbod aan libraries die je kan gebruiken om bvb data af te handelen dat de keuze van een bepaalde library moeilijk maakt (word die vaak upgedate, goeie community e.d.).

## Performance

Hier volgen een aantal performance tests. Bon : [http://www.stefankrause.net/js-frameworks-benchmark6/webdriver-ts-results/table.html](http://www.stefankrause.net/js-frameworks-benchmark6/webdriver-ts-results/table.html)

(links) cijfers in millisecone ± afwijking, (rechts) cijfers in MBs ± afwijking

| Name | angular-v4.1.2-keyed | react-v15.5.4-keyed | react-v15.5.4-redux-v3.6.0 |
|---|---|---|---|
| create rows<br>Duration for creating 1000 rows after the page loaded. | 193.1 ± 7.9 (1.0) | 188.9 ± 10.9 (1.0) | 212.2 ± 14.2 (1.1) |
| replace all rows<br>Duration for updating all 1000 rows of the table (with 5 warmup iterations). | 197.4 ± 5.3 (1.0) | 201.0 ± 6.4 (1.0) | 206.7 ± 7.3 (1.0) |
| partial update<br>Time to update the text of every 10th row (with 5 warmup iterations). | 13.0 ± 4.5 (1.0) | 16.5 ± 2.3 (1.0) | 18.0 ± 1.6 (1.1) |
| select row<br>Duration to highlight a row in response to a click on the row. (with 5 warmup iterations). | 3.4 ± 2.3 (1.0) | 8.8 ± 3.4 (1.0) | 8.7 ± 2.9 (1.0) |
| swap rows<br>Time to swap 2 rows on a 1K table. (with 5 warmup iterations). | 13.4 ± 1.0 (1.0) | 14.7 ± 0.9 (1.0) | 17.1 ± 1.3 (1.1) |
| remove row<br>Duration to remove a row. (with 5 warmup iterations). | 46.1 ± 3.2 (1.0) | 47.2 ± 3.2 (1.0) | 52.4 ± 1.7 (1.1) |
| create many rows<br>Duration to create 10,000 rows | 1946.0 ± 41.8 (1.1) | 1852.4 ± 29.0 (1.0) | 1931.7 ± 35.6 (1.0) |
| append rows to large table<br>Duration for adding 1000 rows on a table of 10,000 rows. | 324.6 ± 10.1 (1.0) | 345.6 ± 10.4 (1.1) | 366.4 ± 10.9 (1.1) |
| clear rows<br>Duration to clear the table filled with 10.000 rows. | 379.9 ± 11.3 (1.0) | 398.4 ± 8.2 (1.0) | 410.9 ± 9.8 (1.1) |
| startup time<br>Time for loading, parsing and starting up | 84.3 ± 2.6 (1.2) | 70.0 ± 2.9 (1.0) | 93.8 ± 6.9 (1.3) |
| slowdown geometric mean | 1.03 | 1.02 | 1.11 |

| Name | angular-v4.1.2-keyed | react-v15.5.4-keyed | react-v15.5.4-redux-v3.6.0 |
|---|---|---|---|
| ready memory<br>Memory usage after page load. | 4.8 ± 0.0 (1.1) | 4.5 ± 0.1 (1.0) | 4.9 ± 0.1 (1.1) |
| run memory<br>Memory usage after adding 1000 rows. | 10.9 ± 0.1 (1.1) | 9.7 ± 0.1 (1.0) | 10.8 ± 0.1 (1.1) |

Hieruit kunnen we opmaken er niet veel verschil is in performantie tussen de twee frameworks. Alhoewel react een betere startup tijd maar in combinatie met redux geeft dit ongeveer hetzelfde cijfer.

## Vergelijk : todo-list

Om het vergelijk compleet te maken heb ik twee dezelfde projecten gemaakt, een todo-list in Angular en React.

## Een component in Angular

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { TodoServiceService } from '../services/todo-service.service';
import 'rxjs/add/operator/map';
@Component({
  selector: 'app-todo-list',
  templateUrl: './todo-list.component.html',
  styleUrls: ['./todo-list.component.css']
})
export class TodoListComponent implements OnInit {
  currentStatus = '';
  constructor(private todoService: TodoServiceService, private route: ActivatedRoute)
{ }
  ngOnInit() {
      this.route.params
          .map(params => params.status)
          .subscribe((status) => {
              this.currentStatus = status;
          });
  }
    remove(id) {
        this.todoService.remove(id);
    }
    update() {
        this.todoService.persist();
    }
    getTodos() {
        if (this.currentStatus == 'completed') {
            return this.todoService.getCompleted();
        } else if (this.currentStatus == 'active') {
            return this.todoService.getRemaining();
        } else {
            return this.todoService.todos;
        }
    }
    allCompleted() {
        return this.todoService.allCompleted();
    }
    setAllTo(toggleAll) {
        this.todoService.setAllTo(toggleAll.checked);
    }
}

<section class="todoapp">
  <app-header></app-header>
  <section class="main" *ngIf="getTodos().length">
    <input class="toggle-all" type="checkbox" #toggleall [checked]="allCompleted()"
(click)="setAllTo(toggleall)">
    <ul class="todo-list">
      <app-todo-item *ngFor="let todo of getTodos()" [todo]="todo"
(itemRemoved)="remove($event)" (itemModified)="update($event)"></app-todo-item>
    </ul>
  </section>
  <app-footer></app-footer>
</section>
```

## Een component in react

```
import React, {Component} from 'react';
import TodoListItem from './Todo-List-Item';
class TodoList extends Component {
    setCompleted(id){
       this.props.toggleComplete(id);
    }
    deleteTodo(id){
        this.props.deleteTodo(id);
    }
    updateTodo(todo){
        this.props.updateTodo(todo);
    }
    render() {
        const {todos } = this.props;
        const mappedTodos = todos.map((todo, i) => <TodoListItem key={i}
editTodo={this.updateTodo.bind(this)}
toggleComplete={this.setCompleted.bind(this)}
deleteTodo={this.deleteTodo.bind(this)} completed={todo.completed}
text={todo.text} id={todo.id}/>);
        return (
            <section className="main">
                <input className="toggle-all" type="checkbox"/>
                <ul className="todo-list">
                    {mappedTodos}
                </ul>
            </section>
        );
    }
}
export default TodoList;
```

Voor het volledige project zie folders : angular-todo en react-todo
run :
npm install
npm start

## Conclusie

Of het nu Angular of React is ze hebben allemaal hun eigen voor en nadelen. Het zijn beide goede front-end frameworks die gebruik maken van de nieuwste technologieën. Het is vooral persoonlijke voorkeur en de manier waarop je ontwikkeld. Voor projecten met grotere teams zou ik eerder opteren voor Angular door de vaste structuur en de leesbaarheid. Voor projecten die veel verandering van de DOM en updates nodig hebben zou ik eerder opteren voor React.