Live webcollege
Front-end frameworks

Mark Nijenhuis

Achter de schermen: Marielle en Anouschka

Even een korte introductie

Vanaf de oertijd (ca. 1986) houd ik me vooral bezig met CAD-tekenen, illustreren, fotograferen, webontwikkeling, grafische vormgeving en lesgeven. Naast de ‘visuele’ aspecten houd ik me de laatste tijd ook veel bezig met coderen in talen als JavaScript en Typescript en in het verleden ook Actionscript (Flash).
Infographics hebben mijn speciale aandacht.

Voor LOI ontwikkel ik lesmateriaal en geef les aan de mbo-opleiding Mediavormgever en de hbo-opleiding CMD.

Ik hoop in dit webcollege een tipje van de sluier op te kunnen lichten met betrekking tot de snel veranderende wereld van front-end frameworks.

En wie bent u?

- Wie volgt voor het eerst een live web college?
- En wie is al professioneel werkzaam als webdesigner of front-end developer?
- Wie heeft al eens gewerkt met één van deze frameworks?

Geef uw antwoord in de openbare chat.

Opbouw van dit live webcollege

Onderdeel 1 – *Angular, React en VueJS*

- *Een korte introductie en vergelijking*

Onderdeel 2 – *Kernconcepten*

- *Reactive programming, asynchronous programming, data-oriented programming, state management, routing*

Onderdeel 3 – *Live demo VueJs en Angular*

- *Lokaal ontwikkelen met MSC, een Angular-project inrichten, voorbeeldprojecten*

Onderdeel 1 – *Angular, React en VueJS*

Introductie en vergelijking

Libraries en frameworks (recap)

- Bij gebruik van een library is de ontwikkelaar ‘in control’ en deze bepaalt wanneer en waar de library binnen de code gebruikt wordt.
  Bijvoorbeeld: Bootstrap en jQuery.

- Bij gebruik van een framework is het framework ‘in control’ en werkt de ontwikkelaar binnen de grenzen van het framework. Een framework is een geheel van softwarecomponenten dat gebruikt kan worden bij het programmeren van applicaties. Echter, ook afspraken hoe die componenten gebruikt worden binnen een groep ontwikkelaars en welke codestandaarden en bibliotheken gebruikt worden, kunnen ook onderdeel zijn van een framework.
  Bijvoorbeeld: Ionic, VueJS, Angular en React.

Web development in twee kampen verdeeld

Aan mijn linkerkant

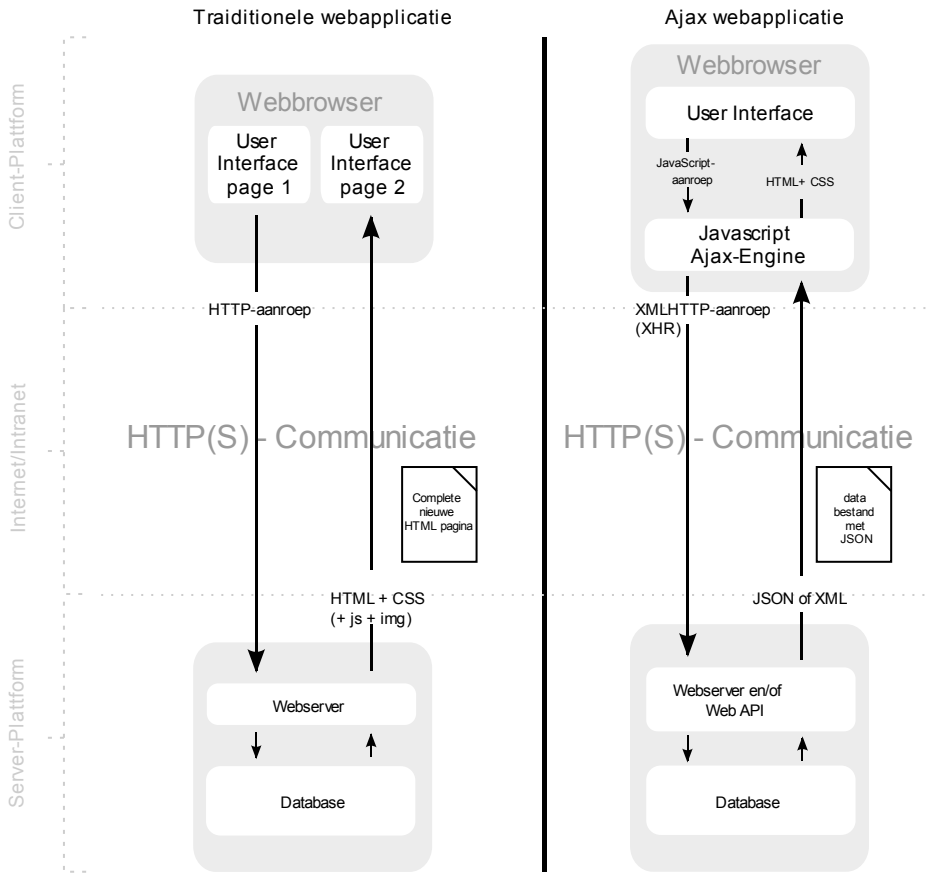
- De wereld van (vooral open source) cms-systemen zoals Wordpress, Joomla, Drupal en Magento.
- Grotendeels gebaseerd op PHP-templates.
- Gebaseerd op http aanroepen (nieuwe pagina's bij elke klik).
- JavaScript is alleen ter ondersteuning van de UI/UX (effecten, carousels, dropdownmenu's en formuliervalidatie).
- Front end en back end zijn 'tight coupled'.

Aan mijn rechterkant

- De wereld van custom built webapplicaties.
- Gebouwd met JavaScript/TypeScript.
- Gebaseerd op de XmlHttpRequest óf de nieuwe fetch() API en daardoor een Single page application ("look mom, no page refreshes!").
- De hele applicatie draait onder JavaScript.
- Front end en back end zijn via API's 'loosely coupled'.

LOI

Schematisch weergegeven

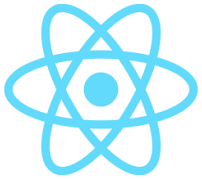


Bron: Wikipedia, bewerkt door OGN.

De drie ‘grote’ JavaScript Front-end frameworks

Angular (Google)

- Groot en schaalbaar
- Gebruikt door overheden en banken
- Steile leercurve door gebruik van TypeScript en concepten uit de wereld van ‘echte’ programmeurs (JAVA, etc)
- Goed gedocumenteerd
- Handige Command Line Interface



React (van Facebook)

- Meest gebruikt en meest populair
- Steile leercurve maar minder steil dan van Angular

VueJS

- Volgens de ontwikkelaar: al het goede van Angular
- Lichtgewicht en naar eigen inzicht toe te passen op delen van een webapplicatie
- Toegankelijk en goed gedocumenteerd

- Angular is het meest volwassen van de frameworks, heeft een goede ondersteuning in termen van bijdragers en is een compleet pakket. De leercurve is echter steil en ontwikkelingsconcepten in Angular kunnen nieuwe ontwikkelaars afschrikken. Angular is een goede keuze voor bedrijven met grote teams en ontwikkelaars die al TypeScript gebruiken.
- React is net oud genoeg om volwassen te zijn en heeft een enorm aantal bijdragen van de community. Het is algemeen aanvaard. De arbeidsmarkt voor React is erg goed en de toekomst voor dit framework ziet er rooskleurig uit. React lijkt een goede keuze voor iemand die aan de slag gaat met front-end JavaScript-frameworks, startups en ontwikkelaars die van enige flexibiliteit houden. De mogelijkheid om naadloos te integreren met andere frameworks geeft het een groot voordeel voor diegenen die wat flexibiliteit in hun code willen.
- Vue is het nieuwste in de arena, zonder de steun van een groot bedrijf. Het heeft de afgelopen jaren echter heel goed gepresteerd om uit te komen als een sterke concurrent voor Angular en React. Dit speelt misschien een rol bij veel Chinese reuzen zoals Alibaba en Baidu die Vue kiezen als hun primaire front-end JavaScript-framework. Het valt echter nog te bezien hoe het in de toekomst zal gaan en men moet er voorzichtig mee zijn. Vue is de juiste keuze als u de voorkeur geeft aan eenvoud, maar ook aan flexibiliteit.

Bron: Codeinwp.com/blog/angular-vs-vue-vs-react

Vraag 1

Afgaand op de informatie over React, VueJS en Angular, tot welk framework voel u zich intuïtief het meest aangetrokken?

Vraag 1

Afgaand op de informatie over React, VueJS en Angular, tot welk framework voel u zich intuïtief het meest aangetrokken?

a. Angular
b. React
c. VueJS
d. (Nog) geen idee.

Vragen?

Stel ze gerust

Bron: 123RF.

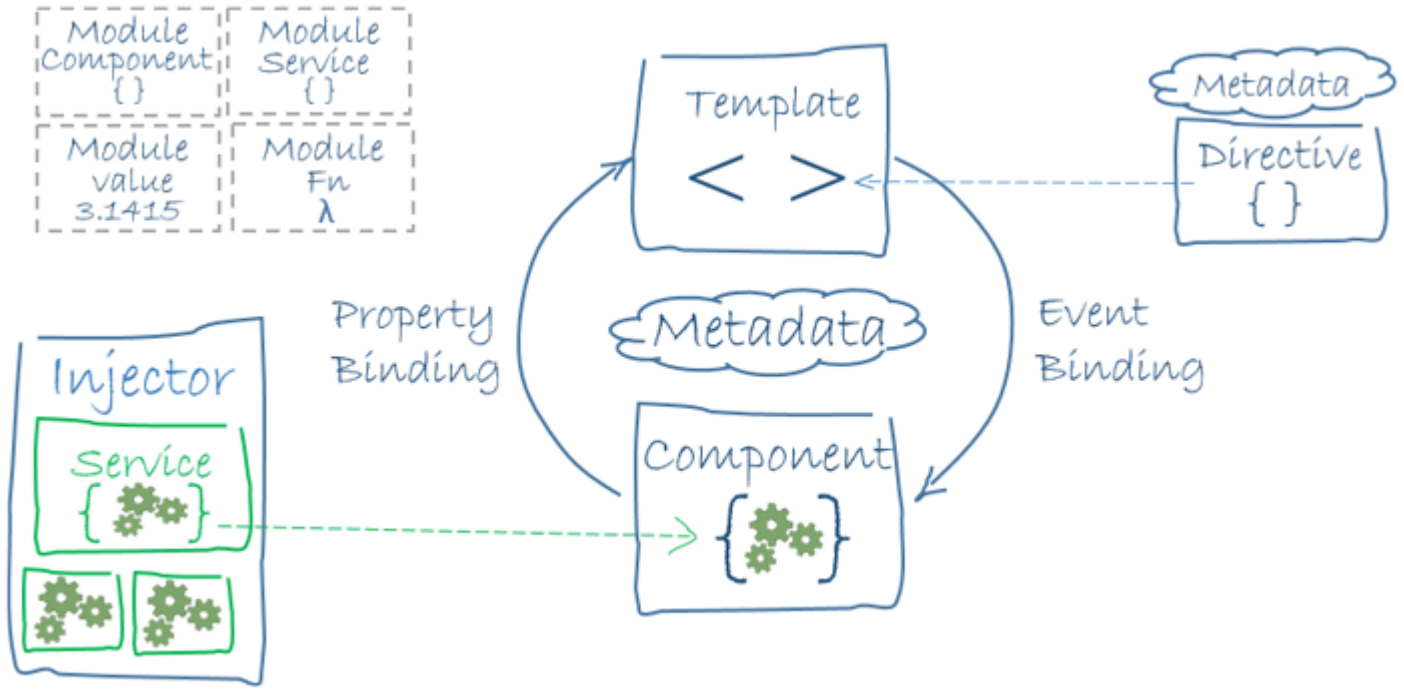
Onderdeel 2 – *Kernconcepten*

Reactive programming, asynchronous programming, data-oriented programming, state management, routing

Reactive programming

- In een reactive framework is er een change detection routine ingebouwd. Bij elke wijziging in de data die binnen de applicatie gebruikt wordt, zal de UI direct aangepast worden.
- RxJS is een JavaScript-bibliotheek die speciaal daarvoor ontwikkeld is.
- Gebruikt binnen Angular, React en VueJS.
- Binnen reactive programming gaat men uit van 'asynchronous data streams'. Data komt op enig moment binnen, wanneer en hoe vaak is niet van tevoren bekend. Dit kan externe data zijn (uit een database) maar ook interne data (de invoer van een gebruiker). Denk daarbij ook aan events (click events, mouse events, etc.).
- Net als bij events en event listeners is er een 'observer' (ook wel subscriber) die de data stream ontvangt in de vorm van een specifiek object, een observable. Het volgt het observer design pattern
- Met andere woorden, als ontwikkelaar hoeft u alleen de UI (het HTML template) te koppelen aan de data van de applicatie, het framework doet de rest en zorgt voor het dynamisch updaten van de UI. We noemen dit data binding. Dit kan ook met events (van de UI naar de applicatie). Dan spreken we van event binding.
- Een combinatie van de beide vormen zien we terug in VueJS en Angular als 'two-way binding'.
- Via zgn. directives (specifieke instructies) wordt het template verteld wat te doen met de binnengekomen data.

Voorbeeld databinding Angular



Bron: Angluar.io.

Voorbeeld code VueJS: data binding en event binding

HTML

```
<div id="app-5">
  <p>{{ message }}</p>
  <button v-on:click="reverseMessage">Reverse Message</button>
</div>
```

JS

```
var app5 = new Vue({
  el: '#app-5',
  data: {
    message: 'Hello Vue.js!'
  },
  methods: {
    reverseMessage: function () {
      this.message = this.message.split('').reverse().join('')
    }
  }
})
```

Hello Vue.js!

Reverse Message

Two way binding

HTML

```
<div id="app-6">
  <p>{{ message }}</p>
  <input v-model="message">
</div>
```

JS

```
var app6 = new Vue({
  el: '#app-6',
  data: {
    message: 'Hello Vue!'
  }
})
```

Hello Vue!

Hello Vue!

n.b.

In de HTML template zien we een ‘**directive** staan’, een template-instructie. Deze directives vertellen het template hoe te gedragen.

In VueJS wordt voor two-way binding de **v-model** directive gebruikt, in Angular de **ngModel** directive.

Angular en VueJS directives

Functie	Angular	VueJS
Voorwaardelijk tonen	ngIf	v-if / v-else
Herhalen (loop) voor elk dataobject	ngFor	v-for
Actie afhankelijk van een waarde	ngSwitch	-
Een class toekennen aan een HTML-element afhankelijk van een waarde (boolean)	ngClass	v-bind:class
Een style toekennen aan een HTML-element afhankelijk van een waarde (boolean)	ngStyle	v-bind:style

Asynchronous vs synchronous

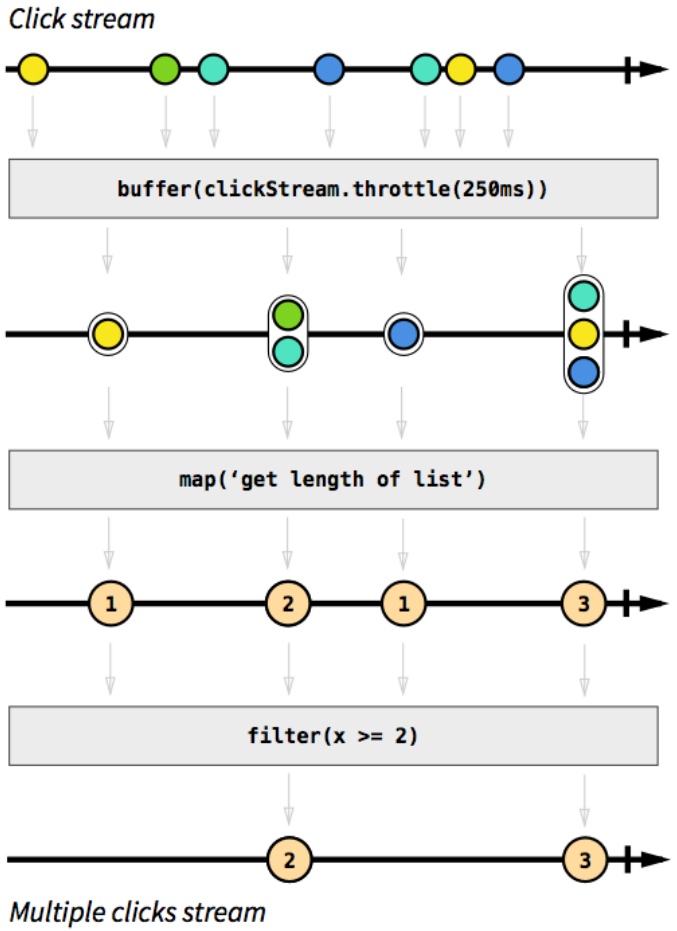
Synchroon:

- Een (traditionele) page refresh is altijd ‘synchroon’. Pas als de nieuwe pagina binnen is, kan de gebruiker weer interactie met de pagina hebben. Bij een actie van de gebruiker wordt een nieuwe pagina opgehaald. De gebruiker kan ondertussen niets doen en moet wachten. De lokale data loopt altijd synchroon met de server data.

Asynchroon:

- Bij het werken met data streams is nooit bekend wanneer iets gebeurt
  Voorbeeld: er wordt een request naar de API gestuurd om nieuwe data op te halen. Wanneer en of de data terugkomt is niet bekend, dit kan binnen milliseconden zijn maar ook uren (als er geen netwerkverbinding is).
- Dit geldt ook voor user events, dit is een ‘click stream’ die asynchroon binnenkomt
- De applicatie moet zo gebouwd zijn dat deze reageert op het moment dat data binnenkomt (of events binnenkomen), maar daar in ieder geval niet op gaat staan wachten. De UI mag niet geblokkeerd worden.
  Daar zorgt het reactive framework (bijvoorbeeld RxJS) voor.

Click stream



Bron: André Stalz, github.

Data-oriented programming en functional programming

- Data-oriented programming benadert codering anders dan OOP (object-oriented programming). In plaats van de nadruk op objecten ligt de nadruk op data en hoe deze is georganiseerd. Dit scheidt de functionaliteit en data. Ze zijn niet langer met elkaar verweven door een specifieke benadering. In DOP hebben uw functies een algemeen doel en worden ze toegepast op grote hoeveelheden gegevens.
- In JavaScript front-end frameworks wordt de data binnen de applicatie in data-objecten bewaard en daar geüpdatet. Deze data-objecten zijn de bron van de data en daarmee 'the source of truth'.
- Hieraan gerelateerd is het principe van functional programming. De functies die we schrijven zijn bij voorkeur 'pure functions' zonder eigen 'state', die alleen data manipuleren en retourneren die in de functie ingevoerd wordt, geen globale data.

```
let myName;
//impure function
function changeName(someJsonData) {
    this.myName = someJsonData.name; // fout: functie manipuleert globale data
}
changeName(getNewData());

// pure function
function changeName(someJsonData){
    return someJsonData.name; // correct: functie retourneert alleen een waarde
}
myName = changeName(getNewData());
```

Data object in een VueJS app

```
new Vue({
    el: "#app",
    data: {
        timer_txt : "",
        deck: [],
        boardClass : "",
        step : 0,
        numberOfSets : 0,
        card1 : "",
        card2 : "",
        score : 0,
        tries : 0,
        timer : "",
        minutes : 0,
        seconds : 0,
        hours : 0,
        time: "",
        clickEnabled : false,
        cardSet : [],
        active : false
    },
```

Pure vs. impure

```
let myName;
//impure function
function changeName(someJsonData) {
    this.myName = someJsonData.name; // fout: functie manipuleert globale data
}
changeName(getNewData());

// pure function
function changeName(someJsonData){
    return someJsonData.name; // correct: functie retourneert alleen een waarde
}
myName = changeName(getNewData());
```

State management en routing

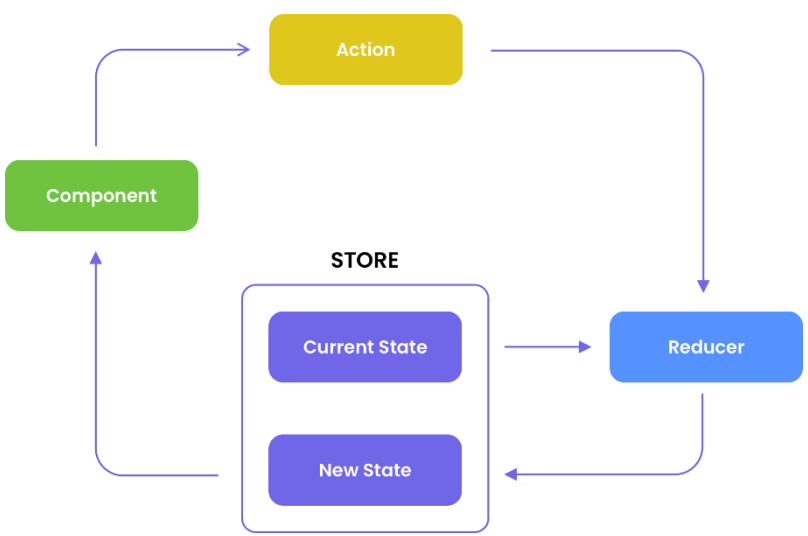
- Al deze principes zorgen voor een solide basis voor ‘state management’ oftewel het bijhouden van de status van de applicatie op elk moment.

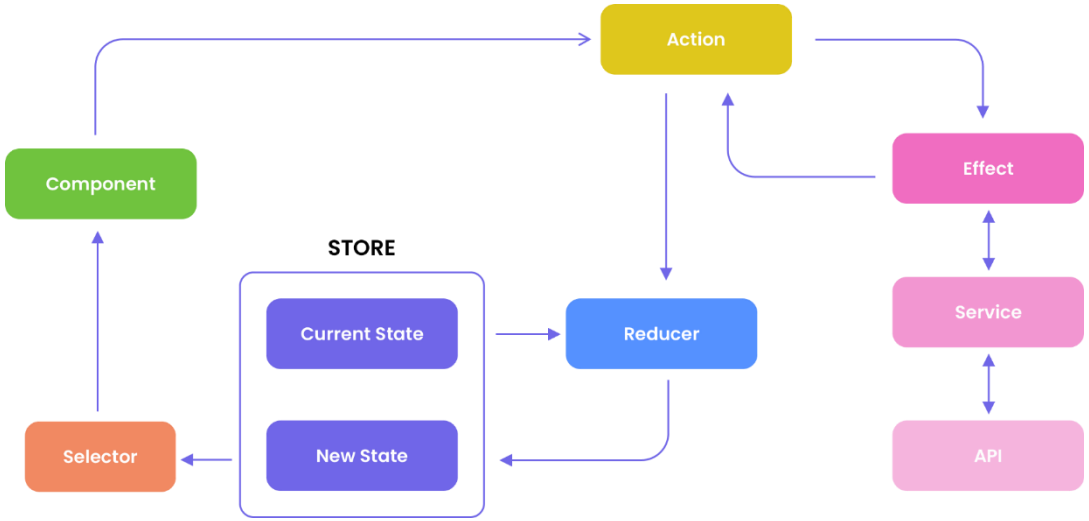
Voorbeelden:

- U bekijkt een YouTube-video op uw telefoon maar stopt halverwege. Later gaat u verder kijken op uw pc en u kunt verderkijken waar u bent gebleven.
- U rijdt met Google maps naar een locatie. Uw positie, uw ETA en uw route worden voortdurend bijgehouden en aangepast. Ook de verkeerssituatie kan zorgen voor een melding voor een alternatieve route.
- U update uw persoonlijke informatie in één deel van de applicatie. De aangepaste data wordt direct correct weergeven in een ander deel.
- De state van een applicatie wordt vastgehouden in data-objecten (data store).
- Middels ‘services’ kan de state gedeeld worden met alle onderdelen van de applicatie.
- Voor de geschiedenis van het navigeren binnen de applicatie gebeurt dit binnen de routing module.
- Alle andere soorten states (custom data) kunnen middels een extern framework (RxJS, NgRx) geprogrammeerd worden.

State management-regels

- De data mag maar één ‘source of truth’ hebben (de store).
- De data stroomt altijd maar één kant op.
- Elke verandering wordt als nieuwe ‘state’ opgeslagen, oude states blijven ongewijzigd.
- Middels een ‘reducer’ wordt alle relevante data in één state samengevoegd (gereduceerd).
- Change detection zorgt voor de update van de component.





Bron: 2muchcoffee, Medium.com.

Vraag 2

DOP (Data riented Programming) en OOP (Object Oriented Programming) zijn een soort van tegenhangers.

In welke benadering zal de data in de applicatie het meest versnipperd aanwezig zijn?

Vraag 2

DOP (Data riented Programming) en OOP (Object Oriented Programming) zijn een soort van tegenhangers.

In welke benadering zal de data in de applicatie het meest versnipperd aanwezig zijn?

a. DOP, omdat de data gescheiden bewaard en verwerkt moet worden.
b. OOP, omdat de data verstopt en verspreid zit in de objecten.
c. Dat hangt er helemaal van af hoe de programmeur zijn classes bouwt.
d. Ik heb geen flauw idee.

Vragen

Onderdeel 3 – *Live demo VueJs en Angular*

Lokaal ontwikkelen met MSC, de CLI, een Angular project inrichten, voorbeeldprojecten

Live demo

- Een VueJS-project
- Een Angular-project

Vragen

Bron: Pixabay.

Een extraatje
Demo memory game

De complete code (VueJS en Ionic)

Bedankt voor uw deelname!

Hierna verschijnt de evaluatie in beeld