Inleiding tot React

Leeruitkomst
De student kan uitleggen waarom er in een gegeven situatie voor het framework React gekozen is of gekozen kan worden. De student begrijpt JSX en kan uitleggen hoe JSX gebruikt wordt met React.

Prestatie indicatoren

- De student kan de toepassing en functionaliteiten van React beschrijven.
- De student kan de benodigde applicaties voor de verschillende besturingssystemen effectief installeren en operationeel maken zodat met React gewerkt kan worden.
- De student kan de verschillende elementen en principes van JavaScript beschrijven en aangeven wanneer deze toegepast kunnen worden.
- De student begrijpt de achterliggende technologie van JSX.
- De student kan de principes van JSX toepassen om werkende code te schrijven.
- De student kan schone code schrijven door principes toe te passen die clean code dicteren.

1. Introductie

1.1 Introductie tot React

React: een JavaScript-library voor het bouwen van gebruikersinterfaces.

Wat is React?

React is een manier om gebruikersinterfaces te bouwen. React maakt dit heel eenvoudig door elke pagina in stukken te knippen. We noemen deze stukjes componenten.

Hieronder staat een voorbeeld van het knippen van een pagina in componenten:

Elke sectie die hierboven is gemarkeerd, wordt als een component beschouwd. Maar wat betekent dit voor een ontwikkelaar?

Een React-component is een stukje code dat een deel van de pagina vertegenwoordigt. Elke component is een JavaScript-functie die een stuk code retourneert waardoor een stuk van een webpagina wordt getoond.

Om een pagina te bouwen roepen we deze functies in een bepaalde volgorde aan, stellen het resultaat samen en laten het aan de gebruiker zien.

Kenmerken van React

Op componenten gebaseerd
React gebruikt componenten om UI's te maken. Componenten lijken op HTML-elementen met toegevoegde functionaliteiten om ze herbruikbaar te maken. In React bouw je ingekapselde componenten die hun eigen staat beheren en vervolgens stel je deze componenten samen om complexe UI's te maken.
Leer eenmalig, schrijf overal
React maakt geen aannames over de rest van je technologiestack, dus je kunt nieuwe functies ontwikkelen in React zonder de bestaande code te herschrijven.

Voordelen van React

- Het renderen van componenten met React is zeer snel.
- React speelt goed met andere programmeertalen samen.
- React geeft de mogelijkheid om op elk gewenst moment code-elementen te hergebruiken, hetgeen een enorm tijdbesparend effect heeft.

React is een JavaScript framework. Dit framework is ooit gestart om de grootste verschillen tussen de verschillende browsers voor de meeste programmeurs onder de motorkap te verbergen en is langzamerhand uitgegroeid tot een van de meest gebruikte frameworks voor het ondersteunen van scherm gebruikersinteractie in HTML. Dit kan zijn voor browsers op telefoons, desktop computers en tablets. Het framework levert een standaard interface voor het afhandelen van grafische gebeurtenissen op het scherm. Dit kan zijn het verversen van data, die reeds op het scherm zijn getoond of het reageren op input. Ook is het mogelijk om gegevens uit verschillende bronnen te combineren en samen te voegen in functies die HTML als uitvoer hebben. Dat is dan ook een van de meest krachtige concepten binnen React: HTML en code zijn één.

In deze leerunit worden de JavaScript taal en de JavaScript Xml (JSX) extensie onder de loep genomen. We gaan ermee aan de slag en kijken hoe de taal werkt en hoe JavaScript samenwerkt met React om te komen tot een snelle manier van het koppelen van gegevens, die dan vervolgens aan de gebruiker worden getoond. Het leerdoel is dan ook: goed kunnen programmeren in JavaScript, het begrijpen van JSX en hoe deze samenwerken.

Naast het gebruik van JSX om HTML te genereren, gaan we ook een afbeelding creëren als Scalable Vector Graphics (SVG). Om een goede en mooie user interface te maken is het handig om te weten hoe je dit doet.

1.2 Introductie tot Scrimba

In deze module zullen we gebruik maken van interactieve coding screencasts. Hieronder vind je een voorbeeld met daarin instructies hoe Scrimba gebruikt wordt.

scrimba

2. Installatie van React

2.1 Node.js

Node.js is een softwareplatform voor het maken van (web)applicaties die snel werken en makkelijk uit te breiden zijn.

Node.js is het best te omschrijven als een JavaScript-interpreter. Dat betekent dat je JavaScript kunt ontwikkelen en testen zonder dat je daarbij een browser nodig hebt. De kracht van Node.js is dat het efficiënt, lichtgewicht en extreem schaalbaar is. Node.js bestaat uit 2 onderdelen: de Core en de Modules.

Core

Een key-techniek is dat Node.js JavaScript op de server kan uitvoeren. Hiervoor wordt gebruik gemaakt van de core, dit is de Chrome V8 JavaScript engine. Deze zogenaamde JIT-compiler (van Just In Time) compileert JavaScript naar machinecode voordat de code wordt uitgevoerd. Zo kan je dus JavaScript uitvoeren op een server als vervanger voor bijvoorbeeld PHP.

Daarnaast bevat de core een ingebouwde profiler die, waar nodig, optimalisaties doorvoert in de code. Ook garbage collection wordt door de core voor zijn rekening genomen.

Libuv

De modules bestaan uit een enkele C++ library met de naam Libuv. Deze neemt alle asynchrone I/O en de zogenaamde main event loop voor zijn rekening, zodat de V8 engine alle andere requests kan verwerken.

Node.js is bedoeld als een serverprogramma. Het komt dus niet met een mooie grafische interface. Node.js maakt gebruik van commando's die worden ingetypt in een commandline prompt. Je kunt een commandline applicatie starten door het programma cmd.exe te starten uit Windows. Je krijgt nu een zwart scherm met een knipperende cursor. Door nu het volgende in te typen: DIR gevolgd door een enter, geef je de computer een commando dat hetzelfde werkt als het oproepen van een bestandenlijst met de bestanden app.

Node.js is wel makkelijk te installeren. Daarvoor ga je naar https://nodejs.org. Daar zie je meestal twee versies: een Current versie en een Long Time Support (LTS) versie. Je mag beide kiezen alleen de leerstof in deze cursus gaat uit van de Current versie. Kies tijdens het installeren voor de standaard opties.

Als alles is gedownload en geïnstalleerd, dan kan er getest worden of de installatie is gelukt.

Testen van de installatie van Node.js

Het testen of Node.js daadwerkelijk goed is geïnstalleerd, kun je op de volgende manier uitvoeren:

De eerste test:

- Open command prompt (cmd.exe), als je met Windows werkt.
- Mac-gebruikers openen de terminal (hulpprogramma's).
- Toets in:

```
node -v
```

Als Node.js beschikbaar is, dan zul je een aantal nummers zien, die vooraf worden gegaan door de letter v en gescheiden zijn door punten, bijvoorbeeld v8.9.3. Als dit niet gebeurt, voer dan de installatie opnieuw uit met default instellingen.

De tweede test:

- Toets in:

```
npm -v
```

Het resultaat is hetzelfde zonder de letter v aan het begin.
Gebeurt er niets of krijg je een foutmelding, installeer dan een recentere versie van Node.js

Voor de derde en laatste test maken we een bestand aan met behulp van een text editor.

- Geef het bestand de naam hallo.js
- Bewaar dit bestand in de root map.
- In dit bestand schrijf je de volgende regel:

```
console.log('Hoera, hij doet het!');
```

Vervolgens ga je weer naar de terminal en typ je in een command prompt:

Node hallo.js

Als er op het scherm komt te staan: Hoera, hij doet het!, dan is de installatie geslaagd.
Komt er een foutmelding, herhaal dan de laatste test met een nieuw gevuld bestand of installeer Node.js opnieuw.

2.2 IDE's voor React

Er zijn verschillende IDE's voor React.

Om aan de slag te kunnen gaan met React zul je een Integrated Development Environment (IDE) nodig hebben die het framework ondersteunt. Hier zijn drie populaire opties:

- Atom
- Visual Studio Code
- Eclipse

Welke je kiest, is aan jou. Soms vraagt een werkgever om met een specifieke IDE te werken, dus het is aan te raden om een paar verschillende IDE's te proberen, zodat je op de hoogte bent van hoe ze van elkaar verschillen.

2.3 React Chrome plugin

De Chrome plugin helpt je bij het debuggen van je code.

De Chrome plugin helpt je bij het debuggen van de code die je maakt. Voor de installatie ga je naar de web store van Chrome en zoek je op die pagina naar React Developer Tools. Klik op Toevoegen aan Chrome (blauwe knop) en de extensie wordt geladen en ingeschakeld.

Je bent nu klaar om de plugin te gebruiken.

2.4 ESLint

ESLint is een plugin om gemakkelijk JavaScript errors te herkennen en op te lossen.

JavaScript is vooral gevoelig voor ontwikkelaarsfouten. Zonder het voordeel van een compilatieproces, wordt JavaScript-code meestal uitgevoerd om syntaxis of andere fouten te vinden. Linting-tools zoals ESLint stellen ontwikkelaars in staat om problemen met hun JavaScript-code te ontdekken zonder deze uit te voeren.

De belangrijkste reden dat ESLint is gemaakt, was om ontwikkelaars hun eigen lintingregels te laten maken. ESLint is ontworpen om alle regels volledig plugbaar te maken. De standaardregels worden geschreven zoals elke plugin-regel zou zijn. Ze kunnen allemaal hetzelfde patroon volgen, zowel voor de regels zelf als voor tests. Hoewel ESLint enkele ingebouwde regels bevat om het vanaf het begin nuttig te maken, kun je regels op elk moment dynamisch laden.

Het is aan te raden om deze plugin te installeren. De plugin kan je op weg helpen om JavaScript beter te begrijpen en tijd te besparen op het opspeuren van foutjes in je code.

De ESLint Config van AirBnb bevat ook ondersteuning voor React. Door deze te installeren wordt naast JavaScript ook de React code gecontroleerd.

Hieronder zijn de URL's te vinden naar de plugin:

ESLint webpage

https://eslint.org

Airbnb Config

https://www.npmjs.com/package/eslint-config-airbnb

3. JavaScript

JavaScript is misschien wel de belangrijkste programmeertaal van het web en JavaScript is misschien wel een van de meest onderkende elementen van het web. Uit onderzoek blijkt dat 95% van de websites gebruik maken van JavaScript.

3.1 JavaScript

Programmeren met JavaScript

JavaScript is misschien wel de belangrijkste programmeertaal van het web en JavaScript is misschien wel een van de meest onderkende elementen van het web. Uit onderzoek blijkt dat 95% van de websites gebruik maakt van JavaScript. Het is dus een essentiële tool voor elke software engineer. Des te meer omdat een groot aantal van de webapplicaties geen gebruik meer maakt van middleware zoals J2EE (java middleware stack) of .NET (Microsoft), de zogenaamde 3-tier stack. Echter meer en meer maken organisaties gebruik van een 2-tier stack gecombineerd met een Database as a Service (DaaS).

3.2 ES6

ECMA Script 6

Ook JavaScript kent versies. De versie van JavaScript in deze cursus is ES6. Deze versie wordt tegenwoordig in de meeste browsers ondersteund. ES6 heeft als voordeel een eenvoudige syntax, die goed te onderhouden is. Bovendien heeft de object oriëntatie nu een plaats in de taal, hetgeen goed ontwikkelen als software engineer mogelijk maakt.

3.3 JavaScript en Mocha

De geschiedenis van JavaScript

Tot voor kort waren JavaScript en HTML onlosmakelijk verbonden. Met de komst en volwassenheid van Node.js is de taal uitgegroeid tot een alternatief voor de gangbare ontwikkeltalen zoals Java, C# en Python. Des te meer omdat een groot aantal ontwikkelaars kennis heeft van JavaScript. In deze cursus wordt JavaScript gebruikt, waarvoor JavaScript bedoeld was: het manipuleren van HTML.

JavaScript is bedacht in de jaren negentig. Voordat JavaScript zijn intrede deed, was het web een statische bedoening. De webpagina's uit die tijd waren opgezet als documenten of flyers, veel tekst en weinig grafische elementen. De overgang van telefoon internet naar een bekabeld netwerk (in die tijd werd ADSL geïntroduceerd) maakte het mogelijk om meer grafische elementen en dynamische aspecten toe te voegen. Daarvoor ontwikkelde Netscape, een van de bedrijven die een browser verkocht, een scripting taal genaamd JavaScript. JavaScript was gebaseerd op C.

Mocha (zo heette JavaScript toen) was ook een antwoord op de marketing push van Sun (de toenmalige eigenaar van Java) om Java in een plugin te laten gebruiken in browsers. Java was echter meer een enterprise ontwikkeltaal en niet eenvoudig te leren. Om Java te gebruiken was een (relatief) zware plugin nodig en de interactie met de browser was minimaal.

JavaScript pakte dit hiaat in de behoefte langzaam op. Flash was een tijd lang zeer populair om websites dynamisch te maken, waarbij Flash na verloop van tijd dezelfde problemen kende als de Java plugin: te groot en te ingewikkeld. JavaScript is na initiële compatibiliteitsproblemen uitgegroeid tot een volwassen ontwikkelomgeving, die beschikbaar is op bijna alle computers uitgerust met een browser.

Met de uitvinding van de DaaS (Database as a Service) lijkt er een nieuwe tijd aangebroken voor JavaScript. Een aantal organisaties heeft ervoor gekozen om hun applicaties weer in een 2-tier vorm te bouwen waarbij DaaS ook de autorisatie verzorgt. Daardoor wordt de behoefte voor middleware tot een minimum beperkt.

3.4 DOM centraal

Het Document Object Model (DOM) staat centraal.

Eigenlijk bestaat er maar een ding in de wereld van JavaScript: het Document Object Model (DOM). Het object dat gewone mensen de pagina noemen. Op deze pagina staat HTML en de specifieke vorm van de pagina wordt door de browser vastgelegd in het DOM. Met JavaScript kunnen we het DOM gaan manipuleren, zoals elementen toevoegen, verwijderen en verplaatsen.

Het manipuleren van het DOM gebeurt op een browserspecifieke manier. Programmeurs waren in de beginjaren van JavaScript dan ook veel tijd kwijt met het debuggen van JavaScript in verschillende browsers. ReactJS, AngularJS, JQuery en verschillende andere frameworks zijn dan ook reacties op het debugproces. Allen op hun eigen manier en insteek.

Zoek eens naar de verschillende JavaScript Frameworks die er zijn.

3.5 Functies

Functies in JavaScript zien er als volgt uit:

```
function myFunction(p1, p2) {
   return p1 * p2; // The function returns the product of p1 and p2
}
```

De aanroep gebeurt dan door myFunction(3,4). Het resultaat moet dan 12 zijn.
Functies zijn nodig voor het samenbrengen van soortgelijke functionaliteit en hergebruik van dezelfde code.
Daarnaast maken functies het mogelijk om functionaliteit te groeperen en een relevante naam te geven.

```
function berekenSalaris(geslacht, leeftijd, functie, dienstjaren) {
   return <zeer ingewikkelde salaris berekening>;
}
```

Maak voor jezelf eens een berekening van bruto naar netto en vul de bovenstaande functie aan.
Gaat dat makkelijk of heb je iets nodig dat de status van de berekening voor je beheert?

Voorbeeld van een JavaScript functie

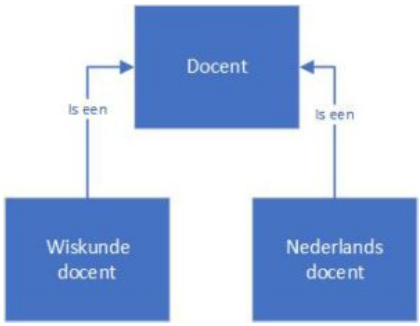
3.6 Objecten

Soms hebben een aantal functies een relatie met hetzelfde ding. In dat geval is het in JavaScript mogelijk om een object te creëren. Een object kun je het best vergelijken met een rol zoals een docent, politievrouw, vader of verzorger. Eén persoon kan verschillende rollen vervullen en bij een rol hoort specifiek gedrag dat die rol met zich meebrengt. Een politievrouw schrijft een bekeuring uit, dat doet een docent niet. Een docent beoordeelt werk van studenten. Soms kan het voorkomen dat gedrag bij verschillende rollen voorkomt, die een relatie met elkaar hebben. In het geval van JavaScript betekent dit dat klassen, dus de rollen, van elkaar overerven door middel van het keyword extends. De vuistregel is dat de meest specialistische klasse, dat is de laagste in de hiërarchie, het meeste gedrag heeft.

```
class Docent {
    constructor(les) {
```

```
        this.les = les;
    }
    geeft() {
        return '(' + this.les + ')';
    }
}

class WiskundeDocent extends Docent {
    constructor() {
        super('Wiskunde');
    }
    geeft() {
        return super.toString();
    }

    kalend() {

  return true;

    }
}

let wil = new WiskundeDocent();

wil.geeft();
```

Om gedrag te kunnen vertonen hebben objecten een status. Deze status wordt gevangen in attributen. In het bovenstaande voorbeeld wordt de status gevangen in het attribuut les. The parameter this wijst naar de instantie van de klasse zelf. Om dezelfde analogie te gebruiken met de rol: de instantie is in de rol-analogie de persoon, die de rol heeft. Er zijn immers meer lesgevende docenten op de wereld. De omschrijving super verwijst naar de bovenliggende generalisatie. De onderstaande figuur geeft het voorgaande grafisch weer.



3.7 Hello World!

Deze paragraaf gaat over Hello World, het minimale voorbeeld in React

Zie de onderstaande code:

```
document.getElementById("demo").innerHTML = "Hello JavaScript";
```

Daarbij vallen een aantal dingen op:

- Er zijn objecten beschikbaar die toegang geven tot de browser en de getoonde HTML.
- Er is geen start of signaal of methode waar iets begint.
- Het eerste stuk JavaScript dat de browser tegenkomt, wordt uitgevoerd.
- Direct assignment van een waarde en dus geen overbodige getters en setters.

3.8 Goede code (Clean code)

JavaScript is net zoiets als je schuur, je kamer, je huis of je bureau. Je kiest er zelf voor of het een rommeltje is of een nette opgeruimde boel. JavaScript is een makkelijke taal zonder al te veel regels. Het effect is dan ook dat veel programmeurs zich bevrijd voelen van de restricties die talen als Java, C(++) en Python met zich meebrengen. Het is dan ook goed om je programmeercompetenties vol in te zetten om tot goede code te komen:

- Gebruik SOLID.
- Codeer DRY.
- JavaScript is code en code wordt getest.
- Wees typesafe wanneer dat nodig is.
- Maak kleine programma's.
- Volg een applicatie-architectuur.
- Design by Contract.

Gebruik SOLID

SOLID staat voor:
Single responsibility: Je object, functie of script is alleen verantwoordelijk voor de eigen activiteit.
Open/closed principe: Open om uit te breiden door middel van extensies, gesloten voor modificatie.
Liskov principe: Je objecten moeten vervangen kunnen worden door een generalisatie, het gedrag van een object zal daardoor niet van gedrag veranderen.
Interfaces splitsen: Veel interfaces is beter dan één interface met heel veel methoden.
Dependency inversie: Je code zou zoveel mogelijk van generalisaties gebruik maken in plaats van specialisaties.

Codeer DRY

Don't Repeat Yourself (DRY) is een principe van softwareontwikkeling gericht op:

- het verminderen van herhaling van softwarepatronen;
- het vervangen door abstracties;
- het gebruik van data normalisatie om redundantie te voorkomen.

Het DRY-principe staat vermeld als: "Elk stukje kennis moet één enkele, ondubbelzinnige, gezaghebbende weergave binnen een systeem hebben". Het principe is geformuleerd door Andy Hunt en Dave Thomas in hun boek The Pragmatic Programmer. Wanneer het DRY-principe met succes wordt toegepast, vereist een wijziging van een enkel element van een systeem geen verandering in andere logisch niet-gerelateerde elementen. Bovendien veranderen elementen die logisch gerelateerd zijn, voorspelbaar en uniform en worden ze dus gesynchroniseerd.

JavaScript is code en code wordt getest

Software moet stabiel en correct zijn en werken zoals bedoeld. Software schrijven zonder bugs is vrijwel onmogelijk. Daarom is het zaak om zoveel mogelijk bugs zo snel mogelijk te vinden. Dat doen we door middel van unittesten. Ook voor React is er een mogelijkheid om je code te testen en dat vinden we zo belangrijk dat het er nog een paragraaf aan gewijd is.

Wees typesafe wanneer dat nodig is

De waarde '1' en 1 zijn gelijk en deze code:

```
let a = '1' == 1
document.getElementById("demo").innerHTML = a;
```

geeft als antwoord waar. Probeer maar.

.

```
let a = '1' === 1
document.getElementById("demo").innerHTML = a;
```

geeft als antwoord false. Probeer maar.

Typesafety inJavaScript is vooral belangrijk als het om vergelijkingen gaat.

Maak kleine programma's

Kleine stukjes code zijn beter te begrijpen dan hele pagina's vol, zeker in het geval dat je een applicatie met anderen maakt. Hoare (1969) geeft aan wat de minimale vorm van een applicatie is: {P} C {Q}. Daarbij is P de preconditie is, waaraan voldaan moet zijn om C uit te voeren. Q is het resultaat van C. C is de transformatie die ervoor zorgt, dat het resultaat Q wordt gerealiseerd. Door P buiten de software te plaatsen in een contract (zie Design by Contract) en alleen C en Q te programmeren wordt de softwarecode geminimaliseerd en daarmee het mogelijk aantal fouten.

Volg een applicatie-architectuur

Als iedere programmeur gebruik maakt van dezelfde broncode, dan is het vinden van een specifiek stukje broncode een stuk gemakkelijker.

Design by Contract (DbC)

DbC is een manier van samenwerken. Door op een juiste manier de code te voorzien van commentaar zorgt de code voor meer uitleg en dus minder source code. Dit gebeurt door in commentaarregels aan te geven:

- op welke manier een methode of klasse moet worden aangeroepen, zodat deze werkt;
- welke variabelen er aangepast gaan worden.

Invariantie is daarbij wel het streven. Zie de S en de O van SOLID.

Design by Contract wordt verder uitgediept op deze pagina.

4. JSX

JSX is de manier in ReactJs om HTML om te zetten in Javascript. Het grote voordeel is echter dat JSX gelijk gebruik kan maken van Javascript expressies.

4.1 JSX

JSX is een extensie van JavaScript voor het gebruik van XML of HTML in JavaScript.

JSX is een extensie van JavaScript voor het gebruik van XML of HTML in JavaScript.

Het meest eenvoudige gebruik van JSX is:

```
const element = <h1>Hello, world!</h1>;
```

Daarbij zijn de rood aangegeven elementen de HTML-elementen. De React precompiler (Babel) zal de bovenstaande broncode vertalen naar JavaScript zodat de code te gebruiken is als element. Buiten HTML zijn ook andere vormen van XML te gebruiken, zoals Scalable Vector Graphics (SVG): een manier om grafisch meer coherente elementen te maken. In HTML met zijn tekst gebaseerde insteek is dat erg moeilijk. Denk hierbij aan grafieken, kaarten, animaties of een mooie lay-out, die op elk scherm op de juiste wijze en in de beste kwaliteit wordt getoond.

Waarom JSX?

- Bij veel applicaties is de presentatie zeer verweven met de logica van de applicatie. Het ligt dus voor de hand om dit op computertaal niveau te integreren.
- Een applicatie is makkelijker te maken als dingen die bij elkaar horen ook als een blok zijn te gebruiken. In de software engineering wereld heet dit Separation of Concerns (Dijkstra, 1982). Daarom is het makkelijker als deze blokken zowel de schermopmaak als de logica bevatten. Deze blokken heten in React componenten.
- Het voorkomt aanvallen op de applicatie door middel van injectie van code.

Dijkstra, E. W. (1982). On the role of scientific thought. In *Selected writings on computing: a personal perspective* (pp. 60-66). Springer New York.

4.2 JSX-elementen

Om JSX dynamisch te maken is het mogelijk om stukjes JavaScript op te nemen in JSX.

Om JSX dynamisch te maken is het mogelijk om stukjes Javascript op te nemen in JSX. Dit zijn de zogenaamde expressies. Je kunt elk valide brokje JavaScript opnemen door dit brokje te plaatsen tussen accolades:

```
let username = “Jaap”;
```

```
const element = <h1>Hello, {username}!</h1>;
```

Het element zou in bovenstaand voorbeeld Hello Jaap bevatten. Andere voorbeelden van expressies zijn: 1+1, formatDate(birthdate), “een ” + “plus ” + “twee ” + “ is drie”, Math.PI etc.

Het verrassende is dat JSX ook weer vertaald kan worden in expressies en dat deze expressies kunnen worden gebruikt in andere expressies. Hetgeen weer leidt tot een groot aantal componenten, die kunnen worden gebruikt in een organisatie. Je zult dan ook zien dat in organisaties die ReactJs gebruiken de componenten vaak gedeeld worden tussen de verschillende applicaties, die de organisatie maakt.
Ook mag je meer geneste HTML-tags gebruiken.
Zie ook dat de h1-tags opgenomen zijn als kleine letters.
Dit is een voorwaarde: grote letters worden niet geaccepteerd door Babel (de precompiler van JSX).

Ook is het mogelijk om SVG te gebruiken in een JSX-statement:

```
const circle = <svg viewBox="0 0 120 120"><circle cx="60" cy="60" r="50"/></svg>;
```

De gemaakte elementen kunnen vervolgens aan een render worden gegeven:

```
ReactDOM.render(
  circle,
  document.getElementById('root')
);
```

Nog makkelijker is het om de elementen te gebruiken in componenten.

4.3 JSX-elementen en veiligheid

Injection safe
Het kan gebeuren dat je data die door derden worden aangeleverd, wil tonen in een JSX-expressie. Dat kan zonder dat je jezelf druk hoeft te maken dat die expressie geïnjecteerd kan worden met code van anderen. ReactJS zorgt ervoor dat geïnjecteerde code onschadelijk wordt gemaakt. Tijdens het vertalen door Babel (de preprocessor, die van JSX weer JavaScript maakt) wordt er rekening gehouden met injectie.

5. Andere frameworks

Welke andere frameworks zijn er en wanneer worden deze gebruikt.

5.1 AngularJS

AngularJS is een framework dat een soortgelijk doel nastreeft als ReactJS. Het verschil zit in het gebruik van componenten. ReactJS heeft de mogelijkheid om componenten te maken en te hergebruiken. AngularJS heeft die mogelijkheid niet en wordt daarom meer gebruikt voor webapplicaties die eenmalig zijn.

Als je meer wilt weten over AngularJS, dan is dit een goed startpunt.

5.2 VueJS

Naast Angular en React is er nog een framework dat vaak wordt gebruikt.

VueJS is een JavaScript-frontend framework dat is ontworpen om webontwikkeling te organiseren en te vereenvoudigen.

Het project richt zich op het toegankelijker maken van ideeën in webontwikkeling en UI-ontwikkeling (componenten, declaratieve gebruikersinterface, hot-reloading, debugging, enzovoorts). Het probeert minder uitgesproken te zijn en dus gemakkelijker voor ontwikkelaars.
VueJS heeft een incrementeel toepasbare architectuur. De kernbibliotheek is gericht op declaratieve rendering en componentsamenstelling en kan worden ingesloten in bestaande pagina's. Geavanceerde functies die vereist zijn voor complexe applicaties zoals routing, state management en build-tooling worden aangeboden via officieel onderhouden, ondersteunende bibliotheken en pakketten.

Meer informatie over VueJS kun je vinden op de website: https://vuejs.org

5.3 JQuery

JQuery is ondertussen weer een beetje op zijn retour, maar wordt nog regelmatig ingezet.

JQuery kun je beschouwen als een minimaal framework voor het maken van dynamische websites. Waar AngularJS en ReactJs faciliteiten hebben voor templates en componenten, heeft JQuery dit niet. AngularJS en ReactJs maken dan ook impliciet gebruik van JQuery om de DOM te manipuleren,

JQuery is van origine een framework dat de verschillende browsers uniform maakt. Voor meer informatie over JQuery zie: https://jquery.com/.