# React 1 – Basis

## *Les 1 – Introductie React & Ontwikkelomgeving*

---

## ? Leerdoelen

Na deze les kun je:

- uitleggen wat React is;
- uitleggen waarom React wordt gebruikt;
- het verschil beschrijven tussen Front-end en Back-end;
- een nieuw React-project maken met Vite;
- de belangrijkste React-bestanden herkennen;
- je eerste React-pagina aanpassen;
- je project opslaan in GitHub.

---

## ? Wat ga je bouwen?

Tijdens deze module werk je aan één project:

## ? Game Collection

Je bouwt stap voor stap een moderne webapplicatie waarin gebruikers games kunnen bekijken, zoeken en beheren.

Aan het einde van deze module kun je onder andere:

- □□ Games weergeven
- □□ Games zoeken

- ❤ Favorieten toevoegen
- ◫ Detailpagina's maken
- ◫ Componenten bouwen
- ⚡ Dynamische React-applicaties ontwikkelen

> afbeelding.png

---

# ? Waarom React?

Stel je voor dat je de Game Collection maakt met alleen HTML.

Voor iedere game zou je opnieuw moeten schrijven:

- titel;
- afbeelding;
- platform;
- knop;
- beschrijving.

Heb je 200 games? Dan schrijf je dezelfde code 200 keer.

Dat is natuurlijk niet handig. React lost dit probleem op.

React maakt gebruik van **componenten**.

Een component maak je één keer. Daarna gebruik je hem zo vaak als je wilt.

---

# ? Wat is React?

React is een **JavaScript-library** waarmee je moderne gebruikersinterfaces bouwt.

React is ontwikkeld door **Meta (Facebook)**.

React wordt gebruikt door onder andere:

- Facebook

- Instagram
- Netflix
- Discord
- Airbnb

React is tegenwoordig één van de meest gebruikte technieken voor Front-end ontwikkeling.

---

# ?? Benodigdheden

Voor deze module heb je nodig:

- Visual Studio Code
- Node.js (LTS-versie)
- Git
- GitHub-account

Controleer of alles correct is geïnstalleerd.

Open een terminal en voer uit:

```
node -v
npm -v
git --version
```

Wanneer alle versies zichtbaar zijn, ben je klaar om te beginnen.

---

# ? Componenten

Een React-applicatie bestaat uit kleine bouwstenen.

Deze noemen we componenten.

Onze Game Collection zou er bijvoorbeeld zo uit kunnen zien.

```
App

├── Header
```

```
├── SearchBar
├── GameList
│   ├── GameCard
│   ├── GameCard
│   ├── GameCard
│   └── GameCard
└── Footer
```

Iedere GameCard wordt slechts één keer gemaakt.
React hergebruikt deze component daarna automatisch. Dat scheelt enorm veel programmeerwerk.

> afbeelding.png

---

# ? Front-end versus Back-end

Een webapplicatie bestaat meestal uit twee delen.

| Front-end | Back-end |
|---|---|
| Alles wat de gebruiker ziet | Verwerking achter de schermen |
| HTML | Database |
| CSS | API |
| JavaScript | Server |
| React | Laravel (behandelen we in deel 2) |

Tijdens deze module richten we ons uitsluitend op de **Front-end**.

In de vervolgreeks koppelen we React aan een Back-end.

>

> afbeelding.png

---

# ?? Opdracht 1 – React installeren

Tijdens deze module gebruiken we **Vite**.

Vite is de officiële manier om een nieuw React-project te maken.

# Stap 1

Maak een nieuw React-project.

```
npm create vite@latest game-collection -- --template react
```

| | |
|---|---|
| **Which linter to use?** | **ESLint** |

> **Waarom ESLint?**
>
> ESLint is de meest gebruikte codecontrole voor React-projecten. Het helpt je om fouten sneller te ontdekken en zorgt ervoor dat je code volgens de gangbare React-standaarden wordt geschreven.

# ? Waarom gebruiken we Vite?

Om met React te kunnen werken hebben we een ontwikkelomgeving nodig.
Tegenwoordig wordt bijna altijd **Vite** gebruikt.

**Vite** is: Sneller, moderner, eenvoudiger, de standaard binnen nieuwe React-projecten.

Daarom gebruiken wij Vite tijdens deze module.

React gebruikt een techniek die **Virtual DOM** heet. Hierdoor wordt niet de hele pagina opnieuw opgebouwd.

Alleen het onderdeel dat verandert wordt opnieuw weergegeven. Daardoor voelt een React-applicatie snel en soepel aan.

# Stap 2

Open het project in Visual Studio Code. Zorg ervoor dat je de juiste directory hebt geopend.

Ga naar de projectmap.

```
cd game-collection
```

# Stap 3

Installeer alle packages.

```
npm install
```

# Stap 4

Start de ontwikkelomgeving in dien je het niet is gestart.

Terminal in VScode

```
npm run dev
```

Je ziet nu ongeveer:

```
> npx
> create-vite game-collection --template react

|
◇ Which linter to use?
| ESLint
|
◇ Install with npm and start now?
| Yes
|
◇ Scaffolding project in C:\REACT-1\game-collection...
|
◇ Installing dependencies with npm...

added 135 packages, and audited 136 packages in 8s

31 packages are looking for funding
 run `npm fund` for details
```

```
found 0 vulnerabilities
|
◇ Starting dev server...

> game-collection@0.0.0 dev
> vite

 VITE v8.1.4 ready in 428 ms

 ➜ Local: http://localhost:5173/
 ➜ Network: use --host to expose
 ➜ press h + enter to show help
```

Als alles goed is zie je via de link http://localhost:5173/  de standaard React pagina

> afbeelding.png

---

# ?? Opdracht 2 – Verken de projectstructuur

React-projecten bestaan uit verschillende mappen.

Zoek onderstaande bestanden en mappen op in je verkenner.

| Bestand | Functie |
| --- | --- |
| `src` | Hier staat bijna alle React-code |
| `App.jsx` | De hoofdcomponent van de applicatie |
| `main.jsx` | Startpunt van de React-app |
| `public` | Afbeeldingen en andere statische bestanden |
| `package.json` | Overzicht van alle packages |

In deze module ga je vooral hieraan werken.

# ?? Opdracht 3 – Pas de eerste React-pagina aan

Open:

```
src/App.jsx
```

Vervang de bestaande inhoud door:
Vul je eigen naam, studentennummer en klas in.

```
function App() {
 return (
 <>
 <h1>□□Game Collection</h1>

 <h2>Welkom bij mijn eerste React-applicatie!</h2>

 <p>Naam: Jouw naam</p>

 <p>Studentnummer: Jouw studentnummer</p>

 <p>Klas: Jouw klas</p>
 </>
 );
}

export default App;
```

Sla het bestand op.
De browser wordt automatisch bijgewerkt.

afbeelding.png

# ? Wist je dat?

React past de browser automatisch aan zodra je een bestand opslaat.

Dit heet **Hot Reloading**.

Daardoor zie je direct het resultaat van je wijzigingen, zonder de pagina handmatig te verversen.

---

# ? GitHub

Nu de applicatie werkt is het tijd om de eerste werkende versie op te slaan.

Open **Source Control** in Visual Studio Code.

Maak een commit met het bericht:

```
Les 1 - React omgeving gereed
```

Synchroniseer daarna met GitHub.
Vanaf nu maak je **na iedere werkende opdracht** een nieuwe commit.

---

# ? AI-opdracht

Gebruik ChatGPT.

Prompt:

> Leg uit wat React is alsof je het uitlegt aan een eerstejaars mbo Software Developer.

Vergelijk de uitleg met deze les.

Beschrijf in maximaal 100 woorden welke uitleg jij duidelijker vond.

---

# ? Kennischeck

1. Wat is React?
2. Waarom gebruiken veel bedrijven React?
3. Wat is het verschil tussen Front-end en Back-end?
4. Waarom gebruiken we Vite?
5. Wat is de functie van `App.jsx`?
6. Wat is de functie van `main.jsx`?
7. Waarom gebruiken we GitHub tijdens deze module?

---

# ? Inleveren

Lever de volgende onderdelen in via Canvas.

- ☐☐ Screenshot van de draaiende React-applicatie.
- ☐☐ Screenshot van `App.jsx`.
- ☐☐ Screenshot van de eerste GitHub-commit.
- ☐☐ PDF of TXT met de antwoorden op de kennischeck.
- ☐☐ De uitwerking van de AI-opdracht.

# *Les 2 – JSX: De taal van React*

---

## ? Leerdoelen

Na deze les kun je:

- uitleggen wat JSX is;
- HTML schrijven in JSX;
- JSX-elementen aanpassen;
- `className` gebruiken;
- JavaScript combineren met JSX;
- de eerste versie van de Game Collection uitbreiden.

---

## ? Wat ga je vandaag maken?

In de vorige les heb je jouw eerste React-applicatie gemaakt.

Vandaag ga je deze verder uitbreiden.

Je leert hoe React HTML en JavaScript met elkaar combineert. Aan het einde van deze les ziet jouw Game Collection er ongeveer zo uit.

De pagina bevat straks:

> **[Screenshot eindresultaat]**

Dat lijkt nog eenvoudig, maar je leert vandaag de belangrijkste basis van React.

# ? Wat is JSX?

React gebruikt **JSX**.

JSX lijkt heel erg op HTML. Toch is het geen HTML.

JSX is een uitbreiding op JavaScript waarmee je HTML kunt schrijven binnen je JavaScript-code.

React zet deze JSX vervolgens automatisch om naar gewone JavaScript-code.

Bijvoorbeeld:

```
<h1>Game Collection</h1>
```

React gebruikt **JSX**.

JSX lijkt heel erg op HTML, Toch is het geen HTML.

JSX is eigenlijk JavaScript waarin HTML geschreven mag worden.

React zet deze JSX later automatisch om naar gewone JavaScript-code.

---

# ? Waarom gebruikt React JSX?

Met JSX kun je veel makkelijker een pagina opbouwen.

Vergelijk eens.

Gewone **JavaScript:**

```
document.createElement(...)
```

**JSX:**

```
<h1>Game Collection</h1>
```

Welke leest prettiger? Vrijwel iedere React-programmeur kiest daarom voor JSX.

---

# ? JSX lijkt op HTML, maar...

Hoewel JSX veel op HTML lijkt, zijn er enkele verschillen.

| HTML | JSX |
| --- | --- |
| class | className |
| for | htmlFor |
| onclick | onClick |

Het belangrijkste verschil dat je vandaag gebruikt is:

```
className
```

---

# ?? Opdracht 1 – Breid de pagina uit

Open:

```
src/App.jsx
```

Voeg onder de titel toe:

```
<h2>Mijn favoriete games</h2>
```

Maak daarna een lijst.

```
<ul>

<li>□□Minecraft</li>

<li>□□Mario Kart</li>

<li>□□FIFA 26</li>

<li>□□Rocket League</li>

</ul>
```

Controleer of de browser automatisch wordt bijgewerkt.

> afbeelding.png

---

# ? JavaScript in JSX

Een groot voordeel van JSX is dat je JavaScript kunt gebruiken.

Dat doe je met **accolades**.

Bijvoorbeeld:

```
{4}
```

React laat vervolgens gewoon het getal zien.

---

# ?? Opdracht 2 – Toon het aantal games

Voeg onder de lijst toe.

```
<p>Aantal games: {4}</p>
```

De browser laat nu zien:

```
Aantal games: 4
```

Dat lijkt misschien eenvoudig.

Maar later gaan we hier echte berekeningen uitvoeren.

---

# ? Waarom gebruiken we accolades?

Alles wat tussen:

```
{

}
```

staat, wordt behandeld als JavaScript.

Bijvoorbeeld.

```
{10 + 5}
```

Resultaat.

```
15
```

---

# ?? Opdracht 3 – Rekenen in JSX

Voeg de onderstaande regels toe.

```
<p>2 + 3 = {2 + 3}</p>

<p>10 × 5 = {10 * 5}</p>
```

Controleer de browser.

React berekent de uitkomst automatisch.

---

# ?? Opdracht 4 – Gebruik een variabele

Maak boven de `return`.

```
const student = "Jouw naam";
```

Gebruik deze variabele.

```
<p>Developer: {student}</p>
```

React toont nu jouw naam.

> **[Screenshot browser]**

---

# ? Waarom gebruiken we variabelen?

Nu staat jouw naam nog één keer in de code.

Maar stel dat je dezelfde naam op meerdere plaatsen wilt tonen.

Dan hoef je alleen de variabele aan te passen.

React werkt daarna alles automatisch bij.

---

# ?? Opdracht 5 – Gebruik className

Open:

```
src/App.css
```

Voeg toe.

```
.title {

color:#2563eb;

}

.games {

margin-top:30px;

}
```

Pas daarna App.jsx aan.

```
<h1 className="title">

□□Game Collection

</h1>

<div className="games">
```

Controleer de browser.

---

# ? Wist je dat?

React gebruikt JSX inmiddels al meer dan tien jaar.

Vrijwel iedere React-applicatie die je op internet tegenkomt is opgebouwd met JSX.

---

# ? GitHub

Maak een commit.

```
Les 2 - JSX toegepast
```

Synchroniseer daarna met GitHub.

---

# ? AI-opdracht

Vraag ChatGPT:

> “ Leg uit waarom JSX geen gewone HTML is.

Vraag daarna:

> “ Waarom gebruikt React accolades `{}`?

Vergelijk de uitleg met deze les.

---

# ? Kennischeck

1. Wat is JSX?
2. Is JSX hetzelfde als HTML?
3. Waarom gebruikt React JSX?
4. Waarvoor gebruiken we `{}`?
5. Waarom gebruiken we `className`?
6. Wat doet Hot Reloading?
7. Wat is het voordeel van een variabele?

---

# ? Inleveren

Lever de volgende onderdelen in.

- ◻◻ Screenshot van de browser.
- ◻◻ Screenshot van `App.jsx`.
- ◻◻ Screenshot van `App.css`.
- ◻◻ Screenshot van de GitHub-commit.
- ◻◻ PDF of TXT met de antwoorden op de kennischeck.
- ◻◻ De uitwerking van de AI-opdracht.

---

Revision #11
Created 2026-07-10 09:02:27 UTC by yildiz
Updated 2026-07-10 13:02:23 UTC by yildiz