



React








Local development

Modules

Je werkt altijd in de **src** map. Al je bestanden zijn modules. Je code bevat **import** en **export** statements om componenten, css en images te laden.

De **module bundler** bouwt je app automatisch in de **public** map.

Op **codesandbox** gebeurt dit automatisch. Lokaal moet je zelf een module bundler installeren.

 public src About.js Apidemo.js App.js PokeCard.js Pokedex.js index.js logo.png styles.css package.json



PARCEL

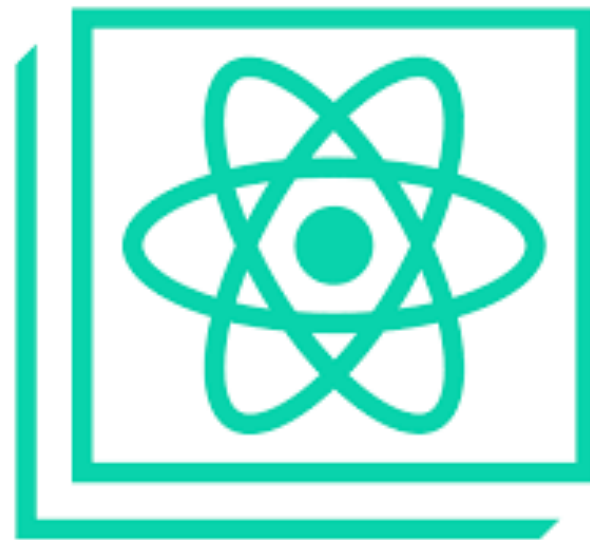
Blazing fast, zero configuration web application bundler



rollup.js



webpack



Create React App



[Home](#) > [Extensions](#) > React Developer Tools



React Developer Tools

Offered by: Facebook

★★★★★ 1,298 | [Developer Tools](#) | 👤 2,000,000+ users

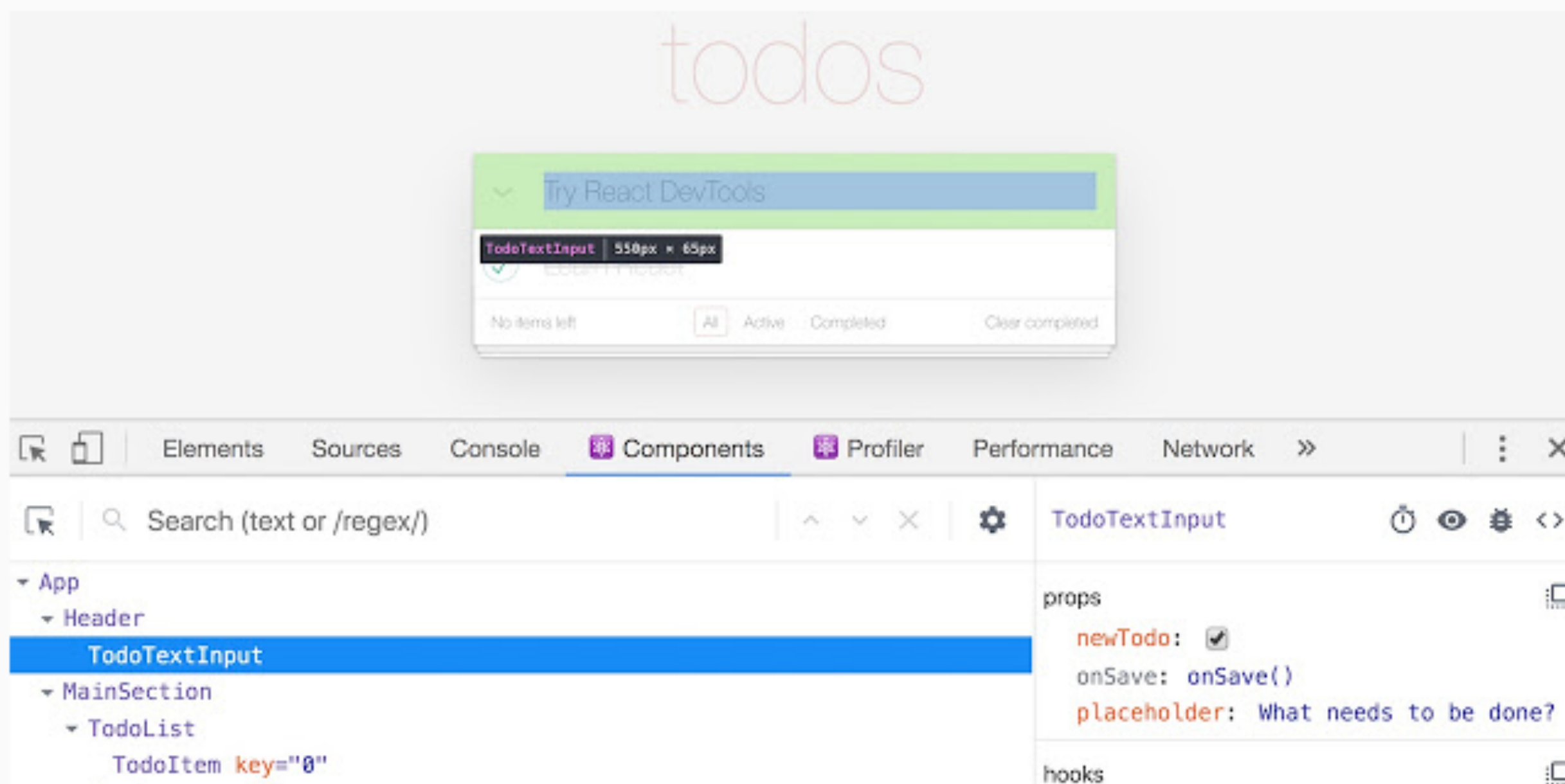
Remove from Chrome

Overview

Reviews

Support

Related





PARCEL

Blazing fast, zero configuration web application bundler

<https://parceljs.org>

<https://parceljs.org/recipes/react/>

npm project

Installeer NodeJS. Maak een project met **npm init -y**
Installeer parcel

Installeer react

Kopieer het voorbeeld
index.html, index.js, App.js van
<https://parceljs.org/recipes/react/>

Plaats deze files in de **src** folder

Start de live development server in watch mode. Open
<http://localhost:1234>

Je kan de server stoppen met ctrl+c

Als je project af is maak je de final build. Open deze in
localhost of upload naar je server.

```
npm init -y  
npm install --save-dev parcel
```

```
npm install react react-dom
```

```
> node_modules  
└─ src  
   ├── JS App.js  
   ├── <> index.html  
   ├── JS index.js  
   ├── {} package-lock.json  
   └── {} package.json
```

```
npx parcel src/index.html
```

```
parcel build src/index.html
```


npm start

Het is handig om Parcel's development en build commando's in je **package.json** te plaatsen

```
{  
  ...  
  "scripts": {  
    "start": "parcel src/index.html",  
    "build": "parcel build src/index.html --dist-dir docs --public-url ./"  
  },  
  ...  
}
```

```
npm run start
```

```
npm run build
```

optioneel

Typescript

Hernoem je .js modules naar **.ts** modules.
Installeer type information.

```
npm install @types/react @types/react-dom --dev
```

Sass

Installeer node-sass en je favoriete sass library

```
npm install node-sass  
npm install bulma
```



Github

package.json bevat informatie over alle libraries die je gebruikt

Deze libraries staan in de **node_modules** map

Het heeft geen nut om alle libraries naar je **GitHub** repo te uploaden. Maak daarom een **.gitignore** file.

Als iemand jouw GitHub repo uit checkt, kan hij/zij zelf de **node_modules** map aanmaken.

```
{
  "devDependencies": {
    "@babel/core": "^7.16.0",
    "@babel/plugin-transform-react-jsx": "^7.16.0",
    "parcel": "^2.0.1"
  },
  "dependencies": {
    "react": "^17.0.2",
    "react-dom": "^17.0.2"
  }
}
```

package.json

```
node_modules
src
docs
package.json
```

```
.DS_Store
.vscode
node_modules
```

.gitignore

```
npm install
```


Github Pages

Verander de output map van je project naar **docs** via **package.json**

Na het **build** commando heb je nu een **docs** map waarin je project staat.

Je kan nu je hele project pushen naar GitHub. Activeer Github Pages en kies de **main** branch, **docs** folder voor de live output!

```
{
  "name": "parcelreact-v2",
  "version": "1.0.0",
  "description": "",
  "scripts": {
    "start": "parcel src/index.html",
    "build": "parcel build src/index.html --dist-dir docs --public-url ./",
  },
  ...
}
```


package.json

	KokoDoko First commit	
	docs	First commit
	src	First commit
	.gitignore	First commit
	README.md	First commit
	package-lock.json	First commit
	package.json	First commit



GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

 Your site is ready to be published at <https://kokodoko.github.io/react-pokedex/>

Source

Your GitHub Pages site is currently being built from the /docs folder in the main branch. [Learn more.](#)

 Branch: main ▼

 /docs ▼

Save



React


Demo


JSON laden part 2





React Pokémon API


Amount of pokémon: 1118


Pidgeotto

normal flying
keen-eye
tangled-feet
big-pecks


Pidgeot

normal flying
keen-eye
tangled-feet
big-pecks


Rattata

normal
run-away
guts
hustle

Raticate

normal
run-away
guts
hustle

Spearow

normal flying
keen-eye
sniper

Fearow

normal flying
keen-eye
sniper

Ekans

poison
intimidate
shed-skin
unnerve

Arbok

poison
intimidate
shed-skin
unnerve

✗ Component updates

Een state update gebeurt niet meteen na de setState aanroep. Onderstaande code kan daardoor fout gaan!

```
import { useState } from 'react';

function BlogPost() {

  const [count, setCount] = useState(0);

  const handleClick(() => {
    setCount(count + 1)
    // de waarde van count is hieronder nog niet aangepast
    document.title = `Post was liked ${count} times`;
  });

  return (
    <div>
      <p>Liked {count} times</p>
      <button onClick={clickHandler}>Like</button>
    </div>
  );
}
```

✓ Component updates

Als je code wil uitvoeren nadat de state is veranderd, dan heb je **useEffect** nodig.

```
import { useState, useEffect } from 'react';

function BlogPost() {

  const [count, setCount] = useState(0);

  const clickHandler(() => {
    setCount(count + 1)
  });

  useEffect(() => {
    document.title = `Post was liked ${count} times`;
  });

  return (
    <div>
      <p>Liked {count} times</p>
      <button onClick={clickHandler}>Like</button>
    </div>
  );
}
```


✓ Component updates

Je kan een **array van variabelen meegeven**, dan wordt de **useEffect** functie alleen aangeroepen zodra een van die variabelen is veranderd.

Als je een **lege array** meegeeft, dan wordt de code alleen uitgevoerd wanneer het component voor het eerst opstart.

Je kan meerdere useEffect calls aanmaken.

<https://reactjs.org/docs/hooks-effect.html>

```
import { useState, useEffect } from 'react';

function BlogPost() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    document.title = `Liked ${count} times`;
  }, [count]);

  useEffect(() => {
    console.log("Starting up the blog post!");
  }, []);

  return (
    <div>
      <p>Liked {count} times</p>
      <button onClick={() => setCount(count + 1)}>Like</button>
    </div>
  );
}
```


Detail data laden

De pokeApi laadt in eerste instantie alleen een lijst van namen met urls.

```
[{
  name:"bulbasaur"
  url:"https://pokeapi.co/api/v2/pokemon/1/"
},{
  name:"ivysaur"
  url:"https://pokeapi.co/api/v2/pokemon/2/"
},{
  name:"venusaur"
  url:"https://pokeapi.co/api/v2/pokemon/3/"
}]
```

Je kan die url doorgeven als **prop** aan een **detail component**.

```
import { useState } from 'react';

function App() {

  const [pokemon, setPokemon] = useState(null)

  const loadJson = () => {
    //..
  }

  const allPokemon = pokemon.map((pok, index) => (
    <Detail key={index} url={pok.url} />
  ));

  return (
    <div>
      { allPokemon }
    </div>
  );
}
```

Detail component

Het detail component krijgt een url binnen via **props**:

`"https://pokeapi.co/api/v2/pokemon/1/"`

Het detail component kan vervolgens deze data weer laden met fetch.

```
import { useState } from 'react';

function DetailPage(props) {

  const [detail, setDetail] = useState(null)

  const loadJson = () => {
    fetch(props.url)
      .then(response => response.json())
      .then(data => setDetail(data))
      .catch(error => console.error("oh halps"))
  }

  return (
    <div>
      { detail.area }
      { detail.generation }
      { detail.baselevel }
    </div>
  );
}
```

Loading status

Zo lang de detail data nog aan het laden is kan je die niet weergeven.

Je kan de **conditional &&** gebruiken om de details alleen te tonen als die ook echt bestaan.

```
import { useState } from 'react';

function DetailPage(props) {

  const [detail, setDetail] = useState(null)

  const loadJson = () => {
    fetch(props.url)
      .then(response => response.json())
      .then(data => setDetail(data))
      .catch(error => console.error("oh halps"))
  }

  return (
    <div>
      { details && <p>{ details.name }</p> }
    </div>
  );
}
```

Loading status

Je kan met een IF statement een "loading" message renderen als het details object nog leeg is.

```
import { useState } from 'react';

function DetailPage(props) {

  const [detail, setDetail] = useState(null)

  const loadJson = () => {
    //...
  }

  if(!details) {
    return (<div>Still loading...</div>)
  }

  return (
    <div>
      <p>{ details.name }</p>
    </div>
  );
}
```

Starten met laden

Met behulp van **useEffect** kan je de **loadJson** functie aanroepen zodra het component voor het eerst in de DOM is gerenderd.

```
import { useEffect, useState } from 'react';

function DetailPage(props) {

  const [detail, setDetail] = useState(null)

  const loadJson = () => {
    fetch(props.url)
      .then(response => response.json())
      .then(data => setDetail(data))
      .catch(error => console.error("oh halps"))
  }

  useEffect(loadJson, []);

  return (
    <div>
      { details && <p>{ detail.name }</p> }
    </div>
  );
}
```

Maak de opdrachten van vorige week in je lokale React site.

Opdracht

Plaats je codesandbox project van vorige week in je lokale project.

Test of alles nog werkt!

Eevee

Pikachu

Mankey

Opdracht

Laad data van je eigen REST api.

Laad detail data in een `<Detail />` component.

Bouw je site met het **build** command.
De site verschijnt in je public map.

Als je RESTful service online staat, dan kan je de public folder van React ook online zetten! Bv. Met GitHub pages.

Bonus slides

- Troubleshooting
- Pagination
- Update parent state
- Animation
- Libraries

Secret slides

- CSS modules
- Children as props
- Composition
- Fragments
- Context

Zoek de secret PDF in Cum Laude



React

Bonus slides

Parcel Build: Troubleshooting 🤔

⚠️ Na build: problemen met "entry point" in "package.json" ? *Verwijder het **main** veld.*

⚠️ Build gaat goed, maar browser toont: "index.js23423 not found" ? *Voeg de **public-url** variabele toe aan je build script:*

```
{
  "name": "parcelreact-v2",
  "version": "1.0.0",
  "description": "",
  "main": "index.html",
  "scripts": {
    "start": "parcel src/index.html",
    "build": "parcel build src/index.html --dist-dir docs --public-url ./"
  },
  ...
}
```

package.json

⚠️ Na build: React is not defined: *Installeer de babel JSX plugin handmatig.*

```
npm install -D @babel/core @babel/plugin-transform-react-jsx
```

⚠️ Na build: parcelRequire is not defined: *dit is een bug in parcel. Dirty fix:*

```
<body>
  <script>
    window.global = window;
    var parcelRequire;
  </script>
  <script type="module" src="index.js"></script>
</body>
```

index.html

Pagination 🤯💥😱

Als de PAGE variabele verandert, wordt de JSON opnieuw geladen.

```
import { useState, useEffect } from 'react';

function BlogPost() {

  const [page, setPage] = useState(0);

  const loadJson = () => {
    let url = `https://pokeapi?page=${page}`
    //...
  }

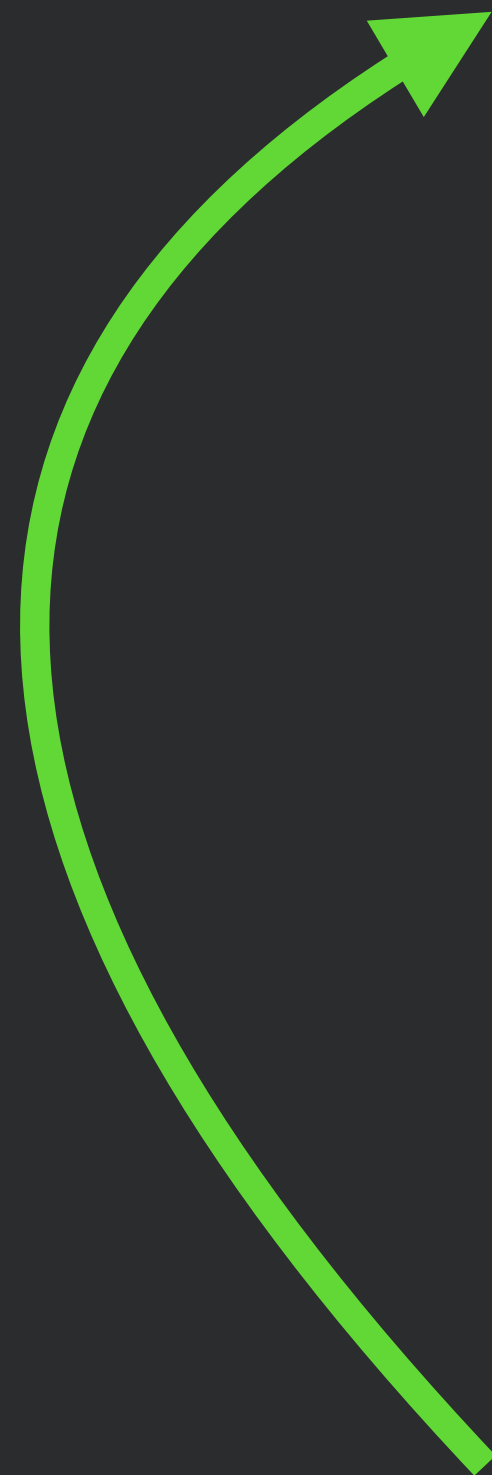
  useEffect(loadJson, [page])

  return (
    <div>
      <p>Page {page}</p>
      <button onClick={() => setPage(page + 1)}>Next</button>
    </div>
  );
}
```

Update parent state

Je kan een event handler meegeven als een prop.

Dit is de manier om de **state** van een **parent** te veranderen vanuit een **child**!



```
import About from "../About";

export default function App() {

  const [likes, setLikes] = useState(0);

  const likePost = () => {
    console.log("the total number of likes increased!")
    setLikes(likes + 1)
  }

  return (
    <div className="App">
      <h1>React Tutorial</h1>
      <About name="Erik" clickHandler="likePost"></About>
      <About name="Bas" clickHandler="likePost"></About>
    </div>
  );
}
```

```
export default function About(props) {
  return (
    <div className="About">
      <h4>About {props.name}</h4>
      <button onClick={props.clickHandler}>Like</button>
    </div>
  );
}
```

Animation

Animaties zijn belangrijk om de gebruiker te laten zien dat de state is veranderd.

In dit voorbeeld heeft het `` item uit de array een fade-in animatie

Door de **animationDelay** steeds iets hoger te maken, zullen de items een voor een infaden.

```
import { useState, useEffect } from 'react';

export function Pokedex() {
  const [pokemon, setPokemon] = useState(["Eevee", "Pikachu", "Poliwhirl"]);

  const listItems = pokemon.map((name, i) =>
    <li key={i} style={{animationDelay: `${i*0.4}s`}}>{name}</li>
  );

  return (<ul>{listItems}</ul>);
}
```

CSS

```
li {
  opacity: 0;
  transform: translateY(10px);
  animation: fadein forwards 0.4s ease-out;
}

@keyframes fadein {
  from {
    opacity: 0;
    transform: translateY(10px);
  }
  to {
    opacity: 1;
    transform: translateY(0px);
  }
}
```

Router

Met React router kan je deeplinken naar een bepaalde state van je applicatie.

<https://reactrouter.com>

```
<App>
  <Sales>
    <Invoices />
  </Sales>
</App>
```



example.com/sales/invoices

● Fastbooks

Overview

Subscriptions

Invoices

Customers

Deposits

Dashboard

Accounts

Overdue: \$10,800

Due soon: \$62,000

Material UI

React component UI library

<https://mui.com>

December 2021



S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Yesterday

Today

Tomorrow



50°C

25°C

Buy now

Buy now



React



Javascript



Spring

Animaties voor je componenten

<https://react-spring.io>



Framer

UI Animation

<https://www.framer.com/motion/>

Framer Motion

A production-ready motion library for React. Utilize the power behind Framer,

React Native

Bouw native iOS / Android apps

<https://reactnative.dev>



Gesture

Drag, swipe gestures

<https://use-gesture.netlify.app>



@use-gesture

Docs API GitHub Twitter Settings

move it, zoom it, drag it, scroll it,
pinch it



@use-gesture is the only gesture lib you'll need.

Browse Docs

Next.js

Server side rendering voor React

<https://nextjs.org>

The React Framework for Production

Next.js gives you the best developer experience with all the features you need for production: hybrid static & server-rendering, TypeScript support, smart bundling, route pre-fetching, and more. No config needed.

Recoil

Een externe state manager

<https://recoiljs.org>

Recoil

A state management library for React

Get Started