

Peter Eijgermans



@EijgermansPeter

React

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

Contents

What is a component in React?

JSX

Function versus Class component

Tutorial Building product cart

useState hook

useReducer hook

useEffect hook

Custom hooks



Clone this repo

```
git clone https://github.com/petereijgermans11/react-hooks
```



What is a component?





















Components are **independent** and **reusable** bits of code



React components?

Class component **versus** ***Function*** components



	Functional Component	Class Component
Use state inside component		
Use lifecycle methods		
Stateful		
Smart component		
Sharing state with Hooks		
Sharing Logic		
Easy readable		
Easy testable		
Using Hooks		
Functional programming style		

Syntax class component

```
class Product extends React.Component {  
  render( ) {  
  }  
}
```

IN HTML:

```
<Product />
```



Syntax class component

```
class Product extends React.Component
  render( ) {
    return <div>I am a {this.props.message} Product</div>;
  }
}
```

IN HTML:

```
<Product message="nice" />
```



Syntax function component

```
function Product(props) {  
    return <div> { props.message } </div> <== JSX  
}  
  
export default Product;
```



Components are like JavaScript functions

$$\text{fn}(d) = V$$


Rendering a Component

```
ReactDOM.render(  
  <Product />,  
  document.getElementById('root')  
);
```

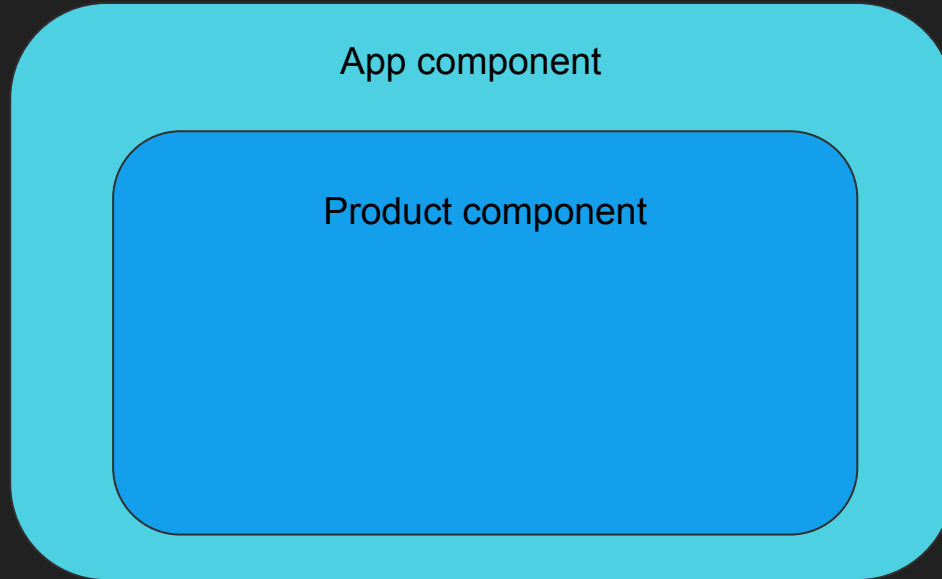


index.html

```
<div id="root"> </div>
```



Components in components



Components in components

```
function Product(props) {  
  
  return <h2>I am a {props.message} Product!</h2>;  
}  
  
function App() {  
  return (  
    <>  
      <h1>My shopping cart:</h1>  
      <Product message="red" />  
    </>  
  );  
}  
  
ReactDOM.render(<App />, document.getElementById('root'));
```



Components in files

create a new file with a **.js** file extension



Steps tutorial

Make Product component with Hooks:

Shopping cart

Products part to select

What are hooks?

`useState` hook for Shopping cart

`useReducer` hook

`useEffect` hook

Make a `Custom` hook



What are hooks?

Hooks allow us to "hook" into React features such as **state**
and can handle **side effects**



Why hooks?

Hooks let you split one component into smaller functions

Easy testing

Reuse stateful logic



useState Hook



useState Hook

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

```
const [color, setColor] = useState( "red" );
```



useState Hook

```
import { useState } from "react";
import ReactDOM from "react-dom";

function FavoriteColor() {
  const [color, setColor] = useState("red");

  return (
    <>
      <h1>My favorite color is {color}!</h1>
      <button
        type="button"
        onClick={() => setColor("blue")}>Blue
      </button>
    </>
  )
}

ReactDOM.render(<FavoriteColor />, document.getElementById('root'));
```



Hook Rules

Hooks can only be called inside React function components

Hooks can only be called at the top level of a component

Hooks cannot be conditional



Shopping Cart

18 total items

Total price: 10000 Euro



3



5



10



Checkout



Create a Product component

```
import React from 'react';
import './Product.css';

export default function Product() {
  return(
    <div className="wrapper">
      <div>
        Shopping Cart: 0 total items.
      </div>
      <div>Total: 0</div>

      <div className="product"><span role="img" aria-label="gitar">🎸</span></div>
      <button>+</button>
      <button>-</button>
    </div>
  )
}
```

Shopping Cart

18 total items

Total price: 10000 Euro



3



5



10



Checkout



useState Hook

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

```
const [cart, setCart] = useState( [ ] );
```

```
const [total, setTotal] = useState(0);
```



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

const products = [
  {
    emoji: '\uD83C\uDFB8',
    name: 'gitar',
    price: 500
  },
  {
    emoji: '\uD83C\uDFB7',
    name: 'saxophone',
    price: 1200,
  },
];

export default function Product() {
  const [cart, setCart] = useState([]);
  const [total, setTotal] = useState(0);

  function add(product) {
    setCart(current => [...current, product.name]);
    setTotal(current => current + product.price);
  }

  return(
    <div className="wrapper">
      <div>Shopping Cart: {cart.length} total items.</div>
      <div>Total: {total}</div>
      <div>
        {products.map(product => (
          <div key={product.name}>
            <div className="product">
              <span role="img" aria-label={product.name}>{product.emoji}</span>
            </div>
            <button onClick={() => add(product)}> + </button>
            <button> - </button>
          </div>
        ))}
      </div>
    </div>
  )
}
```

useReducer Hook



useReducer Hook

`useReducer(<reducer-function>, <initialState>)`

`||`
`∨`

```
function cartReducer(state, product) {  
  return [...state, product.name]  
}
```



useReducer Hook

```
import React, { useReducer } from 'react';  
  
const [cart, setCart] = useReducer(cartReducer, []);  
const [total, setTotal] = useReducer(totalReducer, 0);
```

```
function cartReducer(state, product) {  
  return [...state, product.name]  
}
```

⇐= Reducer functie

```
function totalReducer(state, product) {  
  return state + product.price;  
}
```

⇐= Reducer functie



```

function cartReducer(state, product) {
  return [...state, product.name]
}

function totalReducer(state, product) {
  return state + product.price;
}

export default function Product() {
  const [cart, setCart] = useReducer(cartReducer, []);
  const [total, setTotal] = useReducer(totalReducer, 0);

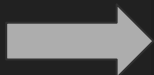
  function add(product) {
    setCart(product);
    setTotal(product);
  }

  return(
    <div className="wrapper">
      <div>
        Shopping Cart: {cart.length} total items.
      </div>
      <div>Total: {total}</div>
      <div>
        <div>
          {products.map(product => (
            <div key={product.name}>
              <div className="product">
                <span role="img" aria-label={product.name}>{product.emoji}</span>
              </div>
              <button onClick={() => add(product)}>+</button>
              <button>-</button>
            </div>
          ))}
        </div>
      </div>
    </div>
  )
}

```


Action - object

```
export default function Product() {  
  const [cart, setCart] = useReducer(cartReducer, []);  
  
  function add(product) {  
    setCart(product);  
  }  
}
```



```
export default function Product() {  
  const [cart, setCart] = useReducer(cartReducer, []);  
  
  function add(product) {  
    setCart({ product, type: 'add' });  
  }  
  
  function remove(product) {  
    setCart({ product, type: 'remove' });  
  }  
}
```

```
function cartReducer(state, action) {
  switch(action.type) {
    case 'add':
      return [...state, action.product];
    case 'remove':
      const productIndex = state.findIndex(item => item.name === action.product.name);
      if(productIndex < 0) {
        return state;
      }
      const update = [...state];
      update.splice(productIndex, 1)
      return update
    default:
      return state;
  }
}
```

```
export default function Product() {
  const [cart, setCart] = useReducer(cartReducer, []);

  function add(product) {
    setCart({ product, type: 'add' });
  }

  function remove(product) {
    setCart({ product, type: 'remove' });
  }
}
```



Demo



useEffect hook



useEffect hook

The `useEffect` Hook allows you to perform **side effects** in your components.

```
useEffect(<function>, <dependency>)
```



useEffect hook

```
export default function Product() {  
  const [cart, setCart] = useReducer(cartReducer, []);  
  
  useEffect(() => {  
    |  | setProducts(fetchProductData())  
  }, []);  
  
  const [products, setProducts] = useState([]);  
}
```

Demo



No dependency passed:

```
useEffect(() => {
```

```
  //Runs on every render
```

```
});
```



No dependency passed:

```
useEffect(() => {
```

```
  //Runs on every render
```

```
});
```

An empty array:

```
useEffect(() => {
```

```
  //Runs only on the first render
```

```
}, []);
```



No dependency passed:

```
useEffect(() => {  
    //Runs on every render  
});
```

An empty array:

```
useEffect(() => {  
    //Runs only on the first render  
}, []);
```

Props values as dependency:

```
useEffect(() => {  
    //Runs on the first render  
    //And any time any dependency value changes  
}, [prop]);
```



Custom hooks



Custom hooks

For logic that needs to be **reused** by multiple components



How to reuse useState hook?

Create a separate javascript file for the hook(s)

Return the data and functions from our custom hook

Import the custom hook in your component



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

const useStatesHooks = () => {
  const [cart, setCart] = useReducer(cartReducer, []);
  const [products, setProducts] = useState([]);
  return { cart, products, setCart, setProducts };
}

function cartReducer(state, action) {
  switch(action.type) {
    case 'add':
      return [...state, action.product];
    case 'remove':
      const productIndex = state.findIndex(item => item.name === action.product.name);
      if(productIndex < 0) {
        return state;
      }
      const update = [...state];
      update.splice(productIndex, 1)
      return update
    default:
      return state;
  }
}

export default useStatesHooks;
```

Use custom hook in your component

```
import useStatesHooks from './useStatesHooks';

export default function Product() {
  const { cart, products, setCart, setProducts } = useStatesHooks();
```

Demo



GET request using `axios` with React hooks

Installing Axios with npm

```
npm install axios
```

```
import axios from 'axios';
```

```
useEffect(() => {  
  axios.get('https://Some_URL')  
    .then(response => setProducts(response.ProductsData));  
}, []);
```

Opdracht Axios

getting local JSON data with axios

```
useEffect(() => {  
  axios.get('data/data.json')  
    .then(response => {  
      setProducts(response.data)  
    })  
});  
, []);
```

ReactJS to TypeScript

Add TypeScript

Add tsconfig.json

Start simple

Convert all files

Increase strictness

Clean it up

ReactJS to TypeScript

Summary

What is a component in React?

JSX

Function versus Class component

useState hook

useReducer hook

useEffect hook

Custom hooks




[Edit Profile](#)

Peter Eijgermans

Java WebDeveloper, Ordina

Peter is an adventurous and passionate CodeSmith and Front-end developer at Ordina Netherlands. He likes to travel around the world with his bike. Always seeking for the unexpected and unknown. For his job he tries out the latest techniques and frameworks. He loves to share his experience by speaking at conferences over the World and writing for the Dutch Java magazine and DZone. He believes that you as the Front-end developer are the spider in the web to bring the user, the team and the product together.

📍 Culemborg, NL 🏆 Gold Member

For inquiries regarding your profile data, visit our [GDPR info page](#).

Peter Eijgermans's Content


[Articles](#)
[Links](#)
[Comments](#)
[★ Saved](#)
[Awards\(0\)](#)


Hooks by Example: Convert a Tesla Battery Range Calculator to Functional... 👍 Like (9) +



In this article, I change the TeslaBattery Class Component to a Functional Component with React hooks.

2021-10-13 by Peter Eijgermans · Web Dev · 23800 Views



Micro Frontends With Example 👍 Like (14) 2

Monolithic frontends are difficult to maintain, develop, test, and deploy. The solution is micro frontends. It is a type of architecture that can increase effectiveness and efficiency across teams.

2021-07-27 by Peter Eijgermans · Microservices · 25136 Views

Any questions



@EijgermansPeter

