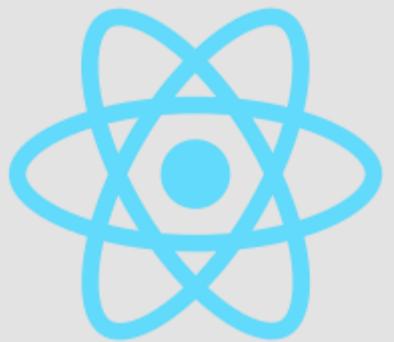
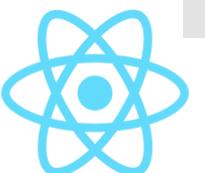


# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb

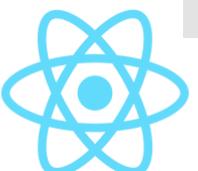


# Course Goals



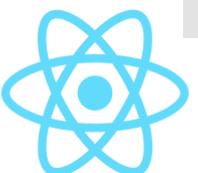
# Course Goals

- **Getting started** with functional components and hooks
- Go **beyond** the basic use cases



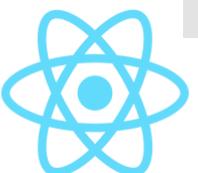
# Why this course?

- React hooks are deceptively **simple**
  - But you can still go wrong with them



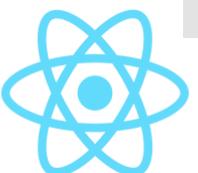
# Basic Hooks

- We will cover **the basics** of React hooks first



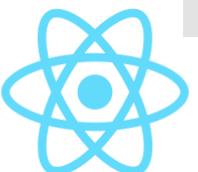
# Custom Hooks

- Creating **custom hooks**
  - What can't be done with custom hooks



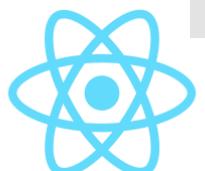
# The Rules

- Why do you need to **follow the rules?**
  - Take a look under the covers of hooks and find out



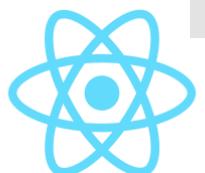
# Beyond the Basics

- We are going **beyond** the basic hooks
  - When should you use the more advanced hooks?

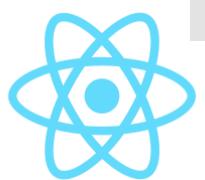


# Putting it all together

- Build a more **complex example**
  - A Formik like forms over data library
  - Combining hooks with other React features

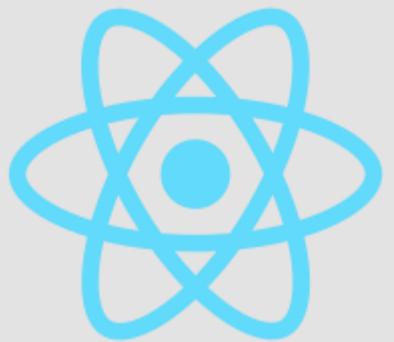


See you in the next video

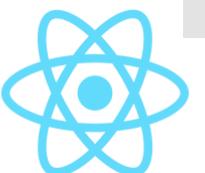


# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



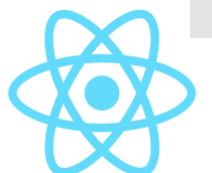
# Personal introduction



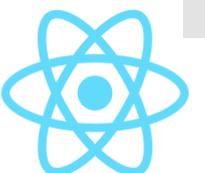


# Maurice de Beijer

Independent software developer and trainer



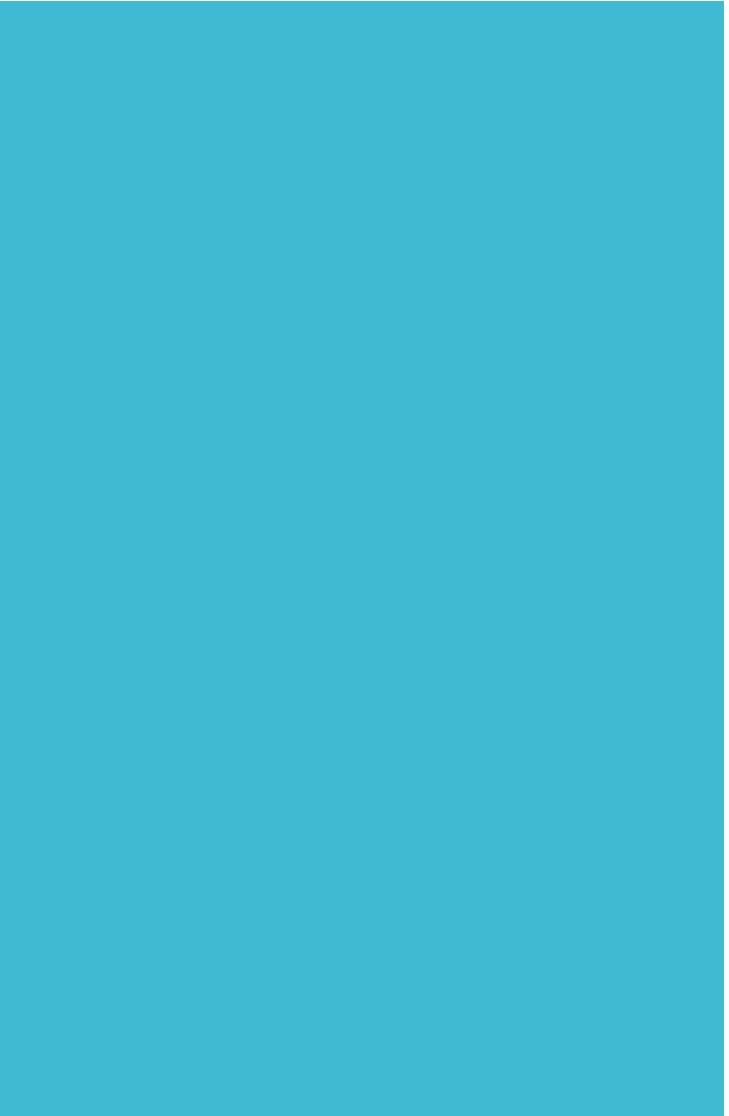
# The Netherlands



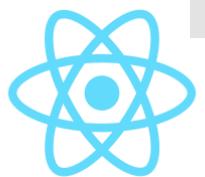








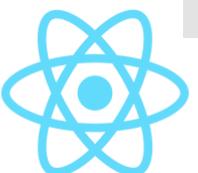
Happily married

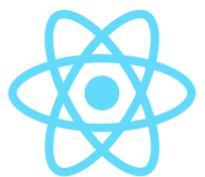




# Independent software developer & instructor

Since 1995

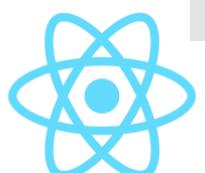




# The React Newsletter

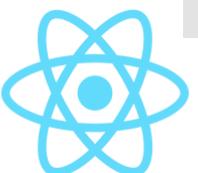
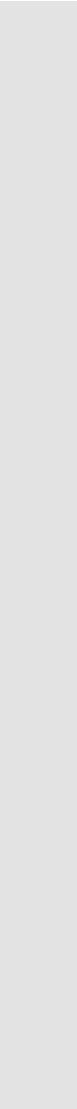


The screenshot shows an email from "The React Newsletter". The header includes links for "Subscribe", "Past Issues", "Translate", and "RSS". The main content features the React logo and the word "React". Below that, it says "The React Newsletter" and "Hi Maurice,". A thumbnail image titled "Form validation with React Hooks WITHOUT a library" shows a registration form and a blue background with the text "Form validation with React Hooks" and a fishhook icon. The text below the thumbnail reads: "In the early days of the Internet, HTML forms were the first way of interacting with websites. The internet has come a long way since, but forms are still an essential component of any web application. In React, you can validate forms in many different ways. There are some libraries out there that intend to make this task easier for you."



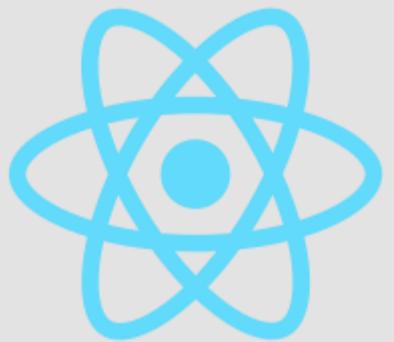


See you in the next video



# React Hooks Tips Only the Pros Know

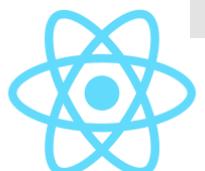
Maurice de Beijer - @mauricedb



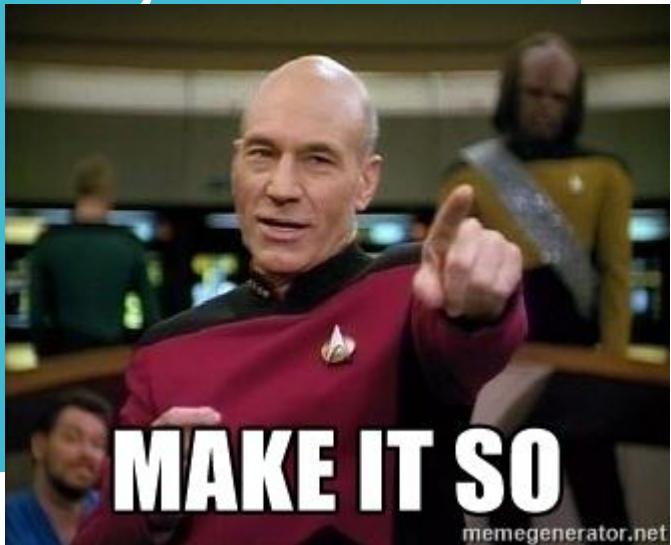
# Prerequisites

Install Node & NPM

Install the GitHub starter repository

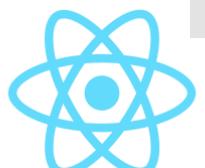


# Following Along



- Slides:
  - <http://theproblemsolver.nl/react-hooks-tips-only-the-pros-know-course.pdf>
- Starter repository:
  - <http://bit.ly/react-hooks-tips-only-the-pros-know-code>

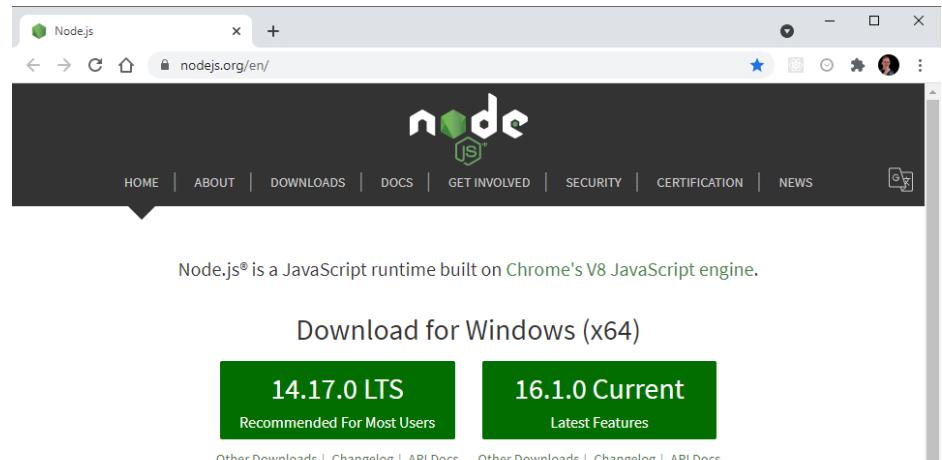
```
12  export function usePerson(
13    initialPerson: Person
14  ): [Person | null, (person: Person | null) => void] {
15    const [person, setPerson] = useState<Person | null>(null);
16    const loaded = useRef(false);
17
18    useLayoutEffect(() => {
19      loaded.current = true;
20
21      return () => {
22        loaded.current = false;
23      };
24    });
25
26    useEffect(() => {
27      const getPerson = async () => {
28        await sleep(2500);
29        const person = await localforage.getItem<Person>("person");
30
31        if (!loaded.current) {
32          setPerson(person ?? initialPerson);
33        }
34      };
35      getPerson();
36    }, [initialPerson]);
37  }
```



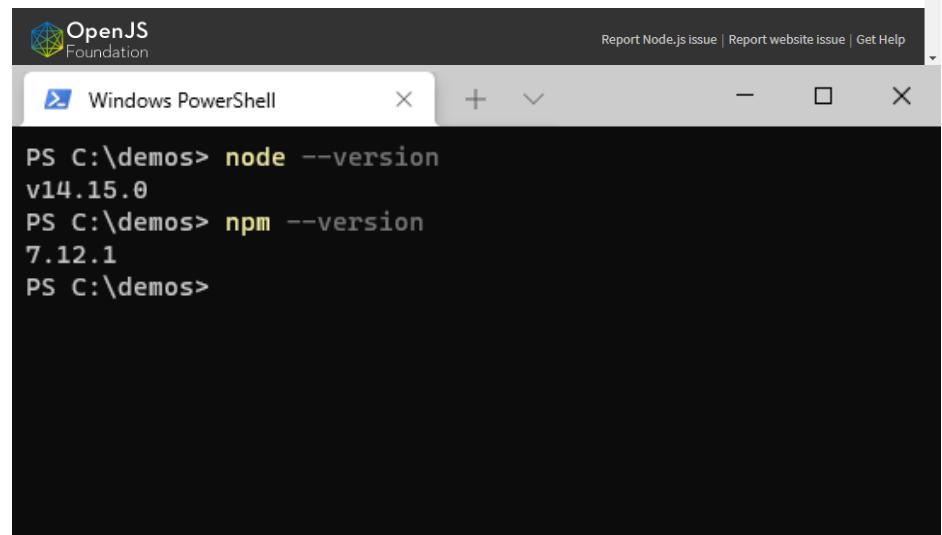
# Install Node.js & NPM



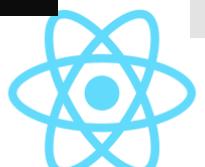
- Minimal:
  - Node version 12
  - NPM version 6



A screenshot of the official Node.js website ([nodejs.org/en/](https://nodejs.org/en/)). The page features the Node.js logo and navigation links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, and NEWS. A prominent call-to-action button says "Download for Windows (x64)". Below it, two other download options are shown: "14.17.0 LTS Recommended For Most Users" and "16.1.0 Current Latest Features". At the bottom, links for "Other Downloads", "Changelog", "API Docs", and "Long Term Support (LTS) schedule" are provided.



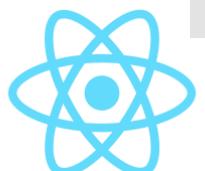
A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command "PS C:\demos> node --version" is run, resulting in the output "v14.15.0". The command "PS C:\demos> npm --version" is also run, resulting in the output "7.12.1". The OpenJS Foundation logo is visible in the top left corner of the window.



# Clone the GitHub Repository



```
PS C:\Temp> git clone git@github.com:mauricedb/react-hooks-tips-only-the-pros-know-course.git
Cloning into 'react-hooks-tips-only-the-pros-know-course'...
remote: Enumerating objects: 112, done.
remote: Counting objects: 100% (112/112), done.
remote: Compressing objects: 100% (72/72), done.
Receiving objects: 55% (62/112) used 107 (delta 32), pack-reused 0 receiving objects: 38% (43/112)
Receiving objects: 100% (112/112), 91.52 KiB | 520.00 KiB/s, done.
Resolving deltas: 100% (34/34), done.
PS C:\Temp> |
```



# Install NPM Packages



```
Windows PowerShell

PS C:\Temp\react-hooks-tips-only-the-pros-know-course> npm install
npm WARN deprecated @types/classnames@2.3.1: This is a stub types definition. classnames provides its
own type definitions, so you do not need this installed.

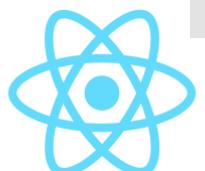
> react-hooks-tips-only-the-pros-know-course@0.0.0 prepare
> husky install

husky - Git hooks installed

added 370 packages, and audited 371 packages in 6s

78 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
PS C:\Temp\react-hooks-tips-only-the-pros-know-course> |
```



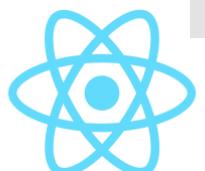
# Start the application



```
PS C:\Temp\react-hooks-tips-only-the-pros-know-course> npm start
> react-hooks-tips-only-the-pros-know-course@0.0.0 start
> vite

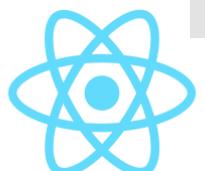
vite v2.3.3 dev server running at:
> Local: http://localhost:3000/
> Network: use '--host' to expose

ready in 421ms.
```



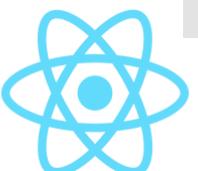
Type it out  
by hand?

*"Typing it drills it into your brain much better than simply copying and pasting it. You're forming new neuron pathways. Those pathways are going to help you in the future. Help them out now!"*



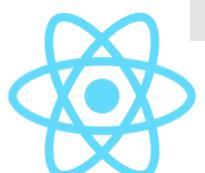
# TypeScript

- All slides and exercises use **TypeScript**
  - It's highly recommended in any serious project
- Not used to TypeScript?
  - Just give it a try 😊
  - Or set *strict* to *false* in the *tsconfig.json*



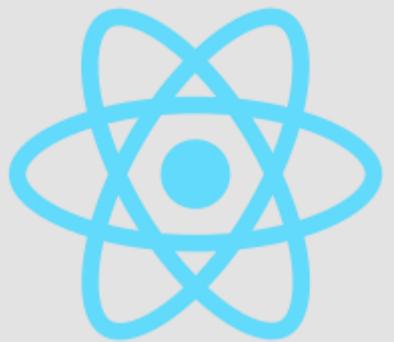


See you in the next video



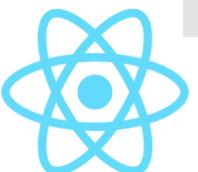
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



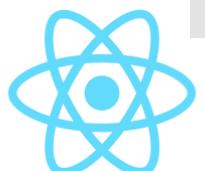
# React hooks

## The basics



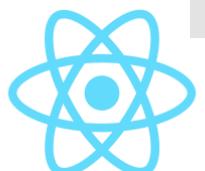
# The basics of React hooks

- Three basic React hooks
  - useState()
  - useEffect()
  - useContext()
- Hooks can only be **used in functional components**



# useState()

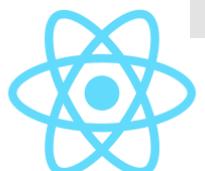
Returns a **stateful value**, and a function to update it



# The useState() Ho

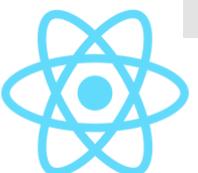
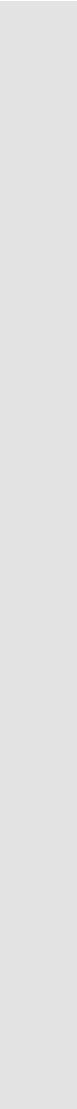


```
TS PersonEditor.tsx M X
src > person-editor > TS PersonEditor.tsx > ...
1 import React, { ReactElement, useState } from "react"
2
3 import { LabeledInput } from "../components"
4 import { initialPerson } from "../utils"
5
6 export function PersonEditor(): ReactElement {
7   const [person, setPerson] = useState(initialPerson)
8
9   return (
10    <form>
11      <h2>Person Editor</h2>
12      <LabeledInput
13        label="Firstname:"
14        value={person.firstname}
15        onChange={(e) => {
16          const newPerson = {
17            ... person,
18            firstname: e.target.value,
19          }
20          setPerson(newPerson)
21        }}
22      />
```



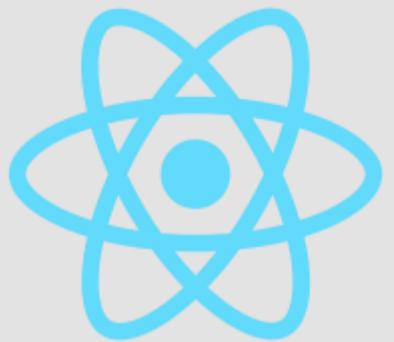


See you in the next video



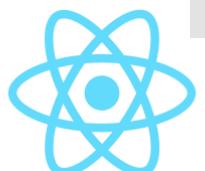
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



# useState()

With callbacks

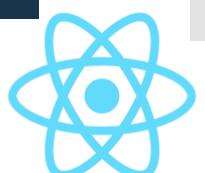


# State hook with functions



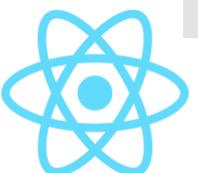
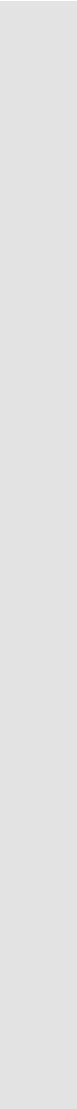
```
TS PersonEditor.tsx M X
src > person-editor > TS PersonEditor.tsx > ...
You, seconds ago | 1 author (You)
1 import React, { ReactElement, useState } from "react" You, 13 minutes
2 *
3 import { LabeledInput } from "../components"
4 import { initialPerson } from "../utils"
5
6 export function PersonEditor(): ReactElement {
7   const [person, setPerson] = useState(() => initialPerson)
8
9   return (
10     <form
11       className="person-editor"
12       onSubmit={(e) => {
13         e.preventDefault()
14         alert(`Submitting\n${JSON.stringify(person, null, 2)}`)
15       }}
16     >
17       <h2>Person Editor</h2>
18       <LabeledInput
19         label="Firstname:"
20         value={person.firstname}
21         onChange={(e) =>
22           setPerson((state) => ({ ...state, firstname: e.target.value }))
23         }
24       />

```



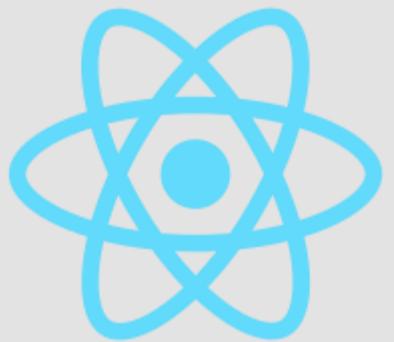


See you in the next video



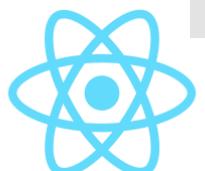
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



# useEffect()

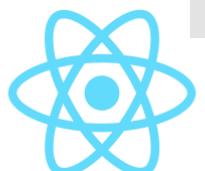
Accepts a function that contains imperative, possibly **effectful** code



# Side effect



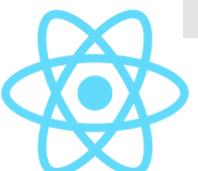
```
TS PersonEditor.tsx M X
src > person-editor > TS PersonEditor.tsx > PersonEditor > <function> > setPerson() callback > <unknown>
10 | function savePerson(person: Person | null): void {
11 |   console.log("Saving", person)
12 |   localforage.setItem("person", person)
13 |
14 |
15 | export function PersonEditor(): ReactElement {
16 |   const [person, setPerson] = useState<Person | null>(null)
17 |
18 |   useEffect(() => {
19 |     const getPerson = async () => {
20 |       const person = await localforage.getItem<Person>("person")
21 |       setPerson(person ?? initialPerson)
22 |     }
23 |
24 |     getPerson()
25 |   }, [])
26 |
27 |   useEffect(() => {
28 |     savePerson(person)
29 |   }, [person])
30 |
31 |   if (!person) {
32 |     return <Loading />
33 |   }
}
```



# Side effect with cleanup

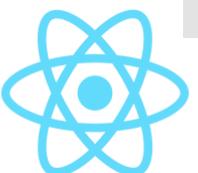
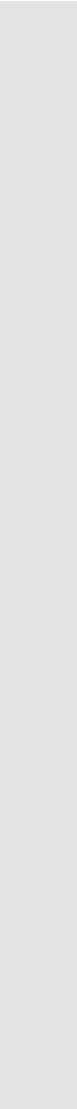
```
useEffect(() => {
  const handle = setInterval(() => {
    // Do something every second
  }, 1000)

  return () => clearInterval(handle)
}, [])
```



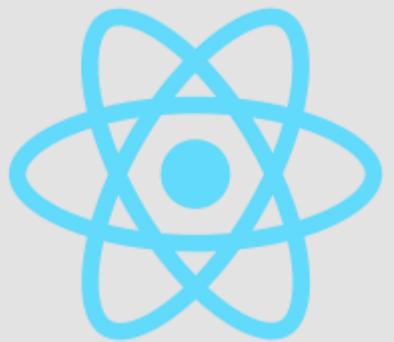


See you in the next video



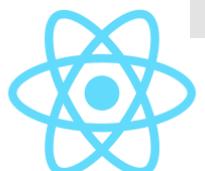
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



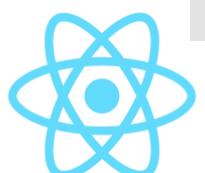
# useContext()

Accepts a **context object** and returns it's current value

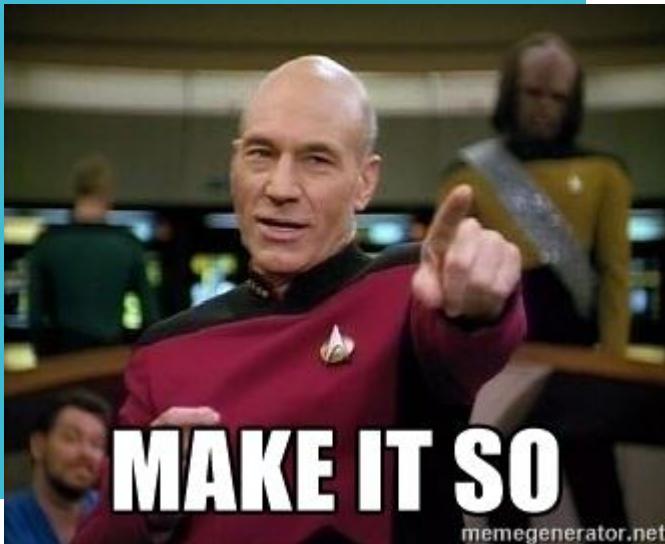


# Context Provider

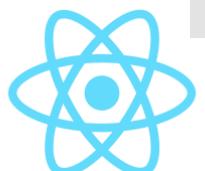
```
18 export const themeContext = createContext<ThemeContext>({
19   style: undefined,
20   setStyle: () => void null,
21 })
22
23 export function ThemeProvider({ children }: Props): ReactElement {
24   const [style, setStyle] = useState<CSSProperties>()
25
26   return (
27     <themeContext.Provider value={{ style, setStyle }}>
28       {children}
29     </themeContext.Provider>
30   )
31 }
```



# useContext()

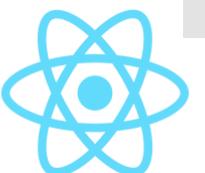
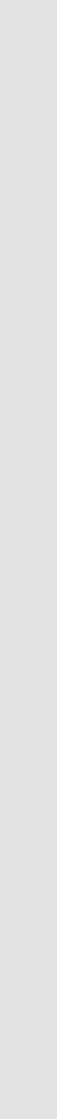


```
9  export function App(): ReactElement {  
10 |   const { style } = useContext(themeContext)  
11 |  
12 |   return (  
13 |     <div className="container" style={style}>  
14 |       <BrowserRouter>  
15 |         <AppNavbar />  
16 |         <Routes />  
17 |       </BrowserRouter>  
18 |     </div>  
19 |   )  
20 }
```



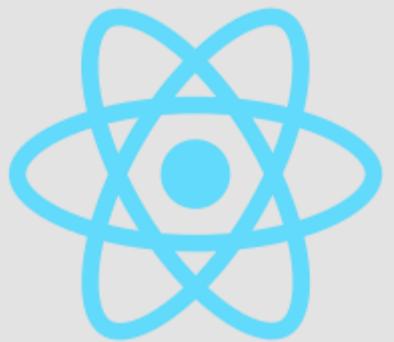


See you in the next video

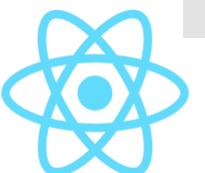


# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb

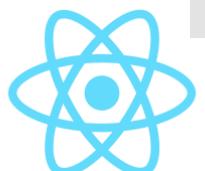


# Custom hooks



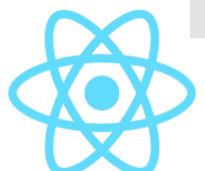
# Custom hooks

- Make component code **reusable**
- Great for **extracting code** from components
  - Even when reuse is not a goal
- Custom hooks can **use other React hooks** as needed



# Creating the usePerson() hook

```
TS usePerson.ts U X  TS PersonEditor.tsx M
src > person-editor > TS usePerson.ts > ...
1 import { useState, useEffect } from "react"
2 import localforage from "localforage"
3
4 import type { Person } from "../types/person"
5
6 function savePerson(person: Person | null): void {
7   console.log("Saving", person)
8   localforage.setItem("person", person)
9 }
10
11 export function usePerson(initialPerson: Person) {
12   const [person, setPerson] = useState<Person | null>(null)
13
14   useEffect(() => {
15     const getPerson = async () => {
16       const person = await localforage.getItem<Person>("person")
17       setPerson(person ?? initialPerson)
18     }
19
20     getPerson()
21   }, [initialPerson])
22
23   useEffect(() => {
24     savePerson(person)
25   }, [person])
26
27   return [person, setPerson] as const
28 }
```



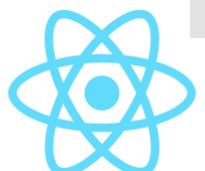
# Using the usePerson() hook



MAKE IT SO

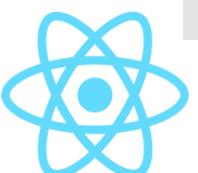
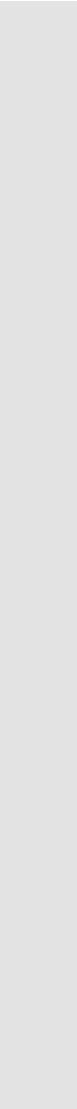
memegenerator.net

```
TS PersonEditor.tsx M X TS usePerson.ts U
src > person-editor > TS PersonEditor.tsx > ...
You, seconds ago | 1 author (You)
1  /* eslint-disable @typescript-eslint/no-non-null-assertion */
2  import React, { ReactElement } from "react"
3
4  import { LabeledInput, Loading } from "../components"
5  import { initialPerson } from "../utils"
6  import { usePerson } from "./usePerson"
7
8  export function PersonEditor(): ReactElement {
9    const [person, setPerson] = usePerson(initialPerson)
10
11   if (!person) {
12     return <Loading />
13   }
```



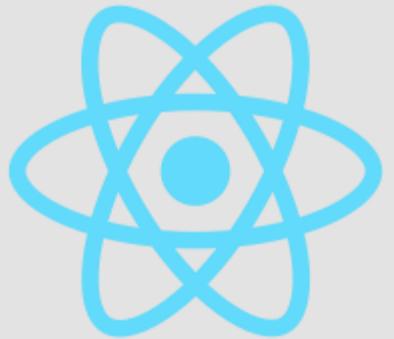


See you in the next video



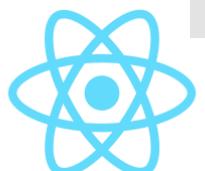
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



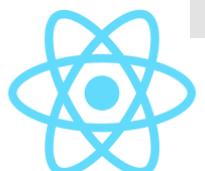
# Rules of Hooks

Custom hook names



# Custom Hook Names

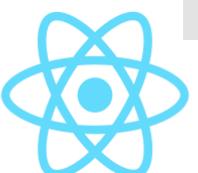
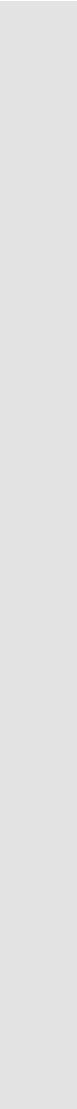
👉 Custom hooks must be functions named `use....` ✌





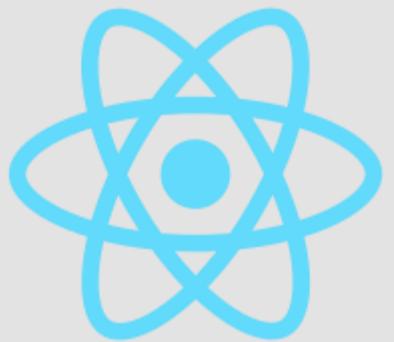


See you in the next video



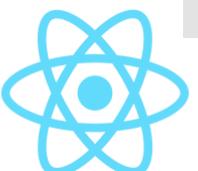
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



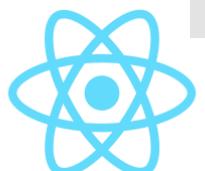
# useRef()

useRef() returns a mutable ref object whose **.current** property is initialized to the passed argument



# useRef()

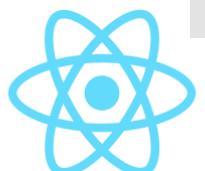
- One of its main purposes is to get a **reference to a DOM element**
- But can be used to keep a **reference to any state**
  - Holds the same state with each render
  - Updating the state doesn't trigger a render
- useRef() is stable and **doesn't need to be added as a dependency**
  - Great way to share values between useEffect() functions



# useRef() with a HTML element

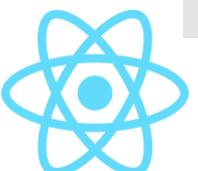


```
TS Counter.tsx M X
src > rules-of-hooks > TS Counter.tsx > ...
You, a minute ago | 1 author (You)
1 import React, { ReactElement, useEffect, useRef, useState } from "react"
2
3 export function Counter(): ReactElement {
4   const [counter, setCounter] = useState(0)
5   const button = useRef<HTMLButtonElement>(null)
6
7   useEffect(() => {
8     setTimeout(() => {
9       button.current?.focus()
10      }, 1000)
11    }, [])
12
13   return (
14     <div>
15       <div>Count: {counter}</div>
16       <div>
17         <button
18           ref={button}
19           className="btn btn-primary"
20           onClick={() => setCounter(counter + 1)}
21         >
22           Increment
23         </button>
24       </div>
25     </div>
26   )
27 }
```

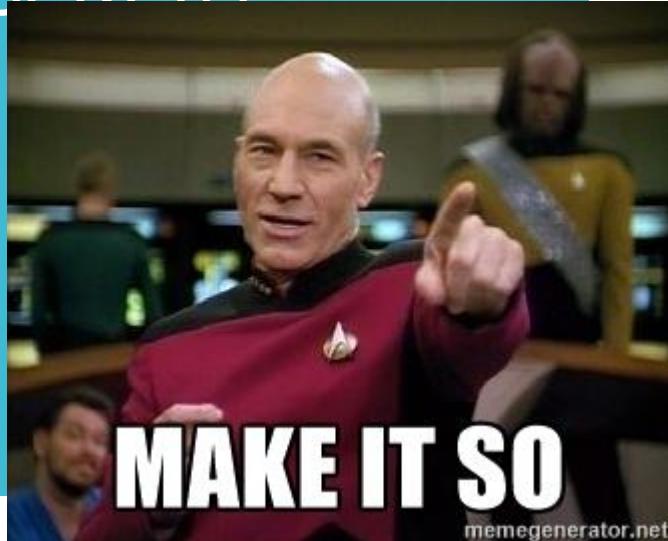


# useRef() with a component

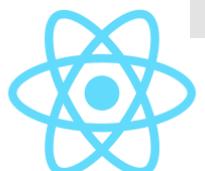
```
TS PersonEditor.tsx 2, M X
src > person-editor > TS PersonEditor.tsx > ...
1 | import React, { ReactElement, useEffect, useRef } from "react"
2 | ?
3 | import { LabeledInput, Loading } from "../components"
4 | import { initialPerson } from "../utils"
5 | import { usePerson } from "./usePerson"
6 |
7 | export function PersonEditor(): ReactElement {
8 |   const [person, setPerson] = usePerson(initialPerson)
9 |   const input = useRef<HTMLInputElement>(null)
10 |
11 |   useEffect(() => {
12 |     setTimeout(() => {
13 |       input.current?.focus()
14 |     }, 1000)
15 |   }, [])
16 |
17 |   if (!person) {
18 |     return <Loading />
19 |   }
20 |
21 |   return (
22 |     <form>
23 |       <h2>Person Editor</h2>
24 |       <LabeledInput
25 |         ref={input}
26 |         label="Firstname:"
27 |         value={person.firstname}
28 |         onChange={(e) => {
29 |           setPerson((person) => ({
```



# Forwarding useRef()

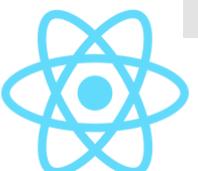


```
TS LabeledInput.tsx M X
src > components > TS LabeledInput.tsx > ...
You, seconds ago | 1 author (You)
1 import React, { forwardRef, InputHTMLAttributes, ReactElement } from "react"
2 import classNames from "classnames"
3
You, a week ago | 1 author (You)
4 interface Props extends InputHTMLAttributes<HTMLInputElement> {
5   label: string
6 }
7
8 export const LabeledInput = forwardRef<HTMLInputElement, Props>(
9   ({ id, label, className, ...props }, ref): ReactElement => {
10     return (
11       <div className={classNames("form-group", className)}>
12         <label htmlFor={id} className="form-label">
13           {label}
14         </label>
15
16         <input {...props} id={id} className="form-control" ref={ref} />
17       </div>
18     )
19   }
20 )
21
22 LabeledInput.displayName = "LabeledInput"
```



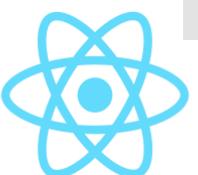
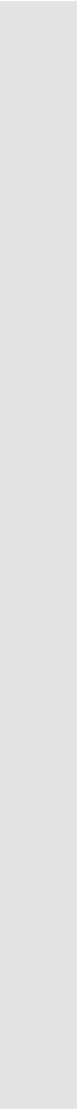
# useRef() for generic state

- With useRef() you can maintain **any state** for a component
  - When you don't want a render on updates



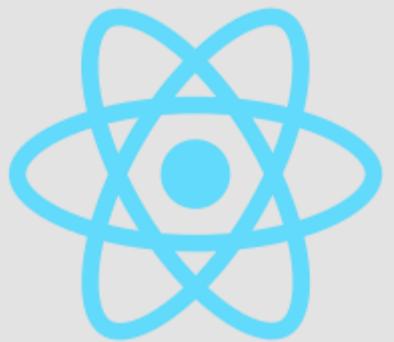


See you in the next video



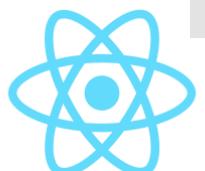
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



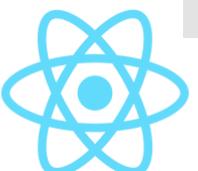
# useLayoutEffect()

Like useEffect() but **fires synchronously** after all DOM mutations

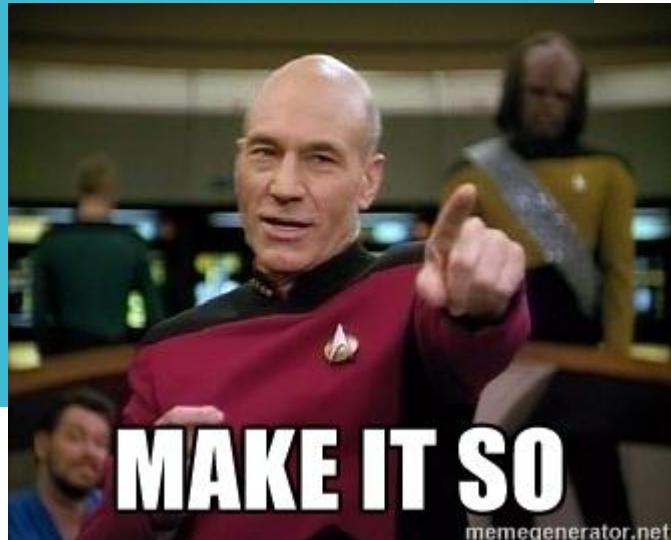


# useEffect() versus useLayoutEffect()

- **useLayoutEffect()** executes **synchronously**
  - After the render but before the component is painted
- While **useEffect()** executes **asynchronously**
  - After the component is painted on screen
- **Normally useEffect() is what you need**
  - Use useLayoutEffect() when you want the synchronous behavior



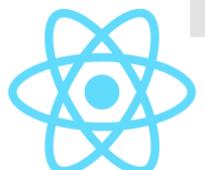
# Synchronous DOM Mutations



MAKE IT SO

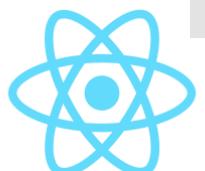
memegenerator.net

```
TS Counter.tsx M X
src > rules-of-hooks > TS Counter.tsx > ...
You, seconds ago | 1 author (You)
1 import React, { You, seconds ago • Uncommitted changes
2   ReactElement,
3   useEffect,
4   useLayoutEffect,
5   useRef,
6   useState,
7 } from "react"
8
9 export function Counter(): ReactElement {
10   const [counter, setCounter] = useState(0)
11   const button = useRef<HTMLButtonElement>(null)
12
13   useEffect(() => {
14     setTimeout(() => {
15       button.current?.focus()
16     }, 1000)
17   }, [])
18
19   useLayoutEffect(() => {
20     if (button.current) {
21       button.current.style.backgroundColor = "green"
22     }
23   }, [])
```



# useIsMounted()

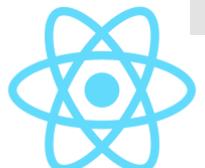
```
TS useIsMounted.ts U X
src > hooks > TS useIsMounted.ts > ...
1 import { MutableRefObject, useRef, useLayoutEffect } from "react"
2 ?
3 export function useIsMounted(): Readonly<MutableRefObject<boolean>> {
4   const isMounted = useRef(false)
5
6   useLayoutEffect(() => {
7     isMounted.current = true
8
9     return () => {
10       isMounted.current = false
11     }
12   }, [])
13
14   return isMounted
15 }
```



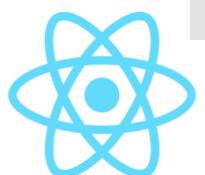
# Using useRef() to maintain state



```
TS usePerson.ts M X
src > person-editor > TS usePerson.ts > ...
1 import { useState, useEffect } from "react"      You, 3 days ago
2 import localforage from "localforage"
3
4 import type { Person } from "../types/person"
5 import { sleep } from "../utils"
6 import { useIsMounted } from "../hooks/useIsMounted"
7
8 function savePerson(person: Person | null): void {
9   console.log("Saving", person)
10  localforage.setItem("person", person)
11 }
12
13 export function usePerson(initialPerson: Person) {
14   const [person, setPerson] = useState<Person | null>(null)
15   const isMounted = useIsMounted()
16
17   useEffect(() => {
18     const getPerson = async () => {
19       const person = await localforage.getItem<Person>("person")
20       await sleep(2500)
21       if (isMounted.current) {
22         setPerson(person ?? initialPerson)
23       }
24     }
25
26     getPerson()
27   }, [initialPerson, isMounted])
```

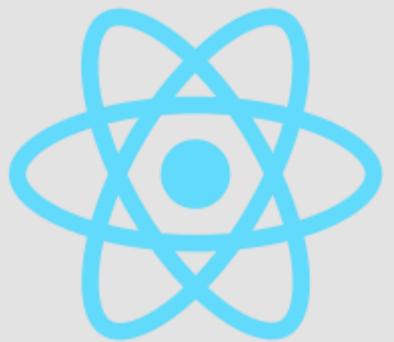


See you in the next video



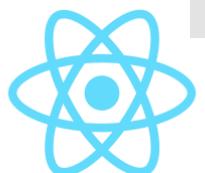
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



# useRef(), useState() and dependencies

👉 The useRef() object and the useState() updater function are not required in dependency arrays ✌



# No dependency

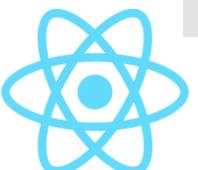
```
TS PersonEditor.tsx M X
src > person-editor > TS PersonEditor.tsx > PersonEditor > <function> > setPerson() callback > <unknown>
10 | function savePerson(person: Person | null): void {
11 |   console.log("Saving", person)
12 |   localforage.setItem("person", person)
13 |
14 |
15 | export function PersonEditor(): ReactElement {
16 |   const [person, setPerson] = useState<Person | null>(null)
17 |
18 |   useEffect(() => {
19 |     const getPerson = async () => {
20 |       const person = await localforage.getItem<Person>("person")
21 |       setPerson(person ?? initialPerson)
22 |     }
23 |
24 |     getPerson()
25 |
26 |
27 |     useEffect(() => {
28 |       savePerson(person)
29 |     }, [person])
30 |
31 |     if (!person) {
32 |       return <Loading />
33 |     }
34 |   }, [])
35 | }
```

Empty

getPerson()

Not empty

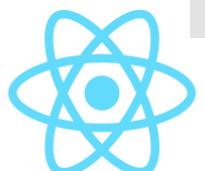
[person]



# No dependency

```
TS useIsMounted.ts U X
src > hooks > TS useIsMounted.ts > ...
1 import { MutableRefObject, useRef, useLayoutEffect } from "react"
2 ?
3 export function useIsMounted(): Readonly<MutableRefObject<boolean>> {
4   const isMounted = useRef(false)
5
6   useLayoutEffect(() => {
7     isMounted.current = true
8
9     return () => {
10       isMounted.current = false
11     }
12   })
13
14   return isMounted
15 }
```

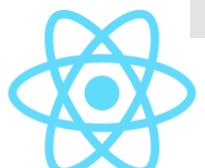
Empty → []



# With Dependency

```
TS usePerson.ts M X
src > person-editor > TS usePerson.ts > ...
1 import { useState, useEffect } from "react"      You, 3 days ago
2 import localforage from "localforage"
3
4 import type { Person } from "../types/person"
5 import { sleep } from "../utils"
6 import { useIsMounted } from "../hooks/useIsMounted"
7
8 function savePerson(person: Person | null): void {
9   console.log("Saving", person)
10  localforage.setItem("person", person)
11 }
12
13 export function usePerson(initialPerson: Person) {
14   const [person, setPerson] = useState<Person | null>(null)
15   const isMounted = useIsMounted()
16
17   useEffect(() => {
18     const getPerson = async () => {
19       const person = await localforage.getItem<Person>("person")
20       await sleep(2500)
21       if (isMounted.current) {
22         setPerson(person ?? initialPerson)
23       }
24     }
25
26   getPerson()
  [initialPerson, isMounted]
}
```

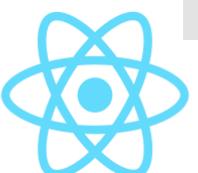
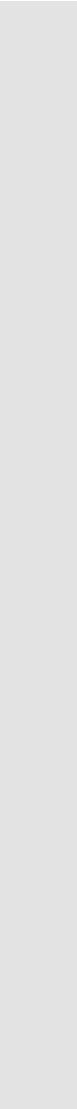
Not empty





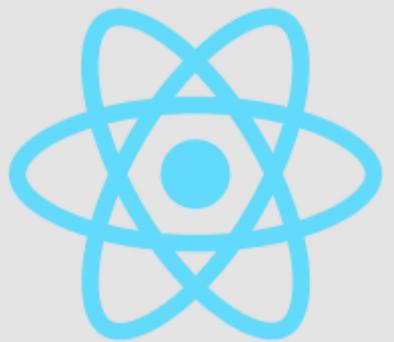


See you in the next video

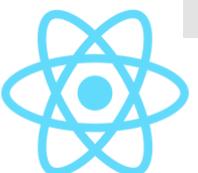


# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb

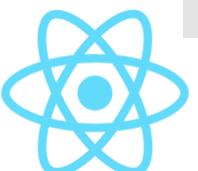


# Debounce and useEffect()



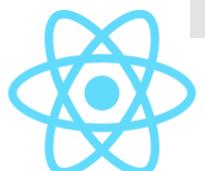
# Debounce

- Creating a **debounce hook with useEffect()** is easy
  - Execute the code via a setTimeout()
  - Cancel the setTimeout() in the cleanup



# useDebounce()

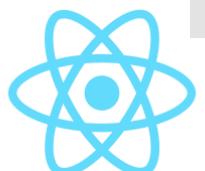
```
TS useDebounce.ts U X
src > hooks > TS useDebounce.ts > ...
1 import { useEffect } from "react"
2
3 export function useDebounce(fn: () => void, timeout: number): void {
4   useEffect(() => {
5     const handle = setTimeout(fn, timeout)
6
7     return () => clearTimeout(handle)
8   }, [fn, timeout])
9 }
```



# Using useDebounce()

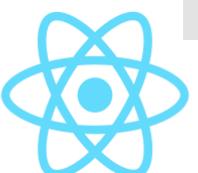
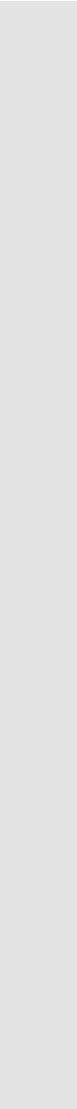


```
TS usePerson.ts 1, M X
src > person-editor > TS usePerson.ts > ...
18  useEffect(() => {
19    const getPerson = async () => {
20      const person = await localforage.getItem<Person>("person")
21      // await sleep(2500)
22      if (isMounted.current) {
23        setPerson(person ?? initialPerson)
24      }
25    }
26
27    getPerson()
28  }, [initialPerson, isMounted])
29
30  useDebounce(() => {
31    savePerson(person)
32  }, 1000)
33
34  return [person, setPerson] as const
35 }
```



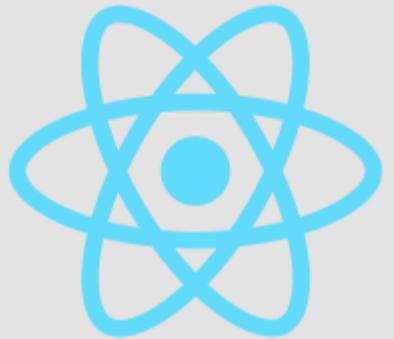


See you in the next video



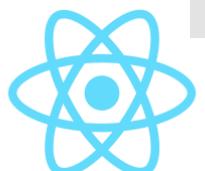
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



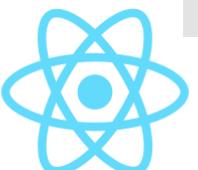
# useCallback()

Returns a memoized callback

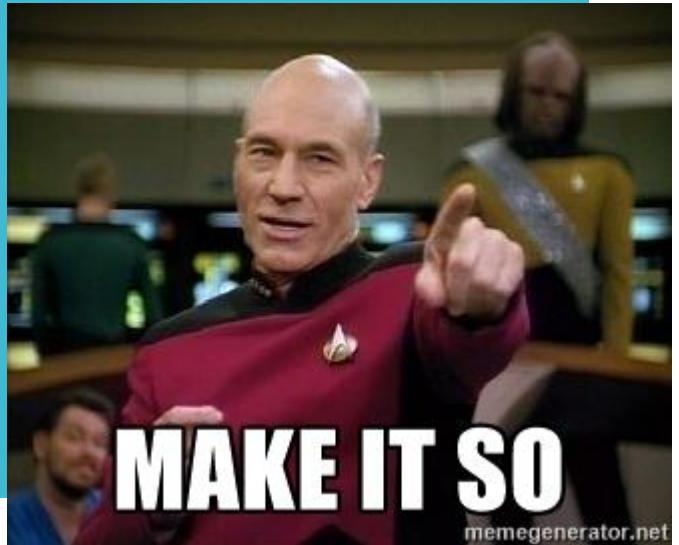


# useCallback()

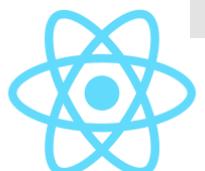
- **Pass a function and dependencies** to useCallback()
  - Returns the same function reference with the same dependencies
- Useful when **passing callbacks to child components**
  - Or other hooks
- **Note:** useMemo() can be used for values



# useCallback()

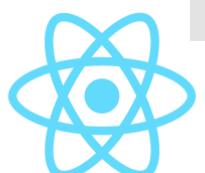


```
36 |   const saveFn = useCallback(() => {  
37 |     savePerson(person)  
38 |   }, [person])  
39 |  
40 |   useDebounce(saveFn, 1000)
```



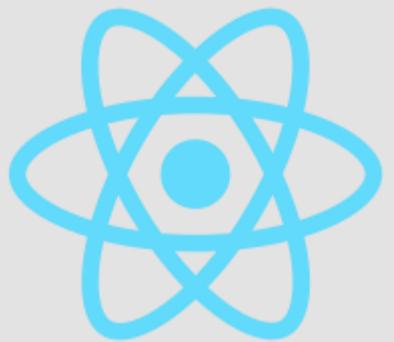


See you in the next video

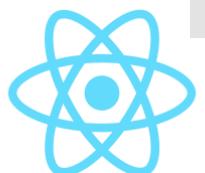


# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb

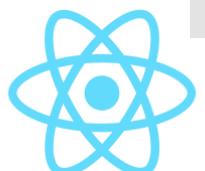


# Unmount and useEffect()



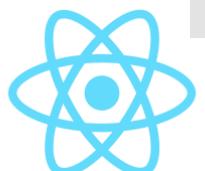
# Unmount

- Creating an **unmount hook with useEffect()** is easy
  - Execute the passed code in the effect cleanup
  - Use a ref so there are no dependencies



# useWillUnmount()

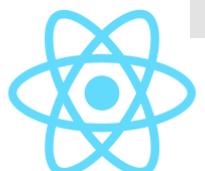
```
TS useWillUnmount.ts U X
src > hooks > TS useWillUnmount.ts > ...
1 import { useEffect, useRef } from "react"
2
3 export function useWillUnmount(fn: () => void): void {
4   const functionRef = useRef(fn)
5   functionRef.current = fn
6
7   useEffect(() => {
8     return () => functionRef.current()
9   }, [])
10 }
```



Save changes  
when unmounting

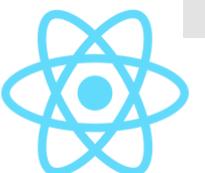
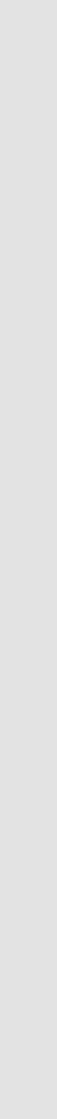


```
TS usePerson.ts 1, M X
src > person-editor > TS usePerson.ts > ...
36 |
37 |   const saveFn = useCallback(() => {
38 |     savePerson(person)
39 |   }, [person])
40 |
41 |   useDebounce(saveFn, 10000)
42 |   useWillUnmount(saveFn)
43 |
44 |   return [person, setPerson] as const
45 }
```



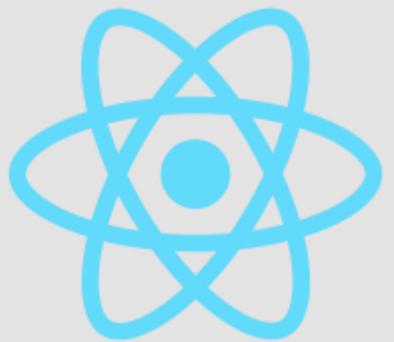


See you in the next video



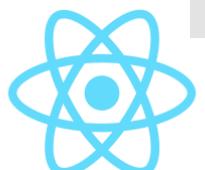
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



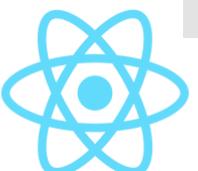


# useThrottle()



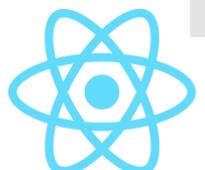
# Throttle versus Debounce

- A debounced function is only called after it has not been called for a specified duration
- A throttled function is called once after every timeout

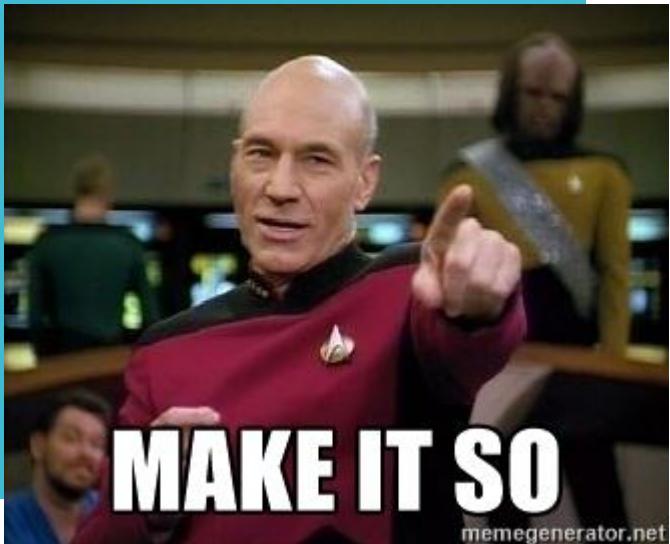


# useThrottle()

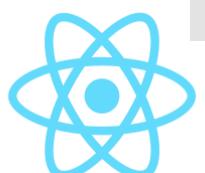
```
TS useThrottle.ts ✘ | src > hooks > TS useThrottle.ts > ...
1 import { useEffect, useRef } from "react"
2
3 export function useThrottle(fn: () => void, timeout: number): void {
4   const previousRef = useRef<(() => void) | null>(null)
5   const currentRef = useRef<(() => void) | null>(fn)
6   if (previousRef.current !== fn) {
7     currentRef.current = fn
8   }
9
10 useEffect(() => {
11   const handle = setInterval(() => {
12     if (currentRef.current) {
13       currentRef.current()
14       previousRef.current = currentRef.current
15       currentRef.current = null
16     }
17   }, timeout)
18
19   return () => clearInterval(handle)
20 }, [timeout])
21 }
```



# usePerson()

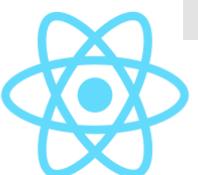
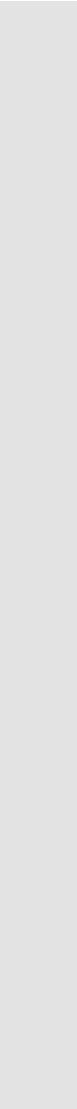


```
TS usePerson.ts 2, M X
src > person-editor > TS usePerson.ts > ...
38  const saveFn = useCallback(() => {
39    |   savePerson(person)
40  }, [person])
41
42  | useThrottle(saveFn, 1000)
43  useWillUnmount(saveFn)
44
45  return [person, setPerson] as const
46 }
```



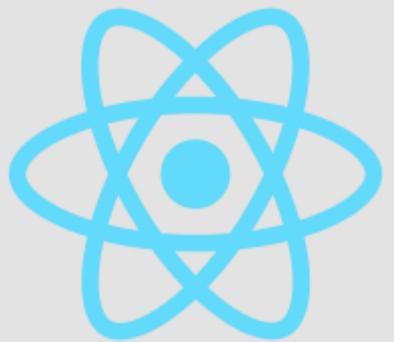


See you in the next video



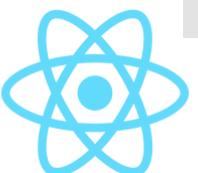
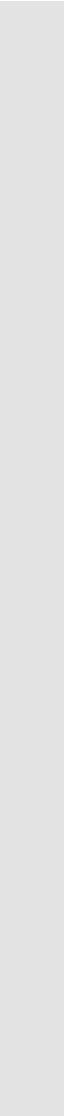
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



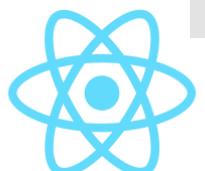


# useDebugValue()

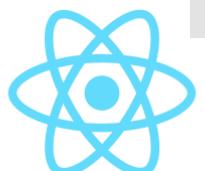
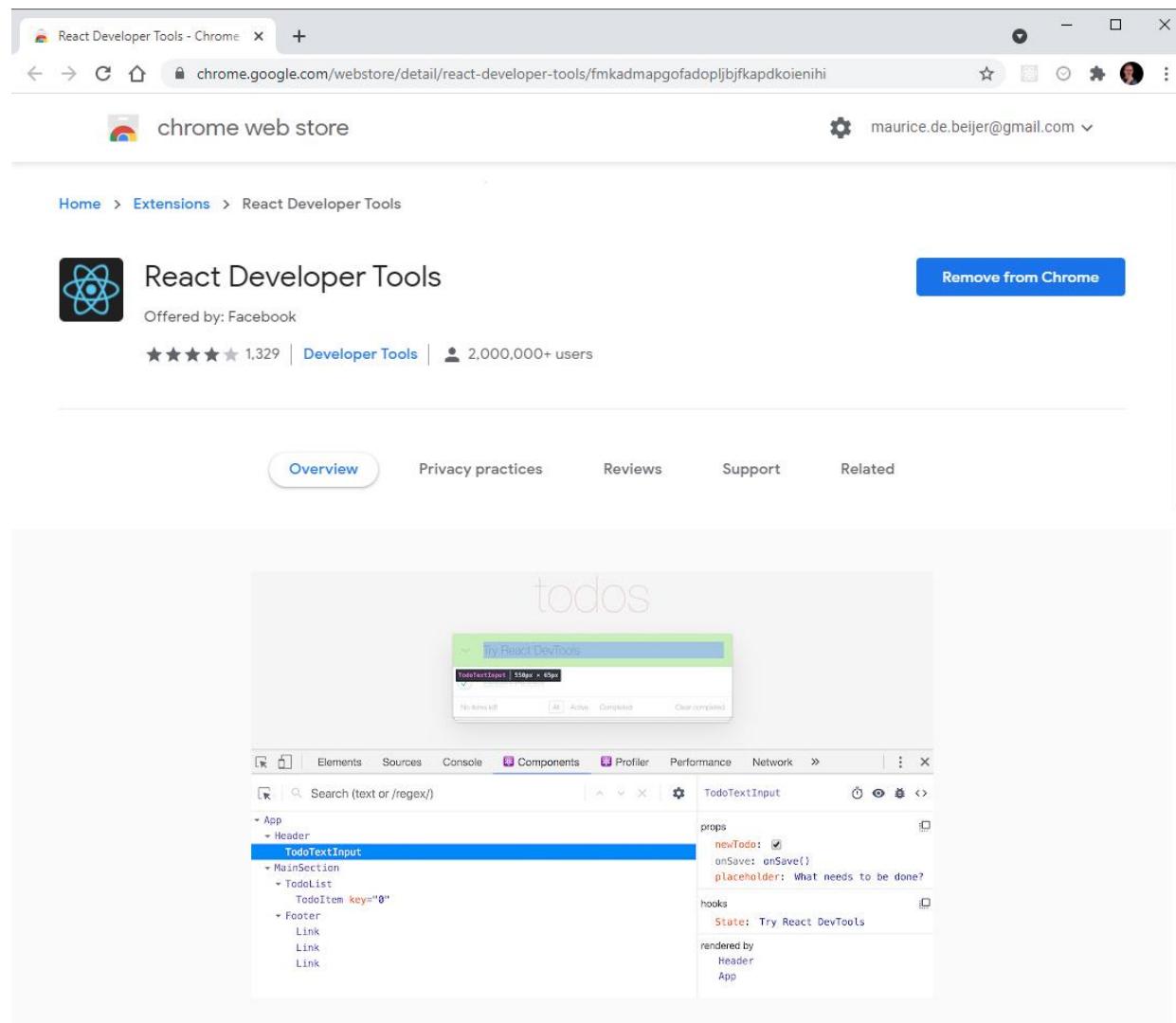


# useDebugValue()

- Used to display a label for custom hooks in React DevTools
- Only recommended for reusable hooks



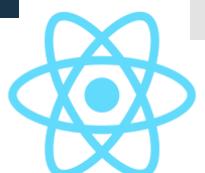
# React Developer Tools



# useDebugValue()

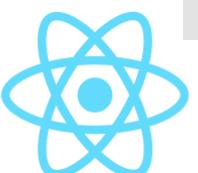
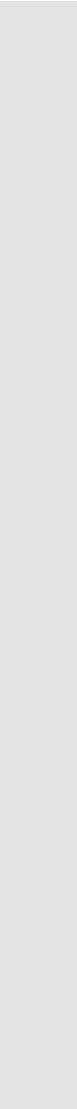


```
TS useThrottle.ts M X
src > hooks > TS useThrottle.ts > ...
You, 2 minutes ago | 1 author (You)
1 import { useEffect, useRef, useDebugValue } from "react"
2
3 export function useThrottle(fn: () => void, timeout: number): void {
4   const previousRef = useRef<(() => void) | null>(null)
5   const currentRef = useRef<(() => void) | null>(fn)
6   if (previousRef.current !== fn) {
7     currentRef.current = fn
8   }
9
10 useDebugValue(currentRef.current, (fn) => fn?.toString())
11
12 useEffect(() => {
13   const handle = setInterval(() => {
14     if (currentRef.current) {
15       previousRef.current = currentRef.current
16       currentRef.current()
17       currentRef.current = null
18     }
19   }, timeout)
20
21   return () => clearInterval(handle)
22 }, [timeout])
23 }
```



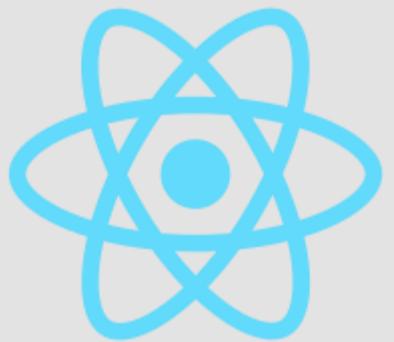


See you in the next video

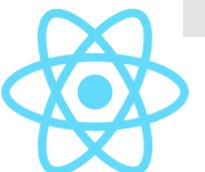


# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb

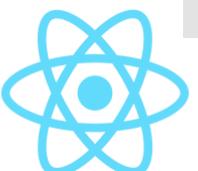


# Rules of Hooks



# Rules of hooks

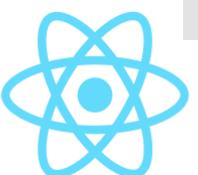
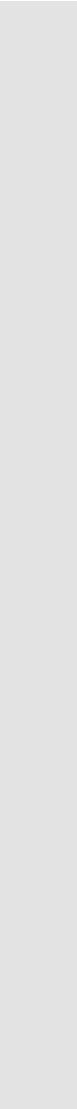
- **Only Call Hooks at the Top Level**
  - Don't call Hooks inside loops, conditions, or nested functions
- **Only Call Hooks from React functions**
  - Call Hooks from React function components
  - Call Hooks from custom Hooks





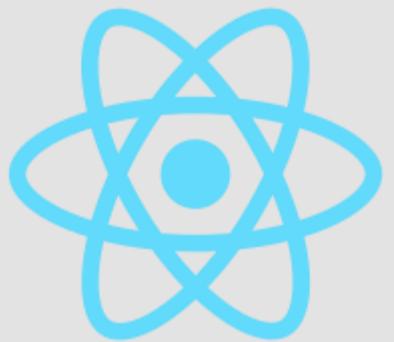


See you in the next video



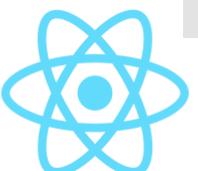
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



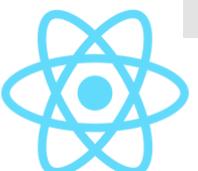
# More complex state

With useState()



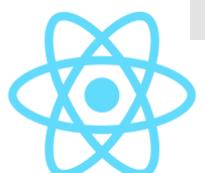
# More complex state

- Track multiple state items
  - The form data as well as meta data about the form
- Use as many `useState()` hooks as needed
  - One for the person object
  - A second for the dirty and valid states



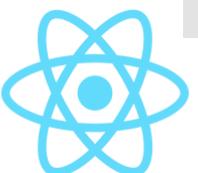
# Two useState() hooks

```
TS usePerson.ts 2, M X
src > person-editor > TS usePerson.ts > ...
22 | interface Metadata {
23 |   isDirty: boolean
24 |   isValid: boolean
25 |
26 |   You, seconds ago • Uncommitted changes
27 | export function usePerson(initialPerson: Person) {
28 |   const [person, setPerson] = useState<Person | null>(null)
29 |   const [metadata, setMetadata] = useState<Metadata>({
30 |     isDirty: false,
31 |     isValid: true,
32 |   })
33 |   const isMounted = useIsMounted()
```



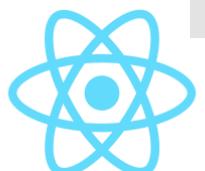
# Multiple updates

```
TS usePerson.ts 2, M ×
src > person-editor > TS usePerson.ts > ...
59  useThrottle(saveFn, 1000)
60  useWillUnmount(saveFn)
61
62  const setPersonAndMeta = (value: SetStateAction<Person | null>) => {
63    setPerson(value)
64    setMetadata((m) => ({ ...m, isDirty: true }))
65  }
66
67  return [person, setPersonAndMeta, metadata] as const
68 }
```



# Exposing more data from a custom hook

```
TS PersonEditor.tsx M X
src > person-editor > TS PersonEditor.tsx > PersonEditor
  8  export function PersonEditor(): ReactElement {
  9    const [person, setPerson, { isDirty, isValid }] = usePerson(initialPerson)
10    const input = useRef<HTMLInputElement>(null)
11
12    useEffect(() => {
13      setTimeout(() => {
14        input.current?.focus()
15      }, 1000)
16    }, [])
17
18    if (!person) {
19      return <Loading />
20  }
```



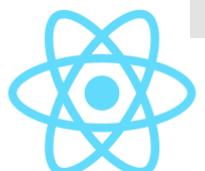
Disable save  
button if there  
are no changes



**MAKE IT SO**

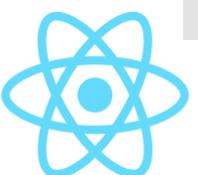
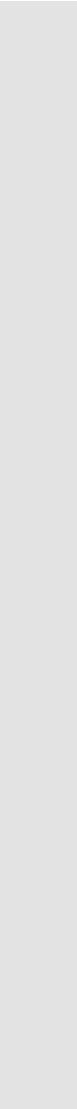
memegenerator.net

```
TS PersonEditor.tsx M X
src > person-editor > TS PersonEditor.tsx > PersonEditor
85   <div className="btn-group">
86     <button
87       type="submit"
88       className="btn btn-primary"
89       disabled={!isDirty || !isValid}
90     >
91       Submit
92     </button>
93   </div>
94   <hr />
95   <pre>{JSON.stringify(person, null, 2)}</pre>
96 </form>
97 )
98 }
```



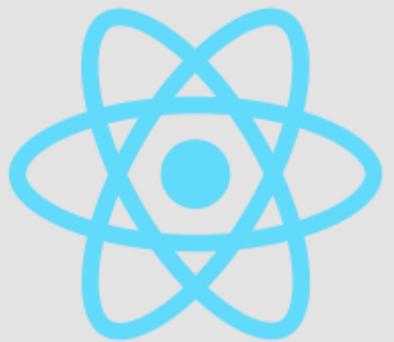


See you in the next video



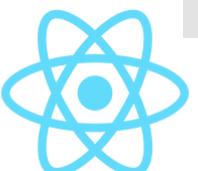
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



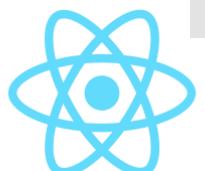
# useReducer()

An alternative to useState()



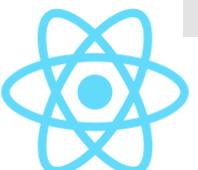
# useReducer()

- **useReducer() is the more powerful brother of useState()**
  - Internally useState() is just a special useReducer()
- **Dispatch action objects** and use a reducer function to create state
  - Very similar to how Redux works
- **The state is still tied to a component**
  - Not global like with Redux



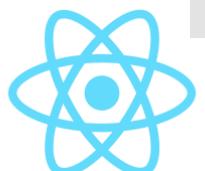
# personEditorReducer()

```
TS personEditorReducer.ts U ×
src > person-editor > TS personEditorReducer.ts > ...
  4 interface Metadata {
  5   isDirty: boolean
  6   isValid: boolean
  7 }
  8
  9 interface ReducerState {
10   person: Person | null
11   metadata: Metadata
12 }
13
14 interface SetPersonAction {
15   type: "set-initial-person"
16   payload: Person
17 }
18
19 type SomeAction = SetPersonAction
20
21 export function personEditorReducer(
22   state: ReducerState,
23   action: SomeAction
24 ): ReducerState {
25   switch (action.type) {
26     case "set-initial-person":
27       return { ...state, person: action.payload }
28     default:
29       return state
30   }
31 }
```



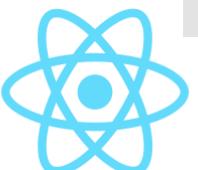
# usePerson()

```
TS usePerson.ts 4, M X
src > person-editor > TS usePerson.ts > usePerson
43 | useEffect(() => {
44 |   const getPerson = async () => {
45 |     const person = await localforage.getItem<Person>("person")
46 |     // await sleep(2500)
47 |     if (isMounted.current) {
48 |       // setPerson(person ?? initialPerson)
49 |       dispatch({
50 |         type: "set-initial-person",
51 |         payload: person ?? initialPerson,
52 |       })
53 |     }
54 |   }
55 |
56 |   getPerson()
57 | }, [initialPerson, isMounted])
```



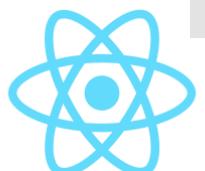
## personEditorReducer()

```
TS personEditorReducer.ts U X
src > person-editor > TS personEditorReducer.ts > ...
24 type SomeAction = SetPersonAction | SetPropertyAction
25
26 export function personEditorReducer(
27   state: ReducerState,
28   action: SomeAction
29 ): ReducerState {
30   switch (action.type) {
31     case "set-initial-person":
32       return { ...state, person: action.payload }
33     case "set-property":
34       return {
35         ...state,
36         metadata: { ...state.metadata, isDirty: true },
37         person: {
38           ...state.person!,
39           [action.payload.name]: action.payload.value,
40         },
41       }
42     default:
43       return state
44   }
45 }
46 }
```



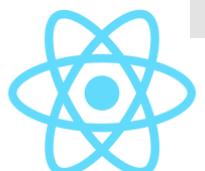
# usePerson()

```
TS usePerson.ts 5, M X
src > person-editor > TS usePerson.ts > ...
78 |   function setProperty(name: keyof Person, value: unknown) {
79 |     dispatch({ type: "set-property", payload: { name, value } })
80 |   }
81 |
82 |   return [person, setProperty, metadata] as const
83 | }
```



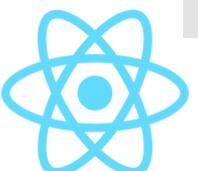
# PersonEditor

```
TS PersonEditor.tsx M X
src > person-editor > TS PersonEditor.tsx > PersonEditor
30   <h2>Person Editor</h2>
31   <LabeledInput
32     ref={input}
33     label="Firstname:"
34     value={person.firstname}
35     onChange={(e) => {
36       // setPerson((person) => ({
37       //   ...person,
38       //   firstname: e.target.value,
39       // }))
40      setProperty("firstname", e.target.value)
}}
```



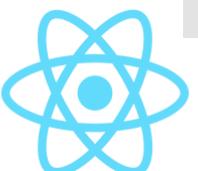
# personEditorReducer()

```
TS personEditorReducer.ts U X
src > person-editor > TS personEditorReducer.ts > ...
31  export function personEditorReducer(
32    state: ReducerState,
33    action: SomeAction
34  ): ReducerState {
35    switch (action.type) {
36      case "set-initial-person":
37        return { ...state, person: action.payload }
38      case "set-property":
39        return {
40          ...state,
41          metadata: { ...state.metadata, isDirty: true },
42          person: {
43            ...state.person!,
44            [action.payload.name]: action.payload.value,
45          },
46        }
47      case "set-properties":
48        return {
49          ...state,
50          metadata: { ...state.metadata, isDirty: true },
51          person: {
52            ...state.person!,
53            ...action.payload,
54          },
55        }
56    }
57  }
```



# usePerson()

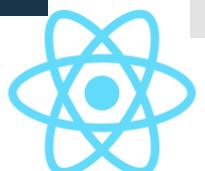
```
TS usePerson.ts 5, M ×
src > person-editor > TS usePerson.ts > ...
78 |   function setProperty(name: keyof Person, value: unknown) {
79 |     dispatch({ type: "set-property", payload: { name, value } })
80 |   }
81 |
82 |   function setProperties(payload: Partial<Person>) {
83 |     dispatch({ type: "set-properties", payload })
84 |   }
85 |
86 |   return [person, setProperty, setProperties, metadata] as const
87 }
```



# PersonEditor

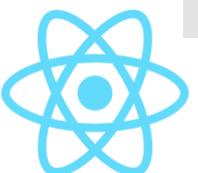
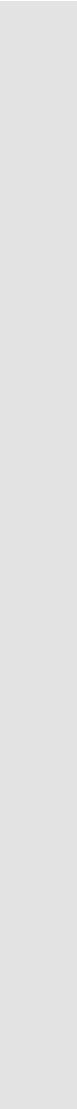


```
TS PersonEditor.tsx M X
src > person-editor > TS PersonEditor.tsx > PersonEditor
32   |     <LabeledInput
33     |       ref={input}
34     |       label="Firstname:"
35     |       value={person.firstname}
36     |       onChange={(e) => {
37       |         setProperty("firstname", e.target.value)
38
39         if (e.target.value === "Ford") {
40           |             setProperties({
41             |               surname: "Prefect",
42             |               address: "Outer space",
43             |               email: "",
44             |               phone: "",
45           |         })
46         }
47       }
48     >
```



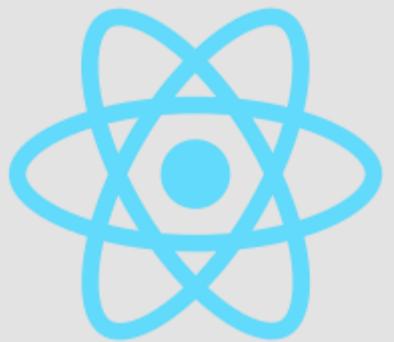


See you in the next video



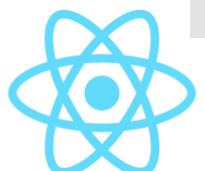
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



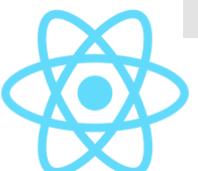
# useMemo()

Returns a memoized value



# useMemo()

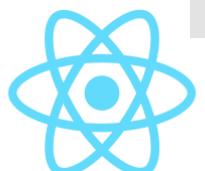
- Uses the creator function to recompute a value
  - Only when the dependencies change
- useMemo() is a performance optimization
  - Not a semantic guarantee
-  **The API allows for memoized values to be forgotten** 
  - In React 17 this doesn't happen



# useMemo()

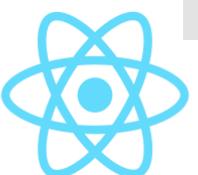
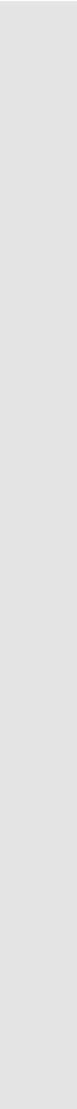


```
TS usePerson.ts 3, M X
src > person-editor > TS usePerson.ts > usePerson > useEffect() callback
38 | const firstAndSurName = useMemo(
39 |   () => {
40 |     firstname: person ?.firstname,
41 |     surname: person ?.surname,
42 |   },
43 |   [person ?.firstname, person ?.surname]
44 | )
```



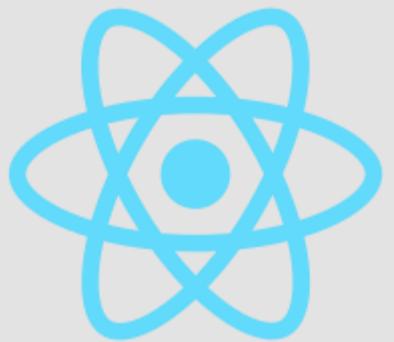


See you in the next video



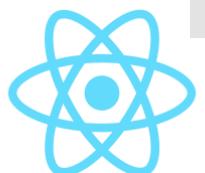
# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



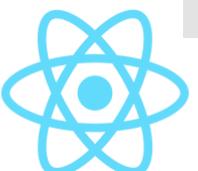
# Kimrof

A [Formik](#) like forms utility



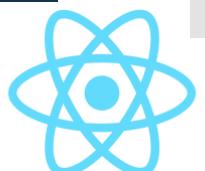
# Kimrof

- Putting it all together
  - Combining hooks, context and components
- Kimrof, the demo library
  - A Formik like forms library

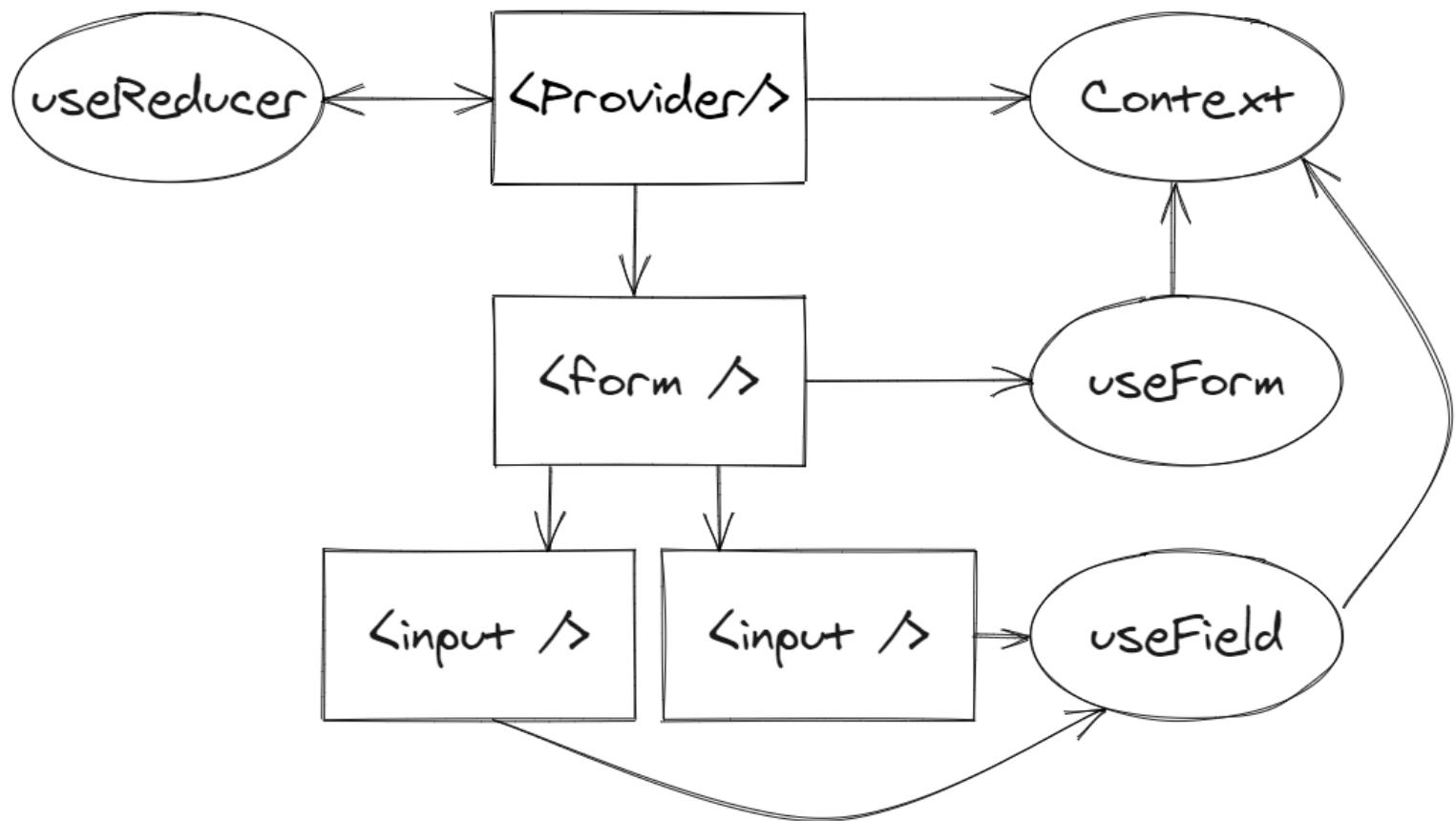


# Kimrof Example

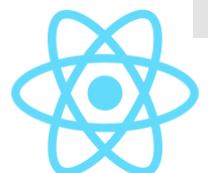
```
TS UserEditor.tsx M X
src > kimrof-user-editor > TS UserEditor.tsx > ...
You, seconds ago | 1 author (You)
1 import React, { ReactElement } from "react"
2 import { KimrofLabeledField, useKimrof, useKimrofForm } from "./kimrof"
3
4 export function UserEditor(): ReactElement {
5   const {
6     metadata: { isDirty, isValid },
7   } = useKimrof()
8   const formProps = useKimrofForm()
9
10  return (
11    <form className="person-editor" { ...formProps}>
12      <h2>Kimrof User Editor</h2>
13      <KimrofLabeledField label="Firstname:" name="firstname" />
14      <KimrofLabeledField label="Surname:" name="surname" />
15      <KimrofLabeledField label="Email:" name="email" />
16      <KimrofLabeledField label="Address:" name="address" />
17      <KimrofLabeledField label="Phone:" name="phone" />
18      <button className="btn btn-primary" disabled={!isDirty || !isValid}>
19        Save
20      </button>
21    </form>
22  )
23 }
```



# Kimrof Structure

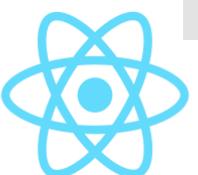
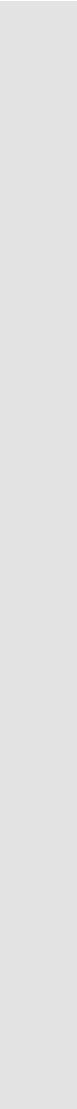


Made with Excalidraw



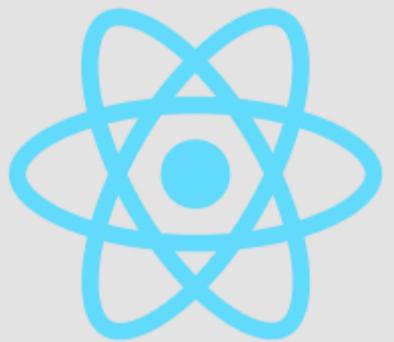


See you in the next video

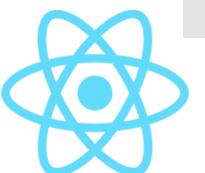


# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb

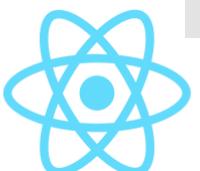


# Context and Provider

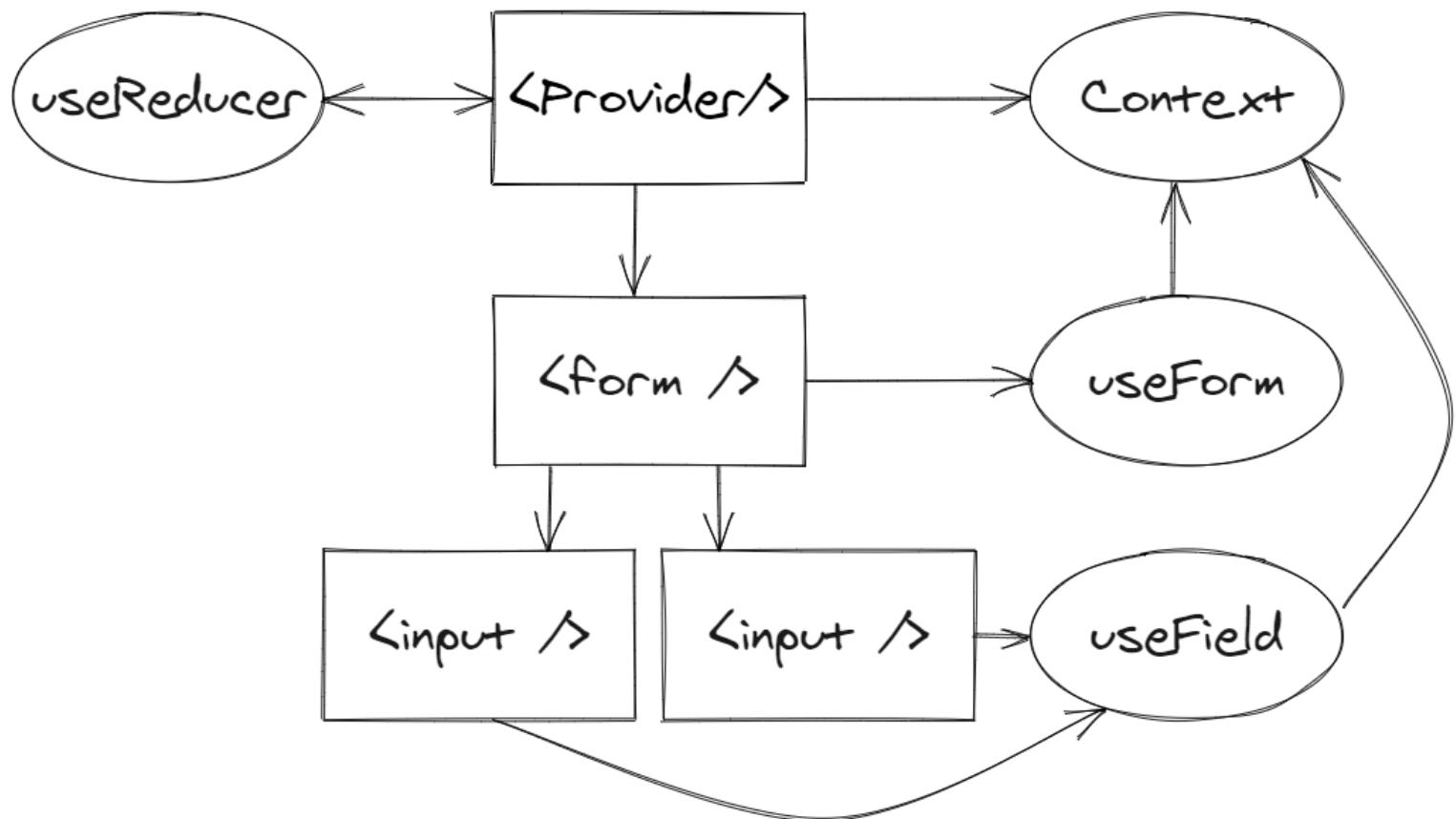


# Context and Provider

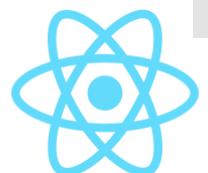
- The provider component will **manage the internal state**
  - The object to edit
  - Updates to it's properties
  - Validity/dirty states
  - Form submission
- The context will **make this available**
  - To the various hooks



# Kimrof Structure

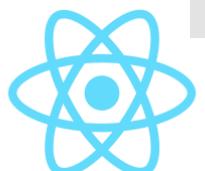


Made with Excalidraw



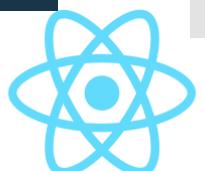
# The kimrofContext

```
TS KimrofContext.ts 2, M ×  
src > kimrof-user-editor > kimrof > TS KimrofContext.ts > ...  
You, 2 minutes ago | 1 author (You)  
1 import { React, { createContext } } from "react"  
2  
3 import { KimrofObject, KimrofProperty } from "./Types"  
4  
You, 2 minutes ago | 1 author (You)  
5 export interface KimrofContext {  
6   values: KimrofObject  
7 }  
8  
9 export const kimrofContext = createContext<KimrofContext>({  
10   values: {},  
11 })
```



# The Kimrof provider

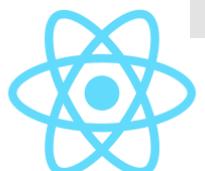
```
TS Kimrof.tsx 1, M X
src > kimrof-user-editor > kimrof > TS Kimrof.tsx > ...
    You, seconds ago | 1 author (You)
1 // Kimrof = Formik reversed :-)
2
3| import React, { ReactElement, ReactNode, useMemo } from "react"
4
5| import { KimrofObject, KimrofProperty } from "./Types"
6| import { KimrofContext, kimrofContext } from "./KimrofContext"
7
    You, seconds ago | 1 author (You)
8 interface Props<TData> {
9   children: ReactNode
10|  initialValues: TData
11 }
12
13 export function Kimrof<TData extends KimrofObject>({
14   children,
15|  initialValues,
16 }: Props<TData>): ReactElement {
17   const values = initialValues
18
19   const context: KimrofContext = useMemo(() => ({ values }), [values])
20
21   return (
22     <kimrofContext.Provider value={context}>{children}</kimrofContext.Provider>
23   )
24 }
```



# Adding the provider

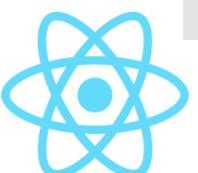
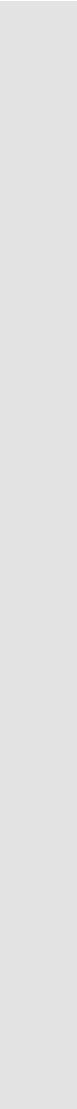


```
TS KimrofUserEditor.tsx M X
src > kimrof-user-editor > TS KimrofUserEditor.tsx > ...
You, seconds ago | 1 author (You)
1 import React, { ReactElement } from "react"
2
3 import { initialPerson } from "../utils"
4 import { IndexedPerson } from "../types/IndexedPerson"
5
6 // Kimrof = Formik reversed :-)
7 import { UserEditor } from "./UserEditor"
8 import { Kimrof } from "./kimrof"
9
10 export function KimrofUserEditor(): ReactElement {
11   return (
12     <Kimrof initialValues={initialPerson as IndexedPerson}>
13       <UserEditor />
14     </Kimrof>
15   )
16 }
```



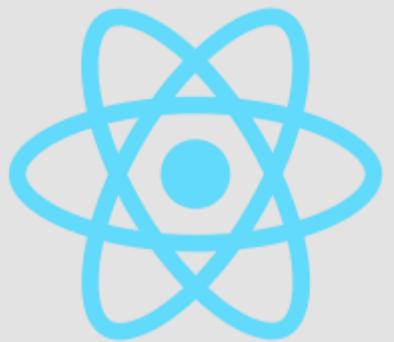


See you in the next video

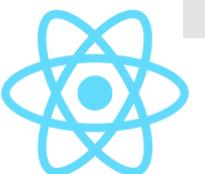


# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb

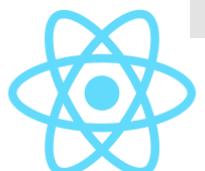


# Displaying the values

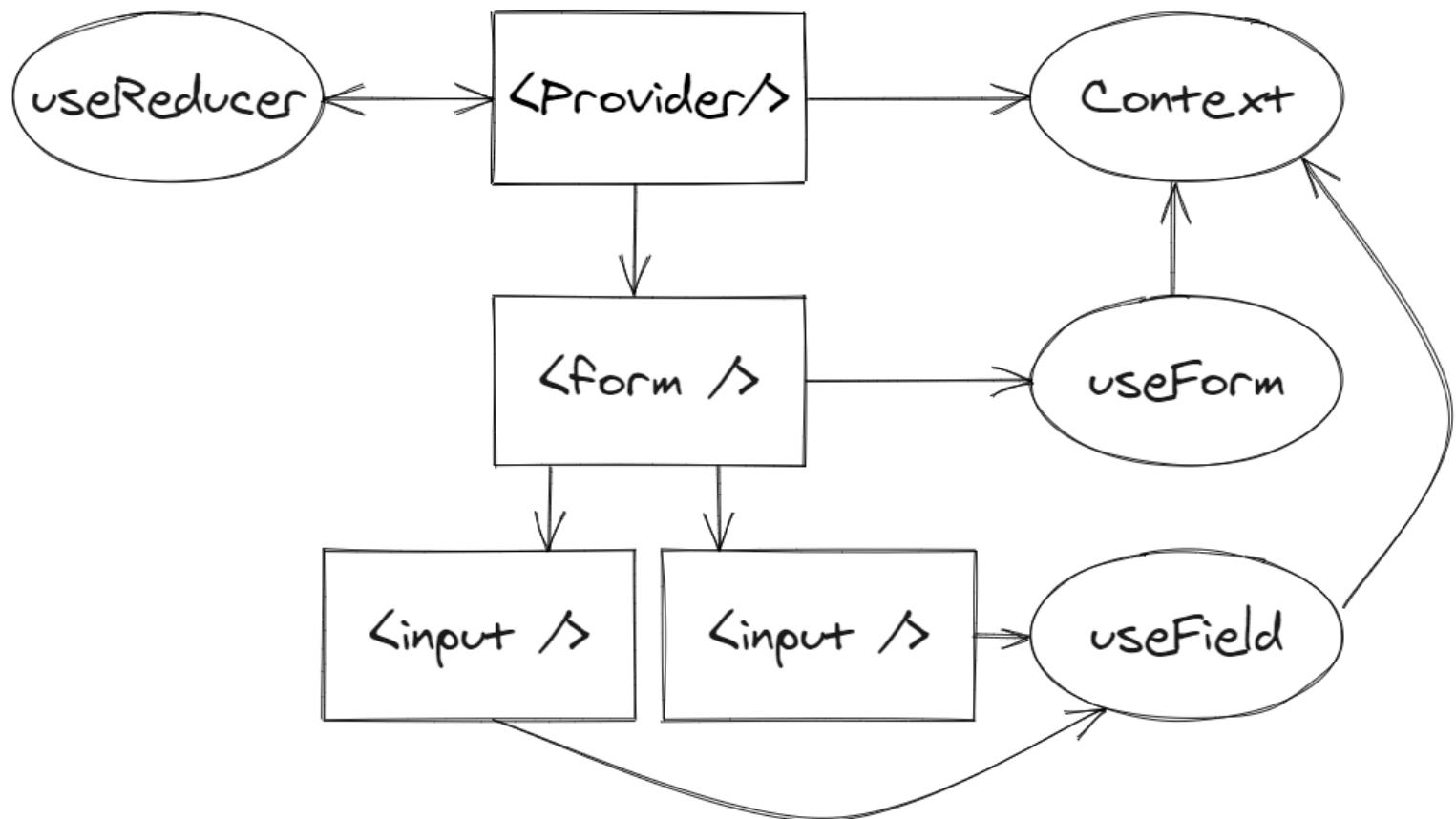


# Displaying the form values

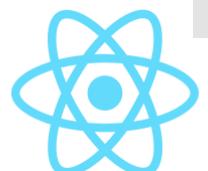
- The **useKimrofField()** hook allows **input components** to work
  - Returns an object with a value and onChanged prop
- **Spread the result** into any HTML <input />



# Kimrof Structure

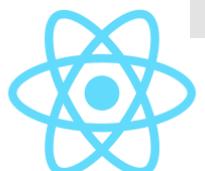


Made with Excalidraw



# The useKimrofField() hook

```
TS useKimrofField.ts 1, M ×  
src > kimrof-user-editor > kimrof > TS useKimrofField.ts > ...  
You, a minute ago | 1 author (You)  
1 import { ChangeEvent, useCallback, useContext } from "react"  
2  
3 import { kimrofContext } from "./KimrofContext"  
4  
5 export function useKimrofField(name: string) {  
6   const { values } = useContext(kimrofContext)  
7  
8   const onChange = useCallback((e: ChangeEvent<HTMLInputElement>) => {  
9     // TODO  
10    }, [])  
11  
12   return {  
13     value: values[name],  
14     onChange,  
15   } as const  
16 }
```



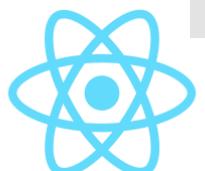
# The KimrofLabeledField component



**MAKE IT SO**

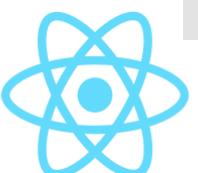
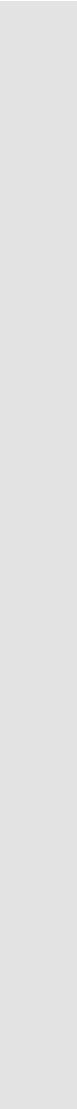
memegenerator.net

```
TS KimrofLabeledField.tsx M X
src > kimrof-user-editor > kimrof > TS KimrofLabeledField.tsx > ...
You, seconds ago | 1 author (You)
1 import React, { ComponentProps, ReactElement } from "react"
2
3 import { LabeledInput } from "../../components"
4 import { useKimroffField } from "./useKimroffField"
5
6 type LabeledInputProps = ComponentProps<typeof LabeledInput>
7
8 You, seconds ago | 1 author (You)
9 interface Props extends Omit<LabeledInputProps, "onChange" | "value"> {
10   name: string
11 }
12 export function KimrofLabeledField(props: Props): ReactElement {
13   const fieldProps = useKimroffField(props.name)
14
15   return <LabeledInput { ...props} { ...fieldProps} />
16 }
```



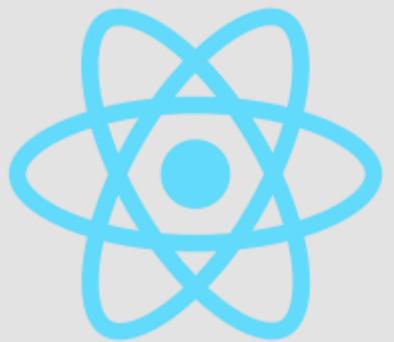


See you in the next video

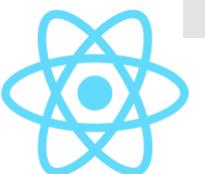


# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb

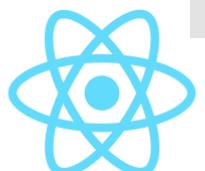


# Editing data



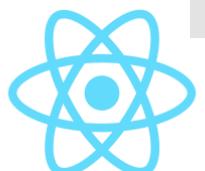
# Editing data

- The **Kimrof state** is maintained with a **useReducer() hook**
  - Supports a set-property action for when an edit is made
- Add the **onChange handler** to the **useKimrofField()** hook
  - Dispatch the set-property action when a change is detected



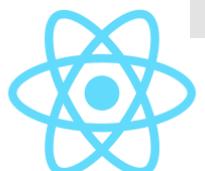
# The kimrofReducer()

```
TS kimrofReducer.ts M X
src > kimrof-user-editor > kimrof > TS kimrofReducer.ts > ...
20  export function kimrofReducer(
21    state: ReducerState,
22    action: SomeAction
23  ): ReducerState {
24    switch (action.type) {
25      case "set-property":
26        return {
27          ... state,
28          metadata: { ... state.metadata, isDirty: true },
29          values: {
30            ... state.values,
31            [action.payload.name]: action.payload.value,
32          },
33        }
34    }
35
36    return state
37 }
```



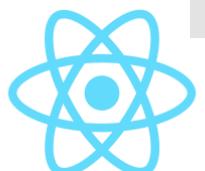
# The KimrofContext

```
TS KimrofContext.ts 1, M X
src > kimrof-user-editor > kimrof > TS KimrofContext.ts > ...
You, seconds ago | 1 author (You)
1 import React, { createContext } from "react"
2
3 import { KimrofObject, KimrofProperty } from "./Types"
4
You, seconds ago | 1 author (You)
5 export interface KimrofContext {
6   values: KimrofObject
7   setFieldValue: (name: string, value: KimrofProperty) => void
8 }
9
10 export const kimrofContext = createContext<KimrofContext>({
11   values: {},
12   setFieldValue: () => void null,
13 })
```



# The Kimrof Provider

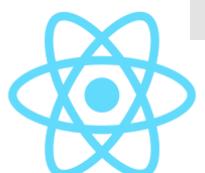
```
TS Kimrof.tsx M 
src > kimrof-user-editor > kimrof > TS Kimrof.tsx > ...
14  export function Kimrof<TData extends KimrofObject>({
15    children,
16    initialValues,
17  }: Props<TData>): ReactElement {
18    const [{ values }, dispatch] = useReducer(kimrofReducer, {
19      values: initialValues,
20      metadata: { isDirty: false, isValid: true },
21    })
22
23    const context: KimrofContext = useMemo(
24      () => ({
25        values,
26        setFieldValue: (name: string, value: KimrofProperty) => {
27          dispatch({ type: "set-property", payload: { name, value } })
28        },
29      }),
30      [values]
31    )
32
33    return (
34      <kimrofContext.Provider value={context}>{children}</kimrofContext.Provider>
35    )
36 }
```



# The useKimrofField() hook

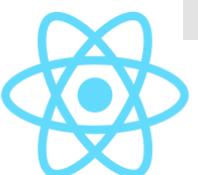
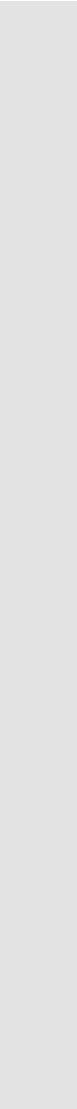


```
TS useKimrofField.ts M X
src > kimrof-user-editor > kimrof > TS useKimrofField.ts > ...
5  export function useKimrofField(name: string) {
6    const { values, setFieldValue } = useContext(kimrofContext)
7
8    const onChange = useCallback(
9      (e: ChangeEvent<HTMLInputElement>) => {
10        setFieldValue(e.target.name, e.target.value)
11      },
12      [setFieldValue]
13    )
14
15    return {
16      value: values[name],
17      onChange,
18    } as const
19 }
```



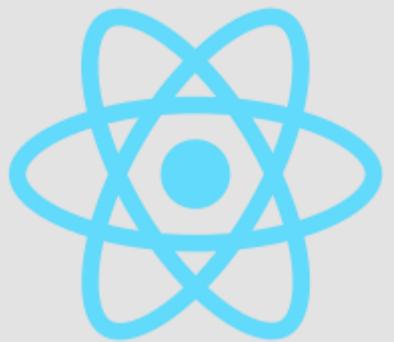


See you in the next video

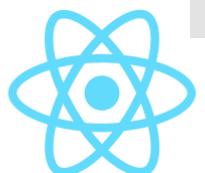


# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb

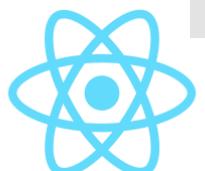


# Submitting the form data

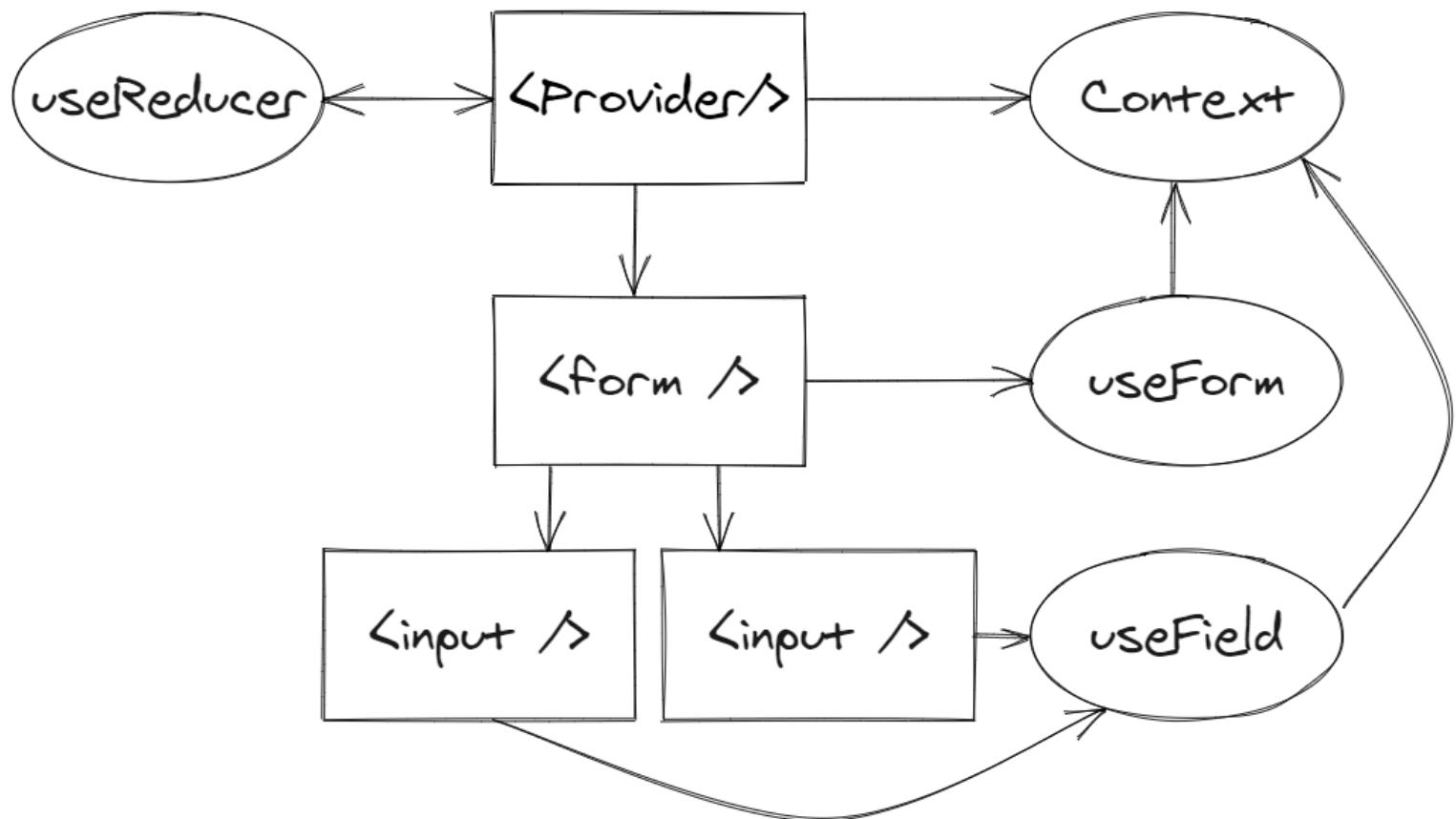


# Submitting the form data

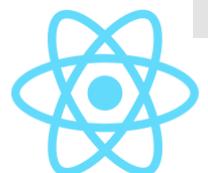
- The **useKimrofForm()** hook allows a form to be **submitted**
  - Returns an object with an onSubmit prop
  - Spread the result into the HTML <form />
- The **useKimrof()** hook returns values and metadata
  - The dirty and valid flags



# Kimrof Structure

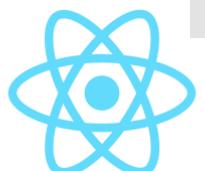


Made with Excalidraw



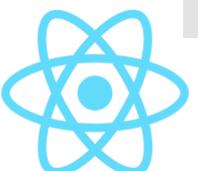
# The kimrofContext

```
TS KimrofContext.ts 1, M X
src > kimrof-user-editor > kimrof > TS KimrofContext.ts > ...
6  export interface KimrofContext {
7    values: KimrofObject
8    metadata: Metadata
9    submitForm: () => void
10   setFieldValue: (name: string, value: KimrofProperty) => void
11 }
12
13 export const kimrofContext = createContext<KimrofContext>({
14   values: {},
15   metadata: { isDirty: false, isValid: true },
16   submitForm: () => void null,
17   setFieldValue: () => void null,
18 })
```



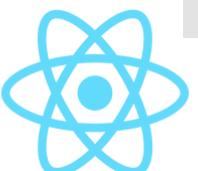
# The Kimrof provider

```
ts Kimrof.tsx M X
src > kimrof-user-editor > kimrof > ts Kimrof.tsx > Kimrof
  9 interface Props<TData> {
10   children: ReactNode
11   initialValues: TData
12 |   onSubmit: (values: TData) => void
13 }
14
15 export function Kimrof<TData extends KimrofObject>(
16   children,
17   initialValues,
18 |   onSubmit,
19 ): Props<TData>: ReactElement {
20 |   const [{ values, metadata }, dispatch] = useReducer(kimrofReducer, {
21 |     values: initialValues,
22 |     metadata: { isDirty: false, isValid: true },
23 |   })
24
25   const context: KimrofContext = useMemo(
26     () => ({
27       values,
28       metadata,
29 |       submitForm: () => onSubmit(values as TData),
30       setFieldValue: (name: string, value: KimrofProperty) => {
31         dispatch({ type: "set-property", payload: { name, value } })
32       },
33     }),
34   [values, metadata, onSubmit]
35 )
```



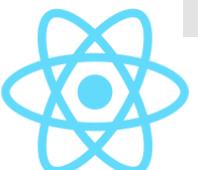
# Adding the onSubmit handler to KimrofUserEditor

```
TS KimrofUserEditor.tsx M X
src > kimrof-user-editor > TS KimrofUserEditor.tsx > ...
10  export function KimrofUserEditor(): ReactElement {
11    return (
12      <Kimrof
13        initialValues={initialPerson as IndexedPerson}
14        onSubmit={(person) => alert(JSON.stringify(person, null, 2))}
15      >
16        <UserEditor />
17      </Kimrof>
18    )
19 }
```



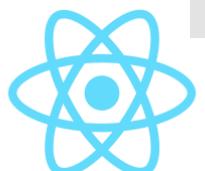
# The useKimrofForm() hook

```
TS useKimrofForm.ts M X
src > kimrof-user-editor > kimrof > TS useKimrofForm.ts > ...
You, seconds ago | 1 author (You)
1 import { useCallback, useContext } from "react"
2
3 import { kimrofContext } from "./KimrofContext"
4
5 export function useKimrofForm() {
6   const { submitForm } = useContext(kimrofContext)
7
8   const onSubmit = useCallback(
9     (e: React.FormEvent) => {
10       e.preventDefault()
11       submitForm()
12     },
13     [submitForm]
14   )
15
16   return {
17     onSubmit,
18   } as const
19 }
```



# The useKimrof() hook

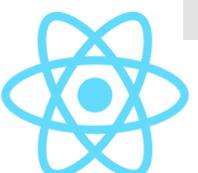
```
TS useKimrof.ts M X
src > kimrof-user-editor > kimrof > TS useKimrof.ts > ...
You, seconds ago | 1 author (You)
1 import { useContext } from "react"
2 import { kimrofContext } from "./KimrofContext"
3
4 export function useKimrof() {
5   const { values, metadata } = useContext(kimrofContext)
6
7   return { values, metadata } as const
8 }
```



# Updating the UserEditor

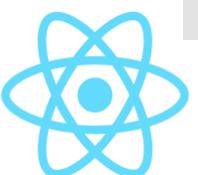
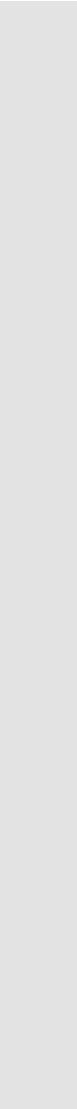


```
TS UserEditor.tsx M X
src > kimrof-user-editor > TS UserEditor.tsx > ...
1 import React, { ReactElement } from "react"
2 import { KimrofLabeledField, useKimrof, useKimrofForm } from "./kimrof"
3
4 export function UserEditor(): ReactElement {
5   const formProps = useKimrofForm()
6   const {
7     values,
8     metadata: { isDirty },
9   } = useKimrof()
10
11   return (
12     <form className="person-editor" { ...formProps}>
13       <h2>Kimrof User Editor</h2>
14       <KimrofLabeledField label="Firstname:" name="firstname" />
15       <KimrofLabeledField label="Surname:" name="surname" />
16       <KimrofLabeledField label="Email:" name="email" />
17       <KimrofLabeledField label="Address:" name="address" />
18       <KimrofLabeledField label="Phone:" name="phone" />
19       <button className="btn btn-primary" disabled={!isDirty}>
20         Save
21       </button>
22       <hr />
23       <pre>{ JSON.stringify(values, null, 2) }</pre>
24     </form>
25   )
26 }
```



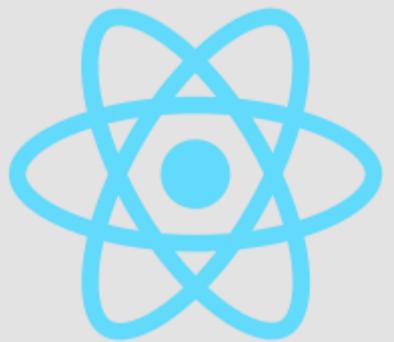


See you in the next video

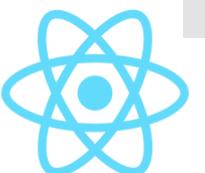


# React Hooks Tips Only the Pros Know

Maurice de Beijer - @mauricedb



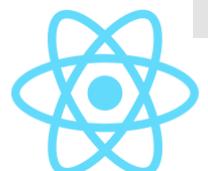
# Form Validation



## Adding validation to the KimrofPersonEditor

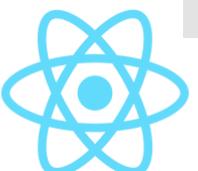


```
10  export function KimrofPersonEditor(): ReactElement {
11    return (
12      <Kimrof
13        initialValues={initialPerson as IndexedPerson}
14        onSubmit={(person) => {
15          alert(`Submitting\n${JSON.stringify(person, null, 2)}`);
16        }}
17      >
18        <PersonEditor />
19      </Kimrof>
20    );
21 }
```



# Conclusion

- **Hooks are simple** but stick to the rules
  - They exist for a good reason
- The implementation of hooks is **simple and elegant**
- **Custom hooks** are extremely useful
  - Keep them small and focused and use hook composition
- Hooks are a great way to **provide non UI functionality**
  - When building non UI React libraries
  - Regardless of inhouse or published to NPM



Maurice de Beijer

@mauricedb

maurice.de.beijer  
@gmail.com

