

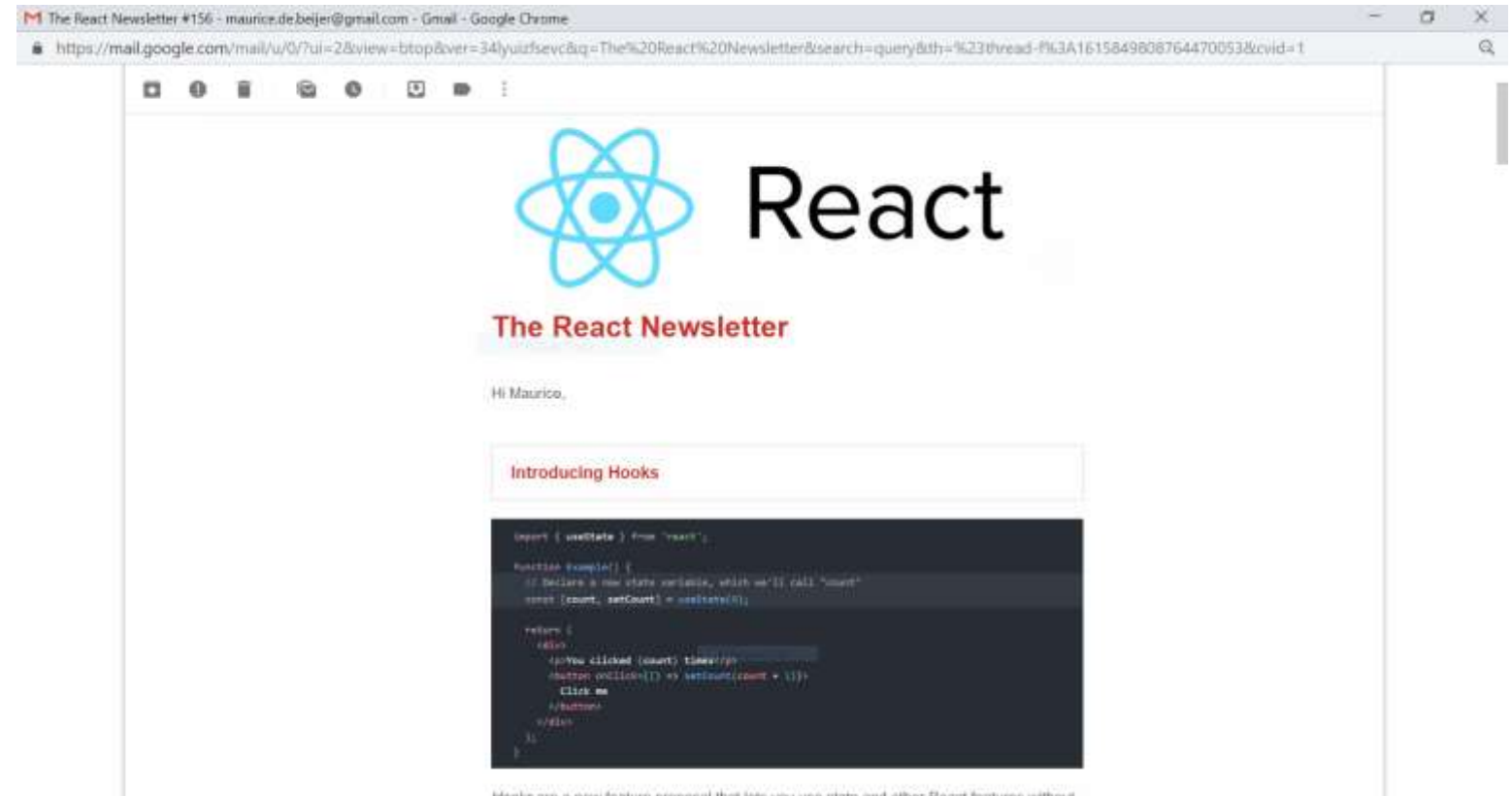
# Why I am hooked on the future of React

Maurice de Beijer - @mauricedb



- Maurice de Beijer
- The Problem Solver
- Microsoft MVP
- Freelance developer/instructor
- Twitter: @mauricedb
- Web: <http://www.TheProblemSolver.nl>
- E-mail: [maurice.de.beijer@gmail.com](mailto:maurice.de.beijer@gmail.com)

# The React Newsletter



<http://bit.ly/ReactNewsletter>

# History of Components



# Original Component API

```
1 import React from 'react';  
2  
3 const Greeter = React.createClass({  
4   render: function() {  
5     return <div>Hello {this.props.firstName}</div>;  
6   }  
7 });  
8  
9 export default Greeter;
```

# Class Components

```
1 import React, { Component } from 'react';  
2  
3 class Greeter extends Component {  
4   render() {  
5     return <div>Hello {this.props.firstName}</div>;  
6   }  
7 }  
8  
9 export default Greeter;
```

# State & Lifecycle

```
1 import React, { Component } from 'react';
2
3 class Greeter extends Component {
4   state = {
5     loaded: false
6   };
7
8   componentDidMount() {
9     this.setState({ loaded: true });
10  }
11
12  render() {
13    return <div>Hello {this.props.firstName}</div>;
14  }
15 }
16
17 export default Greeter;
```



# The problem with classes





this

```
1 class Counter extends Component {  
2   state = { count: 0 };  
3  
4   onClick() {  
5     this.setState({ count: this.state.count + 1 });  
6   }  
7 }  
8  
9 <button onClick={this.onClick}>  
10   Click me  
11 </button>  
12  
13 );  
14 }  
15 }
```

✖ ▶ Uncaught TypeError: Cannot read property 'setState' of undefined  
at onClick (Counter.js:79)  
at HTMLUnknownElement.callCallback (react-dom.development.js:147)  
at Object.invokeGuardedCallbackDev (react-dom.development.js:196)  
at invokeGuardedCallback (react-dom.development.js:250)  
at invokeGuardedCallbackAndCatchFirstError (react-dom.development.js:265)  
at executeDispatch (react-dom.development.js:622)  
at executeDispatchesInOrder (react-dom.development.js:647)

```
12 </button>  
13 );  
14 }  
15 }
```

# Fat arrow for some functions

```
1 class Counter extends Component {  
2   state = { count: 0 };  
3  
4   onClick = () => {  
5     this.setState({ count: this.state.count + 1 });  
6   };  
7  
8   render() {  
9     return (  
10      <button onClick={this.onClick}>  
11        Click me  
12      </button>  
13    );  
14  }  
15 }
```

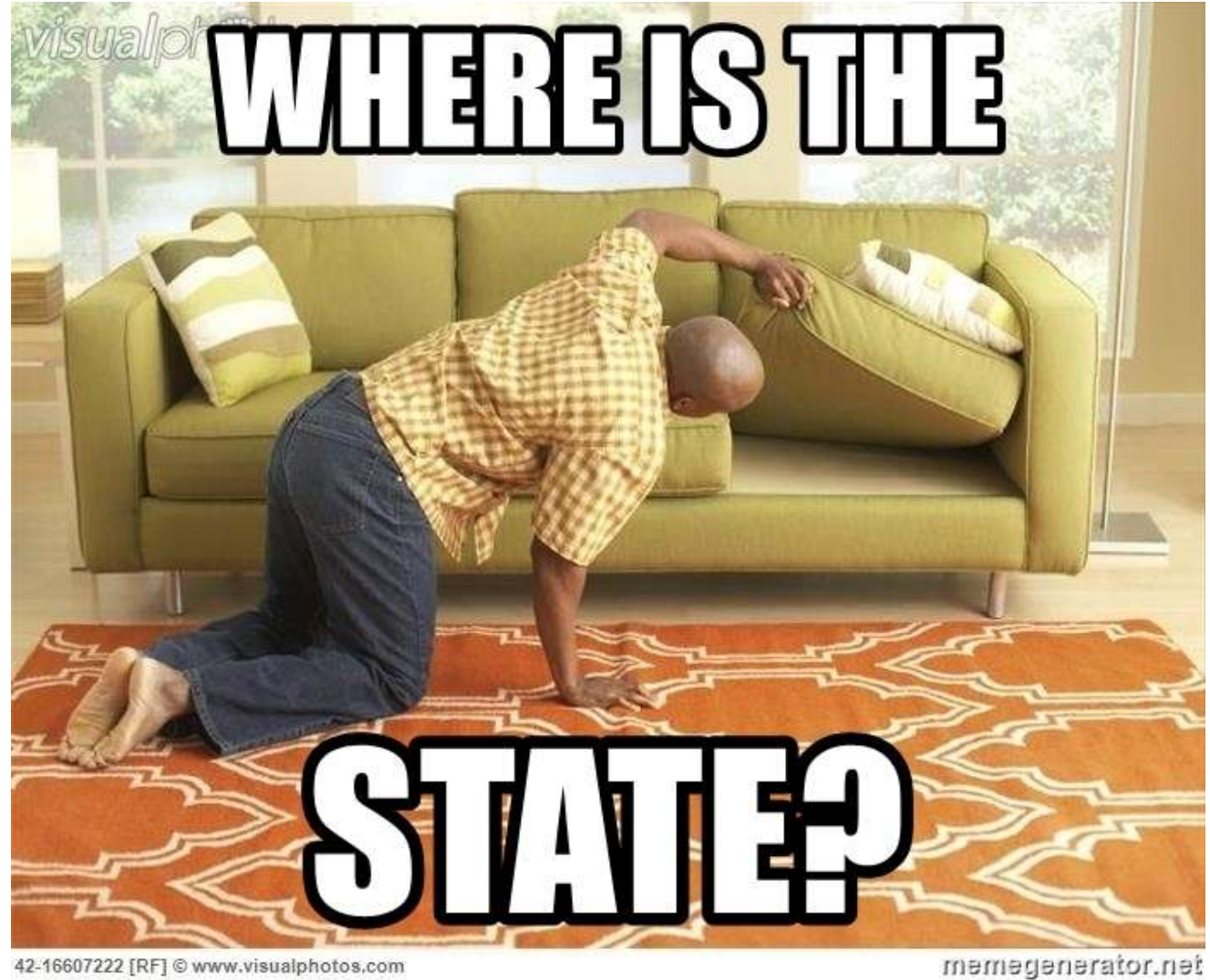
# Single responsibility in multiple functions

```
1 class Clock extends Component {  
2   state = { now: new Date().toLocaleTimeString() };  
3  
4   componentDidMount() {  
5     this.handle = setInterval(  
6       () =>  
7         this.setState({  
8           now: new Date().toLocaleTimeString()  
9         })),  
10    1000  
11  );  
12  }  
13  componentWillUnmount() {  
14    clearInterval(this.handle);  
15  }  
16  render() {  
17    return <div>{this.state.now}</div>;  
18  }  
19 }
```

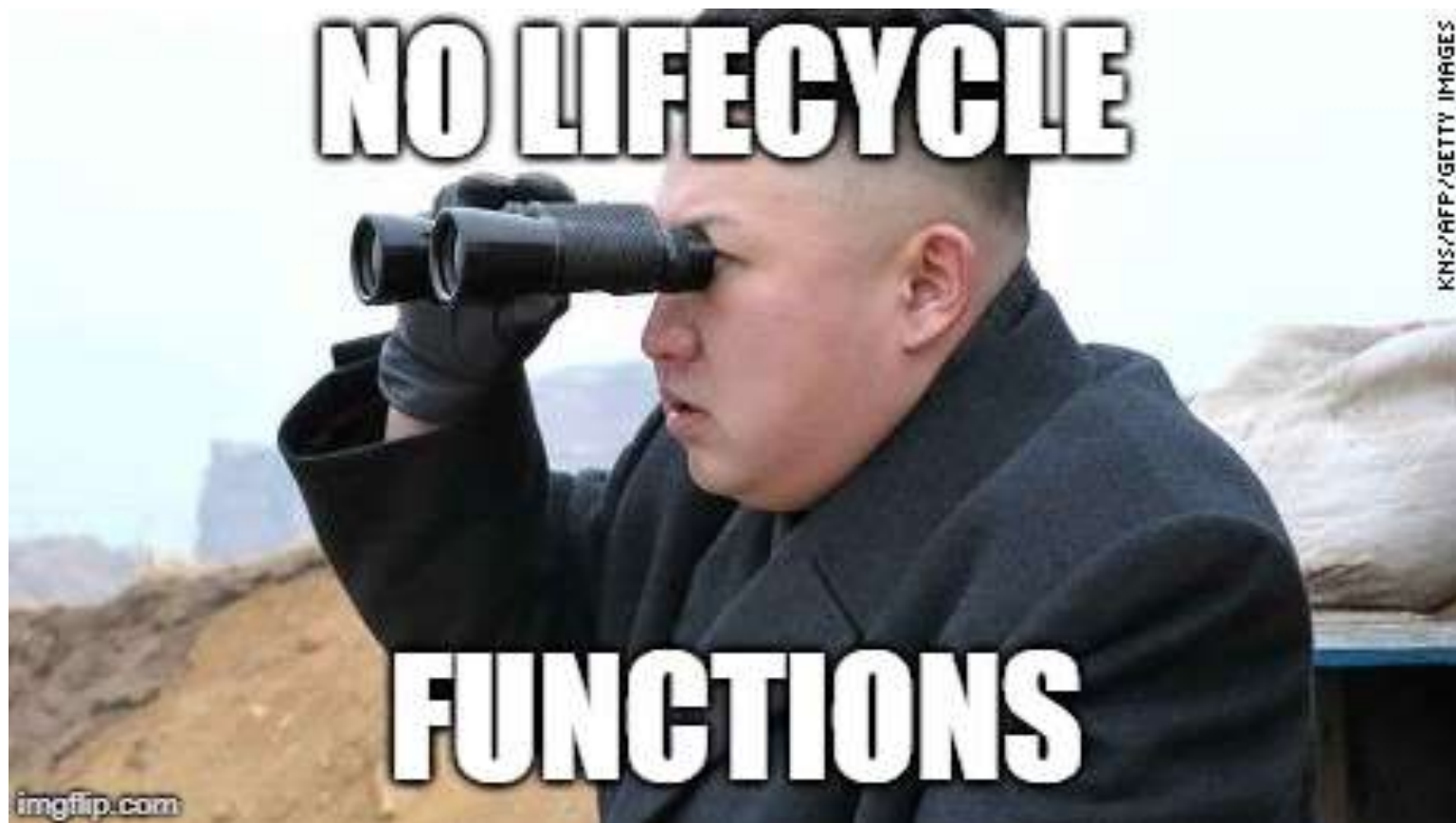
# Functional Components

```
1 import React from 'react';  
2  
3 const Greeter = props => {  
4   return <div>Hello {props.firstName}</div>;  
5 };  
6  
7 export default Greeter;
```

# The Problem



And





If your only  
tool ...





# Deeply Nested Components

[illegible]

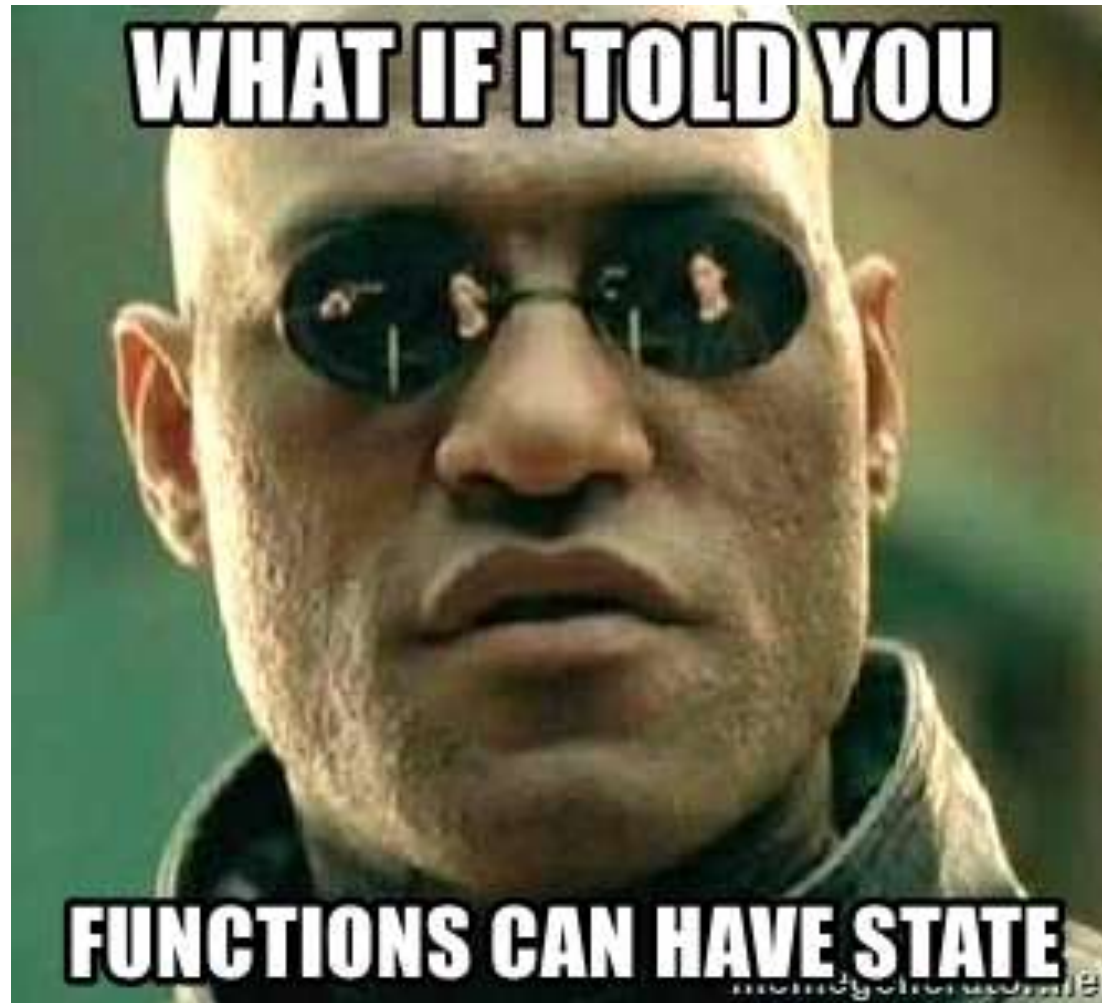
Many are  
wrappers to  
add props

```
▼ <PresTrackedContainer className="billboard-presentation-tracking">
  ▼ <div className="ptrack-container billboard-presentation-tracking">
    ▼ <getTrackingInfoFromContext(createTrackingComponent(billboard)) className="billboard-presentation-trac
      " videoId={80190859}>
      ▼ <createTrackingComponent(billboard) className="billboard-presentation-tracking" imageKey="BILLBOARD
        ▼ <div className="billboard-presentation-tracking ptrack-content">
          ▼ <getTrackingInfoFromContext(createTrackingComponent(boxArt)) className="billboard-presentation
            BILLBOARD|6d853480-ce72-11e8-b627-0e319b527290|en" videoId={80190859}>
            ▼ <createTrackingComponent(boxArt) className="billboard-presentation-tracking" imageKey="BILL
              >
              ▼ <div className="billboard-presentation-tracking ptrack-content">
                ▼ <logPresentationManually(getTrackingInfoFromContext(windowVisibility(inViewport(Connec
                  BILLBOARD|6d853480-ce72-11e8-b627-0e319b527290|en" videoId={80190859} backgroundImageS
                  useAvailablePhase={true}>
                  ▼ <getTrackingInfoFromContext(windowVisibility(inViewport(ConnectToApps(e)))) isMotion
                    BILLBOARD|6d853480-ce72-11e8-b627-0e319b527290|en" videoId={80190859} backgroundIma
                    useAvailablePhase={true}>
                    ▼ <windowVisibility(inViewport(ConnectToApps(e))) isMotionEnabled={true} imageKey="
                      80190859} backgroundImageStartsPlay={false} trackId={254015180} hasScrolled={true}
                      ▼ <inViewport(ConnectToApps(e)) isMotionEnabled={true} imageKey="BILLBOARD|6d853.
                        backgroundImageStartsPlay={false} trackId={254015180} hasScrolled={true} useAv
                        ={false} ignoreElementWithNoDimensions={false}>
                        ▼ <ConnectToApps(e) isMotionEnabled={true} imageKey="BILLBOARD|6d853480-ce72-1
                          backgroundImageStartsPlay={false} trackId={254015180} hasScrolled={true} use
                          defaultInViewportState={false} ignoreElementWithNoDimensions={false} inViewp
                          ▼ <e muted={true} isMotionEnabled={true} imageKey="BILLBOARD|6d853480-ce72-
                            backgroundImageStartsPlay={false} trackId={254015180} hasScrolled={true}
                            ...
```

# Render Props

```
1 import React, { Component } from 'react';
2 import TimeContext from './TimeContext';
3 import ThemeContext from './ThemeContext';
4 import AnalogClock from './AnalogClock';
5
6 class Clock extends Component {
7   render() {
8     return (
9       <TimeContext.Consumer>
10         ({ { time } }) => (
11           <ThemeContext.Consumer>
12             ({ { theme } }) => (
13               <AnalogClock time={time} theme={theme} />
14             )
15           </ThemeContext.Consumer>
16         )
17       </TimeContext.Consumer>
18     );
19   }
20 }
21
22 export default Clock;
```

But...



# Introducing Hooks





Open  
Mind

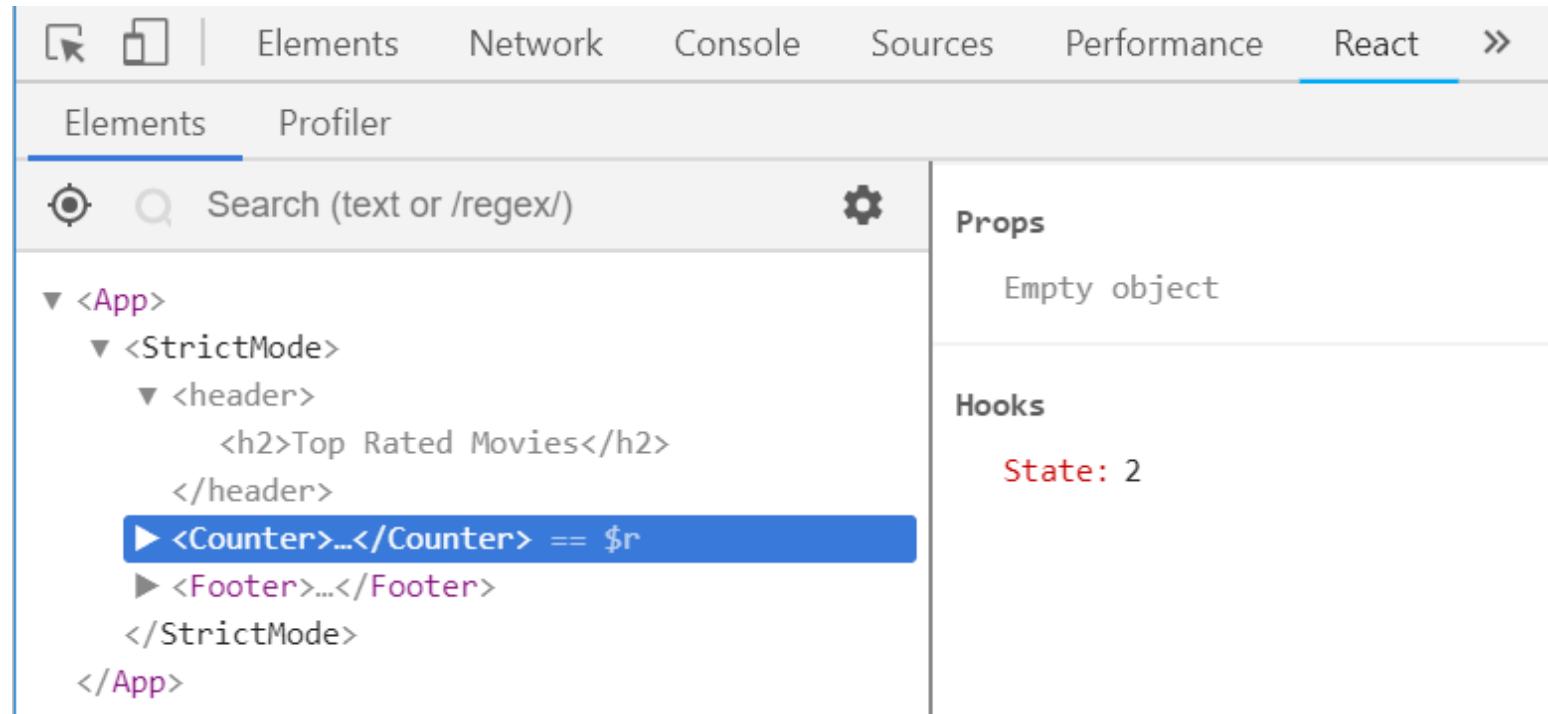


# useState()

```
1 import React, { useState } from 'react';
2
3 const Counter = () => {
4   const [count, setCount] = useState(1);
5
6   return (
7     <div>
8       <p>The counter is: {count}</p>
9       <button onClick={
10         () => setCount(count + 1)}>
11         Increment
12       </button>
13     </div>
14   );
15 };
16
17 export default Counter;
```



# Just State



# useEffect()

```
1 import React, { useEffect, useState } from 'react';
2 import AnalogClock from './AnalogClock';
3
4 const Clock = ({ interval }) => {
5   const [time, setTime] = useState(new Date());
6
7   useEffect(
8     () => {
9       const handle = setInterval(
10         () => setTime(new Date()), interval);
11
12       return () => clearInterval(handle);
13     },
14     [interval]
15   );
16
17   return <AnalogClock time={time} />;
18 };
19
20 export default Clock;
```

# The Context



```
1 import { createContext } from 'react';  
2  
3 const TimeContext = createContext(new Date());  
4  
5 export default TimeContext;
```

# Render Props

```
1 import React, { Component } from 'react';
2 import TimeContext from './TimeContext';
3 import ThemeContext from './ThemeContext';
4 import AnalogClock from './AnalogClock';
5
6 class Clock extends Component {
7   render() {
8     return (
9       <TimeContext.Consumer>
10         ({ { time } }) => (
11           <ThemeContext.Consumer>
12             ({ { theme } }) => (
13               <AnalogClock time={time} theme={theme} />
14             )
15           </ThemeContext.Consumer>
16         )
17       </TimeContext.Consumer>
18     );
19   }
20 }
21
22 export default Clock;
```

# useContext()

```
1 import React, { useContext } from 'react';
2 import TimeContext from './TimeContext';
3 import ThemeContext from './ThemeContext';
4 import AnalogClock from './AnalogClock';
5
6 const Clock = () => {
7   const time = useContext(TimeContext);
8   const theme = useContext(ThemeContext);
9
10  return <AnalogClock time={time} theme={theme}
11  >>;
12
13 export default Clock;
```

# All hooks

- Basic hooks
  - useState()
  - useEffect()
  - useContext()
- Additional Hooks
  - useReducer()
  - useCallback()
  - useMemo()
  - useRef()
  - useDebugValue()
  - useEffect()
  - useImperativeHandle()
- Custom hooks
  - ...

# Custom Hooks

```
1 import { useEffect, useState } from 'react';
2
3 const useTime = interval => {
4   const [time, setTime] = useState(new Date());
5
6   useEffect(
7     () => {
8       const handle = setInterval(
9         () => setTime(new Date()), interval);
10
11       return () => clearInterval(handle);
12     },
13     [interval]
14   );
15   return time;
16 };
17
18 export default useTime;
```



# Using The Hook

```
1 import React from 'react';
2 import useTime from './useTime';
3 import AnalogClock from './AnalogClock';
4
5 const Clock = ({ interval }) => {
6   const time = useTime(interval);
7
8   return <AnalogClock time={time} />;
9 };
10
11 export default Clock;
```

# Abortable Fetch Hook

```
1 import {
2   useState, useEffect, useLayoutEffect, useRef
3 } from 'react';
4 import fetchData from './fetchData';
5
6 const useAbortableFetch = url => {
7   const [state, setState] = useState({
8     json: null,
9     loading: false,
10    error: null,
11    controller: null
12  });
13
14  const isMounted = useRef(false);
15  useLayoutEffect(() => {
16    isMounted.current = true;
17    return () => {
18      isMounted.current = false;
19    };
20  }, []);
21
22  useEffect(
23    () => {
24      const controller = new AbortController();
25      setState({
26        json: null,
27        loading: true,
28        error: null,
29        controller
30      });
31
32      fetchData(url, controller.signal, state => {
33        if (isMounted.current) {
34          setState(state);
35        }
36      });
37
38      return () => controller.abort();
39    },
40    [url]
41  );
42
43  return {
44    json: state.json,
45    loading: state.loading,
46    error: state.error,
47    abort: () => state.controller 66 state.controller.abort()
48  };
49 };
50
51 export default useAbortableFetch;
```

# Fetch Request

```
1 const fetchData = async (url, signal, setState) => {
2   try {
3     const rsp = await fetch(url, { signal });
4     const json = await rsp.json();
5
6     setState(oldState => ({
7       ...oldState,
8       json,
9       loading: false
10    }));
11  } catch (err) {
12    const error = err.name !== 'AbortError' ? err.message :
13    null;
14    setState(oldState => ({
15      ...oldState,
16      error,
17      loading: false
18    }));
19  }
20 };
21
22 export default fetchData;
```

# Rules of Hooks

- Hooks can only be used in functional components
  - Or in other hooks
  - Not in class based components
- Hooks must always be created in the same order
  - Must be outside loops, conditions or nested functions
- Hooks names must be prefixed with `use`
- There is an ESLint plugin to enforce these rules

Hooks are  
optional



Classes will  
keep on  
working



Maurice de Beijer

@mauricedb

