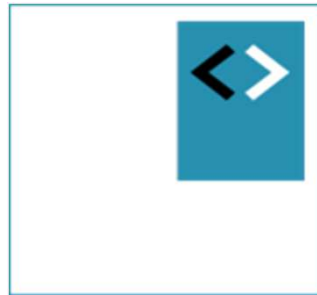


React Fundamentals

Module – Class based components



Peter Kassenaar –
info@kassenaar.com



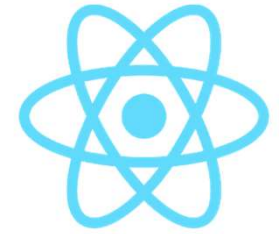
Class-based Components

Using ES6-classes for your components, methods, state and props

Types of components

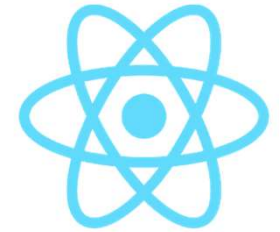
Function Components

Class Components



Class-based components

- In this concept, components are **just classes**
- Classes have a mandatory `render()` function
- They can have an optional `constructor`, methods and a `state` object
 - `state` in the constructor
 - `state` as standalone class property
- *You don't have to learn new concepts, as all components are equal!*
- It's a matter of personal preference



For instance: <DisplayCounter />

Import {Component}
also

```
import React, {Component} from 'react'
```

```
class DisplayCounter extends Component {  
  // Optional constructor, to initialize the component
```

Constructor
w/ props

```
  constructor(props) {
```

```
    super(props);
```

```
    this.state = {
```

```
      // optional: some local state
```

Local state

render()
function

```
  // mandatory render() function, to render the UI
```

```
  render() {
```

```
    return (
```

```
      <h2>
```

```
        {this.props.counter}
```

```
      </h2>
```

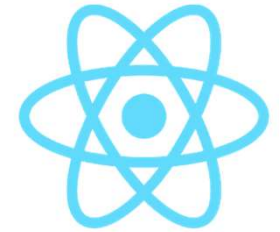
```
    )
```

```
  }
```

```
}
```

Access props and
methods via this
keyword

```
export default DisplayCounter
```



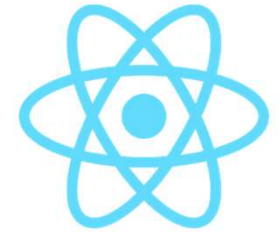
But: no Hooks in classes!

- Use a local `state` object and the method `.setState({...})`

```
class App extends Component {  
  
  state = {  
    counter: 0  
  };  
  
  incrementCounter(val) {  
    const newCounter = this.state.counter + val;  
    this.setState({  
      counter: newCounter  
    });  
  }  
}
```

Update state via
`this.setState(...)`

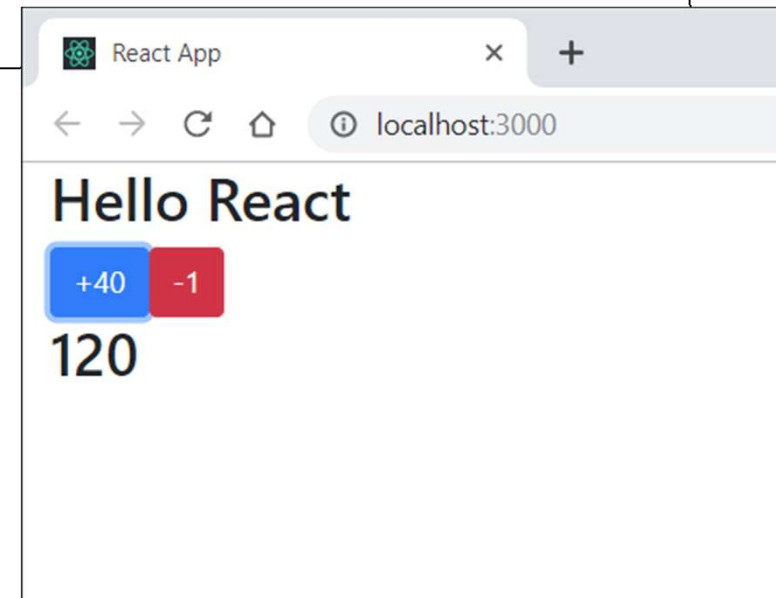
Use arrow function in render



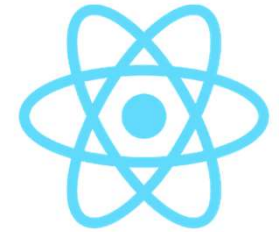
```
render() {  
  return (<div className="container">  
    <h2>Hello React</h2>  
    <Counter increment={() => this.incrementCounter(40)} val={40}/>  
    <DisplayCounter counter={this.state.counter}/>  
  </div>  
)  
};
```


Use the keyword `this` whenever you access state or instance methods

More info: <https://reactjs.org/docs/faq-functions.html>



Passing functions to components



 **React**

Docs Tutorial Blog Community

🔍 Search

v16.10.2 🌐 Languages GitHub

Passing Functions to Components

How do I pass an event handler (like `onClick`) to a component?

Pass event handlers and other functions as props to child components:

```
<button onClick={this.handleClick}>
```

If you need to have access to the parent component in the handler, you also need to bind the function to the component instance (see below).

How do I bind a function to a component instance?

There are several ways to make sure functions have access to component attributes like `this.props` and `this.state`, depending on which syntax and build steps you are using.

INSTALLATION ▾

MAIN CONCEPTS ▾

ADVANCED GUIDES ▾

API REFERENCE ▾

HOOKS ▾

TESTING ▾

CONCURRENT MODE (EXPERIMENTAL) ▾

CONTRIBUTING ▾

FAQ ^

AJAX and APIs

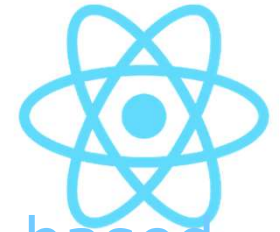
Babel, JSX, and Build Steps

Passing Functions to Components

Component State

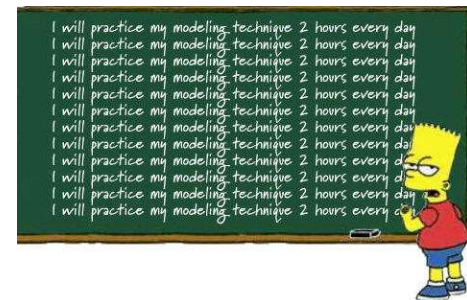
Styling and CSS

File Structure

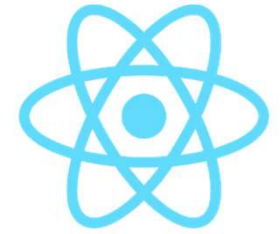


Workshop

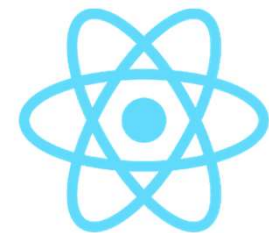
- Rewrite your previous application to use `class-based` components instead of function components
- OR: start with the sample application, and implement the `decrement` function
- Also pass the `className` for the button as a prop
(`btn-primary`, `btn-success`, `btn-info`,...)
- If necessary - update `<App />` component
- Example [../140-class-components](#)



Checkpoint



- You know about **class based** components
- You know how to define and access **state** in a class based component
- You know how to pass **props** and **functions** down to child components in a class-based component



Verdict 2022

classes are used less and less often.

Focus is now on React Hooks. But

you still need to know and learn

them!