

Thinking in React

Understand React's Basic Concepts and Gain Hands-On Experience

Walter Wallner | relwiwa.io

Motivation

- Cooperation between Eddisrupt and FreeCodeCampLisbon
- Get you started with React
- Convince you that React is not too advanced for you, especially with create-react-app client
- Encourage you to invest time discovering React or another frontend library or framework instead of dealing with jQuery
- Follow good coding practices as soon as possible to stop producing non-DRY "spaghetti code"

Today's agenda

- Understanding (60 minutes)
 - Explanation of only a few of React's basic concepts
 - Best practices workflow of building React apps: „Thinking in React“
- Hands-On Practice (90 minutes)
 - You will build a Markdown Previewer, the first React challenge in FCC
 - Pair programming, a more experienced with a less experienced coder
 - We use create-react-app client, so no worries about configuration
- Reflection (15 minutes)

Walter Wallner

- Started Web Development in the late 1990s
- Self-taught from HTML and CSS to Javascript; then PHP and Mysql and other server-side languages
- Cultural Studies and Computer Science
- Plan was to become a Developer, but became a Flight Attendant
- Got back into development via FreeCodeCamp
- Finished the Frontend Certificate, almost done with Data Viz and API certificate
- relwiwa.io is my React-based portfolio, that features most FCC projects I've done, live

Understanding React's basic concepts

1. Why Single Page Applications?

Traditional vs. Single Page Applications

2. How does React render HTML on the client?

JSX: Javascript Syntax eXtension

3. `<MarkdownPreviewer />`

React Components and Javascript Modules

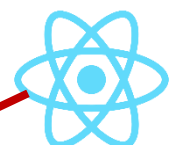
4. How does React update the UI and keep it in sync?

state, immutability

1. Why single page applications?

```
<!doctype html>
<html>
<head class="no-js">
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta http-equiv="x-ua-compatible" content="ie=edge">
  <title>relwiwa's Portfolio | freeCodeCamp Certification</title>
</head>
<body>
  <div id='root'></div>
  <script type="text/javascript" src="/build/index.77b404961eaed09578e5.js"></script></body>
</html>
```

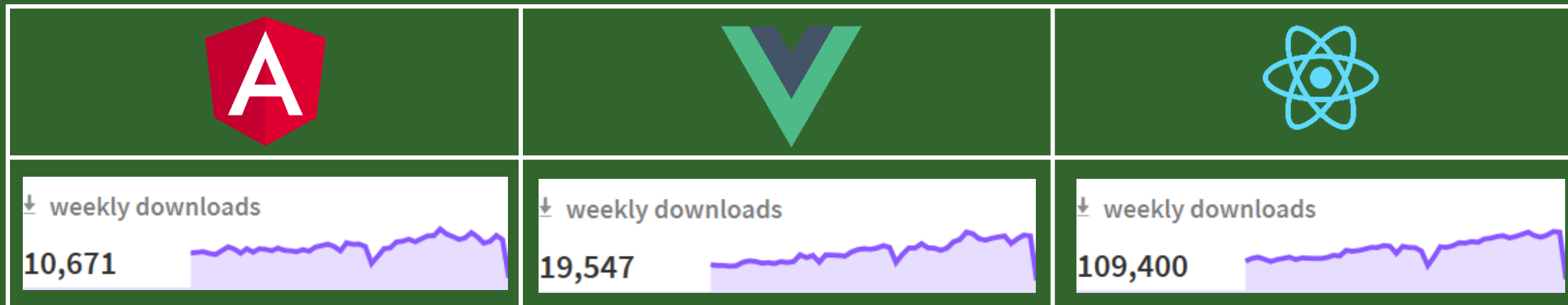
Initial HTML



1. Single page applications, pros and cons

	Traditional Web Apps	Single Page Apps
HTML produced on	server, directly, or via server-side language like PHP	client, via Javascript
Requests	always full HTML page	initial large, then small
Server Workload	big	Small
Traffic	big	Small
SEO	+	- (ServerSideReact)
User Experience	Page Loads, slow	reactive, super-fast

Why React? Compare NPM downloads



2. JSX is how React renders HTML on the client

```
const ExampleComponent = function () {  
  return (  
    <div>  
      <h1>Example Component</h1>  
      <p>Lorem ipsum dolor amet</p>  
    </div>  
  );  
};
```

With Javascript Syntax eXtension (JSX) you can use HTML Tags within Javascript!

Simple as that 😊

2. JSX Specifics

- JSX supports all HTML elements
- Some HTML attributes are renamed, because they are part of Javascript syntax:
 - className instead of class
 - htmlFor instead of for (label element)

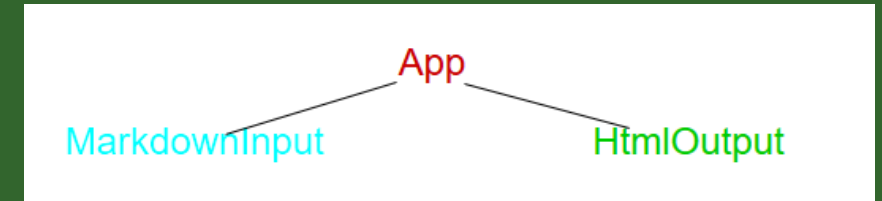
- To access Javascript variables within JSX, you use curly braces:

```
const myComponent = function() {  
  var someVariable = "my Text";  
  return (  
    <p>{someVariable}</p>  
  );  
}
```

- React components are also included via JSX:
 - <MarkdownPreviewer />

3. React components and Javascript modules

- React is based on components
- React creates a hierarchy of components, similar to the DOM
- Every component is a Javascript module
- Every component is in a separate file



Pros:

- + no trouble with global scope
- + good code organization, hard to spaghetti code
- + reusability of components, DRY

Cons:

- Javascript modules are not natively supported in browser, so complex configuration is necessary

3. global scope trouble

fileOne.js:

```
var incredible = "blue";
```

fileTwo.js:

```
var incredible = "red";
```

```
<script src="fileOne.js"></script>
```

```
<script src="fileTwo.js"></script>
```

Both files have a variable incredible, they are both on the global scope. fileTwo's variable will overwrite fileOne's variable

3. global scope trouble

fileOne.js:

```
var incredible = "blue";  
export default incredible;
```

fileTwo.js:

```
var incredible = "red";  
export default incredible;
```

app.js:

```
import incredible from './fileOne.js';  
import incredible as incredible2 from './fileTwo.js';
```

With importing and exporting from modules, we can overcome the trouble of using global scope

3. React components

```
import React from 'react';

const ExampleComponent = function () {
  return (
    <div>
      <h1>Example Component</h1>
      <p>Lorem ipsum dolor amet</p>
    </div>
  );
};

export default ExampleComponent;
```

```
import React from 'react';

import ExampleComponent from './ExampleComponent';

const App = function () {
  return (
    <div>
      <ExampleComponent />
    </div>
  );
};

export default App;
```

4. How does React update the UI?

- There are functional and class-based components in React
- Functional components are "dumb": They only display the UI
- Class-based components are "smart", because they have state
- state represents all the data you need to represent your UI and manage your components
- You should try to have as little components with state as possible
- You should always try to lift state up as much as possible in the component hierarchy

4. How to use state

- state is a regular Javascript object with key-value pairs

- It is defined in the constructor of a class-based component:

```
class exampleComponent extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = {  
      numbers: []  
    };  
  }  
}
```

- Whenever you want to update one or more state values, you have to use the **this.setState()** **method**:

```
this.setState({ numbers: [100, 12, 34, 45] });
```

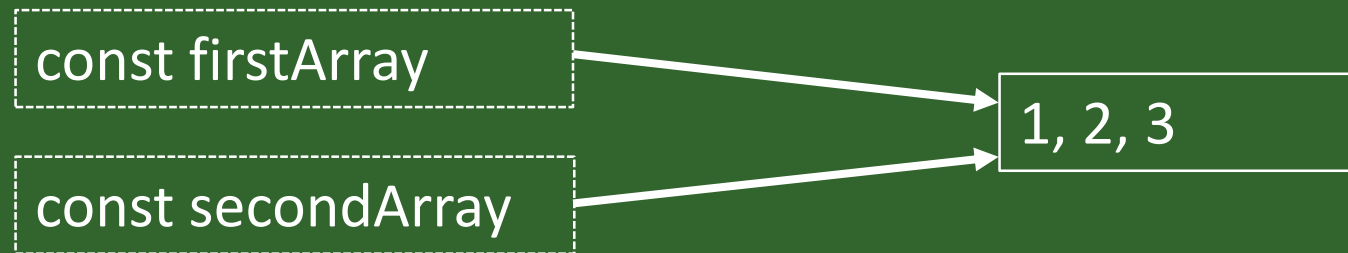
- React will then realize the values that have changed, and start updating all the components affected by the changes
- React uses a virtual DOM to manage components, and only updates the real DOM upon changes

4. Never mutate state

- Whenever you want to update values in state, you use `this.setState()`
- You have to be careful when updating arrays and objects inside state
- In order to find out about the changes, React does not check every entry inside of arrays or objects
- Instead, it only checks, whether the reference of the array or object has changed
- Only if the reference has changed, it looks for the changes inside of the array or object
- Only then React updates the affected components, very efficient

4. Array references

- `const firstArray = [1, 2, 3];`
`const secondArray = firstArray;`
`secondArray[0] = 5;`
`console.log(secondArray[0], firstArray[0]);`
- When assigning arrays to variables, only references to the existing array are assigned, no new array is created!



4. How to update arrays without mutations

- When you want to update an array inside of state, you always have to create a new array
- Never use array-methods that change the array without creating a new one (pop, push, shift, unshift)
- You can create a new array by using the concat method:
`const secondArray = [].concat(firstArray);`
- Or you can use the ES6-Spread Operator:
`const secondArray = [...firstArray];`
- There is also a Spread-Operator for objects