Typeform

React Workshop

Agenda

0. The theory
- History of frameworks
- Why React

0.5. From scratch
- Add React to index.html
- JSX basics

1. Create React App
- Setup environment
- Simple modification
- Structure of the files
- Modules

2. More JSX & Props
- JSX and HTML
- Props

3. Interactions
- Listeners
- State

4. Go deeper!
- Data from API
- Data manipulation
- Go functional

T

Why React?

Story of JS

T


HTML & CSS
design and build websites
JON DUCKETT
WILEY
JAVASCRIPT & JQUERY
JON DUCKETT
WILEY

JS Problems

JS Problems

```
if (window.XMLHttpRequest) { // Mozilla, Safari, IE7+ ...
    httpRequest = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE 6 and older
    httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
}
```

```
function addEvent(evnt, elem, func) {
   if (elem.addEventListener)  // W3C DOM
      elem.addEventListener(evnt,func,false);
   else if (elem.attachEvent) { // IE DOM
      elem.attachEvent("on"+evnt, func);
   }
   else { // No much to do
      elem[evnt] = func;
   }
}
```

- **Cross browser API differences**
- Difficult DOM manipulation
- “Bad” parts
- Unpredictable design patterns, easy spaghetti code

JS Problems

```
if (window.XMLHttpRequest) { // Mozilla, Safari, IE7+ ...
    httpRequest = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE 6 and older
    httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
}
```

- **Browser API differences - ✓ jQuery**
- Difficult DOM manipulation
- “Bad” parts
- Unpredictable design patterns, easy spaghetti code

```
function addEvent(evnt, elem, func) {
   if (elem.addEventListener)  // W3C DOM
      elem.addEventListener(evnt,func,false);
   else if (elem.attachEvent) { // IE DOM
      elem.attachEvent("on"+evnt, func);
   }
   else { // No much to do
      elem[evnt] = func;
   }
}
```

```
parent.on( evType, el, function);
```

```
1 function addClassF(el, cls) {
2   var clss
3   if (typeof cls === "string") {
4     clss = cls.split(" ")
5   } else if (cls instanceof Array) {
6     clss = cls
7   }
8   var i = 0,
9       len = clss.length;
10  for (; i < len; i++) {
11    if (el.classList) {
12      el.classList.add(clss[i])
13    } else {
14      // <= IE8
15      el.className += ' ' + clss[i]
16    }
17  }
18
19 }
20
21 var foo = document.getElementById( "foo" )
22 addClassF(foo, "class1 class2")
```

JS Problems

- Browser API differences
- **Difficult DOM manipulation**
- “Bad” parts
- Unpredictable design patterns, easy spaghetti code

JS Problems

```
1 function addClassF(el, cls) {
2   var clss
3   if (typeof cls === "string") {
4     clss = cls.split(" ")
5   } else if (cls instanceof Array) {
6     clss = cls
7   }
8   var i = 0,
9       len = clss.length;
10  for (; i < len; i++) {
11    if (el.classList) {
12      el.classList.add(clss[i])
13    } else {
14      // <= IE8
15      el.className += ' ' + clss[i]
16    }
17  }
18
19 }
20
21 var foo = document.getElementById( "foo" )
22 addClassF(foo, "class1 class2")
```

- Browser API differences
- **Difficult DOM manipulation - ✓ jQuery**
- “Bad” parts
- Unpredictable design patterns, easy spaghetti code

```
1 var $foo = $("#foo")
2 $foo.addClass("class1 class2")
```

```
1 a = 5
2 if (a == 5){
3   console.log("Hello")
4 }
```

JS Problems

- Browser API differences
- Difficult DOM manipulation
- **“Bad” parts (globals, eval, this, …)**
- Unpredictable design patterns, easy spaghetti code

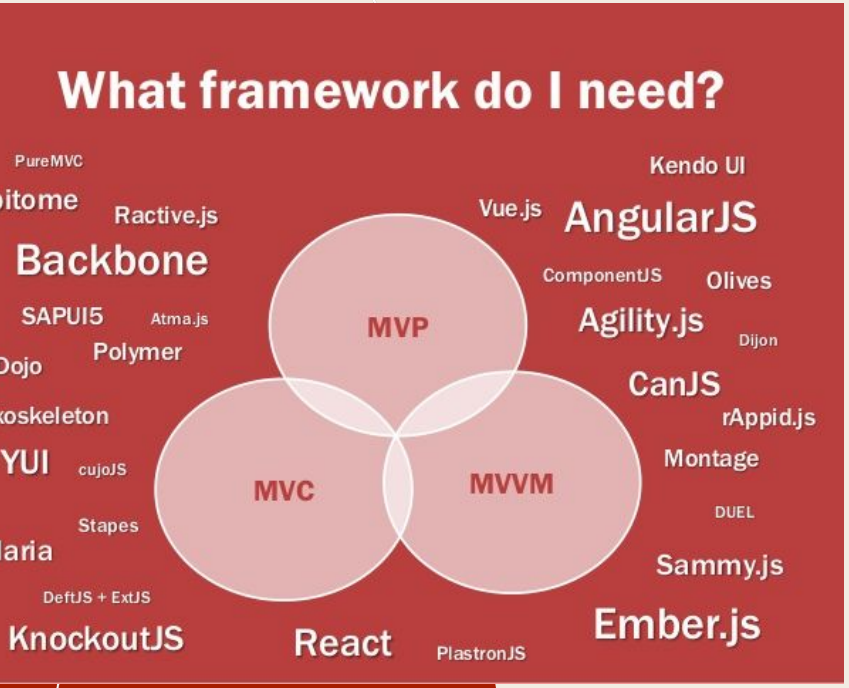
JS Problems

```
1 a = 5
2 if (a == 5){
3   console.log("Hello")
4 }
```

- Browser API differences
- Difficult DOM manipulation
- **“Bad” parts (globals, eval, this, …) -**
  - ✓ **linters & ES6 & strict mode**
- Unpredictable design patterns, easy spaghetti code

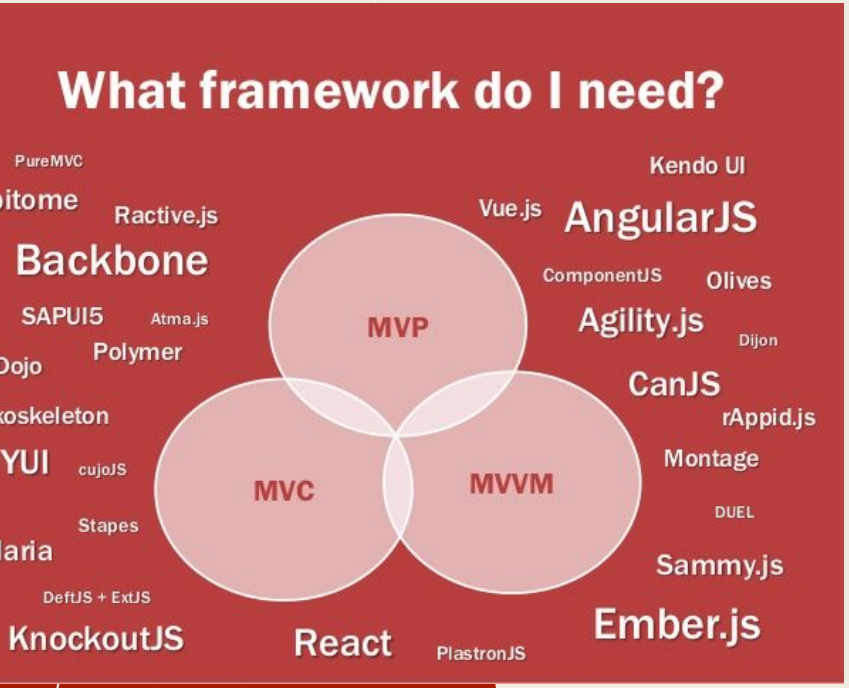
```
1 'a' was used before it was defined.
2 Expected ';' and instead saw 'if'.
3 Expected '===' and instead saw '==='.
4 Expected exactly one space between ')' and '{'.
5 Expected 'console' at column 5, not column 3.
6 Expected ';' and instead saw '}'
```

T



JS Problems

- Browser API differences
- Difficult DOM manipulation
- “Bad” parts (globals, eval, this, …)
- **Easy to write spaghetti code**



JS Problems

- Browser API differences
- Difficult DOM manipulation
- “Bad” parts (globals, eval, this, …)
- **Easy to write spaghetti code - Frameworks and Libraries (Angular, Backbone, Ember, React)**

React

Why React?

Advantages

REACT

- React is a library - responsible only for view layer

REACT

- React is a library - responsible only for view layer
- Has a very tiny API

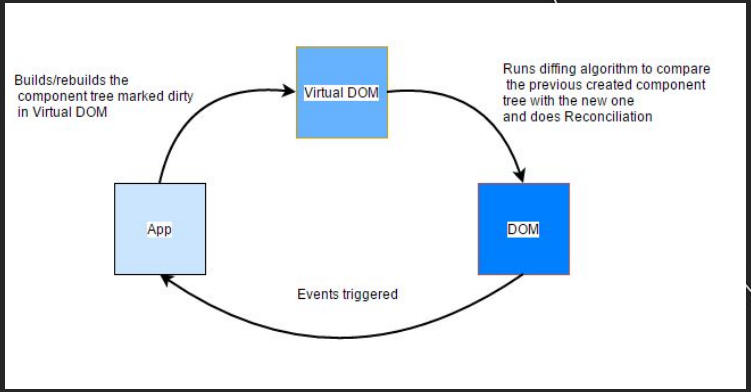
REACT

- React is a library - responsible only for view layer
- Has a very tiny API
- It easy to understand (DOM Manipulation is hell in JS!)

T

REACT

- React is a library - responsible only for view layer
- Has a very tiny API
- It easy to understand (DOM Manipulation is hell in JS!)
- It’s performant (virtual DOM, only updates when necessary)



From Scratch!

What’s JSX

```
var header
```

T

What’s JSX

```
var header
```

```
<h1> Hello </h1>
```

What’s JSX

```
var header
```

```
<h1> Hello </h1>
```

```
var header = <h1> Hello </h1>
```

T

What's JSX

- XML-like syntax extension of JavaScript.
- It means that you can basically write HTML in Javascript.

```
var header
```

```
<h1> Hello </h1>
```

```
var header = <h1> Hello </h1>
```

Let’s start

- Go to: https://github.com/mbondyra/ada-react-workshop
- Clone the repo in one of your folder
- Open from_scratch/index.html file in your IDE (VS Code etc)
- Follow the instructions in github readme file.
- Presentation: https://goo.gl/g7smwM
- Encoding problem? Add this to head:

```
<meta charset="UTF-8">
```

T

Checkpoint 0

- Outcome of the part: all the examples solved
- You should know now:
  - How to add React to index.html file
  - What's JSX
  - How to render JSX Element
  - How to execute javascript in JSX element
  - How to display value of variable
  - How to add listener to JSX element
- Once solved, go to the 1-the-letter branch.

T

Create React App
1. The letter

T

Files structure - HTML page

index.html

```html
<html>
  <head>
    <link href="./style" />
    <script scr="./script.src"></script>
  </head>
  <body />
</html>

```

style.css

```css
body {
  margin: 0;
  background: #e8e8e4;
  min-height: 100vh;
  color: white;
}

```

script.js

```js
document.addEventListener('click', function(){
    console.log("You clicked on page!')
})
```

Files structure - HTML page

index.html

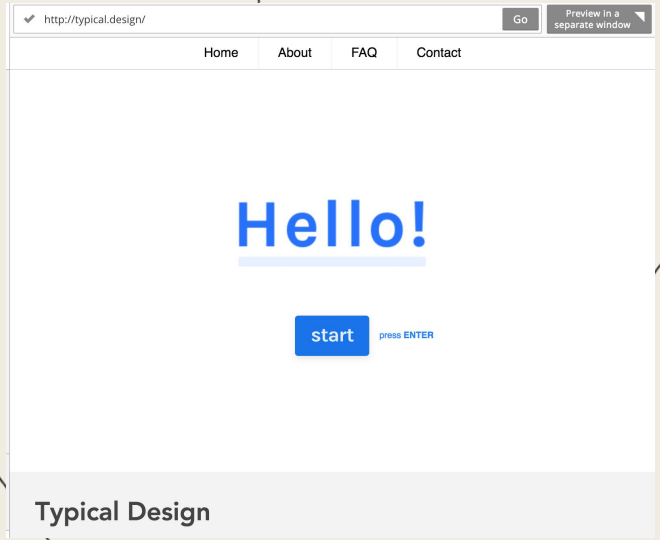
```html
<html>
  <head>
    <link href="./style" />
    <script scr="./script.src"></script>
  </head>
  <body />
</html>
```

style.css

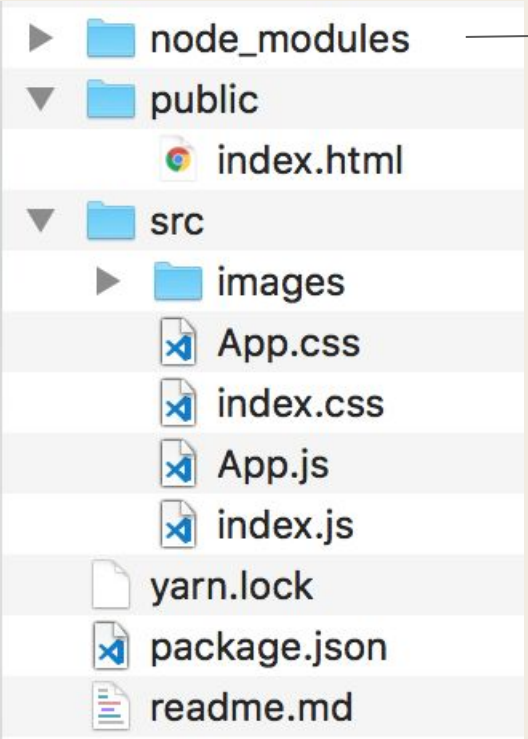
```css
body {
  margin: 0;
  background: #e8e8e4;
  min-height: 100vh;
  color: white;
}
```

script.js

```js
document.addEventListener('click', function(){
    console.log("You clicked on page!')
})
```



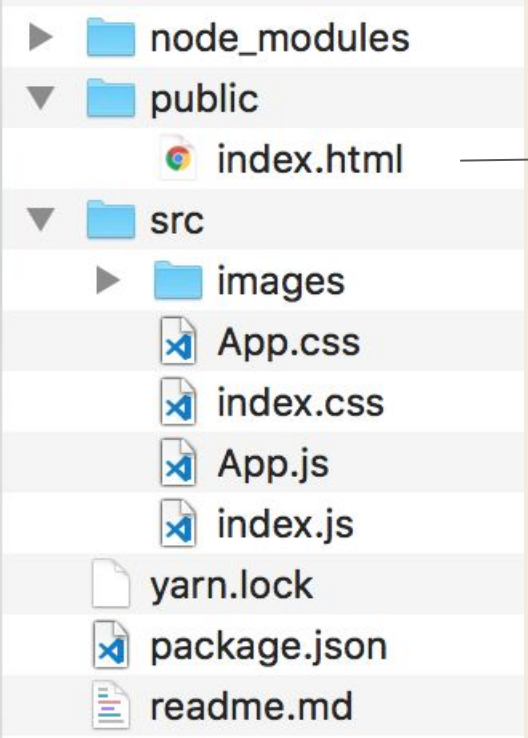
Files structure



External packages that we use in our project, e.g. React

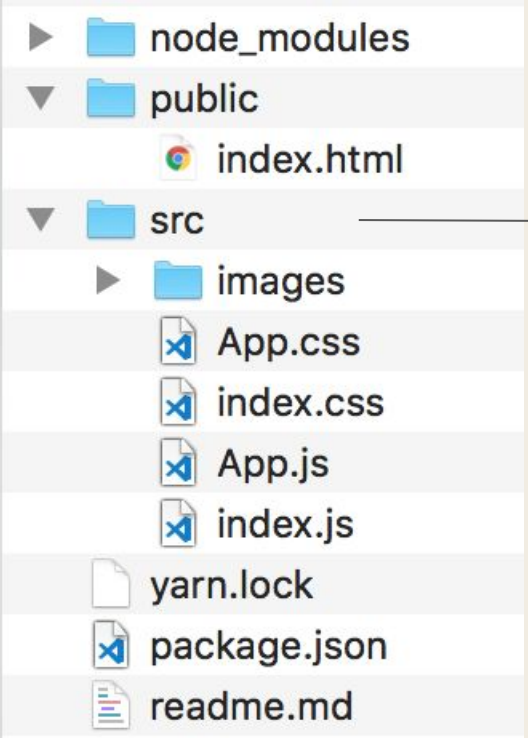
T

Files structure



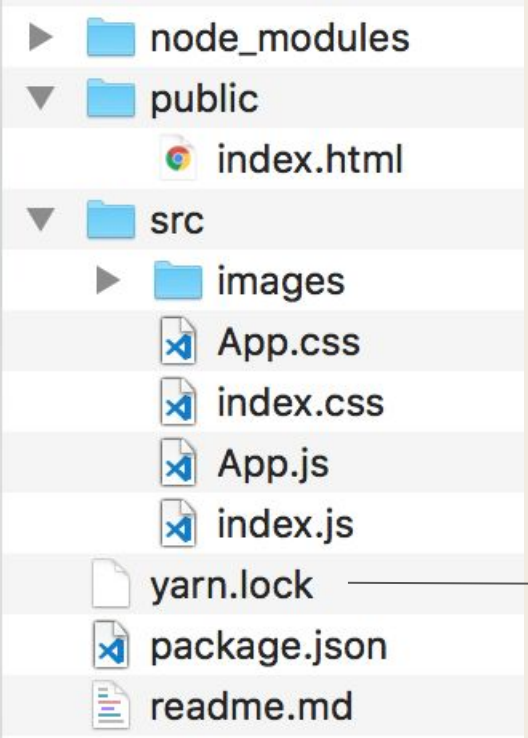
Index.html file
with the structure of html and <div id='root/>'

Files structure



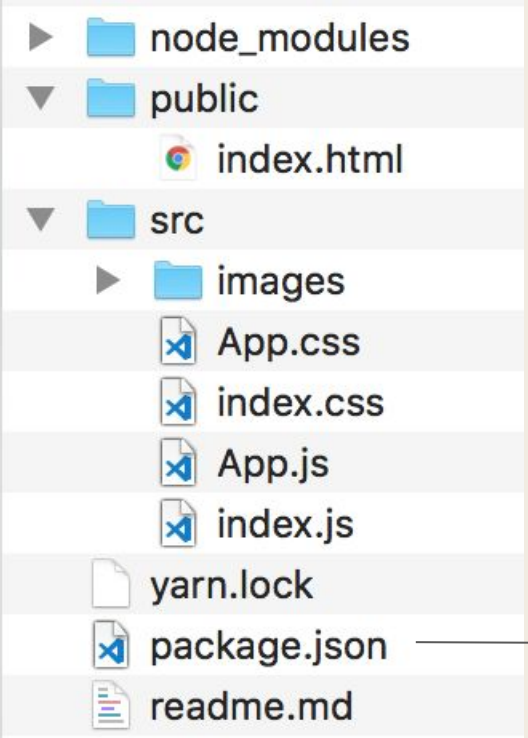
Basically the whole app (it is mounted in index.html <div id='root/>')

Files structure



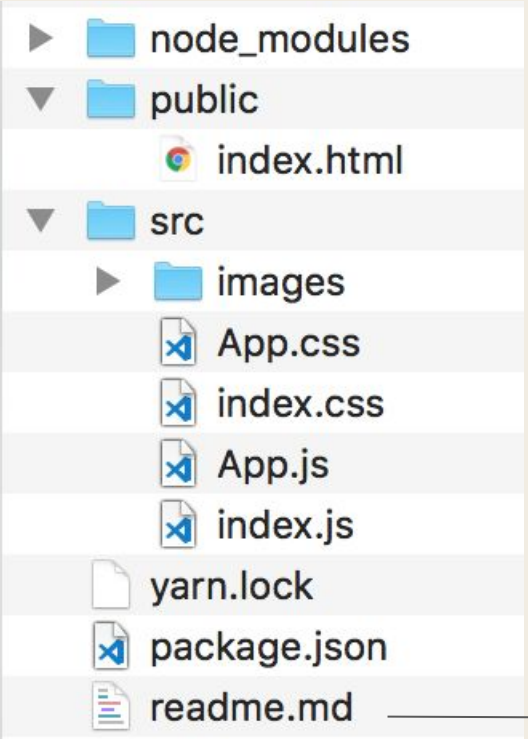
Saves current state of node modules

Files structure



Configuration for our app

Files structure



Instruction files

Files structure

- node_modules
- public
  - index.html
- src
  - images
  - App.css
  - index.css
  - App.js
  - index.js
- yarn.lock
- package.json
- readme.md

public/index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- metatags -->
    <title>React App</title>
  </head>
  <body>
    <noscript>
      You need to enable JavaScript to run this app.
    </noscript>
    <div id="root"></div>
  </body>
</html>
```

Files structure

- node_modules
- public
  - index.html
- src
  - images
  - App.css
  - index.css
  - App.js
  - index.js
- yarn.lock
- package.json
- readme.md

public/index.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <!-- metatags -->
5     <title>React App</title>
6   </head>
7   <body>
8     <noscript>
9       You need enable JavaScript to run this app.
10
```

Webpack adds index.js (no explicit import)

Code structure

src/index.js

```
import React from 'react' // import react
import ReactDOM from 'react-dom' // import react dom
import './index.css'    //import css module
import App from './App' // import main react element from file (JSX)

ReactDOM.render(
  <App />, /* react element to mount */
  document.getElementById('root') /* dom element to mount to */
)
```

Code structure

src/index.js

```
import React from 'react' // import react
import ReactDOM from 'react-dom' // import react dom
import './index.css'    //import css module
import App from './App' // import main react element from file (JSX)

ReactDOM.render(
  <App />, /* react element to mount */
  document.getElementById('root') // dom eleme
)
```

public/index.html

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

Code structure

src/index.js

```
/* ... */
ReactDOM.render(
  <App />, /* react element to mount */
  document.getElementById('root')
)
```

Code structure

src/index.js

```
/* ... */
ReactDOM.render(
  <App />,
  document.getElementBy
)
```

src/App.js

```
import React, { Component } from "react";
import reactLogo from "./images/react-logo.svg";
import adaLogo from "./images/ada-logo.jpeg";
import letter from "./images/letter.jpg";
import "./App.css";

// we'll be modifying code here:

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <h1>
            <img src={reactLogo} className="App-logo" alt="react-logo" />
            workshop with a bit of magic
            <img src={adaLogo} className="App-logo Ada-logo" alt="ada-logo" />
          </h1>
        </header>
        <div className="App-bg" />
      </div>
    );
  }
}

export default App;
```

T

Code structure

src/index.js

```
/* ... */
ReactDOM.render(
  <App />, /* react element
  document.getElementById('r
)
```

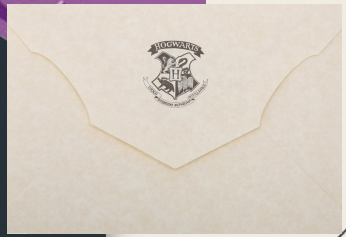
src/App.js

```
import React, { Component } from "react";
import reactLogo from "./images/react-logo.svg";
import adaLogo from "./images/ada-logo.jpeg";
import letter from "./images/letter.jpg";
import "./App.css";

// we'll be modifying code here:

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <h1>
            <img src={reactLogo} className="App-logo" alt="react-logo" />
            workshop with a bit of magic
            <img src={adaLogo} className="App-logo Ada-logo" alt="ada-logo" />
          </h1>
        </header>
        <div className="App-bg" />
      </div>
    );
  }
}

export default App;
```



Code structure

src/index.js

```
/* ... */
ReactDOM.render(
  <App />, /* react element
  document.getElementById('r
)
```

src/App.js

```
import React, { Component } from "react";
import reactLogo from "./images/react-logo.svg";
import adaLogo from "./images/ada-logo.jpeg";
import letter from "./images/letter.jpg";
import "./App.css";

// we'll be modifying code here:

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <h1>
            <img src={reactLogo} className="App-logo" alt="react-log
            workshop with a bit of magic
            <img src={adaLogo} className="App-logo Ada-logo" alt="ada-logo"
          </h1>
        </header>
        <div className="App-bg" />
      </div>
    );
  }
}

export default App;
```

src/App.css

```
/* and here we're modifying csses: */

.Ada-logo {
  margin-right: 10px;
  right: 0;
  position: absolute;
  top: 15px;
  filter: grayscale(100%);
}

.App-logo {
  height: 30px;
  margin: 0 0px 0 10px;
  vertical-align: middle;
  filter: grayscale(100%);
}
```

Go to the instruction :)

Checkpoint 1: Code structure

- Outcome of the part: secret message read
- You should know now:
  - How our app is structurized
  - Where to modify CSSes and content
  - Where React adds elements to DOM

- Switch to next branch: 1-and-3/4-the-letter-modules

T

Checkpoint 1 and 3/4: Modules

- Outcome of the part: Content module created
- You should know now:
  - How to create new component
  - How to import css module
  - How to logically divide your app
- Switch to the next branch: 2-elixirs-class

T

Part 2:

More JSX & Props

HTML vs JSX

index.html

```
<div id='elixir1'></div>
```

HTML vs JSX

index.html

```
<div id='elixir1'></div>
```

index.js

```
<div id='elixir1'></div>
```

HTML vs JSX

index.html

```
<div class='elixir2'></div>
```

T

HTML vs JSX

index.html

```
<div class='elixir2'></div>
```

index.js

```
<div className='elixir2'></div>
```

HTML vs JSX

index.html

```
<div style='color: black'></div>
```

HTML vs JSX

index.html

```
<div style='color: black'></div>
```

index.js

```
<div style={{color: 'black'}}></div>
```

HTML vs JSX

index.html

```
<div class='elixir2'></div>
```

index.js

```
<div className='elixir2'></div>
```

index.html

```
<div style='color: black'></div>
```

index.js

```
<div style={{color: 'black'}}></div>
```

Props

```
<Welcome who='Ada' messageColor='#fff' titleClassName='blue' withEmoji />
```

```
...
<div className='message' style={{ color: this.props.messageColor }}>
          This is your first elixirs lesson, {this.props.who}
...
```

```
//display if prop exists
this.props.withEmoji && <span> 😀 </span>
```

```
//Add class if it was passed
render() {
    let titleClassName = 'title'
    if (this.props.titleClassName) {
      titleClassName = 'title ' + this.props.titleClassName
    }

  ...
  <div className={titleClassName}>Elixirs' class</div>
```

T

Solution - file Potion/index.js

1. Move Content of one potion to Potion file:

```
class Potion extends Component {
 render() {return (
     <div className='potion'>
       <img src={bottle} alt='bottle' />
       <div className='liquid' />
       <div className='label'>id: veritaserum</div>
     </div>
   )}}
```

2. Make it work! - pass prop label:

```
<div className='label'>~~id: veritaserum~~{this.props.label}</div>
```

3. If a prop className was passed, add it:

```
let className = 'liquid'
 if (this.props.className){
   className = 'liquid ' + this.props.className
}
```

4. Use it for a div that was before a liquid

```
 <img src={bottle} alt='bottle' />
 <div className={className}/>
```

T

Solution

1. Import Potion file to Content.js:

```
import Potion from '../Potion/index'
```

2. Create three instances of <Potion/>

```
<Potion label='veritaserum' color='gold'/>
<Potion label='felis-felisis' color='pink'/>
<Potion label='polyjuice' color='grey'/>
```

Checkpoint 2

- Outcome of the part: having a component Potion with set of props. Modifying Welcome component.
- You should know now:
  - How to modify styles with class, id and inline styles.
  - How to pass props to object

- Switch to the next branch: 3-charms-class

T

Part 3

Interactions

Assigning function to a listener

index.js

```
sayHello(){
    alert('hello')
}
...
<div onClick={function(){alert('hello')}    //standard syntax
<div onClick={()=>{alert('hello')}}         //ES6 syntax
<div onClick={this.sayHello}                //assigning to a function of the object
```

State

- State of your bank account - empty or full
- State of a man (things that change):

```
this.state= {
   hungry: false,
   sick: false,
   age: 28,
   mood: ANGRY
}
```

- State of the door: open or closed

State

the outcome of all of the actions

the user has taken since the page loaded

T

Stateless component

```
class Wand extends Component {
  render(){
    return (
      <div className='wandContainer'>
        <img
            src={wand}
            className={this.props.raised}
            alt='wand' />
        <Confetti
            active={this.props.spellCasting}
            config={confettiConfig} />
      </div>
    )
  }
}
```

Stateless component

```
class Wand extends Component {
  render(){
    return (
      <div className='wandContainer'>
        <img
            src={wand}
            className={this.props.raised}
            alt='wand' />
        <Confetti
            active={this.props.spellCasting}
            config={confettiConfig} />
      </div>
    )
  }
}
```

```
const Wand = ({raised, spellCasting}) => (
  <div className='wandContainer'>
    <img
        src={wand}
        className={raised}
        alt='wand' />
    <Confetti
        active={spellCasting}
        config={confettiConfig} />
  </div>
)
```

T

Stateless component

```
class Wand extends Component {
  render(){
    return (
      <div className='wandContainer'>
        <img
            src={wand}
            className={this.props.raised}
            alt='wand' />
        <Confetti
            active={this.props.spellCasting}
            config={confettiConfig} />
      </div>
    )
  }
}
```

```
const Wand = ({raised, spellCasting}) => (
  <div className='wandContainer'>
    <img
        src={wand}
        className={raised}
        alt='wand' />
    <Confetti
        active={spellCasting}
        config={confettiConfig} />
  </div>
)
```

T

Checkpoint 3

- Door open and close on wand click, Wand is going up and down, confetti gets crazy.
- You should know:
  - What’s a state and how to add state to your component
  - How to pass a state as a prop
  - How to trigger events in react
  - How to create stateless component
- Once you’re ready, go to branch 4-advanced-charms

T

Part 4

Let’s go deeper!

Interested in learning more? Here are subjects we haven't covered:

- React lifecycle
- Performance tweaks
- Styled components (not react but very popular now)
- React dev tools

Resources I recommend

- https://reactjs.org/tutorial/tutorial.html
- https://egghead.io/courses/the-beginner-s-guide-to-react
- https://medium.freecodecamp.org/learn-react-js-in-5-minutes-526472d292f4

Typeform

Thank you!

Marta Bondyra

Roast me ;)

https://mbondyra.typeform.com/to/UaFFrC

marta.bondyra@gmail.com