Day 2 - Getting Started with React

What is React?

React is a JavaScript library for building a reusable user interface (UI). React makes creating UI components very easy. When we work with React, we don't interact directly with the DOM. Instead, React has its own way to handle DOM manipulation. React uses its virtual DOM to make changes and updates only the element that needs changing. When building a React application, don't directly interact with the DOM so as to not interfere.

Summary

- React was released in May 2013
- React was created by Facebook
- React is a JavaScript library for building user interfaces
- React is used to build single page applications - An application that has only one HTML page.
- React allows us to create reusable UI components.
- Reacts latest release is 18.2.0

Why React?

- Fast
- Modular
- Scalable
- Flexible
- Big community and popular
- Open source
- High job opportunity

JSX

JSX stands for JavaScript XML. It allows us to write HTML elements with JavaScript code. An HTML element has opening and closing tags, content, and attributes in the opening tag. However, some HTML elements may not have content and a closing tag - self closing elements. To create HTML elements in React, we don't use the createElement() method. Instead, we just use JSX elements. Therefore, JSX makes it easier to write and add HTML elements in React. JSX can be converted to

JavaScript on the browser using a transpiler - babel.js. Babel is a library that transpiles JSX to pure JSX to pure JavaScript and latest JavaScript to an older version.

```
// JSX syntax
// We don't need to use quote with JSX

const jsxElement = <h1>I am a JSX element</h1>
const welcome = <h1>Welcome to 30 Days of React Challenge</h1>
const data = <small>Oct 2, 2020</small>
```

In the examples above, the HTML elements are h1 and small.

JSX Element

JSX has a JavaScript and HTML like syntax. A JSX element could be a single HTML element or it may be many HTML elements wrapped in a parent HTML element.

```
const jsxElement = <h1>I am a JSX element</h1> // JS with HTML
```

```
const title = <h2>Getting Started with React</h2>
```

Every HTML element should be wrapped by an outer HTML element to create a valid JSX element. The title variable should also be changed to header because our JSX element contains almost all of the header of the application.

```
const header = (
  <header>
    <h1>Welcome to 30 Days of React</h1>
    <h2>Getting Started with React</h2>
    <h3>JavaScript Library</h3>
  </header>
)
```

Let’s add more elements to display the author name and year.

```
const header = (
  <header>
    <h1>Welcome to 30 Days Of React</h1>
    <h2>Getting Started React</h2>
    <h3>JavaScript Library</h3>
    <p>Christopher Guerrero</p>
    <small>Mar 3, 2023</small>
  </header>
)
```

Commenting a JSX element

```
{
  /*
 <header>
    <h1>Welcome to 30 Days Of React</h1>
    <h2>Getting Started React</h2>
    <h3>JavaScript Library</h3>
    <p>Asabeneh Yetayeh</p>
    <small>Oct 2, 2020</small>
  </header>

*/
}
```

Rendering a JSX Element

To render a JSX element to an HTML document, we should first create an index.html file. This is the only HTML file we'll have in any React Application.

We can get started with React in one of two ways - either using the CDN or create-react-app. Create-react-app creats a React project boilerplate outbox and because of that, many people have a hard time understanding how react works. To make things clear, let's start with a CDN. For just this section, we'll use the CDN, but after, we'll use create-react-app.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>30 Days Of React Challenge</title>
  </head>

  <body>
    <div class="root"></div>

    <script></script>
  </body>
</html>
```

Above, we have one div with a class root and script. The root div is the gateway to connect all react components to our index.html. In the script tag, we'll write our JavaScript, but the script *type* will be *babel*. Babel will transpile the react JSX to pure JavaScript on the browser. Inside babel, we can write any pure JavaScript, JSX and in general any React code.

```
<!DOCTYPE html>
<html lang="en">
  <head>
```

```
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>30 Days Of React Challenge</title>
  </head>

  <body>
    <div class="root"></div>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
    <script type="text/babel">
       // our JS, JSX, and React code in general goes here
    </script>
  </body>
</html>
```

The babel library is linked to our document above. The next step is importing React and ReactDOM using CDN or link. In order to link React and ReactDOM, we attach both packages from the CDN to the body of index.html. To test if React is linked to the index.html, we can try to check it by doing console.log(React). If we open the browser console, we should get an object.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>30 Days Of React Challenge</title>
  </head>

  <body>
    <div class="root"></div>

    <script
      crossorigin
      src="https://unpkg.com/react@16/umd/react.development.js"
    ></script>
    <script
      crossorigin
      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
    ></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
    <script type="text/babel">
      console.log(React)
    </script>
  </body>
</html>
```

We now have everything we need to write React code. We can get the root element using document.querySelector('.root') and assign it to the variable rootElement. This is the only place we directly interact with the DOM.

Now that we know about JSX and JSX elements, we can render a JSX element to the browser. To do so, we need the React and ReactDOM library. In addition to the React and ReactDOM library, we

need babel to transpile the JSX to JavaScript code. The ReactDOM package has a render() method. It takes two arguments; a JSX element or component and the root document.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>30 Days Of React Challenge</title>
  </head>

  <body>
    <div class="root"></div>

    <script
      crossorigin
      src="https://unpkg.com/react@16/umd/react.development.js"
    ></script>
    <script
      crossorigin
      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
    ></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
    <script type="text/babel">
      // To get the root element from the HTML document
      const rootElement = document.querySelector('.root')

      // JSX element
      const jsxElement = <h1>I am a JSX element</h1>

      // we render the JSX element using the ReactDOM package
      // ReactDOM has the render method and the render method takes two arguments
      ReactDOM.render(jsxElement, rootElement)
    </script>
  </body>
</html>
```

Let’s render more content. To render more content, the JSX element should have more HTML elements. For instance, we can create a header of a website, which may have a title, subtitle, author or date etc. Remember, we can render only one JSX element at a time.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>30 Days Of React Challenge</title>
  </head>

  <body>
    <div class="root"></div>

    <script
      crossorigin
      src="https://unpkg.com/react@16/umd/react.development.js"
    ></script>
```

```
    <script
      crossorigin
      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
    ></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
    <script type="text/babel">
      // To get the root element from the HTML document
      const rootElement = document.querySelector('.root')

      // JSX element
      const header = (
        <header>
          <h1>Welcome to 30 Days Of React</h1>
          <h2>Getting Started React</h2>
          <h3>JavaScript Library</h3>
          <p>Christopher Guerrero</p>
          <small>Mar 3, 2023</small>
        </header>
      )

      // we render the JSX element using the ReactDOM package
      // ReactDOM has the render method and the render method takes two arguments
      ReactDOM.render(header, rootElement)
    </script>
  </body>
</html>
```

What about the main part and footer for our website? Similar to the header, let's create a JSX element for the main and the footer.

```
// JSX element
const main = (
  <main>
    <p>Prerequisite to get started react.js:</p>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
  </main>
)
```

```
// JSX element
const footer = (
  <footer>
    <p>Copyright 2020</p>
  </footer>
)
```

We now have three JSX elements; the header, main and footer. The best way to render all three of the JSX elements is by wrapping them all in a parent JSX element or putting them in an array. To include a JSX element inside another JSX element, we can use the curly bracket and call the name of the JSX inside the curly bracket.

```
// JSX element for the header part of the website
const header = (
  <header>
    <h1>Welcome to 30 Days Of React</h1>
    <h2>Getting Started React</h2>
    <h3>JavaScript Library</h3>
    <p>Asabeneh Yetayeh</p>
    <small>Oct 2, 2020</small>
  </header>
)

// JSX element for the main part of the website
const main = (
  <main>
    <p>Prerequisite to get started react.js:</p>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
  </main>
)

// JSX element for the footer part of the website
const footer = (
  <footer>
    <p>Copyright 2020</p>
  </footer>
)

// JSX element which contain all, it is a container or parent
const app = (
  <div>
    {header}
    {main}
    {footer}
  </div>
)
```

Style and className in JSX

After the emergence of react, inline styling of element became very popular. To add style to a JSX element, we use inline style or className. We inject the style object using {}. Every CSS property becomes a key and every CSS property value becomes a value for the object. For instance, in the example below, border is a key and ‘2px solid orange’ is a value, color is a key and ‘black’ is a value, fontSize is a key and ‘18px’ is a value. All two word CSS properties will change to camelCase when we use them as a key in the CSS object in React or JavaScript

```
const header = (
  <header
    style={{ border: '2px solid orange', color: 'black', fontSize: '18px' }}
  >
    <h1>Welcome to 30 Days Of React</h1>
    <h2>Getting Started React</h2>
    <h3>JavaScript Library</h3>
```

```
    <p>Asabeneh Yetayeh</p>
    <small>Oct 2, 2020</small>
  </header>
)

// or we can write it this way

const style = { border: '2px solid orange', color: 'black', fontSize: '18px' }

const header = (
  <header style={style}>
    <h1>Welcome to 30 Days Of React</h1>
    <h2>Getting Started React</h2>
    <h3>JavaScript Library</h3>
    <p>Asabeneh Yetayeh</p>
    <small>Oct 2, 2020</small>
  </header>
)
```

It’s good practice to open the browser console while developing an application to know if everything is going well.

We can use regular internal styling to style our application. Using regular style, to target an HTML element, we use tag name, id, class, an attribute and other methods. In the React community, people use classes quite a lot instead of id’s.

In JSX elements, we use className instead of class because class is a reserved word in JavaScript. Similar to className, htmlFor is used instead of for in a label tag.

```
const title = <h1 className='title'>Getting Started with React </h1>
const inputField = (
  <div>
    <label htmlFor='firstname'>First Name</label>
    <input type='text' id='firstname' placeholder='First Name' />
  </div>
)
```

The id used in the input element isn’t for styling purposes, and instead refers to the label to the input field. Using class instead of className or for instead of htmlFor will lead to errors in the console.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>30 Days Of React Challenge</title>
  </head>

  <body>
    <div class="root"></div>
```

```
<script
  crossorigin
  src="https://unpkg.com/react@16/umd/react.development.js"
></script>
<script
  crossorigin
  src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
></script>
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
<script type="text/babel">
  // To get the root element from the HTML document
  const rootElement = document.querySelector('.root')

  // style
  const headerStyles = {
    backgroundColor: '#61DBFB',
    fontFamily: 'Helvetica Neue',
    padding: 25,
    lineHeight: 1.5,
  }

  // JSX element, header
  const header = (
    <header style={headerStyles}>
      <div className='header-wrapper'>
        <h1>Welcome to 30 Days Of React</h1>
        <h2>Getting Started React</h2>
        <h3>JavaScript Library</h3>
        <p>Asabeneh Yetayeh</p>
        <small>Oct 2, 2020</small>
      </div>
    </header>
  )

  // JSX element, main
  const mainStyles = {
    backgroundColor: '#F3F0F5',
  }
  const main = (
    <main style={mainStyles}>
      <p>Prerequisite to get started react.js:</p>
      <ul>
        <li>HTML</li>
        <li>CSS</li>
        <li>JavaScript</li>
      </ul>
    </main>
  )

  const footerStyles = {
    backgroundColor: '#61DBFB',
  }
  // JSX element, footer
  const footer = (
    <footer style={footerStyles}>
      <p>Copyright 2020</p>
    </footer>
  )

  // JSX element, app
  const app = (
```

```
        <div className='app'>
          {header}
          {main}
          {footer}
        </div>
      )

      // we render the JSX element using the ReactDOM package
      ReactDOM.render(app, rootElement)
    </script>
  </body>
</html>
```

Below, we'll apply internal styling to style all the JSX.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      href="https://fonts.googleapis.com/css?family=Montserrat:300,400,500|Roboto:300,400,500&display=swap"
      rel="stylesheet"
    />

    <title>30 Days Of React Challenge</title>
    <style>
      /* == General style === */
      * {
        box-sizing: border-box;
        padding: 0;
        margin: 0;
      }

      html,
      body {
        height: 100%;
        line-height: 1.5;
        font-family: 'Montserrat';
        font-weight: 300;
        color: black;
      }

      .root {
        min-height: 100%;
        position: relative;
      }

      .header-wrapper,
      .main-wrapper,
      .footer-wrapper {
        width: 85%;
        margin: auto;
      }

      .header-wrapper,
      .main-wrapper {
        padding: 10px;
        margin: 2px auto;
```

```
      }

      h1 {
        font-size: 70px;
        font-weight: 300;
      }

      h2,
      h3 {
        font-weight: 300;
      }

      header {
        background-color: #61dbfb;
        padding: 10px;
      }

      main {
        padding: 10px;
        padding-bottom: 60px;
        /* Height of the footer */
      }

      ul {
        margin-left: 15px;
      }

      ul li {
        list-style: none;
      }

      footer {
        position: absolute;
        bottom: 0;
        width: 100%;
        height: 60px;
        /* Height of the footer */
        background: #6cf;
      }

      .footer-wrapper {
        font-weight: 400;
        text-align: center;
        line-height: 60px;
      }
    </style>
  </head>

  <body>
    <div class="root"></div>

    <script
      crossorigin
      src="https://unpkg.com/react@16/umd/react.development.js"
    ></script>
    <script
      crossorigin
      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
    ></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
    <script type="text/babel">
      // To get the root element from the HTML document
```

```
      const rootElement = document.querySelector('.root')

      // JSX element, header
      const header = (
        <header>
          <div className='header-wrapper'>
            <h1>Welcome to 30 Days Of React</h1>
            <h2>Getting Started React</h2>
            <h3>JavaScript Library</h3>
            <p>Instructor: Asabeneh Yetayeh</p>
            <small>Date: Oct 1, 2020</small>
          </div>
        </header>
      )

      // JSX element, main
      const main = (
        <main>
          <div className='main-wrapper'>
            <p>
              Prerequisite to get started{' '}
              <strong>
                <em>react.js</em>
              </strong>
              :
            </p>
            <ul>
              <li>HTML</li>
              <li>CSS</li>
              <li> JavaScript</li>
            </ul>
          </div>
        </main>
      )

      // JSX element, footer
      const footer = (
        <footer>
          <div className='footer-wrapper'>
            <p>Copyright 2020</p>
          </div>
        </footer>
      )

      // JSX element, app
      const app = (
        <div className='app'>
          {header}
          {main}
          {footer}
        </div>
      )

      // we render the JSX element using the ReactDOM package
      ReactDOM.render(app, rootElement)
    </script>
  </body>
</html>
```

Inject data to a JSX Element

We can pass different data types as dynamic data to JSX elements. Dynamic data could be a string, number, boolean, array or object. To inject data to a JSX we use the {} bracket.

```
const welcome = 'Welcome to 30 Days Of React'
const title = 'Getting Started React'
const subtitle = 'JavaScript Library'
const authorFirstName = 'Asabeneh'
const authorLastName = 'Yetayeh'
const date = 'Oct 1, 2020'

// JSX element, header
const header = (
  <header>
    <div className='header-wrapper'>
      <h1>{welcome}</h1>
      <h2>{title}</h2>
      <h3>{subtitle}</h3>
      <p>
        Instructor: {authorFirstName} {authorLastName}
      </p>
      <small>Date: {date}</small>
    </div>
  </header>
)
```

Similar to the header JSX element, we can implement data injection to main and footer JSX elements.

Injecting a string to a JSX Element

```
const welcome = 'Welcome to 30 Days Of React'
const title = 'Getting Started React'
const subtitle = 'JavaScript Library'
const firstName = 'Asabeneh'
const lastName = 'Yetayeh'
const date = 'Oct 2, 2020'

// JSX element, header

// JSX element, header
const header = (
  <header>
    <div className='header-wrapper'>
      <h1>{welcome}</h1>
      <h2>{title}</h2>
      <h3>{subtitle}</h3>
      <p>
        Instructor: {firstName} {lastName}
      </p>
      <small>Date: {date}</small>
    </div>
  </header>
)
```

Injecting a number to a JSX Element

```
const numOne = 3
const numTwo = 2

const result = (
  <p>
    {numOne} + {numTwo} = {numOne + numTwo}
  </p>
)

const yearBorn = 1820
const currentYear = new Date().getFullYear()
const age = currentYear - yearBorn
const personAge = <p> {age}</p>
```

With the example above, we can see it is possible to do some arithmetic calculations and ternary operations

Injecting an array to a JSX Element

```
const techs = ['HTML', 'CSS', 'JavaScript']

// JSX element, main
const main = (
  <main>
    <div className='main-wrapper'>
      <p>
        Prerequisite to get started{' '}
        <strong>
          <em>react.js</em>
        </strong>
        :
      </p>
      <ul>{techs}</ul>
    </div>
  </main>
)
```

Injecting an Object to a JSX Element

We can inject a string, number, boolean, or array data to JSX, but we can't directly inject an object. We should extract object values first or destructure the content of the object before we inject the data to the JSX element. For instance, we can write firstName and lastName inside an object and extract them to use them inside JSX.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      href="https://fonts.googleapis.com/css?family=Montserrat:300,400,500|Roboto:300,400,500&display=swap"
      rel="stylesheet"
```

```
    />

    <title>30 Days Of React Challenge</title>
    <style>
      /* == General style === */
      * {
        box-sizing: border-box;
        padding: 0;
        margin: 0;
      }

      html,
      body {
        height: 100%;
        line-height: 1.5;
        font-family: 'Montserrat';
        font-weight: 300;
        color: black;
      }

      .root {
        min-height: 100%;
        position: relative;
      }

      .header-wrapper,
      .main-wrapper,
      .footer-wrapper {
        width: 85%;
        margin: auto;
      }

      .header-wrapper,
      .main-wrapper {
        padding: 10px;
        margin: 2px auto;
      }

      h1 {
        font-size: 70px;
        font-weight: 300;
      }

      h2,
      h3 {
        font-weight: 300;
      }

      header {
        background-color: #61dbfb;
        padding: 10px;
      }

      main {
        padding: 10px 10px 60px;
        /* Height of the footer */
      }

      ul {
        margin-left: 15px;
      }
```

```
      ul li {
        list-style: none;
      }

      footer {
        position: absolute;
        bottom: 0;
        width: 100%;
        height: 60px;
        /* Height of the footer */
        background: #6cf;
      }

      .footer-wrapper {
        font-weight: 400;
        text-align: center;
        line-height: 60px;
      }
    </style>
  </head>

  <body>
    <div class="root"></div>

    <script
      crossorigin
      src="https://unpkg.com/react@16/umd/react.development.js"
    ></script>
    <script
      crossorigin
      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
    ></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
    <script type="text/babel">
      // To get the root element from the HTML document
      const rootElement = document.querySelector('.root')
      // JSX element, header
      const welcome = 'Welcome to 30 Days Of React'
      const title = 'Getting Started React'
      const subtitle = 'JavaScript Library'
      const author = {
        firstName: 'Asabeneh',
        lastName: 'Yetayeh',
      }
      const date = 'Oct 2, 2020'

      // JSX element, header
      const header = (
        <header>
          <div className='header-wrapper'>
            <h1>{welcome}</h1>
            <h2>{title}</h2>
            <h3>{subtitle}</h3>
            <p>
              Instructor: {author.firstName} {author.lastName}
            </p>
            <small>Date: {date}</small>
          </div>
        </header>
      )

      const numOne = 3
```

```
    const numTwo = 2

    const result = (
      <p>
        {numOne} + {numTwo} = {numOne + numTwo}
      </p>
    )

    const yearBorn = 1820
    const currentYear = new Date().getFullYear()
    const age = currentYear - yearBorn
    const personAge = (
      <p>
        {' '}
        {author.firstName} {author.lastName} is {age} years old
      </p>
    )

    // JSX element, main
    const techs = ['HTML', 'CSS', 'JavaScript']

    // JSX element, main
    const main = (
      <main>
        <div className='main-wrapper'>
          <p>
            Prerequisite to get started{' '}
            <strong>
              <em>react.js</em>
            </strong>
            :
          </p>
          <ul>{techs}</ul>
          {result}
          {personAge}
        </div>
      </main>
    )

    const copyRight = 'Copyright 2020'

    // JSX element, footer
    const footer = (
      <footer>
        <div className='footer-wrapper'>
          <p>{copyRight}</p>
        </div>
      </footer>
    )

    // JSX element, app
    const app = (
      <div className='app'>
        {header}
        {main}
        {footer}
      </div>
    )

    // we render the JSX element using the ReactDOM package
    ReactDOM.render(app, rootElement)
  </script>
```

```
  </body>
</html>
```

In order to format a list before we inject it to JSX, we can use map. As a react developer, we should have a very good understanding of functional programming(map, filter, reduce, find, some, every).

```
const techs = ['HTML', 'CSS', 'JavaScript']
const techsFormatted = techs.map((tech) => <li>{tech}</li>)
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      href="https://fonts.googleapis.com/css?family=Montserrat:300,400,500|Roboto:300,400,500&display=swap"
      rel="stylesheet"
    />

    <title>30 Days Of React Challenge</title>
    <style>
      /* == General style === */
      * {
        box-sizing: border-box;
        padding: 0;
        margin: 0;
      }

      html,
      body {
        height: 100%;
        line-height: 1.5;
        font-family: 'Montserrat';
        font-weight: 300;
        color: black;
      }

      .root {
        min-height: 100%;
        position: relative;
      }

      .header-wrapper,
      .main-wrapper,
      .footer-wrapper {
        width: 85%;
        margin: auto;
      }

      .header-wrapper,
      .main-wrapper {
        padding: 10px;
        margin: 2px auto;
      }

      h1 {
```

```
        font-size: 70px;
        font-weight: 300;
      }

      h2,
      h3 {
        font-weight: 300;
      }

      header {
        background-color: #61dbfb;
        padding: 10px;
      }

      main {
        padding: 10px 10px 60px;
        /* Height of the footer */
      }

      ul {
        margin-left: 15px;
      }

      ul li {
        list-style: none;
      }

      footer {
        position: absolute;
        bottom: 0;
        width: 100%;
        height: 60px;
        /* Height of the footer */
        background: #6cf;
      }

      .footer-wrapper {
        font-weight: 400;
        text-align: center;
        line-height: 60px;
      }
    </style>
  </head>

  <body>
    <div class="root"></div>

    <script
      crossorigin
      src="https://unpkg.com/react@16/umd/react.development.js"
    ></script>
    <script
      crossorigin
      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
    ></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
    <script type="text/babel">
      // To get the root element from the HTML document
      const rootElement = document.querySelector('.root')
      // JSX element, header
      const welcome = 'Welcome to 30 Days Of React Challenge'
      const title = 'Getting Started React'
```

```
const subtitle = 'JavaScript Library'
const author = {
  firstName: 'Asabeneh',
  lastName: 'Yetayeh',
}
const date = 'Oct 2, 2020'

// JSX element, header
const header = (
  <header>
    <div className='header-wrapper'>
      <h1>{welcome}</h1>
      <h2>{title}</h2>
      <h3>{subtitle}</h3>
      <p>
        Instructor: {author.firstName} {author.lastName}
      </p>
      <small>Date: {date}</small>
    </div>
  </header>
)

const numOne = 3
const numTwo = 2

const result = (
  <p>
    {numOne} + {numTwo} = {numOne + numTwo}
  </p>
)

const yearBorn = 1820
const currentYear = new Date().getFullYear()
const age = currentYear - yearBorn
const personAge = (
  <p>
    {' '}
    {author.firstName} {author.lastName} is {age} years old
  </p>
)

// JSX element, main
const techs = ['HTML', 'CSS', 'JavaScript']
const techsFormatted = techs.map((tech) => <li>{tech}</li>)

// JSX element, main
const main = (
  <main>
    <div className='main-wrapper'>
      <p>
        Prerequisite to get started{' '}
        <strong>
          <em>react.js</em>
        </strong>
        :
      </p>
      <ul>{techsFormatted}</ul>
      {result}
      {personAge}
    </div>
  </main>
)
```

```
      const copyRight = 'Copyright 2020'

      // JSX element, footer
      const footer = (
        <footer>
          <div className='footer-wrapper'>
            <p>{copyRight}</p>
          </div>
        </footer>
      )

      // JSX element, app
      const app = (
        <div className='app'>
          {header}
          {main}
          {footer}
        </div>
      )

      // we render the JSX element using the ReactDOM package
      ReactDOM.render(app, rootElement)
    </script>
  </body>
</html>
```

We can apply a key to each list element for the tech array to avoid errors about each list element not having a key:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      href="https://fonts.googleapis.com/css?family=Montserrat:300,400,500|Roboto:300,400,500&display=swap"
      rel="stylesheet"
    />

    <title>30 Days Of React Challenge</title>
    <style>
      /* == General style === */
      * {
        box-sizing: border-box;
        padding: 0;
        margin: 0;
      }

      html,
      body {
        height: 100%;
        line-height: 1.5;
        font-family: 'Montserrat';
        font-weight: 300;
        color: black;
      }

      .root {
```

```
        min-height: 100%;
        position: relative;
      }

      .header-wrapper,
      .main-wrapper,
      .footer-wrapper {
        width: 85%;
        margin: auto;
      }

      .header-wrapper,
      .main-wrapper {
        padding: 10px;
        margin: 2px auto;
      }

      h1 {
        font-size: 70px;
        font-weight: 300;
      }

      h2,
      h3 {
        font-weight: 300;
      }

      header {
        background-color: #61dbfb;
        padding: 10px;
      }

      main {
        padding: 10px;
        padding-bottom: 60px;
        /* Height of the footer */
      }

      ul {
        margin-left: 15px;
      }

      ul li {
        list-style: none;
      }

      footer {
        position: absolute;
        bottom: 0;
        width: 100%;
        height: 60px;
        /* Height of the footer */
        background: #6cf;
      }

      .footer-wrapper {
        font-weight: 400;
        text-align: center;
        line-height: 60px;
      }
    </style>
  </head>
```

```
<body>
  <div class="root"></div>

  <script
    crossorigin
    src="https://unpkg.com/react@16/umd/react.development.js"
  ></script>
  <script
    crossorigin
    src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
  ></script>
  <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  <script type="text/babel">
    // To get the root element from the HTML document
    const rootElement = document.querySelector('.root')
    // JSX element, header
    const welcome = 'Welcome to 30 Days Of React Challenge'
    const title = 'Getting Started React'
    const subtitle = 'JavaScript Library'
    const author = {
      firstName: 'Asabeneh',
      lastName: 'Yetayeh',
    }
    const date = 'Oct 2, 2020'

    // JSX element, header
    const header = (
      <header>
        <div className='header-wrapper'>
          <h1>{welcome}</h1>
          <h2>{title}</h2>
          <h3>{subtitle}</h3>
          <p>
            Instructor: {author.firstName} {author.lastName}
          </p>
          <small>Date: {date}</small>
        </div>
      </header>
    )

    const numOne = 3
    const numTwo = 2

    const result = (
      <p>
        {numOne} + {numTwo} = {numOne + numTwo}
      </p>
    )

    const yearBorn = 1820
    const currentYear = 2020
    const age = currentYear - yearBorn
    const personAge = (
      <p>
        {' '}
        {author.firstName} {author.lastName} is {age} years old
      </p>
    )

    // JSX element, main
    const techs = ['HTML', 'CSS', 'JavaScript']
```

```
      const techsFormatted = techs.map((tech) => <li key={tech}>{tech}</li>)

      // JSX element, main
      const main = (
        <main>
          <div className='main-wrapper'>
            <p>
              Prerequisite to get started{' '}
              <strong>
                <em>react.js</em>
              </strong>
              :
            </p>
            <ul>{techsFormatted}</ul>
            {result}
            {personAge}
          </div>
        </main>
      )

      const copyRight = 'Copyright 2020'

      // JSX element, footer
      const footer = (
        <footer>
          <div className='footer-wrapper'>
            <p>{copyRight}</p>
          </div>
        </footer>
      )

      // JSX element, app
      const app = (
        <div className='app'>
          {header}
          {main}
          {footer}
        </div>
      )

      // we render the JSX element using the ReactDOM package
      ReactDOM.render(app, rootElement)
    </script>
  </body>
</html>
```