

# 30 Days Of React: Components



LinkedIn



Follow @asabeneh

776

Author: [Asabeneh Yetayeh](#)

October, 2020

[<< Day 3](#) | [Day 5 >>](#)



- [Components](#)
  - [Big picture of components](#)
  - [JavaScript function](#)
  - [JavaScript Class](#)
  - [Creating React Component](#)
    - [Functional Component](#)
    - [Rendering components](#)
    - [Injecting data to JSX in React Component](#)
    - [Further on Functional components](#)
- [Exercises: Components](#)
  - [Exercises: Level 1](#)
  - [Exercises: Level 2](#)
  - [Exercises: Level 3](#)

## Components

A React component is a small, reusable code, which is responsible for one part of the application UI. A React application is an aggregation of components. React can help us to build reusable components. The following diagram shows different components. All the components have different border colors. In React we assemble

different components together to create an application. We use JavaScript functions or classes to make components. If we use a function, the component will be a functional component, but if we use a class, the component will be a class-based component.

Components can be:

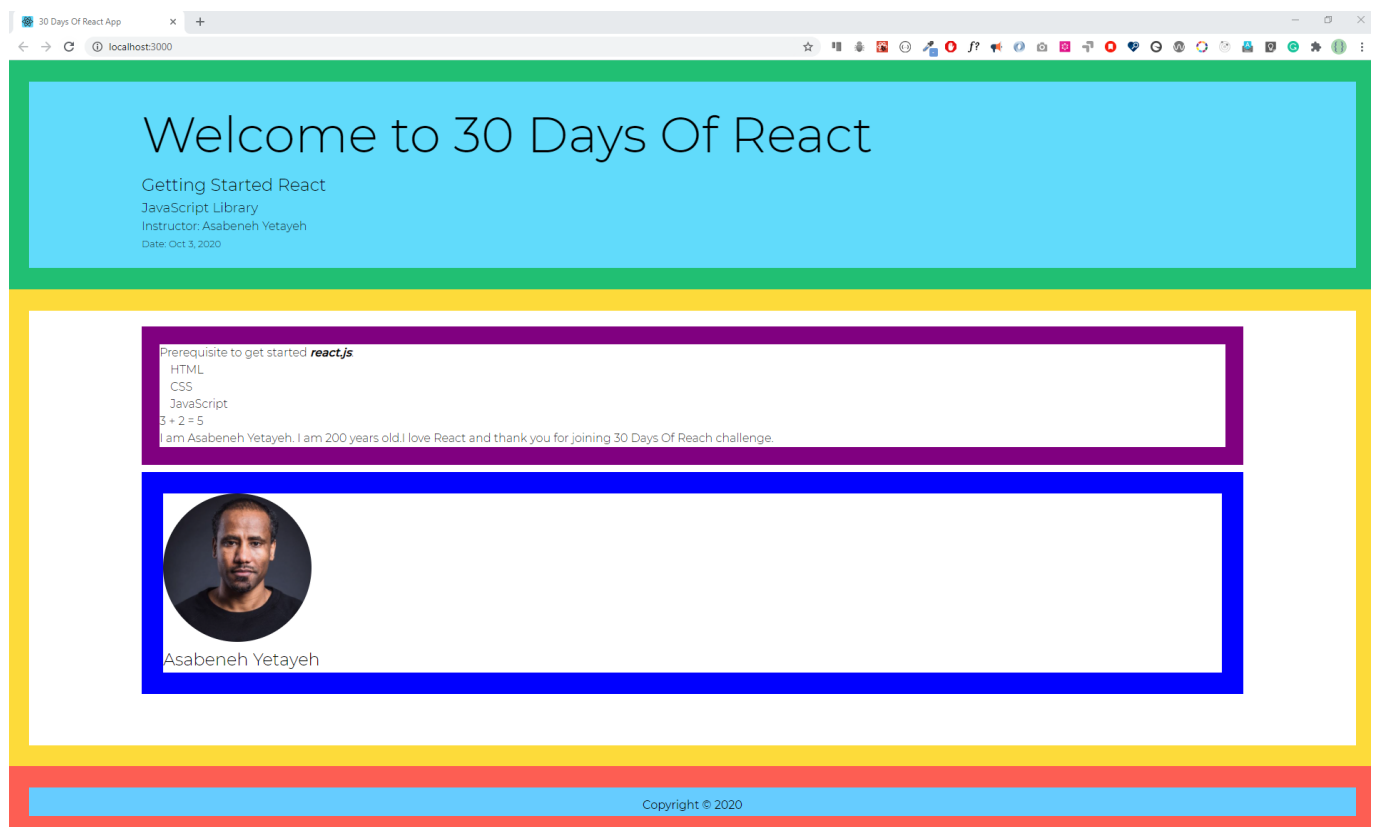
- Functional Component / Presentational Component / Stateless Component / Dumb Component
- Class Component / Container Component / Stateful Component / Smart Component

The classification of components above does not work for the latest version of React, but it is good to know the former definition and how the previous versions work.

So, let us change all the JSX to components. Components in React are JavaScript functions or classes, that return a JSX. Component name must start with an uppercase, and if the name is two words, it should be CamelCase - a camel with two humps.

## Big picture of components

In the previous section we agreed, that a website or an application is made of buttons, forms, texts, media objects, header, section, article and footer. If we have a million-dollar button, we can use this button all the time, instead of recreating it all over again, whenever we need a button. The same goes for input fields, forms, header or footer. That is where the power of the component comes. In the following diagram, the header, main and footer are components. Inside the main there is also a user card component and a text section component. All the different colors represent different components. How many colors do you see? Each color represent a single component. We have five components in this diagram.



Before we jump into React components, let's do some functions and class refreshers.

## JavaScript function

A JavaScript function could be either a regular function or an arrow function. These functions are not exactly the same there is a slight difference between them.

```
const getUserInfo = (firstName, lastName, country, title, skills) => {
  return `${firstName} ${lastName}, a ${title} developer based in ${country}. He
  knows ${skills.join(
    ', '
  )} `
}
// When we call this function we need parameters
const skills = ['HTML', 'CSS', 'JS', 'React']
console.log(
  getUserInfo('Asabeneh', 'Yetayeh', 'Finland', 'FullStack Developer', skills)
)
```

## JavaScript Class

A class is a blueprint of an object. We instantiate a class to create different objects. In addition, we can create children, by inheriting all the methods and properties of the parent.

```
class Parent {
  constructor(firstName, lastName, country, title) {
    // we bind the params with this class object using this keyword
    this.firstName = firstName
    this.lastName = lastName
    this.country = country
    this.title = title
  }
  getPersonInfo() {
    return `${this.firstName} ${this.lastName}, a ${this.title} developer base in
    ${this.country} `
  }
  parentMethod() {
    // code goes here
  }
}

const p1 = new Parent('Asabeneh', 'Yetayeh', 'Finland', 'FullStack Developer')

class Child extends Parent {
  constructor(firstName, lastName, country, title, skills) {
    super(firstName, lastName, country, title)
    this.skills = skills
    // we bind the child params with the this keyword to this child object
  }
  getSkills() {
    let len = this.skills.length
    return len > 0 ? this.skills.join(' ') : 'No skills found'
  }
  childMethod() {
```

```

    // code goes here
  }
}

const skills = ['HTML', 'CSS', 'JS', 'React']

const child = new Child(
  'Asabeneh',
  'Yetayeh',
  'Finland',
  'FullStack Developer',
  skills
)

```

We just briefly covered function and class. React component is made of JavaScript functions or classes, so let's make a React component now.

## Creating React Component

### Functional Component

Using a JavaScript function, we can make a functional React component.

```

// React component syntax
// it can be arrow function, function declaration or function expression
const jsx = <tag> Content </tag>
const ComponentName = () => {
  return jsx
}

```

The following expression is a JSX element.

```

// 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 3, 2020</small>
    </div>
  </header>
)

// React Component
const Header = () => {
  return header
}

```

```
// or we can just return the JSX

const Header = () => {
  return (
    <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 3, 2020</small>
      </div>
    </header>
  )
}

// Even th above code can be written like this
// Explicitly returning the JSX
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 3, 2020</small>
    </div>
  </header>
)
```

## Rendering components

Now, lets change all the JSX elements we had to components. When we call JSX element we use curly brackets and when we call components we do as follows . If we pass an attribute, when we call the component name, we call it props(<ComponentName propsName = {data-type} />). We will talk about props in another section.[Live on code pen](#)

Let's render first the *Header* component.

```
// index.js
import React from 'react'
import ReactDOM from 'react-dom'

// Header Component
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>Asabeneh Yetayeh</p>
        <small>Oct 3, 2020</small>
      </div>
    </header>
  )

  const rootElement = document.getElementById('root')
  // we render the JSX element using the ReactDOM package
  ReactDOM.render(<Header />, rootElement)

```

Now, let's create an App component , that will wrap the Header, Main and Footer. Then the App component will be render on the DOM.

```

// index.js
import React from 'react'
import ReactDOM from 'react-dom'
import asabenehImage from './images/asabeneh.jpg'

// Header Component
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>Asabeneh Yetayeh</p>
      <small>Oct 3, 2020</small>
    </div>
  </header>
)

// User Card Component
const UserCard = () => (
  <div className='user-card'>
    <img src={asabenehImage} alt='asabeneh image' />
    <h2>Asabeneh Yetayeh</h2>
  </div>
)

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

// Main Component
const Main = () => (
  <main>
    <div className='main-wrapper'>
      <p>Prerequisite to get started react.js:</p>

```

```

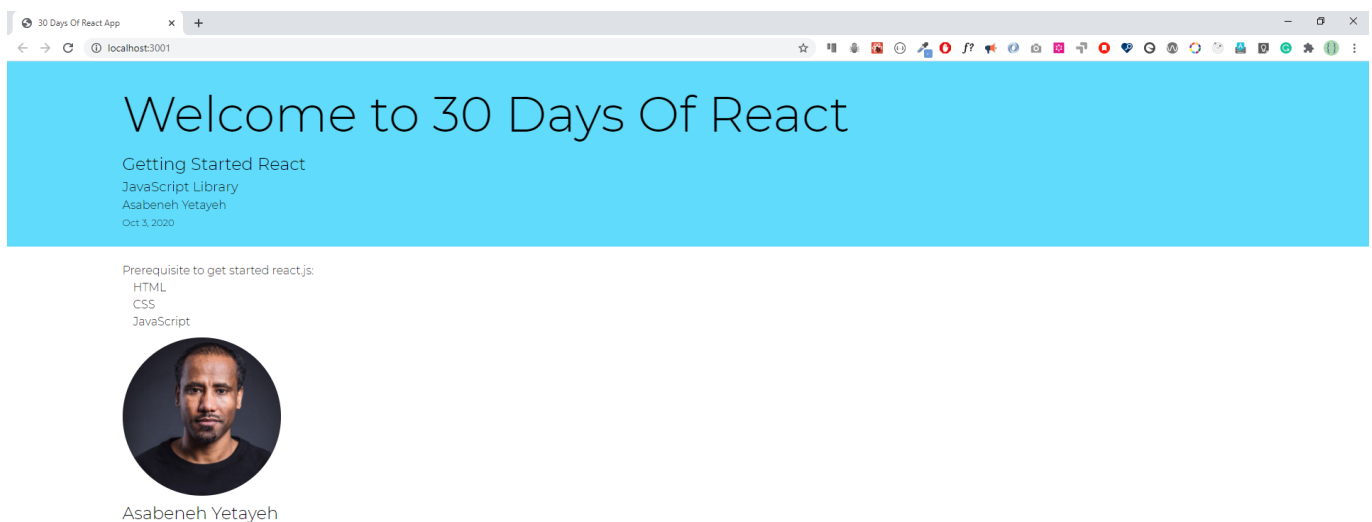
        <ul>
          <TechList />
        </ul>
        <UserCard />
      </div>
    </main>
  )

// Footer Component
const Footer = () => (
  <footer>
    <div className='footer-wrapper'>
      <p>Copyright 2020</p>
    </div>
  </footer>
)

// The App, or the parent or the container component
const App = () => (
  <div className='app'>
    <Header />
    <Main />
    <Footer />
  </div>
)

const rootElement = document.getElementById('root')
// we render the App component using the ReactDOM package
ReactDOM.render(<App />, rootElement)

```



## Injecting data to JSX in React Component

So far, we used static data on the JSX elements. Now let's pass different data types as dynamic data. The dynamic data could be strings, numbers, booleans, arrays or objects. Let us see each of the data types step by step. To inject data to a JSX we use the {} bracket.

In this section we inject only strings

```
import React from 'react'
import ReactDOM from 'react-dom'

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 3, 2020'

// JSX element, header
const header = () => {
  return (
    <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>
  )
}
const rootElement = document.getElementById('root')
// we render the App component using the ReactDOM package
ReactDOM.render(<Header />, rootElement)
```

Similar to the Header component we can implement to Main and Footer component.

```
// 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>
)

// User Card Component
const UserCard = () => (
  <div className='user-card'>
    <img src={asabenehImage} alt='asabeneh image' />
    <h2>
      {author.firstName} {author.lastName}
    </h2>
  </div>
)

// 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'>

```

```

    <div>
      <p>
        Prerequisite to get started{' '}
        <strong>
          <em>react.js</em>
        </strong>
        :
      </p>
      <ul>{techsFormatted}</ul>
      {result}
      {personAge}
    </div>
    <UserCard />
  </div>
</main>
)

const copyRight = '2020'

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

// JSX element, app
const app = () => (
  <div className='app'>
    <Header />
    <Main />
    <Footer />
  </div>
)

// we render the App component using the ReactDOM package
ReactDOM.render(<App />, rootElement)

```

## Further on Functional components

We have transformed all the JSX elements of Day 2 to functional components, and by now you are very familiar with components. Let's create more components. What is the smallest size of a component? A component that returns only a single HTML as JSX is considered as a small component. A button component or an alert box component, or just an input field component.

```
const Button = () => <button>action</button>
```

The *Button* component is made of a single HTML button element. Let's style this button using JavaScript style object. All CSS properties should be camelCase to make a JavaScript CSS object. If we pass a number without unit as CSS value, it is considered as px. See the example below.

```
const buttonStyles = {
  padding: '10px 20px',
  background: 'rgb(0, 255, 0)',
  border: 'none',
  borderRadius: 5,
}
const Button = () => <button style={buttonStyles}> action </button>
```

The Button component is a dumb component, because it does not take any parameters and we cannot change the action text dynamically. We need to pass props to the button, to change the value dynamically. We will see props in the next section. Before we close today's lesson let's make another, more functional component, which displays a random hexadecimal number.

```
import React from 'react'
import ReactDOM from 'react-dom'

// Hexadecimal color generator
const hexaColor = () => {
  let str = '0123456789abcdef'
  let color = ''
  for (let i = 0; i < 6; i++) {
    let index = Math.floor(Math.random() * str.length)
    color += str[index]
  }
  return '#' + color
}

const HexaColor = () => <div>{hexaColor()}</div>

const rootElement = document.getElementById('root')
// we render the App component using the ReactDOM package
ReactDOM.render(<HexaColor />, rootElement)
```

## Exercises: Components

---

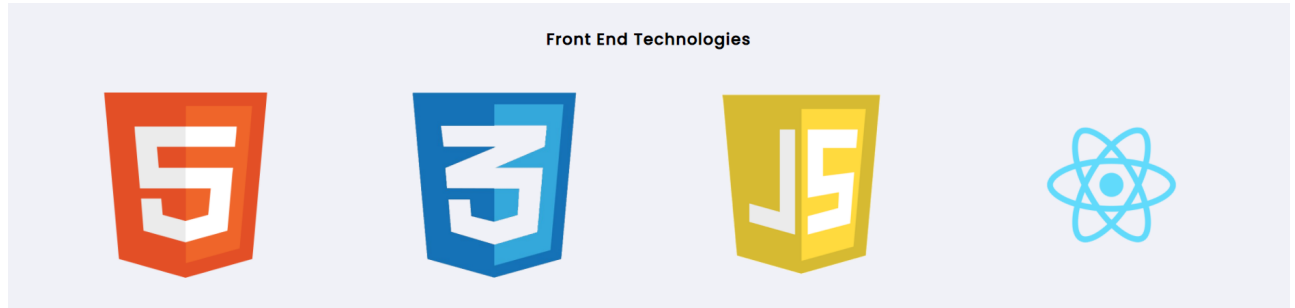
### Exercises: Level 1

1. What is the difference between a regular function and an arrow function?
2. What is a React Component?
3. How do you make a React functional component?
4. What is the difference between a pure JavaScript function and a functional component?
5. How small is a React component?

6. Can we make a button or input field component?
7. Make a reusable Button component.
8. Make a reusable InputField component.
9. Make a reusable alert box component with one div parent element and one p child element of the div(warning alert box, success alert box).

## Exercises: Level 2

1. Create functional components and display the following images

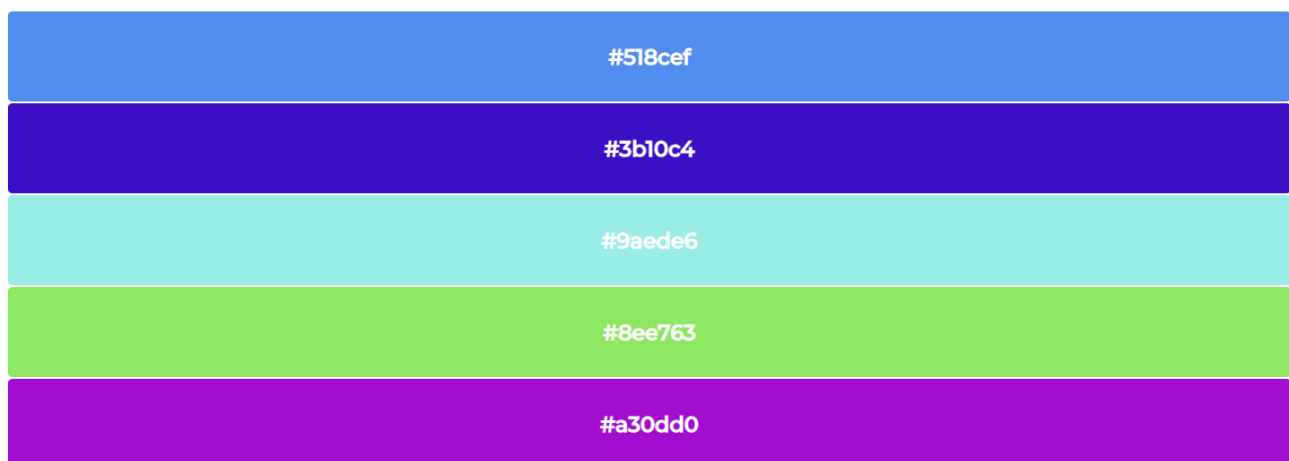


2. Use functional component to create the following design


A subscription form titled "SUBSCRIBE" with the text "Sign up with your email address to receive news and updates." Below the text are three input fields labeled "First name", "Last name", and "Email". At the bottom is a red "Subscribe" button.

## Exercises: Level 3

1. Use the given hexadecimal color generator in the example to create these random colors



2. Use functional component to design the following user card.



**ASABENEH YETAYEH** ✓

Senior Developer, Finland

**SKILLS**

HTML CSS Sass JS React Redux Node MongoDB Python Flask Django NumPy Pandas Data Analysis

MYSQL GraphQL D3.js Gatsby Docker Heroku Git

🕒 Joined on Aug 30, 2020

🎉 CONGRATULATIONS ! 🎉

<< [Day 3](#) | [Day 5](#) >>