Class-Based Component

**Component**

Components are one of the core concepts of React. They are the foundation upon which you build user interfaces (UI), which makes them the perfect place to start your React journey!

To define a React component as a class, extend the built-in Component class and define a [render method:](https://react.dev/reference/react/Component#render)

import { Component } from 'react';

class Greeting extends Component {

render() {

return <h1>Hello, {this.props.name}!</h1>;

}

}

Only the render method is required, other methods are optional.

**Usage**

**Defining a class component**

To define a React component as a class, extend the built-in Component class and define a render method:

import { Component } from 'react';

class Greeting extends Component {

render() {

return <h1>Hello, {this.props.name}!</h1>;

}

}

React will call your render method whenever it needs to figure out what to display on the screen. Usually, you will return some JSX from it. Your render method should be a pure function: it should only calculate the JSX.

Similarly to function components, a class component can receive information by props from its parent component. However, the syntax for reading props is different. For example, if the parent component renders <Greeting name="Taylor" />, then you can read the name prop from this.props, like this.props.name:

/ Pure JavaScript class and child

// Imagine this what we import from React package

class Component {

constructor(props) {}

}

// This how we make class based components by inheriting from the parent

class Child extends Component {

constructor(props) {

super(props)

}

}

Functional React component

// index.js

import React from 'react'

import ReactDOM from 'react-dom'

// Header Component

// Functional 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 6, 2020</small>

</div>

</header>

)

const rootElement = document.getElementById('root')

ReactDOM.render(<Header />, rootElement)

Class based React component is a child of React.Component and it has a built-in render method and it may have a constructor.

//index.js

import React from 'react'

import ReactDOM from 'react-dom'

// class based component

class Header extends React.Component {

render() {

return (

<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 7, 2020</small>

</div>

</header>

)

}

}

const rootElement = document.getElementById('root')

ReactDOM.render(<Header />, rootElement)

Let's see the above component with a constructor

//index.js

import React from 'react'

import ReactDOM from 'react-dom'

// class base component

class Header extends React.Component {

constructor(props) {

super(props)

// the code inside the constructor run before any other code

}

render() {

return (

<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 7, 2020</small>

</div>

</header>

)

}

}

const rootElement = document.getElementById('root')

ReactDOM.render(<Header />, rootElement)

}

const rootElement = document.getElementById('root')

ReactDOM.render(<Header />, rootElement)

Example

import { Component } from 'react';

class Greeting extends Component {

render() {

return <h1>Hello, {this.props.name}!</h1>;

}

}

export default function App() {

return (

<>

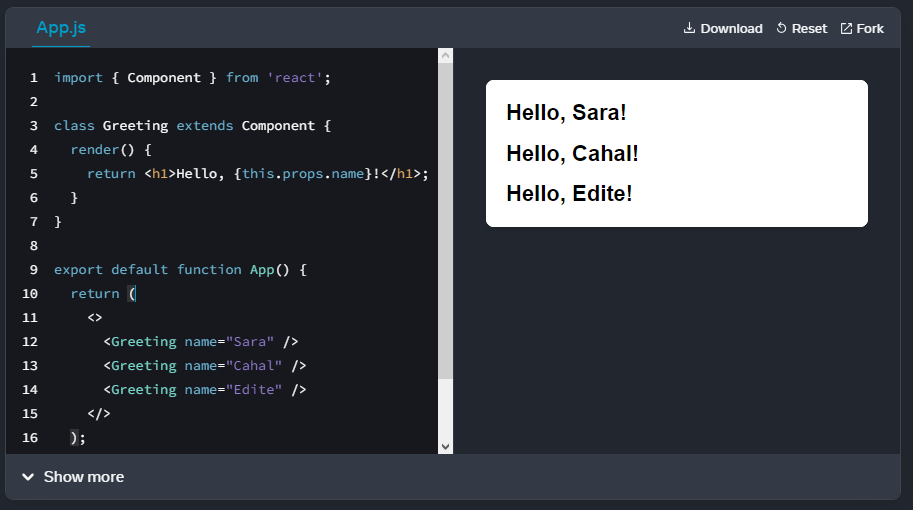
<Greeting name="Sara" />

<Greeting name="Cahal" />

<Greeting name="Edite" />

</>

);



**Adding state to a class component**

import { Component } from 'react';

export default class Counter extends Component {

state = {

name: 'Taylor',

age: 42,

};

handleNameChange = (e) => {

this.setState({

name: e.target.value

});

}

handleAgeChange = () => {

this.setState({

age: this.state.age + 1

});

};

render() {

return (

<>

<input

value={this.state.name}

onChange={this.handleNameChange}

/>

<button onClick={this.handleAgeChange}>

Increment age

</button>

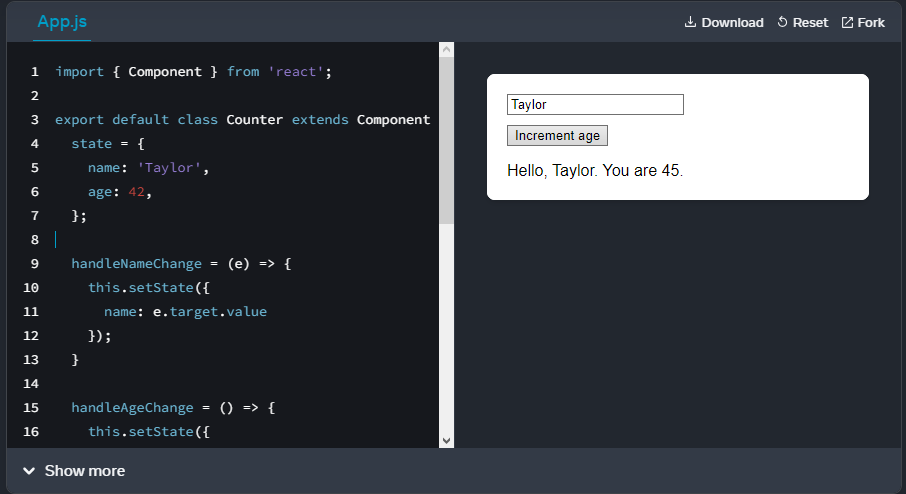
<p>Hello, {this.state.name}. You are {this.state.age}.</p>

</>

);

}

}



**Important methods and Features**

**Accessing props in Class components**

We stated that props is a means to send data from on component to another or we can call it that props is a data carrier. Therefore, we should handle props in class based component too. We can access props of a class based component using the keyword *this*. See the example below.

// index.js

import React from 'react'

import ReactDOM from 'react-dom'

// class based component

class Header extends React.Component {

constructor(props) {

super(props)

// the code inside the constructor run before any other code

}

render() {

return (

<header>

<div className='header-wrapper'>

<h1>{this.props.data.welcome}</h1>

<h2>{this.props.data.title}</h2>

<h3>

{this.props.data.author.firstName} {this.props.data.author.lastName}

</h3>

<small>{this.props.data.date}</small>

</div>

</header>

)

}

}

const App = () => {

const data = {

welcome: 'Welcome to 30 Days Of React',

title: 'Getting Started React',

subtitle: 'JavaScript Library',

author: {

firstName: 'Asabeneh',

lastName: 'Yetayeh',

},

date: 'Oct 7, 2020',

}

return (

<div className='app'>

<Header data={data} />

</div>

)

}

const rootElement = document.getElementById('root')

ReactDOM.render(<App />, rootElement)

As you can see in the above example, to get the data out from props we have write *props.data* every time. We can avoid this repetition using destructuring.

// index.js

import React from 'react'

import ReactDOM from 'react-dom'

// class based component

class Header extends React.Component {

constructor(props) {

super(props)

// the code inside the constructor run before any other code

}

render() {

console.log(this.props.data)

const {

welcome,

title,

subtitle,

author: { firstName, lastName },

date,

} = this.props.data

return (

<header>

<div className='header-wrapper'>

<h1>{welcome}</h1>

<h2>{title}</h2>

<h3>{subtitle}</h3>

<p>

{firstName} {lastName}

</p>

<small>{date}</small>

</div>

</header>

)

}

}

const App = () => {

const data = {

welcome: 'Welcome to 30 Days Of React',

title: 'Getting Started React',

subtitle: 'JavaScript Library',

author: {

firstName: 'Asabeneh',

lastName: 'Yetayeh',

},

date: 'Oct 6, 2020',

}

return (

<div className='app'>

<Header data={data} />

</div>

)

}

const rootElement = document.getElementById('root')

ReactDOM.render(<App />, rootElement)

As you can see, the above code cleaner than the previous. Now, let's clean all the components we have and put all together.

// index.js

import React from 'react'

import ReactDOM from 'react-dom'

// class based component

class Header extends React.Component {

constructor(props) {

super(props)

// the code inside the constructor run before any other code

}

render() {

console.log(this.props.data)

const {

welcome,

title,

subtitle,

author: { firstName, lastName },

date,

} = this.props.data

return (

<header>

<div className='header-wrapper'>

<h1>{welcome}</h1>

<h2>{title}</h2>

<h3>{subtitle}</h3>

<p>

{firstName} {lastName}

</p>

<small>{date}</small>

</div>

</header>

)

}

}

// TechList Component

// class base component

class TechList extends React.Component {

constructor(props) {

super(props)

}

render() {

const { techs } = this.props

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

return techsFormatted

}

}

// Main Component

// Class Component

class Main extends React.Component {

constructor(props) {

super(props)

}

render() {

return (

<main>

<div className='main-wrapper'>

<p>Prerequisite to get started react.js:</p>

<ul>

<TechList techs={this.props.techs} />

</ul>

</div>

</main>

)

}

}

// Footer Component

// Class component

class Footer extends React.Component {

constructor(props) {

super(props)

}

render() {

return (

<footer>

<div className='footer-wrapper'>

<p>Copyright {this.props.date.getFullYear()}</p>

</div>

</footer>

)

}

}

class App extends React.Component {

render() {

const data = {

welcome: 'Welcome to 30 Days Of React',

title: 'Getting Started React',

subtitle: 'JavaScript Library',

author: {

firstName: 'Asabeneh',

lastName: 'Yetayeh',

},

date: 'Oct 7, 2020',

}

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

return (

<div className='app'>

<Header data={data} />

<Main techs={techs} />

<Footer date={new Date()} />

</div>

)

}

}

const rootElement = document.getElementById('root')

ReactDOM.render(<App />, rootElement)

**Methods in Class based component**

We access methods in class based component. Most of the time, we write different methods on the parent component and we pass them to child components. Let's see the implementation.

Let's add a method on this component.

//index.js

import React from 'react'

import ReactDOM from 'react-dom'

// class based component

class Header extends React.Component {

greetPeople = () => {

alert('Welcome to 30 Days Of React Challenge, 2020')

}

render() {

return (

<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 7, 2020</small>

<button onClick={this.greetPeople}> Greet </button>

</div>

</header>

)

}

}

const rootElement = document.getElementById('root')

ReactDOM.render(<Header />, rootElement)

The invoking or calling of the method triggers when the event occurs. Therefore, whenever you pass a method to an event listener do not invoke the method.

Now, let's the code we had add all the necessary methods.

// index.js

import React from 'react'

import ReactDOM from 'react-dom'

import asabenehImage from './images/asabeneh.jpg'

// Fuction to show month date year

// User Card Component

const UserCard = ({ user: { firstName, lastName, image } }) => (

<div className='user-card'>

<img src={image} alt={firstName} />

<h2>

{firstName}

{lastName}

</h2>

</div>

)

// A button component

const Button = ({ text, onClick, style }) => (

<button style={style} onClick={onClick}>

{text}

</button>

)

// CSS styles in JavaScript Object

const buttonStyles = {

backgroundColor: '#61dbfb',

padding: 10,

border: 'none',

borderRadius: 5,

margin: 3,

cursor: 'pointer',

fontSize: 18,

color: 'white',

}

// class based component

class Header extends React.Component {

constructor(props) {

super(props)

// the code inside the constructor run before any other code

}

render() {

console.log(this.props.data)

const {

welcome,

title,

subtitle,

author: { firstName, lastName },

date,

} = this.props.data

return (

<header>

<div className='header-wrapper'>

<h1>{welcome}</h1>

<h2>{title}</h2>

<h3>{subtitle}</h3>

<p>

{firstName} {lastName}

</p>

<small>{date}</small>

</div>

</header>

)

}

}

// TechList Component

// class base component

class TechList extends React.Component {

constructor(props) {

super(props)

}

render() {

const { techs } = this.props

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

return techsFormatted

}

}

// Main Component

// Class Component

class Main extends React.Component {

constructor(props) {

super(props)

}

render() {

return (

<main>

<div className='main-wrapper'>

<p>Prerequisite to get started react.js:</p>

<ul>

<TechList techs={this.props.techs} />

</ul>

<UserCard user={this.props.user} />

<Button

text='Greet People'

onClick={this.props.greetPeople}

style={buttonStyles}

/>

<Button

text='Show Time'

onClick={this.props.handleTime}

style={buttonStyles}

/>

</div>

</main>

)

}

}

// Footer Component

// Class component

class Footer extends React.Component {

constructor(props) {

super(props)

}

render() {

return (

<footer>

<div className='footer-wrapper'>

<p>Copyright {this.props.date.getFullYear()}</p>

</div>

</footer>

)

}

}

class App extends React.Component {

showDate = (time) => {

const months = [

'January',

'February',

'March',

'April',

'May',

'June',

'July',

'August',

'September',

'October',

'November',

'December',

]

const month = months[time.getMonth()].slice(0, 3)

const year = time.getFullYear()

const date = time.getDate()

return ` ${month} ${date}, ${year}`

}

handleTime = () => {

alert(this.showDate(new Date()))

}

greetPeople = () => {

alert('Welcome to 30 Days Of React Challenge, 2020')

}

render() {

const data = {

welcome: 'Welcome to 30 Days Of React',

title: 'Getting Started React',

subtitle: 'JavaScript Library',

author: {

firstName: 'Asabeneh',

lastName: 'Yetayeh',

},

date: 'Oct 7, 2020',

}

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

// copying the author from data object to user variable using spread operator

const user = { ...data.author, image: asabenehImage }

return (

<div className='app'>

<Header data={data} />

<Main

user={user}

techs={techs}

handleTime={this.handleTime}

greetPeople={this.greetPeople}

/>

<Footer date={new Date()} />

</div>

)

}

}

const rootElement = document.getElementById('root')

ReactDOM.render(<App />, rootElement)