States

The state is a built-in React object that is used to contain data or information about the [component.](https://www.simplilearn.com/tutorials/reactjs-tutorial/reactjs-components) A component’s state can change over time; whenever it changes, the component re-renders. The change in state can happen as a response to user action or system-generated events and these changes determine the behavior of the component and how it will render

class Greetings extends React.Component {

state = {

name: "World"

};

updateName() {

this.setState({ name: "Simplilearn" });

}

render() {

return(

<div>

{this.state.name}

</div>

)

}

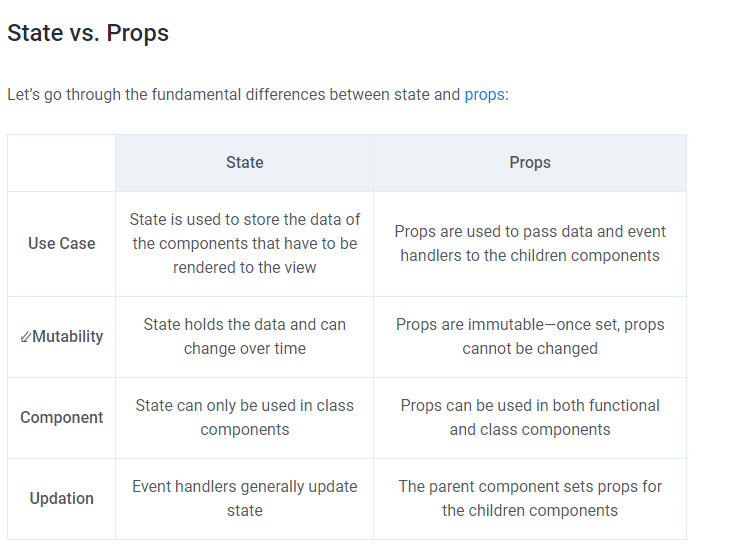
}

* A state can be modified based on user action or network changes
* Every time the state of an object changes, React re-renders the component to the browser
* The state object is initialized in the constructor
* The state object can store multiple properties
* this.setState() is used to change the value of the state object
* setState() function performs a shallow merge between the new and the previous state

**The setState() Method**

State can be updated in response to event handlers, server responses, or prop changes. This is done using the setState() method. The setState() method enqueues all of the updates made to the component state and instructs React to re-render the component and its children with the updated state.

Always use the [setState() method](https://css-tricks.com/understanding-react-setstate/" \o "setState() method" \t "_blank) to change the state object, since it will ensure that the component knows it’s been updated and calls the render() method.



**Example**

class Bike extends React.Component {

  constructor(props) {

    super(props);

     this.state = {

      make: "Yamaha",

      model: "R15",

      color: "blue"

    };

  }

  changeBikeColor = () => {

    this.setState({color: "black"});

  }

  render() {

    return (

      <div>

        <h2>My {this.state.make}</h2>

        <p>

          It is a {this.state.color}

          {this.state.model}.

        </p>

        <button

          type="button"

          onClick={this.changeBikeColor}

        >Change color</button>

      </div>

    );

  }

}

**Resetting a state using a JavaScript method**

Now, let's add some methods which increase or decrease the value of count by clicking a button. Let us add a button to increase and a button to decrease the value of count. To set the state we use react method *this.setState*. See the example below

// index.js

import React from 'react'

import ReactDOM from 'react-dom'

class App extends React.Component {

// declaring state

state = {

count: 0,

}

render() {

// accessing the state value

const count = this.state.count

return (

<div className='App'>

<h1>{count} </h1>

<button onClick={() => this.setState({ count: this.state.count + 1 })}>

Add One

</button>

</div>

)

}

}

const rootElement = document.getElementById('root')

ReactDOM.render(<App />, rootElement)

If you understand the above example, adding minus one method will be easy. Let us add the minus one method on the click event.

// index.js

import React from 'react'

import ReactDOM from 'react-dom'

class App extends React.Component {

// declaring state

state = {

count: 0,

}

render() {

// accessing the state value

const count = this.state.count

return (

<div className='App'>

<h1>{count} </h1>

<div>

<button

onClick={() => this.setState({ count: this.state.count + 1 })}

>

Add One

</button>{' '}

<button

onClick={() => this.setState({ count: this.state.count - 1 })}

>

Minus One

</button>

</div>

</div>

)

}

}

const rootElement = document.getElementById('root')

ReactDOM.render(<App />, rootElement)

Both button work well, but we need to re-structure the code well. Let us create separate methods in the component.

// index.js

import React from 'react'

import ReactDOM from 'react-dom'

class App extends React.Component {

// declaring state

state = {

count: 0,

}

// method which add one to the state

addOne = () => {

this.setState({ count: this.state.count + 1 })

}

// method which subtract one to the state

minusOne = () => {

this.setState({ count: this.state.count - 1 })

}

render() {

// accessing the state value

const count = this.state.count

return (

<div className='App'>

<h1>{count} </h1>

<div>

<button className='btn btn-add' onClick={this.addOne}>

+1

</button>{' '}

<button className='btn btn-minus' onClick={this.minusOne}>

-1

</button>

</div>

</div>

)

}

}

const rootElement = document.getElementById('root')

ReactDOM.render(<App />, rootElement)

Let us do more example about state, in the following example we will develop small application which shows either a dog or cat. We can start by setting the initial state with cat then when it is clicked it will show dog and alternatively. We need one method which changes the animal alternatively. See the code below. If you want to see live click [here](https://codepen.io/Asabeneh/full/LYVxKpq).

# // index.js

# import React from 'react'

# import ReactDOM from 'react-dom'

# class App extends React.Component {

# // declaring state

# state = {

# image: 'https://www.smithsstationah.com/imagebank/eVetSites/Feline/01.jpg',

# }

# changeAnimal = () => {

# let dogURL =

# 'https://static.onecms.io/wp-content/uploads/sites/12/2015/04/dogs-pembroke-welsh-corgi-400x400.jpg'

# let catURL =

# 'https://www.smithsstationah.com/imagebank/eVetSites/Feline/01.jpg'

# let image = this.state.image === catURL ? dogURL : catURL

# this.setState({ image })

# }

# render() {

# // accessing the state value

# const count = this.state.count

# return (

# <div className='App'>

# <h1>30 Days Of React</h1>

# <div className='animal'>

# <img src={this.state.image} alt='animal' />

# </div>

# <button onClick={this.changeAnimal} class='btn btn-add'>

# Change

# </button>

# </div>

# )

# }

# }

# const rootElement = document.getElementById('root')

# ReactDOM.render(<App />, rootElement)

# Now, let's put all the codes we have so far and also let's implement state when it is necessary.

# // index.js

# import React from 'react'

# import ReactDOM from 'react-dom'

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

# // Fuction to show month date year

# const 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}`

# }

# // 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 style={this.props.styles}>

# <div className='header-wrapper'>

# <h1>{welcome}</h1>

# <h2>{title}</h2>

# <h3>{subtitle}</h3>

# <p>

# {firstName} {lastName}

# </p>

# <small>{date}</small>

# </div>

# </header>

# )

# }

# }

# const Count = ({ count, addOne, minusOne }) => (

# <div>

# <h1>{count} </h1>

# <div>

# <Button text='+1' onClick={addOne} style={buttonStyles} />

# <Button text='-1' onClick={minusOne} style={buttonStyles} />

# </div>

# </div>

# )

# // 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() {

# const {

# techs,

# user,

# greetPeople,

# handleTime,

# changeBackground,

# count,

# addOne,

# minusOne,

# } = this.props

# return (

# <main>

# <div className='main-wrapper'>

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

# <ul>

# <TechList techs={techs} />

# </ul>

# <UserCard user={user} />

# <Button

# text='Greet People'

# onClick={greetPeople}

# style={buttonStyles}

# />

# <Button text='Show Time' onClick={handleTime} style={buttonStyles} />

# <Button

# text='Change Background'

# onClick={changeBackground}

# style={buttonStyles}

# />

# <Count count={count} addOne={addOne} minusOne={minusOne} />

# </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 {

# state = {

# count: 0,

# styles: {

# backgroundColor: '',

# color: '',

# },

# }

# 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}`

# }

# addOne = () => {

# this.setState({ count: this.state.count + 1 })

# }

# // method which subtract one to the state

# minusOne = () => {

# this.setState({ count: this.state.count - 1 })

# }

# handleTime = () => {

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

# }

# greetPeople = () => {

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

# }

# changeBackground = () => {}

# 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']

# const date = new Date()

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

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

# return (

# <div className='app'>

# {this.state.backgroundColor}

# <Header data={data} />

# <Main

# user={user}

# techs={techs}

# handleTime={this.handleTime}

# greetPeople={this.greetPeople}

# changeBackground={this.changeBackground}

# addOne={this.addOne}

# minusOne={this.minusOne}

# count={this.state.count}

# />

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

# </div>

# )

# }

# }

# const rootElement = document.getElementById('root')

# ReactDOM.render(<App />, rootElement)

I believe that now you have a very good understanding of state. After this, we will use state in other sections too because state and props is the core of a react application.