9

Conditional Rendering, Lists, Lists & Keys

Objectives

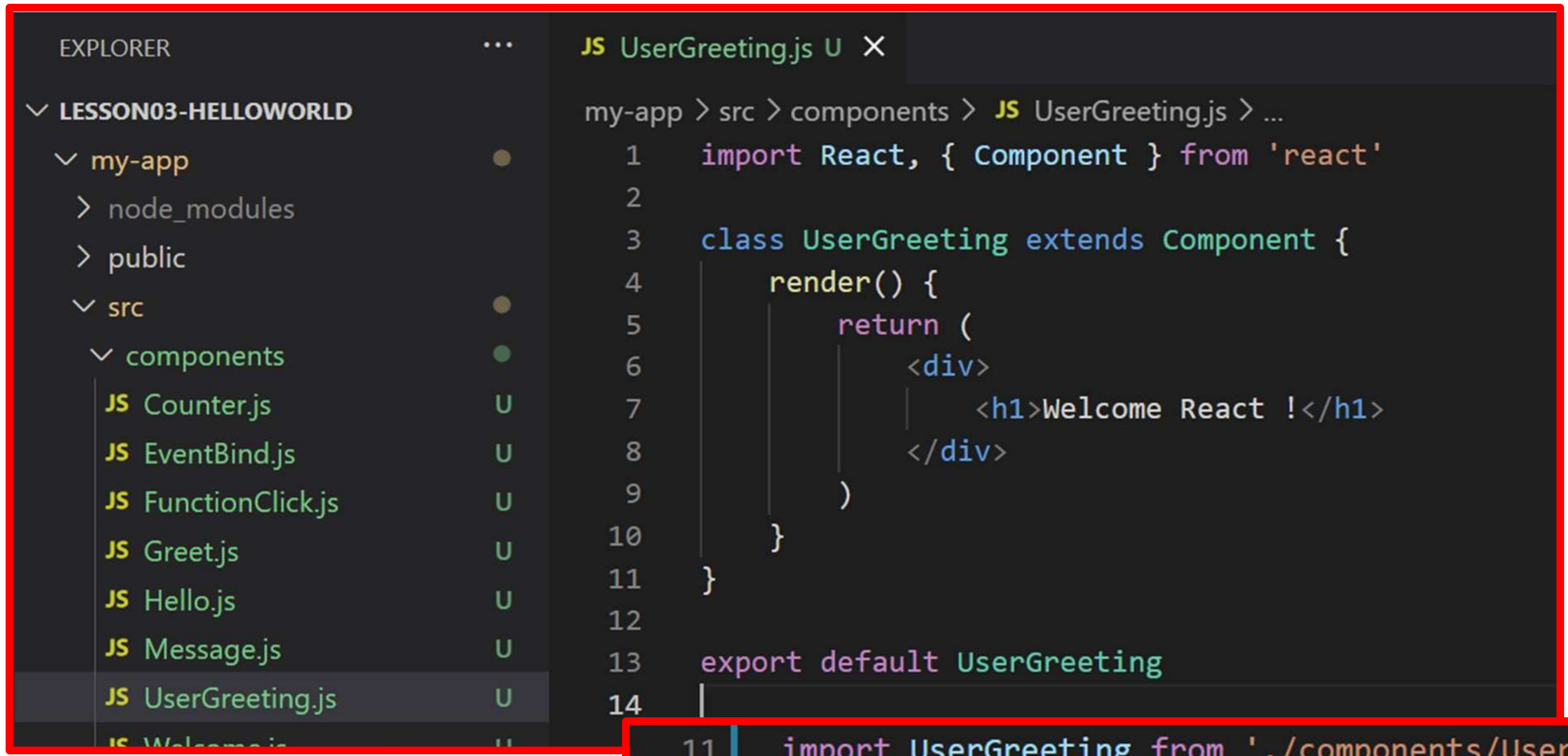
After completing this lesson, you should be able to do the following:

- Basics in State
- setState
- State, Events and Managed Controls

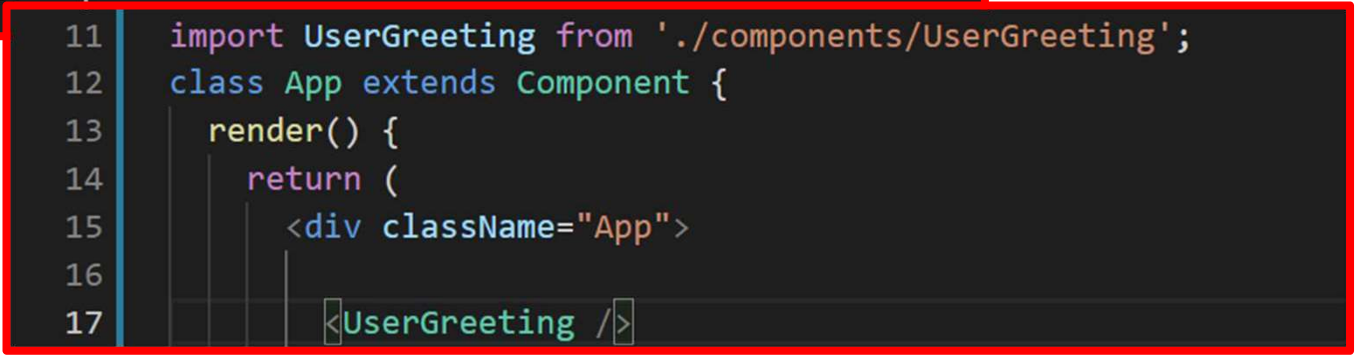
Conditional Rendering

- When you are building react applications you may often need to show or hide some HTML based on a certain condition luckily conditional rendering in react works the same way conditions work in JavaScript
  1. IF/Else
  2. Element Variables
  3. Ternary Conditional Operator
  4. Short Circuit Operator

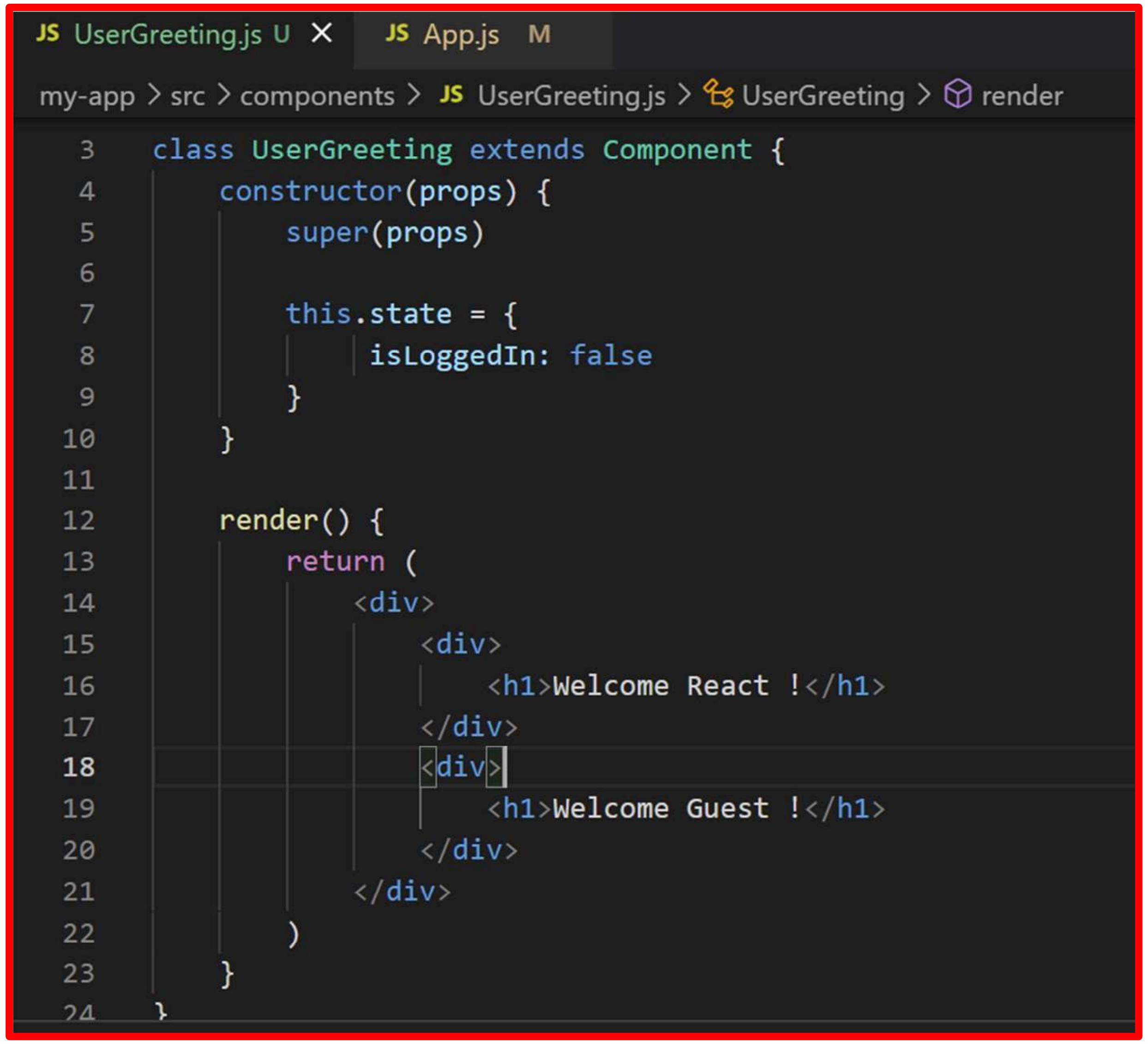
UserGreeting.js



```
import React, { Component } from 'react'

class UserGreeting extends Component {
    render() {
        return (
            <div>
                <h1>Welcome React !</h1>
            </div>
        )
    }
}

export default UserGreeting

```



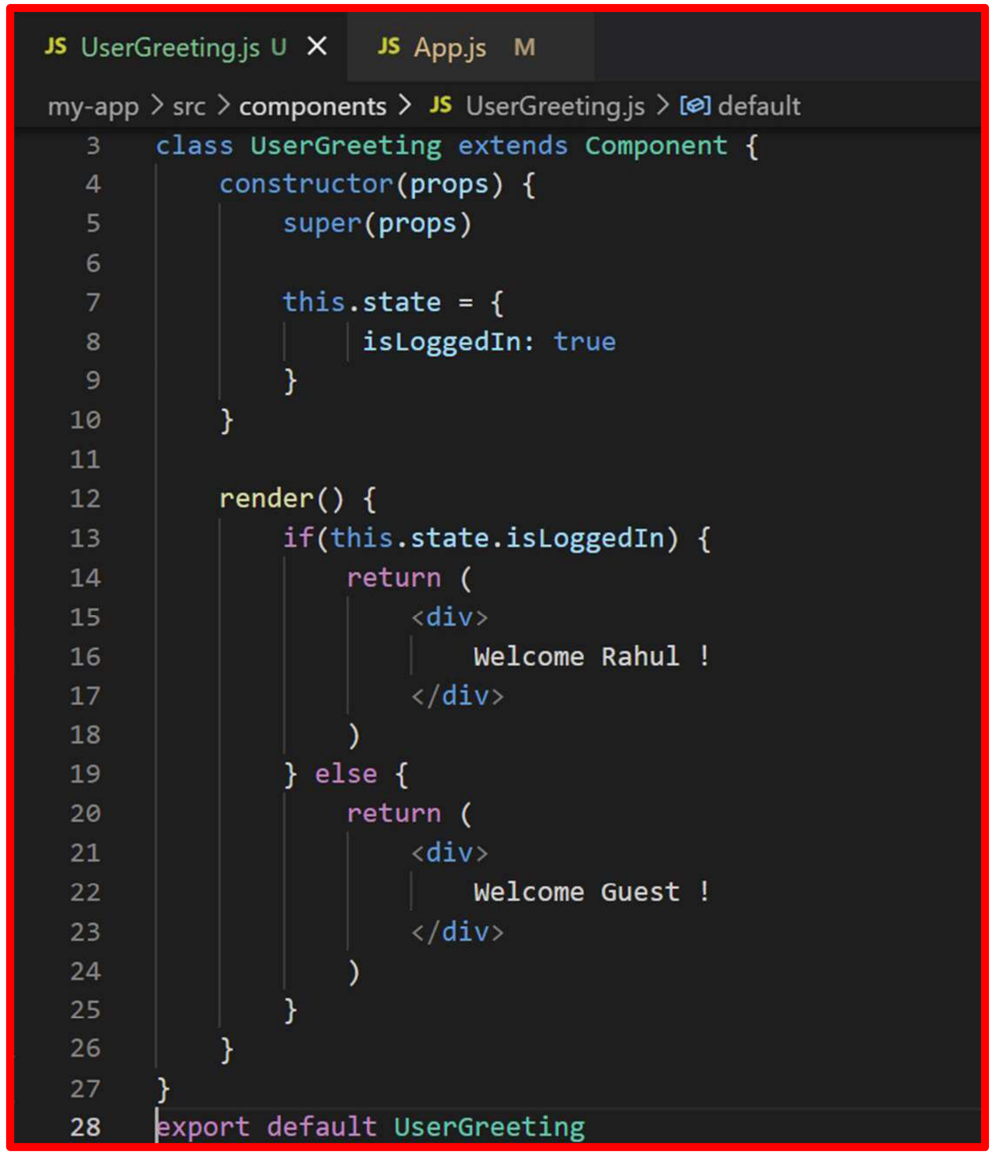
```
import UserGreeting from './components/UserGreeting';
class App extends Component {
  render() {
    return (
      <div className="App">

        <UserGreeting />
```



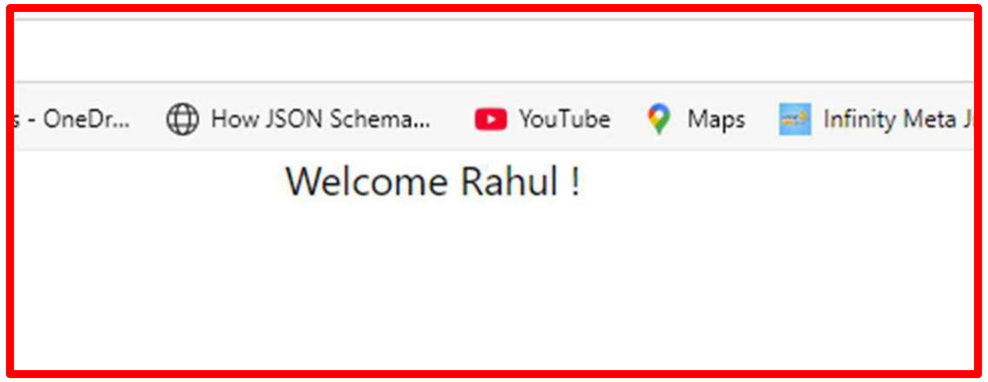
UserGreeting.js U
App.js M
my-app > src > components > UserGreeting.js > UserGreeting > render

```
class UserGreeting extends Component {
    constructor(props) {
        super(props)

        this.state = {
            isLoggedIn: false
        }
    }

    render() {
        return (
            <div>
                <div>
                    <h1>Welcome React !</h1>
                </div>
                <div>
                    <h1>Welcome Guest !</h1>
                </div>
            </div>
        )
    }
}
```

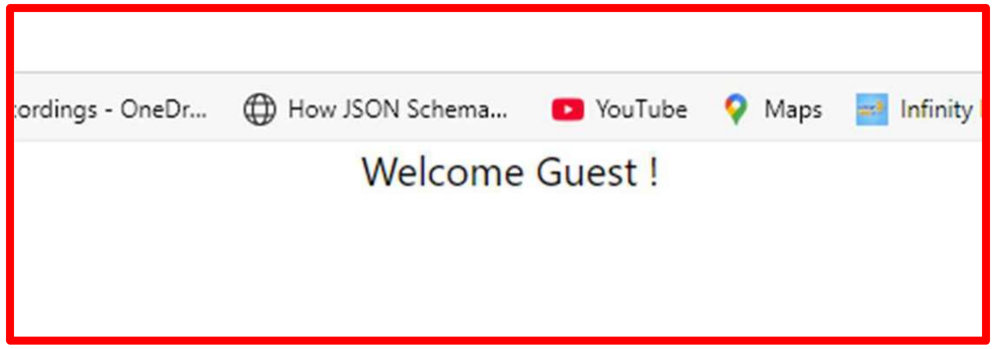
What we want is to Render the Message based on the state isLoggedIn



```
class UserGreeting extends Component {
    constructor(props) {
        super(props)

        this.state = {
            isLoggedIn: true
        }
    }

    render() {
        if(this.state.isLoggedIn) {
            return (
                <div>
                    Welcome Rahul !
                </div>
            )
        } else {
            return (
                <div>
                    Welcome Guest !
                </div>
            )
        }
    }
}
export default UserGreeting
```



Welcome Rahul !

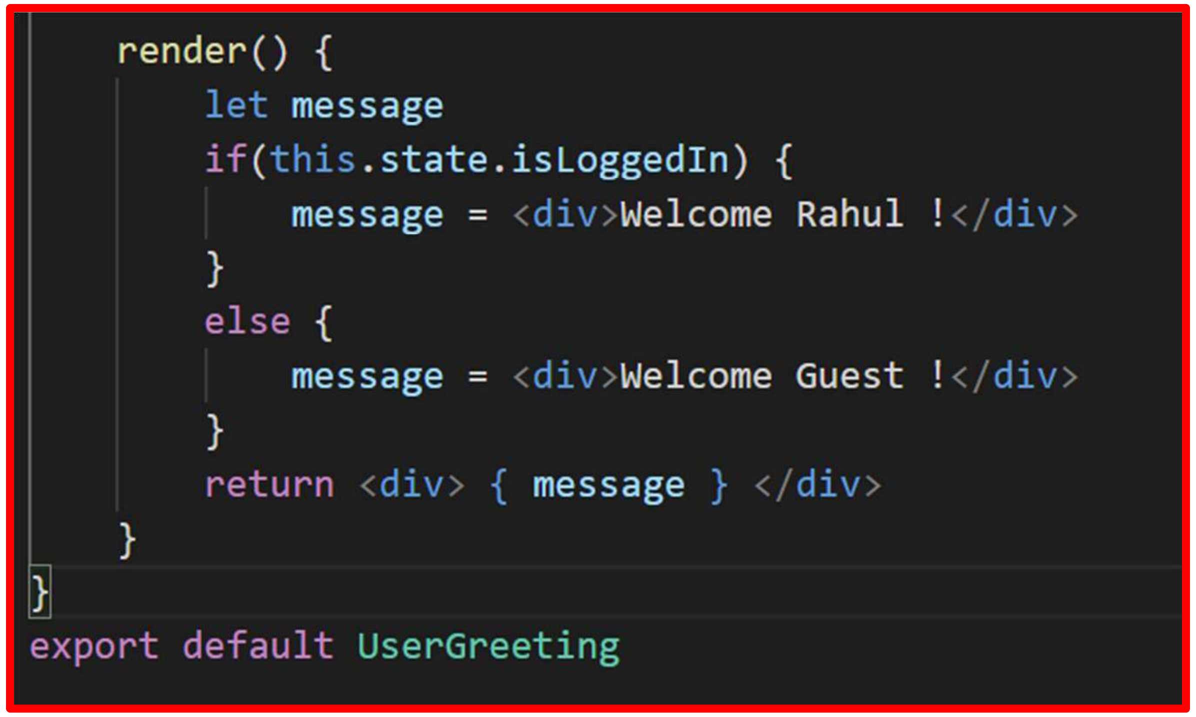


Welcome Guest !

- We might be thinking there's a lot of repetition and the render method
- looks crowded
- can we not simply use the IF/ELSE condition on the message being displayed the answer is no
- Default statements don't work inside the JSX that is because JSX is just syntactic sugar for function calls and object construction adding if else statements within the JSX is not valid
- That is the reason we have if else statements outside the JSX and the entire return statement containing the JSX is placed inside the if or else block

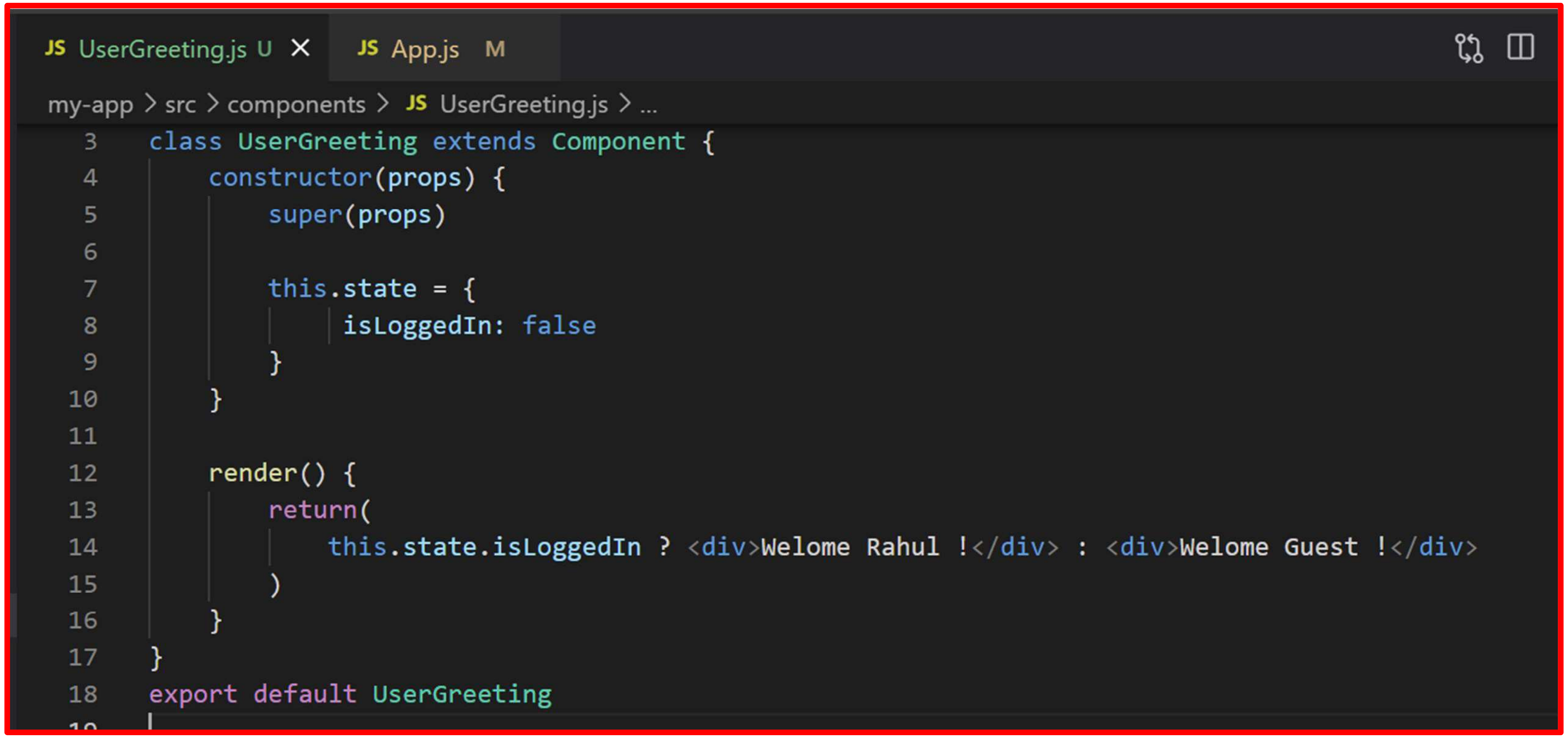
2nd Apporach [Using Element Variables]

- Here we use JS Variables to Store Elements
- This will also help you conditionally render the entire component or only a part of the component as well



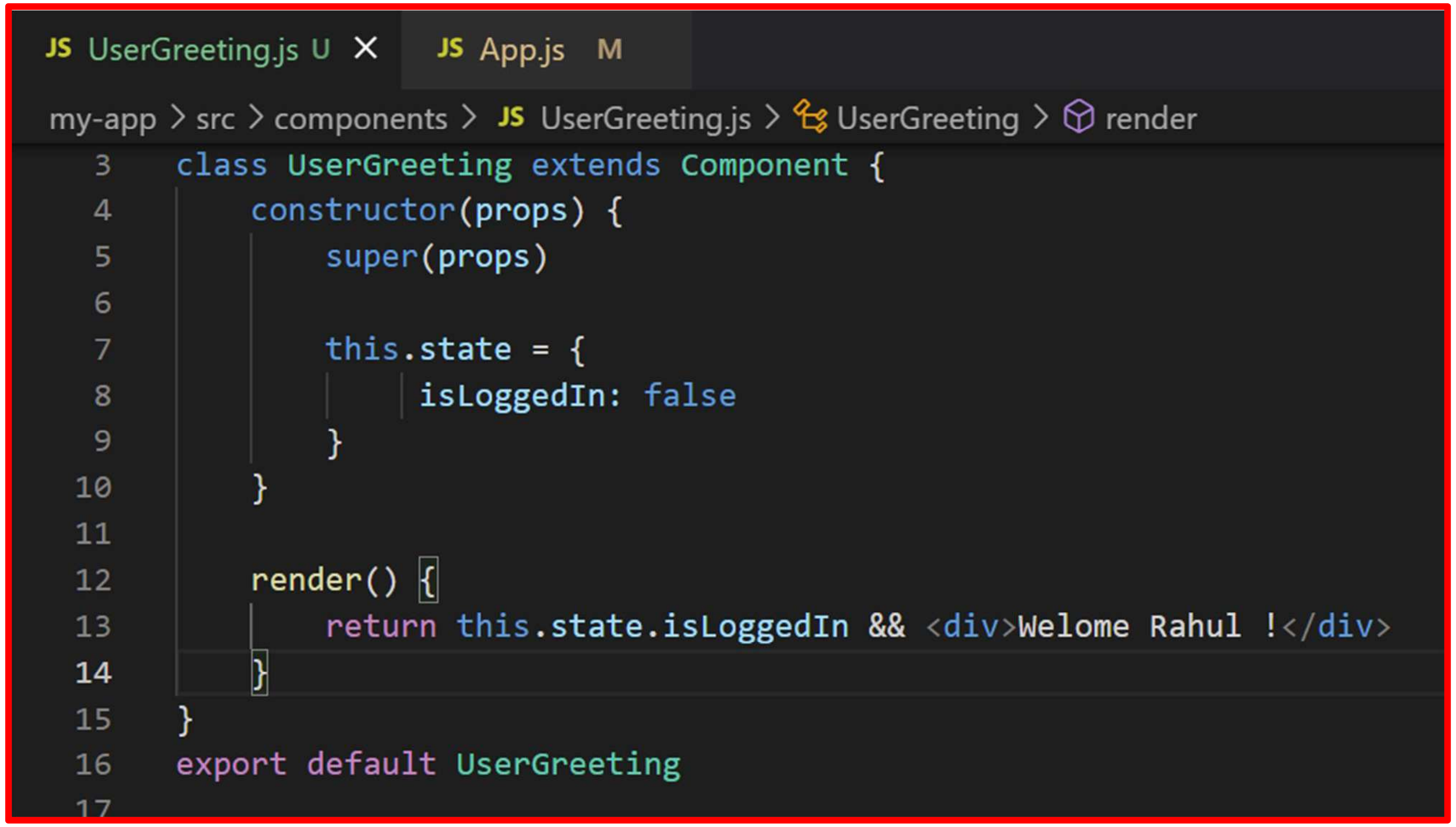
```
    render() {
        let message
        if(this.state.isLoggedIn) {
            message = <div>Welcome Rahul !</div>
        }
        else {
            message = <div>Welcome Guest !</div>
        }
        return <div> { message } </div>
    }
}
export default UserGreeting
```

3rd Approach [Ternary Conditional Operator]



```
class UserGreeting extends Component {
    constructor(props) {
        super(props)

        this.state = {
            isLoggedIn: false
        }
    }

    render() {
        return(
            this.state.isLoggedIn ? <div>Welome Rahul !</div> : <div>Welome Guest !</div>
        )
    }
}
export default UserGreeting
```

4th Approach : Short Circuit Operator


UserGreeting.js U
App.js M
my-app > src > components > UserGreeting.js > UserGreeting > render

```
class UserGreeting extends Component {
    constructor(props) {
        super(props)

        this.state = {
            isLoggedIn: false
        }
    }

    render() {
        return this.state.isLoggedIn && <div>Welome Rahul !</div>
    }
}
export default UserGreeting
```

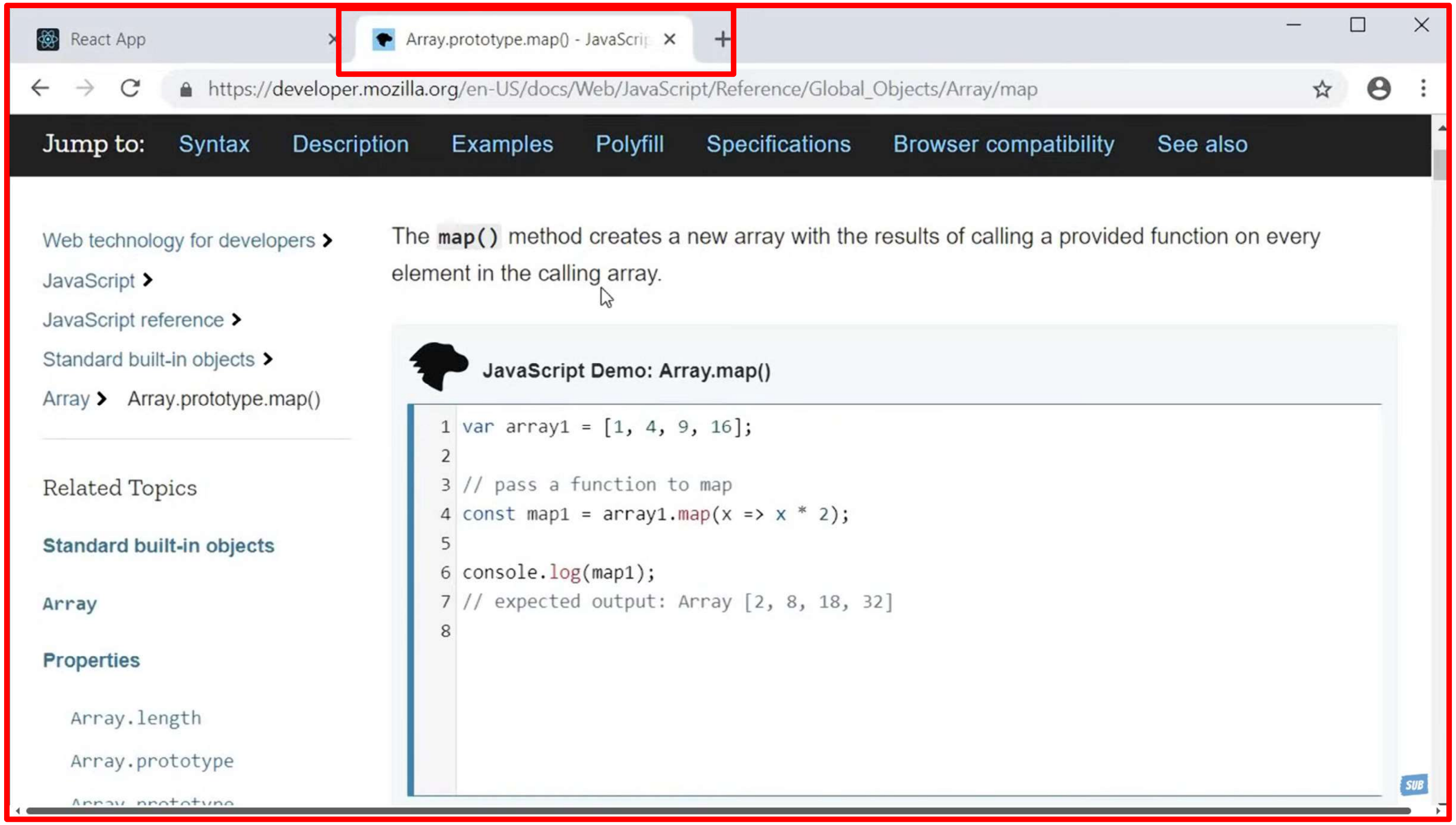
Better One

- Approach
  - 3
  - 4

Rendering List

- When you build web applications a common scenario is to display a list of items For Eg
  - a list of names a list of products
  - a list of courses and so on
- So what we want is to repeat some HTML for each item

Using Map Method



React App

Array.prototype.map() - JavaScrip

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

Jump to: Syntax Description Examples Polyfill Specifications Browser compatibility See also

Web technology for developers > JavaScript > JavaScript reference > Standard built-in objects > Array > Array.prototype.map()

Related Topics

Standard built-in objects

Array

Properties

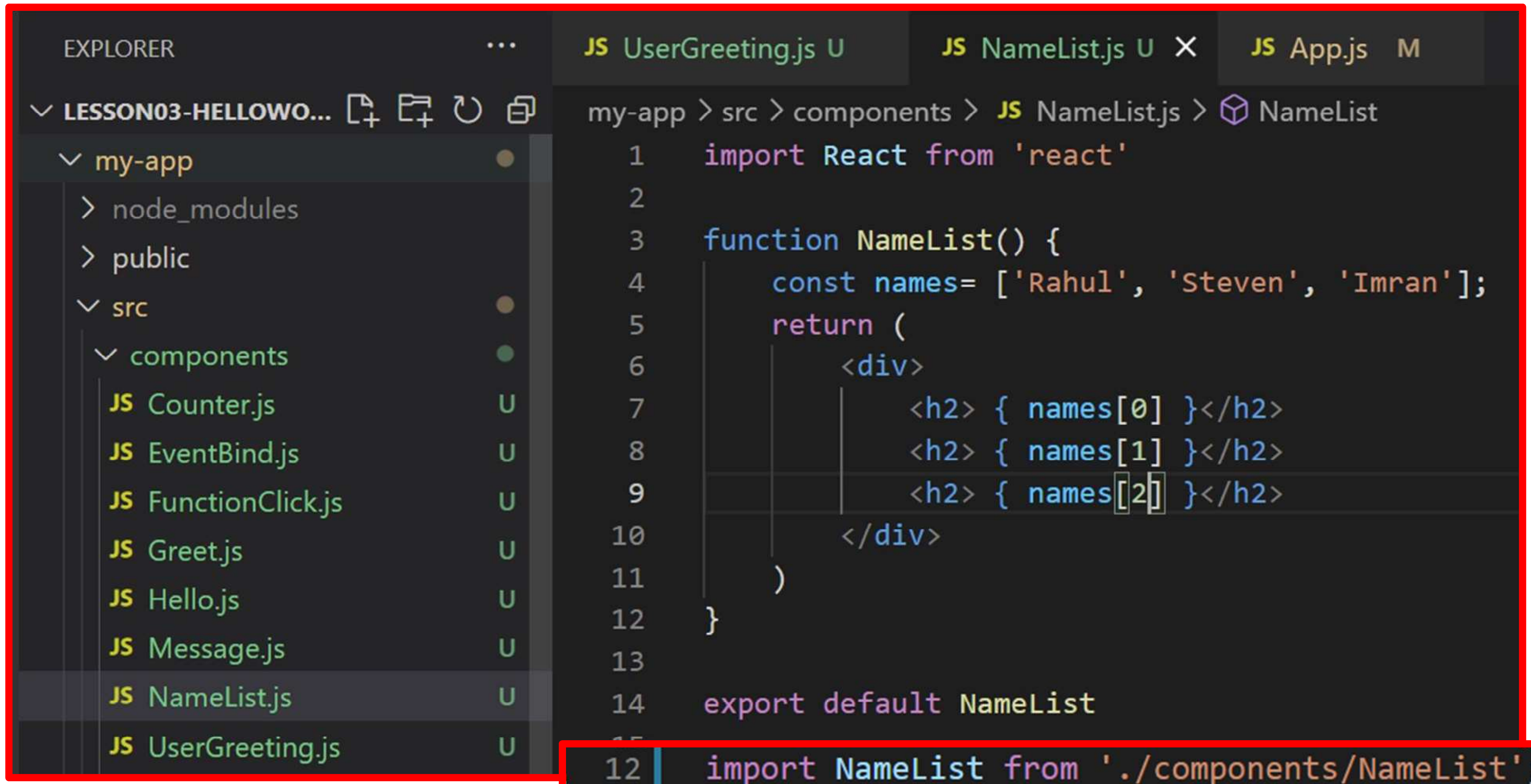
Array.length

Array.prototype

The **`map()`** method creates a new array with the results of calling a provided function on every element in the calling array.

JavaScript Demo: Array.map()

```
var array1 = [1, 4, 9, 16];

// pass a function to map
const map1 = array1.map(x => x * 2);

console.log(map1);
// expected output: Array [2, 8, 18, 32]

```



EXPLORER

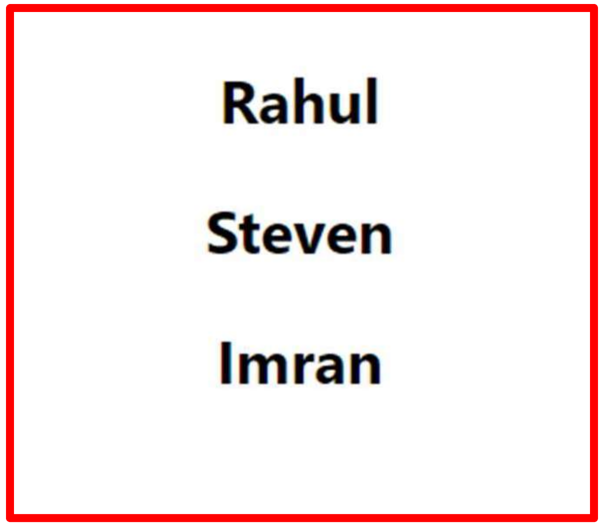
LESSON03-HELLOWO...

- my-app
  - node_modules
  - public
  - src
    - components
      - Counter.js U
      - EventBind.js U
      - FunctionClick.js U
      - Greet.js U
      - Hello.js U
      - Message.js U
      - NameList.js U
      - UserGreeting.js U

UserGreeting.js U | NameList.js U | App.js M

my-app > src > components > NameList.js > NameList

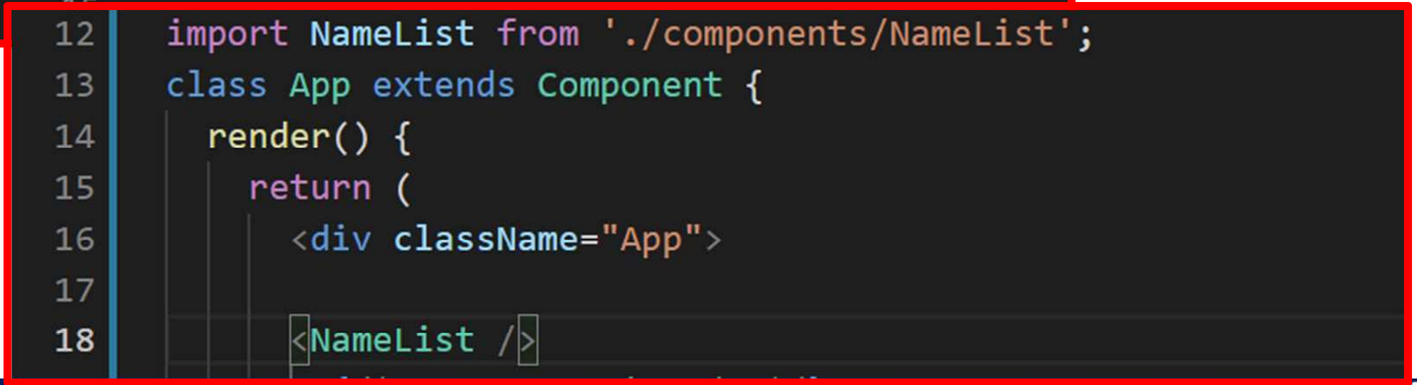
```
import React from 'react'

function NameList() {
    const names= ['Rahul', 'Steven', 'Imran'];
    return (
        <div>
            <h2> { names[0] }</h2>
            <h2> { names[1] }</h2>
            <h2> { names[2] }</h2>
        </div>
    )
}

export default NameList
```



Rahul

Steven

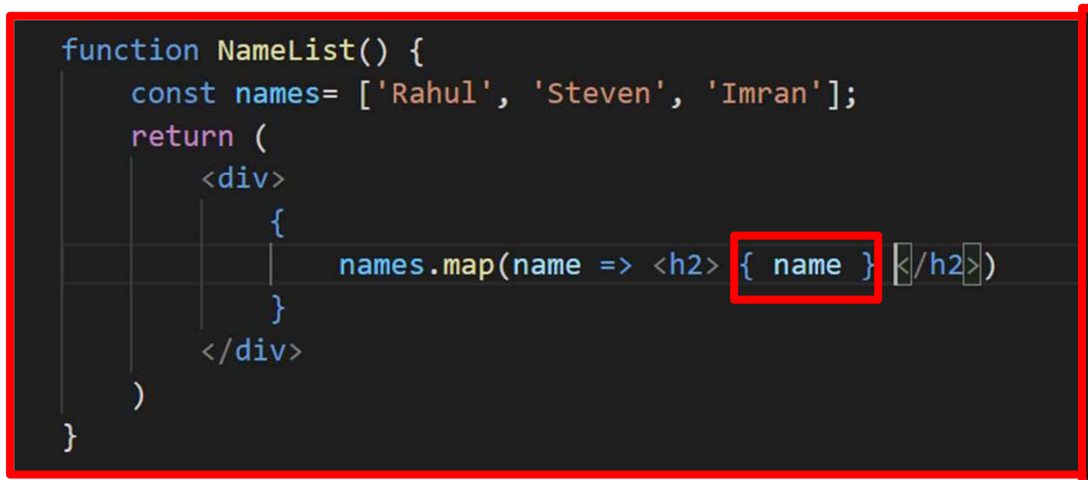
Imran



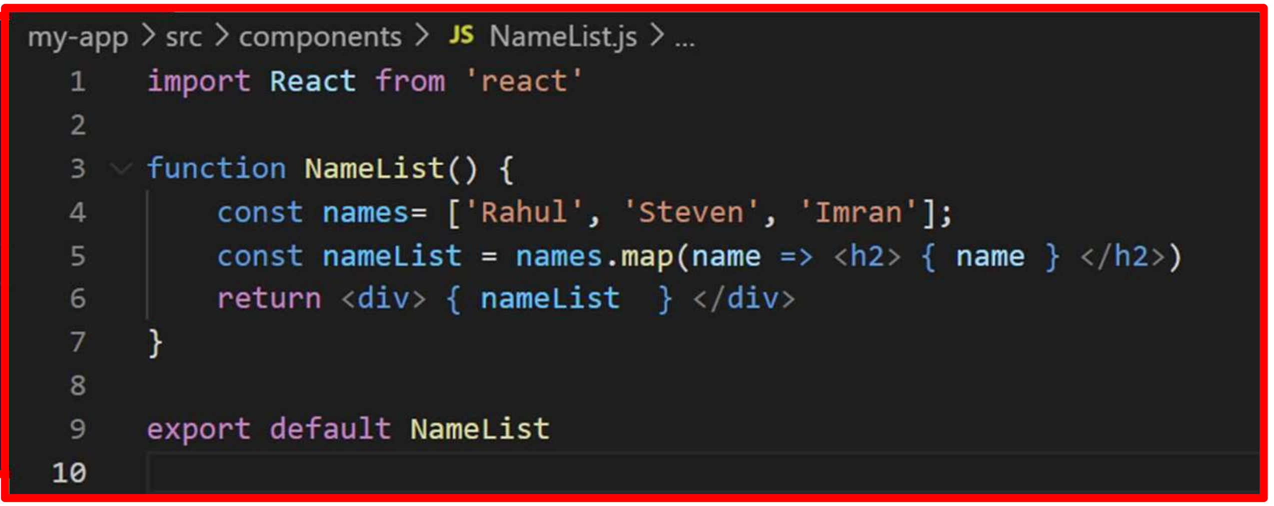
```
import NameList from './components/NameList';
class App extends Component {
  render() {
    return (
      <div className="App">

        <NameList />
```

Using Map Method to Render the Names

- Remember the map method is JavaScript code which needs to be evaluated in curly braces.
- Once you start writing HTML you need to reuse curly braces if you have to evaluate the JavaScript expression so even though we do already have a pair of outer curly braces we need another pair for the name parameter



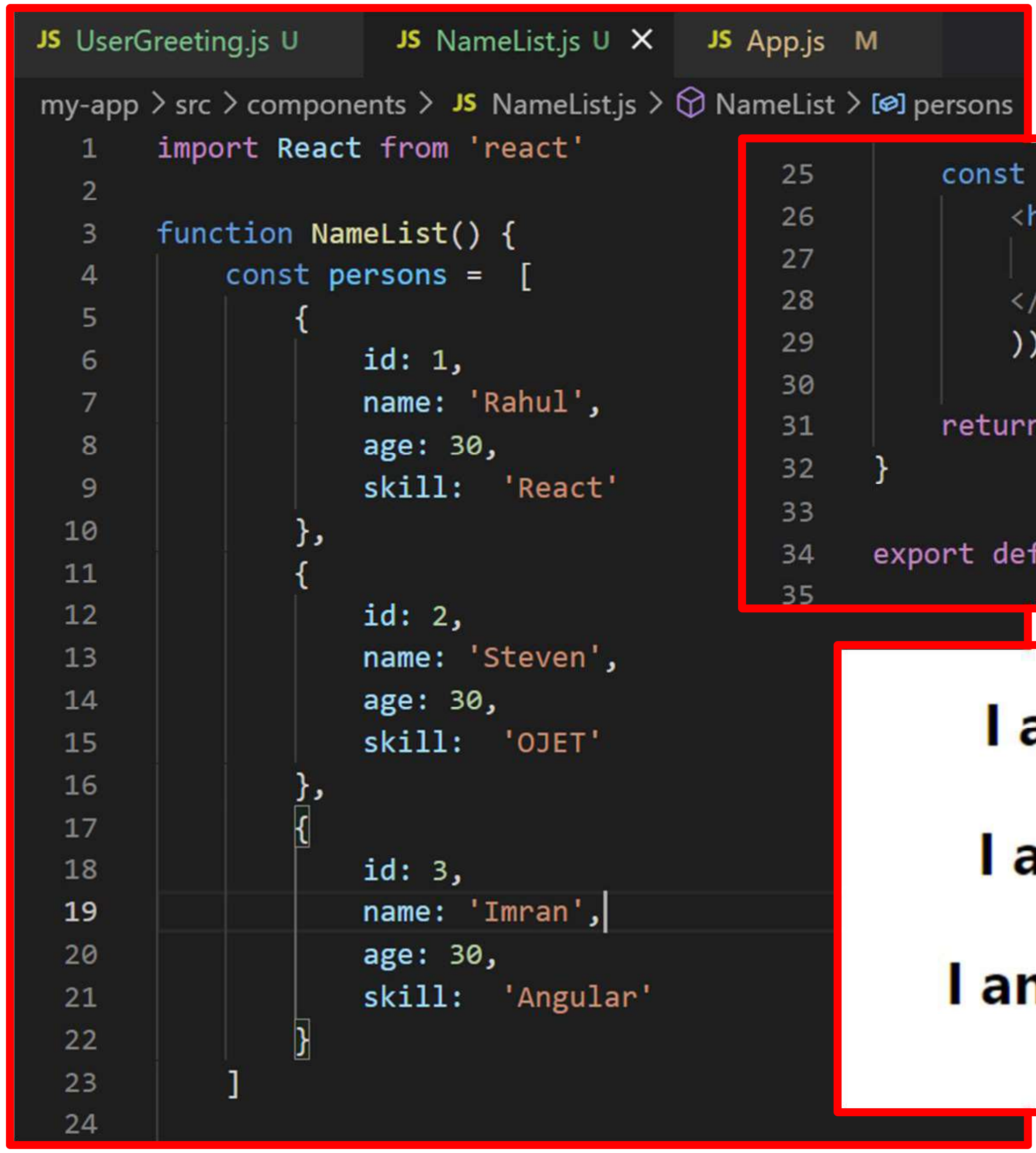
```
function NameList() {
    const names= ['Rahul', 'Steven', 'Imran'];
    return (
        <div>
            {
                names.map(name => <h2> { name } </h2>)
            }
        </div>
    )
}
```



my-app > src > components > JS NameList.js > ...

```
import React from 'react'

function NameList() {
    const names= ['Rahul', 'Steven', 'Imran'];
    const nameList = names.map(name => <h2> { name } </h2>)
    return <div> { nameList  } </div>
}

export default NameList
```

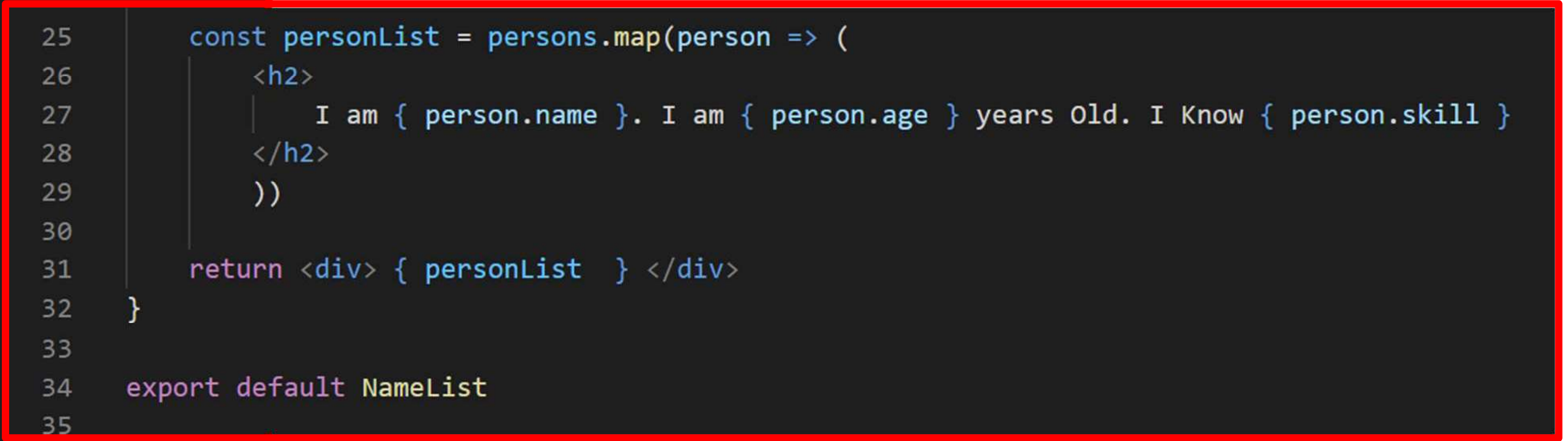
Extending the Functionality by Increasing the Complexity



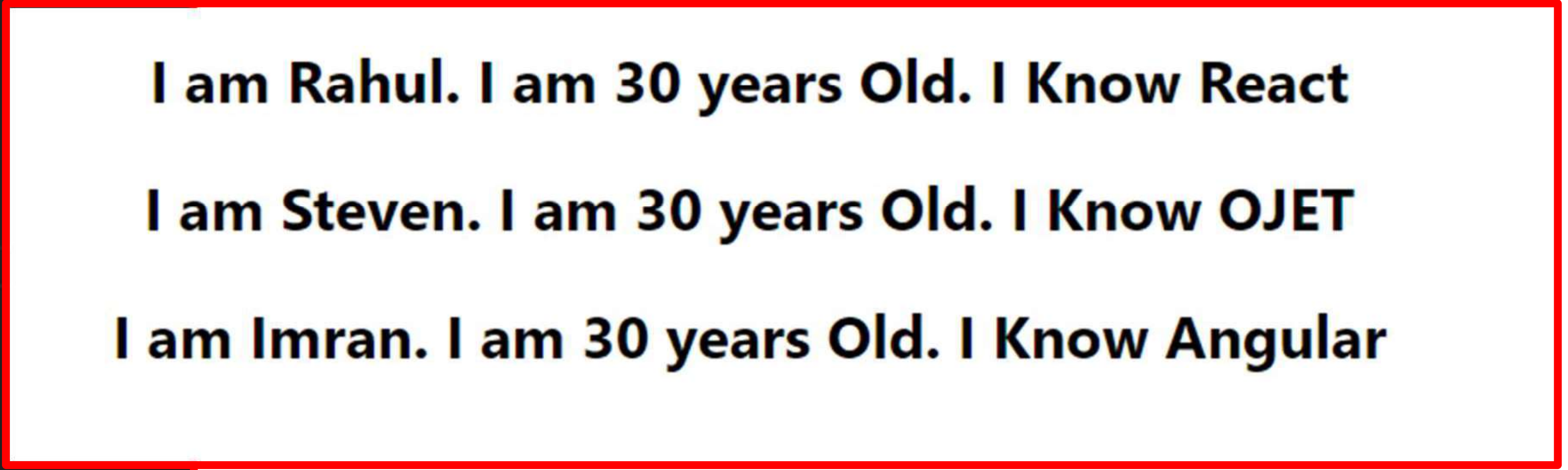
UserGreeting.js U   NameList.js U   App.js M

my-app > src > components > NameList.js > NameList > persons

```
import React from 'react'

function NameList() {
    const persons =  [
        {
            id: 1,
            name: 'Rahul',
            age: 30,
            skill:  'React'
        },
        {
            id: 2,
            name: 'Steven',
            age: 30,
            skill:  'OJET'
        },
        {

            id: 3,
            name: 'Imran',
            age: 30,
            skill:  'Angular'
        }
    ]

```



```
    const personList = persons.map(person => (
        <h2>
            I am { person.name }. I am { person.age } years Old. I Know { person.skill }
        </h2>
        ))

    return <div> { personList  } </div>
}

export default NameList

```

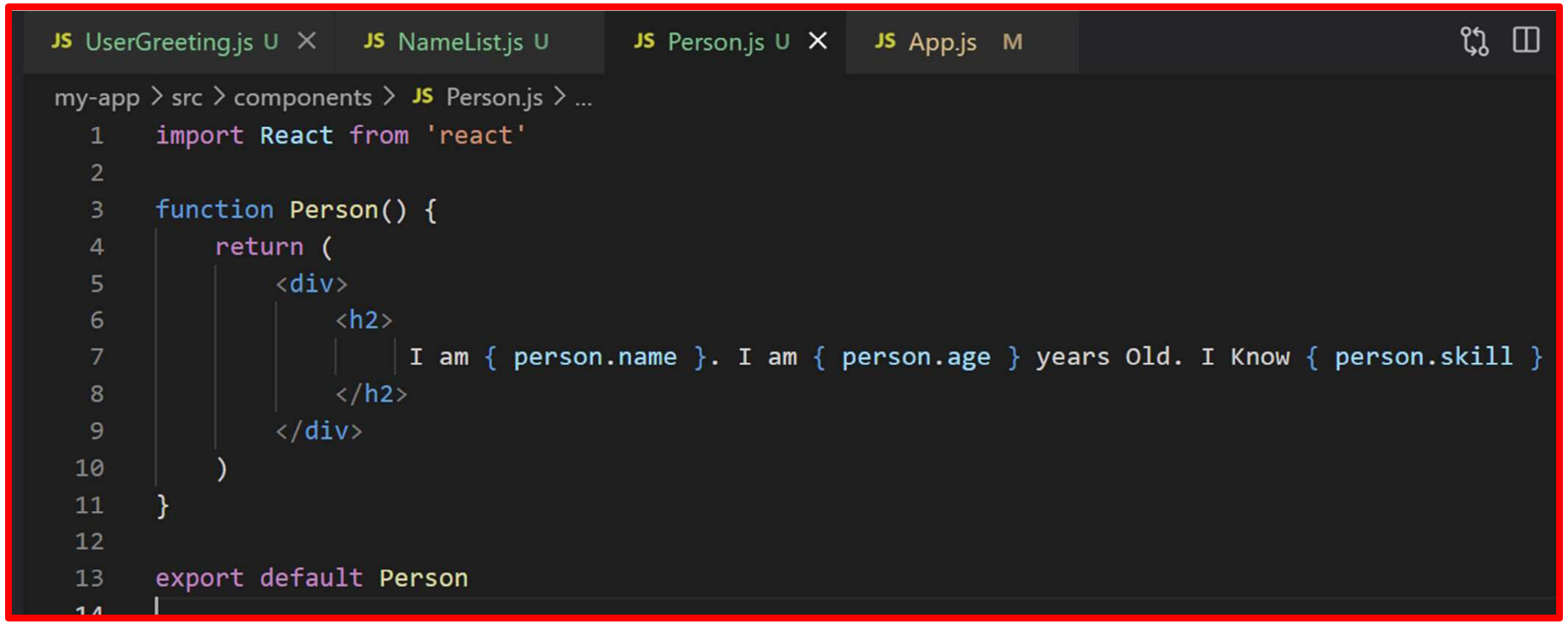


I am Rahul. I am 30 years Old. I Know React

I am Steven. I am 30 years Old. I Know OJET

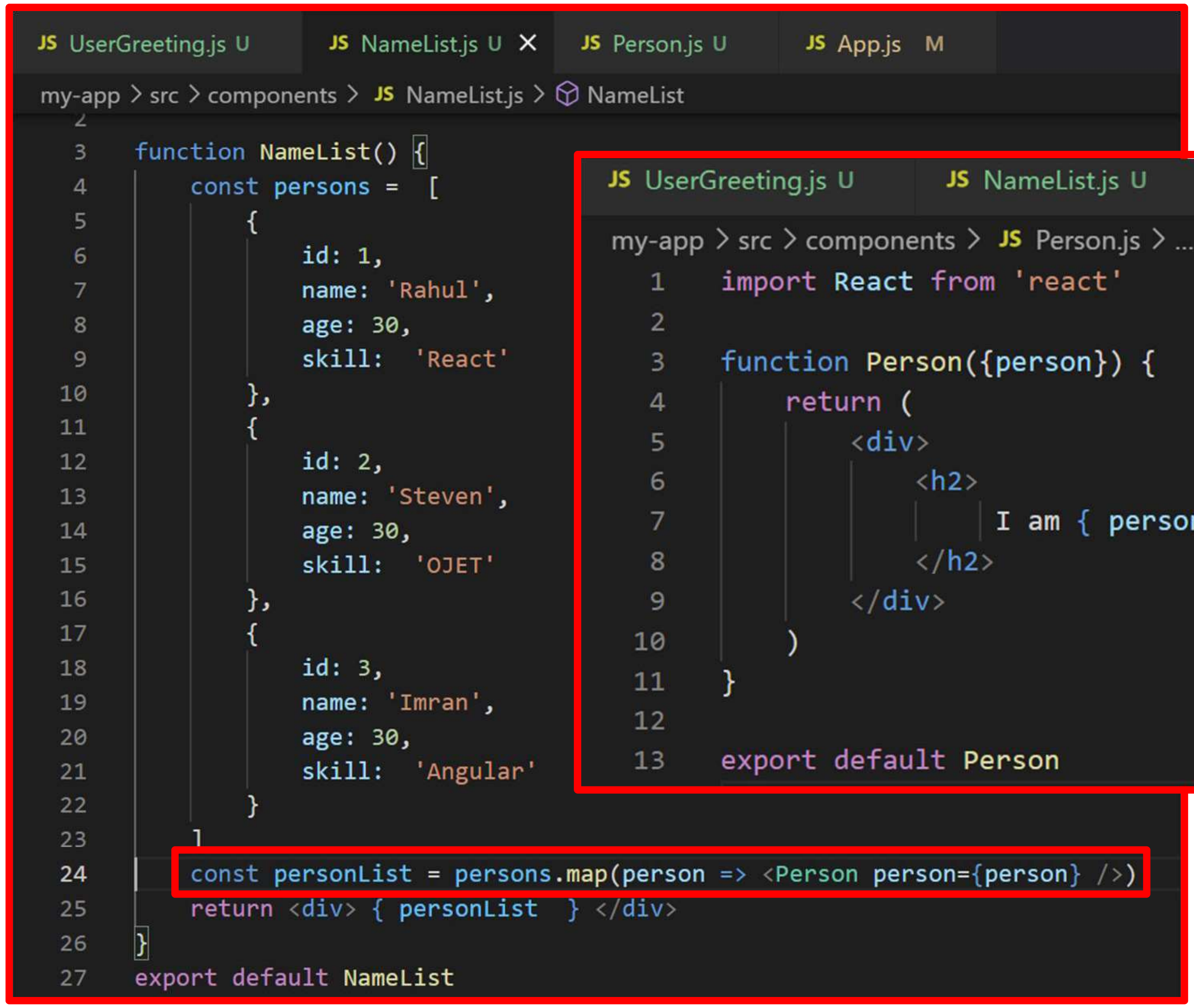
I am Imran. I am 30 years Old. I Know Angular

- There is nothing wrong with the code but the recommended way is to refactor the JSX **Refactor** into a separate component



```
import React from 'react'

function Person() {
    return (
        <div>
            <h2>
                I am { person.name }. I am { person.age } years Old. I Know { person.skill }
            </h2>
        </div>
    )
}

export default Person
```

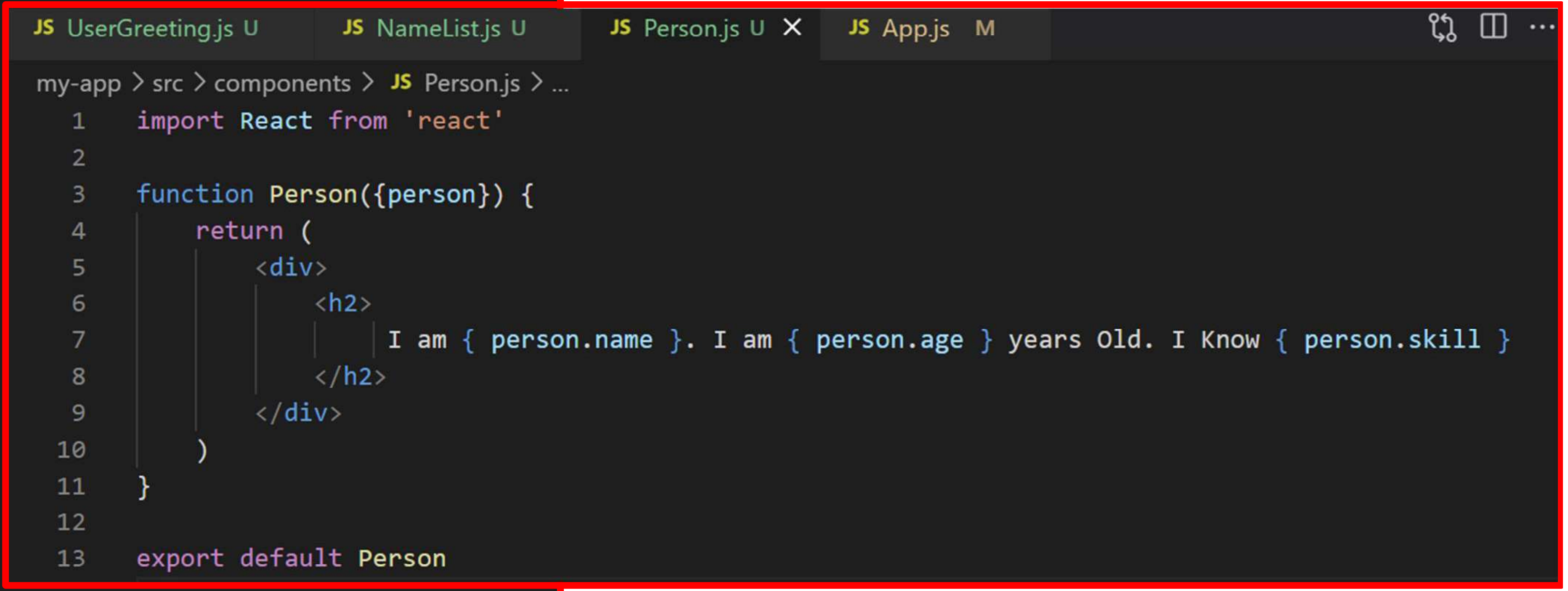
In the List Component lets pass Data to the Person Component using props



JS UserGreeting.js U   JS NameList.js U ×   JS Person.js U   JS App.js M

my-app > src > components > JS NameList.js > NameList

```
function NameList() {
    const persons =  [
        {
            id: 1,
            name: 'Rahul',
            age: 30,
            skill:  'React'
        },
        {
            id: 2,
            name: 'Steven',
            age: 30,
            skill:  'OJET'
        },
        {
            id: 3,
            name: 'Imran',
            age: 30,
            skill:  'Angular'
        }
    ]
    const personList = persons.map(person => <Person person={person} />)
    return <div> { personList  } </div>
}
export default NameList
```

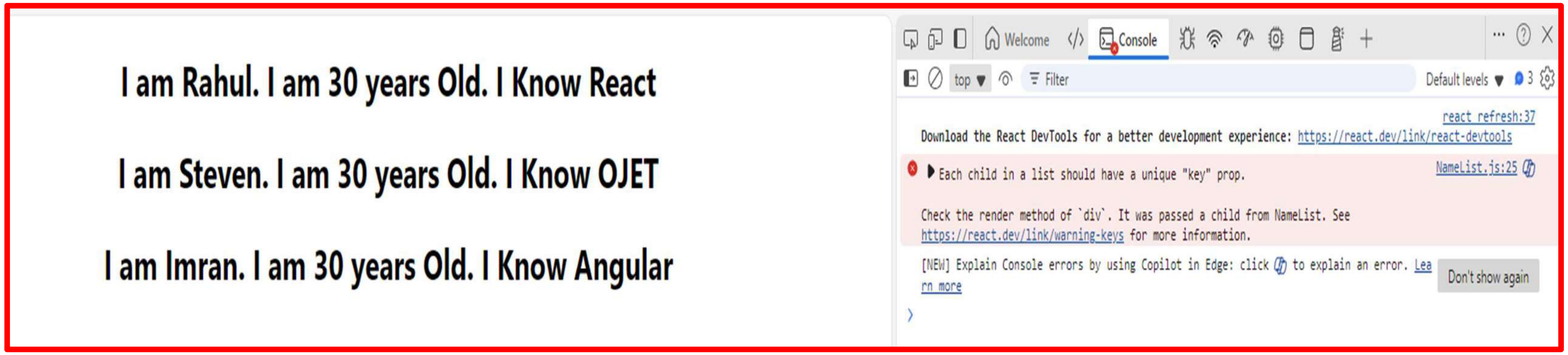


JS UserGreeting.js U   JS NameList.js U   JS Person.js U ×   JS App.js M

my-app > src > components > JS Person.js > ...

```
import React from 'react'

function Person({person}) {
    return (
        <div>
            <h2>
                I am { person.name }. I am { person.age } years Old. I Know { person.skill }
            </h2>
        </div>
    )
}

export default Person
```

But We have a Problem


I am Rahul. I am 30 years Old. I Know React

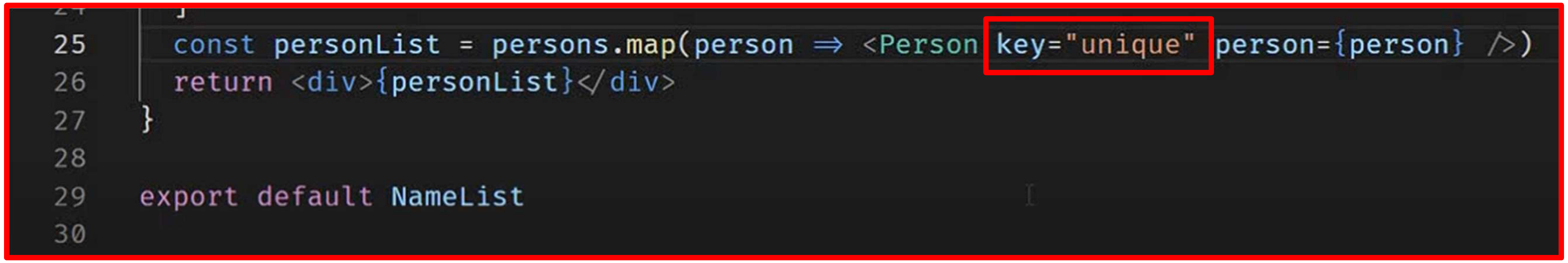
I am Steven. I am 30 years Old. I Know OJET

I am Imran. I am 30 years Old. I Know Angular

```
                                                                          react_refresh:37
Download the React DevTools for a better development experience: https://react.dev/link/react-devtools
▶ Each child in a list should have a unique "key" prop.                  NameList.js:25

Check the render method of `div`. It was passed a child from NameList. See
https://react.dev/link/warning-keys for more information.
[NEW] Explain Console errors by using Copilot in Edge: click  to explain an error. Lea
rn more
```

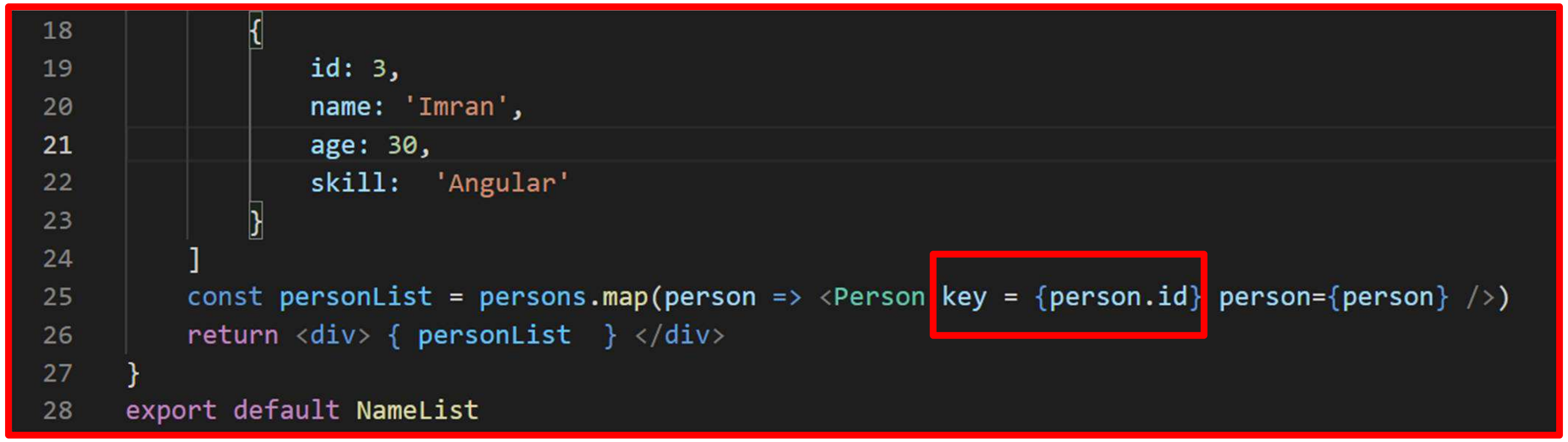
List & Keys

- In the console each child in an array or iterator should have a unique key prop or in simpler words react is telling us hey each item in the list rendered using the map operator should have a prop called key and the value to that prop should be unique within the List

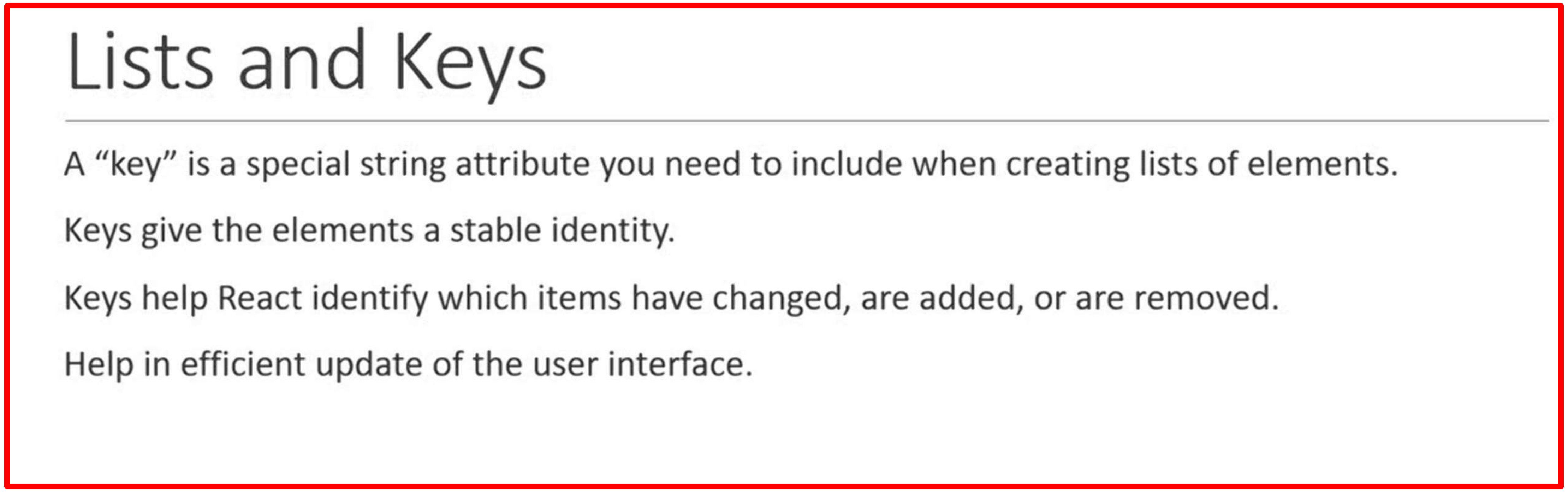


```
  const personList = persons.map(person => <Person key="unique" person={person} />)
  return <div>{personList}</div>
}

export default NameList
```

- Typically the ID of the item is a great choice for the key prop value you can see that every Person has an ID property whose value is unique within the list so we can assign Person.ID as the value to the key prop so key is going to be equal to within curly braces Person.ID



```
        {
            id: 3,
            name: 'Imran',
            age: 30,
            skill:  'Angular'
        }
    ]
    const personList = persons.map(person => <Person key = {person.id} person={person} />)
    return <div> { personList  } </div>
}
export default NameList
```



Lists and Keys

A “key” is a special string attribute you need to include when creating lists of elements.

Keys give the elements a stable identity.

Keys help React identify which items have changed, are added, or are removed.

Help in efficient update of the user interface.

Summary

In this lesson, you should have learned how to:

- Basics in State
- setState
- State, Events and Managed Controls

MENTORLABS℠