

LECTURE #4

By: Aqib Rehman

OUTLINE

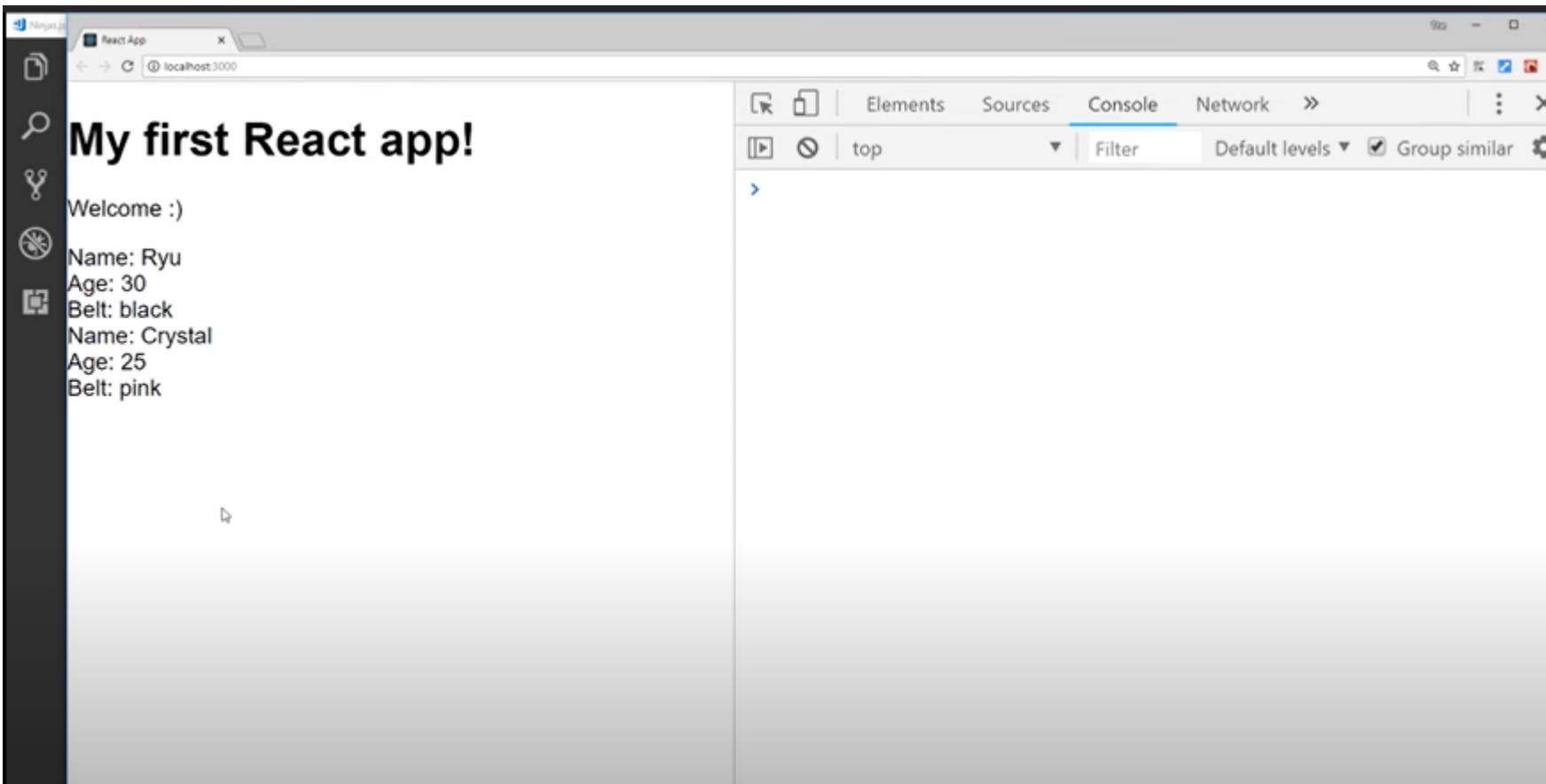
Conditional Output

Forms Revisited

Functions as Props

CONDITIONAL OUTPUT

```
1 import React from 'react';
2
3 const Ninjas = ({ninjas}) => {
4   const ninjaList = ninjas.map(ninja => {
5     if (ninja.age > 20){
6       return (
7         <div className="ninja" key={ninja.id}>
8           <div>Name: { ninja.name }</div>
9           <div>Age: { ninja.age }</div>
10          <div>Belt: { ninja.belt }</div>
11        </div>
12      )
13    } else {
14      return null
15    }
16  })
17  return(
18    <div className="ninja-list">
19      { ninjaList }
20    </div>
21  )
22}
```



```
1 import React from 'react';
2
3 const Ninjas = ({ninjaList}) => {
4     // const ninjaList = ninjas.map(ninja => {
5     //     if (ninja.age > 20){
6     //         return (
7     //             <div className="ninja" key={ninja.id}>
8     //                 <div>Name: { ninja.name }</div>
9     //                 <div>Age: { ninja.age }</div>
10    //                 <div>Belt: { ninja.belt }</div>
11    //             </div>
12    //         )
13    //     } else {
14    //         return null
15    //     }
16    // })
17    const ninjaList = ninjas.map(ninja => {
18        condition ? () : ()
19    })
20    return(
21        <div className="ninja-list">
22            { ninjaList }
23        </div>
24    )
25}
```

```
1 import React from 'react';
2
3 const Ninjas = ({ninjas}) => {
4   // const ninjaList = ninjas.map(ninja => {
5   //   if (ninja.age > 20){
6   //     return (
7   //       <div className="ninja" key={ninja.id}>
8   //         <div>Name: { ninja.name }</div>
9   //         <div>Age: { ninja.age }</div>
10    //        <div>Belt: { ninja.belt }</div>
11    //      </div>
12    //    )
13    //  } else {
14    //    return null
15    //  }
16  // })
17  const ninjaList = ninjas.map(ninja => {
18    return ninja.age > 20 ? () : null
19  })
20  return(
21    <div className="ninja-list">
22      { ninjalist }
23    </div>
24  )}
```

The screenshot shows a Visual Studio Code window with a dark theme. The title bar reads "Ninjas.js - react-redux-complete - Visual Studio Code". The left sidebar has icons for file operations, search, and other settings. The main editor area contains the following code:

```
1 import React from 'react';
2
3 const Ninjas = ({ninja}) => {
4     // const ninjaList = ninjas.map(ninja => {
5     //     if (ninja.age > 20){
6     //         return (
7     //             <div className="ninja" key={ninja.id}>
8     //                 <div>Name: { ninja.name }</div>
9     //                 <div>Age: { ninja.age }</div>
10                <div>Belt: { ninja.belt }</div>
11            </div>
12        )
13    } else {
14        return null
15    }
16 })
17 const ninjaList = ninjas.map(ninja => {
18     return ninja.age > 20 ? (
19         <div className="ninja" key={ninja.id}>
20             <div>Name: { ninja.name }</div>
21             <div>Age: { ninja.age }</div>
22             <div>Belt: { ninja.belt }</div>
23         </div>
24     ) : null;
25 });
26 return(
```

The screenshot shows a Visual Studio Code interface with two tabs open: "App.js" and "Ninjas.js". The "App.js" tab is active, displaying the following code:

```
13 // ...
14 // ...
15 // ...
16 // ...
17 const ninjaList = ninjas.map(ninja => {
18   return ninja.age > 20 ? (
19     <div className="ninja" key={ninja.id}>
20       <div>Name: { ninja.name }</div>
21       <div>Age: { ninja.age }</div>
22       <div>Belt: { ninja.belt }</div>
23     </div>
24   ) : null;
25 );
26
27 return(
28   <div const ninjaList: any=">
29     { ninjaList }
30   </div>
31 )
32 }
33
34 export default Ninjas
```

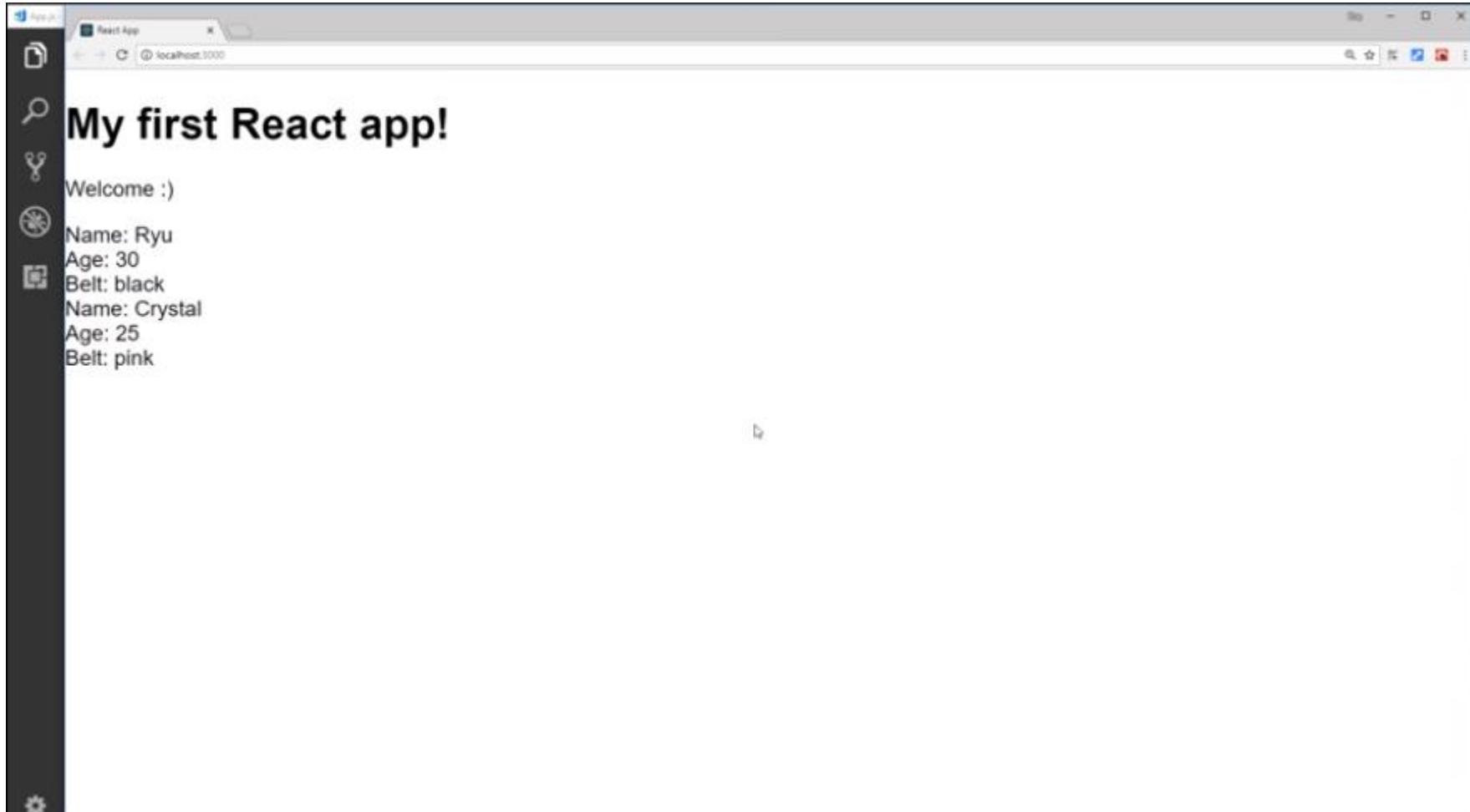
The screenshot shows a Visual Studio Code interface with the title bar "Ninjas.js - react-redux-complete - Visual Studio Code". The left sidebar contains icons for file operations like Open, Save, Find, Replace, and Undo. The main editor area displays the following code:

```
// if empty
14 //     return null
15 //
16 // })
17
18
19 return(
20     <div className="ninja-list">
21         { ninjaList }
22     </div>
23 )
24
25
26 export default Ninjas
```

A screenshot of the Visual Studio Code interface showing the file `Ninjas.js`. The code is a component that maps over an array of ninjas, filtering out those aged 20 or older and rendering their details in a list.

```
// ...  
14 //     return null  
15 // }  
16 // })  
17  
18  
19 return(  
20     <div className="ninja-list">  
21         {  
22             ninjas.map(ninja => {  
23                 return ninja.age > 20 ? (  
24                     <div className="ninja" key={ninja.id}>  
25                         <div>Name: { ninja.name }</div>  
26                         <div>Age: { ninja.age }</div>  
27                         <div>Belt: { ninja.belt }</div>  
28                     </div>  
29                 ) : null;  
30             })  
31         }  
32     </div>  
33     )  
34 }  
35  
36 export default Ninjas
```

FORMS REVISITED



What we have already

WHAT WE GONNA DO

A form in which user can add new ninja of his/her choice.

Programmatically how?

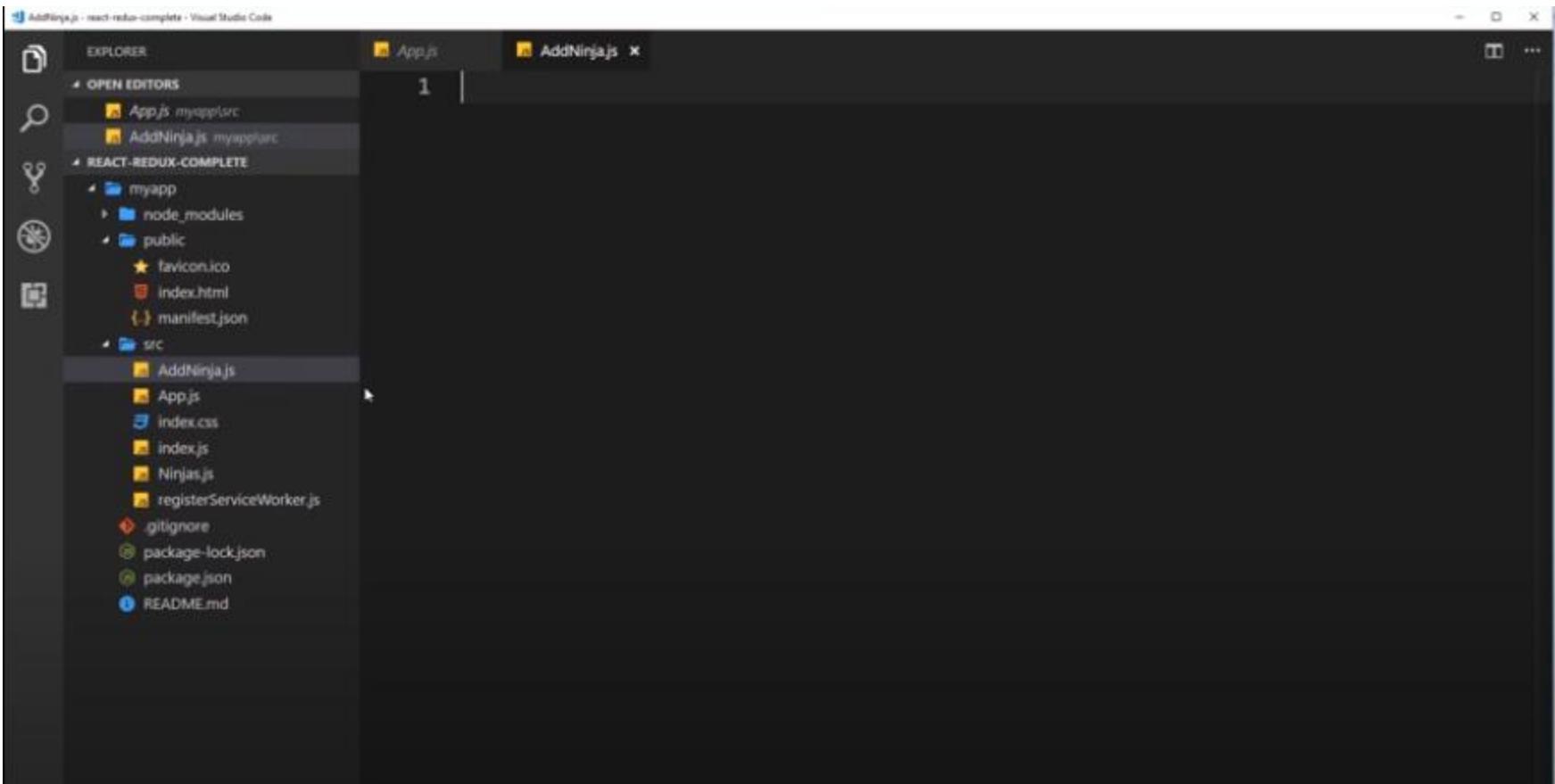
What if by some method we update the ninjas list which we have defined in state of App comp.

This list passes as props to Ninjas comp, where it is iterate and show on the screen.

So when list will be updated the updated one will be sent to Ninjas component and that will be shown to the user.

So lets create a form with 3 fields for name, age, belt and add button.

Create a new component with name AddNinja



What will be type of this comp container or stateless?

As we want to store user input values locally in the state so container comp will be used

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** AddNinja.js - react-redux-complete - Visual Studio Code
- Explorer View:** Shows the project structure:
 - OPEN EDITORS: App.js (myapp/src), AddNinjas.js (myapp/src)
 - REACT-REDUX-COMPLETE: myapp (containing node_modules, public, and src folders), src (containing AddNinjas.js, App.js, index.css, index.js, Ninjas.js, registerServiceWorker.js, gitignore, package-lock.json, package.json, and README.md)
- Editor View:** The file AddNinjas.js is open, displaying the following code:

```
1 import React, { Component } from 'react'
2
3 class AddNinja extends Component {
4   render(){
5     return (
6       <div>
7         <form>
8           <label htmlFor="name">Name:</label>
9           <input type="text" id="name" onChange={} />
10          <label htmlFor="name">Age:</label>
11          <input type="text" id="age" onChange={} />
12          <label htmlFor="name">Belt:</label>
13          <input type="text" id="belt" onChange={} />
14          <button>Submit</button>
15        </form>
16      </div>
17    )
18  }
19 }
```

htmlFor in jsx template not for

The screenshot shows a Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure. The root folder is "myapp". Inside "myapp" are "node_modules", "public" (containing "favicon.ico" and "index.html"), and "manifest.json". A "src" folder contains "AddNinja.js", "App.js", "index.css", "index.js", "Ninjas.js", and "registerServiceWorker.js". There are also ".gitignore", "package-lock.json", "package.json", and "README.md" files.
- Open Editors:** Shows two tabs: "App.js" and "AddNinjas.js".
- AddNinjas.js Content:** The code defines a class "AddNinja" that extends "Component". It has a state object with properties "name", "age", and "belt", all initially set to null. The "render" method returns a form with three text input fields and a submit button, each with an "onChange" event handler.

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

class AddNinja extends Component {
  state = {
    name: null,
    age: null,
    belt: null
  }
  render(){
    return (
      <div>
        <form>
          <label htmlFor="name">Name:</label>
          <input type="text" id="name" onChange={} />
          <label htmlFor="name">Age:</label>
          <input type="text" id="age" onChange={} />
          <label htmlFor="name">Belt:</label>
          <input type="text" id="belt" onChange={} />
          <button>Submit</button>
        </form>
      </div>
    )
  }
}
```

Define state

Need 3 functions for each field.

Right?

No

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure under "REACT-REDUX-COMPLETE".
 - myapp:** Contains node_modules, public (with favicon.ico and index.html), manifest.json, and src.
 - src:** Contains App.js, AddNinja.js (selected), index.css, index.js, Ninjas.js, and registerServiceWorker.js.
- Open Editors (Top):** Shows "App.js" and "AddNinja.js".
- Code Editor (Right):** Displays the content of "AddNinja.js".

```
1 import React, { Component } from 'react'
2
3 class AddNinja extends Component {
4   state = {
5     name: null,
6     age: null,
7     belt: null
8   }
9   handleChange = (e) => {
10     this.setState({
11       [e.target.id]: e.target.value
12     })
13   }
14   render(){
15     return (
16       <div>
17         <form>
18           <label htmlFor="name">Name:</label>
19           <input type="text" id="name" onChange={} />
20           <label htmlFor="name">Age:</label>
21           <input type="text" id="age" onChange={} />
22           <label htmlFor="name">Belt:</label>
23         </form>
24       </div>
25     )
26   }
27 }
```

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

class AddNinja extends Component {
  state = {
    name: null,
    age: null,
    belt: null
  }
  handleChange = (e) => {
    this.setState({
      [e.target.id]: e.target.value
    })
  }
  render(){
    return (
      <div>
        <form>
          <label htmlFor="name">Name:</label>
          <input type="text" id="name" onChange={this.handleChange} />
          <label htmlFor="age">Age:</label>
          <input type="text" id="age" onChange={this.handleChange} />
          <label htmlFor="belt">Belt:</label>
          <input type="text" id="belt" onChange={this.handleChange} />
          <button>Submit</button>
        </form>
      </div>
    )
  }
}
```

Calling handleChange func

```
App.js  AddNinjas.js •
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```

```
    "name": null,
    "age": null,
    "belt": null
}

handleChange = (e) => {
  this.setState({
    [e.target.id]: e.target.value
  })
}

handleSubmit = (e) => {
  e.preventDefault();
  console.log(this.state);
}

render(){
  return (
    <div>
      <form onSubmit={this.handleSubmit}>
        <label htmlFor="name">Name:</label>
        <input type="text" id="name" onChange={this.handleChange} />
        <label htmlFor="age">Age:</label>
        <input type="text" id="age" onChange={this.handleChange} />
        <label htmlFor="belt">Belt:</label>
        <input type="text" id="belt" onChange={this.handleChange} />
      <button>Submit</button>
    </div>
  )
}
```

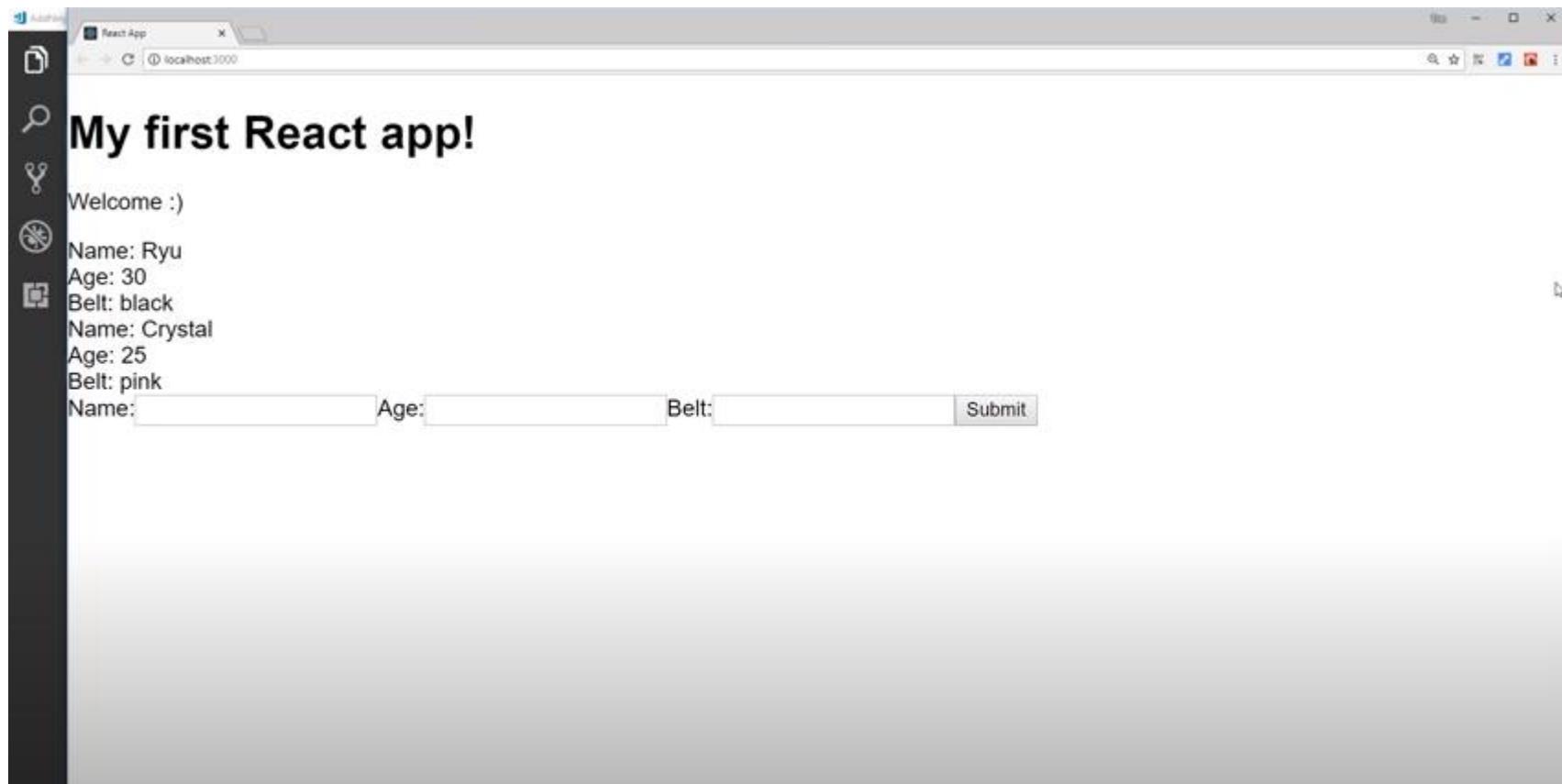
handleSubmit func

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** App.js - react-redux-complete - Visual Studio Code
- Explorer View (Left):** Shows the project structure:
 - OPEN EDITORS: App.js (myapp/src)
 - REACT-REDUX-COMPLETE: myapp (including node_modules, public, src, and various files like index.html, manifest.json, AddNinja.js, App.js, index.css, index.js, and registerServiceWorker.js), .gitignore, package-lock.json, package.json, and README.md.
- Code Editor (Right):** The App.js file content is displayed:

```
1 import React, { Component } from 'react';
2 import Ninjas from './Ninjas';
3 import AddNinja from './AddNinja';
4
5 class App extends Component {
6   state = {
7     ninjas : [
8       { name: 'Ryu', age: 30, belt: 'black', id: 1 },
9       { name: 'Yoshi', age: 20, belt: 'green', id: 2 },
10      { name: 'Crystal', age: 25, belt: 'pink', id: 3 }
11    ]
12  }
13  render() {
14    return (
15      <div className="App">
16        <h1>My first React app!</h1>
17        <p>Welcome :)</p>
18        <Ninjas ninjas={this.state.ninjas} />
19        <AddNinja />
20      </div>
21    );
22  }
}
```

Adding AddNinja comp in the root comp



My first React app!

Welcome :)

Name: Ryu

Age: 30

Belt: black

Name: Crystal

Age: 25

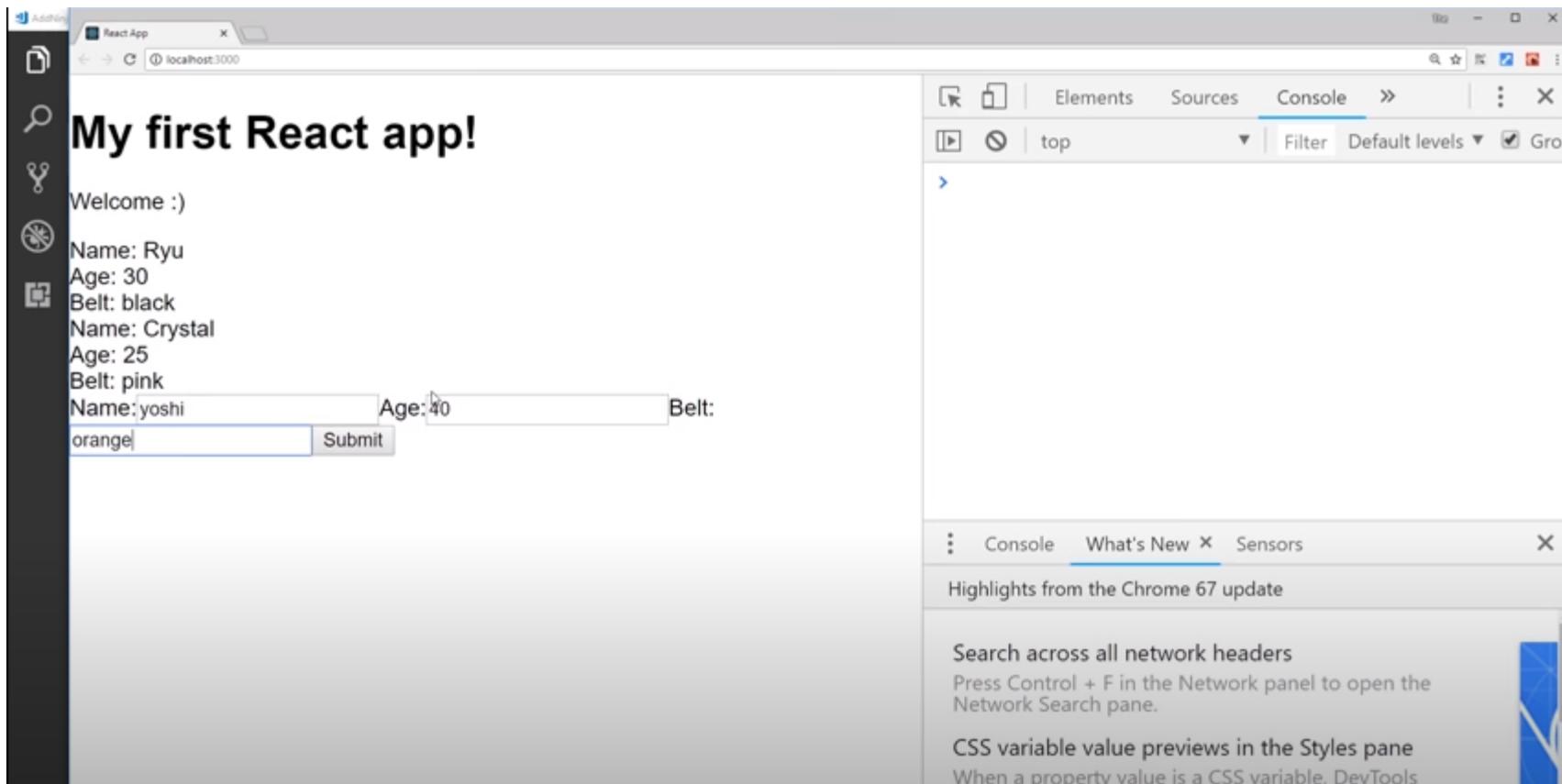
Belt: pink

Name:

Age:

Belt:

Submit



The screenshot shows a browser window with the title "React App" at "localhost:3000". The main content area displays the text "My first React app!" and a list of characters:

- Welcome :)
- Name: Ryu
- Age: 30
- Belt: black
- Name: Crystal
- Age: 25
- Belt: pink

Below this, there is a form with three input fields and a submit button:

- Name: yoshi
- Age: 40
- Belt: orange

The "Submit" button is located to the right of the "Belt" field.

In the bottom right corner of the browser window, there is a small "React" logo.

On the right side of the image, the Chrome DevTools are open. The "Console" tab is selected, showing the following log entry:

```
AddNinja.js:16
▶ {name: "yoshi", age: "40", belt: "orange"}
```

The "What's New" tab is also visible, displaying highlights from the Chrome 67 update:

- Search across all network headers
- Press Control + F in the Network panel to open the Network Search pane.
- CSS variable value previews in the Styles pane
- When a property value is a CSS variable, DevTools

Now what we need:

We need to add the stuff which we got in the last slide from the user to the nijas list which we mentioned in the start.

But how?

FUNCTIONS AS PROPS

The screenshot shows the Visual Studio Code interface with the following details:

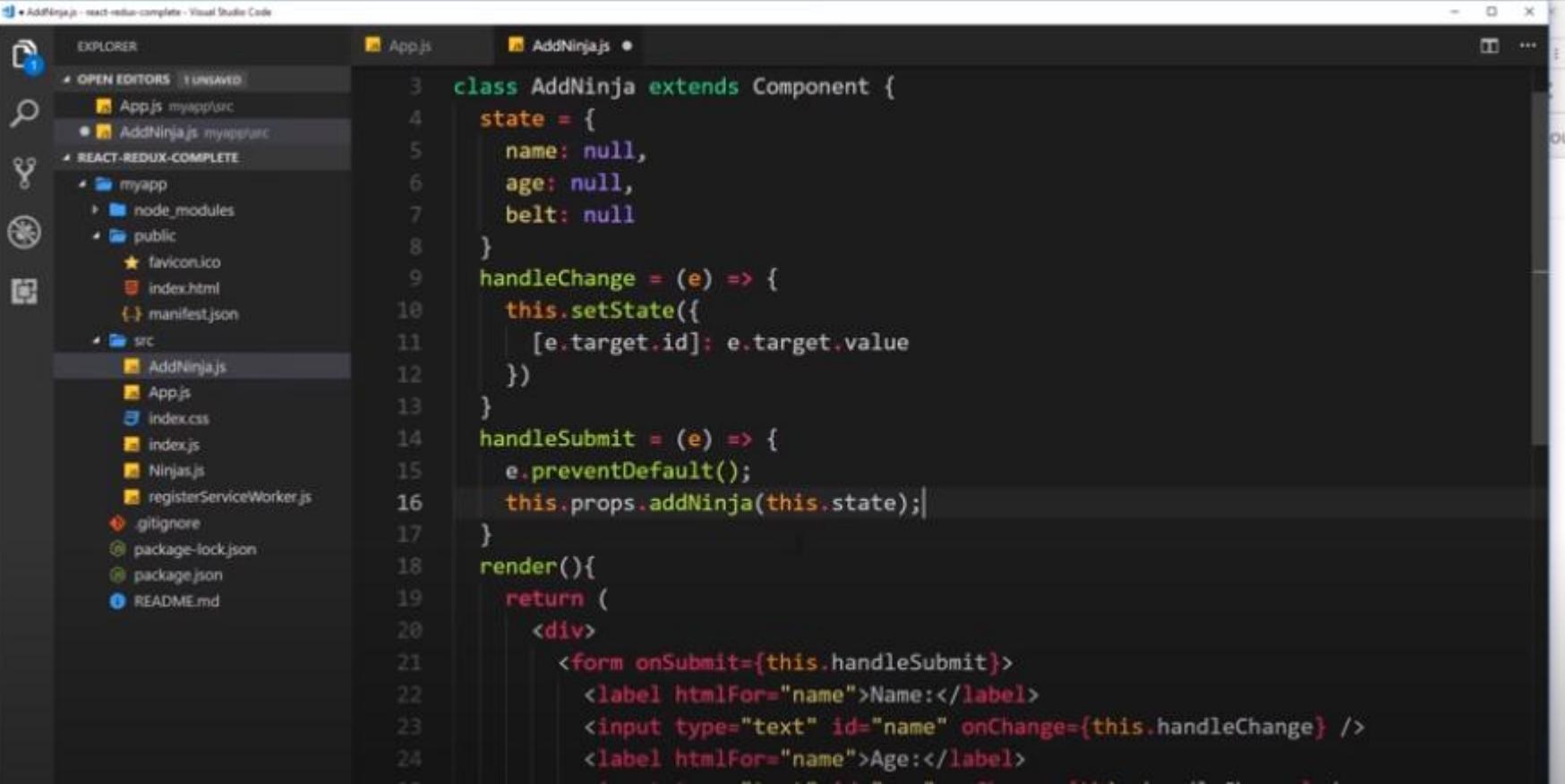
- Title Bar:** App.js - react-redux-complete - Visual Studio Code
- Explorer View (Left):** Shows the project structure:
 - OPEN EDITORS: App.js (myapp/src)
 - REACT-REDUX-COMPLETE: myapp (including node_modules, public, src, .gitignore, package-lock.json, package.json, README.md)
- Editor View (Center):** Displays the contents of App.js:

```
1 import React, { Component } from 'react';
2 import Ninjas from './Ninjas';
3 import AddNinja from './AddNinja';
4
5 class App extends Component {
6   state = {
7     ninjas : [
8       { name: 'Ryu', age: 30, belt: 'black', id: 1 },
9       { name: 'Yoshi', age: 20, belt: 'green', id: 2 },
10      { name: 'Crystal', age: 25, belt: 'pink', id: 3 }
11    ]
12  }
13  addNinja = (ninja) => {
14  }
15  render() {
16    return (
17      <div className="App">
18        <h1>My first React app!</h1>
19        <p>Welcome :)</p>
20        <Ninjas ninjas={this.state.ninjas} />
21        <AddNinja />
22    );
23  }
}
```

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** sidebar:
 - OPEN EDITORS: App.js (myappsrc), AddNinjas.js (myappsrc)
 - REACT-REDUX-COMPLETE: myapp (expanded)
 - node_modules
 - public
 - favicon.ico
 - index.html
 - manifest.json
 - src
 - AddNinjas.js
 - App.js (highlighted)
 - index.css
 - index.js
 - Ninjas.js
 - registerServiceWorker.js
 - .gitignore
 - package-lock.json
 - package.json
 - README.md
- EDITOR**: App.js (myappsrc) is the active editor, showing the following code:

```
1 import React, { Component } from 'react';
2 import Ninjas from './Ninjas';
3 import AddNinja from './AddNinja';
4
5 class App extends Component {
6   state = {
7     ninjas: [
8       { name: 'Ryu', age: 30, belt: 'black', id: 1 },
9       { name: 'Yoshi', age: 20, belt: 'green', id: 2 },
10      { name: 'Crystal', age: 25, belt: 'pink', id: 3 }
11    ]
12  }
13  addNinja = (ninja) => {
14  }
15  render() {
16    return (
17      <div className="App">
18        <h1>My first React app!</h1>
19        <p>Welcome :)</p>
20        <Ninjas ninjas={this.state.ninjas} />
21        <AddNinja addNinja={this.addNinja} />
22      </div>
23    );
24  }
25}
26
27export default App;
```



```
3  class AddNinja extends Component {
4    state = {
5      name: null,
6      age: null,
7      belt: null
8    }
9    handleChange = (e) => {
10      this.setState({
11        [e.target.id]: e.target.value
12      })
13    }
14    handleSubmit = (e) => {
15      e.preventDefault();
16      this.props.addNinja(this.state);
17    }
18    render(){
19      return (
20        <div>
21          <form onSubmit={this.handleSubmit}>
22            <label htmlFor="name">Name:</label>
23            <input type="text" id="name" onChange={this.handleChange} />
24            <label htmlFor="age">Age:</label>
```

Calling a func and passing a ninja obj through state

The screenshot shows the Visual Studio Code interface with a dark theme. The left sidebar displays the file structure of a React application named 'myapp'. The 'src' folder contains components like 'AddNinjas.js', 'App.js', and 'Ninjas.js'. The 'REACT-REDUX-COMPLETE' section is expanded, showing the contents of 'App.js'.

```
import React, { Component } from 'react';
import Ninjas from './Ninjas';
import AddNinja from './AddNinja';

class App extends Component {
  state = {
    ninjas: [
      { name: 'Ryu', age: 30, belt: 'black', id: 1 },
      { name: 'Yoshi', age: 20, belt: 'green', id: 2 },
      { name: 'Crystal', age: 25, belt: 'pink', id: 3 }
    ]
  }
  addNinja = (ninja) => {
    console.log(ninja);
  }
  render() {
    return (
      <div className="App">
        <h1>My first React App</h1>
        <p>Welcome :)</p>
        <Ninjas ninjas={this.state.ninjas} />
        <AddNinja addNinja={this.addNinja} />
      </div>
    );
  }
}
```

A code completion tooltip is visible over the 'ninja' parameter in the 'addNinja' function. It shows the type '(parameter) ninja: any' and lists suggestions: 'ninja' (with a small icon), 'Ninjas' (with a small icon), and 'AddNinja' (with a small icon).

The screenshot shows a browser window with the title "React App" at "localhost:3000". The main content area displays the text "My first React app!" followed by a list of characters: Ryu (Name: Ryu, Age: 30, Belt: black), Crystal (Name: Crystal, Age: 25, Belt: pink), and Mario (Name: mario, Age: 40, Belt: black). Below this list is a form with input fields for Name, Age, and Belt, and a Submit button. The developer tools' Console tab is open, showing the object {name: "mario", age: "40", belt: "black"} with a red "40" and "black", indicating they are being inspected. The file "App.js:14" is mentioned above the object.

```
▶ {name: "mario", age: "40", belt: "black"}  
App.js:14
```

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure:
 - myapp
 - node_modules
 - public
 - favicon.ico
 - index.html
 - manifest.json
 - src
 - AddNinja.js
 - App.js
 - index.css
 - index.js
 - Ninjas.js
 - registerServiceWorker.js
 - .gitignore
 - package-lock.json
 - package.json
 - README.md
- Editor (Center):** Displays the content of `App.js`:

```
1 import React, { Component } from 'react';
2 import Ninjas from './Ninjas';
3 import AddNinja from './AddNinja';
4
5 class App extends Component {
6   state = {
7     ninjas: [
8       { name: 'Ryu', age: 30, belt: 'black', id: 1 },
9       { name: 'Yoshi', age: 20, belt: 'green', id: 2 },
10      { name: 'Crystal', age: 25, belt: 'pink', id: 3 }
11    ]
12  }
13  addNinja = (ninja) => {
14    ninja.id = Math.random();
15  }
16  render() {
17    return (
18      <div className="App">
19        <h1>My first React app!</h1>
20        <p>Welcome :)</p>
21        <Ninjas ninjas={this.state.ninjas} />
22        <AddNinja addNinja={this.addNinja} />
23      </div>
24    );
25  }
26}
```

Adding id in ninja obj as we are not interested to take it from user

Not do this already updating array what of setState

The screenshot shows the Visual Studio Code interface with the file `App.js` open. The code is as follows:

```
1 import React, { Component } from 'react';
2 import Ninjas from './Ninjas';
3 import AddNinja from './AddNinja';
4
5 class App extends Component {
6   state = {
7     ninjas: [
8       { name: 'Ryu', age: 30, belt: 'black', id: 1 },
9       { name: 'Yoshi', age: 20, belt: 'green', id: 2 },
10      { name: 'Crystal', age: 25, belt: 'pink', id: 3 }
11    ]
12  }
13  addNinja = (ninja) => {
14    ninja.id = Math.random();
15    this.ninjas.push(ninja)
16    this.setState({
17      ninjas:
18    })
19  }
20  render() {
21    return (
22      <div className="App">
23        <h1>My first React app!</h1>
```

The line `this.ninjas.push(ninja)` is highlighted with a red squiggle underlining, indicating a potential error or anti-pattern. This is because pushing directly onto the state array does not trigger a re-render. Instead, the code should use the `setState` method to update the state.



Do not alter the state without setState

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** App.js - react-redux-complete - Visual Studio Code
- Explorer View:** Shows the project structure:
 - OPEN EDITORS: 1 UNSAVED
 - REACT-REDUX-COMPLETE
 - myapp
 - node_modules
 - public
 - favicon.ico
 - index.html
 - manifest.json
 - src
 - AddNinjas.js
 - App.js**
 - index.css
 - index.js
 - Ninjas.js
 - registerServiceWorker.js
 - .gitignore
 - package-lock.json
 - package.json
 - README.md
- Editor View:** The current file is App.js, which contains the following code:

```
1 import React, { Component } from 'react';
2 import Ninjas from './Ninjas';
3 import AddNinja from './AddNinja';
4
5 class App extends Component {
6   state = {
7     ninjas: [
8       { name: 'Ryu', age: 30, belt: 'black', id: 1 },
9       { name: 'Yoshi', age: 20, belt: 'green', id: 2 },
10      { name: 'Crystal', age: 25, belt: 'pink', id: 3 }
11    ]
12  }
13  addNinja = (ninja) => {
14    ninja.id = Math.random();
15    let ninjas = [...this.state.ninjas]
16    this.setState({
17      ninjas:
18    })
19  }
20  render() {
21    return (
22      <div className="App">
23        <h1>My first React app!</h1>
24        <p>Welcome :)</p>
25        <Ninjas ninjas={this.state.ninjas} />

```

Generate a copy

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<script>
```

```
const numbers = [1, 2, 3, 4, 5, 6];
```

```
const [one, two, ...rest] = numbers;
```

```
document.write("<p>" + one + "</p>");
```

```
document.write("<p>" + two + "</p>");
```

```
document.write("<p>" + rest + "</p>");
```

```
</script>
```

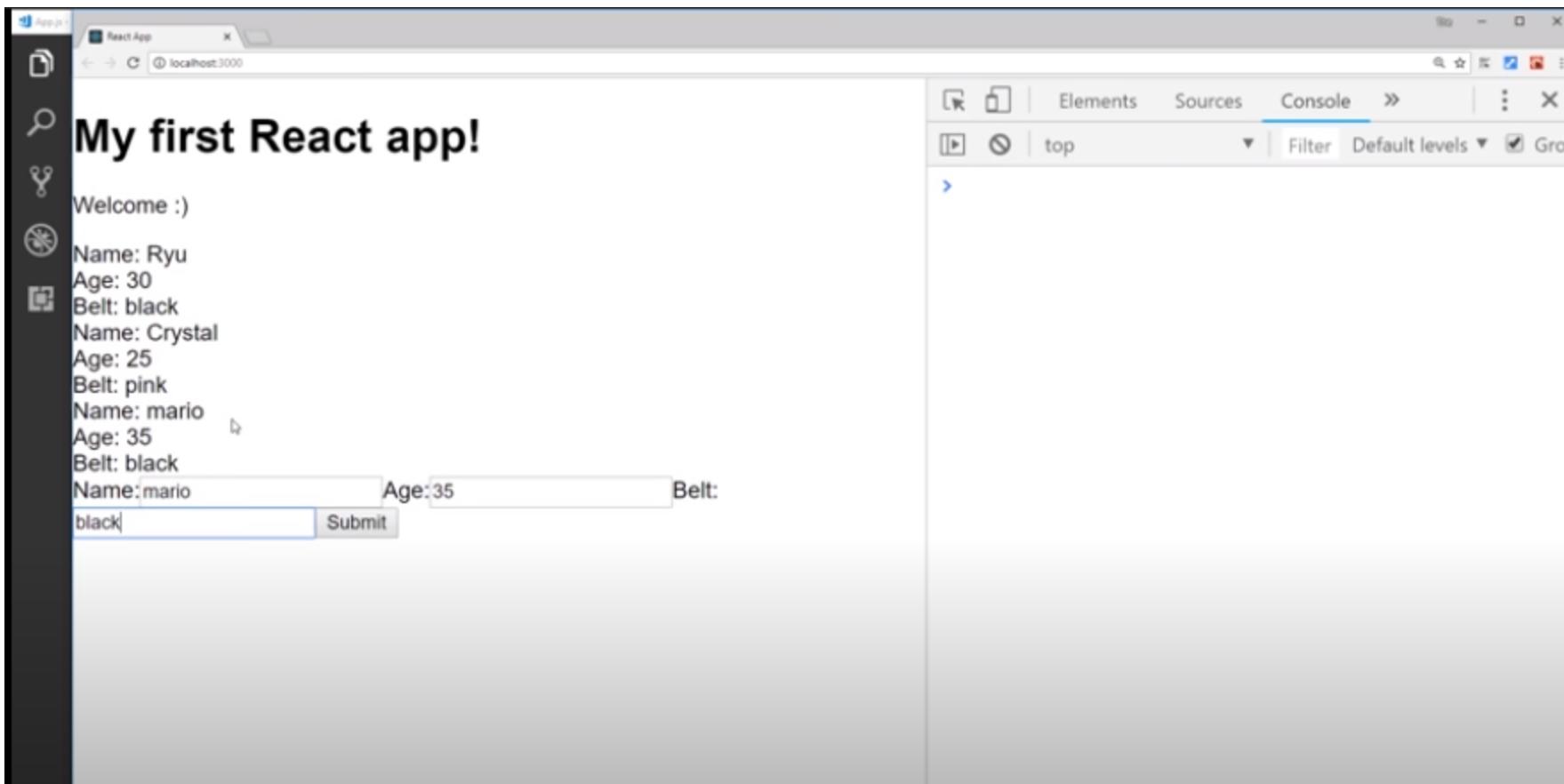
```
</body>
```

```
</html>
```

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** App.js - react-redux-complete - Visual Studio Code
- Explorer View (Left):** Shows the project structure:
 - OPEN EDITORS: App.js (myapp/src)
 - REACT-REDUX-COMPLETE: myapp (node_modules, public, src, .gitignore, package-lock.json, package.json, README.md)
- Code Editor (Right):** The current file is App.js, which contains the following code:

```
1 import React, { Component } from 'react';
2 import Ninjas from './Ninjas';
3 import AddNinja from './AddNinja';
4
5 class App extends Component {
6   state = {
7     ninjas: [
8       { name: 'Ryu', age: 30, belt: 'black', id: 1 },
9       { name: 'Yoshi', age: 20, belt: 'green', id: 2 },
10      { name: 'Crystal', age: 25, belt: 'pink', id: 3 }
11    ]
12  }
13  addNinja = (ninja) => {
14    ninja.id = Math.random();
15    let ninjas = [...this.state.ninjas, ninja];
16    this.setState({
17      ninjas: ninjas
18    })
19  }
20  render() {
21    return (
22      <div className="App">
```



Add delete Ninja functionality in current code