Hands-on Lab - React Application (1 hour 15 mins)

Objective for Exercise:

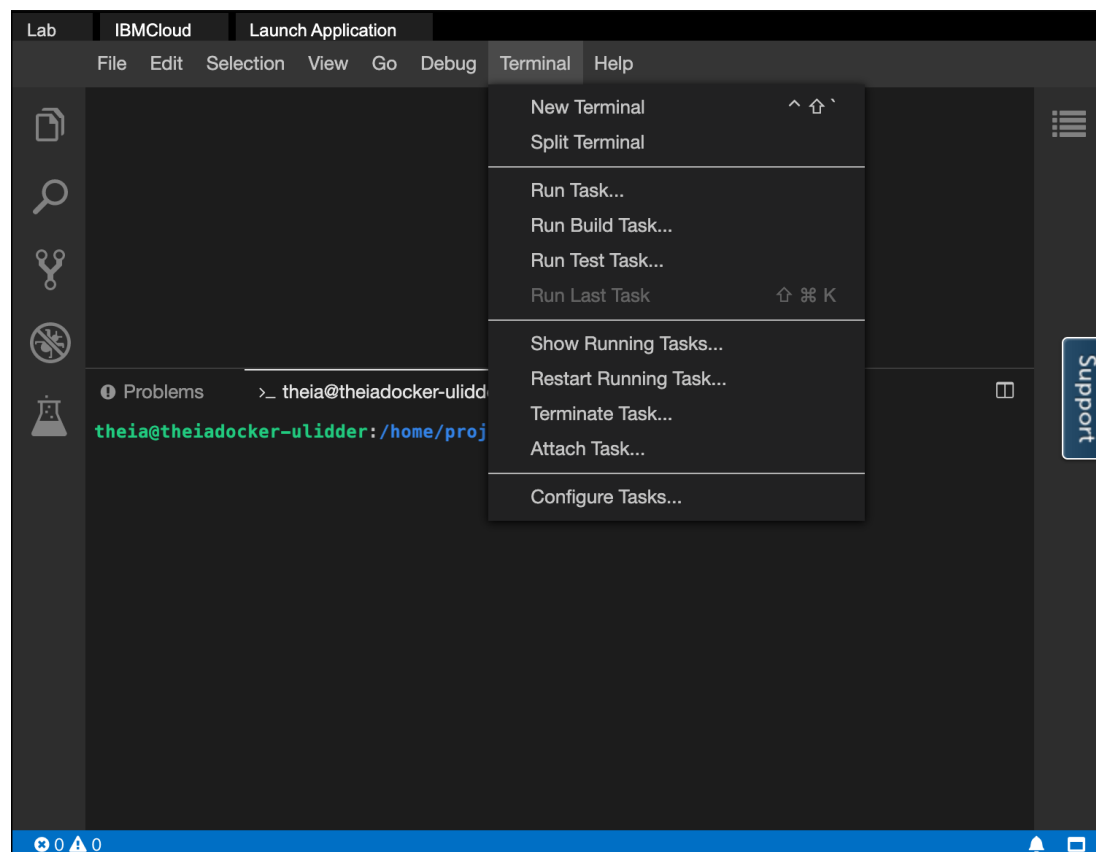By the end of this lab, create HTML Pages with React Components and React applications.

Prerequisites

Before you start, be sure that you meet these prerequisites:

- Basic knowledge of JavaScript
- Basic knowledge of HTML

Set-up : Check for the cloned files

1. Open a terminal window by using the menu in the editor: Terminal > New Terminal.

2. Change to your project folder.

```
1. 1
```

```
1. cd /home/project
```

Copied!

3. Check if you have the folder **lkpho-Cloud-applications-with-Node.js-and-React**

```
1. 1
```

```
1. ls /home/project
```

Copied!

If you do, you can skip to step 5.

4. Clone the git repository that contains the artifacts needed for this lab, if it doesn't already exist.

```
1. 1
```

1. `git clone https://github.com/ibm-developer-skills-network/lkpho-Cloud-applications-with-Node.js-and-React`

Copied!

5. Change to the directory for this lab.

1. 1

1. `cd lkpho-Cloud-applications-with-Node.js-and-React/CD220Labs/`

Copied!

6. List the contents of `react_pages`. This should contain HTML page(s) which you can download and view in your local system.

1. 1

1. `ls react_pages`

Copied!

7. List the contents of `todoapp`. This should contain a directory structure with a pre-installed react app.

1. 1

1. `ls todoapp`

Copied!

8. List the contents of `react-apps`. This should contain a directory structure with a pre-installed react app that you can change to apply what you have learned.
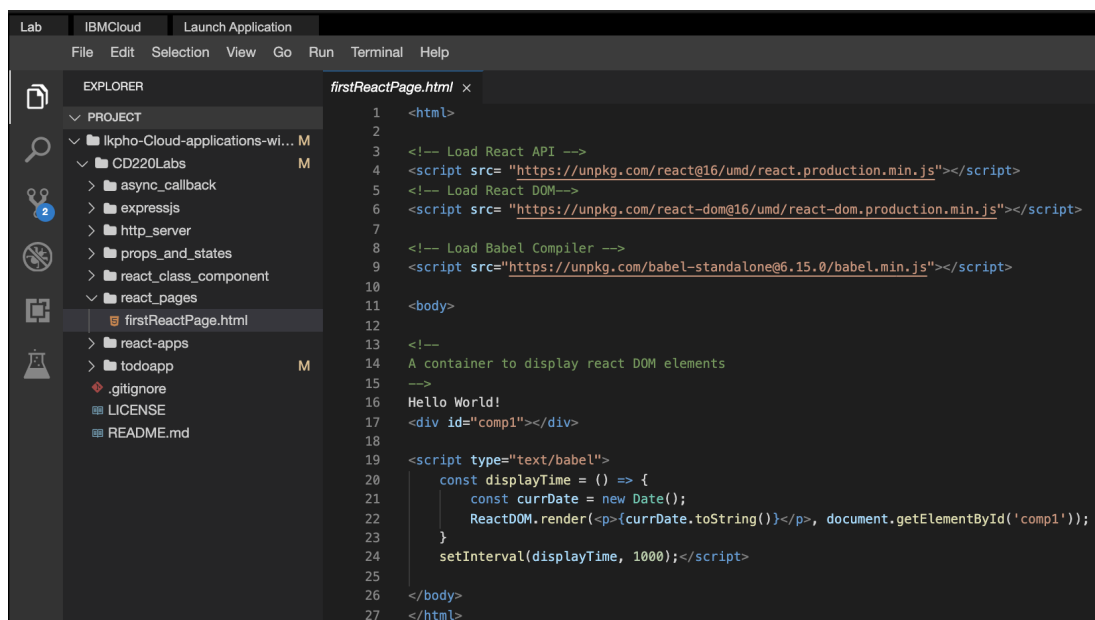
1. 1

1. `ls react-apps`

Copied!

These are the two directories you will be using for this lab.

Exercise 1: Simple HTML with React

1. In the file explorer goto `lkpho-Cloud-applications-with-Node.js-and-React`->`CD220Labs`->`react_pages` and view `firstReactPage.html`. The contents will appear as below. It is a HTML page where the clock is displayed through a react component.
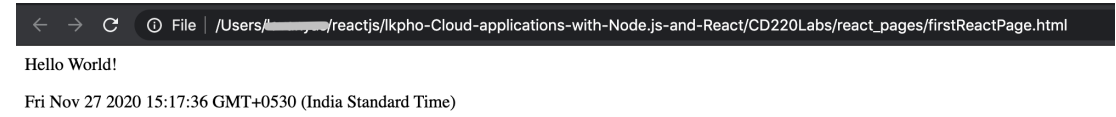

2. Right-click on `firstReactPage.html` and click on `Download`. This will download a copy of this file in your local system.
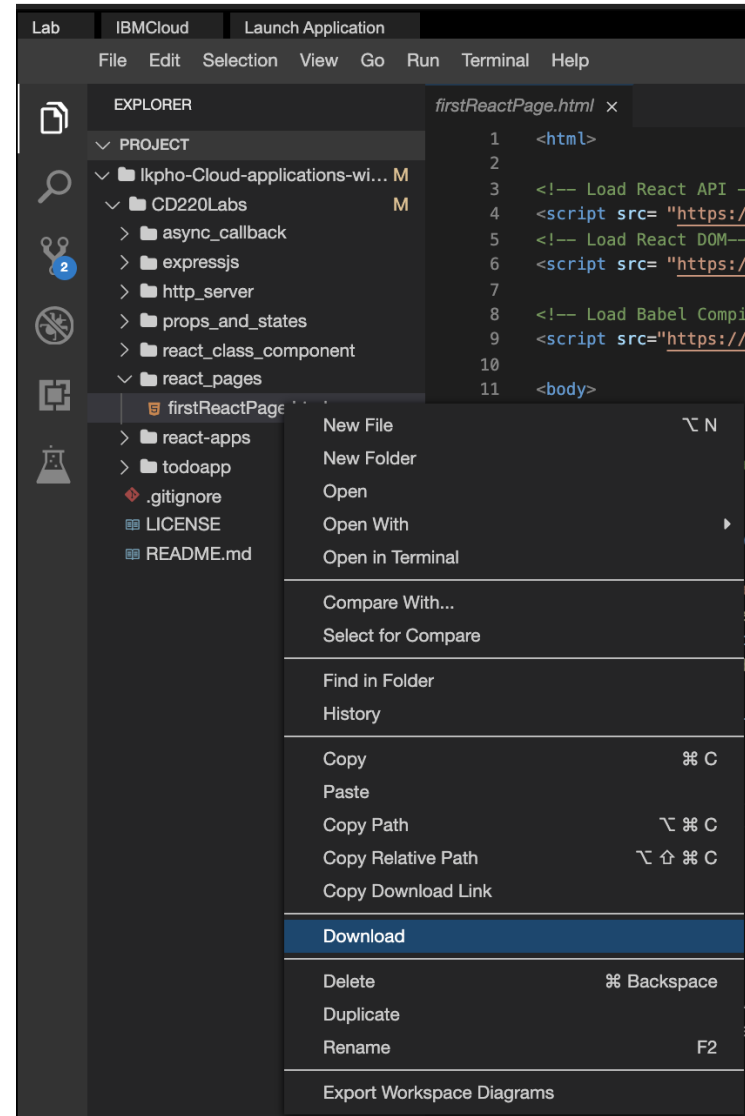
3. Now open the downloaded file in your local browser. The page will appear as below.

Hello World!

Fri Nov 27 2020 15:17:36 GMT+0530 (India Standard Time)

Exercise 2: First React Application

The `todoapp` is a react app that has been created for you using `npx create-react-app todoapp` command. In the files explorer view `todoapp` directory. It would appear like this.
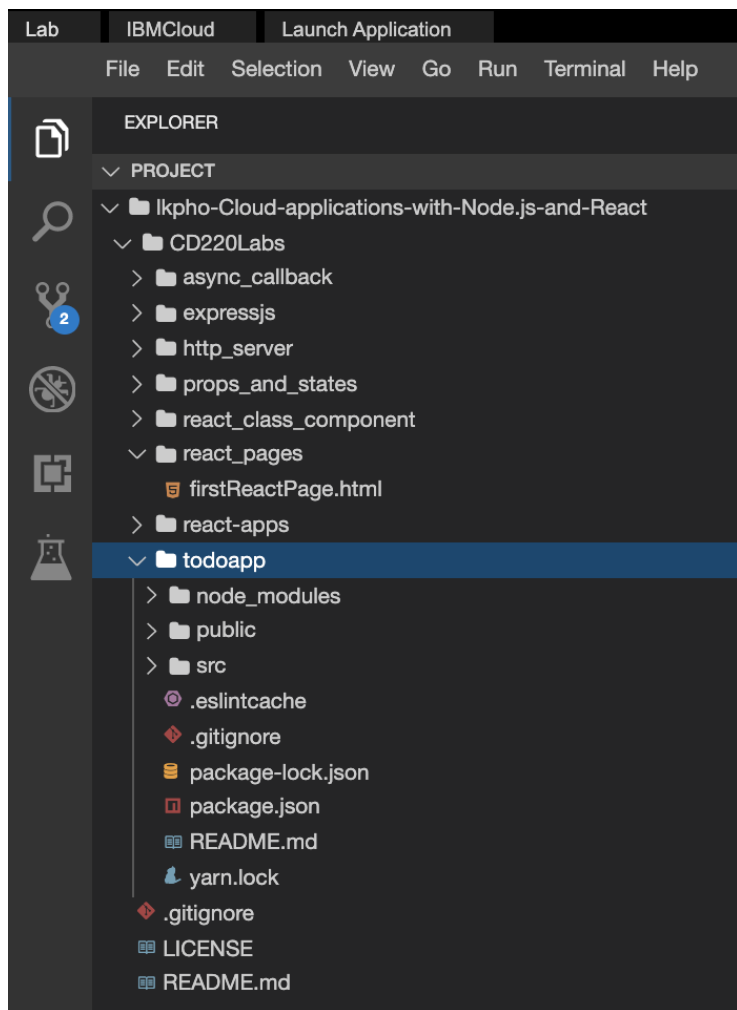
2. In the terminal window run the following commands one after the other.

```
1. 1
2. 2
3. 3
```

```
1. cd todoapp
2. npm install
3. npm start
```

Copied!

The first command changes to the `todoapp` directory. Second command installs the files required for the React app in your lab environment and the third one starts the server.

You will see this output indicating that the server is running.
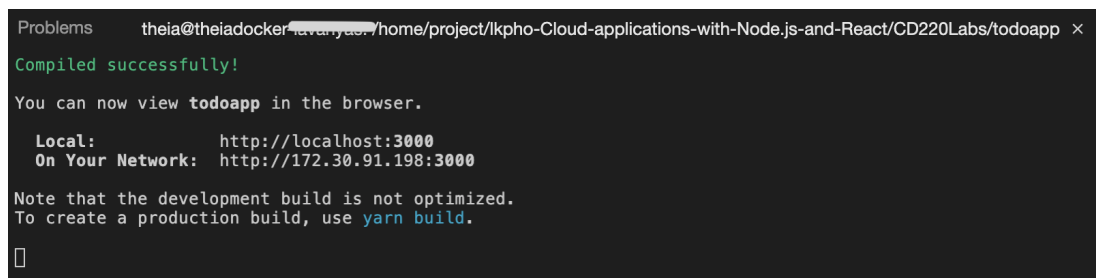
3. Click on the button below to open the React application on new browswer tab or click **Skills Network** button on the left, choose the **Other** option and then click **Launch Application** and connect to port 3000.

REACT Application
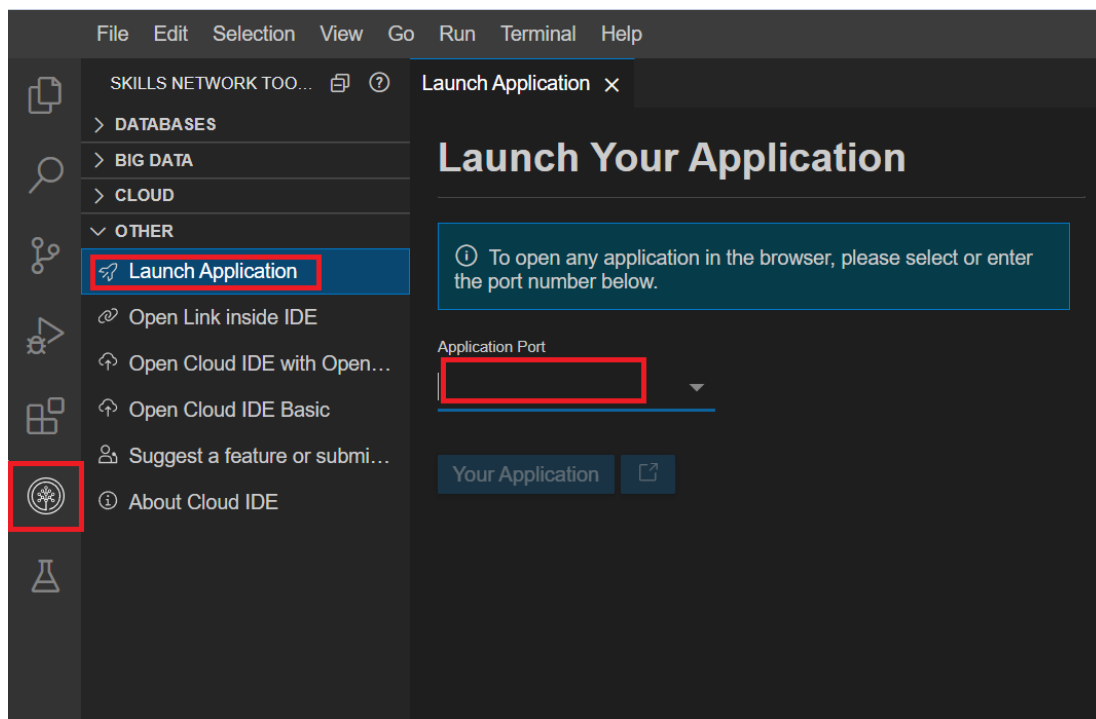
The todoapp UI will appear on the browser as seen in the image below.
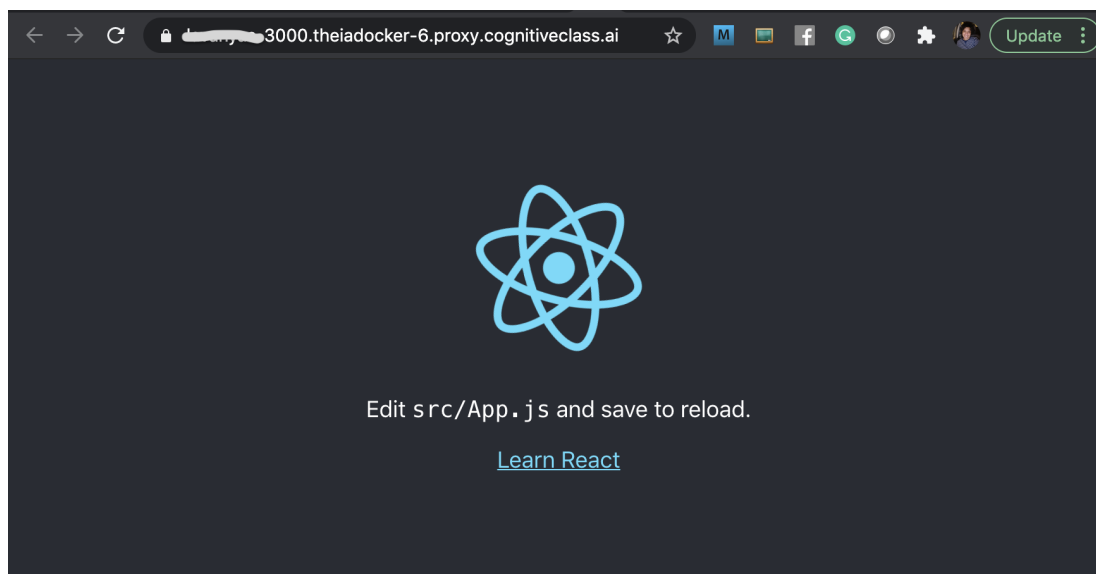
4. To stop the server, go to the main command window and press Ctrl+c to stop the server.

Exercise 3: Function Components, Class Components, Properties and States

The `react-apps` is a react app that has been created for you using `npx create-react-app react-apps` command. In the files explorer view `react-apps` directory. It would appear like this.
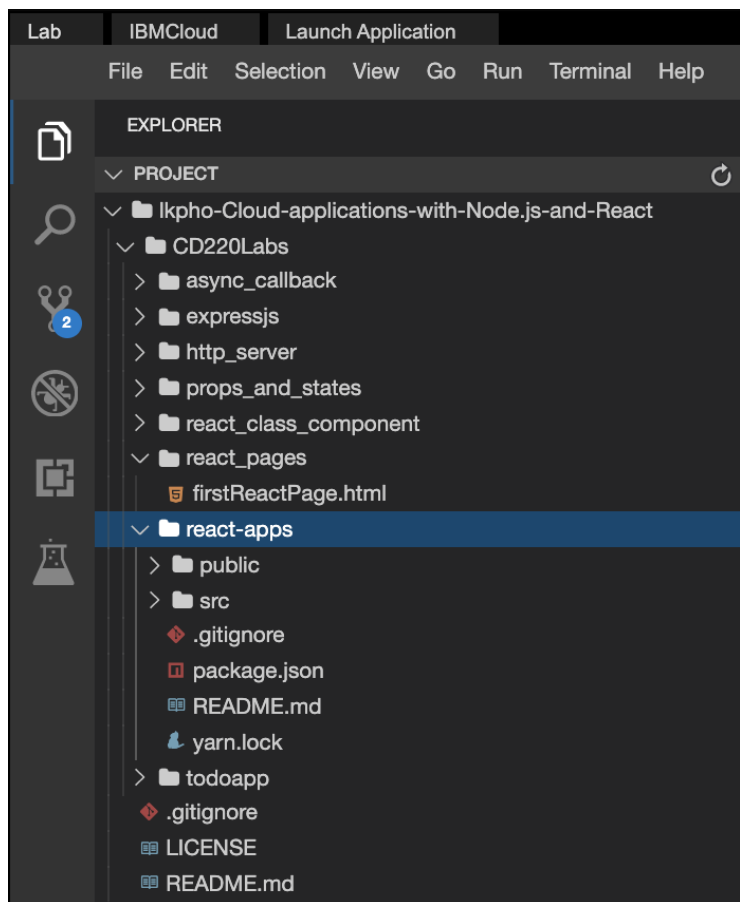
2. In the terminal window run the following commands one after the other.

1. 1
2. 2
3. 3

```
1. cd /home/project/lkpho-Cloud-applications-with-Node.js-and-React/CD220Labs/react-apps
2. npm install
3. npm start
```

Copied!

The first command changes to the `react-apps` directory. Second command installs the files required for the React app in your lab environment and the third one starts the server.

You will see this output indicating that the server is running.

3. To see what UI this React app renders, click on the **Skills Network** button on the left, it will open the "Skills Network Toolbox". Then click the **Other** then **Launch Application**. From there enter the port number the server is running on, which is `3000` and then launch.
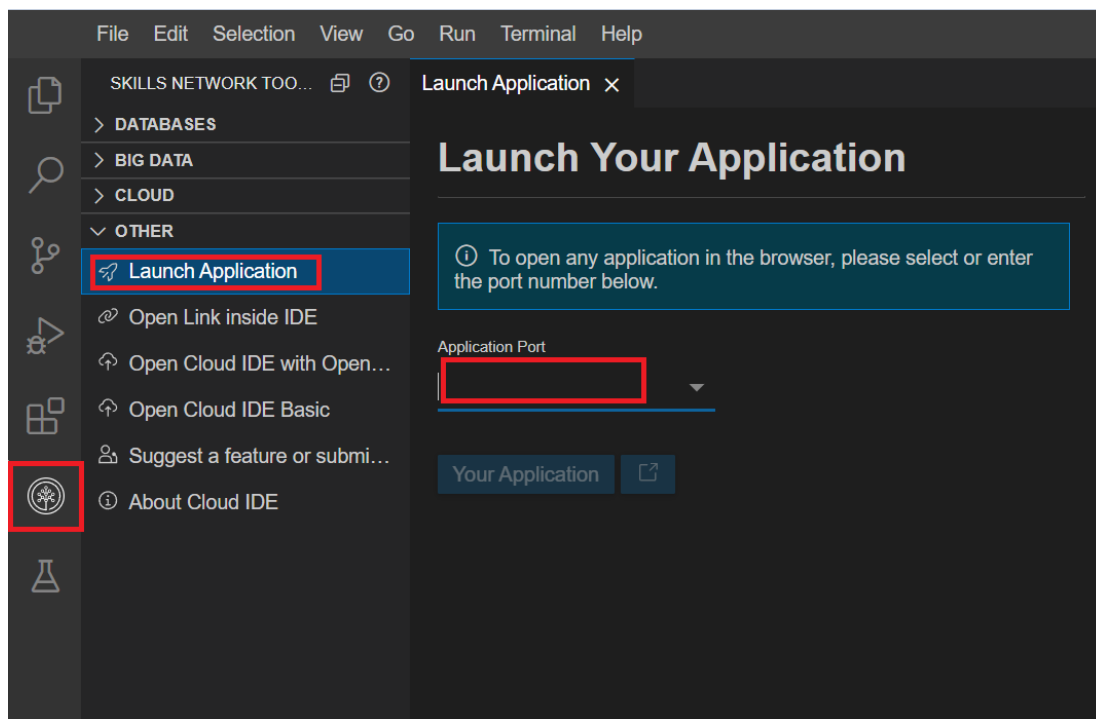
The react-apps UI will appear on the browser. All the styling has been removed and you should see **Hello World** written on the page.

4. View the content on App.js under `react-apps`->`src` in the file explorer. It will appear as below.
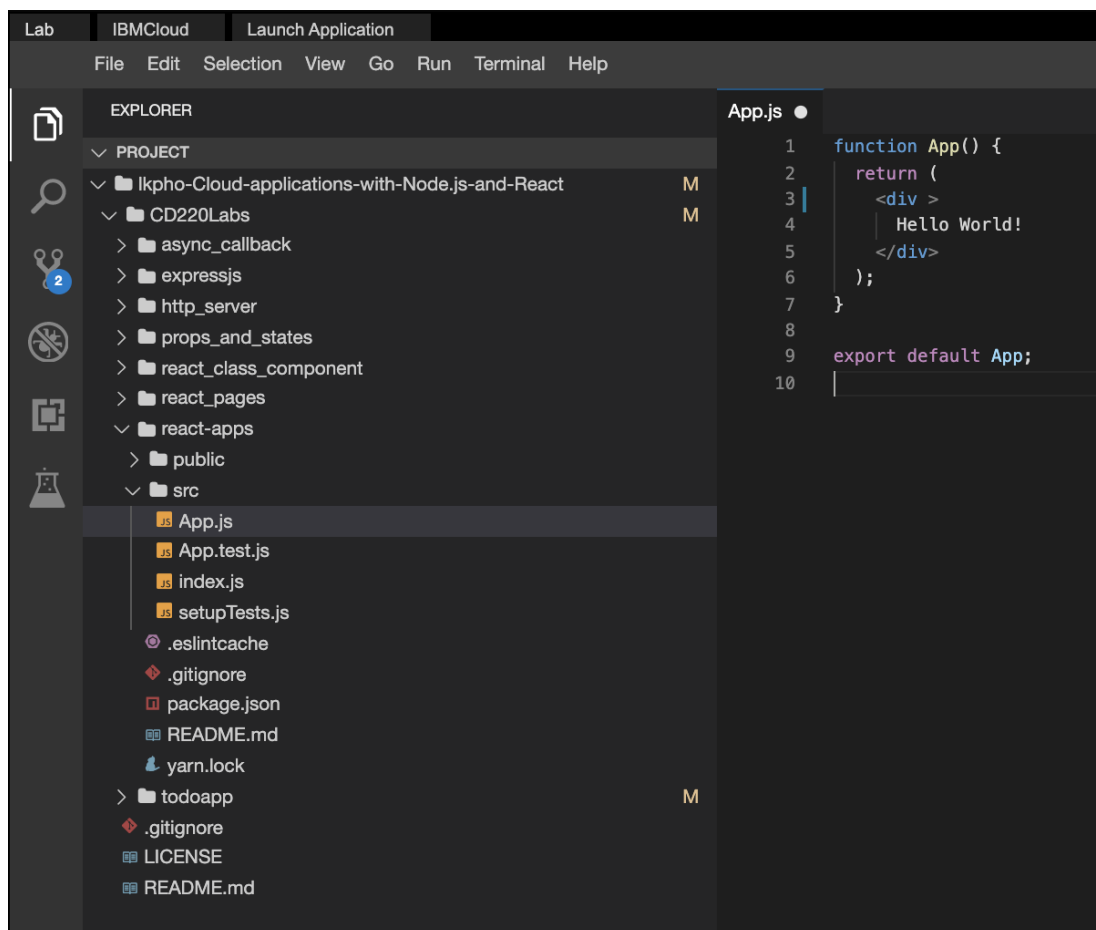
5. Now let's start adding a component to the page. *When the server is running, any change made to the files is automatically picked up by the server.*

The code in App.js will show as given below.

```
1. 1
2. 2
```

```
3.  3
4.  4
5.  5
6.  6
7.  7
8.  8
9.  9
```

```
1.  function App() {
2.    return (
3.      <div >
4.        Hello World!
5.      </div>
6.    );
7.  }
8.
9.  export default App;
```

Copied!

It is a function component. Let's make changes to the code to accept the color as props.

    6. Replace the content of App.js with the following code.

```
1.  1
2.  2
3.  3
4.  4
5.  5
6.  6
7.  7
8.  8
9.  9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
```

```
1.  function App(props) {
2.    const colorStyle = {
3.      color:props.color,
4.      fontSize:props.size+"px"
5.    }
6.    return (
7.      <div>
8.        <div style={colorStyle}>
9.        Hello World!
10.       </div>
11.     </div>
12.   );
13. }
14.
15. export default App;
```

Copied!

We have made changes to App.js to make it a component which will take some style setting as props. We now have to set the properties.

    7. To set the props of the App component, we will make changes in `index.js`. Replace the content of `index.js` with the code given below.

```
1.  1
2.  2
3.  3
4.  4
5.  5
6.  6
7.  7
8.  8
9.  9
10. 10
```

```
1.  import React from 'react';
2.  import ReactDOM from 'react-dom';
```

```
 3. import App from './App';
 4.
 5. ReactDOM.render(
 6.    <React.StrictMode>
 7.      <App color="green" size="20"/>
 8.    </React.StrictMode>,
 9.    document.getElementById('root')
10. );
```

Copied!

As can be seen in the code we are passing the color and size.

8. Now when you refresh the broswer rendering, it will be rerendered with the style.

Practice Exercise

- Add other styling to the components.

9. Now let's change the same to a class component instead of a function component. Replace the App.js code with the following code.

```
 1.  1
 2.  2
 3.  3
 4.  4
 5.  5
 6.  6
 7.  7
 8.  8
 9.  9
10.  10
11.  11
12.  12
13.  13
14.  14
15.  15
16.  16
17.  17
18.  18
19.  19
20.  20
21.  21
22.  22
23.  23
```

```
 1. import React from 'react';
 2.
 3. class App extends React.Component {
 4.    constructor(props) {
 5.      super(props);
 6.    }
 7.
 8.    render() {
 9.      const colorStyle = {
10.      color:this.props.color,
11.      fontSize:this.props.size+"px"
12.      }
13.      return (
14.        <div>
15.          <div style={colorStyle}>
16.          Hello World!
17.          </div>
18.        </div>
19.      );
20.    }
21. }
22.
23. export default App;
```

Copied!

10. Refresh the page that was rendered earlier to see how different it is. You will observe that nothing has changed. All that we have done is replaced the function component with class component. So the props can be passed to both function and class components.

11. Let's set the props from outside and maintain the state of the component inside. The state we will maintain is the number of times the button was clicked. Replace the code in App.js with the following code and save it. Refresh the page that is rendered.

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

```
1.  import React from 'react';
2.
3.  class App extends React.Component {
4.    constructor(props) {
5.      super(props);
6.    }
7.    state = { counter:0 }
8.
9.    incrementCounter = ()=> {
10.     this.setState({counter:this.state.counter+1});
11.   }
12.
13.   render() {
14.     const colorStyle = { color:this.props.color,fontSize:this.props.size+"px"}
15.     return (
16.       <div style={colorStyle}>
17.         React Component
18.         <br /><br />
19.         <button onClick={this.incrementCounter}>Click Me!</button>
20.         <br /><br />
21.         {this.state.counter}
22.       </div>
23.     );
24.   }
25. }
26.
27. export default App;
```

`Copied!`

You will see that the counter state is being refreshed everytime the button is clicked.

12. To stop the server, go to the main command window and press Ctrl+c to stop the server.

Exercise 4: Class Components - componentDidMount

1. Let's make changes in App.js to send a API request to a remote server when the component mounts, and change the state when the remote server responds.

Replace the code in App.js that we used in the previous exercise.

```
1.  1
```

       2. 2
       3. 3
       4. 4
       5. 5
       6. 6
       7. 7
       8. 8
       9. 9
      10. 10
      11. 11
      12. 12
      13. 13
      14. 14
      15. 15
      16. 16
      17. 17
      18. 18
      19. 19
      20. 20
      21. 21
      22. 22
      23. 23
      24. 24
      25. 25
      26. 26
      27. 27
      28. 28
      29. 29
      30. 30
      31. 31
      32. 32
      33. 33
      34. 34
      35. 35
      36. 36
      37. 37
      38. 38
      39. 39
      40. 40
      41. 41
      42. 42
      43. 43
      44. 44

```
 1. import React from 'react';
 2. import axios from 'axios';
 3.
 4. class App extends React.Component {
 5.   state = { APIlist:[] }
 6.
 7.   componentDidMount() {
 8.     let url = "https://api.publicapis.org/entries?category=Animals";
 9.     axios({
10.       method: 'get',
11.       url: url,
12.       responseType: 'json'
13.     }).then(resp => {
14.         let listOfEntries = resp.data.entries;
15.         let listOfEntriesAsArray = Object.entries(listOfEntries);
16.         let entryDetails = listOfEntriesAsArray.map((entryDetail)=>{
17.           return <li style={{color: "green"}}>{entryDetail[1]["API"]}
18.           ------- {entryDetail[1]["Link"]} </li>
19.         })
20.         this.setState({APIlist:<ul>{entryDetails}</ul>})
21.       })
22.     .catch(err => {
23.         console.log(err.toString())
24.     });
25.   }
26.
27.   render() {
28.     const colorStyle = { color:this.props.color,fontSize:this.props.size+"px"}
29.     return (
```

```
30.          <div style={colorStyle}>
31.             <h2>APIs List</h2>
32.             <br/>
33.
34.             <div>
35.                 {
36.                     this.state.APIlist
37.                 }
38.             </div>
39.         </div>
40.       );
41.    }
42. }
43.
44. export default App;
```

Copied!

As you can see initially the state of the component `APIList` is an empty list. When the component mounts, the request is sent through axios asynchronously and then depending on the list returned, the setState is invoked.

2. This code requires the axios module to run. Run the following command to install it.

```
npm install --save axios
```

3. Ensure that you are in the `react-apps` directory. Run the following command to see which directory you are currently in.

```
pwd
```

4. If you are not in `/home/project/lkpho-Cloud-applications-with-Node.js-and-React/CD220Labs/react-apps`, run the following command to change to that directory. Else skip to step 5.

```
cd /home/project/lkpho-Cloud-applications-with-Node.js-and-React/CD220Labs/react-apps
```

5. Run `npm start` in the terminal to start the react app.

6. Click on the button below to open the React application on new browswer tab or click **Skills Network** button on the left, choose the **Other** option and then click **Launch Application** and connect to port 3000.

REACT Application
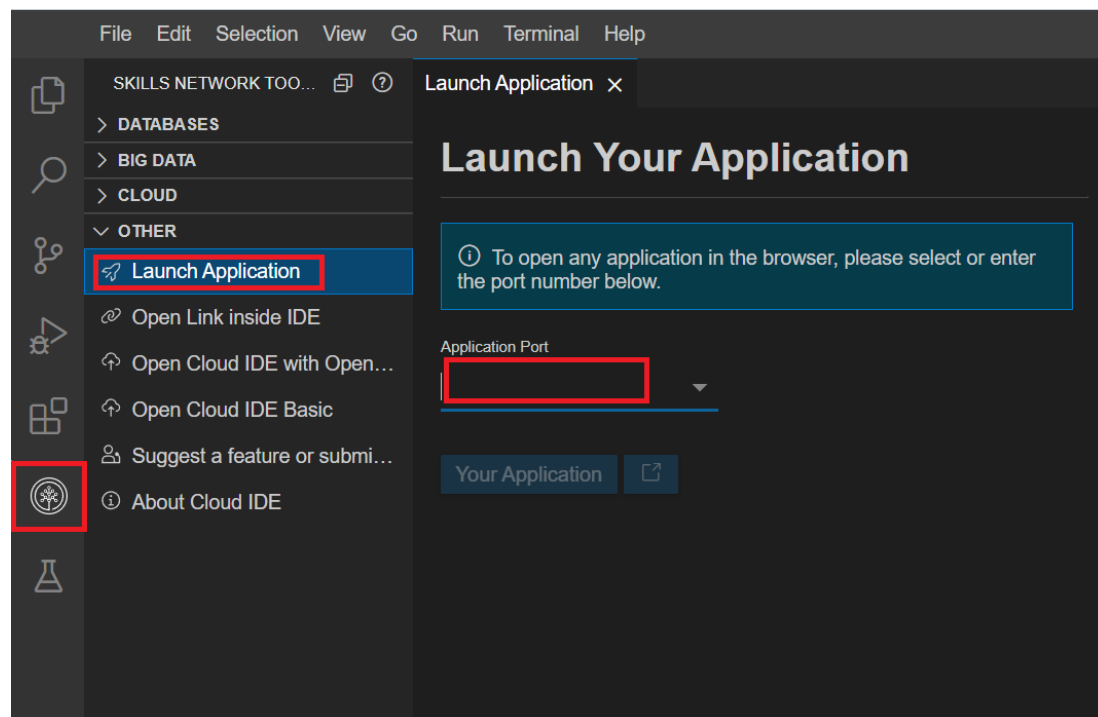
The output would look like this.

APIs List

- AdoptAPet------- https://www.adoptapet.com/public/apis/pet_list.html
- Axolotl------- https://theaxolotlapi.netlify.app/
- Cat Facts------- https://alexwohlbruck.github.io/cat-facts/
- Cataas------- https://cataas.com/
- Cats------- https://docs.thecatapi.com/
- Dog Facts------- https://dukengn.github.io/Dog-facts-API/
- Dog Facts------- https://kinduff.github.io/dog-api/
- Dogs------- https://dog.ceo/dog-api/
- eBird------- https://documenter.getpostman.com/view/664302/S1ENwy59
- FishWatch------- https://www.fishwatch.gov/developers
- HTTP Cat------- https://http.cat/
- HTTP Dog------- https://http.dog/
- IUCN------- http://apiv3.iucnredlist.org/api/v3/docs
- MeowFacts------- https://github.com/wh-iterabb-it/meowfacts
- Movebank------- https://github.com/movebank/movebank-api-doc
- Petfinder------- https://www.petfinder.com/developers/
- PlaceBear------- https://placebear.com/
- PlaceDog------- https://place.dog
- PlaceKitten------- https://placekitten.com/
- RandomDog------- https://random.dog/woof.json
- RandomDuck------- https://random-d.uk/api
- RandomFox------- https://randomfox.ca/floof/
- RescueGroups------- https://userguide.rescuegroups.org/display/APIDG/API+Developers+Guide+Home
- Shibe.Online------- http://shibe.online/
- The Dog------- https://thedogapi.com/
- xeno-canto------- https://xeno-canto.org/explore/api
- Zoo Animals------- https://zoo-animal-api.herokuapp.com/

Task

Modify the code in App.js to use the same URL and to render the API details in the table format given below.

APIs List

AdoptAPet	https://www.adoptapet.com/public/apis/pet_list.html
Axolotl	https://theaxolotlapi.netlify.app/
Cat Facts	https://alexwohlbruck.github.io/cat-facts/
Cataas	https://cataas.com/
Cats	https://docs.thecatapi.com/
Dog Facts	https://dukengn.github.io/Dog-facts-API/
Dog Facts	https://kinduff.github.io/dog-api/
Dogs	https://dog.ceo/dog-api/
eBird	https://documenter.getpostman.com/view/664302/S1ENwy59
FishWatch	https://www.fishwatch.gov/developers
HTTP Cat	https://http.cat/
HTTP Dog	https://http.dog/
IUCN	http://apiv3.iucnredlist.org/api/v3/docs
MeowFacts	https://github.com/wh-iterabb-it/meowfacts
Movebank	https://github.com/movebank/movebank-api-doc
Petfinder	https://www.petfinder.com/developers/
PlaceBear	https://placebear.com/
PlaceDog	https://place.dog
PlaceKitten	https://placekitten.com/
RandomDog	https://random.dog/woof.json
RandomDuck	https://random-d.uk/api
RandomFox	https://randomfox.ca/floof/
RescueGroups	https://userguide.rescuegroups.org/display/APIDG/API+Developers+Guide+Home
Shibe.Online	http://shibe.online/
The Dog	https://thedogapi.com/
xeno-canto	https://xeno-canto.org/explore/api
Zoo Animals	https://zoo-animal-api.herokuapp.com/

▼ Click here for the solution

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

```
29. 29
30. 30
31. 31
32. 32
33. 33
34. 34
35. 35
36. 36
37. 37
38. 38
39. 39
40. 40
41. 41
42. 42
43. 43
44. 44
45. 45
46. 46
```

```jsx
 1. import React from 'react';
 2. import axios from 'axios';
 3.
 4. class App extends React.Component {
 5.    state = { APIlist:[] }
 6.
 7.    componentDidMount() {
 8.      let url = "https://api.publicapis.org/entries?category=Animals";
 9.      axios({
10.        method: 'get',
11.        url: url,
12.        responseType: 'json'
13.      }).then(resp => {
14.          let listOfEntries = resp.data.entries;
15.          let listOfEntriesAsArray = Object.entries(listOfEntries);
16.          let entryDetials = listOfEntriesAsArray.map((entryDetail)=>{
17.            return <tr><td style={{color: "red",border: "1px solid black"}}>{entryDetail[1]["API"]} </td>
18.             <td style={{color: "red",border: "1px solid black"}}> {entryDetail[1]["Link"]} </td>
19.             </tr>
20.          })
21.          this.setState({APIlist:<table style={{border: "1px solid black", margin: "30px"}}><tbody>{entryDe
22.      })
23.      .catch(err => {
24.          console.log(err.toString())
25.      });
26.    }
27.
28.    render() {
29.      const colorStyle = { color:this.props.color,fontSize:this.props.size+"px"}
30.      return (
31.        <div style={colorStyle}>
32.        <h2>APIs List</h2>
33.        <br/>
34.
35.        <div>
36.
37.                {
38.                this.state.APIlist
39.                }
40.          </div>
41.          </div>
42.      );
43.    }
44. }
45.
46. export default App;
```

[Copied!]

Congratulations! You have completed the lab for the first week of this course.

Author(s)

Lavanya

Changelog

Date	Version	Changed by	Change Description
2021-01-18	1.1	Lavanya	Changed the lab exercise as the remote link had some issues
2020-11-27	1.0	Lavanya	Initial version created based videos