

## Tutorial 05: React - Create components using JSX

**NOTE:** If you have problem with installation, you may use CodeSandbox CDE to write React codes for free (<https://codesandbox.io/>) for this tutorial

Refer to sample codes shared on SPeCTRUM.

JSX allows us to write HTML in React by converting HTML into React components.

Two main rules of writing JSX:

- You must explicitly close self-closing elements like `<br />`
- Components can only return a single element
  - If you have multiple elements, only the top element will be returned
  - E.g. Utilize `<div> </div>` as top (parent) element

1. Create a directory named '**components**' inside the '**src**' directory in your react app folder
2. In the '**components**' directory, create a new component directory named 'JobList'
3. In the 'JobList' directory, create new component file name "JobList.js"
  - a. Write a JobList component to return a list of job.
  - b. Export the component: In JSX, '**export default**' statement is used to export a single value or component from a module that is present by default in your script so that others script can import that for use.

### Example 1:

```
export default function Greeter() {  
    return <h1>HELLO!</h1>  
}
```

### Example 2:

```
const JobList = () => {  
    return (  
        <ul jobName="job-list">  
            <li>Barista</li>  
            <li>Manager</li>  
            <li>Trainee</li>  
        </ul>  
    );  
};  
  
export default JobList;
```

4. Styling components: In the '**JobList**' directory, create new css file name '**JobList.css**'
  - a. Write some css codes to design this component. You may create your own css class element to be used in the component.
  - b. In the **JobList.js** file, add the codes to import this css file:

```
import './JobList.css';
```

5. In the **App.js** file (inside '**src**' directory)
  - a. import this component before using it, for example:

```
import JobList from "../components/JobList/JobList";
```

- b. Add the component name using <component name/> in the App.js file, for example:

```
export default function App() {  
  return (  
    <div className="Job-list">  
      <h2>JobList</h2>  
      <JobList />  
    </div>  
  );  
}
```

- c. Refer to sample React codes 2, modify the App.js codes by defining an array named JobClass with three JobClass objects (see Table 1).

**Table 1: Three JobClass objects with code, name and scope properties**

code	name	scope
ba	Barista	Preparing and serving hot and cold drinks such as coffee, tea, artisan and speciality beverages.
ma	Manager	Managing day-to-day operations of the cafe.
tr	Trainee	Supporting daily operations of the cafe.

**Question 2: Using DOM in JavaJam Coffee House website**

In this task, you have to do the following steps:

1. Open the `jobs.html` file in your `javajam5` project folder to modify the dropdown list codes (“*Select job position:*”).

Remove/Comment the following codes from “`jobs.html`”:

```
<select id="select-choice" class="form-control col-sm-4" >
    <option value="Barista">Barista</option>
    <option value="Manager">Manager</option>
    <option value="Trainee">Trainee</option>
</select>
```

2. Write JavaScript codes to replace the codes that have been removed/commented.

```
<label for="select-choice">Select job position:</label>
<script>
//Write codes for 2(a), 2(b), 2(c), 2(d)

</script>
```

- a) Define an array of three `jobclass` objects (see Table 1). Each `jobclass` object should have `code`, `name`, and `scope` properties.

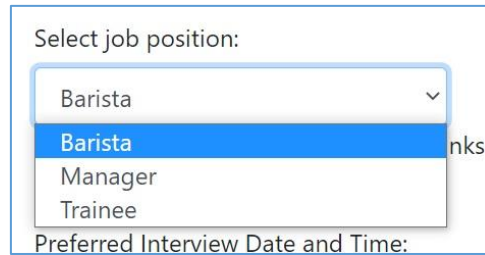
Sample code to define an array:

```
var person = [{firstName:"John", lastName:"Doe", age:46},
               {firstName:"Aminah", lastName:"Ali", age:40},
               ];
```

**Table 1: Three `jobclass` objects with code, name and scope properties**

code	name	scope
ba	Barista	Preparing and serving hot and cold drinks such as coffee, tea, artisan and speciality beverages.
ma	Manager	Managing day-to-day operations of the cafe.
tr	Trainee	Supporting daily operations of the cafe.

- b) Given the provided array of `jobclass`, using `document.write` and a loop to generate a drop-down list that display the `jobclass` name. The `value` attribute for each item in the list should be the `jobclass` code. The sample browser screenshot should look like **Figure 3**.



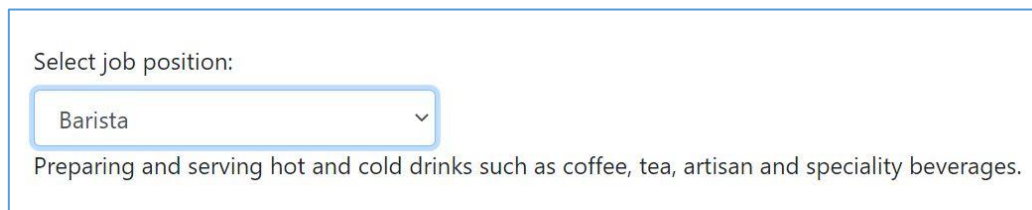
**Figure 3:** Screenshot for the drop-down list that display the list of job positions

- c) Add a new paragraph `<p>` HTML element with an id named "jobScope" after the closing tag of `</script>` to display the scope of each job position (see Figure 4).

```
<script>
```

```
</script>
```

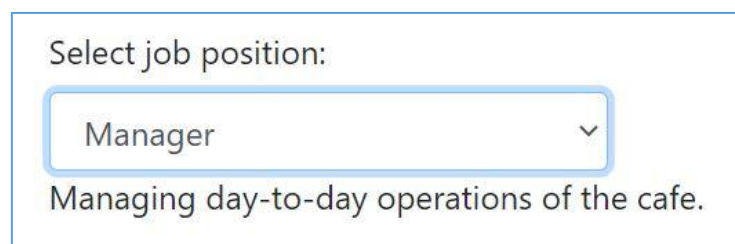
```
<p id="jobScope">Preparing and serving hot and cold drinks such  
as coffee, tea, artisan and speciality beverages.</p>
```



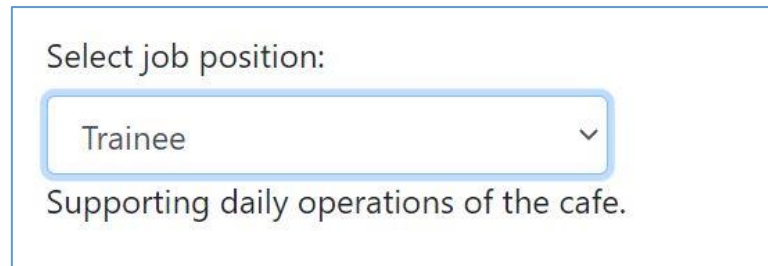
**Figure 4:** Screenshot of the paragraph that display job scope description

- d) Write JavaScript code using DOM to add an **event listener** to handle the **change event** of the drop-down list. This event handler should **display the job scope** of the currently **selected job position** within the paragraph `<p id="jobScope"></p>`. The sample browser screenshots should look like Figure 5 and Figure 6.

**Hint:** Your event handler may use the `value` property of the drop-down list to which show the `'code'` of the `jobclass` in the array in order to retrieve the right job scope (`'scope'`) in the array to be displayed in the `textContent` or `innerText` property of the `<p id="jobScope">`.



**Figure 4:** Screenshot when “**Manager**” job position is selected, the browser should display the description “**Managing day-to-day operations of the cafe.**”



Select job position:

Trainee ▼

Supporting daily operations of the cafe.

**Figure 5:** Screenshot when “**Trainee**” job position is selected, the browser should display the description “**Supporting daily operations of the cafe.**”