Exercise (Instructions): Presentational and Container Components

Objectives and Outcomes

In this exercise we understand about how presentational components deal with the look and feel of the app and container components deal with the data and behavior. At the end of this exercise you will learn about:

- Organizing your React app into presentational and container components
- Enable your presentational components to be concerned with the look and feel of your app
- Enable container components to deal with the state, provide the data and handle user interactions.

Add a Container Component

- Add a new component named *MainComponent.js* in the components folder and update its contents as follows:

```
import React, { Component } from 'react';
import { Navbar, NavbarBrand } from 'reactstrap';
import Menu from './MenuComponent';
import DishDetail from './DishdetailComponent';
import { DISHES } from '../shared/dishes';

class Main extends Component {

  constructor(props) {
    super(props);
    this.state = {
        dishes: DISHES,
        selectedDish: null
    };
  }

  onDishSelect(dishId) {
    this.setState({ selectedDish: dishId});
  }

  render() {
    return (
      <div>
        <Navbar dark color="primary">
          <div className="container">
            <NavbarBrand href="/">Ristorante Con Fusion</NavbarBrand>
          </div>
        </Navbar>
        <Menu dishes={this.state.dishes} onClick={(dishId) => this.onDishSelect
          (dishId)} />
        <DishDetail dish={this.state.dishes.filter((dish) => dish.id === this
          .state.selectedDish)[0]} />
      </div>
    );
  }
}

export default Main;
```

- Update the *App.js* by removing the state related information, and make use of Main Component to render the UI:

```
. . .
import Main from './components/MainComponent';

class App extends Component {

  render() {
    return (
      <div className="App">
        <Main />
      </div>
    );
  }
}

. . .
```

Turn Menu Component into a Presentational Component

- Open MenuComponent.js and update its contents by removing the state and removing the DishdetailComponent reference, and make use of the onClick supplied by MainComponent through the props to deal with the clicking of a menu item:

```
. . .
                    <Card key={dish.id}
                        onClick={() => this.props.onClick(dish.id)}>

. . .
```

- The DishdetailComponent is already structured as a presentational component and hence needs no further update, except wrapping the return value from render() within a with the className as container.
- To print out the date for a comment in a format more suitable for human readability, you can update your renderComment function with the code snippet shown below:

```
{new Intl.DateTimeFormat('en-US', { year: 'numeric', month: 'short', day: '2
  -digit'}).format(new Date(Date.parse(comment.date)))}
```

- Save all the changes and do a Git commit with the message "Presentational and Container Components"

Conclusions

In this exercise you learnt how to structure your app into presentational and container components.

✓ Complete Go to next item