React (JavaScript/JSX) Style Guide {

Before creating Components

Research the available components, layout spacing, etc. in our Bootstrap library here:

https://getbootstrap.com/docs/4.4/components/alerts/

Often times there is no need to create new components, and if so we can build a component consisting of other components. Look at all the available options the component offers as well.

Finally, use their offered spacing classNames instead of creating style variables if possible. They are shown here:

https://getbootstrap.com/docs/4.0/utilities/spacing/

Example:

```
// Good - this gives the Column component a padding of 3
<Col className="p-3">
    <div>
        {this.renderHours()}
    </div>
</Col>
.
```

CSS box model is very important for UI development. A refresher is here:

https://www.geeksforgeeks.org/css-box-model/

Components

Creating components should follow this class declaration format.

```
// bad
const Listing = React.createClass({
  // ...
  render() {
    return <div>{this.state.hello}</div>;
  }
});


// good
class Listing extends React.Component {
  // ...
  render() {
```

```
    return <div>{this.state.hello}</div>;
  }
}
.
```

Naming

Variables should always have meaning and never be single characters or vague. Use .jsx for components and PascalCase for the component and filename

```
// bad
import reservationCard from './ReservationCard';

// good
import ReservationCard from './ReservationCard';
```

Comments and Documentation

Components should be documented at the top of the file with their purpose and how they function.

Before a function that is not within the render of a component, the purpose of the function should be commented. Within the function any logic that is not straightforward should have a brief description.

```
//Good
/*
    This Prop is used to render the cards of the Manager Menu page.
    The menu is a 2d array with the first array containing only
    the title of the food item. The second array contains the category,
    price, calories, picture and whether the item is in stock.

    getStockState is a helper function which takes in_stock property
    which is either 0 or 1 and creates the appropriate text to display.

*/
function MenuProp(props) {
    ...
}



// Good - although uncomplicated logic just for example purposes

/*
```

```
*   Used to loop through a number
*/
function myFunction(number) {
    // Loops through every number up to the parameter and prints it
    for (var j = 0; j < number; j++) {
        console.log(j);
    }
}
```

Alignment

Use a single indent for each block of scope. First curly brace should be on first line and second should be at the same indentation as the first line. Indentation should be **2 spaces**

```
    // if props fit in one line then keep it on the same line
    <Foo bar="bar" />

    // Good
    for (let i = 0; i < 3; i++) {
      let children = []
      //Inner loop to create children
      for (let j = 0; j < 5; j++) {
        children.push(<td>{`Column ${j + 1}`}</td>)
      }
      //Create the parent and add the children
      table.push(<tr>{children}</tr>)
    }
```

Styling

Styling for each variable should be within the component and be outside of the component declaration. The variable names should be declared as constants and be meaningful. The Javascript object should have the key and values as strings.

```
//Bad
var itemStyle = {
    display: "flex",
    borderBottom: "grey solid 1px",
};

//Good
const cardHeaderStyle = {
  'background-color': '#0b658a',
  'color': '#ffffff'
}
```

Props

Always use camelCase for prop names

```
// bad
<Foo UserName="hello" phone_number={12345678} />

// good
<Foo userName="hello" phoneNumber={12345678} />
```

Parenthesis

Wrap JSX tags in parentheses when they span more than one line

```
// bad
render() {
  return <MyComponent variant="long body" foo="bar">
           <MyChild />
         </MyComponent>;
}

// good
render() {
  return (
    <MyComponent variant="long body" foo="bar">
      <MyChild />
    </MyComponent>
  );
}

// good, when single line
render() {
  const body = <div>hello</div>;
  return <MyComponent>{body}</MyComponent>;
}
```

Full example file

The export of the component should be at the bottom of the file.

```
import React from "react";
import Order from "./Order";
import Container from 'react-bootstrap/Container';

class Orders extends React.Component{
    /*
    This Prop is used to render the orders for the Cook page.
```

```
    The orders are an array of object containing order details.
Look at the Order component for more details on the format of an order object.

    renderOrders is a helper function which takes all the orders,
    converts them to Order objects and returns a single JSX object that
    will be exported as an Orders
    component.
*/
  constructor(props) {
    super(props);
    this.state = {
      orders: [
        {table: 1, items: [{quantity: 1, title: "Hamburger"}]},
        {table: 2, items: [{quantity: 1, title: "Hamburger"}]}
      ]
    };
  }

  renderOrders() {
    // Returns every order stored in the components state as an individual
    Order component
    return this.state.orders.map((item, key) =>
      <Order key={key} id={key} order={item}/>
    );
  }

  render() {
    return (
      <Container fluid>
        <div style={ordersStyle}>
          {this.renderOrders()}
        </div>
      </Container>
    )
  };
}

const ordersStyle = {
  'display': 'flex',
  'margin': '30px',
};

export default Orders;
```