CODE GUID FOR REACT-JSX

## 1. Basic Rules

1.1 Only include one React component per file.

1.2 Always use JSX syntax.

1.3 Do not use React.createElement unless you're initializing the app from a file that is not JSX.

**2. Class vs React.createClass vs stateless**

2.1 If you have internal state and/or refs, prefer class extends React.Component over React.createClass

// 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>;

}

}

2.2 if you don't have state or refs, prefer normal functions (not arrow functions) over classes

// bad

class Listing extends React.Component {

render() {

return <div>{this.props.hello}</div>;

}

}

// bad (since arrow functions do not have a "name" property)

const Listing = ({ hello }) => (

<div>{hello}</div>

);

// good

function Listing({ hello }) {

return <div>{hello}</div>;

}

**3. Naming**

3.1 **Extensions**: Use .jsx extension for React components.

3.2 **Filename**: Use PascalCase for filenames. E.g., ReservationCard.jsx.

3.3 **Reference Naming**: Use PascalCase for React components and camelCase for their instances.

// bad

import reservationCard from './ReservationCard';

// good

import ReservationCard from './ReservationCard';

// bad

const ReservationItem = <ReservationCard />;

// good

const reservationItem = <ReservationCard />;

**3.4 Component Naming:** Use the filename as the component name

// bad

import Footer from './Footer/Footer';

// bad

import Footer from './Footer/index';

// good

import Footer from './Footer';

## 4. Quotes

4.1 Always use double quotes (") for JSX attributes, but single quotes for all other JS.

// bad

<Foo bar='bar' />

// good

<Foo bar="bar" />

// bad

<Foo style={{ left: "20px" }} />

// good

<Foo style={{ left: '20px' }} />

**5. Props**

5.1 Always use camelCase for prop names.

// bad

<Foo

UserName="hello"

phone\_number={12345678}

/>

// good

<Foo

userName="hello"

phoneNumber={12345678}

/>

**6. Parentheses**

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

// bad

render() {

return <MyComponent className="long body" foo="bar">

<MyChild />

</MyComponent>;

}

// good

render() {

return (

<MyComponent className="long body" foo="bar">

<MyChild />

</MyComponent>

);

}

// good, when single line

render() {

const body = <div>hello</div>;

return <MyComponent>{body}</MyComponent>;

}

**7. Tags**

7.1 Always self-close tags that have no children

// bad

<Foo className="stuff"></Foo>

// good

<Foo className="stuff" />

7.2 If your component has multi-line properties, close its tag on a new line.

// bad

<Foo

bar="bar"

baz="baz" />

// good

<Foo

bar="bar"

baz="baz"

/>

**8. Methods**

8.1 Use arrow functions to close over local variables.

function ItemList(props) {

return (

<ul>

{props.items.map((item, index) => (

<Item

key={item.key}

onClick={() => doSomethingWith(item.name, index)}

/>

))}

</ul>

);

}

8.2 Bind event handlers for the render method in the constructor.

// bad

class extends React.Component {

onClickDiv() {

// do stuff

}

render() {

return <div onClick={this.onClickDiv.bind(this)} />

}

}

// good

class extends React.Component {

constructor(props) {

super(props);

this.onClickDiv = this.onClickDiv.bind(this);

}

onClickDiv() {

// do stuff

}

render() {

return <div onClick={this.onClickDiv} />

}

}

8.3 Do not use underscore prefix for internal methods of a React component.

// bad

React.createClass({

\_onClickSubmit() {

// do stuff

},

// other stuff

});

// good

class extends React.Component {

onClickSubmit() {

// do stuff

}

// other stuff

}

**9. Ordering**

9.1 Ordering for class extends React.Component

- optional static methods

- constructor

- getChildContext

- componentWillMount

- componentDidMount

- componentWillReceiveProps

- shouldComponentUpdate

- componentWillUpdate

- componentDidUpdate

- componentWillUnmount

- clickHandlers or eventHandlers like onClickSubmit() or onChangeDescription()

- getter methods for render like getSelectReason() or getFooterContent()

- Optional render methods like renderNavigation() or renderProfilePicture()

- render

How to define propTypes, defaultProps, contextTypes, etc

import React, { PropTypes } from 'react';

const propTypes = {

id: PropTypes.number.isRequired,

url: PropTypes.string.isRequired,

text: PropTypes.string,

};

const defaultProps = {

text: 'Hello World',

};

class Link extends React.Component {

static methodsAreOk() {

return true;

}

render() {

return <a href={this.props.url} data-id={this.props.id}>{this.props.text}</a>

}

}

Link.propTypes = propTypes;

Link.defaultProps = defaultProps;

export default Link;