# React 编程规范

## 基本规则

* 每个文件只包含一个 React 组件
* 使用 JSX 语法
* 除非是从一个非 JSX 文件中初始化 app，否则不要使用 React.createElement

**Class vs React.createClass**

* 除非有更好的理由使用混淆(mixins)，否则就使用组件类继承 React.Component。eslint 规则：[react/prefer-es6-class](https://github.com/yannickcr/eslint-plugin-react/blob/master/docs/rules/prefer-es6-class.md)

// bad

const Listing = React.createClass({

render() {

return <div />;

}

});

// good

class Listing extends React.Component {

render() {

return <div />;

}

}

## 命名

* **扩展名:** 使用 jsx 作为 React 组件的扩展名
* **文件名:** 文件命名采用帕斯卡命名法，如：ReservationCard.jsx
* **引用名:** 组件引用采用帕斯卡命名法，其实例采用驼峰式命名法。eslint rules: [react/jsx-pascal-case](https://github.com/yannickcr/eslint-plugin-react/blob/master/docs/rules/jsx-pascal-case.md)

// bad

const reservationCard = require('./ReservationCard');

// good

const ReservationCard = require('./ReservationCard');

// bad

const ReservationItem = <ReservationCard />;

// good

const reservationItem = <ReservationCard />;

* **组件命名:** 使用文件名作为组件名。例如：ReservationCard.jsx 组件的引用名应该是 ReservationCard。然而，对于一个目录的根组件，应该使用 index.jsx 作为文件名，使用目录名作为组件名。

// bad

const Footer = require('./Footer/Footer.jsx')

// bad

const Footer = require('./Footer/index.jsx')

// good

const Footer = require('./Footer')

## 声明

* 不要通过 displayName 来命名组件，通过引用来命名组件

// bad

export default React.createClass({

displayName: 'ReservationCard',

// stuff goes here

});

// good

export default class ReservationCard extends React.Component {

}

## 对齐

* 对于 JSX 语法，遵循下面的对齐风格。eslint rules: [react/jsx-closing-bracket-location](https://github.com/yannickcr/eslint-plugin-react/blob/master/docs/rules/jsx-closing-bracket-location.md)

// bad

<Foo superLongParam="bar"

anotherSuperLongParam="baz" />

// good

<Foo

superLongParam="bar"

anotherSuperLongParam="baz"

/>

// if props fit in one line then keep it on the same line

<Foo bar="bar" />

// children get indented normally

<Foo

superLongParam="bar"

anotherSuperLongParam="baz"

>

<Spazz />

</Foo>

## 引号

* 对于 JSX 使用双引号，对其它所有 JS 属性使用单引号

为什么？因为 JSX 属性[不能包含被转移的引号](http://eslint.org/docs/rules/jsx-quotes)，并且双引号使得如 "don't" 一样的连接词很容易被输入。常规的 HTML 属性也应该使用双引号而不是单引号，JSX 属性反映了这个约定。

eslint rules: [jsx-quotes](http://eslint.org/docs/rules/jsx-quotes" \t "_blank)

// bad

<Foo bar='bar' />

// good

<Foo bar="bar" />

// bad

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

// good

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

## 空格

* 在自闭和标签之前留一个空格

// bad

<Foo/>

// very bad

<Foo />

// bad

<Foo

/>

// good

<Foo />

## 属性

* 属性名采用驼峰式命名法

// bad

<Foo

UserName="hello"

phone\_number={12345678}

/>

// good

<Foo

userName="hello"

phoneNumber={12345678}

/>

## 括号

* 当组件跨行时，要用括号包裹 JSX 标签。eslint rules: [react/wrap-multilines](https://github.com/yannickcr/eslint-plugin-react/blob/master/docs/rules/wrap-multilines.md)

/// 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>;

}

## 标签

* 没有子组件的父组件使用自闭和标签。eslint rules: [react/self-closing-comp](https://github.com/yannickcr/eslint-plugin-react/blob/master/docs/rules/self-closing-comp.md)

// bad

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

// good

<Foo className="stuff" />

* 如果组件有多行属性，闭合标签应写在新的一行上。eslint rules: [react/jsx-closing-bracket-location](https://github.com/yannickcr/eslint-plugin-react/blob/master/docs/rules/jsx-closing-bracket-location.md)

// bad

<Foo

bar="bar"

baz="baz" />

// good

<Foo

bar="bar"

baz="baz"

/>

## 方法

* 不要对 React 组件的内置方法使用 underscore 前缀

// bad

React.createClass({

\_onClickSubmit() {

// do stuff

}

// other stuff

});

// good

class extends React.Component {

onClickSubmit() {

// do stuff

}

// other stuff

});

## 顺序

* 继承 React.Component 的类的方法遵循下面的顺序
  1. constructor
  2. optional static methods
  3. getChildContext
  4. componentWillMount
  5. componentDidMount
  6. componentWillReceiveProps
  7. shouldComponentUpdate
  8. componentWillUpdate
  9. componentDidUpdate
  10. componentWillUnmount
  11. clickHandlers or eventHandlers like onClickSubmit() or onChangeDescription()
  12. getter methods for render like getSelectReason() or getFooterContent()
  13. Optional render methods like renderNavigation() or renderProfilePicture()
  14. render
* 怎么定义 propTypes，defaultProps，contextTypes 等等…

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;

* 使用 React.createClass 时，方法顺序如下：
  1. displayName
  2. propTypes
  3. contextTypes
  4. childContextTypes
  5. mixins
  6. statics
  7. defaultProps
  8. getDefaultProps
  9. getInitialState
  10. getChildContext
  11. componentWillMount
  12. componentDidMount
  13. componentWillReceiveProps
  14. shouldComponentUpdate
  15. componentWillUpdate
  16. componentDidUpdate
  17. componentWillUnmount
  18. clickHandlers or eventHandlers like onClickSubmit() or onChangeDescription()
  19. getter methods for render like getSelectReason() or getFooterContent()
  20. Optional render methods like renderNavigation() or renderProfilePicture()
  21. render

eslint rules: [react/sort-comp](https://github.com/yannickcr/eslint-plugin-react/blob/master/docs/rules/sort-comp.md)