React01

官方网站: https://react.docschina.org/

React是由 Facebook 公司开发并维护的 JS框架.

可以用于开发网页, 也可以使用扩展 ReactNative 开发手机端App

- 网站web: vue框架目前最主流
- 手机端App开发: ReactNative最主流

微信小程序 就是 参考 React 和 vue 最终借鉴出现的形态!

React开发分两种方式:

- 脚本方式: 类似于 jQuery, 在 html 文件中引入 一个脚本源代码, 就可以使用jQuery
- 脚手架方式: 按照脚手架之后生成项目包, 更加强大, 适用于实际开发!

HelloWorld

原生写法

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <div id="app"></div>

    <script>
      // 要求: 使用DOM方式, 向 id='app' 的元素中,
      // 添加一个 <b class="danger" title="东哥威武">HelloWorld</b>

      // 1.创建一个b元素
      let b = document.createElement("b");
      b.className = "danger";
      b.title = "东哥威武";
      b.innerText = "HelloWorld";

      // 2.找到 id='app' 的元素, 把b加到其子中
      let app = document.getElementById("app");
      app.appendChild(b);

      /**
       * jquery:
       * $('#app').append('<b class="danger" title="东哥威武">HelloWorld</b>')
       */
```

```
    </script>
  </body>
</html>
```

官方提供的地址:

> 此网站 在某些地区可能被封锁, 无法访问!

```
<script crossorigin
src="https://unpkg.com/react@16/umd/react.production.min.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-
dom.production.min.js"></script>
```

国内必定可以访问的

https://www.runoob.com/react/react-install.html

```
<script src="https://unpkg.com/react@16/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
</script>
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <!-- 引入: react的源代码 -->
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
</script>

    <!--
      引入的文件 分两个:
      * react : 负责创建元素
      * react-dom : 负责渲染元素到页面上

      Facebook工程师的简化思路: 利用函数封装的方式 简化代码
     -->
    <div id="app"></div>

    <script>
      //  React来创建 <b class="danger" title="东哥威武">HelloWorld</b>

      // Facebook工程师 封装了createElement函数
      // 固定3个参数:
```

```
      // 参数1: 元素名
      // 参数2: 元素的属性对象
      // 参数3: 元素 双标签中的内容
      let b = React.createElement('b', {className:'danger', title:'东哥威武'},
'HelloWorld');

      // react-dom 实现元素的添加
      ReactDOM.render(b, app)
    </script>
  </body>
</html>
```

JSX

babel编译工具: `https://www.babeljs.cn/`

```
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

<!-- 把想要用babel翻译的代码, 放到带有 type="text/babel" 属性的 标签中! -->
<script type="text/babel">

</script>
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <!--
      JSX语法 : 在JS原生语法的框架内, 已经无法达到更高的简化易用性!!
      必须通过 自制语法, 来实现更加简单的操作
      JSX: JS + XML(xhtml)
    -->

    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
</script>

    <!-- babel编译工具 -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

    <div id="app"></div>

    <!-- JSX语法 不被浏览器所识别; 运行时要翻译成浏览器识别的代码:  翻译软件 babel -->
    <script type="text/babel">
      // JSX语法:  在js中书写的 html 代码;
      // 下方写法在运行时 会自动转化成 02_react.html 中的那个封装写法
```

```
      // 注意: 此处属性 className 代表代码虽然格式与HTML完全相同, 但是本质上是 对DOM写法的一个改变!
      let b = (
        <b className="danger" title="东哥威武">
          HelloWorld
        </b>
      );

      ReactDOM.render(b, app);

      // $('#app').append('<b class="danger" title="东哥威武">HelloWorld</b>')
    </script>
  </body>
</html>
```

组件

三大框架都具备的核心概念 -- 属于 模块化操作

- 把一段 UI 封装起来, 实现复用;

React组件有两种方式制作:

- 函数方式: 适合简单组件
- 类方式: 适合复杂组件, 例如带有生命周期, 事件 等.

函数组件

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <!-- react源代码: 引入顺序必须正确, 有依赖关系:  react -> react-dom -->
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
</script>

    <!-- babel编译工具 -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

    <div id="app"></div>

    <script type="text/babel">
      // 组件传参 : 父子传参
      function HelloName(props) {
        // 在 angular 和 vue中,  html里使用js代码:  {{ js代码 }}
        // 在 react 中, jsx 中使用js代码: { js代码 }
```

```
        return <h1>Hello, {props.name}</h1>;
      }

      // 原生写法
      // ReactDOM.render(HelloName({ name: "东东" }), app);

      // JSX写法
      // ReactDOM.render(<HelloName name="东东" />, app);

      // 复用
      let more = (
        <div>
          <HelloName name="亮亮" />
          <HelloName name="然然" />
          <HelloName name="宝宝" />
        </div>
      );

      ReactDOM.render(more, app);
    </script>
  </body>
</html>
```

类方式组件

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <!-- react源代码: 引入顺序必须正确, 有依赖关系:  react -> react-dom -->
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
</script>

    <!-- babel编译工具 -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

    <div id="app"></div>

    <script type="text/babel">
      // 类方式的组件:  类型大驼峰命名法!
      // 类能够实现复杂组件功能全靠 继承父类!  父类中拥有强大的各种属性和方法
      class HelloWorld extends React.Component {
        // render(): 是固定的方法名,  JSX写法  <HelloWorld /> 固定调用 render()
        render() {
          return <h1>Hello World!</h1>;
        }
      }

      // let obj = new HelloWorld();   实例化
```

```
      // obj.render()
      // ReactDOM.render(new HelloWorld().render(), app);

      // JSX写法
      ReactDOM.render(<HelloWorld />, app);
    </script>
  </body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <!-- react源代码: 引入顺序必须正确, 有依赖关系:  react -> react-dom -->
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
</script>

    <!-- babel编译工具 -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

    <div id="app"></div>

    <script type="text/babel">
      // 类组件的传参
      class HelloName extends React.Component {
        // 父类的构造方法会完成:
        // constructor(props) {
        //   this.props = props;
        // }

        render() {
          return <h1>Hello, {this.props.name}</h1>;
        }
      }

      // ReactDOM.render(new HelloName({ name: "东东" }).render(), app);

      // JSX
      ReactDOM.render(<HelloName name="东东" />, app);
    </script>
  </body>
</html>
```

事件

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <!-- react源代码: 引入顺序必须正确, 有依赖关系:  react -> react-dom -->
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
</script>

    <!-- babel编译工具 -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

    <div id="app"></div>

    <script type="text/babel">
      // 事件:  必须是class方式的组件 才支持事件
      class Demo extends React.Component {
        // ES6 之前 和 之后 产生了两种函数声明方式
        // 非箭头函数 和 箭头函数:  this指向
        // * 箭头函数: 忠贞      函数体中的this 只与出生时环境有关  :身在曹营心在汉 --徐庶
        // * 非箭头函数: 渣    谁调用,函数体中的this就是谁:  三姓家奴 --吕布

        // 成员方法
        show() {
          alert("中午吃点啥呢?");
        }

        show1 = () => {
          alert("我是箭头函数");
        };

        render() {
          // 由于老师安装了 Prettier插件,所以多行代码有自动格式化 --- 详见 Angular/01 的
文档
          return (
            <div>
              <h3>事件</h3>
              {/* jsx中的注释写法 */}
              {/* 原生DOM 的点击事件: onclick */}
              {/* 注意: React 的点击事件: onClick */}
              {/* 在JSX中, {} 中的代码会在页面显示时 自动触发! */}
              <button onClick={this.show}>点我</button>

              <button onClick={this.show1}>箭头函数</button>

              <button
                onClick={() => {
                  alert("123");
                }}
              >
                箭头函数写JSX里
              </button>

              {/* 箭头函数体 {} 中只有一行代码, 可以去掉{}. 与if相同 */}
```

```
            <button onClick={() => alert("123")}>语法糖,箭头函数</button>
          </div>
        );
      }
    }

    ReactDOM.render(<Demo />, app);
  </script>
 </body>
</html>
```

事件的this

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Document</title>
 </head>
 <body>
  <!-- react源代码: 引入顺序必须正确, 有依赖关系:  react -> react-dom -->
  <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
</script>

  <!-- babel编译工具 -->
  <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

  <div id="app"></div>

  <script type="text/babel">
    // 事件中的 this
    class Demo extends React.Component {
      name = "东东";

      // DOM: 点击事件触发的函数, 本质上是由 window 调用的;
      // window.show();  此处window是undefined
      show() {
        //Cannot read property 'name' of undefined
        //不能够 对undefined 读取 属性  'name'
        console.log("this:", this);
        console.log(this.name);
      }

      render() {
        return (
          <div>
            {/* 指定普通函数中的 this 指向?  ES6之前使用 apply bind call来实现this的
绑定

            {}中的代码在页面刷新时就会执行, 即页面刷新时就会执行 bind(this)
            点击按钮时 触发的是 绑定了 this 的 show方法
```

```
              */}
              <button onClick={this.show.bind(this)}>普通函数</button>

              {/* ES6来解决this的绑定:
              点击时触发的是箭头函数 ()=>{}
              箭头函数特色: this指向声明时所在的位置!
               */}
              <button onClick={() => this.show()}>箭头函数</button>
            </div>
          );
        }
      }

      ReactDOM.render(<Demo />, app);
    </script>
  </body>
</html>
```

状态值

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <!-- react源代码: 引入顺序必须正确, 有依赖关系:  react -> react-dom -->
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
</script>

    <!-- babel编译工具 -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

    <div id="app"></div>

    <script type="text/babel">
      // 状态值 state 和 setState()
      class Demo extends React.Component {
        // num = 1;

        // 数据变化UI 则UI变化, 要求此数据必须放到固定名称的 state 属性中
        state = { num: 1 };

        doAdd() {
          // this.num++;
          // console.log(this.num);

          // 更新数据 + 更新页面 ==  setState()
          this.setState({ num: this.state.num + 1 });
        }
```

```
        render() {
          return (
            <div>
              {/*
              问题: 点击时 写的 this.doAdd   所以是this触发的 doAdd啊??
              类比: 点击时 打 志豪的媳妇;  按照上方的理解: 志豪在打自己媳妇

              此处: this.doAdd 是一个整体, 代表点击时触发 当前对象中的doAdd
              点击事件:是 window 在触发 this.doAdd 这个方法, 随意是 window在触发!

                */}
              <button onClick={this.doAdd.bind(this)}>
                计数器: {this.state.num}
              </button>
            </div>
          );
        }
      }

      ReactDOM.render(<Demo />, app);
    </script>
  </body>
</html>
```

循环渲染

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <!-- react源代码: 引入顺序必须正确, 有依赖关系:  react -> react-dom -->
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
</script>

    <!-- babel编译工具 -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

    <div id="app"></div>

    <script type="text/babel">
      // 循环显示
      class Demo extends React.Component {
        names = ["亮亮", "然然", "东东", "华华", "新新"];

        tags = [
          "SQL",
          "MYSQL",
```

```
    "SERVER",
    "NODE.JS",
    "CSS",
    "HTML",
    "BOOTSTRAP",
  ];

  emps = [
    { name: "亮亮", age: 28, sex: "男" },
    { name: "然然", age: 30, sex: "男" },
    { name: "东东", age: 32, sex: "男" },
    { name: "华华", age: 33, sex: "男" },
    { name: "新新", age: 31, sex: "男" },
  ];

  showEmps() {
    let arr = [];

    this.emps.forEach((item, index) => {
      let jsx = (
        <tr>
          <td>{index + 1}</td>
          <td>{item.name}</td>
          <td>{item.age}</td>
          <td>{item.sex}</td>
        </tr>
      );
      arr.push(jsx);
    });

    return arr;
  }

  showTags() {
    let arr = [];

    this.tags.forEach((item) => {
      let jsx = <button>{item}</button>;
      arr.push(jsx);
    });

    return arr;
  }

  showNames() {
    let arr = [];

    this.names.forEach((item) => {
      // 数组中的每一个元素 拿出来, 放到JSX语法里, 再存放到新的数组里
      let jsx = <li>{item}</li>;
      arr.push(jsx);
    });

    return arr;
  }

  render() {
    return (
```

```
            <div>
              <p>循环展示</p>
              {/* 数组中的值会自动 全部显示到页面上! */}
              <div>{this.names}</div>
              <ul>
                {/* vue的写法 */}
                {/* <li v-for="(item,index) in names" :key="index">{{item}}</li>
*/}

                {/* ng的写法 */}
                {/* <li *ngFor="let item of names; let i=index">{{item}}</li>
*/}

                {/* react的写法: 采用原生的JS循环实现 */}
                {this.showNames()}

                {/* new Demo().render()  所以render()中直接运行的代码 就是当前对象触发
的*/}
                {/* 绑定this的写法 是 点击按钮之后 由window触发 */}
              </ul>

              <div>{this.showTags()}</div>

              <table border="1">
                <tr>
                  <td>序号</td>
                  <td>姓名</td>
                  <td>年龄</td>
                  <td>性别</td>
                </tr>
                {this.showEmps()}
              </table>
            </div>
          );
        }
      }

      ReactDOM.render(<Demo />, app);
    </script>
  </body>
</html>
```

map

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <!-- react源代码: 引入顺序必须正确, 有依赖关系:  react -> react-dom -->
```

```
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
</script>

    <!-- babel编译工具 -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

    <div id="app"></div>

    <script type="text/babel">
      // map 方法实现循环
      class Demo extends React.Component {
        names = ["亮亮", "然然", "东东", "华华", "新新"];

        showNames() {
          return this.names.map((item, index) => {
            // 通过for循环生成的元素, 必须给 key 即唯一标识:  同 微信wx:key 和  vue
的:key
            return <li key={index}>{item}</li>;
          });
        }

        render() {
          return (
            <div>
              <h2>map方法的使用</h2>
              <ul>{this.showNames()}</ul>
            </div>
          );
        }
      }

      ReactDOM.render(<Demo />, app);
    </script>
  </body>
</html>
```

脚手架的安装

安装命令:

> angular01 的文档中 有npm详细介绍

- 中国镜像
- 已存在, 则需要手动删除 再次安装
- npm 非内部命令

```
npm i -g create-react-app
```

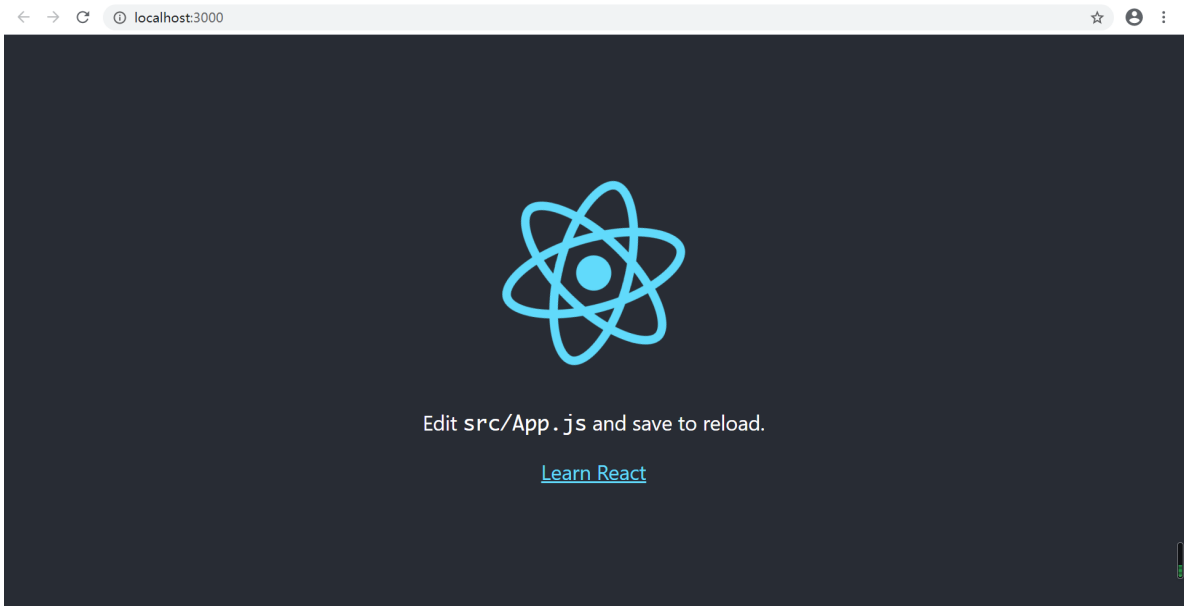
```
+ create-react-app@3.4.1
updated 3 packages in 20.98s
```

生成项目包，找到你希望生成项目的目录下 打开cmd

```
create-react-app 项目名(只能小写字母)

例如:
create-react-app reactpro
```

启动命令：必须在项目执行

```
npm start
```



启动流程

- 启动服务器, 默认端口 3000
- 浏览器访问 端口是3000的程序: `localhost:3000`
- 服务默认有设置: 默认访问的页面名: `index.html`
- `public/index.html` : 此文件的body中有标签: `<div id="root"></div>`
- webpack: 此打包工具会自动把 `index.js` 文件打包引入到 `index.html` 中
- `index.js` 中 把 App.js 的内容加载到 `id='app'` 的元素中

练习

```
// rcc :安装插件之后,快捷提示
import React, { Component } from "react";

export default class App extends Component {
  tags = ["vue", "vuex", "mintUI", "angular", "ionic", "react", "python"];

  showTags() {
    return this.tags.map((item, index) => {
      return <button key={index}>{item}</button>;
    });
```

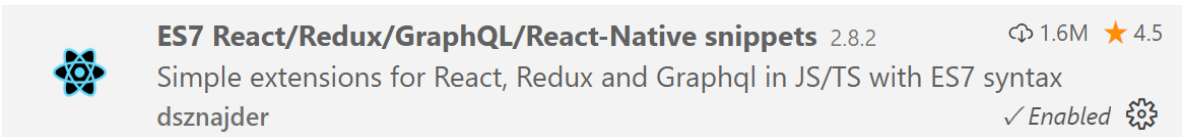
```
  }

  // 专门的 state 属性, 用来 响应式: 数据和UI联动
  state = { num: 5 };

  doAdd() {
    //Cannot read property 'state' of undefined
    //无法 对 未定义 读取属性 state
    // this.state.num++;
    // 只有用 setState() 来修改 state 才会刷新页面

    // ++a 和 a++
    // b = ++a;  先把a+1 然后复制给b
    // a = a++;  先把a赋值给b 再+1
    this.setState({ num: this.state.num + 1 });
  }

  doMinus() {
    this.setState({ num: this.state.num - 1 });
  }

  render() {
    return (
      <div>
        {/* 要想有合适的代码提示, 必须告诉vscode 当前文件是 React 代码! */}
        <h1>Hello World!</h1>
        <div>{this.showTags()}</div>
        <div>
          <button onClick={() => this.doMinus()}>-</button>
          <b>{this.state.num}</b>
          <button onClick={this.doAdd.bind(this)}>+</button>
        </div>

        {/* 通过传参 +1 和 -1 来实现 */}
        <div>
          {/* bind(this, 参数...) */}
          <button onClick={this.changeNum.bind(this, -1)}>-</button>
          <b>{this.state.num}</b>
          <button onClick={() => this.changeNum(1)}>+</button>
        </div>
      </div>
    );
  }

  changeNum(delta) {
    let num = this.state.num + delta;
    // 语法糖: {num: num}  简化为 {num}
    this.setState({ num });
  }
}
```

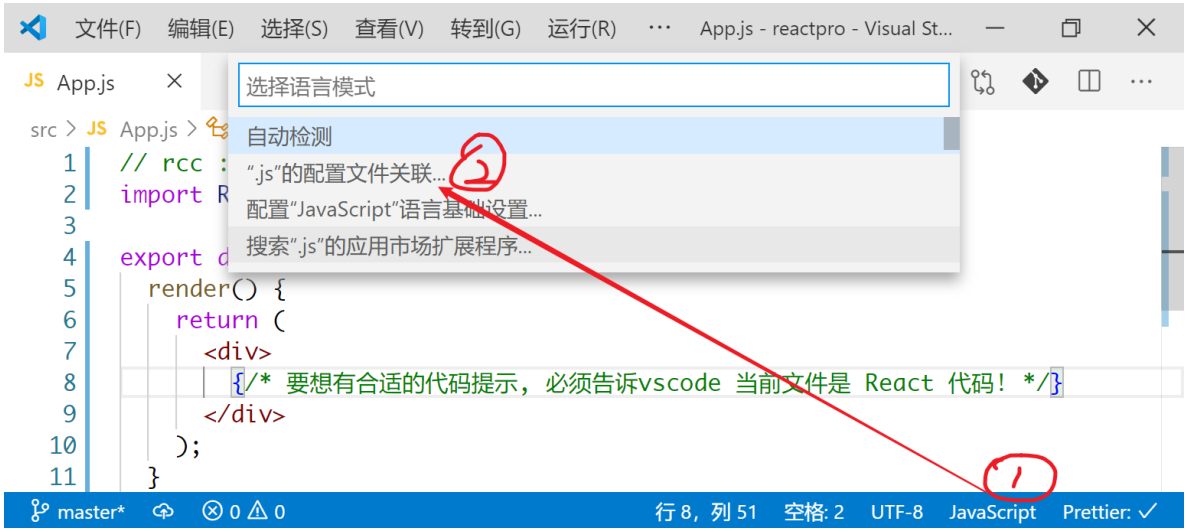
样式

```jsx
// rcc

// 样式: 内联样式 和 外部样式
import React, { Component } from "react";

/**
 * 引入css文件的写法:
 * html引入:  <link ...>
 * css引入: @import 'css文件路径'
 */

// 必须写 ./前缀, 系统才会认为这是个路径 而非 模块名!
import "./App.css";

export default class App extends Component {
  render() {
    return (
      <div>
        {/* 本质是DOM的另一种形式: dom的属性叫 className */}
        <button className="danger">危险</button>
        <button className="success">成功</button>

        <ul>
          {/* DOM中的 style 是对象类型 */}
          {/* 属性名不能带 - ; 所以 font-size 必须写成 fontSize */}
          <li style={{ color: "green", fontSize: "30px" }}>亮亮</li>
          <li></li>
          <li></li>
        </ul>
      </div>
    );
  }
}
```

插件

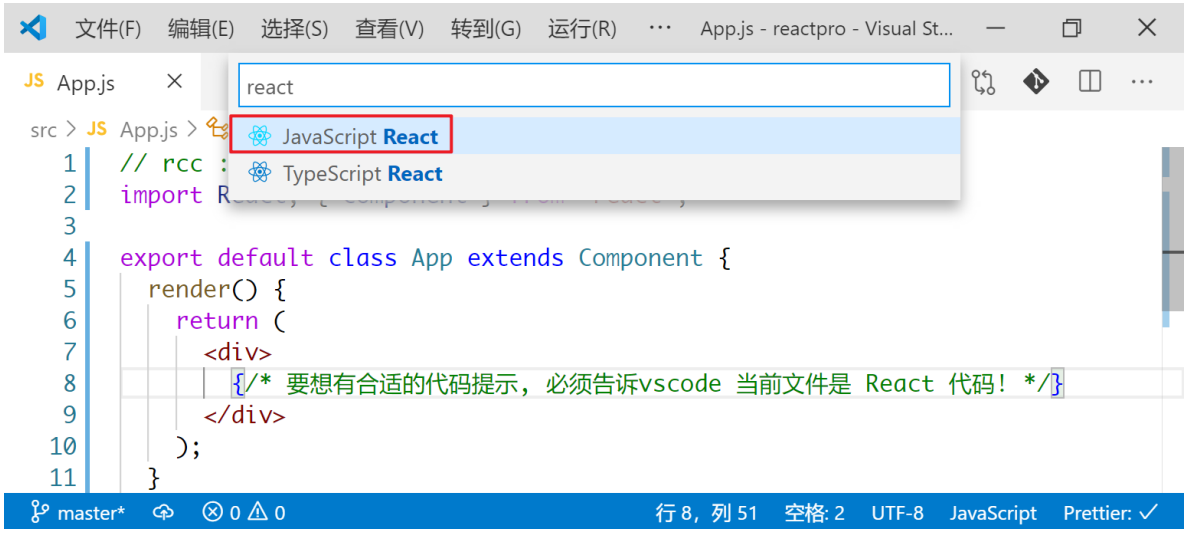
禁用与 angular 相关的插件: `angular`, `ionic`




文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) ··· App.js - reactpro - Visual St...
App.js
选择语言模式
自动检测
"js"的配置文件关联...
配置"JavaScript"语言基础设置...
搜索".js"的应用市场扩展程序...
src > App.js >
// rcc :
import R
export d
render() {
return (

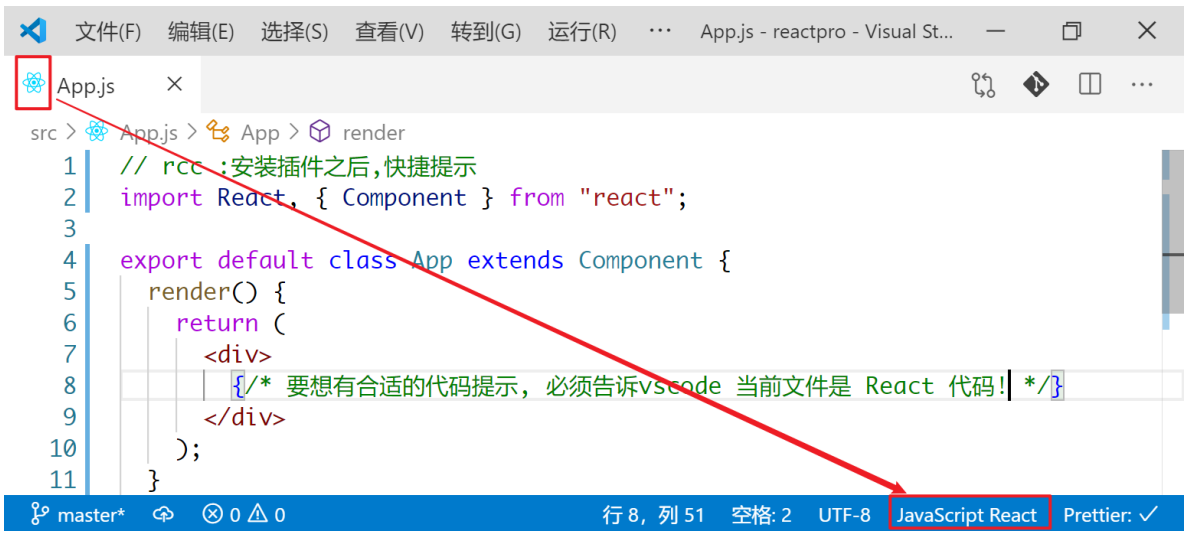
{/* 要想有合适的代码提示，必须告诉vscode 当前文件是 React 代码！ */}

);
}
master*
0 0
行 8，列 51
空格: 2
UTF-8
JavaScript
Prettier:


文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) ··· App.js - reactpro - Visual St...
App.js
react
JavaScript React
TypeScript React
src > App.js >
// rcc :
import R
export default class App extends Component {
render() {
return (

{/* 要想有合适的代码提示，必须告诉vscode 当前文件是 React 代码！ */}

);
}
master*
0 0
行 8，列 51
空格: 2
UTF-8
JavaScript
Prettier:


文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) ··· App.js - reactpro - Visual St...
App.js
src > App.js > App > render
// rcc :安装插件之后，快捷提示
import React, { Component } from "react";
export default class App extends Component {
render() {
return (

{/* 要想有合适的代码提示，必须告诉vscode 当前文件是 React 代码！ */}

);
}
master*
0 0
行 8，列 51
空格: 2
UTF-8
JavaScript React
Prettier:

作业

作业1:

作业2:

制作一个数组, 循环显示出; 每一项后面有个删除按钮, 点击后即可删除对应项目

* SQL 删除
* HTML 删除
* MYSQL 删除
* VUE 删除
* ANGULAR 删除
* PYTHON 删除
* REACT 删除