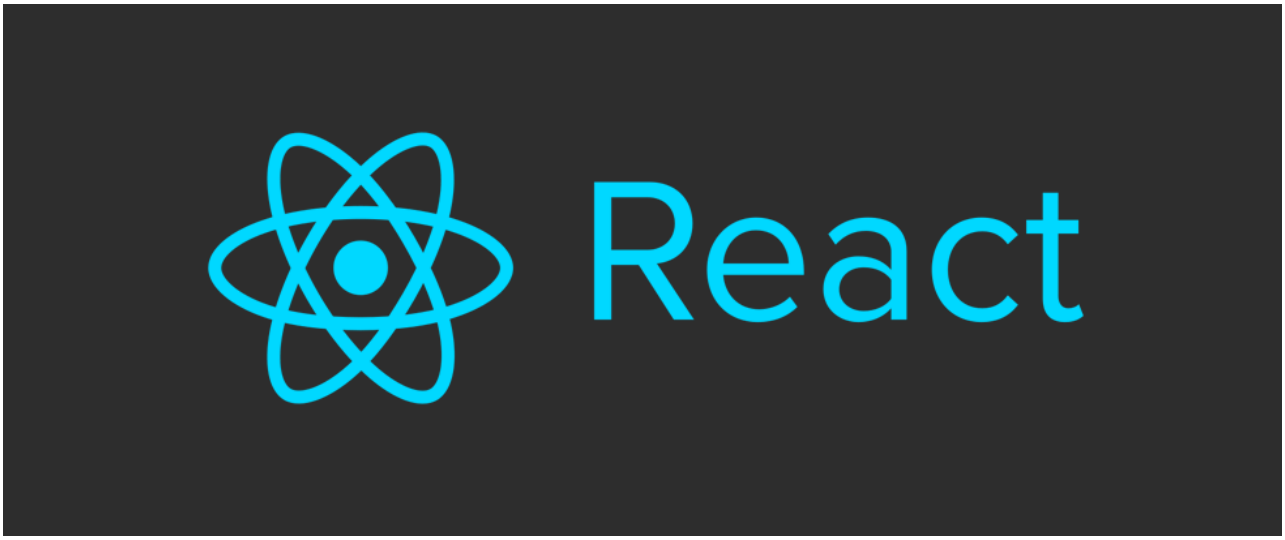
#基础知识



React 是一个用于构建用户界面的 JAVASCRIPT 库。

#创建项目

下面来创建每一个 REACT 项目

1. 首先登录

官网 (opens new window)下载安装NODEJS (opens new window)最新版本

2. Yarn

(opens new window)会缓存它下载的每个包所以无需重复下载，安装速度之快前所未有

```
npm install -g yarn@berry

```

3. 使用

Create React App (opens new window)安装 REACT 项目非常方便，下面来创建项目 houdunren

```
npx create-react-app houdunren

```

4. 进入目录并启动项目

```
cd houdunren
npm start

```

#开发工具

建议使用 VSCODE (opens new window)做为开发工具，需要安装的插件如下（点开链接后，点击 Install 按钮安装）

1. Reactjs code snippets(opens new window)
2. React Extension Pack(opens new window)
3. ES7 React/Redux/GraphQL/React-Native snippets(opens new window)

#声明组件

#基本声明

下面在 index.js 入口文件中创建组件最简单的组件。下面的组件像 HTML 但只是像而已。它被称为 JSX 是一个 JavaScript 的语法扩展，它具有 JavaScript 的全部功能。

- 渲染后的内容将放在 public/index.html 中 ID 为 root 的标签中

```
import React from "react";
import ReactDom from "react-dom";

ReactDom.render(<div>houdunren</div>, document.querySelector("#root"));


```

在 JSX 中可以使用 JS 的功能，要求使用花扩号包裹

```
const name = "后盾人";
ReactDom.render(<div>{name}</div>, document.querySelector("#root"));

```

#函数声明

使用函数返回组件，渲染组件时可以传递参数供组件使用

```
const App = props => {
  return <div>{props.name}</div>;
};

ReactDom.render(App({ name: "后盾人" }), document.querySelector("#root"));

```

调用组件也可以直接使用标签形式，参数以属性形式传递

- 要求首字母大写

```
import React from "react";
import ReactDom from "react-dom";
const App = props => {
  return <div>{props.name}</div>;
};

ReactDom.render(<App name="houdunren.com" />, document.querySelector("#root"));

```

#类的声明

我们知道 JS 中的类也是函数，REACT 也可以使用类的方式声明组件，但要保证返回 JSX 组件标签

```
import React, { Component } from "react";
import ReactDom from "react-dom";
class App {
  (props) {
    this.props = props;
  }
  render() {
    return <div>{this.props.name}</div>;
  }
}

ReactDom.render(
  new App({ name: "后盾人" }).render(),
  document.querySelector("#root")
);
```

```
16
```

如果继承了 Component 基类，会自动绑定参数到 props

```
import React, { Component } from "react";
import ReactDom from "react-dom";
class App extends Component {
  render() {
    return <div>{this.props.name}</div>;
  }
}

ReactDom.render(
  new App({ name: "后盾人" }).render(),
  document.querySelector("#root")
);

```

更好的是，当继承了 Component 基类

- 可以使用标签形式调用组件
- 系统会自动将标签参数绑定到属性 props
- 注意要求首字母大写

```
import React, { Component } from "react";
import ReactDom from "react-dom";
class App extends Component {
  constructor(props) {
    super(props);
    this.props = props;
  }
  render() {
    return (
      <div>
        {this.props.name}
      </div>
    );
  }
}

```

```
ReactDom.render(<App name="后盾人" />, document.querySelector("#root"));

```

基类会帮助我们绑定数据到 props，所以不写构造函数也可以正常执行

```
import React, { Component } from "react";
import ReactDom from "react-dom";
class App extends Component {
  render() {
    return (
      <div>
        {this.props.name}
      </div>
    );
  }
}

ReactDom.render(<App name="后盾人" />, document.querySelector("#root"));

```

#组件嵌套

下面 App 组件内部引入了 Hd 组件

```
import React, { Component } from "react";
import { render } from "react-dom";

class Hd extends Component {
  render() {
    return <div>Hd组件: {this.props.name} </div>;
  }
}

class App extends Component {
  render() {
    return (
      <div>
        <Hd name="houdunren.com" />
        App: {this.props.name}
```

```
      </div>
    );
  }
}
render(<App name="后盾人" />, document.getElementById("root"));

```

#根组件

根据件就像 HTML 标签中的 **html** 一样，所有其它标签都在它的里面。根组件也是这个特点，在里面构建不同组件产生不同界面。

组件一般都是独立的文件，下面创建 App.js 文件构建根组件

```
import React, { Component } from "react";
export default class App extends Component {
  render() {
    return <div>后盾人</div>;
  }
}

```

在入口文件中导入组件并渲染

```
import React, { Component } from "react";
import { render } from "react-dom";
import App from "./App";
render(<App />, document.querySelector("#root"));

```

#注释规范

组件中的注释使用 JS 注释规范，因为是 JS 所以要使用花扩号包裹。

```
class App extends Component {
  render() {
    return (
      <div>
        {/* 后盾人 */}
```

```
        {this.props.name}
      </div>
    );
  }
}

```

#样式处理

下面介绍多种样式的处理方式

#行级样式

REACT 中定义样式也非常简单，下面是定义 STYLE 行样式

```
class App extends Component {
  render() {
    return <div style={{ color: "red" }}>App: {this.props.name}</div>;
  }
}
render(<App name="后盾人" />, document.getElementById("root"));

```

以对象形式声明样式

```
class App extends Component {
  render() {
    const style = {
      backgroundColor: "red",
      color: "blue"
    };
    return <div style={style}>后盾人</div>;
  }
}
render(<App name="后盾人" />, document.getElementById("root"));

```

#类的声明

下面来体验类样式的定义

1. 组件同级目录定义 App.css，内容如下

```
.bg-color {
  background: red;
}

```

2. 在 index.js 中使用 className 属性来声明类

```
import React, { Component } from "react";
import { render } from "react-dom";
import "./App.css";
class App extends Component {
  render() {
    return <div className="bg-color">App: {this.props.name}</div>;
  }
}
render(<App name="后盾人" />, document.getElementById("root"));

```

当然也可以使用 JS 程序计算，下面是使用三元表达式的计算

```
class App extends Component {
  render() {
    return (
      <div className={true ? "bg-color" : "hd"}>App: {this.props.name}</div>
    );
  }
}

```

#第三方库

classnames

classnames (opens new window)是一个动态设置样式类的库，比如不同用户组使用不同样式。

首先来安装库

```
npm i classnames

```

在 index.js 声明的组件中使用

```
import className from "classnames";
import "./App.css";
class App extends Component {
  render() {
    return (
      <div className={className("bg-color", { hd: true })}>
        App: {this.props.name}
      </div>
    );
  }
}
render(<App name="后盾人" />, document.getElementById("root"));

```

styled-components

使用社区提供的第三方库来控制样式，下面是使用 styled-components (opens new window)组件来控制样式

安装扩展包

```
npm i styled-components

```

下面在组件中使用

```
...

//声明样式组件Wrapper 最终渲染成section
const Wrapper = styled.section`
  padding: 4em;
  background: papayawhip;
`;

class App extends Component {
```

```
  render() {
    return (
      <Wrapper>
        <div>App: {this.props.name}</div>
      </Wrapper>
    );
  }
}
render(<App name="后盾人" />, document.getElementById("root"))

```

#实例操作

下面来开发用户展示模块，只实现页面 UI 的展示，具体业务功能需要后面其他知识点

编号	姓名	年龄
1	后盾人	18

首先确定几点

- 组件会牵扯到多个文件，所以组件最好在独立目录中存放如 components/User
- 多个组件使用统一文件合并导出 components/index.js

目录结构如下

```
src/components
├── Add
|   ├── index.js        #搜索组件
|   └── index.css #组件样式
├── User
|   ├── User.js         #用户记录组件
|   ├── index.js        #用户列表组件
|   └── index.css #组件样式
├── index.js                    #组件合并导出文件
├── App.js                              #根组件
└── App.css                             #根组件样式

```

#代码展示

下面是代码展示，有几点需要说明

- 每个组件单独一个文件夹
- 组件文件夹中存在组件需要的其他文件，如 index.css 样式文件
- User/User.js 是 User/index.js 用户列表组件分离出的私有组件

src/components/Add/index.js

```
import React, { Component } from "react";
import "./index.css";
export default class Add extends Component {
  render() {
    return (
      <div className="add">
        <input/>
      </div>
    );
  }
}

```

src/components/Add/index.css

```
.add {
    margin-bottom: 10px;
}

.add input {
    border: solid 2px #34495e;
    padding: 10px;
    font-size: 20px;
    width: 95%;
}

```

src/components/User/index.js

```
import React, { Component } from "react";
```

```
import User from "./User";
import "./index.css";
export default class List extends Component {
  render() {
    return (
      <div>
        <table border="1" width="100%">
          <thead>
            <tr>
                <th>编号</th>
              <th>姓名</th>
              <th>年龄</th>
            </tr>
            <User />
          </thead>
        </table>
      </div>
    );
  }
}

```

src/components/User/user.js

```
import React, { Component } from "react";

export default class User extends Component {
  render() {
    return (
      <tr align="center">
        <td>1</td>
        <td>后盾人</td>
        <td>18</td>
      </tr>
    );
  }
}

```

src/components/User/index.css

```css
.add {
    margin-bottom: 10px;
}

.add button {
    background: #34495e;
    color: white;
    font-size: 20px;
    border: none;
    cursor: pointer;
}

.add input {
    border: solid 2px #34495e;
    padding: 10px;
    font-size: 20px;
    min-width: 100px;
    font-weight: bold;
}

.add button:focus,
.add input:focus {
    outline: none
}

```

app.js

```jsx
import React, { Component } from "react";
import { List, Add } from "./components/index";
import "./App.css";
export default class App extends Component {
  render() {
    return (
      <div className="app">
        <Add />
        <List />
      </div>
```

```
    );
  }
}

```

app.css

```
.app {
  background: #f3f3f3;
  padding: 10px;
  border: solid 2px #ddd;
}

```

#顶级标签

下面介绍顶级标签产生的问题，及解决方法

#基本知识

组件必须必须存在一个顶级标签，下面的是正确格式

```
export default class App extends Component {
  render() {
    return (
      <div>
        <Add />
        <List />
      </div>
    );
  }
}

```

下面是错误的格式

```
export default class App extends Component {
  render() {
    return (
```

```
        <Add />
        <List />
    );
  }
}

```

#问题说明

以前面讲解的学生模块为例，因为每个组件都有一个顶级标签，最终生成的 HTML 标签结构如下

```
▼<div id="root">
  ▼<div class="app">
    ▼<div>
       <input class="input">
     </div>
    ▼<div>
     ▶<table border="1" width="100%">…</table>
     </div>
   </div> == $0
 </div>
```


但是现在我们发现一个问题，就是文本框不能 100%对齐

编号	姓名	年龄
1	后盾人	18

#解决问题

针对上面的问题，我们希望使用 FLEX 来解决，这就要求每个组件不能有顶级标签，有以下几种解决方案

fragment

使用 `Fragment` 组件可以让最终生成的 HTML 没有顶级标签

```
import React, { Component, Fragment } from "react";

```

```
export default class Add extends Component {
  render() {
    return (
      <Fragment>
        <input className="input"/>
      </Fragment>
    );
  }
}

```

空标签

也可以使用以下特殊标签实现 fragment 相同的效果

```
function App() {
  return (
    <>
      houdunren.com
    </>
  )
}

```

现在将 src/components/User/index.js 与 src/components/Add/index.js 两个组件使用 fragment 或空标签来处理。最终生成的 HTML 结构如下图所示

现在我们来修改相关样式文件

App.css

```
.app {
  background: #f3f3f3;
  display: flex;
  flex-direction: column;
```

```
  padding: 10px;
  border: solid 2px #ddd;
}

```

在顶级组件 App.js 中设置类

```
import React, { Component } from "react";
import { List, Add } from "./components/index";
import "./App.css";
export default class App extends Component {
  render() {
    return (
      <div className="app">
        <Add />
        <List />
      </div>
    );
  }
}

```

最终修正后的效果如图

编号	姓名	年龄
1	后盾人	18

#搜索按钮

下面来搜索添加按钮，需要修改两个组件文件和两个样式文件

src/components/Add/index.js 因为页面结构有变化，所以删除 fragment 改变 div 标签

```
import React, { Component, Fragment } from "react";
import "./index.css";
export default class Add extends Component {
  render() {
    return (
      <div className="search">
        <input/>
        <button>后盾人</button>
      </div>
    );
  }
}

```

src/components/Add/index.css

```
.add {
    display: flex;
    margin-bottom: 10px;
}

button {
    background: #34495e;
    color: white;
    font-size: 20px;
    border: none;
    cursor: pointer;
}

```

```
input {
    border: solid 2px #34495e;
    padding: 10px;
    font-size: 20px;
    flex: 1;
    min-width: 100px;
}

button:focus,
input:focus {
    outline: none
}

```