React

渲染 UI 的 JavaScript 库

React 介绍

- React 是一个用来渲染用户界面（UI）的 JavaScript 库
- 开发者：Meta（Facebook）
- React 是世界上应用最广泛的前端库，是国内一线大厂（阿里、字节等）前端的首选
- 特色：
  - React 完全基于 JavaScript，只要有 JS 基础，就可以上手 React 开发
  - 使用 JS 来编写 HTML，完全符合 JS 的编程习惯

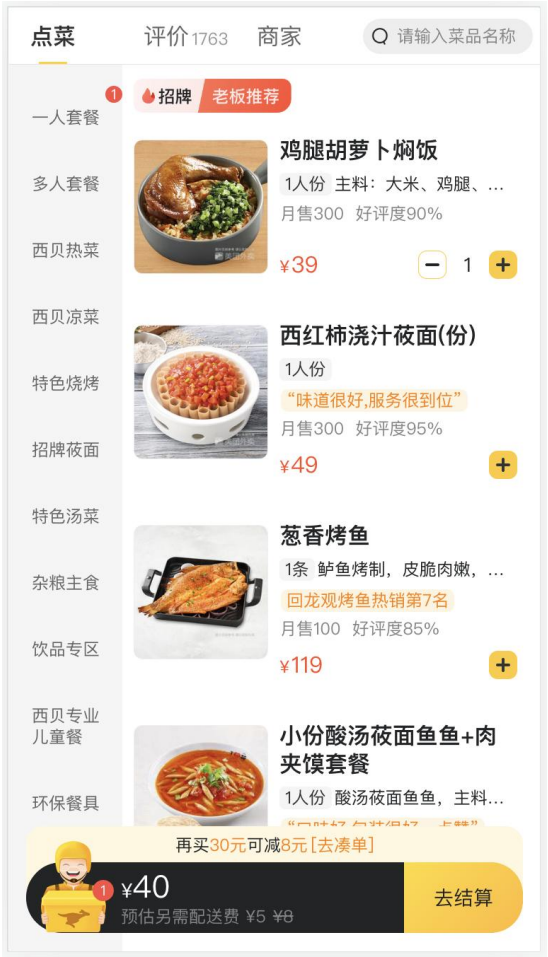
React 快速体验

- 步骤
  1. 使用 create-react-app 创建项目
  2. 修改 App.js 体验用 JavaScript 生成结构、使用样式及控制逻辑
- 收获
  - React 完全基于 JavaScript，只要有 JS 基础，就可以上手 React 开发

React 学习路径

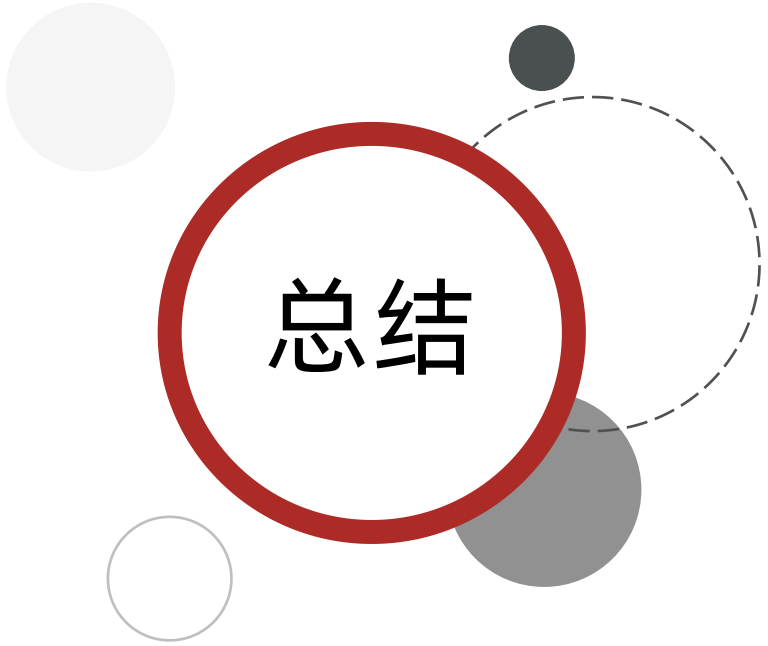
1. JSX（编写页面结构）
2. 组件、组件通讯
3. React Hooks
4. Redux（状态管理工具）
5. 路由

- 综合案例



React 课程收获

1. 开发 H5 或 PC 端项目
2. 单页应用程序（SPA）
3. 组件化
4. 状态管理
5. 提升 JavaScript 的编程能力



1. React 是渲染用户界面（UI）的 JavaScript 库

2. React 完全基于 JavaScript

   用 JavaScript 可以生成结构、使用样式及控制逻辑

3. 学习路径

   JSX → 组件 → Hooks → 状态管理 → 路由 → 综合案例

开发环境搭建

React 脚手架

Getting started with a minimal toolchain

If you're **learning React,** we recommend Create React App. It is the most popular way to try out React and build a new single-page, client-side application. It's made for React but isn't opinionated about routing or data fetching.

官方推荐，使用 React 脚手架 CRA（Create React App）搭建 React 的开发环境

使用 CRA 搭建开发环境（项目）分两步：

1. 创建项目
2. 启动项目

创建项目和启动项目

创建项目命令：

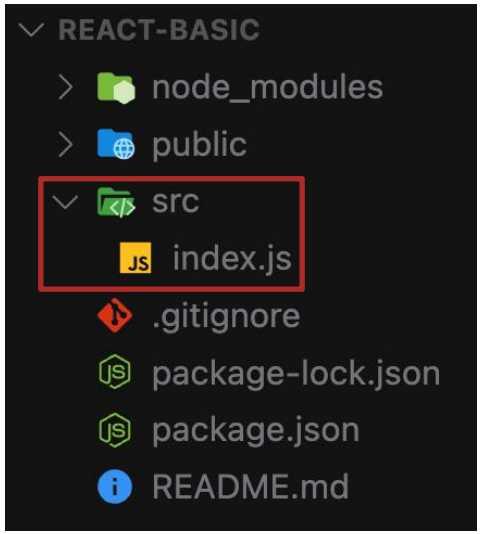
npx create-react-app react-basic

React 脚手架命令名称

项目名称

```
Success! Created react-basic at /Users/zqran/ReactCourse/react-basic
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd react-basic
  npm start

Happy hacking!
```

目录结构说明和调整



- public 目录：不做任何改动，不要删除 index.html 模板
- src 目录：只保留 index.js 项目入口文件，并清空 index.js 内容

React 初体验

效果演示

需求：渲染一个 h1 元素，内容为：Hello, React

步骤：

1. 导包

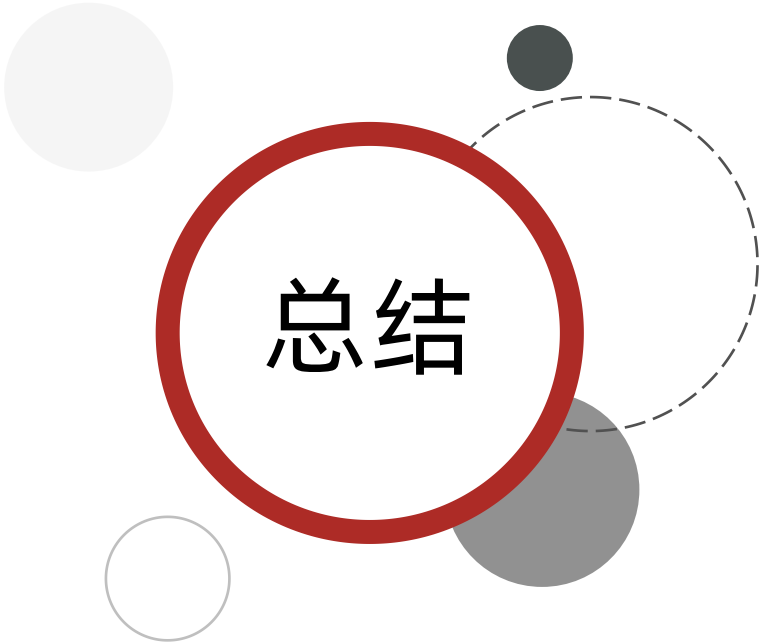
```
import { createRoot } from 'react-dom/client'
```

2. 创建 React 根对象

```
const root = createRoot(DOM对象)
```

3. 使用根对象，渲染 React 内容

```
root.render(React内容)
```



1. React 渲染内容（比如，h1 标签）到页面中，分哪三步？

```
// 1. 导包
import { createRoot } from 'react-dom/client'
// 2. 创建 React 根对象
const root = createRoot(document.querySelector('#root'))
// 3. 渲染 React 内容
root.render(<h1>Hello, React</h1>)
```

2. 要渲染什么内容，直接写类似 HTML 的结构即可，它叫做：JSX

```
root.render(<h1>Hello, React</h1>)
```

JSX（写在 JS 中的 HTML）

JSX 的使用和原理

JSX 的使用和原理

> *JSX* is a syntax extension for JavaScript that lets you write HTML-like markup inside a JavaScript file.

JSX 允许你在 JS 文件中写类似 HTML 的标记，是 JS 的语法扩展

1. 作用：写 React 页面结构
2. 原理：JSX 是 JS 的语法扩展

```
root.render(<h1>Hello, React</h1>)
```

JSX 的原理

JSX 本质上是 JS 对象

体验地址

JSX

```
<h1>Hello World</h1>
```

JS 的语法扩展

JSX 转换器（CRA 提供）

```
React.createElement('h1', null, 'Hello World')
```

标准的 JS 语法

JS 对象
{ type, props }

返回值

JSX 的使用和原理

1. JSX 的使用

作用：写 React 页面结构，类似 HTML

规则：① 唯一根节点 ② 标签闭合 ③ 属性名称驼峰命名法

```
<img className="avatar" src="..." />
```

2. JSX 的原理

本质：JS 对象 —— { type, props }

JS 的语法扩展　　JSX 转换器（CRA 提供）　　返回值

写 JSX 推荐的 VSCode 配置

写 JSX 推荐的 VSCode 配置

```
/**
1. JS 文件中启用 Emmet 语法支持
"emmet.includeLanguages": {
  "javascript": "javascriptreact”
},

2. 按照两个插件：
- Prettier - Code formatter  自动格式化 JSX 代码
    - 保存时自动格式化代码： "editor.formatOnSave": true
    - 设置为默认格式化工具： "editor.defaultFormatter": "esbenp.prettier-vscode",
- Auto Rename Tag  自动修改成对的标签名称
    - 无需额外配置
*/
```

JSX 中 {} 的应用

JSX 中 {} 的应用

```
<div>{ JavaScript 表达式 }</div>
```

作用：让 JSX 变为动态的，可以用在标签内容、属性中

```
const categories = [
  { id: 1, name: '推荐' },
  { id: 2, name: '一人套餐' },
  { id: 3, name: '西贝凉菜' },
  { id: 4, name: '西贝热菜' },
  { id: 5, name: '杂粮主食' },
]
```

映射

map　　&& 或 ? :　　className

常见应用场景：① 列表渲染　② 条件渲染　③ 样式处理

React 事件绑定

React 事件绑定

语法：on + 事件名称 = { 事件处理程序 }，整体上遵循驼峰命名法

```
<div onClick={() => {}} />
```

单独创建事件处理程序函数：

```
const handleClick = () => {}
<div onClick={handleClick} />
```

事件对象：

```
<a onClick={e => e.preventDefault()} href="..."></a>
```

注意：

```
<div onClick={handleClick()} />
<div onClick={handleClick} />
```

React 组件

React 组件

> a React component is a JavaScript function that you can *sprinkle with markup.*

React 组件是一个 JavaScript 函数，函数中可以添加 JSX 标记

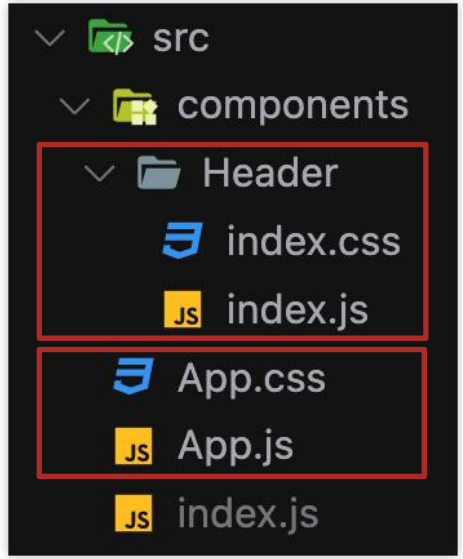
创建组件：

```
const App = () => {
  return <div>React Component</div>
}
```

渲染组件：

```
root.render(<App />)
```

注意点：组件名称首字母大写



组件的状态

组件的状态

状态（state）是可以让页面内容发生变化的数据

使用状态：useState 函数

```
const [count, setCount] = useState(10)
```

存储状态的变量 修改状态的函数 默认值


```
setCount(count + 1)
```

状态最新值

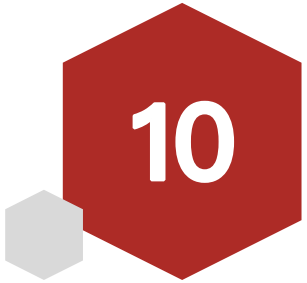
修改状态的函数：

1. 修改状态
2. 触发组件重新渲染

useState：

1. 第一次调用：返回默认值
2. 以后每次调用：返回状态最新值

React 也是**数据驱动视图**的开发模式



修改状态的规则

修改状态的规则

规则：不要直接修改当前状态的值，应该创建新值 —— 状态不可变

注意：如果没有遵循该规则，会导致报错或组件无法重新渲染

简单类型：

```
const [count, setCount] = useState(0)
```

不要：

```
count += 1
setCount(count)
```

应该：

```
setCount(count + 1)
```

复杂类型：

```
const [list, setList] = useState(['苹果'])
const [user, setUser] = useState({ name: '传智' })
```

不要：

```
list.push('橘子')
setList(list)
```

```
user.name = '黑马'
setUser(user)
```

应该：

```
setList([...list, '橘子'])
```

```
setUser({ ...user, name: '黑马' })
```

案例：B站评论

B站评论案例

- 使用 App 组件，搭建结构（① 使用 SASS 样式 ② 使用本地图片）

使用组件搭建结构

导航 Tab 的渲染和操作

评论列表的渲染和操作

评论列表的渲染和操作

1. 根据状态渲染评论列表
2. 删除评论
3. 喜欢和不喜欢

使用组件搭建结构 √

导航 Tab 的渲染和操作

★ 评论列表的渲染和操作

导航 Tab 的渲染和操作

1. 渲染导航 Tab 和高亮
2. 评论列表排序

使用组件搭建结构 √

★ 导航 Tab 的渲染和操作

评论列表的渲染和操作 √

B站评论案例总结

使用组件搭建结构 √

- 组件的使用
- SASS 样式处理

评论列表的渲染和操作 √

导航 Tab 的渲染和操作 √

B站评论案例总结

评论列表的渲染和操作 √

- 状态的使用

```
const [list, setList] = useState(defaultList)
```

- 列表渲染（数组的 map）

```
{list.map(item => {
  return (
    <div key={item.rpid} className="reply-item">…
    </div>
  )
})}
```

- 条件渲染（逻辑与 &&）

```
{user.uid === item.user.uid && (
  <span className="delete-btn">删除</span>
)}
```

- 删除数组元素（filter）

```
setList(list.filter(item => item.rpid !== rpid))
```

- 修改数组元素（map）

```
setList(list.map(item => {
  if (item.rpid === rpid) { ... }
  return item
}))
```

B站评论案例总结

导航 Tab 的渲染和操作 √

- 导航 Tab 高亮

```
const tabs = [
  { type: 'hot', text: '最热' },
  { type: 'time', text: '最新' },
]
const [activeTab, setActiveTab] = useState('hot')
```

- lodash 的 orderBy 实现排序

```
orderBy(要排序的数组, 按照谁排, 顺序)
orderBy(list, 'like', 'desc')
```

使用 classnames 优化类名处理

使用 classnames 优化类名处理

问题：项目开发中，当多个类名都是动态的，手动处理会变得非常困难

```
<button className={disabled ? 'btn btn-disabled' : 'btn'}></button>
```

推荐：使用 classnames 包来优化类名处理

1. 装包：npm i classnames
2. 导包：`import classNames from 'classnames'`
3. 使用：
   1. 逻辑与（&&）—— 处理单个类名

```
classNames('btn', disabled && 'btn-disabled')
```

   2. 对象语法 —— 处理多个类名

```
classNames('btn',
  {
    'btn-disabled': disabled,
    'btn-small': size === 'small',
    'btn-large': size === 'large',
  }
)
```

传智教育旗下高端IT教育品牌