React介绍

React是什么

React由Meta公司研发，是一个用于 构建Web和原生交互界面的库

React的优势

相较于传统基于DOM开发的优势

- 组件化的开发方式
- 不错的性能

相较于其它前端框架的优势

- 丰富的生态
- 跨平台支持

React的市场情况

全球最流行，大厂必备

(index)	name	counts
0	'react'	18130822
1	'vue'	13642794
2	'element-ui'	4912319
3	'antd'	3873399
4	'element-plus'	2138717
5	'ant-design-vue'	1888752
6	'@angular/core'	1027802
7	'next'	918282
8	'nuxt'	422214
9	'@mui/material'	383488
10	'vuetify'	265469
11	'svelte'	226658

职位描述

React　TypeScript　RESTful API　Git

职责描述：

1. 使用 React 及 TypeScript 开发 RightCapital 核心系统
2. 参与代码审核、设计审核等工作
3. Web 前沿技术研究和新技术调研

职位要求：

1. 具有 React 及 TypeScript 项目开发经验
2. 熟悉 React 生态
3. 具有英文文档阅读能力
4. 具备良好的团队协作精神
5. 对前端技术有持续的热情，逻辑性强
6. 有良好的代码习惯

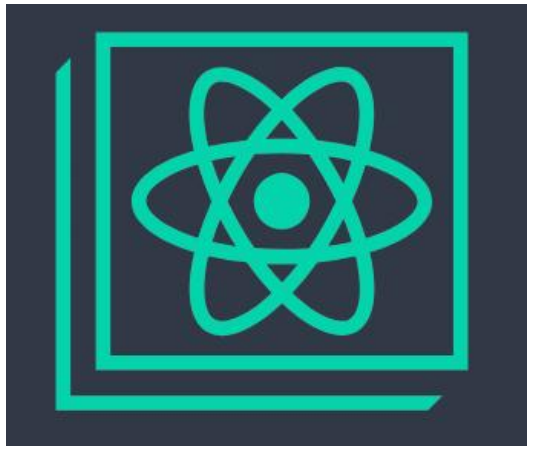
所使用的技术栈：

1. React
2. TypeScript
3. RESTful API
4. Git 版本控制， Gitflow 工作流

开发环境搭建

使用create-react-app快速搭建开发环境

create-react-app是一个快速 创建React开发环境的工具，底层由Webpack构建，封装了配置细节，开箱即用



执行命令：

```
npx create-react-app react-basic
```

1. npx Node.js工具命令，查找并执行后续的包命令
2. create-react-app 核心包（固定写法），用于创建React项目
3. react-basic React项目的名称（可以自定义）

创建React项目的更多方式

https://zh-hans.react.dev/learn/start-a-new-react-project

JSX基础–概念和本质

什么是JSX

概念：JSX是JavaScript和XML（HTML）的缩写，表示在JS代码中编写HTML模版结构,它是React中编写UI模版的方式

```
const message = 'this is message'
function App () {
  return (
    <div>
      <h1>this is title</h1>
      {message}
    </div>
  )
}
```

优势：

1. HTML的声明式模版写法　2. JS的可编程能力

JSX的本质

JSX并不是标准的JS语法，它是JS的语法扩展，浏览器本身不能识别，需要通过解析工具做解析之后才能在浏览器中运行

```
<div>
  this is div
</div>

```



```
import { jsx as _jsx } from
"react/jsx-runtime";
/*#__PURE__*/_jsx("div", {
  children: "this is div"
});
```

JSX基础–高频场景

JSX中使用JS表达式

在JSX中可以通过 大括号语法{} 识别 JavaScript中的表达式，比如常见的变量、函数调用、方法调用等等

1. 使用引号传递字符串
2. 使用JavaScript变量
3. 函数调用和方法调用
4. 使用JavaScript对象

注意：if语句、switch语句、变量声明属于语句，不是表达式，不能出现在{}中

JSX中实现列表渲染

```
<ul>
  <li>凯瑟琳·约翰逊：数学家</li>
  <li>马里奥·莫利纳：化学家</li>
  <li>穆罕默德·阿卜杜勒·萨拉姆：物理学家</li>
  <li>珀西·莱温·朱利亚：化学家</li>
  <li>苏布拉马尼扬·钱德拉塞卡：天体物理学家</li>
</ul>
```

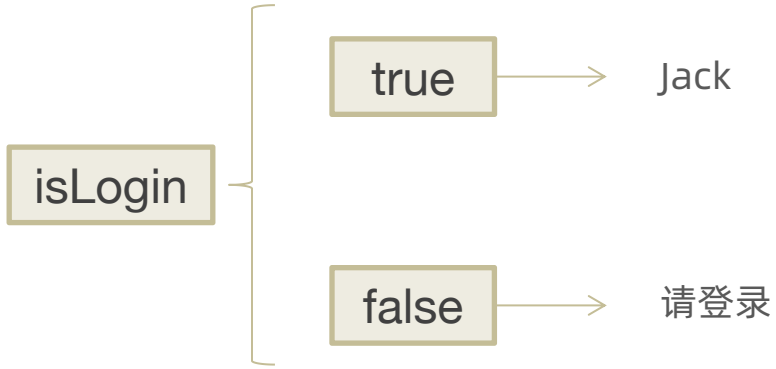
语法：在JSX中可以使用原生JS中的map方法遍历渲染列表

```
const list = [
  { id: 1001, name: 'Vue' },
  { id: 1002, name: 'React' },
  { id: 1003, name: 'Angular' }
]
```



```
<ul>
  {list.map(item =><li key={item.id}>{item}</li>)}
</ul>
```

JSX中实现条件渲染



语法：在React中，可以通过逻辑与运算符&&、三元表达式（?:）实现基础的条件渲染

```
{flag && <span>this is span</span>}
```

```
{loading ? <span>loading...</span> : <span>this is span</span>}
```

JSX中实现复杂条件渲染

穷小子请婚假被开除，不料临走时敲了几下键盘，下秒整个公司瘫痪

香港电影看不停　121评论　1月前

拜登离开不到一周，越南总理访华，见华为高层，将做美不喜欢的事

贺文萍　4评论　1小时前

官方再通报“遵义一医院问题招聘”：区卫健局长等9人被查

澎湃新闻　35评论　1小时前

需求：列表中需要根据文章状态适配三种情况，单图，三图，和无图三种模式

解决方案：自定义函数 + if判断语句

React中的事件绑定

React 基础事件绑定

语法：on + 事件名称 = { 事件处理程序 }，整体上遵循驼峰命名法

```
function App () {
  const clickHandler = () => {
    console.log('button按钮点击了')
  }
  return (
    <button onClick={clickHandler}></button>
  )
}
```

使用事件对象参数

语法：在事件回调函数中设置形参e

```
function App() {
  const clickHandler = (e) => {
    console.log('button按钮点击了', e)
  }
  return <button onClick={clickHandler}>click me</button>
}
```

传递自定义参数

语法：事件绑定的位置改造成箭头函数的写法，在执行clickHandler实际处理业务函数的时候传递实参

```
function App() {
  const clickHandler = (name) => {
    console.log('button按钮点击了', name)
  }
  return <button onClick={() => clickHandler('jack')}>click me</button>
}

```

注意：不能直接写函数调用，这里事件绑定需要一个函数引用

同时传递事件对象和自定义参数

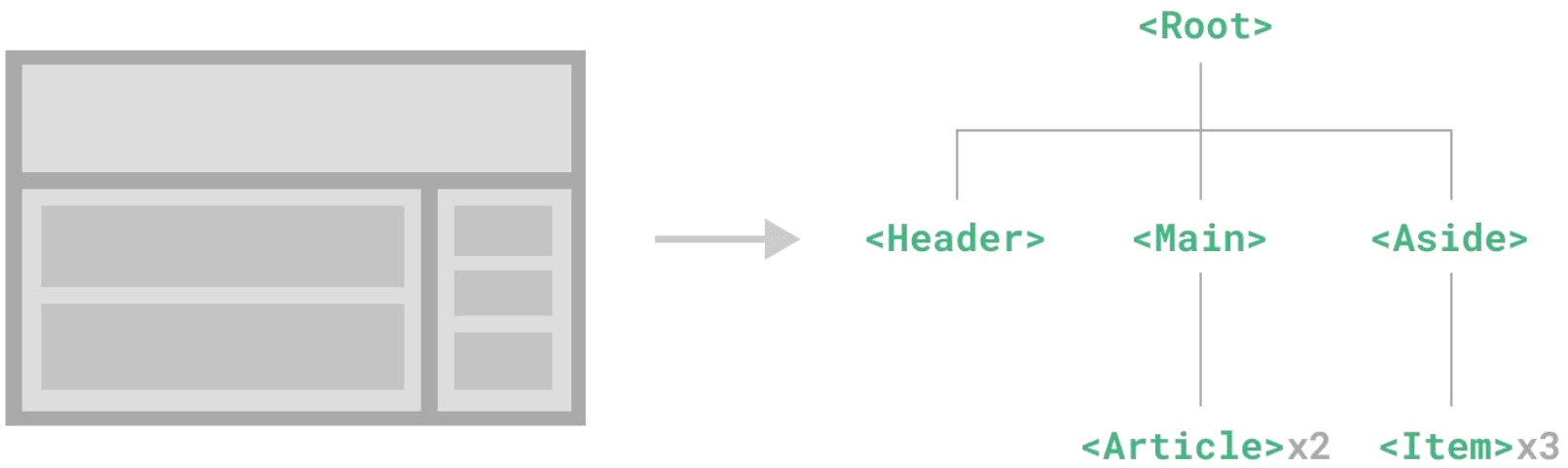
语法：在事件绑定的位置传递事件实参e和自定义参数，clickHandler中声明形参，注意顺序对应

```
function App() {
  const clickHandler = (name, e) => {
    console.log('button按钮点击了', name, e)
  }
  return <button onClick={(e) => clickHandler('jack', e)}>click me</button>
}
```

React中的组件

组件是什么

概念：一个组件就是用户界面的一部分，它可以有自己的逻辑和外观，组件之间可以互相嵌套，也可以复用多次



组件化开发可以让开发者像搭积木一样构建一个完整的庞大的应用

React组件

在React中，一个组件就是首字母大写的函数，内部存放了组件的逻辑和视图UI，渲染组件只需要把组件当成标签书写即可

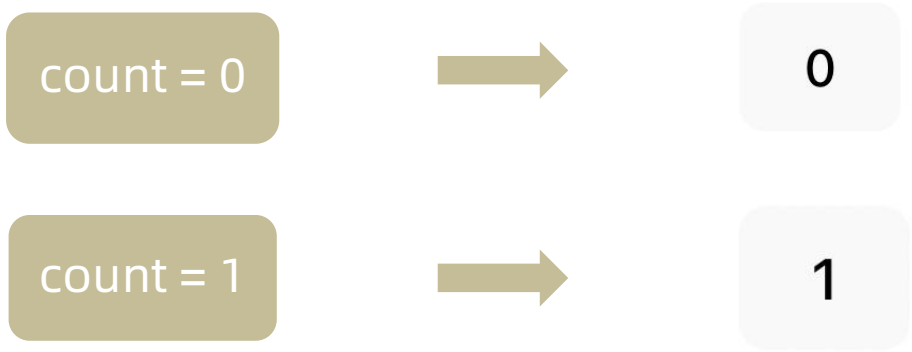
```
// 1. 定义组件
function Button() {
  // 组件内部逻辑
  return <button>click me</button>
}
```

```
// 2. 使用组件
function App() {
  return (
    <div>
      {/* 自闭和 */}
      <Button />
      {/* 成对标签 */}
      <Button></Button>
    </div>
  )
}
```

useState

useState基础使用

useState 是一个 React Hook（函数），它允许我们向组件添加一个状态变量，从而控制影响组件的渲染结果



本质：和普通JS变量不同的是，状态变量一旦发生变化组件的视图UI也会跟着变化（数据驱动视图）

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

1. useState是一个函数，返回值是一个数组
2. 数组中的第一个参数是状态变量，第二个参数是set函数用来修改状态变量
3. useState的参数将作为count的初始值

修改状态的规则

状态不可变

在React中，状态被认为是只读的，我们应该始终替换它而不是修改它，直接修改状态不能引发视图更新

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

const handleClick = () => {
  // 直接修改 无法引发视图更新
  count++
  console.log(count)
}
```

```
const handleClick = () => {
  // 作用:
  // 1. 用传入的新值修改count
  // 2. 重新使用新的count渲染UI
  setCount(count + 1)
}
```

修改对象状态

规则：对于对象类型的状态变量，应该始终传给set方法一个全新的对象来进行修改

```
const [form, setForm] = useState({
  name: 'jack',
})

const handleChangeName = () => {
  form.name = 'john'
}
```

直接修改原对象，不引发视图变化

```
const [form, setForm] = useState({
  name: 'jack',
})

const handleChangeName = () => {
  setForm({
    ...form,
    name: 'john',
  })
}
```

调用set传入新对象用于修改

组件的样式处理

组件基础样式方案

React组件基础的样式控制有俩种方式

1. 行内样式（不推荐）

```jsx
<div style={{ color: 'red' }}>this is div</div>
```

2. class类名控制

```css
.foo {
  color: red;
}
```

index.css

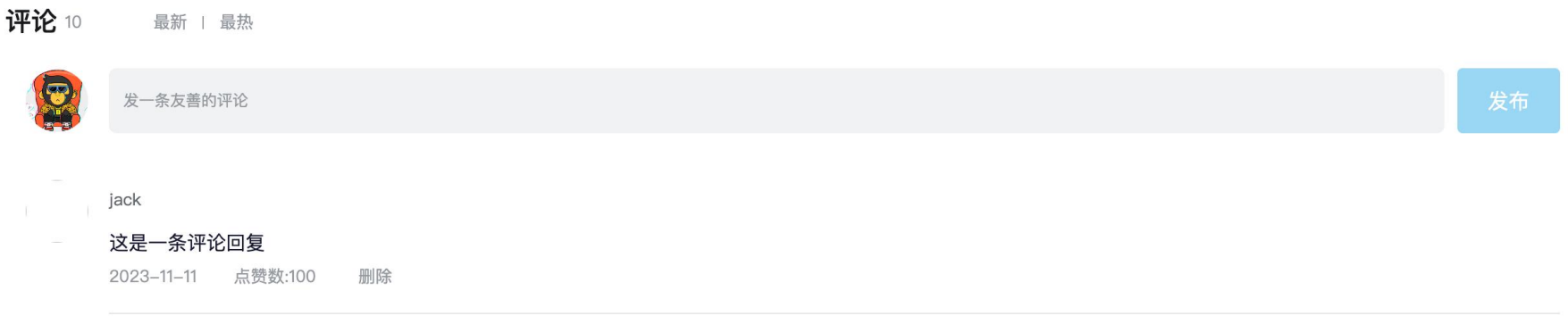
```jsx
import './index.css'

function App () {
  return (
    <div>
      <span className='foo'>this is span</span>
    </div>
  )
}
```

App.js

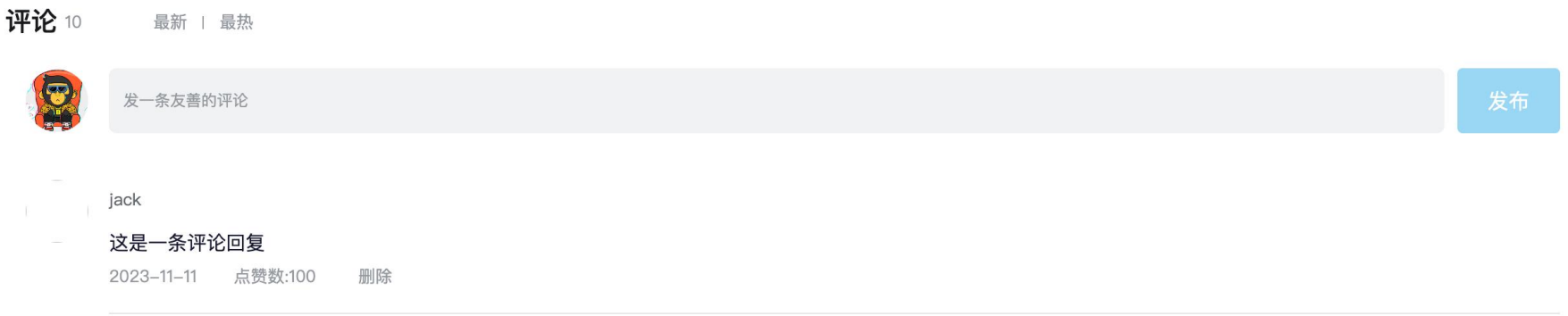
案例：B站评论

B站评论案例



1. 渲染评论列表
2. 删除评论实现
3. 渲染导航Tab和高亮实现
4. 评论列表排序功能实现

渲染评论列表

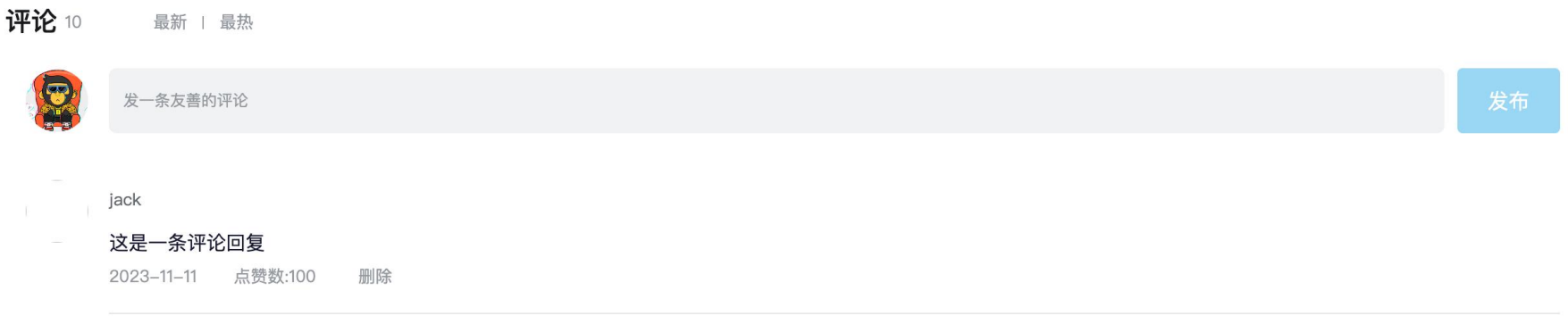


核心思路：

1. 使用useState维护评论列表

2. 使用map方法对列表数据进行遍历渲染（别忘了加key）

实现评论删除



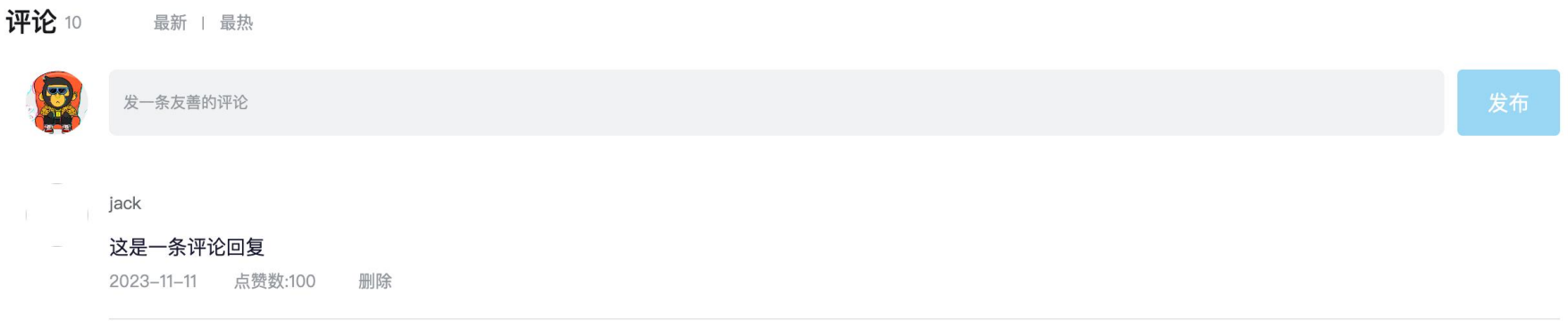
需求：

1. 只有自己的评论才显示删除按钮
2. 点击删除按钮，删除当前评论，列表中不再显示

核心思路

1. 删除显示 - 条件渲染　　2. 删除功能 - 拿到当前项id以id为条件对评论列表做filter过滤

渲染Tab+点击高亮实现



需求：

点击哪个tab项，哪个做高亮处理

核心思路：

点击谁就把谁的type（独一无二的标识）记录下来，然后和遍历时的每一项的type做匹配，谁匹配到就设置负责高亮的类名

classnames优化类名控制

classnames是一个简单的JS库，可以非常方便的通过条件动态控制class类名的显示

```
<span
  key={item.type}
  onClick={() => handleTabChange(item.type)}
  className={`nav-item ${type === item.type && 'active'}`}>
  {item.text}
</span>)}
```

现在的问题：字符串的拼接方式不够直观，也容易出错

```
className={classNames('nav-item', { active: type === item.type })}
```

静态的类名　　动态类名 key表示要控制的类名，value表示条件，true的时候类名显示

传智教育旗下高端IT教育品牌