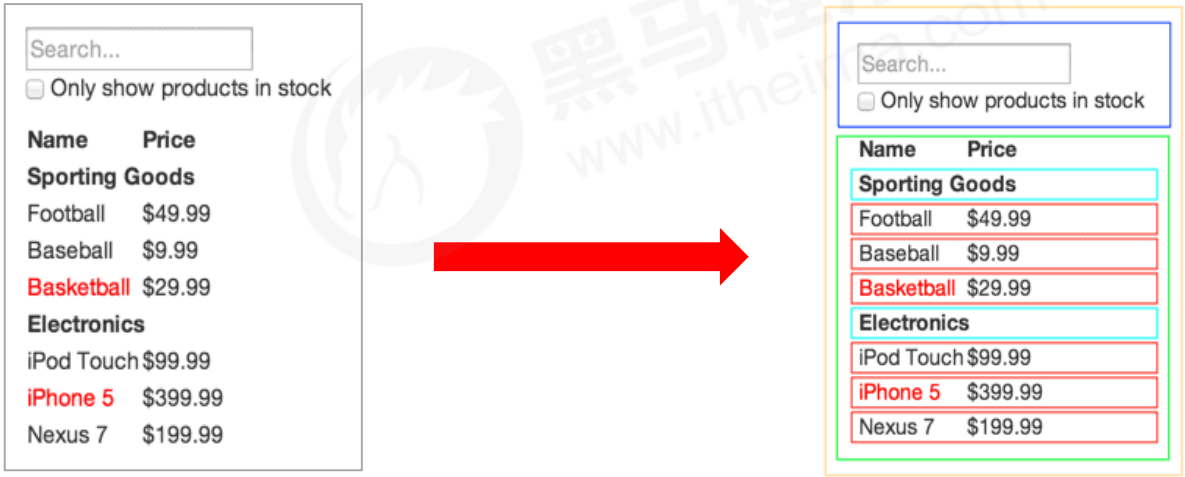

黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
React 组件基础

目录 Contents

- React 组件介绍
- React 组件的两种创建方式
- React 事件处理
- 有状态组件和无状态组件
- 组件中的 state 和 setState()
- 事件绑定 this 指向
- 表单处理

1. React 组件介绍

- 组件是 React 的**一等公民**，使用 React 就是在用组件
- 组件表示页面中的部分功能
- 组合多个组件实现完整的页面功能
- 特点：可复用、独立、可组合



目录 Contents

- React 组件介绍
- React 组件的两种创建方式
- React 事件处理
- 有状态组件和无状态组件
- 组件中的 state 和 setState()
- 事件绑定 this 指向
- 表单处理

2. React 组件的两种创建方式

1. 使用函数创建组件
2. 使用类创建组件

2. React 组件的两种创建方式

2.1 使用函数创建组件

- 函数组件：使用 JS 的函数（或箭头函数）创建的组件
- 约定1：函数名称必须以大写字母开头
- 约定2：函数组件必须有返回值，表示该组件的结构
- 如果返回值为 null，表示不渲染任何内容

```
function Hello() {
  return (
    <div>这是我的第一个函数组件！</div>
  )
}
```

2. React 组件的两种创建方式

2.1 使用函数创建组件

- 渲染函数组件：用函数名作为组件标签名
- 组件标签可以是单标签也可以是双标签

```
function Hello() {
  return (
    <div>这是我的第一个函数组件！</div>
  )
}
ReactDOM.render(<Hello />, root)
```

2. React 组件的两种创建方式

2.1 使用函数创建组件

- 使用JS中的函数创建的组件叫做：函数组件
- 函数组件必须有返回值
- 组件名称必须以大写字母开头， React 据此区分 组件 和 普通的React 元素
- 使用函数名作为组件标签名

```
function Hello() {
  return (
    <div>这是我的第一个函数组件！</div>
  )
}
ReactDOM.render(<Hello />, root)
```

2. React 组件的两种创建方式

2.2 使用类创建组件

- 类组件：使用 ES6 的 class 创建的组件
- 约定1：类名称也必须以大写字母开头
- 约定2：类组件应该继承 React.Component 父类，从而可以使用父类中提供的方法或属性
- 约定3：类组件必须提供 render() 方法
- 约定4：render() 方法必须有返回值，表示该组件的结构

```
class Hello extends React.Component {
  render() {
    return <div>Hello Class Component!</div>
  }
}
ReactDOM.render(<Hello />, root)
```

2. React 组件的两种创建方式

2.3 抽离为独立 JS 文件

- 思考：项目中的组件多了之后，该如何组织这些组件呢？
- 选择一：将所有组件放在同一个JS文件中
- 选择二：将每个组件放到单独的JS文件中
- 组件作为一个独立的个体，一般都会放到一个单独的 JS 文件中

2. React 组件的两种创建方式

2.3 抽离为独立 JS 文件

1. 创建Hello.js
2. 在 Hello.js 中导入React
3. 创建组件（函数 或 类）
4. 在 Hello.js 中导出该组件
5. 在 index.js 中导入 Hello 组件
6. 渲染组件

```
// index.js
import Hello from './Hello'
// 渲染导入的Hello组件
ReactDOM.render(<Hello />, root)
```

```
// Hello.js
import React from 'react'
class Hello extends React.Component {
  render() {
    return <div>Hello Class Component!</div>
  }
}
// 导出Hello组件
export default Hello
```

目录 Contents

- React 组件介绍
- React 组件的两种创建方式
- React 事件处理
- 有状态组件和无状态组件
- 组件中的 state 和 setState()
- 事件绑定 this 指向
- 表单处理

3. React 事件处理

1. 事件绑定
2. 事件对象

3. React 事件处理

3.1 事件绑定

- React 事件绑定语法与 DOM 事件语法相似
- 语法：on+事件名称={事件处理程序}，比如：onClick={() => {}}
- 注意：React 事件采用**驼峰命名法**，比如：onMouseEnter、onFocus
- 在函数组件中绑定事件：

```
class App extends React.Component {
  handleClick() {
    console.log('单击事件触发了')
  }
  render() {
    return (
      <button onClick={this.handleClick}></button>
    )
  }
}
```

```
function App() {
  function handleClick() {
    console.log('单击事件触发了')
  }

  return (
    <button onClick={handleClick}>点我</button>
  )
}
```

3. React 事件处理

3.2 事件对象

- 可以通过事件处理程序的参数获取到事件对象
- React 中的事件对象叫做：合成事件（对象）
- 合成事件：兼容所有浏览器，无需担心跨浏览器兼容性问题

```
function handleClick(e) {
  e.preventDefault()
  console.log('事件对象', e)
}
<a onClick={handleClick}>点我，不会跳转页面</a>
```

目录 Contents

- React 组件介绍
- React 组件的两种创建方式
- React 事件处理
- 有状态组件和无状态组件
- 组件中的 state 和 setState()
- 事件绑定 this 指向
- 表单处理

4. 有状态组件和无状态组件

- 函数组件又叫做无状态组件，类组件又叫做有状态组件
- 状态（state）即数据
- 函数组件没有自己的状态，只负责数据展示（静）
- 类组件有自己的状态，负责更新 UI，让页面“动” 起来

比如计数器案例中，点击按钮让数值加 1 。0 和 1 就是不同时刻的状态，而由 0 变为 1 就表示状态发生了变化。状态变化后，UI 也要相应的更新。React 中想要实现该功能，就要使用有状态组件来完成。

目录 Contents

- React 组件介绍
- React 组件的两种创建方式
- React 事件处理
- 有状态组件和无状态组件
- 组件中的 state 和 setState()
- 事件绑定 this 指向
- 表单处理

5. 组件中的 state 和 setState

1. state的基本使用
2. setState()修改状态

5. 组件中的 state 和 setState

5.1 state的基本使用

- 状态（state）即数据，是组件内部的私有数据，只能在组件内部使用
- state 的值是对象，表示一个组件中可以有多个数据

```
class Hello extends React.Component {
  constructor() {
    super()
    // 初始化state
    this.state = {
      count: 0
    }
  }
  render() {
    return (
      <div>有状态组件</div>
    )
  }
}
```

```
class Hello extends React.Component {
  // 简化语法
  state= {
    count: 0
  }
  render() {
    return (
      <div>有状态组件</div>
    )
  }
}
```

5. 组件中的 state 和 setState

5.1 state的基本使用

- 获取状态：this.state

```
class Hello extends React.Component {
  // 简化语法
  state= {
    count: 0
  }
  render() {
    return (
      <div>有状态组件，{this.state.count}</div>
    )
  }
}
```

5. 组件中的 state 和 setState

5.1 state的基本使用

- 状态即数据
- 状态是私有的，只能在组件内部使用
- 通过 this.state 来获取状态

```
class Hello extends React.Component {
  // 简化语法
  state= {
    count: 0
  }
  render() {
    return (
      <div>有状态组件，{this.state.count}</div>
    )
  }
}
```

5. 组件中的 state 和 setState

5.2 setState()修改状态

- 状态是可变的
- 语法：this.setState({ 要修改的数据 })
- 注意：不要直接修改 state 中的值，这是错误的！！！
- setState() 作用：1. 修改 state 2. 更新UI
- 思想：数据驱动视图


```
// 正确
this.setState({
  count: this.state.count + 1
})
// 错误
this.state.count += 1
```

5. 组件中的 state 和 setState

从 JSX 中抽离事件处理程序

- JSX 中掺杂过多 JS 逻辑代码，会显得非常混乱
- 推荐：将逻辑抽离到单独的方法中，保证 JSX 结构清晰

- 原因：事件处理程序中 this 的值为 undefined
- 希望：this 指向组件实例（render方法中的this即为组件实例）

目录 Contents

- React 组件介绍
- React 组件的两种创建方式
- React 事件处理
- 有状态组件和无状态组件
- 组件中的 state 和 setState()
- 事件绑定 this 指向
- 表单处理

6. 事件绑定 this 指向

1. 箭头函数
2. Function.prototype.bind()
3. class 的实例方法

6. 事件绑定 this 指向

1. 箭头函数

- 利用箭头函数自身不绑定this的特点
- render() 方法中的 this 为组件实例，可以获取到 setState()

```
class Hello extends React.Component {
  onIncrement() {
    this.setState({ … })
  }
  render() {
    // 箭头函数中的this指向外部环境，此处为：render()方法
    return (
      <button onClick={() => this.onIncrement()}></button>
    )
  }
}
```

6. 事件绑定 this 指向

2. Function.prototype.bind()

- 利用ES5中的bind方法，将事件处理程序中的this与组件实例绑定到一起

```
class Hello extends React.Component {
  constructor() {
    super()
    this.onIncrement = this.onIncrement.bind(this)
  }
  // ...省略 onIncrement
  render() {
    return (
      <button onClick={this.onIncrement}></button>
    )
  }
}
```

6. 事件绑定 this 指向

3. class 的实例方法

- 利用箭头函数形式的class实例方法
- 注意：该语法是实验性语法，但是，由于babel的存在可以直接使用

```
class Hello extends React.Component {
  onIncrement = () => {
    this.setState({ … })
  }
  render() {
    return (
      <button onClick={this.onIncrement}></button>
    )
  }
}
```

6. 事件绑定 this 指向

总结

1. 推荐：使用class的实例方法
2. 箭头函数
3. bind

```
class Hello extends React.Component {
  onIncrement = () => {
    this.setState({ … })
  }
  render() {
    return (
      <button onClick={this.onIncrement}></button>
    )
  }
}
```

目录 Contents

- React 组件介绍
- React 组件的两种创建方式
- React 事件处理
- 有状态组件和无状态组件
- 组件中的 state 和 setState()
- 事件绑定 this 指向
- 表单处理

7. 表单处理

1. 受控组件
2. 非受控组件（DOM方式）

7. 表单处理

7.1 受控组件

- HTML 中的表单元素是可输入的，也就是有自己的可变状态
- 而，React 中可变状态通常保存在 state 中，并且只能通过 setState() 方法来修改

计数器：0

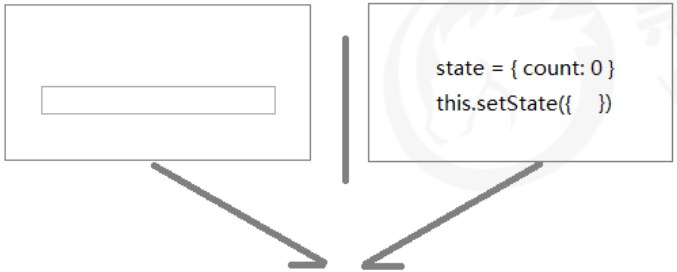
+1

```
state = {
  count: 0
}
this.setState({
  count: this.state.count + 1
})
```

7. 表单处理

7.1 受控组件

- HTML 中的表单元素是可输入的，也就是有自己的可变状态
- 而，React 中可变状态通常保存在 state 中，并且只能通过 setState() 方法来修改
- React将 state 与表单元素值value绑定到一起，由 state 的值来控制表单元素的值
- 受控组件：其值受到 React 控制的表单元素




```
<input type="text" value={this.state.txt} />
```

7. 表单处理

7.1 受控组件

步骤：

1. 在 state 中添加一个状态，作为表单元素的value值（控制表单元素值的来源）

```
state = { txt: '' }
```

```
<input type="text" value={this.state.txt} />
```

7. 表单处理

7.1 受控组件

步骤：

1. 在 state 中添加一个状态，作为表单元素的value值（控制表单元素值的来源）
2. 给表单元素绑定 change 事件，将 表单元素的值 设置为 state 的值（控制表单元素值的变化）

```
state = { txt: '' }
```

```
<input type="text" value={this.state.txt}
  onChange={e => this.setState({ txt: e.target.value })}
/>
```

7. 表单处理

7.1 受控组件

示例：

1. 文本框、富文本框、下拉框
2. 复选框

7. 表单处理

7.1 受控组件

示例总结：

1. 文本框、富文本框、下拉框 操作value属性
2. 复选框 操作checked属性

7. 表单处理

7.1 受控组件

多表单元素优化：

- 问题：每个表单元素都有一个单独的事件处理程序处理太繁琐
- 优化：使用一个事件处理程序同时处理多个表单元素

7. 表单处理

7.1 受控组件

多表单元素优化步骤：

1. 给表单元素添加name属性，名称与 state 相同
2. 根据表单元素类型获取对应值
3. 在 change 事件处理程序中通过 [name] 来修改对应的state

```
<input
  type="text"
  name="txt"
  value={this.state.txt}
  onChange={this.handleForm}
/>
```

```
// 根据表单元素类型获取值
const value = target.type === 'checkbox'
    ? target.checked
    : target.value

// 根据name设置对应state
this.setState({
  [name]: value
})
```

7. 表单处理

7.2 非受控组件

- 说明：借助于 ref，使用原生 DOM 方式来获取表单元素值
- ref 的作用：获取 DOM 或组件

7.2 非受控组件

使用步骤：

1. 调用 React.createRef() 方法创建一个 ref 对象

```
constructor() {
  super()
  this.txtRef = React.createRef()
}
```

2. 将创建好的 ref 对象添加到文本框中

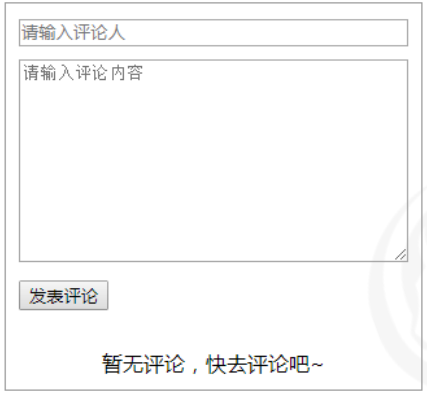
```
<input type="text" ref={this.txtRef} />
```

3. 通过 ref 对象获取到文本框的值

```
Console.log(this.txtRef.current.value)
```

React组件基础

案例：评论列表



React组件基础

案例：需求分析

① 渲染评论列表（列表渲染）
② 没有评论数据时渲染：暂无评论（条件渲染）
③ 获取评论信息，包括评论人和评论内容（受控组件）
④ 发表评论，更新评论列表（setState()）

案例：实现步骤

1. 渲染评论列表

① 在 state 中初始化评论列表数据
② 使用数组的map方法遍历state中的列表数据
③ 给每个被遍历的li元素添加key属性

案例：实现步骤

2. 渲染暂无评论

① 判断列表数据的长度是否为0

② 如果为0，则渲染暂无评论

案例：实现步骤

3. 获取评论信息

① 使用受控组件方式处理表单元素

案例：实现步骤

4. 发表评论

① 给按钮绑定单击事件
② 在事件处理程序中，通过state获取评论信息
③ 将评论信息添加到state中，并调用 setState() 方法更新state
④ 边界情况：清空文本框
⑤ 边界情况：非空判断

React 组件基础

1. 组件的两种创建方式：函数组件和类组件
2. 无状态（函数）组件，负责静态结构展示
3. 有状态（类）组件，负责更新 UI ，让页面动起来
4. 绑定事件注意 this 指向问题
5. 推荐使用受控组件来处理表单
6. 完全利用 JS 语言的能力创建组件，这是 React 的思想

传智播客旗下高端IT教育品牌