React 组件

从入门到工作 - React 全解

版权声明

联系方式

如果你想要购买本课程
请微信联系 xiedaimala02 或 xiedaimala03

如果你发现有人盗用本课程
请微信联系 xiedaimala02 或 xiedaimala03

目录

- 类组件和函数组件
- 如何使用 props 和 state
- 如何绑定事件
- 复习 this + 两个面试题

组件

Component

Element V.S. Component

- 元素与组件
  - `const div = React.createElement('div',...)`
  - 这是一个 React 元素（d小写）
  - `const Div = ()=>React.createElement('div'..)`
  - 这是一个 React 组件（D大写）
- 什么是组件
  - 能跟其他物件组合起来的物件，就是组件
  - 组件并没有明确的定义，靠感觉理解就行
  - 就目前而言，一个返回 React 元素的函数就是组件
  - 在 Vue 里，一个构造选项就可以表示一个组件

React 两种组件

- 一、函数组件

```
function Welcome(props){
  return <h1>Hello, {props.name}</h1>;
}
```

使用方法： `<Welcome name="frank"/>`

- 二、类组件

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>
  }
}
```

使用方法： `<Welcome name="frank"/>`

<Welcome />

- 会被翻译成什么
  - 会被翻译为 React.createElement('div')
  - <Welcome /> 翻译为 React.createElement(Welcome)
  - 可以用 babel online 直接翻译给你看
- React.createElement 的逻辑
  - 如果传入一个字符串 'div'，则会创建一个 div
  - 如果传入一个函数，则会调用该函数，获取其返回值
  - 如果传入一个类，则在类前面加个 new（这会导致执行 constructor），获取一个组件对象，然后调用对象的 render 方法，获取其返回值

小试牛刀

- 代码示例
  - ✓ 链接
- 添加 props（外部数据）
  - ✓ 链接
  - ✓ 类组件直接读取属性 this.props.xxx
  - ✓ 函数组件直接读取参数 props.xxx
- 添加 state（内部数据）
  - ✓ 链接
  - ✓ 类组件用 this.state 读，this.setState 写
  - ✓ 函数组件用 useState 返回数组，第一项读，第二项写

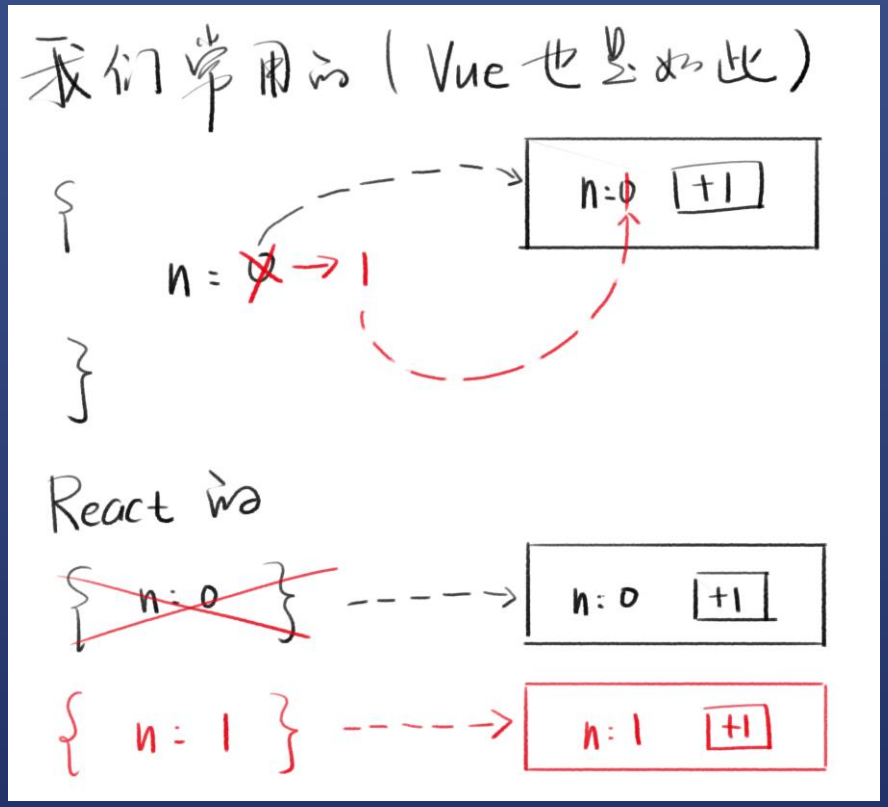
类组件注意事项

- this.state.n += 1 无效?
  - ✓ 其实 n 已经改变了，只不过 UI 不会自动更新而已
  - ✓ 调用 setState 才会触发 UI 更新（异步更新）
  - ✓ 因为 React 没有像 Vue 监听 data 一样监听 state
- setState 会异步更新 UI
  - ✓ setState 之后，state 不会马上改变，立马读 state 会失败
  - ✓ 更推荐的方式是 setState(函数)
- this.setState(this.state) 不推荐?
  - ✓ React 希望我们不要修改旧 state（不可变数据）
  - ✓ 常用代码：setState({n: state.n+1})
- 总结
  - ✓ 这是一种理念（函数式），喜欢就用，不喜欢就换 Vue

函数组件注意事项

- 跟类组件类似的地方
  - ✓ 也要通过 setX(新值) 来更新 UI
- 跟类组件不同的地方
  - ✓ 没有 this，一律用参数和变量

两种编程模型



Vue 的编程模型

一个对象，对应一个虚拟 DOM
当对象的属性改变时，
把属性相关的 DOM 节点全部更新

注：Vue 为了其他考量，也引入了
虚拟 DOM 和 DOM diff，此图只是简化

React 的编程模型

一个对象，对应一个虚拟 DOM
另一个对象，对应另一个虚拟 DOM

对比两个虚拟 DOM，找不同(DOM diff)
最后局部更新 DOM

复杂 state

- 如果 state 里不止有 n 怎么办
  - ✓ 类组件里有 n 和 m
  - ✓ 函数组件里有 n 和 m
  - ✓ 函数组件另一种不推荐的写法，你会发现 m 被置空
- 总结
  - ✓ 类组件的 setState 会自动合并第一层属性
  - ✓ 但是并不会合并第二层属性，那怎么办？
  - ✓ 使用 Object.assign 或者 ... 操作符 吧
  - ✓ 函数组件的 setX 则完全不会帮你合并
  - ✓ 要合并你自己用 ... 操作符合并去吧 :)

事件绑定

onClick、onKeyPress ...

事件绑定

- 类组件的事件绑定

```
<button onClick={() => this.addN()}>n+1</button>
```

- 传一个函数给 onClick 即可，注意 C 大写
- 思考一个问题，下面这样写行不行

```
<button onClick={this.addN}>n+1</button>
```

- 有问题，这样会使得 this.addN 里的 this 变成 window
- 这里建议复习一下我的this教程
- 还有一种写法：

```
<button onClick={this.addN.bind(this)}>n+1</button>
```

- 这样写是可以的，因为它返回一个绑定了当前 this 的新函数
- 但是这样写也太麻烦了吧，还不如第一种
- 但第一种写法依然太长，可用 this._addN = ()=> this.addN()
- 给箭头函数取个名字，然后写成

```
<button onClick={this._addN}>n+1</button>
```

事件绑定

- 类组件的事件绑定
  - 给箭头函数取个名字，然后写成

```
<button onClick={this._addN}>n+1</button>
```

  - 但这样写又要多想一个名字，不优雅
  - 不如直接写成

```
constructor(){
  this.addN = ()=> this.setState({n: this.state.n + 1})
}
render(){
  return <button onClick={this.addN}>n+1</button>
}
```

  - 但这样写不如直接声明 addN 结构清晰
  - 最终，我们得到了一个最好的写法，见下页

事件绑定

- 类组件的事件绑定最终写法

```
class Son extends React.Component{
  addN = () => this.setState({n: this.state.n + 1});
  render(){
    return <button onClick={this.addN}>n+1</button>
  }
}
```

- ✓ 这里的 addN 等价于上页 PPT 中 constructor 中的 addN
- ✓ 那么问题来了

```
class Son extends React.Component{
  addN = () => this.setState({n: this.state.n + 1});
  addN(){
    this.setState({n: this.state.n + 1})
  }
} // 两个 addN 有什么区别?
```

事件绑定

- 蓝色写法和红色写法有什么区别

```
class Son extends React.Component{
  addN = () => this.setState({n: this.state.n + 1});
  constructor(){
    this.addN = ()=> this.setState({n: this.state.n + 1})
  } // 两个蓝色写法完全等价

  addN(){ // 两个红色写法完全等价
    this.setState({n: this.state.n + 1})
  }
  addN: functioin(){
    this.setState({n: this.state.n + 1})
  }
}
// 一、两个颜色的 addN 有什么区别？蓝色在对象上，红色在原型上
// 二、为什么红色 addN 的 this 可变成 window，蓝色 addN 不会
```

回答

- 蓝色和红色的区别
  - ✓ 蓝色函数是对象本身的属性，这意味着每个 Son 组件都有自己的 addN，如果有两个 Son，就有两个 addN
  - ✓ 红色函数是对象的共用属性（也就是原型上的属性），这以为着所有 Son 组件共用一个 addN
  - ✓ 可以用代码证明

回答

- 为什么 this 会变/不会变
  - ✓ 所有函数的 this 都是参数，由调用决定，所以可变
  - ✓ 唯独箭头函数的 this 不变，因为箭头函数不接受 this
  - ✓ React 的特点：能不做的，我都不做。
  - ✓ Vue 的特点是：能帮你做的，都帮你做了

结论

- 你说了这么多 this 我没懂
  - ✓ 那就直接记住结论，用蓝色 addN 的写法即可
  - ✓ 或者你可以使用函数组件，它完全不用 this
  - ✓ 以后的趋势也是优先采用函数组件
- 我懂了，但是觉得 this 很烧脑
  - ✓ 如果你觉得烧脑，那就是没懂
- 总结
  - ✓ 要么你写博客把 this 搞透
  - ✓ 要么你就不要用 this，面试通过背题搞定

复习 this

JS 三座大山：this、原型链和 AJAX

React 又一次强迫你把 JS 学好

如果你的 this 已经学好了，这一节视频不用看

我关于 this 的三篇文章

- this 的值到底是什么？一次说清楚
  - https://zhuanlan.zhihu.com/p/23804247
  - 给 this 定性
- 你怎么还没搞懂 this?
  - https://zhuanlan.zhihu.com/p/25991271
  - 如何确定 this
- JS 里为什么会有 this
  - https://zhuanlan.zhihu.com/p/30164164
  - 为什么要设计 this

this 面试题

```
var a = {
    name:"里面的name",
    sayName:function(){
        console.log("this.name = " + this.name);
    }
};
var name = "外面的name";
function sayName(){
    var sss = a.sayName;
    sss(); //this.name = ?
    a.sayName(); //this.name = ?
    (a.sayName)(); //this.name = ?
    (b = a.sayName)();//this.name = ?
}
sayName();
// 忠告: 你要是不用我的转换代码改写程序，你就挂了
```

改写代码

```
var a = {
    name:"里面的name",
    sayName:function(){
        console.log("this.name = " + this.name);
    }
};
var name = "外面的name";
function sayName(){
    var sss = a.sayName;
    sss.call(undefined); //this 是 undefined
    a.sayName.call(a); //this 是 a
    (a.sayName).call(a); //this 是 a
    b.call(undefined);//this 是 undefined
}
sayName();
```

再说一次

不用我的转换代码改写程序，this 面试题你就挂定了

this 面试题 2

- 检验一下自己搞清楚 this 没

```
var length = 10;
function fn() {
    console.log(this.length);
}

var obj = {
  length: 5,
  method: function(fn) {
    fn();
    arguments[0]();
  }
};

obj.method(fn, 1);//输出是什么?
// 忠告: 你要是不用我的转换代码改写程序，你就挂了
```

this 面试题 2

- 改写之后

```
var length = 10;
function fn() {
    console.log(this.length);
}

var obj = {
  length: 5,
  method: function(fn) {
    fn.call(undefined);
    // this 是 undefined
    arguments[0].call(arguments);
    // this 是 arguments，也就是 [fn,1]
  }
};

obj.method.call(obj, fn, 1);//输出是什么?
```

再说一次

拿到题目，先动手，不要动脑

总结

- 关于 React 的知识点
  - ✓ 两种方式引入 React & ReactDOM
  - ✓ React.createElement('div' | 函数 | 类)
  - ✓ 类组件、函数组件如何获取外部数据 props
  - ✓ 类组件、函数组件如何使用内部数据 state
  - ✓ 类组件如何绑定事件，this 很麻烦，用蓝色写法搞定
  - ✓ 函数组件绑定事件非常简单，不用 this
  - ✓ React 的特点：能不帮你做，就不能你做
- 其他知识点
  - ✓ Vue 的特点：能帮你做，就帮你做
  - ✓ this 的复习

React V.S. Vue

- 相同点

✓ 都是对视图的封装，React 是用类和函数表示一个组件，而 Vue 是通过构造选项构造一个组件

✓ 都提供了 createElement 的 XML 简写，React 提供的是 JSX 语法，而 Vue 是提供的是模板写法（语法巨多）

- 不同点

✓ React 是把 HTML 放在 JS 里写(HTML in JS)，而 Vue 是把 JS 放在 HTML 里写(JS in HTML)