React 起手式

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

- 如何引入 React
- 函数的本质——延迟
- JSX 的用法
- 条件判断与循环

如何引入 React

两种方式

从 CDN 引入 React

- 这可有点麻烦
  - 不像 Vue，直接进入 cdn 链接即可
  - React 的引入相比之下麻烦了许多
- CDN 引入（注意顺序）
  - 引入 React：https://.../react.x.min.js
  - 引入 ReactDOM：https://.../react-dom.x.min.js
- cjs 和 umd 的区别
  - cjs 全称 CommonJS，是 Node.js 支持的模块规范
  - umd 是统一模块定义，兼容各种模块规范（含浏览器）
  - 理论上优先使用 umd，同时支持 Node.js 和浏览器
  - 最新的模块规范是使用 import 和 export 关键字

通过 webpack 引入 React

- import ... from ...
  - ✓ yarn add react react-dom
  - ✓ import React from 'react'
  - ✓ import ReactDOM from 'react-dom'
  - ✓ 注意大小写，尽量保持一致
  - ✓ 不用自己试，因为等会用 create-react-app 代替
- 其他
  - ✓ 除 webpack 外，rollup、parcel 也支持上面写法

如何引入 React

新手用 create-react-app，老手用 webpack/rollup

用 React 实现 +1

- 相当难
  - 失败的示例
  - 想知道失败原因，全靠你的 JS 基础知识
  - 失败是因为 App = React.create... 只执行了一次
  - 如何让它重新执行，以获取 n 的最新值?
  - 记不记得 6 个 6，函数可以获取最新值
- 6 个 6

```
let i
for(i=0; i<6; i++){
  setTimeout(()=>{console.log(i)},1000)
}
```

  - 一秒钟后打印 6 个 6，因为函数执行时遇到外部变量会去读取其最新值
  - 正好上面的失败可以通过把 App 变为函数解决

题外话

React 促使我们思考函数的本质

对比普通代码与函数

- 这是普通代码
  - let b = 1 + a
- 这是函数（不讨论参数）
  - let f = () => 1 + a
  - let b = f()
- 看出来什么
  - 普通代码立即求值，读取 a 的当前值
  - 函数会等调用时再求值，即延迟求值。且求值时才会读取 a 的最新值

对比React元素和函数组件

- 对比
  - App1 = React.createElement('div', null, n)
  - App1 是一个 React 元素
  - App2 = ()=> React.createElement('div', null, n)
  - App2 是一个 React 函数组件
- 启示
  - 函数 App2 是延迟执行的代码，会在被调用的时候执行
  - 注意我说的是代码执行的时机，没有在说同步和异步
  - 同步和异步关注的是得到结果的时机，忘了可以自己复习

目前我们知道

- React 元素
  - ✓ createElement 的返回值 element 可以代表一个 div
  - ✓ 但 element 并不是真正的 div（DOM 对象）
  - ✓ 所以我们一般称 element 为虚拟 DOM 对象
- () => React 元素
  - ✓ 返回 element 的函数，也可以代表一个 div
  - ✓ 这个函数可以多次执行，每次得到最新的虚拟 div
  - ✓ React 会对比两个虚拟 div，找出不同，局部更新视图
  - ✓ 不信的话看我怎么证明（调试技巧）
  - ✓ 找不同的算法叫做 DOM Diff 算法

可是你不觉得代码很丑吗

```
let n = 0;
const App = () =>
  React.createElement("div", null, [
    n,
    React.createElement(
      "button",
      {
        onClick: () => {
          n += 1;
          console.log(n); //这一句是精髓
          ReactDOM.render(App(), document.querySelector("#app"));
        }
      },
      "+1"
    )
  ]);

ReactDOM.render(App(), document.querySelector("#app"));
```

比 Vue 的写法复杂多了
别急，有办法

JSX

X 表示扩展，所以 JSX 就是 JS 扩展版

回忆一下

- Vue 有 vue-loader
  - .vue 文件里写 <template> <script> <style>
  - 通过 vue-loader 变成一个构造选项
- React 有 JSX
  - 把 <button onClick="add">+1</button> 变成 React.createElement('button', {onClick:...}, '+1') 不就行了
  - 这其中用到了编译原理（我的阶段三课程教）
  - 那么，是不是用 jsx-loader 就可以做到这些
  - 猜对一半，实际上 jsx-loader 被 babel-loader 取代了
  - 而 babel-loader 被 webpack 内置了
  - vue-loader 为什么没有被 webpack 内置……
  - 看，Vue 多难呀，以一人之力对抗多个团队

使用 JSX

- 方法一：CDN
  - 引入 babel.min.js
  - 把 <script> 改成 <script type="text/babel">
  - babel 会自动进行转译(可以理解为翻译)，此处画图
  - 这种方式似乎并不支持 src，示例
- 忠告
  - 永远不要再生产环境使用方法一，因为效率太低
  - 它要下载一个 babel.min.js
  - 它还要在浏览器端把 JSX 翻译成 JS
  - 为什么不在 build 的时候做呢？看方法二、三

使用 JSX

- 方法二：webpack + babel-loader
  - ✓ 新手跳过，因为有方法三
- 方法三：使用 create-react-app
  - ✓ 跟 @vue/cli 用法类似
  - ✓ 全局安装 yarn global add create-react-app
  - ✓ 初始化目录 create-react-app react-demo-1
  - ✓ 进入目录 cd react-demo-1
  - ✓ 为了方便学习，src 目录只留下两个文件
  - ✓ 为了方便学习，把 public/index.html 中多余的删掉
  - ✓ 开始开发 yarn start
  - ✓ 看看 App.js，是不是默认就使用了 jsx 语法?
  - ✓ 那是因为 webpack 让 JS 默认走 babel-loader

使用 JSX 的注意事项

- 注意 className
  - n

被转译为

  - React.createElement('div', {className:'red'}, "n")

- 插入变量
  - 标签里面的所有 JS 代码都要用 {} 抱起来
  - 如果你需要变量n，那么就用 {} 把 n 包起来
  - 如果你需要对象，那么就要用 {} 把对象包起来，如 {{name:'frank'}}

- 习惯 return 后面加 ()

现在 Vue 和 React 打平了

都可以写 HTML

Vue 写在 .vue 文件的 <template> 里

React 把 HTML 混在 JS / JSX 文件里

你喜欢哪个?

if...else...

条件控制语句

条件判断

- 在 Vue 里

```
<template>
  <div>
    <div v-if="n%2===0">n是偶数</div>
    <span v-else>n是奇数</span>
  </div>
</template>
```

JSX 的条件判断

- 在 React 里

```
const Component = () => {
  return n%2===0 ? <div>n是偶数</div> :
<span>n是奇数</span>
}
// 如果需要外面的 div，可以写成
const Component = () => {
  return (
    <div>
      { n%2===0 ? <div>n是偶数</div> : <span>n
是奇数</span> }
    </div>
  )
}
```

JSX 的条件判断

- 还可以写成

```
const Component = () => {
  const content = (
    <div>
      { n%2===0 ? <div>n是偶数</div> : <span>n
是奇数</span> }
    </div>
  )
  return content
}
```

JSX 的条件判断

- 还可以写成

```
const Component = () => {
  const inner = n%2===0 ? <div>n是偶数</div> :
<span>n是奇数</span>
  const content = (
    <div>
      { inner }
    </div>
  )
  return content
}
```

JSX 的条件判断

- 还可以写成

```
const Component = () => {
  let inner
  if (n%2===0) {
    inner = <div>n是偶数</div>
  } else {
    inner = <span>n是奇数</span>
  }
  const content = (
    <div>
      { inner }
    </div>
  )
  return content
}
```

结论

- 在 Vue 里
  - ✓ 只能用 Vue 提供的语法写条件判断
- 在 React 里
  - ✓ 你想™怎么写就™怎么写，你就是在写 JS 而已

循环语句

loop

循环语句

- 在 Vue 里可以遍历数组和对象

```
<template>
  <div>
    <div v-for="(n, index) in numbers"
         :key="index">
      下标 {{index}}，值为 {{n}}
    </div>
  </div>
</template>
```

在 React 里

```
const Component = (props) => {
  return props.numbers.map((n,index)=>{
    return <div>下标{index}值为{n}</div>
  })
}
// 如果你需要外面的 div，可以写成
const Component = (props) => {
  return (<div>
    { props.numbers.map((n,index)=>{
      return <div>下标{index}值为{n}</div>
    }) }
  </div>)
}
```

在 React 里还可以写

```
const Component = (props) => {
  const array = []
  for(let i=0;i<props.numbers.length;i++){
    array.push(<div>下标{i}值为
{props.numbers[i]}</div>)
  }
  return <div>{ array }</div>
}
```

结论

- 在 Vue 里
  - ✓ 只能用 Vue 提供的语法写循环
- 在 React 里
  - ✓ 你想™怎么写就™怎么写，你就是在写 JS 而已

Vue 和 React 的区别突显

你是愿意用 Vue 给你封装好的 v-*

还是愿意在 React 里写原生 JS

明显偷懒的新人跟适合 Vue

小结：目前学了什么

- 引入 React & ReactDOM
  - CDN 方式：react.js、react-dom.js、babel.js
  - 也可以直接 import React from 'react'
- React.createElement
  - 创建虚拟 DOM 对象
  - 函数的作用：多次创建虚拟 DOM 对象
  - DOM Diff 是什么
- JSX
  - 将 XML 转译为 React.createElement
  - 使用 { } 插入 JS 代码
  - create-react-app 默认将 JS 当作 JSX 处理
  - 条件判断、循环要用原生 JS 实现

该学 Vue 还是学 React

- 怂
  - ✓ 喜欢哪个就学哪个
  - ✓ 我总不能强迫你学你不喜欢的那个
  - ✓ 现实生活中你没有喜欢的妹子，我总不能把你喜欢的框架也剥夺掉
- 全都要
  - ✓ 学完一个，请立马开始学另一个

再见

下节课继续讲 React