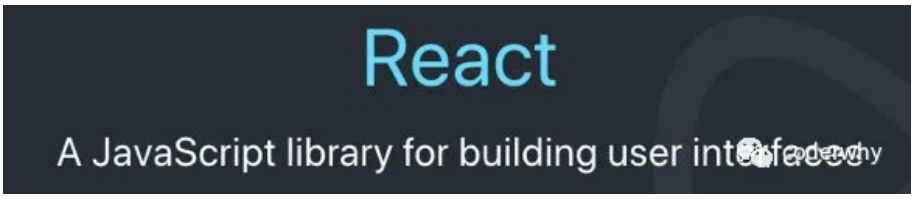
而React本身绝对是优雅和美丽的，所以从这里开始，我们和React来一场美丽的邂逅吧！

一. 认识React

1.1. React是什么?

React是什么呢？相信每个做前端的人对它都或多或少有一些印象。

这里我们来看一下官方对它的解释：用于构建用户界面的 JavaScript 库。



React是什么?

我们知道对于前端来说，主要的任务就是构建用于界面，而构建用于界面离不开三个技术：

- HTML：构建页面的结构
- CSS：构建页面的样式
- JavaScript：页面动态内容和交互

那么使用最原生的HTML、CSS、JavaScript可以构建完整的用户界面吗？当然可以，但是会存在很多问题

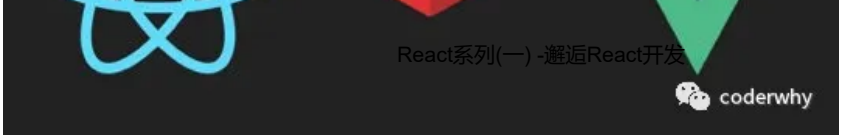
- 比如操作DOM兼容性的问题；
- 比如过多兼容性代码的冗余问题；
- 比如代码组织和规范的问题；

所以，一直以来前端开发人员都在需求可以让自己开发更方便的JavaScript库：

- 在过去的很长时间内，jQuery是被使用最多的JavaScript库；
- 在过去的一份调查中显示，全球前10,000个访问最高的网站中，有65%使用了jQuery，是当时最受欢迎的JavaScript库；
- 但是越来越多的公司开始慢慢不再使用jQuery，包括程序员使用最多的GitHub；

现在前端领域最为流行的是三大框架：

- Vue



三个框架

而Angular在国内并不是特别受欢迎，尤其是Angular目前的版本对TypeScript还有要求的情况下。

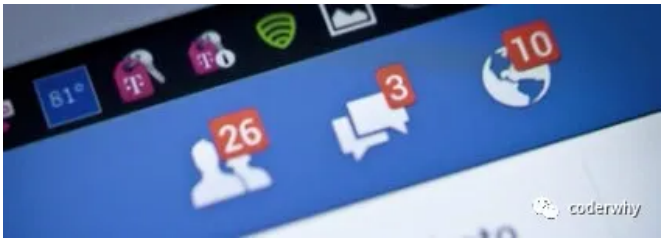
Vue和React是国内最为流行的两个框架，而他们都是帮助我们来构建用户界面的JavaScript库。

- 关于它们的对比，我会另外再写一篇文章

1.2. React的起源

React是2013年，Facebook开源的JavaScript框架，那么当时为什么Facebook要推出这样一款框架呢?

这个源于一个需求，所产生的bug:



Facebook功能需求

该功能上线之后，总是出现bug:

- 三个消息的数字在发生变化时，过多的操作很容易产生bug;

bug是否可以修复呢？当然可以修复，但是Facebook的工程师并不满足于此;

他们开始思考为什么会产生这样的问题;

- 在传统的开发模式中，我们过多的去操作界面的细节；（前端、iOS、Android）
  - 比如说需要掌握和使用大量DOM的API，当然我们可以通过jQuery来简化和适配一些API的使用;
- 另外关于数据（状态），往往会分散到各个地方，不方便管理和维护;

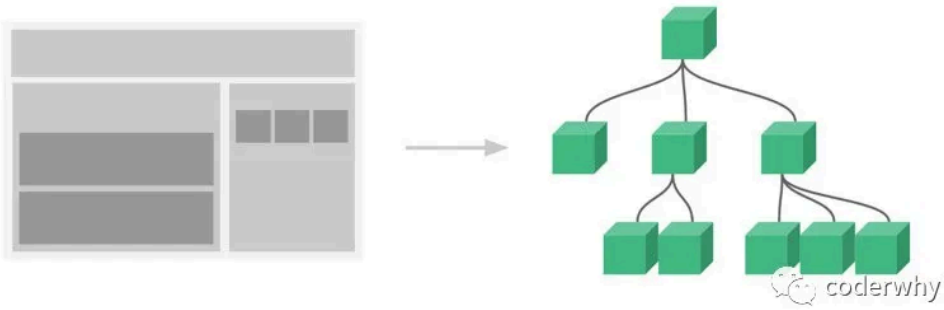
他们就去思考，是否有一种新的模式来解决上面的问题:

- 声明式编程是目前整个大前端开发的模式：Vue、React、Flutter、SwiftUI；
- 它允许我们只需要维护自己的状态，当状态改变时，React可以根据最新的状态去渲染我们的UI界面；

声明式编程

组件化开发：

- 组件化开发页面目前前端的流行趋势，我们会讲复杂的界面拆分成一个个小的组件；
- 如何合理的进行组件的划分和设计也是后面我会讲到的一个重点；



组件化开发

多平台适配：

- 2013年，React发布之初主要是开发Web页面；
- 2015年，Facebook推出了ReactNative，用于开发移动端跨平台；（虽然目前Flutter非常火爆，但是还是有很多公司在使用ReactNative）；

react多平台

1.3.2. React的优势

React由Facebook来更新和维护，它是大量优秀程序员的思想结晶：

- React的流行不仅仅局限于普通开发工程师对它的认可，大量流行的其他框架借鉴React的思想；

Vue.js框架设计之初，有很多的灵感来自Angular和React。

- 包括Vue3很多新的特性，也是借鉴和学习了React
- 比如React Hooks是开创性的新功能（也是我们课程的重点）
- Vue Function Based API学习了React Hooks的思想

Flutter的很多灵感都来自React，来自官网的一段话：（SwiftUI呢）

> Flutter widgets are built using a modern framework that takes inspiration from React. "

来自Flutter官网

- 事实上Flutter中的Widget – Element – RenderObject，对应的就是JSX – 虚拟DOM – 真实DOM

所以React可以说是前端的先驱者，它总是会引领整个前端的潮流。

1.4. React的现状

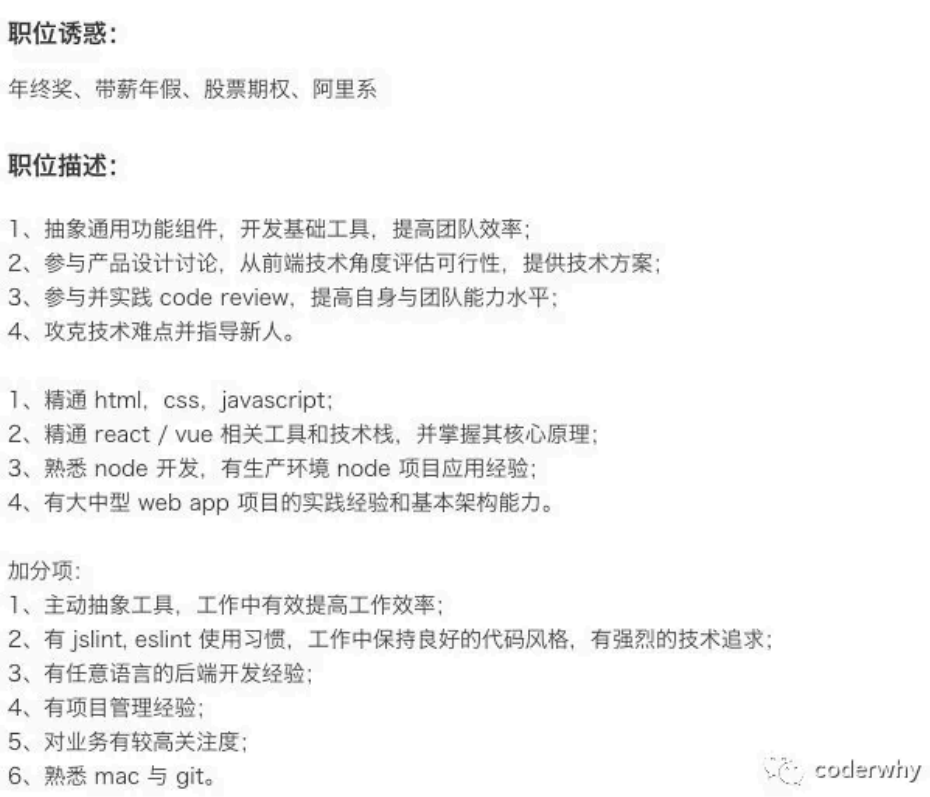
另外在HackerRank中，2020年有一份调用，你更想要学习的framework（框架）：

哪一个是你最想要学习的框架

国内外很多知名网站使用React开发：

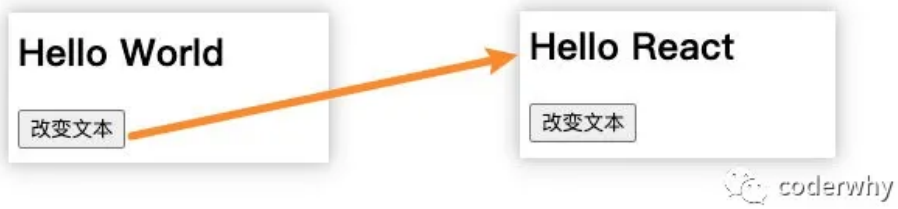
image-20200608115008557

目前国内在大型公司使用React的较多：



高级前端工程师要求

- 在界面显示一个文本：Hello World
- 点击下方的一个按钮，点击后文本改变为Hello React



案例效果

但是，我们使用React实现之前，先使用原生代码来实现，这样更加方便大家对比React和原生：

- 当然，你也可以使用jQuery和Vue来实现，对它们分别进行对比学习

原生实现代码如下：

```
<body>

  <div class="app">
    <h2 class="title">Hello World</h2>
    <button class="change-btn">改变文本</button>
  </div>

  <script>
    // 1.获取dom节点
    const titleEl = document.getElementsByClassName("title")[0];

    // 2.获取数据
    let message = "Hello World";

    // 3.将数据显示到titleEl中
    titleEl.innerHTML = message;

    // 4.改变按钮内容
    const btnEl = document.getElementsByClassName("change-btn")[0];
    btnEl.addEventListener('click', (e) => {
```

开发React必须依赖三个库：

- react：包含react所必须的核心代码
- react-dom：react渲染在不同平台所需要的核心代码
- babel：将jsx转换成React代码的工具

第一次接触React会被它繁琐的依赖搞蒙，对于Vue来说，我们只是依赖一个vue.js文件即可，但是react居然要依赖三个库。

其实呢，这三个库是各司其职的，目的就是让每一个库只单纯做自己的事情：

- 在React的0.14版本之前是没有react-dom这个概念的，所有功能都包含在react里。
- 为什么要进行拆分呢？原因就是react-native。
- react包中包含了react和react-native所共同拥有的核心代码。
- react-dom针对web和native所完成的事情不同：
  - web端：react-dom会讲jsx最终渲染成真实的DOM，显示在浏览器中
  - native端：react-dom会讲jsx最终渲染成原生的控件（比如Android中的Button，iOS中的UIButton）。

babel是什么呢？

- **Babel**，又名 **Babel.js**。
- 是目前前端使用非常广泛的编辑器、转移器。
- 比如当下很多浏览器并不支持ES6的语法，但是确实ES6的语法非常的简洁和方便，我们**开发时**希望使用它。
- 那么编写源码时我们就可以使用ES6来编写，之后通过Babel工具，将ES6转成大多数浏览器都支持的ES5的语法。

React和Babel的关系：

- 默认情况下开发React其实可以不使用babel。
- 但是前提是我们自己使用 `React.createElement` 来编写源代码，它编写的代码非常的繁琐和可读性差。
- 那么我们就可以直接编写jsx（JavaScript XML）的语法，并且让babel帮助我们转换成React.createElement。
- 后续还会讲到；

所以，我们在编写React代码时，这三个依赖都是必不可少的。

那么，如何添加这三个依赖：

- 这里有一个crossorigin的属性，这个属性的目的是为了拿到跨域脚本的错误信息

```
<script src="https://unpkg.com/react@16/umd/react.development.js" crossorigin></script>
<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js" crossorigin></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
```

2.3. Hello World

下面我们通过一个Hello World的案例来看下如何使用React开发。

需求非常简单：通过React，在界面上显示一个Hello World

- 注意：这里我们编写React的script代码中，必须添加 `type="text/babel"` ，作用是可以让babel解析jsx的语法

```
<div id="app"></div>

<script src="https://unpkg.com/react@16/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>

<script type="text/babel">
  // 通过ReactDom对象来渲染内容
  ReactDOM.render(<h2>Hello World</h2>, document.getElementById("app"));
</script>
```

代码解析：

- 依赖不需要多讲，开发React代码必须添加三个依赖；
- ReactDOM.render函数：
  - 这里我们已经提前定义一个id为app的div
  - 这里我们传入了一个h2元素，后面我们就会使用React组件
  - 参数一：传递要渲染的内容，这个内容可以是HTML元素，也可以是React的组件
  - 参数二：将渲染的内容，挂载到哪一个HTML元素上

显示效果：

```
let message = "Hello World";

// 通过ReactDom对象来渲染内容
ReactDOM.render(<h2>{message}</h2>, document.getElementById("app"));
```

2.4. Hello React

按照我们最初的案例，我们已经实现了Hello World，但是我们希望点击一个按钮后，修改为Hello React

2.4.1. 错误的方式

下面的代码是我们正常的执行逻辑，但是会报错：

- 原因是默认情况下 `ReactDOM.render` 会覆盖挂载到的app原生中的所有内容；
- 所以在执行完 `ReactDOM.render` 之后，就不存在button原生了；

```
<body>

  <div id="app">
    <button class="change-btn">改变文本</button>
  </div>

  <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
  <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>

  <script type="text/babel">
    // 将数据定义到变量中
    let message = "Hello World";

    // 通过ReactDom对象来渲染内容
    ReactDOM.render(<h2>{message}</h2>, document.getElementById("app"));

    // 获取btn
    const btnEl = document.getElementsByClassName("change-btn")[0];
    btnEl.addEventListener("click", (e) => {
      console.log(e);
    })
```

```
  <div id="app">
  </div>

  <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
  <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>

  <script type="text/babel">
    // 将数据定义到变量中
    let message = "Hello World";

    // 通过ReactDom对象来渲染内容
    render();

    // 定义一个执行的函数
    function btnClick() {
      message = "Hello React";
      render();
    }

    function render() {
      ReactDOM.render((
        <div>
          <h2>{message}</h2>
          <button onClick={btnClick}>改变文本</button>
        </div>
      ), document.getElementById("app"));
    }
  </script>
</body>
```

2.4.3. 组件的方式

整个逻辑其实可以看做一个整体，那么我们就可以将其封装成一个组件：

- 我们说过 `ReactDOM.render` 第一参数是一个HTML原生或者一个组件；
- 所以我们可以先将之前的业务逻辑封装到一个组件中，然后传入到 `ReactDOM.render` 函数中的第一个参数；

```
class App extends React.Component {
  render() {
    return (<h2>Hello World</h2>)
  }
}

ReactDOM.render(<App/>, document.getElementById("app"));
```

如果我们的Hello World是依赖变量的，并且会根据按钮的点击而改变呢？这里涉及到几个核心点

1.数据在哪里定义

- 在组件中的数据，我们可以分成两类：
  - 参与界面更新的数据：当数据变量时，需要更新组件渲染的内容
  - 不参与界面更新的数据：当数据变量时，不需要更新将组建渲染的内容
- 参与界面更新的数据我们也可以称之为是参与数据流，这个数据是定义在当前对象的state中
  - 我们可以通过在构造函数中 `this.state = {定义的数据}`
- 当我们的数据发生变化时，我们可以调用 `this.setState` 来更新数据，并且通知React进行update操作
  - 在进行update操作时，就会重新调用render函数，并且使用最新的数据，来渲染界面

2.事件绑定中的this

- 在类中直接定义一个函数，并且将这个函数绑定到html原生的onClick事件上，当前这个函数的this指向的是谁呢？
- 默认情况下是undefined
  - 很奇怪，居然是undefined；
  - 因为在正常的DOM操作中，监听点击，监听函数中的this其实是节点对象（比如说是button对象）；
  - 这次因为React并不是直接渲染成真实的DOM，我们所编写的button只是一个语法糖，它的本质React的Element对象；
  - 那么在这里发生监听的时候，react给我们的函数绑定的this，默认情况下就是一个undefined；
- 我们在绑定的函数中，可能想要使用当前对象，比如执行 `this.setState` 函数，就必须拿到当前对象的this

```
    this.state = {
      message: "Hello World"
    };
  }

  render() {
    return (
      <div>
        <h2>{this.state.message}</h2>
        <button onClick={this.changeText.bind(this)}>改变文本</button>
      </div>
    )
  }

  changeText() {
    this.setState({
      message: "Hello React"
    })
  }
}

ReactDOM.render(<App/>, document.getElementById("app"));
```

React系列教程 24

React系列教程 · 目录

下一篇 · React系列二 - 核心JSX语法一

People who liked this content also liked

GWO灰狼优化BP实现时间序列预测（带交叉验证+后续值预测输出）

安安讲代码