

邂逅React开发

itjc8收集整理

课前提醒

■ 学习这个课程需要具备什么样的基础：

□ HTML、CSS、JavaScript (ES6以上)

■ 上课时间：

□ 周一、三、五：晚上20:00 ~ 22:00

□ 最好来听直播，如果没有来听直播，找时间会看录播

□ 课程目录是暂停的，不完全按照目录来讲（可能会调整顺序、可能会插入其他知识点）

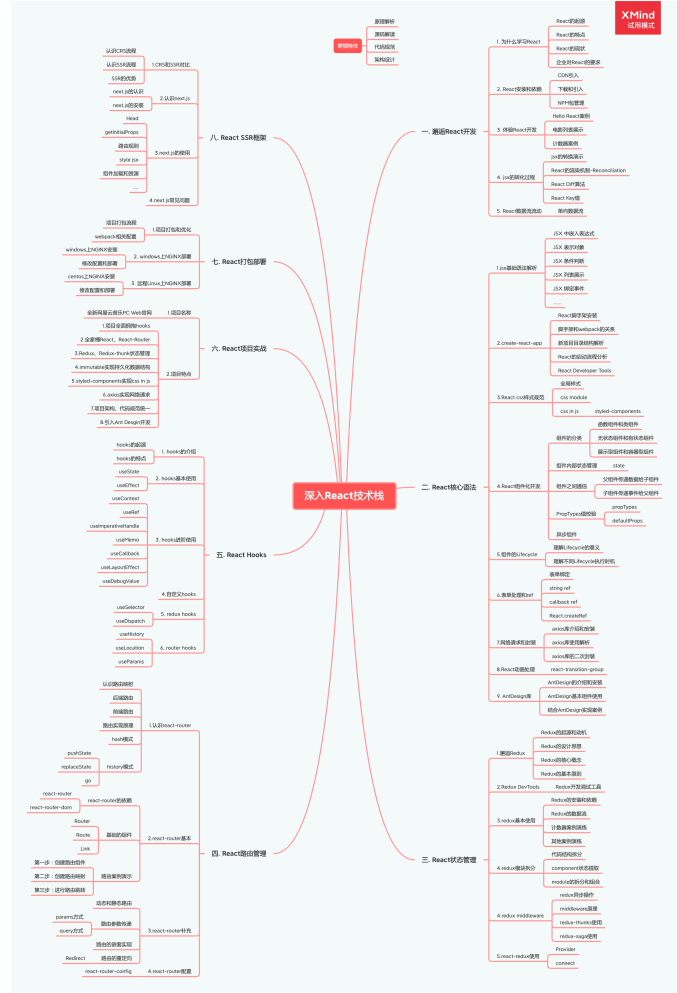
■ 相互尊重，共同进步：

□ 每个人的基础不同，之前学习的经历和方向不同；

□ 我所讲的内容是为大部分同学考虑的，所以每个人相互尊重，共同进步；

课程内容

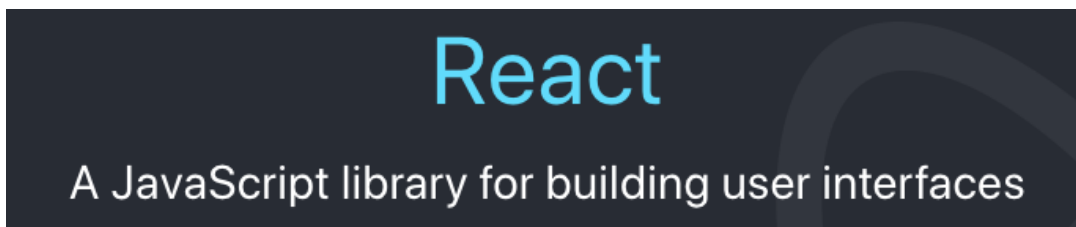
■ 全面讲解React技术栈：



React是什么？

■ React是什么呢？相信每个做开发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。



■ 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；

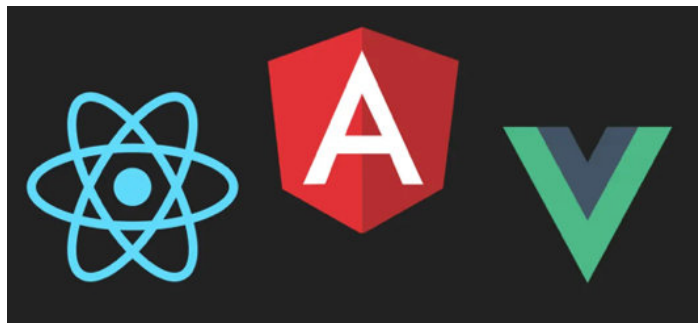
常见的JavaScript库

■ 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
- React
- Angular

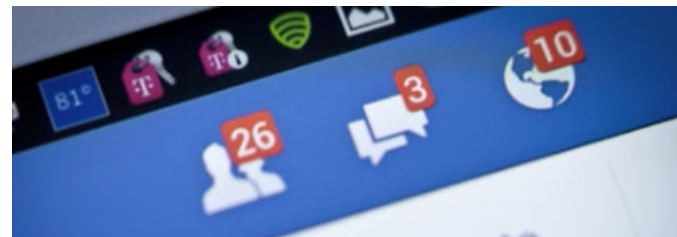


■ 而Angular在国内并不是特别受欢迎，尤其是Angular目前的版本对TypeScript还有要求的情况下。

- 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

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：
 - 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）
 - ✓ 并且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：
 - 1.以组件的方式去划分一个个功能模块
 - 2.组件内以jsx来描述UI的样子，以state来存储组件内的状态
 - 3.当应用的状态发生改变时，通过setState来修改状态，状态发生变化时，UI会自动发生更新



React的特点 – 声明式编程

■ 声明式编程：

- 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；

$$\text{UI} = f(\text{state})$$

The layout
on the screen

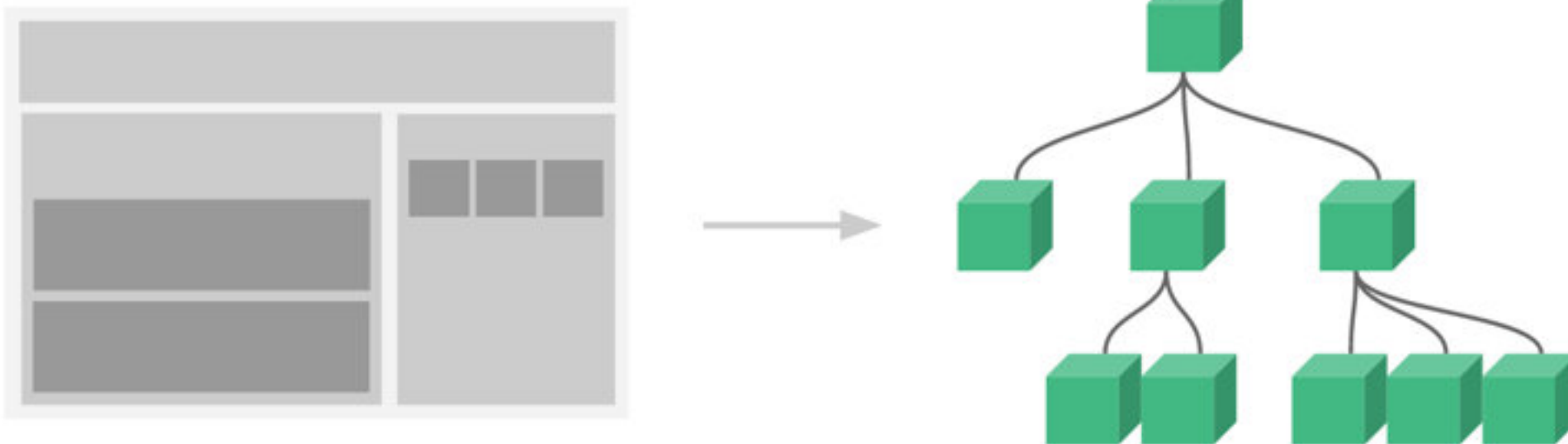
Your
build
methods

The application state

React特点 - 组件化开发

■ 组件化开发：

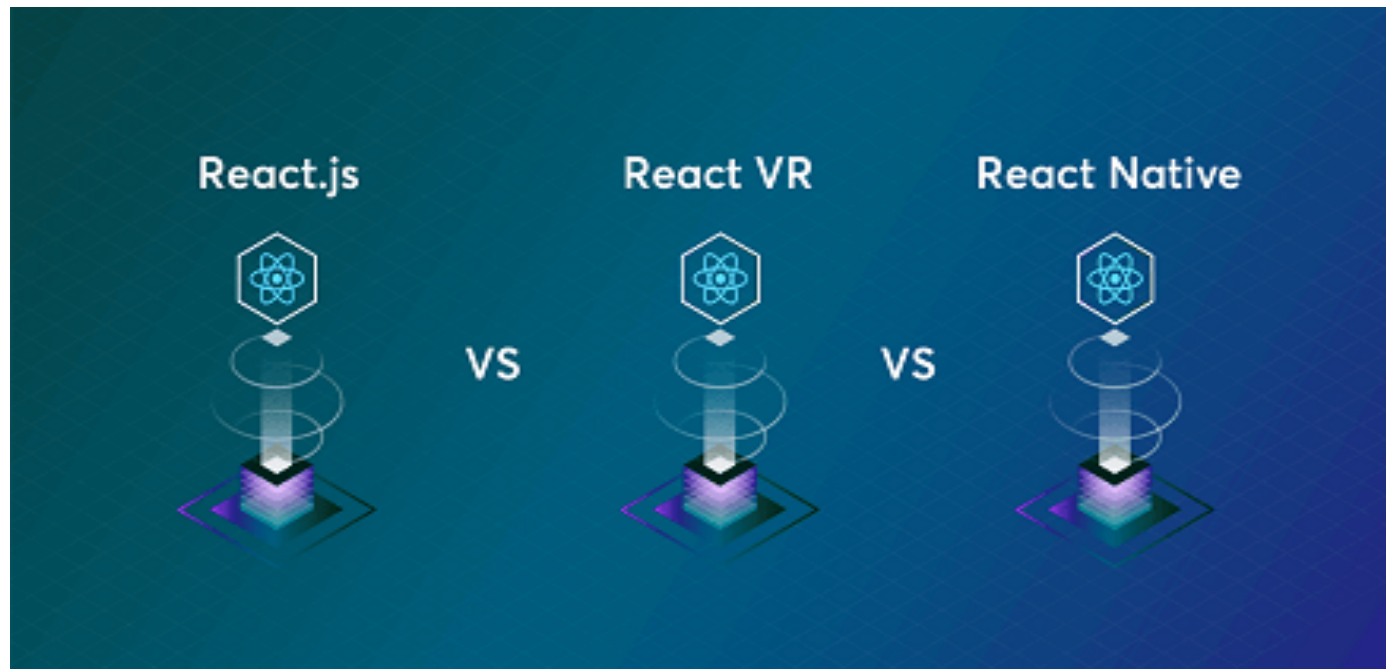
- 组件化开发页面目前前端的流行趋势，我们会讲复杂的界面拆分成一个个小的组件；
- 如何合理的进行组件的划分和设计也是后面我会讲到的一个重点；



React的特点 – 多平台适配

■ 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）；
- 2017年，Facebook推出ReactVR，用于开发虚拟现实Web应用程序；（随着5G的普及，VR也会是一个火爆的应用场景）；



掌握最先进的思想和技术

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

Flutter widgets are built using a modern framework that takes inspiration from [React](#).
- 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。

HackerRank调查显示

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

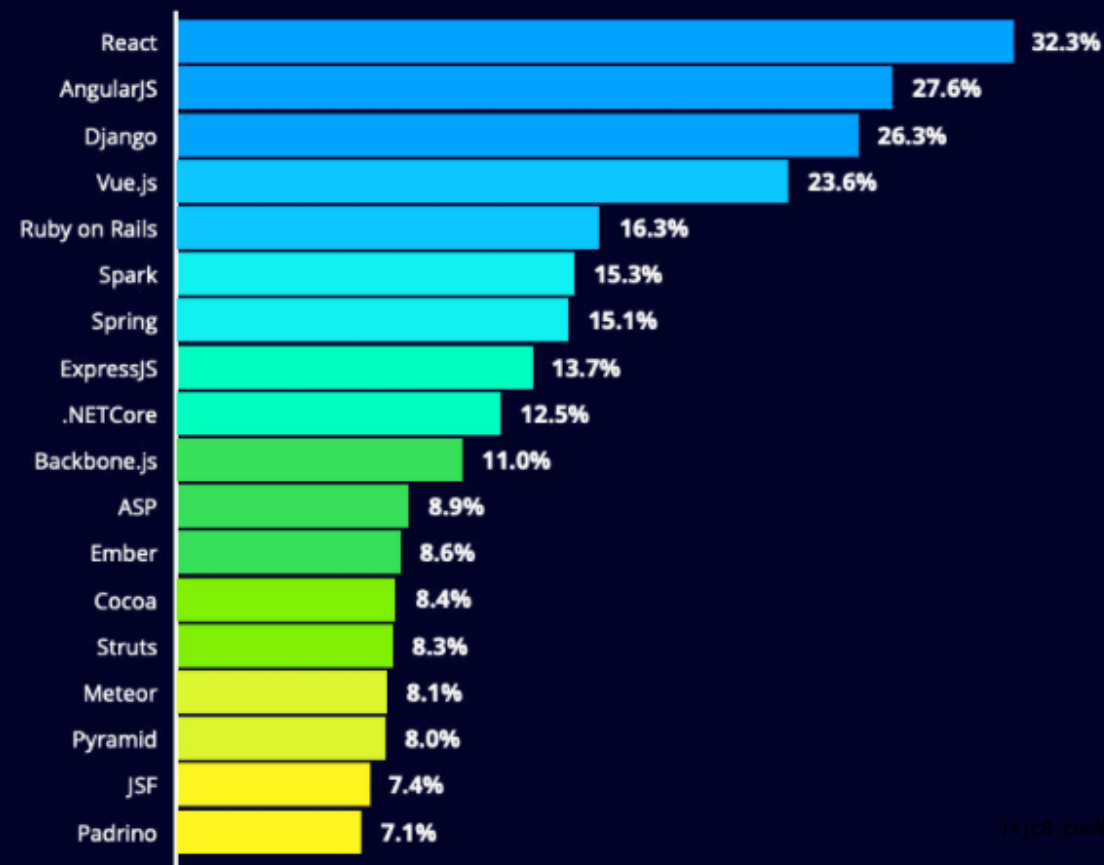
Developers are learning React, AngularJS, and Django

React is the framework most developers want to learn—32% say it's the framework they're learning next.

AngularJS and Django round out the top 3: 28% say they plan to learn AngularJS, and 26% say they're learning Django. All three frameworks are in the top five that hiring managers need.

Last year, we noted that [an increasing number of developers](#) were planning to learn React—and that as a result, it had a chance to become the best known framework in 2020. But in this year's survey, React's position remained static as the second best-known framework.

Which frameworks do you plan on learning next?



React在国内外被广泛使用



职位诱惑：

年终奖、带薪年假、股票期权、阿里系

职位描述：

- 1、抽象通用功能组件，开发基础工具，提高团队效率；
- 2、参与产品设计讨论，从前端技术角度评估可行性，提供技术方案；
- 3、参与并实践 code review，提高自身与团队能力水平；
- 4、攻克技术难点并指导新人。

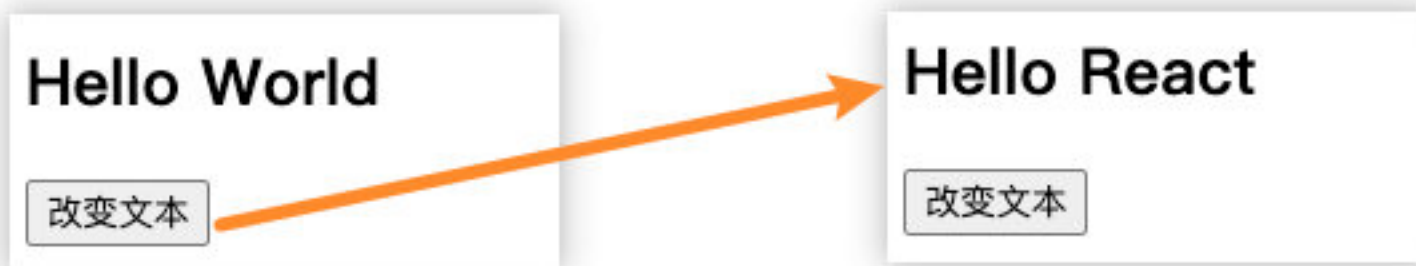
- 1、精通 html, css, javascript;
- 2、精通 react / vue 相关工具和技术栈，并掌握其核心原理；
- 3、熟悉 node 开发，有生产环境 node 项目应用经验；
- 4、有大中型 web app 项目的实践经验和基本架构能力。

加分项：

- 1、主动抽象工具，工作中有效提高工作效率；
- 2、有 tslint, eslint 使用习惯，工作中保持良好的代码风格，有强烈的技术追求；
- 3、有任意语言的后端开发经验；
- 4、有项目管理经验；
- 5、对业务有较高关注度；
- 6、熟悉 mac 与 git。

Hello React案例

- 为了演练React，我们可以提出一个小的需求：
 - 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



- 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

React开发依赖

■ 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）。

认识Babel

■ 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；

引入React依赖

- 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：
 - 方式一：直接CDN引入
 - 方式二：下载后，添加本地依赖
 - 方式三：通过npm管理（后续脚手架再使用）
- 暂时我们直接通过CDN引入，来演练下面的示例程序：
 - 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>
```


Hello World

■ 第一步：在界面上通过React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

```
<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"));
</script>
```

■ ReactDOM.render函数：

□ 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

✓ 这里我们传入了一个h2元素，后面我们就会使用React组件

□ 参数二：将渲染的内容，挂载到哪一个HTML元素上

✓ 这里我们已经提定义一个id为app的div

■ 我们可以通过{}语法来引入外部的变量或者表达式

Hello React – 错误做法

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

```
<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);
  })
</script>
```

Hello React – 正确做法

```
<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>
```

Hello React – 组件化开发

- 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；
- 在React中，如何封装一个组件呢？
- 这里我们暂时使用类的方式封装组件：
 - 1.定义一个类（类名大写，组件的名称是必须大写的，小写会被认为是HTML元素），继承自React.Component
 - 2.实现当前组件的render函数
 - ✓ render当中返回的jsx内容，就是之后React会帮助我们渲染的内容

```
class App extends React.Component {  
  render() {  
    return (<h2>Hello World</h2>)  
  }  
}  
  
ReactDOM.render(<App/>, document.getElementById("app"));
```

组件化 - 数据依赖

- 组件化问题一：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

```
class App extends React.Component {  
  constructor() {  
    super();  
    this.state = {  
      message: "Hello World"  
    };  
  }  
}
```

组件化 – 事件绑定

- 组件化问题二：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
 - 我们就需要在传入函数时，给这个函数直接绑定this
 - 类似于下面的写法： <button onClick={this.changeText.bind(this)}>改变文本</button>

```
changeText() {  
  console.log(this);  
}
```

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

电影列表展示

电影列表

- 星际穿越
- 大话西游
- 盗梦空间
- 少年派

```
class App extends React.Component {
  constructor(props) {
    super(props);

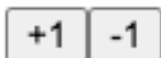
    this.state = {
      movies: ["星际穿越", "大话西游", "盗梦空间", "少年派"]
    }
  }

  render() {
    return (
      <div>
        <h2>电影列表</h2>
        <ul>
          {
            this.state.movies.map((item, index) => {
              return <li>{item}</li>
            })
          }
        </ul>
      </div>
    )
  }
}

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

计数器案例

当前计数:0



```
class App extends React.Component {
  constructor(props) {
    super(props);

    this.state = {
      counter: 0
    }
  }

  render() {
    return (
      <div>
        <h2>当前计数:{this.state.counter}</h2>
        <button onClick={this.increment.bind(this)}>+1</button>
        <button onClick={this.decrement.bind(this)}>-1</button>
      </div>
    )
  }

  increment() {
    this.setState({
      counter: this.state.counter+1
    })
  }

  decrement() {
    this.setState({
      counter: this.state.counter-1
    })
  }
}

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