

React-1

课程内容

React 是什么？

一个用于构建用户界面的 JavaScript 库 中文手册：<https://react.docschina.org/>

命令式编程 和 声明式编程

- 告诉计算机怎么做（How） - 过程
- 告诉计算机我们要什么（What） - 结果

如何使用 React

基于浏览器的模式

- React.js 提供 React.js 核心功能代码，如：虚拟 dom，组件
 - `React.createElement(type, props, children);`
- ReactDOM 提供了与浏览器交互的 DOM 功能，如：dom 渲染
 - `ReactDOM.render(element, container[, callback])`
 - `element`: 要渲染的内容
 - `container`: 要渲染的内容存放容器
 - `callback`: 渲染后的回调函数

babel

babel-standalone.js: 在浏览器中处理 JSX <https://cdn.bootcss.com/babel-standalone/6.26.0/babel.min.js>

JSX

JSX 是一个基于 JavaScript + XML 的一个扩展语法 - 它可以作为值使用 - 它并不是字符串 - 它也不是 HTML - 它可以配合 JavaScript 表达式一起使用

插值表达式

在 JSX 中可以使用 {表达式} 嵌入表达式

表达式：产生值的一组代码的集合

- 变量
- 算术运算
- 函数调用
-

注意：分清楚 表达式 与 语句 的区别，if、for、while 这些都是语句，JSX 不支持语句

各种类型内容在插值中的使用

- 注释 `{/注释/}` `{/* 多行注释 */}`

输出数据类型

- 字符串、数字：原样输出
- 布尔值、空、未定义 会被忽略

列表渲染

- 数组
- 对象 扩展：虚拟 DOM（virtualDOM）和 diff

条件渲染

- 三元运算符
- 与或运算符

在属性上使用 表达式

JSX 中的表达式也可以使用在属性上，但是使用的时候需要注意

- 当在属性中使用 `{}` 的时候，不要使用引号包含

JSX 使用注意事项

- 必须有,且只有一个顶层的包含元素
- JSX 不是html，很多属性在编写时不一样
 - className
 - style
- 列表渲染时，必须有 key 值
- 在 jsx 所有标签必须闭合
- 组件的首字母一定大写，标签一定要小写

XSS 为了有效的防止 XSS 注入攻击，React DOM 会在渲染的时候把内容（字符串）进行转义，所以字符串形式的标签是不会作为 HTML 标签进行处理的

基于自动化的集成环境模式 - create-react-app - 脚手架

介绍

通过前面 script 的方式虽然也能完成 React.js 的开发，但是有一个现在前端很重要的特性 - 模块化，无法使用。

Create React App 是一个使用 Node.js 编写的命令行工具，通过它可以帮助我们快速生成 React.js 项目，并内置了 Babel、Webpack 等工具帮助我们实现 ES6+ 解析、模块化解析打包，也就是通过它，我们可以使用 模块化 以及 ES6+ 等更新的一些特性。同时它还内置 ESLint 语法检测工具、Jest 单元测试工具。还有一个基于 Node.js 的 WebServer 帮助我们更好的在本地预览应用，其实还有更多。

这些都通过 Create React App 帮助我们安装并配置好了，开箱即用

安装与使用

通过 `npm`、`yarn`、`npx` 都可以

安装

`npm`

```
npm i -g create-react-app
```

`yarn`

```
yarn global add create-react-app
```

使用

安装完成以后，即可使用 `create-react-app` 命令

```
create-react-app <项目名称>
```

或者通过 `npx` 的方式

`npx`

```
npx create-react-app <项目名称>
```

项目目录结构说明

运行命令以后，就会在运行命令所在目录下面创建一个以项目名称为名的目录

```
my-app/  
  README.md  
  node_modules/  
  package.json  
  public/  
    index.html  
    favicon.ico  
  src/  
    App.css  
    App.js  
    App.test.js
```

```
index.css
index.js
logo.svg
```

命令脚本

`create-react-app` 同时也提供了其它一些命令来帮助我们进行开发

`npm start`

启动一个内置的本地 `WebServer`，根目录映射到 `./public` 目录，默认端口：3000

`npm test`

运行 `Jest` 测试

`npm run build`

打包应用（准备上线）

组件

对具有一定独立功能的数据与方法的封装，对外暴露接口，有利于代码功能的复用，且不用担心冲突问题。


函数式组件

- 函数的名称就是组件的名称
- 函数的返回值就是组件要渲染的内容

类式组件

- 组件类必须继承 `React.Component`
- 组件类必须有 `render` 方法

不从实际案例开始的知识点就是要 LM

 image-20190713113649374

样式

```
.friend-list {
  border: 1px solid #000000;
  width: 200px;
}
.friend-group dt {
  padding: 10px;
  background-color: rgb(64, 158, 255);
  font-weight: bold;
}
```

```
.friend-group dd {  
  padding: 10px;  
  display: none;  
}  
.friend-group.expanded dd {  
  display: block;  
}  
.friend-group dd.checked {  
  background: green;  
}
```

创建 FriendList 组件

```
class FriendList extends React.Component {  
  render() {  
    return (  
      <div className="friend-list">  
        <div className="friend-group">  
          <dt>家人</dt>  
          <dd>爸爸</dd>  
          <dd>妈妈</dd>  
        </div>  
        <div className="friend-group">  
          <dt>朋友</dt>  
          <dd>张三</dd>  
          <dd>李四</dd>  
          <dd>王五</dd>  
        </div>  
        <div className="friend-group">  
          <dt>客户</dt>  
          <dd>阿里</dd>  
          <dd>腾讯</dd>  
          <dd>头条</dd>  
        </div>  
      </div>  
    );  
  }  
}
```

组件复用 - 数据抽取

为了提高组件的复用性，通常会把组件中的一些可变数据提取出来

```
let datas = {  
  family: {  
    title: '家人',  
    list: [  
      {name: '爸爸'},  
      {name: '妈妈'}  
    ]  
  }  
}
```

```
    ]
  },
  friend: {
    title: '朋友',
    list: [
      {name: '张三'},
      {name: '李四'},
      {name: '王五'}
    ]
  },
  customer: {
    title: '客户',
    list: [
      {name: '阿里'},
      {name: '腾讯'},
      {name: '头条'}
    ]
  }
};
```

props 和 state

- props 父组件传递过来的参数
- state 组件自身状态
 - setState
 - 多个 setState 合并

props 与 state 的区别

state 的主要作用是用于组件保存、控制、修改自己的可变状态，在组件内部进行初始化，也可以在组件内部进行修改，但是组件外部不能修改组件的 **state** **props** 的主要作用是让使用该组件的父组件可以传入参数来配置该组件，它是外部传进来的配置参数，组件内部无法控制也无法修改 **state** 和 **props** 都可以决定组件的外观和显示状态。通常，**props** 做为不变数据或者初始化数据传递给组件，可变状态使用 **state**

组件通信与数据流

在 **React.js** 中，数据是从上自下流动（传递）的，也就是一个父组件可以把它的 **state / props** 通过 **props** 传递给它的子组件，但是子组件不能修改 **props** - **React.js** 是单向数据流，如果子组件需要修改父组件状态（数据），是通过回调函数方式来完成。

- 父级向子级通信
- 子级向父级通信

React 中的事件

- 大小写问题
- this 问题

练习

- 今天练习，把好友列表自己实现一遍

总结

下节课内容

- 组件间通信
- 受控组件和非受控组件
- 生命周期
- children
- ref
- context