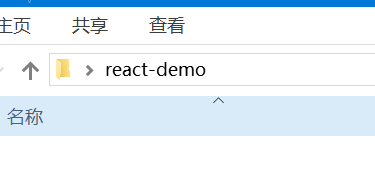
## 环境搭建

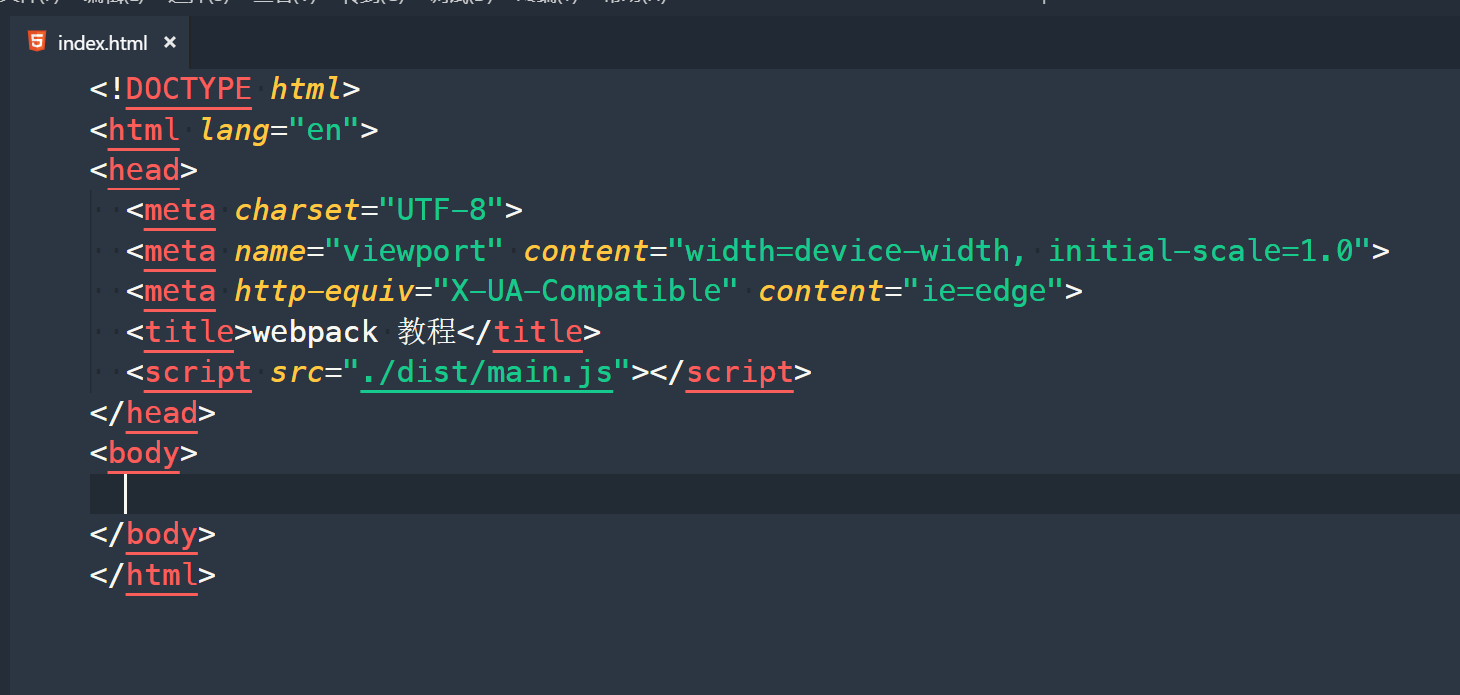
第一步、创建一个目录react-demo



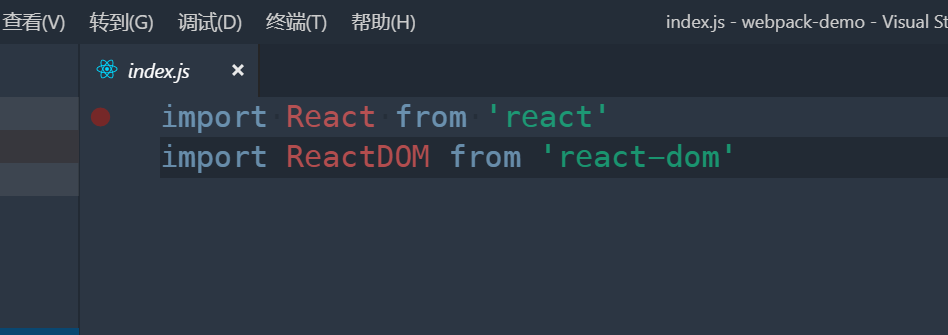
第二步、执行npm init -y



第三步、创建index.html



第四步、创建src目录，在里面创建index.js，代码如下：



第五步、安装react相关的模块

cnpm install react react-dom prop-types scheduler -S

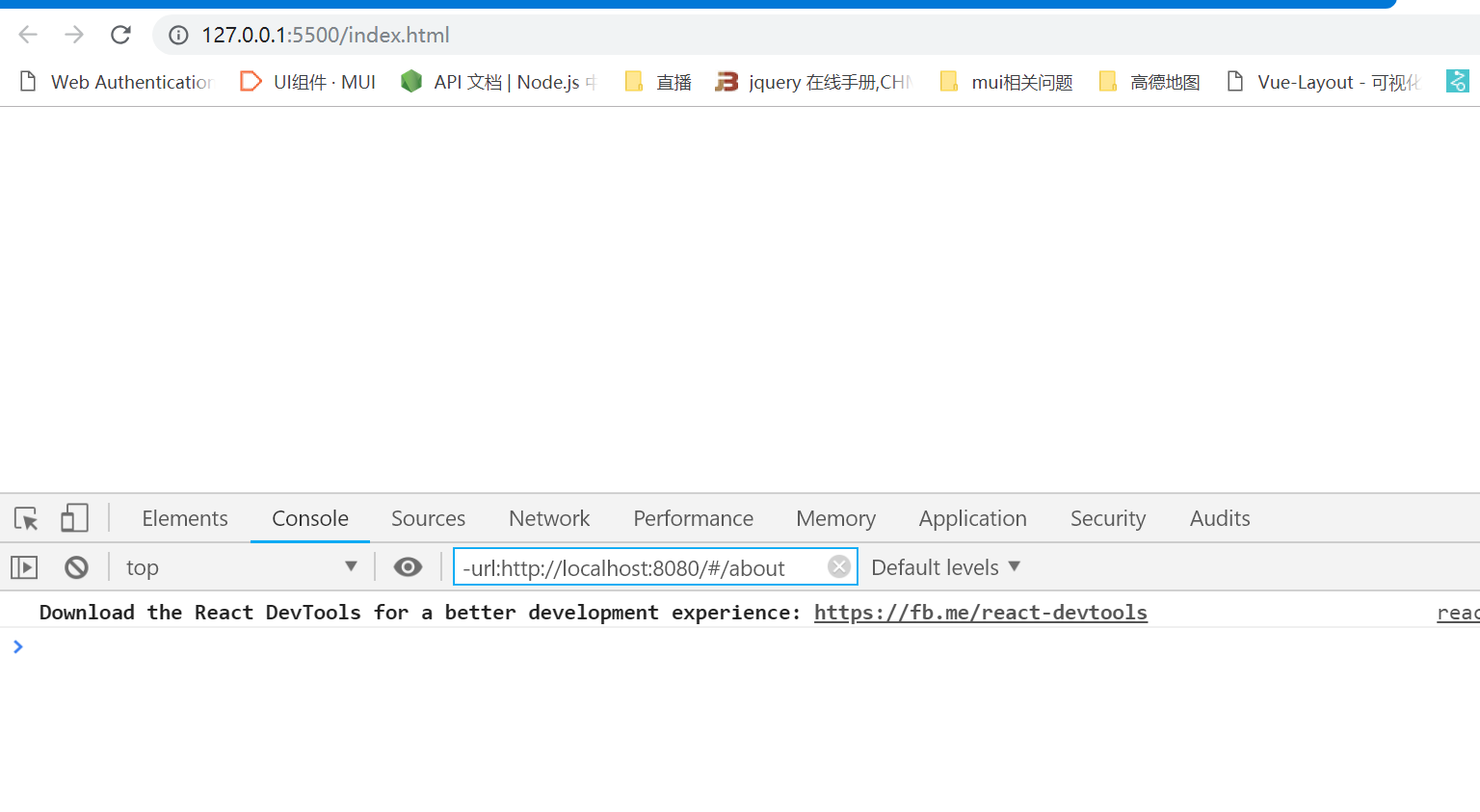
第六步、安装webpack,webpack-cli

cnpm install -D webpack webpack-cli

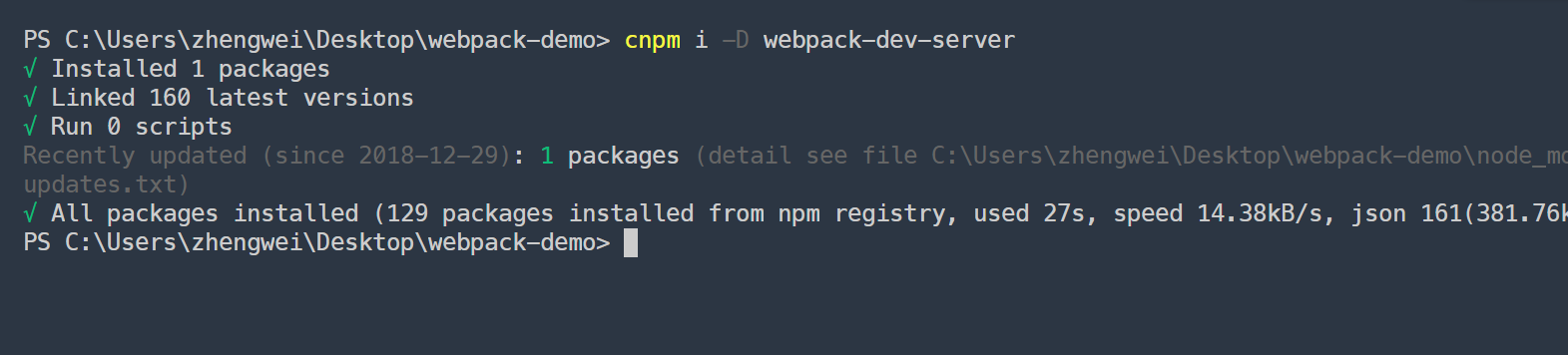
第七步、编译js文件

npx webpack --mode development

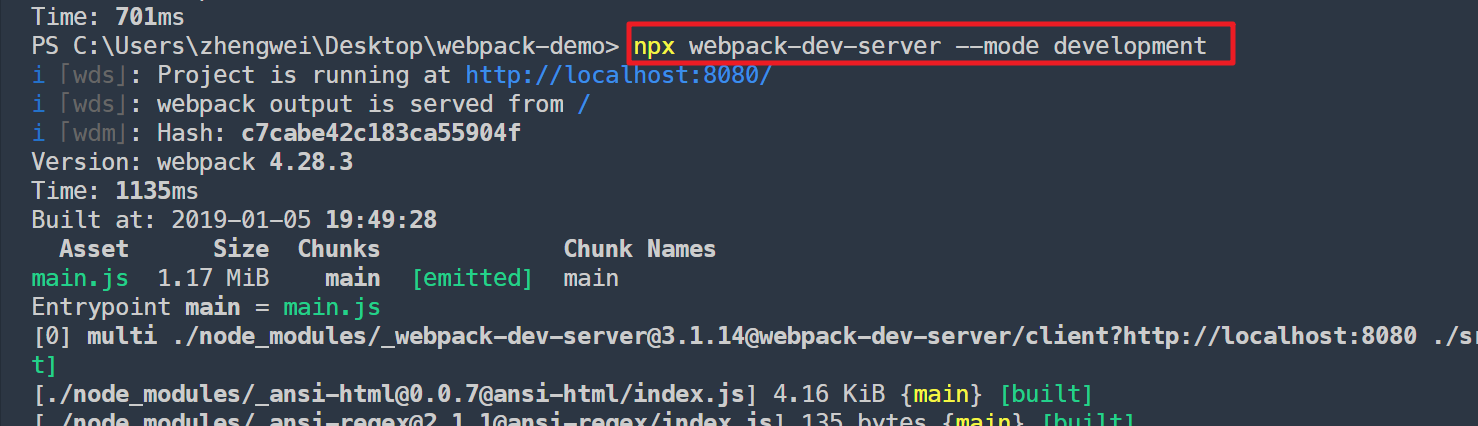
第八步、打开live-server，如果没报错，说明编译成功



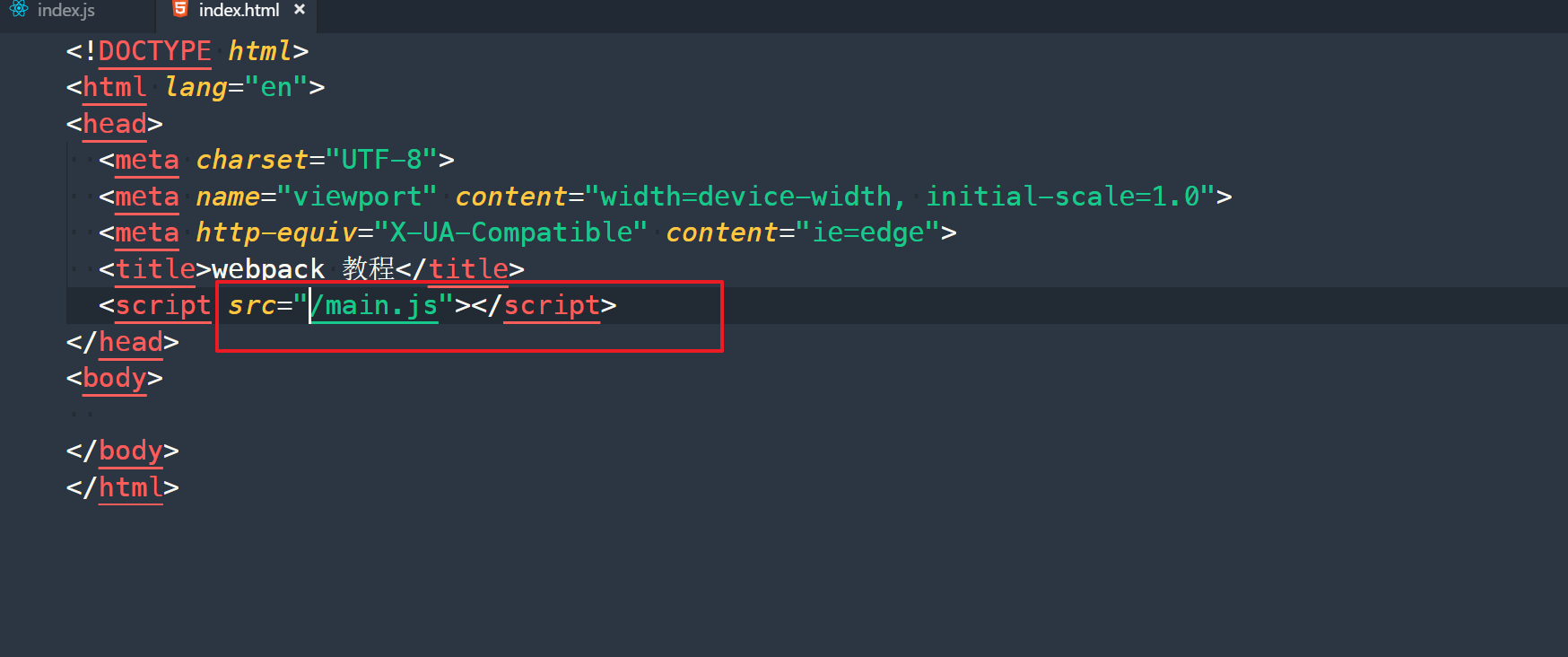
上面的操作的缺点是：只要我们src/index.js修改了，我们就必须要重新编译，所以，我们接下来可以安装webpack-dev-server



接下来，我们执行的命令换成了：



这时候，我们需要修改index.html中的路径指向：



我们现在就可以打开http://localhost:8080/，当你现在修改代码的时候，可以实现实时编译

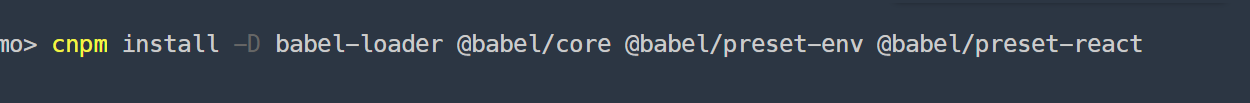
接下来，我们可以安装react的chrome插件



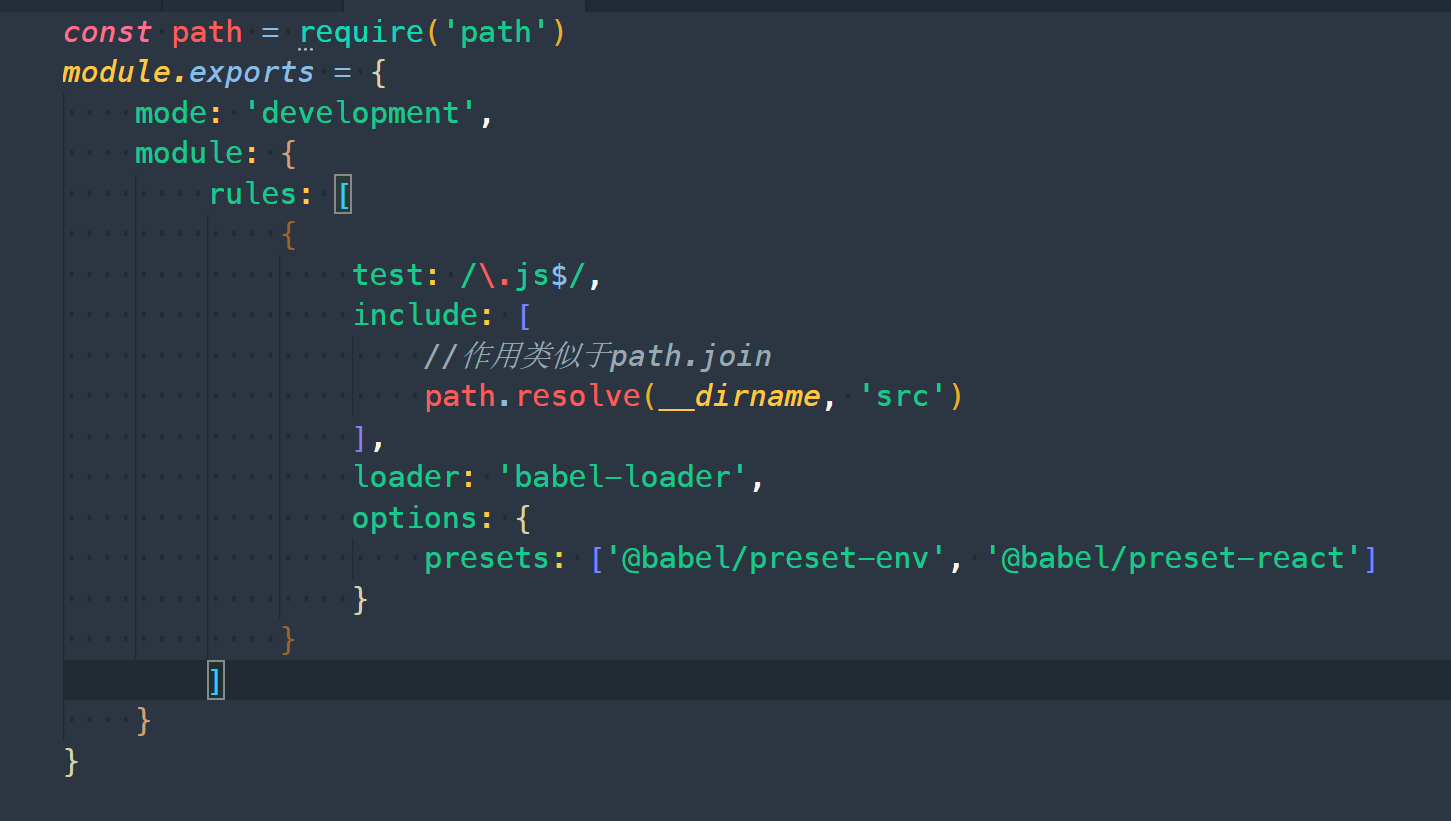
接下来，我们可以在index.js添加一行代码



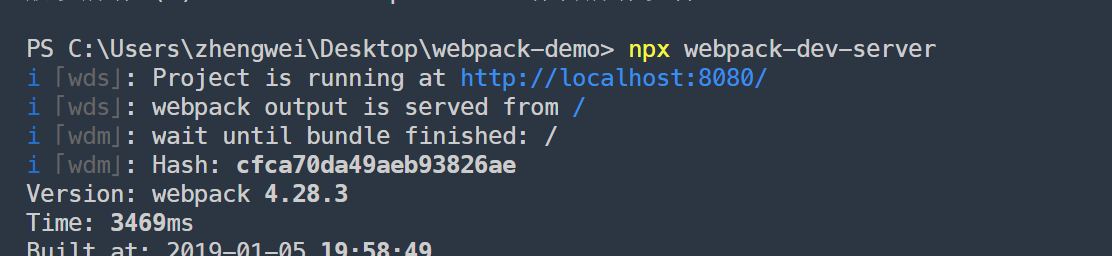
这时候浏览器会报错，因为我们现在用到了jsx语法，所以我们需要安装babel来让其支持



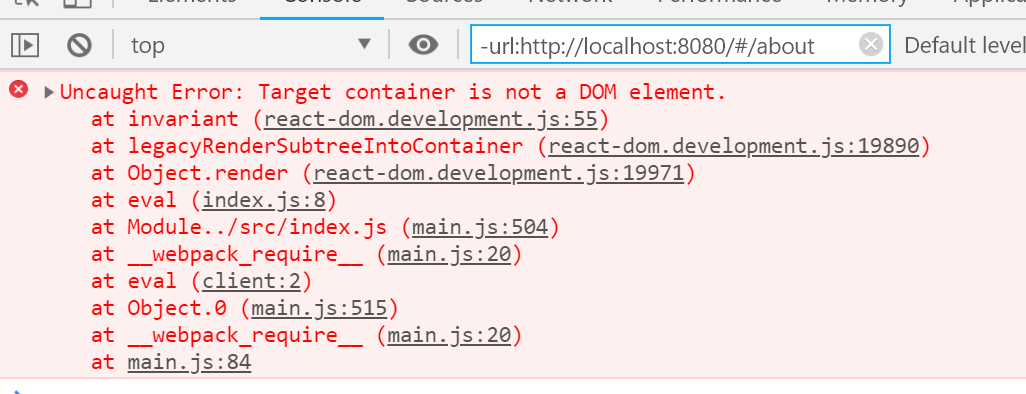
创建webpack.config.js文件，代码如下：



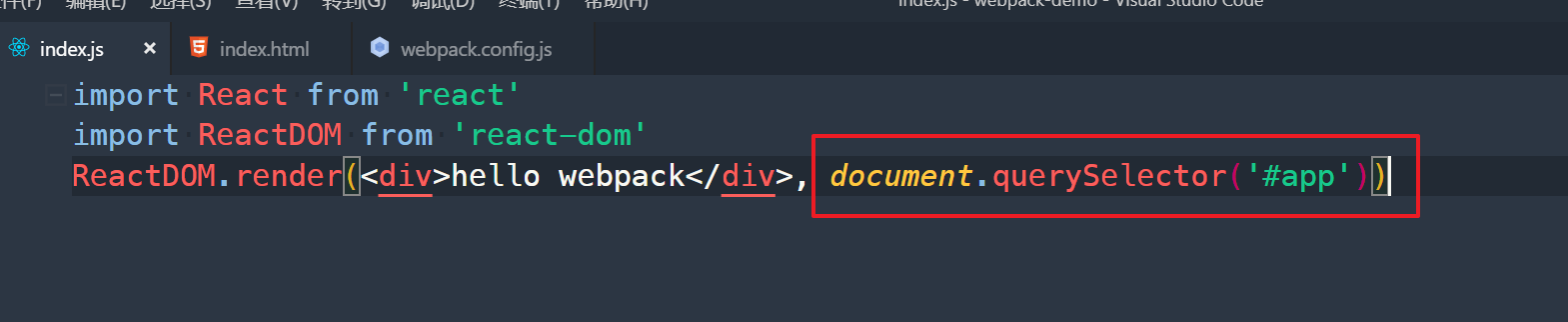
这时候，打开控制台，执行



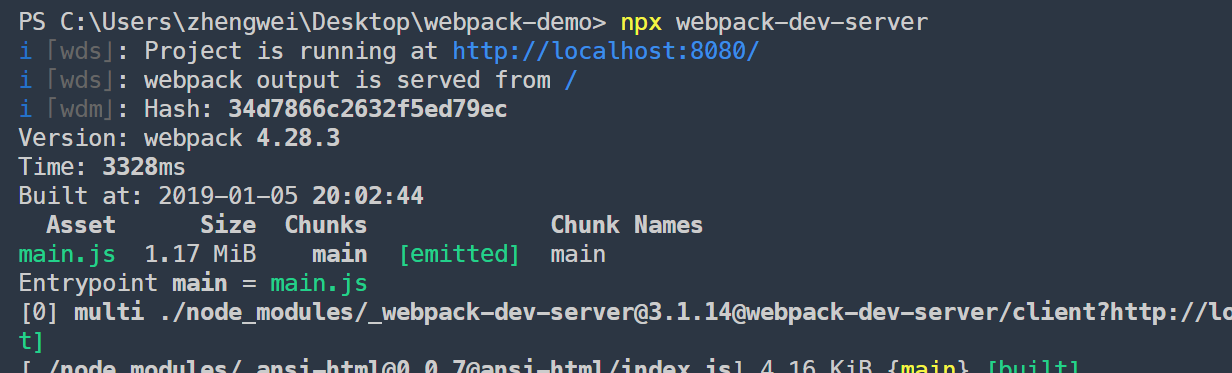
打开浏览器，会发现报错如下：



这是因为react不支持把元素挂载到body身上，调整html和js代码如下：

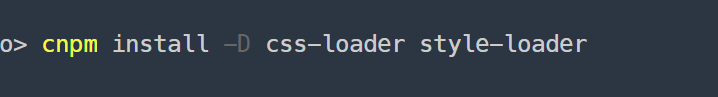


再次重启命令就没错了：

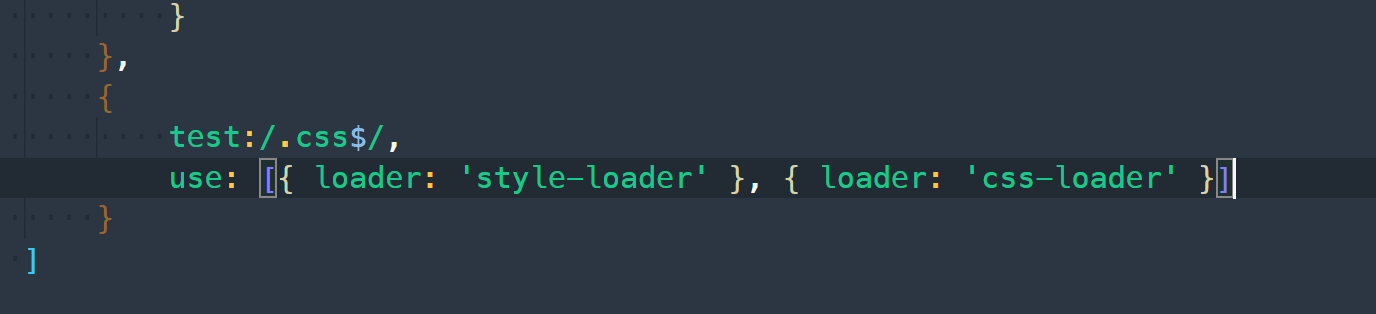


加载css

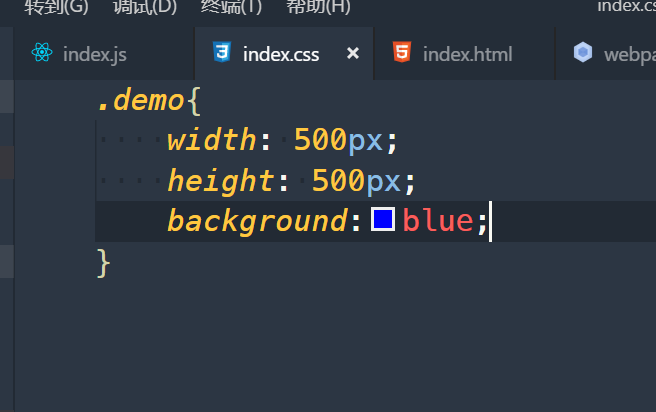
第一步、安装css加载器



第二步、修改webpack.config.js



第三步、在src中创建index.css



第四步、修改index.js,创建我们的第一个组件



## ES6相关语法学习

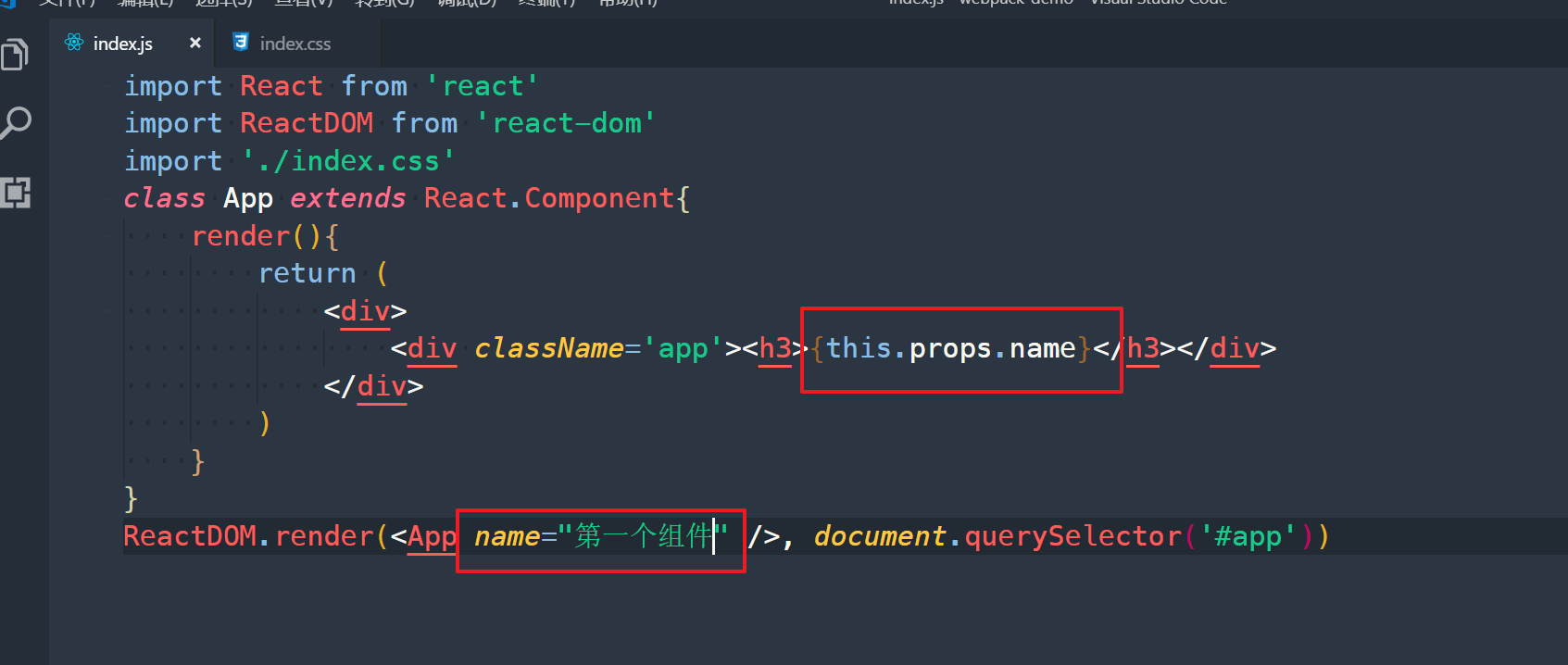
类、继承

## 相关概念学习

## jsx

在jsx的世界里面，只有两种符号要理解：一种是<>表示，接下来是html代码，一种是{}表示接下来是js代码

## 父组件向子组件传值



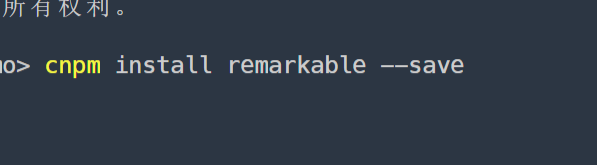
## 计时器

|  |
| --- |
| import React from 'react'  import ReactDOM from 'react-dom'  import './index.css'  class App extends React.Component{  render(){  return (  <div>  <div className='app'><h3>{this.props.name}</h3></div>  <Timer></Timer>  </div>  )  }  }  //计时器  class Timer extends React.Component{  constructor(props){  super(props);  this.state = {seconds:0};  }  tick(){  this.setState(preState=>({  seconds: preState.seconds+1  }))  }  componentDidMount(){  this.interval = setInterval(()=>this.tick(),1000);  }  render(){  return (  <div>  Seconds:{this.state.seconds}  </div>  )  }  }  ReactDOM.render(<App name="第一个组件" />, document.querySelector('#app')) |

## Todo案例

|  |
| --- |
| import React from 'react'  import ReactDOM from 'react-dom'  class TodoApp extends React.Component{  constructor(props){  super(props);  this.state = {items:[],text:''};  this.handleChange = this.handleChange.bind(this);  this.handleSubmit = this.handleSubmit.bind(this);  }  render(){  return (  <div>  <h3>TODO</h3>  <TodoList items={this.state.items} />  <form onSubmit={this.handleSubmit}>  <input  onChange={this.handleChange}  value={this.state.text}  />  <button>  Add #{this.state.items.length + 1}  </button>  </form>  </div>  )  }  handleChange(e){  this.setState({text:e.target.value});  }  handleSubmit(e){  e.preventDefault();  if(!this.state.text.length){  return;  }  const newItem = {  text:this.state.text,  id:Date.now()  };  this.setState(prevState=>({  items:prevState.items.concat(newItem),  text:''  }));  }  }  class TodoList extends React.Component{  render(){  return (  <ul>  {this.props.items.map(item=>(  <li key={item.id}>{item.text}</li>  ))}  </ul>  )  }  }  ReactDOM.render(<TodoApp />, document.querySelector('#app')); |

## 使用第三方库remarkable



|  |
| --- |
| import React from 'react'  import ReactDOM from 'react-dom'  import Remarkable from 'remarkable'  class MarkdownEditor extends React.Component {  constructor(props) {  super(props);  this.handleChange = this.handleChange.bind(this);  this.state = { value: 'Type some \*markdown\* here!' };  }  handleChange(e) {  this.setState({ value: e.target.value });  }  getRawMarkup() {  const md = new Remarkable();  return { \_\_html: md.render(this.state.value) };  }  render() {  return (  <div className="MarkdownEditor">  <h3>Input</h3>  <textarea  onChange={this.handleChange}  defaultValue={this.state.value}  />  <h3>Output</h3>  <div  className="content"  dangerouslySetInnerHTML={this.getRawMarkup()}  />  </div>  );  }  }  ReactDOM.render(<MarkdownEditor />, document.querySelector('#app')); |

接下来，大家可以开始把官方示例照着做一下：

<https://react.docschina.org/tutorial/tutorial.html>

然后看一遍官方文档：

<https://react.docschina.org/docs/hello-world.html>

然后学会使用脚手架

<https://facebook.github.io/create-react-app/>

react-router --> redux -->

接下来就可以找一个案例视频照着玩了