# 1什么是JSX？

Jsx其实就是对js的一种扩展，他使得我们可以在js文件中直接编写html代码，他通过{}进行插值和代码逻辑控制实现，避免了繁琐的字符串拼接。编程更加方便。其实它只有一个地方和js文件不一样，就是可以在里面写标签，其他和js都是一样的。

# 2.JSX基本语法

## 语法1

### 以下语法合法

|  |
| --- |
| **const** h1 = <**h1**>你好</**h1**> |

### 以下代码不合法

|  |
| --- |
|  |

### 也就是说，标签需要嵌套写，不能并排写。多于一个并且的时候需要一个父亲节点。这样子是可以的

|  |
| --- |
| **const** h1 = <**div**><**h1**>你好</**h1**><**h1**>你好</**h1**></**div**> |

### 也是可以换行的

|  |
| --- |
| **const** h1 = <**div**>  <**h1**>你好</**h1**>  <**h1**>你好</**h1**> </**div**> |

### 然后可以用插值表达式{}来渲染这个h1

|  |
| --- |
| *// import { useState } from 'react'* **import *React* from "react"**;  **function** *App*() {  *// const [count, setCount] = useState(0)* **const** h1 = <**div**>  <**h1**>你好</**h1**>  <**h2**>我们一起来学习react！！！</**h2**>  </**div**>  **return** (  <>  <**h1**> {h1}</**h1**>  <**div className="card"**>   </**div**>   </>  ) }  **export default** *App* |

### 效果如下：

|  |
| --- |
|  |

### 也可以把他们放进一对小括号里面，然后我们定义一个person对象，然后用插值的方式把person的属性嵌入hello对象在，然后再把hello嵌入到根对象中，如下

|  |
| --- |
| *// import { useState } from 'react'* **import *React* from "react"**;  **function** *App*() {  *// const [count, setCount] = useState(0)* **let** person = {  **name**:**'Jessie'**,  **gender**:**'female'**,  **age**:19,  **email**:**'Jessie12345@gmail.com'** }  **let** hello = (  <**div**>  <**h2**>个人资料</**h2**>  <**p**>姓名：{person.**name**}</**p**>  <**p**>性别：{person.**gender**}</**p**>  <**p**>年龄：{person.**age**}</**p**>  <**p**>email：{person.**email**}</**p**>  </**div**>  )  **return** (  <>  <**h1**> {hello}</**h1**>  <**div className="card"**>   </**div**>   </>  ) }  **export default** *App* |

### 运行效果如下

|  |
| --- |
|  |

## 语法2.

### 我们把App.jsx的代码注释起来（只需要把<App/>注释起来），然后在main.jsx写一些代码（实际开发中别那么做）

|  |
| --- |
| **import *React* from 'react' import** ReactDOM **from 'react-dom/client' import** *App* **from './App.jsx'** *// import './index.css'* **let** Jessie = (  <**div**>  <**h2**>个人资料</**h2**>  <**p**>姓名：Jessie</**p**>  <**p**>性别：female</**p**>  <**p**>年龄：20</**p**>  <**p**>email：Jessie123@hotmail.com}</**p**>  </**div**> )  ReactDOM.*createRoot*(***document***.getElementById(**'root'**)).render(  *// <React.StrictMode> //严格模式  // <App />  // </React.StrictMode>,  // <App />* Jessie ) |

### 效果如下

|  |
| --- |
|  |

## 语法3：

### 可以在并且里面使用循环，我们修改一下App.jsx的代码

|  |
| --- |
| *// import { useState } from 'react'* **import *React* from "react"**;   **function** *App*() {  *// const [count, setCount] = useState(0)*  **let** languages = [**'java'**,**'cpp'**,**'python'**]  **return** (  <>  <**div**>  <**ul**>  {languages.map(item=>(  <**li**>{item}</**li**>  ))}  </**ul**>  </**div**>   </>  ) }  **export default** *App* |

### 效果如下

|  |
| --- |
|  |

## 语法4.

### {}里面不能直接写if语句来判断如果需要判断，需要使用三目运算符

### 我们把App.jsx的相关引用注释掉，把main.jsx的代码改动一下

|  |
| --- |
| **import *React* from 'react' import** ReactDOM **from 'react-dom/client' import** *App* **from './App.jsx'** *// import './index.css'* **let** age =19 **let** Jessie = (  <**div**>  <**h2**>个人资料</**h2**>  <**p**>姓名：Jessie</**p**>  <**p**>性别：female</**p**>  <**p**>年龄：{age}</**p**>  <**p**>email：Jessie123@hotmail.com}</**p**>  <**p**>{age>=18?**'已经成年'**:**'尚未成年'**}</**p**>  </**div**> )  ReactDOM.*createRoot*(***document***.getElementById(**'root'**)).render(  *// <React.StrictMode> //严格模式  // <App />* Jessie ) |

### 运行，效果如下

|  |
| --- |
|  |

## 语法5：

### 函数返回值，也就是说可以在{}里面直接调用函数回去他的返回值渲染到元素中，如

|  |
| --- |
| **import *React* from 'react' import** ReactDOM **from 'react-dom/client' import** *App* **from './App.jsx'** *// import './index.css'* **let** age =19 **function** sayHello(){  **return 'Hello World of React!!!'** } **let** Jessie = (  <**div**>  <**h2**>个人资料</**h2**>  <**p**>姓名：Jessie</**p**>  <**p**>性别：female</**p**>  <**p**>年龄：{age}</**p**>  <**p**>email：Jessie123@hotmail.com}</**p**>  <**p**>{age>=18?**'已经成年'**:**'尚未成年'**}</**p**>  <**p**>{sayHello()}</**p**>  </**div**> )  ReactDOM.*createRoot*(***document***.getElementById(**'root'**)).render(  *// <React.StrictMode> //严格模式  // <App />* Jessie ) |

## 语法6：

## 标签的特殊属性：

### class:需要替换成className,

### 2.label标签的for属性: 需要替换成htmlFor=对应的控件的id值，注意是id，class没有效果

### 如，我们修改一下main.jsx的代码

|  |
| --- |
| **import *React* from 'react' import** ReactDOM **from 'react-dom/client' import** *App* **from './App.jsx'** *// import './index.css'* **let** age =19 **function** sayHello(){  **return 'Hello World of React!!!'** } **let** Jessie = (  <**div**>  <**h2**>个人资料</**h2**>  <**p**>姓名：Jessie</**p**>  <**p**>性别：female</**p**>  <**p**>年龄：{age}</**p**>  <**p**>email：Jessie123@hotmail.com}</**p**>  <**p**>{age>=18?**'已经成年'**:**'尚未成年'**}</**p**>  <**p**>{sayHello()}</**p**>  <**label htmlFor="username" className='username'**>用户名：</**label**>  <**input id='username' type="text"**/>  </**div**> )  ReactDOM.*createRoot*(***document***.getElementById(**'root'**)).render(  *// <React.StrictMode> //严格模式  // <App />* Jessie ) |

### 运行一下，效果如下

|  |
| --- |
|  |

### 当你在用户名上面点击一下，文本框会自动获得收入焦点

## 语法7：

### 列表渲染，也就是语法3的内容，我们把它的写法稍微改动一下，注意这里是在App.jsx,我们注释掉Jessie，打开<App />

|  |
| --- |
| *// import { useState } from 'react'* **import *React* from "react"**;   **function** *App*() {  *// const [count, setCount] = useState(0)* **let** person = {  **name**:**'Jessie'**,  **gender**:**'female'**,  **age**:19,  **email**:**'Jessie12345@gmail.com'** }  **let** hello = (  <**div**>  <**h2**>个人资料</**h2**>  <**p**>姓名：{person.**name**}</**p**>  <**p**>性别：{person.**gender**}</**p**>  <**p**>年龄：{person.**age**}</**p**>  <**p**>email：{person.**email**}</**p**>  </**div**>  )  **let** languages = [**'java'**,**'cpp'**,**'python'**]  **return** (  <>  {*/\*<h1> {hello}</h1>\*/*}  <**div**>  <**ul**>  {languages.map(item=>{  **return** <**li**>{item}</**li**>  })}  </**ul**>  </**div**>   </>  ) }  **export default** *App* |

### 运行代码，效果如下

|  |
| --- |
|  |

### 还可以简写，在箭头函数里面去掉return和{}

|  |
| --- |
| *// import { useState } from 'react'* **import *React* from "react"**;   **function** *App*() {  *// const [count, setCount] = useState(0)* **let** person = {  **name**:**'Jessie'**,  **gender**:**'female'**,  **age**:19,  **email**:**'Jessie12345@gmail.com'** }  **let** hello = (  <**div**>  <**h2**>个人资料</**h2**>  <**p**>姓名：{person.**name**}</**p**>  <**p**>性别：{person.**gender**}</**p**>  <**p**>年龄：{person.**age**}</**p**>  <**p**>email：{person.**email**}</**p**>  </**div**>  )  **let** languages = [**'java'**,**'cpp'**,**'python'**]  **return** (  <>  {*/\*<h1> {hello}</h1>\*/*}  <**div**>  <**ul**>  {languages.map(item=> <**li**>{item}</**li**>)}  </**ul**>  </**div**>   </>  ) }  **export default** *App* |

### 运行一下，效果是一样的

|  |
| --- |
|  |

### 注意：我们打开开发者工具，发现有一个错误说：遍历出来的子元素需要一个key，我们给它加上

|  |
| --- |
| **// import { useState } from 'react'**  **import React from "react";**  **function App() {**  **// const [count, setCount] = useState(0)**  **let person = {**  **name:'Jessie',**  **gender:'female',**  **age:19,**  **email:'Jessie12345@gmail.com'**  **}**  **let hello = (**  **<div>**  **<h2>个人资料</h2>**  **<p>姓名：{person.name}</p>**  **<p>性别：{person.gender}</p>**  **<p>年龄：{person.age}</p>**  **<p>email：{person.email}</p>**  **</div>**  **)**  **let languages = ['java','cpp','python']**  **return (**  **<>**  **{/\*<h1> {hello}</h1>\*/}**  **<div>**  **<ul>**  **{languages.map(item=> <li key={item}>{item}</li>)}**  **</ul>**  **</div>**  **</>**  **)**  **}**  **export default App** |

### 我们运行程序，发现报错没有了

|  |
| --- |
|  |

### 语法说明：

#### 1.jsx中可以直接解析并且渲染数组，所以我们通常只需要通过map方法遍历数组拼接html标签即可

#### 2.通过遍历数组生成的子元素都需要一个唯一的key属性（不能够有两个key相同）

## 语法8：

### 条件渲染，其实就是语法4，我们现在用它来控制标签的显示或者隐藏，我们打开Jessie，把<App />把代码修改一下

|  |
| --- |
| **import *React* from 'react' import** ReactDOM **from 'react-dom/client' import** *App* **from './App.jsx'** *// import './index.css'* **let** age =19 **function** sayHello(){  **return 'Hello World of React!!!'** } **let** Jessie = (  <**div**>  <**h2**>个人资料</**h2**>  <**p**>姓名：Jessie</**p**>  <**p**>性别：female</**p**>  <**p**>年龄：{age}</**p**>  <**p**>email：Jessie123@hotmail.com}</**p**>  {*/\*<p>{age>=18?'已经成年':'尚未成年'}</p>\*/*}  <**p**>{age>=18?<**span**>已经成年</**span**>:<**span**>尚未成年</**span**>}</**p**>  <**p**>{sayHello()}</**p**>  <**label htmlFor="username" className='username'**>用户名：</**label**>  <**input id='username' type="text"**/>  </**div**> )  ReactDOM.*createRoot*(***document***.getElementById(**'root'**)).render(  *// <React.StrictMode> //严格模式  // <App />* Jessie ) |

### 运行一下，发现效果如下

|  |
| --- |
|  |

### 条件渲染还有一种方法就是使用函数，注意不要给这个函数传递参数，否则得不到正确结果

#### 我们把上面的代码修改一下

|  |
| --- |
| **import *React* from 'react' import** ReactDOM **from 'react-dom/client' import** *App* **from './App.jsx'** *// import './index.css'* **let** age =19 **function** sayHello(){  **return 'Hello World of React!!!'** } **function** getAgeSate() {*//注意：这里不要传参，否则会得到错误的结果* **let** age\_span  **if**(age >=18){  **return** <**span**>是成年人</**span**>  }  **return** <**span**>尚未成年</**span**>  } **let** Jessie = (  <**div**>  <**h2**>个人资料</**h2**>  <**p**>姓名：Jessie</**p**>  <**p**>性别：female</**p**>  <**p**>年龄：{age}</**p**>  <**p**>email：Jessie123@hotmail.com</**p**>  {*/\*<p>{age>=18?'已经成年':'尚未成年'}</p>\*/*}  {*/\*<p>{age>=18?<span>已经成年</span>:<span>尚未成年</span>}</p>\*/*}  <**p**>{getAgeSate()}</**p**>  <**p**>{sayHello()}</**p**>  <**label htmlFor="username" className='username'**>用户名：</**label**>  <**input id='username' type="text"**/>  </**div**> )  ReactDOM.*createRoot*(***document***.getElementById(**'root'**)).render(  *// <React.StrictMode> //严格模式  // <App />* Jessie ) |

### 运行程序，效果如下

|  |
| --- |
|  |

#### 其实，如果代码的逻辑比较简单我们完全可以使用三目运算符比较好，只有三目运算符难以实现的复杂逻辑，才用函数