# React中组件分为2种

## 1.函数组件: 以js中函数的形式创建的组件

## 2.类组件：以ES6中类的形式创建的组件

### 其中我们学习函数组件为主，类组件也要学习一点

# 创建组件

## React中通常以一个jsx文件作为一个组件

打开我们的项目：react-app-yarn，打开<**App** />，，为了更好学习，新建一个项目：react-app-yarn2-comp1.在这个项目来编辑，注释掉Jessie在src项目项目新建一个文件夹01-component-basic，然后在里面新建2个jsx文件：FuncComponent.jsx和ClassComponent.jsx

|  |
| --- |
|  |

### 注意1： 如果要上传到gitee或者GitHub最好不要复制项目以免出现问题

### 注意2：组件文件的命名规则：

1. 首字母大写
2. 建议文件名和组件名称一致

## 创建函数组件，注意这个函数组件中的函数必须要有返回值

## 就是在jsx文件里面定义一个函数，然后导出，如我们的FuncComponent.jsx文件中写的那样

|  |
| --- |
| **function** *FuncComponent*() {   }  **export default** *FuncComponent* |

### 还可以怎么写

|  |
| --- |
| **let** *FuncComponent* =**function**() {  }  **export default** *FuncComponent* |

### 或者这么写

|  |
| --- |
| **let** *FuncComponent* =() =>{  }  **export default** *FuncComponent* |

## 很不幸，上面的项目给gitee搞丢了，我们新建一个react-project来重新创建项目，这一次先管理gitee，开发完成后再上传到gitee，

## 注意不要复制项目，最好都是新建项目，然后可以复制文件内容

## 我们的新项目结构如下

|  |
| --- |
|  |

### 我们的第一个函数组件是reactdemo3-components/src/01-compoment-basic/ClassComponent.jsx，内容如下

|  |
| --- |
| **import *React* from "react"**; *// function FuncComponent() { // // } // let FuncComponent =function() { // // }* **let** *FuncComponent* =() =>{  **let** person = {  **name**:**'Linda'**,  **gender**:**'female'**,  **age**:20,  **email**:**'Linda12345hello@gmail.com'** }  **return** (  <>  <**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**>  </>  ) }    **export default** *FuncComponent* |

### 可以使用快捷键rsc回车创建文件骨架

### 然后修改一下main.jsx的内容

|  |
| --- |
| **import *React* from 'react' import** ReactDOM **from 'react-dom/client' import** *App* **from './App.jsx'** *// import './index.css'* **import** *FuncComponent* **from "./01-compoment-basic/FuncComponent"**;  ReactDOM.*createRoot*(***document***.getElementById(**'root'**)).render(   *// <App />* <**FuncComponent**/> ) |

## 运行项目，效果如下

|  |
| --- |
|  |

### 在Vscode中有一个快捷键可以快速生成函数组件：输入rafce回车，会自动生成函数组件并且导出

### PHP storm也有快捷键：****rsc回车键****

### 我们来看看类组件，我们在ClassComponent.jsx中添加代码

|  |
| --- |
| **import *React* from "react"**; **class** ClassComponent **extends *React***.Component{  render() {  **return** (  <**div**>  <**h1**>这是类组件里面的内容</**h1**>  </**div**>  )  } }  **export default** ClassComponent; |

### 然后修改一种main.jsx的内容

|  |
| --- |
| **import *React* from 'react' import** ReactDOM **from 'react-dom/client' import** *App* **from './App.jsx'** *// import './index.css'* **import** *FuncComponent* **from "./01-compoment-basic/FuncComponent"**; **import** ClassComponent **from "./01-compoment-basic/ClassComponent"**;  ReactDOM.*createRoot*(***document***.getElementById(**'root'**)).render(   *// <App />  // <FuncComponent/>* <**ClassComponent**/> ) |

### 运行一下，效果如图

|  |
| --- |
|  |

#### 注意：使用组件的方法就是

##### 把组件引入

|  |
| --- |
|  |

##### 利用标签的形式来使用组件

|  |
| --- |
|  |

#### 注意：

#### 在Vscode中有一个快捷键生成类组件：rcc+回车

#### Phpstorm中也有快捷键：rcc+tab键

### 其实，我们可以在App.jsx中导入我们自己写的组件，然后在main.jsx中只导入App.jsx组件就可以了，我们修改一下App.jsx

|  |
| --- |
| *// import reactLogo from './assets/react.svg' // import viteLogo from '/vite.svg' // import './App.css'* **import** *FuncComponent* **from "./01-compoment-basic/FuncComponent"**; **import** ClassComponent **from "./01-compoment-basic/ClassComponent"**;  **function** *App*() {  **return** (  <>   <**h1**>React demo3</**h1**>  <**div className="card"**>  <**FuncComponent**/>  <**ClassComponent**/>  </**div**>   </>  ) }  **export default** *App* |

### 注意：在App.jsx里面可以同时加载多个组件而在main.jsx只能加载一个组件因为main.jsx其实加载的是根组件所以只能有一个

### 然后修改一下main.jsx

|  |
| --- |
| **import *React* from 'react' import** ReactDOM **from 'react-dom/client' import** *App* **from './App.jsx'** *// import './index.css'* **import** *FuncComponent* **from "./01-compoment-basic/FuncComponent"**; **import** ClassComponent **from "./01-compoment-basic/ClassComponent"**;  ReactDOM.*createRoot*(***document***.getElementById(**'root'**)).render(  <**App** />  ) |

### 运行一下，效果如下

|  |
| --- |
|  |

# 渲染组件

## 其实上面App.jsx和main.jsx中的代码就是渲染组件

|  |
| --- |
| **//App.jsx**  **import** *FuncComponent* **from "./01-compoment-basic/FuncComponent"**; **import** ClassComponent **from "./01-compoment-basic/ClassComponent"**; **function** *App*() {  **return** (  <>  <**h1**>React demo3</**h1**>  <**div className="card"**>  <**FuncComponent**/>  <**ClassComponent**/>  </**div**>  </>  ) }  **export default** *App* |
| **//main.jsx**  **import *React* from 'react' import** ReactDOM **from 'react-dom/client' import** *App* **from './App.jsx'** *// import './index.css'* **import** *FuncComponent* **from "./01-compoment-basic/FuncComponent"**; **import** ClassComponent **from "./01-compoment-basic/ClassComponent"**;  ReactDOM.*createRoot*(***document***.getElementById(**'root'**)).render(   <**App** /> ) |

# 空的根元素

## 其实很多时候我们需要根元素只是因为react的语法要求两个以上的组件必须在一个根元素里面，所以其实这个根元素可以是空：<></>，如我们把App.jsx里面的div标签删除，如下

|  |
| --- |
| **import** { *useState* } **from 'react'** **import** *FuncComponent* **from "./01-compoment-basic/FuncComponent"**; **import** ClassComponent **from "./01-compoment-basic/ClassComponent"**; **function** *App*() {  **return** (  <>  <**h1**>React demo3</**h1**>  <**FuncComponent**/>  <**ClassComponent**/>    </>  ) }  **export default** *App* |

### 重新运行，效果是一样的

|  |
| --- |
|  |