# 什么是props？

## 在react中，props就是指properties的简写，就是指属性，它可以用于在组件之间进行数据菜单

# 有参数的函数组件

## 我们以前使用的函数组件都是没有参数的，但是react支持带参数的函数组件

## 实例

### 我们还是使用reactdemo3-component项目，在src目录下面新建一个文件夹：11react-props在这个文件夹里面新建一个FunctionProps.jsx组件文件，用快捷键rsc生成骨架，并且添加一些渲染代码

|  |
| --- |
| **import *React* from 'react'**;  **const** *FunctionProps* = (props) => {  **return** (  <**div**>  <**h4**>name : {props.**name**}</**h4**>  <**h4**>age: {props.**age**}</**h4**>  <**h4**>gender:{props.**gender**}</**h4**>  <**h4**>money:{props.money}</**h4**>  </**div**>  ); };  **export default** *FunctionProps*; |

## 然后在App.jsx文件里面添加对组件的引用，注意此时必须传递参数

|  |
| --- |
| **import** { *useState* } **from 'react'** *// 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"**; **import** ClassEvent **from "./02-class-component-events/ClassEvent"**; **import** ClassState **from "./03class-component-state/ClassState"**; **import** ClassComputed **from "./04class-computed-property/ClassComputed" import** ClassLifeCycle **from "./05class-comp-life-cycle/ClassLifeCycle"**; *// import SideMenu from "./side-menu/SideMenu";* **import** SideMenu2 **from "./side-menu-v2/SideMenu2"**; **import** ComponentStyle **from "./06component-style/ComponentStyle"**; **import** *FunctionState* **from "./07readct-hook-useState/FunctionState"**; **import** *FunctionLifeCycle* **from "./08react-Hook-useEffect/FunctionLifeCycle"**; **import** *FunctionComputed* **from "./09react-Hook-useMemo/FunctionComputed"**; **import** *FunctionComputed2* **from "./09react-Hook-useMemo/FunctionComputed2"**; **import** *FunctionRef* **from "./10react-Hook-useRef/FunctionRef"**; **import** *TodoList* **from "./exercise-todoList/TodoList"**; **import** *FunctionProps* **from "./11react-props/FunctionProps"**;  **function** *App*() {   **return** (  <>  {*/\*<h1>React demo3</h1>\*/*}  {*/\*<FuncComponent/>\*/*}  {*/\*<ClassComponent/>\*/*}  {*/\*<ClassEvent/>\*/*}  {*/\* <ClassState></ClassState>\*/*}  {*/\* <ClassComputed></ClassComputed>\*/*}  {*/\* <ClassLifeCycle></ClassLifeCycle>\*/*}  {*/\* <SideMenu></SideMenu>\*/*}  {*/\* <SideMenu2/>\*/*}  {*/\* <ComponentStyle></ComponentStyle>\*/*}  {*/\* <FunctionState></FunctionState>\*/*}  {*/\* <FunctionLifeCycle></FunctionLifeCycle>\*/*}  {*/\* <FunctionComputed></FunctionComputed>\*/*}  {*/\* <FunctionComputed2></FunctionComputed2>\*/*}  {*/\* <FunctionRef/>\*/*}  {*/\*<TodoList/>\*/*}  <**FunctionProps name = 'jack' age=**{30} **gender='male' money=**{100000} />  </>  ) }  **export default** *App* |

### 效果

|  |
| --- |
|  |

## App.jsx还可以怎么写

|  |
| --- |
| **import** { *useState* } **from 'react'** *// 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"**; **import** ClassEvent **from "./02-class-component-events/ClassEvent"**; **import** ClassState **from "./03class-component-state/ClassState"**; **import** ClassComputed **from "./04class-computed-property/ClassComputed" import** ClassLifeCycle **from "./05class-comp-life-cycle/ClassLifeCycle"**; *// import SideMenu from "./side-menu/SideMenu";* **import** SideMenu2 **from "./side-menu-v2/SideMenu2"**; **import** ComponentStyle **from "./06component-style/ComponentStyle"**; **import** *FunctionState* **from "./07readct-hook-useState/FunctionState"**; **import** *FunctionLifeCycle* **from "./08react-Hook-useEffect/FunctionLifeCycle"**; **import** *FunctionComputed* **from "./09react-Hook-useMemo/FunctionComputed"**; **import** *FunctionComputed2* **from "./09react-Hook-useMemo/FunctionComputed2"**; **import** *FunctionRef* **from "./10react-Hook-useRef/FunctionRef"**; **import** *TodoList* **from "./exercise-todoList/TodoList"**; **import** *FunctionProps* **from "./11react-props/FunctionProps"**;  **function** *App*() {  **let** props={  **name** :**'jack'** ,  **age**:35,  **gender**:**'male'** ,  **money**:9000000  }  **return** (  <>  {*/\*<h1>React demo3</h1>\*/*}  {*/\*<FuncComponent/>\*/*}  {*/\*<ClassComponent/>\*/*}  {*/\*<ClassEvent/>\*/*}  {*/\* <ClassState></ClassState>\*/*}  {*/\* <ClassComputed></ClassComputed>\*/*}  {*/\* <ClassLifeCycle></ClassLifeCycle>\*/*}  {*/\* <SideMenu></SideMenu>\*/*}  {*/\* <SideMenu2/>\*/*}  {*/\* <ComponentStyle></ComponentStyle>\*/*}  {*/\* <FunctionState></FunctionState>\*/*}  {*/\* <FunctionLifeCycle></FunctionLifeCycle>\*/*}  {*/\* <FunctionComputed></FunctionComputed>\*/*}  {*/\* <FunctionComputed2></FunctionComputed2>\*/*}  {*/\* <FunctionRef/>\*/*}  {*/\*<TodoList/>\*/*}  {*/\* <FunctionProps name = 'jack' age={30} gender='male' money={100000} />\*/*}  <**FunctionProps** {...props} />  </>  ) }  **export default** *App* |

### 注意：从这一节课快速，我们需要学习多组件开发，比如组件的嵌套，父组件，子组件兄弟组件等等

# 明白了上面的原理后我们来深入学习

# 父组件和子组件的参数传递

## 在11react-props文件夹里面新建两个组件文件：FatherProp.jsx和ChildProp.jsx文件，用快捷键rsc回车创建文件骨架，并且在父组件中添加对子组件的引用，

### FatherProp.jsx

|  |
| --- |
| **import *React* from 'react'**; **import** *ChildProp* **from "./ChildProp"**;  **const** *FatherProp* = () => {  **return** (  <**div**>  <**h3**>FatherProp</**h3**>  <**ChildProp**/>  </**div**>  ); };  **export default** *FatherProp*; |

### ChildProp.jsx

|  |
| --- |
| **import *React* from 'react'**;  **const** *ChildProp* = () => {  **return** (  <**div**>  <**h3**>ChildProp</**h3**>  </**div**>  ); };  **export default** *ChildProp*; |

### 然后在App.jsx中添加对FatherProp.jsx的引用

|  |
| --- |
| **import** { *useState* } **from 'react'** *// 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"**; **import** ClassEvent **from "./02-class-component-events/ClassEvent"**; **import** ClassState **from "./03class-component-state/ClassState"**; **import** ClassComputed **from "./04class-computed-property/ClassComputed" import** ClassLifeCycle **from "./05class-comp-life-cycle/ClassLifeCycle"**; *// import SideMenu from "./side-menu/SideMenu";* **import** SideMenu2 **from "./side-menu-v2/SideMenu2"**; **import** ComponentStyle **from "./06component-style/ComponentStyle"**; **import** *FunctionState* **from "./07readct-hook-useState/FunctionState"**; **import** *FunctionLifeCycle* **from "./08react-Hook-useEffect/FunctionLifeCycle"**; **import** *FunctionComputed* **from "./09react-Hook-useMemo/FunctionComputed"**; **import** *FunctionComputed2* **from "./09react-Hook-useMemo/FunctionComputed2"**; **import** *FunctionRef* **from "./10react-Hook-useRef/FunctionRef"**; **import** *TodoList* **from "./exercise-todoList/TodoList"**; **import** *FunctionProps* **from "./11react-props/FunctionProps"**; **import** *FatherProp* **from "./11react-props/FatherProp"**;  **function** *App*() {  **let** props={  **name** :**'jack'** ,  **age**:35,  **gender**:**'male'** ,  **money**:9000000  }  **return** (  <>  {*/\*<h1>React demo3</h1>\*/*}  {*/\*<FuncComponent/>\*/*}  {*/\*<ClassComponent/>\*/*}  {*/\*<ClassEvent/>\*/*}  {*/\* <ClassState></ClassState>\*/*}  {*/\* <ClassComputed></ClassComputed>\*/*}  {*/\* <ClassLifeCycle></ClassLifeCycle>\*/*}  {*/\* <SideMenu></SideMenu>\*/*}  {*/\* <SideMenu2/>\*/*}  {*/\* <ComponentStyle></ComponentStyle>\*/*}  {*/\* <FunctionState></FunctionState>\*/*}  {*/\* <FunctionLifeCycle></FunctionLifeCycle>\*/*}  {*/\* <FunctionComputed></FunctionComputed>\*/*}  {*/\* <FunctionComputed2></FunctionComputed2>\*/*}  {*/\* <FunctionRef/>\*/*}  {*/\*<TodoList/>\*/*}  {*/\* <FunctionProps name = 'jack' age={30} gender='male' money={100000} />\*/*}  {*/\* <FunctionProps {...props} />\*/*}  <**FatherProp**/>  </>  ) }  **export default** *App* |

### 测试一下，正常

|  |
| --- |
|  |

## 好我们在父组件里面调用子组件，我们想给子组件传递一些值，如下

|  |
| --- |
| //FatherPop.jsx  **import *React***, {*useState*} **from 'react'**; **import** *ChildProp* **from "./ChildProp"**;  **const** *FatherProp* = () => {  **let** [gender,setGender] = *useState*(**'男'**)  **return** (  <**div**>  <**h2**>FatherProp</**h2**>  <**ChildProp name='Jack' age=**{25} **gender=**{gender}/>  </**div**>  ); };  **export default** *FatherProp*; |

## 此时不会报错，但是子组件没有接收到数据，因为子组件压根没有此时

## 我们可以给子组件添加一个形参props对象来接受父组件传递过来的值，然后使用

|  |
| --- |
| **import *React* from 'react'**;  **const** *ChildProp* = (props) => {  **return** (  <**div**>  <**h3**>ChildProp</**h3**>  <**p**>name:{props.**name**}</**p**>  <**p**>age:{props.**age**}</**p**>  <**p**>gender:{props.**gender**}</**p**>  </**div**>  ); };  **export default** *ChildProp*; |

### 效果

|  |
| --- |
|  |

### 注意传递数据时，只有字符串类型可以直接写，其他类型都需要花括号，也可以使用解构表达式来获取数据

|  |
| --- |
| //ChildProp.jsx  **import *React* from 'react'**;  **const** *ChildProp* = (props) => {  **let** {name,age,gender} = props  **return** (  <**div**>  <**h3**>ChildProp</**h3**>  <**p**>name:{name}</**p**>  <**p**>age:{age}</**p**>  <**p**>gender:{gender}</**p**>  </**div**>  ); };  **export default** *ChildProp*; |

### 效果是一样的

## 还可以怎么写

|  |
| --- |
| //ChildProp.jsx  **import *React* from 'react'**;  **const** *ChildProp* = ({name,age,gender}) => {  **return** (  <**div**>  <**h3**>ChildProp</**h3**>  <**p**>name:{name}</**p**>  <**p**>age:{age}</**p**>  <**p**>gender:{gender}</**p**>  </**div**>  ); };  **export default** *ChildProp*; |

### 不过不建议怎么写

# Props默认值

## 我们可以给props设置默认值，注意这个默认值写在组件外面，使用的是组件的defaultProps

|  |
| --- |
| //ChildProp.jsx  **import *React* from 'react'**;  **const** *ChildProp* = (props) => {  **let** {name,age,gender} = props  **return** (  <**div**>  <**h3**>ChildProp</**h3**>  <**p**>name:{name}</**p**>  <**p**>age:{age}</**p**>  <**p**>gender:{gender}</**p**>  </**div**>  ); }; *ChildProp*.**defaultProps**={  **gender**:**'女'** } **export default** *ChildProp*; |

## 此时如果父组件不传递gender，也可以正常显示

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**; **import** *ChildProp* **from "./ChildProp"**;  **const** *FatherProp* = () => {  **let** [gender,setGender] = *useState*(**'男'**)  **return** (  <**div**>  <**h2**>FatherProp</**h2**>  <**ChildProp name='Jack' age=**{25} />  </**div**>  ); };  **export default** *FatherProp*; |

### 效果

|  |
| --- |
|  |

# props的验证

## 可以在需要接收数据的组件里面指定需要接收数据的数据类型，如果外部组件传递的数据类型不匹配就会抛异常

## 需要一个插件：prop-types

## 我们使用yarn安装：yarn add prop-types

### 我们在子组件里面添加验证代码，还是在组件外面设置

|  |
| --- |
| **import *React* from 'react'**; **import** pt **from 'prop-types' const** *ChildProp* = (props) => {  **let** {name,age,gender} = props  **return** (  <**div**>  <**h3**>ChildProp</**h3**>  <**p**>name:{name}</**p**>  <**p**>age:{age}</**p**>  <**p**>gender:{gender}</**p**>  </**div**>  ); }; *ChildProp*.**defaultProps**={  **gender**:**'女'** } *ChildProp*.**propTypes** = {  **name**:pt.***string***,  **age**:pt.***number***,  **gender**:pt.***string*** } **export default** *ChildProp*; |

## 此时我们在FatherProp的组件引用里面把age的值改为“25”

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**; **import** *ChildProp* **from "./ChildProp"**;   **const** *FatherProp* = () => {  **let** [gender,setGender] = *useState*(**'男'**)  **return** (  <**div**>  <**h2**>FatherProp</**h2**>  <**ChildProp name='Jack' age='25'** />  </**div**>  ); };  **export default** *FatherProp*; |

### 运行，发现报错了

|  |
| --- |
|  |

### 说明验证生效了

## 我们把age重新改为number，它有正常了

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**; **import** *ChildProp* **from "./ChildProp"**;   **const** *FatherProp* = () => {  **let** [gender,setGender] = *useState*(**'男'**)  **return** (  <**div**>  <**h2**>FatherProp</**h2**>  <**ChildProp name='Jack' age=**{30} />  </**div**>  ); };  **export default** *FatherProp*; |
|  |

## prop-types常见类型

### 以pt来代表prop-types

|  |  |
| --- | --- |
| 类型 | 说明 |
| **pt.string** | **字符串** |
| **pt.number** | **数字** |
| **pt.bool** | **布尔值** |
| **pt.array** | **数组** |
| **pt.object** | **对象** |
| **pt.func** | **函数** |
| **pt.oneOfType([pt.type1,pt.type2,pt.type3])** | **是指定类型的其中一种** |

#### 我们把上面的代码修改一下

#### App.jsx

|  |
| --- |
| *// eslint-disable-next-line no-unused-vars* **import** { *useState* } **from 'react'** *// import FunctionRef from "./10react-Hook-useRef/FunctionRef";* **import** *FunctionProps* **from "./11react-props/FunctionProps"**; *// import FunctionComputed from "./09react-Hook-useMemo/FunctionComputed"; // import FunctionLifeCycle from "./08react-Hook-useEffect/FunctionLifeCycle"; // import FunctionState from "./07readct-hook-useState/FunctionState"; // import ComponentStyle from "./06component-style/ComponentStyle"; // import reactLogo from './assets/react.svg' // import viteLogo from '/vite.svg' // import './App.css' // import FuncComponent from "./01-compoment-basic/FuncComponent";* **function** *App*() {  **let** props={  **name** :**'jack'** ,  **age**:35,  **gender**:**'male'** ,  **money**:9000000  }  **return** (  <>  {*/\*<FuncComponent></FuncComponent>\*/*}  {*/\* <ComponentStyle></ComponentStyle>\*/*}  {*/\* <FunctionState></FunctionState>\*/*}  {*/\* <FunctionLifeCycle></FunctionLifeCycle>\*/*}  {*/\* <FunctionComputed></FunctionComputed>\*/*}  {*/\* <FunctionRef></FunctionRef>\*/*}  <**FunctionProps**></**FunctionProps**>  </>  ) }  **export default** *App* |

#### FunctionProps.jsx

|  |
| --- |
| **import *React* from 'react'**; **import** *FatherProp* **from "./FatherProp"**;   **const** *FunctionProps* = () => {  **return** (  <**div**>  <**FatherProp name="Jackline" age=**{30} **gender="female"**/>  </**div**>  ); };  **export default** *FunctionProps*; |

#### FatherProp.jsx

|  |
| --- |
| **import *React* from 'react'**; **import** *ChildProp* **from "./ChildProp"**;   **const** *FatherProp* = (props) => {  **return** (  <**div**>  <**ul**>  <**li**>Name:{props.**name**}</**li**>  <**li**>age:{props.**age**}</**li**>  <**li**>gender:{props.**gender**}</**li**>  </**ul**>  <**ChildProp chinese=**{100} **english=**{80} **chemestry=**{70}></**ChildProp**>  </**div**>  ); };  **export default** *FatherProp*; |

#### ChildProp.jsx

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**;  **const** *ChildProp* = (props) => {    **return** (  <**div**>  <**p**>{props.**chinese**}</**p**><**p**>{props.**english**}</**p**><**p**>{props.**chemestry**}</**p**>  </**div**>  ); };  **export default** *ChildProp*; |

### 效果如下

|  |
| --- |
|  |

# 上面是父亲传递数据给孩子，那么孩子这么传递数据给父亲

## 方法1.使用父亲传递给孩子的回调函数，我们修改一下代码

#### App.jsx

|  |
| --- |
| *// eslint-disable-next-line no-unused-vars* **import** { *useState* } **from 'react'** *// import FunctionRef from "./10react-Hook-useRef/FunctionRef";* **import** *FunctionProps* **from "./11react-props/FunctionProps"**; *// import FunctionComputed from "./09react-Hook-useMemo/FunctionComputed"; // import FunctionLifeCycle from "./08react-Hook-useEffect/FunctionLifeCycle"; // import FunctionState from "./07readct-hook-useState/FunctionState"; // import ComponentStyle from "./06component-style/ComponentStyle"; // import reactLogo from './assets/react.svg' // import viteLogo from '/vite.svg' // import './App.css' // import FuncComponent from "./01-compoment-basic/FuncComponent";* **function** *App*() {  **let** props={  **name** :**'jack'** ,  **age**:35,  **gender**:**'male'** ,  **money**:9000000  }  **return** (  <>  {*/\*<FuncComponent></FuncComponent>\*/*}  {*/\* <ComponentStyle></ComponentStyle>\*/*}  {*/\* <FunctionState></FunctionState>\*/*}  {*/\* <FunctionLifeCycle></FunctionLifeCycle>\*/*}  {*/\* <FunctionComputed></FunctionComputed>\*/*}  {*/\* <FunctionRef></FunctionRef>\*/*}  <**FunctionProps**></**FunctionProps**>  </>  ) }  **export default** *App* |

#### FunctionProps.jsx

|  |
| --- |
| **import *React* from 'react'**; **import** *FatherProp* **from "./FatherProp"**;   **const** *FunctionProps* = () => {  **return** (  <**div**>  <**FatherProp name="Jackline" age=**{30} **gender="female"**/>  </**div**>  ); };  **export default** *FunctionProps*; |

#### FatherProp.jsx

|  |
| --- |
| **import *React* from 'react'**; **import** *ChildProp* **from "./ChildProp"**;   **const** *FatherProp* = (props) => {  **let** onClick = (value)=>{  ***console***.log(**"value from my child:"**,value)  }  **return** (  <**div**>  <**ul**>  <**li**>Name:{props.**name**}</**li**>  <**li**>age:{props.**age**}</**li**>  <**li**>gender:{props.**gender**}</**li**>  </**ul**>  <**ChildProp chinese=**{100} **english=**{80} **chemestry=**{70} **onClick=**{onClick}></**ChildProp**>  </**div**>  ); };  **export default** *FatherProp*; |

#### ChildProp.jsx

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**;  **const** *ChildProp* = (props) => {  **let** onChildClick = ()=>{  **let** avg = (props.**chinese**+props.**english**+props.**chemestry**)/3  props.**onClick**(avg)  }  **return** (  <**div**>  <**p**>{props.**chinese**}</**p**><**p**>{props.**english**}</**p**><**p**>{props.**chemestry**}</**p**>  <**button onClick=**{onChildClick}>给父亲传递数据</**button**>  </**div**>  ); };  **export default** *ChildProp*; |

#### 效果：

|  |
| --- |
|  |

## 方法2.使用ref，需要把子组件改为forwardRef，比较复杂

|  |
| --- |
| 父组件代码  js  复制代码  const App = () => {  const childrenRef = useRef();  const onFatnerClick = () => {  const {setCountChildren, countChildren} = childrenRef.current || {};  setCountChildren?.(countChildren + 1);  }  return <>  <button onClick={onFatnerClick}>点击父组件，修改ref子组件的count</button>  <RefChildrenComponent ref={childrenRef}/>  </>  }  子组件代码  js  复制代码  const RefChildrenComponent = forwardRef((props, ref) => {  const [ countChildren, setCountChildren ] = useState(0);  useImperativeHandle(ref,() => {  return {  setCountChildren,  countChildren  }  },[countChildren])  return <>  ref的子组件{countChildren}  </>  })  子组件通过useImperativeHandle向父组件传递了setCountChildren和countChildren，父组件触发自己的点击事件后，调用了子组件的setCountChildren方法，改变了子组件的countChildren。从而使页面更新。 |

# 还有兄弟组件的传递数据有2种方法通过父组件或者使用事件总线（第三方库）