# 受控组件

## 是指表单受react控制，不受以后控制

在 HTML 中，表单元素（如<input>、 <textarea> 和 <select>）通常自己维护 state，并根据用户输入进行更新。而在 React 中，可变状态（mutable state）通常保存在组件的 state 属性中，并且只能通过使用 [setState()](https://zh-hans.legacy.reactjs.org/docs/react-component.html#setstate)来更新。

我们可以把两者结合起来，使 React 的 state 成为“唯一数据源”。渲染表单的 React 组件还控制着用户输入过程中表单发生的操作。被 React 以这种方式控制取值的表单输入元素就叫做“受控组件”。

### 简单实例

|  |
| --- |
| 我们可以将表单写为受控组件：  **class NameForm extends React.Component {**  **constructor(props) {**  **super(props);**  **this.state = {value: ''};**  **this.handleChange = this.handleChange.bind(this);**  **this.handleSubmit = this.handleSubmit.bind(this);**  **}**  **handleChange(event) {**  **this.setState({value: event.target.value});**  **}**  **handleSubmit(event) {**  **alert('提交的名字: ' + this.state.value);**  **event.preventDefault();**  **}**  **render() {**  **return (**  **<form onSubmit={this.handleSubmit}>**  **<label>**  **名字:<input type="text" value={this.state.value} onChange={this.handleChange} />**  **</label>**  **<input type="submit" value="提交" />**  **</form>**  **);**  **}**  **}** |

### 把上面的内容保存为一个jsx文件，然后在App.jsx中引用该组件，运行程序，效果如下

|  |
| --- |
|  |

## 我们再写一个受控组件的小例子在src目录下面新建一个文件夹：15react-form在这个文件夹下面创建一个FunctionFormControlled.jsx组件文件，用rsc创建骨架在App.jsx文件中添加对该组件的引用，然后在组件文件中添加一些代码

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**;  **const** *FunctionFormControlled* = () => {  **let** [msg,setMsg] = *useState*(**'hello'**)  **let** [isChecked,setIsChecked] = *useState*(**true**)  **return** (  <**div**>  <**h3**>受控组件：受react控制</**h3**>  <**input type="text" value=**{msg}/>  </**div**>  ); };  **export default** *FunctionFormControlled*; |

## 一运行代码发现报错了

|  |
| --- |
|  |

### 因为我们没有给他设置onChange事件

## 我们给他添加onChange事件

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**;  **const** *FunctionFormControlled* = () => {  **let** [msg,setMsg] = *useState*(**'hello'**)  **let** [isChecked,setIsChecked] = *useState*(**true**)  **let** handleChange=(e)=>{  setMsg(e.**target**.**value**)  *alert*(**'hello'**)  }  **return** (  <**div**>  <**h3**>受控组件：受react控制</**h3**>  <**input type="text" value=**{msg} **onChange=**{()=>handleChange}/>  </**div**>  ); };  **export default** *FunctionFormControlled*; |

### 我们发现此时运行程序，用户是无法修改输入框的值的，那怎么样才能编辑内容？

### 需要在onChange事件里面传递一个事件对象，然后获取他的target的value再用setXXX来设置值这个xxx是输入框的value属性的值

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**;  **const** *FunctionFormControlled* = () => {  **let** [msg,setMsg] = *useState*(**'hello'**)  **let** [isChecked,setIsChecked] = *useState*(**true**)  **let** handleChange=(e)=>{  ***console***.log(e.**target**)  setMsg(e.**target**.**value**)   }  **return** (  <**div**>  <**h3**>受控组件：受react控制</**h3**>  <**input type="text" value=**{msg} **onChange=**{(e)=>{  //***console***.log(e.**target**.**value**)  setMsg(e.**target**.**value**)  }} />  </**div**>  ); };  **export default** *FunctionFormControlled*; |

## 我们来看一下受控的单选按钮组件，文件在组件文件里面添加两个单选按钮

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**;  **const** *FunctionFormControlled* = () => {  **let** [msg,setMsg] = *useState*(**'hello'**)  **let** [gender,setGender] = *useState*(**'女'**)  **let** [isChecked,setIsChecked] = *useState*(**true**)  **let** handleChange=(e)=>{  ***console***.log(e.**target**)  setMsg(e.**target**.**value**)   }  **return** (  <**div**>  <**h3**>受控组件：受react控制</**h3**>  <**input type="text" value=**{msg} **onChange=**{(e)=>{  ***console***.log(e.**target**.**value**)  setMsg(e.**target**.**value**)  }} />  <**input type="radio" checked=**{gender===**'男'**} **onChange=**{(e)=>{  setGender(**'男'**)  }}/>男  <**input type="radio" checked=**{gender===**'女'**} **onChange=**{(e)=>{  setGender(**'女'**)  }}/>女  </**div**>  ); };  **export default** *FunctionFormControlled*; |

## 我们来看看复选框受控组件，复选框有很多用法，需要根据实际情况来使用

### 我的版本（只写了一个其他都是一样的写法）

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**;  **const** *FunctionFormControlled* = () => {  **let** [msg,setMsg] = *useState*(**'hello'**)  **let** [gender,setGender] = *useState*(**'女'**)  **let** [isChecked,setIsChecked] = *useState*(**true**)  **let** [likes,setLikes] = *useState*([])  **let** handleChange=(e)=>{  ***console***.log(e.**target**)  setMsg(e.**target**.**value**)   }  **return** (  <**div**>  <**h3**>受控组件：受react控制</**h3**>  <**input type="text" value=**{msg} **onChange=**{(e)=>{  ***console***.log(e.**target**.**value**)  setMsg(e.**target**.**value**)  }} />  <**p**>性别：{gender}</**p**>  <**input type="radio" checked=**{gender===**'男'**} **onChange=**{(e)=>{  setGender(**'男'**)  }}/>男  <**input type="radio" checked=**{gender===**'女'**} **onChange=**{(e)=>{  setGender(**'女'**)  }}/>女  <**ul**>  <**li**><**h4**>兴趣爱好</**h4**></**li**>  {likes.map((item,index)=>{  **return** <**li key=**{index}>{item}</**li**>  })}  </**ul**>  <**input type="checkbox" checked=**{likes.includes(**'泡妞'**)} **onChange=**{()=>{  **if**(!likes.includes(**'泡妞'**)){  setLikes([  ...likes,  **'泡妞'** ])  } **else** {  likes.splice(likes.indexOf(**'泡妞'**),1)  setLikes([...likes])  }   }} />泡妞  <**input type="checkbox" checked=**{likes.includes(**'逛街'**)} **onChange=**{()=>{   }} />逛街  <**input type="checkbox" checked=**{likes.includes(**'上管子'**)} **onChange=**{()=>{   }} />上管子   </**div**>  ); };  **export default** *FunctionFormControlled*; |

### 老师的版本+我的改进包括方法抽取

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**;  **function** setMyItem(e, setLikes, likes,myItem) {  **if** (e.**target**.**checked**) {  setLikes([  ...likes,  myItem  ])  } **else** {  setLikes(likes.filter(item => item !== myItem))  } }  **const** *FunctionFormControlled* = () => {  **let** [msg,setMsg] = *useState*(**'hello'**)  **let** [gender,setGender] = *useState*(**'女'**)  **let** [isChecked,setIsChecked] = *useState*(**true**)  **let** [likes,setLikes] = *useState*([])  **let** handleChange=(e)=>{  ***console***.log(e.**target**)  setMsg(e.**target**.**value**)   }  **return** (  <**div**>  <**h3**>受控组件：受react控制</**h3**>  <**input type="text" value=**{msg} **onChange=**{(e)=>{  ***console***.log(e.**target**.**value**)  setMsg(e.**target**.**value**)  }} />  <**p**>性别：{gender}</**p**>  <**input type="radio" checked=**{gender===**'男'**} **onChange=**{(e)=>{  setGender(**'男'**)  }}/>男  <**input type="radio" checked=**{gender===**'女'**} **onChange=**{(e)=>{  setGender(**'女'**)  }}/>女  <**ul**>  <**h4**>兴趣爱好</**h4**>  {likes.map((item,index)=>{  **return** <**li key=**{index}>{item}</**li**>  })}  </**ul**>  <**input type="checkbox" checked=**{likes.includes(**'泡妞'**)} **onChange=**{(e)=>{  setMyItem(e, setLikes, likes,**'泡妞'**);  }} />泡妞  <**input type="checkbox" checked=**{likes.includes(**'逛街'**)} **onChange=**{(e)=>{  setMyItem(e, setLikes, likes,**'逛街'**)  }} />逛街  <**input type="checkbox" checked=**{likes.includes(**'上管子'**)} **onChange=**{(e)=>{  setMyItem(e, setLikes, likes,**'上管子'**)  }} />上管子   </**div**>  ); };  **export default** *FunctionFormControlled*; |

#### 可以看到复选框比较复杂，不过慢慢体会也觉得很有道理

## 现在我们来看看受控下拉列表

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**;  **function** setMyItem(e, setLikes, likes,myItem) {  **if** (e.**target**.**checked**) {  setLikes([  ...likes,  myItem  ])  } **else** {  setLikes(likes.filter(item => item !== myItem))  } }  **const** *FunctionFormControlled* = () => {  **let** [msg,setMsg] = *useState*(**'hello'**)  **let** [gender,setGender] = *useState*(**'女'**)  **let** [isChecked,setIsChecked] = *useState*(**true**)  **let** [likes,setLikes] = *useState*([])  **let** [addr,setAddr] = *useState*(**'云南'**)  **let** handleChange=(e)=>{  ***console***.log(e.**target**)  setMsg(e.**target**.**value**)   }  **return** (  <**div**>  <**h3**>受控组件：受react控制</**h3**>  <**input type="text" value=**{msg} **onChange=**{(e)=>{  ***console***.log(e.**target**.**value**)  setMsg(e.**target**.**value**)  }} />  <**p**>性别：{gender}</**p**>  <**input type="radio" checked=**{gender===**'男'**} **onChange=**{(e)=>{  setGender(**'男'**)  }}/>男  <**input type="radio" checked=**{gender===**'女'**} **onChange=**{(e)=>{  setGender(**'女'**)  }}/>女  <**ul**>  <**h4**>兴趣爱好</**h4**>  {likes.map((item,index)=>{  **return** <**li key=**{index}>{item}</**li**>  })}  </**ul**>  <**input type="checkbox" checked=**{likes.includes(**'泡妞'**)} **onChange=**{(e)=>{  setMyItem(e, setLikes, likes,**'泡妞'**);  }} />泡妞  <**input type="checkbox" checked=**{likes.includes(**'逛街'**)} **onChange=**{(e)=>{  setMyItem(e, setLikes, likes,**'逛街'**)  }} />逛街  <**input type="checkbox" checked=**{likes.includes(**'上管子'**)} **onChange=**{(e)=>{  setMyItem(e, setLikes, likes,**'上管子'**)  }} />上管子  <**p**>所在省份：{addr}</**p**>  <**select value=**{addr} **onChange=**{(e)=>{  setAddr(e.**target**.**value**)  }}>  <**option value="四川"**>四川</**option**>  <**option value="广东"**>广东</**option**>  <**option value="云南"**>云南</**option**>  <**option value="贵州"**>贵州</**option**>  </**select**>  </**div**>  ); };  **export default** *FunctionFormControlled*; |

## 总结，如果需要修改受控组件的值，需要3步，

### 第一步需要用useState这个Hook生成一条数据，包含set方法，

### 第二步，把控件的value设置为这条数据，

### 第三步，给控件添加onChange方法，在这个方法中调用set方法来修改这条数据

### 每一个受控组件都必须有一个value属性和一个onChange方法，否则无法修改数据

### 注意复选框的操作稍微有点复杂，需要慢慢品味

# 非受控组件

## 是值表单不受react控制

在大多数情况下，我们推荐使用 [受控组件](https://zh-hans.legacy.reactjs.org/docs/forms.html#controlled-components) 来处理表单数据。在一个受控组件中，表单数据是由 React 组件来管理的。另一种替代方案是使用非受控组件，这时表单数据将交由 DOM 节点来处理。

要编写一个非受控组件，而不是为每个状态更新都编写数据处理函数，你可以 [使用 ref](https://zh-hans.legacy.reactjs.org/docs/refs-and-the-dom.html) 来从 DOM 节点中获取表单数据。

### 例如，下面的代码使用非受控组件接受一个表单的值：

|  |
| --- |
| **class NameForm extends React.Component {**  **constructor(props) {**  **super(props);**  **this.handleSubmit = this.handleSubmit.bind(this);**  **this.input = React.createRef();**  **}**  **handleSubmit(event) {**  **alert('A name was submitted: ' + this.input.current.value);**  **event.preventDefault();**  **}**  **render() {**  **return (**  **<form onSubmit={this.handleSubmit}>**  **<label>**  **Name:<input type="text" ref={this.input} />**  **</label>**  **<input type="submit" value="Submit" />**  **</form>**  **);**  **}**  **}** |

## 我们来写一个非受控表单的小例子

## 还是使用reactdemo3项目，在src目录下15react-form在这个文件夹下面创建一个FunctionFormUnControlled.jsx函数组件文件，可以使用rsc创建骨架

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

## 在App.jsx文件中添加对该组件的引用

|  |
| --- |
| **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"**; **import** *FatherEvent* **from "./12comp-communication/FatherEvent"**; **import** *GrandFather* **from "./13multilayer-comp-communication/GrandFather"**; **import** *FatherMemo* **from "./14func-component-optimize/FatherMemo"**; **import** NameForm **from "./NameForm"**; **import** *FunctionFormUnControlled* **from "./15react-form/FunctionFormUnControlled"**;   **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/>\*/*}  {*/\* <FatherEvent/>\*/*}  {*/\* <GrandFather/>\*/*}  {*/\* <FatherMemo/>\*/*}  {*/\* <NameForm/>\*/*}  <**FunctionFormUnControlled**/>  </>  ) }  **export default** *App* |

## 我们在组件文件里面添加一个输入框，代码如下

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**;  **const** *FunctionFormUnControlled* = () => {  **let** [msg,setMsg] = *useState*(**'hello'**)  **return** (  <**div**>  <**h3**>非受控组件</**h3**>  <**form action=""**>  <**input type="text" value=**{msg}/>  </**form**>  </**div**>  ); };  **export default** *FunctionFormUnControlled*; |

## 运行程序，发现我们并不能修改输入框的内容，然后打开开发工具，发现他说我们需要设置defaultValue，我们修改一下代码

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**;  **const** *FunctionFormUnControlled* = () => {  **let** [msg,setMsg] = *useState*(**'hello'**)  **return** (  <**div**>  <**h3**>非受控组件</**h3**>  <**form action=""**>  <**input type="text" defaultValue=**{msg}/>  </**form**>  </**div**>  ); };  **export default** *FunctionFormUnControlled*; |

## 运行程序，发现它现在可以修改了

|  |
| --- |
|  |
|  |

### 怎么体现控件不受react控制？

### 我们添加一个按钮，和一个p标签在p里面显示msg的内容，我们给按钮添加点击事件，使他能够修改msg，把form标签删除

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**;  **const** *FunctionFormUnControlled* = () => {  **let** [msg,setMsg] = *useState*(**'hello'**)  **return** (  <**div**>  <**h3**>非受控组件</**h3**>    <**p**>{msg}</**p**>  <**input type="text" defaultValue=**{msg}/>  <**button onClick=**{()=>setMsg(**'你好'**)}>change</**button**>   </**div**>  ); };  **export default** *FunctionFormUnControlled*; |

### 我们发现点击按钮后p标签是内容被修改了但是输入框的文字没有改变

|  |
| --- |
|  |
|  |

### 总结：我们给一个控件设置了一个defaultValue后，他就是一个非受控组件

### 我们还可以使用非受控单选按钮

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**;  **const** *FunctionFormUnControlled* = () => {  **let** [msg,setMsg] = *useState*(**'hello'**)  **let** [isChecked,setIsChecked] = *useState*(**true**)  **return** (  <**div**>  <**h3**>非受控组件</**h3**>   <**p**>{msg}</**p**>  <**input type="text" defaultValue=**{msg}/>  <**button onClick=**{()=>setMsg(**'你好'**)}>change</**button**><**br**/>  <**input type="radio" name='gender' defaultChecked=**{isChecked} />男  <**input type="radio" name='gender'** />女  </**div**>  ); };  **export default** *FunctionFormUnControlled*; |

### 复选框也是一样的，我们看一下下拉列表

|  |
| --- |
| **import *React***, {*useState*} **from 'react'**;  **const** *FunctionFormUnControlled* = () => {  **let** [msg,setMsg] = *useState*(**'hello'**)  **let** [isChecked,setIsChecked] = *useState*(**true**)  **return** (  <**div**>  <**h3**>非受控组件</**h3**>   <**p**>{msg}</**p**>  <**input type="text" defaultValue=**{msg}/>  <**button onClick=**{()=>setMsg(**'你好'**)}>change</**button**><**br**/>  <**input type="radio" name='gender' defaultChecked=**{isChecked} />男  <**input type="radio" name='gender'** />女  <**select defaultValue="cpp"**>  <**option value="java"**>java</**option**>  <**option value="cpp"**>cpp</**option**>  <**option value="python"**>python</**option**>  </**select**>  </**div**>  ); };  **export default** *FunctionFormUnControlled*; |

### 注意非受控下拉列表的默认值需要设置在select标签上而不是option上

## 非受控组件总体来说用的不多，了解即可

# react图片处理

## 在react中是无法有传统的方法来渲染本地图片的。

## 我们通过一个例子来说明

## 还是使用reactdemo3的项目，我们在src目录下面创建一个文件夹16react-image-rendering，然后在这个文件夹下面创建一个ImageRender.jsx组件文件，用快捷键rsc创建骨架，然后在App.jsx文件中引用这个组件，另外还需要在同一级目录下面新建一个images文件夹，放置一张图片img1.jpg,项目结构如下

|  |
| --- |
|  |

### 我们使用以前的方法在组件文件里面的div里面添加一个img然后设置他的src为图片路径

|  |
| --- |
| **import *React* from 'react'**;  **const** *ImageRender* = () => {   **return** (  <**div**>  <**h3**>ImageRender</**h3**>  <**img src="./images/img1.jpg" alt="图片"**/>  </**div**>  ); };  **export default** *ImageRender*; |

### 运行程序，我们看不到图片，因为报了一个错误说找不到图片

|  |
| --- |
|  |

### 出现这个问题的原因是什么？

#### 答：因为浏览器其实不能识别jsx语法，我们的项目是需要构建工具如vite等等将这个jsx转化为js代码，然后浏览器才能执行，但是经过处理后，jsx文件中图片的路径已经发生了改变，不是原来的路径了，所以图片找不到

## 解决办法：

### 1.通过import来引入图片，这个一个通用方法，但是你必须指定图片的路径

我们把上面的代码修改一下

|  |
| --- |
| **import *React* from 'react'**; **import** img1 **from './images/img1.jpg' const** *ImageRender* = () => {   **return** (  <**div**>  <**h3**>ImageRender</**h3**>  <**img src=**{img1} **alt="图片"**/>  </**div**>  ); };  **export default** *ImageRender*; |

#### 然后我们运行程序，发现图片可以正常显示了

|  |
| --- |
|  |

### 2.vite创建的react项目的解决方法：new URL引入图片，参数比较多，写法有点怪

|  |
| --- |
| **import *React* from 'react'**; **const** *ImageRender* = () => {  *// console.log(img1)* **return** (  <**div**>  <**h3**>ImageRender</**h3**>  <**img src=**{**new *URL***(**'./images/img1.jpg'**,**import**.**meta**.**url**).**href**} **alt="图片"**/>  </**div**>  ); };  **export default** *ImageRender*; |

### 效果

|  |
| --- |
|  |

### 3.用CRA创建的项目中还可以使用require方法来引入图片

|  |
| --- |
|  |

#### 有时候可能需要default

|  |
| --- |
|  |