收件地址组件

l 手写modal弹框

l 手写form组件

l 手写HOC高阶组件

l 手写Form表单核心功能

 **by eric wen**

分析UI

1、新增地址，弹出表单

2、选择当次收货地址

3、modal弹框组件

4、表单相关的功能

组件设计

Props		
Events	onChange 传递用户选择的地址	
Funtions	createAdress 添加新地址 handleSelect 选择地址	

新增地址弹框组件

Props	Visible 控制弹框显隐	
	onClose 关闭弹框事件	
	onSubmit 传递表单数据	

Form 组件

Input		
select	<Select options={[{ name: '北京', value: 'beijing' }, { name: '上海', value: 'shanghai' }]}></Select>	
radio	<Radio label="先生" >	
radioGroup	<RadioGroup> <Radio label="先生" ></Radio> <Radio label="小姐" ></Radio> </RadioGroup>	
checkbox		
FormItem	HOC高阶组件 用来装饰input\select\radio\checkbox等，使其拥有validate、defalutValue、value、onChange、errors等表单属性及事件	
Form	解析formitem包装的组件，统一返回结构化数据、结构化errors	

组件开发

1、form组件开发

2、新增地址表单，验证form组件

3、地址组件，保存地址到本地，从本地加载地址，选择地址逻辑

Form组件

1、先开发input

2、开发formitem 学习高阶组件

3、开发form学习form表单的核心知识点

4、开发select

5、开发radio

6、开发radioGroup 学习React.clone

程序设计

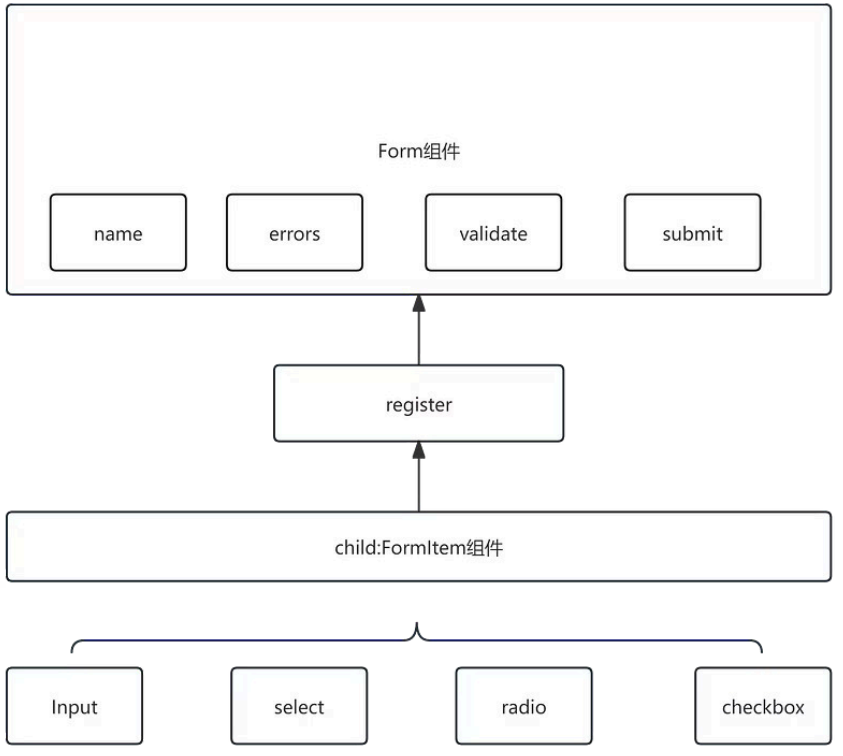
希望使用方式是这样的

```
const handleSubmit = (result) => {
    const {data,errors}=result;
    // data:{name:111,phone:222,Email:333,address:444,paymentMethod:555}
    // errors:{name:'未填写'}
    if(errors){
        console.log(errors);
        return;
    }
    // 提交数据
}
return (


    <Form onSubmit={handleSubmit} >
        <Input name="name" errorText='未填写' required />

        <Input name="phone" required />
        <Input name="Email" required validate={new RegExp(/^[^\s@]+@[^\s@]+\.[^\s@]+$/)} />
        <select name="address" options={[]}/>
        <RadioGroup name="paymentMethod" required>
            <Radio value="credit">信用卡</Radio>
            <Radio value="debit">借記卡</Radio>
        </RadioGroup>

        <button type="submit">提交</button>
    </Form>
)
```



Input

1、符合设计搞的ui

formItem

高阶组件

高阶组件（HOC）是 React 中的一种设计模式，用于增强或复用组件的逻辑。它本质上是一个函数，接收一个组件作为参数，并返回一个新的增强后的组件。

定义： const EnhancedComponent = higherOrderComponent(WrappedComponent); WrappedComponent 是被包装的原始组件。EnhancedComponent 是经过增强后的新组件，通常会添加额外的 props、状态管理、生命周期方法或其他功能。

1. 高阶组件的核心思想 高阶组件的核心思想是通过函数式编程的方式实现组件的逻辑复用和抽象。它不修改原始组件，而是通过组合的方式增强组件的功能。

特点： 复用性： 可以将通用的逻辑（如数据获取、状态管理、校验等）提取到 HOC 中，供多个组件复用。 解耦： HOC 将逻辑与 UI 分离，使得组件更加专注于渲染逻辑。 灵活性： HOC 可以动态地注入 props 或行为，适应不同的场景。 2. 高阶组件的基本结构 以下是一个简单的高阶组件示例：

```
function withLogger(WrappedComponent) {
  return function EnhancedComponent(props) {
    console.log("Props received:", props);

    return <WrappedComponent {...props} />;
  };
}
```

解释： withLogger 是一个高阶组件，接收一个 WrappedComponent。 返回的新组件 EnhancedComponent 在渲染时会打印接收到的 props，然后渲染原始组件。 使用时：

```
const ButtonWithLogger = withLogger(Button);
```

业务逻辑开发

1、订单创建页，基础UI准备

2、地址组件基础UI准备

3、开发弹框modal组件

4、使用我们的form库，地址新增表单

5、保存地址，渲染地址，选择地址

表单优化：

1、分离表单布局类型div

2、暴露更多form实例方法