react中TS类组件的写法

原创 ICPunk 于 2022-06-25 17:34:39 发布 1084 收藏 版权
分类专栏: react 文章标签: react.js javascript 前端

> 类组件最多的就是Typescript接口类型的应用，限制组件状态或者属性符合接口定义

1. state.tsx

```tsx
state = { name: 'xx' }
setState({
    name: 100 // 这个值如果是后台返回的，你不知道类型，运行时才能看到报错；或者调用别人的
})
// 换种写法再试试
interface IState {
    name: string
}
export ...
state: IState = { name: 'xx' }
//
setState({ // 此时这样改并不会提示，因为不是直接改，this.state.name = 100这样才有提示
    name: 100
}
// react有专门的写法
export default class App extends Component<约定属性，约定状态>
export default class App extends Component<any, IState> { // 属性不约束，状态约束
    state = {
        name: 'xxx' // 这时就会提示类型校验
    }
    render() {
        return (
            // ...
            setState({name: "yyy"})
            // ...
        )
    }
}
```

2. todo.tsx

```tsx
interface IState {
  text: string,
  list: string[]
}
export default class App extends Component<any, IState> {
    state = {
        text: "",
        list: []
    }
    myRef = React.createRef<HTMLInputElement>() // 需要指定ref的类型
    render() {
        return (
            <div>
                // <input type="text" value={this.state.text}onChange={
                //     (evt) => {
                //         setState({
                //             text: evt.target.value
                //         })
                //     }
                // }>
                // 或者试试ref
                <input ref={this.myRef}>
                <button onClick={
                    // console.log(state.name)
                    // console.log(this.myRef.current.value)
                    console.log((this.myRef.current as HTMLInputElement).value)
                    this.setSate({
                        list: [...this.sate.list, (this.myRef.current as HTMLInp
                    })
                }></button>
                {
                    this.state.list.map(item=>
```

```
                        <li key={item}>{item}</li>
                    )
                }
            </div>
        )
    }
}
```

3. props.tsx

```tsx
export default class App extends Component {
    render() {
        return (
            <div>
                <Child name="aaa">
            </div>
        )
    }
}
interface IProps {
    name: string
}
class Child extends Component<IProps, any> {
    render() {
        return <div>
            {this.props.name}
        </div>
    }
}

```

4. drawer.tsx

```tsx
export default class App extends Component {
  state = {
    isShow: true,
  };
  render() {
    return (
      <div>
        app
        <Navbar
          title="first page"
          cb={() => {
            this.setState({
              isShow: !this.state.isShow,
            });
          }}
        />
        {this.state.isShow && <Sidebar></Sidebar>}
      </div>
    );
  }
}
interface IProps {
  title: string;
  cb: () => void;
}
class Navbar extends Component<IProps, any> {
  render() {
    return (
      <div>
        Navbar-{this.props.title}
        <button
          onClick={() => {
            this.props.cb();
          }}
        ></button>
      </div>
    );
  }
}
class Sidebar extends Component {
  render() {
    return <div>Sidebar</div>;
  }
}
```