link: null
title: 珠峰架构师成长计划
description: src\index.tsx
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=42 sentences=160, words=1039

1. 安装和初始化

```
$ yarn create react-app antd-form
$ cd antd-form
$ yarn start
```

2. 引入 antd

```
$ yarn add antd
```

3. 支持typescript

```
yarn add typescript @types/node @types/react @types/react-dom @types/jest
```

- typescript
- @types/node 可以获得有关node.js的API的类型说明文件
- @types/react react的声明文件
- @types/react-dom react-dom的声明文件
- @types/jest jest的声明文件

4.表单组件

src\index.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import UserForm from './UserForm';

ReactDOM.render(<UserForm />, document.getElementById('root'));
```

src\UserForm.tsx

```
import * as React from 'react'
import Form  from 'antd/lib/form';
interface Props {
    form: any
}
class UserForm extends React.Component<Props> {
    handleSubmit = (event: React.FormEvent) => {
        event.preventDefault();

        this.props.form.validateFields((error: any, values: any) => {
            console.log(error);
            console.log(values);
        });
    }
    render() {
        let { getFieldDecorator } = this.props.form;
        return (
            <div>
                <Form onSubmit={this.handleSubmit}>
                    <Form.Item label="用户名">
                        {getFieldDecorator('username', {
                            rules: [{ required: true, message: '用户名必须输入' }]
                        })(<input />)}
                    Form.Item>
                    <Form.Item label="密码">
                        {getFieldDecorator('password', {
                            rules: [
                                { required: true, message: '密码必须输入' },
                                { min: 6, message: '密码长度不能小于6位' },
                                { max: 8, message: '密码长度不能大于8位' }
                            ]
                        })(
                            <input />
                        )}
                    Form.Item>
                    <Form.Item>
                        <button>提交button>
                    Form.Item>
                Form>
            div>
        )
    }
}
export default Form.create()(UserForm);
```

5.实现表单组件

src\UserForm.tsx

```
import * as React from 'react'
import Form  from './antd/lib/form';
interface Props {
    form: any
}
class UserForm extends React.Component<Props> {
    render() {
        return (
                <Form>
                    <Form.Item label="用户名">
                      <input />
                    Form.Item>
                    <Form.Item label="密码">
                      <input />
                    Form.Item>
                    <Form.Item>
                        <button>提交button>
                    Form.Item>
                Form>
        )
    }
}
export default Form.create()(UserForm);
```

src\antd\lib\form\index.tsx

```
import Form from './Form';
export default Form;
```

src\antd\lib\form\Form.tsx

```
import  * as React from  'react';
import FormItem from './FormItem';
import create from './create';

export default class  extends React.Component{
    static Item = FormItem;
    static create = create;
    render(){
      return <form {...this.props}/>
    }
}
```

src\antd\lib\form\FormItem.tsx

```
import React, { Component } from 'react'

export interface FormItemProps {
  label?:string
}
export default class FormItem extends React.Component<FormItemProps, any> {
  render(){
    const {label,children} = this.props;
    return (
      <div>
        <label>{label}label>
        {children}
      div>
    );
  }
}
```

src\antd\lib\form\create.tsx

```
import * as React from 'react';
interface WrapProps {}
interface WrapState {}
export default function(){
    return function decorate(WrappedComponent:any) {
        class WrapComponent extends React.Component<WrapProps,WrapState>{
            render(){
                const props = {};
                return <WrappedComponent {...props}/>;
             }
        }
        return WrapComponent;
    }
}
```

6. 获取数据

src\UserForm.tsx

```
import * as React from 'react'
import Form  from './antd/lib/form';
interface Props {
    form: any
}
class UserForm extends React.Component<Props> {
    handleSubmit = (event:React.FormEvent)=>{
        event.preventDefault();
        let values = this.props.form.getFieldsValue();
        console.log(values);
    }
    render() {
        let {getFieldDecorator} = this.props.form;
        return (
            <Form onSubmit={this.handleSubmit}>
                <Form.Item label="用户名">
                    {getFieldDecorator('username')(<input/>)}
                Form.Item>
                <Form.Item label="密码">
                    {getFieldDecorator('password')(<input/>)}
                Form.Item>
                <Form.Item>
                    <button>提交button>
                Form.Item>
            Form>
        )
    }
}
export default Form.create()(UserForm);
```

src/antd/lib/form/Form.tsx

```
import  * as React from  'react';
import FormItem from './FormItem';
import create from './create';
interface FormProps{
  onSubmit:any
}
export default class  extends React.Component<FormProps>{
    static Item = FormItem;
    static create = create;
    render(){
      return <form {...this.props}/>
    }
}
```

src/antd/lib/form/create.tsx

```
import * as React from 'react';
interface FormProps {

}
interface Objects{
    [propName: string]: any;
}
interface FormState {
    values:Objects,
}
export default function(){
    return function decorate(WrappedComponent:any) {
        class Form extends React.Component<FormProps,FormState>{
            state = {values:{}}
            handleChange = (event:React.ChangeEvent,name:string)=>{
                this.setState({
                   values:{...this.state.values,[name]:event.target.value}
                });
            }
            getFieldDecorator = (name:string,fieldOption:any)=>{
                return (fieldElement:any)=>{
                    let values:Objects = this.state.values;
                    let props:Objects = {
                        value:values[name]||'',
                        onChange:(event:React.ChangeEvent)=>this.handleChange(event,name)
                    }
                    return React.cloneElement(fieldElement, props);
                }
            }
            getFieldsValue = ()=>{
                return this.state.values;
            }
            render(){
                const props = {
                    form:{
                        getFieldDecorator:this.getFieldDecorator,
                        getFieldsValue:this.getFieldsValue
                    }
                };
                return <WrappedComponent {...props}/>;
             }
        }
        return Form;
    }
}
```

7. 表单校验

src\UserForm.tsx

```
import * as React from 'react'
import Form  from './antd/lib/form';
interface Props {
    form: any
}
class UserForm extends React.Component<Props> {
    handleSubmit = (event:React.FormEvent)=>{
        event.preventDefault();

        this.props.form.validateFields((error:any,values:any)=>{
            console.log(error);
            console.log(values);
        });
    }
    render() {
        let {getFieldDecorator} = this.props.form;
        return (
            <div>
                <Form onSubmit={this.handleSubmit}>
                    <Form.Item label="用户名">
                        {getFieldDecorator('username',{
                            rules:[{required:true,message:'用户名必须输入'}]
                        })(<input/>)}
                    Form.Item>
                    <Form.Item label="密码">
                        {getFieldDecorator('password',{
                            rules:[
                                {required:true,message:'密码必须输入'},
                                {min:3,message:'密码长度不能小于6位'},
                                {max:8,message:'密码长度不能大于8位'}
                            ]
                        })(<input/>)}
                    Form.Item>
                    <Form.Item>
                        <button>提交button>
                    Form.Item>
                Form>
            div>
        )
    }
}
export default Form.create()(UserForm);
```

src/antd/lib/form/create.tsx

```
import * as React from 'react';
interface FormProps {

}
interface Objects{
    [propName: string]: any;
}
interface FormState {
    values:Objects,
    errors:Objects
}
type ValidateCallback = (errors: any, values: V) => void;
export default function(){
    return function decorate(WrappedComponent:any) {
        class Form extends React.Component<FormProps,FormState>{
            state = {values:{},errors:{}}
            rules:Objects={}
            handleChange = (event:React.ChangeEvent,name:string)=>{
                this.setState({
                   values:{...this.state.values,[name]:event.target.value}
                },()=>this.validateFields([name]));
            }
            validateFields = (fields:Array|ValidateCallback,callback?:Array|ValidateCallback)=>{
                if(typeof fields == 'function'){
                    callback = fields;
                    fields = Object.keys(this.rules);
                }
                let errors:Objects = this.state.errors;
                (fields as Array).forEach((field:string)=>{
                    let rules = this.rules[field];
                    if(rules && rules.length>0){
                        let values:Objects = this.state.values;
                        let value = values[field];
                        let fieldErrors = rules.map((rule:Objects)=>{
                            if((rule.required && !value)||
                            (rule.min && value && value.length < rule.min)||
                                (rule.max && value && value.length > rule.max)
                            ){
                            return {field,message:rule.message};
                        }}).filter((item:any)=>item);
                        if(fieldErrors.length>0){
                            errors[field] = {errors:fieldErrors};
                        }else{
                            delete errors[field];
                        }
                    }
                });
                let error = Object.keys(errors).length>0?errors:null;
                this.setState({errors},()=>{
                    callback&& (callback as ValidateCallback)(error,this.state.values);
                });
            }
            getFieldDecorator = (name:string,fieldOption:any)=>{
                if(fieldOption.rules){
                    this.rules[name] = fieldOption.rules;
                }
                return (fieldElement:any)=>{
                    let values:Objects = this.state.values;
                    let props:Objects = {
                        value:values[name]||'',
                        onChange:(event:React.ChangeEvent)=>this.handleChange(event,name)
                    }
                    let errors:Objects = this.state.errors;
                    let fieldErrors = errors[name];
                    let messages = [];
                    if(fieldErrors && fieldErrors.errors.length>0){
                        props.style={border:'1px solid red'};
                        messages = fieldErrors.errors.map((item:any)=>item.message).map((message:string,index:number)=><p key={index} style={{color:'red'}}>
{message}p>);
                    }
                    let inputElement = React.cloneElement(fieldElement, props);
                    return <div>{inputElement} {messages.length>0&&messages}div>
                }
            }
            getFieldsValue = ()=>{
                return this.state.values;
            }
            render(){
                const props = {
                    form:{
                        getFieldDecorator:this.getFieldDecorator,
                        getFieldsValue:this.getFieldsValue,
                        validateFields:this.validateFields
                    }
                };
                return <WrappedComponent {...props}/>;
             }
        }
        return Form;
    }
}
```