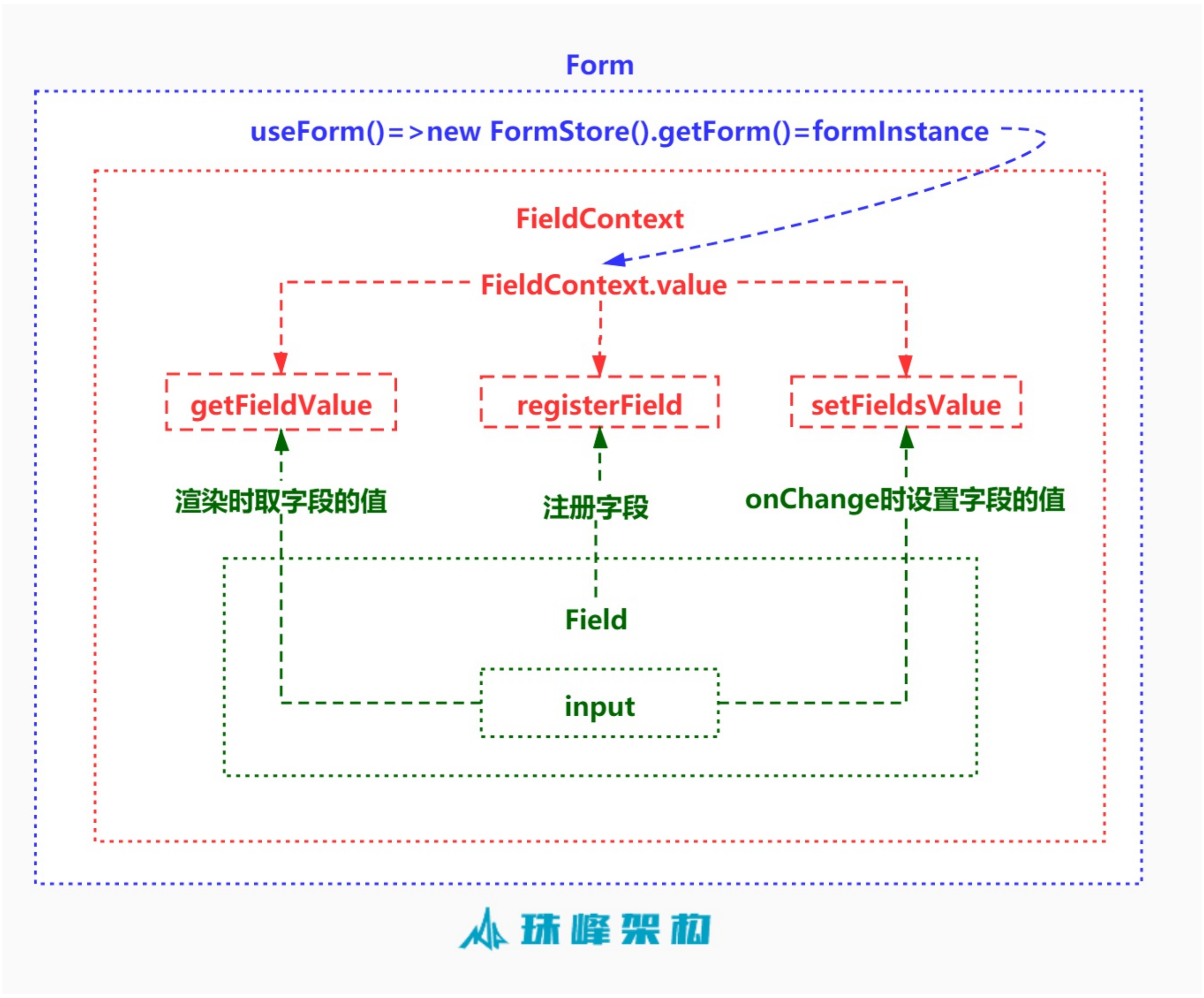
link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=82 sentences=249, words=1648

1.生成项目

- rc-field-form (https://www.npmjs.com/package/rc-field-form)
- async-validator (https://github.com/yiminghe/async-validator)

```
create-react-app zhufeng_antdesign-form
cd zhufeng_antdesign-form
yarn add rc-field-form
yarn start
```

2.使用Form



2.1 src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import Form, { Field } from 'rc-field-form';
ReactDOM.render(
  <Form
    initialValues={{username:'',password:''}}
    onFinish={values => {
      console.log('完成:', values);
    }}
  >
    <Field name="username">
      <input placeholder="用户名" />
    Field>
    <Field name="password">
      <input placeholder="密码" />
    Field>
    <button>提交button>
  Form>,
  document.getElementById('root')
);
```

3.实现双向数据绑定

3.1 src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';

import Form,{Field } from './rc-field-form';
ReactDOM.render(
  <Form
    initialValues={{username:'',password:''}}
    onFinish={values => {
      console.log('完成:', values);
    }}
  >
    <Field name="username">
      <input placeholder="用户名" />
    Field>
    <Field name="password">
      <input placeholder="密码" />
    Field>
    <button>提交button>
  Form>,
  document.getElementById('root')
);
```

3.2 rc-field-form\index.js

src\rc-field-form\index.js

```
import Form from "./Form";
import Field from "./Field";
import useForm from "./useForm";
export default Form;
export {
    Field,useForm
}
```

3.3 Form.js

src\rc-field-form\Form.js

```
import React from "react";
import useForm from "./useForm";
import FieldContext from "./FieldContext";

const Form = ({initialValues,onFinish,children}) => {
  const [formInstance] = useForm();
  formInstance.setCallbacks({
    onFinish
  });
  const mountRef = React.useRef(null);
  formInstance.setInitialValues(initialValues, !mountRef.current);
  if (!mountRef.current) {
    mountRef.current = true;
  }
  return (
    <form
      onSubmit={event => {
        event.preventDefault();
        event.stopPropagation();
        formInstance.submit();
      }}>
      <FieldContext.Provider value={formInstance}>
        {children}
      FieldContext.Provider>
    form>
  );
}
export default Form;
```

3.4 FieldContext.js

src\rc-field-form\FieldContext.js

```
import React from "react";

const warningFunc = () => {
    console.warn(false, '无法找到FormContext. 请确定你是在Form下面使用Field');
};

const Context = React.createContext({
  getFieldValue: warningFunc,
  getFieldsValue: warningFunc,
  setFieldsValue: warningFunc,
  submit: warningFunc,
});

export default Context;
```

3.5 Field.js

src\rc-field-form\Field.js

```
import React from "react";
import FieldContext from "./FieldContext";

class Field extends React.Component {
  static contextType = FieldContext;
  componentDidMount() {
    this.context.registerField(this);
  }

  onStoreChange = () => {
    this.forceUpdate();
  };

  getControlled = (childProps) => {
    const {name} = this.props;
    const {getFieldValue,setFieldsValue} = this.context;
    return {
      ...childProps,
      value: getFieldValue(name),
      onChange: event => {
        setFieldsValue({[name]: event.target.value});
      }
    };
  };

  render() {
    const { children } = this.props;
    const returnChildNode = React.cloneElement(children, this.getControlled(children.props));
    return returnChildNode;
  }
}
export default Field;
```

3.6 useForm.js

src\rc-field-form\useForm.js

```
import React from 'react';
class FormStore {
    store = {};
    fieldEntities = [];
    initialValues = {};
    callbacks = {};
    constructor(forceRootUpdate) {
        this.forceRootUpdate = forceRootUpdate;
    }
    getForm = () => ({
        getFieldValue: this.getFieldValue,
        getFieldsValue: this.getFieldsValue,
        setFieldsValue: this.setFieldsValue,
        setInitialValues: this.setInitialValues,
        setCallbacks: this.setCallbacks,
        registerField: this.registerField,
        submit: this.submit,
    });
    setInitialValues = (initialValues) => {
        this.store = { ...initialValues };
    };
    setCallbacks = (callbacks) => {
        this.callbacks = callbacks;
    };
    getFieldValue = (name) => {
        return this.store[name];
    };
    getFieldsValue = () => {
        return this.store;
    }
    registerField = (entity) => {
        this.fieldEntities.push(entity);
    };
    setFieldsValue = (store) => {
        this.store = { ...this.store, ...store };
        this.fieldEntities.forEach(({ onStoreChange }) => {
            onStoreChange();
        });
    };
    submit = () => {
        const { onFinish } = this.callbacks;
        if (onFinish) {
            onFinish(this.store);
        }
    };
}

export default function useForm(form) {
    const formRef = React.useRef();
    const [, forceUpdate] = React.useState({});
    if (!formRef.current) {
        if (form) {
            formRef.current = form;
        } else {
            const forceReRender = () => {
                forceUpdate({});
            };
            const formStore = new FormStore(forceReRender);
            formRef.current = formStore.getForm();
        }
    }
    return [formRef.current];
}
```

4.实现表单校验

4.1 src\index.js

src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
//import Form, { Field } from 'rc-field-form';
import Form,{Field } from './rc-field-form';
ReactDOM.render(
   {
      console.log('完成:', values);
    }}
+     onFinishFailed={(errorInfo)=>{
+       console.log('失败:',errorInfo);
+     }}
  >
+

+


    提交
  ,
  document.getElementById('root')
);
```

4.2 Form.js

src\rc-field-form\Form.js

```
import React from "react";
import useForm from "./useForm";
import FieldContext from "./FieldContext";

+const Form = ({initialValues,onFinish,onFinishFailed,children}) => {
  const [formInstance] = useForm();
  formInstance.setCallbacks({
    onFinish,
+     onFinishFailed
  });
  const mountRef = React.useRef(null);
  formInstance.setInitialValues(initialValues, !mountRef.current);
  if (!mountRef.current) {
    mountRef.current = true;
  }
  return (
     {
        event.preventDefault();
        event.stopPropagation();
        formInstance.submit();
      }}>

        {children}

  );
}

export default Form;
```

4.3 useForm.js

src\rc-field-form\useForm.js

```
import React from 'react';
import AsyncValidator from './async-validator';
class FormStore {
    store = {};
    fieldEntities = [];
    initialValues = {};
    callbacks = {};
    constructor(forceRootUpdate) {
        this.forceRootUpdate = forceRootUpdate;
    }
    getForm = () => ({
        getFieldValue: this.getFieldValue,
        getFieldsValue: this.getFieldsValue,
        setFieldsValue: this.setFieldsValue,
        setInitialValues: this.setInitialValues,
        setCallbacks: this.setCallbacks,
        registerField: this.registerField,
        submit: this.submit,
    });
    setInitialValues = (initialValues) => {
        this.store = { ...initialValues };
    };
    setCallbacks = (callbacks) => {
        this.callbacks = callbacks;
    };
    getFieldValue = (name) => {
        return this.store[name];
    };
    getFieldsValue = () => {
        return this.store;
    }
    registerField = (entity) => {
        this.fieldEntities.push(entity);
    };
    setFieldsValue = (store) => {
        this.store = { ...this.store, ...store };
        this.fieldEntities.forEach(({ onStoreChange }) => {
            onStoreChange();
        });
    };
+    submit = () => {
+      this.validateFields()
+      .then(values => {
+        const { onFinish } = this.callbacks;
+        if (onFinish) {
+            onFinish(values);
+        }
+      })
+      .catch(errorInfo => {
+        const { onFinishFailed } = this.callbacks;
+        if (onFinishFailed) {
+          onFinishFailed(errorInfo);
+        }
+      });
+    };
+    validateFields = () => {
+      let values = this.getFieldsValue();
+      let descriptor = this.fieldEntities.reduce((descriptor,entity)=>{
+        let rules = entity.props.rules;
+        if(rules && rules.length){
+            let config = rules.reduce((config,rule)=>{
+                config = {...config,...rule};
+                return config;
+            },{});
+            descriptor[entity.props.name]=config;
+        }
+        return descriptor;
+      },{});
+      return new AsyncValidator(descriptor).validate(values);
+    }
}

export default function useForm(form) {
    const formRef = React.useRef();
    const [, forceUpdate] = React.useState({});
    if (!formRef.current) {
        if (form) {
            formRef.current = form;
        } else {
            const forceReRender = () => {
                forceUpdate({});
            };
            const formStore = new FormStore(forceReRender);
            formRef.current = formStore.getForm();
        }
    }
    return [formRef.current];
}
```

4.4 async-validator.js

src\async-validator.js

```
class Schema {
    constructor(descriptor) {
        this.descriptor = descriptor;
    }
    validate(values) {
        return new Promise((resolve,reject) => {
            let errorFields = [];
            for (let name in this.descriptor) {
                let rules = this.descriptor[name];
                if (rules) {
                    let ruleKeys = Object.keys(rules);
                    let errors = [];
                    for(let i=0;ilet ruleKey = ruleKeys[i];
                        if (ruleKey === 'required') {
                            if (rules[ruleKey] && !values[name]) {
                                errors.push(`${name} is required`);
                            }
                        } else if (ruleKey === 'type') {
                            if (typeof values[name] !== rules[ruleKey]) {
                                errors.push(`${name} is not ${rules[ruleKey]}`);
                            }
                        }else if (ruleKey === 'min') {
                            if (values[name].length <  rules[ruleKey]) {
                                errors.push(`${name} must be at least ${rules[ruleKey]} characters` );
                            }
                        }
                    }
                    if(errors && errors.length){
                        errorFields.push({name,errors});
                    }
                }
            }
            if(errorFields && errorFields.length>0){
                reject({errorFields,values});
            }else{
                resolve(values);
            }
        });
    }
}

module.exports = Schema;
```

5.实现异步校验 [#](#)

5.1 src\index.js [#](#)

src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
//import Form, { Field } from 'rc-field-form';
import Form,{Field } from './rc-field-form';
+let uniqueValidator = (rule, value, callback, form) => {
+  return new Promise((resolve,reject)=>{
+    setTimeout(()=>{
+      if(value === 'zhufeng'){
+        resolve(`用户名已经被占用`);
+      }else{
+        resolve('');
+      }
+    },3000);
+  });
+}
ReactDOM.render(
   {
      console.log('成功:', values);
    }}
    onFinishFailed={(errorInfo)=>{
      console.log('失败:',errorInfo);
    }}
  >
+

    提交
  ,
  document.getElementById('root')
);
```

5.2 async-validator.js [#](#)

src\rc-field-form\async-validator.js

```
class Schema {
    constructor(descriptor) {
        this.descriptor = descriptor;
    }
    validate(values) {
+        return new Promise(async (resolve,reject) => {
            let errorFields = [];
            for (let name in this.descriptor) {
                let rules = this.descriptor[name];
                if (rules) {
                    let ruleKeys = Object.keys(rules);
                    let errors = [];
                    for(let i=0;i+                        }else if (ruleKey === 'validator') {
+                            let validator = rules[ruleKey];
+                            let result = await validator(rules[ruleKey], values[name]);
+                            if(result.length>0){
+                                errors.push(`${name}不符合自定义验证器的需求!`);
+                            }
+                        }
                    }
                    if(errors && errors.length){
                        errorFields.push({name,errors});
                    }
                }
            }
            if(errorFields && errorFields.length>0){
                reject({errorFields,values});
            }else{
                resolve(values);
            }
        });
    }
}
//export default Schema;
module.exports = Schema;
```

6.按需加载

6.1 生成项目

```
create-react-app zhufeng-antdesign-form
cd  zhufeng-antdesign-form
yarn add react-app-rewired customize-cra babel-plugin-import
npm start
```

6.1 使用antd

6.1.1 安装依赖

```
yarn add react-app-rewired customize-cra babel-plugin-import less less-loader
```

6.1.2 config-overrides.js

config-overrides.js

```
const {override,fixBabelImports, addLessLoader} = require('customize-cra');
module.exports = override(
    fixBabelImports('import',{
        libraryName:'antd',
        libraryDirectory:'es',
        style:true
    }),
    addLessLoader({
        lessOptions:{
            javascriptEnabled: true,
            modifyVars:{'@primary-color':'red'}
        }
    })
);
```

6.1.3 package.json

package.json

```
+  "scripts": {
+    "start": "react-app-rewired start",
+    "build": "react-app-rewired build",
+    "test": "react-app-rewired test",
+    "eject": "react-app-rewired eject"
+  }
```

6.1.4 src\index.js

src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import {Button} from 'antd';
ReactDOM.render(
  <Button type="primary">点击Button>,
  document.getElementById('root')
);
```