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=232 sentences=601, words=3824

1.formily

- formilyjs (https://formilyjs.org/zh-CN/guide)
- formily (https://github.com/alibaba/formily)
- 内容大纲
  - 实现 `@formily/reactive`核心
  - 实现 `@formily/reactive-react`核心
  - 实现 `@formily/core`核心
  - 实现 `@formily/react`核心
  - 实现 `@formily/antd`核心

1.1 安装

```
pnpm create vite
pnpm install @formily/reactive @formily/reactive-react @formily/core @formily/react @formily/antd antd moment less --save
```

1.2 vite.config.ts

vite.config.ts

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
import path from 'path'
export default defineConfig({
  plugins: [react()],
  resolve: {
    alias: [
      { find: /^~/, replacement: '' },
      { find: "@", replacement: path.resolve('src') }
    ]
  },
  css: {
    preprocessorOptions: {
      less: {
        javascriptEnabled: true,
      }
    }
  }
})
```

2. @formily/reactive

- 依赖@formily/reactive (https://reactive.formilyjs.org/zh-CN)响应式解决方案，构建响应式表单的领域模型实现精确渲染

2.1 observable

- 主要用于创建不同响应式行为的 `observable` 对象
- 一个 `observable`对象，字面意思是可订阅对象，我们通过创建一个可订阅对象，在每次操作该对象的属性数据的过程中，会自动通知订阅者
- @formily/reactive (https://reactive.formilyjs.org/zh-CN) 创建observable (https://reactive.formilyjs.org/zh-CN/api/observable) 对象主要是通过 `ES Proxy` 来创建的，它可以做到完美劫持数据操作

2.2 Reaction

- reaction (https://reactive.formilyjs.org/zh-CN/guide/concept#reaction)在响应式编程模型中，它就相当于是可订阅对象的订阅者
- 它接收一个 `tracker` 函数，这个函数在执行的时候，如果函数内部有对 `observable` 对象中的某个属性进行读操作会进行依赖收集，那当前 `reaction` 就会与该属性进行一个绑定(依赖追踪)，该属性在其它地方发生了写操作，就会触发 `tracker` 函数重复执行
- 从订阅到派发订阅，其实是一个封闭的循环状态机，每次 `tracker` 函数执行的时候都会重新收集依赖，依赖变化时又会重新触发 `tracker`执行

2.3 autorun

- autorun (https://reactive.formilyjs.org/zh-CN/api/autorun)可以创建一个自动执行的响应器
- 接收一个 `tracker` 函数，如果函数内部有消费 `observable` 数据，数据发生变化时， `tracker` 函数会重复执行

2.4 创建深度可观察对象

- observable (https://reactive.formilyjs.org/zh-CN/api/observable)主要用于创建不同响应式行为的 observable 对象，同时可以作为 annotation 给 define 用于标记响应式属性
- autorun (https://reactive.formilyjs.org/zh-CN/api/autorun)接收一个 tracker 函数，如果函数内部有消费 observable 数据，数据发生变化时，tracker 函数会重复执行
- reaction (https://reactive.formilyjs.org/zh-CN/api/reaction)接收一个 tracker 函数，与 callback 响应函数，如果 tracker 内部有消费 observable 数据，数据发生变化时，tracker 函数会重复执行，但是 callback 执行必须要求 tracker 函数返回值发生变化时才执行
- define (https://reactive.formilyjs.org/zh-CN/api/define)手动定义领域模型，可以指定具体属性的响应式行为，也可以指定某个方法为 batch 模式
- toJS (https://reactive.formilyjs.org/zh-CN/api/to-js)深度递归将 observable 对象转换成普通 JS 对象
- tracker (https://reactive.formilyjs.org/zh-CN/api/tracker)主要用于接入 React/Vue 的手动追踪依赖工具，在依赖发生变化时不会重复执行 tracker 函数，需要用户手动重复执行，只会触发 scheduler

2.4.1 src\main.jsx

src\main.jsx

```
import { observable, autorun } from '@/@formily/reactive'
const values = { username: 'zhufeng', home: { name: 'beijing' } }
const observableValues = observable(values)
console.log(observableValues);
console.log(observableValues.username);
console.log(observableValues.home);
console.log(observableValues.home);
```

2.4.2 reactive\index.jsx

src@formily\reactive\index.jsx

```
export * from './observable'
export * from './autorun'
```

2.4.3 observable.jsx

src@formily\reactive\observable.jsx

```
import { createObservable } from './internals';
export function observable(target) {
    return createObservable(null, null, target)
}
```

2.4.4 handlers.jsx

src@formily\reactive\handlers.jsx

```
import { isObservable } from './externals'
import { createObservable } from './internals'
import { RawProxy } from './environment'
export const baseHandlers = {
    get(target, key) {
        const result = target[key]
        const observableResult = RawProxy.get(result)
        if (observableResult) {
            return observableResult
        }
        if (!isObservable(result)) {
            return createObservable(target, key, result)
        }
        return result;
    },
    set(target, key, value) {
        const newValue = createObservable(target, key, value)
        target[key] = newValue
        return true;
    }
}
```

2.4.5 environment.jsx

src@formily\reactive\environment.jsx

```

export const RawProxy = new WeakMap()

export const ProxyRaw = new WeakMap()
```

2.4.6 checkers.jsx

src@formily\reactive\checkers.jsx

```
const toString = Object.prototype.toString
export const isPlainObj = (val) => toString.call(val) === '[object Object]'
export const isNormalType = (target) => {
    return isPlainObj(target)
}
```

2.4.7 internals.jsx

src@formily\reactive\internals.jsx

```
import { baseHandlers } from './handlers'
import { isNormalType } from './checkers';
import { ProxyRaw, RawProxy } from './environment';
export const createObservable = (target, key, value) => {
    if (typeof value !== 'object') return value;
    const raw = ProxyRaw.get(value)
    if (raw) {
        return value
    }
    if (isNormalType(value)) return createNormalProxy(value)
    return value
}
const createNormalProxy = (target) => {
    const proxy = new Proxy(target, baseHandlers)
    ProxyRaw.set(proxy, target)
    RawProxy.set(target, proxy)
    return proxy
}
```

2.4.8 externals.jsx

src@formily\reactive\externals.jsx

```
import { ProxyRaw } from './environment'
export const isObservable = (target) => {
    return ProxyRaw.has(target)
}
```

2.4.9 autorun.jsx

src@formily\reactive\autorun.jsx

```
export const autorun = (tracker) => {}
```

2.5 实现autorun

2.5.1 src\main.jsx

src\main.jsx

```
import { observable, autorun } from '@/@formily/reactive'
const values = { username: 'zhufeng', home: { name: 'beijing' } }
const observableValues = observable(values)
+autorun(() => {
+  console.log(observableValues.username);
+})
+observableValues.username = 'jiagou';
```

2.5.2 autorun.jsx

src@formily\reactive\autorun.jsx

```
import { ReactionStack } from './environment'
export const autorun = (tracker) => {
    const reaction = () => {
        ReactionStack.push(reaction)
        tracker()
        ReactionStack.pop()
    }
    reaction()
}
```

2.5.3 environment.jsx

src@formily\reactive\environment.jsx

```
//RawProxy.set(target, proxy) 普通对象=>代理对象
export const RawProxy = new WeakMap()
//ProxyRaw.set(proxy, target) 代理对象=>原生对象
export const ProxyRaw = new WeakMap()
+export const RawReactionsMap = new WeakMap()
+export const ReactionStack = []
```

2.5.4 handlers.jsx

src@formily\reactive\handlers.jsx

```
import { isObservable } from './externals'
import { createObservable } from './internals'
import { RawProxy } from './environment'
+import { bindTargetKeyWithCurrentReaction, runReactionsFromTargetKey } from './reaction'
export const baseHandlers = {
    get(target, key) {
        const result = target[key]
+       bindTargetKeyWithCurrentReaction({ target, key })
        const observableResult = RawProxy.get(result)
        if (observableResult) {
            return observableResult
        }
        if (!isObservable(result)) {
            return createObservable(target, key, result)
        }
        return result;
    },
    set(target, key, value) {
        const newValue = createObservable(target, key, value)
        target[key] = newValue
+       runReactionsFromTargetKey({ target, key })
        return true;
    }
}
```

2.5.5 reaction.jsx

src@formily\reactive\reaction.jsx

```
import { isFn } from './checkers'
import { ReactionStack, RawReactionsMap } from './environment'
const addRawReactionsMap = (target, key, reaction) => {
    const reactionsMap = RawReactionsMap.get(target)
    if (reactionsMap) {
        const reactionSet = reactionsMap.get(key)
        if (reactionSet) {
            reactionSet.add(reaction)
        } else {
            let reactionSet = new Set();
            reactionSet.add(reaction);
            reactionsMap.set(key, reactionSet);
        }
        return reactionsMap
    } else {
        let reactionSet = new Set();
        reactionSet.add(reaction);
        const reactionsMap = new Map([[key, reactionSet]])
        RawReactionsMap.set(target, reactionsMap)
        return reactionsMap
    }
}

export const bindTargetKeyWithCurrentReaction = (operation) => {
    let { key, target } = operation
    const current = ReactionStack[ReactionStack.length - 1]
    if (current) {
       addRawReactionsMap(target, key, current)
    }
}

export const runReactionsFromTargetKey = (operation) => {
    let { key, target } = operation
    runReactions(target, key)
}
const runReactions = (target, key) => {
    const reactions = getReactionsFromTargetKey(target, key)
    if(reactions){
        for (let reaction of reactions) {
            reaction();
        }
    }
}
const getReactionsFromTargetKey = (target, key) => {
    const reactionsMap = RawReactionsMap.get(target);
    if (reactionsMap) {
        return reactionsMap.get(key)
    }
}
```

2.5.6 checkers.jsx

src@formily\reactive\checkers.jsx

```
const toString = Object.prototype.toString
export const isPlainObj = (val) => toString.call(val)
export const isNormalType = (target) => {
    return isPlainObj(target)
}
+export const isFn = (val) => typeof val === 'function'
```

2.6 实现define

2.6.1 src\main.jsx

src\main.jsx

```
+import { observable, autorun, define } from '@/@formily/reactive'
+const form = {
+  values: { username: { value: 'zhufeng' } },
+  fields: { username: { name: '用户名' } }
+}
+define(form, {
+  values: observable,
+  fields: observable.shallow
+});
autorun(() => {
+ console.log(form.values, form.values.username, form.values.username.value);
+ console.log(form.fields, form.fields.username, form.fields.username.name);
})
+form.values.username.value = 'jiagou'
+form.fields.username.name = '密码'
```

2.6.2 reactive\index.jsx

src@formily\reactive\index.jsx

```
export * from './observable'
export * from './autorun'
+export * from './model'
```

2.6.3 model.jsx

src@formily\reactive\model.jsx

```
import { getObservableMaker } from './internals';
import { isObservable, isAnnotation } from './externals'
export function define(target, annotations) {
    if (isObservable(target)) return target
    for (const key in annotations) {
        const annotation = annotations[key]
        if (isAnnotation(annotation)) {
            getObservableMaker(annotation)({ target, key })
        }
    }
    return target
}
```

2.6.4 internals.jsx #

src@formily\reactive\internals.jsx

```
import { baseHandlers } from './handlers'
+import { isNormalType, isFn } from './checkers';
+import { ProxyRaw, RawProxy, MakeObservableSymbol, RawShallowProxy } from './environment';
+export const createObservable = (target, key, value, shallow) => {
    if (typeof value !== 'object') return value;
    const raw = ProxyRaw.get(value)
    if (raw) {
        return value
    }
+   if (target) {
+       const parentRaw = ProxyRaw.get(target) || target
+       const isShallowParent = RawShallowProxy.get(parentRaw)
+       if (isShallowParent) return value
+   }
+   if (shallow) return createShallowProxy(value)
    if (isNormalType(value)) return createNormalProxy(value)
    return value
}
+const createShallowProxy = (target) => {
+    if (isNormalType(target)) return createNormalProxy(target, true)
+    return target
+}
+const createNormalProxy = (target, shallow) => {
    const proxy = new Proxy(target, baseHandlers)
    ProxyRaw.set(proxy, target)
+   if (shallow) {
+       RawShallowProxy.set(target, proxy)
+   } else {
+       RawProxy.set(target, proxy)
+   }
    return proxy
}
+export const createAnnotation = (maker) => {
+    const annotation = (target) => {
+        return maker({ value: target })
+    }
+    if (isFn(maker)) {
+        annotation[MakeObservableSymbol] = maker
+    }
+    return annotation
+}
+export const getObservableMaker = (target) => {
+    if (target[MakeObservableSymbol]) {
+        if (!target[MakeObservableSymbol][MakeObservableSymbol]) {
+            return target[MakeObservableSymbol]
+        }
+        return getObservableMaker(target[MakeObservableSymbol])
+    }
+}
```

2.6.5 externals.jsx #

src@formily\reactive\externals.jsx

```
+import { ProxyRaw, MakeObservableSymbol } from './environment'
export const isObservable = (target) => {
    return ProxyRaw.has(target)
}
+export const isAnnotation = (target) => {
+    return target && target[MakeObservableSymbol]
+}
```

2.6.6 environment.jsx #

src@formily\reactive\environment.jsx

```
//RawProxy.set(target, proxy) 普通对象=>代理对象
export const RawProxy = new WeakMap()
//ProxyRaw.set(proxy, target) 代理对象=>原生对象
export const ProxyRaw = new WeakMap()
export const RawReactionsMap = new WeakMap()
+export const ReactionStack = []
+export const MakeObservableSymbol = Symbol('MakeObservableSymbol')
+//RawShallowProxy.set(target, proxy) 原生对象=>代理对象
+export const RawShallowProxy = new WeakMap()
```

2.6.7 observable.jsx #

src@formily\reactive\observable.jsx

```
import { createObservable } from './internals';
+import * as annotations from './annotations'
+import { MakeObservableSymbol } from './environment';
export function observable(target) {
    return createObservable(null, null, target)
}
+observable.shallow = annotations.shallow
+observable[MakeObservableSymbol] = annotations.observable
```

2.6.8 annotations\index.jsx

src@formily\reactive\annotations\index.jsx

```
export * from './observable'
export * from './shallow'
```

2.6.9 observable.jsx

src@formily\reactive\annotations\observable.jsx

```
import { createAnnotation, createObservable } from '../internals'
import { bindTargetKeyWithCurrentReaction, runReactionsFromTargetKey } from '../reaction';
export const observable = createAnnotation(
    ({ target, key, value }) => {
        const store = {
            value: createObservable(target, key, target[key]),
        }
        function get() {
            bindTargetKeyWithCurrentReaction({ target, key })
            return store.value
        }
        function set(value) {
            value = createObservable(target, key, value)
            store.value = value
            runReactionsFromTargetKey({ target, key })
        }
        Object.defineProperty(target, key, {
            set,
            get,
            enumerable: true,
            configurable: false
        })
        return store.value
    }
)
```

2.6.10 shallow.jsx

src@formily\reactive\annotations\shallow.jsx

```
import { createAnnotation, createObservable } from '../internals'
import { bindTargetKeyWithCurrentReaction, runReactionsFromTargetKey } from '../reaction';
export const shallow = createAnnotation(
    ({ target, key, value }) => {
        const store = {
            value: createObservable(target, key, target[key], true),
        }
        function get() {
            bindTargetKeyWithCurrentReaction({ target: target, key: key })
            return store.value
        }

        function set(value) {
            value = createObservable(target, key, target[key], true)
            store.value = value
            runReactionsFromTargetKey({ target, key })
        }
        if (target) {
            Object.defineProperty(target, key, {
                set,
                get,
                enumerable: true,
                configurable: false
            })
            return target
        }
        return store.value
    }
)
```

2.7 实现Tracker

2.7.1 src\main.jsx

src\main.jsx

```
import { observable, Tracker } from '@/@formily/reactive'
const values = { username: 'zhufeng', home: { name: 'beijing' } }
+const observableValues = observable(values)
+const tracker = new Tracker(() => {
+    console.log('forceUpate');
+})
+tracker.track(() => {
+    console.log(observableValues.username);
+})
+observableValues.username = 'jiagou';
```

2.7.2 tracker.jsx

src@formily\reactive\tracker.jsx

```
import { ReactionStack } from './environment'
export class Tracker {
    constructor(scheduler) {
        this.track.scheduler = scheduler;
    }
    track = (view) => {
        ReactionStack.push(this.track)
        return view();
    }
}
```

2.7.3 reactive\index.jsx

src@formily\reactive\index.jsx

```
export * from './observable'
export * from './autorun'
export * from './model'
+export * from './tracker'
```

2.7.4 reaction.jsx

src@formily\reactive\reaction.jsx

```
import { ReactionStack, RawReactionsMap } from './environment';
/**
 * 把某个对象的某个key和当前的reaction进行绑定
 * @param {*} operation {target,key}
 */
export const bindTargetKeyWithCurrentReaction = (operation) => {
    const { target, key } = operation;
    //最后一个Reaction就是currentReaction
    const currentReaction = ReactionStack[ReactionStack.length - 1];
    if (currentReaction) {
        addRawReactionsMap(target, key, currentReaction)
    }
}
const addRawReactionsMap = (target, key, reaction) => {
    //判断此target对象在RawReactionsMap里有没有值
    const reactionsMap = RawReactionsMap.get(target);
    if (reactionsMap) {
        const reactionSet = reactionsMap.get(key);
        if (reactionSet) {
            reactionSet.add(reaction);
        } else {
            let reactionSet = new Set();
            reactionSet.add(reaction);
            reactionsMap.set(key, reactionSet);
        }
        return reactionsMap;
    } else {
        //ArraySet 元素唯1的数组
        let reactionSet = new Set();//源码里作者自己封装了一个ArraySet
        reactionSet.add(reaction);
        const reactionsMap = new Map([[key, reactionSet]]);
        RawReactionsMap.set(target, reactionsMap);
        return reactionsMap;
    }
}

export const runReactionsFromTargetKey = (operation) => {
    const { target, key } = operation;
    runReactions(target, key);
}
function runReactions(target, key) {
    const reactions = getReactionsFromTargetKey(target, key);
    if(reactions){
      for (let reaction of reactions) {
        if (isFn(reaction.scheduler)) {
           reaction.scheduler(reaction)
        } else {
           reaction()
        }
      }
    }
}
const getReactionsFromTargetKey = (target, key) => {
    const reactionsMap = RawReactionsMap.get(target);
    if (reactionsMap) {
        return reactionsMap.get(key)
    }
}
```

3. @formily/reactive-react

- observer (https://reactive.formilyjs.org/zh-CN/api/react/observer)接收一个 Function RenderProps，只要在 Function 内部消费到的任何响应式数据，都会随数据变化而自动重新渲染，也更容易实现局部精确渲染 -在 React 中，observer (https://reactive.formilyjs.org/zh-CN/api/react/observer#observer-1)将 Function Component 变成 Reaction，每次视图重新渲染就会收集依赖，依赖更新会自动重新渲染

3.1 src\main.tsx

src\main.tsx

```
import React from 'react'
import { createRoot } from 'react-dom/client'
import Counter from './Counter';
createRoot(document.getElementById('root')).render(<Counter />);
```

3.2 src\Counter.jsx

src\Counter.jsx

```
import { observable } from '@formily/reactive'
import { observer } from '@formily/reactive-react'
const counter = observable({
    number: 1
});
const Counter = observer(() => {
    return (
        <div>
            <p>{counter.number}p>
            <button onClick={() => counter.number++}>+button>
        div>
    )
});
export default Counter;
```

3.3 reactive-react\index.tsx

src@formily\reactive-react\index.tsx

```
export * from './observer'
```

3.4 observer.jsx

src@formily\reactive-react\observer.jsx

```
import { useObserver } from './hooks/useObserver'
export function observer(component) {
    const wrappedComponent = (props) => {
        return useObserver(() => component(props))
    }
    return wrappedComponent;
}
```

3.5 useObserver.jsx

src@formily\reactive-react\hooks\useObserver.jsx

```
import { useState, useCallback, useRef } from 'react';
import { Tracker } from '@/@formily/reactive'
export const useObserver = (view) => {
    const [, setState] = useState([])
    const forceUpdate = useCallback(() => setState([]), [])
    const instRef = useRef(null)
    if (!instRef.current) {
        instRef.current = new Tracker(forceUpdate)
    }
    return instRef.current.track(view)
}
```

4. @formily/core

- @formily/core (https://core.formilyjs.org/)的核心意义是将领域模型从UI框架中抽离出来
- Formily内核其实是一个 `@formily/reactive` 领域模型
- 实际消费领域模型则主要是依赖 `@formily/reactive`的响应器机制做依赖追踪来消费
- 我们可以在响应器 `Reactions`中消费 `Form/Field/ArrayField/ObjectField/VoidField`模型中的任意属性，依赖的属性发生变化，响应器就会重复执行

4.1 src\main.jsx

src\main.jsx

```
import { createForm } from '@/@formily/core'
const form = createForm({
    values: {
        username: 'zhufeng'
    },
})
console.log(form);
const field = form.createField({ name: 'username', title: '用户名', value: 'zhufeng' });
console.log(field);
```

4.2 core\index.jsx

src@formily\core\index.jsx

```
export * from './shared/externals';
export * from './models';
```

4.3 externals.jsx

src@formily\core\shared\externals.jsx

```
import { FormPath } from '@/@formily/shared'
import { Form } from '../models'

const createForm = (options) => {
    return new Form(options)
}

export {
    FormPath,
    createForm
}
```

4.4 models\index.jsx

src@formily\core\models\index.jsx

```
export * from './Form'
export * from './Field'
```

4.5 Form.jsx

src@formily\core\models\Form.jsx

```
import { define, observable } from '@/@formily/reactive'
import { Field } from './Field'
import { FormPath } from '@/@formily/shared'
export class Form {
    values={}
    fields = {}
    constructor(props) {
        this.initialize(props)
        this.makeObservable()
        this.makeValues()
    }
    initialize(props) {
        this.props = { ...props }
    }
    makeObservable() {
        define(this, {
            values: observable,
            fields: observable.shallow
        })
    }
    makeValues() {
        this.values = this.props.values
    }
    createField(props) {
        const address = FormPath.parse().concat(props.name)
        new Field(address, props, this)
        return this.fields[address.entire]
    }
}
```

4.6 Field.jsx

src@formily\core\models\Field.jsx

```
import { define, observable } from '@/@formily/reactive'
export class Field {
    constructor(address, props, form) {
        this.props = { ...props };
        this.form = form;
        this.locate(address)
        this.initialize()
        this.makeObservable()
    }
    initialize() {
        this.value = this.props.value;
    }
    makeObservable() {
        define(this, {
            value: observable
        })
    }
    locate(address) {
        this.form.fields[address.entire] = this
    }
}
```

4.7 path\index.jsx

src@formily\path\index.jsx

```
const parse = (pattern) => {
    if (!pattern) {
        return {
            entire: '',
            segments: []
        }
    }
    return {
        entire: pattern,
        segments: pattern.split('.')
    }
}
export class Path {
    constructor(input = '') {
        const { segments, entire } = parse(input)
        this.entire = entire
        this.segments = segments
    }
    static parse() {
        return new Path();
    }
    concat = (...args) => {
        const path = new Path('')
        path.segments = this.segments.concat(...args)
        path.entire = path.segments.join('.')
        return path
    }
}
```

src@formily\shared\index.jsx

```
export * from './path'
```

4.9 path.jsx

src@formily\shared\path.jsx

```
import { Path as FormPath } from '@/@formily/path'
export { FormPath }
```

5.@formily/antd

5.1 src\main.jsx

src\main.jsx

```
import React from "react";
import ReactDOM from "react-dom/client";
import { createForm } from "@/@formily/core";
import { FormProvider, Field } from "@/@formily/react";
import { FormItem, Input } from "@/@formily/antd";
const form = createForm();
const App = () => {
    return (
        <FormProvider form={form}>
            <Field
                name="username"
                title="用户名"
                value="jiagou"
                decorator={[FormItem]}
                component={[Input]}
            />
            <button onClick={() => {
                form.submit(console.log)
            }}>提交button>
        FormProvider>
    )
};
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<App />);
```

5.2 Form.jsx

src@formily\core\models\Form.jsx

```
import { define, observable } from '@/@formily/reactive'
import { Field } from './Field'
import { FormPath } from '@/@formily/shared'
+import { batchSubmit } from '../shared/internals'
export class Form {
    values = {}
    fields = {}
    constructor(props) {
        this.initialize(props)
        this.makeObservable()
        this.makeValues()
    }
    initialize(props) {
        this.props = { ...props }
    }
    makeObservable() {
        define(this, {
            values: observable,
            fields: observable.shallow
        })
    }
    makeValues() {
+       this.values = Object.assign({}, this.props.values);
    }
    createField(props) {
        const address = FormPath.parse().concat(props.name)
        new Field(address, props, this)
        return this.fields[address.entire]
    }
+   setValuesIn = (pattern, value) => {
+       this.values[pattern.entire] = value;
+   }
+   getValuesIn = (pattern) => {
+       return this.values[pattern.entire];
+   }
+   submit = (onSubmit) => {
+       return batchSubmit(this, onSubmit)
+   }
}
```

5.3 Field.jsx

src@formily\core\models\Field.jsx

```
import { define, observable } from '@/@formily/reactive'
export class Field {
    constructor(address, props, form) {
        this.props = { ...props };
        this.form = form;
        this.locate(address)
        this.initialize()
        this.makeObservable()
    }
    initialize() {
        this.value = this.props.value;
+       this.decorator = this.props.decorator
+       this.component = this.props.component
    }
    makeObservable() {
        define(this, {
            value: observable
        })
    }
    locate(address) {
        this.form.fields[address.entire] = this
+       this.path = address;
    }
+   get value() {
+       return this.form.getValuesIn(this.path)
+   }
+   set value(value) {
+       this.form.setValuesIn(this.path, value)
+   }
+   get decorator() {
+       return [this.decoratorType]
+   }
+   set decorator(value) {
+       this.decoratorType = value[0]
+   }
+   get component() {
+       return [this.componentType]
+   }
+   set component(value) {
+       this.componentType = value[0]
+   }
+   onInput = (e) => {
+       const newValue = e.target.value;
+       this.value = newValue;
+       this.form.values[this.props.name] = newValue;
+   };
}
```

5.4 externals.jsx

src@formily\reactive\externals.jsx

```
import { ProxyRaw, MakeObservableSymbol } from './environment'
+import { isPlainObj } from './checkers';
export const isObservable = (target) => {
    return ProxyRaw.has(target)
}
export const isAnnotation = (target) => {
    return target && target[MakeObservableSymbol]
}
+export const toJS = (values) => {
+    const visited = new Set()
+    const _toJS = (values) => {
+        if (visited.has(values)) {
+            return values
+        }
+        if (isPlainObj(values)) {
+            visited.add(values)
+            const res = {}
+            for (const key in values) {
+                res[key] = _toJS(values[key])
+            }
+            return res
+        }
+        return values
+    }
+    return _toJS(values)
+}
```

5.5 reactive\index.jsx

src@formily\reactive\index.jsx

```
export * from './observable'
export * from './autorun'
export * from './model'
export * from './tracker'
+export * from './externals'
```

5.6 internals.jsx

src@formily\core\shared\internals.jsx

```
import { toJS } from '@/@formily/reactive'
export const batchSubmit = (target, onSubmit) => {
    onSubmit(toJS(target.values))
}
```

5.7 antd\index.jsx

src@formily\antd\index.jsx

```
export * from './form-item'
export * from './input'
```

5.8 form-item\index.jsx

src@formily\antd\form-item\index.jsx

```
import { connect, mapProps } from '@/@formily/react'
export const BaseItem = ({ children, label }) => {
    return (
        <div>
            <span>{label}span>
            {children}
        div>
    )
}

export const FormItem = connect(
    BaseItem,
    mapProps((props, field) => {
        return { label: field.props.title }
    })
)

export default FormItem
```

5.9 input\index.jsx

src@formily\antd\input\index.jsx

```
import { connect, mapProps } from '@/@formily/react'
import { Input as AntdInput } from 'antd'
export const Input = connect(
    AntdInput,
    mapProps((props) => {
        return { ...props }
    })
)
export default Input
```

5.10 react\index.jsx

src@formily\react\index.jsx

```
export * from './components'
export * from './hooks'
export * from './shared'
```

5.11 Field.jsx

src@formily\react\components\Field.jsx

```
import React from 'react'
import { useForm } from '../hooks'
import { ReactiveField } from './ReactiveField'
import { FieldContext } from '../shared'
export const Field = (props) => {
  const form = useForm()
  const field = form.createField(props)
  return (
    <FieldContext.Provider value={field}>
      <ReactiveField field={field}>{props.children}ReactiveField>
    FieldContext.Provider>
  )
}
```

5.12 FormProvider.jsx

src@formily\react\components\FormProvider.jsx

```
import React from 'react'
import { FormContext } from '../shared'
export const FormProvider = (props) => {
  const form = props.form
  return (
    <FormContext.Provider value={form}>{props.children}FormContext.Provider>
  )
}
```

5.13 components\index.jsx

src@formily\react\components\index.jsx

```
export * from './FormProvider'
export * from './Field'
```

5.14 ReactiveField.jsx

src@formily\react\components\ReactiveField.jsx

```
import React from 'react';
import { observer } from '@/@formily/reactive-react'
const ReactiveInternal = (props) => {
    const field = props.field
    const renderDecorator = (children) => {
        return React.createElement(
            field.decoratorType,
            {},
            children
        )
    }
    const renderComponent = () => {
        const value = field.value;
        const onChange = (...args) => {
            field.onInput(...args)
        }
        return React.createElement(
            field.componentType,
            {
                value,
                onChange
            }
        )
    }
    return renderDecorator(renderComponent())
}
export const ReactiveField = observer(ReactiveInternal)
```

5.15 hooks\index.jsx

src@formily\react\hooks\index.jsx

```
export * from './useForm'
export * from './useField'
```

5.16 useField.jsx

src@formily\react\hooks\useField.jsx

```
import { useContext } from 'react'
import { FieldContext } from '../shared'

export const useField = () => {
    return useContext(FieldContext)
}
```

5.17 useForm.jsx

src@formily\react\hooks\useForm.jsx

```
import { useContext } from 'react'
import { FormContext } from '../shared'

export const useForm = () => {
  return useContext(FormContext)
}
```

5.18 connect.jsx

src@formily\react\shared\connect.jsx

```
import React from 'react';
import { observer } from '@/@formily/reactive-react';
import { useField } from '../hooks'
export function mapProps(...args) {
    return (target) => {
        return observer(
            (props) => {
                const field = useField()
                const results = args.reduce(
                    (props, mapper) => {
                        return Object.assign(props, mapper(props, field))
                    },
                    { ...props }
                )
                return React.createElement(target, results)
            }
        )
    }
}
export function connect(target, ...args) {
    const Target = args.reduce((target, mapper) => {
        return mapper(target)
    }, target)
    return (props) => {
        return React.createElement(Target, { ...props })
    }
}
```

5.19 context.jsx

src@formily\react\shared\context.jsx

```
import { createContext } from 'react'
export const FormContext = createContext(null)
export const FieldContext = createContext(null)
```

src@formily\react\shared\index.jsx

```
export * from './context'
export * from './connect'
```

6.字段验证