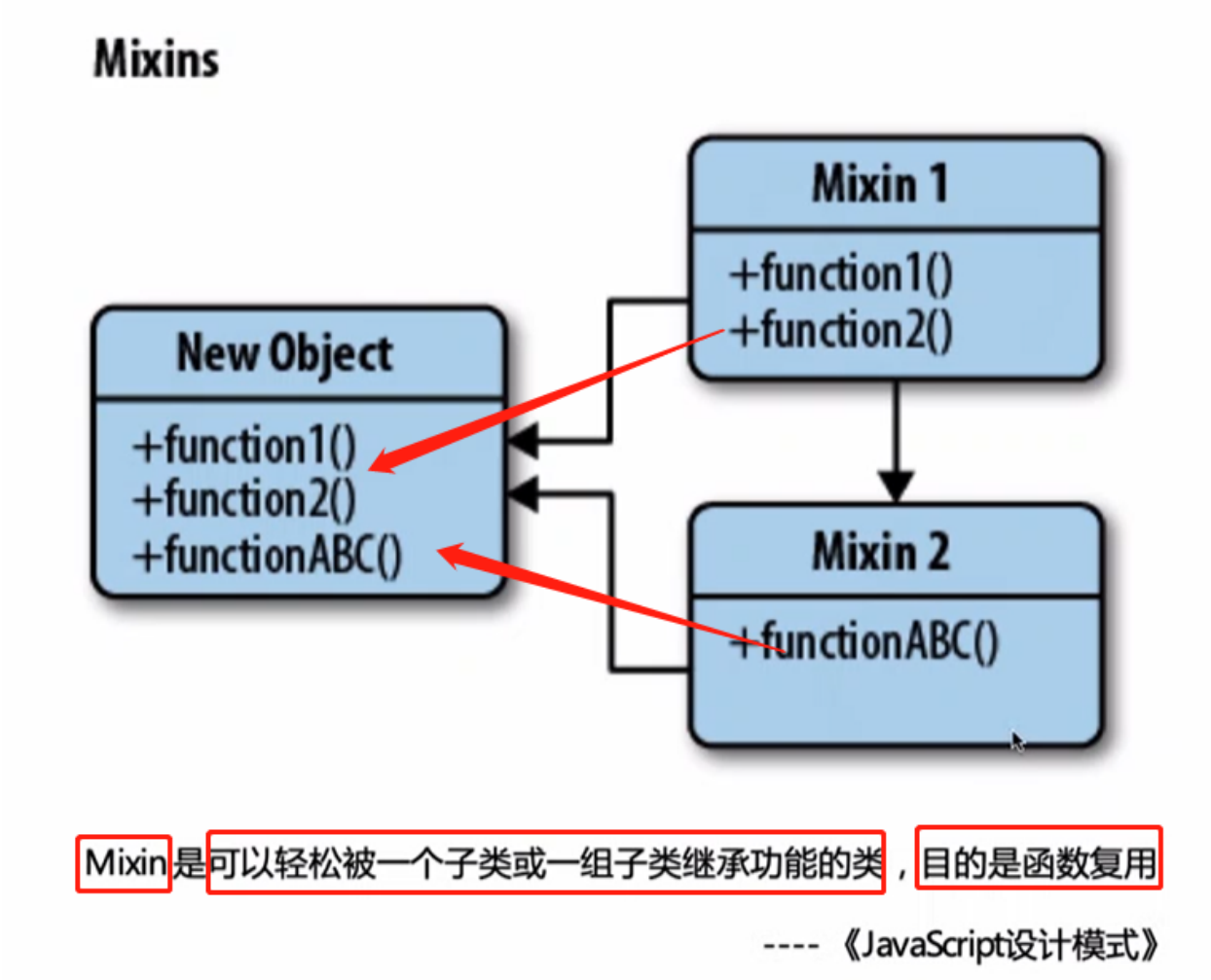
课程目标

Mixin模式



全局Mixins

使用Vue.mixin(mixin)，但是全局注入的ximin，会影响所有创建的Vue实例。(慎用！！！)

```js
const mixin = {
  created: function() {
    console.log("mixin created");
  },
  methods: {
    foo: function() {
      console.log("foo");
    },
    conflicting: function() {
      console.log("from mixin");
    }
  }
};
Vue.mixin(mixin);
```

```js
export default {
  created() {
    console.log("component created");

    this.conflicting();
  },
  methods: {
    conflicting: function() {
      console.log("from components");
    }
  }
};
```

- 同名钩子函数将合并为一个数组，混入对象的钩子将在组件自身钩子之前调用。
- 二者的 methods、components 和 directives，将被合并为同一个对象。若对象键名冲突时，取组件对象的键值对。

可局部混入

插件

Vue.use(plugin)

@/src/main.js

```js
Vue.use(Vuex);
Vue.use(VueRouter);
Vue.use(ElementUI);
```

```ts
type PluginInstallFunction = (app: App, ...options: any[]) => any

export type Plugin =
  | PluginInstallFunction & { install?: PluginInstallFunction }
  | {
      install: PluginInstallFunction
    }
```

Vue.use 接收一个 函数 或者 提供install 方法的对象 作为参数

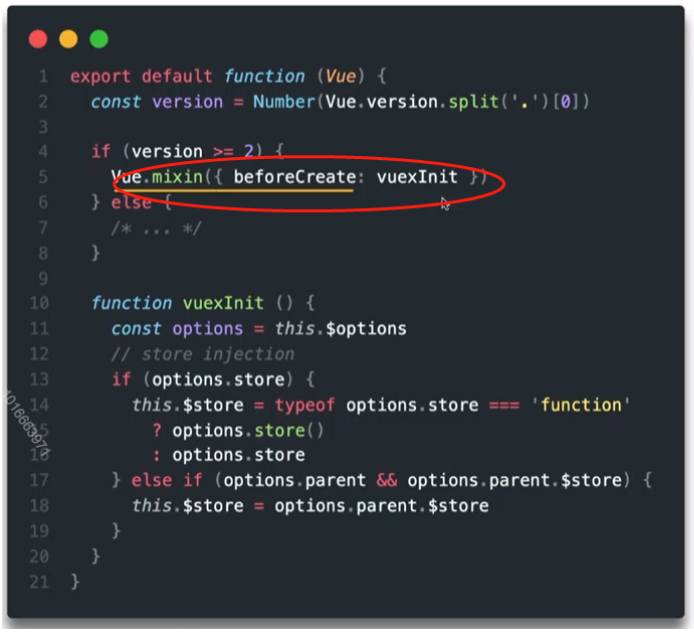
https://cn.vuejs.org/v2/api/#Vue-use

我们可以看vuex，vuex export的是一个含有install方法的对象：下面是vuex源码导出。

```ts
import { Store, install } from './store'
import { mapState, mapMutations, mapGetters, mapActions,
 createNamespacedHelpers } from './helpers'

export default {
  Store,
  install,
  version: '__VERSION__',
  mapState,
  mapMutations,
  mapGetters,
  mapActions,
  createNamespacedHelpers
}
```

install 方法 我们可以去看一下install的方法



```
export default function (Vue) {
  const version = Number(Vue.version.split('.')[0])

  if (version >= 2) {
    Vue.mixin({ beforeCreate: vuexInit })
  } else {
    /* ... */
  }

  function vuexInit () {
    const options = this.$options
    // store injection
    if (options.store) {
      this.$store = typeof options.store === 'function'
        ? options.store()
        : options.store
    } else if (options.parent && options.parent.$store) {
      this.$store = options.parent.$store
    }
  }
}
```

- beforeCreate 钩子函数在组件最初初始化的时候被调用。
- 且 插件 的 beforeCreate 钩子函数也会在具体组件的 beforeCreate 钩子函数前触发。

实践

实现一个vue插件，使得

- 能够通过Vue.use(这个插件);
- 并且可以在任意vue组件中 使用 this.$notify() 来在页面上生成一个提示框;
- 且this.$notify可以传入自定义模板作为参数并显示在提示框中。

Vue.use(接收的是一个的有install方法的对象，或者一个函数)

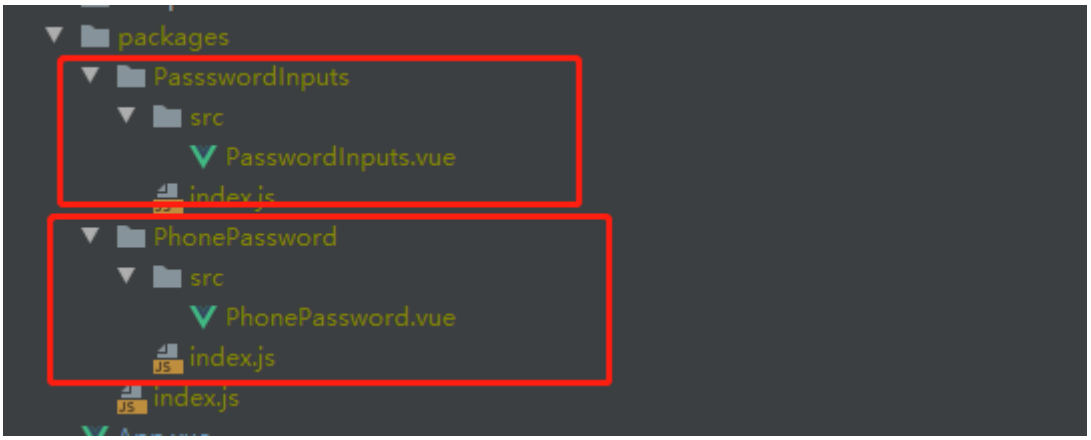
插件发布到npm

流程

1. 使用vue-cli3搭建项目，组件库文件夹存放位置
2. 全局注册组件库
3. 修改编辑package.json，准备发布npm
4. 注册并登录npm，发布

组件库存放和编辑

1. 在src下面新建plugins/文件夹、packages/index.js文件（用于注册所有组件库）、packages/PasswordInputs/index.js文件（对外暴露组件）



2.packages/PasswordInputs/index.js文件(这里只是举例说明一个，多个的话，需要再每个组件里面都新建index.js,以便按需引入，这里是只有两个)

```
//导入组件PasswordInputs
import PasswordInputs from "./src/PasswordInputs";

//局部引入组件 添加install

PasswordInputs.install =(Vue)=>{
 Vue.component(PasswordInputs.name,PasswordInputs)
}
//导出组件
 export default PasswordInputs
```

```
//导入组件PhonePassword
import PhonePassword from "./src/PhonePassword";

//局部引入组件 添加install

PhonePassword.install =(Vue)=>{
 Vue.component(PhonePassword.name,PhonePassword)
}

 //导出组件

 export default PhonePassword
```

3.packages/index.js 用于全局

```
//组件库出口 对外暴露的
import PasswordInputs from "./PasssswordInputs";
import PhonePassword from "./PhonePassword";


//存储所有组件
const components = [PasswordInputs,PhonePassword]


 //如果使用use的话 所有的组件都会被注册
 const install = (Vue)=>{
  // 判断是否安装
  if (install.installed) return
  components.map(item=>Vue.component(item.name,item))
 }


  // 判断是否是直接引入文件 以CDN的方式引入
  if (typeof window !== 'undefined' && window.Vue) {
  install(window.Vue)
  }

  //导出的组件,必须含有install
 export default {
  install,
  PasswordInputs,
  PhonePassword
 }

```

全局注入组件库作为demo

```
import Vue from 'vue'
import App from './App.vue'

//可以试一下打包后的全局引入
// import password from "../lib/passwordComponent.umd.min";
// import '../lib/passwordComponent.css'

//打包前，passsword可以说是一个UI组件库，
```

```
//包含PasswordInputs和PhonePassword组件
import password from './packages/index'

Vue.use(password)
Vue.config.productionTip = false

new Vue({
 render: h => h(App),
}).$mount('#app')
```

修改packagejson，准备发布npm

（1）在pakage.json文件中的scripts添加lib，用于编译为库的命令

--target 构建目标

--name 名称

--dest 输出目录

[entry] 入口文件，默认为 `src/App.vue`。这里我们指定编译 src/packages/ 组件库目录。

```
{
 "name": "vue-password-component",
 "version": "0.1.0",
 "private": false,
 //这里一定要在main 中指定打包lib的js
 // 不然用户在全局引入的时候，import的组件内容是空的
 "main": "lib/passwordComponent.umd.min.js",
 "scripts": {
 "serve": "vue-cli-service serve",
 "build": "vue-cli-service build",
 //这里相关配置
 "lib": "vue-cli-service build --target lib --name passwordComponent --dest lib src/packages/index.js ",
 "lint": "vue-cli-service lint"
 }
 }
```

（2）相关pakage.json配置

name：组件库名称，在npm组件中确保名称唯一，否则不能注册成功

version：版本

description： 组件库描述

main： 入口文件，编译后的包文件(编译后的js)

style：设置路径样式(编译后样式)

keyword：关键字 用户可以根据关键字搜索出组件库

private： 是否私有，为false才能发布到npm

license：开源协议，可以为MIT

（3）执行编译组件库

```
npm run lib
```

注册登录npm,发布

如果配置了淘宝镜像，先设置回npm镜像：

```
npm config set registry http://registry.npmjs.org
```

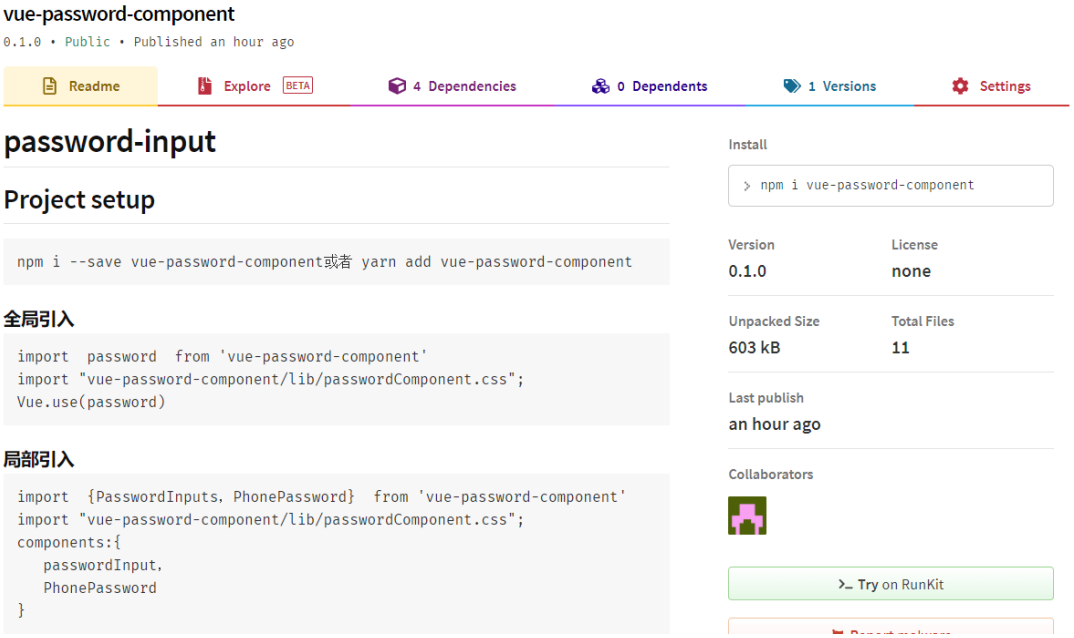
1. 登录

```
npm login
```

2. 发布到npm

```
npm publish
```

3. 发布后可以npm上查询



4. 下载相关组件(可全局 可局部 可CDN)

全局方式

```
<PasswordInputs ref="inputs"
                :pwdTexts.sync="pwdTexts"
                :currentFocusIndex="currentFocusIndex"
                @pwdsChange="pwdsChange"
                @currentFocusIndexChange="currentFocusIndexChange" />

<PhonePassword  :show="ConfirmShowKeyboard"
                :fatherNum="ConfirmValue"
                @numberEvent="ConfirmOnInput"
                @deleteEvent="ConfirmOnDelete" />
```

按需引入

```
<template>
  <div id="app">
      <Select v-model="selected"   :options="options" :multiple="true" />
      <PasswordInputs ref="inputs"
                      :pwdTexts="pwdTexts"
                      :currentFocusIndex="currentFocusIndex"
                      @pwdsChange="pwdsChange"
                      @currentFocusIndexChange="currentFocusIndexChange" />
      <p>{{`输入的密码${ConfirmValue}`}}</p>
      <PhonePassword  :show="true"
                      :fatherNum="ConfirmValue"
                      @numberEvent="ConfirmOnInput"
                      @deleteEvent="ConfirmOnDelete" />
  </div>
</template>

<script>
import  password from 'vue-password-component'
import 'vue-password-component/lib/passwordComponent.css'
export default {
  name: 'App',
  provide:()=>({...}),
  components:{
    PasswordInputs:password.PasswordInputs,
    PhonePassword:password.PhonePassword
  },
  data(){...},
  created() {
    this.$notify('mmp')
  }
```

按需引入，这里为什么需要不能类似Element import {} 引入呢？因为我们自定义的组件是以export defaut {} 导出的 只能以一个变量去接收 这里我们还要说明与export和exprot default

效果：


请选择数据
输入的密码
1
2
3
4
5
6
7
8
9
关闭
0
删除