饿了么基于Vue 2.0的通用组件库开发之路

原创 2016-11-09 Element 开发团队 前端之巅

本文为2016年11月9日，『前端之巅』微信群在线分享活动总结整理而成，转载请在文章开头处注明来自『前端之巅』公众号。加群学习请关注『前端之巅』公众号并发送“加群”。

Element 是由饿了么UED设计、饿了么大前端开发的一套基于 Vue 2.0 的桌面端组件库。今天我们要分享的就是开发 Element 的一些心得。

一、设计目的

大部分项目起源都是源于业务方的需求，Element 也是一样。随着公司的业务发展，内部开始衍生出很多后台系统，UED 部门也接到越来越多的设计需求。分析这整个过程，我们发现如下问题：

- 日渐增多的后台产品设计需求
- 设计资源有限，没办法支持所有业务线
- 公司内部诸多后台产品使用体验不一致

于是我们决定：

- 设计一套后台支撑框架，提升后台系统的可用性和一致性
- 套用此框架，即使没有设计师参与，也能让产品或开发设计出一套好用的后台系统

二、设计阶段

下面简单说一下设计 Element 经历的几个阶段。

了解业务并熟悉公司内各后台产品，寻找业务上的共性问题

设计的目的是为了业务服务。第一步我们从内部系统开始入手，了解公司内部在使用的各种后

台系统，将其组件抽象剥离，寻找共性特征。

专注业务组件设计

总结了公司不同系统不同组件的使用情况后，我们打算从业务组件入手，因为这部份是由公司特殊需求衍生的解决方案。解决了这些棘手的问题，也能给其他后台产品带来好的设计引导。

寻求开发支持

到这一步，我们开始寻找公司内部的开发团队，并在这时才得知不同团队里使用着不同的前端框架，有 Vue、React、Angular 等。

与大前端合作

大前端作为独立的前端团队，有能力开发底层的工具去服务不同业务，并且 Vue 也是一套年轻且发展方向很好的一个技术栈。UED 与大前端的合作一拍即合。

方向转变，专注于基础组件

跟大前端接触后，才发现最开始的方向并不正确，因为业务变化过快，即使有通用的业务组件，也很难跟上需求的变化，而基础组件才是所有开发团队都需要的通用组件。这时候我们开始把方向调整为基础组件的设计。

组件交互完成，进行视觉封装，并搭建主体网站

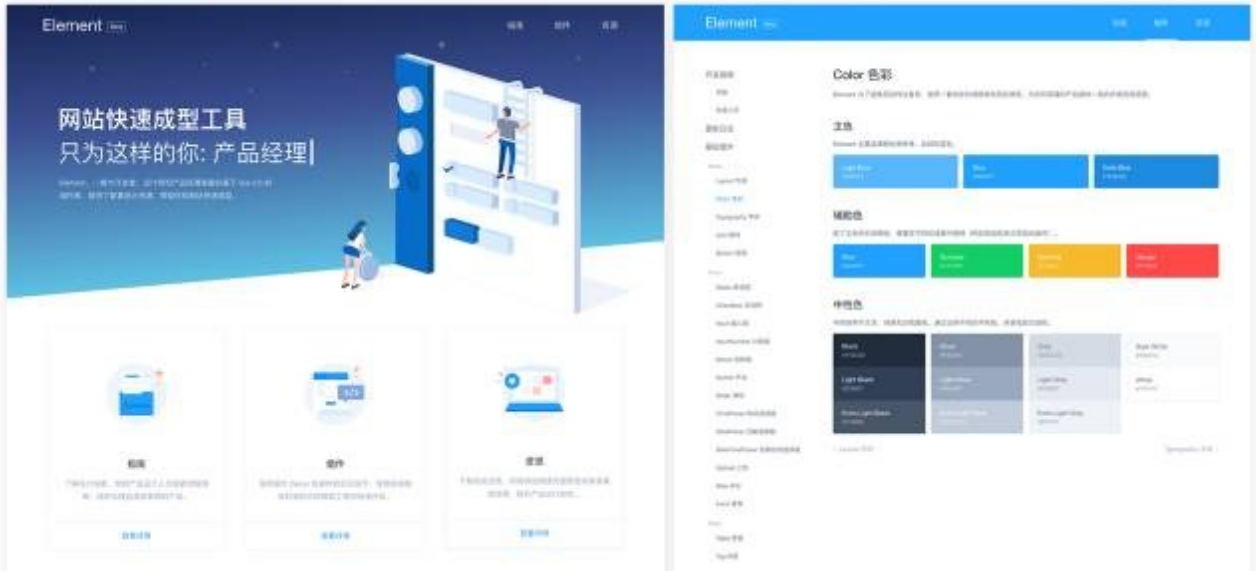
前期的设计工作主要是由交互设计师进行设计，等确认完所有组件的功能和交互后，开始进行视觉阶段，这中间包括制定颜色、字体等各类规范，也同时进行主体网站的设计。

> 输出 UI Kit 文件，统一设计规范

> 第一版网站设计，此处的「特殊组件」即业务组件。

网站二次设计

第一版网站上线后视觉效果并不好，我们内部进行了调整，再次上线后就是大家现在看到的样子。



设计过程简单来说就经历了这几个阶段，如还有问题可以继续交流，下面进入开发阶段。

三、开发目的

- 后台系统缺乏一套完整的基础组件库
- Vue 在公司内部是一个比较年轻的技术栈，希望做一些基础设施的建设
- 提升公司在技术社区的影响力

四、开发流程

进入开发阶段后，在总体架构方面我们做了一些尝试，下面按照时间顺序分享给大家：

1.如何与设计师进行配合

经过项目初期开发和设计的磨合，我们提炼了一套组件开发流程：

a. 根据交互稿和视觉稿进行开发，期间与设计师保持沟通

b. 开发完成后自测，之后提交设计师验收

c. 设计师提出修改意见，根据意见进行修改

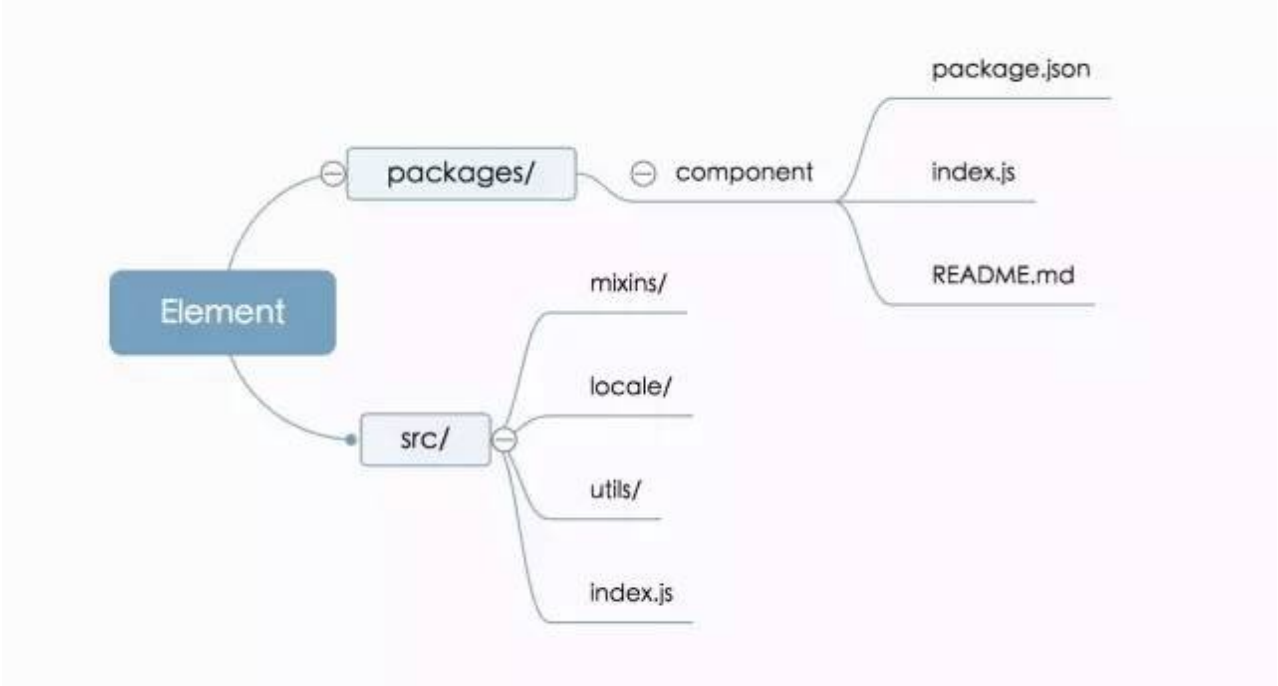
d. 完成组件开发，为网站编写例子和文档

2.如何管理多组件项目

在开发之初，我们就在思考如何降低组件的耦合度，确保组件可以独立工作。这样的目的是可以保证组件可以依赖其他组件、让用户只加载其中几个组件甚至在安装时只安装需要的组件。最先想到的做法是一个组件单独一个仓库，而组件库项目就是把组件作为依赖引入。

但是由于人手不足，这样的机制导致开发太耗时间，每个组件都需要单独维护和打包，同时还要维护组件库项目的各依赖的版本号。我们只能另寻方案。后来参考了 babel 项目的管理方式：所有子项目放在 `packages/` 目录里，一个子项目可以当作一个独立的仓库。通过 lerna 来管理子项目的依赖和发布。

结合自身项目的特点以及 babel 的这套机制，我们重构了目录结构：组件可单独作为一个项目放在 `packages/` ，共用函数放在 `src/` 里。最后的打包结果会将整个组件打包成一个文件、组件分别打包成独立文件，同时发布时还将发布组件库和独立组件，满足不同用户的使用需求。



3.如何解决自定义主题

开发一套组件库就离不开定制主题的需求。类名要足够友好，尽量避免存在样式层级嵌套，这样在直接覆盖样式或者单独写一套主题都会方便许多。所以我们采用 BEM 的方式管理类名，同时尽可能将属性值用变量代替，维护一份变量文件便于直接修改变量就能定制一套主题。

考虑到不同用户的使用习惯，我们没有选用 Less 或 Sass 之类的有各自风格的预处理器，而是选用了更接近未来标准的 CSS4 风格的语法，用 PostCSS 和整合了 postcss-bem 和 postcss-cssnext 等插件的 postcss-salad 开发。

为了降低用户自定义主题的上手成本，我们还提供了命令行工具指导用户快速自定义一套主题。

4.如何提供一份直观的文档

文档不仅是让用户看起来直观，也要让编写者写起来直观。所以最简单的方式是用 Markdown 写文档。但是就会产生另一个问题：如何在文档里写可运行的示例？常规的做法是把文档写在 Vue 文件里，这样就可以在里面调用其他组件，但是这样就违背了写「直观」文档的初衷。

经过几番尝试，结合 Vue 的特点。我们写了一套处理 Markdown 文件的 webpack loader，可以将 Markdown 转成 Vue 文件，不仅降低了文档的维护成本，同时也将文档里运行组件示例变成可能。

5.多语言官网如何配置和管理

Element 在立项之初其实并没有考虑国际化的问题。项目开源之后，我们陆续收到了一些外国开发者的反馈，希望能够增加英文文档。不久之后，国内的一个翻译团队主动联系到了我们，

为 Element 贡献了整套英文文档。

有了英文文档就需要有英文网站，这就需要对官网的现有结构进行修改和升级；同时为了面向未来，需要官网能够兼容除英语外的其他多语言。为此我们做了以下工作：

（1）路由

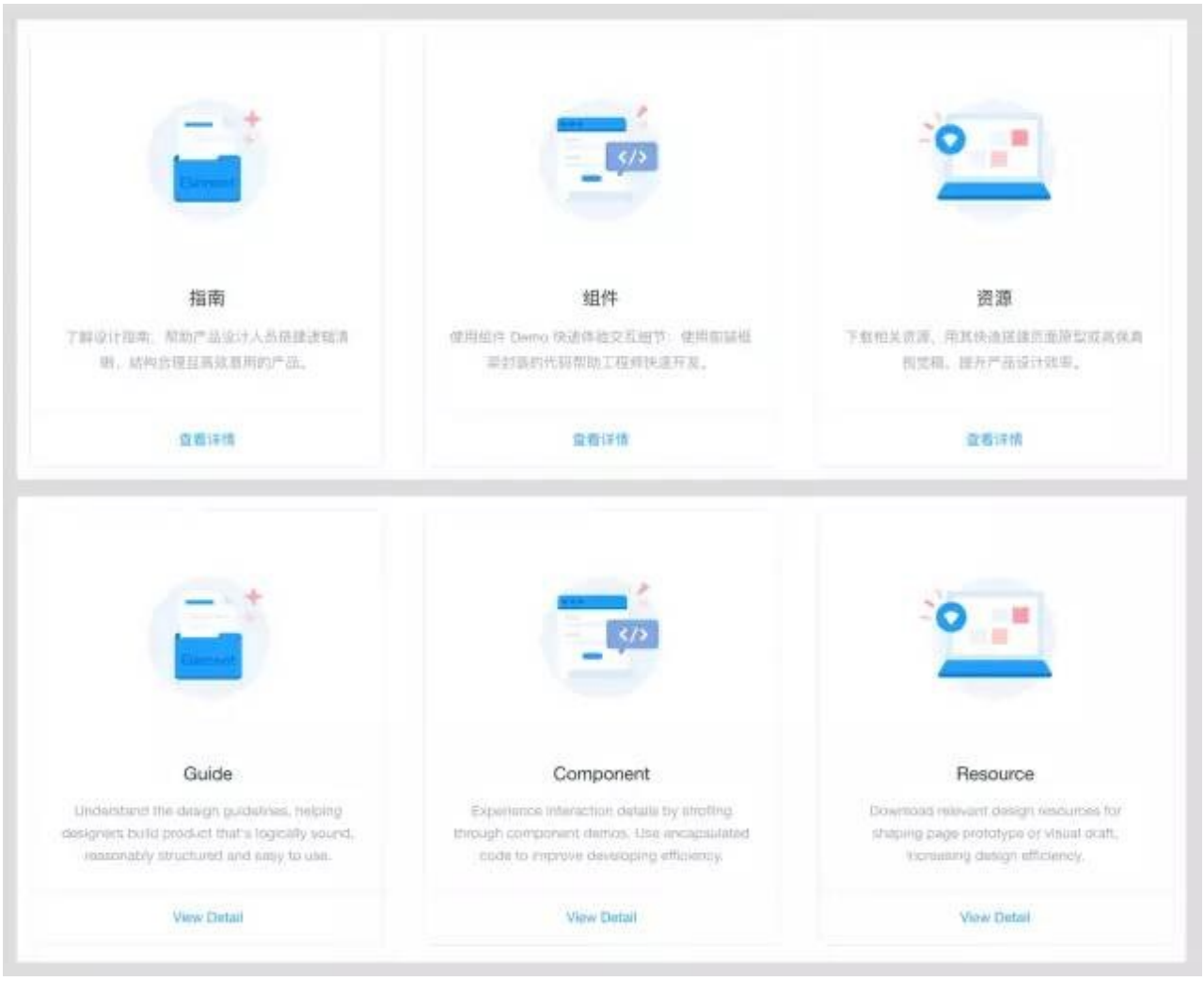
官网的路由是根据一个记录了导航信息的 `json` 文件自动生成的。因此需要在这个 `json` 文件中添加对应于其他语言的字段，并且根据新的数据结构修改路由生成的逻辑。

（2）页面

官网中除了文档外，还有一些介绍性质的页面。这些页面中文字比较多，如果人工管理每种语言的页面，若需要修改则必须去每个页面相应的位置进行编辑，有些繁琐。我们的做法是：每个页面对应一个模板，模板中的文字全部抽取到一个语言配置文件中，并且写了一个脚本生成最终的页面。这样在需要修改时，只需在语言配置文件中编辑对应的字段即可。

（3）网站组件

对于 `header` 、 `footer` 等通用的页面组件，我们采取了和上面类似的策略。但由于组件内的文字较少，于是没有再使用模板，而是通过路由判断应该显示何种语言。



> 中英文网站的显示效果

至此，我们也逐渐完善了技术栈。用 ES2015 和 CSS4 作开发语言、Lerna 负责管理组件、用 Karma 搭配 Mocha 和 Chai 等工具在 Travis CI 里做持续集成测试，最后用 Markdown 结合 Vue 写文档。我们甚至还在 CI 里实现了自动部署网站和推送主题仓库代码等功能，提升了不少开发效率。

6.开发过程中遇到的问题

具体到组件层面，在开发的过程中不可避免地会遇到一些问题。下面是我们的一些应对策略，希望能够抛砖引玉，引发大家的思考和讨论。

如何应对父子组件间事件派发机制的改变

在 Vue 2.0 中，用于父子组件间事件通信的 `$dispatch` 和 `$broadcast` 被移除了。官方的考虑是，基于组件树结构的事件流方式让人难以理解，并且在组件结构扩展的过程中会变得越来越脆弱。但是类似 Element 这样的组件库有几个特点：首先，父子组件间互相通信的场景非常常见，比如在一个带有验证功能的表单里，每个表单项在 `change` 或 `blur` 时需要通知表单组件进行校验；其次，组件的结构相对来说比较固定。

出于以上考虑，我们实现了简化版的 `dispatch` 和 `broadcast` ，并把它们包装成了一个 `mixin` ，方便在需要时调用。其中的 `dispatch` 代码如下：

```
dispatch(componentName, eventName, params) {
        var parent = this.$parent || this.$root;
        var name = parent.$options.componentName;
        while (parent && (!name || name !== componentName)) {
            parent = parent.$parent;
```

```
            if (parent) {
                name = parent.$options.componentName;
            }
        }
        if (parent) {
            parent.$emit.apply(parent, [eventName].concat(params));
        }
    }
```

可以看出，我们的实现需要在调用时传入 `componentName` （在各个组件中定义），这样就确保了事件只会在正确的组件中触发。

是否需要为用户代理自定义组件上的原生事件

在 Vue 2.0 中的自定义组件上使用 `v-on` 只会监听自定义事件（即组件用 `$emit` 触发的事件）。如果要监听原生事件，必须使用 `.native` 修饰符：

```
<mt-component @click.native="handleClick"></my-component>
```

这样一来，很多不太熟悉 Vue 2.0 语法的用户会发现给 Element 的组件绑定原生事件总是不生效。事实上，我们从开源以来收到的 issue 里被问得最多的一个问题是：如何给 `Button` 组件绑定 `click` 事件？

事实上我们只需要添加一行代码就能解决问题，但是关于是否需要让用户可以直接监听原生事件这件事在我们内部有两种不同的观点：一边认为应该遵循 Vue 的设计思想，原生事件要加 native；另一边认为 button 最常用的就是 click 事件，帮助用户做这件事可以降低学习成本。后来我们专门咨询了尤雨溪本人，他的观点是，对于一些组件的常用事件，可以允许用户直接监听原生事件，同时在文档中说明哪些事件可以直接监听，哪些事件需要加 `.native` 修饰符。最后我们决定从易用性的角度出发，让用户在使用 `Button` 组件时可以监听原生 `click` 事件，因为对于桌面端来说， `Button` 在绝大部分场景下都是需要监听点击事件的。 现在的 `Button` 支持以下两种写法：

```
<el-button @click.native="handleClick">Click Me!</el-button>
    <el-button @click="handleClick">Click Me!</el-button>
```

版本迭代的过程中，若 API 发生变化，如何友好地提示用户

在历次迭代中，我们会尽量保持 API 的一致。但是在一些万不得已的情况下，需要对 API 作出一些更新。对于老版本的用户而言，如果使用了被移除的 API，升级到新版后会出现一些意料之外的报错信息。为了友好地帮助用户尽快找到报错的来源，我们编写了一个 `mixin` ，当组件的 API 发生变化时，在组件中引入这个 `mixin` 并列出变化前后的字段名即可。

`mixin` 的核心代码为：

```
    const { props, events } = this.getMigratingConfig();
     const { data, componentOptions } = this.$vnode;
     const definedProps = data.attrs || {};
     const definedEvents = componentOptions.listeners || {};
```

```
    for (let propName in definedProps) {
        if (definedProps.hasOwnProperty(propName) && props[propName]) {
            console.warn(`[Element Migrating][Attribute]: ${props[propName]}`);
        }
    }
    for (let eventName in definedEvents) {
        if (definedEvents.hasOwnProperty(eventName) && events[eventName]) {
            console.warn(`[Element Migrating][Event]: ${events[eventName]}`);
        }
    }
```

引用了这个 `mixin` 的组件需要在 `methods` 中添加一个名为 `getMigratingConfig` 的方法，返回一个包含发生变化的 API 字段名和对应提示信息的对象：

```
getMigratingConfig() {
    return {
        props: {
            'selection-mode': 'Table: selection-mode has been removed.'
        },
        events: {
            cellclick: 'Table: cellclick has been renamed to cell-click.'
        }
    };
}
```

五、issue 处理方式

我们选择使用 Tower 来配合 GitHub 进行 issue 的追踪和处理。首先在 Tower 上建立几个清单：Plan、Design、Develop 和 Release。随后具体的操作流程如下：

- 从各渠道收集反馈
- 若不需设计，则由开发回复
- 若需设计跟进，则在 GitHub 上添加标签 `design` ，并在 Tower 的 Plan 清单中添加相应任务
- 开始处理任务后，为 GitHub 的对应 issue 添加 `working in progress` 标签，同时把任务拖进 Tower 的 Design 清单
- 设计完成后，开发接手，同时把任务拖进 Develop 清单
- 开发完成，经过设计师验收通过后将改动推送至 GitHub 仓库，关闭相应 issue，最后将任务拖进 Release 清单

六、总结

Element 从立项至今已经走过了五个月的时间。总的来说，这段时间就是一个不断发现问题和解决问题的过程，也是每个参与者自身成长的过程。开发时 Vue 2.0 正处于 RC 阶段，我们随着它的版本迭代踩到了不少坑，同时也给 Vue 提了一些 issue，并且都得到了 Vue 团队的处理。在此向Vue 团队的专业精神表示感谢。

自从 9 月开源以来，在社区的帮助下，Element 逐渐成熟，我们也在今天发布了它的第一个正式版本。希望越来越多的人能够参与进来，和我们一起把 Element 做得更好。

七、参考资料

- https://github.com/babel/babel
- https://github.com/lerna/lerna
- https://github.com/ElemeFE/postcss-salad
- https://github.com/QingWei-Li/vue-markdown-loader
- https://github.com/ElemeFE/cooking

精彩问答

淀粉提问：Element官网是多少？

Element开发团队：官网是 http://element.eleme.io

淀粉提问：如何修改样式？

Element开发团队：简单的样式可以通过覆盖来修改；对于大规模的自定义，我们提供了一套自定义主题的工具，文档看这里：https://github.com/ElemeFE/element/blob/master/custom-theme.md 。简单来说，通过修改样式变量、编译主题、引入主题，就可以实现自定义主题。

淀粉提问：Vue的作者给出了一套学习Vue路径，那Element是否有阅读源码的路径呢，怎么样才可以较为方便的理解源码，并且在基础组件不能满足自己业务的时候写出自己的组件呢？

Element开发团队：阅读源码的话，可以先 clone 项目后，先试试用 npm run new 指令创建一个新组件，看看我们的一个组件包含了哪些东西。要理解源码的话就自己边改代码边测试效果吧。

淀粉提问：怎么看待Vue添加redux,而又保留双向绑定的数据方式？

Element开发团队：不太明白想问什么。

淀粉提问：国际化网站怎么做的，是每种语言对应一个页面吗，还是统一的一个页面？

Element开发团队：Element 的主页是一个 SPA，每种语言对应了一个 .vue 文件，而这些 .vue 文件是通过一个统一的模板和语言配置文件生成出来的。

淀粉提问：请问会推出专门针对移动端的Vue2组件库吗？

Element开发团队：目前没有将 Element 移植到移动端的计划。不过，我们已经有一套移动端

组件库了：https://github.com/ElemeFE/mint-ui ，它有两个版本，分别兼容 Vue 1.x 和 2.0。

淀粉提问：写在 Vue 文件中的 MarkDown 输出在哪里？这并不是显示效果的一部分呀？

Element开发团队：Vue 和 MarkDown 的结合，我们是自己做了一个 vue-markdown-loader, 作用是将 MarkDown 文件转成 Vue 组件（生成的文件在插件的 .cache 目录里），最后通过 vue-loader 处理。可以去看看这个 loader 的源码就明白了。

淀粉提问：既然是基于Vue了，那么Element还有继续的必要吗？而且目前框架那么多，Vue也得到了人的认可。Element以后得生态如何保证？毕竟还只有目前饿了么一个团队在用。

Element开发团队：不太清楚“Element还有继续的必要”是什么意思，据我们在 Gitter 对用户的了解，现在已经有不少用户将 Element 实用到他们公司的产品开发里。Element 的生态发展除了我们团队本身以外还需要依靠开源的力量来进行优化和发展。

淀粉提问：在组件开发中,有对复用性很高的业务组件做过积累吗？如果做过～是怎么维护这些业务组件的？也是同逻辑组件的维护方法一样吗？

Element开发团队：组件开发中会不断收到各种的功能需求的反馈，通过 GitHub 仓库 issue 来推动我们组件的功能更新和维护。不太清楚“逻辑组件”的含义，Element 里除了按钮这样特殊的组件外都是带有逻辑的。

淀粉提问：请问将常见的jq插件或者说jq动效写成Vue组件的过程中有什么不顺畅的地方吗？

Element开发团队：在组件开发过程中没有参考任何 jq 插件的动效。实际上基于 mvvm 框架下的组件库开发相对于 jq 是轻松很多的，因为你不需要手动地处理事件绑定和视图的更新。

淀粉提问：分享的文档是否有些陈旧没有更新，Vue 目前已经不是 RC阶段了。

Element开发团队：文档中说「开发时」Vue 还处于 RC 阶段，主要指的是今年的八九月份，那时 Vue 每更新一个 RC 版本，我们就跟着做一遍测试，然后发现几个 bug 的情景还历历在目。Vue 2.0 是上个月正式发布的，我们也在今天发布了 Element 的正式版。

淀粉提问：有没有模板可以参考？

Element开发团队：有，看这里：https://github.com/ElementUI/element-starter。如果熟悉 cooking 或 laravel，我们也提供了相应的模板：https://github.com/ElementUI/element-cooking-starter 、 https://github.com/ElementUI/element-in-laravel-starter 。

淀粉提问：Element的开发者都是饿了么前端团队的吗，团队外的开发者占多少？

Element开发团队：是的。不过自从 Element 开源以来，社区出现了一批热心用户，他们也贡献了自己的代码。所有贡献者可以从这里看到：https://github.com/ElemeFE/element/graphs/contributors 。

淀粉提问：几年前，不计算Gzip 90kb的jQuery遭到了人们的嫌弃：太大了，还是用原生吧。几年后，开启Gzip后 仍有300kb的基于React的项目，人们觉得：区区300kb而已，算很小了。巨型库的概念流行后，人们似乎忘了啪啪啪在脸上有多疼。你怎么看？

Element开发团队：这几年网速已经提升不少，同时前端项目的复杂度也变得越来越高。比起库的体积，可能现在开发效率才是开发者所关心的。

淀粉提问：Element这套组件和Ant Design感觉有点类似，是否有借鉴过他们的设计？

Element开发团队：有借鉴过，我们不仅借鉴过 AntDesign，国内外大大小小的 Design Language 都有借鉴学习。很多信息前人已经总结过，我们希望可以快速的获得这些知识，以便更快的走到前方去探索更前沿的设计。

淀粉提问：感谢分享。目前有没有优秀应用案例可以分享？

Element开发团队：目前我们还没有精力去收集整理使用了 Element 的项目，不过按照最近一段时间在 Gitter 里讨论看到了一些开发者分享的项目链接，完成质量还是挺高的。过段时间我们会在 issue 里开始征集大家使用 Element 组件库的作品链接做分享。

淀粉提问：如何定制CSS，是覆盖还是改源码？改了源码如果Element版本更新之后 样式就没了？

Element开发团队：可以用 element-theme 主题自定义工具，或者直接下载 element-theme-default 主题包自己修改主题。如果只是简单的修改，建议直接覆盖样式。

淀粉提问：您好，我是个初学者，看不大懂编程，初学要做前端，应该先从哪开始入手学习？

Element开发团队：freecodecamp 和 codecademy 都是很好的入门途径。

淀粉提问：在技术选型的时候，基于什么考虑，选择了Vue，而不是React？

Element开发团队：和 React 相比，我们公司前端使用 Vue 的更多。为了照顾到大多数人，我们选用 Vue 作为 Element 的框架。

淀粉提问：我在写 Vue 的组件通信中，也发现事件比较好用但不容易被控制，因此一般是给自定义事件加上命名空间，如 “$dispatch('AComponent::rotate', 90)，”Element 团队有这样的实践吗？

Element开发团队：在 Element 的组件中对组件通信这一块并没有用到全局的 event bug，而是通过dispatch和broadcast来进行相互依赖的组件间的通信。虽然Vue2.0里弃用了$dispatch和$broadcast的api，但我们自己在组件库中封装了一遍。

淀粉提问：Element 团队使用 Vuex 时的一些具体情景是什么？

Element开发团队： Element 没有用到 Vuex 的情景。

淀粉提问：后续是否会开放一些常见业务组件，比如城市选择等？

Element开发团队：与后台业务关联很大的组件应该不会直接开放，不过这个组件如果能解决类似的其他问题，我们会抽离业务属性将之作为「基础组件」开放出来。城市选择只用了比较基础的 Select 组合而已。

淀粉提问：Element 在做动画效果这个方向上有些什么成绩呢？

Element开发团队：动画效果方面我们还没有人力去探索和研究，主要精力还是放在业务系统的搭建上。

嘉宾简介

Element 开发团队

照片从左到右依次为：
FuryBean、cinwell、杨奕、曾海平、敖天羽、Fishpaw、Fredddli、梓非徐

Element 开发团队由饿了么 UED 的设计师和大前端的工程师组成，他们协作开发了 Element——一套基于 Vue 2.0 的桌面端组件库。

今日荐文

Vue相关内容推荐：

- Vue作者尤雨溪：Vue 2.0，渐进式前端解决方案
- Vue 2.0 快速上手指南
- 更轻更快的Vue.js 2.0与其他框架对比
- Vue.js作者尤雨溪加盟Weex项目担任技术顾问

往期精彩分享：

- 滴滴：公司级组件库以及MIS系统的技术实践

长按二维码关注

前 端 之 巅

紧跟前端发展
共享一线技术
不断学习进步
攀登前端之巅