从 0 - 1 实现内部组件库设计

前端组件系统设计思路及模式

设计先行原则：对于组件库实现是一方面，但具体样式的设计一定要与业务相结合。

详细设计接口：这里以 form 为例，讲解一下作为一个合格的组件库，我们需要设计哪些接口。

组件库开发、管理及调试模式

组件库一般会搭配 lerna 、 babel-plugin 或者一些可视化工具来进行设计、开发和管理。

lerna 是一个多 package 的包管理工具，地址：https://github.com/lerna/lerna

它的作用是处理多个包在有相互依赖，都需要发布的时候，通过 lerna 的一个命令，就能同时更新多个包的版本和代码。我们就能方便的组织我们的代码库结构。

lerna 的使用也非常简单，只有两个命令 `lerna bootstrap` 和 `lerna publish` 。

lerna bootstrap 使 lerna 初始化整个项目，publish 使 lerna 来发布所有被索引的所有模块。

（看一个 lerna 使用的例子）

以 babel-plugin-import 为例，来讲下组件库的优化。

我们使用 `import {Button} from 'antd'` 从 ant design 中引入一个 Button 组件，但是大部分情况下，我们的 webpack 等打包工具会把所有的 antd 组件打包进来，当然在后期配合 tree-shaking 可以达到优化的目的。

在不支持 tree-shaking 的打包环境下，我们就可以使用 babel 插件 import，来将它最终形如 `import Button from 'antd/Button'` 。这样，我们打包工具就只会夹在 antd/Button 文件的内容，而不会把所有 antd 加载进来。

在我们业务实现的过程中，尤其在复杂场景，需要我们自定义一些插件来对我们最终的组件库进行一个优化，以便更好的让业务方来使用。

（看一个使用 import 插件的例子，同时实现一个简单的 import 插件）

代码开发完成之后，我们通常需要预览，调试，并且向大家展示最终所有组件库的配置项。一般这种时候我们可以使用 storybook 等工具，来快速展示我们的组件内容，当然这类的文档工具有很多，我们可以自行选择。

组件库常见开发问题

1. 样式代码如何设计?

   常见的组件库，我们的样式都是单独写在组件的文件中，最终所有的样式会进行一个合并打包，这就造成了我们在调用组件库的时候，还需要额外引入组件的样式。

   这种方式对组件库的使用来说不算特别友好，同时全局的样式业务方在调用时容易覆盖。

例如，我们在使用 antd 的时候，必须引入它内部组件的样式，当然我们也可以按照组件来手动引入，不过那样确实使用起来很麻烦。

```
@import '~antd/dist/antd.css';
```

另一种方式，就是我们使用 css-in-js 方案，使用了这种方案之后，业务方在调用组件时，只需要引入一个 js 组件即可，组件的样式已经内连到了我们的组件 DOM 结构上。

这样做的好处当然就是业务方比较轻松，因为不需要做太多额外的工作，但是这样做也有弊端，那就是组件库设计的过程中，必须有足够的弹性，能够应付部分业务需求对样式的定制，往往有些定制的需求，导致业务方在使用的时候需要使用 important，在设计时我们需要注意这种情况。

（对比介绍一下 css-in-js 方案，详细对比两者之间的优劣）

2. 主题、国际化、可访问性设计和单元测试的设计

   主题、国际化、可访问性则需要我们在组件库中提前想好相关的设计，标签上预留对应的接口。同时也需要增加一些运行时配置项，保证可以通过接口进行获取更新。

   在具体的实现中，组件库属于基础架构部分，所以对于组件库的发布和上线需要额外小心，一旦某个版本产生了问题，业务引入后可能会产生非常严重的后果。所以我们需要尽量写一部分单元测试，来保证代码发布过程中的 UI 一致和功能一致。