|  |
| --- |
| 信息分院 |
| 前端工程化 |
| [模块化-组件化-自动化-打包] |
|  |
| **李创** |
| **2019/2/12** |

|  |
| --- |
| 前端开发已经进入新的开发模式有别于传统开发的形式 |

目录

[前端工程化思想 3](#_Toc873538)

[模块化思想 3](#_Toc873539)

[自动化构建 3](#_Toc873540)

[做事情 4](#_Toc873541)

[所有构建工具的鼻祖是Node和npm 4](#_Toc873542)

[构建之后的文件仅仅就是一个准生产版本 5](#_Toc873543)

# 前端工程化思想

工程：设计（可行性分析、需求分析、概设、详设）、开发、测试、运维等各个环节分模块化（便于协作），组件化（便于维护），自动化构建（重复的劳动），打包完成项目

<https://www.cnblogs.com/ihardcoder/p/5378290.htmll>

<https://github.com/fouber/blog/issues/10>

# 模块化思想

传统开发中没有模块化开发思想，随着应用变得越来越复杂，需要多人协作开发所以有了模块的思想。

commJS模块化

requireJS模块化-AMD

seaJS模块化-CMD

UMD模块化

ES6模块化

# 自动化构建

构建工具的核心功能无非就两个：安装 vs 做事情。

安装包工具例如：npm、Bower，Yeoman可以安装几乎所有的东西。他们可以安装前端框架，比如Angular.js或者React.js。他们可以为你的开发环境安装服务器。他们可以安装测试框架。他们甚至帮你安装其他的前端构建工具。

做事情的工具比如Grunt、Webpack、Require.js、Brunch和Grulp。这些工具有些复杂。

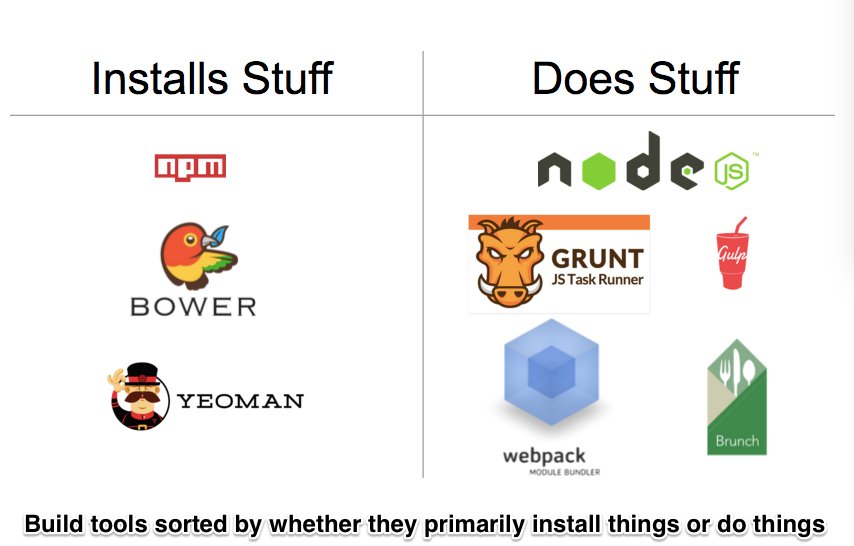
做事情的工具的目的就是自动化，它避免了前端开发中的烦琐事和一些易于犯错的地方。

为了做这些任务，他们经常使用他们自己生态链中的软件包和插件。每个工具都以不同的方式写任务。这些工具也并不都是一样的。一些“做”工具试图处理你的任何任务（Grunt，Gulp等）。其他的工具则专注于一件事情，比如处理Javascript中模块依赖关系的（Browseriry、Require.js等）

有时候你会发现你在一个项目中使用了多个这样的工具。

## 做事情

1.替换一个文件中的文本字符串  
2.创建一个目录然后往这个目录中移动一些文件  
3.用一条命令执行单元测试  
4.当我保存文件时刷新我的浏览器  
5.把我所有的JavaScript文件打包成一个文件，把我所有的CSS文件合并成一个文件  
6.压缩我的Javascript文件以及我的CSS文件  
7.修改网页中的script标签的位置



# 所有构建工具的鼻祖是Node和npm

Node和npm安装和运行所有的这些构建工具，所以你在项目中总是可以看到他们。正因如此，许多开发者在他们安装另外工具时都会多次使用到这两个工具。Node和NPM也掉进了我们的“做”和“安装”二分法。

Node就是做事情的工具，npm就是那个安装工具。

Npm可以安装像Angular或者React.js这样的框架。它也可以给你的应用安装一个服务器来方便你本地开发。它甚至可以安装一些工具来压缩你的代码。

Node，另一方面他可以做的事情还包括运行JavaScript文件，服务器等等。如果你想开始学习构建工具，那就从Node+npm，好好学。当你的项目越来越大时，Node和npm的功能可能就到达了瓶颈，这时你可以引入另外的构建工具来帮助你。

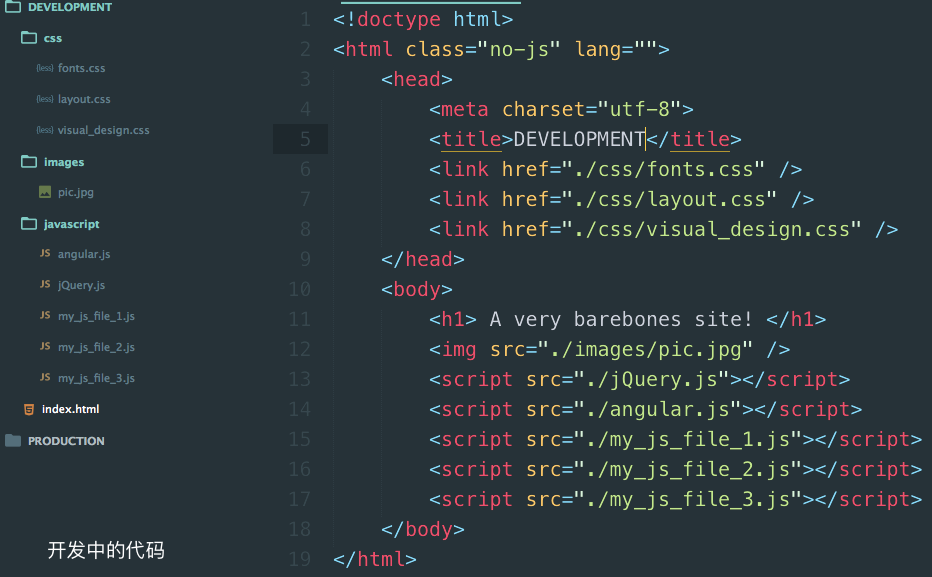
# 构建之后的文件仅仅就是一个准生产版本

开发者经常将JavaScript和css文件分为多个文件。每一个文件都可以让你专注于写某一个模块的功能。这样就可以减少你阅读代码的认知负担。（如果你觉得把文件分为多个文件有点迷惑的话，你想想假如你把一个文件写成5000行时，你怎么去读呢，这是你就知道分散文件的好处了）

但是当你需要把你的应用上线时，多个JavaScript和CSS文件是不现实的。当用户访问你的网站时，每一个js或者css文件都会发起一个新的http请求，这会让你的网站加载起来很慢。

所以为了改进这个问题，你可以创建一个build文件，这个文件把所有的css文件合并到一起，js文件也都合并到一个文件。这样，你就减少了用户请求的数量。为了创建build文件，你就需要一个构建工具。

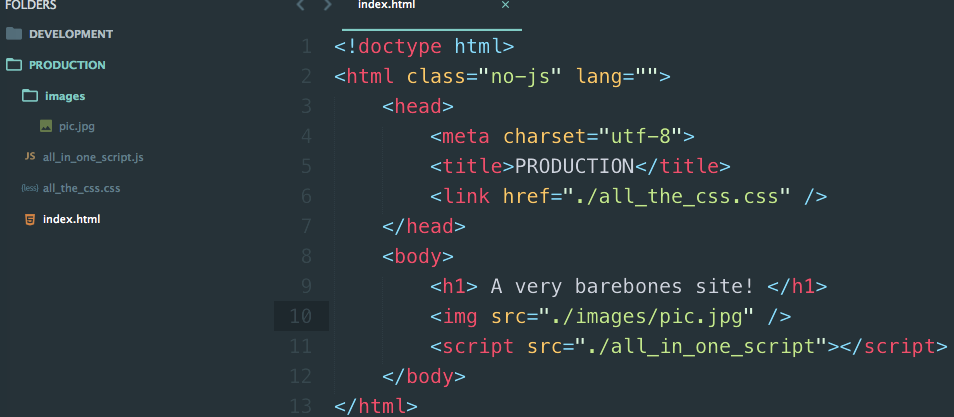
下面就是开发应用中的一个快照。注意它包含5个script标签和3个link标签了吗？如果你看左侧，注意到开发文件夹下有10个文件了吗？



然后下面就是同一个应用构建完之后的代码情况。

注意到我们只有一个script和一个link标签了吗？然后开发文件夹中只有4个文件了吗，之前我们可是有10个文件呢。

应用跟之前是一模一样的，我们只不过把他转化成一个称之为构建之后的代码小包。



你可以想知道为什么需要构建呢，难道就是为了节省用户几毫秒的时间吗？好吧，如果你创建了属于你或者几个人的网站的话，你不需要有这些困扰。生成一个构建之后的应用只有在高访问量的情况下才有必要。（或者那些你觉得以后可能会被高访问）

如果你正在学习开发或者仅仅在开发一个流量不怎么高的网站的话，生成一个构建的应用可能不是很有必要。

https://www.jianshu.com/p/e290f9f53b7e