# 现代 Javascript 工具漫游指南 -- 张鸿飞

![01](images/games-development-01.jpg)

文章翻译：[张鸿飞]( <http://weibo.com/u/5573672045/home?topnav=1&wvr=6>)

发表时间：2015 年 7 月 26 日

原文作者：Marcin Grzywaczewski

文章分类：Web 开发

## 关于本文

许多初学者开始 JavaScript ，但是不知道该采取什么样的代码生成工作。 JavaScript 中包含了很多小工具，本文通过列举 Babel.js、Webpack、Gulp、NPM、Bower 等常用的 Javascript 小工具，初步简单介绍了各工具的功能和用途，以便帮助初学者更好地学习理解和学习 Javascript。

## 文章内容

很多开发者是被 React.js 吸引才来到 JavaScript 的世界,但却迷茫于选择用于生产现代 JavaScript 代码的工具。 WebPACK、Babel、ESLint、Mocha、Karma、Grunt...应该用哪一个，每个工具又是做什么的? JavaScript 新手往往来自于类似 Ruby 和 Java 的社区，这些社区存在主观和全栈的解决方案。像 Rails 上的 Ruby 框架提供了很多立即可用的功能 - 事实上有可能你遇到的 JavaScript 问题是由此导致的。你不用去考虑你的代码生成工具 - 你的模板语言处理器、资源管道、缓存中间件和很多其他的东西都是预先配置的并且他们都在透明地工作。

JavaScript 工具往往由小工具、实用程序和库组成，在浏览器中生成你使用的代码。它们允许在发生变化后重新生成项目、运行测试套件、热重载你的代码等等。你可能会在这个世界迷失 - 当我第一次试图用 ES2015 代码建立 JavaScript 栈来工作时，我就迷失了。

事实上，这是你希望存在的一个问题 - 小零件组成的系统和流行的框架相比较整体更容易维护并且更灵活。但从这种小工具开始是非常困难的。

我想给大家简短介绍一下流行工具的用途 - 以及你需要与否。

## 现代 JavaScript 代码库的独特需求

通过现代化的 JavaScript 代码库工作与诸如 Ruby 之类的代码库稍有不同。这是因为环境（或构建目标）可能会显著不同。这也是因为您的主要环境，如浏览器不能直接读取你的代码并运行。由于 ES2015 在流行的浏览器中没有完全实现，你的代码需要事先准备好。这个过程被称为转换编译并有所谓的转换编译器工具来处理。

你的代码模块化也存在问题。 JavaScript 没有附带内置的解决方案来提供某种模块或API来实现这一点。这些东西都是特定环境的 - Node.js 带有自己的模块系统（基于 CommonJS 规范），浏览器实现模块的某些功能 – 如AMD、CommonJS、System、UMD 等。他们指定如何在 JavaScript 中引入和定义模块。还有叫做模块包的工具，可以把你的代码通过这种特定的模块系统编写，并从它创建了一个静态的 JavaScript 文件，可用于浏览器。

在使用 JavaScript 时，这两件事情都非常独特 - 大多数语言中都拥有自己的一套模块化技术包括在内。大多数语言也直接解释/编译来运行您的程序 - 这是不符合现代的 JavaScript 以前必须得转换编译的情况的。

考虑到这一点，让我们跳的流行工具列表来看看他们能做什么。

## 在现代 JS＆React 代码库中常用的流行工具和库

## [Babel.js](http://babeljs.io/)

Babel.js是一种转换编译器 - 它将你的代码通过 2015 版 ECMAScript 标准书写，再以旧浏览器可以运行的旧标准来生成代码，。它还允许您启用在测试的 2016 版 ECMAScript （又名 ECMAScript7 或 ES7 ）功能，并具有内置的 JSX 转换编译器。它可以采取 React 使用的 JSX 语法，并通过它生成 JavaScript 代码。

还有其他适用于最新 ECMAScript 标准的转换编译器 – 如[Traceur]( https://github.com/google/traceur-compiler)、[ES6-transpiler]( https://github.com/termi/es6-transpiler)以及[更多]( <https://github.com/addyosmani/es6-tools>)。Babel.js 是迄今为止转换编译代码最广泛采取的解决方案。

JavaScript 还带有多种类似版本 – 并且他们都有转换编译器。如[PureScript]( http://www.purescript.org/)、[CoffeeScript]( http://coffeescript.org/)、[TypeScript]( http://www.typescriptlang.org/)以很多语言带有自己转换编译器。你可以使用他们，如果你不想使用 2015 版 ECMAScript 来写代码，并且仍然在客户端浏览器上运行它。

## [Webpack](http://webpack.github.io/)

Webpack 是一个模块包。这需要模块化发展你的代码库，并且产生一个输出，将模块转换成浏览器可以理解的文件。

Webpack中自带开发服务器，它允许你在开发环境中承载你的代码。这种服务器同样也看着你的文件，并在每次做出改变时重新包装代码。

模块包还保持文件之间依赖关系的曲线图。这种特性是为什么其他如转换编译之类的编译步骤通常与这样的工具集成的原因。 Webpack 中确实使用这样的装载机来整合。您可以转换编译你的代码，压缩合并，从你的代码库中删除无用代码等等。

Webpack 也能够打包资源而不像 JavaScript - 你可以包含和转换编译样式表，优化静态资源等等。所有这些都集成都由于 Webpack 提供的装载机功能得以实现。

Webpack 有很多可替代工具，但现在只有一条路可走 - 这是在 React 社区中执行模块打包最流行的工具。同样的还有一些其他工具如[Browserify]( http://browserify.org/)、[JSPM]( http://jspm.io/)、[SystemJS]( <https://github.com/systemjs/systemjs>) 等可以实现。

## [Gulp](http://gulpjs.com/)

Gulp 是一种构建系统或任务执行器。它比模块打包机工作在一个更高的水平上。虽然模块打包机工作于模块和依赖树，Gulp 适用于你的文件结构 - 并能在其上执行任务。这与 UNIX 环境下的一个 make 工具非常类似。它基本上是一种序列化运行其他工具的方式。

在 Gulp 中你可以定义手动运行的任务。您可以定义这些任务需要在哪些文件上运行，关于他们需要完成哪些需求。

Gulp 可以操作工作流抽象。构建过程是一组工作流 – gulp 提供一个文件流并且通过另一个流来传送该流 - 如“压缩合并”流需要文件流作为输入，并产生压缩合并文件作为输出。然后你可以把它传送到另一个流，在压缩合并文件上实现相同功能等等。

所以基本上这个工具是关于一组文件工作的流程编排。这些工具可以压缩合并你的代码、修改、优化你的资源、压缩合并、从注释来编译文件、生成源地图、运行测试、部署你的代码....并且他们是非常通用的 - 对于文件同样有效。Gulp 本身对于模块或你的文件的意义并非一无所知- 但它可以让其他工具来对这些文件执行任务。

Gulp 还带有基本的观察器 - 你可以观察一组文件，并在他们改变时对其执行任务。

Gulp 是市场上较新版本的编译系统中的一个。还有像 [Grunt]( <http://gruntjs.com/>) 之类的更丰富的工具。

构建系统可以做很多模块打包机可以做的事情。但他们可能通过一个不太优化的方式来实现，因为他们都工作于文件 - 他们不知道模块打包机知道的抽象概念。你也可以在你的构建系统中运行你的模块打包机。所以这是一个比模块打包机更高层次的工具。在决定应该在您的环境使用什么时应该把它考虑在内。

## [npm](https://www.npmjs.com/)

NPM 是一个包管理器。它和系统软件包管理器做同样的工作。但对 JavaScript，这是下载你的所有环境的工具。NPM 负责下载包，解决对他们的依赖并且在项目周围提供包的抽象。所以，当另一家开发商想与你的代码库工作时，他所需要做的是发出 NPM install 命令然后所有的依赖会自动安装。在这样的包中，你还可以包含许可证信息、姓名、关键词、版本、描述和许多其他元数据的代码。如果你正在开发一个库，NPM 还可以帮助您在以后发布它并使其可用于所有在Node.js 的环境中工作的开发人员。

所以，如果你想安装另一个很酷的库来使用，或 Webpack 的其他装载机，或者Webpack 自身 - NPM是可为您处理它的工具。

到目前为止，据我了解，还没有 NPM 的替代品。

## [Bower](http://bower.io/)

Bower 是静态资源的软件包管理器。同时 NPM 允许你安装软件包，可用于构建你的代码，Bower 可以帮助你，如果你想要一个独立的库，而不是通过构建或引入它。它所做的是下载少量的 JS 文件、样式表和图像 - 你可以将它们导入到您的浏览器并马上工作。

当你建立你的环境时，Bower 可以为您提供您可以使用的插入式静态文件。当你不想要建立你的代码时，可以令人眼前一亮，但对静态资源有效果。对于简单的网站和原型更有用 - 获取一个库，并立即在浏览器上与所需的所有资源来使用它。

如果您使用的是基于 Rails 的编译环境，[Rails-Assets ]( <https://rails-assets.org/>)就是 RubyGems 和 Bower 之间的代理 - 这样你就可以通过把它包含在你的 Gemfile 中在你的代码库使用 Bower 包。

《React.js通过实例就在这里！》

想要通过建立流行小部件来了解 React.js 和 2015版 ECMAScript？

使用优惠码 LEARNREACT 来得到这本书早期版本访问的 30％优惠！大多数例子的库都包含在内。

[Learn More](<http://reactkungfu.com/react-by-example/>)

![02](images/games-development-02.png)

## [Mocha](http://mochajs.org/)

Mocha 是一个测试框架。它所做的就是运行测试。它还提供了设置测试用例的API - 让你的测试分成若干小块。

它能够在浏览器或 Node.js 环境中运行测试。它也可以根本无需运行浏览器来运行测试你的浏览器代码 - 使用如 [PhantomJS]( <http://phantomjs.org/>) 的无头浏览器。

你可以使用任何你喜欢的测试库。Mocha 不会假设你在测试中会做什么。

您可以在测试你的代码前配置 Mocha 来执行任务 – 类似运行 Webpack 然后再测试诸如此类。

对 Mocha 最显着的替代是[Karma]( http://karma-runner.github.io/0.13/index.html)。做的是相当类似的工作，并都被社区所喜爱。

## [Jasmine](http://jasmine.github.io/2.3/introduction.html)

Jasmine 是用于测试的库。它为关于你代码的声明或期望提供一个 API。它比Mocha的工作层次较低，它运行你的测试 – Jasmine 定义了测试是如何创建的。它是用于测试 JavaScript 代码的最流行的库。

还有像 [Chai]( <http://chaijs.com/>) 的替代品。也有更多的专业库，可以帮助你测试，如 [Sinon.js]( <http://sinonjs.org/>) 的 stubbing。

### ## [[ESLint](http://eslint.org/)]( <http://eslint.org/>)

### ESLint 是2015版ECMAScript 代码的剥绒机。虽然转换编译器可以告诉你，您的代码有什么语法错误，剥绒机却可以告诉你，你的代码没有遵循最佳实践。当你想保持高质量代码时，这将是一个很大的帮助，。它也可以变得很讨厌，因为它为您的代码假定某种风格指南 - 但它是高度可配置的，所以你可以改变行为，甚至禁用一些检查。您可以将 ESLint 与你的编辑器整合成一体 - 像 Sublime Text（使用 SublimeLinter）或 VIM （使用 syntastic），所以根据 ES2015 良好代码原则，你的代码可以被经常检查。

### 也有 ESLint 插件可以大体上检查 JSX 的最佳实践和 React 做法 – 例如[eslint-plugin-react]( https://www.npmjs.com/package/eslint-plugin-react)。

### 这是保证代码质量的一个很好的工具。如果你工作在团队中，它也可以用来建立项目的某种准则- 所以许多人书写的的代码可以根据最佳实践进行检查。且不说它可以集成到你的连续部署服务器中，所以质量如此差的代码将被停止进入生产环境当中。真是利落！

### 有一个有点过时称为 JSLint 的替代物可以完成同样的事情。

### 虽然有用，ESLint 却是高度可选的，如果你不想，你可以不需要使用它。

## [Yeoman]( <http://yeoman.io/>)

正如已经说过的，将这些工具拼接起来对初学者来说是一个艰巨的任务。Yeoman就是用来简化这个过程的。它是一个 \_scaffolding 工具。你可以发现你想拥有的一系列功能列表，并且找到一个 Yeoman 生成器来提供这些功能。发出一个命令之后，你可以拥有一个包含你需要的功能的预配置环境。

这也是一个伟大的工具，如果你建立了一个项目结构，并想在其他项目中重新使用它。你只需要创建一个 Yeoman 生成器出来，然后你可以在几秒钟内引导具有类似配置的项目。

## 总结

噢，这是一个很大的工具！拥有长处和短处。这样的小工具的方法最大的缺点是，它是很难学会使用什么和如何配置它。好处是，如果是你不喜欢的工具，你可以很容易地取代它 – 我在这里描述的大多数工具都是有替代品的。

如果您认为有一个非常受欢迎的工具，值得考虑加入该名单，[让我们知道]( https://twitter.com/reactkungfu)。我希望这对于理解 JavaScript 构建系统和一般使用的工具将会是一个很好的开始。

祝你学习现代JavaScript顺利！

> 更多IT技术干货: [wiki.jikexueyuan.com](wiki.jikexueyuan.com)

> 加入极客星球翻译团队: [http://wiki.jikexueyuan.com/project/wiki-editors-guidelines/translators.html](http://wiki.jikexueyuan.com/project/wiki-editors-guidelines/translators.html)

> 版权声明：

> 本译文仅用于学习和交流目的。非商业转载请注明译者、出处，并保留文章在极客学院的完整链接

> 商业合作请联系 wiki@jikexueyuan.com

> 原文地址：[http://reactkungfu.com/2015/07/the-hitchhikers-guide-to-modern-javascript-tooling/]( http://reactkungfu.com/2015/07/the-hitchhikers-guide-to-modern-javascript-tooling/)