2015 年末必备前端工具集

JavaScript (/lib/tag/JavaScript) APP (/lib/tag/APP) 2016-02-28 19:14:11 发布

来自： http://www.wtoutiao.com/p/170tG4M.html (http://www.wtoutiao.com/p/170tG4M.html)

本文选自《开发者头条》2 月 27 日最受欢迎文章 Top 3，作者 Zindex ，感谢 @leohxj 分享。

欢迎分享 ： http://toutiao.io/contribute

“Javascript没法胜任大型应用，因为它甚至不能确定一个变量的类型，而且很难重构”~一大堆困惑的人

当我初识Javascript的时候，只有一种浏览器需要关心：NetScape。它在微软开始捆绑销售IE和操作系统之前完全统治了世界。在那些日子里，Javascript的开发者工具很弱这种观点的确是对的。

不过这个观点已经被推翻很久了，今天，Javascript已经拥有了在我见过的所有语言中最好的开发工具生态系统。

请注意，我没有说“最好的IDE”。如果你正在寻找一款统一了不同开发工具使用体验的集成式IDE，请试试为C#打造的微软Visual Studio，和Unity一起使用风味更佳。虽然我本人并没有使用过，但是听我信任的人说这很靠谱。

我用过C++和虚幻引擎。当我第一次试用的时候，我意识到web平台的开发工具仍然有很长的路要走。

不过我们已经走过了很长一段路，现在我们在JS中使用的工具让IDE神奇的自动补全看起来就像是 小孩的玩具。尤为是JavaScript的运行时工具，在我见过的所有其他语言中都没有对手。

“Javascript拥有在我见过的所有语言中最好的开发工具生态系统。”

什么是开发者工具？

开发者工具是一套让开发者更轻松的软件集合。传统上，我们主要将IDE，linter，编译器，调试器，和性能分析器认为是开发者工具。

不过JavaScript是一种动态语言，伴随它的动态特性而来的是对运行时开发者工具的需求。JavaScript对此的需求程度很高。

为了实现我写这篇文章的初衷，我将包括对运行时工具的介绍，甚至包括一些能提升运行时开发者工具可视化和调试体验的库。开发工具与库之间的界线将开始模糊。与之而来的将是令人震惊的结果。

开发者工具一览表

- Atom & atom-ternjs
- Chrome Dev Tools
- PageSpeed Insights
- FireFox Developer Edition
- BrowserSync
- TraceGL
- ironNode
- ESLint
- rtype (规范) & rfx (库) 提示: 这些是未完成的开发预览版
- Babel
- Greenkeeper.io & updtr
- React
- Webpack + Hot module replacement
- Redux + Redux DevTools

关于这些工具

你的开发者生涯将围绕着这两个东西展开： 编辑器 ，和你的 运行环境 （比如，浏览器，平台，和你代码的目标设备）

编辑器：我是用着像Borland IDE，微软Visual Studio，Eclipse和WebStorm这样的大型，重量级，高度集成的IDE开始我的职业生涯的。我认为这些IDE中最好的是 **WebStorm** 和 **Visual Studio** 。

但是我对这些IDE体积的膨胀感到厌倦，所以在最近几年里，我的代码都是在更精简的编辑器中写成的。主要是Sublime Text，不过我最近切换到了 **Atom** [1] 。你一定会需要 atom-ternjs[2] 来启用JavaScript智能感知特性。你可能也会对Visual Studio Code感兴趣。这是一个简约版Visual Studio，专为喜欢像Sublime Text和Atom这样的小型可拓展编辑器的人打造。

我也使用vim在终端里进行快速编辑。

调试器：在我开始web编程之旅时，我想念那些集成的调试器。不过Chrome和FireFox团队将运行时调试提升到了一个全新的水准。今天似乎每个人都听说过Chrome DevTools，并且知道如何逐步调试代码。不过你知道它有对性能及内存进行记录和审查（profiling and auditing）的高级特性吗？你用过flame charts或者the dominators view吗？

说到性能审查，你需要了解 PageSpeed Insights[3] :

在这些好东西之上，Chrome DevTools也有一些酷炫的特性，比如像CSS实时编辑，以及可以帮助你编辑动画的超酷特性。去了解Chrome DevTools吧，你不会后悔的。

为了不被超过，FireFox有一个专为开发者打造的浏览器 FireFox Developer Edition[4] :

BrowserSync: BrowserSync[5] 可以一次同时控制几个浏览器，这是检测你的响应式布局的一种好办法。换句话说，你可以使用BrowserSync CLI来在桌面，平板和手机上打开你的app。

你可以设定文件监视（watch files），然后当文件改动时，几个同步的浏览器会自动刷新。滚动，点击，以及表单互动这些动作都将会被同步到所有设备，所有你可以简单的测试你的app的工作流，确保事情在任何设备上都看起来都对劲。

TraceGL: TraceGL[6] 是一个运行时调试工具，它让你可以观察软件中实时发生的所有函数调用，而不是逐步手动调试你的代码，一次一步。这个是一个超级强大和有用的功能。

ironNode: ironNode[7] 是一个用于调试Node的桌面app。由Electron，一个桌面跨平台运行时驱动。Electron也驱动了Atom编辑器。就像node-inspector，ironNode允许你使用类似Chrome DevTools的特性来追踪你的代码。

将ironNode和Babel一起使用，我使用如下的 *debug.js* 脚本：

```
require('babel-core/register');
require('./index');
```

加载调试器：

```
iron-node source/debug.js
```

这就像魔法一样，不是吗？

Linting: ESLint[8] 是目前为止我用过的各种语言的linter中最好的。我喜欢ESLint甚于JSHint，ESLint比JSHint好太多了。如果你不确定使用什么，别担心，使用ESLint。为什么它这么酷呢？

- 可配置性高 - 每一个选项都可以被开启或关闭。这些选项甚至可以接收参数。
- 创造你自己的规则。你有你想要在你的团队中强制执行的代码规范吗？在linter中可能已经有了这样的规则，不过如果没有，你可以写你自己的规则。
- 支持插件 - 使用了某些特殊语法？ES6+或者未来版本JavaScript的实验性特性？没问题。使用了React的JSX语法打造简洁的UI组件？没问题。使用了你自己的实验性JavaScript语法拓展？没问题。

  类型支持：JavaScript具有松散的类型，这意味着你不必注解所有的类型。过去数年我在C++和Java这样的语言中注解所有东西。当我开始使用JavaScript之后，我感到如释重负。类型注解在你的源文件中制造了杂音。函数通常在没有类型注解时更易用。

  和大众认知相反， **JavaScript**是有类型的 ，但是JavaScript在 值 层面区别类型而不是变量层面。变量类型可以被类型推断识别并预测出来（这就是Atom TernJS插件的作用）。

  这就是说，类型注解和签名（signature）声明是为了一个目的：它们对于开发者来说是不错的文档。它们也使JavaScript引擎以及编译器作者的一些重要性能优化成为可能。作为一个构建app的JavaScript程序员，你不应该担心性能问题。把这些留给引擎和制定规范的团队吧。

  不过关于类型注解我最喜欢的一点是运行时类型反射。使用类型反射可以开启运行时开发者工具。想知道这样的工具是什么样的，请阅读 “The Future of Programming: WebAssembly and Life After JavaScript”[9] 。

数年来，我使用JSDoc来注解类型，编写文档以及类型推断。不过我对其麻烦的限制感到厌倦。这感觉就像你使用一种不同的语言编写代码，之后将它挤压成JavaScript（这是真的）。

我也对TypeScript的结构化类型方案感到印象深刻。

不过TypeScript存在一些问题：

- TypeScript不是标准的JavaScript - 选择TypeScript意味着选择TypeScript编译器以及工具生态 - 这通常导致你无法选择为JavaScript标准设计的方案。
- TypeScript很大程度上基于class。这与JavaScript的原型和对象组合特性八字不合。
- 目前为止，TypeScript不提供运行时解决方案… - 他们正在使用实验性的新JavaScript **Reflect** API构建。不过接下来你可能会依靠这些实验性极高的规范特性，这些特性也许会成为最终标准，也许不会。

因为这些，我启动了（目前还未完成） rtype[10] 和 rfx[11] 项目。 **rtype** 是一个函数和接口反射规范，对于了解JavaScript的读者来说，这一规范形成了不言自明的文档。 **rfx** 是一个用于封装已经存在的JS函数及对象然后添加类型元数据的库。同时，它也可以加入自动运行时类型检查。我正在积极的与人们合作以改进rtype和rfx。也欢迎你们的贡献。

你要记得rtype和rfx还非常年轻，并且在短期之内几乎必定会有革命性的变化。

Babel: Babel[12] 是一个让你立即在JavaScript代码中使用还不被支持的ES6+, JSX以及其他特性的编译器。它的原理是将你的代码翻译成等价的ES5代码。一旦你开始使用它，我敢说你将很快对新语法上瘾，因为ES6为这门语言提供了一些真正有价值的语法拓展，像解构赋值（destructuring assignment），默认参数值，剩余和展开参数（rest parameters and spread），简洁对象字面量（concise object literals），以及更多… 阅读 “How to Use ES6 for Universal JavaScript Apps”[13] 来了解细节。

Greenkeeper.io: Greenkeeper[14] 监控你的项目依赖并且自动向你的项目提交一个pull request。你要确保你已经设定了CI解决方案来自动测试pull requests。如果测试通过，只要点击“merge”，就完工了。如果测试失败，你可以手动跟进并且找出哪里需要修复，或者直接关闭PR。

如果你偏爱手动的方法，看看 **updtr** [15] 。在你第一次开启Greenkeeper之前，我推荐先在你的项目上运行updtr。

Webpack: Webpack[16] 将模块和依赖打包成浏览器可用的静态资源。它支持大量有趣的特性，比如模块热替换，这让你正在为浏览器编写的代码在文件更改时自动更新，而不用刷新页面。模块热替换是迈向真正持续实时开发反馈循环的第一步。如果你还没有使用webpack，你应该使用它。为了更快入门，看看 **Universal React Boilerplate** [17] 这个项目里的webpack配置。

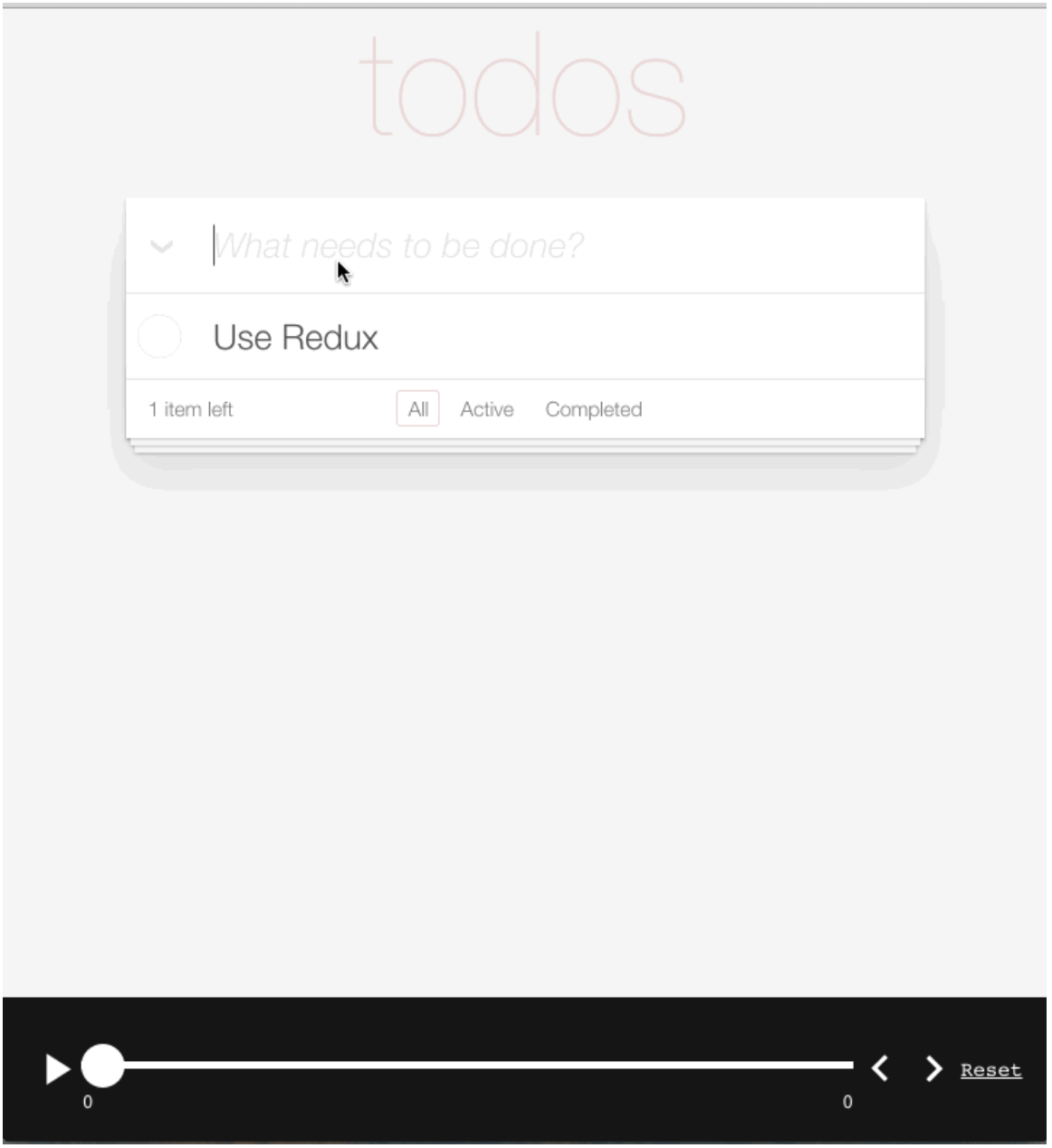
React:这一个有一点跑题，因为 React[18] 严格意义上来说不是一个开发者工具。它和一个UI库有着更多的共同点。请把React想象成现代的jQuery：一种更简单的处理DOM的办法。但React比这更强大。事实上，你可以把React对准一大堆DOM之外的平台，包括原生移动UI APIs(iOS & Android)，WebGL, canvas以及更多。Netflix将React的目标平台设为了他们自己的Gibbon TV设备渲染API。

所以为什么我将React列在开发者工具之中？因为React的抽象层被一些不错的开发者工具使用，来驱动代表未来趋势的惊人工具。特性有热加载（更新你的实时运行代码而不刷新页面），时间旅行（time travel），以及更多… 继续阅读！

Redux + Redux DevTools:Redux是由React/Flux架构和函数式编程的纯函数概念启发而来的一个应用状态管理库。另一个在开发者工具列表中的库？是的，以下是原因：

Redux以及Redux DevTools使得在你的实时运行代码之上进行真正的下一代调试互动成为可能。这让你可以轻松洞察在你的app中已经发生的行为：

它甚至允许你使用时间旅行调试这个特性在时间中来回穿梭。这是它在滚动视图中看起来的样子：



结论

JavaScript有着我所见过的所有语言中最丰富的开发者工具集。你可以看到，这更像是一个拼凑的过程而不是一个统一的IDE环境。不过我们处于JavaScript开发的寒武纪大爆炸时期，在未来，我们也许会看到现成的统一集成开发者工具。与此同时，我们将一瞥编程未来走向的究竟。

随着JavaScript向统一的应用状态（unified application state）和不变性（immutability）（正是这个特性使得Redux DevTools的时间旅行调试成为了可能）的更深处推进，我预测我们将看到更多的实时编程特性上线。

我也相信我们构建的应用和我们用以构建它的开发环境之间的界线会随着时间消逝而渐渐模糊。举个例子，Unreal游戏引擎将蓝图编辑集成进了引擎自身，这允许开发者和设计师从运行的游戏中构建复杂的行为。我思考了很久，我们将开始看到这些特性出现在web和以及原生移动应用中。

JavaScript的linting，运行时监视（runtime monitoring）和时间旅行调试特性在我所知道的任何语言中都没有对手。但我们还可以做更多，比如将同等于Unreal 4引擎中的蓝图系统这样的工具带给我们。我迫不及待的想看接下来会发生什么。

更多优质内容，欢迎安装、使用《开发者头条》iOS、Android 客户端。

体验地址： http://toutiao.io/download

:arrow_down::arrow_down::arrow_down:

同类热门经验

1. Node.js 初体验 (/lib/view/open1326870121968.html)
2. JavaScript开发规范要求 (/lib/view/open1352263831610.html)
3. 使用拖拉操作来自定义网页界面布局并保存结果 (/lib/view/open1325064347889.html)
4. Nodejs入门学习，nodejs web开发入门，npm、express、socket配置安装、nodejs聊天室开发 (/lib/view/open1329050007640.html)
5. 利用HTML5同时上传多个文件 - resumable.js (/lib/view/open1327591300671.html)
6. nide：一个不错的Node.js开发工具IDE (/lib/view/open1325834128750.html)

阅读目录

- 360校园招聘2015届技术类笔试题.pdf (http://www.open-open.com/doc/view/d105ecb84d11448f9c0bc4e41588255d)
- 2015小米校招技术类笔试题.pdf (http://www.open-open.com/doc/view/6a336d1797af460abbbb1754bb793ab3)