用standard来管理JavaScript代码规范

> 链接：http://coderunthings.com/2017/08/12/js-standard/

怎么才能使前端团队写的JavaScript代码都有统一的风格、符合规范呢?

要解决这个问题大概要做下面几件事:

1. 制定规范标准
2. 写代码时执行代码规范
3. 检查代码是否符合规范
4. 修改不符合规范的代码

那么，到底具体应该如何实施以上几个步骤呢？我的回答是你什么都不用做了，有人已经帮你把以上四步全部做好了!

是的，这个好用的工具就是本文要介绍的JavaScript规范库，standarad。

关于standard

standard是什么?

一个开源的JS代码规范库，它做了以下事情

1. 制定了所谓standard(标准)的JS代码规范
2. 配合编辑器插件可以实时检查代码规范以及语法错误
3. 通过执行命令检查代码规范以及语法错误
4. 自动修复(可以直接修复的)不合规范的代码,使其符合规范

关于standard中的代码规范

可以说JS这门语言的魅力之一就是自由、开放的写法，相比python、Go等语言，JS写起来自由的多。但是这种自由本身在团队合作的项目里也带来了很多的不便，于是我们需要指定代码规范，但是应该以什么标准来制定规范呢?缩进到底是4格还是2格、结尾要不要用分号、花括号和if语句在同一行还是另起一行？诸如此类的问题，从功能和逻辑上来讲并没有标准答案，因为无论怎么选，代码都能运行，功能都能实现。所以在JS程序员的世界里经常会有诸如缩进、分号、换行等写法的争论，standard库对此给出的结论是，这种争论对于getting stuff done并无意义，我们要停止这方面的争论，把精力放在解决问题上。

standard官方给出的说法如下:

There are lots of debates online about tabs vs. spaces, etc. that will never be resolved. These debates just distract from getting stuff done. At the end of the day you have to 'just pick something', and that's the whole philosophy of standard – its a bunch of sensible 'just pick something' opinions. Hopefully, users see the value in that over defending their own opinions.

使用standard

好了，开始使用standard吧。

1. 新建一个项目

```
$ mkdir my-project
$ cd my-project
```

2. 安装standard

```
$ npm init
$ npm install standard --save-dev
```

3. 安装snazzy，让代码检查的结果输出更加美观

```
$ npm install snazzy --save-dev
```

4. 配置package.json, 添加一条名为lint的npm script

```
"scripts": {
 "lint": "standard --verbose | snazzy"
}
```

5. 在项目下新建一个app.js文件,随意输入一些代码并保存

```
// app.js              // 错误说明
const a = 1;           // 结尾不应该有分号，而且a定义了没有使用
const f = () => {}     // f定义了但是没有使用

log()                  // log未定义
```

6. 运行代码检查

```
$npm run lint
```

7. 看到检查结果

```
1:7   error  'a' is assigned a value but never used  no-unused-vars
1:12  error  Extra semicolon                         semi
2:7   error  'f' is assigned a value but never used  no-unused-vars
4:1   error  'log' is not defined                    no-undef
```

standard的更多功能

1. 使用编辑器插件，实时检查代码规范
2. 忽略某些不需要执行代码规范的文件
3. 局部禁用代码检查
4. 指定全局变量，以避免变量未定义错误
5. git pre-commit钩子，在每次commit之前检查代码规范
6. 为更多最新的JS语法添加检查规范

7. 自动修复