

ECMAScript6 入门及环境搭建

JavaScript 的历史

诞生

JavaScript 诞生于 1995 年。起初它的主要目的是处理以前由服务器端语言负责的一些表单验证操作。在 JavaScript 问世之前，必须把表单数据发送到服务器端才能确定用户是否没有填写某个必填域，是否输入了无效的值。那个时候，绝大多数因特网用户都使用速度仅为 28.8kbit/s 的“猫”(调制解调器)上网，想象一下；用户填写完一个表单，单击“提交”按钮，然后等待 30 秒钟，最终服务器返回消息说有一个必填字段没有填好……；当时走在技术革新最前沿的 Netscape (网景) 公司，决定着手开发一种客户端语言，用来处理这种装简单的验证。当时就职于 Netscape 公司的布兰登·艾奇开始着手计划将 1995 年 2 月发布的 LiveScript 同时在浏览器和服务端中使用。

Javascript 与 Java 的关系

为了赶在发布日期前完成 LiveScript 的开发，Netscape 与 Sun 公司成立了一个开发联盟。而此时，Netscape 为了搭上媒体热炒 Java 的顺风车，临时把 LiveScript 改名为 JavaScript，所以从本质上来说 JavaScript 和 Java 没什么关系。

Javascript 的版本问题

JavaScript 1.0 获得了巨大的成功，Netscape 随后在 Netscape Navigator 3(网景浏览器)中发布了 JavaScript 1.1。之后作为竞争对手的微软在自家的 IE3 中加入了名为 JScript (名称不同是为了避免侵权)的 JavaScript 实现。而此时市面上意味着有 3 个不同的 JavaScript 版本，IE 的 JScript、网景的 JavaScript 和 ScriptEase 中的 CEnvi。当时还没有标准规定 JavaScript 的语法和特性。随着版本不同暴露的问题日益加剧，JavaScript 的规范化最终被提上日程。

JavaScript 与 ECMAScript 的关系

1997 年，以 JavaScript1.1 为蓝本的建议被提交给了欧洲计算机制造商协会 (ECMA, European Computer Manufactures Association) 该协会指定 39 号技术委员会负责将其进行标准化，TC39 来自各大公司以及其他关注脚本语言发展的公司的程序员组成，经过数月的努力完成了 ECMA-262——定义了一种名为 ECMAScript 的新脚本语言的标准。第二年，ISO/IEC (国际标准化组织和国际电工委员会) 也采用了 ECMAScript 作为标准 (即 ISO/IEC-16262)。

ECMAScript 是什么？

ECMAScript 是 Javascript 语言的标准。

ECMA European Computer Manufactures Association (欧洲计算机制造商协会)，主要任务是研究信息和通讯技术方面的标准并发布有关技术报告。

ECMAScript6：简称 ES6，是 JavaScript 语言的下一代标准，也是目前正在发布的最新 JavaScript 标准，由于 ES6 是在 2015 年发布，所以 ES6 也成为 ECMAScript2015。

ECMAScript 的发展历史

1998 年 6 月，ECMAScript 2.0 版发布。

1999 年 12 月，ECMAScript 3.0 版发布，成为 JavaScript 的通行标准，得到了广泛支持。

2007 年 10 月，ECMAScript 4.0 版草案发布，对 3.0 版做了大幅升级，预计次年 8 月发布正式版本。草案发布后，由于 4.0 版的目标过于激进，各方对于是否通过这个标准，发生了严重分歧。

2008 年 7 月，由于对于下一个版本应该包括哪些功能，各方分歧太大，争论过于激进，ECMA 开会决定，中止 ECMAScript 4.0 的开发，将其中涉及现有功能改善的一小部分，发布为 ECMAScript 3.1，而将其他激进的设想扩大范围，放入以后的版本，由于会议的气氛，该版本的项目代号起名为 Harmony (和谐)。会后不久，ECMAScript 3.1 就改名为 ECMAScript 5。2009 年 12 月，ECMAScript 5.0 版正式发布。Harmony 项目则一分为二，一些较为可行的设想定名为 JavaScript.next 继续开发，后来演变成 ECMAScript 6；一些不是很成熟的设想，则被视为 JavaScript.next.next，在更远的将来再考虑推出。

2011 年 6 月，ECMAScript 5.1 版发布，并且成为 ISO 国际标准 (ISO/IEC 16262:2011)。

2013 年 3 月，ECMAScript 6 草案冻结，不再添加新功能。新的功能设想将被放到 ECMAScript 7。

2013 年 12 月，ECMAScript 6 草案发布。然后是 12 个月的讨论期，听取各方反馈。

2015 年 6 月 17 日，ECMAScript 6 发布正式版本，即 ECMAScript 2015。

ECMA 的第 39 号技术专家委员会 (Technical Committee 39, 简称 TC39) 负责制订 ECMAScript 标准，成员包括 Microsoft、Mozilla、Google 等大公司。TC39 的总体考虑是，ES5 与 ES3 基本保持兼容，较大的语法修正和新功能加入，将由 JavaScript.next 完成。

运行 ECMAScript6 的途径

浏览器 (IE、Firefox、Chrome、Safari、Opera 等)

浏览器对 ES6 的支持情况 <http://kangax.github.io/compat-table/es6/>

使用 Javascript 引擎的系统（如 Node.js）

Node.js 是一个 Javascript 运行环境(runtime)。实际上它是对 Google V8 引擎进行了封装。V8 引擎执行 Javascript 的速度非常快，性能非常好。Google Chrome 浏览器就是用的 V8 引擎。

Node.js 下载地址 <https://nodejs.org/zh-cn/>，目前支持到 97%。

使用 Babel 工具将 ECMAScript6 转换为 ECMAScript5

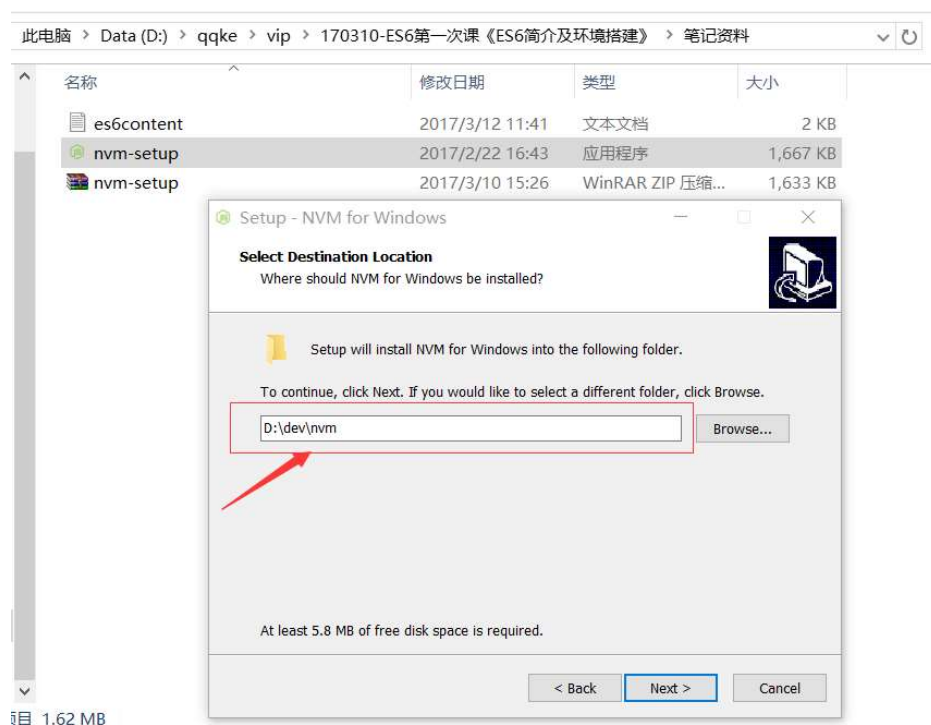
Babel 是 ES2015 语法转化器。这些转化器能让你现在就使用最新的 JavaScript 语法，而不用等待浏览器提供支持。

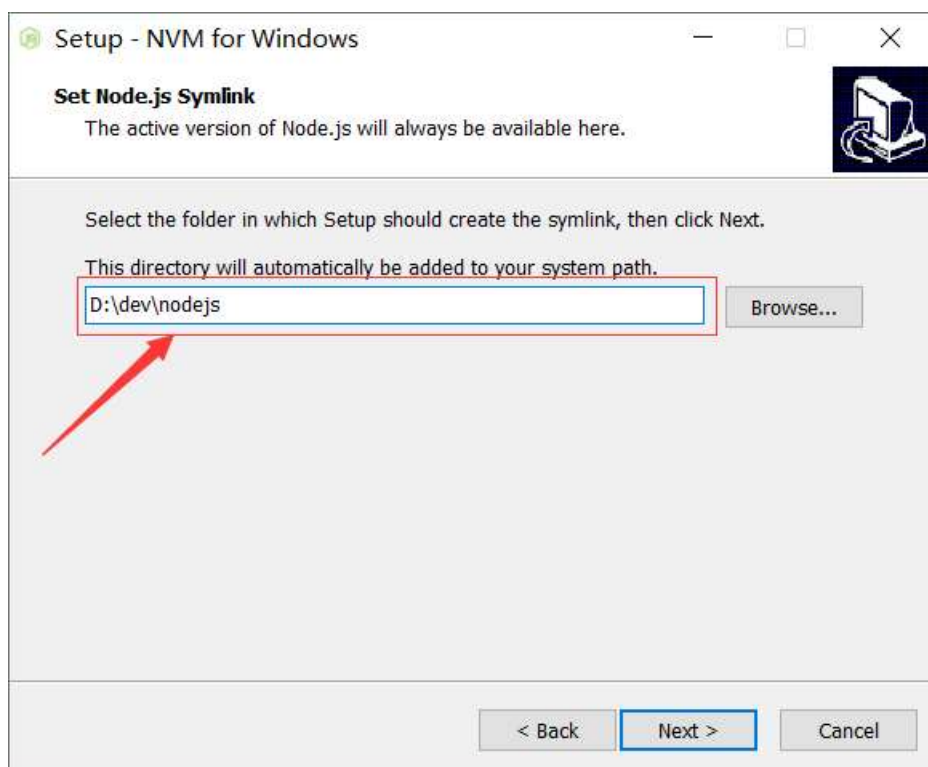
NodeJs 安装步骤

nvm（NodeJs 版本管理工具）安装

Windows 操作系统的 nvm 下载地址 <https://github.com/coreybutler/nvm-windows/releases>

安装过程





检查结果

```
C:\Users\Song>set | findstr "nvm"
NVM_HOME=D:\dev\nvm
Path=C:\Program Files (x86)\Intel\iCLS Client\;C:\Program Files\Intel\iCLS Client\;C:\Windows\system32;C:\Windows;C:\Win
dows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Intel\Intel(R) Management Engine C
omponents\DAL;C:\Program Files\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files (x86)\Intel\Intel(R) Man
agement Engine Components\IPT;C:\Program Files\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files (x86)\NV
IDIA Corporation\PhysX\Common;C:\Program Files\Git\cmd;%NVM_HOME%;%NVM_SYMLINK%;C:\Users\Song\AppData\Local\Microsoft\Wi
ndowsApps;C:\Users\Song\AppData\Local\atom\bin;D:\dev\nvm;D:\dev\nodejs
```

cmd 运行 `nvm -h` 出现如下图所示表示 nvm 安装成功

```
命令提示符
C:\Users\Song>nvm -h
Running version 1.1.2.
Usage:
  nvm arch                    : Show if node is running in 32 or 64 bit mode.
  nvm install <version> [arch] : The version can be a node.js version or "latest" for the latest stable version.
                                Optionally specify whether to install the 32 or 64 bit version (defaults to system arch
                                ).
                                Set [arch] to "all" to install 32 AND 64 bit versions.
                                Add --insecure to the end of this command to bypass SSL validation of the remote downlo
                                ad server.
  nvm list [available]        : List the node.js installations. Type "available" at the end to see what can be installe
                                d. Aliased as ls.
  nvm on                      : Enable node.js version management.
  nvm off                    : Disable node.js version management.
  nvm proxy [url]             : Set a proxy to use for downloads. Leave [url] blank to see the current proxy.
                                Set [url] to "none" to remove the proxy.
  nvm node_mirror [url]       : Set the node mirror. Defaults to https://nodejs.org/dist/. Leave [url] blank to use def
                                ault url.
  nvm npm_mirror [url]        : Set the npm mirror. Defaults to https://github.com/npm/npm/archive/. Leave [url] blank
                                to default url.
  nvm uninstall <version>     : The version must be a specific version.
  nvm use [version] [arch]    : Switch to use the specified version. Optionally specify 32/64bit architecture.
                                nvm use <arch> will continue using the selected version, but switch to 32/64 bit mode.
  nvm root [path]             : Set the directory where nvm should store different versions of node.js.
                                If <path> is not set, the current root will be displayed.
  nvm version                 : Displays the current running version of nvm for Windows. Aliased as v.
```

nvm 配置淘宝镜像（下载更快）

修改 D:\dev\nvm\settings.txt

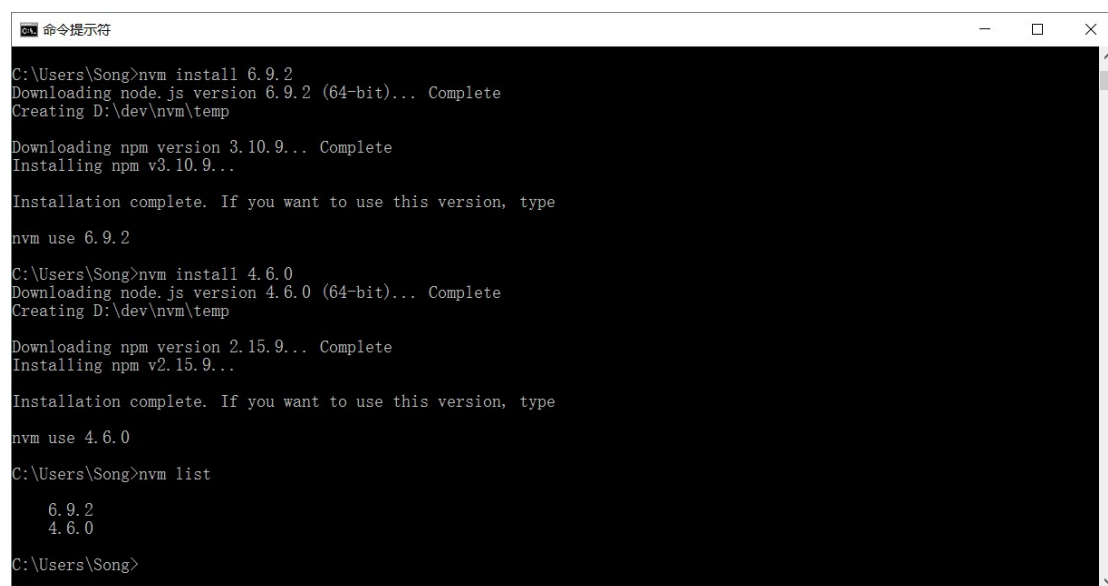
```
root: D:\dev\nvm
path: D:\dev\nodejs
node_mirror: http://npm.taobao.org/mirrors/node/
npm_mirror: https://npm.taobao.org/mirrors/npm/
```

通过 nvm 安装某版本的 nodejs 集 (node、npm (nodejs 包管理工具))

64 位系统：CMD 运行 nvm install 6.9.2
32 位系统：CMD 运行 nvm install 6.9.2 32

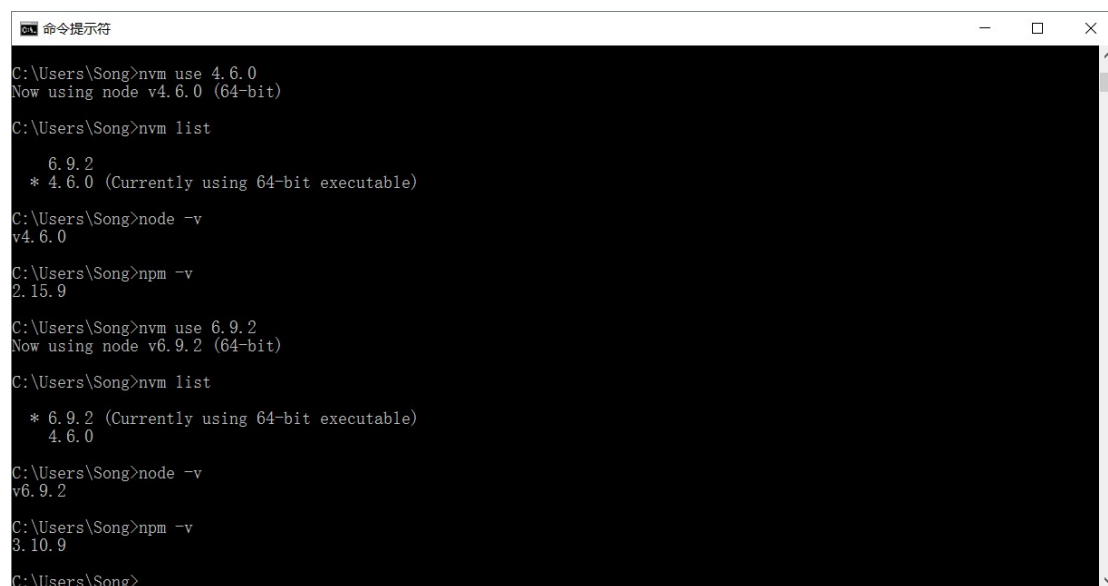
注意：其中 6.9.2 表示安装 node 版本为 6.9.2，这里 npm 是包含在 node 中的不需要单独安装

安装 6.9.2 和 4.6.0 如图所示



```
命令提示符
C:\Users\Song>nvm install 6.9.2
Downloading node.js version 6.9.2 (64-bit)... Complete
Creating D:\dev\nvm\temp
Downloading npm version 3.10.9... Complete
Installing npm v3.10.9...
Installation complete. If you want to use this version, type
nvm use 6.9.2
C:\Users\Song>nvm install 4.6.0
Downloading node.js version 4.6.0 (64-bit)... Complete
Creating D:\dev\nvm\temp
Downloading npm version 2.15.9... Complete
Installing npm v2.15.9...
Installation complete. If you want to use this version, type
nvm use 4.6.0
C:\Users\Song>nvm list
   6.9.2
   4.6.0
C:\Users\Song>
```

使用并切换 node 版本如图所示



```
C:\Users\Song>nvm use 4.6.0
Now using node v4.6.0 (64-bit)

C:\Users\Song>nvm list

   6.9.2
*  4.6.0 (Currently using 64-bit executable)

C:\Users\Song>node -v
v4.6.0

C:\Users\Song>npm -v
2.15.9

C:\Users\Song>nvm use 6.9.2
Now using node v6.9.2 (64-bit)

C:\Users\Song>nvm list

   6.9.2 (Currently using 64-bit executable)
   4.6.0

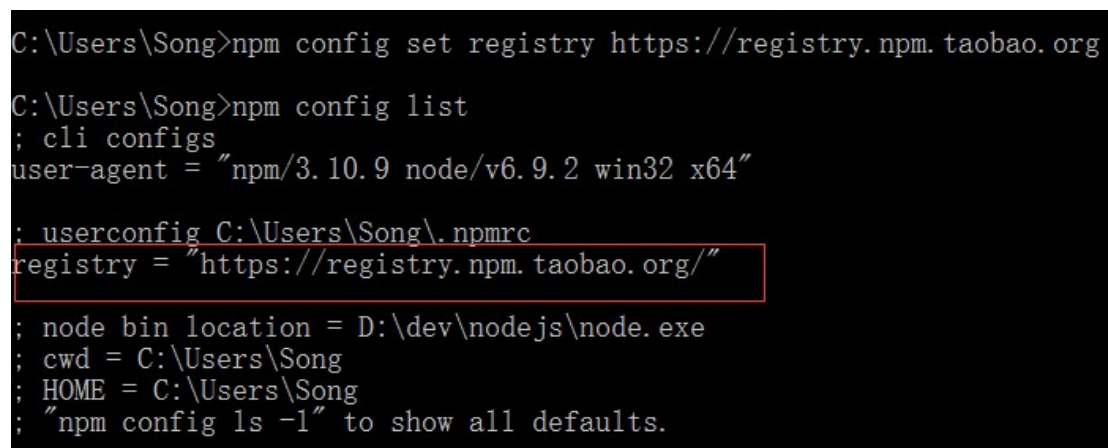
C:\Users\Song>node -v
v6.9.2

C:\Users\Song>npm -v
3.10.9

C:\Users\Song>
```

npm 设置淘宝镜像（下载更快）

npm config set registry <https://registry.npm.taobao.org>



```
C:\Users\Song>npm config set registry https://registry.npm.taobao.org

C:\Users\Song>npm config list
; cli configs
user-agent = "npm/3.10.9 node/v6.9.2 win32 x64"

; userconfig C:\Users\Song\.npmrc
registry = "https://registry.npm.taobao.org/"

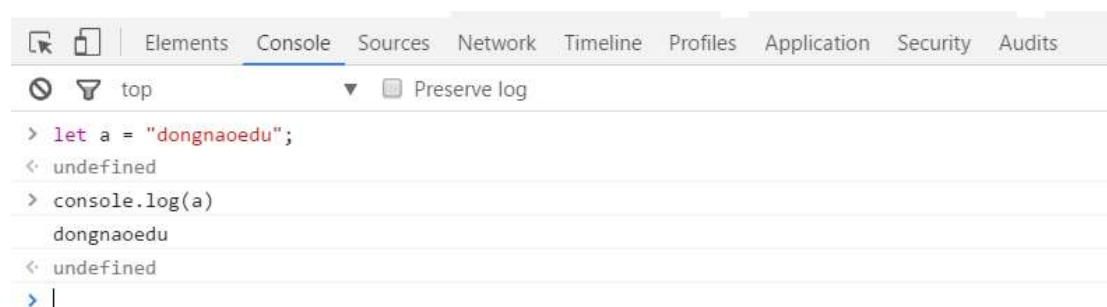
; node bin location = D:\dev\nodejs\node.exe
; cwd = C:\Users\Song
; HOME = C:\Users\Song
; "npm config ls -l" to show all defaults.
```

在不同的环境中使用 ES6

在 NodeJs 环境中运行 ES6

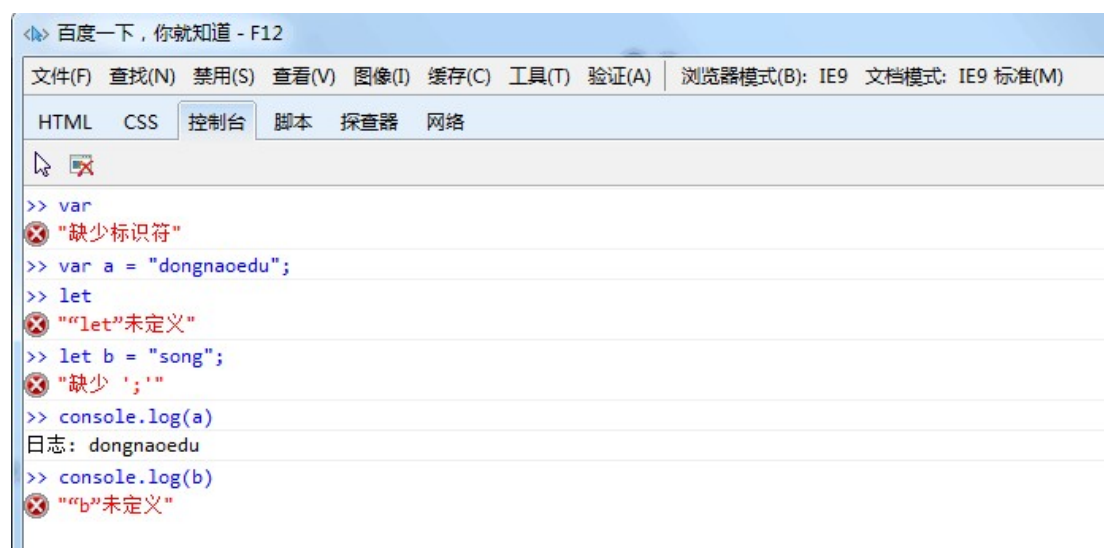
```
C:\Users\Song>node
> let school = "dongnaoedu";
undefined
> console.log(school);
dongnaoedu
undefined
>
```

在 Chrome 56 (64bit) 浏览器中运行 ES6



The screenshot shows the Chrome DevTools Console with the 'Console' tab selected. The code entered is: `> let a = "dongnaoedu";`, `< undefined`, `> console.log(a)`, `dongnaoedu`, `< undefined`, and `> |`. The 'Preserve log' checkbox is checked.

在 IE9 浏览器中运行 ES6



The screenshot shows the Internet Explorer 9 Developer Tools Console with the '控制台' (Console) tab selected. The code entered is: `>> var`, `缺少标识符`, `>> var a = "dongnaoedu";`, `>> let`, `“let”未定义`, `>> let b = "song";`, `缺少 ';'`, `>> console.log(a)`, `日志: dongnaoedu`, `>> console.log(b)`, and `“b”未定义`. The console shows several syntax errors for ES6 features not supported in IE9.

在 IE11 浏览器中运行 ES6

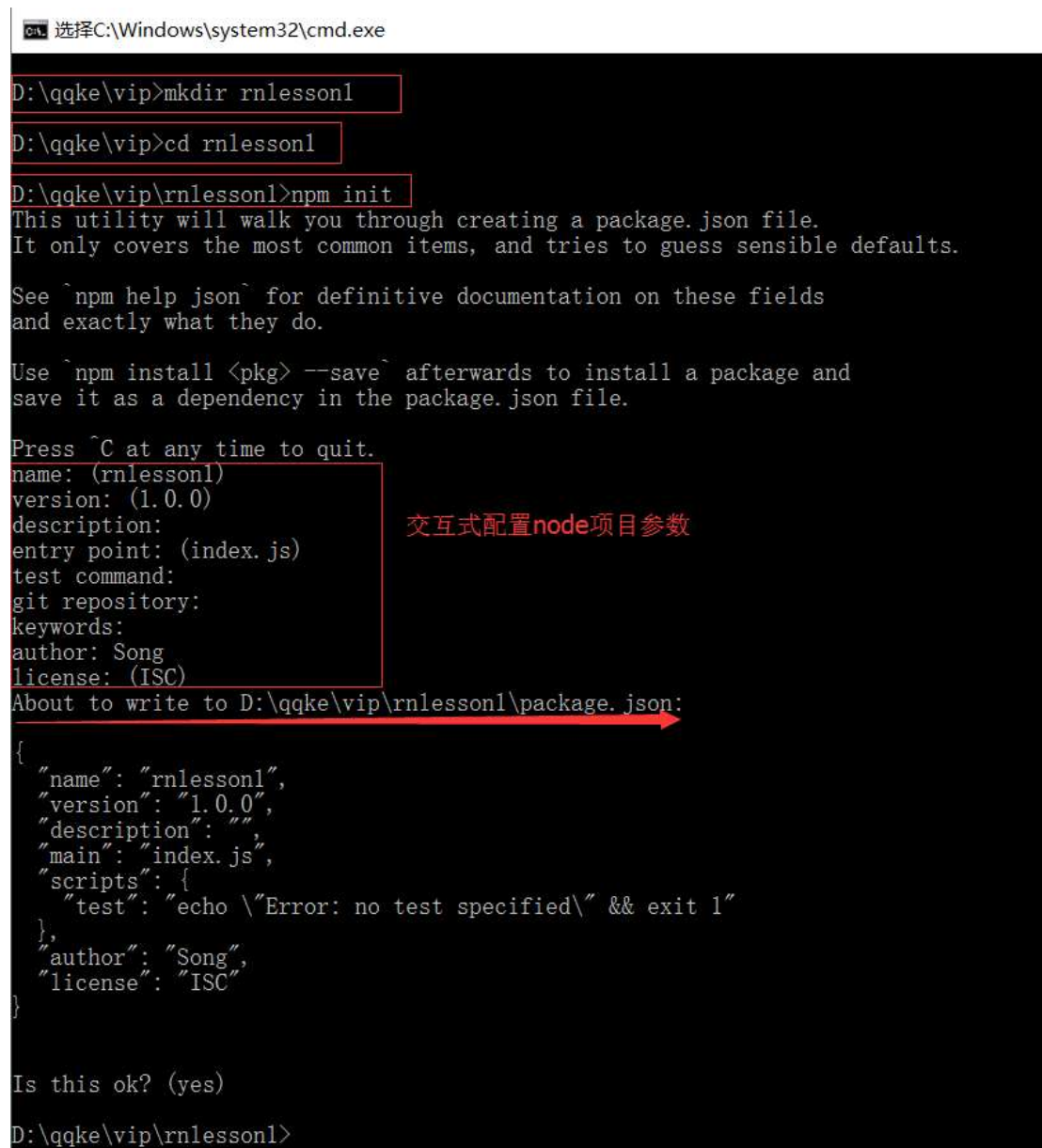


注意：报错啦！是因为 IE9 和 IE11 都不支持 ES6 的 `let` 语法。
ES6 语法支持度：IE9 为 0%；IE11 为 11%；Chrome56 为 97%。

ES6 到 ES5 的转换

使用 Babel 完成 ES6 到 ES5 的转换

使用 npm 创建一个 NodeJs 工程，步骤如图：



```
GA 选择C:\Windows\system32\cmd.exe

D:\qqke\vip>mkdir rnlesson1

D:\qqke\vip>cd rnlesson1

D:\qqke\vip\rnlesson1>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

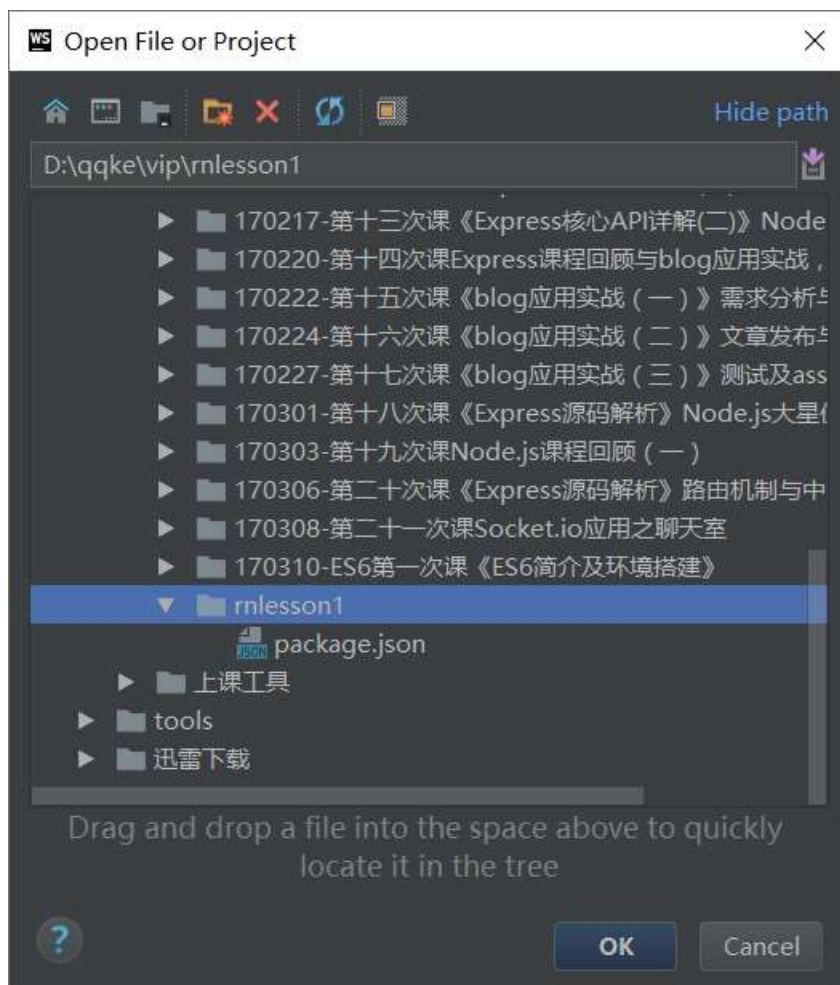
Press ^C at any time to quit.
name: (rnlesson1)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author: Song
license: (ISC)
About to write to D:\qqke\vip\rnlesson1\package.json:
{
  "name": "rnlesson1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Song",
  "license": "ISC"
}

Is this ok? (yes)

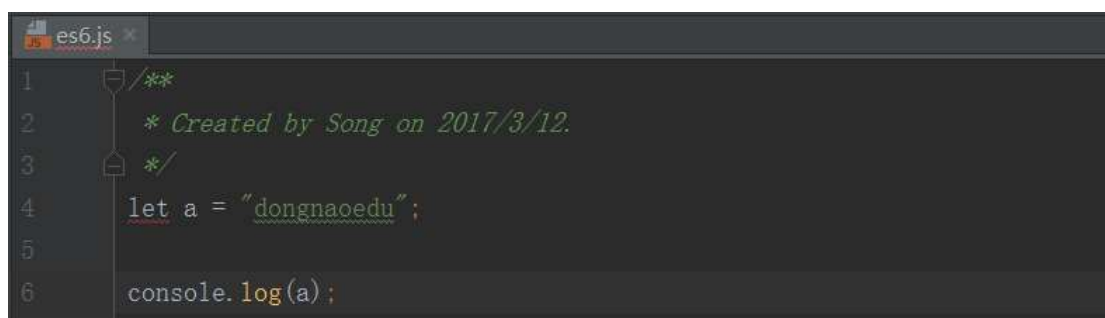
D:\qqke\vip\rnlesson1>
```

交互式配置node项目参数

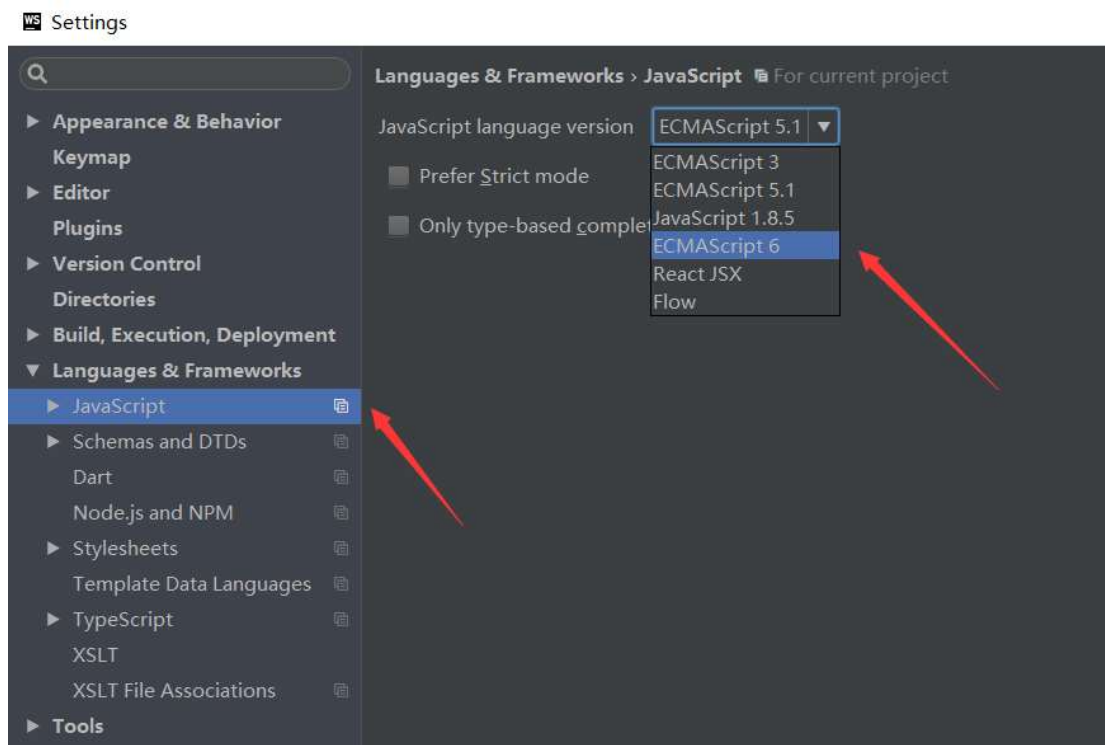
通过 webstorm 打开 rnlesson1 这个项目 File->Open



使用 webstorm 编写 ES6 代码



WebStorm 配置支持 ES6 语法规则



babel 全局安装

npm install -g babel-cli

```
D:\qqke\vip\rnlesson>npm install -g babel-cli
D:\dev\nodejs\babel-external-helpers -> D:\dev\nodejs\node_modules\babel-cli\bin\babel-external-helpers.js
D:\dev\nodejs\babel-doctor -> D:\dev\nodejs\node_modules\babel-cli\bin\babel-doctor.js
D:\dev\nodejs\babel -> D:\dev\nodejs\node_modules\babel-cli\bin\babel.js
D:\dev\nodejs\babel-node -> D:\dev\nodejs\node_modules\babel-cli\bin\babel-node.js
D:\dev\nodejs
-- babel-cli@6.23.0
+-- babel-core@6.23.1
|   +-- babel-code-frame@6.22.0
|   |   +-- chalk@1.1.3
|   |   |   +-- ansi-styles@2.2.1
|   |   |   +-- escape-string-regexp@1.0.5
```

安装转换插件（此插件定义了 ES2015 转码规则，相当于是字典的功能）：

npm install babel-preset-es2015 --save

CMD 命令行转换 ES6 到 ES5

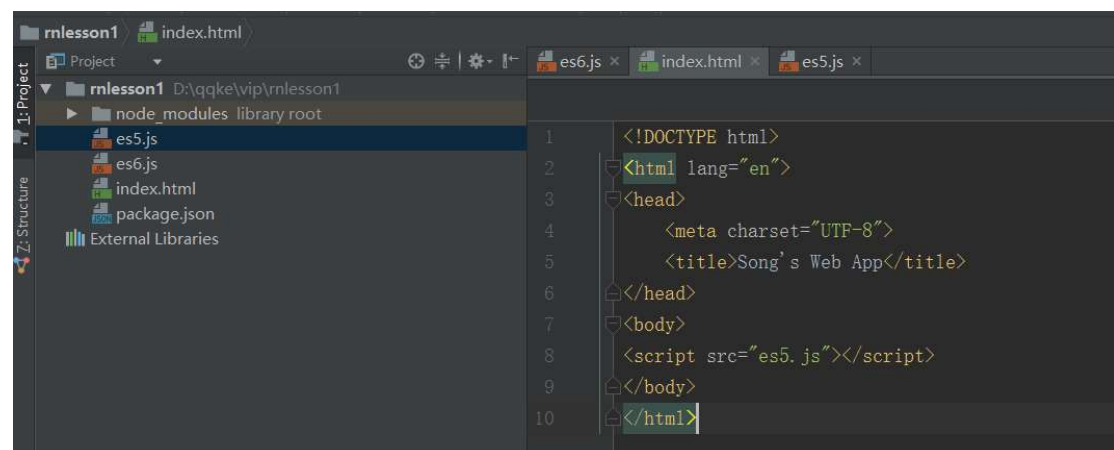
babel es6.js --out-file es5.js --presets es2015

```
D:\qqke\vip\rnlesson1>babel es6.js --out-file es5.js --presets es2015
D:\qqke\vip\rnlesson1>_
```

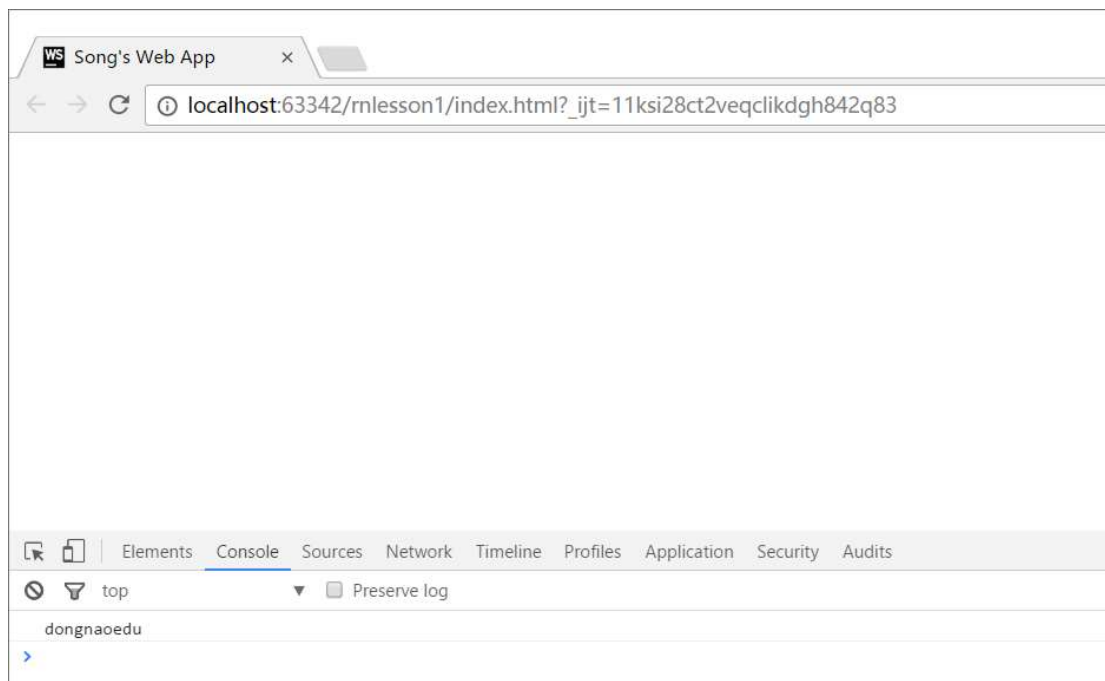
自动转换

babel es6.js -w --out-file es5.js --presets es2015

添加并编写 index.html 文件



运行结果



使用 Browsersync 实时刷新页面和 Babel-Core 实时转换 ES6

Browsersync 能让浏览器实时、快速响应您的文件更改 (html、js、css、sass、less 等) 并自动刷新页面。

安装 Browsersync

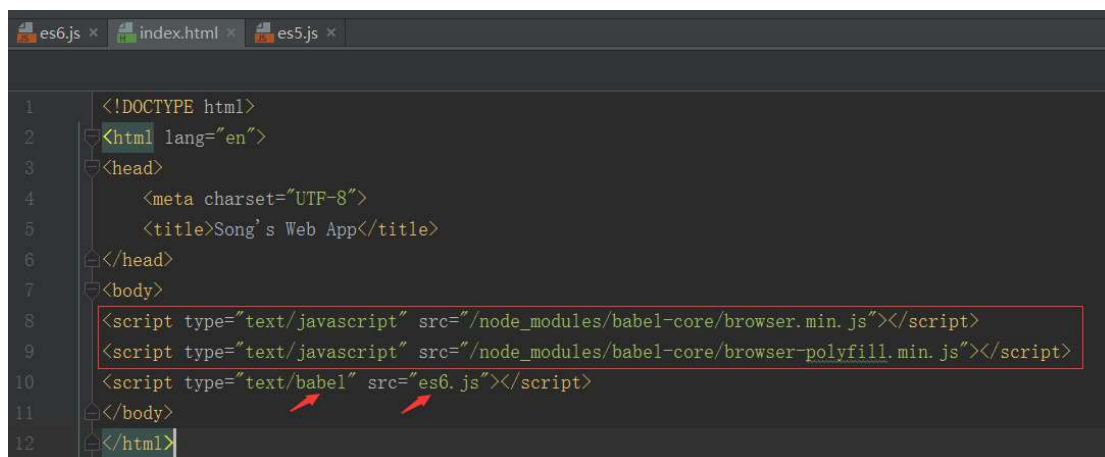
```
npm install -g browser-sync
```

安装 babel-core (在网页上实时转换 ES6 到 ES5)

```
npm install babel-core@5 --save
```

HTML 页面引入 babel-core 实时转换 js 代码

```
/node_modules/babel-core/browser.min.js  
/node_modules/babel-core/browser-polyfill.min.js
```

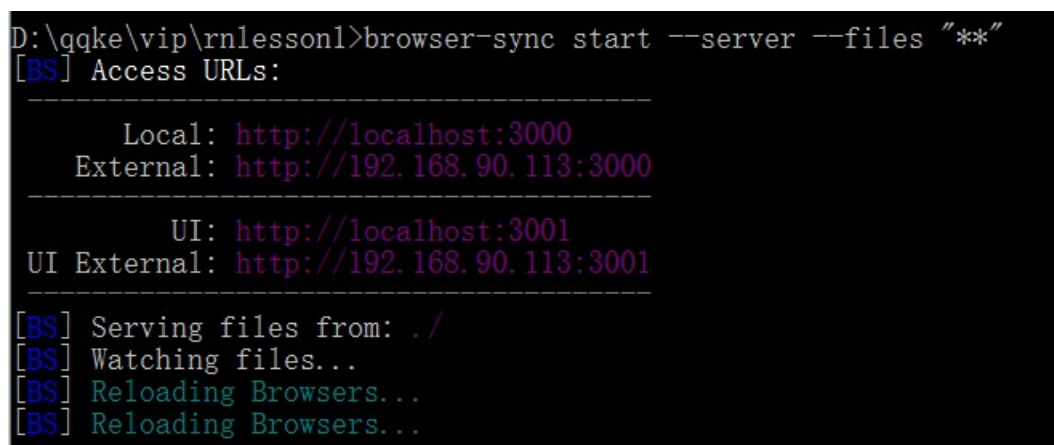


```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <title>Song's Web App</title>
6  </head>
7  <body>
8    <script type="text/javascript" src="/node_modules/babel-core/browser.min.js"></script>
9    <script type="text/javascript" src="/node_modules/babel-core/browser-polyfill.min.js"></script>
10   <script type="text/babel" src="es6.js"></script>
11 </body>
12 </html>
  
```

在当前项目目录下启动 Browsersync, 开始监听

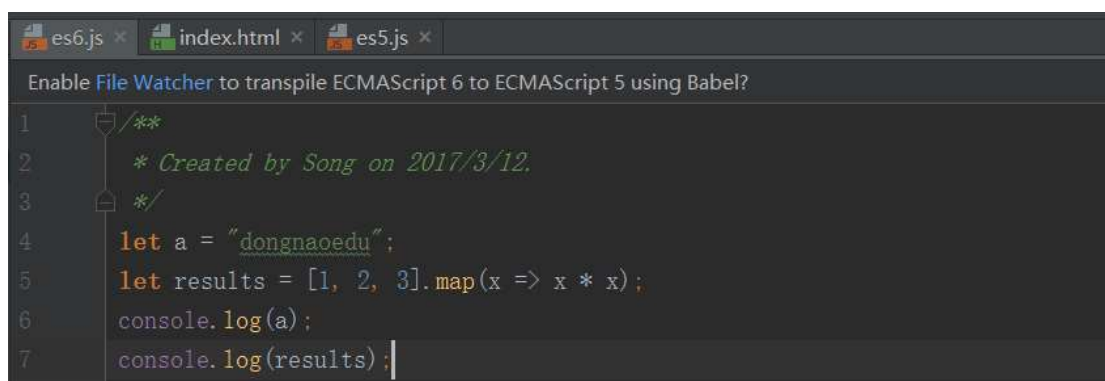
```
browser-sync start --server --files "**"
```



```

D:\qqke\vip\rnlesson1>browser-sync start --server --files "**"
[BS] Access URLs:
-----
    Local: http://localhost:3000
  External: http://192.168.90.113:3000
-----
    UI: http://localhost:3001
  UI External: http://192.168.90.113:3001
-----
[BS] Serving files from: ./
[BS] Watching files...
[BS] Reloading Browsers...
[BS] Reloading Browsers...
  
```

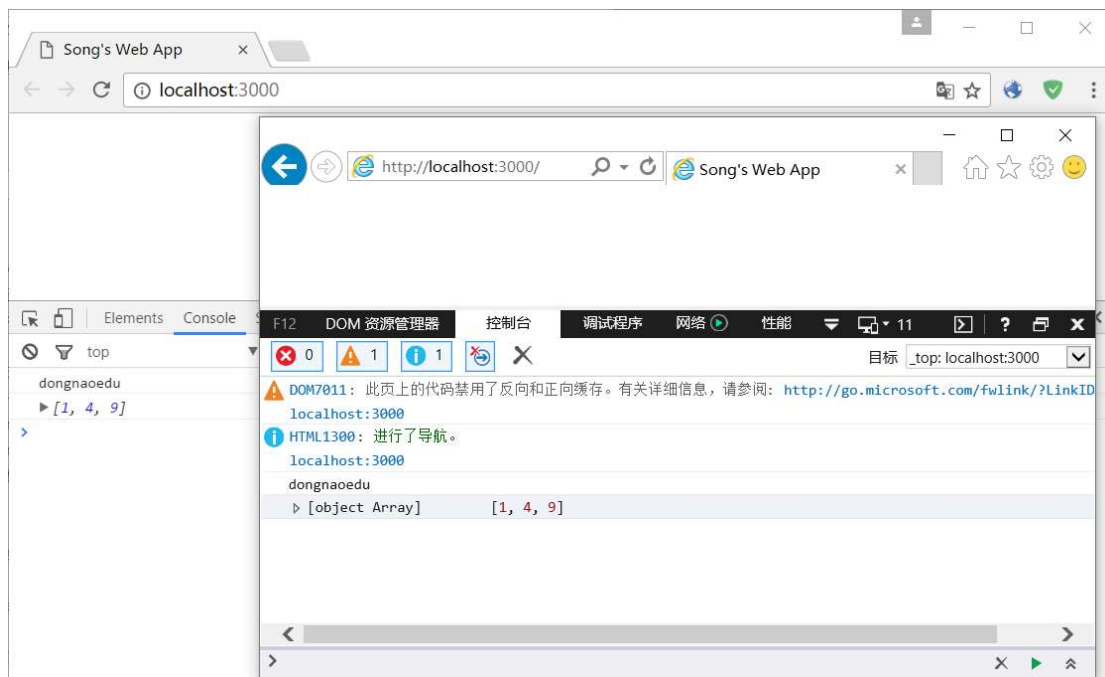
修改 es6.js



```

1  /**
2   * Created by Song on 2017/3/12.
3   */
4   let a = "dongnaoedu";
5   let results = [1, 2, 3].map(x => x * x);
6   console.log(a);
7   console.log(results);
  
```

页面自动刷新，结果如下：



这里的 ES6 到 ES5 的转换时在浏览器端实现的；
可同时刷新 Chrome 和 IE11，现在 ES6 也可以在各种浏览器上使用了。

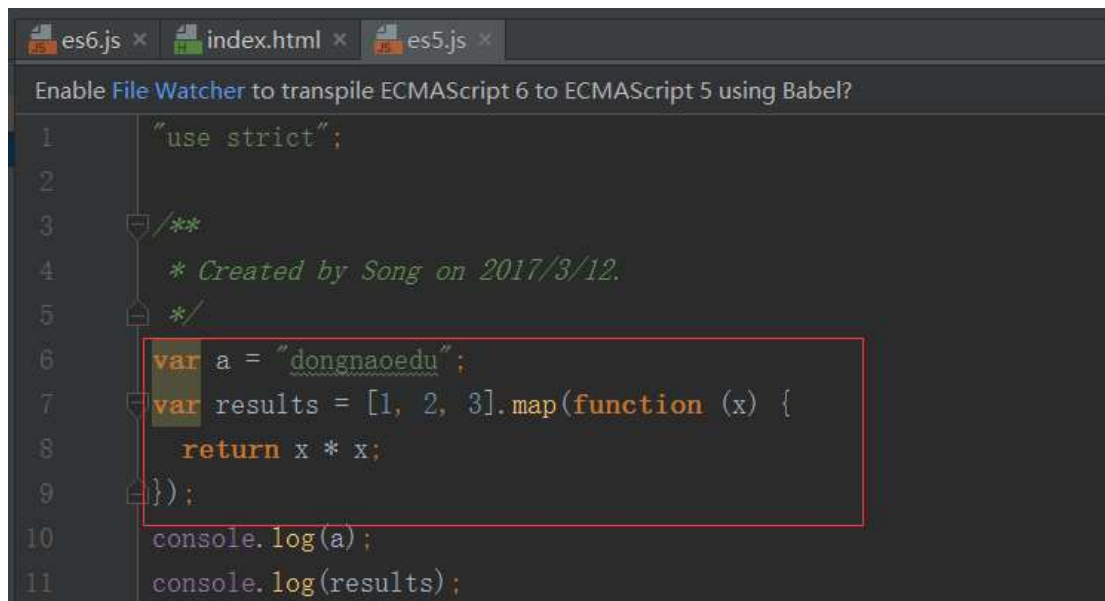
回到命令行的方式再来看看：

CMD 运行 `babel es6.js --out-file es5.js --presets es2015`

```
C:\Windows\system32\cmd.exe

D:\qqke\vip\rnlesson1>babel es6.js --out-file es5.js --presets es2015
D:\qqke\vip\rnlesson1>
```


查看编译后的 es5.js



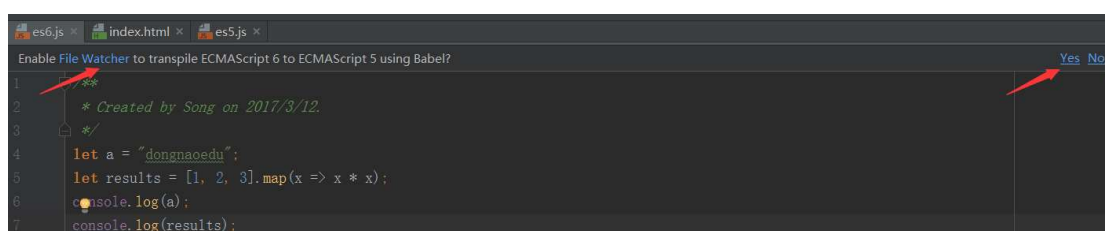
```

1  "use strict";
2
3  /**
4   * Created by Song on 2017/3/12.
5   */
6  var a = "dongnaoedu";
7  var results = [1, 2, 3].map(function (x) {
8     return x * x;
9  });
10 console.log(a);
11 console.log(results);
    
```

上面讲到了 2 两种 es6->es5 的转换：babel 命令行方式和浏览器引入 babel-core 方式

还有第三种方式，那就是 webstorm 自带的 File Watcher 功能

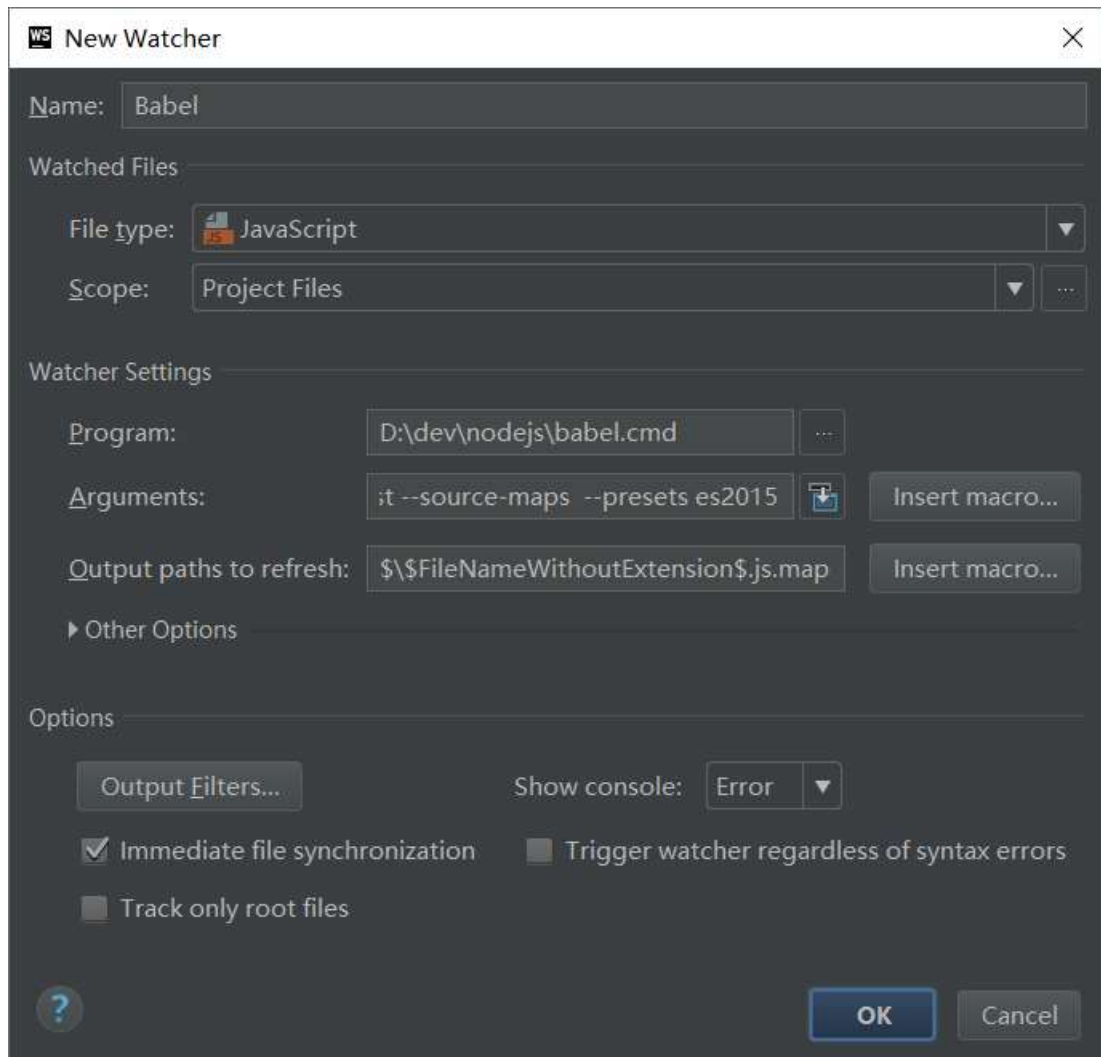
自动配置 File Watcher



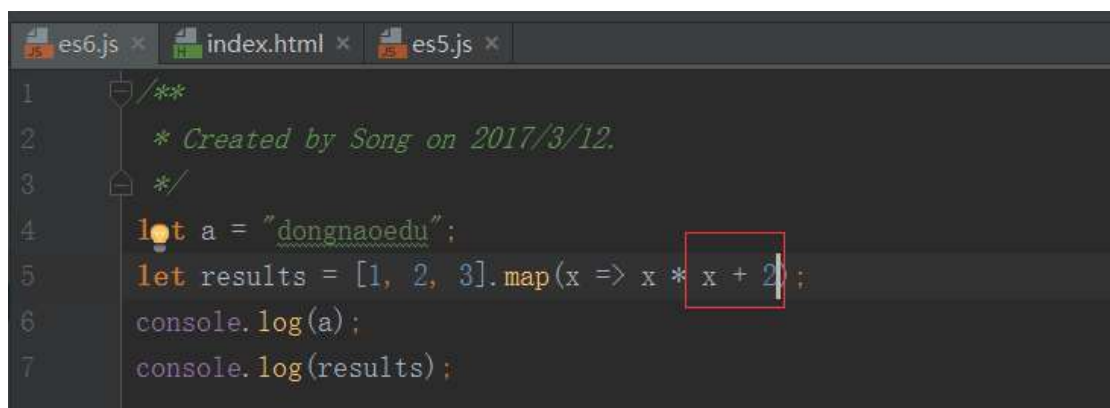
```

1  "use strict";
2
3  /**
4   * Created by Song on 2017/3/12.
5   */
6  let a = "dongnaoedu";
7  let results = [1, 2, 3].map(x => x * x);
8  console.log(a);
9  console.log(results);
    
```

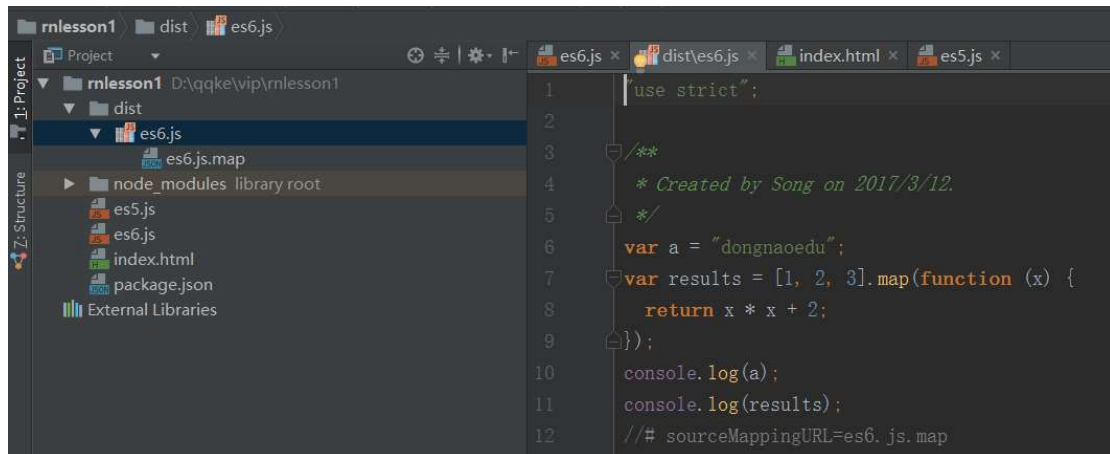
注意：当输入了 es6 语法规则的代码后，webstorm 会自动弹出这种提示。



再次修改 es6.js

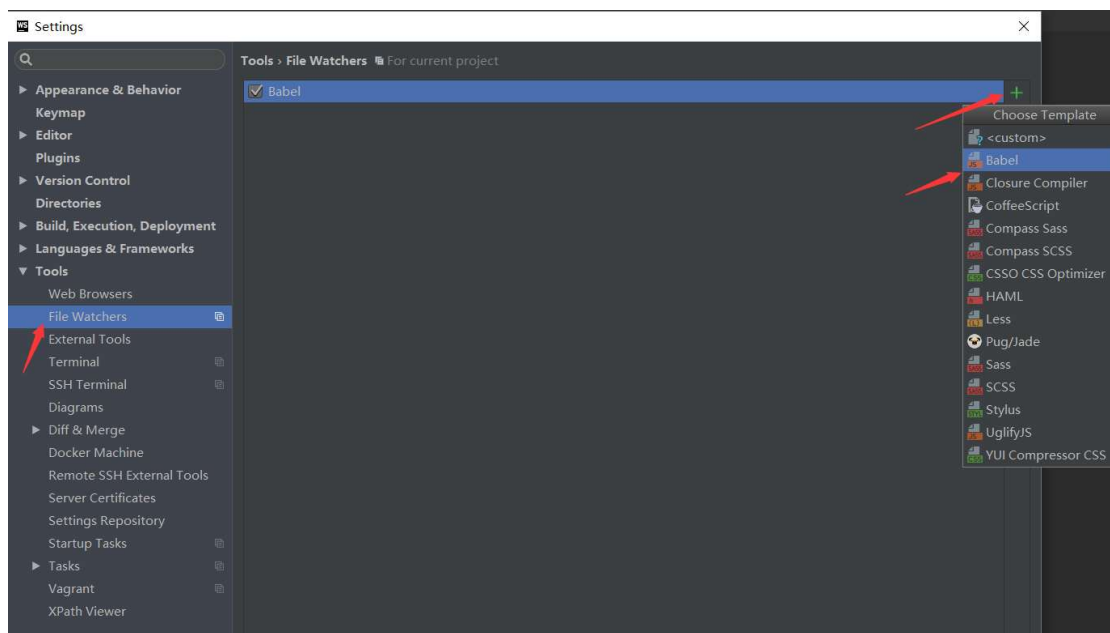


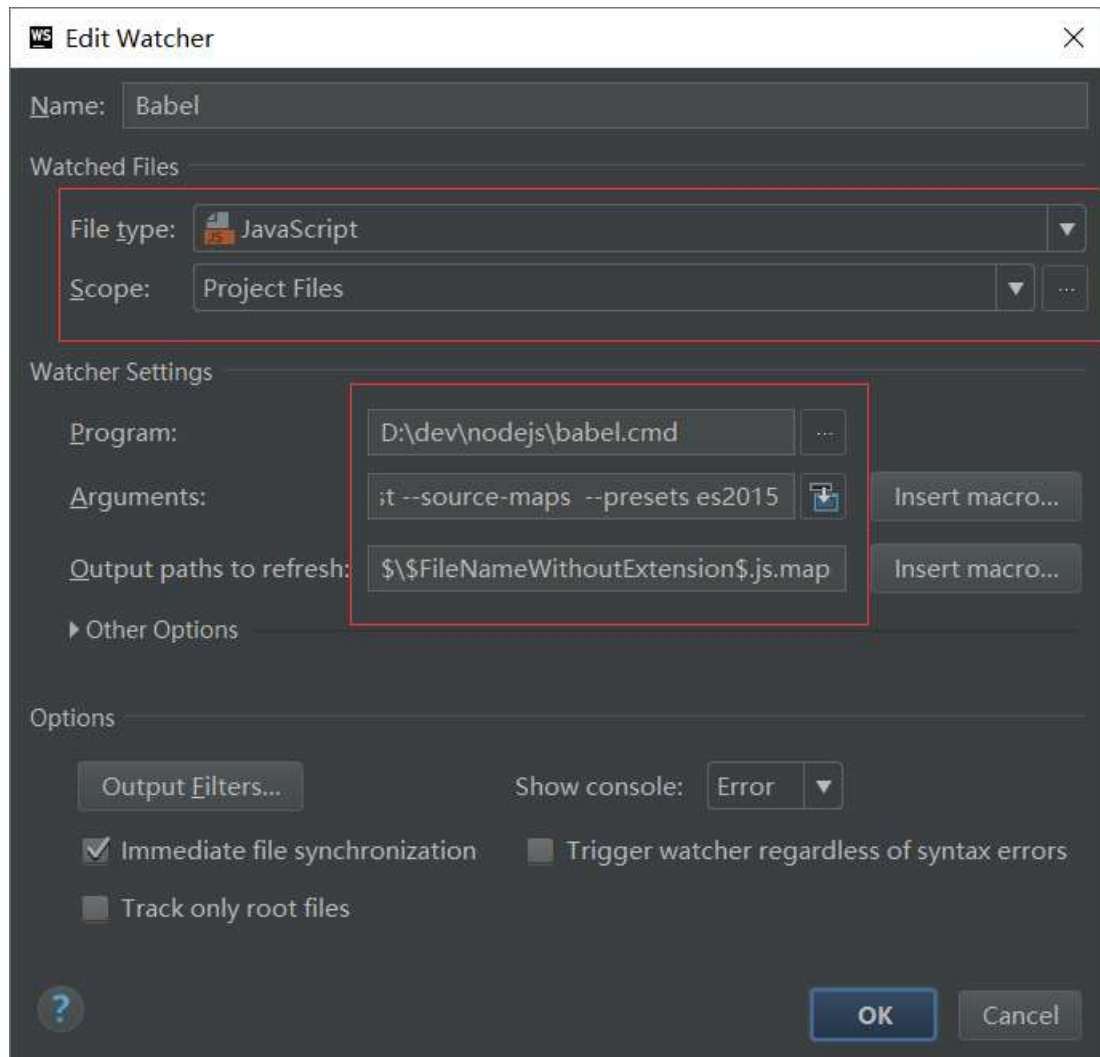
保存 es6.js 通过 webstorm 的 File Watcher 功能会自动生成转换后的 es6.js 和 es6.js.map



手动配置 File Watcher :

File->Settings





`$FilePathRelativeToProjectRoot$ --out-dir dist --source-maps --presets es2015`

`dist\${FileDirRelativeToProjectRoot}\${FileNameWithoutExtension}.js:dist\${FileDirRelativeToProjectRoot}\${FileNameWithoutExtension}.js.map`