

Babel 入门教程

作者： 阮一峰

分享

日期： 2016年1月25日

（说明：本文选自我的新书[《ES6 标准入门（第二版）》](#)的第一章[《ECMAScript 6简介》](#)）

[Babel](#)是一个广泛使用的转码器，可以将ES6代码转为ES5代码，从而在现有环境执行。



这意味着，你可以现在就用ES6编写程序，而不用担心现有环境是否支持。下面是一个例子。

```
// 转码前
input.map(item => item + 1);

// 转码后
input.map(function (item) {
  return item + 1;
});
```

上面的原始代码用了箭头函数，这个特性还没有得到广泛支持，Babel将其转为普通函数，就能在现有的JavaScript环境执行了。

一、配置文件.babelrc

Babel的配置文件是 `.babelrc`，存放在项目的根目录下。使用Babel的第一步，就是配置这个文件。

该文件用来设置转码规则和插件，基本格式如下。

```
{
  "presets": [],
  "plugins": []
}
```

`presets` 字段设定转码规则，官方提供以下的规则集，你可以根据需要安装。

```
# ES2015转码规则
$ npm install --save-dev babel-preset-es2015

# react转码规则
$ npm install --save-dev babel-preset-react

# ES7不同阶段语法提案的转码规则（共有4个阶段），选装一个
$ npm install --save-dev babel-preset-stage-0
$ npm install --save-dev babel-preset-stage-1
$ npm install --save-dev babel-preset-stage-2
$ npm install --save-dev babel-preset-stage-3
```

然后，将这些规则加入 `.babelrc`。

```
{
  "presets": [
    "es2015",
    "react",
    "stage-2"
  ],
  "plugins": []
}
```

注意，以下所有Babel工具和模块的使用，都必须先写好 `.babelrc`。

二、命令行转码babel-cli

Babel提供 `babel-cli` 工具，用于命令行转码。

它的安装命令如下。

```
$ npm install --global babel-cli
```

基本用法如下。

```
# 转码结果输出到标准输出
$ babel example.js

# 转码结果写入一个文件
# --out-file 或 -o 参数指定输出文件
$ babel example.js --out-file compiled.js
# 或者
$ babel example.js -o compiled.js

# 整个目录转码
# --out-dir 或 -d 参数指定输出目录
$ babel src --out-dir lib
# 或者
$ babel src -d lib

# -s 参数生成source map文件
$ babel src -d lib -s
```

上面代码是在全局环境下，进行Babel转码。这意味着，如果项目要运行，全局环境必须有Babel，也就是说项目产生了对环境的依赖。另一方面，这样做也无法支持不同项目使用不同版本的Babel。

一个解决办法是将 `babel-cli` 安装在项目之中。

```
# 安装
$ npm install --save-dev babel-cli
```

然后，改写 `package.json` 。

```
{
  // ...
  "devDependencies": {
    "babel-cli": "^6.0.0"
  },
  "scripts": {
    "build": "babel src -d lib"
  },
}
```

转码的时候，就执行下面的命令。

```
$ npm run build
```

三、babel-node

`babel-cli` 工具自带一个 `babel-node` 命令，提供一个支持ES6的REPL环境。它支持Node的REPL环境的所有功能，而且可以直接运行ES6代码。

它不用单独安装，而是随 `babel-cli` 一起安装。然后，执行 `babel-node` 就进入REPL环境。

```
$ babel-node
> (x => x * 2)(1)
2
```

`babel-node` 命令可以直接运行ES6脚本。将上面的代码放入脚本文件 `es6.js`，然后直接运行。

```
$ babel-node es6.js
2
```

`babel-node` 也可以安装在项目中。

```
$ npm install --save-dev babel-cli
```

然后，改写 `package.json`。

```
{
  "scripts": {
    "script-name": "babel-node script.js"
  }
}
```

上面代码中，使用 `babel-node` 替代 `node`，这样 `script.js` 本身就不用做任何转码处理。

四、babel-register

`babel-register` 模块改写 `require` 命令，为它加上一个钩子。此后，每当使用 `require` 加载 `.js`、`.jsx`、`.es` 和 `.es6` 后缀名的文件，就会先用 Babel 进行转码。

```
$ npm install --save-dev babel-register
```

使用时，必须首先加载 `babel-register`。

```
require("babel-register");
require("./index.js");
```

然后，就不需要手动对 `index.js` 转码了。

需要注意的是，`babel-register` 只会对 `require` 命令加载的文件转码，而不会对当前文件转码。另外，由于它是实时转码，所以只适合在开发环境使用。

五、babel-core

如果某些代码需要调用 Babel 的 API 进行转码，就要使用 `babel-core` 模块。

安装命令如下。

```
$ npm install babel-core --save
```

然后，在项目中就可以调用 `babel-core`。

```
var babel = require('babel-core');

// 字符串转码
babel.transform('code();', options);
// => { code, map, ast }

// 文件转码（异步）
babel.transformFile('filename.js', options, function(err, result) {
  result; // => { code, map, ast }
});

// 文件转码（同步）
babel.transformFileSync('filename.js', options);
// => { code, map, ast }

// Babel AST转码
babel.transformFromAst(ast, code, options);
// => { code, map, ast }
```

配置对象 `options`，可以参看官方文档<http://babeljs.io/docs/usage/options/>。

下面是一个例子。

```
var es6Code = 'let x = n => n + 1';
var es5Code = require('babel-core')
  .transform(es6Code, {
    presets: ['es2015']
  })
  .code;
// '"use strict";\n\nvar x = function x(n) {\n  return n + 1;\n};'
```

上面代码中，`transform` 方法的第一个参数是一个字符串，表示需要转换的ES6代码，第二个参数是转换的配置对象。

六、babel-polyfill

Babel默认只转换新的JavaScript句法（syntax），而不转换新的API，比如Iterator、Generator、Set、Maps、Proxy、Reflect、Symbol、Promise等全局对象，以及一些定

义在全局对象上的方法（比如 `Object.assign`）都不会转码。

举例来说，ES6在 `Array` 对象上新增了 `Array.from` 方法。Babel就不会转码这个方法。如果想让这个方法运行，必须使用 `babel-polyfill`，为当前环境提供一个垫片。

安装命令如下。

```
$ npm install --save babel-polyfill
```

然后，在脚本头部，加入如下一行代码。

```
import 'babel-polyfill';  
// 或者  
require('babel-polyfill');
```

Babel默认不转码的API非常多，详细清单可以查看 `babel-plugin-transform-runtime` 模块的[definitions.js](#)文件。

七、浏览器环境

Babel也可以用于浏览器环境。但是，从Babel 6.0开始，不再直接提供浏览器版本，而是要用构建工具构建出来。如果你没有或不想使用构建工具，可以通过安装5.x版本的 `babel-core` 模块获取。

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

运行上面的命令以后，就可以在当前目录的 `node_modules/babel-core/` 子目录里面，找到 `babel` 的浏览器版本 `browser.js`（未精简）和 `browser.min.js`（已精简）。

然后，将下面的代码插入网页。

```
<script src="node_modules/babel-core/browser.js"></script>  
<script type="text/babel">  
  // Your ES6 code  
</script>
```

上面代码中，`browser.js` 是Babel提供的转换器脚本，可以在浏览器运行。用户的ES6脚本放在 `script` 标签之中，但是要注重 `type="text/babel"`。

另一种方法是使用**babel-standalone**模块提供的浏览器版本，将其插入网页。

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/babel-standalone/6.4.4">
<script type="text/babel">
// Your ES6 code
</script>
```

注意，网页中实时将ES6代码转为ES5，对性能会有影响。生产环境需要加载已经转码完成的脚本。

下面是如何将代码打包成浏览器可以使用的脚本，以 `Babel` 配合 `Browserify` 为例。首先，安装 `babelify` 模块。

```
$ npm install --save-dev babelify babel-preset-es2015
```

然后，再用命令行转换ES6脚本。

```
$ browserify script.js -o bundle.js \
-t [ babelify --presets [ es2015 react ] ]
```

上面代码将ES6脚本 `script.js`，转为 `bundle.js`，浏览器直接加载后者就可以了。

在 `package.json` 设置下面的代码，就不用每次命令行都输入参数了。

```
{
  "browserify": {
    "transform": [ ["babelify", { "presets": ["es2015"] } ] ]
  }
}
```

八、在线转换

Babel提供一个**REPL在线编译器**，可以在线将ES6代码转为ES5代码。转换后的代码，

可以直接作为ES5代码插入网页运行。

九、与其他工具的配合

许多工具需要Babel进行前置转码，这里举两个例子：ESLint和Mocha。

[ESLint](#) 用于静态检查代码的语法和风格，安装命令如下。

```
$ npm install --save-dev eslint babel-eslint
```

然后，在项目根目录下，新建一个配置文件 `.eslint`，在其中加入 `parser` 字段。

```
{
  "parser": "babel-eslint",
  "rules": {
    ...
  }
}
```

再在 `package.json` 之中，加入相应的 `scripts` 脚本。

```
{
  "name": "my-module",
  "scripts": {
    "lint": "eslint my-files.js"
  },
  "devDependencies": {
    "babel-eslint": "...",
    "eslint": "..."
  }
}
```





[Mocha](#) 则是一个测试框架，如果需要执行使用ES6语法的测试脚本，可以修改 `package.json` 的 `scripts.test`。

```
"scripts": {
  "test": "mocha --ui qunit --compilers js:babel-core/register"
}
```

上面命令中，`--compilers` 参数指定脚本的转码器，规定后缀名为 `js` 的文件，都需要使用 `babel-core/register` 先转码。

(完)

文档信息

- 版权声明：自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期：2016年1月25日
- 更多内容：档案 » JavaScript
- 购买文集： 《如何变得有思想》
- 社交媒体： twitter,  weibo
- Feed订阅：

相关文章

■ 2016.04.12: [跨域资源共享 CORS 详解](#)

CORS是一个W3C标准，全称是"跨域资源共享"（Cross-origin resource sharing）。

■ 2016.04.08: [浏览器同源政策及其规避方法](#)

浏览器安全的基石是"同源政策"（same-origin policy）。很多开发者都知道这一点，但了解得不全面。

■ 2016.03.12: [Node 应用的 Systemd 启动](#)

前面的文章介绍了 Systemd 的操作命令和基本用法，今天给出一个实例，如何使用 Systemd 启动一个 Node 应用。

■ 2016.02.13: [React 测试入门教程](#)

越来越多的人，使用React开发Web应用。它的测试就成了一个大问题。

联系方式 | ruanyifeng.com 2003 - 2016