babel

# babel工作原理

babel的功能十分纯粹，以字符串的形式将源代码传给他，它就会返回一段新的代码字符串。（既不会运行代码，也不会将多个代码打包到一起，它就是一个编译器。 输入：ES6+ 输出： ES5）

解析：将代码字符串解析成抽象语法树

变换： 将抽象语法树进行变换操作

再建： 根据变换后的抽象语法树再生成代码字符串

# 运行babel

## .babelrc文件

注意： .babelrc里配置的presets 和plugin都是在第二步（变换）时使用的。

### presets(预设) -语法转义器

可以按需安装

ES2015转码规则

babel-preset-es2015

react转码规则

bable-preset-react

ES7 不同阶段语法提案的转码规则

bable-preset-state-0

bable-preset-state-1

bable-preset-state-2

bable-preset-state-4

#### babel-preset-env

根据运行环境选取 babel 和 polyfill

{

  "presets": [

    ["env", {

      "targets": {

        "node": "current"

      }

    }]

  ]

}

{

  "presets": [

    ["env", {

      "targets": {

        "browsers": ["last 2 versions", "safari >= 7"]

      }

    }]

  ]

}

## 插件-补丁转义器 plugins

主要是转译js新增的api和全局对象。

如

babel-plugin-transform-runtime

支持promise， Object.assign等等

## babel-cli （命令行转码）

文件转码

babel example.js -o compiled.js

文件夹中所有文件转码

babel src -d lib

注意：需要配置 .babelrc

### babel-node ( balel-cli自带命令)

这个命令可以直接运行ES6脚本

## bable-register

这个模块改写了require 命令，为它加上了一个钩子，每当文件中使用require加载 .js

.jsx .es .es6后缀的文件时，就会先用bable进行加载。

需要首先加载 babel-register

require（‘bable-reqister’）

注意：只会对require命令加载的文件转码，而不会对当前文件转码（实时编码，只适合开发环境）

## babel-core（使用babel的API进行转码）

## babel-polyfill (转换新API)

babel只能转换语法（如箭头函数） ，而使用babel-polyfill来支持新的全局变量（Promise， String.padStat ,Set.........），

相当于添加一个垫片

只要在入口文件中引入即可（webpack 中 在entry中引入即可 ）

entry: {

build: ['babel-polyfill', './src/main.js']

},

与其它工具的配合

如 eslint

下载babel-eslint

在.eslint配置文件中加入parser字段，再在package.json中的script加入

{

script：{

“lint”:”eslint my.js “

}

}

mocha

{

script：{

“test”:”mocha --ui qunit --compilers js:babel-core/register

“

}

}

jsx 和 flow

babel可以转换jsx语法并删除类型注释，可以和babel-sublime包一起用（语法高亮）

babel-preset-react