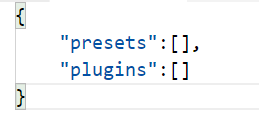
Babel是ES6转ES5的转换器，使用Babel的第一步是在根目录下新建一个.babelrc的配置文件。要用babel第一步最好就写这个配置文件。



格式采用json。

presets字段设置转码规则。官方提供了一些规则集，可以根据需要安装，书中使用的是“latest”、“react”、“stage-2”详见P5。

// 最新转码规则

npm install babel-preset-latest --save-dev

// react转码规则

npm install babel-preset-react --save-dev

// 不同阶段语法提案的转码规则，一共四个阶段，从0开始，最大是3。

// 这里安装的是第2阶段的转码规则。

npm install babel-preset-stage-2 --save-dev

下载安装完转码规则后，配置.babelrc文件，将规则加入presets字段中（JSON叫做字段不是属性）。

Babel还提供了一个命令行转码工具babel-cli，用于命令行的转码。

npm install babel-cli --save-dev

之后就可以利用命令行转码了。

babel index.js

babel index.js --out-file es6.js

// 或者

babel index.js -o es6.js

// babel转码整个目录

babel src --out-dir lib

// 或者

babel src -d lib

Babel-register改写了require命令，为他加上了一个钩子，伺候require.js .jsx .es .es6文件的时候就会自动用Babel进行转码处理。

安装babel-register 不谈了

值得注意的是，register只会对require命令加载的文件进行转码处理。而不会对当前文件进行转码。

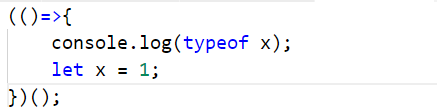
因为迫于练习，而且经过es-checker检测，浏览器支持93%的ES6语法！

所以之后需要找babel的详细教程补充一下知识点。

浏览了ES6书的第二章。复习的时候需要在过一遍，这里只写难的不容易记住的东西。

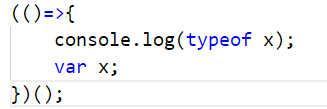
## 暂时性死死区

一个块级作用域一开始就对let const声明的变量形成了封闭的作用域。因此在同一个作用域下，在let和const之前使用变量就会报错，即使是可以容忍变量不存在的错误的typeof也会报错。在声明变量之前就形成了一个死区！





但是改为var就不会出现问题





这里看一下博客：

<https://www.cnblogs.com/jiasm/p/9027389.html>

还有很多隐藏的不容易发现的暂时性死区：

((x=y,y=1)=>{

return;

})()

函数参数默认值的写法只在ES6中有。

在x赋默认值y之前，y并没有被声明，因此会出现报错。



但是如果调换x,y就不会出现问题。

**暂时性死区和let 、const不会出现变量提升。**