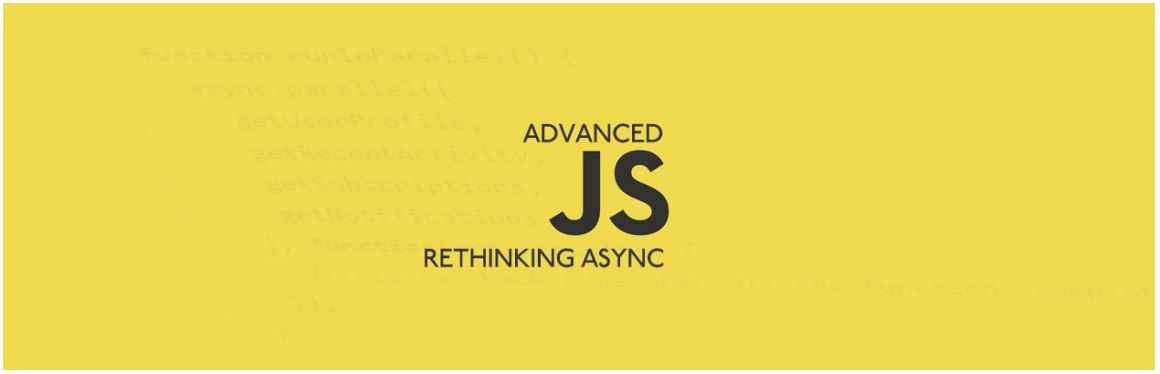
如何优雅地写js异步代码(2)

javascript (/lib/tag/javascript) 2016-04-22 16:22:59 发布



Rock with async/await

本篇文章是作为上一篇 (http://iammapping.com/write-js-async-gracefully/)的续集，考虑到第一篇的篇幅，还有更重要的一点就是上一篇讲的内容已经可以直接应用在最新版本的Node.js和一些高级浏览器（Chrome，FF）中，具体兼容性可参考：https://kangax.github.io/compat-table/es6/ (https://kangax.github.io/compat-table/es6/)。

而这一篇讲的内容，是ECMAScript 2016（ES7）的 `async/await` (https://github.com/tc39/ecmascript-asyncawait)特性，目前的兼容性可参考：http://kangax.github.io/compat-table/esnext/#test-async_functions (http://kangax.github.io/compat-table/esnext/#test-async_functions)，虽然现在来看还不是非常乐观，但是我们可以通过第三方的代码转换工具（如 `Traceur` 和 `Babel` ），将这些新特性的代码转换为当前环境可运行的代码。

一个简单的例子

实现同步的sleep，同步的代码看起来应该是下面的样子：

```
function sleep(timeout) {
    setTimeout(function() {}, timeout);
}

function main() {
    console.time('how long did I sleep');
    sleep(3000);
    console.timeEnd('how long did I sleep');
}

main();
// how long did I sleep: 0ms
```

但是在js中的执行结果却是 `0ms` ，这不是我们预期的呀。

改造

按照这种同步的代码流程，怎么样才能输出 `3000ms` 呢？看过上一篇 (http://iammapping.com/write-js-async-gracefully/)文章的童鞋应该很快就能想到使用 `Generator` 和 `yield` ，没看过的童鞋建议先看完上一篇 (http://iammapping.com/write-js-async-gracefully/)再回来。

sleep(5min)

好，我就当你们都回来了，接下来就说说如何使用 `async/await` 实现“同步”的sleep。

`await` 期望的值是一个 `Promise` 对象，改造 `sleep` 方法：

```
function sleep(timeout) {
    return new Promise(function(resolve, reject) {
        setTimeout(resolve, timeout);
    });
}
```

除此之外， main 方法还需要使用 async 显式声明成异步方法：

```
async function main() {
    console.time('how long did I sleep');
    await sleep(3000);
    console.timeEnd('how long did I sleep');
}
```

将以上代码保存为 async-sleep.js 。前面也说了需要借助第三方代码转换工具，那我们就安装 Babel ：

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

安装后执行：

```
> ./node_modules/babel-cli/bin/babel-node.js async-sleep.js
```

没出意外的话，我们应该看到...WTF，出错了

```
SyntaxError: async-sleep.js: Unexpected token (7:6)
   5 | }
   6 |
>  7 | async function main() {
     |       ^
   8 |     console.time('how long did I sleep');
   9 |     await sleep(3000);
  10 |     console.timeEnd('how long did I sleep');
```

这时需要安装一个 Babel 的插件用于转换 async ：

```
> npm install --save-dev babel-plugin-transform-async-to-generator
```

安装好后，需要在运行目录添加一个配置文件 .babelrc ：

```
> vi .babelrc
{
    "plugins": ["transform-async-to-generator"]
}
```

或在命令中指定：

```
>  ./node_modules/babel-cli/bin/babel-node.js --plugins transform-async-to-generator
async-sleep.js
```

没出意外的话，我们应该看到...WTF，又出错了

阅读目录

```
async-sleep.js:1
(function (exports, require, module, __filename, __dirname) { let main = (() => {
                                                              ^^^

SyntaxError: Block-scoped declarations (let, const, function, class) not yet supporte
d outside strict mode
```

好吧，提示也很明显，我们使用 strict 模式，完整的代码如下：

```
'use strict';

function sleep(timeout) {
    return new Promise(function(resolve, reject) {
        setTimeout(resolve, timeout);
    });
}

async function main() {
    console.time('how long did I sleep');
    await sleep(3000);
    console.timeEnd('how long did I sleep');
}

main();
// how long did I sleep: 3003ms
```

再执行，终于看到我们期望的 3003ms 。呃，怎么不是 3000ms ，不要在意这些细节。难道QQ空间曾经用这个实际延迟的误差来判断客户端CPU的繁忙程度也要告诉你。

看完这个例子，是不是发现 async 类似于 Generator 中的 * ，而 await 类似于 yield ，但是现在不需要再额外封装一个 run 方法了，这还是很方便的。

重写回调地狱的例子

继续重写那个回调地狱的例子：

1. 读取当前目录的package.json
2. 检查backup目录是否存在，如果不存在就创建backup目录
3. 将文件内容写到备份文件

readPackageFile 、 checkBackupDir 和 backupPackageFile 直接使用上一篇中的定义：

```
var readPackageFile = new Promise(function(resolve, reject) {
    fs.readFile('./package.json', function(err, data) {
        if (err) {
            reject(err);
        }

        resolve(data);
    });
});

var checkBackupDir = new Promise(function(resolve, reject) {
    fs.exists('./backup', function(exists) {
        if (!exists) {
            resolve(mkBackupDir);
        } else {
            resolve();
        }
    });
});

var mkBackupDir = new Promise(function(resolve, reject) {
    // throw new Error('unexpected error');
    fs.mkdir('./backup', function(err) {
        if (err) {
            return reject(err);
        }

        resolve();
    });
});

function backupPackageFile(data) {
    return new Promise(function(resolve, reject) {
        fs.writeFile('./backup/package.json', data, function(err) {
            if (err) {
                return reject(err);
            }

            resolve();
        });
    });
};
```

Let's Rock：

```
(async function() {
    try {
        // 1. 读取当前目录的package.json
        var data = await readPackageFile;

        // 2. 检查backup目录是否存在，如果不存在就创建backup目录
        await checkBackupDir;

        // 3. 将文件内容写到备份文件
        await backupPackageFile(data);

        console.log('backup successed');
    } catch (err) {
        console.error(err);
    }
}());
```

总结

js正朝着越来越好的方向发展，不是吗？

参考地址

- Async/Await: The Hero JavaScript Deserved (https://www.twilio.com/blog/2015/10/asyncawait-the-hero-javascript-deserved.html)
- babel 6 async / await: Unexpected token (http://stackoverflow.com/questions/33641593/babel-6-async-await-unexpected-token)
- Babel cli usage (http://babeljs.io/docs/usage/cli/)
  题图引自：http://forwardjs.com/img/workshops/advancedjs-async.jpg (http://forwardjs.com/img/workshops/advancedjs-async.jpg)

  来自：http://iammapping.com/write-js-async-gracefully-2/ (http://iammapping.com/write-js-async-gracefully-2/)

同类热门经验

1. Node.js 初体验 (/lib/view/open1326870121968.html)
2. JavaScript开发规范要求 (/lib/view/open1352263831610.html)
3. 使用拖拉操作来自定义网页界面布局并保存结果 (/lib/view/open1325064347889.html)
4. Nodejs入门学习，nodejs web开发入门，npm、express、socket配置安装、nodejs聊天室开发 (/lib/view/open1329050007640.html)
5. 利用HTML5同时上传多个文件 - resumable.js (/lib/view/open1327591300671.html)
6. nide：一个不错的Node.js开发工具IDE (/lib/view/open1325834128750.html)

- Developing Backbone.js Applications.pdf (http://www.open-open.com/doc/view/adc0965d992c456bae0f42ae2413db24)
- Professional Node.js.pdf (http://www.open-open.com/doc/view/f585e340b9c1459b8e9081c3bcf62883)
- CoffeeScript: Accelerated JavaScript Development.pdf (http://www.open-open.com/doc/view/edce5a755be04b99a524c01edb7832ca)
- node.js入门之buffer类的使用详解.docx (http://www.open-open.com/doc/view/deb470bdbae64c12a1406b120b8c0d24)
- JavaScript 标准参考教程 - Node.js 概述.pdf (http://www.open-open.com/doc/view/09480785c08f43fc8ea0618ccf2847b9)
- javascript学习总结.doc (http://www.open-open.com/doc/view/4526955729c84aaeb9bef57787f4dc9b)