Async/Await？

我们听到了你对async/await在TypeScript中感到兴奋的反馈。Async/await允许开发者去写异步的代码流就像写同步代码那样，不再需要注册事件处理程序或编写回调函数。在c#中你可能见过类似的模式。TypeScript的async/await使用了Promises，就像c#的async/await使用了Tasks。Promises是表示正在进行的异步操作的对象，是ECMAScript 6(ES6)的内置特性。TypeScript的async/await作为了ES2016（又称ES7）提出的实现标准。

我们非常高兴宣布你今天已经可以在Node.js v4及更高版本上使用async/await了！ 在这篇文章中，我们将向您展示如何使用async/await并且告诉你关于async/await的最新进展。

async/await是怎么工作的

JavaScript是单线程顺序运行的：一旦你的函数开始运行，在它完成之前不能中断。对于多任务，Async/await正是开发者所期望和想要的。然而，当一个异步任务（如调用一个Web服务）运行，在你等待这个任务返回时允许JavaScript的其余部分继续运行是非常高效的。Async/await允许你像调用同步方法的方式去调用异步方法，并且不会阻塞异步操作完成。

例如下面的代码，`main`等待异步函数`ping`的结果。因为`main`需要等待，因此它声明为一个异步函数。`ping`函数等待循环中的`delay`函数，因此它也声明为一个异步函数。`delay`函数调用setTimeout在一段时间后返回一个Promise对象。当setTimeout中的promise对象返回时，你将在看到控制台看到字符串‘ping’。

```
async function main() {
 await ping();
}

async function ping() {
 for (var i = 0; i < 10; i++) {
 await delay(300);
 console.log("ping");
 }
}

function delay(ms: number) {
 return new Promise(resolve => setTimeout(resolve, ms));
}
```

```
main();
```

TypeScript使用ES6 生成器去实现异步调用函数返回时重新进入函数能力。当一个函数等待执行时，生成器使用 `yield` 关键字去告诉JavaScript运行时当前执行操作被暂停（或者说冻结）。当函数执行完后，javascript运行时将当前执行操作激活（或者说解冻）并接着继续向下运行代码。对于上面的示例， `ping` 函数被Typescript编译器生成了下面ES6版本的Javascript代码。

```
function ping() {
 return __awaiter(this, void 0, Promise, function* () {
 for (var i = 0; i < 10; i++) {
 yield delay(300);
 console.log("ping");
 }
 });
}
```


`__awaiter` 函数包裹了一个函数块，其中包括 `yield` 语句，和一个作为生成器执行函数的Promise。

在Node.js尝试一下

开始使用Nightly版本的TypeScript（也就是最新的TypeScript版本，写这篇文章时Typescript最新版本为1.7），TypeScript 1.7版本中ES6的targets编译已经支持async/await。你可以使用 `npm install typescript@next` 命令安装最新的TypeScript版本，并且在Node.js v4版本及以上并且支持ES6生成器的环境中尝试。配置你的tsconfig.json文件如下这个样子：

```
"compilerOptions": {
 "target": "ES6",
 "module": "commonjs"
}
```

编译后的JavaScript就可以在Node.js环境中运行了：

如果你当前使用的Node.js是v4或者更高版本，并且今天要尝试一下async/await。我们已经使用GitHub的API创建了一个异步检索repo的pull requests历史的复杂示例。你可以在TypeScriptSamples repo找到源代码下载下来在Node.js环境中运行它。我们很想听听您的意见，如果你发现问题，请告诉我们。还有一件事情要注意：Node.js还没有支持ES6的模块，因此当你编辑TypeScript时选择CommonJs方式作为模块输出，如上述在tsconfig.json所示。

下一步

我们知道很多TypeScript的开发者想要在浏览器和Node.js中使用async/await，也明白无论是从开发者的工作流程角度来看还是由于性能的影响导致额外的编译开销，使用基于额外转译层的临时解决方案不是最佳的。针对众多的浏览器，我们需要使用状态机重写ES6的生成器使之成为可执行的ES5 JavaScript代码。虽然这是需要跨编译器重大变化的一大挑战，但是我们正在努力为之工作。敬请关注；我们将让您及时了解我们的进展！

- *本文章翻译自What about Async/Await?*
- *本人英文水平有限，翻译不正确不通顺的地方，敬请指出。*

相关

[译]TypeScript 2.1 候选版：更好的类型推断、异步函数和其它更多特性
2017年2月7日
在“TypeScript”中

TypeScript 1.7 发布，默认支持 ES6 async/await 特性
2016年1月19日
在“TypeScript”中

[译]TypeScript 2.0 Beta发布
2016年7月13日
在“JavaScript”中

2016年2月17日 小乔布斯 TypeScript、翻译TS官方博客 TypeScript、async、await、ES6、ES2015

《Async/Await？》有1个想法

匿名

2016年2月17日 下午4:48

不错