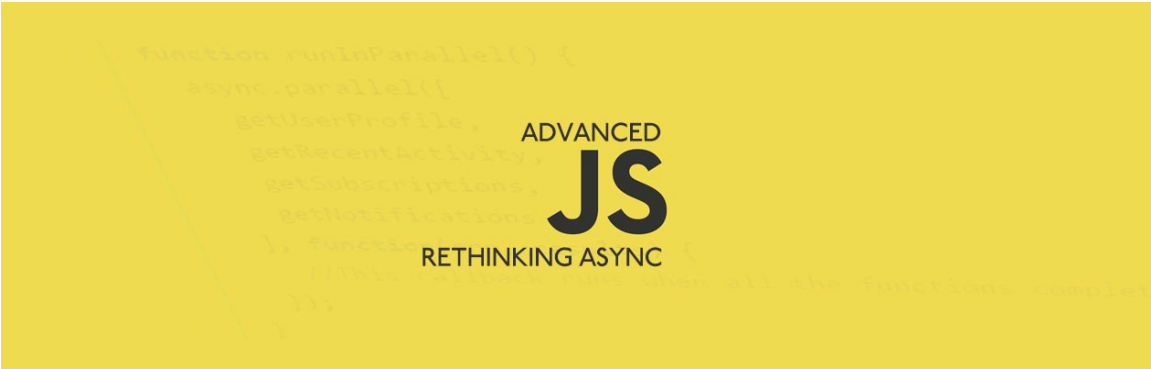


如何优雅地写js异步代码

javascript (/lib/tag/javascript) 2016-04-22 16:20:54 发布

您的评价: 0.0

收藏

2收藏

本文通过一个简单的需求：读取文件并备份到指定目录（详见第一段代码的注释），以不同的js代码实现，来演示代码是如何变优雅的。对比才能分清好坏，想知道什么是优雅的代码，先看看糟糕的代码。

不优雅的代码是什么样的？

1、 回调地狱

阅读目录

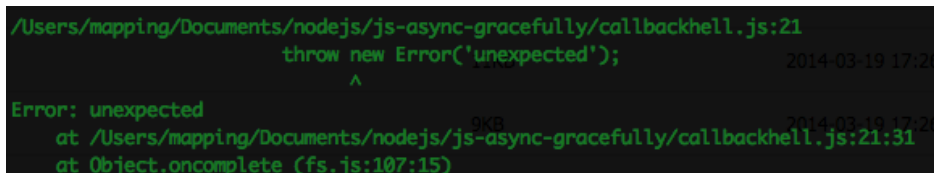
- 不优雅的代码是什
- 1、 回调地狱
- 2、 匿名调试
- 怎样写才能让js回
- 他山之石 可以攻玉
- browser js
- NodeJs
- jQuery Deferred
- Async
- ECMAScript 6
- Promise
- Generator

[总结](#)[参考地址](#)

```
/**
 * 读取当前目录的package.json, 并将其备份到backup目录
 *
 * 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
 */
fs.readFile('./package.json', function(err, data) {
  if (err) {
    console.error(err);
  } else {
    fs.exists('./backup', function(exists) {
      if (!exists) {
        fs.mkdir('./backup', function(err) {
          if (err) {
            console.error(err);
          } else {
            // throw new Error('unexpected');
            fs.writeFile('./backup/package.json', data, function(err) {
              if (err) {
                console.error(err);
              } else {
                console.log('backup succeeded');
              }
            });
          }
        });
      }
    });
  } else {
    fs.writeFile('./backup/package.json', data, function(err) {
      if (err) {
        console.error(err);
      } else {
        console.log('backup succeeded');
      }
    });
  }
});
});
```

2、匿名调试

取消上面代码中抛出异常的注释再执行



```
/Users/mapping/Documents/nodejs/js-async-gracefully/callbackhell.js:21
      throw new Error('unexpected');
      ^
Error: unexpected
    at /Users/mapping/Documents/nodejs/js-async-gracefully/callbackhell.js:21:31
    at Object.oncomplete (fs.js:107:15)
```

wtf, 这个 unexpected 错误从哪个方法抛出来的?

神马? 你觉的这个代码写得很好, 优雅得无可挑剔? 那么你现在可以忽略下文直接去最后的评论写: 楼主敏感词

怎样写才能让js回调看上去优雅?

1. 消除回调嵌套
2. 命名方法

```

fs.readFile('./package.json', function(err, data) {
  if (err) {
    console.error(err);
  } else {
    writeFileContentToBackup(data);
  }
});

function writeFileContentToBackup(fileContent) {
  checkBackupDir(function(err) {
    if (err) {
      console.error(err);
    } else {
      backup(fileContent, log);
    }
  });
}

function checkBackupDir(cb) {
  fs.exists('./backup', function(exists) {
    if (!exists) {
      mkBackupDir(cb);
    } else {
      cb(null);
    }
  });
}

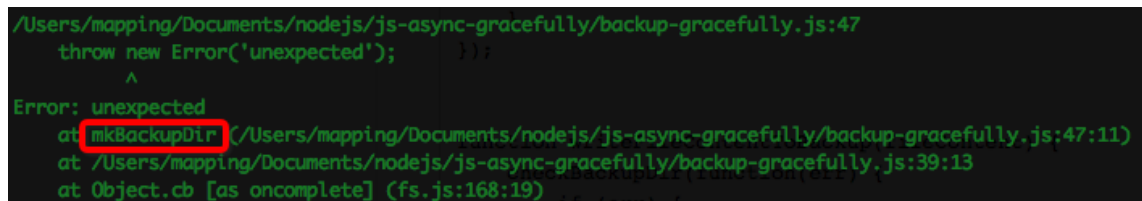
function mkBackupDir(cb) {
  // throw new Error('unexpected');
  fs.mkdir('./backup', cb);
}

function backup(data, cb) {
  fs.writeFile('./backup/package.json', data, cb);
}

function log(err) {
  if (err) {
    console.error(err);
  } else {
    console.log('backup succeeded');
  }
}

```

我们现在可以快速定位抛出异常的方法



```

/Users/mapping/Documents/nodejs/js-async-gracefully/backup-gracefully.js:47
  throw new Error('unexpected');
  ^
Error: unexpected
    at mkBackupDir (/Users/mapping/Documents/nodejs/js-async-gracefully/backup-gracefully.js:47:11)
    at /Users/mapping/Documents/nodejs/js-async-gracefully/backup-gracefully.js:39:13
    at Object.cb [as oncomplete] (fs.js:168:19)

```

他山之石 可以攻玉

借助第三方库，优化异步代码

browser js

- jQuery Deferred
- ajax
- animate

NodeJs

- Async (<https://github.com/caolan/async>)

- `async.each`
- `async.map`
- `async.waterfall`
- ECMAScript 6
- Promise (https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Promise)
- Generator (https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Statements/function*)

jQuery Deferred

在jQuery-1.5中引进，被应用在ajax、animate等异步方法上

一个简单的例子：

```
function sleep(timeout) {
    var dtd = $.Deferred();
    setTimeout(dtd.resolve, timeout);
    return dtd;
}

// 等同于上面的写法
function sleep(timeout) {
    return $.Deferred(function(dtd) {
        setTimeout(dtd.resolve, timeout);
    });
}

console.time('sleep');
sleep(2000).done(function() {
    console.timeEnd('sleep');
});
```

一个复杂的例子：

```
function loadImg(src) {
    var dtd = $.Deferred(),
        img = new Image;

    img.onload = function() {
        dtd.resolve(img);
    }

    img.onerror = function(e) {
        dtd.reject(e);
    }

    img.src = src;

    return dtd;
}

loadImg('http://www.baidu.com/favicon.ico (http://www.baidu.com/favicon.ico)').then(
    function(img) {
        $('body').prepend(img);
    }, function() {
        alert('load error');
    }
)
```

那么问题来了，我想要过5s后把百度Logo显示出来？

普通写法：

```
sleep(5000).done(function() {  
    loadImg('http://www.baidu.com/favicon.ico (http://www.baidu.com/favicon.ico)').  
done(function(img) {  
    $('body').prepend(img);  
    });  
});
```

二逼写法:

```
setTimeout(function() {  
    loadImg('http://www.baidu.com/favicon.ico (http://www.baidu.com/favicon.ico)').d  
one(function(img) {  
    $('body').prepend(img);  
    });  
}, 5000);
```

文艺写法(睡5s和加载图片同步执行):

```
$.when(sleep(5000), loadImg('http://www.baidu.com/favicon.ico (http://www.baidu.com/  
favicon.ico)')).done(function(ignore, img) {  
    $('body').prepend(img);  
});
```

Async

使用方法参考: <https://github.com/caolan/async> (<https://github.com/caolan/async>)

优点:

1. 简单、易于理解
2. 函数丰富, 几乎可以满足任何回调需求
3. 流行

缺点:

1. 额外引入第三方库
2. 虽然简单, 但还是难以掌握所有api

ECMAScript 6

ES6的目标, 是使得JavaScript语言可以用来编写大型的复杂的应用程序, 成为企业级开发语言。

接下来介绍ES6的新特性: Promise对象和Generator函数, 是如何让代码看起来更优雅。

更多ES6的特性参考: ECMAScript 6 入门 (<http://es6.ruanyifeng.com/>)

Promise

Promise对象的初始化以及使用:

```
var promise = new Promise(function(resolve, reject) {
  setTimeout(function() {
    if (true) {
      resolve('ok');
    } else {
      reject(new Error('unexpected error'));
    }
  }, 2000);
});

promise.then(function(msg) {
  // throw new Error('unexpected resolve error');
  console.log(msg);
}).catch(function(err) {
  console.error(err);
});
```

JavaScript Promise 的 API 会把任何包含有 then 方法的对象当作“类 Promise”（或者用术语来说就是 thenable）

与上面介绍的jQuery Deferred对象类似，但api方法和错误捕捉等不完全一样。
可以使用以下方法转换：

```
var promise = Promise.resolve($.Deferred());
```

那怎么使用Promise改写回调地狱那个例子？

```
// 1. 读取当前目录的package.json
readPackageFile.then(function(data) {
  // 2. 检查backup目录是否存在，如果不存在就创建backup目录
  return checkBackupDir.then(function() {
    // 3. 将文件内容写到备份文件
    return backupPackageFile(data);
  });
}).then(function() {
  console.log('backup succeeded');
}).catch(function(err) {
  console.error(err);
});
```

这么简单？

看看 readPackageFile 、 checkBackupDir 和 backupPackageFile 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();
    });
  });
};

```

是不是感觉到满满的欺骗，说好的简单呢，先别打，至少调用起来还是很简单的XD。个人觉得使用**Promise**最大的好处就是让调用方爽。

流程优化，使用js的无阻塞特性，我们发现第一步和第二步可以同步执行：

```

Promise.all([readPackageFile, checkBackupDir]).then(function(res) {
  return backupPackageFile(res[0]);
}).then(function() {
  console.log('backup succeeded');
}).catch(function(err) {
  console.error(err);
});

```

在ES5环境下可以使用的库：

- bluebird (<https://github.com/petkaantonov/bluebird>)
- Q (<https://github.com/krisowal/q>)
- when (<https://github.com/cujojs/when>)
- WinJS (<http://msdn.microsoft.com/en-us/library/windows/apps/br211867.aspx>)
- RSVP.js (<https://github.com/tildeio/rsvp.js>)

Generator

NodeJs默认不支持Generator的写法，但在v0.12后可以添加 `--harmony` 参数使其支持：

```
> node --harmony generator.js
```

允许函数在特定地方像 `return` 一样退出，但是稍后又能恢复到这个位置和状态上继续执行

```
function * foo(input) {  
  console.log('这里会在第一次调用next方法时执行');  
  yield input;  
  console.log('这里不会被执行，除非再调一次next方法');  
}  
  
var g = foo(10);  
  
console.log(Object.prototype.toString.call(g)); // [object Generator]  
console.log(g.next()); // { value: 10, done: false }  
console.log(g.next()); // { value: undefined, done: true }
```

如果觉得比较难理解，就把 `yield` 看成 `return` 语句，把整个函数拆分成许多小块，每次调用 `generator` 的 `next` 方法就按顺序执行一小块，执行到 `yield` 就退出。

告诉你一个惊人的秘密，我们现在可以“同步”写js的 `sleep` 了：

```
var sleepGenerator;  
  
function sleep(time) {  
  setTimeout(function() {  
    sleepGenerator.next(); // step 5  
  }, time);  
}  
  
var sleepGenerator = (function * () {  
  console.log('wait...'); // step 2  
  console.time('how long did I sleep'); // step 3  
  yield sleep(2000); // step 4  
  console.log('weakup'); // step 6  
  console.timeEnd('how long did I sleep'); // step 7  
})();  
  
sleepGenerator.next(); // step 1
```

合体，使用Promise和Generator重写回调地狱的例子

合体前的准备工作，参考Q.async

(<https://github.com/krisKowal/q/blob/db9220d714b16b96a05e9a037fa44ce581715e41/q.js#L500>):


```
function run(makeGenerator) {
  function continuer(verb, arg) {
    var result;
    try {
      result = generator[verb](arg);
    } catch (err) {
      return Promise.reject(err);
    }
    if (result.done) {
      return result.value;
    } else {
      return Promise.resolve(result.value).then(callback, errback);
    }
  }
  var generator = makeGenerator.apply(this, arguments);
  var callback = continuer.bind(continuer, "next");
  var errback = continuer.bind(continuer, "throw");
  return callback();
}
```

`readPackageFile`、`checkBackupDir` 和 `backupPackageFile` 直接使用上面 `Promise` 中的定义，是不是很爽。

合体后的执行：

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

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

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

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

是不是感觉跟写同步代码一样了。

总结

看完本文，如果你感慨：“靠，js还能这样写”，那么我的目的就达到了。本文的写作初衷不是介绍

`Async`、`Deferred`、`Promise`、`Generator` 的用法，如果对于这几个概念不是很熟悉的话，建议查阅其他资料学习。写js就像说英语，不是 *write in js*，而是 *think in js*。不管使用那种方式，都是为了增强代码的可读性和可维护性；如果是在已有的项目中修改，还要考虑对现有代码的侵略性。

续集：如何优雅地写js异步代码(2) (<http://iammapping.com/write-js-async-gracefully-2/>)

参考地址

- 回调地狱 (<http://callbackhell.com/>)
- JavaScript Promise 启示录 (<http://www.alloyteam.com/2014/05/javascript-promise-mode/>)
- Promises/A+ (<https://promisesaplus.com/>)
- ECMAScript 6 入门 (<http://es6.ruanyifeng.com/>)
- JavaScript Promises (<http://www.html5rocks.com/zh/tutorials/es6/promises/>)
- 使用 (Generator) 生成器解决 JavaScript 回调嵌套问题 (<http://huangj.in/765>)
- 拥抱Generator，告别回调 (<http://yaniswang.com/frontend/2014/09/29/es6-generator/>)

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/> (<http://iammapping.com/write-js-async-gracefully/>)

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)

相关文档 — 更多 (<http://www.open-open.com/doc>)

相关经验 — 更多

相关讨论 — 更多 (<http://www.open-open.com/solution>)

- 《Ext JS 3.2 学习指南》(Learning Ext JS 3.2)英文文字版.pdf (<http://www.open-open.com/doc/view/18e48b568c4c44e68cbcf2f18aec9f70>)
- 如何使用ExtJS框架实现无级树结构.pdf (<http://www.open-open.com/doc/view/7c8f07533193417a8ec9f0986b3a4e8c>)
- js操作xml(javascript xml).doc (<http://www.open-open.com/doc/view/bd3630afa33d4159becb61f822b5ceb5>)
- Smashing Node.js: JavaScript Everywhere.pdf (<http://www.open-open.com/doc/view/0b330fc1b2e948feb7b572f83559ee7c>)
- Developing Backbone.js Applications.pdf (<http://www.open-open.com/doc/view/adc0965d992c456bae0f42ae2413db24>)
- JavaScript 面向对象15分钟教程.pdf (<http://www.open-open.com/doc/view/62f0af5fbc4c4ed08220b423d717c792>)
- JavaScript面向对象15分钟教程.pdf (<http://www.open-open.com/doc/view/dc7f4e11e95d484ab2777e145562a323>)
- CoffeeScript: Accelerated JavaScript Development.pdf (<http://www.open-open.com/doc/view/edce5a755be04b99a524c01edb7832ca>)
- jQuery+Ajax+Struts2.js javascript.doc (<http://www.open-open.com/doc/view/3d549e3354494d83861dfcfd00e8949>)
- JavaScript 标准参考教程 - Node.js 概述.pdf (<http://www.open-open.com/doc/view/09480785c08f43fc8ea0618ccf2847b9>)
- 《D3.js数据可视化实战手册》迷你书.pdf (<http://www.open-open.com/doc/view/41ba0b7d5f784890891ab625a196c92e>)
- Professional Node.js Building JavaScript Based Scalable Software.pdf (<http://www.open-open.com/doc/view/470f0f980bb34d47993ff98dd89ff665>)
- Node入门一本全面地Node.js教程.pdf (<http://www.open-open.com/doc/view/78381b3b3c9c40e49d67bfc90d50e41f>)
- Node入门一本全面地node.js教程.pdf (<http://www.open-open.com/doc/view/f680b59006c34260ab1e0fcf42cad6e3>)
- Test-Driven JavaScript Development.pdf (<http://www.open-open.com/doc/view/daeb9f76b4ad4a7ab29c023d0a9b5c55>)
- JavaScript 入门笔记.doc (<http://www.open-open.com/doc/view/392f3dc31f1a4095899fe760d38db0db>)
- JavaScript、jQuery、HTML5、Node.js实例大全mini电子书-
- JavaScript 资源大全中文版 (/lib/view/open1450791728776.html)
- 给 JavaScript 初心者的 ES2015 实战 (/lib/view/open1447222864319.html)
- React.js生态系统概览 [译] (/lib/view/open1446383331963.html)
- 如何优雅地写js异步代码(2) (/lib/view/open1461313379798.html)
- Nodejs学习路线图 (/lib/view/open1403574545233.html)
- Nodejs学习路线图 (/lib/view/open1432785701488.html)
- 史上最全 前端开发面试问题及答案整理 (/lib/view/open1460612941431.html)
- 史上最全 前端开发面试问题及答案整理 (/lib/view/open1437483697115.html)
- 结合个人经历总结的前端入门方法 (/lib/view/open1449542023941.html)
- AngularJS - 下一个大框架 (/lib/view/open1410230442070.html)
- 码农周刊分类整理 (/lib/view/open1416282051852.html)
- iOS 资源大全 (/lib/view/open1454374758667.html)
- 2015前端组件化框架之路 (/lib/view/open1427350415652.html)
- 在 Node.js 上使用 Dojo (/lib/view/open1331824383734.html)
- 浅谈JavaScript编程语言的编码规范 (<http://www.open-open.com/solution/view/1318472833218>)
- 什么是Node.js? (<http://www.open-open.com/solution/view/1318473088937>)
- 我为什么向后端工程师推荐Node.js (<http://www.open-open.com/solution/view/1322451238921>)
- 那些年, 追过的开源软件和技术 (<http://www.open-open.com/solution/view/1425959150201>)
- Web开发人员最喜爱的10款流行前面试问题(二)- 史上最全 前端开发 JavaScript库 (<http://www.open-open.com/solution/view/1379903308476>)
- PHP程序员的技术成长规划 (<http://www.open-open.com/solution/view/1414478644325>)
- 程序员技术练级攻略 (<http://www.open-open.com/solution/view/1319276210452>)

- v1.pdf (<http://www.open-open.com/doc/view/42a52c4308194927ba2e13aa2e27b811>)
- jQuery Ajax全解析.doc (<http://www.open-open.com/doc/view/b67b6f6e69ff42c2a8bfd72c954b0f21>)
 - 深入浅出Node.js.pdf (<http://www.open-open.com/doc/view/3501c93f01894e85bb083a5dc2cc6aa1>)
 - javascript代码规范.doc (<http://www.open-open.com/doc/view/33cc28c3c2eb4f11a2136ae1aad887bf>)
-

©2006-2016 深度开源



(<http://www.open-open.com/>)

浙ICP备09019653号-31

(<http://www.miibeian.gov.cn/>) 站长统计

([http://www.cnzz.com/stat/website.php?](http://www.cnzz.com/stat/website.php?web_id=1257892335)

[web_id=1257892335](http://www.cnzz.com/stat/website.php?web_id=1257892335))