Javascript ...

JavaScript 教程
JavaScript 简介
JavaScript 用法
JavaScript Chrome 中运行
JavaScript 输出
JavaScript 语法
JavaScript 语句
JavaScript 注释
JavaScript 变量
JavaScript 数据类型
JavaScript 对象
JavaScript 函数
JavaScript 作用域
JavaScript 事件
JavaScript 字符串
JavaScript 运算符
JavaScript 比较
JavaScript 条件语句
JavaScript switch 语句
JavaScript for 循环
JavaScript while 循环

JavaScript void

JavaScript Promise

JavaScript 异步编程

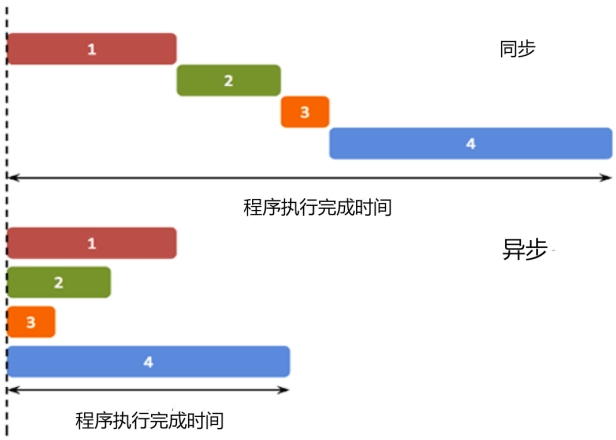
异步的概念

异步（Asynchronous, async）是与同步（Synchronous, sync）相对的概念。

在我们学习的传统单线程编程中，程序的运行是同步的（同步不意味着所有步骤同时运行，而是指步骤在一个控制流序列中按顺序执行）。而异步的概念则是不保证同步的概念，也就是说，一个异步过程的执行将不再与原有的序列有顺序关系。

简单来理解就是：同步按你的代码顺序执行，异步不按照代码顺序执行，异步的执行效率更高。

以上是关于异步的概念的解释，接下来我们通俗地解释一下异步：异步就是从主线程发射一个子线程来完成任务。



什么时候用异步编程

在前端编程中（甚至后端有时也是这样），我们在处理一些简短、快速的操作时，例如计算 1 + 1 的结果，往往在主线程中就可以完成。主线程作为一个线程，不能够同时接受多方面的请求。所以，当一个事件没有结束时，界面将无法处理其他请求。

现在有一个按钮，如果我们设置它的 onclick 事件为一个死循环，那么当这个按钮按下，整个网页将失去响应。

为了避免这种情况的发生，我们常常用子线程来完成一些可能消耗时间足够长以至于被用户察觉的事情，比如读取一个大文件或者发出一个网络请求。因为子线程独立于主线程，所以即使出现阻塞也不会影响主线程的运行。但是子线程有一个局限：一旦发射了以后就会与主线程失去同步，我们无法确定它的结束，如果结束之后需要处理一些事情，比如处理来自服务器的信息，我们是无法将它合并到主线程中去的。

分类导航

HTML / CSS
JavaScript
服务端
数据库
数据分析
移动端
XML 教程
ASP.NET
Web Service
开发工具
网站建设

JavaScript break 和 continue 语句
JavaScript typeof
JavaScript 类型转换
JavaScript 正则表达式
JavaScript 错误
JavaScript 调试
JavaScript 变量提升
JavaScript 严格模式
JavaScript 使用误区
JavaScript 表单
JavaScript 表单验证
JavaScript 验证 API
JavaScript 保留关键字
JavaScript this
JavaScript let 和 const
JavaScript JSON
JavaScript void
JavaScript 异步编程
JavaScript Promise
JavaScript 代码规范
JS 函数
JavaScript 函数定义

为了解决这个问题，JavaScript 中的异步操作函数往往通过回调函数来实现异步任务的结果处理。

回调函数

回调函数就是一个函数，它是在我们启动一个异步任务的时候就告诉它：等你完成了这个任务之后要干什么。这样一来主线程几乎不用关心异步任务的状态了，他自己会善始善终。

实例

```
function print() {
    document.getElementById("demo").innerHTML="RUNOOB!";
}
setTimeout(print, 3000);
```

尝试一下 »

这段程序中的 setTimeout 就是一个消耗时间较长（3 秒）的过程，它的第一个参数是个回调函数，第二个参数是毫秒数，这个函数执行之后会产生一个子线程，子线程会等待 3 秒，然后执行回调函数 "print"，在命令行输出 "RUNOOB!"。

当然，JavaScript 语法十分友好，我们不必单独定义一个函数 print，我们常常将上面的程序写成：

实例

```
setTimeout(function () {
    document.getElementById("demo").innerHTML="RUNOOB!";
}, 3000);
```

尝试一下 »

注意：既然 setTimeout 会在子线程中等待 3 秒，在 setTimeout 函数执行之后主线程并没有停止，所以：

实例

```
setTimeout(function () {
    document.getElementById("demo1").innerHTML="RUNOOB-1!";
}, 3000);
document.getElementById("demo2").innerHTML="RUNOOB-2!";
console.log("2");
```

尝试一下 »

这段程序的执行结果是：

```
RUNOOB-1!
RUNOOB-2!
```

异步 AJAX

除了 setTimeout 函数以外，异步回调广泛应用于 AJAX 编程。有关于 AJAX 详细请参见：https://www.runoob.com/ajax/ajax-tutorial.html

JavaScript 函数参数
JavaScript 函数调用
JavaScript 闭包

JS HTML DOM

DOM 简介
DOM HTML
DOM CSS
DOM 事件
DOM EventListener
DOM 元素
HTMLCollection 对象
NodeList 对象

JS 高级教程

JavaScript 对象
JavaScript prototype
JavaScript Number 对象
JavaScript String
JavaScript Date（日期）
JavaScript Array（数组）
JavaScript Boolean（布尔）
JavaScript Math（算数）
JavaScript RegExp 对象

JS 浏览器BOM

JavaScript Window

XMLHttpRequest 常常用于请求来自远程服务器上的 XML 或 JSON 数据。一个标准的 XMLHttpRequest 对象往往包含多个回调：

实例

```
var xhr = new XMLHttpRequest();

xhr.onload = function () {
    // 输出接收到的文字数据
    document.getElementById("demo").innerHTML=xhr.responseText;
}

xhr.onerror = function () {
    document.getElementById("demo").innerHTML="请求出错";
}

// 发送异步 GET 请求
xhr.open("GET", "https://www.runoob.com/try/ajax/ajax_info.txt", true);
xhr.send();
```

尝试一下 »

XMLHttpRequest 的 onload 和 onerror 属性都是函数，分别在它请求成功和请求失败时被调用。如果你使用完整的 jQuery 库，也可以更加优雅的使用异步 AJAX：

实例

```
$.get("https://www.runoob.com/try/ajax/demo_test.php",function(data,status){
    alert("数据: " + data + "\n状态: " + status);
});
```

尝试一下 »

JavaScript void　　JavaScript Promise

JavaScript Window Screen
JavaScript Window Location
JavaScript Window History
JavaScript Navigator
JavaScript 弹窗
JavaScript 计时事件
JavaScript Cookie

JS 库

JavaScript 库
JavaScript 测试 jQuery
JavaScript 测试 Prototype

JS 实例

JavaScript 实例
JavaScript 对象实例
JavaScript 浏览器对象实例
JavaScript HTML DOM 实例
JavaScript 总结

JS 参考手册

JavaScript 对象
HTML DOM 对象
JavaScript 异步编程

- CSS 实例
- JavaScript 实例
- Ajax 实例
- jQuery 实例
- XML 实例
- Java 实例

- HTML 字符集设置
- HTML ASCII 字符集
- HTML ISO-8859-1
- HTML 实体符号
- HTML 拾色器
- JSON 格式化工具

- CSS clip-path 属性
- CSS @charset 规则
- CSS grid-row 属性
- CSS grid-templa...
- CSS grid-templa...
- CSS grid-templa...
- CSS grid-templa...

- 免责声明
- 关于我们
- 文章归档

关注微信

反馈/建议