同步与异步

调用关系：A 调用 B

同步

在发出一个**调用**，比如：函数的调用或接口的调用，一定在处理完成后才返回结果。

1. 没有得到确定结果，这次调用一定不返回
2. 一旦调用返回，就得到返回值
3. **调用者**一直会等待这个调用结果
4. 最常见的就是，python的函数调用

总结：A调用B，B处理直到获得结果，才返回给A，重点在B。

异步

在发出**调用**后，本次调用立即/直接返回，没有直接结果

1. 异步调用过程不会立即获得结果
2. 一般通过状态，通知，或者回调函数处理这次调用

总结：A调用B，B直接返回。无需等待结果，B通过状态，通知等来通知A或回调函数来处理。

阻塞

阻塞和非阻塞，关注程序在等待调用结果(消息，返回值)时的状态

阻塞调用：调用结果返回之前，当前线程被挂起，调用线程只有得到结果后才会返回

总结：A调用B，A被挂起直到B返回结果给A，A继续执行。主要体现在等待一件事情的处理结果时，你是否还去干点其他的事情，如果不去，则为阻塞。

非阻塞

非阻塞调用：不能立即得到结果时，本次调用不阻塞当前线程

总结：A调用B，A不会被挂起，A可以执行其他操作（但可能A需要轮询检查B是否返回）。非阻塞体现在等待一件事情的处理结果时，你是否还去干点其他的事情，如果去了，则为非阻塞。

组合分析

同步阻塞

A调用B，A挂起，B处理直到获得结果，返回给A，A继续执行

案例：客户端单线程+服务端无回调

同步非阻塞

A调用B，A继续执行，B处理直到获得结果，处理的同时A轮询检查B是否返回结果。

异步阻塞

异步阻塞 I/O 模型的典型流程 (select)。

客户端单线程，服务端多线程回复，再给回调

异步非阻塞

A调用B，B立即返回，A继续执行，B得到结果后通过状态，通知等通知A或回调函数处理。

前端ajax请求，发起请求后，不等待，继续处理其他的事情，后面回来再接着处理