CPU：CPU是计算机的核心，承担着计算机的计算任务。可以比喻为一个工厂。

进程：是操作系统进行资源分配和调度的一个独立单位。可以比喻为工厂的车间，它代表CPU能处理的单个任务。

线程：早期是没有线程的概念的，那时候，进程才是资源分配和调度的最小单位，也是程序执行的最小单位。后来随着计算机的发展，对CPU要求越来越高，但是进程的转换开销实在太大了，于是就发明了线程。线程现在是程序执行流的最小单元。把线程可以比喻为车间的工人。

区别？

进程是操作系统进行资源分配和调度的最小单位，线程是程序执行流中的最小单元。  
一个进程是由一个或者多个线程组成的。  
进程之间相互独立，在同一进程中的线程共享程序的内存空间及一些进程级的资源。  
调度和切换：线程上下文切换比进程上下文的切换要快很多。

为什么浏览器要使用多进程的？

在浏览器刚被设计出来的时候，那时候的页面都很简单，因此一个进程去处理多个页面是可行的。但是在今天，大量的页面都很复杂，如果有一个Tab页崩溃了，将会导致所有的Tab也都崩溃。另外，进程之间是不会共享资源，所以不会存在太多的安全问题。

浏览器是一个多进程的，包括一个Brower主进程，负责协调、主控等。第三方插件进程，只有使用的时候才会创建。GPU进程，最多一个，主要用于3D绘制。浏览器的渲染进程，每个Tab页就是一个渲染进程，Google之类的浏览器会将空白的Tab页都合并为一个进程。

渲染进程是个多线程的，包括

1. GUI渲染进程
2. JavaScript引擎线程
3. 定时触发器线程
4. 事件触发线程
5. 异步http请求线程

GUI渲染线程  
 负责渲染页面的HTML元素，当页面重绘（repaint）或回流（reflow）的时候执行。当JS引擎运行的时候，它会挂起，也就是冻结。  
JS引擎线程  
 负责处理运行JavaScript脚本程序，如V8引擎。  
JS为什么设计为单线程？

这要从JS这门语言诞生的目的开始说起，JS是为了检验表单诞生的，那么检验表单你就势必要操作DOM。如果JS被设计为多线程的，那么可能会同时有两个线程操作一个DOM节点，一个修改一个删除，那么这个时候就需要浏览器来决定哪个生效。当然我们可以借用锁来解决上述问题。但是引入锁之后会带来更大的复杂性，所以还不如单线程。

JS引擎线程和GUI渲染线程是互斥的！

由于JS可以操作DOM，如果在修改这些元素属性的同时去渲染页面，那么渲染前后获得的数据会不一样，简单说就是没法玩了。因此为了防制这种情况的出现，浏览器设置GUI和JS互斥。但是这也就是常说的JS阻塞页面加载，如果JS执行时间过长，会导致渲染不连贯，甚至页面冻结。

定时器触发线程  
 计时的。

事件触发线程

当一个事件被触发的时候，该线程会把该事件放在任务队列的队尾，等待JS引擎的处理。这个事件可以是当前代码执行块的定时任务，也可以是浏览器内核的其他线程如鼠标点击、Ajax请求等，但都要排队。

异步HTTP请求线程

在XMLHttpRequest连接之后浏览器会开一个线程请求，检测状态变化，如果设置有回调函数，异步线程就产生变更事件放到JS引擎任务队列中。

JS引擎内部原理：  
 触发和执行不是一个概念

异步的原理？

JS是单线程的，但是却能执行异步任务，主要是因为JS中存在事件循环（Event Loop）和任务队列（Task Queue）。

事件循环：JS会创建一个类似while（true）的循环，每执行一次循环体的过程称之为Tick。每次的Tick过程就是查看是否有待处理事件，如果有则取出相关事件及回调放入到执行栈中由主线程执行。待处理事件会存储在一个任务队列中，每次Tick就是查看任务队列中是否有需要执行的任务。

任务队列：异步操作会将相关回调添加到任务队列中。

主线程：JS只有一个线程，称之为主线程。事件循环是当主线程中的执行栈中的代码执行完毕之后，才开始执行的。