事件循环

浏览器的进程模型

什么是进程

程序运行需要有自己专属的内存空间，可以把内存空间理解为进程

每个应用至少有一个进程，进程之间相互独立，即使要通信，也需要双方都同意

什么是线程

有进程就可以运行代码了

运行代码的“人”可以叫线程

一个进程至少有一个线程，进程开启后会自动创建一个“主线程”来运行代码，主线程关闭就意味着程序关闭

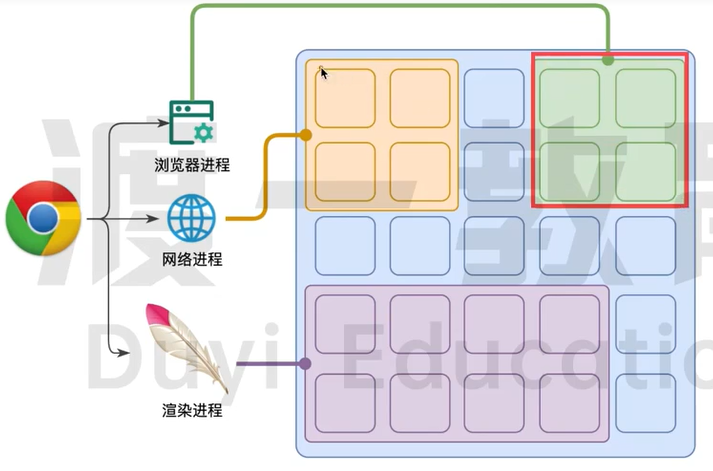
如果程序需要多块代码同时运行，主线程就会启动更多线程（多雇几个人），所以一个进程里可能有多个线程

浏览器有哪些进程和线程

浏览器是一个多进程，多线程的应用程序

浏览器内部工作极其复杂

为了避免相互影响，为了减少连环崩溃的几率，启动浏览器后，会开辟多个进程



其中最主要的进程有：

1. 浏览器进程：负责界面显示，用户交互，子进程管理
2. 网络进程
3. 渲染进程（重点）：进程会开启一个渲染主线程，负责html，css，js代码渲染，每开启一个标签页，就会开启一个新的渲染进程，以保证多个标签页不会相互影响，一个标签页渲染主线程有且只有一个，所以js是单线程编程

渲染主线程处理的任务：

1. 解析html
2. 解析css
3. 计算样式
4. 布局
5. 处理图层
6. 每秒把页面渲染60次（帧生成）
7. 执行全局js代码
8. 执行事件处理函数
9. 执行计时器的回调函数

为什么渲染进程不开启多个线程来分别处理

1. HTML、CSS、JavaScript 是强相关的，不能并行独立处理

这些语言在渲染中是密切耦合的：

* HTML 定义了结构，CSS 样式应用于 HTML 元素。
* JavaScript 可以修改 HTML（DOM）和 CSS（样式表）。
* CSS 的一些计算（比如 em、继承）依赖 DOM。
* JavaScript 执行过程中会阻塞 HTML 的解析（因为可能会修改 DOM）。

举个例子：

<p id="demo">Hello</p>

<script>

document.getElementById("demo").style.color = "red";

</script>

你不能先处理 CSS，再处理 JS，因为 JS 会改变样式。

2. 多线程解析带来的并发问题复杂

如果让不同线程分别处理 HTML、CSS、JS，会导致以下问题：

* 资源依赖管理复杂：JS 会动态操作 DOM，CSS 也依赖 DOM 结构，处理顺序难以确定。
* 状态同步成本高：每个线程都要不断同步 DOM 的状态，会引入性能开销和同步问题。
* 调试复杂度高：出错时难以定位问题来源，因为状态可能在不同线程间改变。

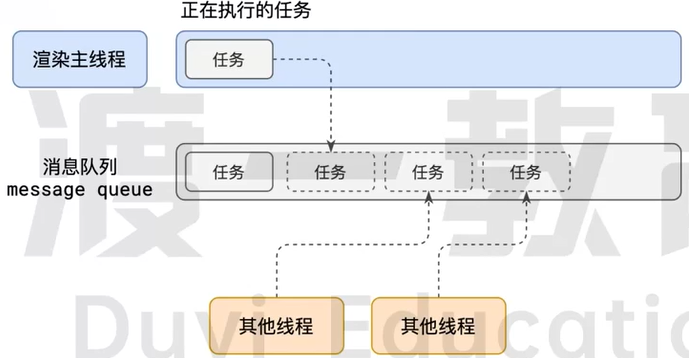
3. 渲染过程需要严格的顺序和同步性

浏览器的渲染流程大致如下：

* 解析 HTML，生成 DOM 树。
* 解析 CSS，生成 CSSOM。
* 合成 DOM 和 CSSOM，生成 Render Tree。
* 布局（Layout），计算每个节点的位置。
* 绘制（Paint），生成像素。

既然不能使用多线程，要处理多个任务，该如何调度任务？

答案就是顺序执行，即“排队”：



也就是说，先让任务排队，不管有多少任务，按顺序执行

1. 在最开始的时候，会开始一个无线循环
2. 每一次循环，都会检查消息队列中是否有任务，如果有，会拿取第一个任务执行，执行完进入下一个循环，如果没有则进入休眠状态
3. 其他所有线程（包括其他进程的线程）可以随时向消息队列里添加任务，新任务会添加到消息队列的末尾，如果此时主线程休眠，消息队列会唤醒主线程继续循环拿取任务

上述整个过程被称为：“事件循环”

什么为异步？

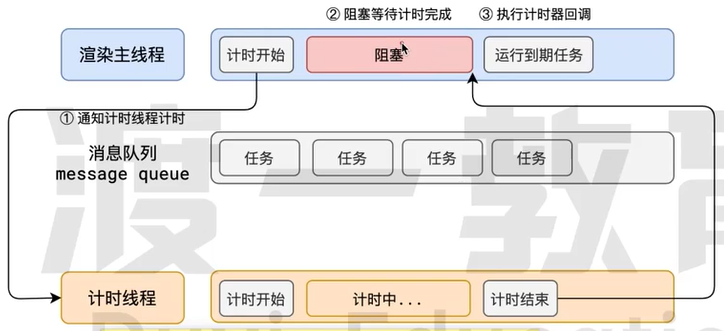
代码在执行过程中，会遇到一些无法立即执行的任务：

计时器：setTimeout，setInterval

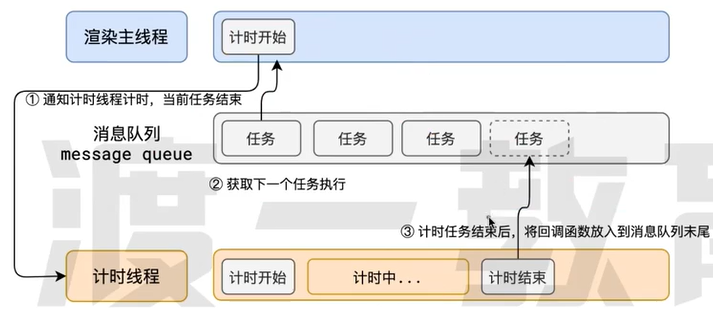
网络通信后需要执行的任务：Fetch

用户操作后要执行的操作：addEventListener

如果让渲染主线程等待这些任务的执行时机，就会导致主线程长期处于“阻塞”状态，从而导致浏览器“卡死”，如下图（“同步”运行）：



浏览器的渲染主线程极其重要，无论如何不能阻塞，所以要使用“异步”模式：



也就是说，当遇到一个无法立即执行的任务时，并不会直接添加到消息队列，而是会直接结束任务执行，将任务交给计时线程，转而去执行后续代码，当计算出到了执行时机，才会将事先传递的回调函数包装成任务，添加进消息队列，然后按顺序执行，使用异步的方式，主线程永不阻塞

任务的优先级

任务没有优先级，都是先来后到按顺序执行

但消息队列是有优先级的，也就是说：

每个任务都有一个任务类型，同一类型的任务必须在同一个队列

不同类型的任务可以分属不同队列

在一次事件循环中，浏览器可以根据实际情况从不同队列里取出任务执行

浏览器必须准备一个微队列，微队列中的任务优先于所以其他任务执行

目前chrome中至少包含了如下队列：

* 延时队列：用于存放计时器到达后的回调任务，优先级【中】
* 交互队列：用于存放用户操作后产生的事件处理任务（点击，滚动等），优先级【高】
* 微队列：用于存放需要立即执行的任务，优先级【最高】

在浏览器中，用户交互的优先级是除了微任务以外最高的

Js中的计时器为什么不能做到精确计时？

* 计算机硬件没有原子钟，无法精确计时
* Js的计时器最终调用的时候操作系统的函数，操作系统的计时器函数本身就有偏差
* 按照W3C的标准，浏览器实现计时器时，如果嵌套层级超过5层，则会带有4ms的最小倒计时，当计时小于4ms时自然会有误差
* 受事件循环的影响，计时器的回调函数只能在主线程空闲时运行，因此又带来了误差