**事件循环**

进程：程序运行需要有自己专属的运行空间，这个运行空间称为进程

线程：运行代码的人称之为线程

一个进程至少有一个线程，所以在进程开启后会自动创建一个线程来运行代码，该线程称之为主线程；如果程序需要同时执行多块代码，主线程就会开启多个线程来执行代码

浏览器是一个多进程多线程的应用程序

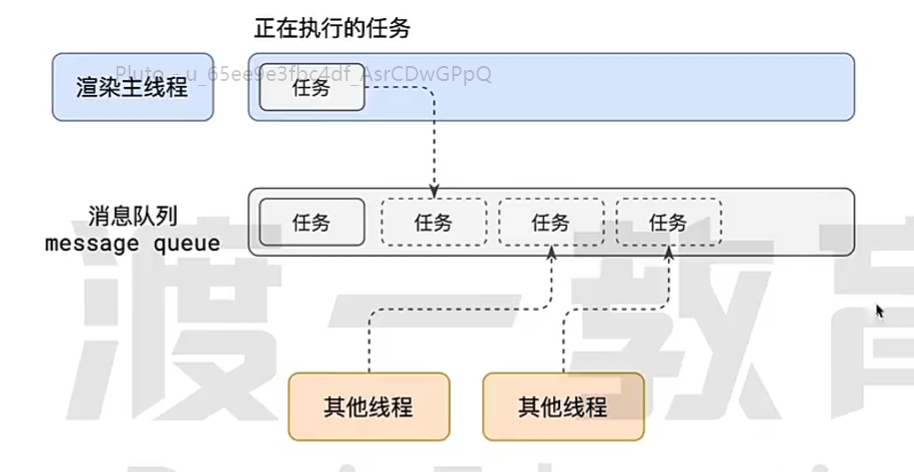
浏览器进程：主要负责界面显示（页面标签、页面按钮等）、用户交互、子进程管理等

网络进程：负责加载网络资源。

渲染进程：渲染进程启动后，会开启一个渲染主线程，主线程负责执行HTML、CSS、JS代码——浏览器会为每一个标签页开启一个新的渲染进程，各页签之间互不影响

渲染主线程：

解析HTML、解析CSS、计算样式、布局、处理图层、每秒把页面画60次、执行全局JS代码、执行时间处理函数、执行计时器的回调函数、……



何为异步：

代码在执行过程中，会遇到一些无法立即处理的任务

计时完成后需要执行的任务—— setTimeout、setInterval

网络通信完成后需要执行的任务—— XHR、Fetch

用户操作后需要执行的任务—— addEventListener

任务有优先级吗？

任务没有优先级，在消息队列中先进先出

但消息队列是有优先级的

每个任务都有一个任务类型，同一个类型的任务必须在一个队列中。

浏览器中必须要有一个微队列（VIP），微队列中的任务优先所有其他任务执行。

谷歌浏览器中的队列：

微队列：用户存放需要最快执行的任务——优先级高

交互队列：用于存放用户操作后产生的事件处理任务——优先级中

延时队列：用于存放计时器到达后的回调任务——优先级低

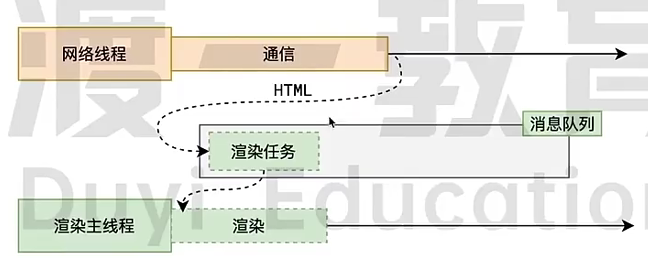
把一个函数立即放入到微队列中：

Promise.resolve().then(函数)

\*\*单线程是异步产生的原因

事件循环是异步的实现方式\*\*

浏览器渲染原理



当浏览器的网络线程收到HTML文档之后，会产生一个渲染任务，并将其传递给渲染主线程的消息队列，在事件循环机制的作用下，渲染主线程

Try…catch

书写的地址（绝对路径、相对路径）--转换🡪 完整的URL路径

绝对路径：与当前页面的path路径无关

当前页面：<https://www.baidu.com/a/b/1.html>

绝对路径：

1. 完整的URL地址：<https://www.sina.com/a.html>
2. 省略协议：//www.sina.com/a.html 🡪 <https://www.sina.com/a.html>
3. 省略协议和域名：/a.html 🡪 <https://www.sina.com/a.html>

相对路径：与当前页面的path路径有关

当前页面：<https://www.baidu.com/a/b/1.html>

相对路径：

1. 相对于当前页面path的最后一个斜杠 / 后面的路径
   * + ./2.html 🡪 <https://www.baidu.com/a/b/2.html>
     + 2.html 🡪 <https://www.baidu.com/a/b/2.html>
2. 相对于当前页面path的倒数第二个斜杠 / 后面的路径

../2.html 🡪 <https://www.baidu.com/a/2.html>

Content-Type 浏览器根据Content-Type来决定行为

。。。。。。

Content-Type根据不同的值，进行不同的处理

* Text/plain：普通的纯文本，浏览器会将响应体原封不动的显示到页面上
* Text/html：html文档，浏览器会将响应体作为页面进行解析渲染
* Text/javascript或application/javascript：js代码，浏览器会使用JS执行引擎将它解析执行
* Text/css：CSS代码，浏览器会将它视为CSS样式
* Image/png：浏览器会将它视为png图片
* Application/octet-stream：二进制数据，会触发浏览器下载功能
* Attachment：附件，会触发下载功能

AJAX 指的是在web应用程序中异步向服务器发送请求，两种发送请求方式：



Ajax 🡪 XHR Fetch

Axios 🡪 XHR

Axios 第三方库

文件上传

流式读取

跨域

源 = 协议 + 域名 + 端口

同源策略 是一套浏览器安全机制

网络通信中AJAX的跨域

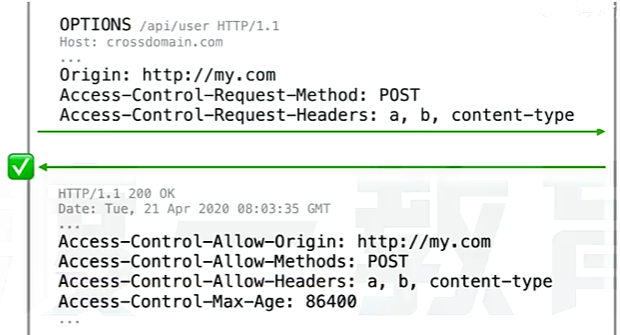
简单请求：

1. 请求方法为GET、POST、HEAD之一
2. 头部字段没有做改动
3. Content-type的值有的话，必须为以下值中的一个
   1. Text/plain
   2. Multipart/form-data
   3. Application/x-www-form-urlencoded

简单请求校验：

客户端发送请求的时候，在头部字段带上字段Origin：http://my.com(当前的页面源)

服务器在响应头中加上字段：Access-Control-Allow-Origin：<http://my.com>

预检请求：除简单请求外都是预检请求

文件上传的请求，根据上传的接口需要什么数据格式来判断，若数据格式为二进制格式或者base64格式，则为预检请求；若数据格式为multipart/form-data，则为简单请求

如何实现精准计时？