**Service workers**

本质上充当Web应用程序与浏览器之间的代理服务器，也可以在网络可用时作为浏览器和网络间的代理。它们旨在（除其他之外）使得能够创建有效的离线体验，拦截网络请求并基于网络是否可用以及更新的资源是否驻留在服务器上来采取适当的动作。他们还允许访问推送通知和后台同步API。

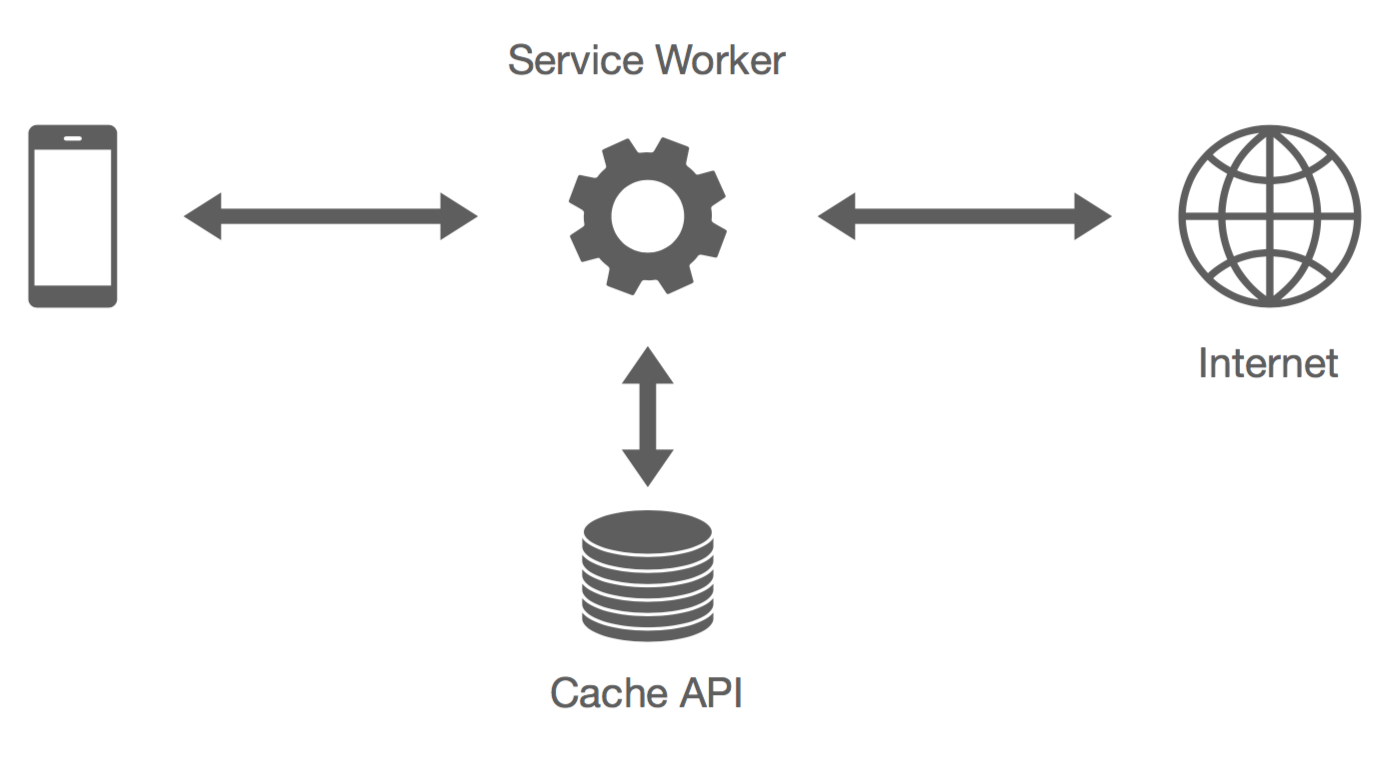
https://developer.mozilla.org/zh-CN/docs/Web/API/Service\_Worker\_API

**Web Workers API**

Worker 接口代表一个可以轻松创建的后台任务，并可以将消息发送回其创建者。创建一个工作程序只要简单的调用Worker() 构造函数，并指定一个要在工作线程中运行的脚本。

**优点：**

1. 完全异步，运行在worker上下文，不会与主JS形成阻塞
2. 只能用HTTPS承载，安全性更高
3. 推送消息功能（pushManager）
4. 强大的离线缓存能力



**兼容性**

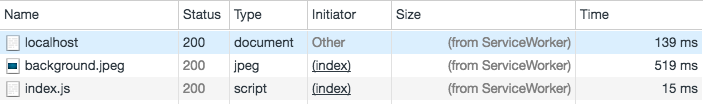


**Service Worker生命周期**

service worker拥有一个完全独立于Web页面的生命周期

1. 注册service worker，在网页上生效
2. 安装成功，激活 或者 安装失败（下次加载会尝试重新安装）
3. 激活后，在sw的作用域下作用所有的页面，首次控制sw不会生效，下次加载页面才会生效。
4. sw作用页面后，处理fetch（网络请求）和message（页面消息）事件 或者 被终止（节省内存）。





**降级方案：**

if (支持SW) {

fetch(开关接口)

.then(() => {

if (降级) {

*// 注销所有已安装的 SW*

} else {

*// 注册 SW*

}

})

}

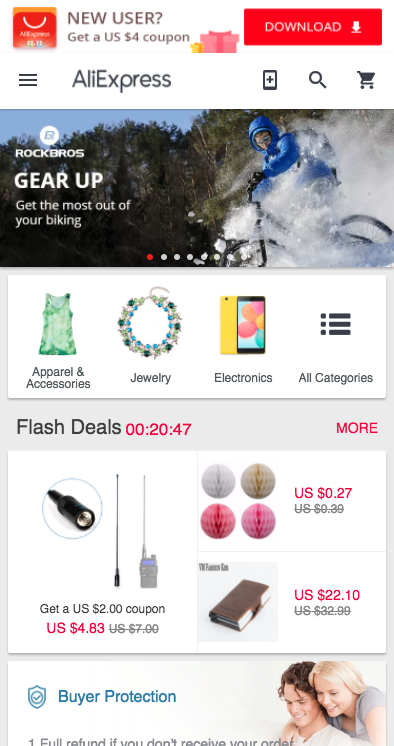
**Cache API**

https://developer.mozilla.org/zh-CN/docs/Web/API/Cache

**Fetch API**

<https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API>

**国内正在使用的网站：**

****

****