PWA

1. 概念

PWA(Progressive Web Apps)PWA，渐进式增强 WEB 应用

1. 生命周期

install -> installed -> actvating -> Active -> Activated -> Redundant

1. service worker

3.1 service worker注册

if ('serviceWorker' in navigator) {  
 navigator.serviceWorker  
 .register('./sw.js', {scope: './'})  
 .then(*registration* => console.log('ServiceWorker 注册成功！作用域为: ', *registration*.scope))  
 .catch(*err* => console.log('ServiceWorker 注册失败: ', *err*));  
}

3.2 service安装

 列举要默认缓存的静态资源，一般用于离线使用 ，建议缓存文件不要太大。太大的文件会增加安装失败率

const *version* = 'gt\_v5';  
const *offlineResources* = [  
 './',  
 './offline.html',  
 './jquery.min.js'  
 // '/wp-content/themes/Kratos/images/default.jpg'  
];

function onInstall(*event*) {

// event.waitUtil 用于在安装成功之前执行一些预装逻辑

// 但是建议只做一些轻量级和非常重要资源的缓存，减少安装失败的概率  
 *event*.waitUntil(  
 caches.open(cacheKey('offline'))  
 .then(*cache* => *cache*.addAll(*offlineResources*))  
 .then(() => log('installation complete! version: ' + *version*))  
 .then(() => self.skipWaiting())  
 );  
}

3.3 service worker 更新

如果 sw.js 文件的内容有改动，当访问网站页面时浏览器获取了新的文件，它会认为有更新，于是会安装新的文件并触发 install 事件。但是此时已经处于激活状态的旧的 Service Worker 还在运行，新的 Service Worker 完成安装后会进入 waiting 状态。直到所有已打开的页面都关闭，旧的 Service Worker 自动停止，新的 Service Worker 才会在接下来打开的页面里生效。

如果希望在有了新版本时，所有的页面都得到及时更新怎么办呢？

可以在 install 事件中执行 skipWaiting 方法跳过 waiting 状态，然后会直接进入 activate 阶段。接着在 activate 事件发生时，通过执行 clients.claim 方法，更新所有客户端上

### 3.4 Service Worker 相关事件处理

### 3.4.1 install 事件

当前脚本被安装时，会触发 install 事件

### 3.4.2 fetch 事件

这里实现了一个缓存优先、降级处理的策略逻辑：监控所有 http 请求，当请求资源已经在缓存里了，直接返回缓存里的内容；否则使用 fetch API 继续请求，如果是 图片或 css、js 资源，请求成功后将他们加入缓存中；如果是离线状态或请求出错，则降级返回预缓存的离线内容。