Atitt pwa 技术原理一些解释

caches 是一个特殊的 [CacheStorage](https://developer.mozilla.org/zh-CN/docs/Web/API/CacheStorage) 对象，它能在 Service Worker 指定的范围内提供数据存储的能力（Service Worker 在注册时，第二个参数是选填的，可以被用来指定你想让 Service Worker 控制的内容的子目录，译者注）。因为 Web Storage 的执行是同步的（此处理解为 Web Storage 并不返回一个 Promise，译者注），在 Service Worker 中使用 [Web Storage](https://developer.mozilla.org/zh-CN/docs/Web/API/Web_Storage_API) 将不会有效果，所以我们使用 Cache API 作为替代。

## 激活

还有一个 activate 事件，它的用法跟 install 事件相同。这个事件通常用来删除那些我们已经不需要的文件或者做一些清理工作。因为在我们的 App 里面没有使用到，这里我们暂时跳过它。

### [响应请求](https://developer.mozilla.org/zh-CN/docs/Web/Progressive_web_apps/Offline_Service_workers" \l "%E5%93%8D%E5%BA%94%E8%AF%B7%E6%B1%82" \o "Permalink to 响应请求)

## fetch 事件

对我们很有用，它在每次应用发起 HTTP 请求的时候被触发。这个事件对我们来说非常有用，它允许我们拦截请求并对请求作出自定义的响应，下面是一个简单的例子：

self.addEventListener('fetch', function(e) {

console.log('[Service Worker] Fetched resource '+e.request.url);});

# 离线缓存机制

[FetchEvent.respondWith](https://developer.mozilla.org/zh-CN/docs/Web/API/FetchEvent/respondWith) 方法将会接管响应控制，它会作为服务器和应用之间的代理服务。它允许我们对每一个请求作出我们想要的任何响应：Service Worker 会处理这一切，从缓存中获取这些数据，并在需要的情况下对它们进行修改。

就是这样，我们的应用会在 install 触发时缓存资源，并且在 fetch 事件触发时返回缓存中的资源，这就是它甚至在离线状态下也能使用的原因。当我们添加新的内容时，它也会随时将其缓存下来。

**适合于**： 如果整个网站无法离线工作，您可以允许用户选择他们需要离线可用的内容。 例如，YouTube 上的某个视频、维基百科上的某篇文章、Flickr 上的某个特定图库。

为用户提供一个“Read later”或“Save for offline”按钮。在点击该按钮后，从网络获取您需要的内容并将其置于缓存中。

document.querySelector('.cache-article').addEventListener('click', function(event) {

event.preventDefault();

var id = this.dataset.articleId;

caches.open('mysite-article-' + id).then(function(cache) {

fetch('/get-article-urls?id=' + id).then(function(response) {

// /get-article-urls returns a JSON-encoded array of

// resource URLs that a given article depends on

return response.json();

}).then(function(urls) {

cache.addAll(urls);

});

});

});复制代码

[caches API](https://link.juejin.cn/?target=https://developer.mozilla.org/en-US/docs/Web/API/Cache" \o "https://developer.mozilla.org/en-US/docs/Web/API/Cache" \t "_blank)可通过页面以及service workers获取，这意味着您不需要通过service workers向缓存添加内容。

#### On network r

作者：木子朗  
链接：https://juejin.cn/post/6844903927763189767  
来源：掘金  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

self.addEventListener('fetch', function(event) {

var requestURL = new URL(event.request);

event.respondWith(

Promise.all([

caches.match('/article-template.html').then(function(response) {

return response.text();

}),

caches.match(requestURL.path + '.json').then(function(response) {

return response.json();

})

]).then(function(responses) {

var template = responses[0];

var data = responses[1];

return new Response(renderTemplate(template, data), {

headers: {

'Content-Type': 'text/html'

}

});

})

);

});复制代码

目录

* [原文](" \l "heading-0" \o "原文)
* [前言](" \l "heading-1" \o "前言)
* [何时进行缓存](" \l "heading-2" \o "何时进行缓存) 
  + [On install - as a dependency](" \l "heading-3" \o "On install - as a dependency)
  + [On install - not as a dependency](" \l "heading-4" \o "On install - not as a dependency)
  + [On activate](" \l "heading-5" \o "On activate)
  + [On user interaction](" \l "heading-6" \o "On user interaction)
  + [On network response](" \l "heading-7" \o "On network response)
  + [Stale-while-revalidate](" \l "heading-8" \o "Stale-while-revalidate)
  + [On push message](" \l "heading-9" \o "On push message)
  + [On background-sync](" \l "heading-10" \o "On background-sync)
* [缓存持久化](" \l "heading-11" \o "缓存持久化)
* [缓存策略](" \l "heading-12" \o "缓存策略) 
  + [Cache only](" \l "heading-13" \o "Cache only)
  + [Network Only](" \l "heading-14" \o "Network Only)
  + [Cache First](" \l "heading-15" \o "Cache First)
  + [Cache & network race](" \l "heading-16" \o "Cache & network race)
  + [Network First](" \l "heading-17" \o "Network First)
  + [Cache then network](" \l "heading-18" \o "Cache then network)
  + [Generic fallback](" \l "heading-19" \o "Generic fallback)