Page Visibility API

目录

- 简介__
- document.visibilityState
- document.hidden
- visibilitychange 事件__
- 页面卸载_____
- 参考链接_____

简介

有时候，开发者需要知道，用户正在离开页面。常用的方法是监听下面三个事件。

> - `pagehide`
> - `beforeunload`
> - `unload`

但是，这些事件在手机上可能不会触发，页面就直接关闭了。因为手机系统可以将一个进程直接转入后台，然后杀死。

> - 用户点击了一条系统通知，切换到另一个 App。
> - 用户进入任务切换窗口，切换到另一个 App。
> - 用户点击了 Home 按钮，切换回主屏幕。
> - 操作系统自动切换到另一个 App（比如，收到一个电话）。

上面这些情况，都会导致手机将浏览器进程切换到后台，然后为了节省资源，可能就会杀死浏览器进程。

以前，页面被系统切换，以及系统清除浏览器进程，是无法监听到的。开发者想要指定，任何一种页面卸载情况下都会执行的代码，也是无法做到的。为了解决这个问题，就诞生了 Page Visibility API。不管手机或桌面电脑，所有情况下，这个 API 都会监听到页面的可见性发生变化。

这个新的 API 的意义在于，通过监听网页的可见性，可以预判网页的卸载，还可以用来节省资源，减缓电能的消耗。比如，一旦用户不看网页，下面这些网页行为都是可以暂停的。

> - 对服务器的轮询
> - 网页动画
> - 正在播放的音频或视频

document.visibilityState

这个 API 主要在 `document` 对象上，新增了一个 `document.visibilityState` 属性。该属性返回一个字符串，表示页面当前的可见性状态，共有三个可能的值。

> - `hidden`：页面彻底不可见。
> - `visible`：页面至少一部分可见。
> - `prerender`：页面即将或正在渲染，处于不可见状态。

其中，`hidden` 状态和 `visible` 状态是所有浏览器都必须支持的。`prerender` 状态只在支持“预渲染”的浏览器上才会出现，比如 Chrome 浏览器就有预渲染功能，可以在用户不可见的状态下，预先把页面渲染出来，等到用户要浏览的时候，直接展示渲染好的网页。

只要页面可见，哪怕只露出一个角，`document.visibilityState` 属性就返回 `visible`。只有以下四种情况，才会返回 `hidden`。

> - 浏览器最小化。
> - 浏览器没有最小化，但是当前页面切换成了背景页。
> - 浏览器将要卸载（unload）页面。
> - 操作系统触发锁屏屏幕。

可以看到，上面四种场景涵盖了页面可能被卸载的所有情况。也就是说，页面卸载之前，`document.visibilityState` 属性一定会变成 `hidden`。事实上，这也是设计这个 API 的主要目的。

另外，早期版本的 API，这个属性还有第四个值 `unloaded`，表示页面即将卸载，现在已经被废弃了。

注意，`document.visibilityState` 属性只针对顶层窗口，内嵌的 `<iframe>` 页面的 `document.visibilityState` 属性由顶层窗口决定。使用 CSS 属性隐藏 `<iframe>` 页面（比如 `display: none;`

），并不会影响内嵌页面的可见性。

document.hidden

由于历史原因，这个 API 还定义了 `document.hidden` 属性。该属性只读，返回一个布尔值，表示当前页面是否可见。

当 `document.visibilityState` 属性返回 `visible` 时，`document.hidden` 属性返回 `false`；其他情况下，都返回 `true`。

该属性只是出于历史原因而保留的，只要有可能，都应该使用 `document.visibilityState` 属性，而不是使用这个属性。

visibilitychange 事件

只要 `document.visibilityState` 属性发生变化，就会触发 `visibilitychange` 事件。因此，可以通过监听这个事件（通过 `document.addEventListener()` 方法或 `document.onvisibilitychange` 属性），跟踪页面可见性的变化。

```
document.addEventListener('visibilitychange', function () {
  // 用户离开了当前页面
  if (document.visibilityState === 'hidden') {
    document.title = '页面不可见';
  }

  // 用户打开或回到页面
  if (document.visibilityState === 'visible') {
    document.title = '页面可见';
  }
});
```

上面代码是 Page Visibility API 的最基本用法，可以监听可见性变化。

下面是另一个例子，一旦页面不可见，就暂停视频播放。

```
var vidElem = document.getElementById('video-demo');
document.addEventListener('visibilitychange', startStopVideo);
```

```
function startStopVideo() {
  if (document.visibilityState === 'hidden') {
    vidElem.pause();
  } else if (document.visibilityState === 'visible') {
    vidElem.play();
  }
}
```

页面卸载

下面专门讨论一下，如何正确监听页面卸载。

页面卸载可以分成三种情况。

> - 页面可见时，用户关闭 Tab 页或浏览器窗口。
> - 页面可见时，用户在当前窗口前往另一个页面。
> - 页面不可见时，用户或系统关闭浏览器窗口。

这三种情况，都会触发 `visibilitychange` 事件。前两种情况，该事件在用户离开页面时触发；最后一种情况，该事件在页面从可见状态变为不可见状态时触发。

由此可见，`visibilitychange` 事件比 `pagehide`、`beforeunload`、`unload` 事件更可靠，所有情况下都会触发（从 `visible` 变为 `hidden`）。因此，可以只监听这个事件，运行页面卸载时需要运行的代码，不用监听后面那三个事件。

甚至可以这样说，`unload` 事件在任何情况下都不必监听，`beforeunload` 事件只有一种适用场景，就是用户修改了表单，没有提交就离开当前页面。另一方面，指定了这两个事件的监听函数，浏览器就不会缓存当前页面。

参考链接

- Page Visibility Level 2, W3C
- Page Visibility API, David Walsh
- Using the pageVisbility API, Joe Marini
- Using PC Hardware more efficiently in HTML5: New Web Performance APIs, Part 2, Jatinder Mann
- Don't lose user and app state, use Page Visibility, Ilya Grigorik

□□□ #2
□ □□□ □□□ 2018-10-24 16:20:12 UTC
□ □□□ □□□ 2018-11-05 00:04:40 UTC