Page Visibility API

Quick Guides for Masterminds

J.D Gauchat
www.jdgauchat.com

Cover Illustration by **Patrice Garden**
www.smartcreativz.com

The content of this guide is a collection of excerpts from the book HTML5 for Masterminds. For more information, visit www.formasterminds.com.

The source code for this eBook is available at **www.formasterminds.com**

Copyright Registration Number: 1140725

1st Edition 2018

What is Inside

This guide will teach you how to use the Page Visibility API to detect whether a web page is visible or not. After reading this guide, you will know how to detect when the user hides the page with another tab or window, and how to respond to changes in the visibility state.

About this Guide

This guide is a collection of excerpts from the book HTML5 for Masterminds. The information included in this guide will help you understand a particular aspect of web development, but it will not teach you everything you need to know to develop a website or a web application. If you need a complete course on web development, read our book HTML5 for Masterminds. For more information, visit our website at www.formasterminds.com.

What Do You Need

This guide assumes that you have a basic knowledge of HTML, CSS and JavaScript, and you know how to create files and upload them to a server. If you don't know how to program in HTML, CSS or JavaScript, you can download our guides Introduction to HTML, Introduction to CSS, and Introduction to JavaScript. For a complete course on web development, read our book HTML5 for Masterminds.

IMPORTANT: We recommend you to execute the examples in this guide with the latest versions of Google Chrome and Mozilla Firefox (www.google.com/chrome and www.mozilla.com). You can also check the state of the current implementations at www.caniuse.com. To find examples, resources, links and videos, visit our website at www.formasterminds.com.

The Basics: To create the files for the examples of this guide, you can use a text editor (Notepad or Text Edit), or a professional editor like Atom (www.atom.io). If you do not have a server to test the files, you can install a server in your computer with a package like MAMP (www.mamp.info). For more information, read our free guide Web Development.

Table of Contents

PAGE VISIBILITY API
Visibility
   State
   Full Detection System

QUICK REFERENCE
   State
   Events

Page Visibility API

Visibility

Web applications are getting more sophisticated and are demanding computer resources as never before. Web pages are no longer static documents; JavaScript has turned them into full applications, capable of executing complex processes without interruption, and even without the user's intervention. But there are moments when these processes should be canceled or paused to effectively distribute the resources for a better user experience. To produce applications that are conscious of its own state, browsers include the Page Visibility API. This API informs the application about the current visibility state of the document, reporting when the tab is hidden, or the window is minimized, so our code can decide what to do while nobody is watching.

State

The API provides a property to report the current state and an event to let the application know that something changed.

visibilityState—This property returns the current visibility state of the document. The possible values are **hidden** or **visible** (optional values such as **prerender** and **unloaded** may also be implemented by some browsers).

visibilitychange—This event is fired when the value of the **visibilityState** property changes.

The API is part of the **Document** object, and therefore it is accessible from the **document** property of the **Window** object. The following document implements the property and the event provided by the API to detect when the user moves to a different tab, and it modifies the content of the page to report the new state.

Listing 1: *Reporting the visibility state*

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Page Visibility API</title>
  <script>
    function initiate() {
      document.addEventListener("visibilitychange",
showstate);
    }
    function showstate() {
      var element = document.getElementById("application");
      element.innerHTML += "<br>" +
document.visibilityState;
    }
    window.addEventListener("load", initiate);
  </script>
</head>
<body>
  <section id="application">
    <p>Move to another tab or minimize this window to
change the visibility state</p>
  </section>
</body>
</html>
```

In the code in Listing 1, we add a listener for the **load** event to the window to execute a function called **initiate()** as soon as the document is loaded. This function assigns the **showstate()** function to handle the **visibilitychange** event. When the event is fired, the function shows the value of the **visibilityState** property on the screen to inform the current state of the document. When the tab is replaced with another tab or the window is minimized, the value of the v**isibilityState** property changes to **hidden**, and when the tab or the window is restored, the value is set back to **visible**.

Do It Yourself: Create a new HTML file with the document of Listing 1 and open the document in your browser. Open a new tab or move to another tab already opened. Move back to the tab that contains the application. You should see the words hidden and visible printed on the screen (the document was hidden when you move to another tab and is visible again when you come back).

One of the reasons for the implementation of this API was the high consumption of resources in modern applications, but the API was also designed as a way to enhance the user experience. The next example shows how to achieve the latter with just a few lines of code.

Listing 2: *Responding to the visibility state*

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Page Visibility API</title>
  <script>
    var video;
    function initiate() {
      video = document.getElementById("media");
      document.addEventListener("visibilitychange",
showstate);
      video.play();
    }
    function showstate() {
      var state = document.visibilityState;
      switch(state) {
        case "visible":
          video.play();
          break;
        case "hidden":
          video.pause();
          break;
      }
    }
    window.addEventListener("load", initiate);
  </script>
</head>
<body>
  <video id="media" width="720" height="400">
    <source src="trailer.mp4">
    <source src="trailer.ogg">
  </video>
</body>
</html>
```

The code in Listing 2 plays a video while the document is visible and pauses it when it is not. We use the same functions from the previous example, except this time the value of the **visibilityState** property is checked by a **switch** instruction and the **play()** or **pause()** methods are executed to play or pause the video according to the current state.

Do It Yourself: Update the code in your HTML file with the document of Listing 2. Download the trailer.mp4 and trailer.ogg files from our website. Open the new document in your browser and alternate between tabs to see the application in action. The video should pause when the document is not visible and resume when it is visible again.

Full Detection System

Browsers do not change the value of the **visibilityState** property when the user opens a new window. The API is only capable of detecting the change in visibility when the tab is hidden by another tab, or the window is minimized. To determine the visibility state in any circumstance, we can complement the API with the following events provided by the **Window** object.

blur—This event is fired when the window loses focus (it is also fired by independent elements).

focus—This event is fired when the window regains focus (it is also fired by independent elements).

With the addition of a small function, we can combine all the tools available to build a better detection system.

Listing 3: *Combining the* blur*,* focus *and* visibilitychange *events*

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Page Visibility API</title>
  <script>
    var state;
    function initiate() {
      window.addEventListener("blur", function() {
        changestate("hidden");
      });
      window.addEventListener("focus", function() {
        changestate("visible");
      });
      document.addEventListener("visibilitychange",
function() {
        changestate(document.visibilityState);
      });
    }
    function changestate(newstate) {
      if (state != newstate) {
        state = newstate;
        showstate();
      }
    }
    function showstate() {
      var element = document.getElementById("application");
      element.innerHTML += "<br>" + state;
    }
    window.addEventListener("load", initiate);
  </script>
</head>
```

```
<body>
  <section id="application">
    <p>Move to another window to change the visibility
state</p>
  </section>
</body>
</html>
```

In the **initiate()** function in Listing 3, listeners for the three events are added. The **changestate()** function was defined to respond to the events and process the value corresponding to the current state. This value is determined by each event. The **blur** event sends the value **hidden**; the **focus** event sends the value **visible**, and the **visibilitychange** event sends the current value of the **visibilityState** property. In consequence, the **changestate()** function receives the right value, no matter how the document was hidden (by another tab, window or program). By using the **state** variable, the function compares the previous value with the new one, stores the new value in the variable and calls the **showstate()** function to show the state on the screen.

This process is a little more complicated than the previous one but considers every possible situation in which the document may be hidden, and the state changed.

Do It Yourself: Update the code in your HTML file with the document of Listing 3. Open the new document in your browser, and alternate between the browser window and another window or program. You should see the words hidden and visible printed on the screen reflecting the change in states.

Quick Reference

State

visibilityState—This property returns the current visibility state of the document. The possible values are **hidden** or **visible** (optional values such as **prerender** and **unloaded** may also be implemented by some browsers).

Events

visibilitychange—This event is fired when the value of the **visibilityState** property changes.

blur—This event is fired when the window loses focus (it is also fired by independent elements).

focus—This event is fired when the window regains focus (it is also fired by independent elements).