Page Visibility

There is nothing more anoying than having audio/video playing in a tab when it's in the background or when the browser is minimized.

The Page Visibility API gives us control of what to do with media or any other element of a page when the tab is hidden or not visible. Some of the things we can do with this API:

- Pausing a video when the page has lost user focus.
- Stop an HTML5 canvas animation from running when a user is not on the page.
- Show a notification to the user only when the page is active.
- Stop movement of a slider carousel when the page has lost focus.

The API introduces two new attributes to the `document` element: `document.visibilityState` and `document.hidden`.

`document.visibilityState` holds four different values which are as below:

- **hidden**: Page is not visible on any screen
- **prerender**: Page is loaded off-screen and not visible to user
- **visible**: Page is visible
- **unloaded**: Page is about to unload (user is navigating away from current page)

`document.hidden` is boolean property, that is set to false if page is visible and true if page is hidden.

The first example pauses a video if the container tab is hidden or not visible and plays it otherwise.

We start by adding a `visibilitychange` event listener to the document. Inside the listener we check if the document is hidden and pause the video if it is; otherwise we play it.

```
video = document.getElementById('myVideo');

document.addEventListener('visibilitychange', function () {
```

```
    if (document.hidden) {
        video.pause();
    } else {
        video.play();
    }
});
```

The most obvious use is to control video playback when the tab is not visible. When I wrote about creating custom controls for HTML5 video I used a click event handler like this one to control video play/pause status:

```
play.addEventListener('click', e => {
  // Prevent Default Click Action
  e.preventDefault();
  if (video.paused) {
    video.play();
    playIcon.src = 'images/icons/pause.svg';
  } else {
    video.pause();
    playIcon.sr'images/icons/pause.svg'utton.svg';
  }
});
```

We can further enhance it so that it will only play the video if the document is visible.

We wrap the if block that controls playback in another if block that tests the page's visibility state and only moves forward if we can see the page. If the page is not visible then we pause the video, regardless of whether it's currently playing.

The code now looks like this:

```
play.addEventListener('click', e => {
  // Prevent Default Click Action
  e.preventDefault();
  if (document.visibilityState === 'visible') {
    if (video.paused) {
```

```
      video.play();
      playIcon'visible'mages/icons/pause.svg';
    } else {
      video.pause();
      playIcon.src = 'images/icons/play-button.svg';
    }
  } else {
    video.pause();
    playIcon.src = 'images/icons/play-button.svg';
  }
});
```

With those simple changes we've ensured that the video will not play in the background and that there will be no other distractions when we work on other tabs. We then should do something similar for our keyboard video controls.