Harnessing the Page Visibility API with React

Written by Seth Corker on Benevolent Bytes

The web is a powerful platform that has more to offer than you might expect. There are many APIs that enrich people's experience of the web allow developers to make websites that react in fun and interesting ways.

The Page Visibility API is easy to use but often overlooked. It's a useful tool which can be used to save battery, conserve data or detect the engagement of a user. Let's take a look at what the API does and why you might want to take advantage of it for your project.

What is it?

Detect when a browser tab becomes active/inactive.

That's it. Plain and simple, when using a browser with multiple tabs, the API will detect whether the tab is active or not. When the user switches tabs, the Page Visibility API fires off a JavaScript event which we can listen to. When they come back, we receive another event to let us know the tab is active again.

It it's difficult to image then take a look at the demo;

Why is it useful?

It might not be immediately obvious why you might want to whether the tab is active or inactive but there are some common cases that you should consider.

Save power

A video is playing, maybe a background video that is secondary to the experience. You could pause the video when the user switches tabs and resume it when they come back. This could be done with a carousel too.

Conserve data

You have a dashboard which makes requests to an API every few seconds to stay updated. Why should it update when the user isn't looking at it? You could save the user's data by polling less frequently or stopping the polling entirely until the tab becomes active again.

How do I use it?

A custom React hook was setup to make it easier to use.

Helper methods

A few helper methods were created to abstract the browser differences. The event handler and the property on the document are different depending on the browser so `getBrowserVisibilityProp` and `getBrowserDocumentHiddenProp` are used to ensure we add the correct listener and check the correct prop.

```
export function getBrowserVisibilityProp() {
  if (typeof document.hidden !== "undefined") {
    // Opera 12.10 and Firefox 18 and later support
    return "visibilitychange"
  } else if (typeof document.msHidden !== "undefined") {
    return "msvisibilitychange"
  } else if (typeof document.webkitHidden !== "undefined") {
    return "webkitvisibilitychange"
  }
}

export function getBrowserDocumentHiddenProp() {
  if (typeof document.hidden !== "undefined") {
    return "hidden"
  } else if (typeof document.msHidden !== "undefined") {
    return "msHidden"
  } else if (typeof document.webkitHidden !== "undefined") {
    return "webkitHidden"
```

```
  }
}

export function getIsDocumentHidden() {
  return !document[getBrowserDocumentHiddenProp()]
}
```

The hook itself

The hook is pretty basic, we listen to the visibility change event and store the result in state before returning it. Notice the return value of the `useEffect` hook cleans up the listener.

```
export function usePageVisibility() {
  const [isVisible, setIsVisible] = React.useState(getIsDocumentHidden())
  const onVisibilityChange = () => setIsVisible(getIsDocumentHidden())

  React.useEffect(() => {
    const visibilityChange = getBrowserVisibilityProp()

    document.addEventListener(visibilityChange, onVisibilityChange, false)

    return () => {
      document.removeEventListener(visibilityChange, onVisibilityChange)
    }
  })

  return isVisible
}
```

Using the hook

Once the hook is setup, using it is easy.

```
const isVisible = usePageVisibility()
```

Your component can take advantage of the visibility state, whenever it changes so too will the `isVisible` variable. Use it to pause animations, videos and, carousels or pause fetching until the tab is active again. There are lots of possibilities so why not give it a go.

Resources:

- https://developer.mozilla.org/en-US/docs/Web/API/Page_Visibility_API
- Want the source for the demo? Take a look at the repository on GitHub.

Updates:

The hook code snippet was updated on{" "} 17/09/2020 due to a suggestion by{" "} @tysonmatanich !