WebVR is a javascript API for creating immersive virtual reality (VR) experiences in browsers. In this paper I will first provide a general introduction about VR, following with a critical analysis of WebVR.

VR is about hacking user's perception system to fully immerse the user into a virtual experience. It often leverages computer graphics, optics and sensors to provide visual and auditory experience that let the users suspend their disbelief that they are experiencing something virtual. This suspension of disbelief is what differentiates VR from all other preceding media such as television and book. When you are watching a movie, you can clearly differentiates yourself from the settings happened in the screen; but with evolved VR technologies that are designed to fool human's perception system, your subconscious is tricked to believe that what you are experiencing is real, as demonstrated in this video. Oculus, HTC Vive, Sony PlayStation, and Google Daydream are the four major VR platforms in the market and users consume VR contents through native app stores.

WebVR takes a different approach that allows the user to directly access the contents without installation. This approach has several advantages. First, it provides an easy integration with an existing web app. Even React, the front-end framework that we are using in COMP531, has a VR component called React VR that allows me to integrate a VR page in my front-end app. Since we've already familiar with Javascript's syntax and idioms, WebVR also has a gentle learning curve for millions of web developers like me. Second, web contents are much more accessible than the ones in curated app store. This universal accessibility is beneficial for both the developers and the users. Developers now can deploy the app anytime they want and the content is immediately accessible to the end user without waiting for the review by various app stores. Consumers now do not have to waiting for the installation and can share the contents with their friend with URLs. Third, your code is now automatically cross-platform without tailoring to various operating systems and VR hardware since the library handles have already handled this for you.

However, WebVR also has its own fallbacks that limits its use cases as a VR platform. The biggest limitation is performance. Browsers are not solely designed for VR experiences and its design could limits the performance as a VR rendering engine. For example, rendering in Google's Chrome browser has been capped at 60 FPS, which put WebVR experiences far behind the 90 FPS bar set by Oculus Rift and HTC Vive. Second, developers are forced to use a big development stack, such as WebGL and Javascript, that they might not want to use. Third, this is relatively new platform and not all browsers support the standards. The APIs are subject to change.

Given the above pros and cons, I find WebVR are most suitable for the following use cases. First, 360 degree video and images. It is natural to adjust the camera by rotating your head and can provide much better user experience than the dragging approach. Second, 3D visualization. For example, consumers can get a much better experience if they can view the item in 3D when they are shopping online. Sites can display an interactive 3D model on screen to viewing the walkthrough in VR, giving users the impression of actually being present in the house.

In all, WebVR is easy to integrate with an existing web app and has a lot less hassles for developers and consumers. Even though performance issues and immature APIs limit its use cases, it's very suitable for displaying 360 video and 3D visualization.

Reference:

1. http://www.roadtovr.com/google-chrome-webvr-htc-vive-90-fps-hz-chrome/
2. https://github.com/w3c/webvr/blob/master/explainer.md