Future plans

In its current state, the extension does not yet allow for all the types of enhancements listed so far. In addition, there are some features and use-cases worth discussing more thoroughly that go beyond creation of a single browser extension.

Site translations

This is a potentially very useful feature, but unfortunately it might cause legal problems. Translated version of websites' text would still be covered by copyright of the original and could hence be impossible to distribute legally. In many cases copyright holders of text available online would be OK with someone translating it but asking for permission every time would definitely be unfeasible. A solution could be to encode translations in some way so that the actual translated text can only be reproduced with access to the original. This way we could claim it is only translation and not the copyrighted original that is distributed. Wayback Machine could be leveraged to produce translation of a site that ceases to be available.

Port to Manifest v3

Google is introducing a breaking change to its WebExtensions ecosystem. The new extensions format differs significantly from the previous one, so do the APIs provided. Mozilla follows this path and is also going to implement what is called "Manifest v3".

There's no point hiding that at least one of this change's goals is making things harder for ad blockers (and hence, also content blockers). While it is going to be a hurdle to implement proper content blocking and substitution in this scenario, it should be achievable and we still want to support Chromium-based browsers for as long as reasonably possible.

It is worth mentioning that Mozilla, unlike Google, claims to have plans to retain the webRequest API extensively used by current content-blocking browser extensions. In the end, this might lead us to supporting less features in Chromium-based browsers and more in Gecko-based or even WebKit-based[1] if we get to implementing an analogous functionality there.

Platform for sharing custom content

Besides being able to store pages settings locally and import/export them in JSON format, Haketilo is able to query its repository server, Hydrilla, to find more custom site resources.

The ultimate goal of Hydrilla is to be a platform where anyone can contribute scripts and other works to make internet more user-controlled. We also need some means to impose guidelines and prevent spam and abuse on the platform. A scheme where long-time, credited contributors are more privileged while newcomers' contributions, however welcome, need to be approved, is likely to work best in this case. We are going to base on the practices developed by projects like Debian and Wikipedia that are known to be dealing with this issue well.

The repository software is also released under a libre license. Even though we, project authors, are running our (soon-)official Hydrilla server, the extension (or any other client-side software we develop to work with Hydrilla) is going to make it easily possible for a user instead connect to a third-party repository or even multiple ones. While it is nice and sometimes beneficial to be "the only player", it would be against the spirit of software freedom and free culture to hamper the creation of alternative repositories by those who desire so. If someone can do this better than we, the founders, can, that person should be allowed to.

It is worth mentioning we, current project members, are very concerned about software freedom. It might happen that other, third-party repositories that appear over time will have looser conditions, maybe even allowing proprietary scripts. While this is unfortunate and such repositories should not be endorsed, they should be allowed to operate freely.

Support for WebKit-based[1] browsers

Currently supported Firefox- and Chromium-based browsers are controlled by big entities and may go into the wrong direction in the future. Internet users' freedom should not depend on them and it is certainly not enough if we support just these browsers.

WebKit is a browser engine commonly used in free software projects, partially due to its easy embedding in other applications. There are several minor web browsers based on it. While in the case of Mozilla and Google browsers the most feasible approach to achieve our goal is to develop a WebExtension, in the case of WebKit it is going to be relatively easy to build the required functionality into the actual browser. This brings along a great advantage of not having to rely on some overly restrictive JavaScript APIs. In fact, if this project succeeds, it could make sense for it to start a completely new browser based on WebKit, using the format developed here for extensions.

While there also exist custom browsers based on Firefox and Chromium, they suffer from Google's great influence. Some would

point out that WebKit is way behind the 2 major browser engines in terms of features. While true, this is relatively harmless for our project, since the goal is to be able to run custom JavaScript with web pages and such custom scripts can be further customized to work despite lack of some fancy features WebKit didn't implement. Additionally, a smaller number of features might be a feature of itself, as described below.

Creation of a new, simplified Web standard

Besides the issue of nonfree JavaScript and lack of user control, the "Web" has yet another issue - it is increasingly complex and is getting even worse in that matter. Someone actually made the effort to compute a word count of all "Web" standards combined. This complexity leads to web browsers becoming operating systems of their own, with even big enterprises like Microsoft being unable to keep up with the changes and resorting to using Google's implementation.

Current state of affairs leads to vendor lock-in and is generally dangerous to the entire WWW ecosystem. Some responses to this issue have appeared. One of them is the Gemini protocol which is lighter than the widespread HTTP/HTML/CSS/JavaScript stack. Such project are good and should be promoted. However, even the Gemini Project's overview states it is not going to replace the "Web". What will, then?

A feasible replacement for the "Web" needs to be something that will be appealing to both enterprises and non-technical users. It needs to be sufficient for things like content uploading and user login. One suggestion is to standardize a modest subset of HTML and CSS (without JavaScript) that would be sufficient for most of the use-cases of current "Web". Some important features have been added to HTML and CSS over the years that enable creation of good-looking, responsive content without use of JavaScript. These would be retained. At the same time some advanced use-cases like those that rely on WebSockets or WebRTC would simply be out of this new standard's scope (although a lightweight, compliant browser engine could still be embedded into standalone applications using these technologies).

Once embraced by businesses and organizations, such standard would allow for much more secure, lightweight and stable websites. At the same time, it could be used with most of existing tools. While convincing decision-makers to use it might seem like a difficult task, possible advantages are big enough that intelligent people are expected to consider it seriously. A good name is also important; "Web Zero" seems pretty catchy. Just as Coca-Cola Zero has been advertised as a drink that can help people lose weight, this standard could help "Web" become more lightweight and easier to develop tools for.

How does this all relate to this browser extension and project in general? In the case of most websites, their owners are obviously not going to instantly make them available on Web Zero. The possibility of making custom interfaces for existing websites has already been mentioned. Taking the idea one step further, it would be possible to adapt Hydrilla to also facilitate sharing code capable of proxying certain websites into Web Zero.

While this is indeed a very distant idea, it presents a perfectly valid way this project could help replace the "Web" with a lighter one.

---

1. Throughout this page we use name "WebKit" to refer to Apple WebKit and not Google's fork of it ↩