Site fixes - Site script request/donation #118

Various Small Scripts

05/04/2022 03:16 AM - jacobk

Status:	Closed	**Start date:**	05/04/2022
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

I haven't had as much free time as I would like, so I haven't worked on the Box script much (and I no longer have access to the folders). But I have come across various sites that are missing some functionality without running JavaScript, and in many cases fixing the site with free JavaScript is nearly trivial or very easy, so I have several scripts to submit, including partial fixes for Anbox (viewing the FAQ), Ars Technica (reading comments), Grammarly (dismissing the ad and viewing images (uh, including tracking pixels)), Gyfcat (controlling videos), and more. Many of the scripts are only a few lines. What would be the recommended way to submit these scripts? Should I attach each one to a separate issue (~12 of them), or would it be better if I bundled everything together somehow and submitted it here?

History

#1 - 05/11/2022 03:26 PM - koszko

Sorry again for the delay (just modified my email notification settings in Redmine, so I should be more responsive from now on).

Sure, you can submit them in a single issue. You can also put them in some vcs and just post here a link if that's comfortable for you.

In the end, I will make them all into Hydrilla source packages in REUSE-compliant repositories like here or here (except I will switch to using separate repositories for distinct packages). You could produce such Hydrilla source packages by yourself to make it easier for me but this is by no means necessary

#2 - 06/04/2022 08:04 PM - jacobk

Alright, I've uploaded the scripts to Codeberg: https://codeberg.org/JacobK/site-fixes

Sorry it took me so long, haha. I was able to build all of the fixes with hydrilla-builder, but I'm not sure how to import them to the extension? I guess the import json option was removed? So I'm not sure what the best way to test the scripts is, other than manually copy/pasting like I have been doing.

Most of the scripts are very short, but they all improve the websites they target in some way.

There are some scripts that may not be fit for inclusion:
* The Anbox script is inferior to the scripts that come with the page, and the scripts that come with the page may be free software, so maybe including those would be better. This may be the case for some other scripts too, but I didn't look into it for most pages.
* The Grammarly script seems to make some tracking pixels appear, but I think the website could have just sent normal tracking pixels as part of the HTML, without JavaScript, so this doesn't seem like a problem unless loading the pixel through Haketilo bypasses anti-tracking mechanisms (not sure if it does).
* The Mojang Jira and Nintendo fixes primarily support getting information about proprietary software. Though, this could include information about malware in proprietary software, for example here: https://bugs.mojang.com/browse/MC-237493 (Okay, maybe it's a stretch to call this malware, but it is arguably an anti-feature at least.)
* The TengoInternet fix is for connecting to Wi-Fi, and it it untestable when not at a Wi-Fi portal that uses that software. Additionally, the fix only makes the terms of service visible; connecting to Wi-Fi is possible without JavaScript in this case. (Dealing with scripts to actually connect to Wi-Fi might be challenging because 1. they are hard to test, 2. they may require strange URL patterns, possibly involving local ip addresses where we would see collisions, and 3. The user cannot connect to Wi-Fi to download the script until after they run the script.)
* I wrote the tp link fix primarily to support downloading proprietary software, but some of the downloaded software is free.

I think all of the scripts should be included (otherwise, I wouldn't have uploaded them and linked them here), but I could see these reasons as potentially preventing the scripts from being accepted.

#3 - 06/10/2022 12:44 PM - koszko

> I'm not sure how to import them to the extension? I guess the import json option was removed?

Yes. Or rather, the extension was redesigned and new UI was made for which such import option has not yet been implemented.

> So I'm not sure what the best way to test the scripts is, other than manually copy/pasting like I have been doing.

Another way is to run Hydrilla on localhost, add http://localhost: to the repos in Haketilo, then visit the site and install scripts from the popup. This way we can also test the actual Hydrilla packaging of the scripts.

I admit that with the requirement to actually visit the site first this is still pretty limited and that I also made some other bad design decisions here. I will of course work to improve those.

> I think all of the scripts should be included (otherwise, I wouldn't have uploaded them and linked them here), but I could see these reasons as potentially preventing the scripts from being accepted.

I think the technical issues with the wifi captive portal make us effectively unable to distribute it through Hydrilla. I see no problem with the rest of the scripts :)

#4 - 06/21/2022 02:46 PM - koszko

I just released the libre ReCAPTCHA client for Haketilo, so I can focus on integrating your scripts now :)

#5 - 06/22/2022 06:44 PM - koszko

With respect to Nintendo website, I found that:

Performing a search can be done by navigating to a page like /app/answers/list/st/5/kw/URLENCODED_SEARCH_TERM/page/7.

'7' can be replaced with any results page number in this case.

Site's original JS loads the search results without reloading the page, with a request similar to this one:

```
await fetch("https://en-americas-support.nintendo.com/ci/ajaxRequest/getReportData", {
    "credentials": "include",
```

```
    "headers": {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0",
        "Accept": "*/*",
        "Accept-Language": "en-US,en;q=0.5",
        "X-Requested-With": "xmlhttprequest",
        "RNT_REFERRER": "https://en-americas-support.nintendo.com/app/answers/list/search/1/kw/mario/search/1"
,
        "Content-Type": "application/x-www-form-urlencoded; charset=UTF-8",
        "Sec-Fetch-Dest": "empty",
        "Sec-Fetch-Mode": "cors",
        "Sec-Fetch-Site": "same-origin",
        "Pragma": "no-cache",
        "Cache-Control": "no-cache"
    },
    "referrer": "https://en-americas-support.nintendo.com/app/answers/list/search/1/kw/mario/search/1",
    "body": "
filters=%7B%22recordKeywordSearch%22%3Atrue%2C%22searchType%22%3A%7B%22filters%22%3A%7B%22rnSearchType%22%3A%2
2searchType%22%2C%22fltr_id%22%3A5%2C%22data%22%3A5%2C%22oper_id%22%3A1%2C%22report_id%22%3A126393%7D%2C%22typ
e%22%3A%22searchType%22%7D%2C%22keyword%22%3A%7B%22w_id%22%3A%22KeywordText_5%22%2C%22filters%22%3A%7B%22searc
hName%22%3A%22keyword%22%2C%22data%22%3A%22mario%22%2C%22rnSearchType%22%3A%22keyword%22%2C%22report_id%22%3A1
26393%7D%7D%2C%22p%22%3A%7B%22w_id%22%3A%22NintendoProdCatSearchFilter_12%22%2C%22filters%22%3A%7B%22rnSearchT
ype%22%3A%22menufilter%22%2C%22searchName%22%3A%22p%22%2C%22report_id%22%3A126393%2C%22fltr_id%22%3A2%2C%22ope
r_id%22%3A10%2C%22data%22%3A%5B%5B%5D%5D%7D%7D%2C%22c%22%3A%7B%22w_id%22%3A%22NintendoProdCatSearchFilter_13%2
2%2C%22filters%22%3A%7B%22rnSearchType%22%3A%22menufilter%22%2C%22searchName%22%3A%22c%22%2C%22report_id%22%3A
126393%2C%22fltr_id%22%3A3%2C%22oper_id%22%3A10%2C%22data%22%3A%5B%5B%5D%5D%7D%7D%2C%22page%22%3A3%2C%22report
_id%22%3A126393%2C%22per_page%22%3A0%7D&report_id=126393&r_tok=ZlV6RzhhfnV0N01xMHlUNDRiUWQ0c2RpOElaQTRraH5Bfnd
pdXh4M3l2UWx2Ylh5RXlaMjk5Rk1EMTlsalZKQWdJWnRTNVJleDBJN05aMDUybDlRU2VJbEVCYTFoMmY2NlBMTlNmTzRsMVE0SERSaXJDTjQzc
1BnSHd2YWdtVmlWMG9vcHJqZXlsNDk0SXU4Z2xWX0JuVn4zU1loRUU1cTB1&format=%7B%22truncate_size%22%3A200%2C%22max_wordb
reak_trunc%22%3Anull%2C%22emphasisHighlight%22%3Atrue%2C%22dateFormat%22%3A%22short%22%2C%22urlParms%22%3A%22%
2Fkw%2Fmario%22%2C%22parmList%22%3A%22kw%2Cp%2Cc%22%7D",
    "method": "POST",
    "mode": "cors"
});
```

Making our simple fix work the same would be of course too much work, so I'll instead stick to the simpler approach with reloading

#6 - 06/23/2022 11:00 AM - koszko

Your slated.org fix removes the #seckit-noscript-tag element. However, I couldn't find such element (neither any JS notice) when visiting the page. What was needed (both on the original slated.org and its archive.org snapshot) was the removal of the link#seckit-clickjacking-no-body element which hid page's body.

I wonder why we saw different things in the page. Recent updates to the website cannot be a reason because if this was the case, I'd see the #seckit-noscript-tag element in the archive.org snapshot (and also because the site looks like it hasn't been updated for a long time)

#7 - 06/23/2022 04:07 PM - jacobk

- File Screenshot at 2022-06-23 08-49-41.png added

The #seckit-noscript-tag appears inside of a on the archive.org version I linked as an example. Archive.org snapshots shouldn't differ at all in view-source: right? If you just looked at the inspector console maybe the settings that control whether you see noscript elements affects that. Though, I am not sure why my fix works on the archive.org snapshot when I can also see the CSS file that should make the body invisible.

I don't see any noscript tags in the view-source: of the live version, but I noticed something stranger: there appear to be 2 body elements. The second body is after the CSS file link that should hide the body, so that one I can understand why the fix might work (if the CSS applies only to the first body but the last body is what is shown). Actually, there is only one body shown in view-source: but an error is shown saying the body was opened twice. I have uploaded a screenshot of what I see in the view-source: for the live page.

#8 - 06/23/2022 07:21 PM - koszko

> Archive.org snapshots shouldn't differ at all in view-source: right?

In general they do. Wayback Machine alters the HTML in a few ways. But it would not change this particular detail.

> If you just looked at the inspector console maybe the settings that control whether you see noscript elements affects that.

Indeed. Actually, I don't think Firefox has a feature of showing <noscript> elements nowadays. Most extensions that block scripts do something to force these into visibility. For example, they might convert <noscript>s to s.

Haketilo currently does not force <noscript> elements. I browse with uBlock Origin which handles displaying of <noscript> elements for me. And, of course, when testing a fix I make additional attempts without uBO.

On slated.org, with uBO I didn't see the JS message because the <noscript> tag is inside <head>. Idk what your extensions are doing that causes it to appear.

Interestingly, uBO also makes the #seckit-clickjacking-no-body stylesheet ineffective. Its removal was only needed when I was testing without uBO.

Anyway, one thing is clear now: both #seckit-noscript-tag and #seckit-clickjacking-no-body might cause problems or not depending on user's set of blocking extensions.

> Though, I am not sure why my fix works on the archive.org snapshot when I can also see the CSS file that should make the body invisible.

I bet it's the job of either NoScript or LibreJS

#9 - 06/24/2022 02:32 AM - jacobk

> In general they do. Wayback Machine alters the HTML in a few ways. But it would not change this particular detail.

Oops, by "snapshots shouldn't differ" I meant the same snapshot shouldn't differ depending on who is looking at it (as opposed to a live web page that could serve different HTML based on ip address or something). The way I worded it was confusing.

Regarding the noscript showing up despite being in the head, it seems that LibreJS modifies the page HTML (even in view-source:) to show the element, but I think LibreJS gets confused. Based on the screenshot I uploaded, it seems like LibreJS tries to replace the noscript element with a span element, but for some reason it ends the span element after the link, and it then decides to end the head and start the body (I am not sure if this is intentional or not, or if it should be considered a bug in LibreJS. I guess there are some cases where unmodified browsers (maybe Internet Explorer or something) would render noscript tags in the head, because otherwise why would Slated put the JavaScript message there?). It doesn't open another span tag, so the h1 element with the text saying JavaScript must be enabled appears outside of the span entirely. I guess maybe LibreJS sees the noscript tag in the head with a link in it and thinks that's fine, but when a div appears, those aren't supposed to be in the head, so LibreJS ends the head (and the noscript) immediately. When I use view-source: I do not see the noscript tags because (I think) the page source is being affected by LibreJS before I see it. HAR files seem to show what the HTML was before extension filtering.

On the other hand, NoScript seems to deal with noscript elements (also by replacing with span) as (or after?) the page loads, so view-source: shows the noscript elements still when only Haketilo and NoScript are installed. (NoScript seems to intentionally (in nscl/content/NoscriptElements.js with document.body.insertBefore(noscript, document.body.firstChild);) move noscript elements in the head to the top of the body, and the rest of the head continues as normal.) In some situations Haketilo tries to remove the element *before* NoScript makes it accessible, resulting in an error in the console from Haketilo and the element not being removed after NoScript shows it. I don't know if there's any way to control the order of execution of extensions that modify the page, but it would be good to say "run this script after NoScript" (so the spans are already there), or "run this script before LiberJS" (so we can see denied scripts without the user having to whitelist them (not that important)). Regardless, the first priority should probably be installations with Haketilo as the only extension, so I should test my scripts with Haketilo by itself in the future (in addition to testing with other extensions).

I would guess that uBO enables noscript elements but does not move them to the body, so the stylesheet worked fine for you but the h1 element stayed in the head and was not displayed. But, I've not installed uBO so I don't know for sure.

> I bet it's the job of either NoScript or LibreJS

It turns out the link element that was inside the noscript and is now inside a span (slated.org/modules/seckit/css/seckit.noscript_tag.css) forces the body to be shown, overriding the CSS that hides the body for browsers that do not render noscript tags. So, you are right, it is due to LibreJS (and if not for LibreJS, NoScript effectively does the same thing).

I have updated the script in my repository: https://codeberg.org/JacobK/site-fixes/src/branch/master/Slated/slated-dismiss.js

With only Haketilo installed, the script should work fine.
With Haketilo and NoScript (but not LibreJS) installed, the script should work if NoScript runs first, but it will fail if Haketilo runs first, because the element that covers the screen will be added after Haketilo tries to erase it. If you, in NoScript, allow script or disallow noscript, then the noscript should not be rendered and you should not run into any problems with NoScript.
With Haketilo and LibreJS (NoScript doesn't matter for this case), the script should work fine, because it seems LibreJS always runs before the other 2 extensions, so the elements inside the noscript should be parsed when Haketilo runs.

Also, when I update a resource, am I supposed to change the UUID? I'm not, right?

#10 - 06/30/2022 01:10 PM - koszko

- Status changed from New to Closed

- % Done changed from 0 to 100

I integrated all of your fixes. In the process I improved and rewrote some parts...

I also changed the approach I have been using so far to store the fixes. They are now in separate directories in the monorepo. I think from now on the simplest approach for donating new fixes will be to git clone it, add&commit new fixes and send me either patches or a link to the published fork.

I shall of course still accept fixes as plain JS files for those who find this way easier :)

#11 - 07/01/2022 08:02 PM - jacobk

Thanks!

I agree that cloning the bundle repository will be the simplest way to contribute now that the fixes are split into multiple directories (with their own index files).

Files

Screenshot at 2022-06-23 08-49-41.png	259 KB	06/23/2022	jacobk