Security-isolated JS widgets

Safety first, flexibility too (long version)

Wikimedia Hackathon - Jerusalem
2016-04-03
What is the state of user-supplied JavaScript code in MediaWiki today?
JavaScript from gadgets and user scripts is run directly in the web page’s security context.
**Good:** has access to everything you do!

- Can **read and write data** with your credentials easily
- Can **hook into more or less anything** in the user interface
- Lots of **helper libraries** available to you already
Bad: has access to everything you do!

- Can read private data on-wiki or modify things you didn’t expect
- Can pull in external data that risks your privacy by leaking IP
- Can pull in external code that further risks security through cross-site scripting
- Hooking directly into user interface means code is exposed to unexpected dependency breakage
What do we do about that?
Site JS and Gadgets live in restricted-access “MediaWiki:” namespace

Only a few thousand people can introduce code, instead of billions
User JS only runs for yourself, and using other people’s code requires going to unusual places and cut-n-pasting obscure commands

Ease-of-use barrier prevents most people from running insecure code! ;)
Security isolation through `<iframe>` separation
Example iframe-based widget plugin

- Runs on Wikidata via user-script shim
- The payload has its own HTML and JS libraries
- All that is isolated in an iframe, currently loaded from another site
  - Frame code has no way to access DOM or JS objects from the wiki!
Interactivity through limited API

Example exposes a very limited API to the widget:

- Host wiki sends the URL and title of the page down
- Iframe code can request navigation to another wiki page

Clicking on a related node in the graph navigates to that node, which draws a new graph.
Next steps: richer APIs
But still limited for your safety!
Would be great to allow widgets to plug in to the user interface via clearly defined stable APIs.

These can be provided consistently on both desktop and mobile interfaces, on different and custom skins, or even in native mobile apps through a special web view.

Adding custom tabs, toolbar+dialog plugins to the source and visual editors, in-browser image editors, etc should all be possible within the wiki’s UI this way.
Next steps: privacy

Solving the “web bug” problem
HTML or JS in an iframe can still cause network access through loading images, scripts, etc from third-party sites. This would expose a user’s IP address, risking privacy exposure equivalent to opening an external link.

Thus the iframe isolation alone isn’t safe enough to allow completely unreviewed code to autoload in the content area.
Sane ideas:

**opt-in click-to-play** for unreviewed content widgets, with ability to review & version-lock to allow autoplay in content.

A nice review queue and **tools to help review**. Yeah, that’d be cool.

Crazy ideas:

**JavaScript recompilation and restricted DOM proxies** such as using the [caja framework](https://caja.github.io/).

**Scrapping JavaScript and running lua with custom non-DOM user interface** is probably too crazy, and kills ability to reuse open-source JS libraries.
Next steps: merging disparate projects?

Gadgets, user JS, Widgets, WikiWidgets, oh my!
Modernize old EmbedScript extension’s transport

Older experiment that injects code hosted on the wiki into the iframe.

Modern iframe sandbox options can remove need to use an external site to host the frame.

Integrate old loader shim with the newer message posting. Port the examples!
Review popular Gadgets & user scripts

See how much work it would take to create APIs rich enough for them to be workable!

Some will be a poor fit, but many should be doable.

<table>
<thead>
<tr>
<th>User profile</th>
<th>Appearance</th>
<th>Editing</th>
<th>Recent changes</th>
<th>Watchlist</th>
<th>Notifications</th>
<th>Gadgets</th>
<th>Beta features</th>
</tr>
</thead>
</table>

Below is a list of custom features ("gadgets") you may enable for your account. Most of them require JavaScript to be enabled in your browser. Please note that these tools are not part of the MediaWiki software, and are usually developed and maintained by users on Wikipedia. The numbers of users for each gadget on this wiki are listed at [Gadget usage statistics](#).

Be advised that you take full responsibility for any action performed using these features. For more information, see our policies and guidelines.

**Browsing**

- After rolling back a user’s edit, automatically open their contributions page
- Require confirmation before performing rollback on mobile devices (documentation)
- Disable access keys
- Focus the cursor in the search bar on loading the Main Page
- GoogleTrans: open a translation popup for the selected text or the word under the cursor when pushing the shift button
- ImageAnnotator: view image notes and comments on file description pages
- Redirect image links to Commons for files hosted there
- Navigation popups: article previews and editing functions pop up when hovering over links
- Open external links in a new tab or window
- Open search results in a new tab or window when holding down the Ctrl key
- Print options: control how pages are printed (for example, remove images or backgrounds)
- revisionjumper: quickly navigate between page revisions
- Twinkle: automate common tasks such as reporting vandalism, warning vandals, requesting deletion, welcoming users, and tagging articles (preferences)
- Suppress display of fundraiser banners
- Suppress display of CentralNotices
- Enable the Teahouse "Ask a question" feature
- Reference Tooltips: hover over inline citations to see reference information without moving away from the article text
- FormWizard: a wizard for creating and expanding project pages
A few ‘WikiWidgets’ are in use on es.wikipedia.org

Can be ported to use iframe isolation for safety; transport parameters from the placeholder div via a limited API.
Widgets extension could benefit from iframes

Content widgets should usually not depend much on the parent page. Could be ported to use iframe isolation for additional safety; transport parameters from the placeholder div via a limited API.
the end

https://phabricator.wikimedia.org/T131436