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

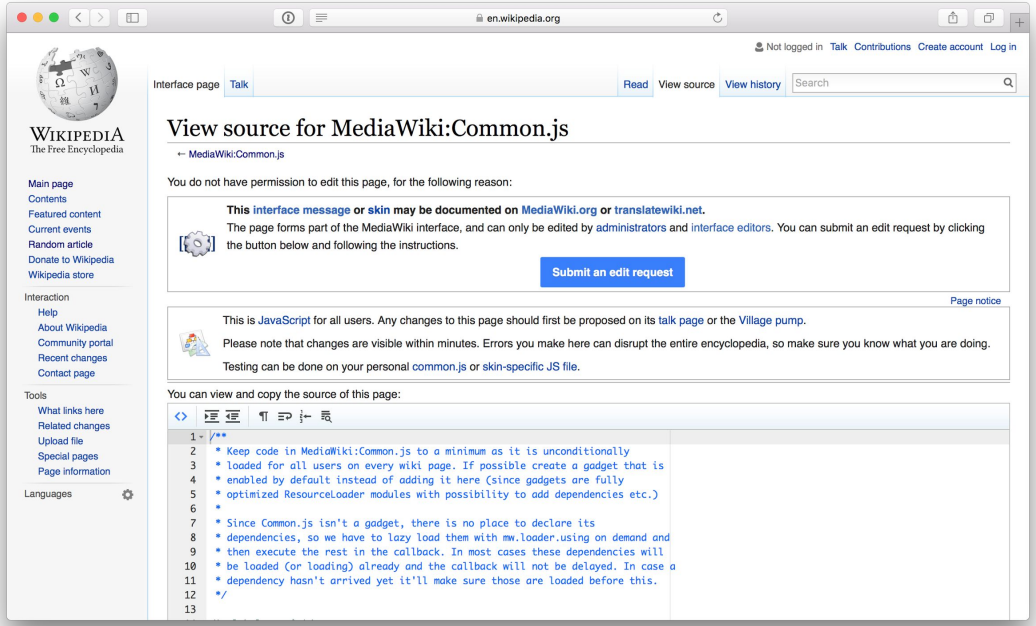
Bad: has access to everything you do!

- 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?

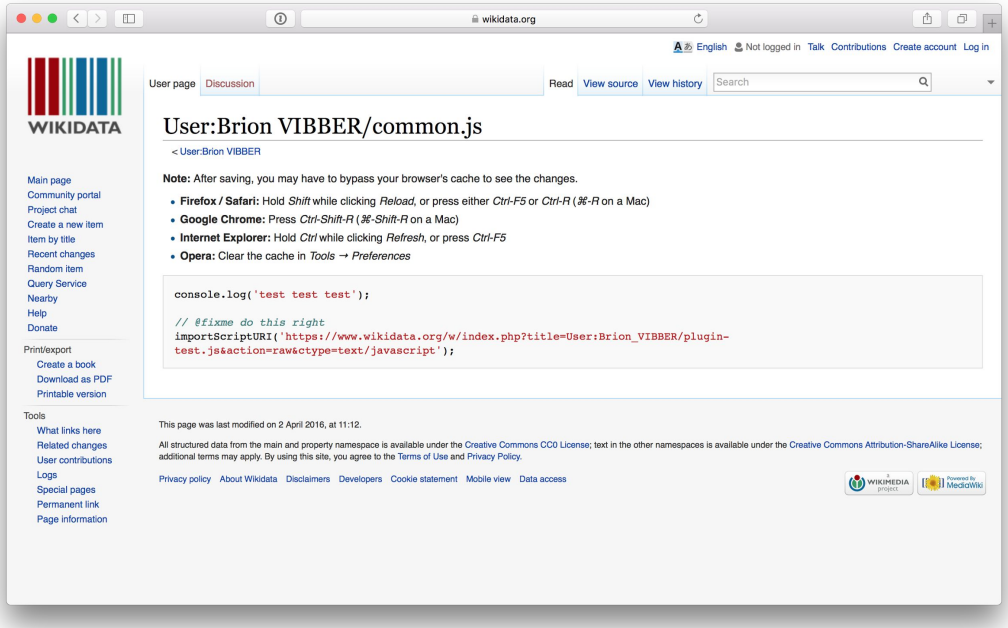
Site JS and Gadgets live in restricted-access “MediaWiki:” namespace

Only a few thousand people can introduce code, instead of billions

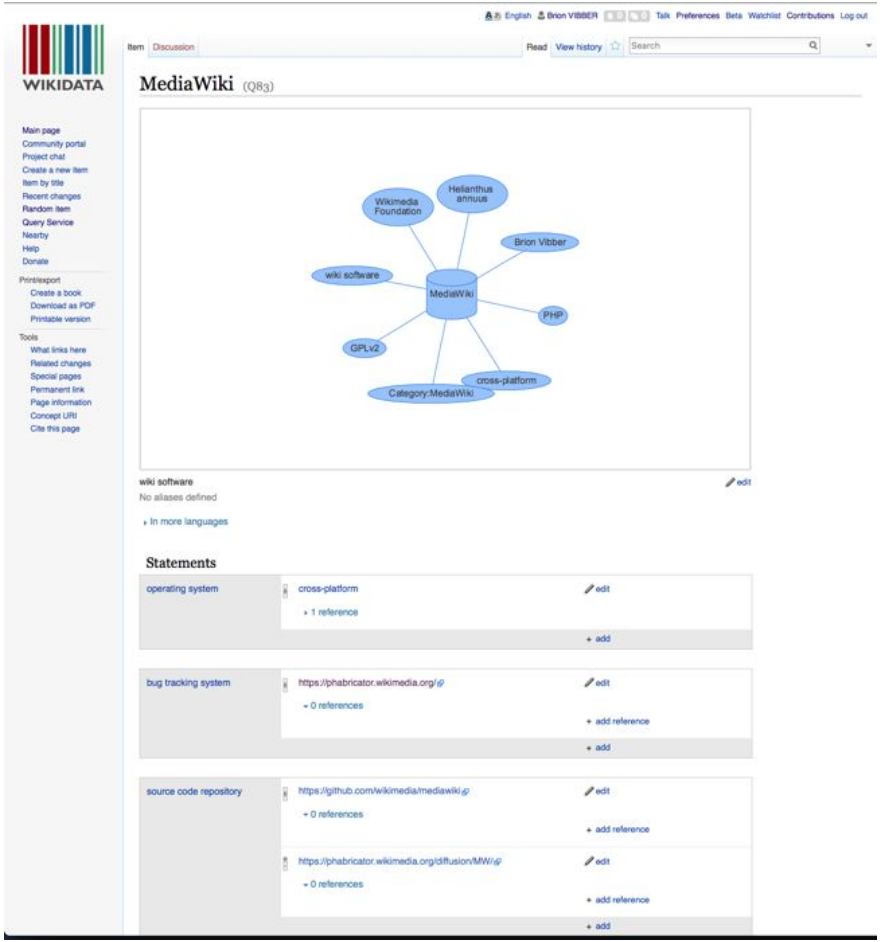


User JS only runs for yourself, and using other people's code requires going to unusual places and cut-n-pasting obscure commands

Ease-of-use barrier prevents most people from running insecure code! ;)

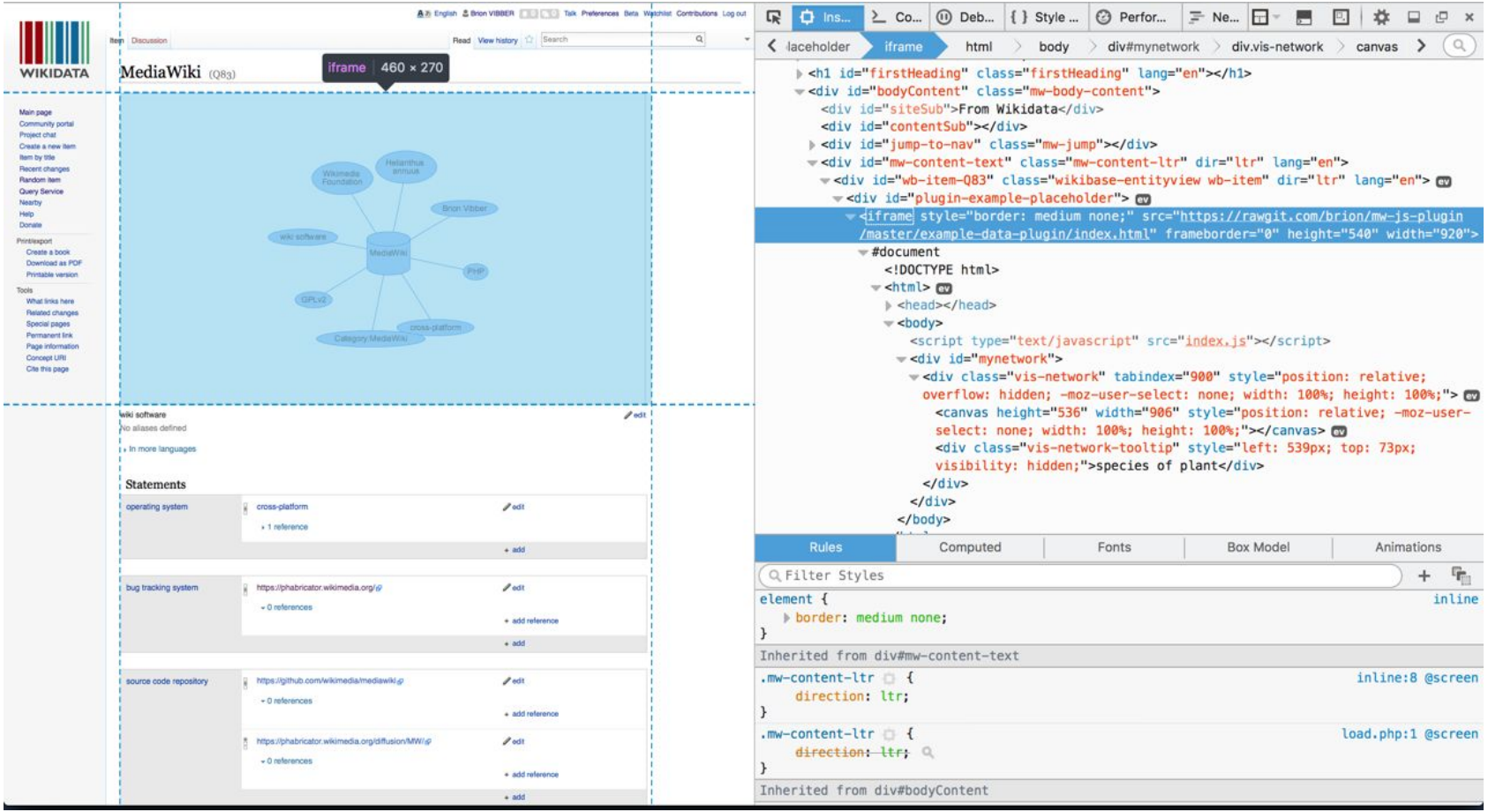


Security isolation through <iframe> separation



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!

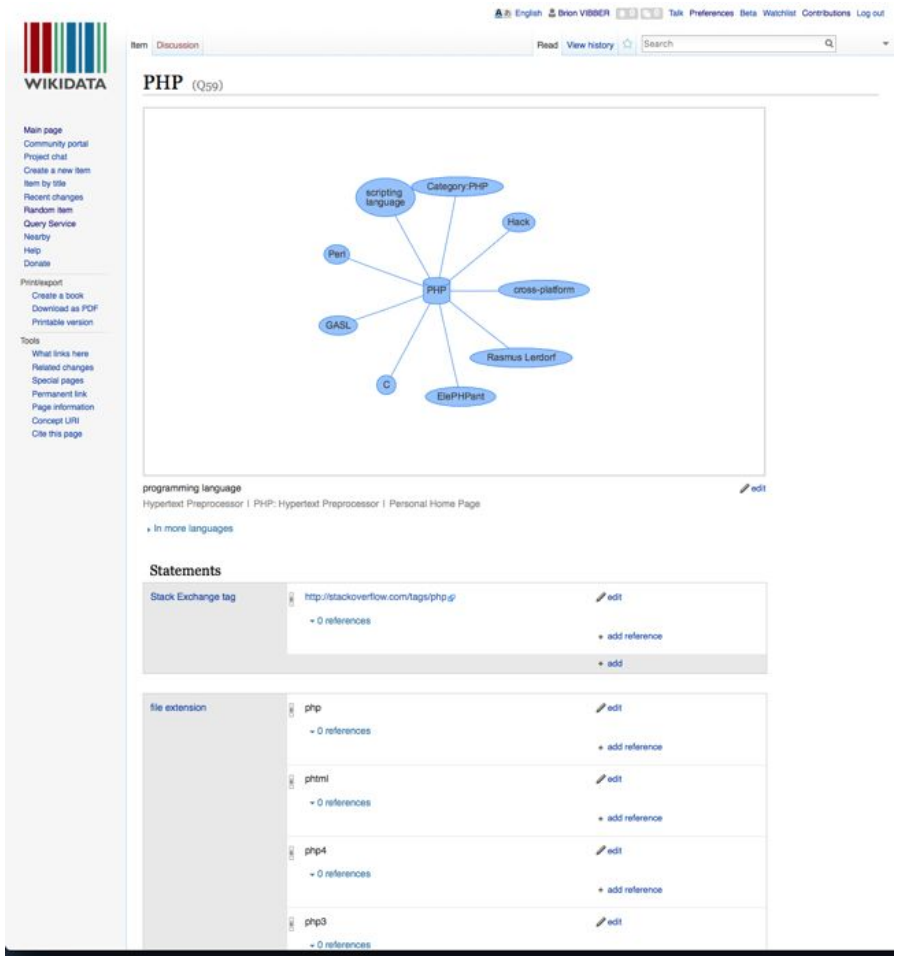

WIKIDATA
MediaWiki (Q83)
iframe 460 × 270
Main page
Community portal
Project chat
Create a new item
Item by title
Recent changes
Random item
Query Service
Nearby
Help
Donate
Print/export
Create a book
Download as PDF
Printable version
Tools
What links here
Related changes
Special pages
Permanent link
Page information
Concept URI
Cite this page
Wikimedia Foundation
Helianthus annuus
Brion Vibber
wiki software
MediaWiki
PHP
GPLv2
cross-platform
Category:MediaWiki
wiki software
No aliases defined
In more languages
Statements
operating system
cross-platform
1 reference
bug tracking system
https://phabricator.wikimedia.org/
0 references
source code repository
https://github.com/wikimedia/mediawiki
https://phabricator.wikimedia.org/diffusion/MW/
add reference
add
Inspector
Console
Debugger
Style Editor
Performance
Network
placeholder
iframe
html
body
div#mynetwork
div.vis-network
canvas
<h1 id="firstHeading" class="firstHeading" lang="en"></h1>

From Wikidata

<iframe style="border: medium none;" src="https://rawgit.com/brion/mw-js-plugin/master/example-data-plugin/index.html" frameborder="0" height="540" width="920">
#document
<!DOCTYPE html>
<html>
<head></head>
<body>
<script type="text/javascript" src="index.js"></script>

<canvas height="536" width="906" style="position: relative; -moz-user-select: none; width: 100%; height: 100%;"></canvas>
species of plant

</body>
Rules
Computed
Fonts
Box Model
Animations
Filter Styles
element { inline
border: medium none;
}
Inherited from div#mw-content-text
.mw-content-ltr { inline:8 @screen
direction: ltr;
}
.mw-content-ltr { load.php:1 @screen
direction: ltr;
}
Inherited from div#bodyContent



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.

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?

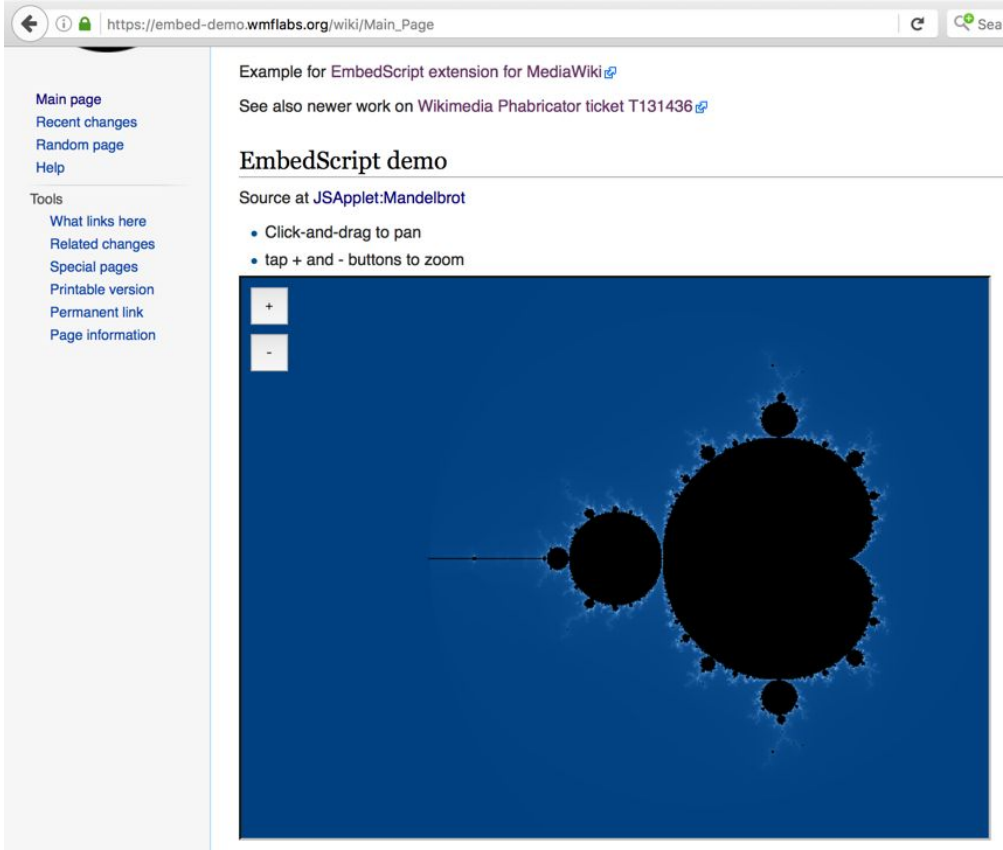
Gadgets, user JS, Widgets, WikiWidgets, oh my!

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.

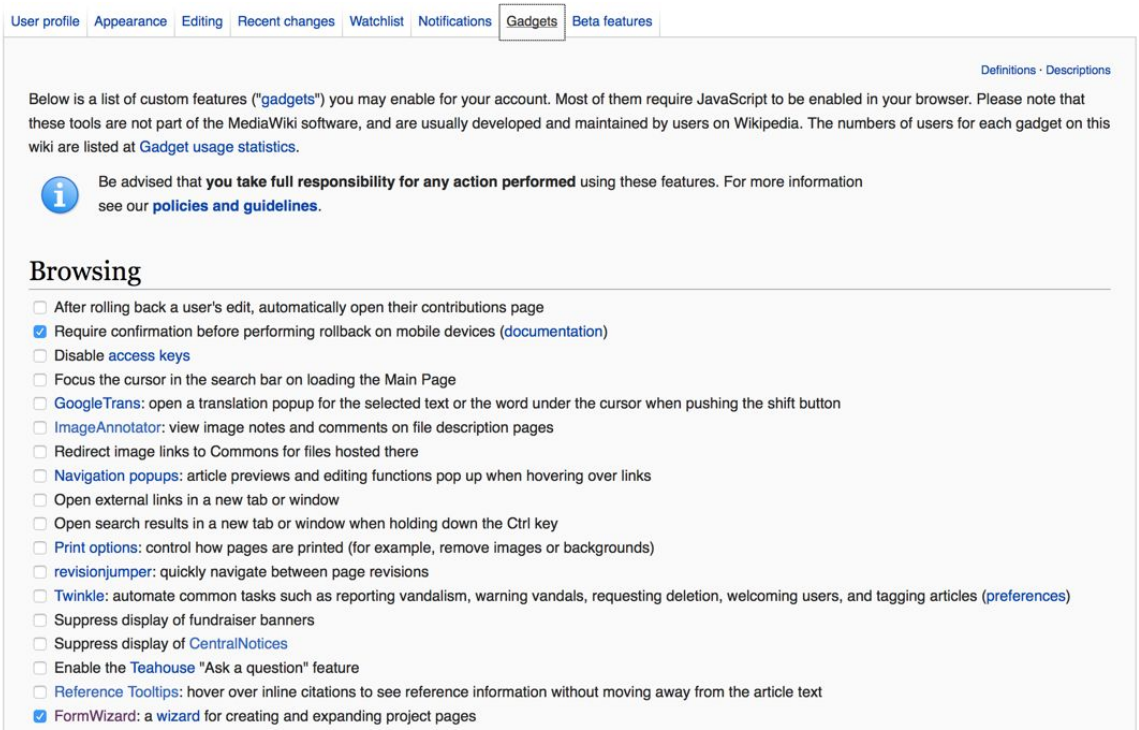
Integrate old loader shim with the newer message posting. Port the examples!



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.



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