Design for Power Users

Redesigning NoScript: A Case Study

September 2019

Simply Secure

Supported by OPEN TECHNOLOGY FUND

Contents

About this project

NoScript is a browser extension for Firefox and Chrome/Chromium that blocks executable web content based on JavaScript, Java, Flash, Silverlight and other plugins, unless the site host is considered trusted by its user and has been previously added to a whitelist. It is considered a trusted open source tool, and regarded as an integral part of a larger suite of browser extensions for additional security. Along with HTTPS Everywhere, NoScript is embedded in Tor Browser by default.

NoScript has 1.5 million users on Firefox and around 50k users on Chrome, in addition to all Tor Browser users (currently around 2.5 million). It is used by activists, governments and corporations alike, and was the first (and is, as of August 2019, the *only*) browser extension to offer client-side protection against cross-site scripting attacks.

NoScript 10 "Quantum", a full rewrite of NoScript 5 "Classic" using the new WebExtensions API, was released at the end of 2018. This was necessary for NoScript to survive past Firefox 57, which deprecated and disabled all legacy add-ons. There was a significant backlash from the existing user base due to the unfamiliar UI.

High-priority usability issues

Although many improvements have been made since the initial release of the new version, many usability issues still required attention. Specifically, we identified three problem areas at the start of the project:

1. Given the technical impossibility to reproduce verbatim the old UI, NoScript took the chance of a complete overhaul including long requested features (such as customizable presets) which couldn't cleanly fit in the previous menu-based model. This, however, resulted in an excessive density of the layout, an excessive use of icons with no labels and therefore a generalized discoverability problem.

2. The switch from XUL to HTML led to accessibility issues, especially regarding keyboard navigation. There is the additional option to display NoScript in "high-contrast mode", but that led to an extremely complex interface.

3. The documentation is outdated, still mostly referring to NoScript 5 "Classic". Community-provided tutorials and guides are available, but have

proven not to be easy to discover. An "official" up-to-date guide is sorely needed, but even better would be the tool to be more self-explanatory.

Familiar patterns in open-source design

In addition to the problem areas, we see familiar patterns that many open source security tools face. NoScript was first released in May 2005, and has been developed and maintained by one developer as a side project. It has an active community of enthusiastic users who rely on it for security both personally and professionally, and wouldn't hesitate to contact the developer or the community for help and discussion. Changes to NoScript were mostly reactive: they were directly based on unstructured incoming user feedback or feature requests. Sometimes changes needed to be fast, as researchers file security issues that were just being discovered (and potentially exploited). *Incredibly, NoScript has always responded within 24 hours to these security fixes in the past 14 years*. However, when it came to UX/UI issues, the fix was not always immediate. This feedback setup only resulted in NoScript reacting to superficial UI/UX problems, rather than offering a coherent and thought-through foundation.

👉 Unstructured incoming feedback meant that NoScript was always one step behind user needs.

Another common issue we see is what has been referred to as the Itch-to-Scratch Model, a developer-centric mindset that prevents valuable open source tools from having wider adoption. The developer of NoScript will readily admit that he *built NoScript for himself, and will always make it work for him first*. At the same time, he and other people working in the information security sector agree that **everyone should use security tools like NoScript**. But how does one design a tool for all when the needs of the developer are the driving force behind the project?

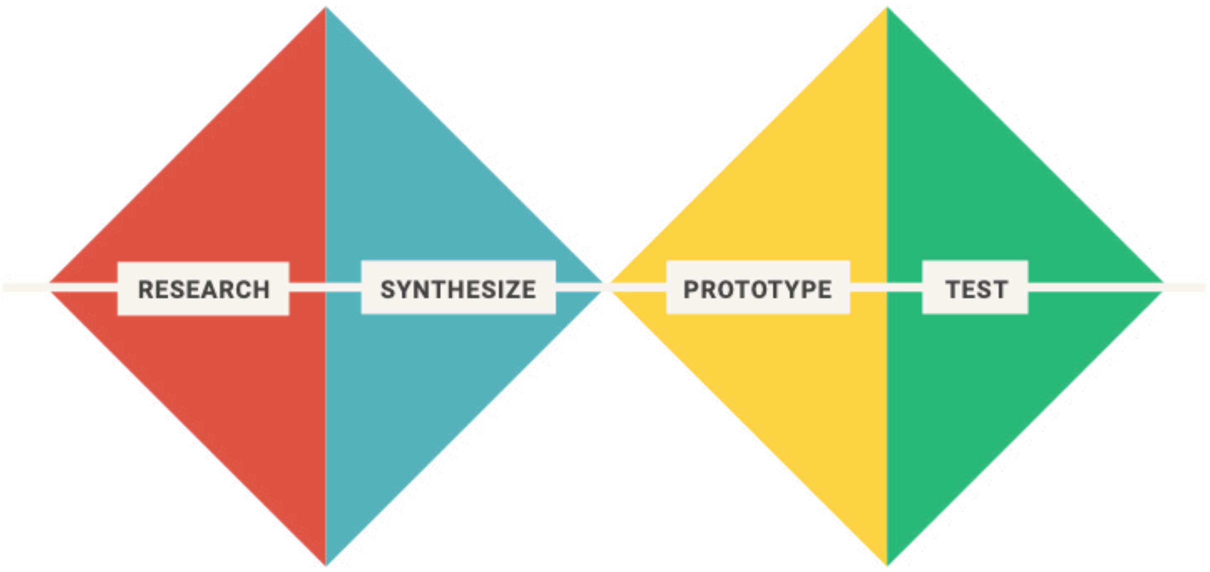
👉 NoScript is caught between *designing for developers* and *making it work for everyone*—it is difficult to balance the needs of very different user groups.

To address the problem areas as well as these underlying issues, Simply Secure was tasked to go back to the drawing board and develop a new and usable UI that fits NoScript users' needs.

Methodology

Our framework for secure usability audits looks at security from a human-centered perspective. Our driving questions are: What do people want? What do people need? What can people use?

Our research and synthesis focuses on user needs by conducting user interviews and usability tests to get get rich, qualitative feedback. In our prototyping and testing phase, our primary focus was that the tool made sense to people.



For this research, we conducted five qualitative interviews in early May 2019. The interviewees identified themselves as: novice users (2), experienced users (2), and security trainers (2). The interview guide is in our appendix.

Scope

We have limited our redesign to the desktop browser extension as it appears on Firefox and Chrome/Chromium. The evaluation and redesign of the mobile interface for Firefox will follow the next release, and minor changes specific to mobile environments are possible.

A note about Tor Browser

We did not, as originally planned, speak to Tor Browser users about their NoScript experience. As of May 2019, Tor Browser is planning to integrate NoScript settings into its general security settings interface (the *slider*), and thus we did not think that evaluating current pain points in the Tor Browser would be helpful. However, we did sketch out some UI designs for Tor use cases that are in line with the redesign. Thanks are due to the Tor Browser team for providing us with feedback they had collected from Tor Browser users, and for many great discussions on design issues and directions in the secure browser extensions space.

Findings

Things that work

NoScript functions as a browser extension for security

The users we spoke to understood that NoScript offers state-of-the-art script blocking, and is probably the most secure choice out there, even if not the most usable one. Users also understood how NoScript might interact with other browser extensions, and generally like "fiddling around with the knobs" to see how things interacted.

> *NoScript is the last line of defence on my browser. I trust it to be working in my interest at all times.*

> *I think everyone needs something like that. But not everyone will understand this tool.*

NoScript also provides information

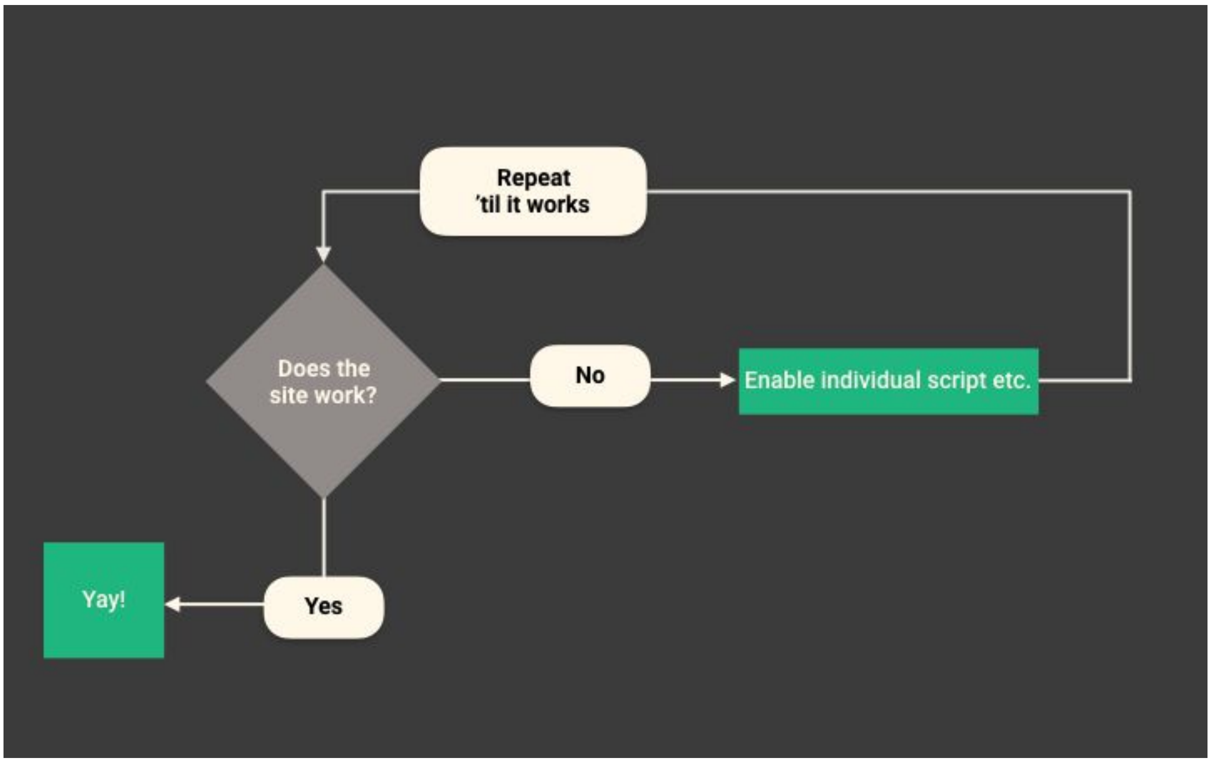
For many users, seeing the scripts themselves is already a feature. If a site uses a lot of scripts, especially scripts that look like they are trackers, users lower their trust to a site.

> *You can see how much cruft is implemented on the website– just seeing this with the slider pushed over gives me a good*

> *feeling—and I get a bad feeling if the slider is on the wrong side!*

People know how to use NoScript effectively

Experienced users had an established workflow of visiting a site → evaluating breakage → temporarily allowing scripts from the parent site → reloading → scrutinising other scripts and toggling their controls for the site to display/function correctly. **This is not a straightforward process**—yet experienced users are able to achieve their respective goals.



Experienced users understood the NoScript workflow.

Users often manage to achieve their goals, though the steps they take may not be the most efficient. Some people always change settings in the `per-site permissions` tab, while others prefer the menu interface with its `allow all temporarily` option.

While novice users understood that NoScript blocks scripts on the website you visit, few of them were able to verbalise NoScript's blocking behavior (that blocking one domain on one site means blocking this domain for all sites, unless settings are temporarily revoked). Relatedly, novice users could only guess that temporary changes are changes **in this current browser session**.

To novice users, the key questions were:

- Which domains are ok? How do I know whether something is a tracker?
- Does NoScript help me decide which scripts are OK? Are there defaults?
- Does NoScript learn from my behavior? From other users' behavior?

Things that don't work

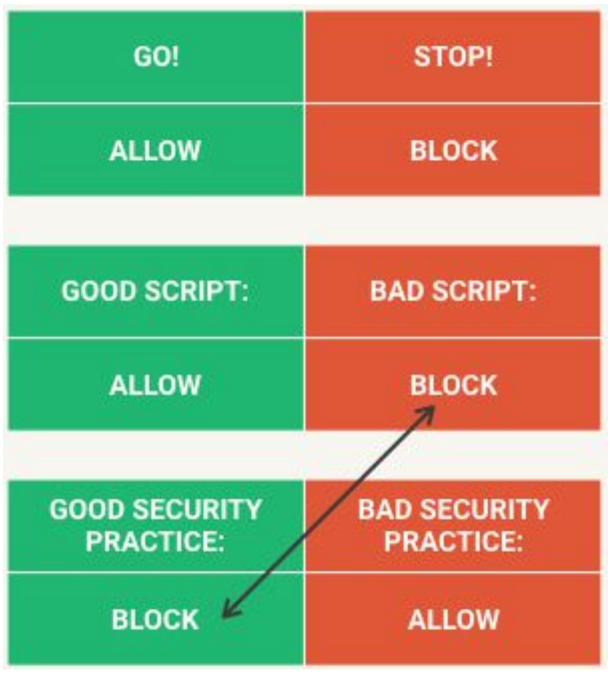
System states and icons do not match

NoScript uses three main icons to indicate differences in states:

- ❗ means that restrictions are disabled (i.e. scripts are allowed),
- ❌ means restrictions are enabled (i.e. scripts are blocked),
- 🕒 means temporary, with the options `temporarily allow` and `revoke temporary allowance`.

These icons were confusing to all users we talked to, most likely because system state (`allow` and `block`) were mixed with recommendations (`secure` and `danger`). This is a phenomenon that happens throughout the interface, and the most obvious place for this was the use of *color*.

If one were to assign a color for UI elements related to blocking, what would it be? Green is commonly associated with "GOOD!", "DO THAT!", "GO!", while red is associated with "BAD!", "DON’T DO THAT!", "STOP!". But adding additional meaning to color complicates things.



When we encounter a good or harmless script, we should allow it. If it is a bad or harmful script, we should block it. However, when we attach the GOOD and BAD to security practices, the colors are flipped: a good or recommended security practice is to block things by default, and a bad or not recommended security practice is to allow scripts in general. Suddenly, a simple color can carry meaning on three levels: the action, the system state, or the recommendation.

NoScript does not use red and green in combination (at least not with regard to `block` and `allow`), but the same problem stills exists in a more nuanced form. An icon with a crossed-out item is usually BAD, and a sign without anything is usually GOOD. But of course, here blocking something is actually GOOD, while blocking nothing is actually BAD. So whichever indicators NoScript chooses, they should **either** aim to indicate system state [allow/block] **or** recommendation [good/bad], but **not both**. And in any case, NoScript should probably stay away from standard colors and icons.

When to block, when to allow

In general, people tripped over allowing and blocking across the interface. For example, the `CUSTOM` preset has checkboxes for **allowing** certain scripts on sites; but users have unilaterally interpreted the checkboxes to be for **blocking** scripts of different kinds. The word "allow" at the top left was not always seen. (Moreover, having the `script` option as a checkbox in that tab was confusing —people rightly asked, "isn't everything a script?") Similarly, users stumbled over the phrase "disable restrictions" as a *double negative* for pausing NoScript on a particular tab. We've seen multiple NoScript users guessing at the meaning of these buttons, only to then test the difference in settings themselves and try to infer the behavior of particular features.

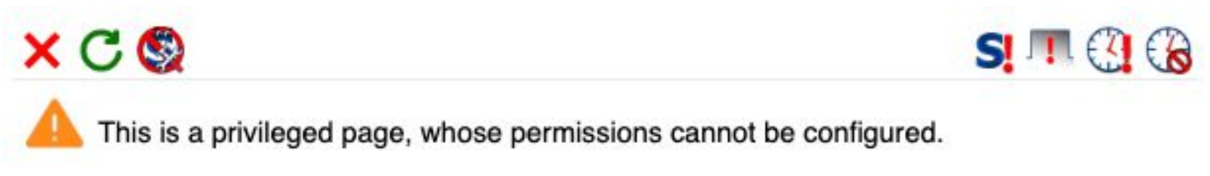
Another source of confusion comes from the usage of "trust". Experienced users know that sites you trust should generally be set to `allowed`, but sometimes it is necessary to allow scripts from sites you don't trust for things to function correctly. Equating the two concepts (blocking and distrusting) was not helpful. Novice users sometimes expect NoScript to be making decisions about trust for them, due to the passive phrasing ("trusted" and "untrusted").

Finally, the green or red lock icon indicating HTTPS status was mistaken as an indicator for a NoScript-trusted site for novice users. Neither novice users nor experienced users knew what it meant.

Unclear language makes actions confusing

We found that some of the labels were unclear to both novice and experienced users. For example, none of the 6 people we talked to could describe what `Cascade top-level documents` or `Match HTTPS content` meant.

After installing NoScript, people often open it while on a tab that is "privileged" (browser settings, for example the `add-ons` page), which does not allow content blocking. The screen for this warning was a first hurdle for novice users, who didn't understand the warning, nor where to go from here.

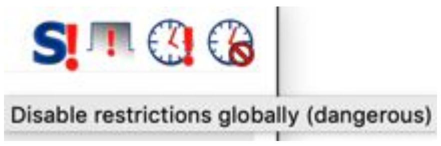

This is a privileged page, whose permissions cannot be configured.

This is often the first screen that new NoScript users will see.

Strict tone and voice, along with dated UI, made people feel insecure

NoScript's language on security is divisive. On the one hand, we repeatedly heard that running NoScript is making its users feel **safer** online. On the other hand, some users reported that they feel guilty when turning it off; or worse, they believed they were completely unprotected and insecure when it's off.

This was most evident when novice users confronted the interface. They were baffled by the fact that not using NoScript was deemed "dangerous".



> *5 minutes ago I was browsing without it, putting 'dangerous' here sounds life-threatening...*

The visual design stood out to many novice users. Upon seeing the settings page, novice users commented that the design looks dated, and does not inspire trust.

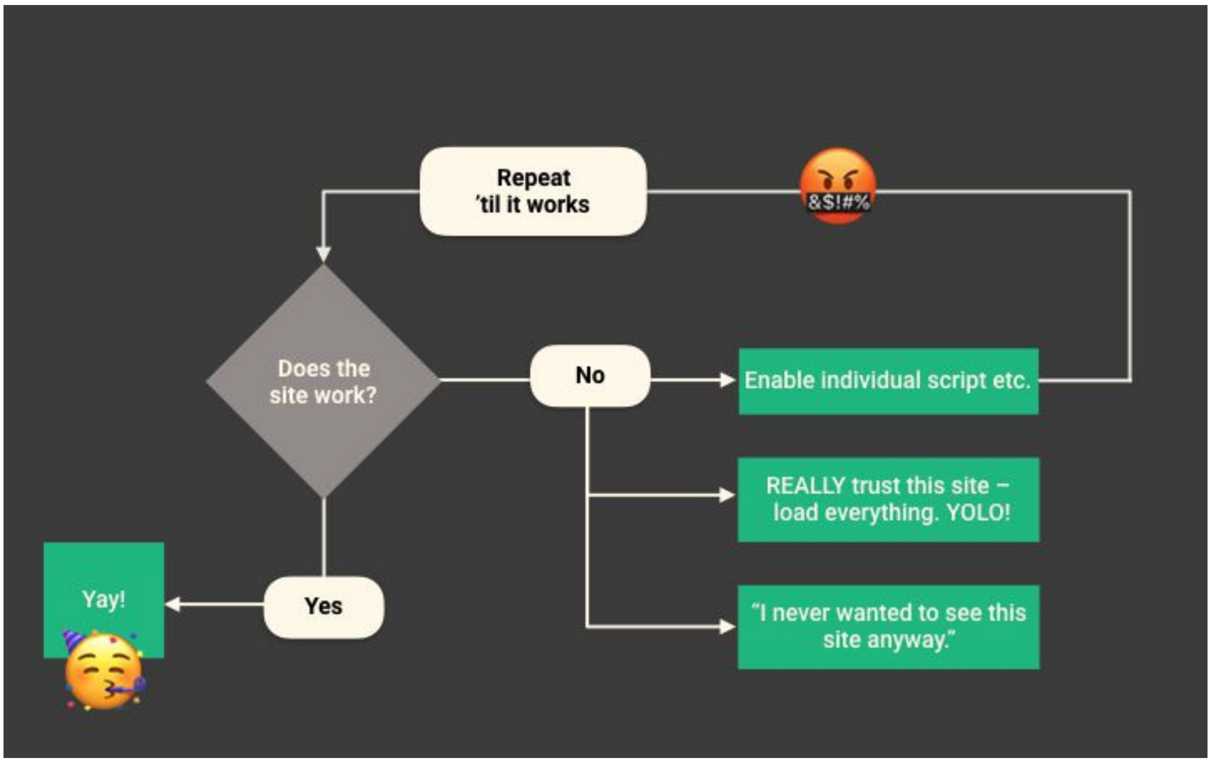
> *I cannot judge whether the tech is good or not, I have to rely on the design to tell me that.*

Users repeatedly commented on the icon design (the in-browser NoScript badge). It is too small for many devices, and not always as informative as they had hoped.

Most people found the logo cute, albeit old-fashioned. Some knew the reference to Ghostbusters. However, the logo is not consistent (store vs app vs settings), and that has thrown some people off.

Page breakage means people abandon NoScript too soon

Even though experienced users understood the ideal workflow of NoScript, few people follow it consistently throughout their web browsing. Page breakage happens frequently even with fine-tuned NoScript settings: either the display of certain elements is removed or altered, or interactive elements don't function correctly. Most people abandon the process of assessing page breakage and adjusting script blocking after one or two attempts, and either allow everything temporarily, or close the tab they wanted to view.


Repeat
'til it works
&$!#%
Does the site work?
No
Enable individual script etc.
REALLY trust this site – load everything. YOLO!
"I never wanted to see this site anyway."
Yes
Yay!

> *If I haven't found the right thing to turn off after the second try, I'll just turn NoScript off.*

Sadly, whenever pages work, the success is attributed to the website that uses javascript wisely; whenever pages break, the failure is attributed to NoScript for breaking the website. By design, NoScript is always associated with frustration.

Recommendations

Based on our user research, we developed five main design recommendations for NoScript.

1. Focus on visualisation and customisability

NoScript users value the ability to **see** which scripts are being used on a site, and to **adjust** settings to only allow the scripts needed for a site to function correctly. Most active users only use the menu interface on the website they are visiting, improving that interface and the visualisation of script information will be most crucial.

As such, NoScript is a tool that *creates friction* rather than removing it, and this sort of friction is seen as valuable and necessary for people who care about security.

> *I always pause before I [allow] a script—what is it, and can I trust it? NoScript makes me think about the websites I visit.*

Design for:

👩‍💻 Rose the Super User

Potentially design for:

👨‍🏫 Emil the Privacy Advocate

Don't design for:

👩‍🎤 Chris the Curious

We recommend sticking with the core offering (surfacing information on scripts, and allowing for individual blocking) and making that more intuitive, rather than offering a script blocker that works silently and frictionlessly. **NoScript is inherently interactive, and not for everyone.**

2. Communicate system state better

Current system states can be opaque or ambiguous, and don't always match the user's mental model. NoScript is a complex tool, and the goal isn't to simplify the options; however, a clear communication around system states and tool behavior that doesn't leave the user guessing is vital for people to achieve their goals.

We recommend using clear language and indicators for `blocking` and `allowing`, instead of trusting and untrusting. New and readable icons could help to indicate system states. We also recommend going through all labels and making sure functions are described in clear, simple and consistent language.

Sample text explaining NoScript

When you visit a `site`, it can have a `script`, from a `script owner`.

The script could present a security risk, or a privacy infringement, or just make browsing unnecessarily slow and cluttered.

But you can decide on what gets loaded into your browser! You can block scripts.

Deciding on what to block, and how to block it, requires some knowledge about scripts and their owners.

Though there are some general rules of thumb, decisions always depend on the site, the type of script, and the owner of that script.

NoScript is a tool that allows you to make these decisions easily.

You can `block` or `allow` a script based on the `site` you're visiting, the `type of script`, and the `owner of that script`.

When you do any of the three things above, you can do it temporarily (on a tab), or permanently (across all visits).

Moreover, you can also save your specific `settings for the sites you visit`, so you don't have to fine-tune your blocking when you visit your most frequent sites.

NoScript blocks every script by default. When you change this behavior for an individual script, or for scripts on a specific site, the new rules get saved. You can view and edit these rules in the settings.

3. Supply better defaults and explain why

While we want to focus on making toggling choices and seeing site/script information easier, it's vital to also consider the default settings for users who will not fine-tune NoScript to their needs. It is important to have on-screen helper texts (for example, hover-overs or information boxes) explaining features and defaults.

> *It would be great if NoScript would take a stand and prescribe settings for people who just wanted to be safer and not mess with it every single minute.*

People also wanted to know why certain defaults were chosen. While the documentation is not the appropriate place to understand how and why things work, it is a good place to look up background information—experienced users said the documentation "made them feel safer".

We recommend default settings that work out-of-the-box, and a walk-through of the main NoScript features in the onboarding process. Background information around browsing security as well as default choices should be provided in the documentation.

4. Update the visual and interface design

The choice of colours, typeface, layout and UI elements could use an upgrade —for a state-of-the-art tool, NoScript *looks* like it was last updated a decade ago.

This may seem like a cosmetic problem that doesn't impact the way the tool works for users. However, users mentioned that an outdated interface:

- Reduces their trust in the tool, and makes them wonder if it has bugs or malware
- Makes them worry that it will not be compatible with their browser
- Leaves them confused as to whether they have the latest version or need to update

Beyond the general update, there are some specific changes we think could address some issues with the interface.

> *I could shoot myself in the foot accidentally with one of these toggle settings and it wouldn't warn me.*

- `Enable/restrict temporarily` should be a toggle with appropriate labels. Toggles should be used with care, making options clear and also throwing good alerts when needed.
- The presets under `Options/General` should be communicated clearer, for example by renaming the tab to "Configure presets" and pushing it further back. (Make sure it's understood as *redefining* `block` `allow` etc.)
- When you change settings that affect the current site, the menu interface should reflect that.

We recommend a complete re-design of the UI, using a simple and clear style that does not distract from the existing complexity.

5. Add a "per-site settings" feature

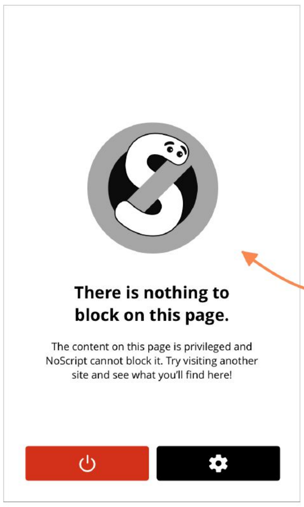
As of now, changing the preferences around a script/domain means changing the preference for that script/domain for all tabs, for all sites visited. But that does not cover site-specific settings. For example, if a user wanted to allow scripts from `googleanalytics.com` when visiting a trusted site, but does not want to allow them when visiting an untrusted site, this change has to be made manually each time. It would be convenient for the user to set these per-site preferences manually and have them saved as well.

A related use case is when people want to allow `first-party` scripts but not `third-party` scripts, e.g. when visiting Facebook.com you want to allow scripts from Facebook.com; but when visiting your local news site you don't want to allow scripts from Facebook.com. This could fall under saved per-site settings, or could be enabled with a shortcut, for example `first-party` or `affiliated scripts`.

We recommend a new interface design for allowing per-site settings. This interface should communicate clearly which domains are allowed or blocked A) permanently and globally, B) permanently for this site, C) temporarily for this tab.

Design decisions & testing

Main Screens


There is nothing to block on this page.

The content on this page is privileged and NoScript cannot block it. Try visiting another site and see what you'll find here!

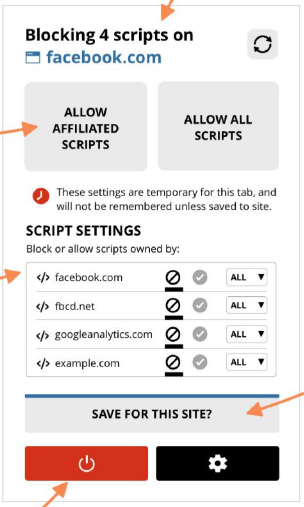
This is often the first screen new users see: when visiting the adds-on page or the browser settings page, NoScript does not block anything. The gray symbol and helper text explain that and direct to next steps.

First important information is on how many scripts are being blocked where.

These are "quick and dirty" options. Using text instead of icons means people don't need to guess what this does.

Below that, we have all the options to fine-tune and trial and error.

A power button to disable the entire browser extension following existing icon designs in HTTPS Everywhere.


Blocking 4 scripts on
facebook.com

ALLOW AFFILIATED SCRIPTS | ALLOW ALL SCRIPTS

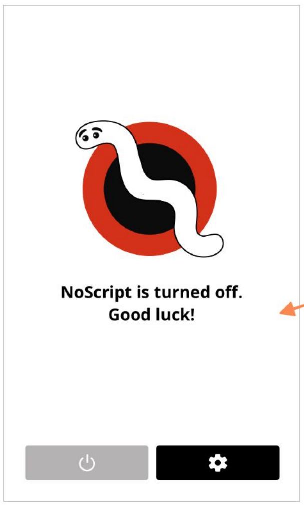
These settings are temporary for this tab, and will not be remembered unless saved to site.

SCRIPT SETTINGS
Block or allow scripts owned by:

facebook.com	ALL
fbcd.net	ALL
googleanalytics.com	ALL
example.com	ALL

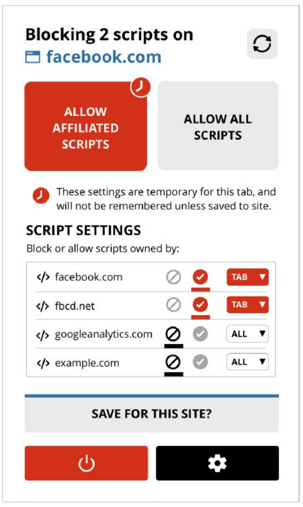
SAVE FOR THIS SITE?

New feature: saving per-site settings.


NoScript is turned off.
Good luck!

When NoScript is turned off, the script snake escapes. "Good luck" here is intended to sound alarming, but not patronizing.

Scope of blocking and allowing


Blocking 2 scripts on
facebook.com

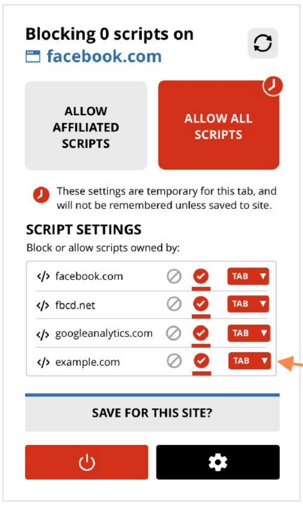
ALLOW AFFILIATED SCRIPTS
ALLOW ALL SCRIPTS

These settings are temporary for this tab, and will not be remembered unless saved to site.

SCRIPT SETTINGS
Block or allow scripts owned by:

facebook.com TAB ▾
fbcd.net TAB ▾
googleanalytics.com ALL ▾
example.com ALL ▾

SAVE FOR THIS SITE?


Blocking 0 scripts on
facebook.com

ALLOW AFFILIATED SCRIPTS
ALLOW ALL SCRIPTS

These settings are temporary for this tab, and will not be remembered unless saved to site.

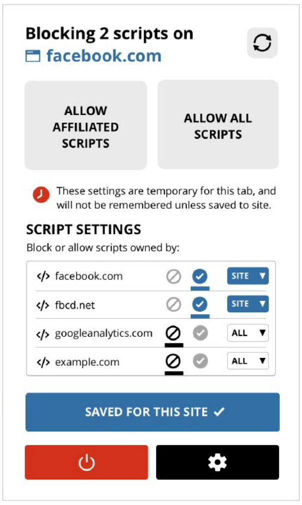
SCRIPT SETTINGS
Block or allow scripts owned by:

facebook.com TAB ▾
fbcd.net TAB ▾
googleanalytics.com TAB ▾
example.com TAB ▾

SAVE FOR THIS SITE?

The quick options apply to the tab and subsequent tabs on the same site (in this case facebook.com) opened from this tab.

Changes in quick options are always directly reflected in the detailed options.


Blocking 2 scripts on
facebook.com

ALLOW AFFILIATED SCRIPTS
ALLOW ALL SCRIPTS

These settings are temporary for this tab, and will not be remembered unless saved to site.

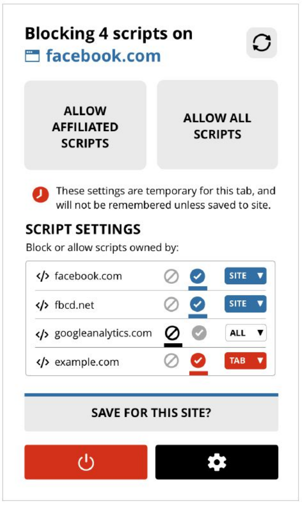
SCRIPT SETTINGS
Block or allow scripts owned by:

facebook.com SITE ▾
fbcd.net SITE ▾
googleanalytics.com ALL ▾
example.com ALL ▾

SAVED FOR THIS SITE ✓

Saving the current site settings changes the color of the blocking and allowing icons. For accessibility, we added an underline as well as text on the right side to indicate the scope.

A single site can have script blocking across different scopes:


Blocking 4 scripts on
facebook.com

ALLOW AFFILIATED SCRIPTS
ALLOW ALL SCRIPTS

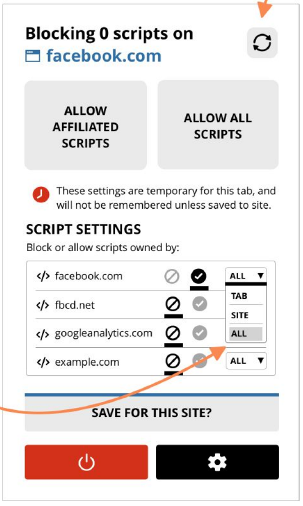
These settings are temporary for this tab, and will not be remembered unless saved to site.

SCRIPT SETTINGS
Block or allow scripts owned by:

facebook.com SITE ▾
fbcd.net SITE ▾
googleanalytics.com ALL ▾
example.com TAB ▾

SAVE FOR THIS SITE?

There are two ways to change individual scopes: by dropdown menu on the right, and by multiclicking on the icons. Changes go into effect on reload.


Blocking 0 scripts on
facebook.com

ALLOW AFFILIATED SCRIPTS
ALLOW ALL SCRIPTS

These settings are temporary for this tab, and will not be remembered unless saved to site.

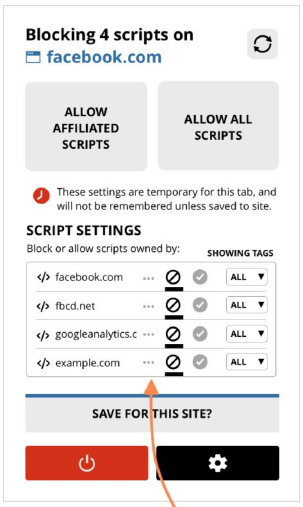
SCRIPT SETTINGS
Block or allow scripts owned by:

facebook.com ALL ▾
fbcd.net TAB
googleanalytics.com SITE
example.com ALL ▾

ALL

SAVE FOR THIS SITE?

Advanced interfaces



Blocking 4 scripts on
facebook.com

ALLOW AFFILIATED SCRIPTS | ALLOW ALL SCRIPTS

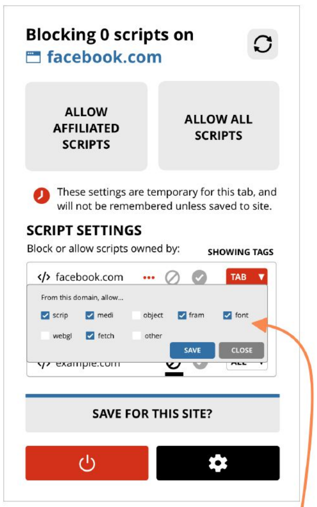
These settings are temporary for this tab, and will not be remembered unless saved to site.

SCRIPT SETTINGS
Block or allow scripts owned by: SHOWING TAGS

facebook.com •••	ALL
fbcd.net •••	ALL
googleanalytics.c •••	ALL
example.com •••	ALL

SAVE FOR THIS SITE?

For more fine-grained blocking, there is the advanced option of showing “tags” in the script setting.



Blocking 0 scripts on
facebook.com

ALLOW AFFILIATED SCRIPTS | ALLOW ALL SCRIPTS

These settings are temporary for this tab, and will not be remembered unless saved to site.

SCRIPT SETTINGS
Block or allow scripts owned by: SHOWING TAGS

facebook.com ••• TAB

From this domain, allow...

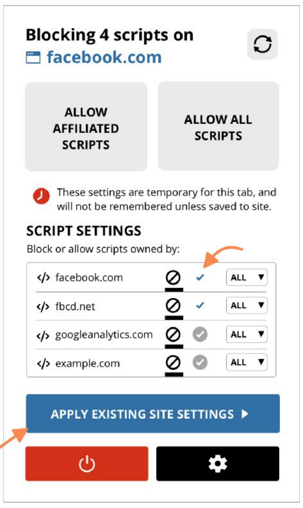
[✓ scrip] [✓ medi] [ ] object [✓ fram] [✓ font]
[ ] webgl [✓ fetch] [ ] other

SAVE CLOSE

example.com

SAVE FOR THIS SITE?

Clicking on it will open an additional interface that blocks scripts by their tag. These settings can be saved with the usual scopes (TAB, SITE, ALL).



Blocking 4 scripts on
facebook.com

ALLOW AFFILIATED SCRIPTS | ALLOW ALL SCRIPTS

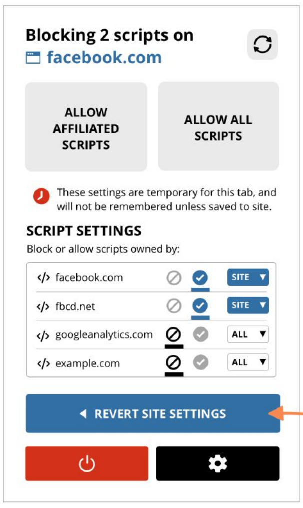
These settings are temporary for this tab, and will not be remembered unless saved to site.

SCRIPT SETTINGS
Block or allow scripts owned by:

facebook.com	ALL
fbcd.net	ALL
googleanalytics.com	ALL
example.com	ALL

APPLY EXISTING SITE SETTINGS ▸

For those wanting an additional step in applying existing per-site settings, we added this option to “apply existing site settings.” The option can be toggled in the NoScript options.



Blocking 2 scripts on
facebook.com

ALLOW AFFILIATED SCRIPTS | ALLOW ALL SCRIPTS

These settings are temporary for this tab, and will not be remembered unless saved to site.

SCRIPT SETTINGS
Block or allow scripts owned by:

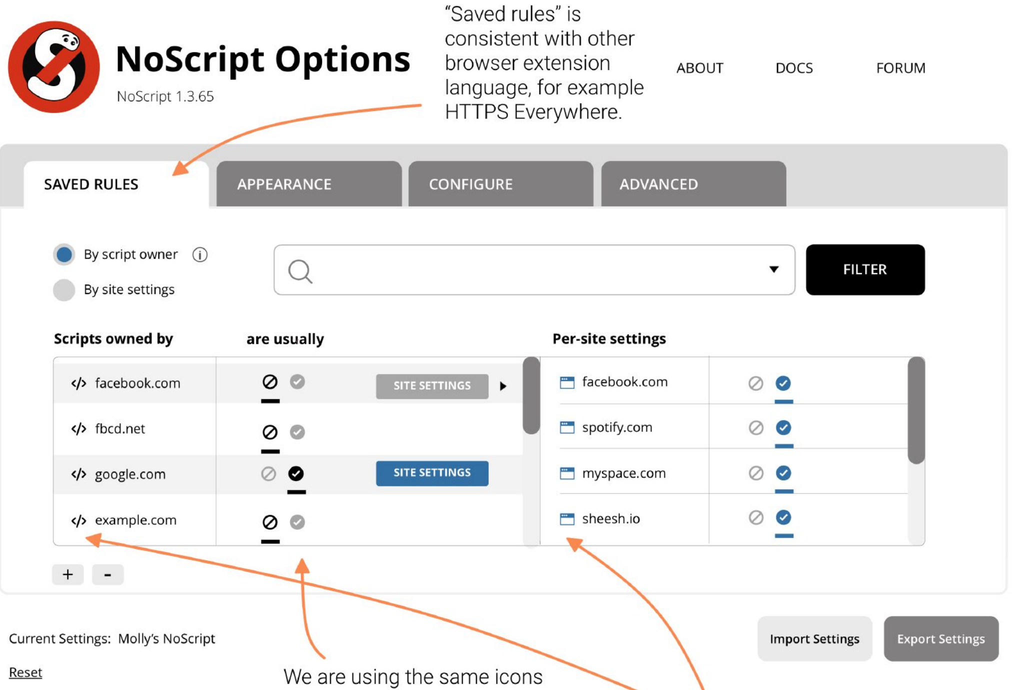
facebook.com	SITE
fbcd.net	SITE
googleanalytics.com	ALL
example.com	ALL

◂ REVERT SITE SETTINGS

Those settings can be reverted at any time; and revisiting the site doesn’t automatically apply the existing settings. This adds another layer of security.

Options: Saved Rules

“Saved rules” is consistent with other browser extension language, for example HTTPS Everywhere.



NoScript Options
NoScript 1.3.65

ABOUT DOCS FORUM

SAVED RULES | APPEARANCE | CONFIGURE | ADVANCED

[✓ By script owner] ⓘ
By site settings

FILTER

Scripts owned by	are usually		Per-site settings	
facebook.com	⊘ ✓	SITE SETTINGS ▸	facebook.com	⊘ ✓
fbcd.net	⊘ ✓		spotify.com	⊘ ✓
google.com	⊘ ✓	SITE SETTINGS	myspace.com	⊘ ✓
example.com	⊘ ✓		sheesh.io	⊘ ✓

+ -

Current Settings: Molly’s NoScript

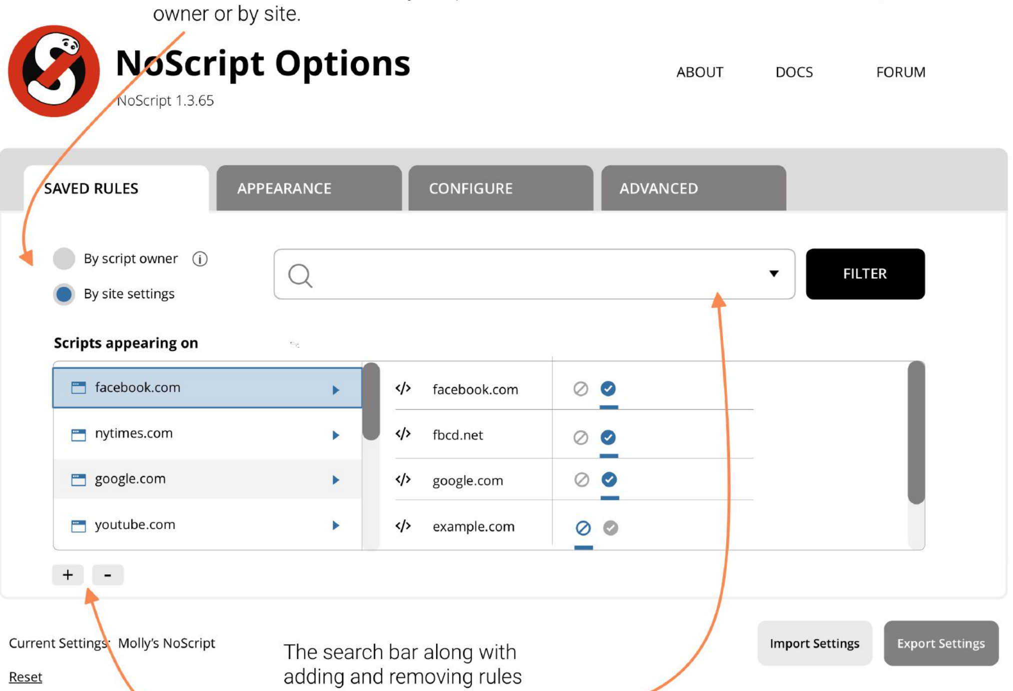
Reset

Import Settings | Export Settings

We are using the same icons and color to indicate scope; this makes it look more familiar.

Per-site settings open up to the right. Different icons indicate when we are referring to domains as the origin of the script (the ***script owner***) as opposed to the ***site*** that has specific settings.

Saved rules can be viewed by script owner or by site.



NoScript Options
NoScript 1.3.65

ABOUT DOCS FORUM

SAVED RULES | APPEARANCE | CONFIGURE | ADVANCED

By script owner ⓘ
[✓ By site settings]

FILTER

Scripts appearing on			
facebook.com ▸	facebook.com	⊘ ✓
nytimes.com ▸	fbcd.net	⊘ ✓
google.com ▸	google.com	⊘ ✓
youtube.com ▸	example.com	⊘ ✓

+ -

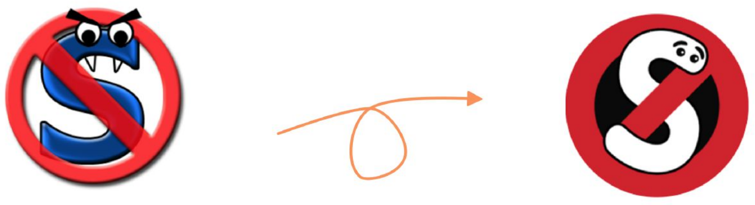
Current Settings: Molly’s NoScript

Reset

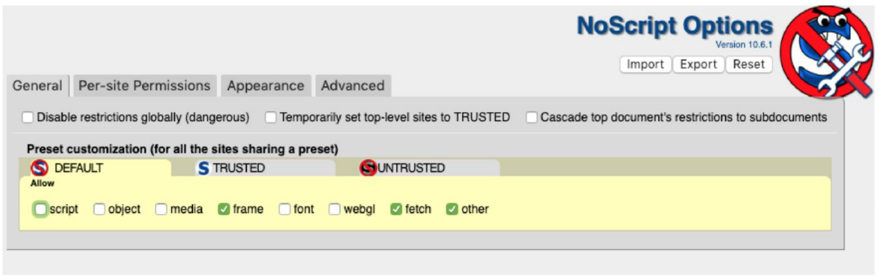
Import Settings | Export Settings

The search bar along with adding and removing rules makes this interface the best place to customize and fine-tune existing rules.

Visual Changes



The logo got a refresh: The essential idea remains the same, but the log is now more readable as a menu icon.



NoScript Options
Version 10.6.1
Import | Export | Reset

General | Per-site Permissions | Appearance | Advanced

☐ Disable restrictions globally (dangerous) ☐ Temporarily set top-level sites to TRUSTED ☐ Cascade top document's restrictions to subdocuments

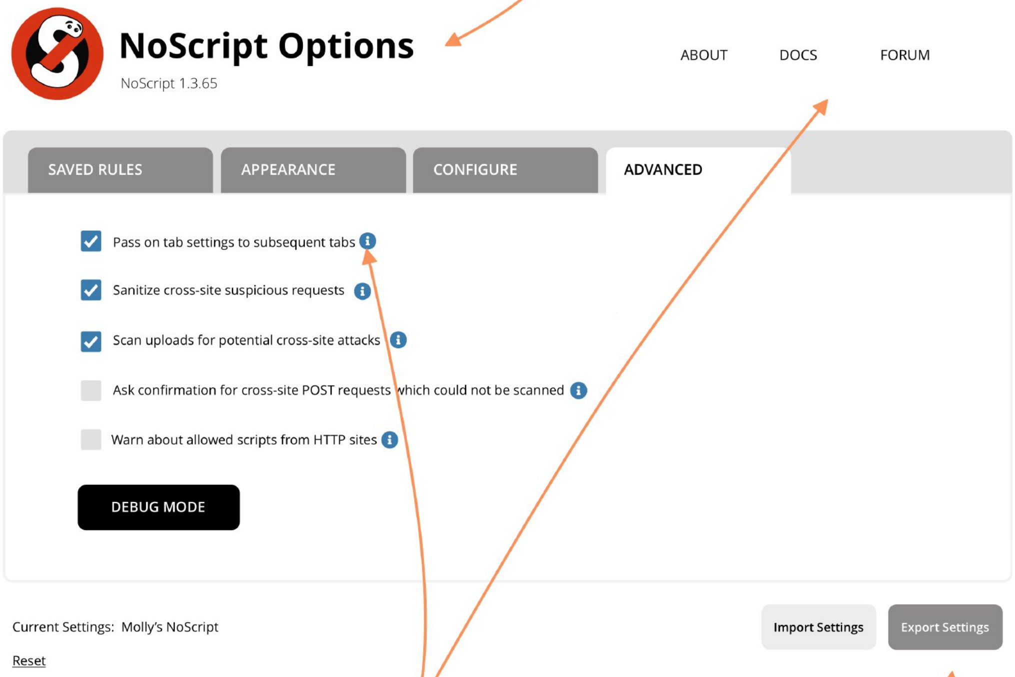
Preset customization (for all the sites sharing a preset)

DEFAULT | TRUSTED | UNTRUSTED

Allow

☐ script ☐ object ☐ media ✓ frame ☐ font ☐ webgl ✓ fetch ✓ other

The options page retains the gray scale design but uses more real estate to lay out information and options.



NoScript Options
NoScript 1.3.65

ABOUT DOCS FORUM

SAVED RULES | APPEARANCE | CONFIGURE | ADVANCED

✓ Pass on tab settings to subsequent tabs ⓘ

✓ Sanitize cross-site suspicious requests ⓘ

✓ Scan uploads for potential cross-site attacks ⓘ

☐ Ask confirmation for cross-site POST requests which could not be scanned ⓘ

☐ Warn about allowed scripts from HTTP sites ⓘ

DEBUG MODE

Current Settings: Molly's NoScript

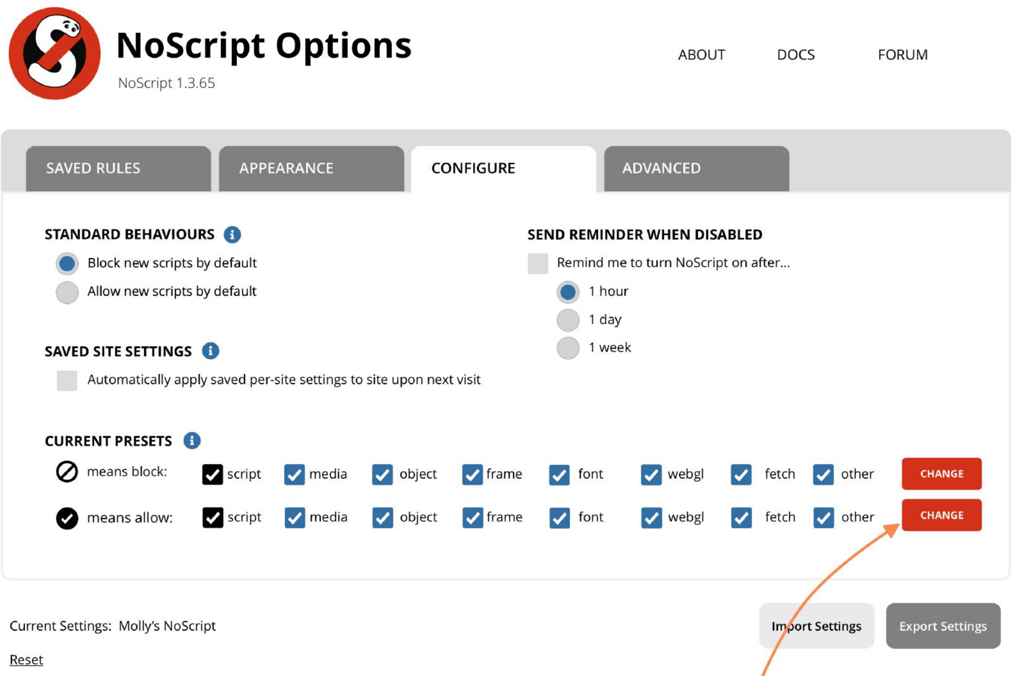
Reset

Import Settings | Export Settings

Additional information helps explain and clarify options. Docs and forum are supplementary resources.

Importing and exporting settings is a less used and advanced option, but because it applies to all settings, we moved it to the bottom.

Options: More advanced configurations


NoScript Options

NoScript 1.3.65

ABOUT DOCS FORUM

SAVED RULES | APPEARANCE | CONFIGURE | ADVANCED

STANDARD BEHAVIOURS

- [✓ Block new scripts by default]
- Allow new scripts by default

SEND REMINDER WHEN DISABLED

- ☐ Remind me to turn NoScript on after...
  - [✓ 1 hour]
  - 1 day
  - 1 week

SAVED SITE SETTINGS

- ☐ Automatically apply saved per-site settings to site upon next visit

CURRENT PRESETS

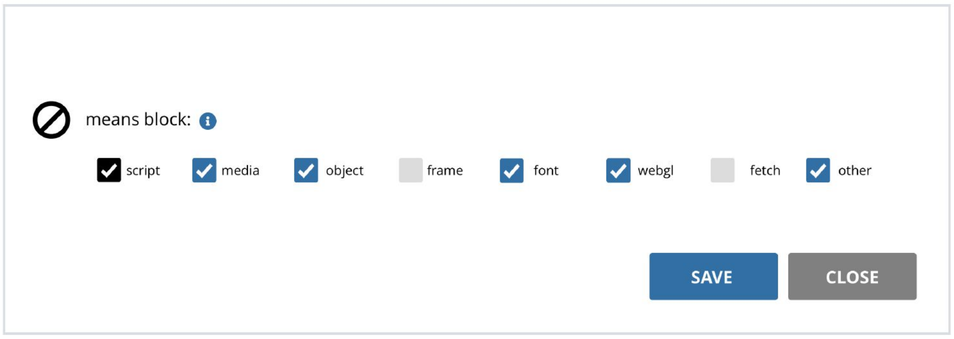
means block:	✓ script	✓ media	✓ object	✓ frame	✓ font	✓ webgl	✓ fetch	✓ other	CHANGE
means allow:	✓ script	✓ media	✓ object	✓ frame	✓ font	✓ webgl	✓ fetch	✓ other	CHANGE

Current Settings: Molly's NoScript

Reset

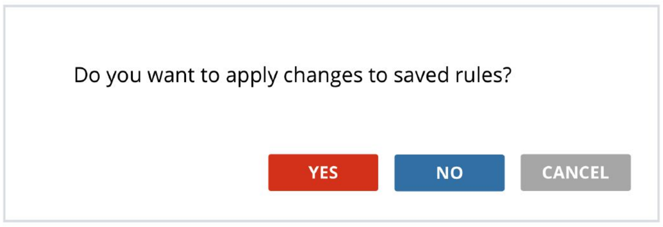
Import Settings | Export Settings

The option to change the definitions of blocking and allowing is now much more hidden. It requires two additional dialogues to do so.


means block:

✓ script	✓ media	✓ object	☐ frame	✓ font	✓ webgl	☐ fetch	✓ other

SAVE | CLOSE


Do you want to apply changes to saved rules?

YES | NO | CANCEL

This dialogue offers the option to apply these changes to existing rules, but also functions as a reminder that these are important changes.

Accessibility

In updating the UI style, we made sure that the contrast ratios of the colors are AAA-compliant. Moreover, we will add keyboard shortcuts to make NoScript more accessible for other input formats, as well as test our helper/hover text with screen readers.

Testing

Our designs (wireframes and clickable prototypes) were tested with experienced NoScript users in July 2019. Iterations have focused on the interfaces surrounding per-site settings. As a next step, we plan to release an alpha version in the fall of 2019. The feedback from this release will further inform the design.

If you would like to participate in alpha testing with us, please get in touch:

ux [at] noscript.net

Credits

Eileen Wagner, Design Lead
Lorraine Chuen, Visual Design
With help from **Molly Wilson** and **Georgia Bullen**
Special thanks to **Susan Farrell**, UX researcher and OTF Advisory Board member
Infinite gratitude to **Giorgio Maone**, developer and maintainer of NoScript
CC BY-SA 4.0 Simply Secure, 2019
Funded by Open Technology Fund's Usability Lab

Simply Secure OPEN TECHNOLOGY FUND

Appendix

Interview Guide

This is a semi-structured interview. Participants are invited to discuss whatever they think is relevant to the conversation.

[existing users]	[new users]

Start

Do you do anything to protect your browsing online? What? What made you do that?

[existing users]	[new users]
How long have you been using NoScript? (Did you stop using it? Why?) Where do you use NoScript?	Have you heard of NoScript?
☐ Firefox [desktop/mobile?]	
☐ Chrome	
☐ Chromium	
☐ Tor Browser	
When is it active and when is it disabled? Why, why not?	What do you think NoScript does?
What about NoScript is most useful to you? Think back to when you first got NoScript… Do you remember why you installed it?	[Scripts are pieces of code that run in your browser. They can be anything from embedded players and sleek animations to ads, trackers, or malicious attacks.] How do you feel about blocking scripts?

Who do you think NoScript is for?

What other browser extensions do you use? Can you show me?
When and why are certain browser extensions disabled?

Tasks

[On separate device] Could you try…

- Download and install NoScript on Firefox
- Go to nytimes.com and make it work as much as you need (at least to display all images and videos) [this could be quickly accomplished using any of the bulk operations, like "Disable restrictions globally / on this tab "Set all on this page as temporarily trusted", or by operating on each script origin individually]
- Allow scripts from google-analytics.com on nytimes.com
- Change settings so that all scripts from cloudflare.com will be allowed globally
- Change settings so that web fonts will be blocked globally
- Adjust settings so that embedded pages (e.g. Youtube movies on other sites) can run even if the parent page has scripts disabled ["cascade top document's restrictions to subdocuments" off]

Screens

Screen of extension: What do you think [trusted | temp. trusted | default | allow scripts globally | close | lock | webgl] does?

Have you ever gone into more fine-grained control? [if yes] When and where was that, and what did you do? [if no] Did you know that they were there? What do you think they might do? Let's look at them now, I'd like to get your opinion about them.

Do you ever look at site information?

Have you seen the new XSS [pop-up | in the settings] ? What do you think about that?

Closing

What's your opinion of NoScript overall? Anything you'd like to tell me about it?

Now that you've seen NoScript, what do you think? Is it different from what you expected?

How would you change NoScript for it to be more useful for you?