Chromium/Tips and tricks

< Chromium

The following tips and tricks should work for both Chromium and Chrome unless explicitly stated.

Contents

Related articles

Chromium

Firefox tweaks

Browsing experience

chrome:// URLs

A number of tweaks can be accessed via Chrome URLs. See **chrome://chrome-urls** for a complete list.

- **chrome://flags** - access experimental features such as WebGL and rendering webpages with GPU, etc.
- **chrome://extensions** - view, enable and disable the currently used Chromium extensions.
- **chrome://gpu** - status of different GPU options.
- **chrome://sandbox** - indicate sandbox status.
- **chrome://version** - display version and switches used to invoke the active `/usr/bin/chromium`.

An automatically updated, complete listing of Chromium switches is available **here (http://peter.sh/experiments/chromium-command-line-switches/)**.

Chromium task manager

Shift+ESC can be used to bring up the browser task manager wherein memory, CPU, and network usage can be viewed.

Chromium overrides/overwrites Preferences file

If you enabled syncing with a Google Account, then Chromium will override any direct edits to the Preferences file found under `~/.config/chromium/Default/Preferences`. To work around this, start Chromium with the `--disable-sync-preferences` switch:

```
$ chromium --disable-sync-preferences
```

If Chromium is started in the background when you login in to your desktop environment, make sure the command your desktop environment uses is:

```
$ chromium --disable-sync-preferences --no-startup-window
```

Search engines

Make sites like **wiki.archlinux.org (https://wiki.archlinux.org)** and **wikipedia.org (https://en.wikipedia.org)** easily searchable by first executing a search on those pages, then going to *Settings* > *Search* and click the *Manage search engines..* button. From there, "Edit" the Wikipedia entry and change its keyword to **w** (or some other shortcut you prefer). Now searching Wikipedia for "Arch Linux" from the address bar is done simply by entering "**w arch linux**".

Note: Google search is used automatically when typing something into the URL bar. A hard-coded keyword trigger is also available using the **?** prefix.

Tmpfs

Cache in tmpfs

Note: Chromium stores its cache separate from its browser profile directory.

To limit Chromium from writing its cache to a physical disk, one can define an alternative location via the `--disk-cache-dir=/foo/bar` flag:

```
$ chromium --disk-cache-dir=/tmp/cache
```

Cache should be considered temporary and will **not** be saved after a reboot or hard lock. Alternatively, use:

```
/etc/fstab
tmpfs   /home/username/.cache   tmpfs   noatime,nodev,nosuid,size=400M  0       0
```

Profile in tmpfs

Relocate the browser profile to a **tmpfs** filesystem, including `/tmp`, or `/dev/shm` for improvements in application response as the entire profile is now stored in RAM.

Use an active profile management script for maximal reliability and ease of use.

`profile-sync-daemon (https://aur.archlinux.org/packages/profile-sync-daemon/)`[AUR] is such a script and is directly available from the **AUR**. It symlinks and syncs the browser profile directories to RAM. Refer to the **Profile-sync-daemon** wiki article for additional information on it.

Launch a new browser instance

When you launch the browser, it first checks if another instance using the same data directory is already running. If there is one, the new window is associated with the old instance. If you want to launch an independent instance of the browser, you must specify separate directory using the `--user-data-dir` parameter:

```
$ chromium --user-data-dir=/path/to/some/directory
```

Note: The default location of the user data is `~/.config/chromium/` .

Directly open *.torrent files and magnet links with a torrent client

By default, Chromium downloads `*.torrent` files directly and you need to click the notification from the bottom-left corner of the screen in order for the file to be opened with your default torrent client. This can be avoided with the following method:

- Download a `*.torrent` file.
- Right-click the notification displayed at the bottom-left corner of the screen.
- Check the "*Always Open Files of This Type*" checkbox.

See **xdg-open** to change the default assocation.

Touch Scrolling on touchscreen devices

You may need to specify which touch device to use. Find your touchscreen device with `xinput list` then launch Chromium with the `--touch-devices=x` parameter, where "**x**" is the id of your device.

Note: If the device is designated as a slave pointer, using this may not work, use the master pointer's ID instead.

Reduce memory usage

By default, Chromium uses a separate OS process for each *instance* of a visited web site. **[1] (https://www.chromium.org/developers/design-documents/process-models#Supported_Models)** However, you can specify command-line switches when starting Chromium to modify this behaviour.

For example, to share one process for all instances of a website:

```
$ chromium --process-per-site
```

To use a single process model:

```
$ chromium --single-process
```

Warning: The single-process model is discouraged because it is unsafe and may contain bugs not present in other models.**[2] (https://www.chromium.org/developers/design-documents/process-models#TOC-Single-process)**

In addition, you can suspend or store inactive Tabs with extensions such as **Tab Suspender (https://chrome.google.com/webstore/detail/tab-suspender/fiabciakcmgepblmdkmemdbbkilneeeh?hl=en)** and **OneTab (https://chrome.google.com/webstore/detail/onetab/chphlpgkkbolifaimnlloiipkdnihall?hl=en)**.

User Agent

The User Agent can be arbitrarily modified at the start of Chromium's base instance via its `--user-agent="[string]"` parameter.

DOM Distiller

Chromium has a similar reader mode to Firefox. In this case it's called DOM Distiller, which is an **open source project (https://github.com/chromium/dom-distiller)**. All you need to do is run Chromium with the `--enable-dom-distiller` flag to unlock the "Distill page" menu option or you can even make it **persistent**. Not only does DOM Distiller provide a better reading experience by distilling the content of the page, it also simplifies pages for print. Even though the latter checkbox option has been removed from the print dialog, you can still print the distilled page, which basically has the same effect.

Running the upper flag, you will find a new "Distill Page" menu item.

You can reach the internal debug page by visiting `chrome://dom-distiller`

Forcing specific GPU

In multi-GPU systems, Chromium automatically detects which GPU should be used for rendering (discrete or integrated). This works 99% of the time, except when it doesn't - if a unavailable GPU is picked (for example, discrete graphics on VFIO GPU passthrough-enabled systems), `chrome://gpu` will complain about not being able to initialize the GPU process. On the same page below **Driver**

Information there'll be multiple GPUs shown (GPU0, GPU1, ...). There's no way to switch between them in a user-friendly way, but you can read the device/vendor IDs present there and configure Chromium to use a specific GPU with flags:

```
$ chromium --gpu-testing-vendor-id=0x8086 --gpu-testing-device-id=0x1912
```

...where `0x8086` and `0x1912` is replaced by the IDs of the GPU you want to use (as shown on the `chrome://gpu` page).

Import bookmarks from Firefox

To ease the transition, you can import bookmarks from **Firefox** into Chromium.

Navigate Chromium to `chrome://settings/importData`

If Firefox is already installed on your computer, you can directly import bookmarks as well as many other things from Firefox.

Make sure **Mozilla Firefox** is selected. Optionally, you can uncheck some unwanted items here. Click the **Import** and then **Done**. You're done with it.

Note: If you haven't created any bookmarks in Chromium yet, the bookmarks will show up in your bookmarks bar. If you already have bookmarks, the bookmarks will be in a new folder labeled "Imported From Firefox"

If you import bookmarks from another PC, you have to export bookmarks from Firefox first.

`Ctrl + Shift + O > Import and Backup > Export Bookmarks To HTML` *in Firefox*

The procedure is pretty much the same. You need to go to `chrome://settings/importData`. However, this time, in the **From** drop-down menu, select **Bookmarks HTML File** and click the **Choose File** button and upload the desired bookmark file.

Enabling native notifications

Go to `chrome://flags#enable-native-notifications` and select *Enabled*.

Profile maintenance

Chromium uses **SQLite** databases to manage history and the like. Sqlite databases become fragmented over time and empty spaces appear all around. But, since there are no managing processes checking and optimizing the database, these factors eventually result in a performance hit. A good way to improve startup and some other bookmarks- and history-related tasks is to defragment and trim unused space from these databases.

profile-cleaner (https://aur.archlinux.org/packages/profile-cleaner/)[AUR] and **browser-vacuum (https://aur.archlinux.org/packages/browser-vacuum/)**[AUR] in the **AUR** do just this.

Security

WebRTC

WebRTC is a communication protocol that relies on JavaScript that can leak one's actual IP address and hardware hash from behind a VPN. While some software may prevent the leaking scripts from running, it's probably a good idea to block this protocol directly as well, just to be safe. As of October 2016, there is no way to disable WebRTC on Chromium on desktop, there are extensions available to disable local IP address leak, one is this **extension (https://chrome.google.com/webstore/detail/webrtc-network-limiter/npeicpdbkakmehahjeeohfdhnlpdklia)**.

One can test WebRTC via **this page (https://www.privacytools.io/webrtc.html)**.

Warning: Even though IP leak can be prevented, Chromium still sends your unique hash, and there is no way to prevent this. Read more on **https://www.browserleaks.com/webrtc#webrtc-disable**

SSL certificates

Chromium does not have an SSL certificate manager. It relies on the NSS Shared DB `~/.pki.nssdb`. In order to add SSL certificates to the database, users will have to use the shell.

Adding CAcert certificates for self-signed certificates

Grab the CAcerts and create an `nssdb`, if one does not already exist. To do this, first install the **`nss (https://www.archlinux.org/packages/?name=nss)`** package, then complete these steps:

```
$ mkdir -p $HOME/.pki/nssdb
$ cd $HOME/.pki/nssdb
$ certutil -N -d sql:.
```

```
$ curl -k -o "cacert-root.crt" "http://www.cacert.org/certs/root.crt"
$ curl -k -o "cacert-class3.crt" "http://www.cacert.org/certs/class3.crt"
```

```
$ certutil -d sql:$HOME/.pki/nssdb -A -t TC -n "CAcert.org" -i cacert-root.crt
$ certutil -d sql:$HOME/.pki/nssdb -A -t TC -n "CAcert.org Class 3" -i cacert-class3.crt
```

Note: Users will need to create a password for the database, if it does not exist.

Now users may manually import a self-signed certificate.

Example 1: Using a shell script to isolate the certificate from TomatoUSB

Below is a simple script that will extract and add a certificate to the user's `nssdb`:

```
#!/bin/sh
#
# usage:  import-cert.sh remote.host.name [port]
#
REMHOST=$1
REMPORT=${2:-443}
exec 6>&1
exec > $REMHOST
echo | openssl s_client -connect ${REMHOST}:${REMPORT} 2>&1 |sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p'
certutil -d sql:$HOME/.pki/nssdb -A -t "P,," -n "$REMHOST" -i $REMHOST
exec 1>&6 6>&-
```

Syntax is advertised in the commented lines.

References:

- **http://blog.avirtualhome.com/adding-ssl-certificates-to-google-chrome-linux-ubuntu**
- **https://chromium.googlesource.com/chromium/src/+/master/docs/linux_cert_management.md**

Example 2: Using Firefox to isolate the certificate from TomatoUSB

The `firefox (https://www.archlinux.org/packages/?name=firefox)` browser can be used to save the certificate to a file for manual import into the database.

Using firefox:

1. Browse to the target URL.
2. Upon seeing the "This Connection is Untrusted" warning screen, click: *I understand the Risks* > *Add Exception...*
3. Click: *View* > *Details* > *Export* and save the certificate to a temporary location (`/tmp/easy.pem` in this example).

Now import the certificate for use in Chromium:

```
$ certutil -d sql:$HOME/.pki/nssdb -A -t TC -n "easy" -i /tmp/easy.pem
```

Note: Adjust the name to match that of the certificate. In the example above, "easy" is the name of the certificate.

Reference:

- **http://sahissam.blogspot.com/2012/06/new-ssl-certificates-for-tomatousb-and.html**

Canvas Fingerprinting

Canvas fingerprinting is a technique that allows websites to identify users by detecting differences when rendering to an HTML5 canvas. This information can be made inaccessible by using the `--disable-reading-from-canvas` flag.

To confirm this is working run **this test (https://panopticlick.eff.org)** and make sure "hash of canvas fingerprint" is reported as undetermined in the full results.

Note: Some extensions require reading from canvas and may be broken by setting `--disable-reading-from-canvas`.

Privacy extensions

Popular privacy extensions for the **Firefox** browser are typically also available for Chromium. See **Firefox/Privacy#Extensions** for details.

Tip: Installing too many extensions might take up much space in the toolbar. Those extensions which you wouldn't interact with anyway (e.g. **HTTPS Everywhere (https://chrome.google.com/webstore/detail/gcbommkclmclpchllfjekcdonpmejbdp)**) can be hidden by right-clicking on the extension and choosing **Hide in Chromium menu**.

Tip: It's not recommended to install all the privacy extensions. It can be counterproductive as they conflict with each other and doesn't increase security whatsoever.

- **HTTPS Everywhere (https://chrome.google.com/webstore/detail/gcbommkclmclpchllfjekcdonpmejbdp)**
- **uBlock Origin (https://chrome.google.com/webstore/detail/ublock-origin/cjpalhdlnbpafiamejdnhcphjbkeiagm?hl=en)**
- **Adblock Plus (https://chrome.google.com/webstore/detail/adblock-plus/cfhdojbkjhnklbpkdaibdccddilifddb?hl=en)**

- **Privacy Badger (https://chrome.google.com/webstore/detail/privacy-badger/pkehgijcmpdhfbdbbnkijodmdjhbjlgp?hl=en)**
- **Disconnect (https://chrome.google.com/webstore/detail/disconnect/jeoacafpbcihiomhlakheieifhpjdfeo?hl=en)**
- **Decentraleyes (https://chrome.google.com/webstore/detail/decentraleyes/ldpochfccmkkmhdbclfhpagapcfdljkj?hl=en)**
- **AdNauseam (https://github.com/dhowe/AdNauseam/releases)**

ScriptBlock

ScriptBlock is similar to NoScript, which is a Firefox add-on. Both extensions stop a website from executing any kind of JavaScript. However, ScriptBlock is a much simpler design thus it's easier to use. It blocks JavaScript by default. You can allow and temporary allow JavaScripts. Once you allow them to run, it lets all the JavaScripts run on that page so you might want ScriptBlock to work in conjunction with **Privacy Badger (https://chrome.google.com/webstore/detail/privacy-badger/pkehgijcmpdhfbdbbnkijodmdjhbjlgp?hl=en)**.

It's also worth checking it's default whitelist, which might be permissive to you.

Extension is available in the **Chrome Web Store (https://chrome.google.com/webstore/detail/scriptblock/hcdjknjpbnhdoabbngpmfekaecnpajba?hl=en-US)**

ScriptSafe

ScriptSafe is a browser extension that gives users control of the web and more secure browsing while emphasizing simplicity and intuitiveness.

Note: Due to the nature of this extension, this will break most sites! It is designed to learn over time with sites that you allow.

Check it on **GitHub (https://github.com/andryou/scriptsafe)** or in the **Chrome Web Store (https://chrome.google.com/webstore/detail/scriptsafe/oiigbmnaadbkfbmpbfijlflahbdbdgdf?hl=en)**.

Vanilla Cookie Manager

A Cookie Whitelist Manager for Chrome that helps protect your privacy. Automatically removes unwanted cookies. Cookies can be used for authentication, storing your site preferences or anything else that can be saved as text data. Unfortunately they can also be used to track you.

You could turn off cookies completely or just shut off third-party cookies. But that would also keep out useful cookies that many web apps rely upon to work (like Google Mail or Calendar).

With Vanilla you can select which cookies you want to keep on a whitelist. All unwanted cookies are deleted automatically (or manually if you prefer).

Vanilla Cookie Manager on **GitHub (https://github.com/laktak/vanilla-chrome)** or in the **Chrome Web Store (https://chrome.google.com/webstore/detail/vanilla-cookie-manager/gieohaicffldbmiilohhggbidhephnjj)**.

TrackMeNot

TrackMeNot periodically issues randomized search-queries to popular search engines and helps you hide your real ones in a cloud of 'ghost' queries.

Extension is available in the **Chrome Web Store (https://chrome.google.com/webstore/detail/track menot/cgllkjmdafllcidaehjejjhpfkmanmka?hl=en-US)**

Do Not Track

To enable **Do Not Track**, visit `chrome://settings`, scroll down to *Advanced* and under *Privacy and security*, check *Send a "Do Not Track" request with your browsing traffic*.

Force a password store

Chromium uses a password store to store your passwords and the *Chromium Safe Storage* key, which is used to encrypt cookie values. **[3] (https://codereview.chromium.org/24734007)**

By default Chromium auto-detects which password store to use, which can lead to you apparently losing your passwords and cookies when switching to another desktop environment or window manager.

You can force Chromium to use a specific password store by launching it with the `--password-store` flag with one of following the values **[4] (https://chromium.googlesource.com/chromium/src/+/master/docs/linux_password_storage.md)**:

- `gnome`, uses **Gnome Keyring**
- `kwallet`, uses **KDE Wallet**
- `basic`, saves the passwords and the cookies' encryption key as plain text in the file `Login Data`
- `detect`, the default auto-detect behavior

For example, to force Chromium to use Gnome Keyring in another desktop or WM use `--password-store=gnome`, see **#Making flags persistent** for making it permanent.

When using a password store of another desktop environment you probably also want to unlock it automatically see: **GNOME/Keyring#Using the keyring outside GNOME** and **KDE Wallet#Unlock KDE Wallet automatically on login**.

Making flags persistent

Note: The `chromium-flags.conf` file and the accompanying custom launcher script are specific to the Arch Linux **chromium (https://www.archlinux.org/packages/?name=chromium)** package. For **google-chrome (https://aur.archlinux.org/packages/google-chrome/)**[AUR], use `chrome-flags.conf` instead.

You can put your flags in a `chromium-flags.conf` file under `$HOME/.config/` (or under `$XDG_CONFIG_HOME` if you have configured that environment variable).

No special syntax is used; flags are defined as if they were written in a terminal.

- The arguments are split on whitespace and shell quoting rules apply, but no further parsing is performed.
- In case of improper quoting anywhere in the file, a fatal error is raised.
- Flags can be placed in separate lines for readability, but this is not required.
- Lines starting with a hash symbol (#) are skipped.

Below is an example `chromium-flags.conf` file that defines the flags `--start-maximized --incognito`:

```
# This line will be ignored.
--start-maximized
--incognito
```

See also

- **Profile-sync-daemon** - Systemd service that saves Chromium profile in tmpfs and syncs to disk
- **Tmpfs** - Tmpfs Filesystem in `/etc/fstab`
- **Official tmpfs kernel Documentation (https://www.kernel.org/doc/Documentation/filesystems/tmpfs.txt)**