How do I auto-reload a Chrome extension I'm developing?

Asked 13 years, 10 months ago   Modified 5 months ago   Viewed 119k times

359

I'd like for my chrome extension to reload every time I save a file in the extension folder, without having to explicitly click "reload" in chrome://extensions/. Is this possible?

Edit: I'm aware I can update the interval at which Chrome reloads extensions, which is a half-way solution, but I'd rather either making my editor (emacs or textmate) trigger on-save a reload or asking Chrome to monitor the directory for changes.

javascript   google-chrome   google-chrome-extension

Share Edit Follow

edited Jul 2, 2016 at 17:21
Greg Sadetsky
5,017 ● 2 ● 39 ● 50

asked Jun 3, 2010 at 4:06
Andrey Fedorov
9,419 ● 21 ● 68 ● 99

2 It seems to be faster when the reload is triggered from the ui enabled at chrome://flags/#enable-apps-devtool-app – Eric Oct 16, 2013 at 22:20

9 This doesn't answer the question as asked, but I want to mention that I stopped wanting this (or any other extension-loading-helper extension) after I realized I can easily reload an extension by hitting ctrl-R or cmd-R (depending on OS) in the extension's background window. I find this fits into my workflow better than anything else I've tried, and it also avoids the problem of auto-reloading when files are in an inconsistent state. – Don Hatch Mar 26, 2016 at 14:45

1 @DonHatch, Where is the "extension's background window"? – Iulian Onofrei Jul 18, 2021 at 10:53

31 Answers

Sorted by: Highest score (default)

1 2 Next

198

You can use "**Extensions Reloader**" for Chrome:

> **Reloads all unpacked extensions using the extension's toolbar button or by browsing to "http://reload.extensions"**
>
> If you've ever developed a Chrome extension, you might have wanted to automate the process of reloading your unpacked extension without the need of going through the extensions page.
>
> "Extensions Reloader" allows you to reload all unpacked extensions using 2 ways:
>
> 1 - The extension's toolbar button.

> 2 - Browsing to "http://reload.extensions".
>
> The toolbar icon will reload unpacked extensions using a single click.
>
> The "reload by browsing" is intended for automating the reload process using "post build" scripts - just add a browse to "http://reload.extensions" using Chrome to your script, and you'll have a refreshed Chrome window.

Update: As of January 14, 2015, the extension is open-sourced and available on GitHub.

Share Edit Follow edited Jan 31, 2015 at 18:51 answered Oct 7, 2012 at 8:18

Arik
6,333 ●2 ●34 ●43

5 This is the best solution so far, but it still doesn't make it easy to update the extension every time I modify a file it's using, which is what I was ideally looking. – Andrey Fedorov Oct 10, 2012 at 2:55

2 Thanks. Wanted to mention that I use PhpStorm and that I added a toolbar button that calls chrome.exe with the "reload.extensions" url. Whenever I want to test the code, I press that button and Chrome popups up with the updated addon. – Arik Oct 10, 2012 at 6:05

32 Woaw, I just managed to get gruntjs to work with this plugin by using grunt-open to open `http://reload.extensions` when a file gets modified. It now acts exactly like livereload, but with an anoying chrome window opening each time I modify one of the specified plugin files :P. Thanks! – GabLeRoux Jul 15, 2013 at 19:20

1 @GabLeRoux I came here just to find if someone already did it. It works like a charm. – polkovnikov.ph Feb 1, 2014 at 15:57

1 @AlexanderMills sadly I wasn't able to do it without opening a new tab each time. If you find a better solution. At least, one can reload plugins with the github.com/arikw/chrome-extensions-reloader – GabLeRoux Jan 3, 2018 at 14:45

66

Update: I have added an options page, so that you don't have to manually find and edit the extension's ID any more. CRX and source code are at: https://github.com/Rob--W/Chrome-Extension-Reloader
Update 2: Added shortcut (see my repository on Github).

*The original code, which includes the **basic functionality** is shown below.*

Create an extension, and use the Browser Action method in conjunction with the `chrome.extension.management` API to reload your unpacked extension.

The code below adds a button to Chrome, which will reload an extension upon click.

`manifest.json`

```
{
    "name": "Chrome Extension Reloader",
    "version": "1.0",
    "manifest_version": 2,
    "background": {"scripts": ["bg.js"] },
    "browser_action": {
        "default_icon": "icon48.png",
        "default_title": "Reload extension"
    },
    "permissions": ["management"]
}
```

`bg.js`

```
var id = "<extension_id here>";
function reloadExtension(id) {
    chrome.management.setEnabled(id, false, function() {
        chrome.management.setEnabled(id, true);
    });
}
chrome.browserAction.onClicked.addListener(function(tab) {
    reloadExtension(id);
});
```

`icon48.png`: Pick any nice 48x48 icon, for example:

Share Edit Follow

edited Aug 18, 2019 at 5:14
Glorfindel
22.3k ● 13 ● 86 ● 114

answered Mar 10, 2012 at 9:41
Rob W
345k ● 85 ● 800 ● 680

1 @trusktr Scott's answer only reloads the background page. However, both answers can be combines (Scotts answer + helper extension), so that an extension is automatically reload when an extension's file changes. I do, however recommend against this practice, since the devtools are closed upon extension reload. I often fiddle with a test extension, and frequently save my changes, even when the syntax is incomplete. That would crash my extension, if a real auto-reload is enabled. – Rob W Jun 9, 2012 at 7:56

1 @Sudarshan You suggested to use `chrome.i18n.getMessage("@@extension_id")`. This, however returns the extensionID of the *current* extension, which is not useful because an extension cannot reload itself using the `management` API (after disabling, it doesn't have a chance to enable the extension again). – Rob W Dec 25, 2012 at 11:47

55

in any function or event

```
chrome.runtime.reload();
```

will reload your extension (docs). You also need to change the **manifest.json** file, adding:

```
...
"permissions": [ "management" , ...]
...
```

Share Edit Follow

edited Nov 11, 2018 at 0:47
philipp
4,153 ● 1 ● 36 ● 35

answered May 22, 2013 at 22:11
Leonardo Ciaccio
3,006 ● 1 ● 16 ● 17

for me this solution only worked if I wrote chrome.runtime.reload() in the script in the index.html where is specified in the manifest file in the follow way: ** "background": { "page": "index.html", ** . In my content_script js file, I couldn't access this function. I believe, it is for security reseons. – titusfx Jun 14, 2017 at 17:54

Your content script can't talk to chrome APIs. For that to work you need to use messages – marcelocra Sep 19, 2017 at 1:10

when you call `chrome.runtime.reload()`, what actually happens? do multiple pages/tabs refresh? – Alexander Mills Jan 1, 2018 at 23:22

5 I am guessing we should call `chrome.runtime.reload()` from the background script – Alexander Mills Jan 1, 2018 at 23:49

1 Worked for me. Thanks – MikeMurko Apr 7, 2018 at 16:11

is there a way to connect to websocket server from the background.js script? – Alexander Mills Nov 26, 2023 at 5:06

54

I've made a simple embeddable script doing hot reload:

https://github.com/xpl/crx-hotreload

It watches for file changes in an extension's directory. When a change detected, it reloads the extension and refreshes the active tab (to re-trigger updated content scripts).

- Works by checking timestamps of files
- Supports nested directories
- Automatically disables itself in the production configuration

Share Edit Follow

edited Nov 9, 2016 at 17:54

answered Nov 6, 2016 at 20:35
Vitaly Gordon
1,059 ● 9 ● 9

3 If you are looking for a "gulp flavored" solution, this is it. – ow3n Jul 7, 2017 at 7:00

1 worked great. Have you considered making an npm package? – pixelearth Sep 15, 2017 at 18:00

1 @MilfordCubicle I would suggest exporting an object that has a method that you could call in your background.js file. So something `import reloader from 'hot-reload'` then in the script reloader.check() – pixelearth Sep 18, 2017 at 19:26

5 The answer needs more upvotes. Works well in combination with Webpack `--watch` as well, because you can just put it in your build folder so it only updates after the build process has finished. No complicated Webpack post-build command plugins. – V. Rubinetti Nov 20, 2018 at 23:04

4 Doesn't work with manifest version 3 – xquilt Jan 14, 2023 at 22:43

29

I am using a shortcut to reload. I don't want to reload all the time when I save a file

So my approach is lightweight, and you can leave the reload function in

manifest.json

```
{
    ...
    "background": {
        "scripts": [
            "src/bg/background.js"
        ],
        "persistent": true
    },
    "commands": {
        "Ctrl+M": {
            "suggested_key": {
                "default": "Ctrl+M",
                "mac": "Command+M"
            },
            "description": "Ctrl+M."
        }
    },
    ...
}
```

src/bg/background.js

```
chrome.commands.onCommand.addListener((shortcut) => {
    console.log('lets reload');
    console.log(shortcut);
    if(shortcut.includes("+M")) {
        chrome.runtime.reload();
    }
})
```

Now press Ctrl + M in the chrome browser to reload

Share Edit Follow

edited Jun 11, 2019 at 13:30

answered May 30, 2019 at 14:40

Johan Hoeksma

3,626 ● 5 ● 29 ● 40

16

TL;DR

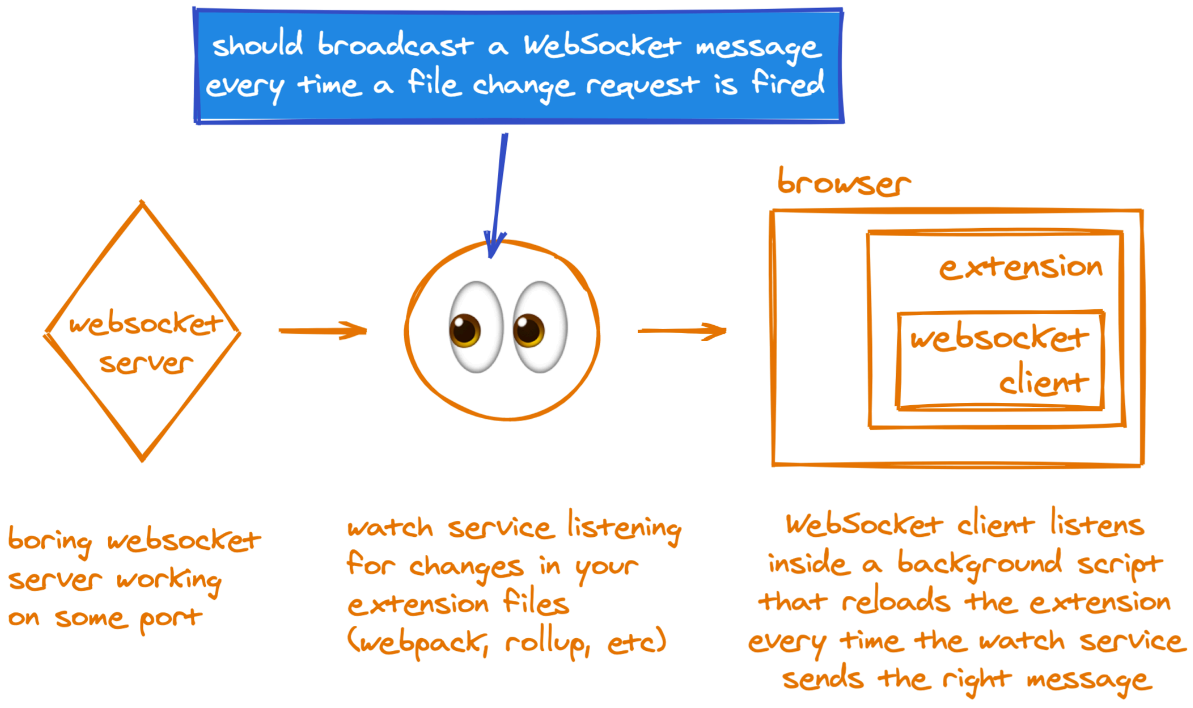
Create a WebSocket server that dispatches a message to a background script that can handle the update. If you are using webpack and don't plan to do it yourself, webpack-run-chrome-extension can help.

Answer

You can create a WebSocket server to communicate with the extension as a WebSocket client (via `window` object). The extension would then listen for file changes by attaching the

WebSocket server to some listener mechanism (like webpack devServer).

Did the file change? Set the server to dispatch a message to the extension asking for updates (broadcasting the ws message to the client(s)). The extension then reloads, replies with "ok, reloaded" and keeps listening for new changes.


should broadcast a WebSocket message every time a file change request is fired
browser
extension
websocket client
websocket server
boring websocket server working on some port
watch service listening for changes in your extension files (webpack, rollup, etc)
WebSocket client listens inside a background script that reloads the extension every time the watch service sends the right message

Plan

1. Set up a WebSocket server (to dispatch update requests)
2. Find a service that can tell you when did the files change (webpack/other bundler software)
3. When an update happens, dispatch a message to client requesting updates
4. Set up a WebSocket client (to receive update requests)
5. Reload the extension

How

For the WebSocket server, use `ws`. For file changes, use some listener/hook (like webpack's `watchRun` hook). For the client part, native WebSocket. The extension could then attach the WebSocket client on a background script for keeping sync persistent between the server (hosted by webpack) and the client (the script attached in the extension background).

Now, to make the extension reload itself, you can either call `chrome.runtime.reload()` in it each time the upload request message comes from the server, or even create a "reloader extension" that would do that for you, using `chrome.management.setEnabled()` (requires `"permissions": [ "management" ]` in manifest).

In the ideal scenario, tools like `webpack-dev-server` or any other web server software could offer support for `chrome-extension` URLs natively. Until that happens, having a server to proxy file changes to your extension seems to be the best option so far.

Available open-source alternative

If you are using webpack and don't want to create it all yourself, I made webpack-run-chrome-extension, which does what I planned above.

Share Edit Follow

answered Dec 29, 2020 at 0:49
Cezar Augusto
9,432 ● 5 ● 31 ● 36

11

Another solution would be to create custom livereload script (extension-reload.js):

```
// Reload client for Chrome Apps & Extensions.
// The reload client has a compatibility with livereload.
// WARNING: only supports reload command.

var LIVERELOAD_HOST = 'localhost:';
var LIVERELOAD_PORT = 35729;
var connection = new WebSocket('ws://' + LIVERELOAD_HOST + LIVERELOAD_PORT +
'/livereload');

connection.onerror = function (error) {
  console.log('reload connection got error:', error);
};

connection.onmessage = function (e) {
  if (e.data) {
    var data = JSON.parse(e.data);
    if (data && data.command === 'reload') {
      chrome.runtime.reload();
    }
  }
};
```

This script connects to the livereload server using websockets. Then, it will issue a chrome.runtime.reload() call upon reload message from livereload. The next step would be to add this script to run as background script in your manifest.json, and voila!

Note: this is not my solution. I'm just posting it. I found it in the generated code of Chrome Extension generator (Great tool!). I'm posting this here because it might help.

Share Edit Follow

answered Apr 8, 2015 at 9:33
refaelio
408 ● 5 ● 10

The great guys at mozilla just released a new https://github.com/mozilla/web-ext that you can use to launch `web-ext run --target chromium`

11

Share Edit Follow

answered Oct 22, 2019 at 19:36
zirqLvgOhM
190 ● 1 ● 2 ● 14

2 This is the best solution for me. I also use it to sign my extension – Johan Hoeksma Oct 27, 2021 at 19:39

9

+350

Chrome Extensions have a permission system that it wouldn't allow it (some people in SO had the same problem as you), so requesting them to "add this feature" is not going to work IMO. There's a mail from Chromium Extensions Google Groups with a proposed solution (theory) using `chrome.extension.getViews()`, but is not guaranteed to work either.

If it was possible to add to the `manifest.json` some Chrome internal pages like `chrome://extensions/`, it would be possible to create a plugin that would interact to the `Reload` anchor, and, using an external program like XRefresh (a Firefox Plugin - there's a Chrome version using Ruby and WebSocket), you would achieve just what you need:

> XRefresh is a browser plugin which will refresh current web page due to file change in selected folders. This makes it possible to do live page editing with your favorite HTML/CSS editor.

It's not possible to do it, but I think you can use this same concept in a different way.

You could try to find third-party solutions instead that, after seeing modifications in a file (I don't know emacs neither Textmate, but in Emacs it would be possible to bind an app call within a "save file" action), just clicks in an specific coordinate of an specific application: in this case it's the `Reload` anchor from your extension in development (you leave a Chrome windows opened just for this reload).

(Crazy as hell but it may work)

Share Edit Follow

edited May 23, 2017 at 11:33
Community Bot
1 ● 1

answered Jun 9, 2010 at 3:48
GmonC
10.9k ● 1 ● 30 ● 38

7

Here's a function that you can use to watch files for changes, and reload if changes are detected. It works by polling them via AJAX, and reloading via window.location.reload(). I suppose you shouldn't use this in a distribution package.

```
function reloadOnChange(url, checkIntervalMS) {
    if (!window.__watchedFiles) {
        window.__watchedFiles = {};
    }
    (function() {
        var self = arguments.callee;
        var xhr = new XMLHttpRequest();

        xhr.onreadystatechange = function() {
            if (xhr.readyState == 4) {
                if (__watchedFiles[url] &&
                    __watchedFiles[url] != xhr.responseText) {
                    window.location.reload();
                } else {
                    __watchedFiles[url] = xhr.responseText
                    window.setTimeout(self, checkIntervalMS || 1000);
                }
            }
        };

        xhr.open("GET", url, true);
        xhr.send();
    })();
}

reloadOnChange(chrome.extension.getURL('/myscript.js'));
```

Share Edit Follow

answered Apr 18, 2012 at 16:11
Scott
4,080 ● 3 ● 22 ● 16

6

Maybe I'm a little late to the party, but I've solved it for me by creating https://chrome.google.com/webstore/detail/chrome-unpacked-extension/fddfkmklefkhanofhlohnkemejcbamln

It works by reloading `chrome://extensions` page, whenever `file.change` events are incoming via websockets.

A Gulp-based example of how to emit `file.change` event upon file changes in an extension folder can be found here: https://github.com/robin-drexler/chrome-extension-auto-reload-watcher

Why reloading the entire tab instead of just using the extensions management api to reload/re-enable extensions? Currently disabling and enabling extensions again causes any open inspection window (console log etc.) to close, which I found to be too annoying during active development.

Share Edit Follow

answered Jul 30, 2015 at 20:47
Robin Drexler
4,357 ● 3 ● 26 ● 28

5

There's an automatic reload plugin if you're developing using webpack: https://github.com/rubenspgcavalcante/webpack-chrome-extension-reloader

```
const ChromeExtensionReloader = require('webpack-chrome-extension-reloader');
```

```
plugins: [
    new ChromeExtensionReloader()
]
```

Also comes with a CLI tool if you don't want to modify webpack.config.js:

```
npx wcer
```

Note: an (empty) background script is required even if you don't need it because that's where it injects reload code.

Share Edit Follow

answered Apr 4, 2019 at 15:25
Eugene Kulabuhov
2,419 ● 1 ● 27 ● 26

3

Maybe a bit late answer but I think crxreload might work for you. It's my result of trying to have a reload-on-save workflow while developing.

Share Edit Follow

answered Aug 29, 2013 at 23:13
Mattias Wadman
11.3k ● 2 ● 44 ● 59

3

Use `npm init` to create a package.json in directory root, then

`npm install --save-dev gulp-open && npm install -g gulp`

then create a `gulpfile.js`

which looks like:

```
/* File: gulpfile.js */

// grab our gulp packages
var gulp  = require('gulp'),
    open = require('gulp-open');

// create a default task and just log a message
gulp.task('default', ['watch']);

// configure which files to watch and what tasks to use on file changes
gulp.task('watch', function() {
  gulp.watch('extensionData/userCode/**/*.js', ['uri']);
});

gulp.task('uri', function(){
  gulp.src(__filename)
  .pipe(open({uri: "http://reload.extensions"}));
});
```

This works for me developing with CrossRider, you might watch to change the path you watch the files at, also assuming you have npm and node installed.

Share Edit Follow

answered Feb 21, 2016 at 17:42

Anthony
13.9k ● 14 ● 62 ● 83

2

Your content files such has html and manifest files are not changeable without installation of the extension, but I do believe that the JavaScript files are dynamically loaded until the extension has been packed.

I know this because of a current project im working on via the Chrome Extensions API, and seems to load every-time i refresh a page.

Share Edit Follow

answered Jun 9, 2010 at 3:52

RobertPitt
57.1k ● 21 ● 114 ● 161

2

Disclaimer: I developed this extension myself.

Clerc - for Chrome Live Extension Reloading Client

> Connect to a LiveReload compatible server to automatically reload your extension every time you save.
>
> Bonus: with a little extra work on your part, you can also automatically reload the webpages that your extension alters.

Most webpage developers use a build system with some sort of watcher that automatically builds their files and restarts their server and reloads the website.

Developing extensions shouldn't need to be that different. Clerc brings this same automation to Chrome devs. Set up a build system with a LiveReload server, and Clerc will listen for reload events to refresh your extension.

The only big gotcha is changes to the manifest.json. Any tiny typos in the manifest will probably cause further reload attempts to fail, and you will be stuck uninstalling/reinstalling your extension to get your changes loading again.

Clerc forwards the complete reload message to your extension after it reloads, so you can optionally use the provided message to trigger further refresh steps.

Share Edit Follow

edited Jun 20, 2020 at 9:12
Community Bot
1 ● 1

answered Sep 21, 2017 at 2:22
skylize
1,411 ● 1 ● 10 ● 21

1

Thanks to @GmonC and @Arik and some spare time, I managet to get this working. I have had to change two files to make this work.

(1) Install LiveReload and Chrome Extension for that application. This will call some script on file change.

(2) Open `<LiveReloadInstallDir>\Bundled\backend\res\livereload.js`

(3) change line `#509` to

`this.window.location.href = "http://reload.extensions";`

(4) Now install another extension `Extensions Reloader` which has useful link handler that reload all development extensions on navigating to `"http://reload.extensions"`

(5) Now change that extension's `background.min.js` in this way

`if((d.installType=="development")&&(d.enabled==true)&&(d.name!="Extensions Reloader"))` replace with

```
if((d.installType=="development")&&(d.enabled==true)&&(d.name!="Extensions Reloader")&&
(d.name!="LiveReload"))
```

Open LiveReload application, hide Extension Reloader button and activate LiveReload extension by clicking on button in toolbar, you will now reload page and extensions on each file change while using all other goodies from LiveReload (css reload, image reload etc.)

Only bad thing about this is that you will have to repeat procedure of changing scripts on every extension update. To avoid updates, add extension as unpacked.

When I'll have more time to mess around with this, I probably will create extension that eliminates need for both of these extensions.

Untill then, I'm working on my extension Projekt Axeman

Share Edit Follow

answered Jan 17, 2013 at 21:01
Aleksandar Toplek
2,801 ● 30 ● 44

1

Just found a newish grunt based project that provides bootstrapping, scaffolding, some automated pre-processing faculty, as well as auto-reloading (no interaction needed).

Bootstrap Your Chrome Extension from Websecurify

Share Edit Follow

answered Oct 7, 2013 at 23:30
ProDataLab
41 ● 1

1

I want to reload (update) my extensions overnight, this is what I use in background.js:

```
var d = new Date();
var n = d.getHours();
var untilnight = (n == 0) ? 24*3600000 : (24-n)*3600000;
// refresh after 24 hours if hour = 0 else
// refresh after 24-n hours (that will always be somewhere between 0 and 1 AM)
setTimeout(function() {
    location.reload();
}, untilnight);
```

Regards, Peter

Share Edit Follow

answered Jul 29, 2016 at 22:17
Peter Driessen
139 ● 1 ● 9

1

I primarily develop in Firefox, where `web-ext run` automatically reloads the extension after files change. Then once it's ready, I do a final round of testing in Chrome to make sure there aren't any issues that didn't show up in Firefox.

If you want to develop primarily in Chrome, though, and don't want to install any 3rd party extensions, then another option is to create a `test.html` file in the extension's folder, and add a bunch of SSCCE's to it. That file then uses a normal `<script>` tag to inject the extension script.

You could use that for 95% of testing, and then manually reload the extension when you want to test it on live sites.

That doesn't identically reproduce the environment that an extension runs in, but it's good enough for many simple things.

Share Edit Follow   edited Jun 24, 2018 at 6:12   answered Jun 23, 2018 at 18:15
Ian Dunn
3,650 ● 6 ● 29 ● 44

1 You are now able to run in chromium also: "web-ext run -t chromium" – Johan Hoeksma Jun 3, 2021 at 8:21

1

MAC ONLY

Using Extensions Reloader:

Using Typescript

1. Add the watcher of your to your project: `yarn add tsc-watch`
2. Add command to `scripts` to `package.json`

```
...
  "scripts": {
    "dev": "tsc-watch --onSuccess \"open -a '/Applications/Google Chrome.app'
'http://reload.extensions'\""
  },
...
```

3. Run script `yarn dev`

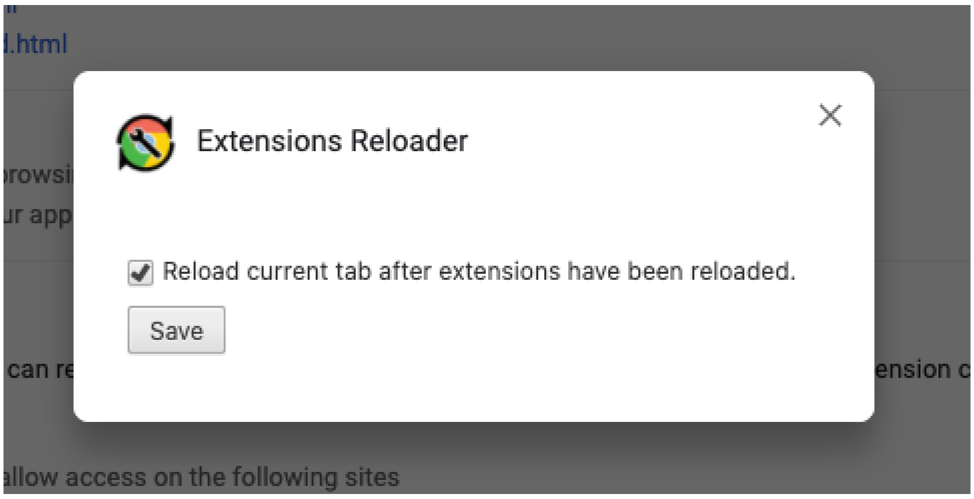
Using JavaScript

1. Add the watcher of your to your project: `yarn add watch-cli`
2. Add command to `scripts` to `package.json`

```
...
  "scripts": {
    "dev": "watch -p \"**/*.js\" -c \"open -a '/Applications/Google Chrome.app' 'http://reload.extensions'\""
  },
...
```

3. Run script `yarn dev`

Bonus: Turn on 'reload current tab' in Extensions Reloader options, so it reloads after a change was made:



☑ Reload current tab after extensions have been reloaded.

Share Edit Follow

edited Feb 19, 2022 at 16:27

answered Feb 12, 2022 at 12:35
e382df99a79509197897 25ceeec126
5,881 ● 4 ● 33 ● 56

1

You can reload the extension by calling `chrome.runtime.reload()`. If you want to reload the extension automatically on each build, you can set up a script to poll for changes.

Here's an example with a package.json file:

```
// package.json
// public is the folder containing the unpacked extension
{
  "name": "package-name",
  "type": "module",
  "scripts": {
    "build": "tsc && vite build",
    "build-and-reload": "echo $(date +%s) > public/.timestamp && npm run build"
  }
}
```

```
// background.ts


import { log } from "./log";

export const initAutoUpdater = () => {
  log("initAutoUpdater");
  const loop = async () => {
    const currentTS = await fetch("./.timestamp").then((_) => _.json());

    while (true) {
      log("checking for updates...");
      const newTS = await fetch("./.timestamp")
        .then((_) => _.json())
        .catch(() => "unknown");
      if (newTS === "unknown" || newTS === currentTS) {
        await new Promise((resolve) => setTimeout(resolve, 1000));
      } else {
        log(`new update found! ${new Date(currentTS)} -> ${new Date(newTS)}`);
        chrome.runtime.reload();
        return;
      }
    }
  };

  loop();
};
```

Here's how I use it:

1. I have a build task mapped to CMD+shift+B in VSCode
2. on each build we re-generate the .timestamp file with the current Unix timestamp
3. background.js polls for that file and compares it with the version it accessed on first load (`req.json()` works as `Number` is a valid JSON value)
4. if the timestamp is different, we reload the runtime

Share Edit Follow

answered May 1, 2023 at 15:41

Rafal Pastuszak
3,150 ● 2 ● 30 ● 31

1

If you are using Vite, here is a simple Vite plugin to reload the chrome extension when developing for mv3.

Usage

```
$ npm i hot-reload-extension-vite -D
```

```
import hotReloadExtension from 'hot-reload-extension-vite';

export default {
  plugins: [
    hotReloadExtension({
      log: true,
      backgroundPath: 'path/to/background' // src/pages/background/index.ts
    })
  ]
};
```

Then run

```
$ NODE_ENV=development vite build --watch
```

Extension and your tab will reload when vite detects file change.

Share Edit Follow

edited Oct 28, 2023 at 10:45

answered Sep 24, 2023 at 14:07
saurssaurav
753 ●1 ●7 ●18

0

I've forked LiveJS to allow for live reloading of Packaged Apps. Just include the file in your app and every time you save a file the app will autoreload.

Share Edit Follow

answered Dec 8, 2013 at 2:27
Oz Ramos
242 ●3 ●11

0

As mentioned in the docs: the following command line will reload an app

```
/Applications/Google\ Chrome.app/Contents/MacOS/Google\ Chrome --load-and-launch-app=
[path to the app ]
```

so I just created a shell script and called that file from gulp. Super simple:

```
var exec = require('child_process').exec;



gulp.task('reload-chrome-build',function(cb){

console.log("reload");

var cmd="./reloadchrome.sh"

exec(cmd,function (err, stdout, stderr) {
    console.log("done: "+stdout);

    cb(err);
    }
);});
```

run your necessary watch commands on scripts and call the reload task when you want to. Clean, simple.

Share Edit Follow

answered Oct 14, 2015 at 3:31
yeahdixon
6,778 ●1 ●41 ●43

0

This is where software such as AutoIt or alternatives shine. The key is writing a script which emulates your current testing phase. Get used to using at least one of them as many technologies do not come with clear workflow/testing paths.

```
Run("c:\Program Files (x86)\Google\Chrome\Application\chrome.exe")
WinWaitActive("New Tab - Google Chrome")
Send("^l")
Send("chrome://extensions{ENTER}")
WinWaitActive("Extensions - Google Chrome")
Send("{TAB}{TAB}{TAB}{TAB}{TAB}{TAB}")
Send("{ENTER}")
WinWaitActive("Extensions - Google Chrome")
Send("{TAB}{TAB}")
Send("{ENTER}")
WinWaitActive("Developer Tools")
Send("^`")
```

Obviously you change the code to suit your testing/iterating needs. Make sure tab clicks are true to where the anchor tag is in the chrome://extensions site. You could also use relative to window mouse movements and other such macros.

I would add the script to Vim in a way similar to this:

```
map <leader>A :w<CR>:!{input autoit loader exe here} "{input script location here}"<CR>
```

This means that when I'm in Vim I press the button above ENTER (usually responsible for: | and \) known as the leader button and follow it with a capital 'A' and it saves and begins my testing phase script.

Please make sure to fill in the {input...} sections in the above Vim/hotkey script appropriately.

Many editors will allow you to do something similar with hotkeys.

Alternatives to AutoIt can be found here.

For Windows: AutoHotkey

For Linux: xdotool, xbindkeys

For Mac: Automator

Share Edit Follow

answered Jan 23, 2016 at 12:22

maelwyn
11 ● 2

0

If you have a Mac, ¡the easiest way is with Alfred App!

Just get *Alfred App with Powerpack*, then add the workflow provided in the link below and customise the hotkey you want (I like to use ⌘ + ⌥ + R). That's all.

Now, every time you use the hotkey, *Google Chrome* will reload, no matter which application you're at that moment.

If you want to use other browser, open the AppleScript inside Alfred Preferences Workflows and replace "Google Chrome" with "Firefox", "Safari", ...

I also will show here the content of the **/usr/bin/osascript** script used in the *ReloadChrome.alfredworkflow file* so you can see what it is doing.

```
tell application "Google Chrome"
  activate
  delay 0.5
  tell application "System Events" to keystroke "r" using command down
  delay 0.5
  tell application "System Events" to keystroke tab using command down
end tell
```

The workflow file is ReloadChrome.alfredworkflow.

Share Edit Follow

answered Feb 13, 2016 at 1:04
Is Ma
934 ● 11 ● 14

0

The author recommended the next version of that webpack plugin: https://github.com/rubenspgcavalcante/webpack-extension-reloader. It works very well for me.

Share Edit Follow

answered May 14, 2019 at 15:44
Hieu Pham
359 ● 1 ● 3 ● 12

-1

Yes,you can do it indirectly! Here is my solution.

In manifest.json

```
{
    "name": "",
    "version": "1.0.0",
    "description": "",
    "content_scripts":[{
        "run_at":"document_end",
        "matches":["http://*/*"],
        "js":["/scripts/inject.js"]
    }]
}
```

In inject.js

```
(function() {
    var script = document.createElement('script');
    script.type = 'text/javascript';
    script.async = true;
    script.src = 'Your_Scripts';
    var s = document.getElementsByTagName('script')[0];
    s.parentNode.insertBefore(script, s);
})();
```

Your injected script can inject other script from any location.

Another benefit from this technic is that you can just ignore the limitation of **isolated world**. see content script execution environment

Share Edit Follow

edited Sep 7, 2015 at 21:56
Stephen Cagle
14.3k ● 16 ● 57 ● 86

answered Nov 27, 2011 at 13:56
wukong
2,519 ● 2 ● 26 ● 34

-2

BROWSER-SYNC

Using the amazing Browser-Sync

- update browsers (any) when source code changes (HTML, CSS, images, etc.)
- support Windows, MacOS and Linux
- you could even watch your code updates (live) using your mobile devices

Instalation on MacOS (view their help to install on other OS)

Install NVM, so you can try any Node version

```
brew install nvm             # install a Node version manager
nvm ls-remote                # list available Node versions
nvm install v10.13.0         # install one of them
npm install -g browser-sync  # install Browser-Sync
```

How to use browser-sync for static sites

Let's see two examples:

```
browser-sync start --server --files . --host YOUR_IP_HERE --port 9000

browser-sync start --server --files $(ack --type-add=web:ext:htm,html,xhtml,js,css --
web -f | tr \\n \ ) --host $(ipconfig getifaddr en0) --port 9000
```

The `--server` option allow you to run a local server anywhere you are in your terminal and `--files` let you specify which files will be tracked for changes. I prefer to be more specific for the tracked files, so in the second example I use `ack` for listing specific file extensions (is important that those files do not have filenames with spaces) and also use `ipconfig` to find my current IP on MacOS.

How to use browser-sync for dynamic sites

In case you are using PHP, Rails, etc., you already have a running server, but it doesn't refresh automatically when you make changes to your code. So you need to use the `--proxy` switch to let browser-sync know where is the host for that server.

```
browser-sync start --files $(ack --type-add=rails:ext:rb,erb,js,css,sass,scss,coffee --
rails -f | tr \\n \ ) --proxy 192.168.33.12:3000 --host $(ipconfig getifaddr en0) --
port 9000 --no-notify --no-open
```

In the above example, I already have a Rails app running on my browser on `192.168.33.12:3000`. It really runs on a VM using a Vagrant box, but I could access the virtual machine using port 3000 on that host. I like `--no-notify` to stop browser-sync sending me a notification alert on the browser every time I change my code and `--no-open` to stop browser-sync behavior that immediately loads a browser tab when the server start.

IMPORTANT: Just in case you're using Rails, avoid using Turbolinks on development, otherwise you will not be able click on your links while using the `--proxy` option.

Hope it would be useful to someone. I've tried many tricks to refresh the browser (even an old post I've submitted on this StackOverflow question using AlfredApp time ago), but this is really the way to go; no more hacks, it just flows.

CREDIT: Start a local live reload web server with one command

Share Edit Follow

edited Nov 25, 2018 at 13:52

answered Nov 25, 2018 at 13:30

Is Ma
934 ● 11 ● 14