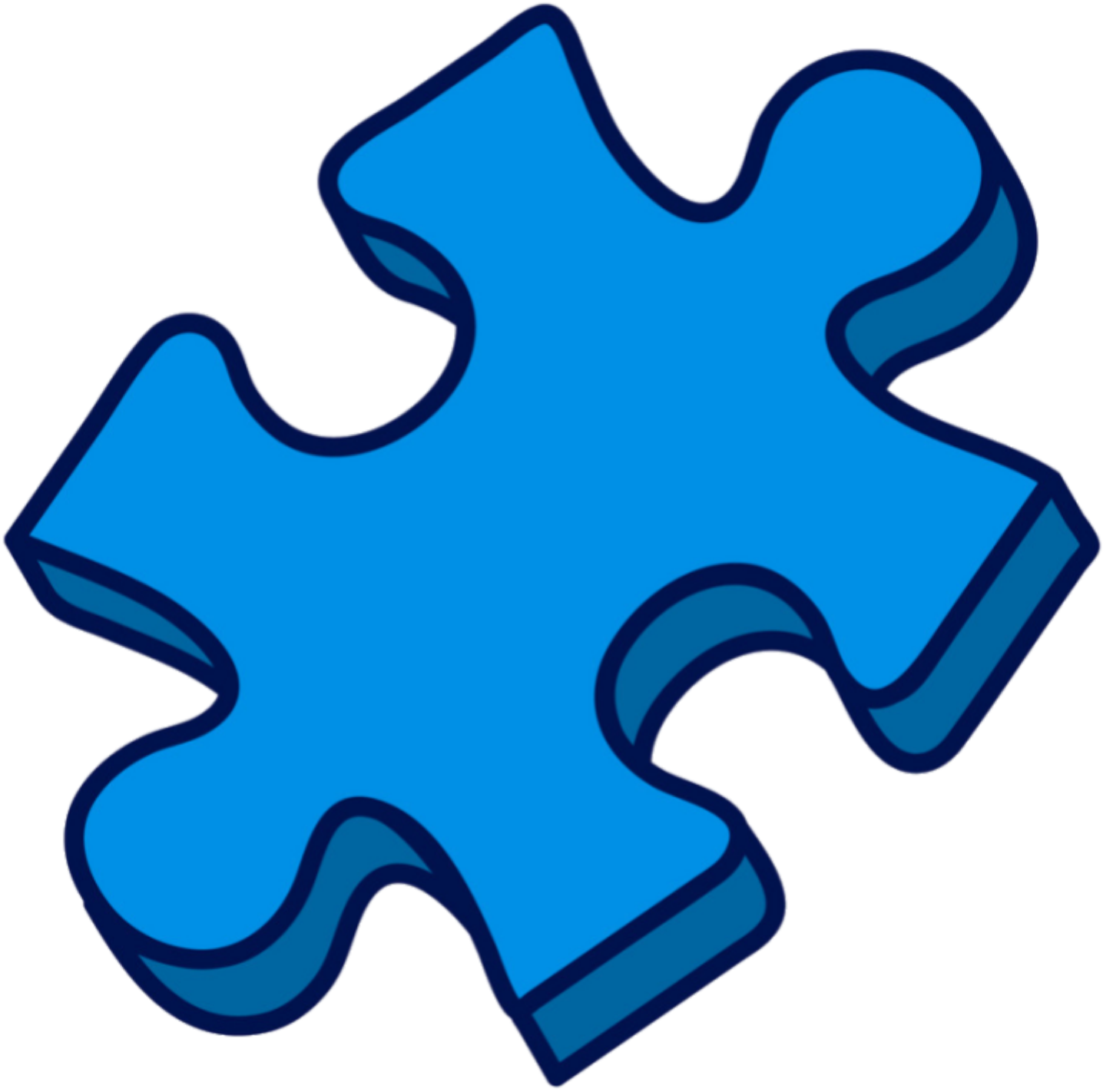
Extensions

MV3



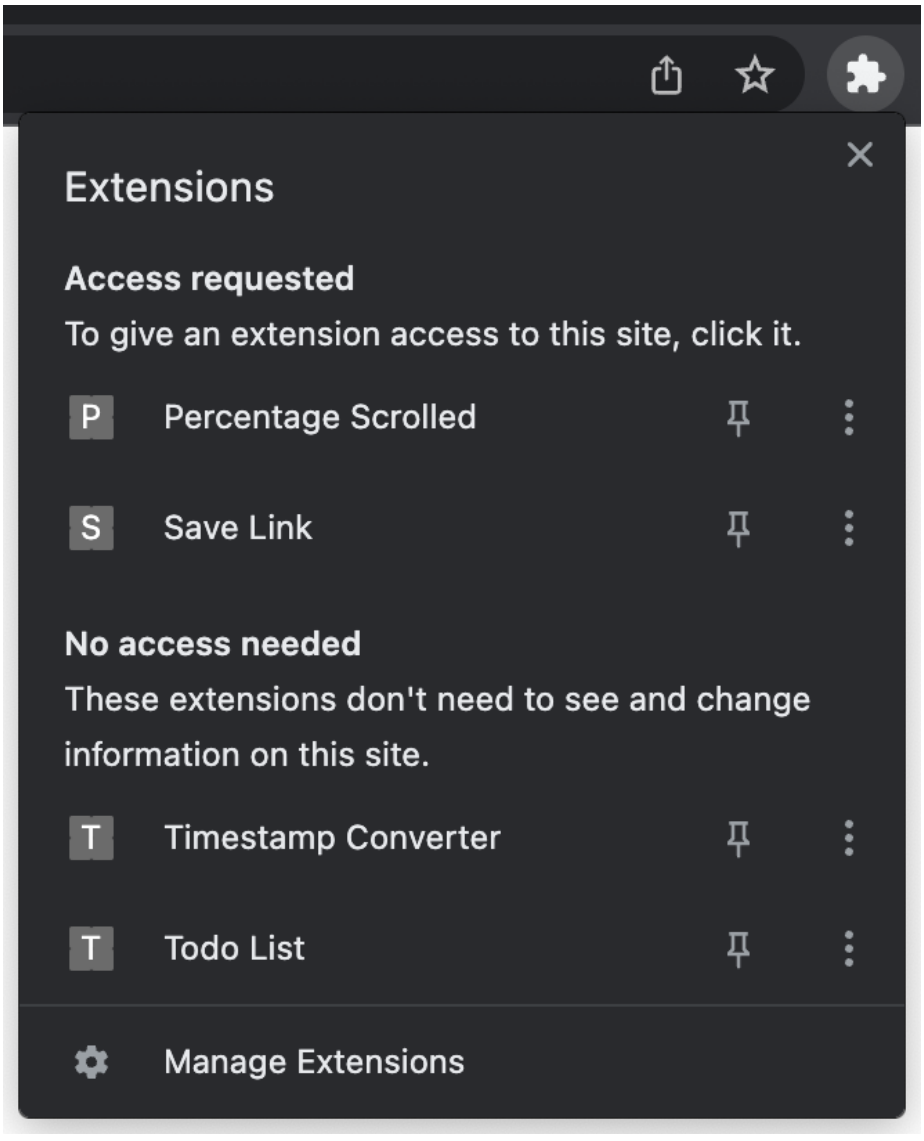
Extensions & Manifest V3

MV3 Chrome Extensions

For developers.

Introduction

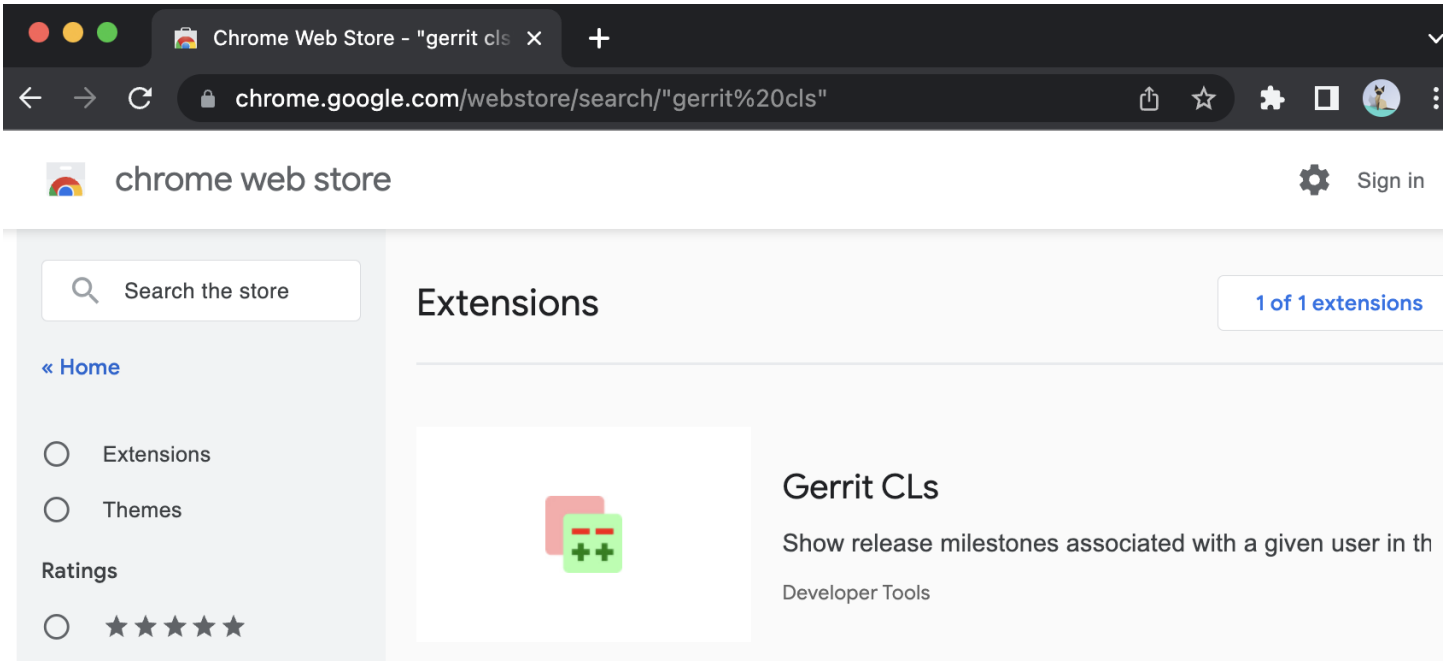
Extensions make it possible to customize your web browsing experience.



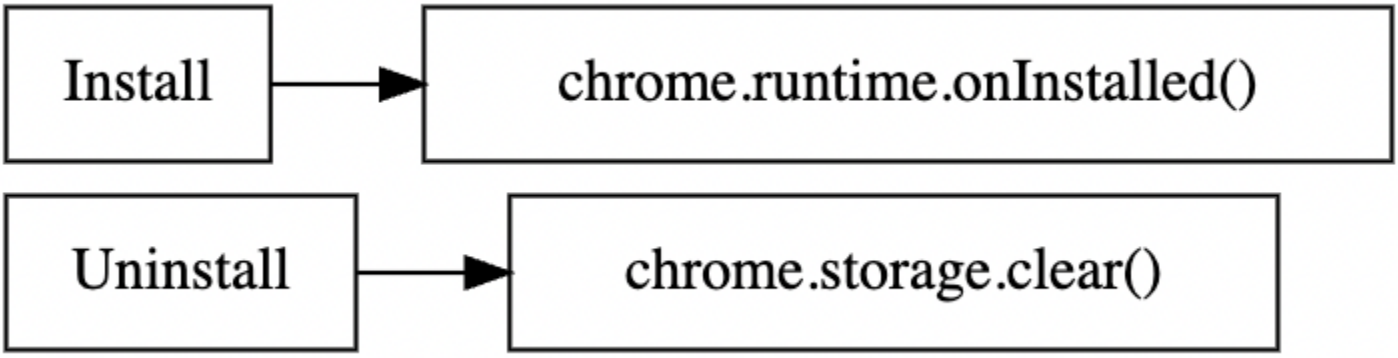
A new API available in MV3 (Manifest V3) is the chrome.tabGroups API.

Install

Chrome Web Store (https://chrome.google.com/webstore/category/extensions)

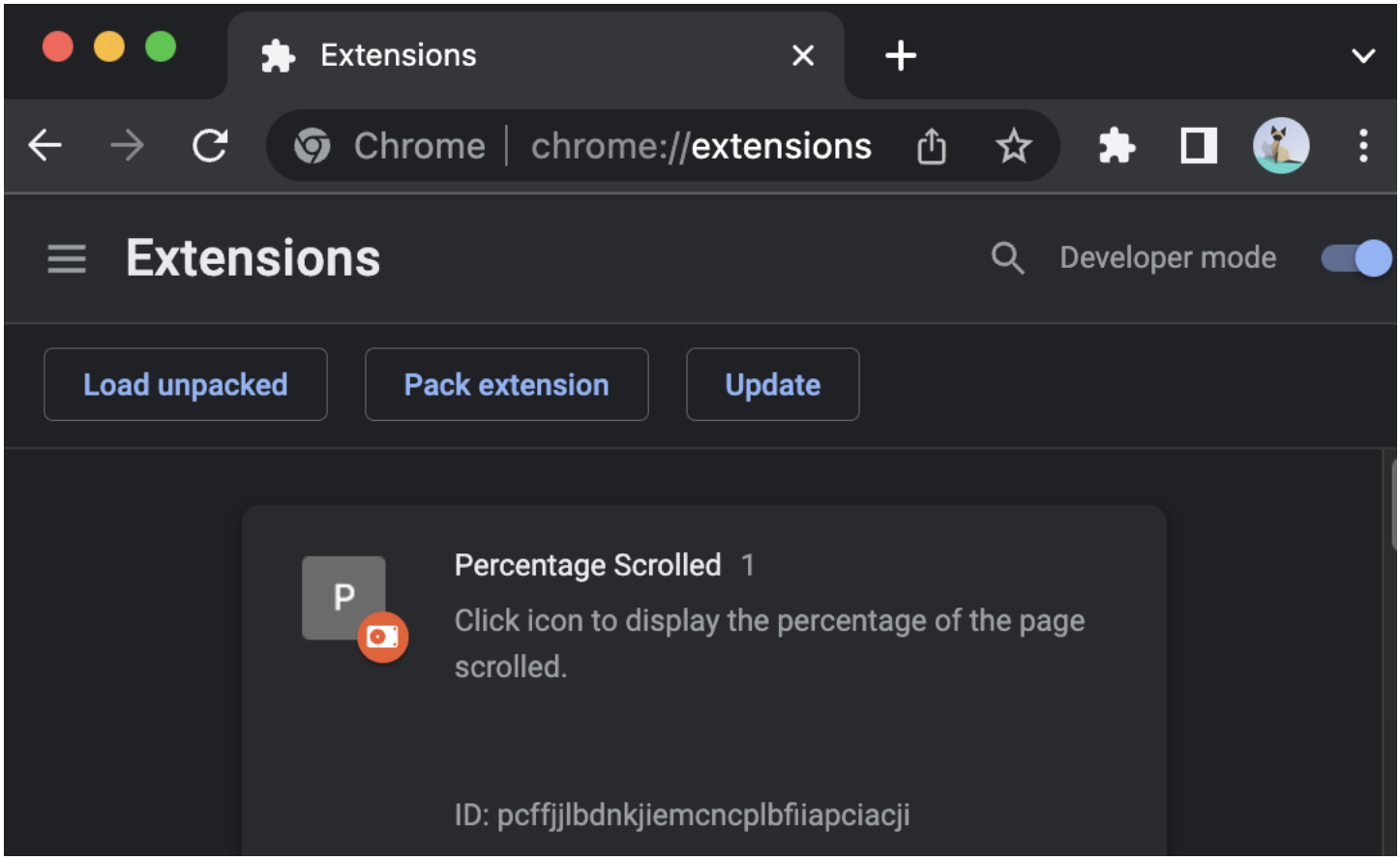


Storage for an extension is cleared when it's uninstalled. An extension can define an onInstalled event action.



Develop

Turn on Developer mode (chrome://extensions).



Create an empty directory. Create a manifest.json file. This is the only required file. Only three manifest keys are required: name, version, and manifest_version.

[manifest.json]

```
{
  "name": "Example Extension",
  "version": "1.0",
  "manifest_version": 3
}
```

Manifest

MV3 (Manifest V3) offers new features that didn't exist in MV2. The changes include manifest.json definitions, in addition to new APIs. For example, manifest.json now allows for more precise definitions for "web_accessible_resources". Resources used to be available to any site, but now developers can define which websites and extensions have access to them.

[manifest.json]

```
{
  "name": "Extension Name",
  "version": "1.0",
```

```
  "manifest_version": 3,
  "action": {"default_popup": "popup.html"},
  "author": "Your name",
  "background": {"service_worker": "worker.js"},
  "content_scripts": [{
    "matches": ["*://*/*"],
    "css": ["content.css"],
    "js": ["content.js"]
  }],
  "description": "Extension Description",
  "icons": {"128": "icon_128.png"},
  "permissions": ["storage", "tabs", "alarms"],
  "web_accessible_resources": [{
    "resources": "resource.html",
    "matches": "resource.html",
    "extension_ids": "resource.html"
  }]
}
```

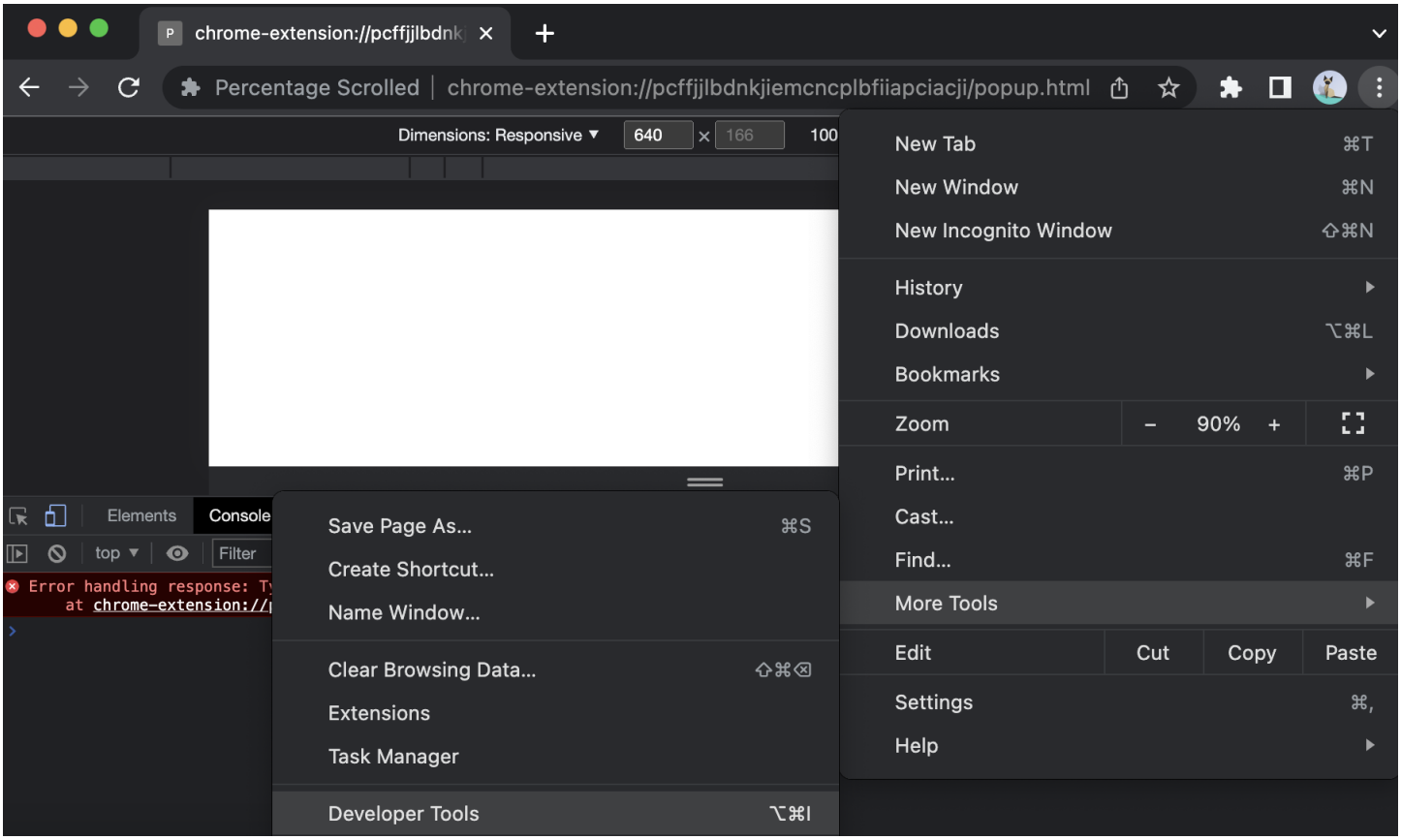
Common keys:

- `name`: (required) The name of the extension.
- `version`: (required) Up to three periods to define a custom version.
- `manifest_version`: (required) 3 is the current version.
- `action: default_poup`: Page to appear when icon is clicked.
- `background: service_worker`: Background script can listen for events and more.
- `content_scripts`: Inject custom css and javascript onto visited websites.
- `icons`: Extension icons of different sizes, specified by keys including 16, 128, and more.
- `web_accessible_resources`: resources accessible from websites.

Details of manifest.json (developer.chrome.com/docs/extensions/mv3/manifest)

Debug

Open Devtools to see popup.html output.



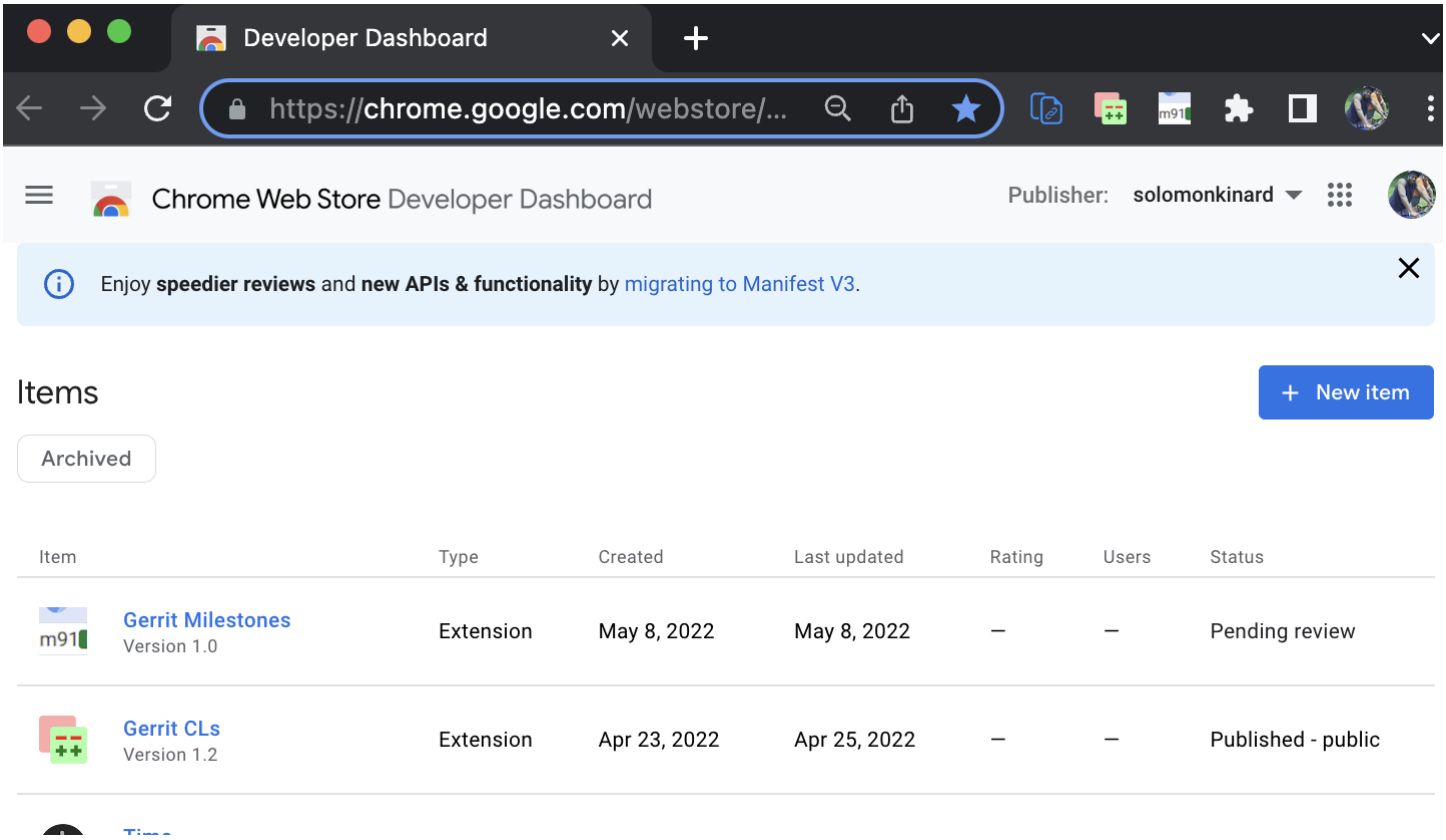
The base extension url is chrome-extension://<id>/. A common path is chrome-extension://<id>/manifest.json. Other common files at the top level are popup.html, worker.js, content.js, and icon.png.

Devtools for service workers are available from chrome://extensions. Devtools for popup.html can be accessed after right clicking the icon to inspect, or loading the page directly yourself using the base extension url.

A useful page for debugging is chrome://extensions-internals. Keyboard shortcuts can be viewed at chrome://extensions/shortcuts.

Publish

Developer Dashboard (https://chrome.google.com/webstore/devconsole)



Click “New Item” to begin. Upload a .zip file. The file must include manifest.json at the top level. The store listing must include a 128x128 .png icon and a 1280x800 .png screenshot. Complete “Privacy practices” next and be sure to include a detailed description as a “Single purpose”.

API

Usage: chrome.<API>.*
Example: chrome.search.query()

List:

accessibilityFeatures	action	alarms	audio
bookmarks	browsingData	certificateProvider	commands
contentSettings	contextMenus	cookies	debugger
declarativeContent	declarativeNetRequest	desktopCapture	devtools.*
documentScan	dom	downloads	enterprise.*
events	extension	extensionTypes	fileBrowserHandler
fileSystemProvider	fontSettings	gcm	history
i18n	identity	idle	input.ime

instanceID	loginState	management	notifications
omnibox	pageCapture	permissions	platformKeys
power	printerProvider	printing	printingMetrics
privacy	proxy	runtime	scripting
search	sessions	storage	system.*
tabCapture	tabGroups	tabs	topSites
tts*	types	virtualKeyboard	vpnProvider
wallpaper	webNavigation	webRequest	windows

Overview

There are several bite sized examples included here to learn from. There are samples available at github.com/GoogleChrome/chrome-extensions-samples. These concepts start small and progressively include more advanced features.

Each project aims to be as small as possible. These extensions are custom made. Each one is meant to solve a specific problem, rather than merely being an exercise. Feel free to copy and paste them into an extension of your own, in addition to making your own changes for your specific needs.

Extensions allow you to do things that would otherwise be almost impossible. It's quite common to want to:

1. Add or remove content directly to/from the current website.
2. Do something special when the extension icon is clicked.
3. Take specific action after a specific amount of time.

Examples

Tiny

manifest.json

```
{
 "name": "Tiny",
 "version": "1.0",
 "manifest_version": 3
}
```

Popup

manifest.json

```
{
 "name": "Popup",
 "version": "1.0",
 "manifest_version": 3,
 "action": {"default_popup": "popup.html"},
 "description": "Click the toolbar icon to see the popup."
}
```

popup.html

```
You clicked the toolbar icon.
```

New Tab

manifest.json

```
{
 "name": "New Tab",
 "version": "1.0",
 "manifest_version": 3,
 "background": {"service_worker": "worker.js"},
 "description": "Create a new tab on install"
}
```

worker.js

```
chrome.runtime.onInstalled.addListener(details => {
 chrome.tabs.create({url: "https://example.com"});
});
```

Next Tab

manifest.json

```
{
 "name": "Next Tab",
 "version": "0.0.1",
 "manifest_version": 3,
 "description": "Open new tab next to the current tab",
```

```
 "action": {},
 "background": {"service_worker": "service_worker.js"}
}
```

service_worker.js

```
chrome.action.onClicked.addListener(tab => {
 chrome.tabs.create({}).then(newTab =>
   chrome.tabs.move(newTab.id, {index: tab.index+1}))
});
```

Message

manifest.json

```
{
 "name": "Message",
 "version": "1.0",
 "manifest_version": 3,
 "action": {"default_popup": "popup.html"},
 "background": {"service_worker": "worker.js"},
 "description": "Click to message the content script."
}
```

popup.js

```
chrome.runtime.sendMessage('Hello', response => {
 document.body.append(response);
});
```

popup.html

```
<script src="popup.js"></script>
```

worker.js

```
chrome.runtime.onMessage.addListener((message, sender, sendResponse) => {
 sendResponse(`Service worker received "${message}" from "${sender.url}".`);
});
```

Alarm

manifest.json

```
{
 "name": "Alarm",
 "version": "1.0",
 "manifest_version": 3,
 "background": {"service_worker": "worker.js"},
 "description": "Create a new tab one minute after installation.",
 "permissions": ["alarms"]
}
```

worker.js

```
chrome.runtime.onInstalled.addListener(details => {
 chrome.alarms.create({delayInMinutes: 1});
});

chrome.alarms.onAlarm.addListener((alarm) => {
 chrome.tabs.create({});
});
```

content.js

```
chrome.runtime.onMessage.addListener((message, sender, sendResponse) => {
 sendResponse(`${sender}: ${message}`);
});
```

Common Manifest

manifest.json

```
{
 "name": "Common manifest",
 "version": "1.0",
 "manifest_version": 3,
 "action": {"default_popup": "popup.html"},
 "author": "Your name",
 "background": {"service_worker": "worker.js"},
 "content_scripts": [{
   "matches": ["*://*/*"],
   "css": ["content.css"],
   "js": ["content.js"]
 }],
 "description": "Extension Description",
 "icons": {"128": "icon_128.png"},
 "permissions": ["storage", "tabs", "alarms"],
 "web_accessible_resources": [{
```

```
   "resources": "resource.html",
   "matches": "resource.html",
   "extension_ids": "resource.html"
 }]
}
```

Chromium Search

manifest.json

```
{
 "name": "Chromium Search",
 "version": "1.0",
 "manifest_version": 3,
 "description": "Right click a word to search Chromium source code.",
 "permissions": ["contextMenus"],
 "background": {"service_worker": "worker.js"}
}
```

worker.js

```
const CONTEXT_MENU_ID = "chromium_quest";

function getword(info, tab) {
 if (info.menuItemId !== CONTEXT_MENU_ID) {
   return;
 }
 chrome.tabs.create({
   url:
`https://source.chromium.org/search?ss=chromium%2Fchromium%2Fsrc&q=${info.selec
tionText}`
 });
}

chrome.contextMenus.create({
 title: "Chromium: %s",
 contexts:["selection"],
 id: CONTEXT_MENU_ID
});

chrome.contextMenus.onClicked.addListener(getword);
```

Dark Mode

manifest.json

```
{
 "name": "Dark Mode",
 "version": "1.0",
 "manifest_version": 3,
 "content_scripts": [{
   "matches": ["*://*/*"],
   "css": ["content.css"],
   "js": ["content.js"]
 }]
}
```

content.css

```
#darkMode {
 position: absolute;
 top: 0.5em;
 left: 0.5em;
 text-decoration: none;
 z-index: 100000000000;
}

#darkMode:hover {
 cursor: pointer;
}
```

content.js

```
const a = document.createElement('a');
a.id = 'darkMode';
a.innerHTML = "&#x1F4A1;";
document.body.appendChild(a);
a.addEventListener('click', event => {
 event.preventDefault();
 const isDark = document.body.style.backgroundColor == 'black';
 document.body.style.backgroundColor = isDark ? 'white' : 'black';
 document.body.style.color = isDark ? 'white' : 'black';
});
```

Favicon

manifest.json

```
{
```

```
 "name": "Favicon - Content Script",
 "version": "1",
 "manifest_version": 3,
 "permissions": ["favicon"],
 "content_scripts": [{
   "matches": ["https://*/*"],
   "css": ["content.css"],
   "js": ["content.js"],
   "run_at": "document_idle"
 }],
 "web_accessible_resources": [{
   "resources": [ "_favicon/*" ],
   "matches": ["<all_urls>"],
   "extension_ids": [ "*" ]
 }],
 "action": {"default_popup": "popup.html"}
}
```

content.css

```
#img {
 position: absolute;
 top: 1em;
 left: 1em;
}
```

popup.html

```
<script src="content_script.js"></script>
<link rel="stylesheet" href="content_script.css">
```

content.js

```
window.onload = event => {
 const img = document.createElement('img');
 const url = location.href;
 img.id = 'img';
 img.src =
`chrome-extension://${chrome.runtime.id}/_favicon/?page_url=${url}&size=64`;
 console.log(img.src);
 document.body.appendChild(img);
}
```

Percent Scrolled

manifest.json

```
{
 "name": "Percent Scrolled",
 "version": "1.0",
 "manifest_version": 3,
 "description": "Click icon to display the percentage of the page scrolled.",
 "action": {"default_popup": "popup.html"},
 "permissions": ["activeTab", "scripting"],
 "host_permissions": ["<all_urls>"]
}
```

popup.js

```
chrome.tabs.query({active: true, currentWindow: true}, tabs => {
 if (tabs[0].url.startsWith("chrome://")) {
   window.close();
   return;
 }
 chrome.scripting.executeScript({
   target: { tabId: tabs[0].id },
   function: getScrollPercent
 });
});

function getScrollPercent() {
 var h = document.documentElement,
   b = document.body,
   st = 'scrollTop',
   sh = 'scrollHeight';
 var percent = (h[st]||b[st]) / ((h[sh]||b[sh]) - h.clientHeight) * 100;
 chrome.runtime.sendMessage({percent: percent.toFixed(1)});
}

chrome.runtime.onMessage.addListener(function(message, sender, sendResponse) {
 document.write(message.percent + '%');
 setTimeout(() => window.close(), 2000);
});
```

popup.html

```
<script src="popup.js"></script>
```

Gerrit Milestones

manifest.json

```
{
 "name": "Gerrit Milestones",
 "version": "1.0",
 "manifest_version": 3,
 "action": {"default_popup": "popup.html"},
 "description": "Click icon to see milestone from crrev.com",
 "permissions": ["activeTab", "storage"],
 "host_permissions": ["https://crrie.com/"]
}
```

popup.js

```
document.addEventListener("DOMContentLoaded", () => {
 chrome.storage.local.get('isDark')
   .then(result => result ? setBackground(result.isDark) : null);
 document.body.addEventListener("click", toggleBackground);
});

function toggleBackground() {
 const isDark = !(document.body.style.backgroundColor == "black");
 setBackground(isDark);
 chrome.storage.local.set({isDark: isDark});
}

function setBackground(isDark) {
 document.body.style.backgroundColor = isDark ? "black" : "white";
 document.body.style.color = isDark ? "white" : "black";
}

// Get the milestone for Chromium Gerrit revision id in tab url.
chrome.tabs.query({active: true, currentWindow: true}, tabs => {
 const div = document.createElement("div");
 document.body.appendChild(div);
 const url = tabs[0].url;
 const search =
"^https://chromium-review.googlesource.com/c/chromium/src/\\+/(\\d+)";
 const match = url.match(search);
 if (match == undefined || match.length != 2)
   return window.close();
 const revid = match[1];
 fetch(`https://crrie.com/c/?r=${revid}`)
   .then(res => res.text())
   .then(text => text != "" ? div.innerText = text : window.close());
});
```

popup.html

```
<script src="popup.js"></script>
```

Timestamp

manifest.json

```
{
 "name": "Timestamp Converter",
 "version": "1.0",
 "manifest_version": 3,
 "description": "Javascript timestamp conversion to human readable",
 "action": { "default_popup": "popup.html" }
}
```

popup.js

```
var input, output;

document.addEventListener('DOMContentLoaded', () => {
 input = document.getElementById("input");
 input.focus();
 input.addEventListener('input', change);
 output = document.getElementById("output");
});

function change() {
 output.innerText = timeConverter(input.value);
}

function timeConverter(UNIX_timestamp) {
 let ts = UNIX_timestamp.toString();
 if (ts.length == 10) ts += "000";
 ts = parseInt(ts);
 const a = new Date(ts);
 if (isNaN(a.getTime())) return "";
 const months =
   ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'];
 const year = a.getFullYear();
 const month = months[a.getMonth()];
 const date = a.getDate();
 const hour = pad(a.getHours());
 const min = pad(a.getMinutes());
 const sec = pad(a.getSeconds());
 return `${date} ${month} ${year} ${hour}:${min}:${sec}`;
}

function pad(value) {
```

```
 return value < 10 ? "0" + value.toString() : value;
}
```

popup.html

```
<script src="popup.js"></script>
<input id="input"></input><br>
<div id="output"></div>
```

Count Extensions

manifest.json

```
{
 "name": "Count Extensions",
 "version": "1.0",
 "manifest_version": 3,
 "description": "Show the number of installed extensions.",
 "action": {"default_popup": "popup.html"},
 "permissions": ["management"]
}
```

popup.js

```
document.addEventListener('DOMContentLoaded', setup, false);

function setup() {
 let sum = 0;
 let enabled = 0;
 let disabled = 0;
 let installType = {};
 chrome.management.getAll(info => {
   sum = info.length;
   info.forEach(item => {
     if (item.enabled) {
       enabled++;
     } else {
       disabled++;
     }

     installType[item.installType] = installType[item.installType] ?
installType[item.installType] + 1 : 1
   });
   let mainElement = document.getElementById("main");
   mainElement.innerHTML = `<div>Total: ${sum}</div>`;
```

```
   mainElement.innerHTML += '<p>Enabled<ul>';
   mainElement.innerHTML += `<li>Enabled: ${enabled}</li>`;
   mainElement.innerHTML += `<li>Disabled: ${disabled}</li></ul></p>`;

   mainElement.innerHTML += '<p>Install Type';
   for (const [key, value] of Object.entries(installType)) {
     mainElement.innerHTML += `<li><span
style="text-transform:capitalize;">${key}</span>: ${value}</li>`;
   }
   mainElement.innerHTML += '</ul></p>';
 });
}
```

popup.html

```
<html><head>
<script src="popup.js"></script>
</head>
<body>
<div id="main"></div>
</body>
```

Clipboard Service Worker

manifest.json

```
{
 "name": "Clipboard Service Worker",
 "version": "1.0",
 "manifest_version": 3,
 "description": "Copy to clipboard from a service worker",
 "background": {"service_worker": "worker.js"},
 "permissions": ["tabs", "clipboardWrite"],
 "action": {}
}
```

worker.js

```
// Save background tab id for message passing.
var tabId;

// Copy text to clipboard from the service worker context.
function copy(text) {
 // Send text to dom.html for copying.
 chrome.runtime.onMessage.addListener((request, sender, sendResponse) => {
   chrome.tabs.sendMessage(tabId, text, response => {});
```

```
 });

 // Create a background tab that has a DOM.
 chrome.tabs.create({url: `dom.html`, active: false}, tab => tabId = tab.id);
}

// Copy the current timestamp to clipboard when the action is clicked.
chrome.action.onClicked.addListener(() => copy(Date.now()));
```

dom.html

```
<script src="dom.js"></script>
```

dom.js

```
// DOM required for manipulation.
document.addEventListener("DOMContentLoaded", () => {
 chrome.runtime.sendMessage({event: "ready"});
});

// Copy message text to clipboard and then close the tab.
chrome.runtime.onMessage.addListener((request, sender, sendResponse) => {
 copyToClipboard(request);
 window.close();
});

// Copy text to clipboard.
function copyToClipboard(text) {
 const textarea = document.createElement("textarea");
 textarea.value = text;
 textarea.style.height = "0";
 textarea.style.overflow = "hidden";
 textarea.style.position = "fixed";
 document.body.appendChild(textarea);
 textarea.focus();
 textarea.select();
 document.execCommand("copy");
 document.body.removeChild(textarea);
}
```

Previous Tab

manifest.json

```
{
 "name": "Previous Tab",
```

```
 "version": "0.0.1",
 "manifest_version": 3,
 "description": "Go back to the previous tabs",
 "action": {},
 "background": {"service_worker": "service_worker.js"},
 "permissions": ["storage"]
}
```

service_worker.js

```
// Listeners.
chrome.tabs.onActivated.addListener(onTabActivated);
chrome.action.onClicked.addListener(onActionClicked);

// Clear storage when installed or when browser starts.
chrome.runtime.onStartup.addListener(init);
chrome.runtime.onInstalled.addListener(init);

// Push window/tab onto the stack w/max depth.
async function onTabActivated(activeInfo) {
 const data = await getData();
 if (!(activeInfo.windowId in data))
   data[activeInfo.windowId] = [];

   // Prevent duplicates here instead of in clicked, by removing leftmost.
 data[activeInfo.windowId] = data[activeInfo.windowId].filter(
   x => x != activeInfo.tabId);
 data[activeInfo.windowId].push(activeInfo.tabId);
 await chrome.storage.local.set({data: data});
}

// Get current window and active tab id.
async function getWindowIdAndTabId() {
 const windowId = (await chrome.windows.getLastFocused()).id;
 const query = {active: true, windowId: chrome.windows.WINDOW_ID_CURRENT};
 const tabId = (await chrome.tabs.query(query))[0].id;
 return [windowId, tabId];
}

// Pop from the stack until window/tab exists to activate.
async function onActionClicked(tab) {
 const data = await getData();
 let [currentWindowId, currentTabId] = await getWindowIdAndTabId();
 if (!(currentWindowId in data))
   return;

 // Remove the current tab from the end and prepend it.
 data[currentWindowId].pop();
```

```
 data[currentWindowId].unshift(currentTabId);

 // Lazily pop tabs from backstack.
 while (data[currentWindowId].length > 1) {
   const tabId = data[currentWindowId].pop();

   // Maybe make tabId active.
   try {
     // Set active. Result can be null if it's already active before update().
     await chrome.tabs.update(tabId, {active: true});
     break;
   } catch(e) {
     // The tab may have already been closed, so it's lazily ignored here.
   }
 }

 // Store data.
 await chrome.storage.local.set({data: data});
}

// Get data from storage.
async function getData() {
 let data = {}; // windowId: [tabId]
 const result = await chrome.storage.local.get(['data']);
 if ('data' in result)
   data = result.data;
 return data;
}

// Replace storage data with the current window and tab id.
async function init() {
 await chrome.storage.local.set({data: {}});
 const [windowId, tabId] = await getWindowIdAndTabId();
 onTabActivated({windowId: windowId, tabId: tabId});
}
```

Alarm Table

manifest.json

```
{
 "name": "Alarm Table",
 "version": "1",
 "manifest_version": 3,
 "action": {"default_popup": "popup.html"},
 "background": {"service_worker": "worker.js"},
```

```
 "description": "Save the time when alarms fire every minute.",
 "permissions": ["alarms", "storage"]
}
```

popup.js

```
window.addEventListener('DOMContentLoaded', (event) => {
 // Get everything from storage.
 chrome.storage.local.get(null).then(items => {
   // Header.
   const div = document.createElement("div");
   const table = document.createElement("table");
   table.style = "width: 500;";
   let rows = [["Name", "Scheduled", "Triggered", "Delta"]];

   // Define rows.
   Object.keys(items).forEach(name => {
     const timestamps = items[name];
     const scheduled = toDateString(timestamps[0]);
     let triggered = "";
     let delta = "";
     if (timestamps.length == 2) {
       triggered = toDateString(timestamps[1]);
       delta = `${subtract(timestamps[1], timestamps[0])} ms`;
     }
     rows.push([name, scheduled, triggered, delta]);
   });

   // Write rows.
   for (i=0; i<rows.length; i++) {
     const row = rows[i];
     const tr = document.createElement("tr");
     row.forEach(col => {
       const cell = document.createElement(i == 0 ? "th" : "td");
       cell.innerText = col;
       tr.appendChild(cell);
     });
     table.appendChild(tr);
   }

   // Footer.
   div.appendChild(table);
   document.body.appendChild(div);
 });
});

// Convert timestamp to human readable, localized time.
function toDateString(timestamp) {
```

```
 const locale = new Date(timestamp).
   toLocaleString('en-US', {timeZone: 'America/Los_Angeles'});
 return [locale, timestamp].join("\n");
}

// Get the difference rounded to the nearest integer.
function subtract(a, b) {
 return Math.round(a - b);
}
```

popup.html

```
<html>
<head>
<script src="popup.js"></script>
<link rel="stylesheet" href="popup.css">
</head>
 <body width="500"></body>
</html>
```

worker.js

```
chrome.runtime.onInstalled.addListener(details => {
 const maxMinutes = 10;
 for (let minutes=1; minutes<=maxMinutes; minutes++) {
   const name = `${minutes}`;
   chrome.alarms.create(name, {delayInMinutes: minutes});
   chrome.alarms.get(name).then(alarm => {
     chrome.storage.local.set({[alarm.name]: [alarm.scheduledTime]}, ()=>{});
   });
 }
});

chrome.alarms.onAlarm.addListener(alarm => {
 const date = Date.now();
 chrome.storage.local.get(alarm.name).then(items => {
   const item = items[alarm.name];
   item.push(date);
   chrome.storage.local.set({[alarm.name]: item}).then(()=>{});
 });
});
```

popup.css

```
body {
 width: 500;
}
```

Native Messaging

Only two files are required in an extension to make native messaging possible. Once manifest.json and service_worker.js exist, install the extension and copy the extension ID. Paste the <extension-id> in “allowed_origins” in com.example.app.json. Compile example.app.cc and save the binary as /tmp/example.app. Reload the extension and click the extension icon to see proof of message passing in the service worker console log.

manifest.json

```
{
 "name": "Native Messaging",
 "version": "0.0.1",
 "manifest_version": 3,
 "background": {"service_worker": "service_worker.js"},
 "permissions": ["nativeMessaging"],
 "action": {}
}
```

service_worker.js

```
// Click the extension icon in the menu to send and receive native messages.
chrome.action.onClicked.addListener(tab => {
 sendNativeMessage();
 connectNative();
});

// One time message must respond before the background goes away.
function sendNativeMessage() {
 chrome.runtime.sendNativeMessage('com.example.app',
   { text: "object required" },
   response => console.log("Received", response)
 );
}

// Connect to keepalive as long as necessary to get a response.
function connectNative() {
 const port = chrome.runtime.connectNative('com.example.app');
 port.postMessage("string allowed");
 port.onMessage.addListener(response => console.log("Received", response));
 port.onDisconnect.addListener(() => console.log(chrome.runtime.lastError));
}
```

/tmp/example.app.cc

```
#include <iostream>
#include <string>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv) {
 // Receive from Chrome the 32 bit message length of up to 4 gb.
 unsigned int length = 0;
 for (int index = 0; index < 4; index++) {
   unsigned int read_char = getchar();
   length = length | (read_char << index * 8);
 }

 // Receive a message from Chrome.
 std::string message = "";
 for (int index = 0; index < length; index++) {
   message += getchar();
 }

 // Set the 32 bit message length of up to 1 gb to send to Chrome.
 unsigned int len = message.length();
 printf("%c%c%c%c", (char) (len & 0xff),
                    (char) (len << 8 & 0xff),
                    (char) (len << 16 & 0xff),
                    (char) (len << 24 & 0xff));

 // Send the message.
 printf("%s", message.c_str());

 return 0;
}
```

~/Library/Application Support/Google/Chrome/NativeMessagingHosts/com.example.app.json

```
{
 "name": "com.example.app",
 "description": "Example App",
 "path": "/tmp/example.app",
 "type": "stdio",
 "allowed_origins": ["chrome-extension://<extension-id>/"]
}
```

Bugs

Use “Component: Platform>Extensions” to file a bug (crbug.com).

Platform

Platform features enable developers to extend the browser.

Overview (crsrc.org/extensions/docs/overview.md)

Code

Browse chromium/src (https://chromium.googlesource.com/chromium/src/)

Chromium Code Search (crsrc.org/c)

Checking out and building Chromium for Mac (https://chromium.googlesource.com/chromium/src/+/main/docs/mac_build_instructions.md)

Install XCode

```
$ ls `xcode-select -p`/Platforms/MacOSX.platform/Developer/SDKs
$ git clone https://chromium.googlesource.com/chromium/tools/depot_tools.git
$ export PATH="$PATH:/path/to/depot_tools"
$ mkdir chromium && cd chromium
$ caffeinate fetch chromium
$ gn args out/debug
```

[out/debug/args.gn]

```
is_debug = false
is_component_build = true
symbol_level = 0
```

```
$ autoninja -C out/Default chrome
$ out/Default/Chromium.app/Contents/MacOS/Chromium
```

Contributing

New API Proposal (crsrc.org/extensions/docs/new_api_proposal.md)

Writing a New Extension API (crsrc.org/extensions/docs/writing_a_new_api.md)

Contributing to Chromium (crsrc.org/docs/contributing.md)

Advanced

Extension Features

Enable

```
$ chrome debug --enable-extension-features=New
```

Check for feature enablement

```
if (base::FeatureList::IsEnabled(extensions_features::kNew) {}
```

[extensions/common/extension_features.h]

```
extern const base::Feature kNew;
```

[extensions/common/extension_features.cc]

```
const base::Feature kNew{"New", base::FEATURE_DISABLED_BY_DEFAULT};
```

[chrome/browser/extensions/api/new/new_apitest.cc]

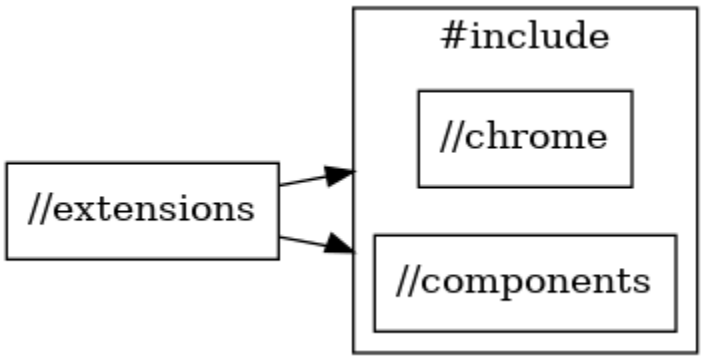
```
class NewApiTest : public ExtensionApiTest {
 public:
  NewApiTest() {
    feature_list_.InitAndEnableFeature(extensions_features::kNew);
  }

 private:
    base::test::ScopedFeatureList feature_list_;
}
```

Channels

The stable channel is the default. Canary is for the latest. Dev is in the middle.

Extensions #include rules



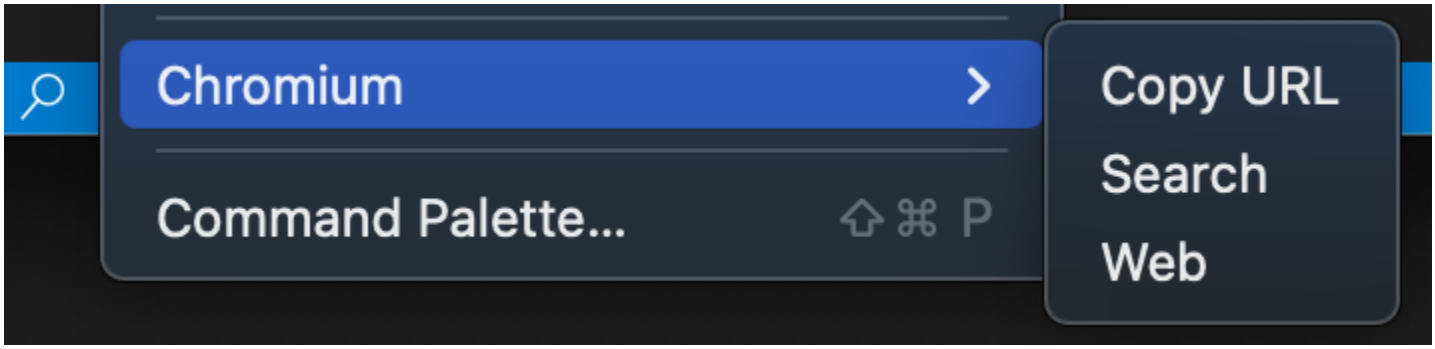
source -> #include

//extensions -> //components
Include //components in //extensions BUILD.gn and DEPS.gn

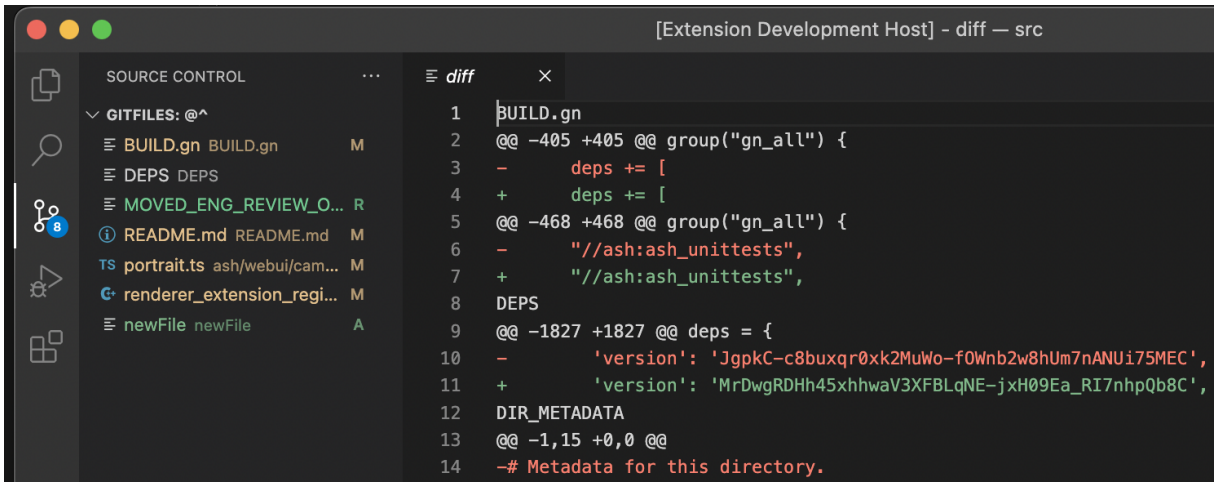
//extensions -> //chrome
Add new method to ExtensionsBrowserClient
Override it in ChromeExtensionsBrowserClient in //chrome

Chromium VSCode Extensions

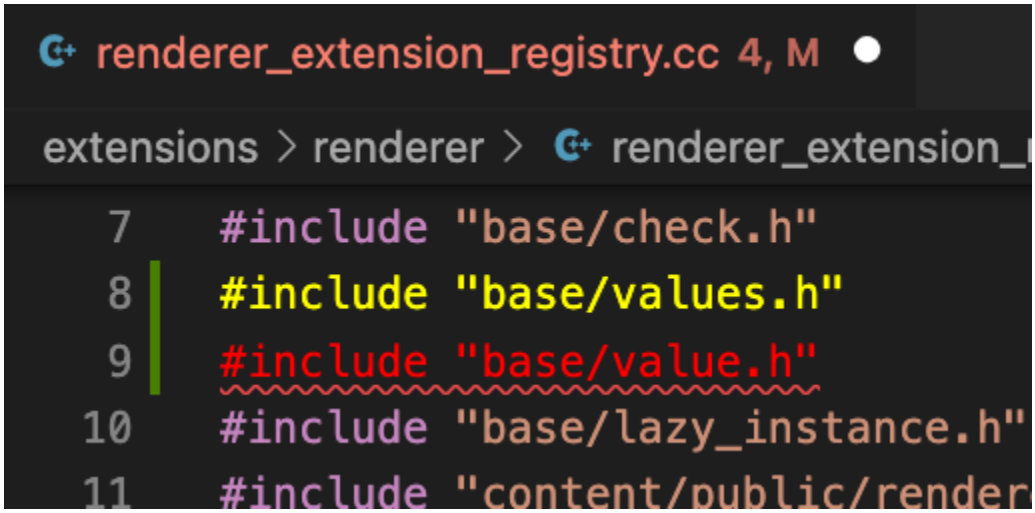
Chromium Context
https://marketplace.visualstudio.com/items?itemName=solomonkinard.chromium-context



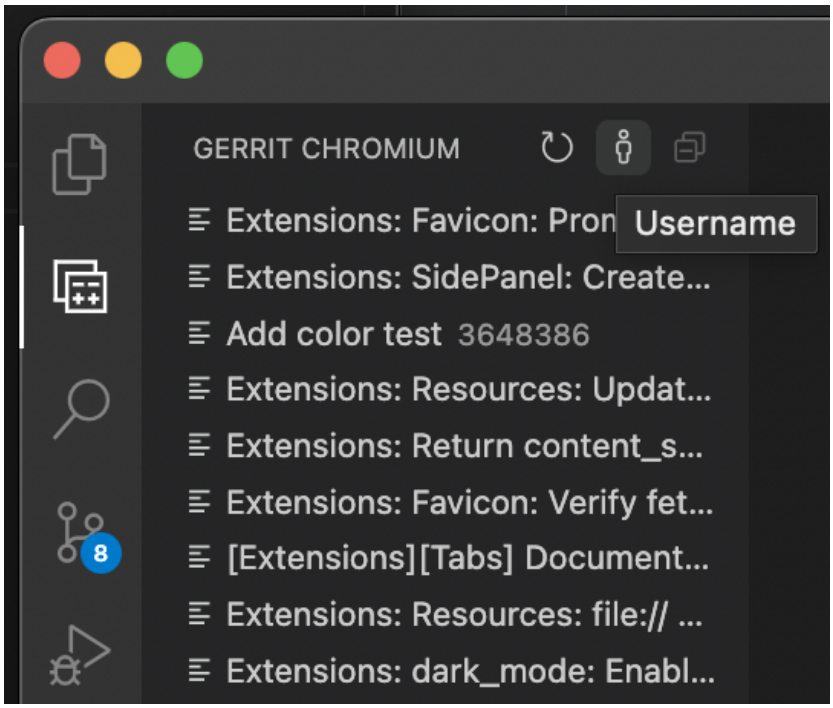
Git Files
Show files modified since the previous commit.
https://marketplace.visualstudio.com/items?itemName=solomonkinard.git-files



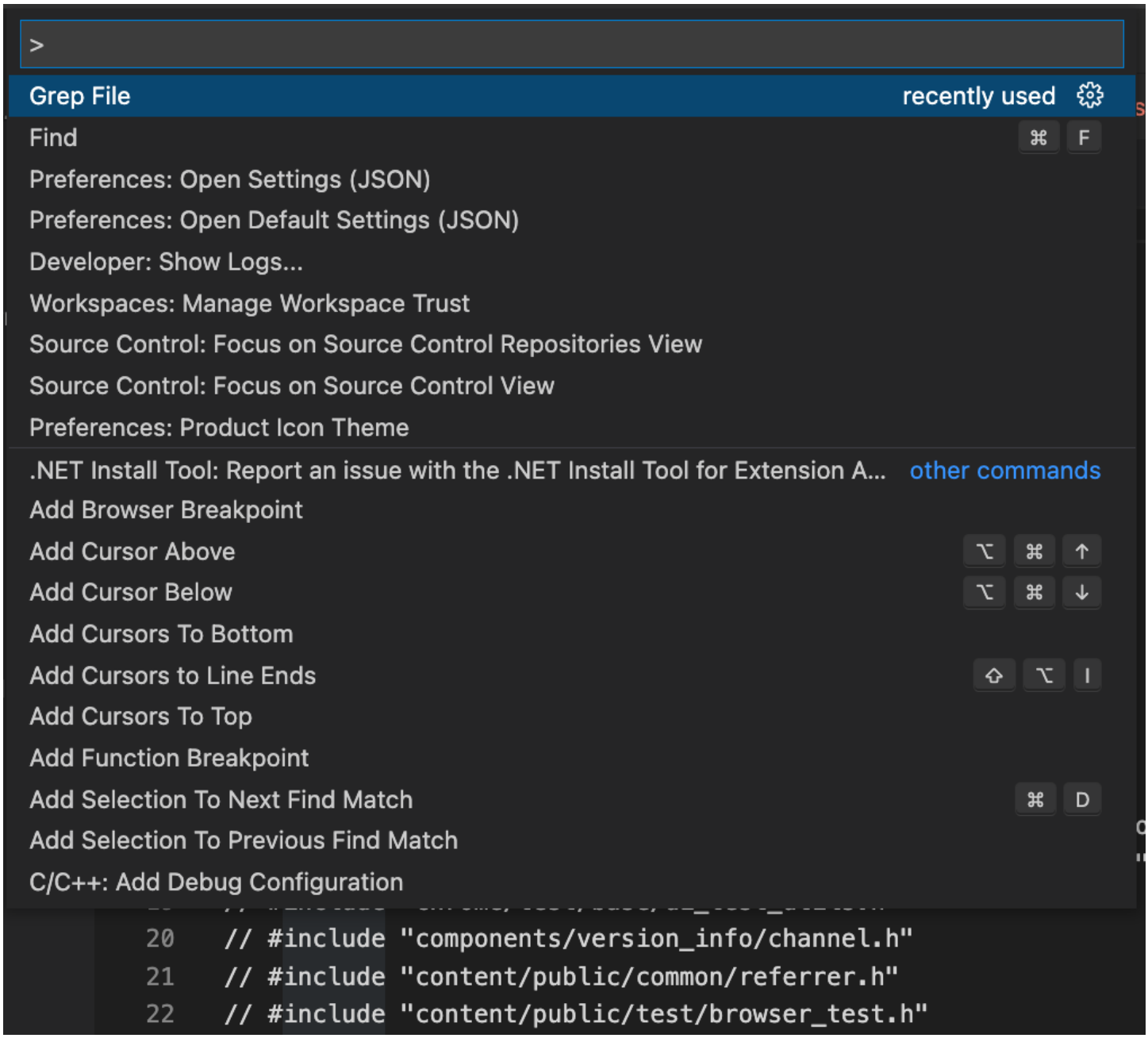
Workspace IWYU
https://marketplace.visualstudio.com/items?itemName=solomonkinard.workspace-include-what-you-use



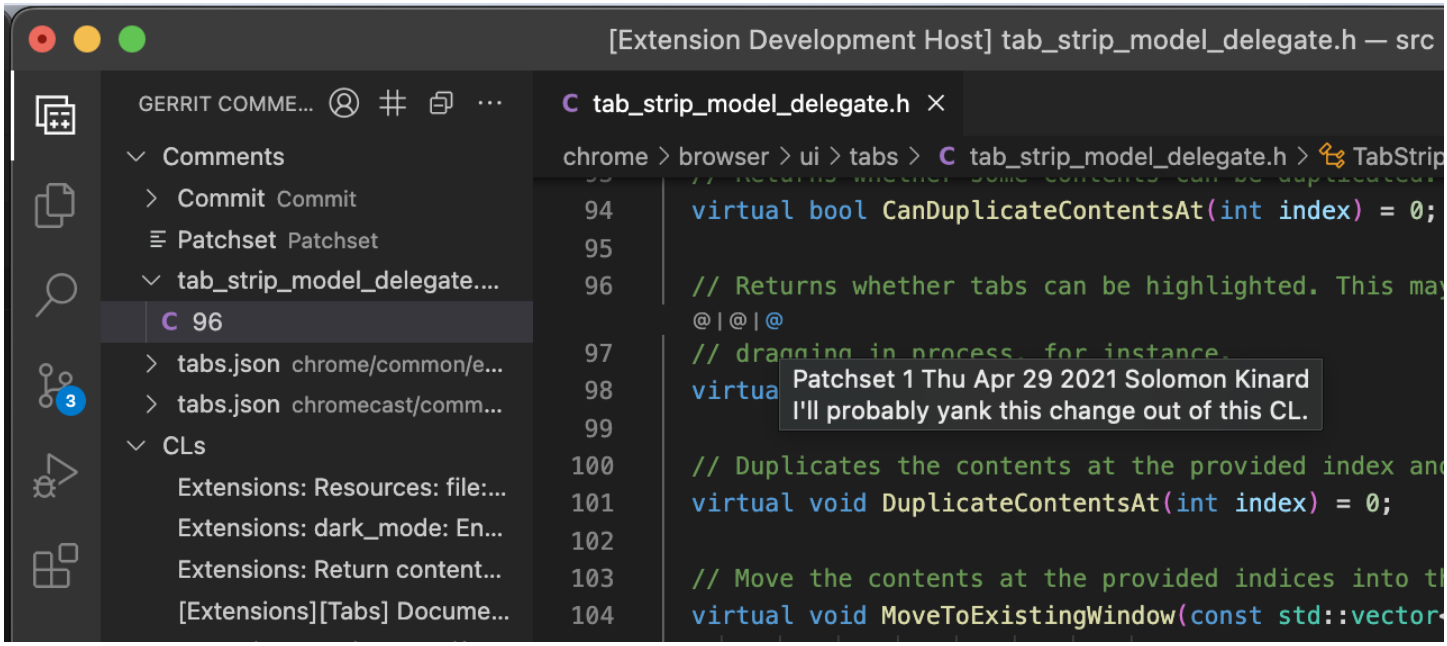
Chromium Gerrit
View your Gerrit CLs. Then click.
https://marketplace.visualstudio.com/items?itemName=solomonkinard.chromium-gerrit



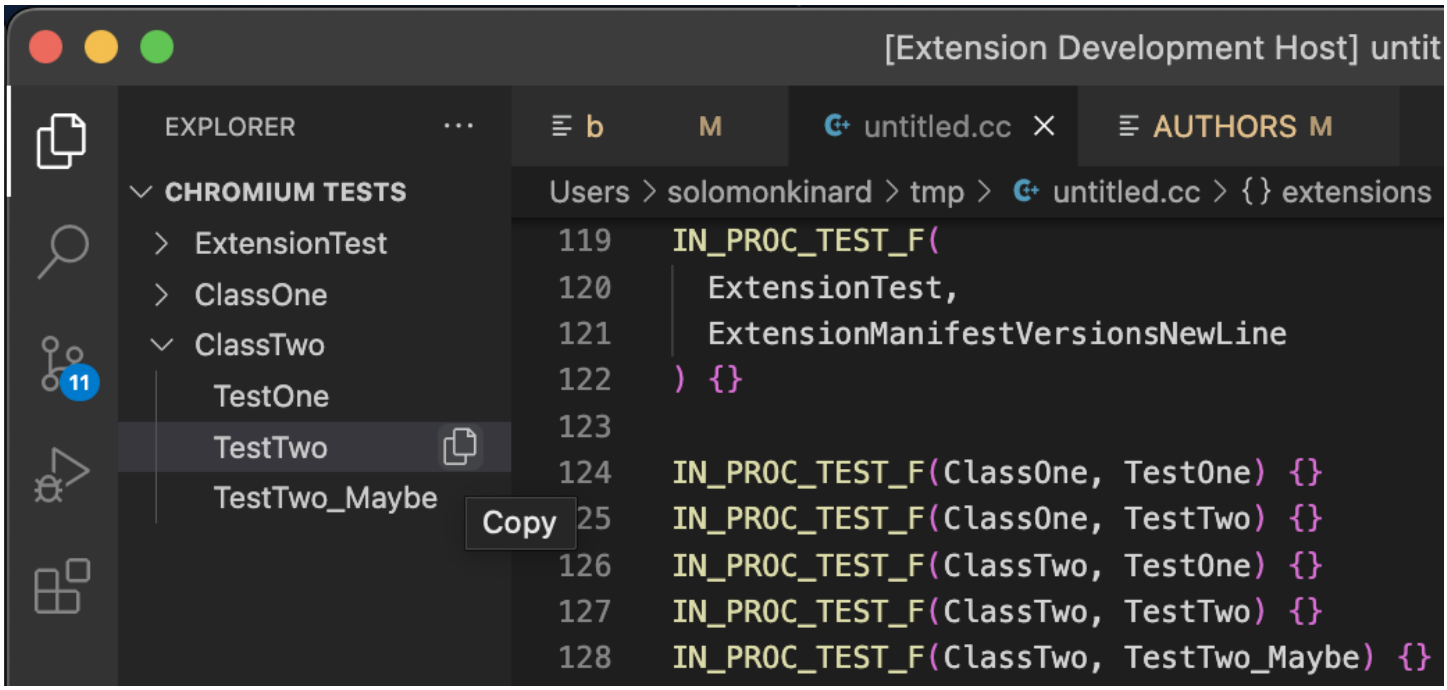
Grep File
https://marketplace.visualstudio.com/items?itemName=solomonkinard.grep-file



Gerrit Comments
https://marketplace.visualstudio.com/items?itemName=solomonkinard.gerrit-comments



Chromium Tests
https://marketplace.visualstudio.com/items?itemName=solomonkinard.chromium-tests

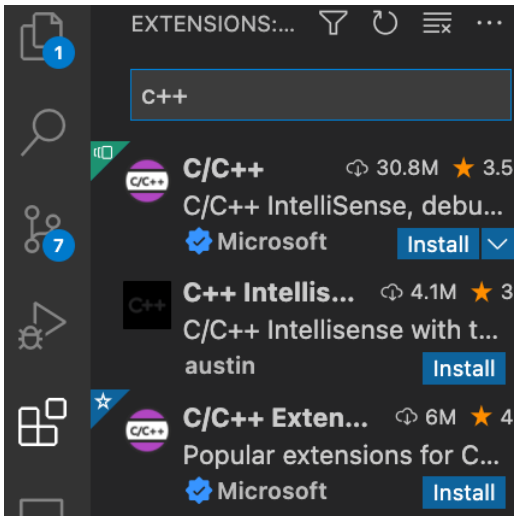


C++

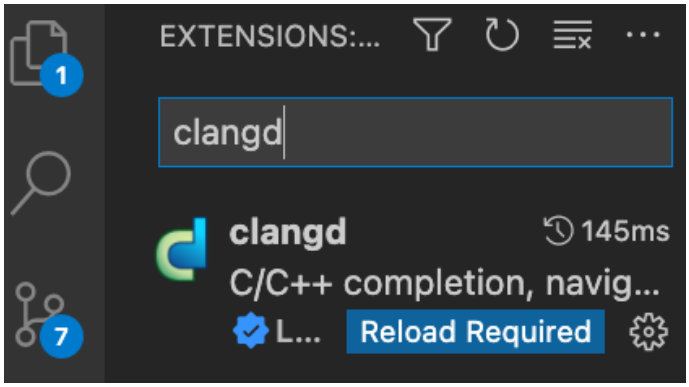
Summary
VSCode (Visual Studio Code) offers inline annotations, typeahead, and right-click to go to definition.

Prerequisites

Uninstall the Microsoft C/C++ extension



Install the clangd extension




```
$ cd ~/chromium/src
```

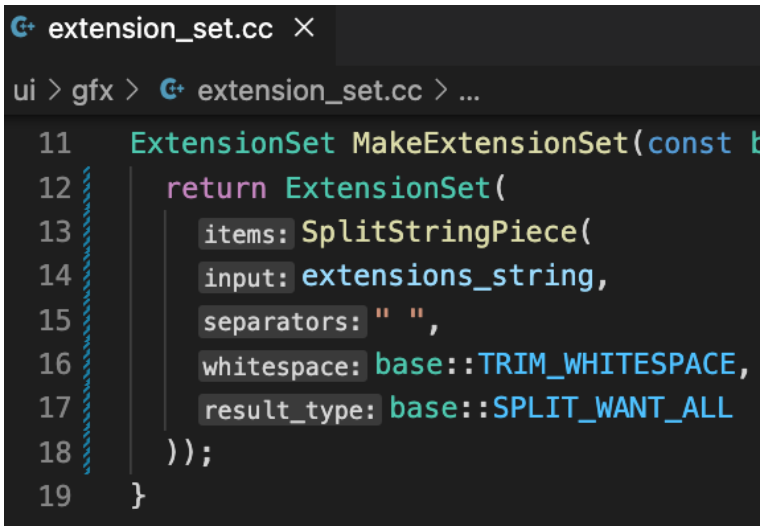
Run

```
$ python3 tools/clang/scripts/generate_compdb.py --target_os linux -p out/debug
-o compile_commands.json
```

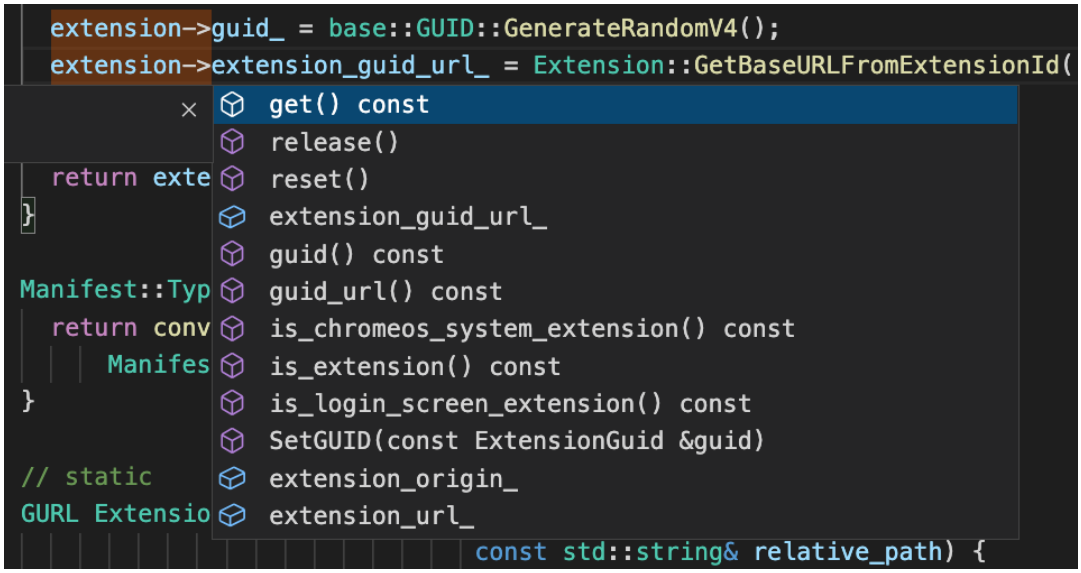
Restart VSCode

Benefits

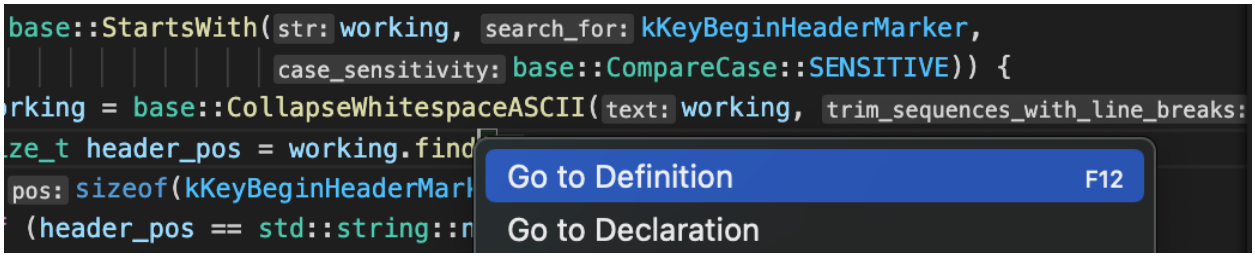
Inline annotations



Intellisense



Right click to go to definition



Milestones

Convert a Gerrit revision ID to a milestone by using the API at crrie.com.

Load the CL (changelist) at crrev.com/c/2930704.

Examples:

```
$ curl https://crrie.com/c/?r=2930704

90

$ curl https://crrie.com/c/ -d '{"r": "2930704,3607290"}' -H 'Content-Type:
application/json'

90,91
```

Definition:

GET One revision ID.
POST One or more revision IDs.

Parameter:

r = revision(s)

Aliases

[~/.gitconfig]

```
[alias]
  # Basic
  a = add
  c = commit
  co = checkout
  cp = cherry-pick
  d = diff
  s = status
  unstash = stash pop

  # Useful
  amend = commit --amend
  cob = checkout -b

  # Rename the current branch.
  # Example: $ git rename new
  rename = "!f() { git branch -m $1; }; f"
```

[~/.bashrc]

```
# Build and run Chrome.
# Example: $ chrome debug
alias chrome='f() { autoninja -C out/$1 chrome && out/$1/chrome; }; f'
```

Git

Git pull before starting work on a new branch

```
$ git rebase update --current
$ gclient sync -Df
```

Create a new branch

```
$ git cob new
```

Format CL

```
$ git cl format
```

Upload CL

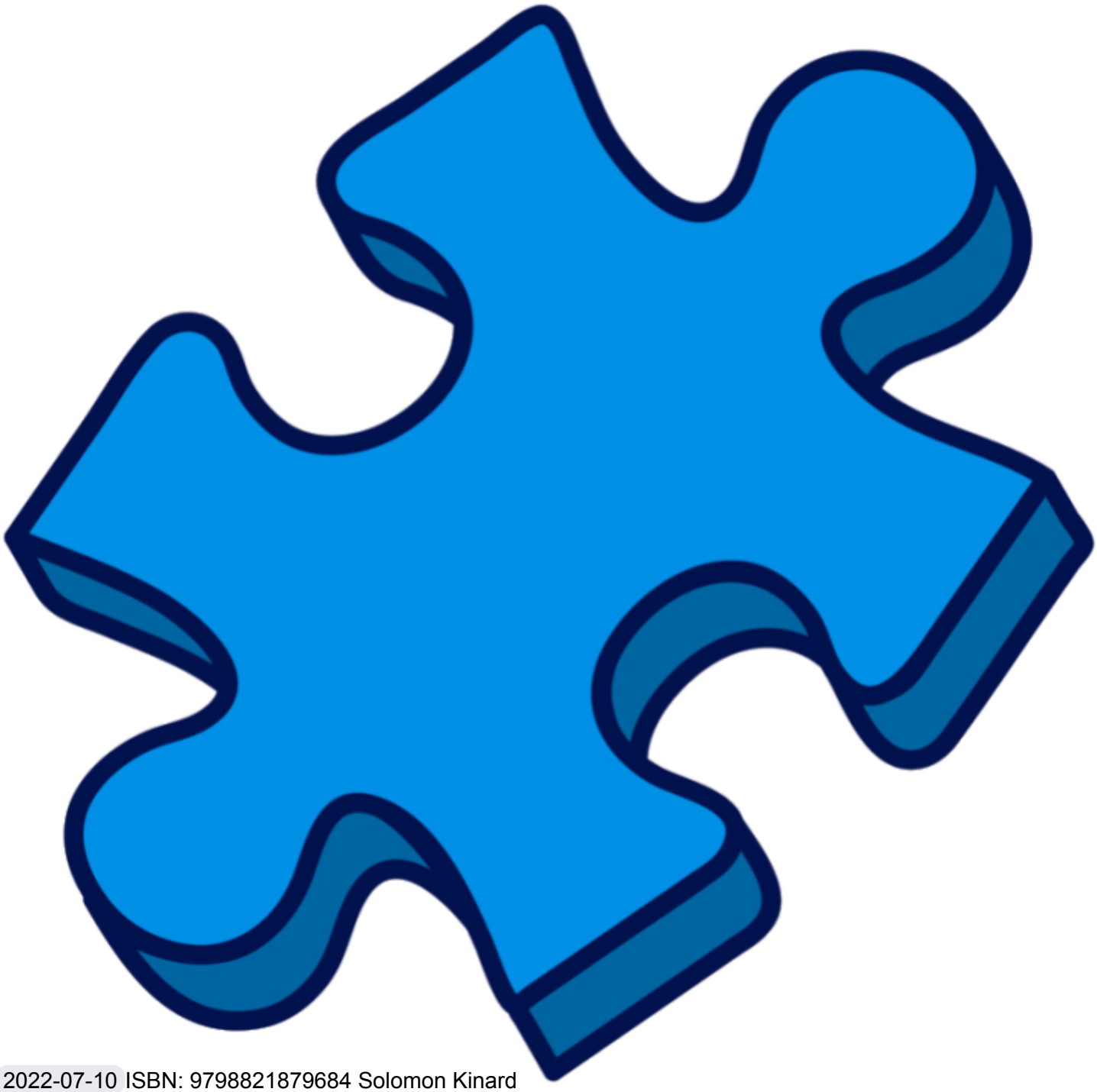
```
$ git cl upload
```

References

1. API Reference - Chrome Developers (https://developer.chrome.com/docs/extensions/reference/)
2. New Extension and Platform App APIs (https://chromium.googlesource.com/chromium/src/+/HEAD/extensions/docs/new_api_proposal.md)

Extensions

MV3



2022-07-10 ISBN: 9798821879684 Solomon Kinard