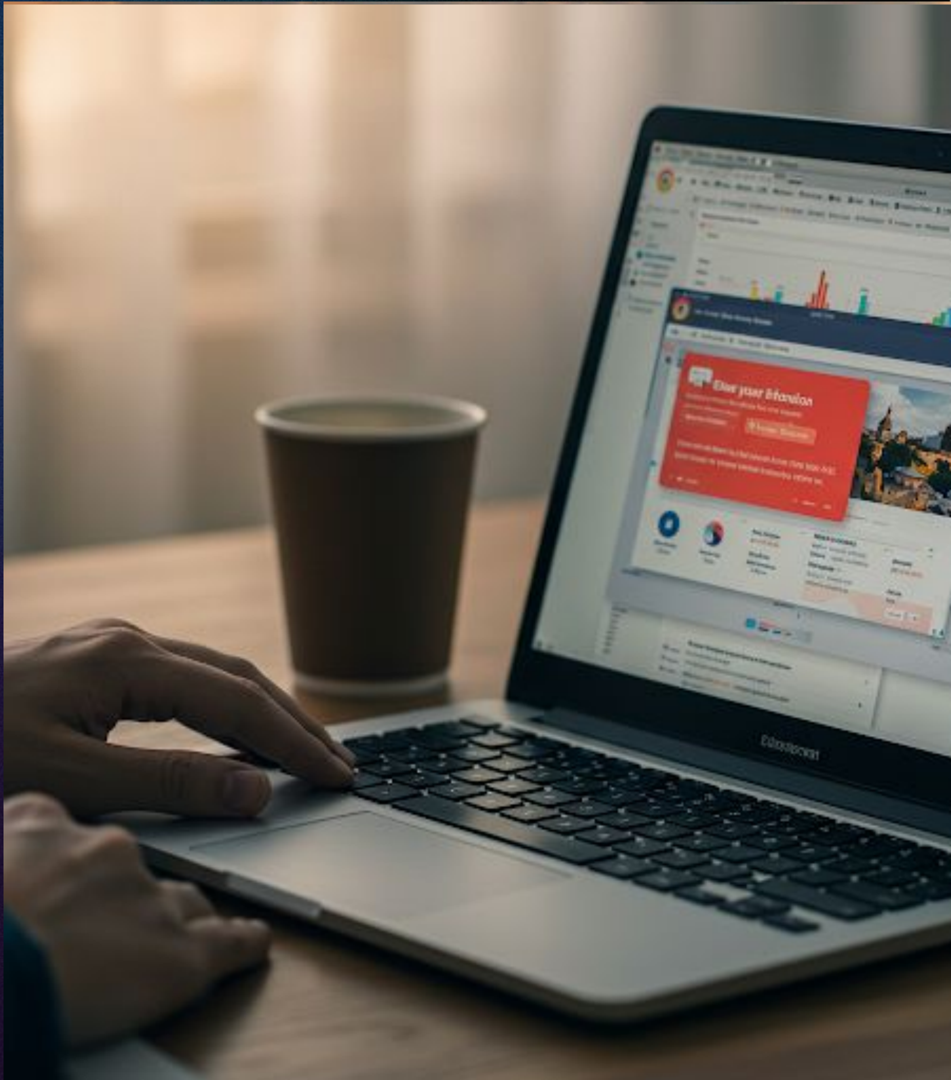
Building Chrome Extensions

08.02.2025
WORKSHOP PRESENTATION

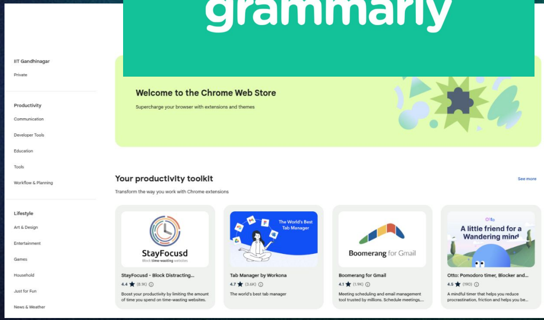
PRESENTORS: TEAM METIS

Agenda

- Introduction to Chrome Extensions
- Understanding the Structure
- Core Components
- Working with Chrome APIs
- Packaging & Testing



Why & What are Chrome Extensions ?



Welcome to the Chrome Web Store

Supercharge your browser with extensions and themes

Your productivity toolkit

Transform the way you work with Chrome extensions

StayFocusd - Block Distracting...
4.4 ★ (8.1K)
Boost your productivity by limiting the amount of time you spend on time-wasting websites.

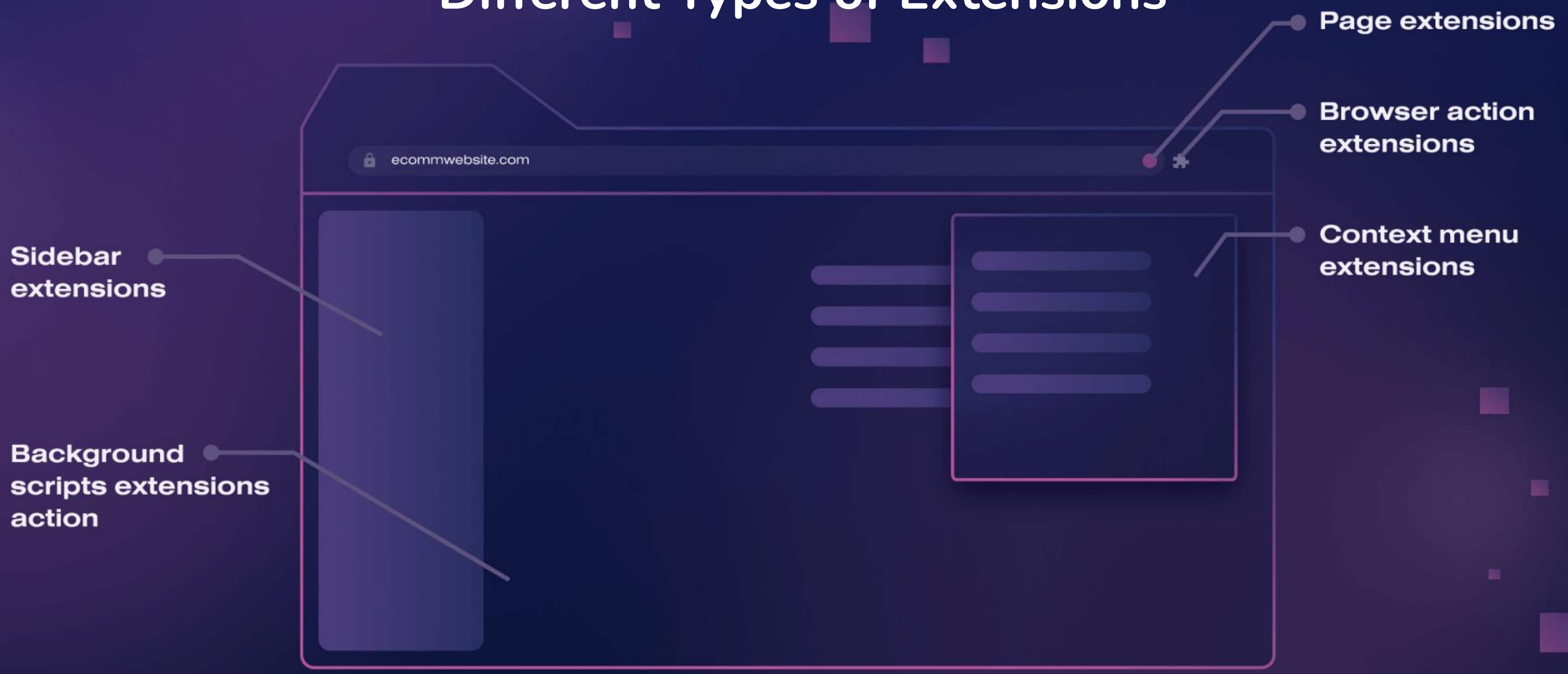
Tab Manager by Workona
4.7 ★ (3.6K)
The world's best tab manager

Boomerang for Gmail
4.1 ★ (1.9K)
Meeting scheduling and email management tool trusted by millions. Schedule meetings,...

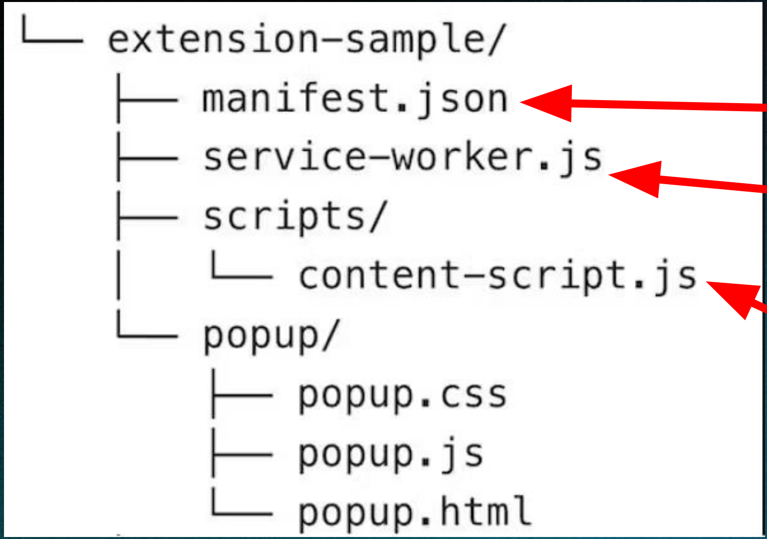
Otto: Pomodoro timer, Blocker and...
4.5 ★ (190)
A mindful timer that helps you reduce procrastination, friction and helps you be...

- To make your browsing experience smarter, faster, and tailored to your needs. Automate tasks, add new features, and customize the web to work exactly how you want it.

Different Types of Extensions



Basic Directory Structure

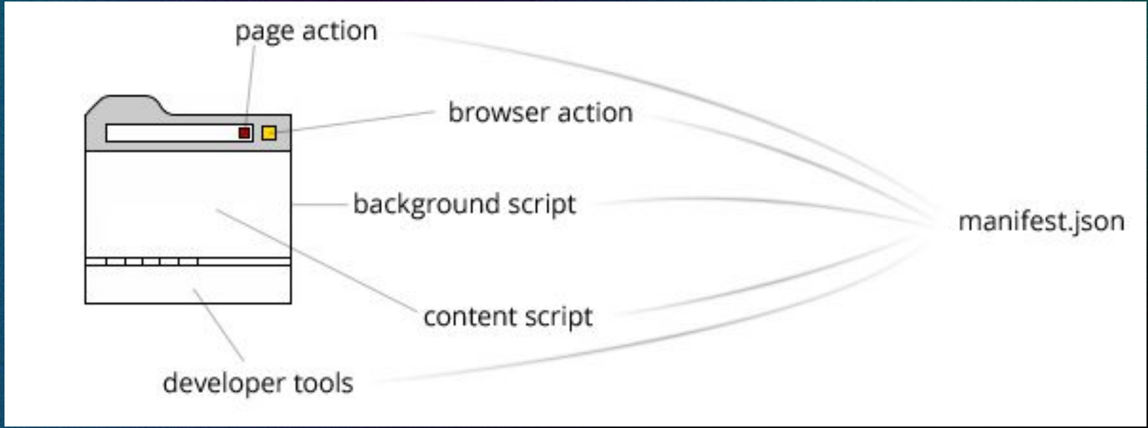


```
└── extension-sample/
    ├── manifest.json
    ├── service-worker.js
    ├── scripts/
    │   └── content-script.js
    └── popup/
        ├── popup.css
        ├── popup.js
        └── popup.html
```

The backbone of your extension

Runs in the background to handle events like network requests, caching, and messaging

Injects code into web pages



Core Components

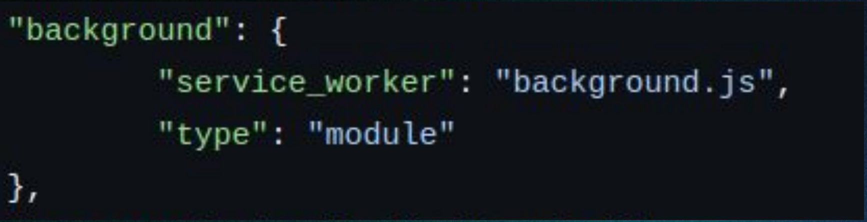
1. Manifest.json

Required

```json
{
        "name": "Library Extension by Metis",
        "manifest_version": 3,
        "version": "0.1.0",

        "permissions": ["contextMenus", "storage", "activeTab", "tabs", "sidePanel", "scripting"],
        "side_panel": {
                "default_path": "popup.html"
        },
        "action": { "default_title": "Generate a summary" },
        "background": {
                "service_worker": "background.js",
                "type": "module"
        },
        "host_permissions": ["<all_urls>"],
        "web_accessible_resources": [
                {
                        "resources": ["dialog.html"],
                        "matches": ["<all_urls>"]
                }
        ]
}
```

Core Components

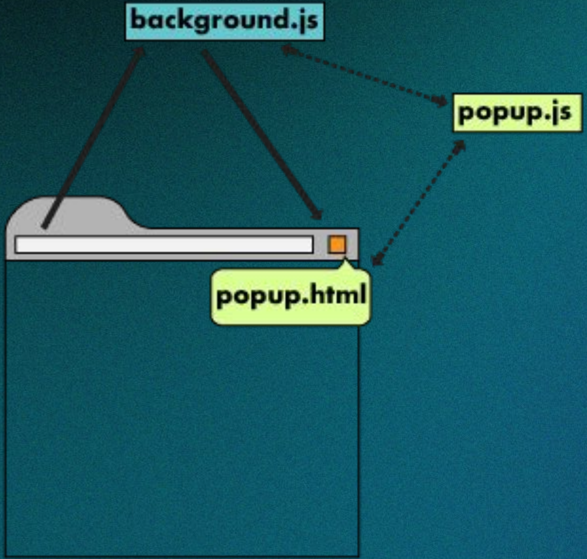
2. Background Scripts

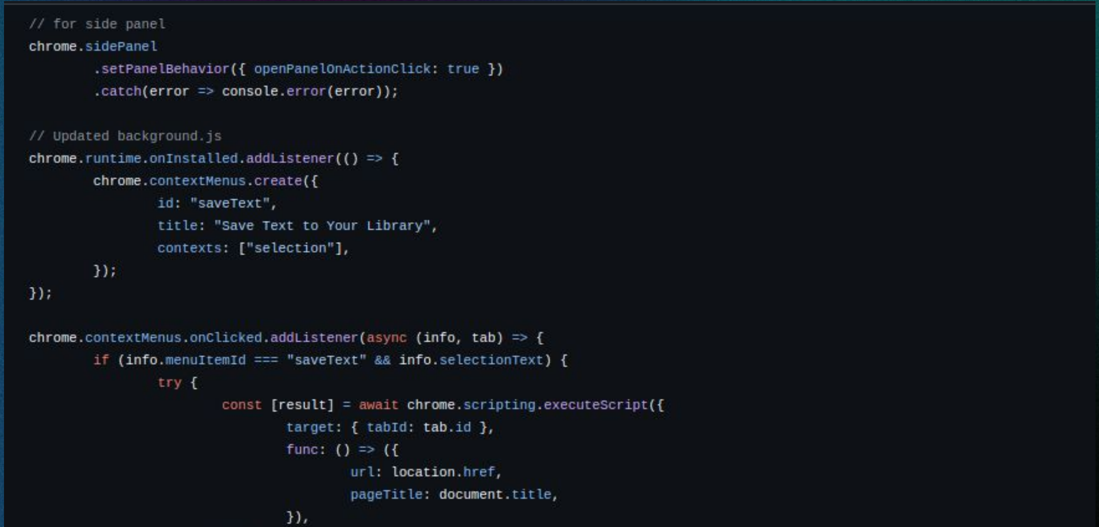


```
"background": {
        "service_worker": "background.js",
        "type": "module"
},
```

These are long-running scripts that handle events like browser startup, tab changes, or network requests.

Think of them as the **"brains"** of your extension, always listening and responding when needed.

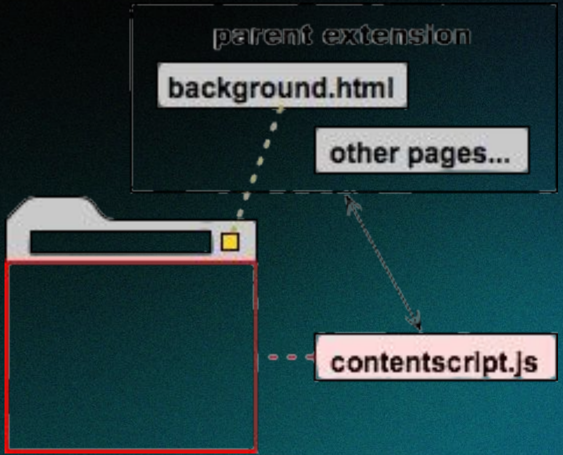


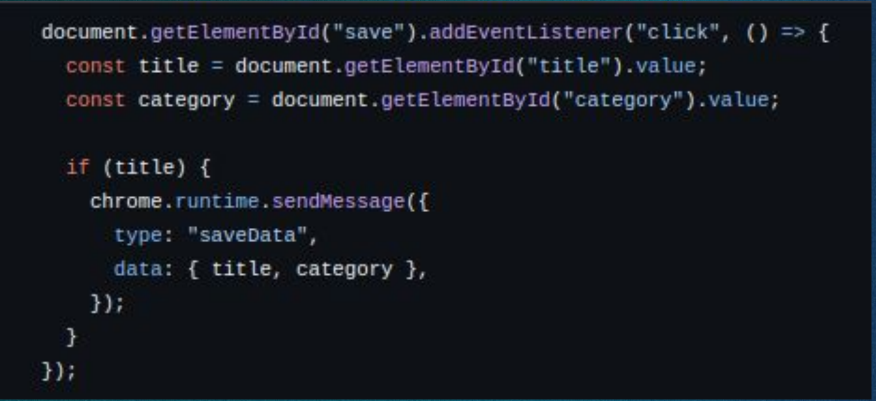


```
// for side panel
chrome.sidePanel
        .setPanelBehavior({ openPanelOnActionClick: true })
        .catch(error => console.error(error));

// Updated background.js
chrome.runtime.onInstalled.addListener(() => {
        chrome.contextMenus.create({
                id: "saveText",
                title: "Save Text to Your Library",
                contexts: ["selection"],
        });
});

chrome.contextMenus.onClicked.addListener(async (info, tab) => {
        if (info.menuItemId === "saveText" && info.selectionText) {
                try {
                        const [result] = await chrome.scripting.executeScript({
                                target: { tabId: tab.id },
                                func: () => ({
                                        url: location.href,
                                        pageTitle: document.title,
                                }),
```

Core Components

3. Content Scripts





```
document.getElementById("save").addEventListener("click", () => {
  const title = document.getElementById("title").value;
  const category = document.getElementById("category").value;

  if (title) {
    chrome.runtime.sendMessage({
      type: "saveData",
      data: { title, category },
    });
  }
});
```

Scripts that run directly inside web pages, allowing you to interact with and modify the DOM of the websites users visit.

Use this to inject custom elements, change styles, or gather data from pages.

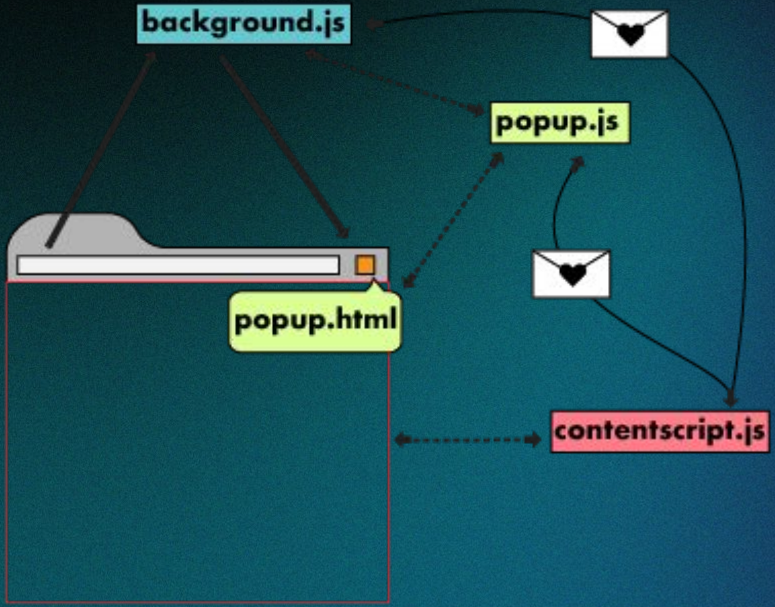
Important: Content scripts have limited access to Chrome APIs, but they can communicate with background scripts.

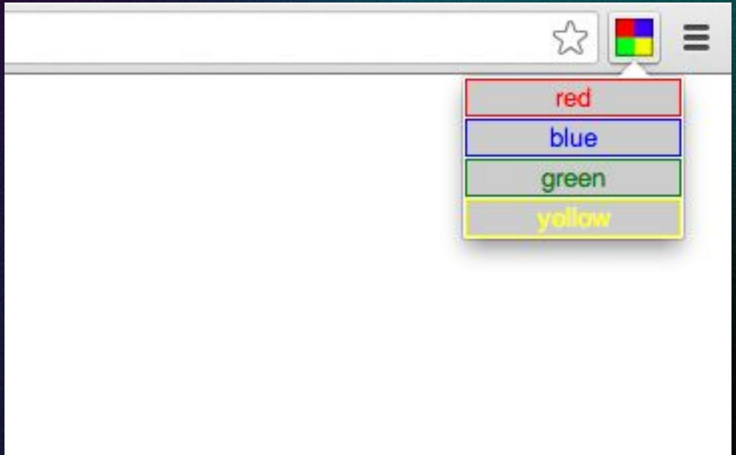
Core Components

4. Popup UI

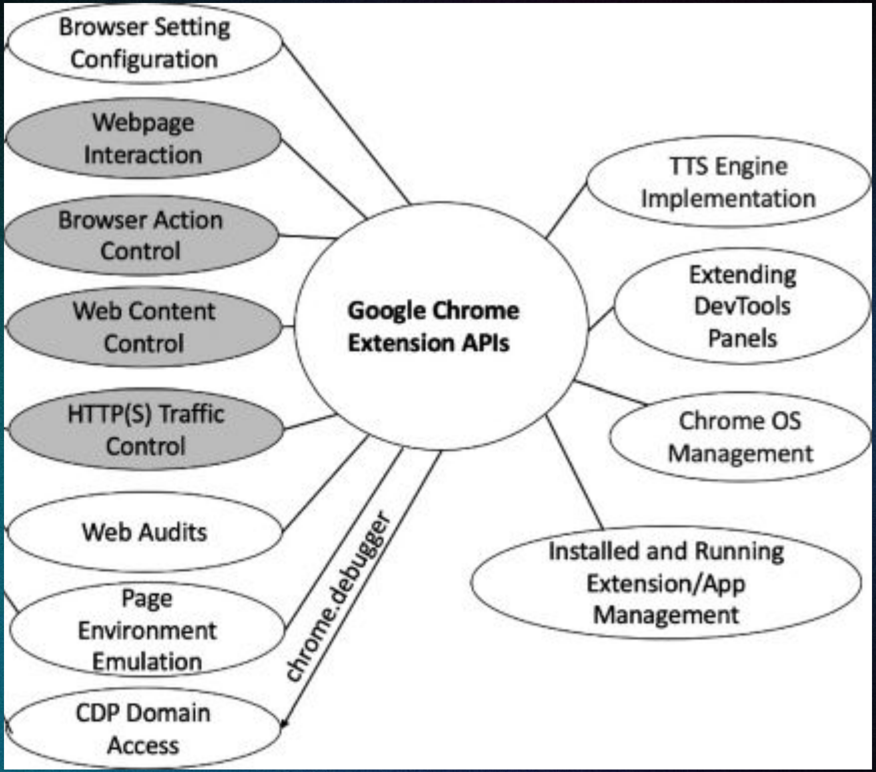
The small window that appears when you click the extension icon.

Built using HTML, CSS, and JavaScript, it's like a mini web page with buttons, forms, or info displays.





Chrome API’s


Browser Setting Configuration
Webpage Interaction
Browser Action Control
Web Content Control
HTTP(S) Traffic Control
Web Audits
Page Environment Emulation
CDP Domain Access
chrome.debugger
Google Chrome Extension APIs
TTS Engine Implementation
Extending DevTools Panels
Chrome OS Management
Installed and Running Extension/App Management

Refer:https://github.com/Metis-IITGandhinagar/ChromeExtensionWorkshop/tree/main?tab=readme-ov-file#chrome-apis

Mini Hackathon Schedule

12:00 PM SATURDAY	KICKOFF
6:30 PM 7:30 PM SATURDAY	MIDWAY CHECKPOINT
12:00 AM SUNDAY	SUBMISSIONS OPEN

HACKATHON

Theme: Productivity

Open Ended - No Problem Statement