



# Building Chrome Extensions

08.02.2025  
WORKSHOP PRESENTATION

PRESENTORS: TEAM METIS



# Agenda

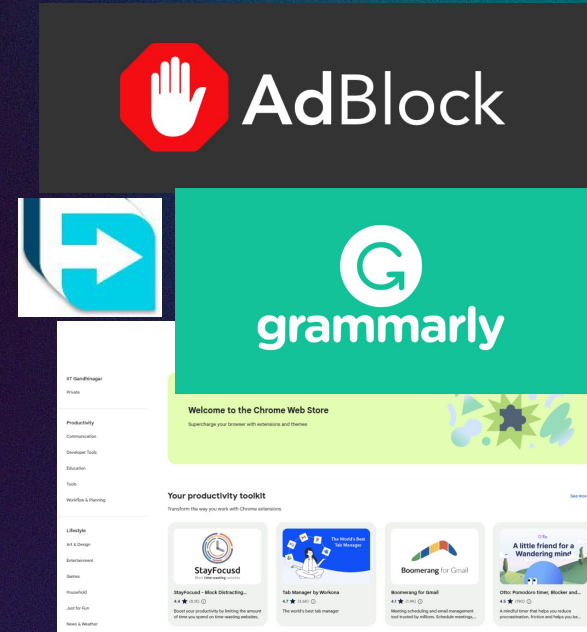
- Introduction to Chrome Extensions
- Understanding the Structure
- Core Components
- Working with Chrome APIs
- Packaging & Testing



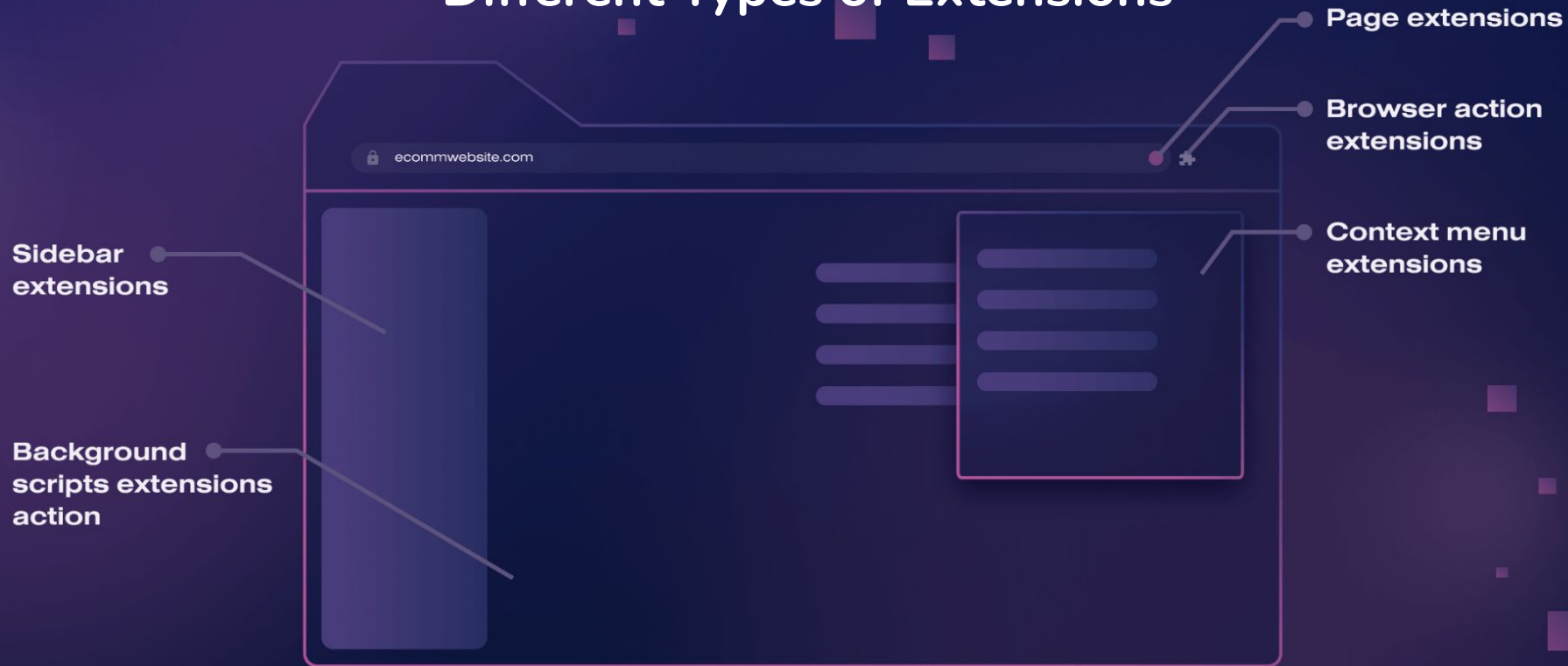


# Why & What are Chrome Extensions ?

- 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





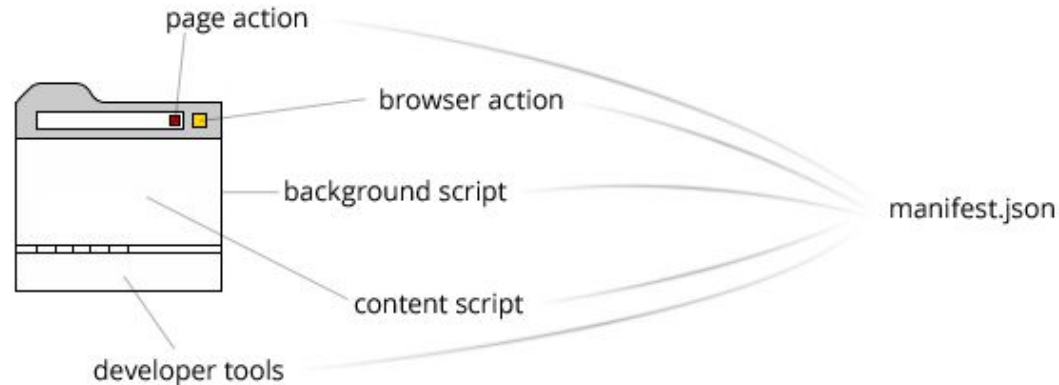
# Basic Directory Structure

```
└─ 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

```
1  {
2      "name": "Library Extension by Metis",
3      "manifest_version": 3,
4      "version": "0.1.0",
5
6      "permissions": ["contextMenus", "storage", "activeTab", "tabs", "sidePanel", "scripting"],
7      "side_panel": {
8          "default_path": "popup.html"
9      },
10     "action": { "default_title": "Generate a summary" },
11     "background": {
12         "service_worker": "background.js",
13         "type": "module"
14     },
15     "host_permissions": ["<all_urls>"],
16     "web_accessible_resources": [
17         {
18             "resources": ["dialog.html"],
19             "matches": ["<all_urls>"]
20         }
21     ]
22 }
```

Required



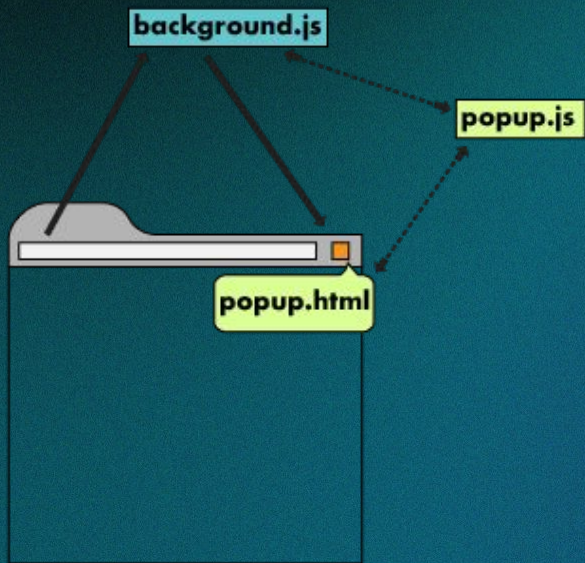
# Core Components

## 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,  
        })),  
    }  
  }  
});
```

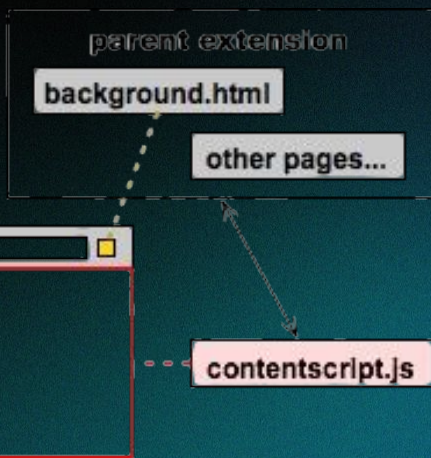


# Core Components

## 3. Content Scripts

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.



```
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 },  
    });  
  }  
});
```

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

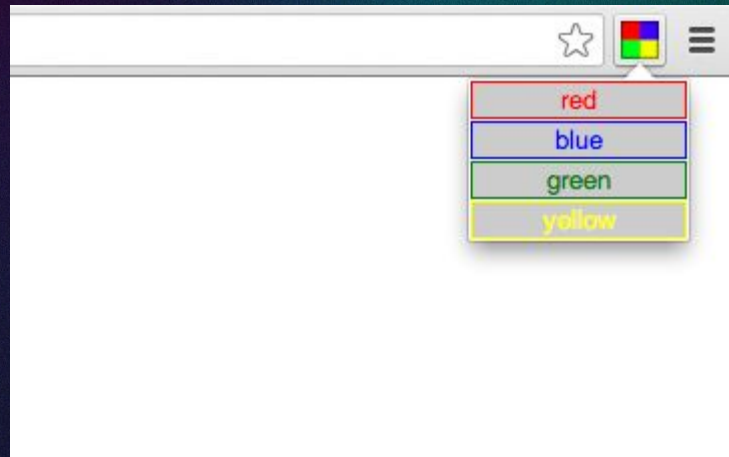
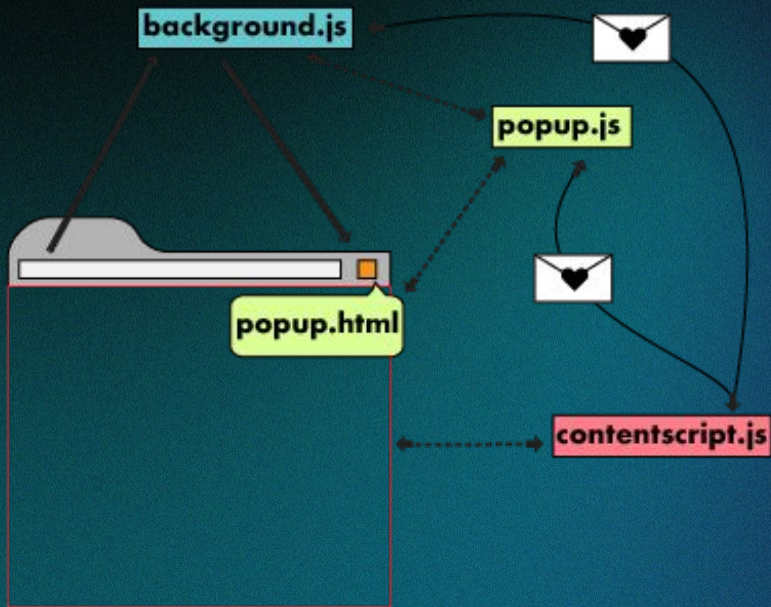


# Core Components

## 4. Popup UI

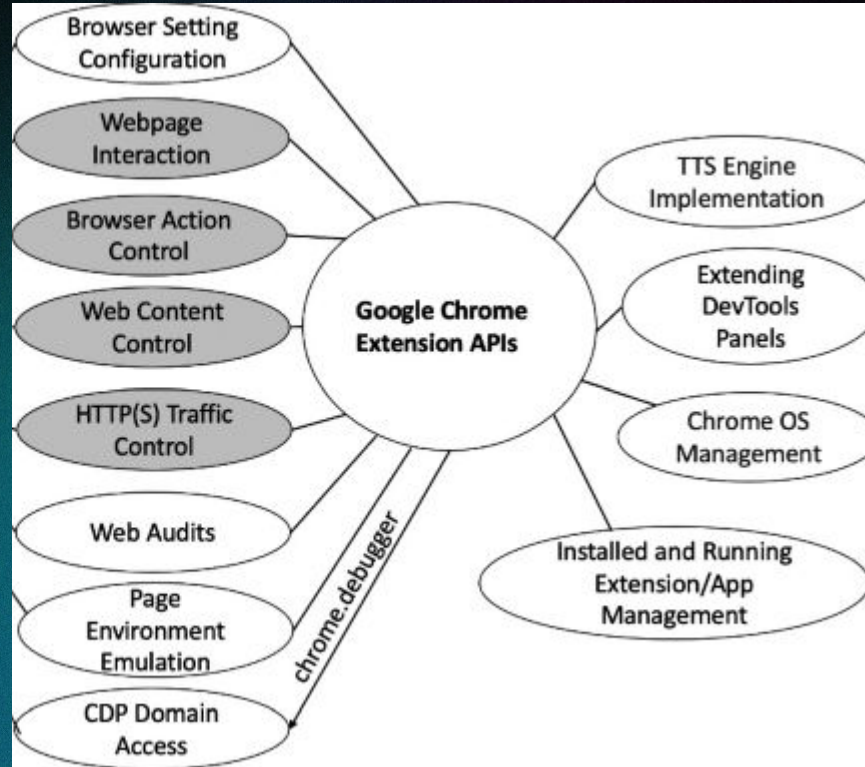
The small window that appears when you click the extension icon.

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





# Chrome API's



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