Demo1:pupup for changing background color

# 大綱

Chrome 提供的擴充功能(Extensions)可以讓程式人員利用 HTML, CSS, 和 JavaScript，來擴充Chrome功能。

這裡的範例，用來改變目前網頁（active page or active tab）的背景顏色。

利用UI element([**browser action**](https://developer.chrome.com/extensions/browserAction))，可以在Chrome的Omnibox加入按鈕。如下圖：



# Something to Declare

第一件事，就是建立manifest檔案manifest.json。利用JSON規格，指定擴充功能的名稱、描述、版本、permissions。

其中，permissions用來限定擴充功能的執行（參考 [**Manifest File Format documentation**](https://developer.chrome.com/extensions/manifest)）。

manifest.json([remote](https://developer.chrome.com/extensions/examples/tutorials/getstarted/manifest.json))

在這個例子中，宣告了一個 [browser action](https://developer.chrome.com/extensions/browserAction)，和 [activeTab permission](https://developer.chrome.com/extensions/activeTab) 允許檢查current tab 的URL和storage ，並用來記住使用者對這個網頁背景顏色的選擇。

{

 "manifest\_version": 2,

 "name": "Getting started example",

 "description": "This extension allows the user to change the background color of the current page.",

 "version": "1.0",

 "browser\_action": {

 "default\_icon": "icon.png",

 "default\_popup": "popup.html"

 },

 "permissions": [

 "activeTab",

 "storage"

 ]

}

|  |
| --- |
| "activeTab" permission, 對目前的active tab開放存取權。然後經由chrome.tabs API可以得到網頁的URLchrome.tabs.query({active: **true**, currentWindow: **true**}, **function**(tabs) { **var** currentURL **=** tabs[0].url; // 在這裡可以利用currentURL (這裡是callback function).});// 這裡不能用currentURL； 因為函數query（）中的第2個參數已經指定是 asynchronous |

## Resources

icon.png :

popup.html.

popup.js

remote( [**icon.png**](https://developer.chrome.com/extensions/examples/tutorials/getstarted/icon.png), [**manifest.json**](https://developer.chrome.com/extensions/examples/tutorials/getstarted/manifest.json), [**popup.html**](https://developer.chrome.com/extensions/examples/tutorials/getstarted/popup.html),[**popup.js**](https://developer.chrome.com/extensions/examples/tutorials/getstarted/popup.js).)

icon.png

##### 【popup.html】

<!doctype html**>**

<!--

 This page is shown when the extension button is clicked, because the

 "browser\_action" field in manifest.json contains the "default\_popup" key with

 value "popup.html".

 -->

**<html>**

 **<head>**

 **<title>**Getting Started Extension's Popup**</title>**

 **<style** type**=**"text/css"**>**

 body {

 margin: 10px;

 white**-**space: nowrap;

 }

 h1 {

 font**-**size: 15px;

 }

 #container {

 align**-**items: center;

 display: flex;

 justify**-**content: space**-**between;

 }

 **</style>**

 <!--

 - JavaScript and HTML must be in separate files: see our Content Security

 - Policy documentation[1] for details and explanation.

 -

 - [1]: https://developer.chrome.com/extensions/contentSecurityPolicy

 -->

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

 **</head>**

 **<body>**

 **<h1>**Background Color Changer**</h1>**

 **<div** id**=**"container"**>**

 **<span>**Choose a color**</span>**

 **<select** id**=**"dropdown"**>**

 **<option** selected disabled hidden value**=**''**></option>**

 **<option** value**=**"white"**>**White**</option>**

 **<option** value**=**"pink"**>**Pink**</option>**

 **<option** value**=**"green"**>**Green**</option>**

 **<option** value**=**"yellow"**>**Yellow**</option>**

 **</select>**

 **</div>**

 **</body>**

**</html>**

##### 【popup.js】

// Copyright (c) 2014 The Chromium Authors. All rights reserved.

// Use of this source code is governed by a BSD-style license that can be

// found in the LICENSE file.

/\*\*

 \* Get the current URL.

 \*

 \* @param {function(string)} callback called when the URL of the current tab

 \* is found.

 \*/

**function** getCurrentTabUrl(callback) {

 // Query filter to be passed to chrome.tabs.query - see

 // https://developer.chrome.com/extensions/tabs#method-query

 **var** queryInfo **=** {

 active: **true**,

 currentWindow: **true**

 };

 chrome.tabs.query(queryInfo, (tabs) **=>** {

 // chrome.tabs.query invokes the callback with a list of tabs that match the

 // query. When the popup is opened, there is certainly a window and at least

 // one tab, so we can safely assume that |tabs| is a non-empty array.

 // A window can only have one active tab at a time, so the array consists of

 // exactly one tab.

 **var** tab **=** tabs[0];

 // A tab is a plain object that provides information about the tab.

 // See https://developer.chrome.com/extensions/tabs#type-Tab

 **var** url **=** tab.url;

 // tab.url is only available if the "activeTab" permission is declared.

 // If you want to see the URL of other tabs (e.g. after removing active:true

 // from |queryInfo|), then the "tabs" permission is required to see their

 // "url" properties.

 console.assert(**typeof** url **==** 'string', 'tab.url should be a string');

 callback(url);

 });

 // Most methods of the Chrome extension APIs are asynchronous. This means that

 // you CANNOT do something like this:

 //

 // var url;

 // chrome.tabs.query(queryInfo, (tabs) => {

 // url = tabs[0].url;

 // });

 // alert(url); // Shows "undefined", because chrome.tabs.query is async.

}

/\*\*

 \* Change the background color of the current page.

 \*

 \* @param {string} color The new background color.

 \*/

**function** changeBackgroundColor(color) {

 **var** script **=** 'document.body.style.backgroundColor="' **+** color **+** '";';

 // See https://developer.chrome.com/extensions/tabs#method-executeScript.

 // chrome.tabs.executeScript allows us to programmatically inject JavaScript

 // into a page. Since we omit the optional first argument "tabId", the script

 // is inserted into the active tab of the current window, which serves as the

 // default.

 chrome.tabs.executeScript({

 code: script

 });

}

/\*\*

 \* Gets the saved background color for url.

 \*

 \* @param {string} url URL whose background color is to be retrieved.

 \* @param {function(string)} callback called with the saved background color for

 \* the given url on success, or a falsy value if no color is retrieved.

 \*/

**function** getSavedBackgroundColor(url, callback) {

 // See https://developer.chrome.com/apps/storage#type-StorageArea. We check

 // for chrome.runtime.lastError to ensure correctness even when the API call

 // fails.

 chrome.storage.sync.get(url, (items) **=>** {

 callback(chrome.runtime.lastError ? null : items[url]);

 });

}

/\*\*

 \* Sets the given background color for url.

 \*

 \* @param {string} url URL for which background color is to be saved.

 \* @param {string} color The background color to be saved.

 \*/

**function** saveBackgroundColor(url, color) {

 **var** items **=** {}; //空物件

 items[url] **=** color; //無法使用 一般物件新增欄位的方式，例如items.url=color 因為url 是字串。

 // [參考](https://developer.chrome.com/apps/storage#type-StorageArea) We omit the

 // optional callback since we don't need to perform any action once the

 // background color is saved.

 chrome.storage.sync.set(items);

}

// This extension loads the saved background color for the current tab if one

// exists. The user can select a new background color from the dropdown for the

// current page, and it will be saved as part of the extension's isolated

// storage. The chrome.storage API is used for this purpose. This is different

// from the window.localStorage API, which is synchronous and stores data bound

// to a document's origin. Also, using chrome.storage.sync instead of

// chrome.storage.local allows the extension data to be synced across multiple

// user devices.

document.addEventListener('DOMContentLoaded', () **=>** {

 getCurrentTabUrl((url) **=>** {

 **var** dropdown **=** document.getElementById('dropdown');

 // Load the saved background color for this page and modify the dropdown

 // value, if needed.

 getSavedBackgroundColor(url, (savedColor) **=>** {

 **if** (savedColor) {

 changeBackgroundColor(savedColor);

 dropdown.value **=** savedColor;

 }

 });

 // Ensure the background color is changed and saved when the dropdown

 // selection changes.

 dropdown.addEventListener('change', () **=>** {

 changeBackgroundColor(dropdown.value);

 saveBackgroundColor(url, dropdown.value);

 });

 });

});

## Load the extension

chrome://extensions

## Code 註解

按鈕相關的tooltip 可以在manifest 檔案中，browser\_action指定，例如：

{

 ...

 "browser\_action": {

 "default\_icon": "icon.png",

 "default\_popup": "popup.html",

 "default\_title": "Click here!"

 },

 ...

}

如果有更動，必須重載，才能正確顯示(課堂操作). 或是直接按F5（或CTRL─R）。  

## 其他參考

[**Chrome Extension Overview**](https://developer.chrome.com/extensions/overview)

[**debugging tutorial**](https://developer.chrome.com/extensions/tut_debugging)

[**chrome.\* APIs documentation**](https://developer.chrome.com/extensions/api_index) : 程式中相關browser actions的APIs。

[**developer's guide**](https://developer.chrome.com/extensions/devguide) 。