建立一個 Chrome 擴展

你有沒有曾經打開太多瀏覽器分頁並希望有一個工具來自動管理它們？在這篇部落格文章中，我們將逐步介紹如何創建一個名為「Tabs Killer」的 Chrome 擴展程序，它會在分頁數量超過用戶定義的限制時自動關閉最舊的分頁。我將分解代碼，解釋其工作原理，並提供見解來幫助你構建自己的 Chrome 擴展程序。

在本文結束時，你將了解 Chrome 擴展程序的結構、如何使用 Chrome API 以及如何創建一個帶有設置的彈出界面。

「Tabs Killer」的功能

「Tabs Killer」是一個 Chrome 擴展程序，它：- 監控打開的分頁數量。- 允許用戶設置最大分頁限制。- 當限制超過時自動關閉最舊的分頁。- 提供白名單功能來保護特定分頁（例如，根據 URL 模式）不被關閉。

擴展程序包括一個彈出界面來配置設置和一個後台腳本來處理分頁管理。

專案結構

這是「Tabs Killer」擴展程序的文件結構：

```
tabs-killer/
  manifest.json          # 擴展配置
  popup.html             # 彈出UI
  popup.js              # 彈出邏輯
  background.html        # 後台頁面
  app.build.js           # 主應用邏輯（假設）
  js/
     lib/               # 外部庫（jQuery, Underscore, Bootstrap, RequireJS）
     tabmanager.js      # 分頁管理邏輯（假設）
     settings.js        # 設置管理（假設）
  css/
     popup.css          # 彈出樣式
  img/
      icon16.png         # 16x16圖標
      icon48.png         # 48x48圖標
      icon128.png        # 128x128圖標
```

第 1 步：清單文件（manifest.json）

manifest.json 文件是任何 Chrome 擴展程序的核心。它定義了元數據、權限和關鍵組件。

```
{
  "manifest_version": 2,
  "name": "Tabs Killer",
  "description": " 當分頁過多時自動關閉最舊的分頁。",
  "version": "1.0",
  "browser_action": {
    "default_icon": "img/icon128.png",
    "default_popup": "popup.html"
  },
  "icons": {
    "128": "img/icon128.png",
    "48": "img/icon48.png",
    "16": "img/icon16.png"
  },
  "background": {
    "page": "background.html"
  },
  "permissions": [
    "tabs",
    "storage"
  ],
  "content_security_policy": "script-src 'self' 'unsafe-eval'; object-src 'self'"
}
```

說明：

- `manifest_version`：必須是 `2`（Chrome 已廢棄版本 1）。
- `name`，`description`，`version`：基本元數據。
- `browser_action`：定義擴展程序的工具欄圖標和彈出界面（`popup.html`）。
- `icons`：不同尺寸的圖標（在 Chrome Web Store 和工具欄中使用）。
- `background`：指定一個後台頁面（`background.html`），該頁面持續運行。
- `permissions`：請求訪問 `tabs` API（管理分頁）和 `storage` API（保存設置）。
- `content_security_policy`：允許 `unsafe-eval` 用於像 RequireJS 這樣的庫（在生產中謹慎使用）。

第 2 步：彈出 UI（popup.html）

當用戶點擊擴展程序圖標時，彈出界面會出現。它使用 Bootstrap 進行樣式設置，並包括一個帶有「選項」部分的分頁界面。

```
<!doctype html>
<html>
<head>
  <title>Tabs Killer 擴展程序的彈出界面</title>
  <link rel="stylesheet" href="js/lib/bootstrap/css/bootstrap.css" type="text/css"/>
  <link rel="stylesheet" href="css/popup.css"/>
  <script src="js/lib/jquery.min.js"></script>
  <script src="js/lib/underscore.js"></script>
  <script src="js/lib/bootstrap/js/bootstrap.min.js"></script>
  <script src="js/lib/bootstrap/js/bootstrap-tab.js"></script>
  <script src="js/lib/require.js"></script>
  <script src="app.build.js"></script>
  <script src="popup.js"></script>
</head>
<body>
  <ul class="nav nav-tabs">
    <li><a href="#tabOptions" target="#tabOptions" data-toggle="tab">選項</a></li>
  </ul>
  <div class="tab-content">
    <div class="tab-pane active" id="tabOptions">
      <form class="well">
        <fieldset>
          <legend>設置</legend>
          <p>
            <label for="maxTabs">最大保留分頁數</label>
            <input type="text" id="maxTabs" class="span1" name="maxTabs"> 分頁
          </p>
        </fieldset>
        <div id="status" class="alert alert-success invisible"></div>
        <fieldset>
          <legend>自動鎖定</legend>
          <label for="white-list-input">包含字符串的分頁 URL：</label>
          <input type="text" id="white-list-input"/>
          <button class="btn-mini add-on" disabled id="white-list-add">添加</button>
          <table class="table table-bordered table-striped" id="white-list">
            <thead>
```

```
            <tr>
              <th>URL 模式</th>
              <th></th>
            </tr>
          </thead>
          <tbody></tbody>
        </table>
      </fieldset>
    </form>
  </div>
</div>
<script type="text/html" id="url-item-template">
  <tr>
    <td><%=url%></td>
    <td><a class="deleteLink" href="#">移除</a></td>
  </tr>
</script>
</body>
</html>
```

說明：

- **庫**：使用 jQuery、Underscore、Bootstrap 和 RequireJS 進行功能和樣式設置。
- **UI 元素**：
  - 一個文本輸入（`#maxTabs`）用於設置最大分頁數。
  - 一個白名單輸入（`#white-list-input`）和「添加」按鈕（`#white-list-add`）來保護特定 URL。
  - 一個表格（`#white-list`）來顯示白名單模式，並帶有「移除」鏈接。
  - 一個狀態消息（`#status`）來顯示保存反饋。
- **模板**：一個 Underscore.js 模板（`#url-item-template`）動態生成表格行。

第 3 步：彈出邏輯（`popup.js`）

這個腳本處理彈出界面的互動性，例如保存設置和管理白名單。

```
require([], function () {
  var GlobalObject = chrome.extension.getBackgroundPage().GlobalObject;

  Popup = {};
```

```
Popup.optionsTab = {};

Popup.optionsTab.init = function (context) {
  function onBlurInput() {
    var key = this.id;
    Popup.optionsTab.saveOption(key, $(this).val());
  }
  $('#maxTabs').keyup(_.debounce(onBlurInput, 200));
  Popup.optionsTab.loadOptions();
};

Popup.optionsTab.loadOptions = function () {
  $('#maxTabs').val(GlobalObject.settings.get('maxTabs'));
  var whiteList = GlobalObject.settings.get('whiteList');
  Popup.optionsTab.buildWhiteListTable(whiteList);

  var $whiteListInput = $('#white-list-input');
  var $whiteListAdd = $('#white-list-add');

  var isValid = function (pattern) {
    return /\S/.test(pattern);
  };

  $whiteListInput.on('input', function () {
    if (isValid($whiteListInput.val())) {
      $whiteListAdd.removeAttr('disabled');
    } else {
      $whiteListAdd.attr('disabled', 'disabled');
    }
  });

  $whiteListAdd.click(function () {
    if (!isValid($whiteListInput.val())) return;
    whiteList.push($whiteListInput.val());
    $whiteListInput.val('').trigger('input').focus();
    Popup.optionsTab.saveOption('whiteList', whiteList);
    Popup.optionsTab.buildWhiteListTable(whiteList);
  });
};
```

```
  Popup.optionsTab.saveOption = function (key, value, hideStatus) {
    if (!hideStatus) $('#status').html('');
    GlobalObject.settings.set(key, value);
    if (!hideStatus) {
      $('#status').removeClass('invisible').css('opacity', '100')
        .html('保存中...').delay(50).animate({opacity: 0});
    }
  };

  Popup.optionsTab.buildWhiteListTable = function (whiteList) {
    var urlItemTemplate = _.template($("#url-item-template").html());
    var $wlTable = $('table#white-list tbody');
    $wlTable.html('');
    for (var i = 0; i < whiteList.length; i++) {
      var $tr = $(urlItemTemplate({url: whiteList[i]}));
      var $deleteLink = $tr.find('a.deleteLink').parent();
      $deleteLink.click(function () {
        whiteList.splice(whiteList.indexOf($(this).data('pattern')), 1);
        Popup.optionsTab.saveOption('whiteList', whiteList, true);
        Popup.optionsTab.buildWhiteListTable(whiteList);
      }).data('pattern', whiteList[i]);
      $wlTable.append($tr);
    }
  };

  $(document).ready(function () {
    $('a[data-toggle="tab"]').on('show', function (e) {
      var tabId = e.target.hash;
      if (tabId === '#tabOptions') {
        Popup.optionsTab.init($('div#tabOptions'));
      }
    });
    $('a[href="#tabOptions"]').click();
  });
});
```

說明：

- **初始化**：連接到後台頁面的 `GlobalObject` 以獲取設置和分頁管理。
- `init`：設置事件監聽器，例如對 `#maxTabs` 的防抖輸入。

- `loadOptions`：加載保存的設置（最大分頁和白名單）並填充 UI。
- `saveOption`：將設置保存到 `GlobalObject.settings` 並顯示「保存中…」動畫。
- `buildWhiteListTable`：動態構建白名單表格，並帶有刪除功能。
- **事件監聽器**：處理輸入驗證、添加白名單條目和分頁切換。

第 4 步：後台邏輯（background.html 和假設腳本）

後台頁面（background.html）持續運行並加載核心邏輯。

```
// background.js（假設，基於提供的片段）
GlobalObject = {};

require(['tabmanager', 'settings'], function (tabmanager, settings) {
  var startup = function () {
    GlobalObject.settings = settings;
    GlobalObject.tabmanager = tabmanager;
    settings.init();
    tabmanager.init();
  };
  startup();
});
```

假設：

- `settings.js`：管理存儲（例如，chrome.storage）設置，如 `maxTabs` 和 `whiteList`。
- `tabmanager.js`：使用 `tabs` API 監控並根據 `maxTabs` 限制和 `whiteList` 關閉分頁。

假設的 `tabmanager.js` 範例：

```
var tabmanager = {
  init: function () {
    chrome.tabs.onCreated.addListener(this.checkTabCount);
  },
  checkTabCount: function () {
    chrome.tabs.query({}, function (tabs) {
      var maxTabs = GlobalObject.settings.get('maxTabs') || 10;
      var whiteList = GlobalObject.settings.get('whiteList') || [];
      if (tabs.length > maxTabs) {
        var tabsToRemove = tabs.filter(tab => !whiteList.some(pattern => tab.url.includes(pattern)));
```

```
        chrome.tabs.remove(tabsToRemove[0].id); // 移除最舊的分頁
      }
    });
  }
};
```

如何測試擴展程序

1. 打開 Chrome 並前往 `chrome://extensions/`。
2. 啟用「開發者模式」（右上角切換）。
3. 點擊「加載未打包的擴展程序」並選擇 `tabs-killer` 文件夾。
4. 點擊擴展程序圖標打開彈出界面並測試設置。

撰寫自己的 Chrome 擴展程序的技巧

1. **從小開始**：從簡單的清單和彈出或後台腳本開始。
2. **使用 Chrome API**：根據需要利用 `chrome.tabs`、`chrome.storage` 等。
3. **調試**：使用 `console.log` 和 Chrome 的開發者工具（右鍵點擊彈出界面 > 檢查）。
4. **安全性**：避免在生產中使用 `unsafe-eval`；使用更嚴格的內容安全策略。
5. **UI 庫**：Bootstrap 和 jQuery 簡化了 UI 開發，但保持擴展程序輕量。

結論

「Tabs Killer」展示了如何結合彈出 UI、後台邏輯和 Chrome API 來創建一個功能擴展程序。有了這個基礎，你可以進一步自定義它——添加通知、精緻化分頁關閉邏輯或增強 UI。

隨意實驗代碼並分享你自己的 Chrome 擴展程序想法！快樂編碼！