



EBook Gratis

APRENDIZAJE google-chrome- extension

Free unaffiliated eBook created from
Stack Overflow contributors.

#google-
chrome-
extension

Tabla de contenido

Acerca de.....	1
Capítulo 1: Empezando con google-chrome-extension.....	2
Observaciones.....	2
TODO: Breve descripción de las extensiones de Chrome.....	2
Documentacion oficial.....	2
Otras lecturas.....	2
TODO: poblar con enlaces a temas importantes de visión general.....	2
Examples.....	2
Ejemplo mínimo absoluto.....	2
Página de fondo.....	3
Guiones de contenido.....	4
Ver también.....	4
Página de opciones.....	5
Versión 2.....	5
Versión 1 (obsoleta).....	6
Almacenamiento.....	6
Documentacion oficial.....	6
Crear una nueva pestaña.....	6
Capítulo 2: Depuración de extensiones de Chrome.....	8
Examples.....	8
Usando las herramientas del desarrollador para depurar su extensión.....	8
Capítulo 3: Guiones de contenido.....	10
Observaciones.....	10
Documentacion oficial.....	10
Examples.....	10
Declarar scripts de contenido en el manifiesto.....	10
Ejemplo minimo.....	10
Nota IMPORTANTE.....	11
Inyectar scripts de contenido desde una página de extensión.....	11

Ejemplo minimo.....	11
Código en línea.....	11
Elegir la pestaña.....	11
Permisos.....	12
Comprobando errores.....	12
Múltiples scripts de contenido en el manifiesto.....	12
Las mismas condiciones, múltiples scripts.....	12
Mismos guiones, múltiples sitios.....	12
Diferentes guiones o diferentes sitios.....	13
Capítulo 4: Integración de herramientas de desarrollador.....	14
Examples.....	14
Indicaciones de punto de interrupción programáticas.....	14
Depuración de la página de fondo / script.....	14
Depurando la ventana emergente.....	15
Capítulo 5: manifest.json.....	16
Observaciones.....	16
Documentacion oficial.....	16
Formato.....	16
Examples.....	17
Absoluto mínimo manifest.json.....	17
Obtención del manifiesto a partir del código de extensión.....	17
Capítulo 6: Páginas de fondo.....	18
Examples.....	18
Declarando página de fondo en el manifiesto.....	18
Capítulo 7: Paso de mensajes.....	19
Observaciones.....	19
Documentacion oficial.....	19
Examples.....	19
Enviar una respuesta de forma asíncrona.....	19
Capítulo 8: Portando a / desde Firefox.....	21
Observaciones.....	21

Examples.....	21
Portando a través de WebExtensions.....	22
Extensiones compatibles basadas en WebExtension.....	22
Una extensión simple que puede funcionar en Firefox y Google Chrome.....	22
Si el complemento actual se basa en el SDK de complemento o XUL.....	24
Creditos.....	26

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [google-chrome-extension](#)

It is an unofficial and free google-chrome-extension ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official google-chrome-extension.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con google-chrome-extension

Observaciones

TODO: Breve descripción de las extensiones de Chrome

Documentacion oficial

- [¿Qué son las extensiones?](#) (centro de documentación)
- [Tutorial de inicio](#) (tutorial básico)
- [Visión general](#)
- [API de JavaScript](#) (lista completa de `chrome.*` API)

Otras lecturas

TODO: poblar con enlaces a temas importantes de visión general

Examples

Ejemplo mínimo absoluto

Cualquier extensión de Chrome se inicia como una *extensión desempaquetada* : una carpeta que contiene los archivos de la extensión.

Un archivo que debe contener es [manifest.json](#) , que describe las propiedades básicas de la extensión. Muchas de las propiedades en ese archivo son opcionales, pero aquí hay un archivo `manifest.json` mínimo absoluto:

```
{
  "manifest_version": 2,
  "name": "My Extension",
  "version": "1.0"
}
```

Cree una carpeta (por ejemplo, `myExtension`) en algún lugar, agregue `manifest.json` como se indica arriba.

Luego, necesitas cargar la extensión en Chrome.

1. Abra `chrome://extensions/` page, accesible a través de **Menú> Más herramientas> Extensiones**.
2. Habilite el **modo de desarrollador** con una casilla de verificación en la parte superior derecha, si aún no está habilitado.
3. Haga clic en el botón **Cargar extensión desempaquetada ...** y seleccione la carpeta `myExtension` creada.

Extensions

2 Developer mode

3 Load unpacked extension... Pack extension...

Update extensions...

¡Eso es! Chrome carga tu primera extensión:



My Extension 1.0

Enabled



[Details](#) [Reload \(Ctrl+R\)](#)

ID: `gdgiijhpbdlebnhblpfplpolomkjbmm`

Loaded from: `C:\Devel\myExtension`

Allow in incognito

Por supuesto, todavía no hace nada, por lo que es un buen momento para leer una [descripción general de la arquitectura de extensión](#) para comenzar a agregar las piezas que necesita.

Importante: cuando realice cambios en su extensión, no olvide volver a `chrome://extensions/` y presione el enlace **Recargar** para su extensión después de realizar los cambios. En el caso de los scripts de contenido, vuelva a cargar la página de destino también.

Página de fondo

Las páginas de fondo son páginas implícitas que contienen scripts de fondo. Una secuencia de comandos de fondo es una secuencia de comandos de larga ejecución para administrar alguna tarea o estado. Existe durante toda la vida de su extensión, y solo una instancia de ella está activa a la vez.

Puedes declararlo así en tu `manifest.json`:

```
"background": {
  "scripts": ["background.js"]
}
```

El sistema de extensión generará una página de fondo que incluye cada uno de los archivos enumerados en la propiedad de `scripts`.

Tienes acceso a todas las API de `chrome.*` Permitidas.

Hay dos tipos de páginas de fondo: páginas de fondo **persistentes** que siempre están abiertas, y **páginas de eventos** que se abren y cierran según sea necesario.

Si desea que su página de fondo no sea persistente, solo tiene que establecer la `persistent` -flag en falso:

```
"background": {
  "scripts": ["eventPage.js"],
  "persistent": false
}
```

Esta secuencia de comandos de fondo solo está activa si se activa un evento en el que tiene un oyente registrado. En general usas un `addListener` para el registro.

Ejemplo: la aplicación o extensión se instala primero.

```
chrome.runtime.onInstalled.addListener(function() {
  console.log("The Extension is installed!");
});
```

Guiones de contenido

Un **script de contenido** es un código de extensión que se ejecuta junto con una página normal.

Tienen acceso completo al DOM de la página web (y, de hecho, son **la única parte de la extensión que puede acceder al DOM de una página**), pero el código JavaScript está aislado, un concepto llamado **Mundo aislado**. Cada extensión tiene su propio contexto de JavaScript de script de contenido invisible para los demás y la página, lo que evita conflictos de código.

Ejemplo de definición en `manifest.json`:

```
"content_scripts": [
  {
    "matches": ["http://www.stackoverflow.com/*"],
    "css": ["style.css"],
    "js": ["jquery.js", "myscript.js"]
  }
]
```

Los atributos tienen el siguiente significado:

Atributo	Descripción
partidos	Especifica en qué páginas se inyectará este script de contenido. Sigue el formato de patrón de coincidencia .
css	Lista de archivos CSS para ser inyectados en páginas coincidentes.
js	Lista de archivos JS para ser inyectados en páginas coincidentes. Ejecutado en el orden indicado.

Los scripts de contenido también se pueden inyectar a pedido mediante `chrome.tabs.executeScript`, que se denomina **inyección programática**.

Ver también

- Documentación oficial: [Scripts de contenido](#)
- Documentación de desbordamiento de pila: [Scripts de contenido](#)

Página de opciones

Las **páginas de opciones** se utilizan para dar al usuario la posibilidad de mantener la configuración de su extensión.

Versión 2

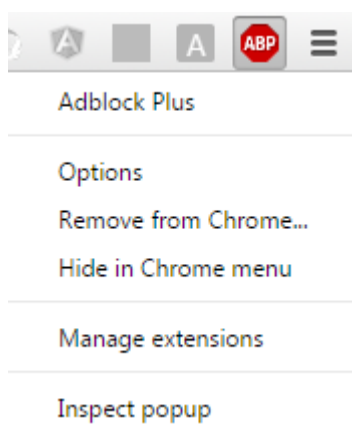
Desde Chrome 40, existe la posibilidad de tener la página de opciones como un diálogo predefinido en `chrome://extensions`.

La forma de definir una página de opciones en `manifest.json` es como la siguiente:

```
"options_ui": {  
  "page": "options.html",  
  "chrome_style": true  
}
```

Esta página de opciones se comportará como un diálogo, se abrirá como una ventana emergente, donde se mostrará el **options.html**. `chrome_style` aplicará una hoja de estilo de Chrome por motivos de coherencia de estilo a su página de opciones.

Las opciones se mostrarán automáticamente a través del menú contextual del **botón de extensión** o la página de **extensiones chrome://**.





Adblock Plus 1.12.1

✔ Enabled



Used by over 50 million people, a free ad blocker that blocks ALL annoying ads, malware and tracking.

[Details](#) [Options](#)

ID: cfhdojbkjhnklbpkdaibdccddilifddb

Inspect views: [background page](#)

Allow in incognito

También puede [abrir la página de opciones mediante programación](#) , por ejemplo, desde una interfaz de usuario emergente:

```
chrome.runtime.openOptionsPage();
```

Versión 1 (obsoleta)

Ejemplo de definición en [manifest.json](#) :

```
"options_page": "options.html"
```

Se recomienda usar la Versión 2, ya que el comportamiento de `options_ui` se aplicará pronto a las páginas de opciones de la Versión 1.

Almacenamiento

Normalmente, la configuración debe persistir, por lo que se recomienda usar la API `chrome.storage` . Los permisos se pueden declarar así en el [manifest.json](#) :

```
"permissions": [  
  "storage"  
]
```

Documentacion oficial

- [Página de opciones - Versión 1](#)
- [Página de opciones - Versión 2](#)
- [API de almacenamiento](#)

Crear una nueva pestaña

En el código de extensión, puedes usar cualquier API de `chrome.*` Si has despachado los permisos requeridos. Además, algunas API solo funcionan desde páginas de fondo, y algunas API solo funcionan desde scripts de contenido.

Puede utilizar la mayoría de los métodos `chrome.tabs` que declaran cualquier permiso. Ahora nos centramos en `chrome.tabs.create`

Nota: la nueva pestaña se abrirá sin ningún aviso `popup` .

```
chrome.tabs.create({
  url:"http://stackoverflow.com",
  selected:false // We open the tab in the background
})
```

Puedes aprender más sobre el objeto de pestaña, en el [desarrollador oficial de Chrome](#)

Lea [Empezando con google-chrome-extension en línea](#): <https://riptutorial.com/es/google-chrome-extension/topic/787/empezando-con-google-chrome-extension>

Capítulo 2: Depuración de extensiones de Chrome

Examples

Usando las herramientas del desarrollador para depurar su extensión

Una extensión de cromo se separa en un máximo de 4 partes:

- la página de fondo
- la página emergente
- uno o más scripts de contenido
- la página de opciones

Cada parte, ya que están separadas de forma innata, requiere depuración individual.

Tenga en cuenta que estas páginas están separadas, lo que significa que las variables no se comparten directamente entre ellas y que un `console.log()` en una de estas páginas no será visible en los registros de ninguna otra parte.

Usando los devtools de cromo:

Las extensiones de Chrome se depuran de forma similar a otras aplicaciones web y páginas web. La depuración se realiza con mayor frecuencia con el uso del inspector de devtools de Chrome utilizando el método abreviado de teclado para Windows y Mac respectivamente: `ctrl + shift + i` y `cmd + shift + i` o haciendo clic derecho en la página y seleccionando inspeccionar.

Desde el inspector, un desarrollador puede verificar los elementos html y cómo los afecta css, o usar la consola para inspeccionar los valores de las variables de javascript y leer los resultados de cualquier `console.log()` que haya configurado el desarrollador.

Puede encontrar más información sobre el uso del inspector en [Chrome Devtools](#) .

Inspeccionando la ventana emergente, la página de opciones y otras páginas accesibles usando chrome: //.....yourExtensionId.../:

Se puede acceder a la *página emergente* y la *página de opciones* simplemente inspeccionándolas cuando están abiertas.

Las páginas html adicionales que forman parte de la extensión, pero que no son la ventana emergente ni la página de opciones, también se depuran de la misma manera.

Inspeccionando la página de fondo:

Para acceder a su *página de fondo* , primero debe navegar a la página de la extensión de [chrome](#) en [chrome: // extensions /](#) . Asegúrese de que la marca de verificación 'Modo desarrollador' esté

habilitada.

Extensions

Load unpacked extension... Pack extension...

Developer mode

Update extensions now

Luego haga clic en el script de fondo junto a "Inspeccionar vistas" para inspeccionar su página de fondo.



My Extension 2.0 Enabled 

Do Stuff!

Details Reload (Ctrl+R)

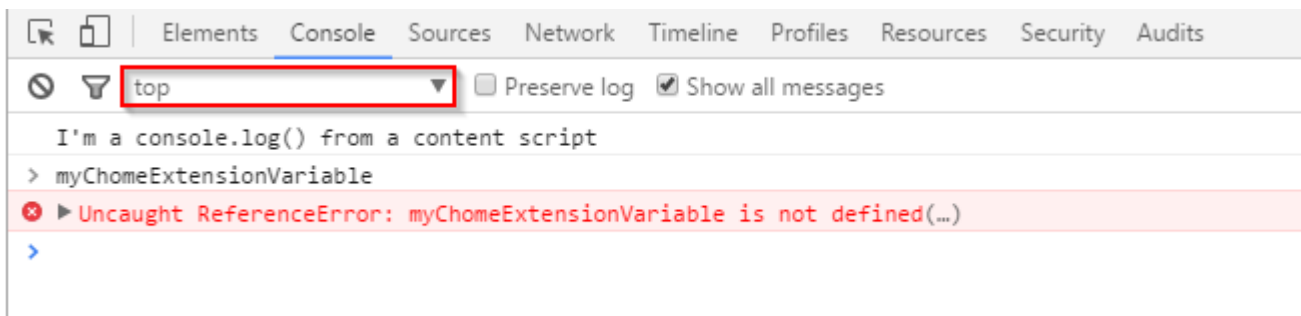
ID: abglihdcgbjkbhalmfachdimiapkfedl

Loaded from: C:\make_page_red

Inspect views: background page

Inspeccionar guiones de contenido:

Los scripts de contenido se ejecutan junto a los sitios web en los que se insertaron. Puede inspeccionar el script de contenido inspeccionando primero el sitio web donde se inserta el script de contenido. En la consola, podrá ver cualquier `console.log()` s generado por su extensión, pero no podrá cambiar ni inspeccionar las variables del script de contenido.



Elements Console Sources Network Timeline Profiles Resources Security Audits

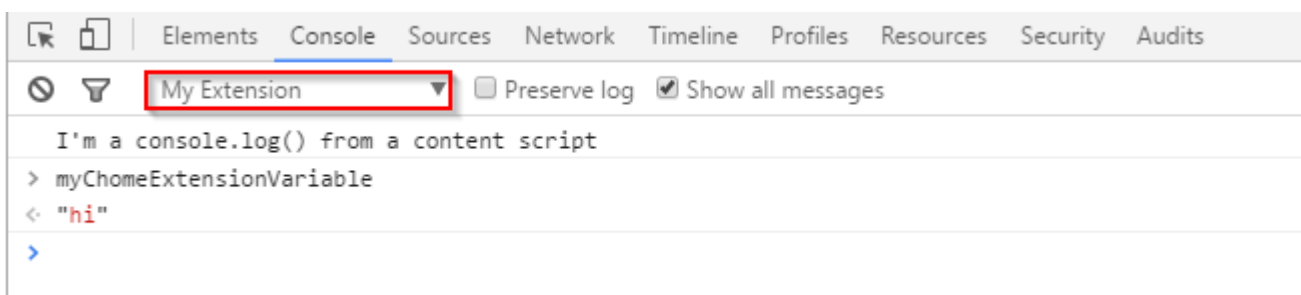
top Preserve log Show all messages

I'm a console.log() from a content script

> myChomeExtensionVariable

Uncaught ReferenceError: myChomeExtensionVariable is not defined(...)

Para solucionar este problema, debe hacer clic en el menú desplegable que normalmente se establece en 'top' y seleccionar su extensión de la lista de extensiones.



Elements Console Sources Network Timeline Profiles Resources Security Audits

My Extension Preserve log Show all messages

I'm a console.log() from a content script

> myChomeExtensionVariable

< "hi"

Desde allí tendrá acceso a las variables dentro de su extensión.

Lea Depuración de extensiones de Chrome en línea: <https://riptutorial.com/es/google-chrome-extension/topic/5730/depuracion-de-extensiones-de-chrome>

Capítulo 3: Guiones de contenido

Observaciones

Documentación oficial

- [Guiones de contenido](#)
- [Política de seguridad de contenido](#)> [Scripts de contenido](#)

Examples

Declarar scripts de contenido en el manifiesto.

Los scripts de contenido se pueden declarar en `manifest.json` para que siempre se inyecten en páginas que coincidan con un conjunto de [patrones de URL](#) .

Ejemplo mínimo

```
"content_scripts" : [  
  {  
    "js": ["content.js"],  
    "css": ["content.css"]  
    "matches": ["http://example.com/*"]  
  }  
]
```

Esta entrada de manifiesto indica a Chrome que inyecte un script de contenido `content.js` , junto con el archivo CSS `content.css` , después de cualquier navegación a una página que coincida con el [patrón de coincidencia](#) `http://example.com/*`

Las teclas `js` y `css` son opcionales: solo puede tener una o ambas dependiendo de lo que necesite.

`content_scripts` **clave** `content_scripts` es una matriz, y puede declarar varias definiciones de script de contenido:

```
"content_scripts" : [  
  {  
    "js": ["content.js"],  
    "matches": ["http://*.example.com/*"]  
  },  
  {  
    "js": ["something_else.js"],  
    "matches": ["http://*.example.org/*"]  
  }  
]
```

Tenga en cuenta que tanto `js` como `matches` son matrices, incluso si solo tiene una entrada.

Más opciones están disponibles en la [documentación oficial](#) y otros ejemplos.

Nota IMPORTANTE

Los scripts de contenido declarados en el manifiesto **solo se inyectarán en las nuevas navegaciones después de la carga de extensión** . No se inyectarán en las pestañas existentes. Esto también se aplica a las recargas de extensión durante el desarrollo y actualizaciones de extensión después del lanzamiento.

Si necesita asegurarse de que las pestañas abiertas actualmente están cubiertas, considere también realizar una inyección programática en el inicio.

Inyectar scripts de contenido desde una página de extensión

Si, en lugar de tener siempre inyectado un script de contenido basado en la URL, desea controlar directamente cuando se inyecta un script de contenido, puede usar la [inyección programática](#) .

Ejemplo mínimo

- JavaScript

```
chrome.tabs.executeScript({file: "content.js"});
```

- CSS

```
chrome.tabs.insertCSS({file: "content.css"});
```

Llamado desde una página de extensión (por ejemplo, fondo o ventana emergente), y suponiendo que tiene permiso para inyectar, esto ejecutará `content.js` o insertará `content.css` como un script de contenido en el marco superior de la pestaña actual.

Código en línea

Puede ejecutar código en línea en lugar de un archivo como un script de contenido:

```
var code = "console.log('This code will execute as a content script');";
chrome.tabs.executeScript({code: code});
```

Elegir la pestaña

Puede proporcionar una ID de pestaña (generalmente de otros métodos o mensajes de `chrome.tabs`) para ejecutar en una pestaña diferente a la activa actualmente.

```
chrome.tabs.executeScript({
  tabId: tabId,
  file: "content.js"
});
```

Hay más opciones disponibles en la [documentación de chrome.tabs.executeScript\(\)](#) y en otros ejemplos.

Permisos

El uso de `chrome.tabs.executeScript()` no requiere el permiso de "tabs" , pero requiere [permisos de host](#) para la URL de la página.

Comprobando errores

Si la inyección de script falla, se puede detectar en la devolución de llamada opcional:

```
chrome.tabs.executeScript({file: "content.js"}, function() {
  if(chrome.runtime.lastError) {
    console.error("Script injection failed: " + chrome.runtime.lastError.message);
  }
});
```

Múltiples scripts de contenido en el manifiesto.

Las mismas condiciones, múltiples scripts.

Si necesita inyectar varios archivos con todas las demás condiciones iguales, por ejemplo, para incluir una biblioteca, puede enumerarlos todos en la matriz "js" :

```
"content_scripts" : [
  {
    "js": ["library.js", "content.js"],
    "matches": ["http://*.example.com/*"]
  }
]
```

Asuntos de orden: `library.js` se ejecutará antes de `content.js` .

Mismos guiones, múltiples sitios.

Si necesita inyectar los mismos archivos en varios sitios, puede proporcionar múltiples patrones de coincidencia:


```
"matches": ["http://example.com/*", "http://example.org/*"]
```

Si necesita insertar básicamente cada página, puede usar patrones de concordancia amplia como `"*://*/*" (coincide con todas las páginas HTTP (S)) o "<all_urls>" (coincide con todas las páginas compatibles).`

Diferentes guiones o diferentes sitios.

"content_scripts" sección "content_scripts" es una matriz, por lo que uno puede definir más de un bloque de script de contenido:

```
"content_scripts" : [
  {
    "js": ["content.js"],
    "matches": ["http://*.example.com/*"]
  },
  {
    "js": ["something_else.js"],
    "matches": ["http://*.example.org/*"]
  }
]
```

Lea Guiones de contenido en línea: <https://riptutorial.com/es/google-chrome-extension/topic/2850/guiones-de-contenido>

Capítulo 4: Integración de herramientas de desarrollador

Examples

Indicaciones de punto de interrupción programáticas

Agregue la declaración del depurador en su script de contenido

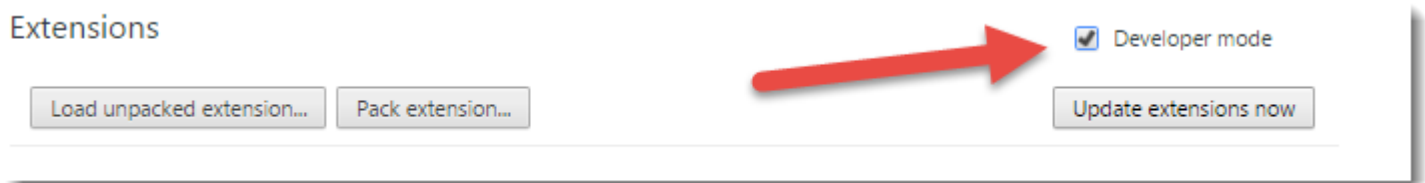
```
var foo = 1;  
debugger;  
  
foo = 2;
```

Abra la Herramienta del desarrollador en la página web donde se inyecta su secuencia de comandos de contenido para ver la pausa de ejecución del código en esas líneas.

Depuración de la página de fondo / script

El script de fondo es como cualquier otro código JavaScript. Puedes depurarlo usando las mismas herramientas que depurasas de otro código JavaScript en Chrome.

Para abrir Chrome Developer Tools, vaya a `chrome://extensions` y active el **modo de desarrollador** :



Ahora puedes depurar cualquier extensión que tenga una página de fondo o un script. Simplemente desplácese hasta la extensión que desea depurar y haga clic en el enlace de la **página de fondo** para inspeccionarlo.



Sugerencia: para volver a cargar la extensión, puede presionar `F5` dentro de la ventana de

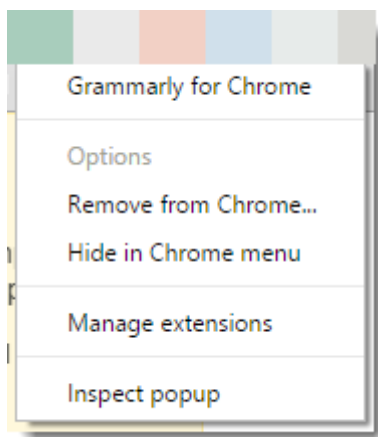
herramientas del desarrollador. Puede poner puntos de interrupción en el código de inicialización antes de volver a cargar.

Consejo: Al hacer clic con el botón derecho en el botón de acción de la extensión y seleccionar "Administrar extensiones", se abrirá la página de `chrome://extensions` desplazada a esa extensión.

Depurando la ventana emergente

Tienes 2 formas de depurar la ventana emergente. Ambas formas son mediante el uso de Chrome DevTools.

Opción 1: haga clic derecho en el botón de acción de la extensión y elija **Inspeccionar ventana emergente**



Opción 2: Abra la ventana emergente, directamente en su navegador como una pestaña.

Por ejemplo, si su ID de extensión es `abcdefghijklmnop`, y su archivo emergente html es `popup.html`. Ir a la dirección y navegar a:

```
chrome-extension://abcdefghijklmnop/popup.html
```

Ahora ves el popup en la pestaña regular. Y puedes presionar `F12` para abrir las herramientas de desarrollo.

Lea Integración de herramientas de desarrollador en línea: <https://riptutorial.com/es/google-chrome-extension/topic/5938/integracion-de-herramientas-de-desarrollador>

Capítulo 5: manifest.json

Observaciones

Documentación oficial

Formato de archivo manifiesto

Formato

El archivo de manifiesto está escrito en formato **JSON** (Notación de objetos de JavaScript).

Este formato difiere de las reglas más sueltas de escribir literales de objetos en código JavaScript. Entre las diferencias importantes:

- Cada nombre de clave y cadena literal **debe estar entre comillas dobles** .
 - **Correcto:** `"key": "value"`
 - **Incorrecto:** `key: "value", 'key': 'value'`
- **No se permiten comentarios** por el formato.
 - **Incorrecto:** `"key": "value" // This controls feature foo`
- Reglas estrictas de coma: **elementos separados por comas, sin comas colgantes** .
 - **Correcto:**

```
{
  "foo": "bar",
  "baz": "qux"
}
```

- **Incorrecto (falta una coma):**

```
{
  "foo": "bar"
  "baz": "qux"
}
```

- **Mal (coma que cuelga):**

```
{
  "foo": "bar",
  "baz": "qux",
}
```

```
}
```

Examples

Absoluto mínimo manifest.json

`manifest.json` proporciona información sobre la extensión, como los archivos más importantes y las capacidades que la extensión podría usar. Entre los campos de manifiesto soportados para extensiones, se requieren los siguientes **tres** .

```
{
  "manifest_version": 2,
  "name": "My Extension",
  "version": "1.0"
}
```

Obtención del manifiesto a partir del código de extensión.

`chrome.runtime.getManifest()` devuelve el manifiesto de la extensión en forma de un objeto analizado.

Este método funciona tanto en los scripts de contenido como en todas las páginas de extensión, no requiere permisos,

Ejemplo, obteniendo la cadena de versión de la extensión:

```
var version = chrome.runtime.getManifest().version;
```

Lea `manifest.json` en línea: <https://riptutorial.com/es/google-chrome-extension/topic/948/manifest-json>

Capítulo 6: Páginas de fondo

Examples

Declarando página de fondo en el manifiesto.

Hay dos formas de registrar una página de fondo en el manifiesto de extensión.

1. La propiedad `scripts`

En el caso común, una página de fondo no requiere ningún marcado HTML. Podemos registrar este tipo de páginas de fondo utilizando la propiedad de `scripts`.

En este caso, el sistema de extensión generará una página de fondo que incluye cada uno de los archivos enumerados en la propiedad de los `scripts`.

```
{
  ...
  "background": {
    "scripts": ["background1.js", "background2.js"],
    "persistent": true
  },
  ...
}
```

2. La propiedad de la `page`

En algunos casos, es posible que deseamos especificar HTML en la página de fondo, podemos lograr eso utilizando la propiedad de la `page`.

```
{
  ...
  "background": {
    "page": "background.html",
    "persistent": true
  },
  ...
}
```

`page scripts` VS

Es difícil decir cuál es mejor. podríamos usar la propiedad de la `page` y tener algunos elementos declarados en la página HTML para uso futuro. También podríamos crear dinámicamente dichos elementos en los `scripts` sin declarar explícitamente la página HTML. Todo depende de las necesidades reales.

Lea Páginas de fondo en línea: <https://riptutorial.com/es/google-chrome-extension/topic/4066/paginas-de-fondo>

Capítulo 7: Paso de mensajes

Observaciones

Documentación oficial

- [Paso de mensajes](#)
- [Mensajería nativa](#)
- [API `chrome.runtime`](#) (la mayoría de las funciones de mensajería y todos los eventos de mensajería)

Examples

Enviar una respuesta de forma asíncrona.

En el intento de enviar una respuesta de forma asincrónica desde la `chrome.runtime.onMessage` llamada `chrome.runtime.onMessage` podemos intentar este **código incorrecto** :

```
chrome.runtime.onMessage.addListener(function(request, sender, sendResponse) {
  $.ajax({
    url: 'https://www.google.com',
    method: 'GET',
    success: function(data) {
      // data won't be sent
      sendResponse(data);
    },
  });
});
```

Sin embargo, encontraríamos que los `data` nunca se envían. Esto sucede porque hemos puesto `sendResponse` dentro de una llamada ajax asíncrona, cuando se ejecuta el método de `success` , el canal de mensaje se ha cerrado.

La solución sería simple, siempre y cuando devolvamos explícitamente `return true;` al final de la devolución de llamada, lo que indica que deseamos enviar una respuesta de forma asíncrona, por lo que el canal de mensajes se mantendrá abierto al otro extremo (llamante) hasta que se ejecute `sendResponse` .

```
chrome.runtime.onMessage.addListener(function(request, sender, sendResponse) {
  $.ajax({
    url: 'https://www.google.com',
    method: 'GET',
    success: function(data) {
      // data would be sent successfully
      sendResponse(data);
    },
  });
});
```

```
    return true; // keeps the message channel open until `sendResponse` is executed
  });
```

Por supuesto, también se aplica a un `return` explícito de la devolución de llamada de `onMessage`:

```
chrome.runtime.onMessage.addListener(function(request, sender, sendResponse) {
  if (request.action == 'get') {
    $.ajax({
      url: 'https://www.google.com',
      method: 'GET',
      success: function(data) {
        // data would be sent successfully
        sendResponse(data);
      },
    });

    return true; // keeps the message channel open until `sendResponse` is executed
  }

  // do something synchronous, use sendResponse

  // normal exit closes the message channel
});
```

Lea Paso de mensajes en línea: <https://riptutorial.com/es/google-chrome-extension/topic/2185/paso-de-mensajes>

Capítulo 8: Portando a / desde Firefox

Observaciones

Si está utilizando una versión de *Firefox* antes de 48, también necesitará una clave adicional en `manifest.json` llamada `aplicaciones`:

```
"applications": {
  "gecko": {
    "id": "borderify@example.com",
    "strict_min_version": "42.0",
    "strict_max_version": "50.*",
    "update_url": "https://example.com/updates.json"
  }
}
```

[aplicaciones](#)

Nota:

[Firma de extensión](#) :

Con el lanzamiento de Firefox 48, la firma de extensión ya no se puede deshabilitar en el lanzamiento y las compilaciones del canal beta usando una preferencia. Tal como se describió cuando se anunció la firma de extensión, estamos publicando compilaciones especializadas que admiten esta preferencia para que los desarrolladores puedan continuar probando contra el código del que se generan las versiones beta y de versión.

Estado de las `WebExtensions` :

Las extensiones web están actualmente en un estado alfa experimental. Desde Firefox 46, puedes publicar `WebExtensions` para usuarios de Firefox, como cualquier otro complemento. Estamos apuntando para un primer lanzamiento estable en Firefox 48.

UPD : *Firefox* 48 lanzado el 02.08.2016.

Campo de golf:

[Estado de soporte de API](#) : la lista de API y su estado.

[Incompatibilidades de cromo](#)

[WebExtensions](#) - API de JavaScript, claves de `manifest.json`, tutoriales, etc.

Examples

Portando a través de WebExtensions

Antes de hablar sobre portar extensiones de *Firefox* de / a, uno debe saber qué es `WebExtensions` .

`WebExtensions` : es una plataforma que representa una API para crear extensiones de *Firefox* .

Utiliza la misma arquitectura de extensión que *Chromium* , como resultado, esta API es compatible de muchas maneras con la API en *Google Chrome* y *Opera* (*Opera*, que se basa en *Chromium*). En muchos casos, las extensiones desarrolladas para estos navegadores funcionarán en *Firefox* con algunos cambios o incluso sin ellos.

MDN [recomienda](#) utilizar `WebExtension` para nuevas extensiones:

En el futuro, `WebExtensions` será la forma recomendada para desarrollar complementos de *Firefox*, y otros sistemas quedarán obsoletos.

En vista de lo anterior, si desea portar extensiones a *Firefox* , debe saber cómo se escribió la extensión.

Las extensiones para *Firefox* pueden basarse en `WebExtension` , `Add-on SDK` o `XUL` .

Extensiones compatibles basadas en WebExtension

Al usar `WebExtension` , uno tiene que mirar a través de la lista de [incompatibilidades](#) , ya que algunas funciones son compatibles total o parcialmente, es decir, uno debe verificar su `manifest.json` .

También permite usar el mismo [espacio de nombres](#) :

En este momento, se puede acceder a todas las API a través del espacio de nombres `chrome.*`. Cuando comencemos a agregar nuestras propias API, esperamos agregarlas al navegador. * Espacio de nombres. Los desarrolladores podrán utilizar la detección de características para determinar si una API está disponible en el navegador. *

Una extensión simple que puede funcionar en Firefox y Google Chrome.

`manifest.json` :

```
{
  "manifest_version": 2,
  "name": "StackMirror",
```

```

"version": "1.0",

"description": "Mirror reflection of StackOverflow sites",

"icons": {
  "48": "icon/myIcon-48.png"
},

"page_action": {
  "default_icon": "icon/myIcon-48.png"
},

"background": {
  "scripts" : ["js/background/script.js"],
  "persistent": false
},

"permissions": ["tabs", "*/**/*.stackoverflow.com/*"]
}

```

guión de background :

```

function startScript(tabId, changeInfo, tab) {

  if (tab.url.indexOf("stackoverflow.com") > -1) {

    chrome.tabs.executeScript(tabId,

      {code: 'document.body.style.transform = "scaleX(-1)";'}, function () {

        if (!chrome.runtime.lastError) {

          chrome.pageAction.show(tabId);

        }

      });

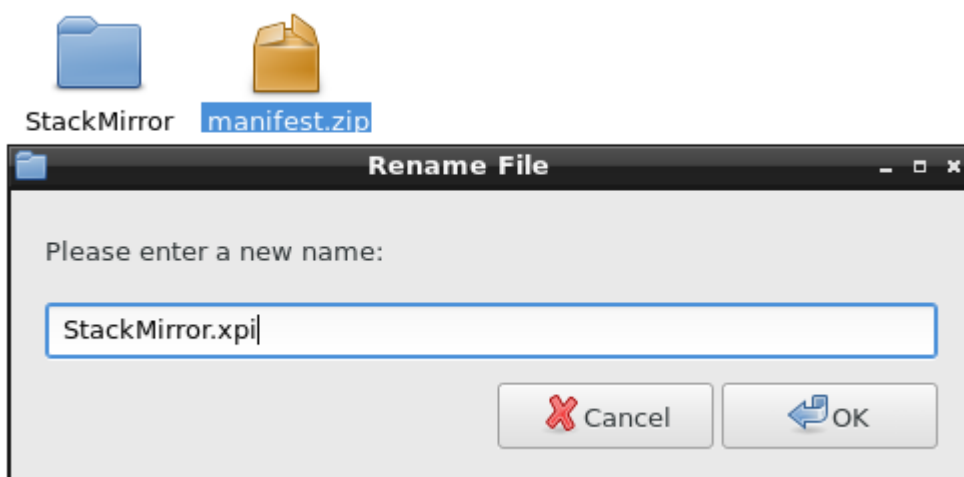
  }

}

chrome.tabs.onUpdated.addListener(startScript);

```

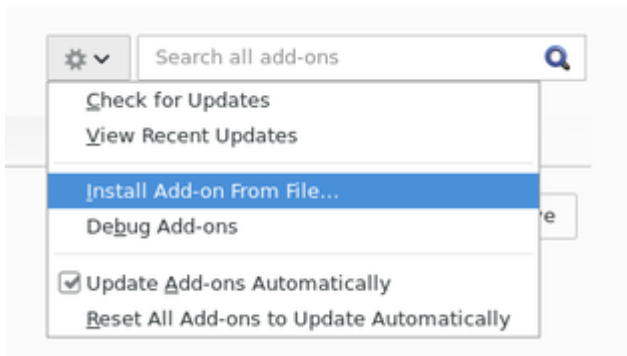
Empaque el proyecto como archivo zip estándar, pero con extensiones .xpi .



Entonces, tienes que cargar la extensión en *Firefox* .

Abra la página `about:addons` , accesible a través de **Menú> Complementos** .

Haga clic en el botón **Herramientas para todos los complementos** .



Cuando se cargue la extensión, la página `about:addons` se verá así:



Las instrucciones para cargar la extensión en Google Chrome se encuentran en otro tema: [cómo comenzar con las extensiones de Chrome](#) .

El resultado de la operación de extensión será el mismo en ambos navegadores (*Firefox / Google Chrome*):



Si el complemento actual se basa en el SDK de complemento o XUL

Cuando la extensión que se va a portar se basa en `Add-on SDK` uno tiene que mirar a través de la tabla de comparación para `Add-on SDK => WebExtensions` , porque estas tecnologías tienen características similares, pero difieren en la implementación. Cada sección de la tabla describe el

equivalente de `Add-on SDK` para `WebExtension` .

[Comparación con el complemento SDK](#)

Un enfoque similar y para las extensiones XUL.

[Comparación con las extensiones XUL / XPCOM](#)

Lea [Portando a / desde Firefox en línea](#): <https://riptutorial.com/es/google-chrome-extension/topic/5731/portando-a---desde-firefox>

Creditos

S. No	Capítulos	Contributors
1	Empezando con google-chrome-extension	Aminadav , Community , Deliaz , Haibara Ai , ScientiaEtVeritas , Xan
2	Depuración de extensiones de Chrome	Marc Guiselin
3	Guiones de contenido	Haibara Ai , Xan
4	Integración de herramientas de desarrollador	Aminadav , Paul Sweatte , Xan
5	manifest.json	Haibara Ai , Xan
6	Páginas de fondo	Haibara Ai , Noam Hacker
7	Paso de mensajes	Haibara Ai , wOxxOm , Xan
8	Portando a / desde Firefox	Deliaz , UserName