chrome.runtime

Description:	Use the `chrome.runtime` API to retrieve the background page, return details about the manifest, and listen for and respond to events in the app or extension lifecycle. You can also use this API to convert the relative path of URLs to fully-qualified URLs.
Availability:	Since Chrome 27.
Content Scripts:	**connect** , **getManifest** , **getURL** , **id** , **onConnect** , **onMessage** and **sendMessage** are supported. **Learn more**
Learn More:	**Event Pages**

Summary

Types
Port
MessageSender
PlatformOs
PlatformArch
PlatformNaclArch
PlatformInfo
RequestUpdateCheckStatus
OnInstalledReason
OnRestartRequiredReason
Properties
lastError
id
Methods
getBackgroundPage – `chrome.runtime.getBackgroundPage(function callback)`
openOptionsPage – `chrome.runtime.openOptionsPage(function callback)`
getManifest – `object chrome.runtime.getManifest()`
getURL – `string chrome.runtime.getURL(string path)`
setUninstallURL – `chrome.runtime.setUninstallURL(string url, function callback)`
reload – `chrome.runtime.reload()`
requestUpdateCheck – `chrome.runtime.requestUpdateCheck(function callback)`
restart – `chrome.runtime.restart()`
restartAfterDelay – `chrome.runtime.restartAfterDelay(integer seconds, function callback)`
connect – `Port chrome.runtime.connect(string extensionId, object connectInfo)`
connectNative – `Port chrome.runtime.connectNative(string application)`
sendMessage – `chrome.runtime.sendMessage(string extensionId, any message, object options, function`

responseCallback)
sendNativeMessage − `chrome.runtime.sendNativeMessage(string application, object message, function responseCallback)`
getPlatformInfo − `chrome.runtime.getPlatformInfo(function callback)`
getPackageDirectoryEntry − `chrome.runtime.getPackageDirectoryEntry(function callback)`
Events
onStartup
onInstalled
onSuspend
onSuspendCanceled
onUpdateAvailable
onBrowserUpdateAvailable
onConnect
onConnectExternal
onMessage
onMessageExternal
onRestartRequired

Types

Port

An object which allows two way communication with other pages. See **Long-lived connections** for more information.

properties		
string	name	The name of the port, as specified in the call to **runtime.connect**.
function	disconnect	Immediately disconnect the port. Calling `disconnect()` on an already-disconnected port has no effect. When a port is disconnected, no new events will be dispatched to this port.
object	onDisconnect	Fired when the port is disconnected from the other end(s). **runtime.lastError** may be set if the port was disconnected by an error. If the port is closed via **disconnect**, then this event is *only* fired on the other end. This event is fired at most once (see also **Port lifetime**). The first and only parameter to the event handler is this disconnected port.
object	onMessage	This event is fired when **postMessage** is called by the other end of the port. The first parameter is the message, the second parameter is the port that received the message.
function	postMessage	Send a message to the other end of the port. If the port is disconnected, an error is thrown. **Parameters**

		any | message | Since Chrome 52. The message to send. This object should be JSON-ifiable.
MessageSender	(optional) sender	This property will **only** be present on ports passed to **onConnect** / **onConnectExternal** listeners.

MessageSender

An object containing information about the script context that sent a message or request.

properties		
tabs.Tab	(optional) tab	The **tabs.Tab** which opened the connection, if any. This property will **only** be present when the connection was opened from a tab (including content scripts), and **only** if the receiver is an extension, not an app.
integer	(optional) frameId	Since Chrome 41. The **frame** that opened the connection. 0 for top-level frames, positive for child frames. This will only be set when `tab` is set.
string	(optional) id	The ID of the extension or app that opened the connection, if any.
string	(optional) url	Since Chrome 28. The URL of the page or frame that opened the connection. If the sender is in an iframe, it will be iframe's URL not the URL of the page which hosts it.
string	(optional) tlsChannelId	Since Chrome 32. The TLS channel ID of the page or frame that opened the connection, if requested by the extension or app, and if available.

PlatformOs

The operating system chrome is running on.

Enum
`"mac"`, `"win"`, `"android"`, `"cros"`, `"linux"`, or `"openbsd"`

PlatformArch

The machine's processor architecture.

Enum
`"arm"`, `"x86-32"`, `"x86-64"`, `"mips"`, or `"mips64"`

PlatformNaclArch

The native client architecture. This may be different from arch on some platforms.

Enum
`"arm"`, `"x86-32"`, `"x86-64"`, `"mips"`, or `"mips64"`

PlatformInfo

Since Chrome 36.

An object containing information about the current platform.

properties		
PlatformOs	os	The operating system chrome is running on.
PlatformArch	arch	The machine's processor architecture.
PlatformNaclArch	nacl_arch	The native client architecture. This may be different from arch on some platforms.

RequestUpdateCheckStatus

Result of the update check.

Enum
`"throttled"`, `"no_update"`, or `"update_available"`

OnInstalledReason

The reason that this event is being dispatched.

Enum
`"install"`, `"update"`, `"chrome_update"`, or `"shared_module_update"`

OnRestartRequiredReason

The reason that the event is being dispatched. 'app_update' is used when the restart is needed because the application is updated to a newer version. 'os_update' is used when the restart is needed because the browser/OS is updated to a newer version. 'periodic' is used when the system runs for more than the permitted uptime set in the enterprise policy.

Enum

"app_update", "os_update", or "periodic"

Properties

object	`chrome.runtime.lastError`	This will be defined during an API method callback if there was an error **Properties** string | (optional) message | Details about the error which occurred.
string	`chrome.runtime.id`	The ID of the extension/app.

Methods

getBackgroundPage

```
chrome.runtime.getBackgroundPage(function callback)
```

Retrieves the JavaScript 'window' object for the background page running inside the current extension/app. If the background page is an event page, the system will ensure it is loaded before calling the callback. If there is no background page, an error is set.

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function(Window backgroundPage) {...};` Window | (optional) backgroundPage | The JavaScript 'window' object for the background page.

openOptionsPage

```
chrome.runtime.openOptionsPage(function callback)
```

Since Chrome 42.

Open your Extension's options page, if possible.

The precise behavior may depend on your manifest's `options_ui` or `options_page` key, or what Chrome happens to support at the time. For example, the page may be opened in a new tab, within chrome://extensions, within an App, or it may just focus an open options page. It will never cause the caller page to reload.

If your Extension does not declare an options page, or Chrome failed to create one for some other reason, the callback will set **lastError**.

Parameters		
function	(optional) callback	If you specify the *callback* parameter, it should be a function that looks like this: `function() {...};`

getManifest

```
object chrome.runtime.getManifest()
```

Returns details about the app or extension from the manifest. The object returned is a serialization of the full **manifest file**.

Returns

The manifest details.

getURL

```
string chrome.runtime.getURL(string path)
```

Converts a relative path within an app/extension install directory to a fully-qualified URL.

Parameters		
string	path	A path to a resource within an app/extension expressed relative to its install directory.

setUninstallURL

```
chrome.runtime.setUninstallURL(string url, function callback)
```

Since Chrome 41.

Sets the URL to be visited upon uninstallation. This may be used to clean up server-side data, do analytics, and implement surveys. Maximum 255 characters.

Parameters		
string	url	Since Chrome 34. URL to be opened after the extension is uninstalled. This URL must have an http: or https: scheme. Set an empty string to not open a new tab upon uninstallation.
function	(optional) callback	Called when the uninstall URL is set. If the given URL is invalid, **runtime.lastError** will be set. If you specify the *callback* parameter, it should be a function that looks like this: `function() {...};`

reload

```
chrome.runtime.reload()
```

Reloads the app or extension. This method is not supported in kiosk mode. For kiosk mode, use chrome.runtime.restart() method.

requestUpdateCheck

```
chrome.runtime.requestUpdateCheck(function callback)
```

Requests an immediate update check be done for this app/extension.

Important: Most extensions/apps should **not** use this method, since chrome already does automatic checks every few hours, and you can listen for the **runtime.onUpdateAvailable** event without needing to call requestUpdateCheck.

This method is only appropriate to call in very limited circumstances, such as if your extension/app talks to a backend service, and the backend service has determined that the client extension/app version is very far out of date and you'd like to prompt a user to update. Most other uses of requestUpdateCheck, such as calling it unconditionally based on a repeating timer, probably only serve to waste client, network, and server resources.

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function( RequestUpdateCheckStatus status, object details) {...};`

RequestUpdateCheckStatus	status	Result of the update check.
object	(optional) details	If an update is available, this contains more information about the available update.

string	version	The version of the available update.

restart

```
chrome.runtime.restart()
```

Since Chrome 32.

Restart the ChromeOS device when the app runs in kiosk mode. Otherwise, it's no-op.

restartAfterDelay

```
chrome.runtime.restartAfterDelay(integer seconds, function callback)
```

Since Chrome 53.

Restart the ChromeOS device when the app runs in kiosk mode after the given seconds. If called again before the time ends, the reboot will be delayed. If called with a value of -1, the reboot will be cancelled. It's a no-op in non-kiosk mode. It's only allowed to be called repeatedly by the first extension to invoke this API.

Parameters

integer	seconds	Time to wait in seconds before rebooting the device, or -1 to cancel a scheduled reboot.
function	(optional) callback	A callback to be invoked when a restart request was successfully rescheduled. If you specify the *callback* parameter, it should be a function that looks like this: `function() {...};`

connect

```
Port chrome.runtime.connect(string extensionId, object connectInfo)
```

Attempts to connect to connect listeners within an extension/app (such as the background page), or other extensions/apps. This is useful for content scripts connecting to their extension processes, inter-app/extension communication, and **web messaging**. Note that this does not connect to any listeners in a content script. Extensions may connect to content scripts embedded in tabs via **tabs.connect**.

Parameters		
string	(optional) extensionId	The ID of the extension or app to connect to. If omitted, a connection will be attempted with your own extension. Required if sending messages from a web page for **web messaging**.
object	(optional) connectInfo	

connectInfo:

string	(optional) name	Will be passed into onConnect for processes that are listening for the connection event.
boolean	(optional) includeTlsChannelId	Since Chrome 32. Whether the TLS channel ID will be passed into onConnectExternal for processes that are listening for the connection event.

connectNative

```
Port chrome.runtime.connectNative(string application)
```

Since Chrome 28.

Connects to a native application in the host machine. See **Native Messaging** for more information.

Parameters		
string	application	The name of the registered application to connect to.

sendMessage

```
chrome.runtime.sendMessage(string extensionId, any message, object options, function
responseCallback)
```

Sends a single message to event listeners within your extension/app or a different extension/app. Similar to **runtime.connect** but only sends a single message, with an optional response. If sending to your extension, the

runtime.onMessage event will be fired in every frame of your extension (except for the sender's frame), or **runtime.onMessageExternal**, if a different extension. Note that extensions cannot send messages to content scripts using this method. To send messages to content scripts, use **tabs.sendMessage**.

Parameters		
string	(optional) extensionId	The ID of the extension/app to send the message to. If omitted, the message will be sent to your own extension/app. Required if sending messages from a web page for **web messaging**.
any	message	The message to send. This message should be a JSON-ifiable object.
object	(optional) options	Since Chrome 32. boolean | (optional) includeTlsChannelId | Whether the TLS channel ID will be passed into onMessageExternal for processes that are listening for the connection event.
function	(optional) responseCallback	If you specify the *responseCallback* parameter, it should be a function that looks like this: `function(any response) {...};` any | response | The JSON response object sent by the handler of the message. If an error occurs while connecting to the extension, the callback will be called with no arguments and **runtime.lastError** will be set to the error message.

sendNativeMessage

```
chrome.runtime.sendNativeMessage(string application, object message, function
responseCallback)
```

Since Chrome 28.

Send a single message to a native application.

Parameters		
string	application	The name of the native messaging host.
object	message	The message that will be passed to the native messaging host.
function	(optional) responseCallback	If you specify the *responseCallback* parameter, it should be a function that looks like this: `function(any response) {...};` any | response | The response message sent by the native messaging host. If an error occurs while connecting to the native messaging host, the callback will be called with no arguments and **runtime.lastError** will be set to the error message.

getPlatformInfo

```
chrome.runtime.getPlatformInfo(function callback)
```

Since Chrome 29.

Returns information about the current platform.

Parameters		
function	callback	Called with results The *callback* parameter should be a function that looks like this: `function( PlatformInfo platformInfo) {...};` **PlatformInfo** | platformInfo

getPackageDirectoryEntry

```
chrome.runtime.getPackageDirectoryEntry(function callback)
```

Since Chrome 29.

Returns a DirectoryEntry for the package directory.

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function(DirectoryEntry directoryEntry) {...};` DirectoryEntry | directoryEntry

Events

onStartup

Fired when a profile that has this extension installed first starts up. This event is not fired when an incognito profile is started, even if this extension is operating in 'split' incognito mode.

addListener

```
chrome.runtime.onStartup.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function() {...};`

onInstalled

Fired when the extension is first installed, when the extension is updated to a new version, and when Chrome is updated to a new version.

addListener

```
chrome.runtime.onInstalled.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function(object details) {...};`

object	details	**OnInstalledReason**	reason	The reason that this event is being dispatched.
		string	(optional) previousVersion	Indicates the previous version of the extension, which has just been updated. This is present only if 'reason' is 'update'.
		string	(optional) id	Since Chrome 29. Indicates the ID of the imported shared module extension which updated. This is present only if 'reason' is 'shared_module_update'.

onSuspend

Sent to the event page just before it is unloaded. This gives the extension opportunity to do some clean up. Note that since the page is unloading, any asynchronous operations started while handling this event are not guaranteed to complete. If more activity for the event page occurs before it gets unloaded the onSuspendCanceled event will be sent and the page won't be unloaded.

addListener

```
chrome.runtime.onSuspend.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function() {...};`

onSuspendCanceled

Sent after onSuspend to indicate that the app won't be unloaded after all.

addListener

```
chrome.runtime.onSuspendCanceled.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function() {...};`

onUpdateAvailable

Fired when an update is available, but isn't installed immediately because the app is currently running. If you do nothing, the update will be installed the next time the background page gets unloaded, if you want it to be installed sooner you can explicitly call chrome.runtime.reload(). If your extension is using a persistent background page, the background page of course never gets unloaded, so unless you call chrome.runtime.reload() manually in response to this event the update will not get installed until the next time chrome itself restarts. If no handlers are listening for this event, and your extension has a persistent background page, it behaves as if chrome.runtime.reload() is called in response to this event.

addListener

```
chrome.runtime.onUpdateAvailable.addListener(function callback)
```

Parameters							
function	callback	The *callback* parameter should be a function that looks like this: `function(object details) {...};`					
		object	details	The manifest details of the available update.			
				string	version	The version number of the available update.	

onBrowserUpdateAvailable

Deprecated since Chrome 33. Please use **runtime.onRestartRequired**.

Fired when a Chrome update is available, but isn't installed immediately because a browser restart is required.

addListener

```
chrome.runtime.onBrowserUpdateAvailable.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function() {...};`

onConnect

Fired when a connection is made from either an extension process or a content script (by **runtime.connect**).

addListener

```
chrome.runtime.onConnect.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function( Port port) {...};`

Port	port	

onConnectExternal

Fired when a connection is made from another extension (by **runtime.connect**).

addListener

```
chrome.runtime.onConnectExternal.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function( Port port) {...};`

Port	port	

onMessage

Fired when a message is sent from either an extension process (by **runtime.sendMessage**) or a content script (by **tabs.sendMessage**).

addListener

```
chrome.runtime.onMessage.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function(any message, MessageSender sender, function sendResponse) {...};`

any	(optional) message	The message sent by the calling script.
MessageSender	sender	
function	sendResponse	Function to call (at most once) when you have a response. The argument should be any JSON-ifiable object. If you have more than one `onMessage` listener in the same document, then only one may send a response. This function becomes invalid when the event listener returns, **unless you return true** from the event listener to

				indicate you wish to send a response asynchronously (this will keep the message channel open to the other end until `sendResponse` is called). The *sendResponse* parameter should be a function that looks like this: `function() {...};`

onMessageExternal

Fired when a message is sent from another extension/app (by **runtime.sendMessage**). Cannot be used in a content script.

addListener

```
chrome.runtime.onMessageExternal.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this: `function(any message, MessageSender sender, function sendResponse) {...};`

any	(optional) message	The message sent by the calling script.
MessageSender	sender	
function	sendResponse	Function to call (at most once) when you have a response. The argument should be any JSON-ifiable object. If you have more than one `onMessage` listener in the same document, then only one may send a response. This function becomes invalid when the event listener returns, **unless you return true** from the event listener to indicate you wish to send a response asynchronously (this will keep the message channel open to the other end until `sendResponse` is called). The *sendResponse* parameter should be a function that looks like this: `function() {...};`

onRestartRequired

Since Chrome 29.

Fired when an app or the device that it runs on needs to be restarted. The app should close all its windows at its earliest convenient time to let the restart to happen. If the app does nothing, a restart will be enforced after a 24-hour grace period has passed. Currently, this event is only fired for Chrome OS kiosk apps.

addListener

```
chrome.runtime.onRestartRequired.addListener(function callback)
```

Parameters

function	callback	The *callback* parameter should be a function that looks like this: `function( OnRestartRequiredReason reason) {...};` **OnRestartRequiredReason** | reason | The reason that the event is being dispatched.