chrome.tabs

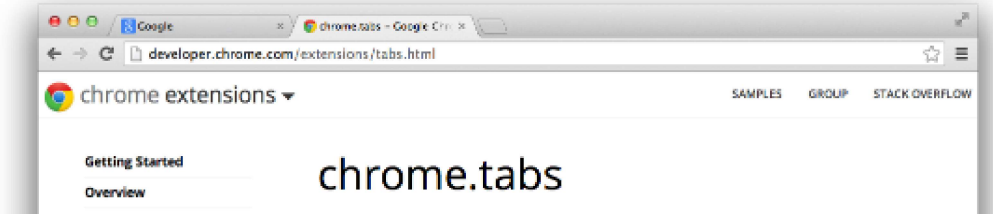
Description:	Use the `chrome.tabs` API to interact with the browser's tab system. You can use this API to create, modify, and rearrange tabs in the browser.
Availability:	Since Chrome 20.
Permissions:	The majority of the `chrome.tabs` API can be used without declaring any permission. However, the `"tabs"` permission is required in order to populate the **`url`**, **`title`**, and **`favIconUrl`** properties of **`Tab`**.

Manifest

You can use most `chrome.tabs` methods and events without declaring any permissions in the extension's **manifest** file. However, if you require access to the **`url`**, **`title`**, or **`favIconUrl`** properties of **`tabs.Tab`**, you must declare the `"tabs"` permission in the manifest, as shown below:

```
{
  "name": "My extension",
  ...
  "permissions": [
    "tabs"
  ],
  ...
}
```

Examples



You can find simple examples of manipulating tabs with the `chrome.tabs` API in the **examples/api/tabs** directory. For other examples and for help in viewing the source code, see **Samples**.

Summary

Types
MutedInfoReason
MutedInfo
Tab
ZoomSettingsMode
ZoomSettingsScope
ZoomSettings
TabStatus
WindowType
Properties
TAB_ID_NONE
Methods
get – chrome.tabs.**get**(integer tabId, **function** callback)
getCurrent – chrome.tabs.getCurrent(**function** callback)
connect – runtime.**Port** chrome.tabs.connect(integer tabId, **object** connectInfo)
sendRequest – chrome.tabs.sendRequest(integer tabId, any request, **function** responseCallback)
sendMessage – chrome.tabs.sendMessage(integer tabId, any message, **object** options, **function** responseCallback)
getSelected – chrome.tabs.getSelected(integer windowId, **function** callback)
getAllInWindow – chrome.tabs.getAllInWindow(integer windowId, **function** callback)
create – chrome.tabs.create(**object** createProperties, **function** callback)
duplicate – chrome.tabs.duplicate(integer tabId, **function** callback)
query – chrome.tabs.query(**object** queryInfo, **function** callback)
highlight – chrome.tabs.highlight(**object** highlightInfo, **function** callback)
update – chrome.tabs.update(integer tabId, **object** updateProperties, **function** callback)
move – chrome.tabs.move(integer **or** array of integer tabIds, **object** moveProperties, **function** callback)
reload – chrome.tabs.reload(integer tabId, **object** reloadProperties, **function** callback)
remove – chrome.tabs.remove(integer **or** array of integer tabIds, **function** callback)
detectLanguage – chrome.tabs.detectLanguage(integer tabId, **function** callback)
captureVisibleTab – chrome.tabs.captureVisibleTab(integer windowId, **object** options, **function** callback)
executeScript – chrome.tabs.executeScript(integer tabId, **object** details, **function** callback)
insertCSS – chrome.tabs.insertCSS(integer tabId, **object** details, **function** callback)
setZoom – chrome.tabs.setZoom(integer tabId, **double** zoomFactor, **function** callback)
getZoom – chrome.tabs.getZoom(integer tabId, **function** callback)
setZoomSettings – chrome.tabs.setZoomSettings(integer tabId, **ZoomSettings** zoomSettings,

`function callback)`
getZoomSettings – `chrome.tabs.getZoomSettings(integer tabId, function callback)`
discard – `chrome.tabs.discard(integer tabId, function callback)`
Events
onCreated
onUpdated
onMoved
onSelectionChanged
onActiveChanged
onActivated
onHighlightChanged
onHighlighted
onDetached
onAttached
onRemoved
onReplaced
onZoomChange

Types

MutedInfoReason

An event that caused a muted state change.

Enum

`"user"`
A user input action has set/overridden the muted state.
`"capture"`
Tab capture started, forcing a muted state change.
`"extension"`
An extension, identified by the extensionId field, set the muted state.

MutedInfo

Since Chrome 46.

Tab muted state and the reason for the last state change.

properties		
boolean	muted	Whether the tab is prevented from playing sound (but hasn't necessarily recently produced sound). Equivalent to whether the muted audio indicator is showing.
MutedInfoReason	(optional) reason	The reason the tab was muted or unmuted. Not set if the tab's mute state has never been changed.
string	(optional) extensionId	The ID of the extension that changed the muted state. Not set if an extension was not the reason the muted state last changed.

Tab

properties		
integer	(optional) id	The ID of the tab. Tab IDs are unique within a browser session. Under some circumstances a Tab may not be assigned an ID, for example when querying foreign tabs using the **sessions** API, in which case a session ID may be present. Tab ID can also be set to chrome.tabs.TAB_ID_NONE for apps and devtools windows.
integer	index	The zero-based index of the tab within its window.
integer	windowId	The ID of the window the tab is contained within.
integer	(optional) openerTabId	The ID of the tab that opened this tab, if any. This property is only present if the opener tab still exists.
boolean	selected	**Deprecated** since Chrome 33. Please use **tabs.Tab.highlighted**. Whether the tab is selected.
boolean	highlighted	Whether the tab is highlighted.
boolean	active	Whether the tab is active in its window. (Does not necessarily mean the window is focused.)
boolean	pinned	Whether the tab is pinned.
boolean	(optional) audible	Since Chrome 45. Whether the tab has produced sound over the past couple of seconds (but it might not be heard if also muted). Equivalent to whether the speaker audio indicator is showing.

boolean	discarded	Since Chrome 54. Whether the tab is discarded. A discarded tab is one whose content has been unloaded from memory, but is still visible in the tab strip. Its content gets reloaded the next time it's activated.
boolean	autoDiscardable	Since Chrome 54. Whether the tab can be discarded automatically by the browser when resources are low.
MutedInfo	(optional) mutedInfo	Since Chrome 46. Current tab muted state and the reason for the last state change.
string	(optional) url	The URL the tab is displaying. This property is only present if the extension's manifest includes the `"tabs"` permission.
string	(optional) title	The title of the tab. This property is only present if the extension's manifest includes the `"tabs"` permission.
string	(optional) favIconUrl	The URL of the tab's favicon. This property is only present if the extension's manifest includes the `"tabs"` permission. It may also be an empty string if the tab is loading.
string	(optional) status	Either *loading* or *complete*.
boolean	incognito	Whether the tab is in an incognito window.
integer	(optional) width	Since Chrome 31. The width of the tab in pixels.
integer	(optional) height	Since Chrome 31. The height of the tab in pixels.
string	(optional) sessionId	Since Chrome 31. The session ID used to uniquely identify a Tab obtained from the **sessions** API.

ZoomSettingsMode

Defines how zoom changes are handled, i.e. which entity is responsible for the actual scaling of the page; defaults to `automatic`.

Enum

`"automatic"`

Zoom changes are handled automatically by the browser.

`"manual"`

Overrides the automatic handling of zoom changes. The `onZoomChange` event will still be dispatched, and it is the responsibility of the extension to listen for this event and manually scale the page. This mode does not support `per-origin` zooming, and will thus ignore the `scope` zoom setting and assume `per-tab`.

`"disabled"`

Disables all zooming in the tab. The tab will revert to the default zoom level, and all attempted zoom changes will be ignored.

ZoomSettingsScope

Defines whether zoom changes will persist for the page's origin, or only take effect in this tab; defaults to `per-origin` when in `automatic` mode, and `per-tab` otherwise.

Enum

`"per-origin"`

Zoom changes will persist in the zoomed page's origin, i.e. all other tabs navigated to that same origin will be zoomed as well. Moreover, `per-origin` zoom changes are saved with the origin, meaning that when navigating to other pages in the same origin, they will all be zoomed to the same zoom factor. The `per-origin` scope is only available in the `automatic` mode.

`"per-tab"`

Zoom changes only take effect in this tab, and zoom changes in other tabs will not affect the zooming of this tab. Also, `per-tab` zoom changes are reset on navigation; navigating a tab will always load pages with their `per-origin` zoom factors.

ZoomSettings

Since Chrome 38.

Defines how zoom changes in a tab are handled and at what scope.

properties

ZoomSettingsMode	(optional) mode	Defines how zoom changes are handled, i.e. which entity is responsible for the actual scaling of the page; defaults to `automatic`.
ZoomSettingsScope	(optional) scope	Defines whether zoom changes will persist for the page's origin, or only take effect in this tab; defaults to `per-origin` when in `automatic` mode, and `per-tab` otherwise.

double	(optional) defaultZoomFactor	Since Chrome 43. Used to return the default zoom level for the current tab in calls to tabs.getZoomSettings.

TabStatus

Whether the tabs have completed loading.

Enum
`"loading"`, or `"complete"`

WindowType

The type of window.

Enum
`"normal"`, `"popup"`, `"panel"`, `"app"`, or `"devtools"`

Properties

`-1`	`chrome.tabs.TAB_ID_NONE`	Since Chrome 46. An ID which represents the absence of a browser tab.

Methods

get

```
chrome.tabs.get(integer tabId, function callback)
```

Retrieves details about the specified tab.

Parameters	
integer	tabId

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

getCurrent

`chrome.tabs.getCurrent(function callback)`

Gets the tab that this script call is being made from. May be undefined if called from a non-tab context (for example: a background page or popup view).

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

connect

`runtime.Port chrome.tabs.connect(integer tabId, object connectInfo)`

Connects to the content script(s) in the specified tab. The **runtime.onConnect** event is fired in each content script running in the specified tab for the current extension. For more details, see **Content Script Messaging**.

Parameters				
integer	tabId			
object	(optional) connectInfo	string	(optional) name	Will be passed into onConnect for content scripts that are listening for the connection event.
		integer	(optional) frameId	Since Chrome 41. Open a port to a specific **frame** identified by `frameId` instead of all frames in the tab.

sendRequest

`chrome.tabs.sendRequest(integer tabId, any request, function responseCallback)`

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

Sends a single request to the content script(s) in the specified tab, with an optional callback to run when a response is sent back. The **extension.onRequest** event is fired in each content script running in the specified tab for the current extension.

Parameters		
integer	tabId	
any	request	
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 request. If an error occurs while connecting to the specified tab, the callback will be called with no arguments and **runtime.lastError** will be set to the error message.

sendMessage

`chrome.tabs.sendMessage(integer tabId, any message, object options, function responseCallback)`

Sends a single message to the content script(s) in the specified tab, with an optional callback to run when a response is sent back. The **runtime.onMessage** event is fired in each content script running in the specified tab for the current extension.

Parameters		
integer	tabId	
any	message	The message to send. This message should be a JSON-ifiable object.
object	(optional) options	Since Chrome 41. integer — (optional) frameId — Send a message to a specific **frame** identified by `frameId` instead of all frames in the tab.
function	(optional)	If you specify the *responseCallback* parameter, it should be a

	responseCallback	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 specified tab, the callback will be called with no arguments and **runtime.lastError** will be set to the error message.

getSelected

`chrome.tabs.getSelected(integer windowId, function callback)`

Deprecated since Chrome 33. Please use **tabs.query** `{active: true}`.

Gets the tab that is selected in the specified window.

Parameters		
integer	(optional) windowId	Defaults to the **current window**.
function	callback	The *callback* parameter should be a function that looks like this: `function( Tab tab) {...};` **Tab**: tab

getAllInWindow

`chrome.tabs.getAllInWindow(integer windowId, function callback)`

Deprecated since Chrome 33. Please use **tabs.query** `{windowId: windowId}`.

Gets details about all tabs in the specified window.

Parameters		
integer	(optional) windowId	Defaults to the **current window**.
function	callback	The *callback* parameter should be a function that looks like this: `function(array of Tab tabs) {...};`

array of **Tab**	tabs	

create

```
chrome.tabs.create(object createProperties, function callback)
```

Creates a new tab.

Parameters

object	createProperties	

integer	(optional) windowId	The window to create the new tab in. Defaults to the **current window**.
integer	(optional) index	The position the tab should take in the window. The provided value will be clamped to between zero and the number of tabs in the window.
string	(optional) url	The URL to navigate the tab to initially. Fully-qualified URLs must include a scheme (i.e. 'http://www.google.com', not 'www.google.com'). Relative URLs will be relative to the current page within the extension. Defaults to the New Tab Page.
boolean	(optional) active	Whether the tab should become the active tab in the window. Does not affect whether the window is focused (see **windows.update**). Defaults to *true*.
boolean	(optional) selected	**Deprecated** since Chrome 33. Please use *active*. Whether the tab should become the selected tab in the window. Defaults to *true*
boolean	(optional) pinned	Whether the tab should be pinned. Defaults to *false*
integer	(optional) openerTabId	The ID of the tab that opened this tab. If specified, the opener tab must be in the same window as the newly created tab.

function	(optional)	If you specify the *callback* parameter, it should be a function that

	callback	looks like this: `function( Tab tab) {...};` **Tab** tab — Details about the created tab. Will contain the ID of the new tab.

duplicate

`chrome.tabs.duplicate(integer tabId, function callback)`

Since Chrome 23.

Duplicates a tab.

Parameters		
integer	tabId	The ID of the tab which is to be duplicated.
function	(optional) callback	If you specify the *callback* parameter, it should be a function that looks like this: `function( Tab tab) {...};` **Tab** (optional) tab — Details about the duplicated tab. The **tabs.Tab** object doesn't contain `url`, `title` and `favIconUrl` if the `"tabs"` permission has not been requested.

query

`chrome.tabs.query(object queryInfo, function callback)`

Gets all tabs that have the specified properties, or all tabs if no properties are specified.

Parameters		
object	queryInfo	boolean — (optional) active — Whether the tabs are active in their windows. boolean — (optional) pinned — Whether the tabs are pinned. boolean — (optional) audible — Since Chrome 45. Whether the tabs are audible. boolean — (optional) muted — Since Chrome 45.

		Whether the tabs are muted.
boolean	(optional) highlighted	Whether the tabs are highlighted.
boolean	(optional) discarded	Since Chrome 54. Whether the tabs are discarded. A discarded tab is one whose content has been unloaded from memory, but is still visible in the tab strip. Its content gets reloaded the next time it's activated.
boolean	(optional) autoDiscardable	Since Chrome 54. Whether the tabs can be discarded automatically by the browser when resources are low.
boolean	(optional) currentWindow	Whether the tabs are in the **current window**.
boolean	(optional) lastFocusedWindow	Whether the tabs are in the last focused window.
TabStatus	(optional) status	Whether the tabs have completed loading.
string	(optional) title	Match page titles against a pattern. Note that this property is ignored if the extension doesn't have the `"tabs"` permission.
string or array of string	(optional) url	Match tabs against one or more **URL patterns**. Note that fragment identifiers are not matched. Note that this property is ignored if the extension doesn't have the `"tabs"` permission.
integer	(optional) windowId	The ID of the parent window, or **windows.WINDOW_ID_CURRENT** for the **current window**.
WindowType	(optional) windowType	The type of window the tabs are in.
integer	(optional) index	The position of the tabs within their windows.

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

highlight

`chrome.tabs.highlight(object highlightInfo, function callback)`

Highlights the given tabs.

Parameters		
object	highlightInfo	integer — (optional) windowId — The window that contains the tabs. array of integer or integer — tabs — One or more tab indices to highlight.
function	(optional) callback	If you specify the *callback* parameter, it should be a function that looks like this: `function( windows.Window window) {...};` windows.Window — window — Contains details about the window whose tabs were highlighted.

update

`chrome.tabs.update(integer tabId, object updateProperties, function callback)`

Modifies the properties of a tab. Properties that are not specified in *updateProperties* are not modified.

Parameters		
integer	(optional) tabId	Defaults to the selected tab of the **current window**.
object	updateProperties	string — (optional) url — A URL to navigate the tab to. boolean — (optional) active — Whether the tab should be active. Does not affect whether the window is focused (see **windows.update**). boolean — (optional) highlighted — Adds or removes the tab from the current selection. boolean — (optional) — **Deprecated** since Chrome 33.

			selected	Please use *highlighted*. Whether the tab should be selected.
		boolean	(optional) pinned	Whether the tab should be pinned.
		boolean	(optional) muted	Since Chrome 45. Whether the tab should be muted.
		integer	(optional) openerTabId	The ID of the tab that opened this tab. If specified, the opener tab must be in the same window as this tab.
		boolean	(optional) autoDiscardable	Since Chrome 54. Whether the tab should be discarded automatically by the browser when resources are low.
function	(optional) callback	If you specify the *callback* parameter, it should be a function that looks like this: `function( Tab tab) {...};` **Tab** (optional) tab — Details about the updated tab. The **tabs.Tab** object doesn't contain `url`, `title` and `favIconUrl` if the `"tabs"` permission has not been requested.		

move

`chrome.tabs.move(integer or array of integer tabIds, object moveProperties, function callback)`

Moves one or more tabs to a new position within its window, or to a new window. Note that tabs can only be moved to and from normal (window.type === "normal") windows.

Parameters				
integer or array of integer	tabIds	The tab or list of tabs to move.		
object	moveProperties	integer	(optional) windowId	Defaults to the window the tab is currently in.

		integer index — The position to move the window to. -1 will place the tab at the end of the window.
function	(optional) callback	If you specify the *callback* parameter, it should be a function that looks like this: `function( Tab or array of Tab tabs) {...};` Tab or array of Tab — tabs — Details about the moved tabs.

reload

```
chrome.tabs.reload(integer tabId, object reloadProperties, function callback)
```

Reload a tab.

Parameters		
integer	(optional) tabId	The ID of the tab to reload; defaults to the selected tab of the current window.
object	(optional) reloadProperties	boolean — (optional) bypassCache — Whether using any local cache. Default is false.
function	(optional) callback	If you specify the *callback* parameter, it should be a function that looks like this: `function() {...};`

remove

```
chrome.tabs.remove(integer or array of integer tabIds, function callback)
```

Closes one or more tabs.

Parameters		
integer or array of integer	tabIds	The tab or list of tabs to close.
function	(optional) callback	If you specify the *callback* parameter, it should be a function that looks like this: `function() {...};`

detectLanguage

```
chrome.tabs.detectLanguage(integer tabId, function callback)
```

Detects the primary language of the content in a tab.

Parameters		
integer	(optional) tabId	Defaults to the active tab of the **current window**.
function	callback	The *callback* parameter should be a function that looks like this: `function(string language) {...};` string — language — An ISO language code such as `en` or `fr`. For a complete list of languages supported by this method, see **kLanguageInfoTable**. The 2nd to 4th columns will be checked and the first non-NULL value will be returned except for Simplified Chinese for which zh-CN will be returned. For an unknown language, `und` will be returned.

captureVisibleTab

```
chrome.tabs.captureVisibleTab(integer windowId, object options, function callback)
```

Captures the visible area of the currently active tab in the specified window. You must have **<all_urls>** permission to use this method.

Parameters		
integer	(optional) windowId	The target window. Defaults to the **current window**.
object	(optional) options	Details about the format and quality of an image. enum of `"jpeg"`, or `"png"` — (optional) format — The format of the resulting image. Default is `"jpeg"`. integer — (optional) quality — When format is `"jpeg"`, controls the quality of the resulting image. This value is ignored for PNG images. As quality is decreased, the resulting image will have more visual

		artifacts, and the number of bytes needed to store it will decrease.

function	callback	The *callback* parameter should be a function that looks like this: `function(string dataUrl) {...};` string dataUrl: A data URL which encodes an image of the visible area of the captured tab. May be assigned to the 'src' property of an HTML Image element for display.

executeScript

`chrome.tabs.executeScript(integer tabId, object details, function callback)`

Injects JavaScript code into a page. For details, see the **programmatic injection** section of the content scripts doc.

Parameters		
integer	(optional) tabId	The ID of the tab in which to run the script; defaults to the active tab of the current window.
object	details	Details of the script to run. Either the code or the file property must be set, but both may not be set at the same time.

string	(optional) code	JavaScript or CSS code to inject. **Warning:** Be careful using the `code` parameter. Incorrect use of it may open your extension to **cross site scripting** attacks.
string	(optional) file	JavaScript or CSS file to inject.
boolean	(optional) allFrames	If allFrames is `true`, implies that the JavaScript or CSS should be injected into all frames of current page. By default, it's `false` and is only injected into the top frame. If `true` and `frameId` is set, then the code is

				inserted in the selected frame and all of its child frames.
		integer	(optional) frameId	Since Chrome 39. The **frame** where the script or CSS should be injected. Defaults to 0 (the top-level frame).
		boolean	(optional) matchAboutBlank	Since Chrome 39. If matchAboutBlank is true, then the code is also injected in about:blank and about:srcdoc frames if your extension has access to its parent document. Code cannot be inserted in top-level about:-frames. By default it is `false`.
		enum of `"document_start"`, `"document_end"`, or `"document_idle"`	(optional) runAt	The soonest that the JavaScript or CSS will be injected into the tab. Defaults to "document_idle".
function	(optional) callback	Called after all the JavaScript has been executed. If you specify the *callback* parameter, it should be a function that looks like this: `function(array of any result) {...};` array of any — (optional) result — The result of the script in every injected frame.		

insertCSS

`chrome.tabs.insertCSS(integer tabId, object details, function callback)`

Injects CSS into a page. For details, see the **programmatic injection** section of the content scripts doc.

Parameters		
integer	(optional) tabId	The ID of the tab in which to insert the CSS; defaults to the active tab of the current window.
object	details	Details of the CSS text to insert. Either the code or the file property must

be set, but both may not be set at the same time.

string	(optional) code	JavaScript or CSS code to inject. **Warning:** Be careful using the `code` parameter. Incorrect use of it may open your extension to **cross site scripting** attacks.
string	(optional) file	JavaScript or CSS file to inject.
boolean	(optional) allFrames	If allFrames is `true`, implies that the JavaScript or CSS should be injected into all frames of current page. By default, it's `false` and is only injected into the top frame. If `true` and `frameId` is set, then the code is inserted in the selected frame and all of its child frames.
integer	(optional) frameId	Since Chrome 39. The **frame** where the script or CSS should be injected. Defaults to 0 (the top-level frame).
boolean	(optional) matchAboutBlank	Since Chrome 39. If matchAboutBlank is true, then the code is also injected in about:blank and about:srcdoc frames if your extension has access to its parent document. Code cannot be inserted in top-level about:-frames. By default it is `false`.
enum of `"document_start"`, `"document_end"`, or `"document_idle"`	(optional) runAt	The soonest that the JavaScript or CSS will be injected into the tab. Defaults to "document_idle".

function	(optional)	Called when all the CSS has been inserted.

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

setZoom

```
chrome.tabs.setZoom(integer tabId, double zoomFactor, function callback)
```

Since Chrome 42.

Zooms a specified tab.

Parameters		
integer	(optional) tabId	Since Chrome 38. The ID of the tab to zoom; defaults to the active tab of the current window.
double	zoomFactor	Since Chrome 38. The new zoom factor. Use a value of 0 here to set the tab to its current default zoom factor. Values greater than zero specify a (possibly non-default) zoom factor for the tab.
function	(optional) callback	Called after the zoom factor has been changed. If you specify the *callback* parameter, it should be a function that looks like this: `function() {...};`

getZoom

```
chrome.tabs.getZoom(integer tabId, function callback)
```

Since Chrome 42.

Gets the current zoom factor of a specified tab.

Parameters		
integer	(optional) tabId	Since Chrome 38. The ID of the tab to get the current zoom factor from; defaults to the active tab of the current window.
function	callback	Called with the tab's current zoom factor after it has been fetched. The *callback* parameter should be a function that looks like this:

```
function(double zoomFactor) {...};
```

double	zoomFactor	The tab's current zoom factor.

setZoomSettings

```
chrome.tabs.setZoomSettings(integer tabId, ZoomSettings zoomSettings, function callback)
```

Since Chrome 42.

Sets the zoom settings for a specified tab, which define how zoom changes are handled. These settings are reset to defaults upon navigating the tab.

Parameters		
integer	(optional) tabId	Since Chrome 38. The ID of the tab to change the zoom settings for; defaults to the active tab of the current window.
ZoomSettings	zoomSettings	Since Chrome 38. Defines how zoom changes are handled and at what scope.
function	(optional) callback	Called after the zoom settings have been changed. If you specify the *callback* parameter, it should be a function that looks like this: `function() {...};`

getZoomSettings

```
chrome.tabs.getZoomSettings(integer tabId, function callback)
```

Since Chrome 42.

Gets the current zoom settings of a specified tab.

Parameters		
integer	(optional) tabId	Since Chrome 38. The ID of the tab to get the current zoom settings from; defaults to the active tab of the current window.
function	callback	Called with the tab's current zoom settings.

The *callback* parameter should be a function that looks like this:

```
function( ZoomSettings zoomSettings) {...};
```

ZoomSettings	zoomSettings	The tab's current zoom settings.

discard

```
chrome.tabs.discard(integer tabId, function callback)
```

Since Chrome 54.

Discards a tab from memory. Discarded tabs are still visible on the tab strip and are reloaded when activated.

Parameters		
integer	(optional) tabId	The ID of the tab to be discarded. If specified, the tab will be discarded unless it's active or already discarded. If omitted, the browser will discard the least important tab. This can fail if no discardable tabs exist.
function	(optional) callback	Called after the operation is completed. If you specify the *callback* parameter, it should be a function that looks like this: `function( Tab tab) {...};` Tab (optional) tab: Discarded tab if it was successfully discarded. Undefined otherwise.

Events

onCreated

Fired when a tab is created. Note that the tab's URL may not be set at the time this event fired, but you can listen to onUpdated events to be notified when a URL is set.

addListener

```
chrome.tabs.onCreated.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this:

```
function( Tab tab) {...};
```

Tab	tab	Details of the tab that was created.

onUpdated

Fired when a tab is updated.

addListener

```
chrome.tabs.onUpdated.addListener(function callback)
```

Parameters

function	callback	The *callback* parameter should be a function that looks like this: `function(integer tabId, object changeInfo, Tab tab) {...};`

integer	tabId	
object	changeInfo	Lists the changes to the state of the tab that was updated.

string	(optional) status	The status of the tab. Can be either *loading* or *complete*.
string	(optional) url	The tab's URL if it has changed.
boolean	(optional) pinned	The tab's new pinned state.
boolean	(optional) audible	Since Chrome 45. The tab's new audible state.
boolean	(optional) discarded	Since Chrome 54. The tab's new

		discarded state.	
		boolean	(optional) autoDiscardable	Since Chrome 54. The tab's new auto-discardable state.
		MutedInfo	(optional) mutedInfo	Since Chrome 46. The tab's new muted state and the reason for the change.
		string	(optional) favIconUrl	Since Chrome 27. The tab's new favicon URL.
		string	(optional) title	Since Chrome 48. The tab's new title.
	Tab	tab	Gives the state of the tab that was updated.	

onMoved

Fired when a tab is moved within a window. Only one move event is fired, representing the tab the user directly moved. Move events are not fired for the other tabs that must move in response. This event is not fired when a tab is moved between windows. For that, see **tabs.onDetached**.

addListener

```
chrome.tabs.onMoved.addListener(function callback)
```

Parameters		
function	callback	The *callback* parameter should be a function that looks like this:

```
function(integer tabId, object moveInfo) {...};
```

integer	tabId	
object	moveInfo	

integer	windowId	
integer	fromIndex	
integer	toIndex	

onSelectionChanged

Deprecated since Chrome 33. Please use **tabs.onActivated**.

Fires when the selected tab in a window changes.

addListener

```
chrome.tabs.onSelectionChanged.addListener(function callback)
```

Parameters

function	callback	The *callback* parameter should be a function that looks like this:

```
function(integer tabId, object selectInfo) {...};
```

integer	tabId	The ID of the tab that has become active.
object	selectInfo	

integer	windowId	The ID of the window the selected tab changed inside of.

onActiveChanged

Deprecated since Chrome 33. Please use **tabs.onActivated**.

Fires when the selected tab in a window changes. Note that the tab's URL may not be set at the time this event fired, but you can listen to **tabs.onUpdated** events to be notified when a URL is set.

addListener

```
chrome.tabs.onActiveChanged.addListener(function callback)
```

Parameters

function	callback	The *callback* parameter should be a function that looks like this: `function(integer tabId, object selectInfo) {...};` integer tabId — The ID of the tab that has become active. object selectInfo integer windowId — The ID of the window the selected tab changed inside of.

onActivated

Fires when the active tab in a window changes. Note that the tab's URL may not be set at the time this event fired, but you can listen to onUpdated events to be notified when a URL is set.

addListener

```
chrome.tabs.onActivated.addListener(function callback)
```

Parameters

function	callback	The *callback* parameter should be a function that looks like this: `function(object activeInfo) {...};` object activeInfo integer tabId — The ID of the tab that has become active. integer windowId — The ID of the window the active tab changed inside of.

onHighlightChanged

Deprecated since Chrome 33. Please use **tabs.onHighlighted**.

Fired when the highlighted or selected tabs in a window changes.

addListener

```
chrome.tabs.onHighlightChanged.addListener(function callback)
```

Parameters

function	callback	The *callback* parameter should be a function that looks like this: `function(object selectInfo) {...};` object selectInfo integer windowId — The window whose tabs changed. array of integer tabIds — All highlighted tabs in the window.

onHighlighted

Fired when the highlighted or selected tabs in a window changes.

addListener

```
chrome.tabs.onHighlighted.addListener(function callback)
```

Parameters

function	callback	The *callback* parameter should be a function that looks like this: `function(object highlightInfo) {...};` object highlightInfo integer windowId — The window whose tabs changed. array of integer tabIds — All highlighted tabs in the window.

onDetached

Fired when a tab is detached from a window, for example because it is being moved between windows.

addListener

```
chrome.tabs.onDetached.addListener(function callback)
```

Parameters

function	callback	The *callback* parameter should be a function that looks like this: `function(integer tabId, object detachInfo) {...};` integer tabId object detachInfo integer oldWindowId integer oldPosition

onAttached

Fired when a tab is attached to a window, for example because it was moved between windows.

addListener

```
chrome.tabs.onAttached.addListener(function callback)
```

Parameters

function	callback	The *callback* parameter should be a function that looks like this: `function(integer tabId, object attachInfo) {...};` integer tabId object attachInfo integer newWindowId integer newPosition

onRemoved

Fired when a tab is closed.

addListener

```
chrome.tabs.onRemoved.addListener(function callback)
```

Parameters

function	callback	The *callback* parameter should be a function that looks like this: `function(integer tabId, object removeInfo) {...};` integer tabId object removeInfo integer windowId Since Chrome 25.

					The window whose tab is closed.
				boolean	isWindowClosing	True when the tab is being closed because its window is being closed.

onReplaced

Since Chrome 26.

Fired when a tab is replaced with another tab due to prerendering or instant.

addListener

```
chrome.tabs.onReplaced.addListener(function callback)
```

Parameters

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

onZoomChange

Since Chrome 38.

Fired when a tab is zoomed.

addListener

```
chrome.tabs.onZoomChange.addListener(function callback)
```

Parameters

function	callback	The *callback* parameter should be a function that looks like this: `function(object ZoomChangeInfo) {...};` object ZoomChangeInfo integer tabId double oldZoomFactor double newZoomFactor ZoomSettings zoomSettings