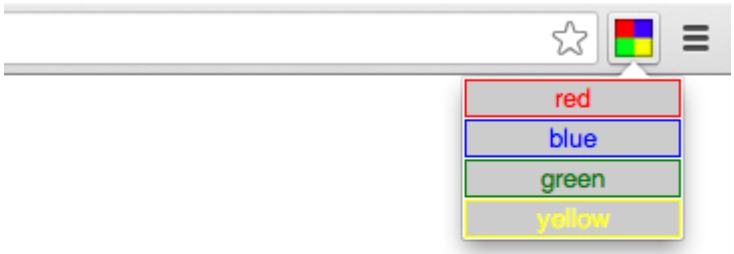
chrome.browserAction

Description:	Use browser actions to put icons in the main Google Chrome toolbar, to the right of the address bar. In addition to its **icon**, a browser action can have a **tooltip**, a **badge**, and a **popup**.
Availability:	Since Chrome 27.
Manifest:	`"browser_action": {...}`

In the following figure, the multicolored square to the right of the address bar is the icon for a browser action. A popup is below the icon.



If you want to create an icon that isn't always visible, use a **page action** instead of a browser action.

Manifest

Register your browser action in the **extension manifest** like this:

```
{
  "name": "My extension",
  ...
  "browser_action": {
    "default_icon": {                    // optional
      "16": "images/icon16.png",           // optional
      "24": "images/icon24.png",           // optional
      "32": "images/icon32.png"            // optional
```

```
    },
    "default_title": "Google Mail",      // optional; shown in tooltip
    "default_popup": "popup.html"        // optional
  },
  ...
}
```

You can provide any size icon to be used in Chrome, and Chrome will select the closest one and scale it to the appropriate size to fill the 16-dip space. However, if the exact size isn't provided, this scaling can cause the icon to lose detail or look fuzzy.

Since devices with less-common scale factors like 1.5x or 1.2x are becoming more common, you are encouraged to provide multiple sizes for your icons. This also ensures that if the icon display size is ever changed, you don't need to do any more work to provide different icons!

The old syntax for registering the default icon is still supported:

```
{
  "name": "My extension",
  ...
  "browser_action": {
    ...
    "default_icon": "images/icon32.png"  // optional
    // equivalent to "default_icon": { "32": "images/icon32.png" }
  },
  ...
}
```

Parts of the UI

A browser action can have an **icon**, a **tooltip**, a **badge**, and a **popup**.

Icon

The browser action icons in Chrome are 16 dips (device-independent pixels) wide and high. Larger icons are resized to fit, but for best results, use a 16-dip square icon.

You can set the icon in two ways: using a static image or using the HTML5 **canvas element**. Using static images is easier for simple applications, but you can create more dynamic UIs — such as smooth animation — using the canvas element.

Static images can be in any format WebKit can display, including BMP, GIF, ICO, JPEG, or PNG. For unpacked extensions, images must be in the PNG format.

To set the icon, use the **default_icon** field of **browser_action** in the **manifest**, or call the **browserAction.setIcon** method.

To properly display icon when screen pixel density (ratio `size_in_pixel / size_in_dip`) is different than 1, the icon can be defined as set of images with different sizes. The actual image to display will be selected from the set to best fit the pixel size of 16 dip. The icon set can contain any size icon specification, and Chrome will select the most appropriate one.

Tooltip

To set the tooltip, use the **default_title** field of **browser_action** in the **manifest**, or call the **browserAction.setTitle** method. You can specify locale-specific strings for the **default_title** field; see **Internationalization** for details.

Badge

Browser actions can optionally display a *badge* — a bit of text that is layered over the icon. Badges make it easy to update the browser action to display a small amount of information about the state of the extension.

Because the badge has limited space, it should have 4 characters or less.

Set the text and color of the badge using **browserAction.setBadgeText** and **browserAction.setBadgeBackgroundColor**, respectively.

Popup

If a browser action has a popup, the popup appears when the user clicks the icon. The popup can contain any HTML contents that you like, and it's automatically sized to fit its contents.

To add a popup to your browser action, create an HTML file with the popup's contents. Specify the HTML file in the **default_popup** field of **browser_action** in the **manifest**, or call the **browserAction.setPopup** method.

Tips

For the best visual impact, follow these guidelines:

- **Do** use browser actions for features that make sense on most pages.
- **Don't** use browser actions for features that make sense for only a few pages. Use **page actions** instead.

- **Do** use big, colorful icons that make the most of the 16x16-dip space. Browser action icons should seem a little bigger and heavier than page action icons.
- **Don't** attempt to mimic Google Chrome's monochrome menu icon. That doesn't work well with themes, and anyway, extensions should stand out a little.
- **Do** use alpha transparency to add soft edges to your icon. Because many people use themes, your icon should look nice on a variety of background colors.
- **Don't** constantly animate your icon. That's just annoying.

Examples

You can find simple examples of using browser actions in the **examples/api/browserAction** directory. For other examples and for help in viewing the source code, see **Samples**.

Summary

Types
ColorArray
ImageDataType
Methods
setTitle – `chrome.browserAction.setTitle(object details, function callback)`
getTitle – `chrome.browserAction.getTitle(object details, function callback)`
setIcon – `chrome.browserAction.setIcon(object details, function callback)`
setPopup – `chrome.browserAction.setPopup(object details, function callback)`
getPopup – `chrome.browserAction.getPopup(object details, function callback)`
setBadgeText – `chrome.browserAction.setBadgeText(object details, function callback)`
getBadgeText – `chrome.browserAction.getBadgeText(object details, function callback)`
setBadgeBackgroundColor – `chrome.browserAction.setBadgeBackgroundColor(object details, function callback)`
getBadgeBackgroundColor – `chrome.browserAction.getBadgeBackgroundColor(object details, function callback)`
enable – `chrome.browserAction.enable(integer tabId, function callback)`
disable – `chrome.browserAction.disable(integer tabId, function callback)`
Events
onClicked

Types

ColorArray

array of integer

ImageDataType

Pixel data for an image. Must be an ImageData object; for example, from a `canvas` element.

Methods

setTitle

```
chrome.browserAction.setTitle(object details, function callback)
```

Sets the title of the browser action. This title appears in the tooltip.

Parameters

object	details	

string	title	The string the browser action should display when moused over.
integer	(optional) tabId	Limits the change to when a particular tab is selected. Automatically resets when the tab is closed.

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

getTitle

```
chrome.browserAction.getTitle(object details, function callback)
```

Gets the title of the browser action.

Parameters		
object	details	integer (optional) tabId — The tab to get the title from. If no tab is specified, the default title is returned.
function	callback	The *callback* parameter should be a function that looks like this: `function(string result) {...};` string result

setIcon

`chrome.browserAction.setIcon(object details, function callback)`

Sets the icon for the browser action. The icon can be specified as the path to an image file, as the pixel data from a canvas element, or as a dictionary of one of those. Either the `path` or the `imageData` property must be specified.

Parameters		
object	details	**ImageDataType** or object (optional) imageData — Either an ImageData object or a dictionary {size -> ImageData} representing an icon to be set. If the icon is specified as a dictionary, the image used is chosen depending on the screen's pixel density. If the number of image pixels that fit into one screen space unit equals `scale`, then an image with size `scale` * n is selected, where *n* is the size of the icon in the UI. At least one image must be specified. Note that 'details.imageData = foo' is equivalent to 'details.imageData = {'16': foo}'

		string or object	(optional) path	Either a relative image path or a dictionary {size -> relative image path} pointing to an icon to be set. If the icon is specified as a dictionary, the image used is chosen depending on the screen's pixel density. If the number of image pixels that fit into one screen space unit equals `scale`, then an image with size `scale` * n is selected, where *n* is the size of the icon in the UI. At least one image must be specified. Note that 'details.path = foo' is equivalent to 'details.path = {'16': foo}'
		integer	(optional) tabId	Limits the change to when a particular tab is selected. Automatically resets when the tab is closed.
function	(optional) callback	If you specify the *callback* parameter, it should be a function that looks like this: `function() {...};`		

setPopup

```
chrome.browserAction.setPopup(object details, function callback)
```

Sets the HTML document to be opened as a popup when the user clicks the browser action icon.

Parameters				
object	details	integer	(optional) tabId	Limits the change to when a particular tab is selected. Automatically resets when the tab is closed.
		string	popup	The HTML file to show in a popup. If set to the empty string (''), no popup is shown.

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

getPopup

`chrome.browserAction.getPopup(object details, function callback)`

Gets the HTML document that is set as the popup for this browser action.

Parameters

object	details	integer (optional) tabId — The tab to get the popup from. If no tab is specified, the non-tab-specific popup is returned.
function	callback	The *callback* parameter should be a function that looks like this: `function(string result) {...};` string result

setBadgeText

`chrome.browserAction.setBadgeText(object details, function callback)`

Sets the badge text for the browser action. The badge is displayed on top of the icon.

Parameters

object	details	string text — Any number of characters can be passed, but only about four can fit into the space. integer (optional) tabId — Limits the change to when a particular tab is selected. Automatically resets when the tab is closed.
function	(optional) callback	If you specify the *callback* parameter, it should be a function that looks like this: `function() {...};`

getBadgeText

`chrome.browserAction.getBadgeText(object details, function callback)`

Gets the badge text of the browser action. If no tab is specified, the non-tab-specific badge text is returned.

Parameters

object	details	integer (optional) tabId — The tab to get the badge text from. If no tab is specified, the non-tab-specific badge text is returned.
function	callback	The *callback* parameter should be a function that looks like this: `function(string result) {...};` string result

setBadgeBackgroundColor

`chrome.browserAction.setBadgeBackgroundColor(object details, function callback)`

Sets the background color for the badge.

Parameters

object	details	string or **ColorArray** color — An array of four integers in the range 0-255 that make up the RGBA color of the badge. Can also be a string with a CSS hex color value; for example, `#FF0000` or `#F00` (red). Renders colors at full opacity. integer (optional) tabId — Limits the change to when a particular tab is selected. Automatically resets when the tab is closed.

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

getBadgeBackgroundColor

`chrome.browserAction.getBadgeBackgroundColor(object details, function callback)`

Gets the background color of the browser action.

Parameters		
object	details	integer (optional) tabId: The tab to get the badge background color from. If no tab is specified, the non-tab-specific badge background color is returned.
function	callback	The *callback* parameter should be a function that looks like this: `function( ColorArray result) {...};` **ColorArray** result

enable

`chrome.browserAction.enable(integer tabId, function callback)`

Enables the browser action for a tab. Defaults to enabled.

Parameters		
integer	(optional) tabId	The ID of the tab for which to modify the browser action.
function	(optional) callback	If you specify the *callback* parameter, it should be a function that looks like this: `function() {...};`

disable

```
chrome.browserAction.disable(integer tabId, function callback)
```

Disables the browser action for a tab.

Parameters		
integer	(optional) tabId	The ID of the tab for which to modify the browser action.
function	(optional) callback	If you specify the *callback* parameter, it should be a function that looks like this: `function() {...};`

Events

onClicked

Fired when a browser action icon is clicked. Does not fire if the browser action has a popup.

addListener

```
chrome.browserAction.onClicked.addListener(function callback)
```

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