When to reload the extension:

The following table shows which components need to be reloaded to see changes:

Extension component	Requires extension reload
The manifest	Yes
Service worker	Yes
Content scripts	Yes (plus the host page)
The popup	No
Options page	No
Other extension HTML pages	No

Manifest file format

Every extension must have a manifest.json file in its root directory that lists important information about the structure and behavior of that extension.

```
{
  "manifest_version": 3,
  "name": "Minimal Manifest",
  "version": "1.0.0",
  "description": "A basic example extension with only required keys",
  "icons": {
    "48": "images/icon-48.png",
    "128": "images/icon-128.png"
  },
}
```

Manifest keys

The following is a list of all supported manifest keys.

Keys required by the Extensions platform:

"manifest_version"

An integer specifying the version of the manifest file format that your extension uses. The only supported value is 3.

"name"

A string that identifies the extension in the Chrome Web Store, the install dialog, and the user's Chrome Extensions page (chrome://extensions). The maximum length is 75 characters. For information on using locale-specific names, see Internationalization.

"version"

A string that identifies the extension's version number. For information on version number formatting, see Version.

A couple of rules apply to the integers:

- The integers must be between 0 and 65535, inclusive.
- Non-zero integers can't start with 0. For example, 032 is invalid because it begins with a zero.
- They must not be all zero. For example, 0 and 0.0.0.0 are invalid while 0.1.0.0 is valid.

Version name

In addition to the "version" field, which is used for update purposes, "version_name" can be set to a descriptive version string and will be used for display purposes if present.

Here are some examples of version names:

- "version_name": "1.0 beta"
- "version_name": "build rc2"
- "version_name": "3.1.2.4567"

If no version_name is present, the version field will be used for display purposes as well.

Keys required by Chrome Web Store

"description"

A string that describes the extension on both the Chrome Web Store and the user's extension management page. The maximum length is 132 characters. For information on localizing descriptions, see Internationalization.

chrome.i18n

Use the chrome.i18n infrastructure to implement **internationalization** across your whole app or extension.

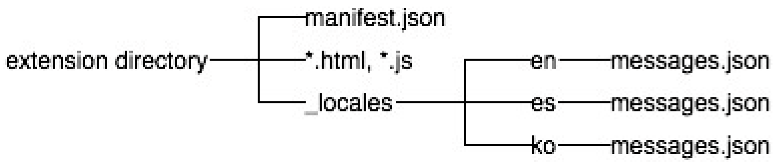
Manifest

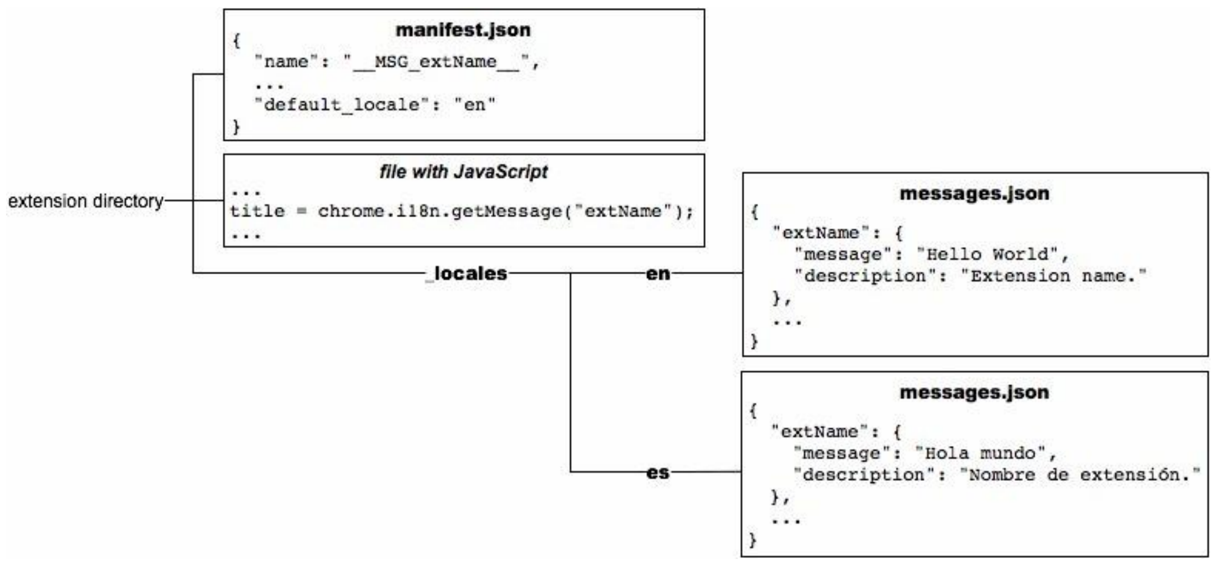
If an extension has a /_locales directory, the manifest must define "default_locale".

Concepts and usage

You need to put all of its user-visible strings into a file named messages.json. Each time you add a new locale, you add a messages file under a directory named /_locales/_localeCode_, where *localeCode* is a code such as en for English.

Here's the file hierarchy for an internationalized extension that supports English (en), Spanish (es), and Korean (ko):





manifest.json

```
{
  "name": "__MSG_extName__",
  ...
  "default_locale": "en"
}
```

file with JavaScript

```
...
title = chrome.i18n.getMessage("extName");
...
```

messages.json

```
{
  "extName": {
    "message": "Hello World",
    "description": "Extension name."
  },
  ...
}
```

messages.json

```
{
  "extName": {
    "message": "Hola mundo",
    "description": "Nombre de extensión."
  },
  ...
}
```

"icons"

One or more icons that represent your extension. For information about best practices, see Icons.

You should always provide a 128x128 icon; it's used during installation and by the Chrome Web Store. Extensions should also provide a 48x48 icon, which is used in the extensions management page (chrome://extensions). You can also specify a 16x16 icon to be used as the favicon for an extension's pages.

Icons should generally be in PNG format, because PNG has the best support for transparency. They can, however, be in any format supported by WebKit, including BMP, GIF, ICO, and JPEG. Here's an example of specifying the icons:

```
"icons": { "16": "icon16.png",

         "48": "icon48.png",

        "128": "icon128.png" },
```

You can also call action.setIcon() to set your extension's icon programmatically by specifying a different image path

Optional keys

"action"

Defines the appearance and behavior of the extension's icon in the Google Toolbar. For more information, see chrome.action.

Use the **chrome.action** API to control the extension's icon in the Google Chrome toolbar.

To use the chrome.action API, specify a "manifest_version" of 3 and include the "action" key in your manifest file.

```
{
 "name": "Action Extension",
 …
 "action": {
  "default_icon": {           // optional
    "16": "images/icon16.png",   // optional
    "24": "images/icon24.png",   // optional
    "32": "images/icon32.png"    // optional
  },
  "default_title": "Click Me",   // optional, shown in tooltip
  "default_popup": "popup.html"  // optional
 },
 …
}
```

Enabled state

By default, toolbar actions are enabled (clickable) on every tab. You can change this default by setting the default_state property in the action key of the manifest. If default_state is set to "disabled", the action is disabled by default and must be enabled programmatically to be clickable. If default_state is set to "enabled" (the default), the action is enabled and clickable by default.

You can control the state programmatically using the **action.enable()** and **action.disable()** methods. This only affects whether the popup (if any) or **action.onClicked** event is sent to your extension; it doesn't affect the action's presence in the toolbar.

Parts of the UI

Icon

The icon is the main image on the toolbar for your extension, and is set by the "default_icon" key in your manifest's "action"

Note: The **action.setIcon()** API is intended to set a static image. Don't use animated images for your icons.

Tooltip (title)

The tooltip, or title, appears when the user holds their mouse pointer over the extension's icon in the toolbar. It's also included in the accessible text spoken by screen readers when the button gets focus.

The default tooltip is set using the "default_title" field of the "action" key in manifest.json. You can also set it programmatically by calling action.setTitle().

Badge

Actions can optionally display a "badge" — a bit of text layered over the icon. This lets you update the action to display a small amount of information about the state of the extension, such as a counter. The badge has a text component and a background color. Because space is limited, we recommend that badge text use four or fewer characters.

To create a badge, set it programmatically by calling **action.setBadgeBackgroundColor()** and **action.setBadgeText().** There isn't a default badge setting in the manifest. Badge color values can be either an array of four integers between 0 and 255 that make up the RGBA color of the badge or a string with a CSS color value.

```
chrome.action.setBadgeBackgroundColor(
  {color: [0, 255, 0, 0]},  // Green
  () => { /* ... */ },
);


chrome.action.setBadgeBackgroundColor(
  {color: '#00FF00'},  // Also green
  () => { /* ... */ },
);
```

```
chrome.action.setBadgeBackgroundColor(
  {color: 'green'},  // Also, also green
  () => { /* ... */ },
);
```

Popup

An action's popup is shown when the user clicks on the extension's action button in the toolbar. The popup can contain any HTML contents you like, and will be automatically sized to fit its contents. The popup's size must be between 25x25 and 800x600 pixels.

The popup is initially set by the "**default_popup**" property in the "action" key in the manifest.json file. If present, this property should point to a relative path within the extension directory. It can also be updated dynamically to point to a different relative path using the **action.setPopup()** method.

"author"

Specifies the email address of the account that was used to create the extension.