Declare Permissions and Warn Users

An extension's ability to access websites and most Chrome APIs is determined by its declared **permissions**. Permissions should be restricted to only what is needed for its functionality. Limiting permissions establishes an extension's capabilities and reduces possible incursion to data if the extension is compromised by an attacker. Protect extensions and their users by implementing explicit, minimal and optional permissions.

Organize Permissions

Permissions are known strings that refer to a Chrome API or **match patterns** that grant access to one or more hosts. They are listed in the manifest and specified as required permissions or **optional permissions**.

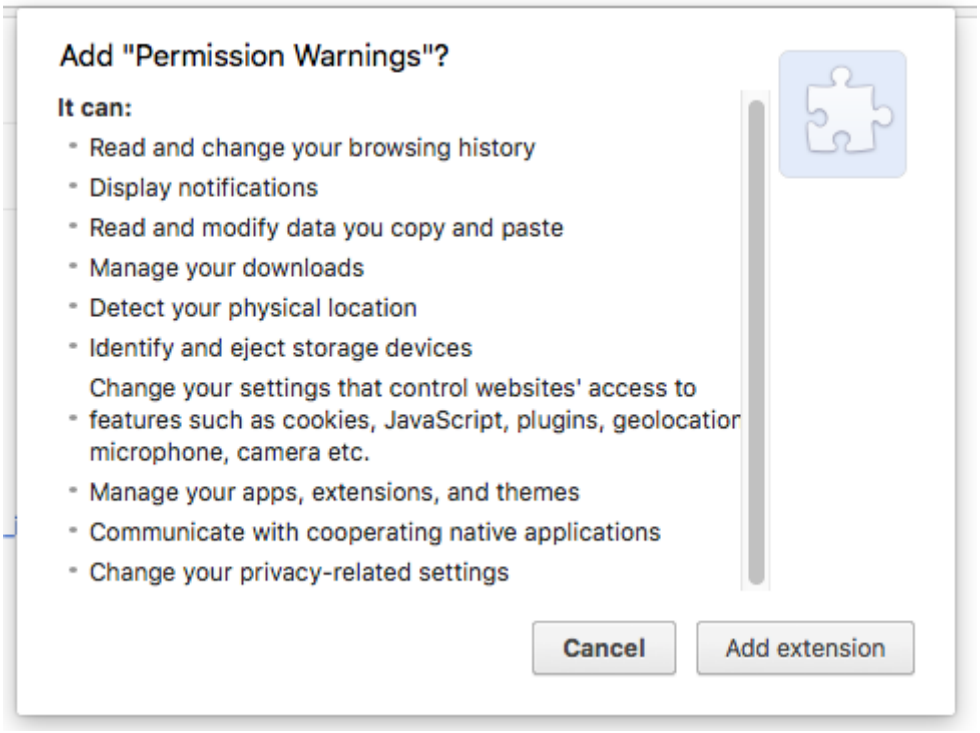
```
{
  "name": "Permissions Extension",
  ...
  // required permissions
  "permissions": [
    "activeTab",
    "contextMenus",
    "storage"
  ],
  // optional permissions
  "optional_permissions": [
    "topSites",
    "http://www.developer.chrome.com/*"
  ],
      ...
  "manifest_version": 2
}
```

Limit required permissions to only what is needed for the extension's core functionality. An extension should not request more permissions than it currently needs; do not future proof by requesting permissions that may be needed with updates.

Permissions needed for optional features should be registered as **optional permissions**. This allows users to decide how much access they are willing to provide an extension and which features are desired.

Identify Required Permissions

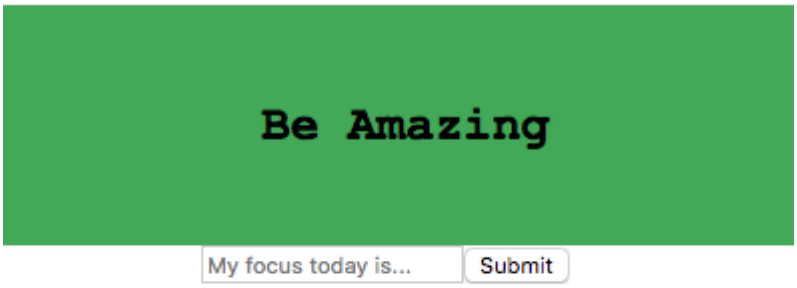
A simple extension may need to request multiple permissions, and many permissions display **warnings** on installation. Users are more likely to trust an extension with limited warnings or when permissions are explained to them.



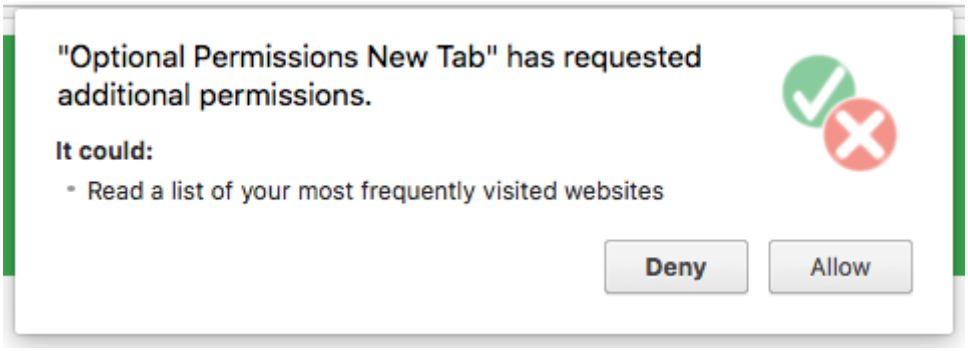
Identify the core functionality of an extension and what permissions are required for it. Consider making features optional if they require permissions with warnings.

Trigger Optional Permissions with Events

The **optional permissions sample extension's** core functionality is overriding the new tab page. One feature is displaying the user's goal of the day. This feature only requires the **storage** permission, which does not include a warning.



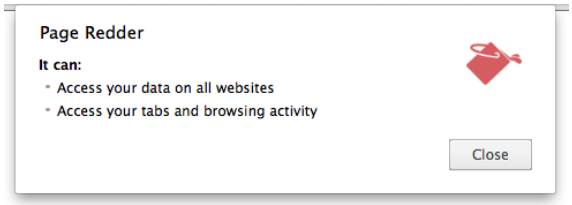
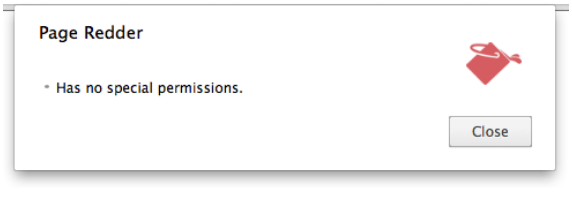
The extension has an additional feature; displaying the user's top sites. This feature requires the **topSites** permission, which has a warning.



Developing features that rely on permissions with warnings as optional and introducing those features organically gives users a risk free introduction to the extension. Additionally, this allows users to further customize their experience with an extension and creates opportunity to explain warnings.

Substitute the activeTab Permission

The `activeTab` permission grants temporary access to the site the user is on and allows the extension to use the **`"tabs"`** permission on the current tab. It replaces the need for `"<all_urls>"` in many cases and displays no warning on installation.

Without activeTab	With activeTab
	

The `activeTab` permission grants an extension temporary access to the currently active tab when the **user invokes** the extension. If the extension is compromised, the attacker would need to wait for the user to invoke the extension before obtaining access, and that access would only last until the tab is navigated or closed.

While the `activeTab` permission is enabled for a tab, an extension can:

- Call **`tabs.executeScript`** or **`tabs.insertCSS`** on that tab.
- Get the URL, title, and favicon for that tab via an API that returns a **`tabs.Tab`** object.
- Intercept network requests in the tab to the tab's main frame origin using the **`webRequest`** API. The 拦截
  extension temporarily gets host permissions for the tab's main frame origin.

The following user gestures enable `activeTab`:

- Executing a **browser action**
- Executing a **page action**

- Executing a **context menu item**
- Executing a keyboard shortcut from the **commands API**
- Accepting a suggestion from the **omnibox API**

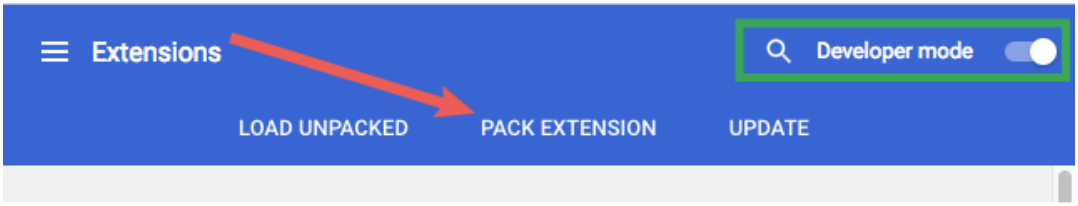
Understanding Permissions

Permission warnings exist to describe the capabilities granted by an API to extension users, but some of these warnings may not be obvious at first. For instance, adding the `"tabs"` permission results in a seemingly unrelated warning: the extension can **Read your browsing activity**. Although the `chrome.tabs` API might be used to only open new tabs, it can also be used to see the URL that is associated with every newly opened tab by using their **tabs.Tab** objects.

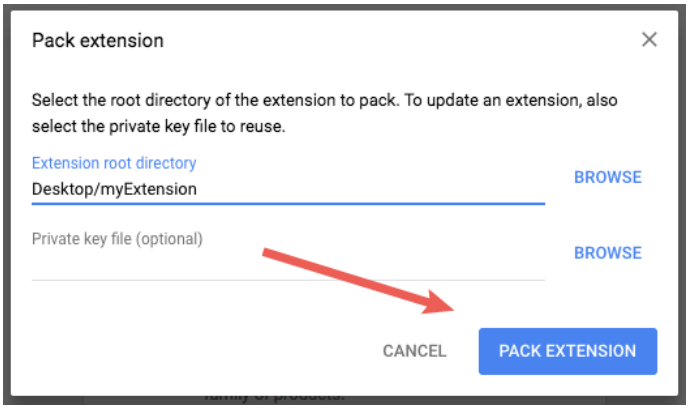
When possible, implement **optional permissions** or a less powerful API to avoid alarming warnings.

Viewing Warnings

No permission warnings will be displayed if an extension is loaded as an unpacked file. To view an extension's permission warnings, navigate to `chrome://extensions`, ensure developer mode is enabled and click **PACK EXTENSION**.



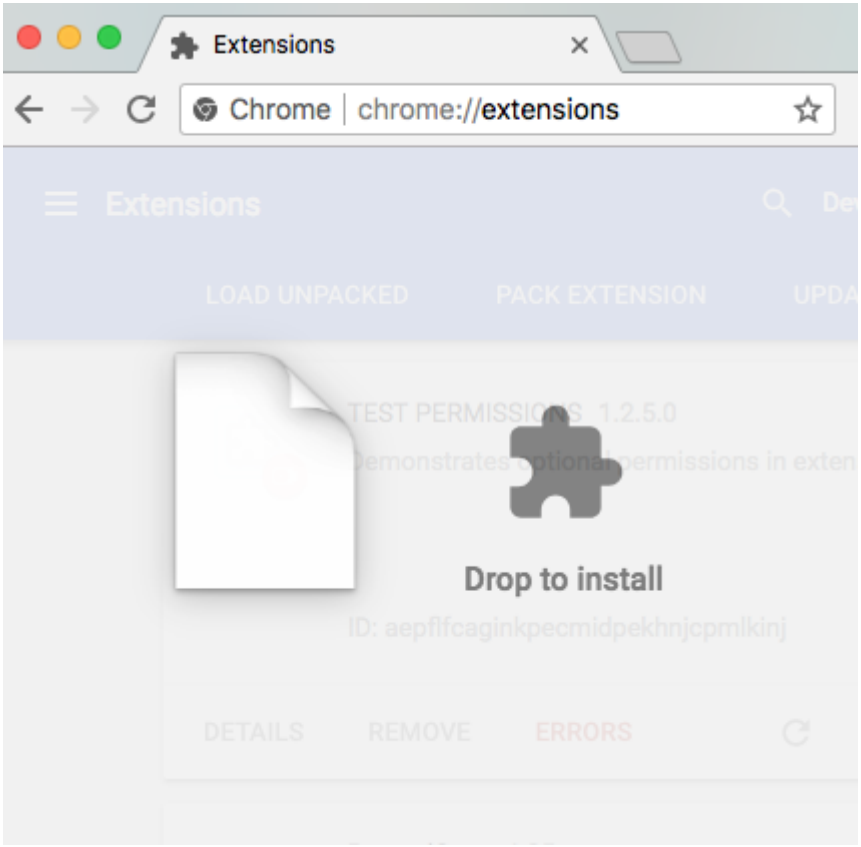
Specify the path to the extension's folder in the Extension root directory field then click the **Pack Extension** button. Ignore the **Private key** field for a first-time package.



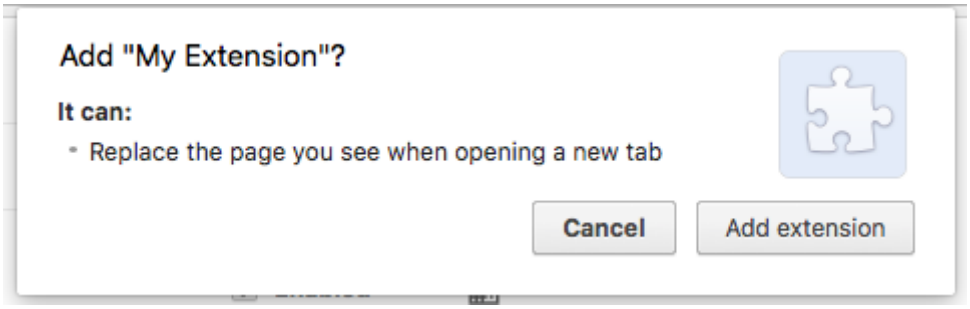
Chrome will create two files, a `.crx` file and a `.pem` file, which contains the extension's private key.

Do not lose the private key! Keep the `.pem` file in a secret and secure place; it will be needed to **update** the extension.

Install the `.crx` file by dropping it into the Chrome Extension's Management page.



After dropping the `.crx` file the browser will ask if the extension can be added and display warnings.



Permissions with Warnings

Note: Permission tables are updated on a best-effort basis and may contain slight discrepancies with the current warnings. Additionally, some permissions may not display warnings when paired with other permissions. For example, the **"tabs"** warning will not show if the extension also requests "<all_urls>". To verify the most recent warnings shown for extension permissions, follow the steps in **Viewing Warnings**.

Permission	Description	Warning
• "http://*/*" • "https://*/*" • "*://*/*" • "<all_urls>"	Grants the extension access to all hosts. It may be possible to avoid declaring any host permissions by using the **activeTab** permission.	**Read and change all your data on the websites you visit**
"https://HostName.com/"	Grants the extension access to "https://HostName.com/". It may be possible to avoid declaring any host permissions by using the **activeTab** permission.	**Read and change your data on HostName.com**
"bookmarks"	Grants your extension access to the **chrome.bookmarks** API.	**Read and change your bookmarks**
"clipboardRead"	Required if the extension uses document.execCommand('paste').	**Read data you copy and paste**
"clipboardWrite"	Indicates the extension uses document.execCommand('copy') or document.execCommand('cut').	**Modify data you copy and paste**
"contentSettings"	Grants your extension access to the **chrome.contentSettings** API.	**Change your settings that control websites' access to features such as cookies, JavaScript, plugins, geolocation, microphone, camera etc.**
"debugger"	Grants your extension access to the **chrome.debugger** API.	• **Access the page debugger backend** • **Read and change all your data on**

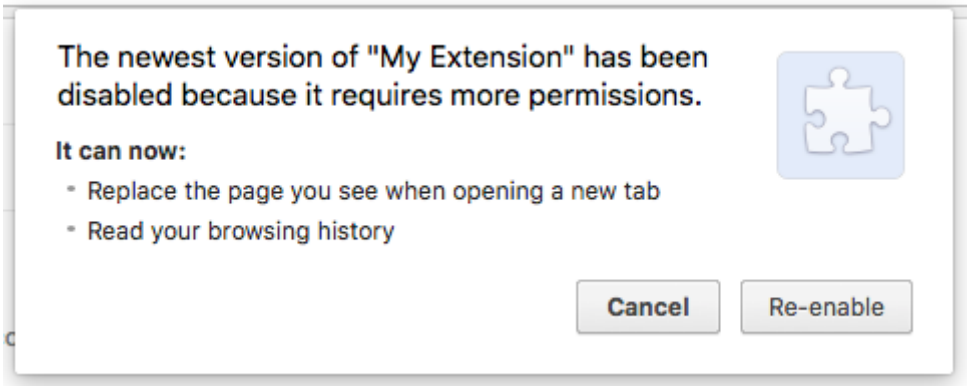
		the websites you visit
`"desktopCapture"`	Grants your extension access to the **chrome.desktopCapture** API.	**Capture content of your screen**
`"downloads"`	Grants your extension access to the **chrome.downloads** API.	**Manage your downloads**
`"geolocation"`	Allows the extension to use the HTML5 **geolocation API** without prompting the user for permission.	**Detect your physical location**
`"history"`	Grants your extension access to the **chrome.history** API.	**Read and change your browsing history**
`"management"`	Grants the extension access to the **chrome.management** API.	**Manage your apps, extensions, and themes**
`"nativeMessaging"`	Gives the extension access to the **native messaging API**.	**Communicate with cooperating native applications**
`"notifications"`	Grants your extension access to the **chrome.notifications** API.	**Display notifications**
`"pageCapture"`	Grants the extension access to the **chrome.pageCapture** API.	**Read and change all your data on the websites you visit**
`"privacy"`	Gives the extension access to the **chrome.privacy** API.	**Change your privacy-related settings**
`"proxy"`	Grants the extension access to the **chrome.proxy** API.	**Read and change all your data on the websites you visit**
`"system.storage"`	Grants the extension access to the **chrome.system.storage** API.	**Identify and eject storage devices**
`"tabCapture"`	Grants the extensions access to the **chrome.tabCapture** API.	**Read and change all your data on**

		the websites you visit
`"tabs"`	Grants the extension access to privileged fields of the **Tab** objects used by several APIs including **chrome.tabs** and **chrome.windows**. In many circumstances the extension will not need to declare the `"tabs"` permission to make use of these APIs.	**Read your browsing history**
`"topSites"`	Grants the extension access to the **chrome.topSites** API.	**Read a list of your most frequently visited websites**
`"ttsEngine"`	Grants the extension access to the **chrome.ttsEngine** API.	**Read all text spoken using synthesized speech**
`"webNavigation"`	Grants the extension access to the **chrome.webNavigation** API.	**Read your browsing history**

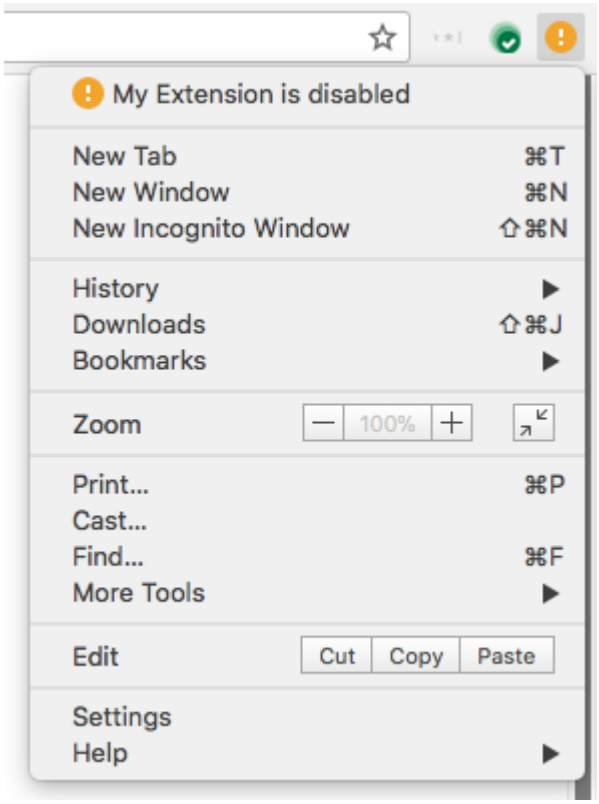
Update Permissions

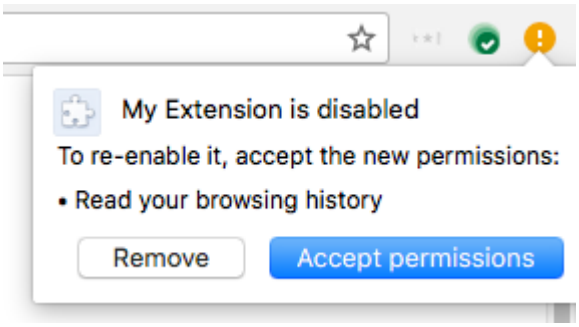
Updating an extension with additional permissions may temporarily disable it. The user will have to re-enable it after agreeing to any new warnings.

If the user manually updates an extension that now includes the **tabs** permission, they will get a warning on the management page.



If the extension is updated automatically it will be disabled until the user agrees to the new permissions.





This can be avoided by making the new feature optional and adding new permission updates to `optional_permissions` in the **manifest**.