Give Users Options

Allow users to customise the behavior of an extension by providing an options page. A user can view an extension's options by right-clicking the extension icon in the toolbar then selecting options or by navigating to the extension management page at `chrome://extensions`, locating the desired extension, clicking **Details**, then selection the options link.

Write the Options Page

Below is an example options page.

```
<!DOCTYPE html>
<html>
<head><title>My Test Extension Options</title></head>
<body>
```

```
Favorite color:
<select id="color">
 <option value="red">red</option>
 <option value="green">green</option>
 <option value="blue">blue</option>
 <option value="yellow">yellow</option>
</select>

<label>
  <input type="checkbox" id="like">
  I like colors.
</label>

<div id="status"></div>
<button id="save">Save</button>

<script src="options.js"></script>
</body>
</html>
```

Save a user's preferred options across devices by using the **storage.sync** API.

```
// Saves options to chrome.storage
function save_options() {
  var color = document.getElementById('color').value;
  var likesColor = document.getElementById('like').checked;
  chrome.storage.sync.set({
    favoriteColor: color,
    likesColor: likesColor
  }, function() {
    // Update status to let user know options were saved.
    var status = document.getElementById('status');
    status.textContent = 'Options saved.';
    setTimeout(function() {
      status.textContent = '';
    }, 750);
  });
}

// Restores select box and checkbox state using the preferences
// stored in chrome.storage.
function restore_options() {
  // Use default value color = 'red' and likesColor = true.
  chrome.storage.sync.get({
    favoriteColor: 'red',
    likesColor: true
  }, function(items) {
    document.getElementById('color').value = items.favoriteColor;
    document.getElementById('like').checked = items.likesColor;
  });
}
document.addEventListener('DOMContentLoaded', restore_options);
document.getElementById('save').addEventListener('click',
    save_options);
```

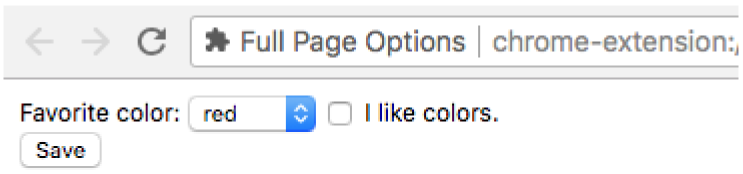
Declare Options Page Behavior

There are two available types of extension options pages, **full page** and **embedded**. The type of options is determined by how it is declared in the manifest.

Full Page Options

An extension's options page will be displayed in a new tab. The options HTML file is listed registered under the `options_page` field.

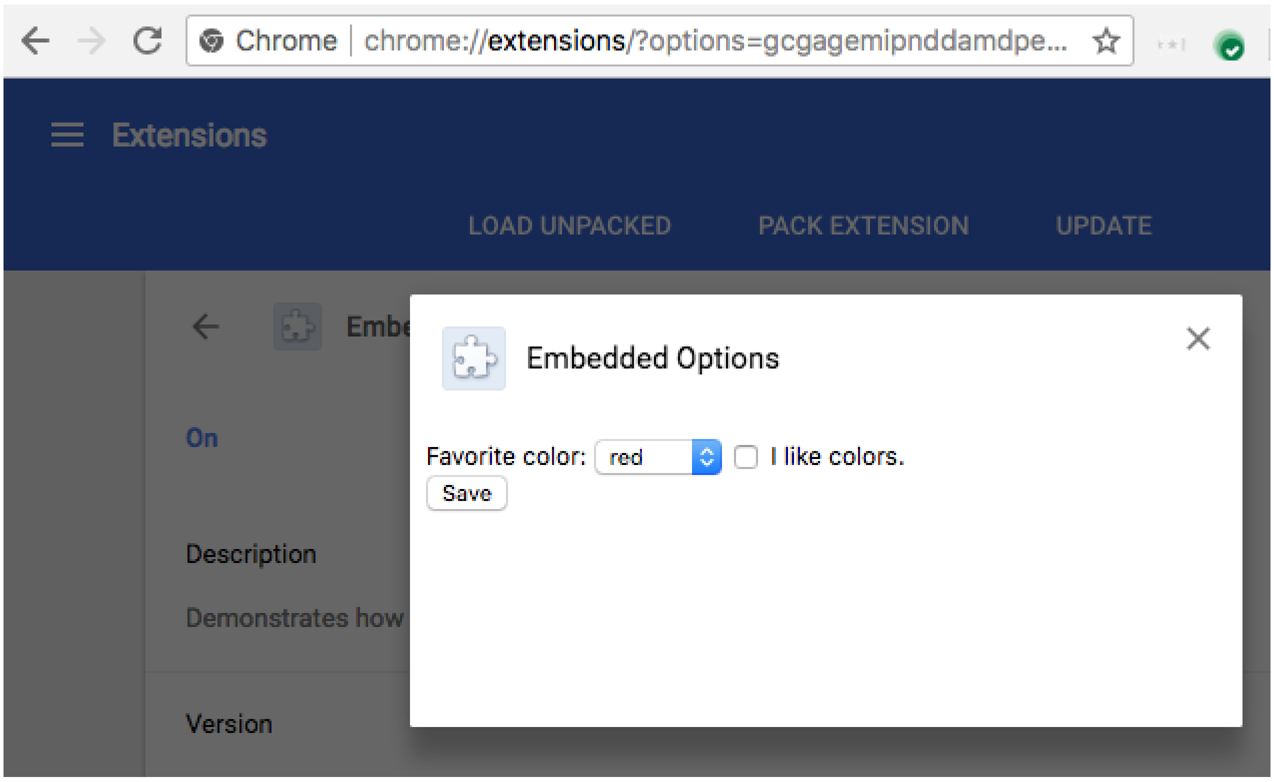
```
{
  "name": "My extension",
  ...
  "options_page": "options.html",
  ...
}
```



Embedded Options

Embedded options allows users to adjust extension options without navigating away from the extensions management page inside an embedded box. To declare an embedded options, register the HTML file under the `options_ui` field in the extension manifest, with the `open_in_tab` key set to false.

```
{
  "name": "My extension",
  ...
  "options_ui": {
    "page": "options.html",
    "open_in_tab": false
  },
  ...
}
```



- page (string)

  Path to the options page, relative to the extension's root.

- open_in_tab (boolean)

  Specify as `false` to declare an embedded options page. If `true`, the extension's options page will be opened in a new tab rather than embedded in *chrome://extensions*.

Consider the Differences

Options pages embedded inside *chrome://extensions* have some subtle behavior differences related to not being hosted inside their own tabs.

Linking to the options page

An extension can link directly to the options page by calling `chrome.runtime.openOptionsPage()` .

```
<button id="go-to-options">Go to options</button>
```

```
document.querySelector('#go-to-options').addEventListener(function() {
  if (chrome.runtime.openOptionsPage) {
    chrome.runtime.openOptionsPage();
  } else {
    window.open(chrome.runtime.getURL('options.html'));
  }
});
```

Tabs API

Extension embedded options page code is not hosted inside a tab, affecting how the **Tabs API** can be used:

- **tabs.query** will never find a tab within an extension's options page URL.
- **tabs.onCreated** will not fire when the options page is opened.
- **tabs.onUpdated** will not fire when the options page load state changes.
- **tabs.connect** or **tabs.sendMessage** cannot be used to communicate with the options page.

Using **runtime.connect** and **runtime.sendMessage** is a work around to these restrictions, if the options page does need to manipulate the containing tab.

Messaging APIs

If an extension's options page sends a message using **runtime.connect** or **runtime.sendMessage**, the **Sender's tab** will not be set, and the **Sender's URL** will be the options page URL.

Sizing

The embedded options should automatically determine its own size based on the page content. However, the embedded box may not find a good size for some types of content. This problem is most common for options pages that adjust their content shape based on window size.

If this is an issue, provide fixed minimum dimensions for the options page to ensure that the embedded page will find an appropriate size.