**1. chrome.action.setIcon**

**Purpose**: Sets the icon of the extension's action button dynamically. You can specify different icon sizes for different resolutions.

**Key Options**:

* path: Path to the image file for the icon.
  + Can be a single path (e.g., "icons/icon48.png") or an object specifying paths for different resolutions (e.g., {"16": "icon16.png", "48": "icon48.png", "128": "icon128.png"}).

**Example Code**:

document.getElementById("setIcon").addEventListener("click", () => {

// Specify icons for different resolutions

chrome.action.setIcon({

path: {

"16": "icons/icon16.png",

"48": "icons/icon48.png",

"128": "icons/icon128.png"

}

}, () => {

console.log("Icon updated with specific resolutions (16, 48, 128 pixels).");

});

});

**Details**:

* Use multiple sizes (e.g., 16x16, 48x48, 128x128) for better rendering on different displays like retina screens.
* Icons must be in .png format for best compatibility.

**2. chrome.action.setBadgeBackgroundColor**

**Purpose**: Sets the background color of the badge displayed on the action button.

**Key Options**:

* color: The color for the badge. Can be a string (e.g., "#FF0000") or an array of RGBA values (e.g., [255, 0, 0, 255]).

**Example Code**:

chrome.action.setBadgeBackgroundColor({ color: "#00FF00" }, () => {

console.log("Badge background color set to green (#00FF00).");

});

**Details**:

* Use hexadecimal color codes or RGBA for transparency.
* Default color is often red (#FF0000).

**3. chrome.action.openPopup**

**Purpose**: Opens the popup window programmatically, without requiring the user to click the action button.

**Key Options**:

* options: Currently, this API doesn't require additional options.
* callback: Optional function to execute after opening the popup.

**Example Code**:

document.getElementById("openPopup").addEventListener("click", () => {

chrome.action.openPopup(() => {

console.log("Popup opened programmatically.");

});

});

**Details**:

* Useful when you want to display the popup in response to some background logic.

**4. chrome.action.isEnabled**

**Purpose**: Checks whether the action button is enabled for a specific tab or globally.

**Key Options**:

* tabId: Optional. The ID of the tab to check. If omitted, it checks the global state.
* callback: A function to receive the result (true or false).

**Example Code**:

document.getElementById("checkEnabled").addEventListener("click", () => {

chrome.action.isEnabled((isEnabled) => {

console.log("Is action enabled globally?", isEnabled);

alert(`Action is ${isEnabled ? "enabled" : "disabled"}`);

});

});

**5. chrome.action.getUserSettings**

**Purpose**: Retrieves user settings related to the extension's action. This can tell you if the user has pinned your extension.

**Key Options**:

* callback: Function that receives the user settings, including whether the action is pinned.

**Example Code**:

chrome.action.getUserSettings((settings) => {

console.log("User settings:", settings);

alert(`Extension is pinned: ${settings.isOnToolbar}`);

});

**Details**:

* The isOnToolbar property tells you if the extension is pinned to the toolbar.

**6. chrome.action.getTitle**

**Purpose**: Retrieves the title (tooltip) of the action button.

**Key Options**:

* details: Includes tabId to specify the tab (optional).
* callback: Function that receives the title as a string.

**Example Code**:

chrome.action.getTitle({}, (title) => {

console.log("Current title:", title);

alert(`Title: ${title}`);

});

**7. chrome.action.getPopup**

**Purpose**: Retrieves the URL of the popup associated with the action button.

**Key Options**:

* details: Includes tabId to specify the tab (optional).
* callback: Function that receives the popup URL.

**Example Code**:

chrome.action.getPopup({}, (popupUrl) => {

console.log("Popup URL:", popupUrl);

alert(`Popup URL: ${popupUrl}`);

});

**8. chrome.action.getBadgeTextColor**

**Purpose**: Gets the current text color of the badge.

**Key Options**:

* details: Includes tabId to specify the tab (optional).
* callback: Function that receives the text color.

**Example Code**:

chrome.action.getBadgeTextColor({}, (color) => {

console.log("Badge text color:", color);

alert(`Badge text color: ${JSON.stringify(color)}`);

});

**9. chrome.action.getBadgeText**

**Purpose**: Retrieves the current text displayed on the badge.

**Key Options**:

* details: Includes tabId to specify the tab (optional).
* callback: Function that receives the badge text.

**Example Code**:

chrome.action.getBadgeText({}, (text) => {

console.log("Badge text:", text);

alert(`Badge text: ${text}`);

});

**10. chrome.action.getBadgeBackgroundColor**

**Purpose**: Gets the background color of the badge.

**Key Options**:

* details: Includes tabId to specify the tab (optional).
* callback: Function that receives the color.

**Example Code**:

chrome.action.getBadgeBackgroundColor({}, (color) => {

console.log("Badge background color:", color);

alert(`Badge background color: ${JSON.stringify(color)}`);

});

**11. chrome.action.enable**

**Purpose**: Enables the action button, either globally or for a specific tab.

**Key Options**:

* tabId: Optional. The ID of the tab to enable the action for.
* callback: Function executed after enabling.

**Example Code**:

chrome.action.enable(() => {

console.log("Action enabled globally.");

});

**12. chrome.action.disable**

**Purpose**: Disables the action button, either globally or for a specific tab.

**Key Options**:

* tabId: Optional. The ID of the tab to disable the action for.
* callback: Function executed after disabling.

**Example Code**:

chrome.action.disable(() => {

console.log("Action disabled globally.");

});

**Implementation Notes:**

1. **Customization**:
   * Icons should be 16x16, 48x48, or 128x128 to ensure clarity across all displays.
   * Badge text should be short, ideally 4 characters, to fit the small badge area.
2. **Testing**:
   * Test tabId-specific behaviors in a multi-tab environment.
   * Use console.log extensively to verify API responses during development.

This breakdown ensures you can not only show but also explain each API's functionality to your students in detail.