MD Bookmarks context menu and keyboard shortcuts

Status:Draft
Author: tsergeant@chromium.org

Objective

Define an element which handles ‘commands’ in the MD Bookmark manager, by controlling context menus and keyboard shortcuts and ensuring that logic is applied consistently across commands.

Background

In the bookmarks manager, it is possible to execute several different editing or browsing commands: eg, deleting bookmarks, or opening a folder of bookmarks in a new window. These commands can be executed from either a context menu (opened by right clicking on a bookmark item or sidebar node), or by keyboard shortcut. Commands may be enabled or disabled based on the current UI or profile state. In some cases (delete), there are other pieces of UI which are also able to execute the command. Further, some commands (edit) need to show additional UI as the action is executed.

In order to manage this, we will create a dedicated element which is able to show the context menu, handle keyboard shortcuts, and display associated UI. By doing this, logic is deduplicated into one place, and new commands are easier to add.

Detailed Design

Initialization

We will create a new element, `<bookmarks-command-manager>`. This will be created and attached to the page as a child of `<bookmarks-app>`. During attached(), the element will start listening for keydown events on the main document.

TBD: It may be necessary to expose this element instance somewhere (like `bookmarks.CommandManager.getInstance()`) to allow other elements to call command handlers.

Adding commands

To add a new command:

1. Add the command name to the enum in `constants.js`
2. Add a case to `canExecute()` to determine when the command is enabled
3. Add a case to `handle()` to execute the command

Then, you can add a keyboard shortcut and/or a context menu item as required (it is possible to skip either step).

Adding keyboard shortcuts

Keyboard shortcuts are registered during `attached()`:

```
this.shortcuts_['copy'] = cr.isMac ? 'meta+c' : 'ctrl+c';
```

Keyboard shortcuts are specified using the syntax of iron-a11y-keys, which is used to parse and check shortcuts. Notably, it is possible to specify multiple shortcuts for one command by separating them with space.

Once a shortcut has been registered, the global keydown handler will check each event to see if it matches any available shortcut. If it does, it will call `canExecute(command)` and then (if allowed) `handle(command)` for the currently selected items.

Adding commands to the context menu

Commands should be added to the context menu in `command_manager.html` following a set template:

```
<button class="dropdown-item"
    command="copy"
    hidden$="[[!canExecute('copy' menuIds_)]]"
    on-tap="handleCommandClick_">
  $i18n{menuCopyURL}
</button>
```

This way, all commands are guaranteed to show/hide and execute in the same way.

Available commands

Context menu

Visible if means the command will appear in the context menu, **Disabled if** means the command will not be clickable in the context menu. Shortcuts will only be enabled if the command is visible and not disabled.

Command	Visible if	Disabled if	Mac shortcut	Non-mac shortcut
Edit (rename)	A single node is selected AND editing is enabled globally	The selected node is unmodifiable	Enter	F2
Copy URL	A single item node is selected		Meta-C	Ctrl-C

Show in folder	A single node is selected AND search is active AND the menu was opened from the list			
Delete	Anything is selected and Editing is enabled globally	Any selected node is unmodifiable (?)	Delete Backspace Meta-Backspace	Delete
Separator line	*Any of the above is true ie, editing is enabled globally OR a single item node is selected*			
Open all OR Open in new tab	Anything is selected	The set of selected nodes (after expanding folders one level) is empty	Meta-enter	Ctrl-enter
Open (all) in new window	Anything is selected	The set of selected nodes (after expanding folders one level) is empty	Shift-Enter	Shift-Enter
Open (all) in incognito	Anything is selected	The set of selected nodes (after expanding folders one level) is empty OR all urls are special chrome urls OR incognito is disabled		

Keyboard shortcuts

Command	**Enabled if**	**Mac shortcut**	**Non-mac shortcut**
Open (in foreground tabs)	Anything is selected	Meta-Down	Enter
Undo	~~The undo service allows us to undo something~~ Always enabled	Meta-Z	Ctrl-Z
Redo	Always enabled	Meta-Shift-Z	Ctrl-Shift-Z Ctrl-Y

Cut	Editing is allowed and the selected nodes are modifiable	Meta-X	Ctrl-X
Paste	Search is not active, and the selected folder's children are modifiable	Meta-V	Ctrl-V
Select all	Always enabled	Meta-A	Ctrl-A
Deselect all	Always enabled	Escape	Escape

Opening URLs

The bookmark manager lets you open bookmarks in several ways: From mouse, keyboard or context menu.

- Open bookmarks in **foreground** tabs in current window.
  - Double click any bookmark (shift or ctrl double-click to do it with multiple selected)
  - Enter or meta-down
  - Note that if you do this to a **single folder**, it will select that folder in the BMM rather than opening the bookmarks
- Open bookmarks in **background** tabs in current window
  - Use the 'Open all' context menu item
  - Ctrl-enter (doesn't work correctly in BMM)
- Open bookmarks in new window
  - Use 'Open in new window' context menu item
  - Shift-enter (doesn't work correctly in BMM)
- Open bookmarks in new incognito window
  - Use context menu item
- Opening bookmarks with middle-click
  - When middle clicking an item, always select that item
  - Middle clicking a bookmark (not a folder) opens that one bookmark in a new background tab (or foreground tab if shift is held)

Context menu items and keyboard shortcuts can be directly handled by the Command Manager. Mouse events will need to be handled by <bookmarks-item> and then passed into the command manager (by event?) to open the relevant items.

Public API

Events

open-item-menu
Opens the context menu for the currently selected set of items at either `e.detail.targetElement`, or `(e.detail.x, e.detail.y)`.

Context menu

openContextMenu(element)
Show the context menu, positioned to cover |element|
TBD: This currently always shows the context menu for the selected set of elements. This may need to change when we make it possible to show a context menu on the sidebar.

openContextMenuAt(x, y)
As above, but at specific screen co-ordinates

closeContextMenu()
Close the context menu if it is open

Command Handlers

canExecute(command, itemIds)
Returns true if the command can be executed for the given set of items

handle(command, itemIds)
Executes the command for the set of items, showing any UI (dialogs, toasts) as necessary.

Caveats/Alternative Approaches

Using <command>

WebUI has an existing system for tying keyboard shortcuts to context menus, <command> and <cr-menu>. This is used extensively in the old bookmark manager. This system predates Polymer, does not natively support Mac shortcuts, and would require significant work to integrate with the new <cr-action-menu>.

Test Plan

What are the sub-units of your system that will be independently testable? Tests must be approved by the code reviewer, and must follow the guidelines in the unittesting document as far as possible. If there are changes envisaged in your future work, would your tests verify the base functionality? If some of your tests cannot be easily automated (e.g. UI tests), how will you document the needed special procedures?

Document History

2017-05-05: Increased detail on how bookmarks are opened from different places. Split apart

when context menu commands are “visible” and “enabled”.
2017-04-24: First draft