Custom action support in chrome.automation - New Chrome API Proposal

Proposal Date
May 31, 2017

Who are the primary Eng and PM contacts for this API? (email addresses)
Eng: yawano@chromium.org
PM: mitsuji@chromium.org, lpalmaro@chromium.org

What are the relevant bug(s)?
https://crbug.com/702659

Which team will be responsible for this API? (team email address)
Chrome / Chrome OS Accessibility

Do you know anyone else, internal or external, that is also interested in this API?

- ChromeVox
- Accessibility services built on top of chrome.automation API should also be interested in the API, e.g. Switch Access.

What's a reasonable Chrome milestone target for landing most of the code on Trunk?
M61

Overview
Android apps can run on some Chrome OS devices. In Android, a widget can provide custom actions which are specific to the UI. See details at AccessibilityActions section in Android developers page. This API is to expose those information in chrome.automation API.

AccessibilityNodeInfo.AccessibilityAction
https://developer.android.com/reference/android/view/accessibility/AccessibilityNodeInfo.AccessibilityAction.html

While the initial primary use case will be to interact with an element in Android UI, Chrome UI also would be able to have custom actions.

This API proposal is an extension to chrome.automation API.

chrome.automation API.

https://developer.chrome.com/extensions/automation

Use cases

- Add custom action support in Android window for ChromeVox

Can you imagine any other similar use cases which your API might address?

1. Some Chrome UI also might want to have custom actions.
2. An automation test tool would be able to use this API to interact with an element.

How would you implement your desired features if this API didn't exist?

- Add virtual nodes to the element which allows an accessibility service to perform custom actions for the element. For example, if an element has “Custom action A” and “Custom action B”, add 2 virtual nodes as children of the element. An user can invoke those custom actions by activating those virtual nodes.

Is this something that could/should be part of the web platform, or an existing chrome.* API?

This API is part of chrome.automation API.

Does this API expose any functionality to the web?

No

Do you expect this API to be fairly stable? How might it be extended or changed in the future?

Yes. The interface is simple and should be stable. Also custom action is defined as a dictionary. It should be easy to add other attributes in the future.

If multiple apps/extensions used this API at the same time, could they conflict with each other? If so, how do you propose to mitigate this problem?

Yes, if 2 extensions perform custom action on the same element at the same time, the action will be performed twice on the element. But chrome.automation API should have this issue in general. I won’t add any additional mitigation for custom action support. I’ll expect that this should be handled as part of chrome.automation API.

List every UI surface belonging to or potentially affected by your API:

No additional system UI for custom action. Accessibility service built on top of chrome.automation API might show additional UI for custom actions. For example, ChromeVox might show custom actions of currently focused element in the ChromeVox menu.

Actions taken with app/extension APIs should be obviously attributable to an app/extension. Will users be able to tell when this new API is being used? How? Can it be spoofed?

No additional mitigation for custom action support. I expect that this should be handled as part

of chrome.automation API.

Does this API impose any requirements on the Chrome Web Store ?
No additional requirement for custom action support. All extensions which can use chrome.automation API can use custom action support.

Does this API have any interaction with other Chrome APIs? Does it impose any restrictions on other APIs that can be used in conjunction?
No, I don't think any additional interaction with other Chrome APIs from chrome.automation API. As the primary use case of this API to interact with an element in Android window, there will be a dependency to the feature that running Android apps on Chrome OS.

How could this API be abused?
An evil extension might be able to perform a custom action on a widget (e.g. reset) without an user intention.

Imagine you're Dr. Evil Extension Writer, list the three worst evil deeds you could commit with your API (if you've got good ones, feel free to add more):

3. An android application can set arbitrary label for each custom action (e.g. it might contain PII). An evil extension might collect those labels and send it to an evil server.
4. An evil extension perform all custom actions without an user intention.
5. An evil extension keep calling an API and make the system slower (or freeze).

What security UI or other mitigations do you propose to limit evilness made possible by this new API?
No additional mitigation for custom actions. I expect that this should be handled as part of chrome.automation API.

Could a consumer of your API cause any permanent change to the user's system using your API that would not be reversed when that consumer is removed from the system?
Yes, as this API allow an extension to interact with an element in the UI. But I believe this is the intention of this API.

Draft Manifest Changes
No new permission or manifest change for custom action support. Every extension which can use chrome.automation API should be able to use custom action.

Draft API spec
CL: https://crrev.com/2873373005

Open questions
No.