Tabindex Focus Navigation Explainer

(was: Better Focus Navigation for Shadow DOM)

Takayoshi Kochi <kochi@chromium.org>

Oct. 30, 2014

last update: Jun 17, 2015

Motivation

As bugs being filed at W3C bug 25473, and at Chromium issue 380445, Custom Elements have issues on handling focus navigation when it involves both shadow DOM and tabindex. This work tries to resolve the case that a shadow host can be focused, while the actual focus needs to be on its internal element, and the author wants to have control over focus navigation order (TAB / SHIFT + TAB). This also tries to explain how an HTML element can be made tab focusable without `tabindex` (like `<a href="...">`) in terms of web components technology, so that a custom element behave exactly in the same manner as such elements natively implemented on browsers.

Problem Description

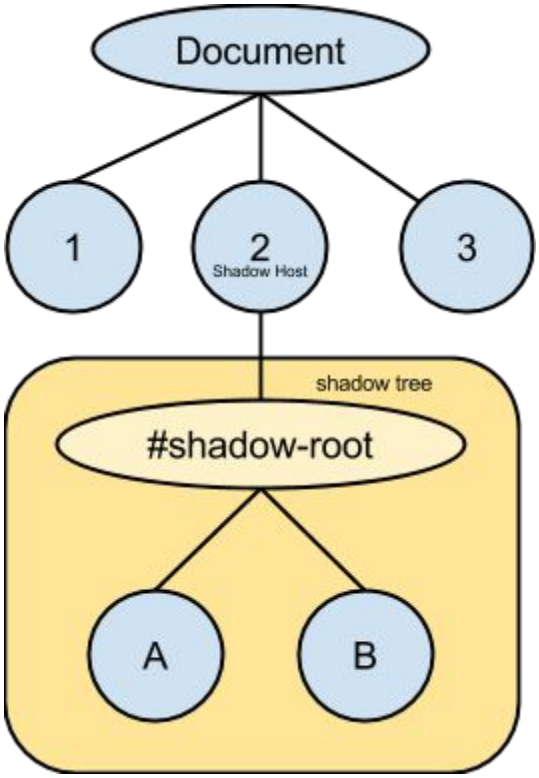
In this document we are tackling 2 problems which are both related to tabindex focus navigation.

The first issue is about the controllability of tab navigation ordering. It is possible for an element to have multiple focusable fields (e.g. `<input type="date">` which has 3 focusable fields on a desktop browser's implementation). Such an element can be tab focusable, and the sequential tab navigation order in the document is controllable via `tabindex` attribute with positive value[1].

When a web author wants to create his/her own one with multiple focusable fields using combination of Shadow DOM and Custom Elements, the default focus navigation behavior (defined in the Shadow DOM spec) works well, but once he/she tries to customize the tab navigation order by giving the custom element (a shadow host) a `tabindex`, tab navigation order breaks in an unexpected way.

- Forward direction: **focus stops at the host element itself**, and will not skip to the first element inside the custom element.
- Backward direction: when focus exits from inside the shadow tree, **focus stops at the host element itself**.

[1] There is some discussions to discourage or prohibit usage of positive values for tabindex. See discussion archived at http://lists.w3.org/Archives/Public/public-html/2014Oct/thread.html#msg27.



In the example diagram above, forward navigation order is 1-2-A-B-3, and backward navigation order is 3-B-A-2-1. Stopping at 2 is the unwanted behavior. This problem is not specific to custom elements, but can apply to any shadow hosts.

The second issue is about tab focusability. In the HTML5 spec, any element can be focusable via TAB key by adding a `tabindex` attribute, which is defined in the tabindex attribute section. The first problem above is a side effect of this attribute. On the other hand, without a `tabindex` attribute, some elements are natively tab focusable (e.g. `<a>` element with `href` attribute). The HTML5 spec introduces an implicit "tabindex focus flag" to explain the behavior of these elements. There is no way at this point to create an element with the same capability with Custom Elements, Shadow DOM or both. If this is resolved, focus behavior of those native elements can be explained in terms of Custom Elements and Shadow DOM.

These 2 issues can be viewed as two sides of the same coin, while they could be orthogonal issues. We are trying to solve these two issues at once.

Non-goals

Some issues are reported on navigation ordering of distributed nodes under shadow DOM (github). It is related, but orthogonal to this.
We don't solve spatial navigation (up, down, left, right) issue within Shadow DOMs or Custom Element. But eventual solution may be easily applied to spatial navigation as well.

Proposed Solution

We would like to solve the problem by decomposing the convoluted `tabIndex` property into 3 properties, `tabIndex`, `focusable`, and `delegatesFocus`, with keeping backward compatibility as much as possible.

- `tabIndex` continues to mean the navigation order when it is read, but when it is written, it affects other properties to be updated (see below)
- `focusable` is a read-only boolean property which reflects whether the element is focusable (i.e. mouse click on the element or calling focus() method will get focus on it). This corresponds to HTML5 spec's "tabindex focus flag". As this is a read-only property, you cannot override this property on regular elements, but for shadow hosts, it can reflect its shadow root's "force focusable" flag, which can be set via `setForceFocusable()` method.
- `delegatesFocus` is a read-only boolean property on a shadow root, which indicates focus activity (tab navigation, mouse click, focus()) on its shadow host will be delegated to its shadow.

The IDL looks like:

```
partial interface Element {
  ShadowRoot createShadowRoot(ShadowRootInit shadowRootInitDict);
  readonly attribute focusable;
}

partial interface ShadowRoot {
  void setForceFocusable(boolean flag);
}

partial dictionary ShadowRootInit {
  boolean delegatesFocus;
}
```

Note: ShadowRootInit dictionary was introduced recently in the shadow DOM spec.

`focusable` returns the value of "tabindex focus flag". By default, focusable is `false` for any HTML element except for those which are defined to be "tabindex focus flag" set in the spec. For those elements whose "tabindex focus flag" is false, the flag can be turned on by setting `tabindex` attribute.

Here are tables of how `focusable` is expected to return.

On non-default-focusable element (e.g. `<div>`, `<a name="">`)

tabIndex	ommitted	-1	>=0
focusable	false	true	true
navigation order	N/A	skipped	value of tabindex

On a default-focusable element (e.g. `<input type=text>`, `<div contenteditable="true">`, `<a href="">`)

tabIndex	omitted	-1	>=0
focusable	true	true	true
navigation order	equivalent to tabindex=0	skipped	value of tabindex

`delegatesFocus` controls how a shadow host behave for focus. By default, `delegatesFocus` is `false` and can be turned on when it is specified for `createShadowRoot()`'s parameter.

Here's the short summary of comparison when `delegatesFocus` is `true` or `false`. The difference is emphasized with yellow background.

Shadow Host (non-default-focusable element, `delegatesFocus=false`, existing behavior)

tabIndex	unspecified	-1	>=0
focusable	false	true	true
navigation order	N/A	skipped (host only, if there is any focusable elements in shadow root, they **are** in the order)	value of tabindex
matches :focus when element in shadow is focused	no	no	no
focus() slides to inner focusable element	no	no	no

To give some context, the above table is also accurate for non-shadow DOM cases, e.g. it would hold for simple s (potentially with focusable children).

Shadow Host (non-default focusable element, `delegatesFocus=true`)

tabIndex	unspecified	-1	>=0
focusable	true	true	true
navigation order	equivalent to tabindex=0	whole focusable elements under shadow root are skipped	value of tabindex
matches :focus when element in shadow is focused	yes	yes	yes
focus() slides to inner focusable element	yes	yes	yes

By using ShadowRoot's `setForceFocusable(boolean flag)` method, you can forcibly set its shadow host's focusable property to be always true regardless of `tabIndex` or `delegatesFocus`, when `true` is given for `setForceFocusable()`. If `false` is given, behavior of shadow host is same as described above.

delegatesFocus details

If a shadow host delegates focus (its shadow root was created with `delegatesFocus=true`), the following behavior is expected.

1. TAB navigation order
   If `tabindex` is 0 or positive and the containing shadow tree has one or more focusable elements, forward tab navigation will skip focus on the host itself and forwards focus to the first focusable element, then to the second, …, to the last focusable element until focus blurs to the next focusable element within the same treescope as the host element. For backward tab navigation, the last focusable element gets focus first, and after the first focusable element, until focus blurs to the previous focusable element in the same treescope as the host element. This behavior applies recursively on a shadow host in a shadow tree.
   If the host element has no focusable element under it, tab navigation just skips the host and its shadow tree.
   In the case of `tabindex=-1`, the whole shadow tree is skipped for the tab navigation. See discussion below for whether or not user agent should skip the whole subtree or not.
2. focus() method behavior
   Invoking focus() method on the host element will delegate the focus the first focusable element in its subtree. This applies recursively for nested shadow trees. If the shadow root doesn't contain any focusable element, the host itself gets focus.

3. autofocus attribute
   If autofocus attribute is specified, the focus is forwarded like `focus()` method when the page load finishes.
4. Response to mouse click
   If focusable area within the shadow tree is clicked, the element gets focus. If non-focusable area is clicked, the shadow host gets focus. This is analogous to when `<div>` with `tabindex` attribute gets focus when its content is clicked. The difference is, that the nearest focusable shadow host up in the tree gets focus, skipping non-focusable shadow hosts.
5. CSS `:focus` pseudo-class
   A selector like `#host:focus` matches when focus is in any of its descendent shadow tree.
6. `document.activeElement` and `shadowroot.activeElement`
   This doesn't change. The shadow host becomes activeElement when an element in its shadow tree has focus.

If you want to control which element gets first focus when forward or backward navigation comes to the component, you can use `tabindex` for each element within the component.

Alternatives Considered

The following solutions are considered.

1. Add `willFocusCallback()` / `willBlurCallback()` with direction (forward/backward) as a parameter for Custom Elements
   At first I pursued the idea of adding these callbacks to Custom Element's callback registry. One thing I noticed was that we need such facility for shadow hosts, not only for custom elements. It is okay to add this to Custom Elements, and have authors use Custom Elements instead of raw shadow host if they need the capability. This certainly adds ability to change low-level behavior of focus for custom elements, but I'm still unsure any viable use cases.
2. Add `direction` attribute in focus event object
   This idea is orthogonal to adding `tabStop`, and can be added independently, not exclusively. Though this alone might be able to solve focusing issues, authors have to handle focus event and change the behavior by script, while with `tabStop` attribute authors can define the behavior declaratively, and it is definitely easier and simpler. Therefore this can be added later if there is viable use case that `tabStop` alone cannot control the subtle behavior.
3. Add `tabStop` attribute to Element
   This is the initial Blink implementation tried to achieve this. This worked to some extent, but was confusing, as `tabStop` attribute value is only honored for shadow hosts and not applicable to e.g. focusable `<div contentEditable="true">`. Also, setting `tabindex` attribute will also change `tabStop` as well to maintain backward

compatibility, which was also confusing and have problems parsing attributes when both attributes are specified on an element (e.g. `<my-element tabindex=0 tabstop="true">`.)

Discussions / Limitations / Unresolved issues

- No focusable element in shadow but `delegatesFocus=true`, what is expected for focus() / tabbing?
  2 options: 1 - skip this element completely, 2 - focus the host element itself. We preferred 1 for tabbing, 2 for focus(). This may be confusing.
- There is a new proposed `:focus-within` pseudo class (spec) which might be used instead of extending `:focus`.
- Consider `aria-activedescendant`, and a11y in general.
- Shadow DOM spec does not explicitly say about the behavior when `tabindex=-1` is specified for a shadow host. As currently implemented in Chrome, only the shadow host is skipped from the sequential navigation order but the nodes in its shadow tree are visited. However it should make sense sequential tab navigation order completely ignore the descendent shadow trees under the shadow host. (additional homework) In that case, tab navigation order within shadow tree will not escape to its host's tree scope? (filed as GitHub issue #86)
- Make the behavior of `delegatesFocus=true` on shadow hosts default? (GitHub issue #105)
- Due to HTML<->DOM reflection, `<a>` behavior cannot be explained via custom elements fully (when `href` attribute is specified, `focusable` turns `true`, while `href` attribute is removed, `focusable` turns `false`), because `focusable` is a read-only attribute.
- What about `onfocus` event handling for `delegatesFocus=true`? The event does not propagate over shadow boundary, thus when an inner element is focused, the host doesn't get focus event, while CSS `:focus` starts to match. Inconsistent?
- Shall we move this doc to GitHub?

Demo

https://takayoshikochi.github.io/tabindex-focus-navigation-explainer/demo-date-input.html
(work-in-progress, the demo still uses old "tabStop" property)
You see <date-input> <input type=date> fields. The former is built with web components (source), the latter is native implementation.

Revision History

Oct. 30, 2014 first version
Dec. 26, 2014 renamed focusPassthrough to delegatesFocus

Jan. 6, 2015 Clarified behavior details of focus(), autofocus, mouse click, CSS :focus when delegatesFocus=true, and declared it is effective only when it is specified on a shadow host.
Jan. 14, 2015 Posted to public-webapps; first public version.
Jan. 21, 2015 Changed title, based on suggestion from Domenic Denicola, and rewrote some contents along the line.
Feb. 5, 2015 Changed the IDL interface from isTabStop() method to isTabStop attribute. Added a table for tabindex/isTabStop combination and expected behaviors.
Feb. 10, 2015 Removed sentences about override'ability of isTabStop attribute.
Feb. 13, 2015 Fixed behavior when non-clickable element in shadow tree is clicked (shadow host gets focus).
Feb. 20, 2015 Added reference to W3C bug 28054 for shadow host default behavior.
Mar. 19, 2015 Clarification about HTML attribute and DOM property reflection, renamed `isTabStop` to `tabStop`. Added a limitation about explaining <a> with custom element + tabStop.
Jun. 1, 2015 Major update to replace `tabStop` with `delegatesFocus`. Now it can be specified for shadow roots only, via `createShadowRoot()`'s parameter. `tabindex="-1"` on shadow host is now defined to skip all the subtree.
Jun. 17, 2015 Added `ShadowRoot.setForceFocusable()` to toggle shadow root's `focusable` programattically.