spatialNavigationSearch API Explained

Written: 2019-11-13, **Updated**: 2019-11-19

spatialNavigationSearch() is a proposed API to get a next target element to be focused via directional inputs. It returns a focusable element as a best candidate element in a given direction. It's only enabled behind a runtime flag (--enable-spatial-navigation) so that it can be used in specific user agents such as smart TV with remote control and touchless smartphone. There seems no performance and security issue as it's just get method.

In CSS Spatial Navigation specification, there are several proposed APIs such as DOM methods, CSS properties and DOM Events. We would like to focus on spatialNavigationSearch API as the most primitive one here.

See also:

- specification, official working draft published in W3C CSS WG.
- demo, working with polyfill located in W3C WICG.

Background

Spatial navigation is the ability to navigate between focusable elements based on their position within a structured document. Spatial navigation is often called 'directional navigation' which enables 4-ways (top/left/bottom/right) directional navigation. Users are usually familiar with the 2-way navigation using both 'tab key' for the forward direction and 'shift+tab key' for the backward direction, but not familiar with the 4-way navigation using arrow keys.

In terms of TV remote control, arrow keys on touchless smartphone and Web accessibility, the spatial navigation has been a rising important input mechanism in several industries. If the web could embrace the spatial navigation functionalities in web engines (i.e. Chromium) and W3C standard APIs, it will be more promising technology for existing products as mentioned above and various upcoming smart devices.

spatialNavigationSearch WebIDL

```
enum SpatialNavigationDirection {
    "up",
    "down",
    "left",
    "right",
};
dictionary SpatialNavigationSearchOptions {
```

```
    Node? container;
    sequence<Node>? candidates;
};
partial interface Element {
    Node? spatialNavigationSearch(SpatialNavigationDirection dir, optional
SpatialNavigationSearchOptions options);
};

// An example of spatialNavigationSearch API
// let elm3 = document.querySelector("#elm3");
// elm3.spatialNavigationSearch("right", {container: "container1", candidates: [elm1, elm2,
elm4, elm5]};
```

spatialNavigationSearch in detail

Basic logic

To properly understand the logic of spatialNavigationSearch, you need to briefly know the current basic processing model that has already been merged in Blink. With the processing model, users moves the focus to the next best target via arrow keys. spatialNavigationSearch is an option that authors can override a default next target to a changed target when pressing arrow keys.

In the processing model, there are three steps to find a next focus target as follows:

1. When the current focused element is a spatial navigation container, a set of candidate elements to be focused become descendant elements of the container element.
2. When the current focused element has some fully or partially overlapped elements, a set of candidate elements to be focused become those overlapped elements.
3. Otherwise, the current focused element has a set of candidate elements in a given direction.

Difference with the processing model

spatialNavigationSearch works similar to the processing model (i.e default behaviors when pressing arrow keys), but the followings describe the difference between the processing model and spatialNavigationSearch.

1. The processing model works recursively when there is no candidate elements in a nearest spatial navigation container. However, spatialNavigationSearch doesn't work recursively so it returns null when there is no candidate elements in a nearest spatial navigation container. Authors can use spatialNavigationSearch APIs in a flexible ways.

2. The processing model considers only visible elements as candidate elements to be focused. However, spatialNavigationSearch can consider invisible elements (i.e. offscreen elements) within a scroll container. For the operation, authors can add those offscreen elements into the candidates parameter.
3. The result of processing model could be scrolling when pressing arrow keys. However, spatialNavigationSearch doesn't do scrolling. In that case spatialNavigationSearch returns null.

Parameters

spatialNavigationSearch API provides three parameters, though two of those are inside the options parameter.

- dir: It has a trivial role as a mandatory parameter. It could be mapped to each arrow key.
- container: It's an optional parameter. If container is specified, the scope of candidate elements to be focused is limited within the container. If container is not specified, the container sets to the nearest spatial naivation container element as a default.
- candidates: It's an optional parameter. If candidates is specified, the scope of candidate elements to be focused is limited among the candidates. If candidates is not specified, the candidates sets to all the descendant elements of the container.

Note: If the intersection between container and candidates specified in each parameter is null set, spatialNavigationSearch returns null. It works as intersection, not union of two parameters.

Sample code

To simply understand the behavior of spatialNavigationSearch() API, the looping example is shown as follows.

```
<div id=container>
  <div id="elm1" tabindex="0"></div>
  <div id="elm2" tabindex="0"></div>
  <div id="elm3" tabindex="0"></div>
  <div id="elm4" tabindex="0"></div>
  <div id="elm5" tabindex="0"></div>
</div>

<script>
const ARROW_KEY_CODE = {37: 'left', 38: 'up', 39: 'right', 40: 'down'};
document.addEventListener('keydown', (e) => {
  let key = ARROW_KEY_CODE[e.keyCode];
  let next = e.target.spatialNavigationSearch(key);
```

```
  if (next === null) {
    if (key === "right") elm1.focus();
    if (key === "left") elm5.focus();
    e.preventDefault();
  }
});
</script>
```

The example code works as the followings: (gif image) spatial navigation on any websites

Frequently Asked Questions

Isn’t it enough just using the relevant web frameworks?

Several web frameworks and extensions for the spatial navigation have been provided until now due to no support from web engines. For examples, [js-spatial-navigation] made by Mozilla is one of the frameworks and its quality would be good to support the features of the spatial navigation. [Spotlight] library developed by LG is also an instance of the frameworks for the spatial navigation. However, the support of spatial navigation from web frameworks has some limits as follows:

- Limitation from JS capability
    - Form elements (slider, text input, etc.), Shadow DOM, OOIPFs
    - e.g. no way to get the current caret position or the current slider bar for now
- Uncertainty on animated pages
    - With animated object on a web page, Blink SpatNav is better than JS SpatNav in terms of responsiveness.
- Uncertainty on keydown handlers
    - Author-defined keydown handlers could affect to the existing SpatNav operation provided by JS SpatNav.
    - e.g. preventDefault in the author-defined keydown handeler could cancel the JS SpatNav logic as well.
- Inconsistency between several JS frameworks
    - Several JS SpatNav provides a bit different user experience. (e.g. rules for next target selection)
- Difficult to align native scroll behaviors with SpatNav when moving the focus to offscreen element
    - With JS, a bit complex codes are needed to support the offscreen element management in a scroll container.
- Difficult to align the virtual focus management with a11y supports

    - The a11y functionality works based on the focus movement, but some JS SpatNav works with virtual focus.
- Performance degradation due to a bit expensive cost of DOM Access whenever pressing arrow keys
    - In case of the huge number of focusable elements in a web page, it works slowly when pressing arrow keys

Spatial navigation seems not a general functionality on the Web. (only for TV?)

- Spatial navigation has a history about for 20 years. First, the feature phone has been a first citizen of the spatial navigation about ten years ago. Before touch-based interface, the majority of interface for mobile was the key-based interface. Recently (as of 2019), KaiOS-based feature phones have been propagated mainly in several developing countries such as Africa, India and Southeast Asia. The device doesn't support a touch interface (touchless), so users need to use the arrow keys to select an item on all applications including the browser.

- In the future, the input mechanism for smart devices will be changed to something like voice command, hand gesture and gaze direction, but the key-based interface will never disappear, even though it would be used as a secondary method. The key-based interface used to be evaluated as one of the most intuitive ways with a strong sense feedback of finger after pushing a key, as if it's inconvenient a touch-based keyboard without any physical keys. The other industries such as IVI(Car), AR, game console could be also future citizen of spatial navigation.

I would like to raise an issue or idea about spatial navigation.

- Please put any question via the following two links:
- Issues in CSS WG for spec issues with css-nav label
- Issues in WICG WG for any other issues
- Everything for the spatial navigation is welcome! :D

Original link

https://github.com/WICG/spatial-navigation/blob/master/docs/spatialNavigationSearch_explainer.md