Reorderer Shift Algorithm

Overview

These notes describe the new "shifting" algorithm for the Reorderer component. This algorithm provides the Reorderer with the smarts to move around reorderable elements spatially while respecting the original DOM location of elements that are not marked specifically as reorderable. This expresses the approach of a typical markup-driven component where arbitrary HTML markup can be passed to the component and manipulated in a manner that is less likely to disrupt the original author's presentational and structural intentions.

Purpose

Given:

1. A document that contains a set of elements that are marked as reorderable.
2. An enclosing element that clearly demarcates the set of elements that are reorderable.
3. That the enclosing element contains other elements that are not reorderable.
4. That the movement of one of the reorderable items to a new position within the reorderable set causes items before and/or after that position potentially to shift.

Then:

- The shift algorithm is responsible for determining the new spatial location of the shifted reorderable items.
- Place them in their new positions.
- Without affecting the position of the elements that are not reorderable.

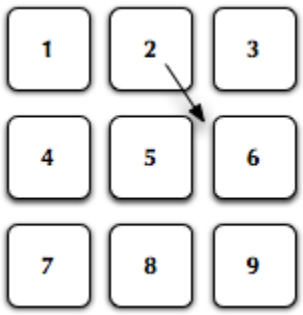
Commentary

The Lightbox is a concrete example of a reorderable set of images that are displayed in a grid. The Lightbox HTML markup consists of a linear set of elements, each containing an image and a caption. Each of these elements is termed a "thumbnail". These thumbnails are arranged visually in a grid, and their spatial location in the display is determined by the associated CSS. That is, each reorderable items' spatial location is determined by CSS, and it is the CSS that causes the set of items to appear in rows and columns.

The Reorderer ideally should know nothing about the fact that it is organizing thumbnails within a Lightbox. All it needs to know is that there is a set of reorderable items, and its job is to reorder those items. The first two parameters noted above are a document containing a set of reorderable items, and an element that contains all of these items. Again, using the Lightbox as a specific example, the main Lightbox contains the items that are reorderable. Requirement three states that this enclosing element may also include elements that are "fixed" in the sense that they are not reorderable. In DOM-speak, the enclosing element is the root of a branch in the DOM that contains the reorderable items, intermingled with other elements.

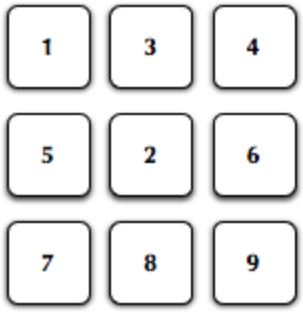
The issues that the Shift Algorithm addresses is how re-render the spatial positions of reorderable items after one has been moved, and what the new order of the items are in the DOM. Figures one and two illustrate a simple case: imagine a three-by-three grid containing nothing but reorderable elements. The user is about to move the second item in the first row to just after the fifth item, as indicated by the arrow in Figure one.

Figure 1. Spatial position of elements on the page before shift.



After the move, element 2 occupies the grid cell that was occupied by element 5. That entails that some of the other items must be shifted, while others are unaffected. Element 5 is shifted to the left and replaces element 4's position. Element 3 also moves to the left to occupy the cell vacated by element 2 (the moved element), and element 4 shifts to element 3's former position. The spatial location of element 1, and elements 6 through 9 does not change.

Figure 2. Spatial position of elements on the page after shift.

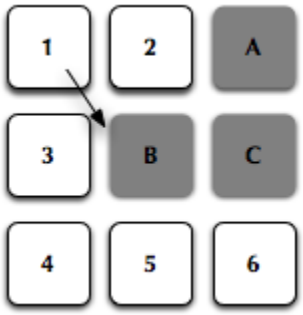


Generalizing from the specific case of a Lightbox, the spatial layout is what is primarily affected by the reorderer. In some sense, the DOM does not matter, since, in the limit, the spatial locations of reorderable elements can be whatever the CSS dictates, and is impossible to derive simply from its model position within the DOM.

Presence of Items that are not reorderable

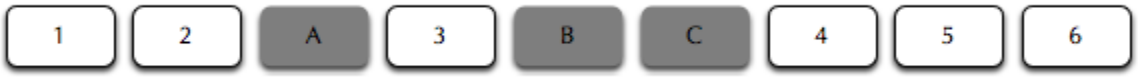
In a more complex case, the reorderable items are intermingled with elements that are not reorderable. This is illustrated in figures 3a and 3b below. In the figures, reorderable items are represented by white numbered rectangles, while elements that are not reorderable are shown as grey rectangles labelled with a letter. In these illustrations, elements A, B, and C are not reorderable.

Figure 3a. Spatial position of elements on the page before shift.



The difference between figure 3a and 3b is that figure 3a shows the spatial rendering of the elements, whereas figure 3b shows their order within the DOM. The arrow in figure 3a indicates that element 1 is to be moved to just after element 3, or, alternatively, between reorderable element 3 and a fixed element B.

Figure 3b. DOM order of elements before shift.



Figures 4a and 4b show the results of the shift after moving element 1 to its new location. With regard to spatial location (figure 4a), the cell occupied by element 1 previous to the move is now empty, and element 2 is shifted left to occupy it. This entails that element 3 is moved up from the beginning of the second row to now occupy element 2's previous location. Element 1, whose movement initiated the shifting, now occupies element 3's old location. Otherwise, all other elements, reorderable or otherwise, stay in their original spatial positions.

Figure 4a. Spatial position of elements on the page after shift.

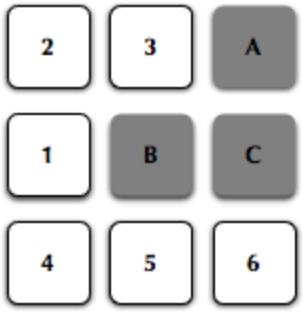


Figure 4b shows the new order of the all of the elements in the DOM. Perhaps not surprisingly, and similarly to the spatial re-location of elements, element 1 has been moved to element 3's former slot, and element 3 and 2 have been shifted up one slot each. The rest of the elements remain where they were originally.

Figure 4b. DOM order of elements after shift.

Figures 5a, 5b, 6a, and 6b illustrate a similar situation to that just discussed. In this case, element 1 is moved to just before element 3. This can be done, for example, using the Lightbox's drag and drop functionally, where the user selects item 1, drags it to a spot just before item 3, and drops it there. However, note that moving element 1 in this way is equivalent to moving it to a spot between element 2 and element A. Note further that, since element A is fixed, this amounts to moving element 1 to a spot between elements 2 and 3.

Figure 5a. Spatial order of elements before shift.

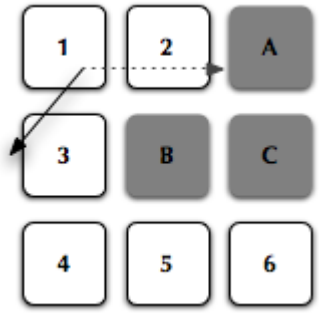
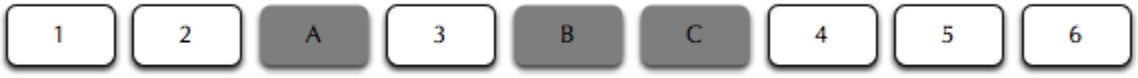


Figure 5b shows the DOM order of the elements prior to the move. It is the same situation as that in the previous example, as shown in figure 3b.

Figure 5b. DOM order of elements before shift.



After the user drops element 1 as illustrated in figure 6a, the shifting algorithm repositions the elements as follows. Since element 1's cell is now unoccupied, element 2 shifts left to occupy it. Element 1 is now positioned "between" elements 2 and 3; and, the remaining elements stay in their original locations.

Figure 6a. Spatial order of elements before shift.

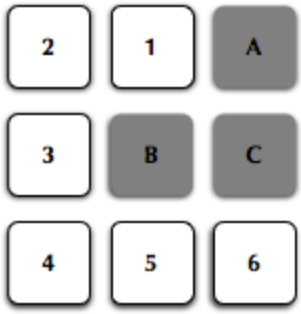
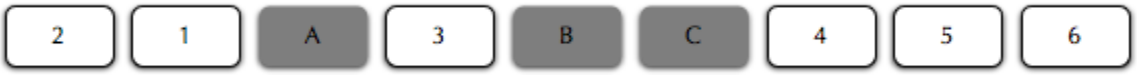


Figure 6b shows the new DOM order of the elements. In summary, moving element 1 to just before element 3 is tantamount to switching the positions of elements 1 and 2.

Figure 6b. DOM order of elements after shift.



Summary

The Reorderer's shifting algorithm allows it to be a truly generalized solution for reordering arbitrary streams of markup. By providing a mechanism by which reorderable elements can be moved around freely without disrupting the placement of non-reorderable elements, the Reorderer is suitable as a building block for creating all kinds of UI components that allow the user to directly manipulate objects on a web page.