Helsingin saavutettavuusmalli

Helsinki Model for Accessible Service Design

Link Box Pattern: Notes on technical implementation

Tero Pesonen

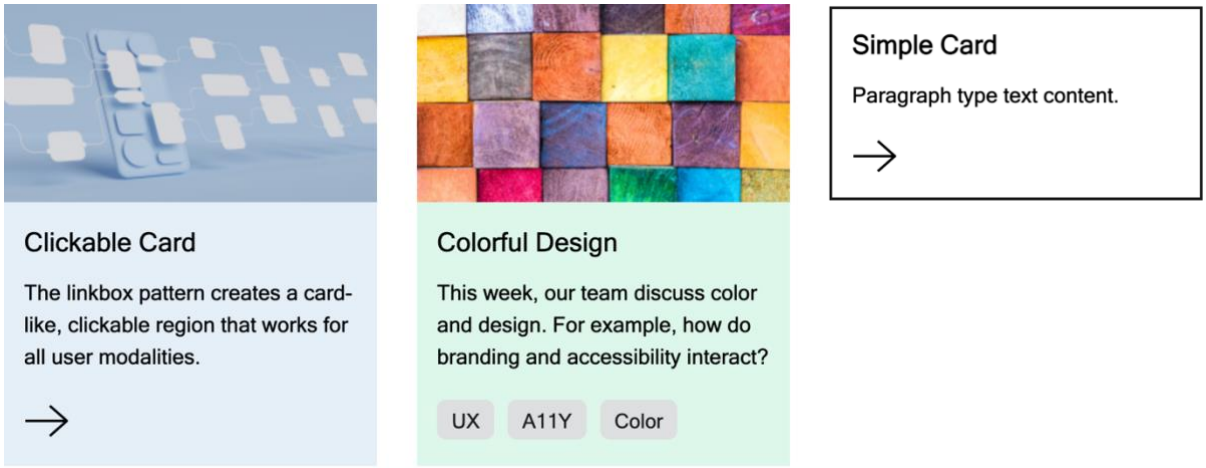
Siteimprove

Version: 6th March 2023

Link Box Pattern

Examples

Examples can be found on the demo page: http://siteimprove-accessibility.net/Demo/Page/



Motivation for the pattern

The linkbox is not a specific element, but a design pattern that is commonly found on modern web sites. UX designers often find it useful to enclose a link with supporting content that is not part of the link text per se. These content types can be a heading, text blurb, image, and even tags describing the type of content linked (e.g. topics of a news piece, blog post, editorial, etc.). Ordinarily, these affiliate content pieces pose no trouble: They are simply laid out as a card or box together with the link text. Moreover, one of these supporting elements can serve a double duty as the link as well (e.g. the heading).

From accessibility point of view, challenges usually only emerge when the design calls for a card or box that is clickable on its entire area. That is, the user can place a pointer over any part of the card and click to activate the link. The same should happen with a keyboard. There is, hence, no specific linked heading or image or text within the box, but all of its area is interactive, and all of the area activates the same link.

A common implementation of this UX pattern is to enclose the entire box in an anchor (link) tag, as that logically renders the box interactive, and the box will behave as described above. Unfortunately, the solution is not always very good for assistive technology users. It can work on very small cards that have but a text blurb, or a heading and a couple of sentences, but even then it limits the design to such use cases only.

One would, rather, prefer a more generic pattern that adapts to any type and amount of content within the box. Individual implementations (small card, large content-rich boxes, etc.) can then be derived from this overall “superclass” of a linkbox.

This document demonstration such a pattern that is also easy to implement.

Accessibility challenges

When the entire box is wrapped within an anchor tag (<a>), the content of the box effectively becomes the accessible name of that anchor. This means that a screenreader can flatten the content into a single string of text that is presented (and often read out loud as a single atomic piece) to the user when they focus the link.

Depending on the SR, the user may or may not be able to parse this content. They may also lose access to the semantical information of each content type. This means that such roles as image, heading, paragraph, “tag group”, etc., disappear.

Clearly, for a link card that has more than one or two pieces of content, this is undesirable. We would like the user to receive all the same semantical and structural information as is available visually.

Pattern solutions

The most straightforward technique to solve the above problem is to dispense with the wrapper link altogether. Instead, one places the anchor element inside the box as its last child. The trick is to place the link with absolute positioning at the top left corner of the card and spread it out to cover all of the box area. (This could be achieved with a pseudo element as well.)

Now, a SR user can parse the box content as normal web page elements, and the user will then see the link as the box’s last element. The link’s name can be given with an aria-label or it may reuse a visual piece of text (a heading, call-to-action button, etc.) from the box. See the examples below.

It is now also straightforward to give keyboard-only users a desired focus indicator as well (is the entire card outlined, for instance, or a particular element within?). Since there Is but one link in the entire box, there is no need to remove any content with tabindexing from the tab order. What is more, the box can now accommodate any number and type of content and yet retain the same degree of technical accessibility.

As a final note, it may be beneficial to define the box as a named region. For larger boxes, in particular, naming may help non-visual users to understand that they have entered – and exited – a link card that behaves unlike a standard page content segment. This does not, of course, replace a good heading structure, which is required regardless, but rather supports it.

Pattern implementation

1. Place all the link card/box content inside a wrapper (such as a or <section>)

   Optional: Give the box a name like so:

   ```
   <div role=”region” aria-label=”Link card” …>
   <section aria-label=”Link card” ..>
   ```

   You can even use aria-labelledby to add the heading (requires an id attribute) to the box’s name:

   ```
   <section id=”linkbox1” aria-label=”Link box” aria-labelledby=”
   linkbox1-heading linkbox1” ..>
   ```

   Small cards could use a `group` role instead,so that they are not made landmarks.

2. Give each child element inside the box its normal tag or role definitions.

    a. Tags can be placed inside a group or list to distinguish them from the other text content:

    ```
    <div role="group" aria-label="Tags">
    <ul aria-label="Tags">
    ```

3. **For image(s)**, an empty ALT (alt="" or just alt) is common. It marks the image as invisible to assistive technology since the link (box) purpose is usually clear even without it.
4. Make sure that the **DOM order** corresponds to a logical **reading order** for non-visual users. If necessary, use CSS if there is need to present the content visually in an order that deviates from the tag's natural reading order.
5. Place the **anchor** tag as the last child element of the box. Then:
    a. give the box wrapper `position: relative`
    b. give the anchor `position: absolute`, and place it e.g. `top: 0px` and `left: 0px`.
    c. give the anchor width and height of 100%.
    d. Apply any desired focus, pointer (or even hover) styles.

    (A CSS pseudo link should also work.)
6. The link accessible name can be defined in a number of ways. The choice depends on the box structure and purpose.

Link accessible name

The name can be given with:

- <a aria-label="Arbitrary name string" ..>
- <a aria-labelledby="id1 id2 …"> → build the name from referenced element names, such as the box heading, which may be a natural link name as well.
- Place a link text inside the anchor:
    1 <a ..><cta-button>Read more</cta-button>

    Now, "Read more" becomes the anchor's name. As this name may be too vague for WCAG 2.4.4 "Link purpose in context", you can supplement it with e.g. the heading to make it unique to each box:

    2 <a id="linkbox1-anchor" aria-labelledby="linkbox1-heading linkbox1-anchor">

WCAG 2.5.3 Label in name

To make sure you do not violate WCAG 2.5.3, only use link accessible names that contain a string that is also visually present in the box and that a voice assistant user, for instance, might use to call out when wanting to click on the box. Incorporating the heading, if present, in the link name can be a good choice. Of course, you can extend the accessible name beyond the visible link name as long as the visible (heading or other) text is included.

- Example: The card title is "UX Design Seminar". The link name could be: aria-label="UX Design Seminar web page".
- Example #2 from above should also work, as it combines the generic CTA button name with the box heading for an accessible name that is not only unique but also visually present.