**Introduction**

* Signifiers are indicators that offer clues on how to interact with some object.
* Showing interactivity and clickability through signifiers is a fundamental aspect of user interface design.

**Browser Link Styles**

* Web browsers use signifiers for navigation through User Agent Stylesheets.
* User agent style sheets define default behavior for rendering content. These styles are used to ensure that a raw HTML file is styled to be reasonably readable by any user.
* User Agent stylesheets include styling for headings, tables, links, and more basic HTML elements. Most users do not see these styles too often, because designers override them with their own custom designs. However, it’s important to note that maintaining a consistent user experience pattern, like the default behavior applied by browsers, is important for creating a consistent experience.
* Traditionally, links are differentiated from regular text using blue text and underlines to draw users’ attention to their clickability. Many websites and user agent stylesheets still use these link styles.
* Browsers also style two other link states: clicked (‘active’), and previously visited. In most user agent stylesheets, clicked (but not yet followed) links appear with red text, and previously visited links are styled with purple text.
* a{
* color:blue;
* text-decoration:underline;
* }

**Link Styling**

* The most important aspect of styling links is differentiating links from surrounding text. The default blue-text, underlined link style accomplishes this differentiation using two CSS properties: color and text-decoration. These are both good ways to differentiate a link, and they are strong, familiar signifiers to most web users. Additionally, you could use CSS properties for background-color, font-weight, or border.
* Link styles should not be replicated in other page text. Link color should sufficiently contrast the text colors in the rest of the design. For example, if links are underlined, other text should not be.
* Provide descriptive anchor text.

**Tooltips and Titles (use title attribute for tooltips on a selector)**

* Additional context for links can be provided using tooltips, images, icons and non-text element.
* For tooltips we use [HTML title attribute](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/title). Although the title attribute can be provided to any HTML element, it is often extremely useful as additional context or advisory text for clickable elements.
* Most browsers will display the text of a title attribute as a [tooltip](https://en.wikipedia.org/wiki/Tooltip), meaning when a user hovers their cursor over an element, the text will appear in a small box near the cursor.
* To add tooltips to a clickable element like a link, add it as the title attribute.

<a href="https://www.codecademy.com" title="Codecademy is an online learning platform">Codecademy</a> is the best place to learn to code!

Mouse over the word “Codecademy” below to see this behavior in action!

[Codecademy](https://www.codecademy.com/) is the best place to learn to code!

**Hover States and Cursors**

* In addition to styling elements themselves, other signifiers and visual feedback can be utilized during user interaction.
* The [CSS pseudo-class :hover](https://developer.mozilla.org/en-US/docs/Web/CSS/:hover) can be used to style elements on mouse hover. For instance, to change the color of link anchor text from blue to orange when a user hovers over it, the following CSS could be used:

a {  
  color: blue;  
}  
   
a:hover {  
  color: orange;  
}

The first rule sets link colors to blue by default, and when a user mouses over a link, the second rule will override the color attribute of the <a> tag and cause the text to turn orange. When the user moves the cursor away from the link, the text color will revert to blue.

* In addition to any text style changes when hovering over a link, the user’s cursor should change from the default appearance to a pointing hand. The [CSS cursor property](https://developer.mozilla.org/en-US/docs/Web/CSS/cursor#Examples) is used to control this behavior. For example, to add this behavior to all <a> tags, the following rule could be used:

a {  
  cursor: pointer;  
}

Luckily, this behavior is generally included in browser user agent stylesheets, and it also exists in the [HTML5 default styles](https://www.w3.org/TR/html5/rendering.html).

* Hover state styling should never be used as the sole indication that something is a link. Users should not be forced to move their mouse over an entire document to tell what might be clickable. Additionally, hover states are not accessible in mobile browsers. Mobile devices do not generally have on-screen cursors, and users must actually touch the device (and possibly trigger a click event) to interact.
* Some Values for cursors

cursor: grab;

cursor: help;

cursor: wait;

cursor: crosshair;

cursor: not-allowed;

cursor: zoom-in;

cursor: grab;

**Link States (Pseudo Classes)**

* Links have four main states: normal (not clicked), hover, active (clicked), and visited. These four states have associated [CSS pseudo-classes](https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes): :link, :hover, :active, and :visited. These four states can be used to give a full range of visual feedback to users about the status of their link interaction.
* Each state should still make it clear that the element in question is a link, meaning it should not make text identical to non-link text or alter the link’s appearance so much that users could perceive its behavior differently.
* The ordering of link state pseudo-class rules is important to reveal the proper information. When a user hovers and then clicks a link, those styles should always override the static styling surrounding a user’s history with the link (unvisited :link and :visited). The proper order of these rules is:
  + :link
  + :visited
  + :hover
  + :active
* This ordering will ensure that the rules [cascade properly](https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Cascade_and_inheritance) and the user can receive the most applicable visual feedback about the state of the link.

**Buttons: Skeuomorphism and Flat Design**

Buttons are another fundamental element of user interaction and navigation. They are called ‘buttons’ because they are often modeled on the appearance and behavior of real-life buttons. The concept of UI elements that replicate or imitate real-life counterparts is known as *skeuomorphism*.

In design, skeuomorphism is helpful for lowering the learning curve for users. If users can draw a metaphor between a familiar real-life object and an interface element, they are more likely to know how to use it without training. In the example of the button, if a web button appears similar to a real-life button on a physical interface, users can reliably figure out that the way to interact with the button is to press it.

*Flat design* is an alternative approach to skeuomorphism which embraces the fact that computer interfaces do not necessarily need to model real-life interfaces. As users grow more familiar with digital displays and interfaces, designers have felt less need to model physical interactions and instead rely on other signifiers to indicate interactive elements. To generalize, flat design uses simplicity and lack of clutter for its UI elements.

**Buttons: Skeuomorphic styling**

Skeuomorphic button design aims to imitate the appearance and interactivity of a real-life button, often including a ‘raised’ appearance while the button is unpressed and a ‘pressed’ appearance when clicked.

Skeuomorphic button design can be implemented using image files, CSS rules, or a combination of both. CSS styles should be preferred over image files if possible, since they are faster to load, have smaller file sizes, and allow for a more consistent scaling and appearance across different screen sizes and resolutions. Modern CSS3 has support for many 2-D and 3-D effects and animations and can create many skeuomorphic button designs on its own.

If using CSS rules, the use of hover and/or active states is important in order to model interaction with a real button. For example, to implement a bare minimum 3-D button design, the following CSS ruleset could be used:

.button {  
  padding: 5px;  
  border: 1px solid black;  
  border-radius: 5px;  
  text-decoration: none;  
  box-shadow: 0px 5px;  
}  
   
.button:hover {  
  cursor: pointer;  
}  
   
.button:active {  
  margin-top: 5px;  
  color: black;  
  box-shadow: 0px 0px;  
}

A button element can then be created with the following HTML:

<div class="button">Click me</div>

The default state of the .button class has some basic ‘buttony’ appearance with a rounded border (border and border-radius properties) and a slightly raised appearance with the use of the box-shadow. The :hover cursor is added for visual feedback. When the button is clicked (:active), the box-shadow is removed, and the margin-top moves the button down by 5px, make it appear to be pressed.

The above example is only one very basic implementation of a 3-D button; there are many additional (and more attractive) ways to create skeuomorphic buttons.

**Buttons: Flat Design**

Flat design is so-called because of its 2-D appearance. Elements appear flat on the screen, and designers must rely on other styling and signifiers to indicate clickability.

Flat design buttons commonly appear as rectangles, rounded rectangles, or circles. These shapes are used ubiquitously for button elements, so users often perceive them as buttons and expect them to be clickable.

Since there are fewer obvious signifiers surrounding buttons in a flat design, they should be visually distinct from other page elements.

In flat designs in particular, button text is very important for clarity—the possibility of user confusion is reduced by pairing buttons with specific, descriptive labels. A button with ‘Click here’ leaves many more open questions than a button that reads ‘Submit form’.

**Buttons: Hover States**

Just as with links, buttons should make use of hover states and the cursor: pointer declaration. All the caveats about the shortcomings of hover states on mobile devices apply to buttons, but their use is still encouraged.

Users expect buttons to be clickable. Since buttons can consist of any number of total elements (rectangular/circular body, text, image(s)), all elements should be clickable, should display their clickability, and should result in the same behavior.

**Review**

Great work! You’ve made this survey site much easier to understand and interact with! Let’s review what changes you made to improve usability:

1. Added styles to make links recognizable and distinguishable from ordinary text.
2. Added link styles that depend upon mouse interaction state, providing users with visual feedback for cursor hovering and mouse clicks.
3. Added additional context text with the HTML title attribute.
4. Styled buttons to be easily recognizable (in both skeuomorphic and flat design styles) using box shapes, border, hover, and active states.

Remember that the styles that you implemented here are only one of countless ways to communicate clickability and provide users with visual feedback. There is a huge amount of opportunity for variety and creativity while still following best practices and common user experience patterns.