**CSS Advanced Selectors**

With powerful tools like the Chrome Developer tools aiding our workflow, it's important to understand how to write cleaner stylesheets with more advanced selectors.

Learning these selectors gives greater flexibility in selecting aspects of our HTML document, thereby enhancing user experience. MDN has a great list of the available selectors [here](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors). Take a look then read through an explanation of the most popular ones below.

**Attribute selectors**

We can target elements with specified attributes by using [] . This is a popular way to style form input elements based on their type attribute. This is useful because of how much input elements differ between browsers.

input[type="radio"] {

-webkit-appearance: none;

width: 27px;

height: 27px;

cursor: pointer;

background-image: url(../assets/stars.png);

vertical-align: bottom;

outline: none;

}

This example selects all input elements with type attributes equal to "radio" and styles several properties.

**N.B.** The -webkit-appearance property is native to the Webkit rendering engine used by Safari and Chrome. The -webkit- part of the property is called a vendor prefix. It's used to declare that this CSS property is from a proprietary rendering engine not yet declared by CSS W3 standards. Other vendor prefixes include -moz- for Firefox, -o- for opera, and -ms- for Internet Explorer.

The outline property is set to none to reset the default browser action of outlining the radio button in blue. If we wanted to do this for all radio buttons in our application, it may make more sense for this to be in the css\_reset.css file.

**Pseudo Selectors**

Most advanced selectors fall under this W3 category called pseudo-class selectors. Check out the [MDN documentation](https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes) for a full reference. Here we have broken down the popular pseudo-class selectors into our own categories for better understanding.

**State selectors**

Some state selectors refer to specific HTML elements like <option> and <select>.

input[type="radio"]:checked {

background-position: 30px 30px;

}

In conjunction with the example in the attributes selector, this example changes the position of the background when the radio button elements are in a **checked** state.

Another common state selector is :hover:

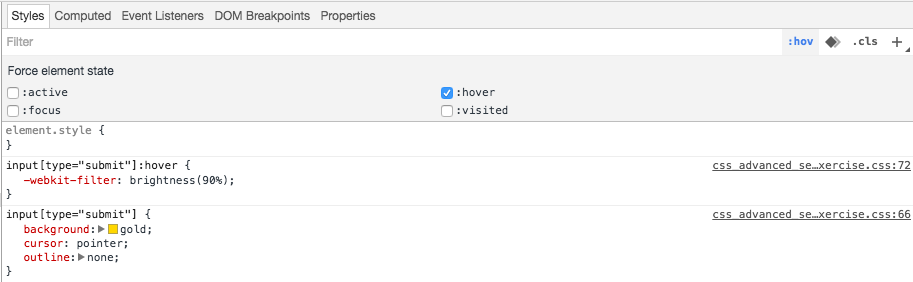
input[type="submit"]:hover {

-webkit-filter: brightness(90%);

}

This example targets the hover state of all submit buttons and uses another advanced webkit property to apply a brightness filter of 90%, thereby darkening the element. This effect adds to our user experience by encouraging them to press the button.

In the Chrome Developer Tools we can toggle element state in order to view styles being applied:



**Structural Selectors**

The following pseudo selectors target based on the structure of the HTML document.

table tr:nth-of-type(2n) {

background-color: whitesmoke;

}

This example uses nth-of-type(2n) to select every other table row element that is a descendent of a table element. It applies a greyish background color.

header img:first-child {

width: 100px

}

The above example could replace a header > img selector to permit more image elements in the header while still applying a width to the first one.

Structural pseudo selectors help target elements without adding extra HTML markup to our code, but beware breaking your structural selectors when making changes to your HTML document.

**Combinators**

We have already seen a couple examples of combining selectors with a space to select descendants or with a > to select children.

We can also use SelectorA + SelectorB to target all SelectorB elements that immediately follow SelectorA elements.

div ~ p {

background-color: gold;

}

This example uses the ~ combinator to make a general selection of subsequent siblings. It would select all <p> elements that follow and are within the same parent as any <div> elements.

There are plenty more selectors in the documentation. Most of the time the basics of CSS will be just fine, but for tough cases where you don't want to make changes to your HTML, these advanced combinators can sometimes really help.

s