CSS Selectors Cheat Sheet - Hackr.io

Whether you are an experienced professional or just a beginner, you can refer to this CSS advanced selectors cheat sheet that will help you brush up on your concepts whenever required. Also, it will help you improve your knowledge of using methods for selecting elements.

In this cheat sheet, we will learn about different types of CSS selectors with simple syntax and examples. But before that, we will take a look at CSS selectors.

What are CSS selectors?

CSS selectors are the crucial part of the CSS rule set that allows you to select the element you want to style.

Consider that you want to apply a specific style to 20 different elements. You normally have to apply the style and change each element one at a time — a real time burner! This is where CSS selectors come into play. They help you make a group of 20 elements based on their identities and apply CSS in a single place.

CSS selectors have helped front-end developers save a lot of manual coding time, allowing them to control and manipulate different CSS elements based on their properties.

Bottom line? CSS selectors allow you to apply a specific style to multiple elements simultaneously with inline CSS using the ‘style’ attribute.

For example:

```
p, h1, #my_id {

        font-weight: bold;
        text-decoration: underline;

}
```

CSS Simple Selectors

CSS simple selectors in CSS do not come with much logic. They are simply used to locate the element with the help of the identifiers and then implement the style accordingly.

CSS Tag Selector

The simplest CSS selector that you can apply is to use the “tag” from the HTML (predefined HTML tags). With the help of the CSS tag selector, you can easily catch all the tags with just a single line of code.

For example:

```
<style>
    h1 {
            color: red;
            font-size: 50px;
          }
  </style>
```

As mentioned in the above example, we selected the “H1” tag from the HTML. We tried to establish the text color as red and changed the font size to 50px.

Whether experienced or beginner, you will regularly use the CSS single selectors for simple changes.

CSS ID Selector

The CSS ID selector is another widely used selector that helps the developers style web pages. Here, the “id” selector specifies the element id that will help implement the styling.

You can use the ID to tag along with various elements. And, you can use CSS selectors or JavaScript to select the elements to carry out desired tasks.

For example, if you want to change a page or section color, you can use the same id for all the elements and implement the functions. CSS ID selectors are used with IDs with the symbol “#” at the beginning.

```
<style>
    #my_id {
            color: red;
            font-size: 50px;
          }

</style>
<p id = "my_id">CSS Selector</p>
```

With the help of the above example, you can easily change the color and font size of all the elements tagged with the id name “my_id.”

CSS Class Selector

The CSS class selector allows the developers to select all the available elements within a particular class name (similar to the id selector). Even if the class name has a single element,

the styling will be applied to it. Like the IDs, class names also help you select multiple elements based on the class name.

In the case of some frameworks, such as Bootstrap, you will get some predefined classes for particular elements, such as “large buttons.” You can seamlessly apply the CSS using the appropriate class selector, and automatically apply the styling accordingly. Thus, the CSS class selector is faster than the CSS ID selector.

```
<style>
    .my_class {
            color: red;
            font-size: 50px;
          }

    </style>
<p class = "my_class">CSS Class Selector</p>
```

The “.” symbol specifies a class name in CSS, and “my_class” is used to select the elements.

How to Group CSS Selectors

Most web page elements come with the same styling configurations to ensure everything belongs to the same space and feels connected. So, developers group CSS multiple selectors rather than copying the configurations multiple times for different classes and IDs. In such a way, the web page will understand that the specified elements share the same styling.

Consider the following example: we grouped the “my_id” and “p” to color them red.

```
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Web Template</title>
    <style>
    #my_id, p {
            color: red;
          }

    </style>
  </head>
  <body>
    <center>
        <h1 id = "my_id">I am a heading</h1>
        <p>I am a paragraph</p>
```

```
    </center>
  </body>
</html>
```

As shown in the above code. You must separate multiple sectors using commas to group them.

Now that we have discussed simple CSS selectors and how to group them, let's look at advanced CSS selectors.

CSS Universal Selectors

Font, styling, and other parameters make each website look different. Font type is a crucial factor in establishing a website signature. A unique and out-of-the-box font type helps customers remember their experience. You can change the complete outlook of the website by **changing its font type entirely.**

Implementing the same settings manually everywhere on the web page might be challenging. But, with the help of the CSS selectors, you can easily reduce this repetitive manual task. Developers can use the CSS universal selector to **select everything available on the web page.**

```
<style>
    * {
            color: red;
          }

 </style>
<h1 id = "my_id">I am a heading</h1>
<p>I am a paragraph</p>
```

In this example, the universal selector selects everything and changes their color to red.

CSS Specific Class Selector

There might be some scenarios where developers want to select only a specific element rather than all elements with the same class. Suppose they just want their "p" tag in red, but they have other elements with the same class name and tag name defined.

In this case, they can use a specific class selector to select only a specific element of the same class.

```
<style>
```

```
    h1.my_class {
            color: red;
          }

    </style>
<h1 class = "my_class">I am a heading</h1>
 <p class = "my_class">I am a paragraph</p>
```

In the above example, all the elements are in the same class “my_class.” But, you need to color H1 as red.

CSS Combinator Selector

This CSS selector is the combination of two or more selectors described above. You can use this selector type to define the relationship between two simple selectors and select only those elements that satisfy the relationship.

The following are various types of CSS combinator selectors.

CSS Descendant Selector

The descendant relationship specifies that the element should be inside another element. This selector type works on the descendant of specified elements. To write the descendant selector, you need to separate two elements with space where the second element must be a descendant of the first element. The implied style will be applied to the descendant element only if the descendant relationship exists.

```
<style>
    div p {
            color: red;
          }
    </style>

  <div>
          <h1>I am a heading</h1>
          <p>I am a paragraph</p>
   </div>
```

In the above example, only the content mentioned within the “p” tag will turn red since it is a descendant of the “div” tag.

CSS Child Selector

This type of selector works similarly to the CSS descendant selector; however, the second element needs to be the first element's child, not descendant. You can define the child selector using the ">" symbol. This symbol specifies that the second element is the child of the specified first element.

Here's an example to help you understand the difference between the child and descendant selectors.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Web Template</title>
    <style>
    div > h3 {
            color: red;
          }

    </style>
  </head>
  <body>
    <br><br>
    <center>
        <div>
          <h3>I am a child</h3>
          <h3>I am a child</h3>
          <span><h3>I am a descendant</h3></span>
          <h3>I am a child</h3>
        </div>
    </center>
  </body>
</html>
```

As per the above example, the third "p" tag is inside the "div" tag but not directly. Therefore, the third "p" will not be considered a child of the "div" tag, whereas it is a child of the span tag. Hence the third "p" tag does not get the red color.

CSS Adjacent Sibling Selector

Another type of CSS combinator selector is CSS adjacent sibling selector. This selector selects the element directly, followed by the first element. If there are other sequences of elements available, all are ignored and it only considers the sequence directly, followed by the element.

Make sure the element is followed rather than being inside another element. You must use the + symbol to use the adjacent sibling selector.

See the example below. We can see the adjacent “h3” tag inside the “div” tag.

```
<style>
    div + h3 {
            color: red;
          }
</style>
<div>
          <h3>I am not adjacent</h3>
          <h3>Me neither</h3>
 </div>
 <h3>I am adjacent</h3>
```

Once you run the above code, you will see that the content of the “H3” will turn red in the browser.

CSS General Sibling Selector

This type of combinator selector is useful when the developers want to select all siblings simultaneously — the generalized form of the adjacent sibling selector in CSS.

For example:

```
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Web Template</title>
    <style>
    div ~ h3 {
            color: red;
          }

    </style>
  </head>
  <body>
    <br><br>
    <center>
        <div>
          <h3>I Am Not Adjacent</h3>
          <h3>Me Neither</h3>
```

```
        </div>
          <h3>I am adjacent</h3>
          <h3>Me Too</h3>
    </center>
  </body>
</html>
```

As per the above example, all the siblings of the “div” tag named “p” will have their color changed to red. Note that sibling, children, and descendant here references the document object model (DOM) of the webpage.

CSS Pseudo-Class Selector

These selectors are useful for styling the elements specifying some conditions. Pseudo-classes will help you control the elements in particular states rather than without any condition, as the CSS simple selectors do.

For example, the pseudo-class hover will select the elements **only when the mouse is moved over to the element.** With Pseudo-classes, developers can style elements quickly without needing to use JavaScript.

CSS Link Pseudo Class Selector

Developers can use this link selector to style the unvisited link of the element they attach this class to. To use this selector, you must use the “link” keyword and pseudo-class symbol “:”.

```
<style>
    a:link {
            color: red;
          }

    </style>
  <a href = "www.google.com">This is an unvisited link</a>
```

As per the above example, the link will be in red until visited. Here, “href” plays a crucial role in the link pseudo-class. Even though the anchor tag is used for the link, it will not be recognized without the “href” tag.

CSS Visited Pseudo Class Selector

As mentioned above, the link pseudo-class selector works on the unvisited links. But, once you click the same link again, the visited flag will turn on, letting users know that they have already visited that link.

This type of pseudo-class selector will apply the style to the visited links. To use this selector, you need to use the keyword “visited” and “:” symbol for the pseudo-class.

```
<style>
    a:visited {
            color: red;
          }

    </style>
<a href = "https://www.google.com" target="_blank">This is a link</a>
```

As per the above code, the visited link will turn red.

CSS Hover Pseudo Class Selector

The hover pseudo-class selector works whenever the mouse hovers over to the element you have attached to the selector. Developers use it to bring out unique style, such as changing image size or background color. To use this selector, you need to use the keyword “hover” and the “:” symbol for the pseudo-class.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Web Template</title>
    <style>
    .my_div {
      background-color: blue;
      width: 300px;
      height: 300px;
    }
    .my_div:hover {
          background-color: red;
      }
    </style>

  </head>
  <body>
    <br><br>
    <center>
        <div class="my_div"></div>
    </center>
  </body>
```

```
</html>
```

As per the above example, the div box includes the hover tag that changes the background color from blue to red once the user hovers the mouse over it.

CSS Active Pseudo-Class Selector

This selector type allows the developers to select the elements with the mouse-click. Whenever you want to use this selector, you need to use the “active” keyword and the “:” pseudo-class symbol.

```
<style>
    .my_div {
      background-color: blue;
      width: 300px;
      height: 300px;
    }
    .my_div:active {
         background-color: red;
      }
    </style>
  <div class="my_div"></div>
```

You can run the above code to observe the clear difference between the hover and active pseudo-classes.

CSS Checked Pseudo-Class Selector

Developers prefer this selector when they want to select an element in its “checked” state. To apply this selector, the “checked” state is necessary. It works on the checkbox and radio buttons only.

Use the “checked” keyword and “:” symbol for this selector.

```
<style>
      input[type = checkbox]:checked + label{
        color: red;
      }
      </style>
  <input type="checkbox" class="my_button" value="Radio Button">
   <label for = "check_button">Radio Button</label>
```

The code will change text color from black to red in the above example once you check the checkbox.

CSS First-Child Pseudo-Class

This type of CSS selector allows the developers to select only the first child among all the occurrences of the elements. You need to use the keyword “first-child” and the “:” symbol to use this selector.

In the following example, the first “p” child will be selected on the web page.

```
<style>
      p:first-child{
        color: red;
      }
      </style>

    <p>This is the first child</p>
    <p>This isn't</p>
```

As per the above example, the first child will be selected and turned red.

Take a look at this next example that uses CSS selector groups and the first-child pseudo-class.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Web Template</title>
      <style>
      ul li:first-child{
        color: red;
      }
      </style>
  </head>
  <body>
    <br><br>
        <ul>
          <li>I am the first-child</li>
          <li>Am I?</li>
          <li>Am I?</li>
          <li>
            <ul>
```

```
            <li>I am the first-child</li>
            <li>Am I?</li>
            <li>Am I?</li>
          </ul>
        </li>
      </ul>
  </body>
</html>
```

Other CSS Pseudo-Classes

The following are other available options for the pseudo-classes that developers can use to make their website more interactive and apply more features.

Pseudo-Class	Meaning	Example
: disabled	When the input is disabled	Input: disabled
: empty	When the element has no children	H1: empty
: enabled	When the input is enabled	Input: enabled
: focus	When the input element is focussed	Input: focus
:last-child	When the element is the last child	P: last-child
:nth-child(n)	When the element is the nth-child	P: nth-child(4)
:only-child	When the element is the only child	P: only-child
: read-only	When the input element has a read-only attribute specified	Input: read-only

How to Combine Pseudo-Classes with CSS Classes

Now that we have discussed all the pseudo-classes and CSS selectors, let's combine them with CSS classes to filter down the selections and make sure that the selectors will work less in finding the elements.

We can select specific classes to tag them with the HTML tags and use the pseudo-class selectors with the different combinations.

In the below example, we have assigned a class to one of the div tags to select later, with the help of the selectors.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Web Template</title>
      <style>
      div {
        background-color: blue;
        width: 300px;
        height: 300px;
      }

      div.my_div:hover{
        background-color: red;
      }
      </style>
  </head>
  <body>
    <br><br>
      <center>
        <div></div>
        <br><br>
        <div class="my_div"></div>
      </center>
  </body>
</html>
```

CSS Pseudo Element Selectors

With the help of the pseudo-classes, you can simply select the complete element and apply the desired style to it. But the pseudo-element allows you to easily grab a specific part of the element and apply the style on it. One scenario where you can use pseudo-elements is inserting elements before a specific tag.

You must use the double colon "::" rather than using a single colon, as in the case of the pseudo-class. It was introduced in CSS 3 and helps distinguish the pseudo-elements from the pseudo-classes. Before CSS3, both were used using the single colon ":"

CSS First-line Pseudo-Element Selector

This is a straightforward selector you can use whenever you want to style the first line of the element. You must add the keyword "first-line" and "::" symbol of the pseudo-element.

```
<style>
    h2::first-line{
      color: green;
    }
      </style>
<h2>Your Text</h2>
```

The above code will change the color of the first line to green. You can use the first-line pseudo-element only on the block-level elements available on the web page.

If you're a developer, use the following HTML or CSS properties for using "first-line:"

- Font
- Color
- Background
- Word and letter spacing
- Text decoration
- Vertical-align
- Text-transform
- Line-height
- clear

CSS First-Letter Pseudo-Element Selector

Another pseudo-element selector is the "first-letter," which helps developers select the first letter of the element we tag. Then you can use the first-letter element to apply the styling to that element. This selector type is very common in article writing. If you want to apply a different styling type to the first letter, you can do it using the "first-letter" keyword and the "::" pseudo-element symbol.

```
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Web Template</title>
      <style>
    h2::first-letter{
      color: green;
      font-size: 50px;
    }
      </style>
  </head>
  <body>
    <br><br>
```

```
      <center>
        <div></div>
        <br><br>
        <h2>Text</h2>
      </center>
  </body>
</html>
```

As per the above code, the first element will be magnified and experience color changes. You can also use this first-letter element for the block-level elements.

If you're a web developer, you can use the below-mentioned properties with the "first-letter" pseudo-element:

- Font
- Color
- Background
- Margin
- Padding
- Border
- Text-decoration
- Vertical-align (only if "float" is "none")
- Text-transform
- Line-height
- Float
- Clear

CSS Before Pseudo Element Selector

If you want to add the content before the target element, you can use the "before" pseudo-element. For example, you can add inverted commas before starting any heading in HTML:

```
<style>
    p::before{
      content: "««";
      font-size: 25px;
    }
 </style>
```

You can use the above code to add the «« before the start of every paragraph.

CSS After Pseudo Element Selector

Opposite the “before” pseudo-element selector, the “after” pseudo-element entails placing the content *after* the target element.

For example:

```
<style>
    p::after{
      content: "««";
      font-size: 25px;
    }
 </style>
```

CSS Marker Pseudo Element Selector

If you want to select and apply a specific style to the marker element from the web page, you can use the marker pseudo-element selector. You can use it to add bullet points, number points, or any other “li” elements. The following example explains how the code will turn the color of markers into red.

```
<style>
      ul li::marker{
        color: red;
      }
      </style>
<ul>
        <li>The first element</li>
        <li>The second element</li>
        <li>The third element</li>
</ul>
```

As per the above code, the bullet points will be turned to red. As we are limited to use the marker only for the specified few elements, we are also limited to perform only specific actions with the marker elements.

Here are the properties you can use with the CSS pseudo-element:

- Font
- Color
- Direction, unicode-bidi and text-combine-upright properties.
- White space
- Content
- All animation and transition properties.

CSS Selection Pseudo-Element Selector

If you want to apply styling to the part of the target element that the user selects, you can use the CSS selection pseudo-element selector.

This type of selector is useful when you want the user to focus on a few words while reading.

Here's an example piece of code that will color the selected text to improve its readability.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Web Template</title>
      <style>
      p{
        font-size: 30px;
      }
      p::selection{
        color: red;
        font-size: 50px;
      }
      </style>
  </head>
  <body>
    <br><br>
        <br><br>
  <p>Your Text</p>
  </body>
</html>
```

The above code changes the selected text to red.

You can also use the following properties with the selection CSS pseudo-element.

- Text decoration
- Text shadow
- All color properties, including background color, stroke color, fill color, and stroke width.
- Cursor
- Outline

Combining Pseudo-Element with CSS Classes

You can combine the pseudo-elements with CSS classes that are similar to the pseudo-classes. With this combination, developers get more control and flexibility with the element. This type of combination brings out their creativity and provides amazing design to the website.

The following code will help you understand how to combine a pseudo-element with the CSS class:

```
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Web Template</title>
      <style>
      p{
        font-size: 30px;
      }
      p.my_class::first-line{
        color: red;
        font-size: 50px;
      }
      </style>
  </head>
  <body>
    <br><br>
        <br><br>
  <p>Your Text</p>
  <br><br>
  <p class="my_class">Your Text</p>
  </body>
</html>
```

In the above example, the “p” tag with the defined class is selected and colored red.

CSS Attribute Selector

The attribute selector is important for designing and creating a webpage.

With the help of the attributes, developers can provide specific instructions to the web elements and change the working as required. Also, attributes give the freedom to play around with the elements using JavaScript or CSS.

CSS developers have added the attributes to the selector library.

The following is the general syntax of CSS attribute selector:

```
element [attribute] {
//styling
}
```

CSS [attribute] Selector

The most commonly used attribute selector is the [attribute] selector. It is easy to use and implement. All you need is to provide the attribute name in the parameters. All the elements within the web page containing this attribute name will be selected for desired styling.

In the following example, the code will perform the styling on the “p” elements.

```
 p[lang]{
        color: red;
        font-size: 20px;
      }
<p lang = "en">Your Text</p>
<p>Your Text</p>
```

Once you run the above code in the browser, you will see the difference in styling in both “p” tags.

CSS [attribute=value] Selector

As a developer, you can simply filter down the selection of the elements. For that, you can provide a specific value to the attribute. Otherwise, everything will work the same as the CSS [attribute] selector.

Syntax:

```
element [attribute = "value"]
{
//styling
}
```

We can make the desired changes to the above code to target only those paragraphs that are written in ‘en’ language.

```
<html lang="en" dir="ltr">
  <head>
```

```
    <meta charset="utf-8">
    <title>Web Template</title>
      <style>
      p[lang = "hi"]{
        color: red;
        font-size: 20px;
      }
      </style>
  </head>
  <body>
    <br><br>
        <br><br>
    <p lang="en">Your Text</p>
    <p lang = "hi">Your Text</p>
  </body>
</html>
```

CSS [attribute ~=”value”] Selector

This selector is an advancement to the CSS [attribute=”value”] but with some added capabilities. This selector can easily select the element if it includes the word “value” in the attribute specified.

In the following example, the code will look for the value “shirt” in the attribute “title.”

```
<style>
      p[title ~= "shirt"]{
        color: red;
        font-size: 20px;
      }
      </style>
<p title="blue shirt">Your Text</p>
<p title="red shirt">Your Text</p>
<p title="hoodie">Your Text</p>
```

The above code will select the first two paragraphs, as they both have a shirt in their title (target attribute).

CSS [attribute |=”value”] Selector

This selector will help you select all the attribute values that start with the word “value.” You can understand it better with the help of the following example.

```
<style>
      p[title |= "shirt"]{
        color: red;
        font-size: 20px;
      }
      </style>
<p title="shirt-blue">Your Text</p>
<p title = "shirt-red">Your Text</p>
<p title = "hoodie">Your Text</p>
```

As per the above code, the first two paras will be colored red. But, this selector will only work only when the value is either a single word in the attribute. It will not select any white space in between to implement the styling. For example, the following code will only select the middle paragraph:

```
<p title="shirt blue">Your Text</p>
<p title = "shirt-red">Your Text</p>
<p title = "hoodie">Your Text</p>
```

CSS [attribute ^=”value”] Selector

CSS[attribute |= “value”] selector limitations are eliminated by the CSS[attribute ^= “value”] selector. With this selector, the attribute value doesn’t have to be a whole word (single or hyphenated).

You can use any value that starts with the word “value” and select it for styling. The following code will show how the CSS [attribute ^= “value”] selector works.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Web Template</title>
      <style>
      p[title ^= "shirt"]{
        color: red;
        font-size: 20px;
      }
      </style>
  </head>
  <body>
    <br><br>
```

```
        <br><br>
    <p title="shirt-blue">Your Text.</p>
    <p title = "shirt red">Your Text.</p>
    <p title = "hoodie">Your Text.</p>
  </body>
</html>
```

CSS [attribute $=”value”] Selector

The CSS[attribute $= “value”] selector works just opposite to the CSS[attribute ^= “value”] selector. The ^ attribute selector looks for the beginning with the word “value”, while on the other hand, the $ attribute selector looks for the ending with the word “value.”

Example:

```
<style>
      p[title $= "shirt"]{
        color: red;
        font-size: 20px;
      }
      </style>
<p title="blue shirt">Your Text</p>
<p title = "shirt red">Your Text</p>
<p title = "hoodie">Your Text</p>
```

The above code colors the first paragraph into red. Note that the CSS [attribute $= “value”] selector does not ask for a single or hyphenated word (one word). The attribute values with multiple attributes are selected until they end with the word “value.”

CSS [attribute*=”value”] Selector

This selector works like the regular expression checker that helps developers select every element with the target attribute composed of “value”. The value doesn’t have to be a whole word. The selector works even if the “value” is a part of the attribute value.

```
<style>
      p[title *= "rt"]{
        color: red;
        font-size: 20px;
      }
      </style>
<p title="blue shirt">Your Text.</p>
<p title = "shirt red">Your Text</p>
```

```
<p title = "hoodie">Your Text</p>
```

The above code will select the first two paragraphs and color them red.