**Graphical user interface, text, application, chat or text message

Description automatically generatedSelectors**

Recall: The beginning of the ruleset used to target (select) the element that will be styled.

1. **Type selector**

The first selector is the *type* selector. The type selector matches the *type* of the element in the HTML document.

p {  
}

**Note:**

* The type selector does not include the angle brackets.
* Since element types are **often referred to by their opening tag name**, the type selector is sometimes referred to as the *tag name* or *element* selector.
* The type selector targets all elements with the tag.

1. **Universal selector**

We know how *type selector* selects all elements of a given type. On the other hand, the *universal selector* selects all elements of any type. The universal selector uses the \* character.

\* {  
}

1. **Class selector**

Side information: in HTML, we can set multiple elements to a same class, using the class attribute. This is also used mainly to be pointed to in the stylesheet (A HTML element can also have multiple classes. If this is the case, that element will be affected by all class sectors)

To select HTML element by its class using CSS, a period (.) must be prepended to the class’s name.

<p class='brand'>Sole Shoe Company</p>

.brand {  
}

1. **ID selector**

If an HTML element needs to be styled uniquely, we can give it an ID using the id attribute (in HTML). Then, we can reference to this element using a number sign (#) and the id name. IDs override the styles of types and classes 🡪 should be limited and only on elements need to always appear the same

<h1 id='large-title'> ... </h1>

#large-title {  
}

1. **Attribute collector**

Remember that some HTML elements use attributes (href, src, class, id, etc.)

The *attribute selector* can be used to target all HTML elements that contain an attribute. To do that, **we surround attribute in square brackets**. (target all elements with href attribute by using [href])

We can also select more specific elements within the attribute by **adding attribute type and/or value** 🡪 selects an element where the attribute contains any instance of the specified value

type[attribute\*=value]

<img src='/images/seasons/cold/winter.jpg'>  
<img src='/images/seasons/warm/summer.jpg'>

img[src\*='winter'] {  
  height: 50px;  
}  
img[src\*='summer'] {  
  height: 100px;  
}

1. **Pseudo-class ♣♣**

Elements can change as we interact with them. For example, we click on an <input> element and a blue border showing it’s in focus; we visit to another page with a blue link text, and the text is purple after....

🡪 These are all examples of pseudo-class selectors in action, including :focus, :visisted, :disabled, :active

A pseudo-class can be attached to any selector (must select something first). It is always written as a **colon : followed by a name**. For example p:hover.

p:hover {  
  background-color: lime; 🡪 When mouse hovers over a p element, its background lime-colored  
}

1. **Chaining**

When writing CSS rules, it’s possible to require an HTML element to have two or more CSS selectors at the same time (and) 🡪 combining multiple selectors, by **writing them both down**.

Since class selectors have (.) and id selectors have (#) in front, we will not afraid of string concat.

h1.special { 🡪 have to be both h1 type, and class “special”  
}

1. **Descendant Combinator**

CSS also supports selecting elements that are nested within other HTML elements, also known as *descendants*. We do so by writing down the **selector for the bigger element, a space, and selector for descendants**.

<ul class='main-list'>  
  <li> ... </li>  
  <li> ... </li> 🡪 .main-list li { 🡪 selecting <li> elements  
  <li> ... </li> }  
</ul>

1. **Specificity**

Specificity is the order by which the browser decides which CSS styles will be displayed (which one overrides which). A best practice in CSS is to style elements while using the lowest degree of specificity so that if an element needs a new style, it is easy to override.

Adding more than one tag, class, or ID to a CSS selector increases the specificity of the CSS selector.

**More than 1 (chaining/descendant) > ID > clasees > type**

1. **Multiple Selectors**

To make CSS more concise, we can add CSS styles to multiple selectors at once (if we want to format various selectors in the same way) 🡪 prevent repetitive code

We can **separate the selectors by a comma** to apply the same style to both

h1 {  
  font-family: Georgia;  
} h1,  
 🡪 .menu {  
.menu { font-family: Georgia;  
  font-family: Georgia; }  
}