Chapter 10: Classes and IDs

Parameter	Details
class	Indicates the Class of the element (non-unique)
id	Indicates the ID of the element (unique in the same context)

Classes and IDs make referencing HTML elements from scripts and stylesheets easier. The class attribute can be used on one or more tags and is used by CSS for styling. IDs however are intended to refer to a single element, meaning the same ID should never be used twice. IDs are generally used with JavaScript and internal document links, and are discouraged in CSS. This topic contains helpful explanations and examples regarding proper usage of class and ID attributes in HTML.

Section 10.1: Giving an element a class

Classes are identifiers for the elements that they are assigned to. Use the `class` attribute to assign a class to an element.

```
<div class="example-class"></div>
```

To assign multiple classes to an element, separate the class names with spaces.

```
<div class="class1 class2"></div>
```

Using classes in CSS

Classes can be used for styling certain elements without changing all elements of that kind. For example, these two `span` elements can have completely different stylings:

```
<span></span>
<span class="special"></span>
```

Classes of the same name can be given to any number of elements on a page and they will all receive the styling associated with that class. This will always be true unless you specify the element within the CSS.

For example, we have two elements, both with the class `highlight`:

```
<div class="highlight">Lorem ipsum</div>
<span class="highlight">Lorem ipsum</span>
```

If our CSS is as below, then the color green will be applied to the text within both elements:

```
.highlight { color: green; }
```

However, if we only want to target `div`'s with the class `highlight` then we can add specificity like below:

```
div.highlight { color: green; }
```

Nevertheless, when styling with CSS, it is generally recommended that only classes (e.g. `.highlight`) be used rather than elements with classes (e.g. `div.highlight`).

As with any other selector, classes can can be nested:

```
.main .highlight { color: red; } /* Descendant combinator */
```

```
.footer > .highlight { color: blue; } /* Child combinator */
```

You can also chain the class selector to only select elements that have a combination of several classes. For example, if this is our HTML:

```
<div class="special left menu">This text will be pink</div>
```

And we want to colour this specific piece of text pink, we can do the following in our CSS:

```
.special.left.menu { color: pink; }
```

Section 10.2: Giving an element an ID

The ID attribute of an element is an identifier which must be unique in the whole document. Its purpose is to uniquely identify the element when linking (using an anchor), scripting, or styling (with CSS).

```
<div id="example-id"></div>
```

You should not have two elements with the same ID in the same document, even if the attributes are attached to two different kinds of elements. For example, the following code is incorrect:

```
<div id="example-id"></div>
<span id="example-id"></span>
```

Browsers will do their best to render this code, but unexpected behavior may occur when styling with CSS or adding functionality with JavaScript.

To reference elements by their ID in CSS, prefix the ID with #.

```
#example-id { color: green; }
```

To jump to an element with an ID on a given page, append # with the element name in the URL.

```
http://example.com/about#example-id
```

This feature is supported in most browsers and does not require additional JavaScript or CSS to work.

Section 10.3: Acceptable Values

For an ID

Version ≥ 5

The only restrictions on the value of an `id` are:

1. it must be unique in the document
2. it must not contain any space characters
3. it must contain at least one character

So the value can be all digits, just one digit, just punctuation characters, include special characters, whatever. Just no whitespace.

So these are valid:

```
<div id="container"> ... </div>
```

```
<div id="999"> ... </div>
<div id="#%LV-||"> ... </div>
<div id="____V"> ... </div>
<div id="⌘⌥"> ... </div>
<div id="♥"> ... </div>
<div id="{}"> ... </div>
<div id="©"> ... </div>
<div id="₲₩¤☆€~¥"> ... </div>
```

This is invalid:

```
<div id=" "> ... </div>
```

This is also invalid, when included in the same document:

```
<div id="results"> ... </div>
<div id="results"> ... </div>
```

Version ≤ 4.01

An `id` value must begin with a letter, which can then be followed only by:

- letters (A-Z/a-z)
- digits (0-9)
- hyphens ("-")
- underscores ("_")
- colons (":")
- periods (".")

Referring to the first group of examples in the HTML5 section above, only one is valid:

```
<div id="container"> ... </div>
```

These are also valid:

```
<div id="sampletext"> ... </div>
<div id="sample-text"> ... </div>
<div id="sample_text"> ... </div>
<div id="sample:text"> ... </div>
<div id="sample.text"> ... </div>
```

Again, if it doesn't start with a letter (uppercase or lowercase), it's not valid.

For a Class

The rules for classes are essentially the same as for an `id`. The difference is that `class` values *do not* need to be unique in the document.

Referring to the examples above, although this is not valid in the same document:

```
<div id="results"> ... </div>
<div id="results"> ... </div>
```

This is perfectly okay:

```
<div class="results"> ... </div>
```

```
<div class="results"> ... </div>
```

Important Note: How ID and Class values are treated outside of HTML

Keep in mind that the rules and examples above apply within the context of HTML.

Using numbers, punctuation or special characters in the value of an `id` or a `class` may cause trouble in other contexts, such as CSS, JavaScript and regular expressions.

For example, although the following `id` is valid in HTML5:

```
<div id="9lions"> ... </div>
```

... it is invalid in CSS:

> **4.1.3 Characters and case**
>
> In CSS, *identifiers* (including element names, classes, and IDs in selectors) can contain only the characters [a-zA-Z0-9] and ISO 10646 characters U+00A0 and higher, plus the hyphen (-) and the underscore (_); ***they cannot start with a digit, two hyphens, or a hyphen followed by a digit***. (emphasis added)

In most cases you may be able to escape characters in contexts where they have restrictions or special meaning.

W3C References

- 3.2.5.1 The `id` attribute
- 3.2.5.7 The `class` attribute
- 6.2 SGML basic types

Section 10.4: Problems related to duplicated IDs

Having more than one element with the same ID is a hard to troubleshoot problem. The HTML parser will usually try to render the page in any case. Usually no error occurs. But the pace could end up in a mis-behaving web page.

In this example:

```
<div id="aDiv">a</div>
<div id="aDiv">b</div>
```

CSS selectors still work

```
#aDiv {
    color: red;
}
```

But JavaScript fails to handle both elements:

```
var html = document.getElementById("aDiv").innerHTML;
```

In this case`html` variable bears only the first `div` content `("a")`.

Chapter 11: Data Attributes

Value	Description
somevalue	Specifies the value of the attribute (as a string)

Section 11.1: Older browsers support

Data attributes were introduced in HTML5 which is supported by all modern browsers, but older browsers before HTML5 don't recognize the data attributes.

However, in HTML specifications, attributes that are not recognized by the browser must be left alone and the browser will simply ignore them when rendering the page.

Web developers have utilized this fact to create non-standard attributes which are any attributes not part of the HTML specifications. For example, the `value` attribute in the line bellow is considered a non-standard attribute because the specifications for the `<img>` tag don't have a `value` attribute and it is not a global attribute:

```
<img src="sample.jpg" value="test" />
```

This means that although data attributes are not supported in older browsers, they still work and you can set and retrieve them using the same generic JavaScript `setAttribute` and `getAttribute` methods, but you cannot use the new `dataset` property which is only supported in modern browsers.

Section 11.2: Data Attribute Use

HTML5 `data-*` attributes provide a convenient way to store data in HTML elements. The stored data can be read or modified using JavaScript

```
<div data-submitted="yes" class="user_profile">
  ... some content ...
</div>
```

- Data attribute structure is `data-*`, i.e. the name of the data attribute comes after the `data-` part. Using this name, the attribute can be accessed.
- Data in string format (including `json`) can be stored using `data-*` attribute.