The `Document` method **`querySelector()`** returns the first `Element` within the document that matches the specified selector, or group of selectors. If no matches are found, `null` is returned.

> **Note**: The matching is done using depth-first pre-order traversal of the document's nodes starting with the first element in the document's markup and iterating through sequential nodes by order of the number of child nodes.

Syntax

```
element = document.querySelector(selectors);
```

Parameters

selectors
A `DOMString` containing one or more selectors to match. This string must be a valid CSS selector string; if it isn't, a `SYNTAX_ERR` exception is thrown. See Locating DOM elements using selectors for more about selectors and how to manage them.

> **Note:** Characters that are not part of standard CSS syntax must be escaped using a backslash character. Since JavaScript also uses backslash escaping, be especially careful when writing string literals using these characters. See Escaping special characters for more information.

Return value

An `HTMLElement` object representing the first element in the document that matches the specified set of CSS selectors, or `null` is returned if there are no matches.

If you need a list of all elements matching the specified selectors, you should use `querySelectorAll()` instead.

Exceptions

`SYNTAX_ERR`
The syntax of the specified *selectors* is invalid.

Usage notes

If the specified selector matches an ID that is incorrectly used more than once in the document, the first element with that ID is returned.

CSS pseudo-elements will never return any elements, as specified in the Selectors API.

Escaping special characters

To match against an ID or selectors that do not follow standard CSS syntax (by using a colon or space inappropriately, for example), you must escape the character with a backslash (" \ "). As the backslash is also an escape character in JavaScript, if you are entering a literal string, you must escape it *twice* (once for the JavaScript string, and another time for `querySelector()`):

```
<div id="foo\bar"></div>
<div id="foo:bar"></div>

<script>
  console.log('#foo\bar');               // "#fooar" (\b is the backspa
  document.querySelector('#foo\bar');    // Does not match anything

  console.log('#foo\\bar');              // "#foo\bar"
  console.log('#foo\\\\bar');            // "#foo\\bar"
  document.querySelector('#foo\\\\bar'); // Match the first div

  document.querySelector('#foo:bar');    // Does not match anything
  document.querySelector('#foo\\:bar');  // Match the second div
</script>
```

Examples

Finding the first element matching a class

In this example, the first element in the document with the class "`myclass`" is returned:

```
var el = document.querySelector(".myclass");
```

A more complex selector

Selectors can also be really powerful, as demonstrated in the following example. Here, the first `<input>` element with the name "login" (`<input name="login"/>`) located inside a `<div>` whose class is "user-panel main" (`<div class="user-panel main">`) in the document is returned:

```
var el = document.querySelector("div.user-panel.main input[name='login'
```

Negation

As all CSS selector strings are valid, you can also negate selectors:

```
var el = document.querySelector("div.user-panel:not(.main) input[name='
```

This will select an input with a parent div with the `user-panel` class but not the `main` class.