The `Document` method **`querySelectorAll()`** returns a static (not live) `NodeList` representing a list of the document's elements that match the specified group of selectors.

> **Note:** This method is implemented based on the `ParentNode` mixin's `querySelectorAll()` method.

Syntax

```
elementList = parentNode.querySelectorAll(selectors);
```

Parameters

`selectors`
A `DOMString` containing one or more selectors to match against. This string must be a valid CSS selector string; if it's not, a `SyntaxError` exception is thrown. See Locating DOM elements using selectors for more information about using selectors to identify elements. Multiple selectors may be specified by separating them using commas.

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

Return value

A non-live `NodeList` containing one `Element` object for each element that matches at least one of the specified selectors or an empty `NodeList` in case of no matches.

> **Note:** If the specified `selectors` include a CSS pseudo-element, the returned list is always empty.

Exceptions

`SyntaxError`

The syntax of the specified `selectors` string is not valid.

Examples

Obtaining a list of matches

To obtain a `NodeList` of all of the `<p>` elements in the document:

```
var matches = document.querySelectorAll("p");
```

This example returns a list of all `<div>` elements within the document with a class of either `note` or `alert`:

```
var matches = document.querySelectorAll("div.note, div.alert");
```

Here, we get a list of `<p>` elements whose immediate parent element is a `<div>` with the class `highlighted` and which are located inside a container whose ID is `test`.

```
var container = document.querySelector("#test");
var matches = container.querySelectorAll("div.highlighted > p");
```

This example uses an attribute selector to return a list of the `<iframe>` elements in the document that contain an attribute named `data-src`:

```
var matches = document.querySelectorAll("iframe[data-src]");
```

Here, an attribute selector is used to return a list of the list items contained within a list whose ID is `userlist` which have a `data-active` attribute whose value is `1`:

```
var container = document.querySelector("#userlist");
var matches = container.querySelectorAll("li[data-active='1']");
```

Accessing the matches

Once the `NodeList` of matching elements is returned, you can examine it just like any array. If the array is empty (that is, its `length` property is 0), then no matches were found.

Otherwise, you can simply use standard array notation to access the contents of the list. You can use any common looping statement, such as:

```
var highlightedItems = userList.querySelectorAll(".highlighted");

highlightedItems.forEach(function(userItem) {
  deleteUser(userItem);
});
```

User notes

`querySelectorAll()` behaves differently than most common JavaScript DOM libraries, which might lead to unexpected results.

HTML

Consider this HTML, with its three nested `<div>` blocks.

```
<div class="outer">
  <div class="select">
    <div class="inner">
    </div>
  </div>
</div>
```

JavaScript

```
var select = document.querySelector('.select');
var inner = select.querySelectorAll('.outer .inner');
inner.length; // 1, not 0!
```

In this example, when selecting `.outer .inner` in the context the `<div>` with the class `select`, the element with the class `.inner` is still found, even though `.outer` is not a descendant of the base element on which the search is performed (`.select`). By default, `querySelectorAll()` only verifies that the last element in the selector is within the search scope.

The `:scope` pseudo-class restores the expected behavior, only matching selectors on descendants of the base element:

```
var select = document.querySelector('.select');
var inner = select.querySelectorAll(':scope .outer .inner');
inner.length; // 0
```