JS JavaScript TUTORIAL

JavaScript querySelector() Method

Summary: in this tutorial, you will learn how to use the JavaScript `querySelector()` and `querySelectorAll()` to find elements based on CSS selectors.

Introduction to JavaScript querySelector() and querySelectorAll() methods

The `querySelector()` is a method of the `Element` interface. The `querySelector()` method allows you to select the first element that matches one or more CSS selectors.

The following illustrates the syntax of the `querySelector()` method:

```
let element = parentNode.querySelector(selector);
```

In this syntax, the `selector` is a CSS selector or a group of CSS selectors to match the descendant elements of the `parentNode` .

If the `selector` is not valid CSS syntax, the method will raise a `SyntaxError` exception.

If no element matches the CSS selectors, the `querySelector()` returns `null` .

> The `querySelector()` method is available on the `document` object or any `Element` object.

Besides the `querySelector()` , you can use the `querySelectorAll()` method to select all elements that match a CSS selector or a group of CSS selectors:

```
let elementList = parentNode.querySelectorAll(selector);
```

The `querySelectorAll()` method returns a static `NodeList` of elements that match the CSS selector. If no element matches, it returns an empty `NodeList` .

> Note that the `NodeList` is an array-like object, not an array object. However, in modern web browsers, you can use the `forEach()` method or the `for...of` loop.

To convert the `NodeList` to an array, you use the `Array.from()` method like this:

```
let nodeList = document.querySelectorAll(selector);
let elements = Array.from(nodeList);
```

Basic selectors

Suppose that you have the following HTML document:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>querySelector() Demo</title>
</head>
<body>
    <header>
        <div id="logo">
            <img src="img/logo.jpg" alt="Logo" id="logo">
        </div>
        <nav class="primary-nav">
            <ul>
                <li class="menu-item current"><a href="#home">Home</a></li>
                <li class="menu-item"><a href="#services">Services</a></li>
                <li class="menu-item"><a href="#about">About</a></li>
                <li class="menu-item"><a href="#contact">Contact</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <h1>Welcome to the JS Dev Agency</h1>
        <div class="container">
            <section class="section-a">
                <h2>UI/UX</h2>
                <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Autem placeat, a
                    laudantium facilis iure adipisci ab veritatis eos neque culpa id nostrum
                    Adipisci, obcaecati repellat.</p>
                <button>Read More</button>
            </section>
            <section class="section-b">
                <h2>PWA Development</h2>
                <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Magni fugiat sir
                    commodi aspernatur, tempora doloribus quod, consectetur deserunt, facilis
                    provident labore nihil in earum.</p>
                <button>Read More</button>
            </section>
            <section class="section-c">
                <h2>Mobile App Dev</h2>
                <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Animi eos culpa l
                    Quibusdam, iure obcaecati. Adipisci deserunt, alias repellat eligendi odi
                    laborum debitis eos?</p>
                <button>Read More</button>
            </section>
        </div>
    </main>
    <script src="js/main.js"></script>
</body>
</html>
```

1) Universal Selector

The universal selector is denoted by `*` that matches all elements of any type:

```
*
```

The following example uses the `querySelector()` selects the first element in the document:

```
let element = document.querySelector('*');
```

And this selects all elements in the document:

```
let elements = document.querySelectorAll('*');
```

2) Type selector

To select elements by node name, you use the type selector e.g., `a` selects all `<a>` elements:

```
elementName
```

The following example finds the first `h1` element in the document:

```
let firstHeading = document.querySelector('h1');
```

The following example finds all `h2` elements:

```
let heading2 = document.querySelectorAll('h2');
```

3) Class selector

To find the element with a given CSS class, you use the class selector syntax:

```
.className
```

The following example finds the first element with the `menu-item` class:

```
let note = document.querySelector('.menu-item');
```

The following example finds all elements with the `menu` class:

```
let notes = document.querySelectorAll('.menu-item');
```

4) ID Selector

To select an element based on the value of its id, you use the id selector syntax:

```
#id
```

The following example finds the first element with the id `#logo` :

```
let logo = document.querySelector('#logo');
```

Since the `id` should be unique in the document, the `querySelectorAll()` is not relevant.

5) Attribute selector

To select all elements that have a given attribute, you use one of the following attribute selector syntaxes:

```
[attribute]
[attribute=value]
[attribute~=value]
[attribute|=value]
[attribute^=value]
[attribute$=value]
[attribute*$*=value]
```

The following example finds the first element with the attribute `[autoplay]` with any value:

```
let autoplay = document.querySelector('[autoplay]');
```

The following example finds all elements that have `[autoplay]` attribute with any value:

```
let autoplays = document.querySelectorAll('[autoplay]');
```

Grouping selectors

To group multiple selectors, you use the following syntax:

```
selector, selector, ...
```

The selector list will match any element with one of the selectors in the group.

The following example finds all `<div>` and `<p>` elements:

```
let elements = document.querySelectorAll('div, p');
```

Combinators

1) descendant combinator

To find descendants of a node, you use the space ( ) descendant combinator syntax:

```
selector selector
```

For example `p a` will match all `<a>` elements inside the `p` element:

```
let links = document.querySelector('p a');
```

2) Child combinator

The `>` child combinator finds all elements that are direct children of the first element:

```
selector > selector
```

The following example finds all `li` elements that are directly inside a `<ul>` element:

```
let listItems = document.querySelectorAll('ul > li');
```

To select all `li` elements that are directly inside a `<ul>` element with the class `nav` :

```
let listItems = document.querySelectorAll('ul.nav > li');
```

3) General sibling combinator

The `~` combinator selects siblings that share the same parent:

```
selector ~ selector
```

For example, `p ~ a` will match all `<a>` elements that follow the `p` element, immediately or not:

```
let links = document.querySelectorAll('p ~ a');
```

4) Adjacent sibling combinator

The `+` adjacent sibling combinator selects adjacent siblings:

```
selector + selector
```

For example, `h1 + a` matches all elements that directly follow an `h1` :

```
let links = document.querySelectorAll('h1 + a');
```

Select the first `<a>` that directly follows an `h1` :

```
let links = document.querySelector('h1 + a');
```

Pseudo

1) Pseudo-classes

The `:` pseudo matches elements based on their states:

```
element:state
```

For example, the `li:nth-child(2)` selects the second `<li>` element in a list:

```
let listItem = document.querySelectorAll('li:nth-child(2)');
```

2) Pseudo-elements

The `::` represent entities that are not included in the document therefore the `querySelector()` method cannot select pseudo-elements.

Summary

- The `querySelector()` finds the first element that matches a CSS selector or a group of CSS selectors.
- The `querySelectorAll()` finds all elements that match a CSS selector or a group of CSS selectors.
- A CSS selector defines elements to which a CSS rule applies.

Quiz