# Introduction to JavaScript getElementById() method

// available only on the document object

The document.getElementById() method returns an Element object that represents an HTML element with an id that matches a specified string.

If the document has no element with the specified id, the document.getElementById() returns null.

Because the id of an element is unique within an HTML document, the document.getElementById() is a quick way to access an element.

Unlike the querySelector() method, the getElementById() is only available on the document object, not other elements.

The following shows the syntax of the getElementById() method:

const element = document.getElementById(id);

In this syntax, the id is a string that represents the id of the element to select. The id is case-sensitive. For example, the 'root' and 'Root' are totally different.

The id is unique within an HTML document. However, HTML is a forgiving language. If the HTML document has multiple elements with the same id, the document.getElementById() method returns the first element it encounters.

## JavaScript getElementById() method example:

Suppose you have the following HTML document:

**<html>**

**<head>**

**<title>**JavaScript getElementById() Method**</title>**

**</head>**

**<body>**

**<p id="message">**A paragraph**</p>**

**</body>**

**</html>**Code language: HTML, XML (xml)

The document contains a <p> element that has the id attribute with the value message:

const p = document.getElementById('message');

console.log(p);Code language: JavaScript (javascript)

Output:

**<p id="message">**A paragraph**</p>**Code language: HTML, XML (xml)

After selecting an element, you can [add styles to the element](https://www.javascripttutorial.net/dom/css/add-styles-to-an-element/), manipulate its [attributes](https://www.javascripttutorial.net/dom/attributes/set-the-value-of-an-attribute/), and traverse to [parent](https://www.javascripttutorial.net/javascript-dom/javascript-get-parent-element-parentnode/) and [child elements](https://www.javascripttutorial.net/javascript-dom/javascript-get-child-element/).

## Summary

* The document.getElementById() returns a DOM element specified by an id or null if no matching element found.
* If multiple elements have the same id, even though it is invalid, the getElementById() returns the first element it encounters.

# JavaScript getElementsByName

// available only on the document object

**Summary**: in this tutorial, you will learn how to use the JavaScript getElementsByName() method to get elements with a given name in a document.

## Introduction to JavaScript getElementsByName() method

Every element on an HTML document may have a name attribute:

**<input type="radio" name="language" value="JavaScript">**

Unlike the id attribute, multiple HTML elements can share the same value of the name attribute like this:

**<input type="radio" name="language" value="JavaScript">**

**<input type="radio" name="language" value="TypeScript">**

To get all elements with a specified name, you use the getElementsByName() method of the document object:

let elements = document.getElementsByName(name);

The getElementsByName() accepts a name which is the value of the name attribute of elements and returns a live NodeList of elements in an array.

The return collection of elements is live. It means that the return elements are automatically updated when elements with the same name are [inserted](https://www.javascripttutorial.net/dom/manipulating/insert-an-element-before-an-existing-element/) and/or [removed](https://www.javascripttutorial.net/javascript-dom/javascript-removechild/) from the document.

## JavaScript getElementsByName() example:

The following example shows a radio group that consists of [radio buttons](https://www.javascripttutorial.net/javascript-dom/javascript-radio-button/) that have the same name (rate).

When you select a radio button and click the submit button, the page will show the selected value such as Very Poor, Poor, OK, Good, or Very Good:

<!DOCTYPE html>

**<html>**

**<head>**

**<meta charset="utf-8">**

**<title>**JavaScript getElementsByName Demo**</title>**

**</head>**

**<body>**

**<p>**Please rate the service:**</p>**

**<p>**

**<label for="very-poor">**

**<input type="radio" name="rate" value="Very poor" id="very-poor">** Very poor

**</label>**

**<label for="poor">**

**<input type="radio" name="rate" value="Poor" id="poor">** Poor

**</label>**

**<label for="ok">**

**<input type="radio" name="rate" value="OK" id="ok">** OK

**</label>**

**<label for="good">**

**<input type="radio" name="rate" value="Good">** Good

**</label>**

**<label for="very-good">**

**<input type="radio" name="rate" value="Very Good" id="very-good">** Very Good

**</label>**

**</p>**

**<p>**

**<button id="btnRate">**Submit**</button>**

**</p>**

**<p id="output"></p>**

**<script>**

let btn = document.getElementById('btnRate');

let output = document.getElementById('output');

btn.addEventListener('click', () => {

let rates = document.getElementsByName('rate');

rates.forEach((rate) => {

if (rate.checked) {

output.innerText = `You selected: ${rate.value}`;

}

});

});

**</script>**

**</body>**

**</html**

How it works:

* First, select the submit button by its id btnRate using the [getElementById()](https://www.javascripttutorial.net/javascript-dom/javascript-getelementbyid/) method.
* Second, listen to the [click](https://www.javascripttutorial.net/javascript-dom/javascript-mouse-events/) event of the submit button.
* Third, get all the radio buttons using the getElementsByName() and show the selected value in the output element.

Notice that you will learn about [events](https://www.javascripttutorial.net/javascript-dom/javascript-events/) like click later. For now, you just need to focus on the getElementsByName() method.

## Summary

* The getElementsByName() returns a live NodeList of elements with a specified name.
* The NodeList is an array-like object, not an [array](https://www.javascripttutorial.net/javascript-array/) object.

# JavaScript getElementsByTagName

// available on the document object and specific element which means you can access an element then you can access the underlying elements by their tag name like you will see in the getElementsByClassName example after this

**Summary**: in this tutorial, you will learn how to use the JavaScript getElementsByTagName() to select elements with a given tag name.

## Introduction to JavaScript getElementsByTagName() method

The getElementsByTagName() is a method of the document object or a specific DOM element.

The getElementsByTagName() method accepts a tag name and returns a live HTMLCollection of elements in an array with the matching tag name in the order which they appear in the document.

The following illustrates the syntax of the getElementsByTagName():

let elements = document.getElementsByTagName(tagName);

The return collection of the getElementsByTagName() is live, meaning that it is automatically updated when elements with the matching tag name are added and/or removed from the document.

Note that the HTMLCollection is an array-like object, like arguments object of a function.

## JavaScript getElementsByTagName() example:

The following example illustrates how to use the getElementsByTagName() method to get the number of H2 tags in the document.

When you click the **Count H2** button, the page shows the number of H2 tags:

<!DOCTYPE html>

**<html>**

**<head>**

**<title>**JavaScript getElementsByTagName() Demo**</title>**

**</head>**

**<body>**

**<h1>**JavaScript getElementsByTagName() Demo**</h1>**

**<h2>**First heading**</h2>**

**<p>**This is the first paragraph.**</p>**

**<h2>**Second heading**</h2>**

**<p>**This is the second paragraph.**</p>**

**<h2>**Third heading**</h2>**

**<p>**This is the third paragraph.**</p>**

**<button id="btnCount">**Count H2**</button>**

**<script>**

let btn = document.getElementById('btnCount');

btn.addEventListener('click', () => {

let headings = document.getElementsByTagName('h2');

alert(`The number of H2 tags: ${headings.length}`);

});

**</script>**

**</body>**

**</html>**Code language: HTML, XML (xml)

How it works:

* First, select the button **Count H2** by using the [getElementById()](https://www.javascripttutorial.net/javascript-dom/javascript-getelementbyid/) method.
* Second, hook the click event of the button to an anonymous function.
* Third, in the anonymous function, use the document.getElementsByTagName() to get a list of H2 tags.
* Finally, show the number of H2 tags using the alert() function.

## Summary

* The getElementsByTagName() is a method of the document or element object.
* The getElementsByTagName() accepts a tag name and returns a list of elements with the matching tag name.
* The getElementsByTagName() returns a live HTMLCollection of elements. The HTMLCollection is an array-like object.

# JavaScript getElementsByClassName

// available only on the document object and any other element

**Summary**: in this tutorial, you will learn how to use the getElementsByClassName() method to select elements by class name.

## Introduction to the getElementsByClassName() method

The getElementsByClassName() method returns an array-like of objects of the child elements with a specified class name. The getElementsByClassName() method is available on the document element or any other elements.

When calling the method on the document element, it searches the entire document and returns the child elements of the document:

let elements = document.getElementsByClassName(names);

However, when calling the method on a specific element, it returns the descendants of that specific element with the given class name:

let elements = rootElement.getElementsByClassName(names);

The method returns the elements which is a live HTMLCollection of the matches elements.

The names parameter is a string that represents one or more class names to match; To use multiple class names, you separate them by space.

## JavaScript getElementsByClassName() method examples

Let’s take some examples of using the getElementsByClassName() method.

Suppose that you have the following HTML document:

<!DOCTYPE html>

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>**JavaScript getElementsByClassName**</title>**

**</head>**

**<body>**

**<header>**

**<nav>**

**<ul id="menu">**

**<li class="item">**HTML**</li>**

**<li class="item">**CSS**</li>**

**<li class="item highlight">**JavaScript**</li>**

**<li class="item">**TypeScript**</li>**

**</ul>**

**</nav>**

**<h1>**getElementsByClassName Demo**</h1>**

**</header>**

**<section>**

**<article>**

**<h2 class="secondary">**Example 1**</h2>**

**</article>**

**<article>**

**<h2 class="secondary">**Example 2**</h2>**

**</article>**

**</section>**

**</body>**

**</html>**Code language: HTML, XML (xml)

### **1) Calling JavaScript getElementsByClassName() on an element example**

The following example illustrates how to use the getElementsByClassName() method to select the <li> items which are the descendants of the <ul> element:

let menu = document.getElementById('menu');

let items = menu.getElementsByClassName('item');

let data = [].map.call(items, item => item.textContent);

console.log(data);Code language: JavaScript (javascript)

Output:

['HTML', 'CSS', 'JavaScript', 'TypeScript']Code language: JavaScript (javascript)

How it works:

* First, select the <ul> element with the class name menu using the [getElementById()](https://www.javascripttutorial.net/javascript-dom/javascript-getelementbyid/) method.
* Then, select <li> elements, which are the descendants of the <ul> element, using the getElementsByClassName() method.
* Finally, create an array of the text content of <li> elements by borrowing the map() method of the Array object.

### **2) Calling JavaScript getElementsByClassName() on the document example**

To search for the element with the class heading-secondary, you use the following code:

let elements = document.getElementsByClassName('secondary');

let data = [].map.call(elements, elem => elem.textContent);

console.log(data);Code language: JavaScript (javascript)

Output:

["Example 1", "Example 2"]Code language: JavaScript (javascript)

This example calls the getElementsByClassName() method on the document object. Therefore, it searches for the elements with the class secondary in the entire document.

## Summary

* Use the JavaScript getElementsByClassName() method to select the child elements of an element that has one or more give class names.

# JavaScript querySelector

**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()](https://www.javascripttutorial.net/javascript-array-foreach/) method or the [for...of](https://www.javascripttutorial.net/es6/javascript-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, atque accusamus voluptas

laudantium facilis iure adipisci ab veritatis eos neque culpa id nostrum tempora tempore minima.

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 similique illo nobis quibusdam

commodi aspernatur, tempora doloribus quod, consectetur deserunt, facilis natus optio. Iure

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 laudantium consequatur ea!

Quibusdam, iure obcaecati. Adipisci deserunt, alias repellat eligendi odit labore! Fugit iste sit

laborum debitis eos?**</p>**

**<button>**Read More**</button>**

**</section>**

**</div>**

**</main>**

**<script src="js/main.js"></script>**

**</body>**

**</html>**Code language: HTML, XML (xml)

### **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('\*');Code language: JavaScript (javascript)

And this select all elements in the document:

let elements = document.querySelectorAll('\*');Code language: JavaScript (javascript)

### **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');Code language: JavaScript (javascript)

And the following example finds all h2 elements:

let heading2 = document.querySelectorAll('h2');Code language: JavaScript (javascript)

### **3) Class selector**

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

.classNameCode language: CSS (css)

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

let note = document.querySelector('.menu-item');Code language: JavaScript (javascript)

And the following example finds all elements with the menu class:

let notes = document.querySelectorAll('.menu-item');Code language: JavaScript (javascript)

### **4) ID Selector**

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

#idCode language: CSS (css)

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

let logo = document.querySelector('#logo');Code language: JavaScript (javascript)

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]

Code language: JSON / JSON with Comments (json)

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

let autoplay = document.querySelector('[autoplay]');Code language: JavaScript (javascript)

And the following example finds all elements that have [autoplay] attribute with any value:

let autoplays = document.querySelectorAll('[autoplay]');Code language: JavaScript (javascript)

## 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');Code language: JavaScript (javascript)

## 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');Code language: JavaScript (javascript)

### **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');Code language: JavaScript (javascript)

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

let listItems = document.querySelectorAll('ul.nav > li');Code language: JavaScript (javascript)

### **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');Code language: JavaScript (javascript)

### **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');Code language: JavaScript (javascript)

And select the first <a> that directly follows an h1:

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

Code language: JavaScript (javascript)

## Pseudo

### **1) Pseudo-classes**

The : pseudo matches elements based on their states:

element:stateCode language: CSS (css)

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

let listItem = document.querySelectorAll('li:nth-child(2)');Code language: JavaScript (javascript)

### **2) Pseudo-elements**

The :: represent entities that are not included in the document.

For example, p::first-line matches the first line of all p elements:

let links = document.querySelector('p::first-line'); Code language: JavaScript (javascript)

## 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.