DOM Access Interface Methods

[**https://www.javascripttutorial.net/javascript-dom/javascript-queryselector/**](https://www.javascripttutorial.net/javascript-dom/javascript-queryselector/)

Often, with JavaScript, you want to manipulate HTML elements.

To do so, you have to find the elements first. There are several ways to do this:

1. Finding HTML elements by id

const element = document.getElementById("intro");

1. Finding HTML Elements By Name

const element = document.getElementsByName("p");

1. Finding HTML elements by tag name

const element = document.getElementsByTagName("p");

1. Finding HTML elements by class name

const x = document.getElementsByClassName("intro");

1. Finding HTML elements by CSS selectors

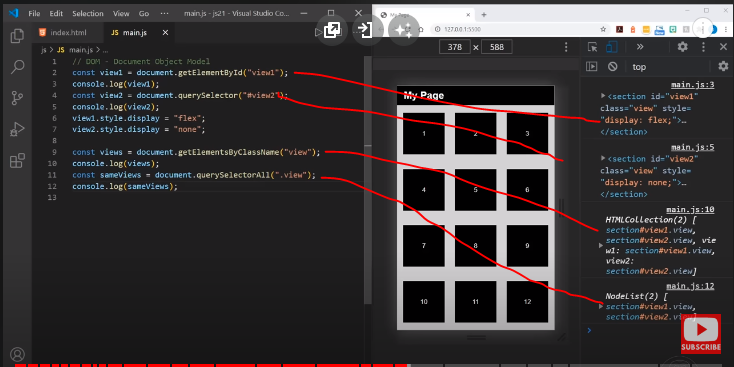
const x = document.querySelectorAll("p.intro");

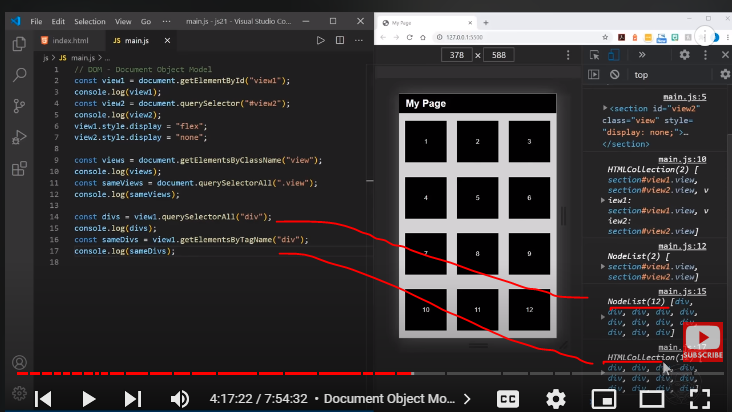
const x = document.querySelector("p.intro");

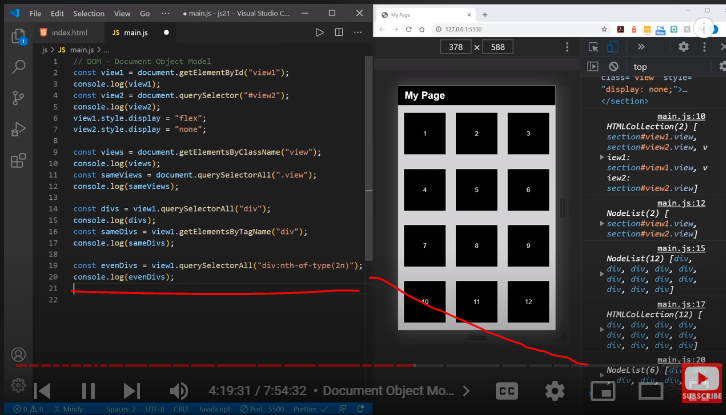
1. Finding HTML elements by HTML object collections

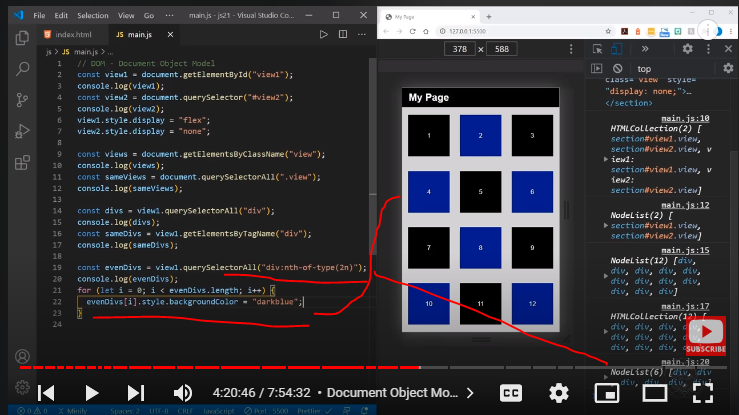
document.getElementById("demo").innerHTML = text;

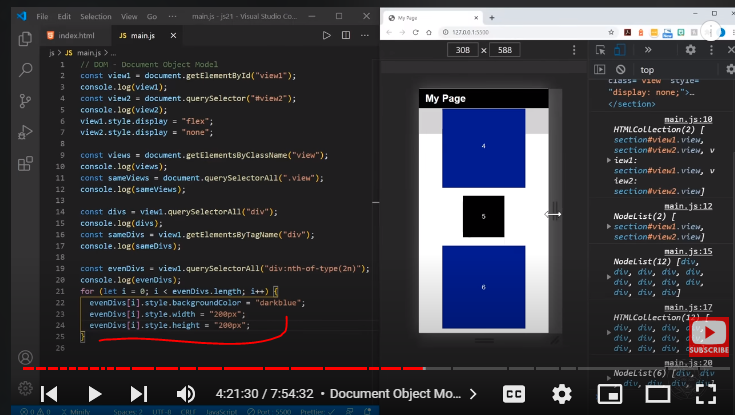
examples

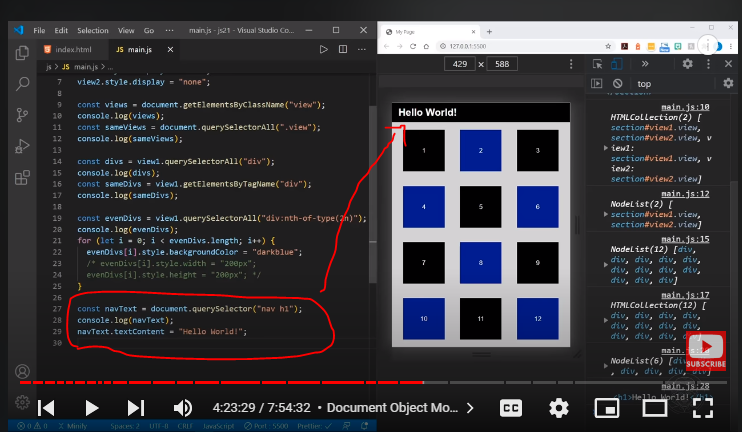


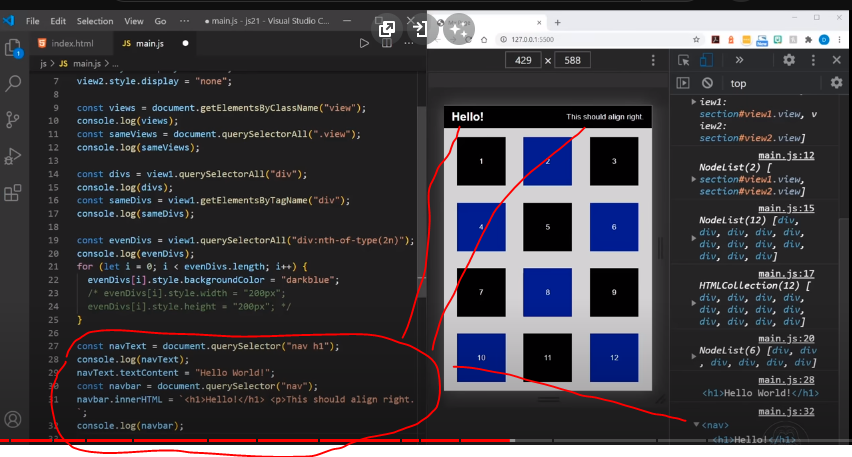












Finding HTML elements by id

const element = document.getElementById("intro");

The getElementById() method returns the elements that have given an ID which is passed to the function.

<script type="text/javascript">

function getcube(){

var number=document.getElementById("number").value;

alert(number\*number\*number);

}

</script>

<form>

Enter No:<input type="text" id="number" name="number"/><br/>

<input type="button" value="cube" onclick="getcube()"/>

</form>

Finding HTML elements by names

Syntax

let elements = document.getElementsByName(name);

<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("language");

Example

JavaScript getElementsByName() example

The following example shows a radio group that consists of radio buttons 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>

Finding HTML elements by tag name

const element = document.getElementsByTagName("p");

elements by tag name is used to select the selector of html tags like h1, p, main, body, html.

<script type="text/javascript">

function countpara(){

var totalpara=document.getElementsByTagName("p");

alert("total p tags are: "+totalpara.length);

}

</script>

<p>This is a pragraph</p>

<p>Here we are going to count total number of paragraphs by getElementByTagName() method.</p>

<p>Let's see the simple example</p>

<button onclick="countpara()">count paragraph</button>

Finding HTML elements by class name

const x = document.getElementsByClassName("intro");

The getElementsByClassName method of Document interface returns an array-like object of all child elements which have all of the given class name(s).

When called on the document object, the complete document is searched, including the root node. You may also call getElementsByClassName() on any element; it will return only elements which are descendants of the specified root element with the given class name(s).

Warning: This is a live HTMLCollection. Changes in the DOM will reflect in the array as the changes occur. If an element selected by this array no longer qualifies for the selector, it will automatically be removed. Be aware of this for iteration purposes.

<html lang="en">

<body>

<div id="parent-id">

<p>hello world 1</p>

<p class="test">hello world 2</p>

<p>hello world 3</p>

<p>hello world 4</p>

</div>

<script>

const parentDOM = document.getElementById("parent-id");

const test = parentDOM.getElementsByClassName("test"); // a list of matching elements, \*not\* the element itself

console.log(test); // HTMLCollection[1]

const testTarget = parentDOM.getElementsByClassName("test")[0]; // the first element, as we wanted

console.log(testTarget); // <p class="test">hello world 2</p>

</script>

</body>

</html>

Finding HTML elements by CSS selectors

const x = document.querySelectorAll("p.intro");

const x = document.querySelector("p.intro");

**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);Code language: JavaScript (javascript)

## 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');

## Combinators

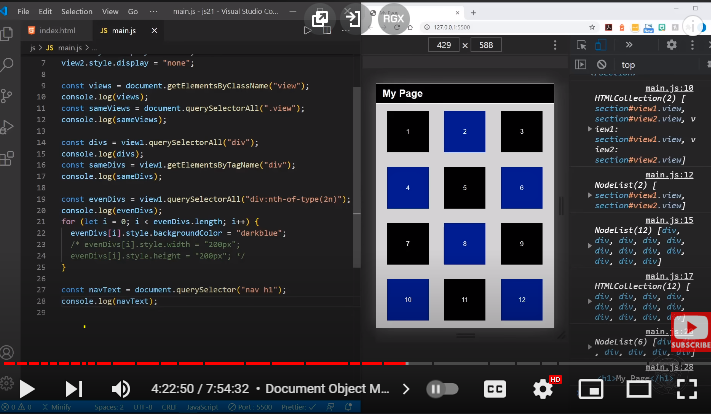
### **1) descendant combinator**

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');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');