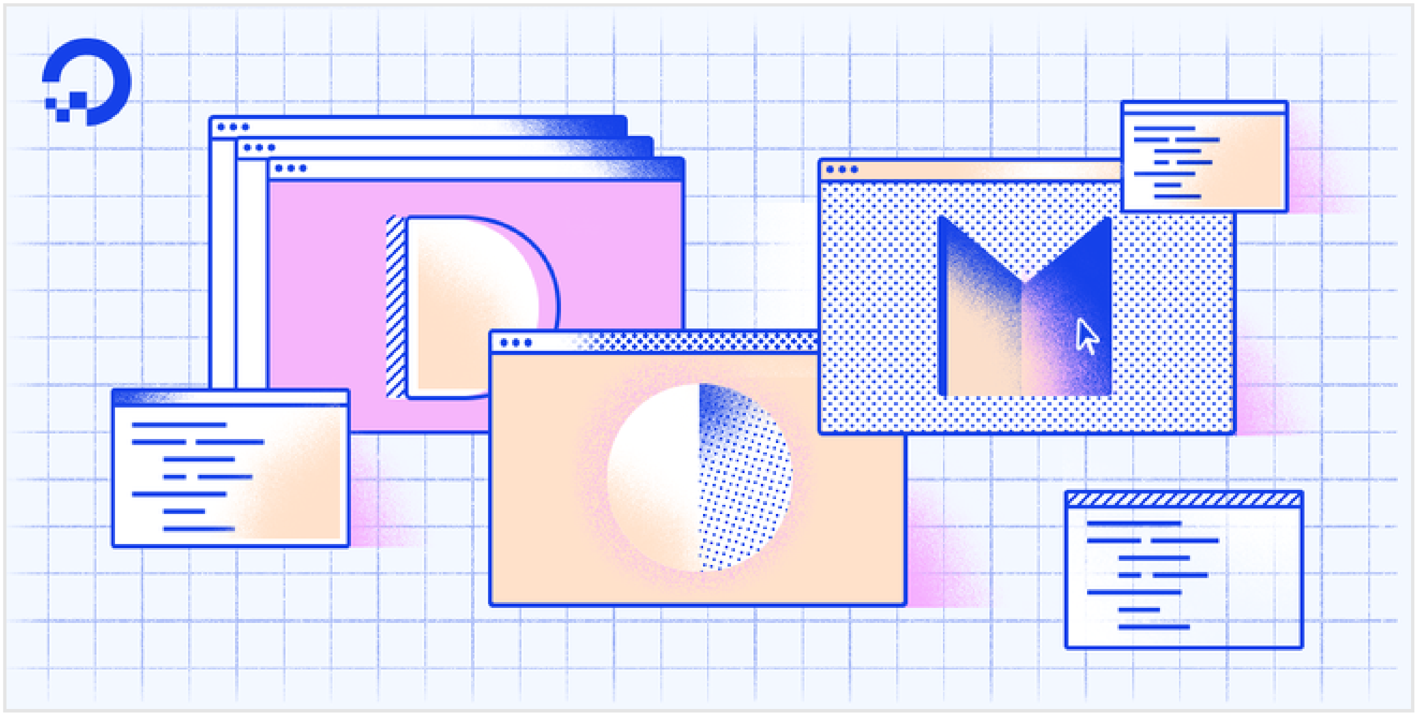
Contents



How To Access Elements in the DOM

Posted November 20, 2017 166.1k JAVASCRIPT DEVELOPMENT

By Tania Rascia
Become an author

Introduction

In Understanding the DOM Tree and Nodes, we went over how the DOM is structured as a tree of objects called nodes, and that nodes can be text, comments, or elements. Usually when we access content in the DOM, it will be through an HTML element node.

In order to be proficient at accessing elements in the DOM, it is necessary to have a working ...ng of H

Overview

Here is a table overview of the five methods we will cover in this tutorial.

Gets	Selector Syntax	Method
ID	`#demo`	`getElementById()`
Class	`.demo`	`getElementsByClassName()`
Tag	`demo`	`getElementsByTagName()`
Selector (single)		`querySelector()`
Selector (all)		`querySelectorAll()`

It is important when studying the DOM to type the examples on your own computer to ensure that you are understanding and retaining the information you learn.

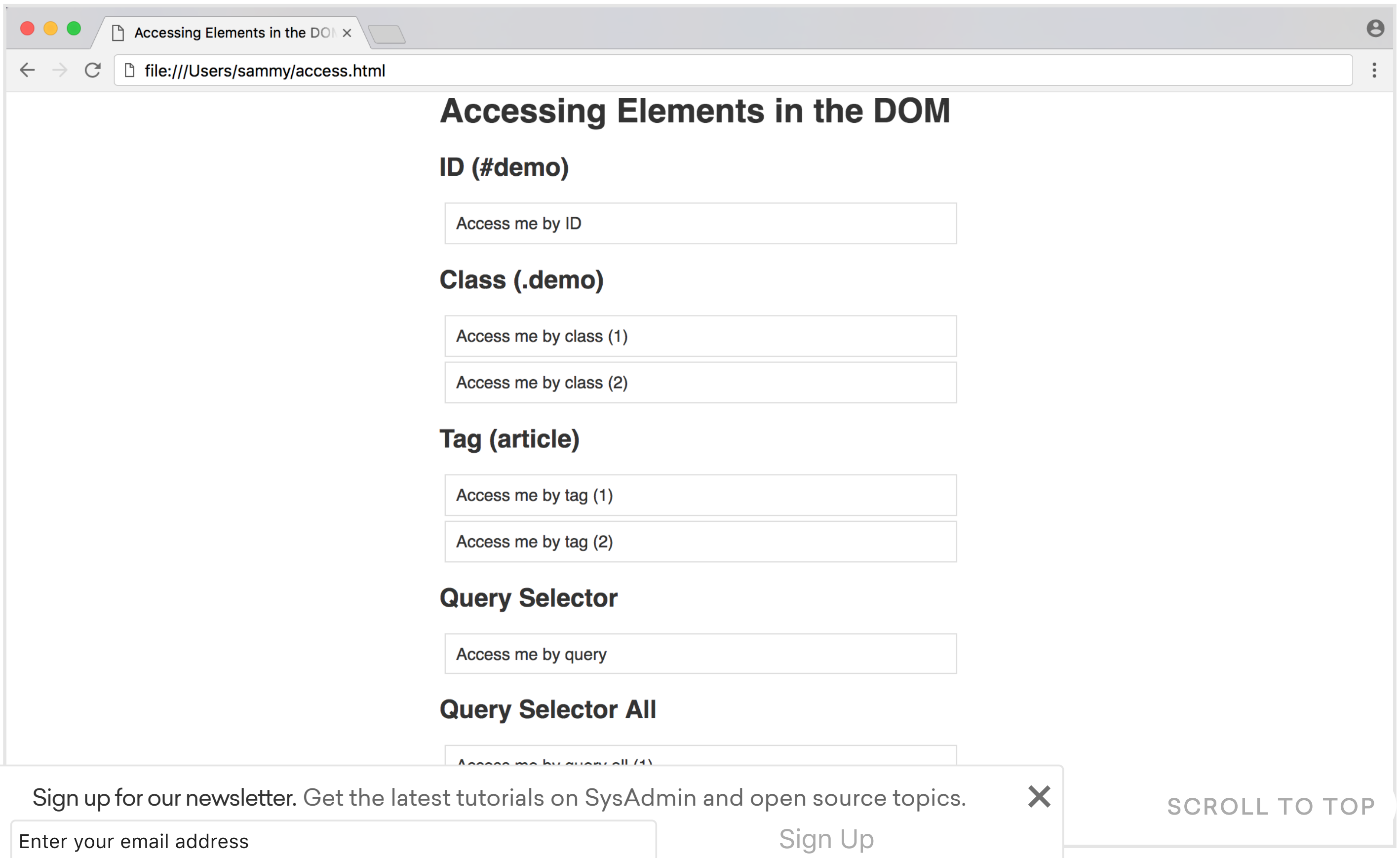
You can save this HTML file, `access.html`, to your own project to work through the examples along with this article. If you are unsure how to work with JavaScript and HTML locally, review our How To Add JavaScript to HTML tutorial.

access.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Accessing Elements in the DOM</title>

  <style>
    html { font-family: sans-serif; color: #333; }
    body { max-width: 500px; margin: 0 auto; padding: 0 15px; }
    div, article { padding: 10px; margin: 5px; border: 1px solid #dedede; }
  </style>

</head>
```

```
    <h2>Class (.demo)</h2>
    <div class="demo">Access me by class (1)</div>
    <div class="demo">Access me by class (2)</div>

    <h2>Tag (article)</h2>
    <article>Access me by tag (1)</article>

    <article>Access me by tag (2)</article>

    <h2>Query Selector</h2>
    <div id="demo-query">Access me by query</div>

    <h2>Query Selector All</h2>
    <div class="demo-query-all">Access me by query all (1)</div>
    <div class="demo-query-all">Access me by query all (2)</div>

</body>

</html>
```

In this HTML file, we have many elements that we will access with different `document` methods. When we render the file in a browser, it will look similar to this:


Accessing Elements in the DOM
file:///Users/sammy/access.html

Accessing Elements in the DOM

ID (#demo)

Access me by ID

Class (.demo)

Access me by class (1)

Access me by class (2)

Tag (article)

Access me by tag (1)

Access me by tag (2)

Query Selector

Access me by query

Query Selector All

SCROLL TO TOP

Accessing Elements by ID

The easiest way to access a single element in the DOM is by its unique ID. We can grab an element by ID with the `getElementById()` method of the `document` object.

```
document.getElementById();
```

In order to be accessed by ID, the HTML element must have an `id` attribute. We have a `div` element with an ID of `demo`.

```
<div id="demo">Access me by ID</div>
```

In the *Console*, let's get the element and assign it to the `demoId` variable.

```
> const demoId = document.getElementById('demo');
```

Logging `demoId` to the console will return our entire HTML element.

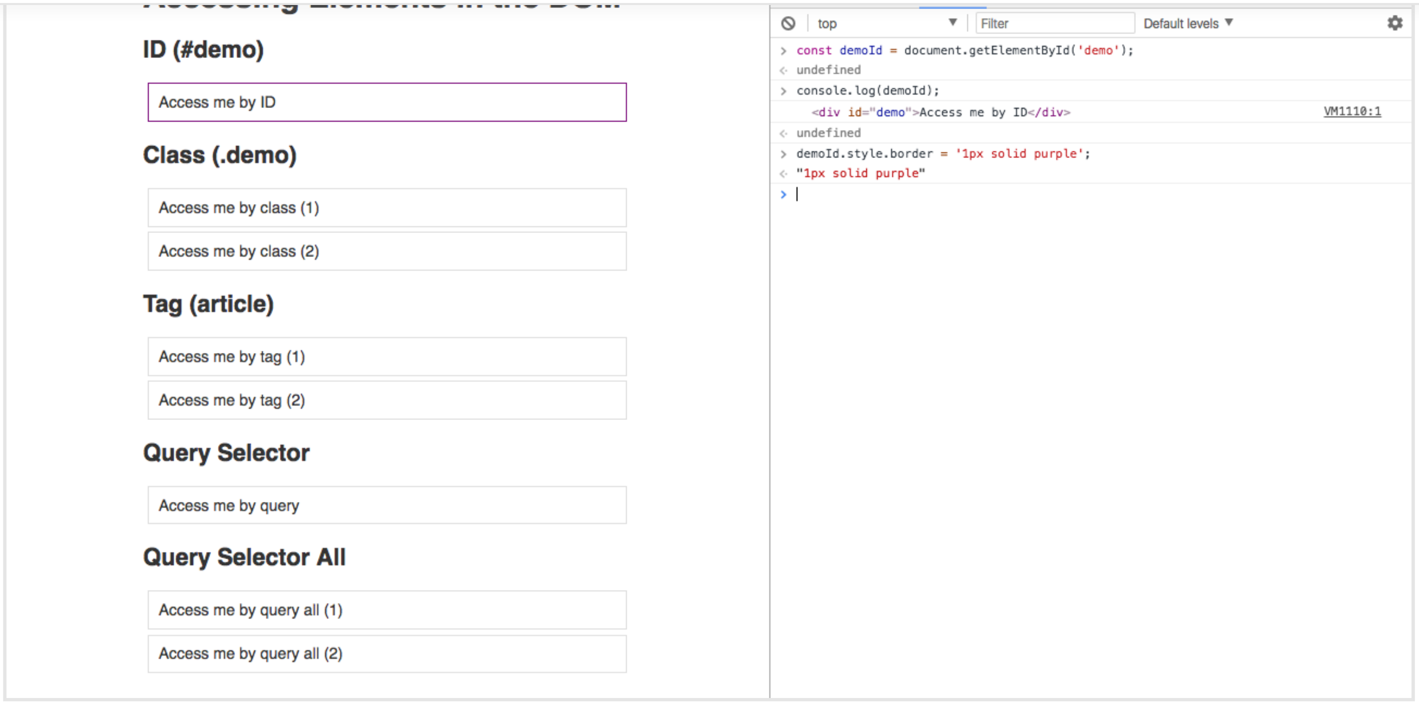
```
> console.log(demoId);
```

```
Output
<div id="demo">Access me by ID</div>
```

We can be sure we're accessing the correct element by changing the `border` property to `purple`.

```
> demoId.style.border = '1px solid purple';
```

Once we do so, our live page will look like this:



```
> const demoId = document.getElementById('demo');
<· undefined
> console.log(demoId);
    <div id="demo">Access me by ID</div>                    VM1110:1
<· undefined
> demoId.style.border = '1px solid purple';
<· "1px solid purple"
>
```

Accessing an element by ID is an effective way to get an element quickly in the DOM. However, it has drawbacks; an ID must always be unique to the page, and therefore you will only ever be able to access a single element at a time with the `getElementById()` method. If you wanted to add a function to many elements throughout the page, your code would quickly become repititious.

Accessing Elements by Class

The class attribute is used to access one or more specific elements in the DOM. We can get all the elements with a given class name with the `getElementsByClassName()` method.

```
document.getElementsByClassName();
```

Now we want to access more than one element, and in our example we have two elements with a `demo` class.

```
<div class="demo">Access me by class (1)</div>
<div class="demo">Access me by class (2)</div>
```

Let's access our elements in the *Console* and put them in a variable called `demoClass`.

elements to orange, we will get an error.

```
> demoClass.style.border = '1px solid orange';
```

```
Output
Uncaught TypeError: Cannot set property 'border' of undefined
```

The reason this doesn't work is because instead of just getting one element, we have an array-like object of elements.

```
> console.log(demoClass);
```

```
Output
(2) [div.demo, div.demo]
```

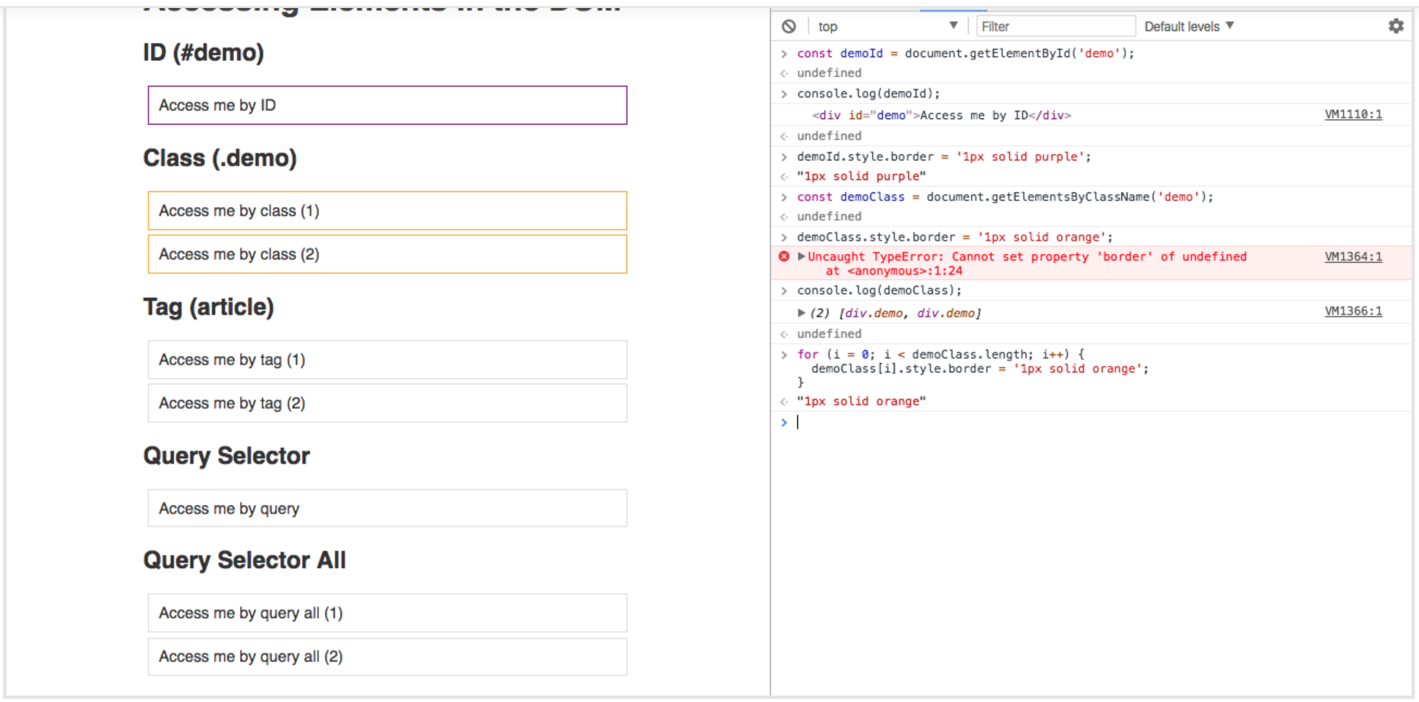
JavaScript arrays must be accessed with an index number. We can therefore change the first element of this array by using an index of `0`.

```
> demoClass[0].style.border = '1px solid orange';
```

Generally when accessing elements by class, we want to apply a change to all the elements in the document with that particular class, not just one. We can do this by creating a `for` loop, and looping through every item in the array.

```
> for (i = 0; i < demoClass.length; i++) {
>   demoClass[i].style.border = '1px solid orange';
> }
```

When we run this code, our live page will be rendered like this:



ID (#demo)

Access me by ID

Class (.demo)

Access me by class (1)

Access me by class (2)

Tag (article)

Access me by tag (1)

Access me by tag (2)

Query Selector

Access me by query

Query Selector All

Access me by query all (1)

Access me by query all (2)

```
> const demoId = document.getElementById('demo');
<· undefined
> console.log(demoId);
    <div id="demo">Access me by ID</div>                              VM1110:1
<· undefined
> demoId.style.border = '1px solid purple';
<· "1px solid purple"
> const demoClass = document.getElementsByClassName('demo');
<· undefined
> demoClass.style.border = '1px solid orange';
✖ ▶Uncaught TypeError: Cannot set property 'border' of undefined       VM1364:1
      at <anonymous>:1:24
> console.log(demoClass);
  ▶ (2) [div.demo, div.demo]                                           VM1366:1
<· undefined
> for (i = 0; i < demoClass.length; i++) {
    demoClass[i].style.border = '1px solid orange';
  }
<· "1px solid orange"
> |
```

We have now selected every element on the page that has a `demo` class, and changed the `border` property to `orange`.

Accessing Elements by Tag

A less specific way to access multiple elements on the page would be by its HTML tag name. We access an element by tag with the `getElementsByTagName()` method.

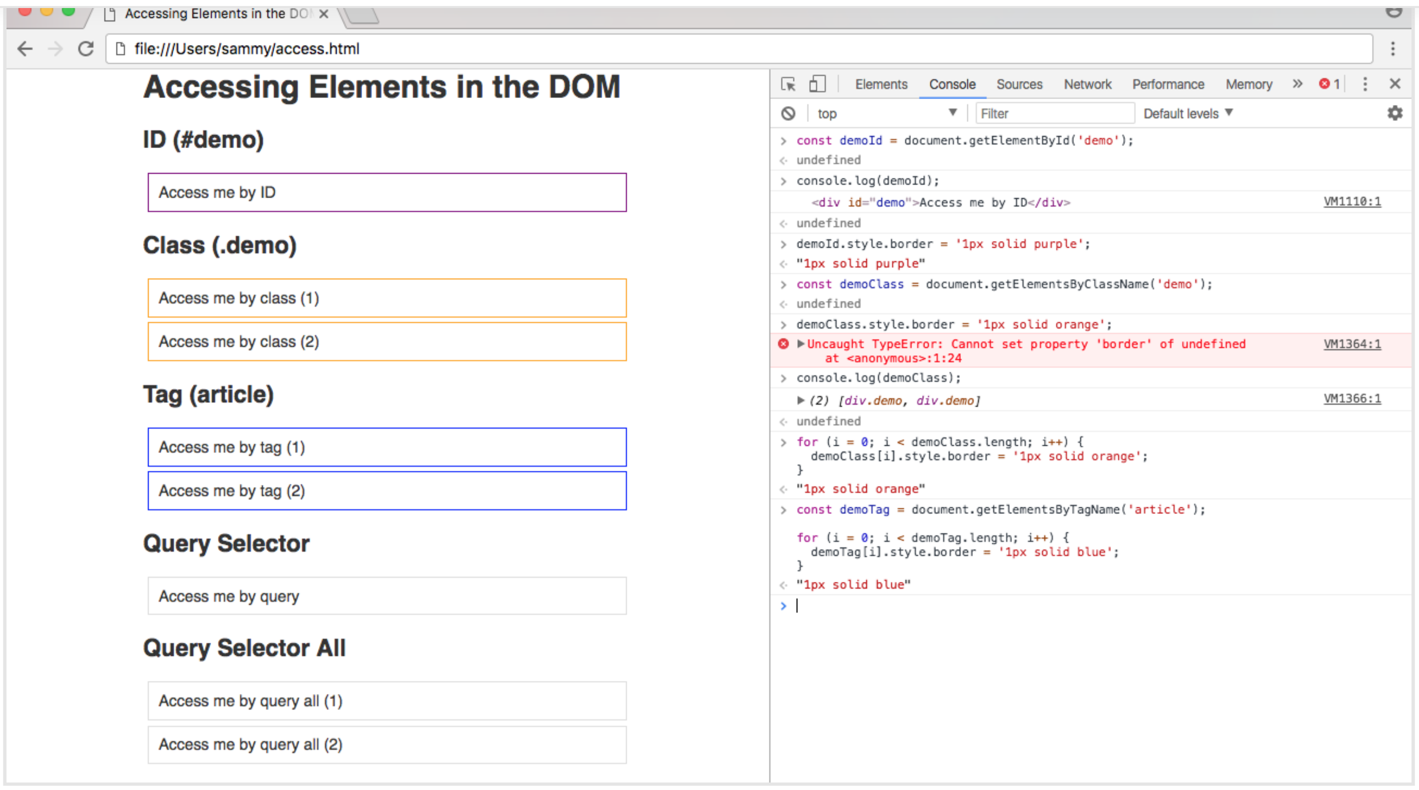
```
document.getElementsByTagName();
```

For our tag example, we're using `article` elements.

```
<article>Access me by tag (1)</article>
<article>Access me by tag (2)</article>
```

Just like accessing an element by its class, `getElementsByTagName()` will return an array-like object of elements, and we can modify every tag in the document with a `for` loop.

```
> const demoTag = document.getElementsByTagName('article');
>
> for (i = 0; i < demoTag.length; i++) {
```



Accessing Elements in the DOM

ID (#demo)

Access me by ID

Class (.demo)

Access me by class (1)

Access me by class (2)

Tag (article)

Access me by tag (1)

Access me by tag (2)

Query Selector

Access me by query

Query Selector All

Access me by query all (1)

Access me by query all (2)

```
> const demoId = document.getElementById('demo');
< undefined
> console.log(demoId);
    <div id="demo">Access me by ID</div>                    VM1110:1
< undefined
> demoId.style.border = '1px solid purple';
< "1px solid purple"
> const demoClass = document.getElementsByClassName('demo');
< undefined
> demoClass.style.border = '1px solid orange';
✖ ▶Uncaught TypeError: Cannot set property 'border' of undefined   VM1364:1
      at <anonymous>:1:24
> console.log(demoClass);
  ▶ (2) [div.demo, div.demo]                                VM1366:1
< undefined
> for (i = 0; i < demoClass.length; i++) {
    demoClass[i].style.border = '1px solid orange';
  }
< "1px solid orange"
> const demoTag = document.getElementsByTagName('article');

  for (i = 0; i < demoTag.length; i++) {
    demoTag[i].style.border = '1px solid blue';
  }
< "1px solid blue"
> |
```

The loop changed the `border` property of all `article` elements to `blue`.

Query Selectors

If you have any experience with the jQuery API, you may be familiar with jQuery's method of accessing the DOM with CSS selectors.

```
$('#demo'); // returns the demo ID element in jQuery
```

We can do the same in plain JavaScript with the `querySelector()` and `querySelectorAll()` methods.

```
document.querySelector();
document.querySelectorAll();
```

To access a single element, we will use the `querySelector()` method. In our HTML file, we have a `demo-query` element

```
<div id="demo-query">Access me by query</div>
```

```
> const demoQuery = document.querySelector('#demo-query');
```

In the case of a selector with multiple elements, such as a class or a tag, `querySelector()` will return the first element that matches the query. We can use the `querySelectorAll()` method to collect all the elements that match a specific query.

In our example file, we have two elements with the `demo-query-all` class applied to them.

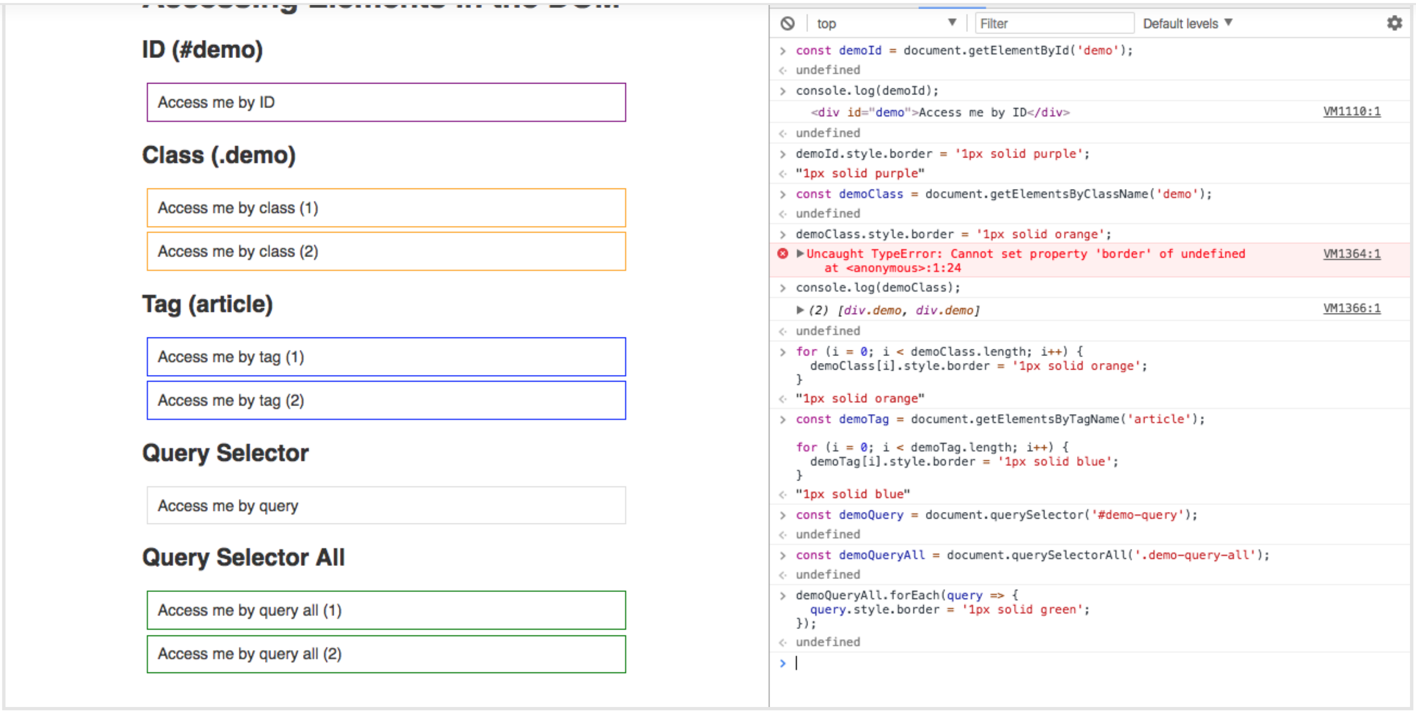
```
<div class="demo-query-all">Access me by query all (1)</div>
<div class="demo-query-all">Access me by query all (2)</div>
```

The selector for a `class` attribute is a period or full stop (`.`), so we can access the class with `.demo-query-all`.

```
> const demoQueryAll = document.querySelectorAll('.demo-query-all');
```

Using the `forEach()` method, we can apply the color `green` to the `border` property of all matching elements.

```
> demoQueryAll.forEach(query => {
>   query.style.border = '1px solid green';
> });
```


Accessing Elements in the DOM

ID (#demo)

Access me by ID

Class (.demo)

Access me by class (1)

Access me by class (2)

Tag (article)

Access me by tag (1)

Access me by tag (2)

Query Selector

Access me by query

Query Selector All

Access me by query all (1)

Access me by query all (2)

```
> const demoId = document.getElementById('demo');
< undefined
> console.log(demoId);
    <div id="demo">Access me by ID</div>                              VM1110:1
< undefined
> demoId.style.border = '1px solid purple';
< "1px solid purple"
> const demoClass = document.getElementsByClassName('demo');
< undefined
> demoClass.style.border = '1px solid orange';
✖ ▶Uncaught TypeError: Cannot set property 'border' of undefined      VM1364:1
      at <anonymous>:1:24
> console.log(demoClass);
  ▶ (2) [div.demo, div.demo]                                          VM1366:1
< undefined
> for (i = 0; i < demoClass.length; i++) {
    demoClass[i].style.border = '1px solid orange';
  }
< "1px solid orange"
> const demoTag = document.getElementsByTagName('article');

  for (i = 0; i < demoTag.length; i++) {
    demoTag[i].style.border = '1px solid blue';
  }
< "1px solid blue"
> const demoQuery = document.querySelector('#demo-query');
< undefined
> const demoQueryAll = document.querySelectorAll('.demo-query-all');
< undefined
> demoQueryAll.forEach(query => {
    query.style.border = '1px solid green';
  });
< undefined
> |
```

With `querySelector()`, comma-separated values function as an OR operator. For example, `querySelector('div, article')` will match `div` *or* `article`, whichever appears first in the document. With `querySelectorAll()`, comma-separated values function as an AND operator, and `querySelectorAll('div, article')` will match all `div` *and* `article` values in the document.

Using the query selector methods is extremely powerful, as you can access any element or group of elements in the DOM the same way you would in a CSS file. For a complete list of selectors, review CSS Selectors on the Mozilla Developer Network.

Complete JavaScript Code

Below is the complete script of the work we did above. You can use it to access all the elements on our example page. Save the file as `access.js` and load it in to the HTML file right before the closing `body` tag.

access.js

```
// Assign all elements
const demoId = document.getElementById('demo');
const demoClass = document.getElementsByClassName('demo');
const demoTag = document.getElementsByTagName('article');
const demoQuery = document.querySelector('#demo-query');
const demoQueryAll = document.querySelectorAll('.demo-query-all');
```

```
  demoClass[i].style.border = '1px solid orange';
}

// Change border of tag demo to blue
for (i = 0; i < demoTag.length; i++) {
  demoTag[i].style.border = '1px solid blue';
}

// Change border of ID demo-query to red
demoQuery.style.border = '1px solid red';

// Change border of class query-all to green
demoQueryAll.forEach(query => {
  query.style.border = '1px solid green';
});
```

Your final HTML file will look like this:

access.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Accessing Elements in the DOM</title>

  <style>
    html { font-family: sans-serif; color: #333; }
    body { max-width: 500px; margin: 0 auto; padding: 0 15px; }
    div, article { padding: 10px; margin: 5px; border: 1px solid #dedede; }
  </style>

</head>

<body>

  <h1>Accessing Elements in the DOM</h1>

  <h2>ID (#demo)</h2>
  <div id="demo">Access me by ID</div>
  <article>Access me by tag (1)</article>
  <article>Access me by tag (2)</article>

  <h2>Query Selector</h2>
  <div id="demo-query">Access me by query</div>

  <h2>Query Selector All</h2>
  <div class="demo-query-all">Access me by query all (1)</div>
  <div class="demo-query-all">Access me by query all (2)</div>

  <script src="access.js"></script>

</body>

</html>
```

You can continue to work on these template files to make additional changes by accessing HTML elements.

Conclusion

In this tutorial, we went over 5 ways to access HTML elements in the DOM — by ID, by class, by HTML tag name, and by selector. The method you will use to get an element or group of elements will depend on browser support and how many elements you will be manipulating. You should now feel confident to access any HTML element in a document with JavaScript through the DOM.

By Tania Rascia

Editor: Lisa Tagliaferri

Tutorial Series

manipulate the content, structure, and style of a website. JavaScript is the client-side scripting language that connects to the DOM in an internet browser.

Show Tutorials

Related Tutorials

How To Build a Search Bar with RxJS

How To Build a Weather App with Angular, Bootstrap, and the APIXU API

How To Build a Blog with Nest.js, MongoDB, and Vue.js

How to Use Node.js and Github Webhooks to Keep Remote Projects in Sync

How to Generate a Short and Unique Digital Address for Any Location Using AngularJS and PHP

2 Comments

Leave a comment...

Log In to Comment

jmarinfotecnico *March 9, 2019*

he array idea behind these

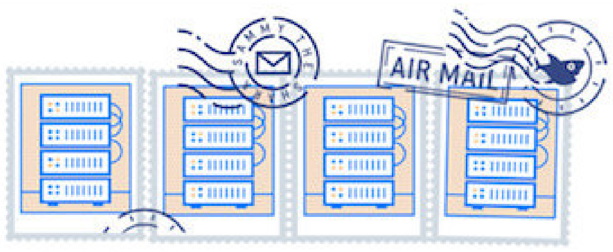
SCROLL TO TOP

paroche *April 25, 2019*

0 Hi Tania, I too thought it was a great tutorial -- all of them are so far. Really helpful. There is one thing, though, that I think could be improved. When you are processing elements that were accessed using .getElementsByClassName, or .getElementsByTagName, you handle the returned variable with a for loop, but when you are processing elements that were accessed using .querySelectorAll you handle the returned variable with the .forEach method. I think it would have been good to have some explanation for why -- which, if I understand correctly, is because the "getElementsBy...()" methods return "HTMLCollection" objects, which cannot be processed with .forEach like arrays, while the querySelectorAll() method returns "NodeList" objects, which, in modern browsers, CAN be processed with .forEach like arrays. I think a little explanation about that would have been helpful.

CONNECT WITH OTHER DEVELOPERS

Find a DigitalOcean Meetup near you.



GET OUR BIWEEKLY NEWSLETTER

Sign up for Infrastructure as a Newsletter.

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the cloud and scale up as you grow – whether you're running one virtual machine or ten thousand.

Learn More

jonsmith
jonsmith DEFAULT
→ Move Resources