



Home

Book

About

Contacts

Get element(s) by data attribute using JavaScript



Borislav Hadzhiev

Last updated: Jan 11, 2023

Reading time · 7 min



BobbyHadz . com

Get element(s) by data attribute using JavaScript

JS

Table of Contents

1. [Get element by data attribute using JavaScript](#)
2. [Get element by Partially matching a Data Attribute](#)
3. [Get all Elements by data attribute using JavaScript](#)
4. [Get all DOM elements by Partial Match of a Data Attribute](#)
5. [Get all data-* attributes of a DOM Element](#)

Get element by data attribute using JavaScript

Use the `querySelector` method to get an element by data attribute.

The `querySelector` method returns the first element that matches the provided selector or `null` if no element matches the selector in the document.

Here is the HTML for the examples.

index.html



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>bobbyhadz.com</title>
  </head>





  <body>
    <div data-id="box1">Box 1</div>
    <span data-id="box2">Box 2</span>

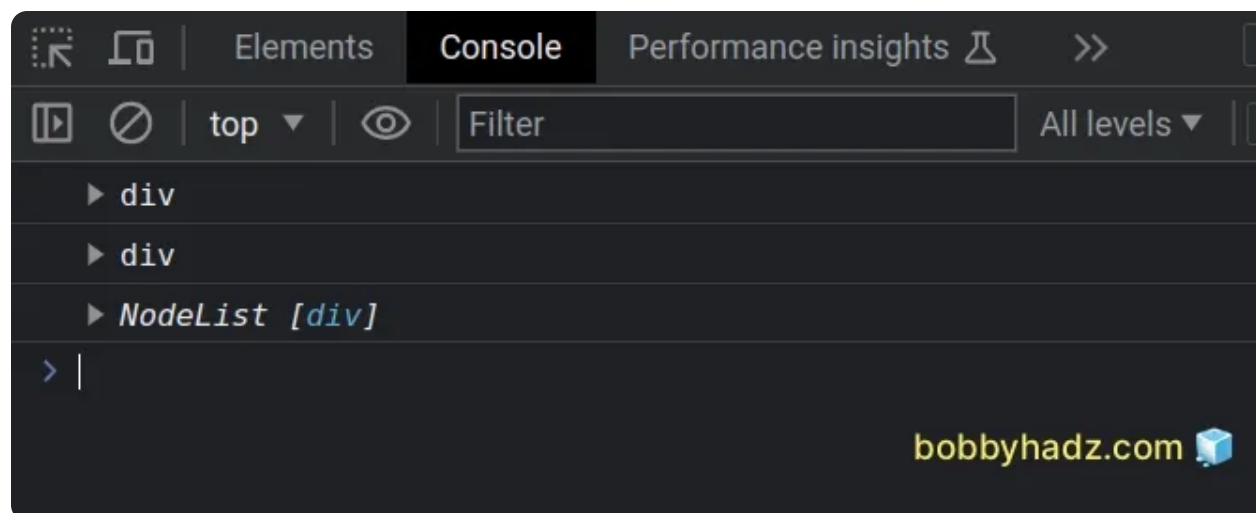
    <script src="index.js"></script>
  </body>
</html>
```

And here is the related JavaScript code.

index.js

```
//  Get the first element with data-id = `box1`
const el1 = document.querySelector('[data-id="box1"]');
console.log(el1); //  div
```

```
// -----  
  
//  Get the first element that has data-id attribute set  
const el2 = document.querySelector('[data-id]');  
console.log(el2); //  div  
  
// -----  
  
//  Get all elements with data-id = `box1`  
const elements = document.querySelectorAll('[data-id="box1"]');  
console.log(elements); //  [div]
```



-  If you need a collection with all elements that have a specific data

attribute, use the `querySelectorAll` method. The method takes the same parameter as `querySelector`.

We used the [document.querySelector](#) method to get the first element in the DOM that has a `data-id` attribute equal to `box1`.

- i If you have to partially match a data attribute, scroll down to the next subheading.

The second example shows how to get the first element that has the `data-id` attribute set to any value.

index.js

```
const el2 = document.querySelector('[data-id]');
```

You can also narrow things down by looking for a specific type of element, e.g. a `div` that has a data attribute set to a certain value.

index.js

```
const el1 = document.querySelector('div[data-id="box1"]');
```

```
console.log(e11); // 📌 div
```

This example only selects `div` elements that have a `data-id` attribute set to `box1`.

- 📘 If you need a collection of elements, instead of the first element that matches a selector, simply replace `querySelector` with `querySelectorAll`.

Get element by Partially matching a Data Attribute

To get an element by partially matching a data attribute, use the `querySelector` method with a selector that matches a data attribute whose value starts with, ends with or contains a specific string.

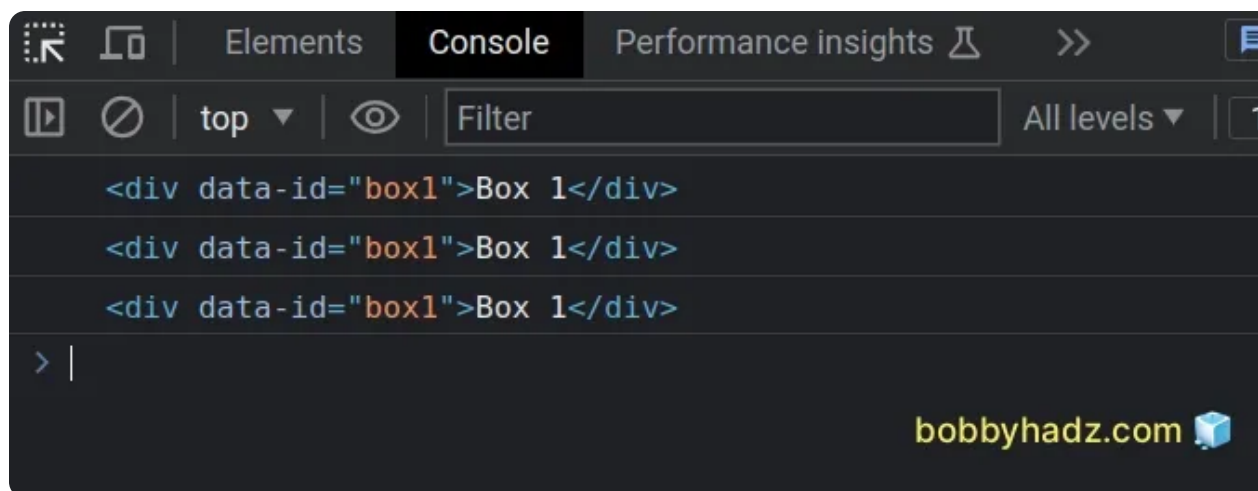
`index.js`

```
// ✅ get element where data-id starts with `bo`  
const e11 = document.querySelector('[data-id^="bo"]');
```

```
console.log(e11); // 📌 div

// ✅ get element where data-id ends with `ox1`
const e12 = document.querySelector('[data-id$="ox1"]');
console.log(e12); // 📌 div

// ✅ get element where data-id contains `box`
const e13 = document.querySelector('[data-id*="box"]');
console.log(e13); // 📌 div
```




- ❗ Any of the examples above can be replaced with the `querySelectorAll` method to get a collection of elements matching the selector.

The first example selects the first DOM element that has a `data-id` attribute, whose value starts with `bo`.

index.js

```
const el1 = document.querySelector('[data-id^="bo"]');
```

-  You might be familiar with the caret `^` symbol, which has the same meaning when used with regular expressions.

The second example selects the first DOM element that has a `data-id` attribute that ends with `ox1`.


index.js

```
const el2 = document.querySelector('[data-id$="ox1"]');
```

The third example selects the first DOM element that has a `data-id` attribute that contains `box`.

index.js

```
const el3 = document.querySelector('[data-id*="box"]');
```

-  The string ``box`` could be located anywhere in the value of the ``data-id`` attribute for the condition to be met.

You could also prefix the selector with a specific type of element that you want to match to narrow down the results.

`index.js`

```
const el1 = document.querySelector('div[data-id^="bo"]');
```

The example selects a ``div`` element that has a ``data-id`` attribute that starts with ``bo``.

Want to learn more about working with data attributes? Check out these resources: [How to check if an attribute exists using JavaScript](#), [Access data-* attributes from the Event object in JavaScript](#).

Get all Elements by data attribute using JavaScript

You can use the `querySelectorAll` method to get all elements by data attribute.

The `querySelectorAll` method returns a `NodeList` containing the elements that match the specified selector.

Here is the HTML code for this example.

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>bobbyhadz.com</title>
  </head>

  <body>
    <div data-id="box1">Box 1</div>
    <div data-id="box2">Box 2</div>
```

```
    <script src="index.js"></script>
  </body>
</html>
```

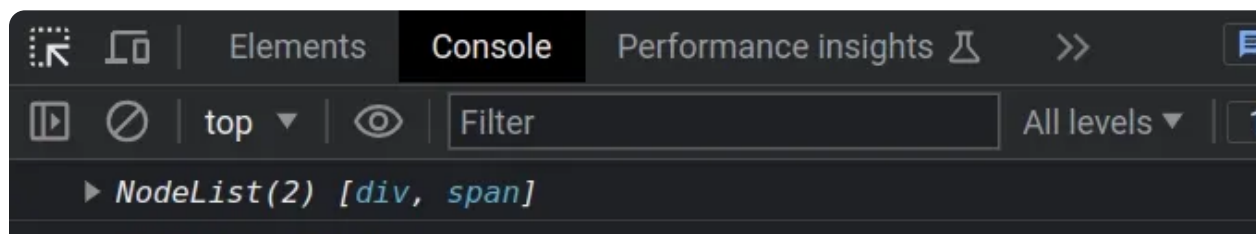
And here is the related JavaScript code.

index.js


```
// ✅ Get all elements with `data-id` attribute
const elements1 = document.querySelectorAll('[data-id]');
console.log(elements1); // 👉 [div, div]

// ✅ Get only DIV elements with `data-id` attribute
const elements2 = document.querySelectorAll('div[data-id]');
console.log(elements2); // 👉 [div, div]

// ✅ Get only elements where data-id = box1
const elements3 = document.querySelectorAll(`[data-id="box1"]`);
console.log(elements3); // 👉 [div]
```



```
▶ NodeList [div]
▶ NodeList [div]
>
```

bobbyhadz.com 

We passed different selectors to the [document.querySelectorAll](#) method to get a `NodeList` containing the specific DOM elements.

Note that the `querySelectorAll` method returns a `NodeList`, not an array. If you need to convert the response to an array, pass it to the `Array.from()` method.

`index.js`

```
const arr = Array.from(document.querySelectorAll('[data-id]'));
```

Get all elements that have data- attribute set

The first example shows how to get all DOM elements that have the `data-id` attribute set.

`index.js`

```
const elements1 = document.querySelectorAll('[data-id]');
```

Get all div elements that have data- attribute set

In the second example, we narrow things down to only `div` elements that have the `data-id` attribute set.

`index.js`

```
const elements2 = document.querySelectorAll('div[data-id]');
```

If the DOM contained any `span` or `p` elements that have the `data-id` attribute set, they wouldn't be included in the return value of the `querySelectorAll` method.


Get all elements by exact match data-attribute

The third example selects elements that have the `data-id` attribute set to `box1`.

`index.js`

```
const elements3 = document.querySelectorAll('[data-id="box1"]');
```







Note that the `data-id` attribute has to be set to exactly `box1`.

-  For selecting elements based on a partial match of a specific attribute value, scroll down to the next example.

Get all DOM elements by Partial Match of a Data Attribute

The `querySelectorAll` method can also be used to get all DOM elements by partial match of a data attribute.

`index.js`

```
//  Get all where value of data-id starts with `bo`  
const elements1 = document.querySelectorAll('[data-id^="bo"]');  
console.log(elements1); //  [div, div]  
  
//  Get all where value of data-id ends with `ox1`  
const elements2 = document.querySelectorAll('[data-id$="ox1"]');  
console.log(elements2); //  [div]  
  
//  Get all where value of data-id contains with `box`  
const elements3 = document.querySelectorAll('[data-id*="box"]');  
console.log(elements3); //  [div, div]
```

Get all DOM elements with data- attribute starting with

The first example selects all DOM elements where the value of the `data-id` attribute starts with `bo`.

index.js

```
const elements1 = document.querySelectorAll('[data-id^="bo"]');
```

- ❗ You might be familiar with the caret `^` symbol, which has the same meaning when used in regular expressions.

Get all DOM elements with data- attribute ending with

The second example selects all DOM elements where the value of the `data-id` attribute ends with `ox1`.

index.js

```
const elements2 = document.querySelectorAll('[data-id$="ox1"]');
```

Get all DOM elements with data- attribute containing

The third example selects all DOM elements where the value of the `data-id` attribute contains the string `box`.

index.js

```
const elements3 = document.querySelectorAll('[data-id*="box"]');
```

- ❗ The string `box` could be located anywhere in the value of the `data-id` attribute for the condition to be met.

You could also prefix the selector with a specific type of element you want to match to narrow down the results.

index.js

```
const elements1 = document.querySelectorAll('div[data-id^="bo"]');
```

The example selects only `div` elements that have a `data-id` attribute set and the attribute's value starts with `bo`.

Get all data-* attributes of a DOM Element

Use the `dataset` property to get all of the data-* attributes of a DOM element,

e.g. ``el.dataset``.

The ``dataset`` property provides read and write access to the custom data attributes of the element. The property returns a ``Map`` of strings that can be converted to an object.

Here is the ``HTML`` for the example.

`index.html`

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>bobbyhadz.com</title>
  </head>

  <body>
    <div id="box" data-bar="foo" data-id="example">Box 1</div>

    <script src="index.js"></script>
  </body>
</html>
```

And here is the related JavaScript code.

index.js

```
const el = document.getElementById('box');

// 📌 DOMStringMap {bar: 'foo', id: 'example'}
console.log(el.dataset);

// ✅ Optionally convert it to Object
const obj = {...el.dataset};
console.log(obj); // 📌 {bar: 'foo', id: 'example'}
```

 For an alternative solution, scroll down to the next example.

We used the [dataset](#) property on the HTML element.

The property returns a read-only `Map` of strings containing the custom data attributes of the element.

The `dataset` property itself is [read-only](#). To update a specific property, we must access the nested property.

index.js

```
console.log(el.dataset.id); // 📌 "example"

el.dataset.id = 'updated id';

console.log(el.dataset.id); // 📌 "updated id"
```

To update the `data-id` attribute, we accessed the `id` property on the `dataset` object and assigned a new value to it.

If you want to convert the `dataset` Map of strings to a native JavaScript object, you can use the spread syntax (...).

index.js

```
// ✅ Optionally convert it to Object
const obj = {...el.dataset};
console.log(obj); // 📌 {bar: 'foo', id: 'example'}
```

⚠ However, if you now update a specific property on the object, the changes will not get reflected in the DOM.

You could also implement a minimal replacement for the `dataset` property.

`index.js`

```
const el = document.getElementById('box');

const dataAttrs = el.getAttributeNames().reduce((obj, name) => {
  if (name.startsWith('data-')) {
    return {...obj, [name.slice(name.indexOf('-') + 1)]: el.getAttribute(name)}
  }
  return obj;
}, {});

console.log(dataAttrs); // 📦 {bar: 'foo', id: 'example'}
```

We used the `getAttributeNames` method to get the names of all of the attributes of the DOM element.

The next step is to use the `reduce()` method to iterate over the array of attribute names.

On each iteration, we check if the attribute's name starts with `data-` and if it does we assign the key/value mapping to the object.

This approach has many differences from the `dataset` property and is only supposed to be used if for some reason you don't have access to the `dataset` property.

Additional Resources

You can learn more about the related topics by checking out the following tutorials:

[Access data-* attributes from the Event object in JavaScript](#)

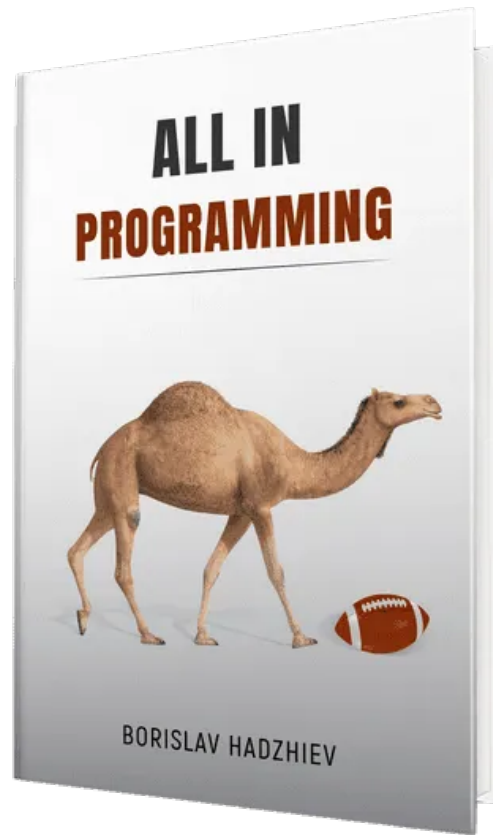
[How to check if an attribute exists using JavaScript](#)

[querySelector Attribute contains/starts with Examples in JS](#)

[Set multiple Attributes on an Element using JavaScript](#)

[How to Get the information from a meta tag using JavaScript](#)

- ✓ I wrote [a book](#) in which I share everything I know about how to become a better, more efficient programmer.



- ✓ You can use the search field on my [Home Page](#) to filter through all of my articles.



[About](#)

[Contacts](#)

[Policy](#)

[Terms & Conditions](#)



Copyright © 2023 Borislav Hadzhiev
