Document: getElementById() method

- Syntax
  - Parameters
  - Return value
- Examples
  - HTML
  - JavaScript
  - Result
- Usage notes
  - Example
- Specifications
- Browser compatibility
- See also

The `getElementById()` method of the Document interface returns an Element object representing the element whose id property matches the specified string. Since element IDs are required to be unique if specified, they're a useful way to get access to a specific element quickly.

If you need to get access to an element which doesn't have an ID, you can use querySelector() to find the element using any selector.

> **NOTE:** IDs should be unique inside a document. If two or more elements in a document have the same ID, this method returns the first element found.

Syntax

```
getElementById(id)
```

> **NOTE:** The capitalization of `"Id"` in the name of this method *must* be correct for the code to function; `getElementByID()` is *not* valid and will not work, however natural it may seem.

Parameters

- `id`
  - : The ID of the element to locate. The ID is a case-sensitive string which is unique within the document; only one element should have any given ID.

Return value

An Element object describing the DOM element object matching the specified ID, or `null` if no matching element was found in the document.

Examples

HTML

```
<p id="para">Some text here</p>
<button>blue</button>
<button>red</button>
```

JavaScript

```
function changeColor(newColor) {
  const elem = document.getElementById("para");
  elem.style.color = newColor;
}

document.querySelectorAll("button").forEach((button) => {
```

```
  button.addEventListener("click", (event) => {
    changeColor(event.target.textContent.toLowerCase());
  });
});
```

Result

Usage notes

Unlike some other element-lookup methods such as Document.querySelector() and Document.querySelectorAll(), `getElementById()` is only available as a method of the global `document` object, and *not* available as a method on all element objects in the DOM. Because ID values must be unique throughout the entire document, there is no need for "local" versions of the function.

Example

```
<div id="parent-id">
  <p>hello word1</p>
  <p id="test1">hello word2</p>
  <p>hello word3</p>
  <p>hello word4</p>
</div>
```

```
const parentDOM = document.getElementById("parent-id");
const test1 = parentDOM.getElementById("test1");
```

If there is no element with the given `id`, this function returns `null`. Note that the `id` parameter is case-sensitive, so `document.getElementById("Main")` will return `null` instead of the element `<div id="main">` because "M" and "m" are different for the purposes of this method.

Elements not in the document are not searched by `getElementById()`. When creating an element and assigning it an ID, you have to insert the element into the document tree with Node.insertBefore() or a similar method before you can access it with `getElementById()`:

```
const element = document.createElement("div");
element.id = "test";
```

```
const el = document.getElementById("test"); // el will be null!
```

In non-HTML documents, the DOM implementation must have information on which attributes are of type ID. Attributes with the name "id" are not of type ID unless so defined in the document's DTD. The `id` attribute is defined to be of ID type in the common cases of XHTML, XUL, and others. Implementations that do not know whether attributes are of type ID or not are expected to return `null`.

Specifications

Browser compatibility

See also

- Document reference for other methods and properties you can use to get references to elements in the document.
- Document.querySelector() for selectors via queries like `'div.myclass'`
- Document.evaluate() - has a utility method for selecting by `xml:id` in XML documents

---

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:33:46 UTC