The `Document` method **`getElementById()`** 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.

Syntax

```
var element = document.getElementById(id);
```

Parameters

`id`
The ID of the element to locate. The ID is case-sensitive string which is unique within the document; only one element may 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.

Example

HTML

```
<html>
<head>
  <title>getElementById example</title>
</head>
<body>
```

```
  <p id="para">Some text here</p>
  <button onclick="changeColor('blue');">blue</button>
  <button onclick="changeColor('red');">red</button>
</body>
</html>
```

JavaScript

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

Result

Usage notes

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.

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

```
<!doctype html>
<html>
<head>
```

```
    <meta charset="UTF-8">
    <title>Document</title>
</head>

<body>
    <div id="parent-id">
        <p>hello word1</p>
        <p id="test1">hello word2</p>
        <p>hello word3</p>
        <p>hello word4</p>
    </div>
    <script>
        var parentDOM = document.getElementById('parent-id');
        var test1 = parentDOM.getElementById('test1');
        //throw error
        //Uncaught TypeError: parentDOM.getElementById is not a functio
    </script>
</body>
</html>
```

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()`:

```
var element = document.createElement('div');
element.id = 'testqq';
var el = document.getElementById('testqq'); // el will be null!
```

Non-HTML documents. The DOM implementation must have information that says 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 other. Implementations that do not know whether attributes are of type ID or not are expected to return `null`.