Get Started with the DOM in JavaScript

The DOM represents a web page as a hierarchy of objects, where each object corresponds to a node in the nested HTML element tree. DOM objects have properties and methods that you can manipulate with JavaScript.

WE'LL COVER THE FOLLOWING

- Access the DOM with the `document` Variable
- Discover a Node's Type
- Access a Node's Children
- Browse Child Nodes
- Access a Node's Parent

Access the DOM with the `document` Variable

When a JavaScript program runs in the context of a web browser, it can access the root of the DOM using the variable `document`. This variable matches the `<html>` element. `document` is an object that has head and body properties which allow access to the `<head>` and `<body>` elements of the page.

JavaScript

```
const h = document.head; // "h" variable contains the contents of the DOM's head
const b = document.body; // "b" variable contains the contents of the DOM's body
```

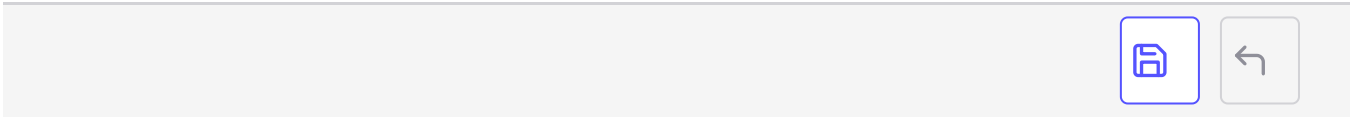
Discover a Node's Type

Each object has a property called `nodeType` which indicates its type. The value of this property is `document.ELEMENT_NODE` for an "element" node (otherwise

known as an HTML tag) and `document.TEXT_NODE` for a text node.

JavaScript

```
if (document.body.nodeType === document.ELEMENT_NODE) {
  console.log("Body is an element node!");
} else {
  console.log("Body is a textual node!");
}
```



Console

Clear

```
Body is an element node!
```

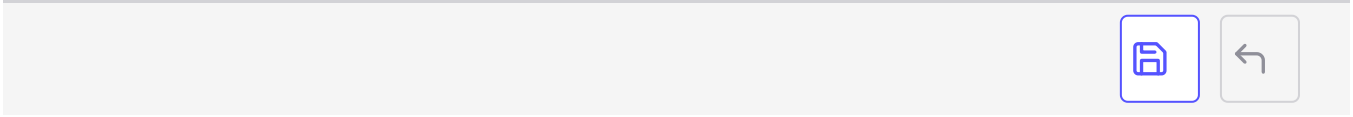
As expected, the DOM object body is an element node because it's an HTML tag.

Access a Node's Children

Each element-typed object in the DOM has a property called `childNodes`. This is an ordered collection containing all its child nodes as DOM objects. You can use this array-like collection to access the different children of a node. The `childNodes` property of an element node is not a real JavaScript array, but rather a NodeList object. Not all of the standard array methods are applicable to it. The following code would display the first child of the `body` node.

JavaScript

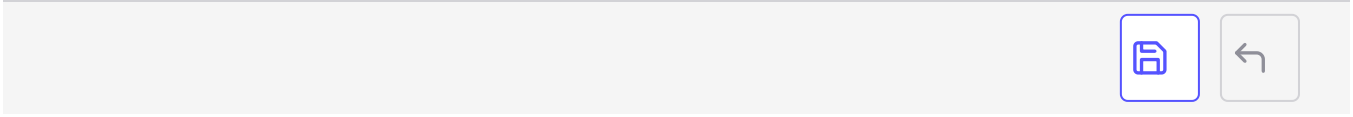
```
// Access the first child of the body node
console.log(document.body.childNodes[0]);
```



Wait... Why isn't the first child node h1, since that's the first element in the body's HTML? That's because spaces between tags and line returns in HTML code are considered text nodes by the browser. The node h1 is therefore the second child node of the body. Let's double check that:

JavaScript

```
// Access the second child of the body node
console.log(document.body.childNodes[1]);
```



To eliminate these text nodes between tags, you could have written the HTML page in a more condensed way.

```
<body> <h1>My web page</h1><!-- ... -->
```

It's better, however, to take the text nodes between tags into account than to sacrifice visibility and code indentation.

Browse Child Nodes

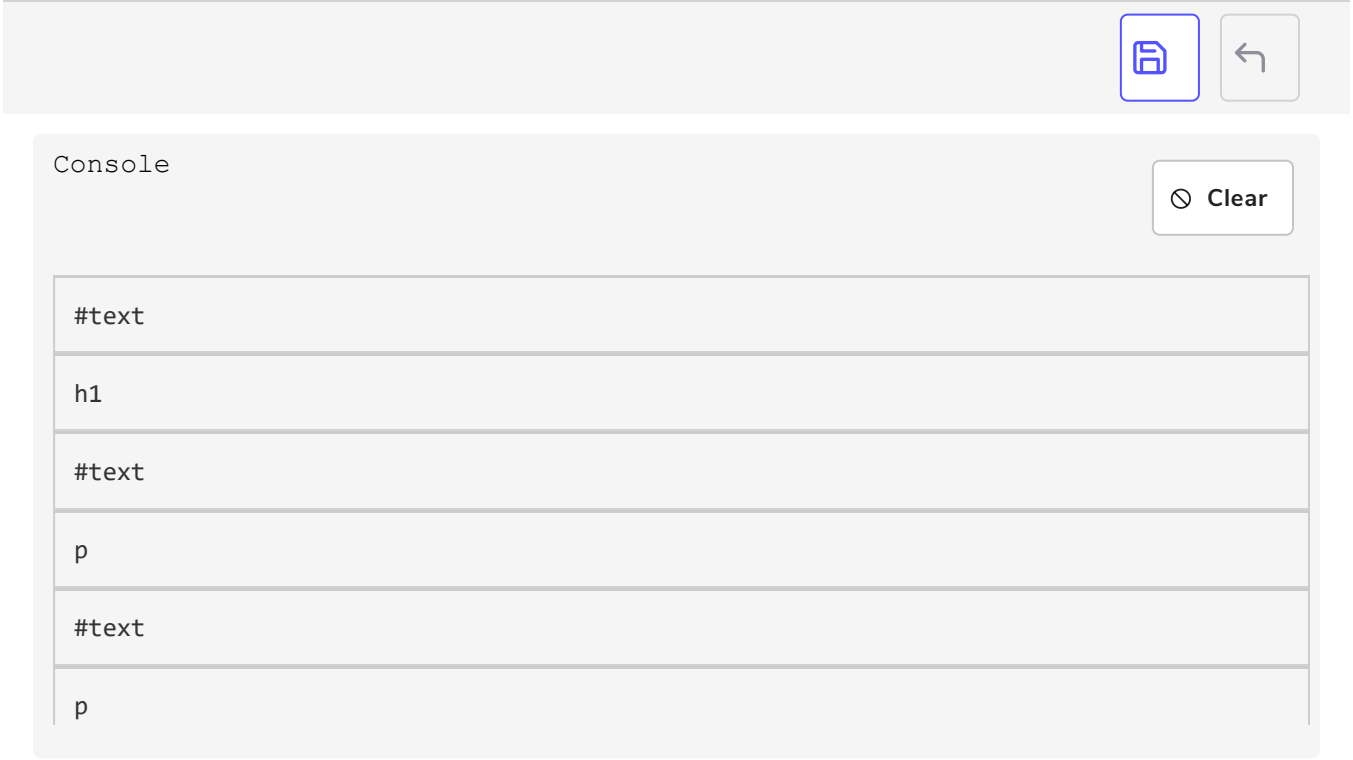
To browse a list of child nodes, you can use a classical `for` loop, the `forEach()` method or the newer `for-of` loop as seen below:

JavaScript

```
// Browse the body node's children using a for loop@
for (let i = 0; i < document.body.childNodes.length; i++) {
  console.log(document.body.childNodes[i]);
}

// Browse the body node's children using the forEach() method
document.body.childNodes.forEach(node => {
  console.log(node);
});

// Browse the body node's children using a for-of loop
for (const node of document.body.childNodes) {
  console.log(node);
}
```



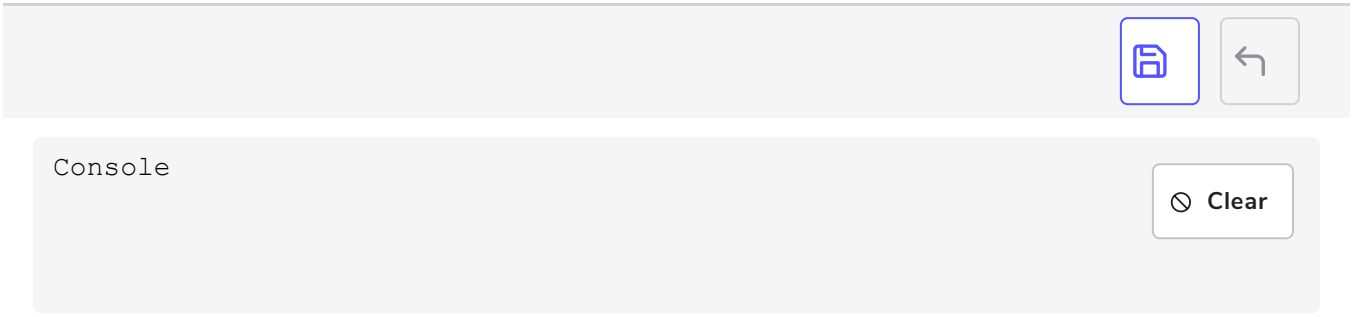
Again, spaces and line returns count as text nodes in the DOM.

Access a Node's Parent

Each DOM object has a property called parentNode that returns its parent node as a DOM object. For the `DOM` root node (`document`), the value of `parentNode` is null since it has no parent node.

JavaScript

```
const h1 = document.body.childNodes[1];
console.log(h1.parentNode);       // Show the body node
console.log(document.parentNode); // Will show null, since body has no parent node
```



There are other properties that we will not discuss here that let you navigate through the DOM, like `firstChild`, `lastChild` or `nextSibling`.