Selecting Elements

This lesson will you how to select elements in JavaScript by index, class or HTML Tag.

The Limits of Node-by-Node Traversal

In the previous chapter, you saw how to navigate the DOM node structure of a web page beginning with the root node and using the `childNodes` property to move down levels in the structure of the page.

Suppose you want to select the title `"Wonders from Antiquity"` of our web page. Taking into account the text nodes between elements, this node is the second child node of the sixth child node of the body element. So you could write something like this.

Output

JavaScript

HTML

```javascript
// Show the "Wonders from Antiquity" h2 element
console.log(document.body.childNodes[5].childNodes[1].innerHTML);
```

This technique is pretty awkward and error-prone. The code is difficult to read and must be updated if new elements are further inserted in the web page. Fortunately, there are much better solutions.

Selecting Items According to HTML Tag

All DOM elements have a method called `getElementsByTagName()`. This returns, under the form of a NodeList object, a list of items that have the name of the tag that's passed as a parameter. The search happens through all the sub-elements of the node on which the method is called – not only its direct children.

With the `getElementsByTagName()` method, selecting the first `h2` element becomes super easy:

Output
JavaScript
HTML

```javascript
// Get all h2 elements into an array
const titleElements = document.getElementsByTagName("h2");

console.log(titleElements[0].innerHTML);    // Show the first h2
console.log(titleElements.length); // 3 (total number of h2 elements in the page)
```

Suffixing JavaScript variables associated to DOM element nodes with Element (or Elements when the variable contains several nodes) is a popular naming convention. We'll stick to it throughout this book.

Selecting Items According to Class

DOM elements also feature a method called `getElementsByClassName()`. This method returns a NodeList object of elements with the class name as a parameter. Again, the search covers all sub-elements of the node on which the method is called.

It's important to note that `NodeList` objects are not real JavaScript arrays, so not all array operations are applicable to them. To turn a `NodeList` object into an array, use the `Array.from()` method. To select and display all document elements with a class `"wonders"`, you can write the following code.

Output
JavaScript
HTML

```javascript
// Show all elements that have the class "exists"
const existingElements = Array.from(document.getElementsByClassName("exists"));
existingElements.forEach(element => {
  console.log(element.innerHTML);
});
```

Console   ⊘ Clear

Selecting an Item According to its ID

Lastly, the `document` variable provides a method called `getElementById()` that returns the element with the specified ID among all elements of the document. It returns `null` if no associated element can be found. The following code selects and displays the list with ID `"new"`.

Output
JavaScript
HTML

```
// Show element with the ID "new"
console.log(document.getElementById("new").innerHTML);
```

Console   ⊘ Clear

Beware: contrary to others, the `getElementById()` method does not contain any `'s'` after the word `"Element"`.