Summary DOM access methods

Hide tutorial navigation

What methods can we use?

We showed how you can use the following methods to find an element or elements in your webpage:

- `document.getElementById(id)`
- `document.getElementsByClassName(className)`
- `document.getElementsByTagName(tagName)`
- `document.querySelector(cssSelector)`
- `document.querySelectorAll(cssSelector)`

What do they return?

There are two methods that return a single `Element` object, `getElementById` and `querySelector`:

```
var salsMotto = document.getElementById("salsMotto");
salsMotto.innerHTML = "Math is cool";
```

The methods `getElementsByClassName` and `getElementsByTagName` return an `HTMLCollection` object that acts like an array. That collection is "live", which means the collection is updated if additional elements with tag name or class name are added to the document.

```
var teamMembers = document.getElementsByClassName("team-member");
for (var i = 0; i < teamMembers.length; i++) {
   console.log(teamMembers[i].innerHTML);
}
```

The method `querySelectorAll()` returns a NodeList, which also acts like an array. That list is "static", which means that the list won't update even if new matching elements are added to the page. Most likely, you won't run into the difference between the two return data types when you're using these methods, but it's good to keep in mind.

```
var teamMembers = document.querySelectorAll(".team-member");
for (var i = 0; i < teamMembers.length; i++) {
   console.log(teamMembers[i].innerHTML);
}
```

Accessing with sub-queries

Once you've found an element, you can do subqueries on it using the methods we've just shown. For example:

```
// find the element with that ID
var salsMotto = document.getElementById("salsMotto");
// find the spans inside that element:
var mottoWords = salsMotto.getElementsByTagName("span");
```

```
// log out how many there are
console.log(mottoWords.length);
```

Traversing the DOM

Another way to access elements is to "traverse" the DOM tree. Each element has properties that point to elements related to it:

- `firstElementChild`
- `lastElementChild`
- `nextElementChild/nextElementSibling`
- `previousElementChild/previousElementSibling`
- `childNodes`
- `childElementCount`

For example:

```
var salsMotto = document.getElementById("salsMotto");
for (var i = 0; i < salsMotto.childNodes.length; i++) {
   console.log(salsMotto.childNodes[i]);
}
```

These properties are **not** available on `Text` nodes, only on `Element` nodes. To make sure you can traverse an element, you can check its `nodeType`/`nodeValue` properties. You likely will not need or want to use DOM traversal, but it is another option available to you.