Lesson Weekend

Introduction to Programming (/introduction-to-programming) / Arrays and Looping (/introduction-to-programming/arrays-and-looping) / Document Query Methods that Return Collections

Text Cheat sheet

`document.querySelectorAll()`

Just like `document.querySelector()`, we can input any valid CSS selector into `document.querySelectorAll()` (https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelectorAll) to get HTML element objects returned to us.

```
> document.querySelectorAll("span");
NodeList(9) [span#person1a, span#person2a, span#animal, spa
n#exclamation, span#person1b, span#verb, span#person1c, spa
n#person2b, span#noun]
```

```
> document.querySelectorAll("span:nth-child(4)");
NodeList [span#exclamation]
```

If we want to get one item from the `NodeList` collection, we'll use bracket notation, passing in the index of the element we want (starting from 0). Just like with arrays, if we pass in 0, we'll get the first element returned:

```
> document.querySelectorAll("span:nth-child(odd)")[0];
<span id="person1a">_________</span>
```

`document.getElementsByClassName()`

The `document.getElementsByClassName()` (https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementsByClassName) method gets all elements that have the same value for their `class` attribute. Since the Mad Libs project doesn't use any classes, the return is an empty collection.

```
> document.getElementsByClassName('x');
HTMLCollection []length: 0[[Prototype]]: HTMLCollection
```

If we want to get an element from the `HTMLCollection`, we'll also use bracket notation and pass in the index (starting at 0) of the element that we want to get.

```
> document.getElementsByClassName('x')[0];
undefined
```

Since our collection is empty, we get `undefined` returned to us.

`document.getElementsByTagName()`

The `document.getElementsByTagName()` (https://developer.mozilla.org/en-US/docs/Web/API/Element/getElementsByTagName) method gets all elements by their tag name. The same tag name that's returned from the `Element.tagName` (https://developer.mozilla.org/en-US/docs/Web/API/Element/tagName) property.

When we use `document.getElementsByTagName()`, we don't have to capitalize the tag name, even though that's how tag names are returned to us from accessing the `Element.tagName` property.

```
> document.getElementsByTagName("h1");
HTMLCollection(2) [h1, h1]
```

To get a single element from the collection, we also use bracket notation. Here, we're getting the second element:

```
> const secondH1 = document.getElementsByTagName("h1")[1];
> secondH1;
<h1>A fantastical adventure</h1>
> Object.prototype.toString.call(secondH1);
"[object HTMLHeadingElement]"
```

`document.getElementsByName()`

The method `document.getElementsByName()` (https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementsByName) will get all elements that have the same value for their `name` attribute. Here's an example:

```
> document.getElementsByName("person1Input");
NodeList [input#person1Input]
```

This method would likely be more useful for radio buttons or checkboxes (as we'll learn), since all inputs of those types must share the same `name` attribute for them to function properly.

Just like in previous examples, if we want to get a single element from the list, we'll use bracket notation.

```
> document.getElementsByName("person1Input")[0];
<input id="person1Input" type="text" name="person1Input">
```

Objects that Look and Act like Arrays

Objects can be structured to look and act like an array. An array-like object:

- Has a `length` property.
- Has properties indexed from zero.
- Do not have access to JavaScript Array methods.
- Uses bracket notation (passing in an index number) to access individual properties.
- Have names; since they are an object, they can be named to be a specific type of object.
- We're working with two array-like objects called `NodeList` and `HTMLCollection`, both of which are Web APIs.

These exist in Web APIs as well as JS proper, and we'll encounter more of these. At this time, we don't need to understand why these exist, or anything deeper about how they are set up. We just need to know how to use them!

Turning `NodeList` and `HTMLCollection` Objects into Arrays

```
> const headingCollection = document.getElementsByTagName
("h1");
> headingCollection;
HTMLCollection(2) [h1, h1]
> const headingArray = Array.from(headingCollection);
> headingArray;
(2) [h1, h1]
> Object.prototype.toString.call(headingArray);
'[object Array]'
```

Notably the `Array.from()` method is called on the array object type. We know it's the array object type, because `Array` is capitalized and we don't use `prototype` in the method's name. We'll revisit this type of method in an upcoming lesson.

Take note that there are some limitations for using `Array.from()` in older browsers, but that's true for a lot of JavaScript! As always, if you run into any issues, visit documentation like MDN.

MDN Documentation Links

The `NodeList` and `HTMLCollection` objects are two of the many objects that make up the DOM. We won't explore them in depth like we did in the last course section with `Element`, `HTMLElement`, and other objects that also are a part of the DOM.

Here are direct links to the objects and methods we learned about in this lesson:

- `NodeList` (https://developer.mozilla.org/en-US/docs/Web/API/NodeList)
- `HTMLCollection` (https://developer.mozilla.org/en-US/docs/Web/API/HTMLCollection)

- `document` object methods (https://developer.mozilla.org/en-US/docs/Web/API/document)
- `Array.from()` (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/from)
- Reference for CSS Selectors (https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors))

Previous (/introduction-to-programming/arrays-and-looping/comparing-and-cloning-arrays)
Next (/introduction-to-programming/arrays-and-looping/adding-and-removing-html-elements)

Lesson 9 of 50
Last updated February 28, 2023

disable dark mode

(http://www.epicodus.com)