Week 03

Thursday, May 6, 2021 10:15 AM

This

Actions are represented in JavaScript by functions in properties

From <https://javascript.info/object-methods>

To access the object, a method can use the `this` keyword.
'this' represents the object

The value of `this` is defined at run-time.

- When a function is declared, it may use `this`, but that `this` has no value until the function is called.
- A function can be copied between objects.
- When a function is called in the "method" syntax: `object.method()`, the value of `this` during the call is `object`.

Please note that arrow functions are special: they have no `this`. When `this` is accessed inside an arrow function, it is taken from outside.

From <https://javascript.info/object-methods>

Objects

An object in JavaScript is a self-contained set of related values and functions. They act as a collection of named properties that map to any JavaScript value such as strings, numbers, booleans, arrays and functions. If a property's value is a function, it is known as a method.

All objects are mutable at any time when a program is running. This means that its properties and methods can be changed or removed, and new properties and methods can be added to the object, even if it was declared using const

Objects Literals are declared

const object_name = {};

Computed Properties
This means that JavaScript code can be placed inside square brackets and the property key will be the return value of that code

Object.key() returns an array with all the keys of an object
Object.value() returns an array with all the values of an object
Object.entries() returns an array with all the key and values of an object

The **parse()** method takes a string of data in JSON format and returns a JavaScript object

The **stringify()** method does the opposite, taking a JavaScript object and returning a string of JSON data, as can be seen in the example:

A regular expression (or RegExp, for short) is a pattern that can be used to search strings for matches to the pattern.

The Document Object Model

The Document Object Model, or DOM for short, represents an HTML document as a network of connected nodes that form a tree-like structure.
The DOM treats everything as a node.

You can check the type of a node with nodeType

```
const body = document.body;
body.nodeType;
<< 1
```

Code	Type
1	Element
2	Attribute
3	Text
8	Comment
9	Body

You can also find the name

```
body.nodeName;
 << "BODY"
```

When you get elements by id you are referencing that element, you have access to several functions, like appendChild.
This is because the idea of id's is that there should be unique.

You don't have for example the appendChild property on getElementsByClassName, this is because you are returning a live node list of all elements, which is not the same as referencing the element as with getElementById.

Query Selectors
The document.querySelector() method allows you to use CSS notation to find the first element in the document that matches a CSS selector provided as an argument. If no elements match, it will return null.

The document.querySelectorAll() method also uses CSS notation but returns a node list of all the elements in the document that match the CSS query selector. If no elements match, it will return an empty node list.

They are also very powerful because the will return the reference to the element as the getElementById does, but this time you can use any CSS property syntax to identify it.

There are also some properties like removeChild and replaceChild similar to what appendChild does

Updating CSS

You need to select the node and then use the . operator to access 'styles'. Inside there you can start

making modifications.
Any CSS property names that are separated by dashes must be written in camelCase notation, so the dash is removed and the next letter is capitalized because dashes are not legal characters in property names.

```
superman.style.backgroundColor = 'blue';     //superman is a node
<< "blue"
```

Or you can also use the bracket notation to select the CSS property

```
superman.style['background color'] = 'blue';    //bracket notaion used here
<< "blue"
```

Disappearing Act
One particularly useful CSS property often employed is the display property. This can be used to make elements disappear and reappear on the page as needed:

```
superman.style.display = 'none';
<< "none"
```

The element can be made to reappear by changing the display property back to block:

```
superman.style.display = 'block';
<< "block"
```

Events

Events are another part of the DOM and they are what provides the link between the web page and user interactions.

You can place event listeners on your nodes that will fire the code to do something, an example would be:

```
document.body.addEventListener("click", doSomething);
```

The recommended way of dealing with events, and the current standard, is to use event listeners.
The addEventListener() method is called on a node object, the node to which the event listener is being applied. For example, this code will attach an event listener to the document's body:

```
document.body.addEventListener('click',doSomething);
```

Whenever an event handler is triggered by an event, the callback function is called. This function is automatically passed an event object as a parameter that contains information about the event.

Website with a list of all types of events
https://developer.mozilla.org/en-US/docs/Web/Events