JavaScript HTML DOM

Notes by rajeeshvengalapurath@gmail.com

Intro

When a web page is loaded, the browser creates a Document Object Model of the page. In the DOM, all HTML elements are defined as objects
JavaScript can:

- Change all the HTML elements in the page
- Change all the HTML attributes in the page
- Change all the CSS styles in the page
- Remove existing HTML elements and attributes
- Add new HTML elements and attributes
- React to all existing HTML events in the page
- Create new HTML events in the page

Example

```
document.getElementById("demo").innerHTML = "Hello World!";
```

`getElementById` `is a method,` **`innerHTML`** `is a property`

getElementById Method

Return type of getElementById is an object

```
document.getElementById("idOfApElement") //is [object HTMLParagraphElement]
```

innerHTML Property

Useful for getting or replacing the content of HTML elements. Type is string

document object

The HTML DOM document object is the owner of all other objects representing the web page.

Finding HTML Elements

```
var myElement = document.getElementById("intro");
var x = document.getElementsByTagName("p");
var x = document.getElementsByClassName("intro");
var x = document.querySelectorAll("p.intro");
```

Finds all elements under a <form> with id=”frm1” and display element values

```
<body>
	<form id="frm1" action="/action_page.php">
		First name: <input type="text" name="fname" value="Donald"><br>
		Last name: <input type="text" name="lname" value="Duck"><br><br>
		<input type="submit" value="Submit">
	</form>
	<button onclick="myFunction()">Try it</button>
	<p id="demo"></p>

	<script>
		function myFunction() {
			var x = document.forms["frm1"];
			var text = "";
			var i;
			for (i = 0; i < x.length ;i++) {
				text += x.elements[i].value + "<br>";
			}
			document.getElementById("demo").innerHTML = text;
		}
	</script>
</body>
```

Object Collections

```
document.forms
document.anchors
document.body
document.documentElement
document.embeds
document.forms
document.head
document.images
document.links
document.scripts
document.title
```

Changing HTML

Changing the HTML Output Stream

```
<body>
	<div>A Div</div>
	<script>
		document.write("<h1>An H1 Element</h1>");
	</script>
</body>
```

Output:
A Div

An H1 Element

Changing HTML Content

```
document.getElementById("p1").innerHTML = "New text!";
```

Changing the Value of an Attribute

```
<img id="myImage" src="smiley.gif">
<script>
	document.getElementById("myImage").src = "landscape.jpg";
</script>
```

Changing CSS

```
document.getElementById("p2").style.color = "blue";
```

Animation

```
<div id="mydiv">A Div</div>
<script>
	var div = document.getElementById("mydiv")
	var id = setInterval(() => {
		if(div.style.color=="red")
			div.style.color = "black"
		else
			div.style.color = "red"
	}, 1000); /*1000 milliseconds*/
</script>
```

To stop animation use `clearInterval(id);`

Events

Examples of HTML events:

- When a user clicks the mouse
- When a web page has loaded
- When an image has been loaded
- When the mouse moves over an element
- When an input field is changed
- When an HTML form is submitted
- When a user strokes a key

Example

```
1. <h1 onclick="this.innerHTML = 'Ooops!'">Click on this text!</h1>
2.
<h1 onclick="changeText(this)">Click on this text!</h1>
```

```
<script>
	function changeText(id) {
		id.innerHTML = "Ooops!";
	}
</script>
```

Assign Events Using the HTML DOM

```
<div id="mydiv">A Div</div>
<script>
	document.getElementById("mydiv").onmousemove = exec;
	function exec(){
		document.getElementById("mydiv").innerHTML = "<b>Welcome</b>"
	}
</script>
```

onload and onunload Events

Triggered when the user enters or leaves the page.

onload

Can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

Deal with Cookies

The onload and onunload events can be used to deal with cookies

onchange Event

Often used in combination with validation of input fields

```
<input type="text" id="fname" onchange="upperCase()">
```

addEventListener() method

```
document.getElementById("myBtn").addEventListener("click", displayDate);
element.addEventListener("click", function(){ alert("Hello World!"); });
```

Add Many Event Handlers to the Same Element

```
element.addEventListener("click", myFunction);
element.addEventListener("click", mySecondFunction);
```

Can add events of different types to the same element

```
element.addEventListener("mouseover", myFunction);
element.addEventListener("click", mySecondFunction);
```

Add an Event Handler to the window Object

Add an event listener that fires when a user resizes the window:

```
window.addEventListener("resize", function(){
       document.getElementById("demo").innerHTML = sometext;
});
```

Passing Parameters

When passing parameter values, use an "anonymous function" that calls the specified function with the parameters:

```
element.addEventListener("click", function(){ myFunction(p1, p2); });
```

Event Bubbling, Event Capturing

Bubbling

The innermost element's event is handled first and then the outer: the <p> element's click event is handled first, then the element's click event.

Capturing

The outermost element's event is handled first and then the inner: the element's click event will be handled first, then the <p> element's click event.

```
addEventListener(event, function, useCapture);

document.getElementById("myP").addEventListener("click", myFunction, true);
document.getElementById("myDiv").addEventListener("click", myFunction, true);
```

removeEventListener

```
element.removeEventListener("mousemove", myFunction);
```

DOM Navigation

DOM Nodes

- The entire document is a document node
- Every HTML element is an element node
- The text inside HTML elements are text nodes
- Every HTML attribute is an attribute node (deprecated)
- All comments are comment nodes

Node Relationships

- In a node tree, the top node is called the root (or root node)

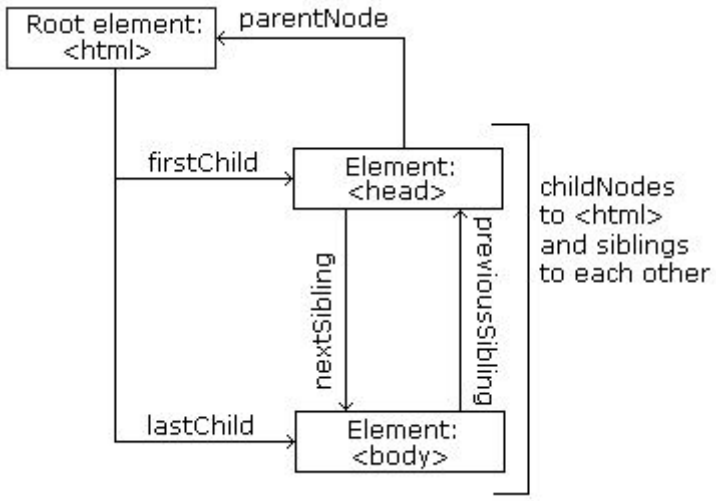
- Every node has exactly one parent, except the root (which has no parent)
- A node can have a number of children
- Siblings (brothers or sisters) are nodes with the same parent

```
<html>

  <head>
    <title>DOM Tutorial</title>
  </head>

  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>

</html>
```



Navigating Between Nodes

Properties to navigate between nodes:

- `parentNode`
- `childNodes[nodenumber]`
- `firstChild`
- `lastChild`
- `nextSibling`
- `previousSibling`

Child Nodes and Node Values

The value of the text node can be accessed by the node's innerHTML property

```
var myTitle = document.getElementById("demo").innerHTML;
var myTitle = document.getElementById("demo").firstChild.nodeValue;
var myTitle = document.getElementById("demo").childNodes[0].nodeValue;
```

DOM Root Nodes

There are two special properties that allow access to the full document:

```
document.body - The body of the document
document.documentElement - The full document
```

nodeName Property

The nodeName property specifies the name of a node.

- nodeName is read-only
- nodeName of an element node is the same as the tag name
- nodeName of an attribute node is the attribute name
- nodeName of a text node is always #text
- nodeName of the document node is always #document

```
document.getElementById("id01").nodeName;
```

nodeValue Property

- nodeValue for element nodes is null
- nodeValue for text nodes is the text itself
- nodeValue for attribute nodes is the attribute value

```
<input type="text" value="test" id="i">
<script>
       alert(document.getElementById("i").attributes[0].nodeValue) /*text*/
       alert(document.getElementById("i").attributes[1].nodeValue) /*test*/
</script>
```

nodeType Property

```
document.getElementById("id01").nodeType;
```

The most important nodeType properties are:

Node	Type	Example
ELEMENT_NODE	1	<h1 class="heading">W3Schools</h1>
ATTRIBUTE_NODE	2	class = "heading" (deprecated)
TEXT_NODE	3	W3Schools
COMMENT_NODE	8	<!-- This is a comment -->
DOCUMENT_NODE	9	The HTML document itself (the parent of <html>)
DOCUMENT_TYPE_NODE	10	<!Doctype html>

Nodes (DOM Elements)

Creating

appendChild()

Appended the new element as the last child of the parent

```
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);

var element = document.getElementById("div1"); /*Existing*/
element.appendChild(para);
```

insertBefore()

```
var para = document.createElement("p");
var element = document.getElementById("div1");
var child = document.getElementById("p1");
element.insertBefore(para, child);
```

Removing Existing HTML Elements

```
var elmnt = document.getElementById("p1");
elmnt.remove();
```

For browsers that does not support the remove() method, you have to find the parent node to remove an element

```
parent.removeChild(child);
```

Replacing HTML Elements

```
parent.replaceChild(para, child);
```

Collections

HTMLCollection Object

```
var x = document.getElementsByTagName("p");
y = x[1];
x.length;

for (i = 0; i < myCollection.length; i++) {
       myCollection[i].style.color = "red";
}
```

NodeList Object

A NodeList object is almost the same as an HTMLCollection object.

```
var myNodeList = document.querySelectorAll("p");
y = myNodeList[1];
for (i = 0; i < myNodelist.length; i++) {
       myNodelist[i].style.color = "red";
}
```

HTMLCollection vs NodeList

HTMLCollection	**NodeList**
Collection of HTML elements	Collection of document nodes
Can be accessed by their name, id, or index number	Can only be accessed by their index number
N/A	Can contain attribute nodes and text nodes
N/A	A node list is not an array