JavaScript and DOM – Class notes - *Accessing and Manipulating DOM*

To manipulate and work with HTML using JavaScript we need to be able to

1. Access or get reference to HTML elements (form elements, div, paragraphs etc)
2. Access or read the values of the elements we have references to
3. Manipulate the values and properties / attributes of the elements
4. Create new elements and add them to the DOM Tree
5. Remove elements from the DOM tree

Let’s look at how we can do each of the steps listed above using JavaScript

# Access or get reference to HTML elements (form elements, div, paragraphs etc)

We can access HTML elements using one of the following functions of the document object

1. **document.getElementById(id)** : finds and returns a reference to an element by the value of the ID attribute of the html element

|  |
| --- |
| HTML |
| <div **id = “div1”**> |
| JavaScript : *to access the div1 div element we have this in javaScript* |
| * 1. var div1 = document.getElementById(“div1”); |

1. **document.getElementsByName(name)** : Finds elements by name and returns a NodeList object. NodeList object is a collection, which behaves like an Array. This function can be used to get child elements of a certain name inside other parent elements.

|  |
| --- |
| HTML |
| <div id = “div1” **name = “dv2”** > |
| JavaScript |
| var divs = document.getElementsByName("dv2")[0] |

1. **document.getElementsByTagName(tag name)** : Finds elements by tag name and returns a NodeList object. NodeList object is a collection, which behaves like an Array. This function can be used to get child elements of a certain tag inside other parent elements.

|  |
| --- |
| HTML |
| <**div** id = “div1” > |
| JavaScript : |
| var divs = document.getElementsByTagName("div")[0] |

1. **document.getElementsByClassName (name)** **:** Finds elements by class name and returns a NodeList object. NodeList object is a collection, which behaves like an Array. This function can be used to get child elements of a certain class inside other parent elements. Multiple class names can be used to by separating them by space when passing them to the function.

|  |
| --- |
| HTML |
| <div id = “div1” **class=”test1”**> |
| JavaScript : |
| var divs = document.getElementsByClassName("test1")[0] (using single class name test1)  document.getElementsByClassName("test1 test2")[0] (using multiple class name test1 and test |

1. **var x = document.querySelectorAll(“css query”):** Finds all the html elements that match the specified css selector. This function also returns a NodeList collection object.

|  |
| --- |
| HTML |
| <div id = “d1” class = “test1”>  <div id = “d1” class = “test2”> |
| JavaScript : *to get a reference to only the div tags with the test1 class name* |
| var x = document.querySelectorAll("div.test1"); |

1. **Using HTML object collections:** In addition to the functions in the document object that are mentioned above, the document, object also has a number of collection that hold reference to certain type of elements. Accessing the elements using these collection is the same for all the collections.

* [document.anchors](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_anchors)
* [document.body](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_body)
* [document.documentElement](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_element)
* [document.embeds](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_embeds)
* [document.forms](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_forms)
* [document.head](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_head)
* [document.images](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_images)
* [document.links](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_links)
* [document.scripts](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_scripts)
* [document.title](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_title)
* Document.forms

To get all the forms in the document

**var forms = document.forms;**

To get the first form in the document

**var forms = document.forms[0];**

Note: The code above shows how to get the forms in the document using the forms collection of the document object. To other collections are used the same way.

Create a registration form in your HTML page with text boxes for first name and last name, a drop down / select box for color preferences (with a few options to choose from), a radio button with three options, a checkbox for consent, a clear form button and a submit button. When the form is submitted, call a validation function that will get references to the form, all the elements inside the form and show the references using console.log. You don’t need to show the values of the form elements yet. Make sure the form is not allowed to be submitted from your JavaScript code (hint: return false from your function).

# Access or read the **values** of the elements we have references to

In DOM the HTML elements have method / functions and properties that can be used to get the values of the elements. For a complete list of element properties and methods, visit

<https://www.w3schools.com/jsref/dom_obj_all.asp>

The examples below are to demonstrate how to get values from form elements and non form elements in HTML

1. **Getting values from form elements using .value shortcut**
   1. Almost all form elements have the value property that will allow JavaScript get the values that the user enters in the form element using a the .value shortcut
      1. Text box
      2. Text area
      3. Button
      4. Select box
      5. Radio button
      6. Check box

|  |
| --- |
| HTML |
| <input type = “Text” id = “txt1” /> |
| JavaScript |
| var txt1 = document.getElementById(“txt1”); *(in javaScript)*  var myVal = txt1.**value;** |

The code above shows how to get the value from a text box using the .value shortcut. The same can be done with the other form elements. **Note** that to get the selected values from radio button, select box and checkbox we will need to check which options has been checked/selected.

1. **Getting values of attributes from HTML elements (getAttribute() and getAttributeNode() functions):**
2. getAttribute(“attribute Name”):

To get the value of an attribute in an HTML element, you can use *element*.getAttribute(). This function gives you the value of an attribute that is in the element. Note that you will need to get a reference to the html element first and then call the getAttribute() function on the element.

|  |
| --- |
| HTML |
| <div id = “abc” name = “div1”> |
| JavaScript |
| var div = document.getElementById(“abc”);  var divName = div.getAttribute(“name”); |

1. getAttributeNode(“attribute Name”)

The getAttributeNode returns the attribute node that meets the attribute name passed into the function. Instead of returning the value of the attribute, it returns the attribute node that contains the value of the attribute in the node

|  |
| --- |
| HTML |
| <div id = “abc” name = “div1”> |
| JavaScript |
| var div = document.getElementById(“abc”);  var divName = div.getAttributeNode(“name”);  \*\* at this point the divName is not the value of the name attribute, to get the value of the name attribute we need to do the following  var divNameValue = divName.value; |

1. **The .id shortcut**

With almost all html elements, you can use the .id shortcut to get the id of an element.

|  |
| --- |
| HTML |
| <div id = “abc” name = “div1”> |
| JavaScript : |
| var div = document.getElementById(“abc”);  console.log(div.id); |

1. **Getting the value between the opening tag and closing tag (innerHTML)**

The .value shortcut as well as the attribute(“value”) option to get the content of the value attribute of the html element. In order to get the content that is between the opening and closing tag of the html element you can use the innerHTML property of the element.

|  |
| --- |
| HTML |
| <div id = “abc” name = “div1”> This is div 1 </div |
| JavaScript : |
| var div = document.getElementById(“abc”);  console.log(div.innerHTML); |

Exercise:

Now that you have access to the form elements, get the values that have been entered into those forms element and display the values using console.log.

# Set the **values** of the elements we have references to

Setting the values of attributes can be done by calling the element.setAttribute(“attributename”, “value”) function

|  |
| --- |
| HTML |
| <div id = “abc” name = “div1”> This is div 1 </div> |
| JavaScript : |
| var div = document.getElementById(“abc”);  div.setAttribute(“name”,”divTest”); This will change the value of the name attribute to “divTest” |

Setting the value of an HTML element using the .value shortcut can be done using the following code

*element*.value = “new value”;

using the .innerHTML property you can change the content between the opening and closing tag.

*element*.innerHTML = “new value”;

Exercise

For each of the form element, set some default values before the page load using your javaScript code

# Creating and adding new element to the dom tree

To create a an HTML element, you can use a the document.createElement() or the document.createTextNode() function

**document.createElement(ELEMENT NAME):** Lets you create an element node based on the element name passed into the function. Once created, the element will need to be added to the DOM tree as well. This can be done using appendChild() function of the element node. Note: Remember that the appendChild() function is not a function of the document object but the element object

|  |
| --- |
| HTML |
| <div id = “abc” name = “div1”> This is div 1 </div> |
| JavaScript : |
| var d1 = document.createElement(“DIV”);  Note: at this point, the d1 variable is a reference to a <div> element. But it will not show up on the html page as it has not been added to the dom tree. To add it to the dom tree, we will need to add it as a “Child” element of an existing element. To do this, we can  If I want to add the new div to the existing div in HTML, I will need to get a reference to the existing div first  var div1 = document.getElementById(“div1”);  now to add the new div  div1.appendChild(div1);  Now the new div has been appended as a child node to the existing div element in html.  **Note: any attributes we want to set in the new div tag should be done before we append it to the DOM tree** |

**document.createTextNode(“text”):** is used to create elements that are textual. For example to create the text that needs to go between the opening and closing tag of your new paragraph element. After the text node is created, it need to be appended to an existing element in the DOM tree or to a new element being created in JavaScript before it is added to the DOM tree.

|  |
| --- |
| HTML |
| <div id = “abc” name = “div1”> This is div 1 </div> |
| JavaScript : |
| var divText = document.createTextNode(“More text for div1”);  var div1 = document.getElementById(“div1”);  div1.appendChild(divText); |

Cloning a node and the tree :

Using element.cloneNode(true/false) you can clone an existing node. If you pass in “true” javascript will clone the node and the underlying children. If you pass in false it will only copy the node. This cloned node can then be added to the document.

Exercise:

1. Add a hyperlink on your html page that will call a javascript function when clicked. This javascript function will add a paragraph with a unique ID and name as well as a unique text the html page as a child node to an existing element on the page.
2. Using cloneNode, clone an element in your html document that has children nodes and add the clone to the html document

# remove or replace element and attribute from the dom tree

To remove an element from the dom tree we can call the element.removeChild() function. Note that this is a function of the element and not the document object.

We will need to use the removeChild() function in conjunction with the element.childNodes property, element.firstChild property and element.hasChildNodes() function

* element.childNodes: provides an array with the list of child nodes in it.
* element.firstChild: returns a reference to the first child of the parent element. This will keep changing as we are removing the first child.
* Element.hasChildNodes() function returns true if the childNodes array still contains any child nodes.

|  |
| --- |
| HTML |
| <div id = “abc” name = “div1”> This is div 1  <div id = “innerDiv1> inner div 1 </div>  <div id = “innerDiv2> inner div 2 </div>  </div> |
| JavaScript : |
| I will need to get a reference to the existing parent div first  var div1 = document.getElementById(“div1”);  to remove the first child node  div1.removeChild( div1.childNodes[0] );  this will remove the first child node of the div1 div in html  if I want remove all the child nodes in div1 we will need a while loop  while (div1.hasChildNodes())  {        div1.removeChild(list.firstChild); } |

To replace an element with another element we can use **element.replaceChild(newNode, oldNode)**

To remove attributes from an element you can use ***element*.removeAttribute(“attribute name”);**

# traversing / walking the dom tree

You can use this article as a reference to understand how the DOM tree is structure and the functions available to traverse the tree

<http://javascript.info/tutorial/traversing-dom>

The following properties and functions allow javaScript to traverse the DOM tree and get access to elements in HTML

1. document.documentElement : references the <HTML> root element
2. document.body : references the <body> element
3. childNodes: property of any element in DOM that gives a references to an array with all the child nodes of that element. This includes text nodes as well
4. element.children: provides an array with non-textual nodes
5. firstChild and lastChild : give reference to the first and last child of the element/node.
6. parentNode: references the nodes parentNode.
7. previousSibling and nextSibling: references the previous and next sibling of the current node/element. This properties do not ignore text and comment nodes.
8. previousElementSibling and nextElementSibling: references the previous and next sibling of the current node/element. This properties will ignore text and comment nodes.