Community Post

Learn JavaScript HTML DOM

Divya (/@divyae) December 04, 2017

HTML DOM:

- When ever the browser loads the page, the browser will create a DOM of the web page.
- HTML DOM model is constructed in the form of a tree of Objects.
- With this Object Model the JavaScript becomes more powerful and create dynamic HTML.
- JavaScript can change, remove or add all the HTML elements and attributes in the page.
- It can also change the CSS styles of the web page.
- It can react to all existing HTML events in the web page or it can add new events as well.

DOM Structure Image:
[(https://cdn.scotch.io/55044/psa8GaspSKmCtB7JiF8z_dom_tree.png
(https://cdn.scotch.io/55044/psa8GaspSKmCtB7JiF8z_dom_tree.png))]

DOM means Document Object Model. Its a World Wide Web Consortium (W3C) standard.

DOM definition:

The DOM - W3C is a platform and a kind of interface that allows programs and scripts to dynamically access and update the content, structure, and style of a HTML document.

The DOM standard is of 3 types:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

HTML DOM Means:

HTML DOM is a standard object model and programming interface for HTML.

The HTML DOM is a standard for how to get, change, add, or delete HTML elements of a web page.

HTML DOM Methods:

- In the DOM structure, all HTML elements are in the form as objects.
- The programming interface is the properties and methods of each object.

- **HTML DOM Method is an action that we perfom on HTML Element like add or deleting an HTML element**

- **HTML DOM properties are values of HTML element that we can change or set like changing the content of an HTML Element.**

Example and explanation:

```
<!DOCTYPE html>
<html>
<body>
<h1>My Name is Divya</h2>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = "Hello Divya! Welcome..!!";
</script>
</body>
</html>

Here innerHTML is a property and  getElementById is a method
```

The innerHTML Property means:

The innerHTML property is used for getting or replacing the content of HTML elements. It's the best way to get the content of an element by using this property.

- The innerHTML property can also be used to get or change any HTML elements including html and body.

The getElementById Method means:

The getElementById method uses id="demo" to find the element and of which the content should be changed. The easy and common way to access an HTML element is to use the id of the element.

HTML DOM Document:

The document object represents our web page. If we want to access any element in an HTML page, we should always start with accessing the document object.

Few examples of how we access and manipulate HTML using Document object:

To Find HTML Elements:

- document.getElementsByClassName(name) ------ Finding elements by class name
- document.getElementById(id) ------ Finding an element by element id
- document.getElementsByTagName(name) ------ Finding elements by tag name

To Change HTML Elements:

- element.innerHTML = new html content ------ Change the inner HTML of an element
- element.setAttribute(attribute, value) ------ Change the attribute value of an HTML element
- element.style.property = new style ------ Change the style of an HTML element
- element.attribute = new value ------ Change the attribute value of an HTML element

To Add and Delete Elements:

- document.createElement(element) ------ Create an HTML element
- document.write(text) ------ Write into the HTML output stream
- document.appendChild(element) ------ Add an HTML element
- document.replaceChild(element) ------ Replace an HTML element
- document.removeChild(element) ------ Remove an HTML element

To Add Events Handlers:

- document.getElementById(id).onclick = function(){code} -- Adding event handler code to an onclick event

Few more objects, collections, and properties were added in in HTML DOM Level 1 and Level 3.

Some of them are:

- document.doctype --- Returns the document's doctype
- document.anchors --- Returns all a tag elements that have a name attribute
- document.embeds --- Returns all elements
- document.documentElement --- Returns the html element
- document.body --- Returns the body element
- document.forms --- Returns all form elements
- document.head --- Returns the head element
- document.images --- Returns all img elements
- document.title --- Returns the title element

- document.domain --- Returns the domain name of the document server
- document.baseURI --- Returns the absolute base URI of the document
- document.cookie --- Returns the document cookie
- document.documentURI --- Returns the URI of the document
- document.URL --- Returns the complete URL of the document
- document.domain --- Returns the domain name of the document server
- document.implementation --- Returns the DOM implementation
- document.lastModified --- Returns the date and time the document was updated
- document.strictErrorChecking --- Returns if error checking is enforced

HTML DOM Elements:

The DOM Element is used to find and access HTML elements in an HTML page.

We can find and access HTML elements in the following ways:

Finding HTML elements by id:

```
Syntax or Example:
var myElement = document.getElementById("intro");
Here we will get element with id = intro
```

Finding HTML elements by tag name:

```
Syntax or Example:
var x = document.getElementsByTagName("p");
Here we will find all p tag elements
```

Finding HTML elements by class name:

```
Syntax or Example:
var x = document.getElementsByClassName("demo");
Here we will get element with classname = intro
```

Finding HTML elements by CSS selectors:

```
Syntax or Example:
var x = document.querySelectorAll("p.intro");
Here we will get a list of all p elements with class="intro"
```

Finding HTML elements by HTML object collections:

```
Syntax or Example:
var a = document.forms["frm1"];
var text = "";
var x;
for (x = 0; x < a.length; x++) {
    text += a.elements[x].value + "<br>";
}
document.getElementById("demo").innerHTML = text;

Here we will find the form element with id="frm1", in the forms collection, and displays al
```

HTML DOM - Changing HTML:

The HTML DOM helps us to change the content of HTML elements using JavaScript.

Changing the HTML Output Stream:

- Using JavaScript can create dynamic HTML content.
- We use document.write() to write directly to the HTML output stream.

```
Example:
document.write(Date());
```

Changing HTML Content:

- To modify the content of an HTML element by using the innerHTML property.

```
Syntax:
document.getElementById(id).innerHTML = new HTML
```

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

Changing the Value of an Attribute:

- We can also change the value HTML attribute.

```
Syntax:
document.getElementById(id).attribute = new value

Example:
document.getElementById("myImg").src = "rocks.jpg";
```

HTML DOM - Changing CSS:

Changing HTML Style: We can change the style of an HTML element.

```
Syntax:
document.getElementById(id).style.property = new style

Example:
document.getElementById("p1").style.color = "blue";
```

HTML DOM Events:

Using HTML DOM we will react on HTML events using JavaScript.

Reacting to Events:

- We will execute the JavaScript when an event occur, example when a user clicks on an element.
- So, to execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

```
Syntax:
onclick=JavaScript code which needs to be executed

Example:
<h2 onclick="this.innerHTML = 'Hello Good Morning..!!'">Click on this Content!</h1>
```

We can also Call a function from the event handler.

```
Example:
<h2  onclick="changeContent(this)">Click on this content!</h1>
<script>
function changeContent(id) {
    id.innerHTML = "Hello Good Morning!";
}
</script>
```

Examples of HTML events:

- When an input field is changed
- When an HTML form is submitted
- When a user clicks the mouse on an element
- When a web page has loaded
- When the pointer hovers over an element

HTML Event Attributes Or by using HTML DOM

We can assign events to HTML elements using event attributes or even the HTML DOM allows us to assign events to HTML elements by using JavaScript.

```
Example for event Attribute:
<button onclick="displayDate()">Click Me</button>
Here we are assigning  an onclick event to a button element.

Example by using HTML DOM:
<script>
document.getElementById("btn").onclick = displayDate;
</script>
```

HTML DOM EventListener:

We can attach an event handler to the specified element by using addEventListener() method. Event listener is nothing but that fires when a user clicks a button.

```
Syntax:
element.addEventListener(event, function, useCapture);

Here the first parameter is the type of the event like "click" or "mousehover".

The second parameter is the function which we want to call when the event occurs.

The third parameter is a boolean value specifying whether to use event bubbling or event ca

Example:
document.getElementById("btn").addEventListener("click", displayDate);
```

About addEventListener():

- The addEventListener() method attaches an event handler to an element without overwriting existing event handlers.
- We can add many event handlers to one element.

```
Example:
element.addEventListener("click", function(){ alert("Hello Divya..!!!"); });
```

- We can also add many event handlers of the same element, i.e two "click" events and that too by without overwriting existing events.

```
Example:
element.addEventListener("click", myFirstFunction);
element.addEventListener("click", mySecondFunction);
```

- We add event listeners to any DOM object not only HTML elements. i.e the window object.

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

Event Bubbling and Event Capturing:

Event Bubbling:

In event bubbling the inner most element's event is handled first and then the outer. So, the p element's click event is handled first, then the div element's click event.

Event Capturing:

In event capturing the outer most element's event is handled first and then the inner. So, the div element's click event will be handled first, then the p element's click event.

Removing Event Listener:

The removeEventListener() method removes event handlers that's been attached with the addEventListener() method.

```
Example:
element.removeEventListener("mousehover", myFunc);
```

HTML DOM Navigation:

We can navigate the node tree using node relationships.

Every thing in an HTML document is a node.

- 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

- All comments are comment nodes

Node Relationships:

The nodes in the node tree have a hierarchical relationship to each other.

Node Relationship Image:
(https://cdn.scotch.io/55044/gHOqSS1WSCaXyhb5FBiP_navigate.gif
(https://cdn.scotch.io/55044/gHOqSS1WSCaXyhb5FBiP_navigate.gif))

- In a node tree, the top node is called the root.
- 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

- We can access the innerHTML property using the node value:

```
Example:
var myTitle = document.getElementById("demo").firstChild.nodeValue;
```

DOM Root Nodes:

We have two kinds properties that allow access to the full document:

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

HTML DOM Collections:

The getElementsByTagName() method returns an HTMLCollection object.

An HTMLCollection object is an array-like list (collection) of HTML elements and index starts with 0.

HTML DOM Node Lists:

- A NodeList object is a list (collection) of nodes from a document.
- A NodeList object is almost the same as an HTMLCollection object.
- All browsers return a NodeList object for the property childNodes.
- Most browsers return a NodeList object for the method querySelectorAll()

The Difference Between an HTMLCollection and a NodeList:

- A NodeList and an HTML collection is a collection of document nodes.
- Both are array list of objects.
- Both have a length property defining the number of items in the list or collection.
- Both provide an index starts with 0 to access each item like an array.

But

- HTMLCollection items can be accessed by their name, id, or index number.
- NodeList items can only be accessed by their index number and can contain attribute nodes and text nodes.

(/@divyae)
Divya (/@divyae)

4 posts (/@divyae)

(https://github.com/https://github.com/Divya-Etikala)

Popular In the Community

MONGODB 3.6: JSON SCHEMA VALIDATION...
RedFlower
4d
I take it all the tags in the code are...

BUILD A SCROLL-SPY NAVBAR (SOLUTION TO...
Zammy13
7 May
I like the challenges you're doing. But I'm...

PASSWORD STRENGTH METER IN REACT
CyanCone
8 May
Code is very unfriendly written

WEB SCRAPING SCOTCH: THE NODE WAY
OrangePlane
7h
Python's Scrapy might have been an...

BUILDING A WEATHER APP IN DJANGO
OrangeMagnet
2h
Great tutorial. The only hangup I had w...

ORGANISABLE AND MAINTAINABLE SCSS...
GreenScissors
5 May
Great article! Never thought before that...

8 AWESO FEATURE
Do you informa

💖 A side project brought to you from Las Vegas and DC by... (/about)

(https://scotch.io/@chris)

Chris Sevilleja (https://scotch.io/@chris)

Follow @chrisoncode 8,006 followers

(https://scotch.io/@nick)

Nick Cerminara (https://scotch.io/@nick)

Follow @whatnicktweets 2,354 followers

(https://bit.ly/2tDcLEK)

Easiest Local Dev Environment

(https://scotch.io/courses/getting-started-with-vue)

Get Started with Vue.js

scotch

Top shelf learning. Informative tutorials explaining the code **and the choices behind it all.**

(https://github.com/scotch-scotchdevelopment)

(https://twitter.com/scotchio) (https://facebook.com/scotchio)

FAQ (/faq) Privacy (/privacy) Terms (/terms) Rules (/rules) Hosted by Digital Ocean (https://m.do.co/c/7a59e9361ab7)