Document Object Model

Document Object Model (DOM) is a Web Application Programming Interface (API) that connects Web pages and a programming API for HTML and XML documents. It represents a document with a logical tree, and each branch ends in a node. In DOM, nodes are elements within an HTML document, and each node is an individual branch.

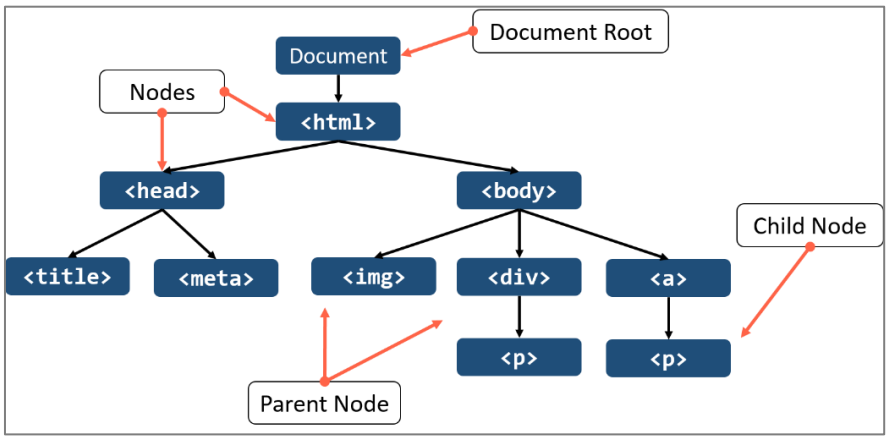

Figure 1. DOM Tree

Document Object

- This represents the HTML document that contains properties and methods. Some functions are used to modify each element or set its value or function.
- The following are the methods or objects used for calling or finding the HTML element:
  - **`getElementById()`**
    - Finding the element by its ID
  - **`getElementByClassName()`**
    - Finding the element by class name
- When we modify an element, there are objects or methods that can be used to modify its content. Below are the common objects and methods that can be used to define an element that will modify its function and how it works:
  - ***`element`*`.innerHTML`**
    - The innerHTML is used to change the HTML content.
  - ***`element`*`.style.`*`property`**
    - This is used to change the style of an HTML element, followed by the property that may be used. It can be a color, border, background color, etc.
- There are also methods that can add or delete an element. These are the following:
  - **`document.createElement()`**
    - This method adds an HTML element.
  - **`document.removeElement()`**
    - This method removes the HTML element.
  - **`document.replaceElement()`**
    - This method replaces the HTML element.

Event

JavaScript events are actions that can be detected only by JavaScript. **Events** are user actions that are initiated or can be generated in the browser. For example, when the user clicks or hovers over an element in a Web page containing an event, the event will be triggered. Some events are common for most of the programming languages or scripts, and this is called **onclick event**. When the user clicks an element like a button, the event will be triggered, and a line of code is executed. In JavaScript, there are two (2) ways or approaches when using an event. It can be an inline event handler or a listener.

Inline Event Handler

- This is one of the common approaches for using an event. The event is declared in the HTML element. It may be considered as an attribute like ID, name, or class.
  - **`<button onclick="functionShowInfo()" >Click this for demo</button>`**
    - In this example, the onclick event is declared in the button element. It means that once the button is clicked, the onclick event that contains a function named **`functionShowInfo()`** will be displayed.

Listener

- The listener event is declared in the JavaScript file. Including an event in the HTML element may be useful, but it may be difficult to maintain the codes. To be able to maintain it and to have a simple markup, an event like the example syntax below may be used:

```
function listenerEvent(){
    alert("Listener Event");
}
var btnClick = document.getElementById('demo');
btnClick.onclick = listenerEvent;
```

JavaScript Events

- **onclick**
  - The mouse is clicked on an HTML element.

```
<button onclick="onClickEvent()">
    Click me!
</button>

<script>
function onClickEvent(){
    alert("Sample onclick");
}
</script>
```

- **onchange**
  - This event is used when the user types something or selects a new choice.

```
<h3>onchange event</h3>
<select id="selectFruit" onchange="onChangeEvent()">
    <option>Strawberry</option>
    <option>Mango</option>
    <option>Banana</option>
    <option>Apple</option>
</select>
<h2 id="showText"></h2>
<script>
    function onChangeEvent(){
        var frt = document.getElementById("selectFruit").
        value;
        document.getElementById("showText").innerHTML = frt;
    }
</script>
```

- **onmouseover**
  - The mouse is moved over an HTML element but not clicked.
- **onmouseout**
  - The mouse moves off of an HTML element.

```
<h3>onmouseover and onmouseout</h3>
<h2 id="demoMouseEvent" onmouseover="onMouseOver()"
onmouseout="onMouseOut()">Mouse: </h2>
<script>
    /////mouse over and out
    var mouseEvent = document.getElementById(
    "demoMouseEvent");
    function onMouseOver(){
        mouseEvent.innerHTML = "Mouse: Over";
        mouseEvent.style.color = "dodgerblue";
    }
    function onMouseOut(){
        mouseEvent.innerHTML = "Mouse: Out";
        mouseEvent.style.color = "tomato";
    }
</script>
```

- **onload**
  - This event shows an action when a document or object has been loaded.

```
<body onload="onClickEvent()">

<script>
function onLoadEvent(){
    alert("Sample onload Event");
}
<script>

</body>
```

- **onkeyup**
  - This event happens last – when the user releases a key that is down.

```
<h3>keyup event</h3>
<input type="text" id="demoKeyup" onkeyup=
"onKeyUpEvent()">
<h2 id="keyUpOutput"></h2>
<script>
    ///keyup event
    function onKeyUpEvent(){
        var getValue = document.getElementById(
        "demoKeyup").value;
        document.getElementById("keyUpOutput").
        innerHTML = getValue.toUpperCase();
    }
</script>
```

- **onkeydown**
  - The function of this event is that when the user is pressing a key, this event happens first.

```
<h3>keydown event</h3>
<input type="text" id="demoKeydown" onkeydown=
"onKeyDownEvent()">
<h2 id="keyDownOutput"></h2>
<script>
    ///keydown event
    function onKeyDownEvent(){
        var getValue = document.getElementById(
        "demoKeydown").value;
        var showValue = document.getElementById(
        "keyDownOutput");

        showValue.innerHTML = getValue.toLowerCase();
        showValue.style.color = "dodgerblue";
    }
</script>
```

- **onkeypress** and **keyCode**
  - The **onkeypress** event happens after **onkeydown** when the user presses a key.
  - The **keyCode** property returns the Unicode character code of the key when it is called in an **onkeypress** event.

```
<h3>keypress event</h3>
<input type="text" id="demoKeypress" onkeypress=
"onKeyPressEvent(event)">
<h2 id="keyPressOutput"></h2>
<script>
    //keypress output
    function onKeyPressEvent(event){
        var getKey = event.keyCode;
        document.getElementById("keyPressOutput").
        innerHTML = getKey;
        if(event.keyCode === 13){
            console.log(document.getElementById(
            "demoKeypress").value);
        }
    }
</script>
```

Key	Code	Key	Code	Key	Code	Key	Code
backspace	8	2	50	o	79	multiply	106
tab	9	3	5	p	80	add	107
enter	13	4	52	q	81	subtract	109
shift	16	5	53	r	82	decimal point	110
ctrl	17	6	54	s	83	divide	111
alt	18	7	55	t	84	semi-colon	186
pause/break	19	8	56	u	85	equal sign	187
caps lock	20	9	57	v	86	comma	188
escape	27	a	65	w	87	dash	189
(space)	32	b	66	x	88	period	190
page up	33	c	67	y	89	forward slash	191
page down	34	d	68	z	90	open bracket	219
end	35	e	69	numpad 0	96	back slash	220
home	36	f	70	numpad 1	97	close bracket	221
left arrow	37	g	71	numpad 2	98	single quote	222
up arrow	38	h	72	numpad 3	99	open bracket	219
right arrow	39	i	73	numpad 4	100		
down arrow	40	j	74	numpad 5	101		
insert	45	k	75	numpad 6	102		
delete	46	l	76	numpad 7	103		
0	48	m	77	numpad 8	104		
1	49	n	78	numpad 9	105		

REFERENCES:

Connolly, R. & Hoar, R. (2015). *Fundamentals of web development*. New Jersey: Pearson Education, Inc.

Lemay, L., Colburn, R., & Kyrnin, J. (2016). *Sams teach yourself HTML, CSS and JavaScript web publishing in one hour a day* (7th Ed.). New Jersey: Pearson Education, Inc.

Krause, J. (2016). *Introducing web development*. California: Apress Media, LLC.