

Browser Environment

Browser Object Model (BOM)

- It is commonly used for talking or interacting with the browser. It manipulates the properties and methods that are connected to the Web browser.
- It has the following common objects which are used in manipulating or interacting with the browser: window, location, history, pop-up message, and timer.

Window

- This object has four (4) common methods to interact in the browser:
 - **open(url, name, windowFeatures)**
 - This method opens another tab or a new window by calling the HTML file.
 - It contains three (3) parameters:
 - **url**
 - This parameter identifies the URL name that will open in a new tab or window.
 - If the URL is not set, only a blank window will open.
 - **name**
 - It specifies the name of the new window, but this is not considered as the title of the new window that is opened.
 - **windowFeatures**
 - It specifies the common features of a new window. An example of this is changing its size and scroll bar.
 - **close()**
 - This method is only used to close the opened new window using the script **window.open()**.

```
<button onclick="openBrowser()">Open</button>
<button onclick="closeBrowser()">Close</button>
<script>
    var newWindow;
    function openBrowser(){
        newWindow=window.open("samplewindow.html", "Test",
        "width=350, height=200");
    }
    function closeBrowser(){
        //close the new window
        newWindow.close();
        //close the current window
        window.close();
    }
</script>
```

- **moveTo(x, y)** and **moveBy(x, y)**
 - These two (2) methods change the default position of the opened new window. It contains two (2) parameters that specify the x and y coordinate in the Web browser.
 - The **moveTo(x, y)** method is used to move the opened new window using the script **window.open()** to the specified coordinate.
 - The **moveBy(x, y)** method is used to move the current window from its current position or coordinate.

- **focus()**

- This method only focuses on the window that is open or the current window.

```
<script>
    var newWindow;
    function openBrowser(){
        newWindow=window.open("", "newWindow", "width=350,
        height=200");
        newWindow.document.write("<p>Hello World</p>");
    }
    function moveToWindow(){
        //move the new window 200px left - 350px top
        newWindow.moveTo(200,350);
        //focuses on the current window
        newWindow.focus();
    }
    function moveByWindow(){
        //move the current window that is relative to its current
        position
        newWindow.moveBy(75,50);
        //focuses on the current window
        newWindow.focus();
    }
</script>
```

Location

- The location, or window.location contains information on the current location of the document.
- **window.location.assign()**
 - This method is used for loading a new document.

```
<h3>window.location.assign()</h3>
<button onclick="funcLocation()">open</button>
<script>
    function funcLocation() {
        window.location.assign("newdocument.html");
    }
</script>
```

- **window.location.reload()**
 - This is used for reloading the current page.

```
<script>
    function funcReload(){
        var reloadPage = confirm("Reload this page?");
        if(reloadPage){
            //location.reload();
            window.location.reload();
            document.write("Reloaded!");
        }else{
            alert("Reload Cancelled!");
        }
    }
</script>
```

History

- The window.history object grants a privilege to access the browser's history.
- There are some limitations when using this object since the security, or the privacy of every user should be considered.

- This object has two (2) methods that are used to go back or forward in the browser:
 - **window.history.back();**
 - This method has a function of going backward; it loads the previous URL.
 - **window.history.forward();**
 - This method loads the next URL from the history of the Web browser.

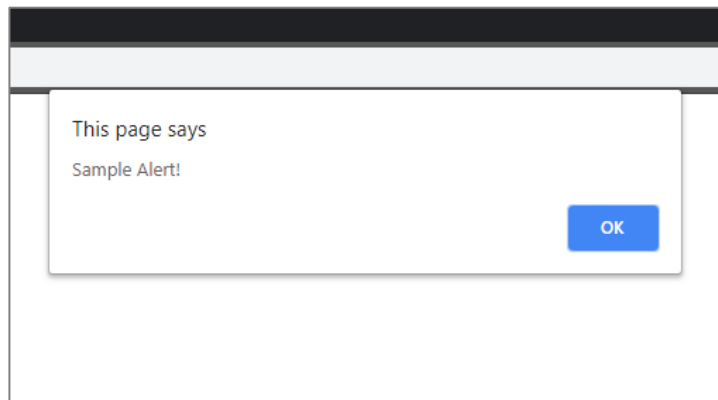
```
<script>
    function funcForward() {
        window.history.forward();
    }
    function funcBackward() {
        window.history.back();
    }
</script>
```

Pop-up Message

- Pop-up message is a common function in an application. It pops up and displays an error message or a confirmation. In JavaScript, it has three (3) functions that are commonly used to deliver a message to the user:

- **alert()**
 - The alert function displays a basic message that only contains one (1) button.

```
<button onclick="funcAlert()">Alert</button>
<script>
    function funcAlert() {
        alert("Sample Alert!");
    }
</script>
```



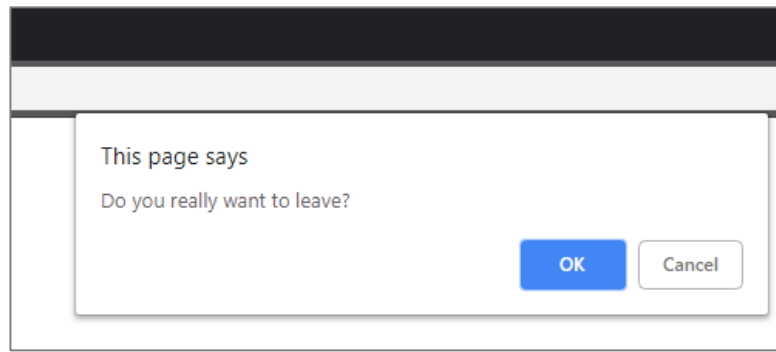
- **confirm()**
 - The confirm function pops up a message that validates if you want to proceed or cancel.

```
<h4 id="message"></h4>
<button onclick="funcConfirm()">Confirm</button>
<script>

    function funcConfirm() {

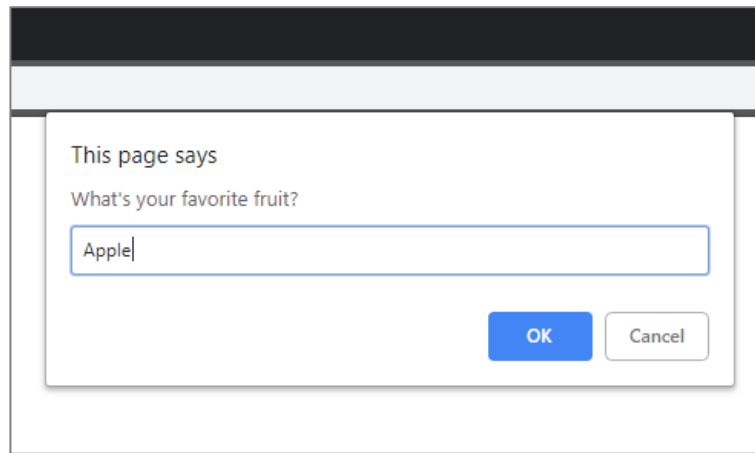
        var sampleConfirm = confirm("Do you really want to leave?");
        var getMessage;
        if(sampleConfirm){
            getMessage = "Thanks for visiting!";
        }else{
            getMessage = "";
        }
        document.getElementById("message").innerHTML = getMessage;
    }

</script>
```



- **prompt()**
 - The prompt function responds to the user's input and validates before entering the page.

```
<script>
    function funcPrompt() {
        var fruit = prompt("What's your favorite fruit?");
        var getMessage;
        if(fruit == null || fruit == ""){
            getMessage = "User cancelled the prompt.";
        }else{
            if(fruit.toLowerCase() == "apple"){
                getMessage = "Apple is my favorite too!";
            }else{
                getMessage = "Not my favorite";
            }
        }
        document.getElementById("message").innerHTML =
        getMessage;
    }
</script>
```



Timer

- In JavaScript, there are (2) methods for setting the time intervals, namely: `setTimeout()` and `setInterval()`.
- **`setTimeout(function, milliseconds)`**
 - This method is used to set the time in milliseconds and evaluates the function after the specified time.
 - There are two (2) parameters inside this method: the function or method to be called and the time that is set in milliseconds. Below is the syntax of how this `setTimeout()` method is used.

```
function timeOut(){  
    alert("Hello Guest!");  
}  
  
setTimeout(timeOut, 3000);
```

- **`setInterval(function, milliseconds)`**
 - This method is the same as the `setTimeout()` method, but what makes this method different is that the `setInterval()` repeats the function continuously.

```
function timeInterval(){  
    setInterval(timeInterval, 1000);  
    var y = new Date();  
  
    document.getElementById("sample").innerHTML =  
    y.toLocaleTimeString();  
}
```

REFERENCES:

- Connolly, R. & Hoar, R. (2015). *Fundamentals of web development*. New Jersey: Pearson Education, Inc.
- Lemay, L., Colburn, R., & Kyrnin, J. (2016). *Sam's 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.