18BCS53C – HTML & JavaScript

UNIT V

Window Object : Timers - Browser Location and Navigation - Browsing History - Browser and Screen Information - Dialog Boxes. Scripting Documents : Overview of the DOM - Selecting Document Elements - Document Structure and Traversal - Attributes - Element Content - Creating, Inserting, and Deleting Nodes- Generating a Table of Contents. Scripting CSS : Overview of CSS - Important CSS Properties - Scripting Inline Styles - Querying Computed Styles - Scripting CSS Classes - Scripting Stylesheets.

JavaScript Window - The Browser Object Model

The Browser Object Model (BOM) allows JavaScript to "talk to" the browser.

The Browser Object Model (BOM)

- There are no official standards for the **B**rowser **O**bject **M**odel (BOM).
- Since modern browsers have implemented (almost) the same methods and properties for JavaScript interactivity, it is often referred to, as methods and properties of the BOM.

The Window Object

- The window object is supported by all browsers. It represents the browser's window.
- All global JavaScript objects, functions, and variables automatically become members of the window object.
- Global variables are properties of the window object.
- Global functions are methods of the window object.
- Even the document object (of the HTML DOM) is a property of the window object:

```
window.document.getElementById("header");
```

is the same as:

```
document.getElementById("header");
```

Window Size

- Two properties can be used to determine the size of the browser window.
- Both properties return the sizes in pixels:
  - window.innerHeight - the inner height of the browser window (in pixels)
  - window.innerWidth - the inner width of the browser window (in pixels).

Other Window Methods

Some other methods:

- window.open() - open a new window
- window.close() - close the current window
- window.moveTo() - move the current window
- window.resizeTo() - resize the current window

JavaScript Timing Events

- JavaScript can be executed in time-intervals.
- This is called timing events.

Timing Events

- The window object allows execution of code at specified time intervals.
- These time intervals are called timing events.
- The two key methods to use with JavaScript are:
- setTimeout(*function, milliseconds*)
  Executes a function, after waiting a specified number of milliseconds.
- setInterval(*function, milliseconds*)
  Same as setTimeout(), but repeats the execution of the function continuously.
- The setTimeout() and setInterval() are both methods of the HTML DOM Window Object.

The setTimeout() Method

- window.setTimeout(*function, milliseconds*);
- The window.setTimeout() method can be written without the window prefix.
- The first parameter is a function to be executed.
- The second parameter indicates the number of milliseconds before execution.

How to Stop the Execution?

- The clearTimeout() method stops the execution of the function specified in setTimeout().
- window.clearTimeout(*timeoutVariable*)
- The window.clearTimeout() method can be written without the window prefix.
- The clearTimeout() method uses the variable returned from setTimeout():
- myVar = setTimeout(*function, milliseconds*);
  clearTimeout(myVar);
- If the function has not already been executed, you can stop the execution by calling the clearTimeout() method:

The setInterval() Method

- The setInterval() method repeats a given function at every given time-interval.
- window.setInterval(*function, milliseconds*);
- The window.setInterval() method can be written without the window prefix.
- The first parameter is the function to be executed.
- The second parameter indicates the length of the time-interval between each execution.

How to Stop the Execution?

The clearInterval() method stops the executions of the function specified in the setInterval() method.
window.clearInterval(*timerVariable*)
The window.clearInterval() method can be written without the window prefix.
The clearInterval() method uses the variable returned from setInterval():
myVar = setInterval(*function, milliseconds*);
clearInterval(myVar);

Window History

- The window.history object contains the browsers history.
- The window.history object can be written without the window prefix.
- Some methods:

- history.back() - same as clicking back in the browser
- history.forward() - same as clicking forward in the browser

Window History Back

- The history.back() method loads the previous URL in the history list.

Window History Forward

- The history.forward() method loads the next URL in the history list.

JavaScript Window Screen

- The window.screen object contains information about the user's scree

Window Screen

- The window.screen object can be written without the window prefix.
- Properties:
  - screen.width
  - screen.height
  - screen.availWidth
  - screen.availHeight
  - screen.colorDepth
  - screen.pixelDepth

Window Screen Width

- The screen.width property returns the width of the visitor's screen in pixels.

Window Screen Height

- The screen.height property returns the height of the visitor's screen in pixels.

Window Screen Available Width

- The screen.availWidth property returns the width of the visitor's screen, in pixels, minus interface features like the Windows Taskbar.

Window Screen Available Height

- The screen.availHeight property returns the height of the visitor's screen, in pixels, minus interface features like the Windows Taskbar.

Window Screen Color Depth

- The screen.colorDepth property returns the number of bits used to display one color.
- All modern computers use 24 bit or 32 bit hardware for color resolution:
  - 24 bits = 16,777,216 different "True Colors"
  - 32 bits = 4,294,967,296 different "Deep Colors"
- Older computers used 16 bits: 65,536 different "High Colors" resolution.
- Very old computers, and old cell phones used 8 bits: 256 different "VGA colors".

Window Screen Pixel Depth

- The screen.pixelDepth property returns the pixel depth of the screen.

JavaScript Popup Boxes

- JavaScript has three kind of popup boxes:
    - Alert box
    - Confirm box
    - Prompt box

Alert Box

- An alert box is often used if you want to make sure information comes through to the user.
- When an alert box pops up, the user will have to click "OK" to proceed.
- Syntax

  window.alert("*sometext*");
- The window.alert() method can be written without the window prefix.

Confirm Box

- A confirm box is often used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns **true**.
- If the user clicks "Cancel", the box returns **false**.
- Syntax

  window.confirm("*sometext*");
- The window.confirm() method can be written without the window prefix.

Prompt Box

- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.
- Syntax

  window.prompt("*sometext*","*defaultText*");
- The window.prompt() method can be written without the window prefix.

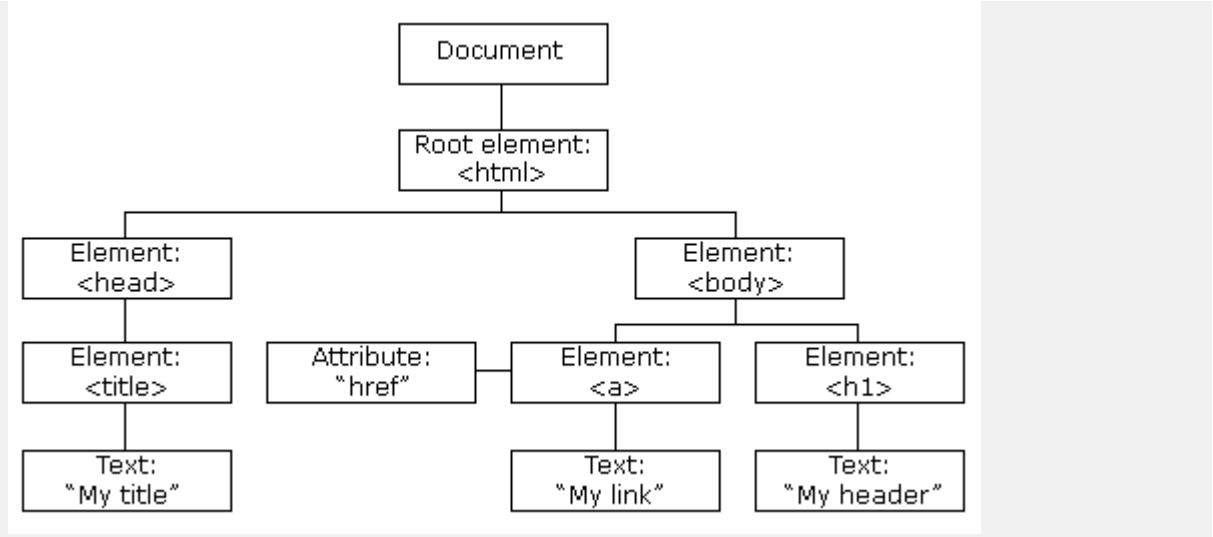
Line Breaks

- To display line breaks inside a popup box, use a back-slash followed by the character n.

JavaScript HTML DOM

- With the HTML DOM, JavaScript can access and change all the elements of an HTML document.
- The HTML DOM (Document Object Model)
- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects**:

The HTML DOM Tree of Objects



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What is the DOM?

- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM defines a standard for accessing documents:

  "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."
- The W3C DOM standard is separated into 3 different parts:
  - Core DOM - standard model for all document types
  - XML DOM - standard model for XML documents
  - HTML DOM - standard model for HTML documents

What is the HTML DOM?

- The HTML DOM is a standard **object** model and **programming interface** for HTML.
- It defines:
  - The HTML elements as **objects**
  - The **properties** of all HTML elements
  - The **methods** to access all HTML elements
  - The **events** for all HTML elements
- In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

JavaScript - HTML DOM Methods

- HTML DOM methods are **actions** you can perform (on HTML Elements).
- HTML DOM properties are **values** (of HTML Elements) that you can set or change.

The DOM Programming Interface

- The HTML DOM can be accessed with JavaScript (and with other programming languages).
- In the DOM, all HTML elements are defined as **objects**.

- The programming interface is the properties and methods of each object.
- A **property** is a value that you can get or set (like changing the content of an HTML element).
- A **method** is an action you can do (like add or deleting an HTML element).

The getElementById Method

- The most common way to access an HTML element is to use the id of the element.

The innerHTML Property

- The easiest way to get the content of an element is by using the innerHTML property.
- The innerHTML property is useful for getting or replacing the content of HTML elements.
- The innerHTML property can be used to get or change any HTML element, including <html> and <body>.

JavaScript HTML DOM Elements

Finding HTML Elements

- There are several ways to do this:
    - Finding HTML elements by id
    - Finding HTML elements by tag name
    - Finding HTML elements by class name
    - Finding HTML elements by CSS selectors
    - Finding HTML elements by HTML object collections

Finding HTML Element by Id

- The easiest way to find an HTML element in the DOM, is by using the element id.
- This example finds the element with id="intro":
- Example
    - var myElement = document.getElementById("intro");

- If the element is found, the method will return the element as an object (in myElement).
- If the element is not found, myElement will contain null.

Finding HTML Elements by Tag Name

- This example finds all <p> elements:
- Example
    - var x = document.getElementsByTagName("p");
- This example finds the element with id="main", and then finds all <p> elements inside "main":
- Example
    - var x = document.getElementById("main");
      var y = x.getElementsByTagName("p");

Finding HTML Elements by Class Name

- If you want to find all HTML elements with the same class name, use getElementsByClassName().
- This example returns a list of all elements with class="intro".
- Example
    - var x = document.getElementsByClassName("intro");

Finding HTML Elements by CSS Selectors

- If you want to find all HTML elements that match a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelectorAll() method.
- This example returns a list of all <p> elements with class="intro".
- Example
    - var x = document.querySelectorAll("p.intro");

Finding HTML Elements by HTML Object Collections

- This example finds the form element with id="frm1", in the forms collection, and displays all element values:
- Example
    - ```
      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;
      ```

JavaScript HTML DOM Elements (Nodes)

Adding and Removing Nodes (HTML Elements)

Creating New HTML Elements (Nodes)

- To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.
- Example
    - ```
      <div id="div1">
        <p id="p1">This is a paragraph.</p>
        <p id="p2">This is another paragraph.</p>
      </div>

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

      var element = document.getElementById("div1");
      element.appendChild(para);
      </script>
      ```
- Example Explained

- This code creates a new <p> element:
  var para = document.createElement("p");
- To add text to the <p> element, you must create a text node first. This code creates a text node:
  var node = document.createTextNode("This is a new paragraph.");
- Then you must append the text node to the <p> element:
  para.appendChild(node);
- Finally you must append the new element to an existing element.
- This code finds an existing element:
  var element = document.getElementById("div1");
- This code appends the new element to the existing element:
  element.appendChild(para);

Creating new HTML Elements - insertBefore()

- The appendChild() method is used to append the new element as the last child of the parent.
- The insertBefore() method can be used instead of appendChild():

Removing Existing HTML Elements

- To remove an HTML element, use the remove() method:

Find the element you want to remove:

- var elmnt = document.getElementById("p1");
- Then execute the remove() method on that element:
  elmnt.remove();
- The removeChild() can be used instead of remove() method.

Replacing HTML Elements

- To replace an element to the HTML DOM, use the replaceChild() method:

JavaScript HTML DOM Collections

The HTMLCollection Object

- The getElementsByTagName() method returns an HTMLCollection object.
- An HTMLCollection object is an array-like list (collection) of HTML elements.

The HTMLCollection Length

- The length property defines the number of elements in an HTMLCollection
- The length property is useful when you want to loop through the elements in a collection

REFERENCES

1. www.w3schools.com
2. www.guru99.com/javascript
3. www.tutorialspoint.com/javascript

----------xxxxxxxxxxxx----------