07

Object Models

WHAT YOU WILL LEARN

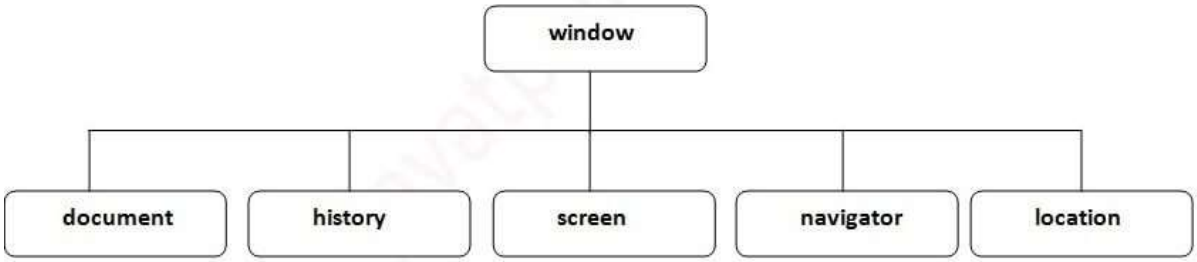
- Browser Object Model
- Document Object Model
- JavaScript Events
- Exception Handling

Browser Object Model

The Browser Object Model (BOM) is used to interact with the browser.

- The default object of browser is window means you can call all the functions of window by specifying window or directly. For example:
- window.alert("hello Engineer"); or alert("hello DEV");

BOM Hierarchy



We will cover Windows and Document object model

- Window Object
- Document
- History
- Screen
- Navigator
- Location

Window Object

The **window object** represents a window in browser. An object of window is created automatically by the browser.

Window is the object of browser, it is not the object of javascript.

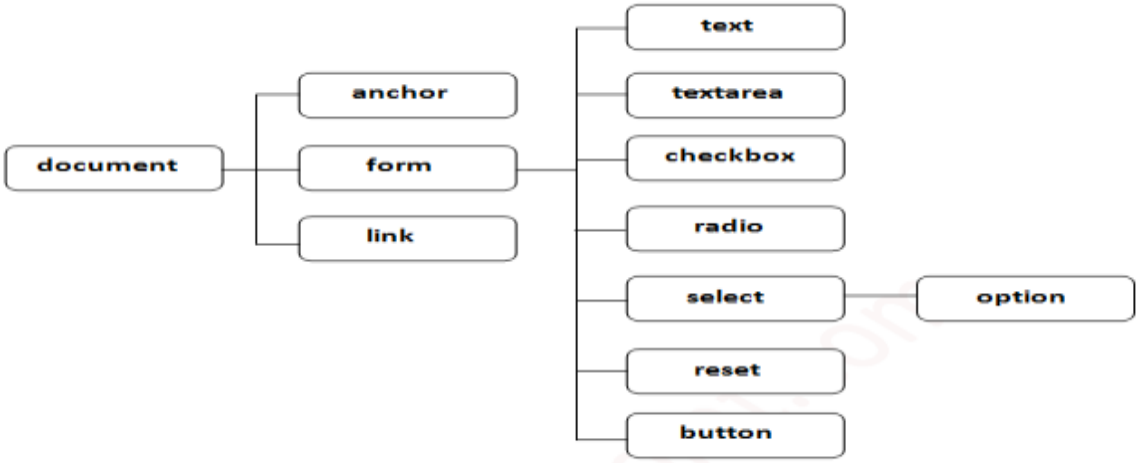
Methods of window object

Method	Description
alert()	displays the alert box containing message with ok button.
confirm()	displays the confirm dialog box containing message with ok and cancel button.
prompt()	displays a dialog box to get input from the user.
open()	opens the new window.
close()	closes the current window.
setTimeout()	performs action after specified time like calling function, evaluating expressions etc.

Document

- The document object represents the whole html document.
- When html document is loaded in the browser, it becomes a document object.
- It is the root element that represents the html document.

➢ Properties of document object



➢ Methods of document object

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

JavaScript Event

- The change in the state of an object is known as an Event.
- here are various events which represents that some activity is performed by the user or by the browser.
- This process of reacting over the events is called Event Handling.
- js handles the HTML events via Event Handlers.

Mouse events:

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Keyboard events:

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

Form events

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

Window/Document events

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

addEventListener()

- ✓ The addEventListener() method is used to attach an event handler to a particular element.
- ✓ The addEventListener() method is an inbuilt function of JavaScript.

Syntax - element.addEventListener(event, function, useCapture);

Parameter Values -

1. **event**: It is a required parameter. It can be defined as a string that specifies the event's name.
2. function: It is also a required parameter. It is a JavaScript function which responds to the event occur.
3. useCapture: It is an optional parameter. It is a Boolean type value that specifies whether the event is executed in the bubbling or capturing phase.

- ➢ onclick event –
  - Example using onclick html attribute

```
<script>
document.addEventListener("mouseover", myFunction);
document.addEventListener("click", mySecondFunction);
document.addEventListener("mouseout", myThirdFunction);

function myFunction() {
  document.getElementById("demo").innerHTML = "Moused over!"
}

function mySecondFunction() {
  document.getElementById("demo").innerHTML = "Clicked!<br>"
}

function myThirdFunction() {
  document.getElementById("demo").innerHTML = "Moused out!<br>"
}
</script>
```

- When executing JavaScript code, different errors can occur.
- Errors can be coding errors made by the programmer, errors due to wrong input, and other unforeseeable things.

JavaScript try and catch

The `try` statement allows you to define a block of code to be tested for errors while it is being executed.

The `catch` statement allows you to define a block of code to be executed, if an error occurs in the try block.

The JavaScript statements `try` and `catch` come in pairs:

The `finally` statement lets you execute code, after try and catch, regardless of the result:

```
try {
  Block of code to try
}
catch(err) {
  Block of code to handle errors
}
```

JavaScript Throws Errors

When an error occurs, JavaScript will normally stop and generate an error message.

The `throw` statement allows you to create a custom error.

Technically you can **throw an exception (throw an error)**.

The exception can be a JavaScript `String`, a `Number`, a `Boolean` or an `Object`:

The Error Object

- ✓ JavaScript has a built in error object that provides error information when an error occurs.
- ✓ The error object provides two useful properties: name and message.

Error Object Properties

Property	Description
name	Sets or returns an error name
message	Sets or returns an error message (a string)

Range Error

A `RangeError` is thrown if you use a number that is outside the range of legal values.

- ➢ For example: You cannot set the number of significant digits of a number to 500.

Reference Error

- ➢ A `ReferenceError` is thrown if you use (reference) a variable that has not been declared

Syntax Error

- ➢ A `SyntaxError` is thrown if you try to evaluate code with a syntax error.

Type Error

- ➢ A `TypeError` is thrown if you use a value that is outside the range of expected types

URI (Uniform Resource Identifier) Error

- ➢ A `URIError` is thrown if you use illegal characters in a URI function: