JavaScript

JavaScript: Scripting language which is used to enhance the functionality and appearance of web pages.

❖ JavaScript Where To

- ### The <script> Tag

In HTML, JavaScript code must be inserted between <script> and </script> tags.

Ex:

```
<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
```

➢ **JavaScript Functions and Events**

✓ A JavaScript **function** is a block of JavaScript code, that can be executed when "called" for.

✓ A function can be called when an **event** occurs, like when the user clicks a button.

- ### JavaScript in <head> or <body>

✓ You can place any number of scripts in an HTML document.

✓ Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

Ex:

```
<!DOCTYPE html>
<html>
<body>

<h1>A Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
<script>
function myFunction() {
   document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>

</body>
</html>
```

Note: placing scripts at the bottom of the <body> element improves the display speed, because script compilation slows down the display.

- **External JavaScript**

Scripts can also be placed in external files:

Ex:

```
function myFunction() {
   document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

✓ External scripts are practical when the same code is used in many different web pages.

✓ JavaScript files have the file extension **.js**.

✓ To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

Ex:

```
<!DOCTYPE html>
<html>
<body>
  <script src="myScript.js"></script>
</body>
</html>
```

Placing scripts in external files has some advantages:

- It separates HTML and code
- It makes HTML and JavaScript easier to read and maintain
- Cached JavaScript files can speed up page load

❖ What JavaScript Can Do

✓ One of many JavaScript HTML methods is **getElementById()**.

- **Change HTML Content**

This example uses the method to "find" an HTML element (with id="demo") and changes the element content (**innerHTML**) to "Hello JavaScript"

Ex:

```
document.getElementById("demo").innerHTML = "Hello JavaScript";
```

Note: JavaScript accepts both double and single quotes.

- **Change HTML Attributes**

  This example changes an HTML image by changing the src (source) attribute of an <img> tag:

  Ex:

```
<button
onclick="document.getElementById('myImage').src='pic_bulbon.gif'">Turn
on the light</button>
<img id="myImage" src="pic_bulboff.gif" style="width:100px">
<button
onclick="document.getElementById('myImage').src='pic_bulboff.gif'">Tur
n off the light</button>
```

- **Change HTML Styles (CSS)**

  Changing the style of an HTML element, is a variant of changing an HTML attribute:

  Ex:

```
document.getElementById("demo").style.fontSize = "35px";
```

- **Hide HTML Elements**

  Hiding HTML elements can be done by changing the display style:

  Ex:

```
document.getElementById("demo").style.display = "none";
```

- **Show HTML Elements**

  Showing hidden HTML elements can also be done by changing the display style:

  Ex:

```
document.getElementById("demo").style.display = "block";
```

❖ JavaScript Output

JavaScript can "display" data in different ways:

- Writing into an HTML element, using **innerHTML**.
- Writing into the HTML output using **document.write()**.
- Writing into an alert box, using **window.alert()**.
- Writing into the browser console, using **console.log()**.

- **Using innerHTML**

✓ To access an HTML element, JavaScript can use the **document.getElementById (id)** method.

✓ The **id** attribute defines the HTML element. The **innerHTML** property defines the HTML content:

Ex:

```
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>
```

- **Using document.write()**

For testing purposes, it is convenient to use **document.write()**:

Ex:

```
<script>
document.write(5 + 6);
</script>
```

- **Using window.alert()**

You can use an alert box to display data:

Ex:

```
<script>
window.alert(5 + 6);
</script>
```

- **Using console.log()**

For debugging purposes, you can use the **console.log()** method to display data.

Ex:

```
<script>
console.log(5 + 6);
</script>
```

❖ JavaScript Comments

- ✓ Single line comments start with //.
- ✓ Multi-line comments start with /* and end with */.

❖ JavaScript Variables

- ✓ All JavaScript **variables** must be **identified** with **unique names**.
- ✓ These unique names are called **identifiers**.
- ✓ A variable declared without a value will have the value undefined.

- **JavaScript Data Types**

JavaScript variables can hold numbers like 100 and text values like "John Doe".

Ex:

```
var pi = 3.14;
var person = "John Doe";
var answer = 'Yes I am!';
```

❖ JavaScript Arithmetic

As with algebra, you can do arithmetic with JavaScript variables, using operators like = and +:

Ex:

```
var x = 5 + 2 + 3;
```

You can also add strings, but strings will be concatenated:

Ex:

```
var x = "John" + " " + "Doe";
```

If you put a number in quotes, the rest of the numbers will be treated as strings, and concatenated.

Ex:

```
var x = "5" + 2 + 3
```

But try This:

```
var x = 2 + 3 + "5";
```

- **JavaScript Comparison Operators**

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

❖ JavaScript Functions

✓ A JavaScript function is a block of code designed to perform a particular task.

✓ A JavaScript function is executed when "something" invokes it (calls it).

- **JavaScript Function Syntax**

✓ A JavaScript function is defined with the **function** keyword, followed by a **name**, followed by parentheses **()**.

✓ Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

✓ The parentheses may include parameter names separated by commas: ***(parameter1, parameter2, ...)***

✓ The code to be executed, by the function, is placed inside curly brackets: **{}**

```
function name(parameter1, parameter2, parameter3) {
    code to be executed
}
```

✓ Inside the function, the arguments (the parameters) behave as local variables.

❖ JavaScript Objects

✓ In real life, a car is an **object**.
✓ A car has **properties** like weight and color, and **methods** like start and stop:

Object	Properties	Methods
	car.name = Fiat car.model = 500 car.weight = 850kg car.color = white	car.start() car.drive() car.brake() car.stop()

- **JavaScript Objects**

This code assigns a **simple value** (Fiat) to a **variable** named car:

```
var car = "Fiat";
```

Objects are variables too. But objects can contain many values.

This code assigns **many values** (Fiat, 500, white) to a **variable** named car:

```
var car = {type:"Fiat", model:"500", color:"white"};
```

The values are written as **name:value** pairs (name and value separated by a colon).

- **Accessing Object Properties**

You can access object properties in two ways:

objectName.propertyName → *Ex:* `person.lastName;`

Or

objectName["propertyName"] → *Ex:* `person["lastName"];`

- **Accessing Object Methods**

✓ A method is actually a function definition stored as a property value.

You access an object method with the following syntax:

objectName.methodName() → *Ex:* `name = person.fullName();`

❖ JavaScript Scope

Scope determines the accessibility (visibility) of variables.

In JavaScript there are two types of scope:

- Local scope
- Global scope

JavaScript has function scope: Each function creates a new scope.

- **Local JavaScript Variables**
  - ✓ Variables declared within a JavaScript function, become **LOCAL** to the function.
  - ✓ Local variables have **local scope**: They can only be accessed within the function.

Ex:

```
// code here can not use carName

function myFunction() {
    var carName = "Volvo";
    // code here can use carName
}
```

- **Global JavaScript Variables**
  - ✓ A variable declared outside a function, becomes **GLOBAL**.
  - ✓ A global variable has **global scope**: All scripts and functions on a web page can access it.

Ex:

```
var carName = " Volvo";
// code here can use carName

function myFunction() {
    // code here can use carName
}
```

- **The Lifetime of JavaScript Variables**
  - ✓ The lifetime of a JavaScript variable starts when it is declared.
  - ✓ Local variables are deleted when the function is completed.
  - ✓ In a web browser, global variables are deleted when you close the browser window (or tab), but remains available to new pages loaded into the same window.

❖ JavaScript Events

✓ HTML events are "**things**" that happen to HTML elements.

✓ When JavaScript is used in HTML pages, JavaScript can "**react**" on these events.

- **HTML Events**

Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

✓ JavaScript lets you execute code when events are detected.

✓ HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

1- With single quotes:

```
<element event='some JavaScript'>
```

2- With double quotes:

```
element event="some JavaScript">
```

In the following example, an onclick attribute (with code), is added to a button element:

Ex:

```
<button onclick="document.getElementById('demo').innerHTML =
Date()">The time is?</button>
```

In the next example, the code changes the content of its own element (using **this**.innerHTML):

Ex:

```
<button onclick="this.innerHTML = Date()">The time is?</button>
```

- **Common HTML Events**

Here is a list of some common HTML events:

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

Event handlers can be used to handle, and verify, user input, user actions, and browser actions:

- Things that should be done every time a page loads
- Things that should be done when the page is closed
- Action that should be performed when a user clicks a button
- Content that should be verified when a user inputs data
- And more ..