JAVASCRIPT

1) JavaScript is the world's most popular programming language.

2) JavaScript is the programming language of the Web.

3) JavaScript is easy to learn.

The <script> Tag

In HTML, JavaScript code is inserted between `<script>` and `</script>` tags.

Example

```
<!DOCTYPE html>
<html>

<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph
changed.";
}
</script>
</head>
<body>
```

```
<h1>A Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

✓ 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 `alert()`.

• 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:

Example

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

<h1>My First Web Page</h1>
```

```
<p>My First Paragraph</p>

<p id="demo"></p>

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

</body>
</html>
```

- ## Using document.write()

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

Example

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

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

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

</body>
</html>
```

- **Using alert()**

You can use an alert box to display data:

Example

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

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
alert(5 + 6);
</script>

</body>
</html>
```

✓ Variables

- JavaScript variables are containers for storing data values.
- In this example, `x`, `y`, and `z`, are variables, declared with the `var` keyword:

Example

```
var x = 5;
var y = 6;
var z = x + y;
```

From the example above, you can expect:

- x stores the value 5
- y stores the value 6
- z stores the value 11

✓ Operators

Arithmetic Operators :-

Operator	**Description**
+	Addition
-	Subtraction
*	Multiplication
/	Division

%	Modulus (Division Remainder)
++	Increment
--	Decrement

Example

```
var x = 5;         // assign the value 5 to x
var y = 2;         // assign the value 2 to y
var z = x + y;     // assign the value 7 to z (5 + 2)
```

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

Logical Operators :-

Operator	**Description**
&&	logical and
||	logical or

✓ Comments

1. JavaScript comments can be used to explain JavaScript code, and to make it more readable.
2. JavaScript comments can also be used to prevent execution, when testing alternative code.

• Single Line Comments

Single line comments start with `//`.

Any text between `//` and the end of the line will be ignored by JavaScript (will not be executed).

• Multi-line Comments

Multi-line comments start with `/*` and end with `*/`.

Any text between `/*` and `*/` will be ignored by JavaScript.

✓ if else and else if

In JavaScript we have the following conditional statements:

- Use `if` to specify a block of code to be executed, if a specified condition is true
- Use `else` to specify a block of code to be executed, if the same condition is false
- Use `else if` to specify a new condition to test, if the first condition is false

• if Statement

Syntax

```
if (condition) {
  //  block of code to be executed if the condition is true
}
```

Example

```
if (hour < 18) {
  greeting = "Good day";
}
```

• The else Statement

Use the `else` statement to specify a block of code to be executed if the condition is false.

```
if (condition) {
  //  block of code to be executed if the condition is true
} else {
  //  block of code to be executed if the condition is false
}
```

Example

```
if (hour < 18) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

- ## The else if Statement

Use the `else if` statement to specify a new condition if the first condition is false.

Syntax

```
if (condition1) {
  //  block of code to be executed if condition1 is true
}
else if (condition2) {
  //  block of code to be executed if the condition1 is
false and condition2 is true
}
 else {
  //  block of code to be executed if the condition1 is
false and condition2 is false
}
```

Example

```
if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

✓ Switch Statement

The `switch` statement is used to perform different actions based on different conditions.

Syntax

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

Example

```
switch (0) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
     day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
}
```

• break Keyword

When JavaScript reaches a `break` keyword, it breaks out of the switch block.

This will stop the execution inside the switch block.

• default Keyword

The `default` keyword specifies the code to run if there is no case match:

Example

```
switch (1) {
  case 6:
    text = "Today is Saturday";
    break;
  case 0:
    text = "Today is Sunday";
    break;
  default:
    text = "Looking forward to the Weekend";
}
```

✓ Loop

Loops can execute a block of code a number of times.

Different Kinds of Loops

JavaScript supports different kinds of loops:

- `for` - loops through a block of code a number of times
- `while` - loops through a block of code while a specified condition is true
- `do/while` - also loops through a block of code while a specified condition is true

For Loop

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed.

Example

```
for (i = 0; i < 5; i++) {
  text += "The number is " + i + "<br>";
}
```

• While Loop

The `while` loop loops through a block of code as long as a specified condition is true.

Syntax

```
while (condition) {
  // code block to be executed
}
```

Example

```
while (i < 10) {
  text += "The number is " + i;
  i++;
}
```

• Do/While Loop

The `do/while` loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

```
do {
  // code block to be executed
}
while (condition);
```

Example

```
do {
  text += "The number is " + i;
  i++;
}
while (i < 10);
```

- ## Continue Statement

The `continue` statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

Example

```
for (i = 0; i < 10; i++) {
  if (i === 3) { continue; }
  text += "The number is " + i + "<br>";
}
```

✓ Arrays

JavaScript arrays are used to store multiple values in a single variable.

Example

```
var cars = ["Saab", "Volvo", "BMW"];
```

• Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:

```
var array_name = [item1, item2, ...];
```

• Access the Elements of an Array

You access an array element by referring to the **index number**.

This statement accesses the value of the first element in `cars`:

```
var name = cars[0];
```

• length Property

The `length` property of an array returns the length of an array (the number of array elements).

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.length;   // the length of fruits is 4
```

- ## Looping Array Elements

The safest way to loop through an array, is using a `for` loop:

Example

```
var fruits, text, fLen, i;
fruits = ["Banana", "Orange", "Apple", "Mango"];
fLen = fruits.length;

text = "<ul>";
for (i = 0; i < fLen; i++) {
  text += "<li>" + fruits[i] + "</li>";
}
text += "</ul>";
```

✓ Functions

1) A JavaScript function is a block of code designed to perform a particular task.
2) A JavaScript function is executed when "something" invokes it (calls it).

• Function Syntax

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

Example

```
function myFunction(p1, p2) {
  return p1 * p2;   // The function returns the product of
p1 and p2
}
```

• Local Variables

Variables declared within a JavaScript function, become **LOCAL** to the function.

Local variables can only be accessed from within the function.

Example

```
// code here can NOT use carName

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

// code here can NOT use carName
```

✓ Events

1. HTML events are **"things"** that happen to HTML elements.
2. When JavaScript is used in HTML pages, JavaScript can **"react"** on these events.

Event	**Description**
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

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

With single quotes:

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

With double quotes:

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

Example

```
<button onclick="Date();">The time is?</button>
```

✓ DOM - Changing HTML

The HTML DOM allows JavaScript to change the content of HTML elements.

Changing HTML Content

The easiest way to modify the content of an HTML element is by using the `innerHTML` property.

To change the content of an HTML element, use this syntax:

```
document.getElementById(id).innerHTML = new HTML
```

This example changes the content of a `<p>` element:

Example

```
<html>
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML = "New
text!";
</script>

</body>
</html>
```

Changing the Value of an Attribute

To change the value of an HTML attribute, use this syntax:

`document.getElementById(`*`id`*`).`*`attribute = new value`*

This example changes the value of the src attribute of an `<img>` element:

Example

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

<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src = "landscape.j
```

```
pg";
</script>

</body>
</html>
```

✓ DOM - Changing CSS

The HTML DOM allows JavaScript to change the style of HTML elements.

• Changing HTML Style

To change the style of an HTML element, use this

syntax:

document.getElementById(*id*).style.*property* = *new style*

Example

```
<html>
<body>

<p id="p2">Hello World!</p>
```

```
<script>
document.getElementById("p2").style.color = "blue";
</script>

<p>The paragraph above was changed by a script.</p>

</body>
</html>
```

✓ Timing Events

- `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.

- ## setTimeout() Method

Example

```
<button onclick="setTimeout(myFunction, 3000)">Try
it</button>

<script>
```

```
function myFunction() {
  alert('Hello');
}
</script>
```

• How to Stop the Execution?

The `clearTimeout()` method stops the execution of the function specified in setTimeout()

The `clearTimeout()` method uses the variable returned from `setTimeout()`:

```
myVar = setTimeout(function, milliseconds);
clearTimeout(myVar);
```

Example

```
<button onclick="myVar = setTimeout(myFunction,
3000)">Try it</button>

<button onclick="clearTimeout(myVar)">Stop
it</button>
```

• The setInterval() Method

The `setInterval()` method repeats a given function at every given time-interval.

Example

Display the current time:

```
var myVar = setInterval(myTimer, 1000);

function myTimer() {
  var d = new Date();
  document.getElementById("demo").innerHTML =
d.toLocaleTimeString();
}
```

There are 1000 milliseconds in one second.

✓ Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

1. Alert Box

When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
alert("sometext");
```

Example

```
alert("I am an alert box!");
```

2. Confirm Box

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");
```

Example

```
if (confirm("Press a button!")) {
  txt = "You pressed OK!";
} else {
  txt = "You pressed Cancel!";
}
```

3. Prompt Box

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

Syntax

```
window.prompt("sometext","defaultText");
```

Example

```
var person = prompt("Please enter your name", "Harry
Potter");

if (person == null || person == "") {
  txt = "User cancelled the prompt.";
} else {
  txt = "Hello " + person + "! How are you today?";
}
```

4. Line Breaks

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

Example

```
alert("Hello\nHow are you?");
```

✓ Scroll Back To Top Button

Step 1) Add HTML:

Create a button that will take the user to the top of the page when clicked on:

Example

```
<button onclick="topFunction()" id="myBtn" title="Go to
top">Top</button>
```

Step 2) Add CSS:

Style the button:

Example

```
#myBtn {
  display: none; /* Hidden by default */
  position: fixed; /* Fixed/sticky position */
  bottom: 20px; /* Place the button at the bottom of the
page */
  right: 30px; /* Place the button 30px from the right */
  z-index: 99; /* Make sure it does not overlap */
  border: none; /* Remove borders */
  outline: none; /* Remove outline */
  background-color: red; /* Set a background color */
  color: white; /* Text color */
  cursor: pointer; /* Add a mouse pointer on hover */
  padding: 15px; /* Some padding */
  border-radius: 10px; /* Rounded corners */
  font-size: 18px; /* Increase font size */
}

#myBtn:hover {
  background-color: #555; /* Add a dark-grey background on
hover */
}
```

Step 3) Add JavaScript:

Example

```
// When the user scrolls down 20px from the top of the
document, show the button
window.onscroll = function() {scrollFunction()};

function scrollFunction() {
  if (document.body.scrollTop > 20 || document.documentEleme
nt.scrollTop > 20) {
    document.getElementById("myBtn").style.display = "block"
;
  } else {
    document.getElementById("myBtn").style.display = "none";
  }
}

// When the user clicks on the button, scroll to the top of
the document
function topFunction() {
  document.body.scrollTop = 0; // For Safari
  document.documentElement.scrollTop = 0; // For Chrome,
Firefox, IE and Opera
}
```

BE SAFE AND KEEP PRACTICING

brighten your future....

The Institute of Commerce & Vocational Studies