

**COURSE NAME: UI Web DevelopmentCourse Code: MR23-1CS0102**

**UNIT-IV**

**JAVASCRIPT-II**

CONDITIONAL STATEMENTS:

Conditional statements are used to perform different actions based on different conditions. if statement is used to specify a block of code to be executed, if a specified condition is true. The else statement is used to specify a block of code to be executed, if the same condition is false. The else if  statement is used to specify a new condition to test, if the first condition is false. The switch statement is used to specify many alternative blocks of code to be executed.

Conditional Statements: (if Statement):

Syntax:

if (condition)

{

//block of code to be executed if the condition is true

}

Example Program:

<html>

<head>

<title>IF Statments!!!</title>

<script>

var age = prompt("Please enter your age");

if(age>=18)

document.write("You are an adult <br />");

if(age<18)

document.write("You are NOT an adult <br />");

</script>

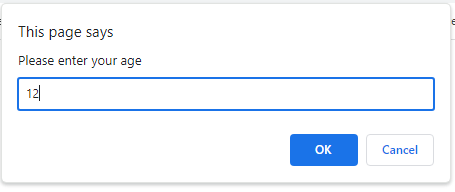
</head>

<body>

</body>

</html>

OUTPUT:



You are NOT an adult

Conditional Statements(else Statement):

Syntax:

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 Program:

<html>

<head>

<title>If...Else Statments!!!</title>

</head>

<body>

<h1>if - else conditional statements</h1>

<script>

// Get the current hours

var d =new Date();

hours=d.getHours();

if(hours<12)

document.write("Good Morning!!!<br>");

else

document.write("Good Afternoon!!!<br>");

</script>

</body>

</html>

OUTPUT:

if - else conditional statements

Good Afternoon!!!

NOTE: new Date().getHours()- prints system time and date

Conditional Statements(If….Else If….Else):

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 Program:

<html>

<head>

<script>

var one = prompt("Enter the first number");

var two = prompt("Enter the second number");

if (one == two)

document.write(one + " is equal to " + two + ".");

else if (one<two)

document.write(one + " is less than " + two + ".");

else

document.write(one + " is greater than " + two + ".");

</script>

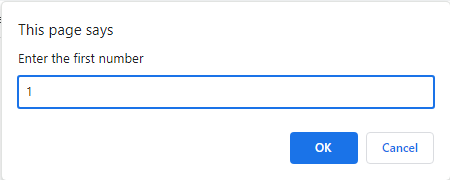
</head>

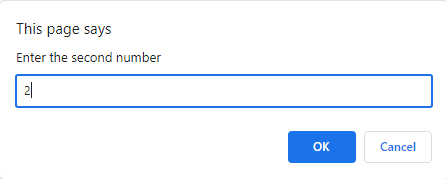
<body>

</body>

</html>

OUTPUT:





1 is less than 2.

SWITCH:

SYNTAX:

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

Example Program:

!DOCTYPE html>

<html>

<body>

<h2>JavaScript switch</h2>

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

<script>

let day;

switch (new Date().getDay()) {

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";

}

document.getElementById("demo").innerHTML = "Today is " + day;

</script>

</body>

</html>

OUTPUT:

**JavaScript switch**

Today is Sunday

LOOPING:

Loops in JavaScript allow you to execute a block of code multiple times.

1. For Loop
2. For...In Loop
3. For...Of Loop
4. While Loop
5. Do while Loop

For Loop:

The for statement creates a loop with 3 optional expressions:

SYNTAX:

for (expression 1; expression 2; expression 3) {

// code block to be executed

}

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

Expression 2 defines the condition for executing the code block.

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

Example Program:

<!DOCTYPE html>

<html>

<head>

<title>For Loop Statement in JavaScript</title>

</head>

<body>

<h2>For loop Statement in JavaScript</h2>

<script>

var i;

for(i=0;i<=10;i++)

{

document.write("The number is :" +i +"<br/>");

}

</script>

</body>

</html>

OUTPUT:

**For loop Statement in JavaScript**

The number is :0  
The number is :1  
The number is :2  
The number is :3  
The number is :4  
The number is :5  
The number is :6  
The number is :7  
The number is :8  
The number is :9  
The number is :10

For ….in Loop:

The JavaScript for in statement loops through the properties of an Object:

Syntax:

for (key in object) {  
  // code block to be executed  
}

Example Program:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Arrays</h1>

<h2>For In Loops</h2>

<p>The for in statement can loops over array values:</p>

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

<script>

const numbers = [45, 4, 9, 16, 25];

let txt = "";

for ( x in numbers) {

txt += numbers[x] + "<br>";

}

document.write(txt)

</script>

</body>

</html>

OUTPUT:

JavaScript Arrays

For In Loops

The for in statement can loops over array values:

45

4

9

16

25

For of Loop:

SYNTAX:

for (variable of iterable) {  
  // code block to be executed  
}

The JavaScript for of statement loops through the values of an iterable object.It lets you loop over iterable data structures such as Arrays, Strings, Maps, NodeLists, and more.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript For Of Loop</h2>

<p>The for of statement loops through the values of an iterable object.</p>

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

<script>

let language = "JavaScript";

let text = "";

for (let x of language) {

text += x + "<br>";

}

document.getElementById("demo").innerHTML = text;

</script>

</body>

</html>

OUTPUT:

JavaScript For Of Loop

The for of statement loops through the values of an iterable object.

J

a

v

a

S

c

r

i

p

t

While Loop

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

SYNTAX:

while (condition)

{

Execute these statements; //true

Increment/decrement;

}

Example Program:

<!DOCTYPE html>

<html>

<body>

<h1> Demonstration of while loop</h1>

<script>

let n = 0;

while (n < 3) {

document.write(n +"<br>" )

n++;

}

</script>

</body>

</html>

OUTPUT:

Demonstration of while loop

0

1

2

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

{

Execute the statements;

Increment/decrements;

}

while(condition) //true

Example Program:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Do While Loop</h2>

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

<script>

let text = ""

let i = 0;

do {

text += "<br>The number is " + i;

i++;

}

while (i < 10);

document.getElementById("demo").innerHTML = text;

</script>

</body>

</html>

OUTPUT:

**JavaScript Do While Loop**

The number is 0

The number is 1

The number is 2

The number is 3

The number is 4

The number is 5

The number is 6

The number is 7

The number is 8

The number is 9

JAVASCRIPT FUNCTIONS:

* A **function**is a set of statements that take inputs, do some specific computation, and produce output.
* The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can call that function.

Syntax:

functionfunctionName(Parameter1, Parameter2, ...)

{

// Function body

}

Example Program:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Functions</h1>

<script>

functionmyFunction(p1, p2) {

return p1 \* p2;

}

let result = myFunction(4, 3);

document.write(result)

</script>

</body>

</html>

OUTPUT:

**JavaScript Functions**

12

FUNCTION PARAMETERS:

Function parameters are the names listed in the function definition. The parameter rules are listed below:

* JavaScript function definitions do not specify data types for parameters.
* JavaScript functions do not perform type checking on the passed arguments.
* JavaScript functions do not check the number of arguments received.

FUNCTION INVOCATION:

The code inside a function is not executed when the function is **defined**.The code inside a function is executed when the function is **invoked**.It is common to use the term "**call a function**" instead of "**invoke a function**".

FUNCTION CALL:

The Javascript Function call is a predefined javascript method, that is used to write methods for different objects. The keyword this refers to the “owner” of the function or the object it belongs to.

// function that returns product of two numbers

function product(a, b) {

return a \* b;

}

// Calling product() function

let result = product.call(20, 5);

Document.write(result);

FUNCTION APPLY:

* The call() method takes arguments **separately**.
* The apply() method takes arguments as an **array**.

Example Code:

function sum(a, b) {

return a + b;

}

var numbers = [5, 10];

var result = sum.**apply**(null, numbers);

console.log(result); //

Output: **15**

NOTE: The first argument of `apply()` is set to `null` because we don't need to set a specific context for the `sum` function.

ARRAY:

An array is a special variable, which can hold more than one value. For example: constcars = ["Saab", "Volvo", "BMW"].

Accessing elements of array:

<script>

const cars = ["Saab", "Volvo", "BMW"];

document.write(**cars[1]);**

OUTPUT:

Volvo

Changing an array element:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Arrays</h1>

<script>

const cars = [“Mecedes", "Volvo", "BMW"];

cars[0]=“Maruti"

document.write(cars);

</script>

</body>

</html>

OUTPUT:

**JavaScript Arrays**

Maruti,Volvo,BMW

To find length of an array element:

<!DOCTYPE html>

<html>

<body>

<script>

const fruits = ["Banana", "Orange", "Apple", "Mango"];

let size = fruits.length;

document.write(size);

</script>

</body>

</html>

OUTPUT:

4

Example Code:

// Creating an array

var fruits = ['apple', 'banana', 'orange'];

// Accessing elements in the array

document.write(fruits[0]); // Output: 'apple'

// Adding elements to the array

fruits.push('grape');

document.write(fruits); // Output: ['apple', 'banana', 'orange', 'grape']

// Removing elements from the array

fruits.pop();

console.log(fruits); // Output: ['apple', 'banana', 'orange']

// Checking the length of the array

console.log(fruits.length); // Output: 3

JAVASCRIPT **Array** METHODS

1. JavaScript Array **length**

Returns the number of elements in an array

let city = [“Hyderabad", “Ramanthapur", “Secunderabad"];

let len = city.length;

console.log(len);

1. JavaScript Array **reverse()**

Returns the array in reverse order

let numbers = [1, 2, 3, 4, 5];

let result= numbers.reverse();

console.log(result);

1. JavaScript Array **sort()**

Sorts the elements of an array in specific order

let city = [“Hyderabad", “Ramanthapur", “Secunderabad"];

let result = city.sort();

console.log(result);

1. JavaScript Array **toSort()**

Returns a string with elements of the array separated by commas

let items = ["JavaScript", 1, "a", 3];

let result = items.toString();

console.log(result);

1. JavaScript Array **pop()**

Removes and returns the last array element

let cities = [“Hyderabad", “Ramanthapur", “Secunderabad"];

let result= cities.pop();

console.log(cities);

console.log(result);

1. JavaScript Array **push()**

Adds elements to end of array & returns its length

let city = [“Hyderabad", “Ramanthapur", “Secunderabad"]; city.push(“Bangalore");

console.log(city);

JAVASCRIPT STRING:

JavaScript string is a primitive data type that is used to work with texts.

Example: const name = 'John';

Create JavaScript Strings

* In JavaScript, strings are created by surrounding them with quotes. There are three ways you can use quotes.
* Single quotes: 'Hello'
* Double quotes: "Hello"
* Backticks: `Hello`

Example

const name = ‘Ram';

const name1 = “Sita";

const result = `The names are ${name} and ${name1}`;

Backticks are generally used when you need to include variables or expressions into a string. This is done by wrapping variables or expressions with ${variable or expression} as shown above.

You can also write a quote inside another quote. For example,

const name = 'My name is “Ram".';

However, the quote should not match the surrounding quotes. For example,

const name = 'My name is ‘Ram'.'; // error

JAVASCRIPT **STRING** METHODS

**1. concat()**

joins two or more strings

**2. toUpperCase()**

returns the passed string in upper case

**3. trim()**

removes whitespace from the strings

**4. split()**

converts the string to an array of strings

**5. slice(start, end)**

returns a part of a string

6. **match()**

returns an array containing the results of matching a string against a string

7. **search()**

searches a string for a string (or a regular expression) and returns the position of the match

EXAMPLE CODES:

The search() Method:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Strings</h1>

<h2>The search() Method</h2>

<p>The search() method returns the position of the first occurrence of a string in a string.</p>

<p>The position of the first occurrence of "locate" is:</p>

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

<script>

let text = "Please locate where 'locate' occurs!";

let index = text.search("locate");

document.getElementById("demo").innerHTML = index;

</script>

</body>

</html>

OUTPUT:

JavaScript Strings

The search() Method

The search() method returns the position of the first occurrence of a string in a string.

The position of the first occurrence of "locate" is: 7

The match() Method:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Strings</h1>

<h2>The match() Method</h2>

<p>Perform a search for "ain":</p>

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

<script>

let text = "The rain in SPAIN stays mainly in the plain";

constmyArr = text.match("ain");

document.getElementById("demo").innerHTML = myArr.length + " " + myArr;

</script>

</body>

</html>

OUTPUT:

JavaScript Strings

The match() Method

Perform a search for "ain":

1 ain

The split() Method:

A string can be converted to an array with the split() method:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript String Methods</h1>

<p>Display the first array element, after a string split:</p>

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

<script>

let text = "a,b,c,d,e,f";

const myArray = text.split(",");

document.getElementById("demo").innerHTML = myArray[0];

</script>

</body>

</html>

OUTPUT:

JavaScript String Methods

Display the first array element, after a string split:

a

To find the sum of ‘n’ Natural Numbers:

<!DOCTYPE html>

<html>

<head>

<title>sum of n Natural Numbers</title>

</head>

<body>

<h1> SUM OF n NATURAL NUMBERS </h1>

<script>

const number = prompt('Enter a positive integer: ');

let sum = 0;

// looping from i = 1 to number

// in each iteration, i is increased by 1

for (let i = 1; i <= number; i++) {

sum += i;

}

document.write('The sum of natural numbers:'+sum);

</script>

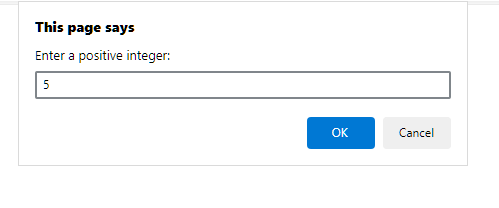
</body>

</html>

OUTPUT:

**SUM OF n NATURAL NUMBERS**

The sum of natural numbers:15



THREE JavaScript Global Functions

1. eval() function

The eval() function in JavaScript is used to evaluate the expression. It is JavaScirpt's global function, which evaluates the specified string as JavaScript code and executes it.

The parameter of the eval() function is a string. If the parameter represents the statements, eval() evaluates the statements. If the parameter is an expression, eval() evaluates the expression. If the parameter of eval() is not a string, the function returns the parameter unchanged.

EXAMPLE PROGRAM:

<html>

<head>

<script>

var a = 10, b = 20, c = 30, sum, mul, sub;

sum = eval(" a + b + c ");

mul = eval(" a  \* b \* c");

sub = eval(" a  - b");

document.write(sum + "<br>");

document.write(mul + "<br>");

document.write(sub);

</script>

</head>

<body>

</body>

</html>

OUTPUT:

60

6000

-10

1. Number()

Converts an object's value to a number

Convert booleans and dates to a number:

Number(true);

Number(false);

Number(new Date());

The Number() method converts a value to a number.

If the value cannot be converted, NaN is returned.

Example PROGRAM:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Global Methods</h1>

<h2>The Number() Method</h2>

<p>Number() converts a value to a number if possible:</p>

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

<script>

document.getElementById("demo").innerHTML =

Number(true) + "<br>" +

Number(false) + "<br>" + );

</script>

</body>

</html>

OUTPUT:

JavaScript Global Methods

The Number() Method

Number() converts a value to a number if possible:

1

0

1. parseInt()

Parses a string and returns an integer

The parseInt method parses a value as a string and returns the first integer.

A radix parameter specifies the number system to use:

2 = binary, 8 = octal, 10 = decimal, 16 = hexadecimal.

If radix is omitted, JavaScript assumes radix 10. If the value begins with "0x", JavaScript assumes radix 16.

Example PROGRAM:

<!DOCTYPE html>

<html>

<body>

<script>

document.write(parseInt('123'));

// 123 (default base-10)

document.write(parseInt('123', 10));

// 123 (explicitly specify base-10)

document.write(parseInt(' 123 '));

</script>

</body>

</html>

OUTPUT:

123123123

**JavaScript code which checks the contents entered in a forms text element. If the text entered is in the lower cases convert to upper case.**

<html>

<head>

<script type = "text/javascript">

function upper()

{

var upper = document.form1.string.value.toUpperCase();

document.form1.string.value = upper;

}

</script>

</head>

<body>

<form name = "form1">

<table>

<tr>

<td>Enter String</td>

<td><input type = "text" name = "string" /></td>

</tr>

<tr>

<td><input type = "button" value = "upper" onclick = "upper()"></td>

</tr>

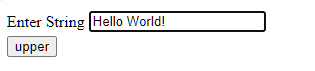
</table>

</form>

</body>

</html>

OUTPUT:



**JavaScript program to find factorial of a number**

### Logic to find factorial of the number

* Declared fact variable with initialization value 1.
* Take a number from the user and store it in a variable a.
* Iterate the loop from 1 to user value i.e a,
* inside the loop calculate the factorial of a given number by using code   fact=fact\*i;
* when the loop breaks factorial value will get stored in a fact variable, after the loop end print the factorial value.

<!DOCTYPE html>

<html>

<body>

<script>

function factorial()

{

var fact=1,i;

var a=prompt("enter a number:");

for(i=1;i<=a;i++)

{

fact=fact\*i;

}

document.write("factorial of number "+a+" is:",fact);

}

</script>

<form>

<input type="button" value="factorial" onclick="factorial();">

</form>

</body>

</html>

OUTPUT:

