2.2 INTRODUCTION TO JAVASCRIPT

JavaScript is a client side scripting language developed by Netscape for use within HTML web pages. JavaScript is loosely based on Java and it is built into all the major modern browsers like Internet Explorer, Firefox, Chrome, Safari etc.

Features of JavaScript

- JavaScript is a lightweight, interpreted scripting language that is directly embedded into web pages.
- It is used for creating network-centric applications. It is complementary to and integrated with Java and HTML.

- It is an open and cross-platform scripting language.It adds interactivity to HTML pages.

Capabilities of JavaScript

- JavaScript acts as a programming tool for web designers.They can add dynamic features into an HTML page.
- JavaScript can react to events.JavaScript can read and write HTML elements and validate input data.JavaScript can be used to create cookies and much more.

Placement of JavaScript in a HTML File:

The following are the ways to include JavaScript in the HTML file:

- Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections. Here JavaScript is included at both head and body of the HTML. The above three are **inline** JavaScripts
- Script in and external file and then include in <head>...</head> section. Here the JavaScript is an external file and the JavaScript file is linked with the HTML file in the header section. This is **external** JavaScript.

Inline JavaScript

In Inline JavaScript, the scripts can be placed **anywhere** on the page. The output of a page will appear where the script block is in the HTML file. For instance if the JavaScript blocks are placed in the header region of the HTML document, then the dynamic content will appear in the header part of the web page and if the script blocks are at the body region of the HTML document, then the dynamic content will appear in the body part of the web page.

It is a good practice to place the scripts at the bottom of the HTML document. The reason is that each time the browser encounters a <script> tag it has to pause, compile the script, execute the script, then continue on generating the page. This takes time.

External JavaScript

External JavaScript allows the **reuse of same block of code** on several different web pages. The JavaScript code will be written on a separate page and the web pages can make use of this code by including the page in the src attribute of the script tag.

The biggest advantage to have an external JavaScript file is that once the file has been loaded, the script will remain in the the browser's cache area. So the next time the page will be loaded from the browser's cache instead of having to reload it over the Internet. This enables faster execution.

Syntax: <script type='text/javascript' src='filename.js'>

</script>

When the browser encounters this block it will load filename.js and execute it.

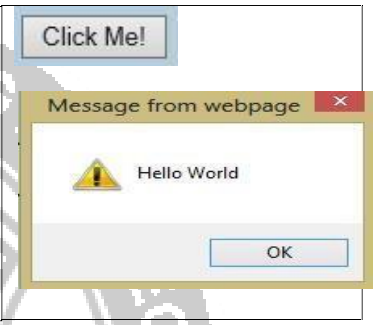
Advantages of external JavaScript

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

External JavaScript

external.js

```
function popup()
{alert("Hello World")}
<html><head><script src="external.js"></script></head>
<body>
<input type="button" onclick="popup()" value="Click Me!">
</body></html>
```



Hello World

The external.js is a JavaScript file. It cannot contain <script></script>.

Differences between inline and external javascript

Inline JavaScript	**External JavaScript**
The JavaScript code will be embedded in the same html document.	The JavaScript code will be included in the src attribute of the <script> in the html document. The JavaScript code will not be a part of the html document.
Difficult to maintain and slow.	Easy to maintain and faster execution since the external file is stored in brower's cache.

Creating a simple web page with JavaScript

JavaScript is embedded inside a HTML code.A JavaScript consists of set of JavaScript statements that are placed within the <script>... </script> HTML tags in a web page. The <script>tag alert the browser program to begin interpreting all the text between these tags as a script.

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. Usage of semi colons is optional. But it is a good programming practice to use semi colons where ever necessary to enhance the readability of the code. JavaScript is a **case-sensitive** language. The identifiers "CAT" and "cat" are two different tokens in JavaScript because of their cases.

Syntax: <script>   JavaScript code</script>

The attributes of the script tag are:

- **Language:**

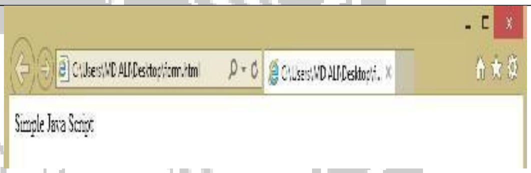
  This specifies the scripting language used.In case of JavaScript, the value will be javascript. If other scripting languages are used, then this attribute will take the names of the language used.This is an optional attribute.

- **Type**

  This attribute specifies the type of code.Its value should be set to text/javascript.**Example:**<script language="JavaScript" type="text/javascript">

  JavaScript code </script>

Simple JavaScript

<html><body> *<script language="javascript" type="text/javascript">* *document.write("Simple Java Script")* *</script></body></html>*	 Simple Java Script

- The first method is to use is the document.writeln(string). This is used while the web page is being constructed. After the page has finished loading a new document.writeln(string) command will delete the page in most browsers.

- As the page is loading, JavaScript will encounter this script and it will output " Simple Java Script " exactly where the script block appears on the page.The problem with writeln is that if this method is used after the page has loaded the browser will destroy the page and start constructing a new one.

Comments in JavaScript

JavaScript supports both C-style and C++-style comments.The text between // and the end of a line is treated as a comment and is ignored by JavaScript. This is single line comment.The text between the characters /* and */ is treated as a comment. This is multi-line comment.JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment. The closing of HTML comment should be written as //-->.

Data types

JavaScript allows three primitive data types:Numbers, Strings and Boolean. JavaScript also defines two trivial data types, null and undefined both definesonlya single value.

Reserved words

The reserved words or keywords cannot be used as JavaScript variables, functions, methods, loop labels, or any object names. The following are the keywords in JavaScript:

Abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

JavaScript Variables

Variables are named containers. JavaScript is not a strongly typed language. The programmer needs to care only what the variable is storing. In JavaScript the variables can store anything, even functions. Before using a variable in a JavaScript program, it must be declared.

Syntax:varvariable_name;

Here var is the keyword and is optional. Any variable in JavaScript is declared without specifying its data type. The variable takes the type of the value it holds.

1. var s= 'This is a string'; //now s is of string data type
2. var s = 25; //now s is of number or integer data type
3. var s = true; // now s is of Boolean data type.
4. var s = [0, 'one', 2, 3, '4', 5]; // now s is of array data type
5. var s = { 'color': 'red'} //now s is of object data type. Color is a JavaScript object
6. var s = function()

   { return “example function” }

   // The compiler executes the function and stores the return value of the function which is// " example function" in the variable s.

Naming variables

- Keywords in JavaScript cannot be used as a valid variable name.JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or the underscore character. JavaScript variable names are casesensitive.

Scope of a variable

The lifetime of the JavaScript variables starts when they are declared, and ends when the page is closed. The **scope of a variable** is the region of the program in which it is defined. JavaScript variables will have only two scopes:

Global Variables: A global variable has global scope which means it can be accessed everywhere in the JavaScript code of the web page. If a function defines a new variable without using the var keyword, that variable will be a global variable.

Local Variables: A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Special Keywords

JavaScript has a few pre-defined variables with special or fixed meaning. The following are those special keywords:

- NaN (Not a Number)-This is generated when an arithmetic operation returns an invalid result.
- Infinity is a keyword which is returned when an arithmetic operation overflows JavaScript's precision which is in the order of 300 digits.
- Null is a reserved word that means "empty". In boolean operations null evaluates as false.JavaScript supports true and false as boolean values.
- Undefined value-If a variable has not been declared or assigned yet then that variable will be given a special undefined value. In boolean operations undefined evaluates as false.

Arithmetic Operators: The following are the arithmetic operators supported by JavaScript: +, -, *, /, % (modulus), ++ and ___

Comparison Operators: Javascript supports ==, !=, >, <, >= and <= operators.

Logical Operators: The following are the logical operators supported by JavaScript:&&, || and !.

Bitwise Operators: The following are the bitwise operators supported by JavaScript: &, | and ^.

Assignment Operators: The following are the assignment operator formats supported by JavaScript: =, +=, -=, *=, /=, %=.

Conditional Operator or ternary operator (? :): This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation.

if (operand1 conditonal operator operand 2)? statement1 :statement2

Example:if(a= =b)?1:0

typeof Operator

The typeof is a unary operator. This operator returns the data type of the operand. The typeof operator valuates to number, string, or boolean depending on the value taken by the operand.

Syntax: typeof(operand)	**Example:** typeof(1) //This returns number

JavaScript Statements

Statements define what the script will do and how it will be done. The end of a statement is indicated with a semicolon(;). The following are the types of statements in JavaScript:Conditional Statements, Loop Statements, Object Manipulation Statements, Comment Statements and Exception Handling Statements

Comment Statements

Comment statements are used to prevent the browser from executing certain parts of code that you designate as non-code. The single line comment is just two slashes (//) and the multiple line comment starts with (/*) and ends with (*/).

Exception Handling Statements

These statements are safety mechanisms, so that the code handles common problems that may arise. The try...catch statement tries to execute a piece of code and if it fails, the catch should handle the error gracefully.

Conditional Statements

Java script supports the following conditional control statements:Simple if, If else, If else ladder and Switch

a) Simple if

The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

b) if else statement

The if...else statement is a form of control statement that allows JavaScript to execute statements in more controlled way. If the condition evaluates to true then one block of statements will get executed and if the condition is false other block of statements will get executed.

c) if-else ladder

The if else ladder statement is an advanced form of control statement that allows JavaScript to make correct decision out of several conditions. A normal If Statement must be placed before the use the else If statement. This is because the else if statement is an add- on to the simple if Statement. Any number of else if statements can be included in a program.

If-else ladder

```
<html><script  type="text/javascript">
var a= "first letter";
if(a == “first number”)
{document.write("1 is the first natural number");}
else if(a == "first letter")
{               document.write("a is the first letter”);}
else {          document.write("nothing”);}
</script></html>
```

Output:

a is the first letter

d) Switch statement

The basic syntax of the switch statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

```
<html>
<script type="text/javascript">
var grade='A';
switch (grade)
{ case 'A':document.write("Distinction<br />"); break;
case 'B': document.write("First Class<br />"); break;
case 'C': document.write("Second Class<br />");break;
case 'F':document.write("Failed<br />"); break;
 default: document.write("Did not appear for exams <br />")
}</script></html>
```

Output:

Distinction

Looping Statements

The following are the looping statements in JavaScript:While loop, Do while loop and For loop

a) While loop

The most basic loop in JavaScript is the while loop. There are two key parts to a JavaScript while loop: The conditional statement which must be true for the while loop's code to be executed. The while loop's code that is contained in curly braces "{ and }" will be executed if the condition is True.

When a while loop begins, the JavaScript interpreter checks if the condition statement is true. If it is, the code between the curly braces is executed. The same procedure is repeated until the condition stays true. If the condition statement is always True, then you will never exit the while loop.

b) do-while loop

The do...while loop is similar to the while loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is false.

Do-while

```
<html>
<script type="text/javascript">
var loop= 0;
varlinebreak = "<br />";
do{
            document.write("loop= " + loop);
            document.write(linebreak);
            loop++;
}while(loop<5);
</script></html>
```

Output:

```
loop = 0
loop = 1
loop = 2
loop = 3
loop = 4
loop = 5
```

c) for loop

The for loop is the most compact form of looping and includes the following three important parts: The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.The **test statement** which will test if the given condition is true or not. If condition is true then code given inside the loop will be executed otherwise loop will come out.The **iteration** statement where counter value is incremented or decremented.

For loop

<html><script type="text/javascript">	**Output:**
varlinebreak = "<br />";	**Output:**
loop=0;	Counter = 0
for(i = 0; i < 4; i++)	Counter = 1
{ document.write("Counter = " +i);	Counter = 2
document.write(linebreak);	Counter = 3
}</script></html>	

Break and Continue Statements

The break statement is used to exit a loop early. It breaks the execution of the code from that block.The continue statement tells the interpreter to immediately start the next iteration of the loop and skip remaining code block.When a continue statement is encountered, program flow will move to the loop check expression immediately and if condition remain true then it start next iteration otherwise control comes out of the loop.

Functions in JavaScript

A JavaScript function contains some code that will be executed only by an event or by a call to that function.The function can be called from anywhere within the page or from other external pages.Functions can be defined either <head> or <body>.The most common way to define a function (optional), and a statement block surrounded by curly braces. As a convention, they are typically defined in the <head> section.

***Syntax**: <script type="text/javascript">*

```
Function functionname(parameter-list)
{ statements}
</script>
```

Calling a Function:

The syntax to invoke a function is:

```
<script type="text/javascript">functionname(parameter-list) </script>
```

Functions

<html><head><script type="text/javascript">	**Output:**
function firstfunction()	
{document.write("This is my first function")}	This is my first function

</script></head> *<body><form>* *<input type="button" value="Click me!" onclick="firstfunction()" >* *</form></body></html>*	

Dialog boxes

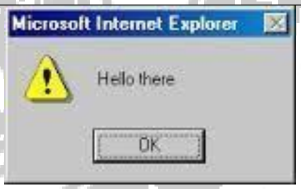
JavaScript supports three types of dialog boxes:Alert dialog box, Prompt dialog box and Confirmation dialog box

Alert dialog box

An alert dialog box is used to give a warning message to the users. It pops up a message box displaying some contents with an OK button. JavaScript alerts are used in the following situations:

- To see a message before doing anything on the website.
- To warn the user about something.
- It can be used as an error indication.

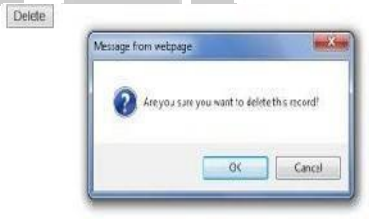
Alert Dialog box

<head> <script type="text/javascript"> alert("Hello there"); </script></head	Microsoft Internet Explorer Hello there OK

Confirmation Dialog Box:

A confirmation dialog box is mostly used to take user's consent on any option. It displays a dialog box with two buttons: OK and Cancel. If the user clicks on OK button the window method confirm() will return true. If the user clicks on the Cancel button confirm() returns false.

Confirmation dialog box

<head><script type="text/javascript"> *varretVal = confirm("Are you sure you want to delete this record ?");* *if(retVal == true){* *alert("User wants to delete!");return true; }*	Delete Message from webpage Are you sure you want to delete this record? OK Cancel

else{ *alert("User does not want to delete!"); return false; }* *</script></head>*	

Prompt Dialog Box:

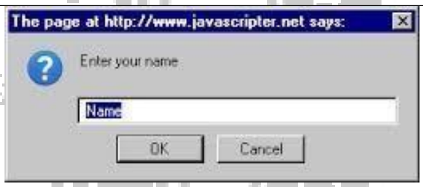
The prompt dialog box is very useful when a pop-up text box to used to get user input. Thus it enables to interact with the user. The user needs to fill in the field and then click OK. This dialog box is displayed using a method called prompt() which takes two parameters:

(i) A label which you want to display in the text box

(ii) A default string to display in the text box.

This dialog box with two buttons: OK and Cancel. If the user clicks on OK button the window method prompt() will return entered value from the text box. If the user clicks on the Cancel button the window method prompt() returns null.

Prompt dialog box

<head> <script type="text/javascript"> varretVal = prompt("Enter your name :”); </script></head>	 The page at http://www.javascripter.net says: Enter your name Name