18ITU404A **SCRIPTING LANGUAGE** **3H – 3C**

Instruction Hours / week: L: 3 T: 0 P: 0 **Marks:** Int : **40** Ext : **60** Total: **100**
End Semester Exam: **3 Hours**

UNIT I - INTRODUCTION TO VBSCRIPT VB Script: Introduction- Embedding VBScript Code in an HTML Document Comments-Variables- Operators-Procedures-Conditional Statements- Looping Constructs - Objects and VBScript – Cookies.

UNIT II - INTRODUCTION TO JAVA SCRIPT

JavaScript- Introduction, simple programming, Obtaining User Input with prompt Dialogs, Operators (arithmetic, Decision making, assignment, logical, increment and decrement. Functions - program modules in JavaScript, programmer defined functions, function definition, Random-number generator, scope rules, global functions, recursion.

UNIT III - FUNCTIONS, ARRAYS AND OBJECTS

JavaScript: Arrays, Objects - Math Object, String Object, Date Object, Boolean & Number Object, document and window Objects. Handling event using java script

UNIT IV - CLIENT SIDE TECHNOLOGIES - AJAX– Evolution of AJAX – AJAX Framework – Web applications with AJAX – AJAX with PHP – AJAX with Databases- Ajax Client Server Architecture-XML Http Request Object-Call Back Methods.

UNIT V - SERVER SIDE SCRIPTING- JSP

Servlet Overview – Life cycle of a Servlet – Handling HTTP request and response – Using Cookies – Session tracking – Java Server Pages – Anatomy of JSP – Implicit JSP Objects – JDBC – Java Beans – Advantages – Enterprise Java Beans – EJB Architecture – Types of Beans – EJB Transactions

Suggested Readings

1. Jeffrey C. Jackson, "Web Technologies--A Computer Science Perspective", Pearson Education, 2006.
2. Deitel, Deitel, Goldberg, “Internet & World Wide Web How to Program”, Third Edition, Pearson Education, 2006.
3. Bryan Basham, Kathy Siegra, Bert Bates, “Head First Servlets and JSP”, Second Edition
4. Uttam K Roy, “Web Technologies”, Oxford University Press, 2011.
5. Robert. W. Sebesta, "Programming the World Wide Web", Fourth Edition, Pearson Education, 2007.
6. Marty Hall and Larry Brown, ‖ Core Web Programming‖ Second Edition, Volume I and II, Pearson Education, 2001.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2018 onwards)

DEPARTMENT OF COMPUTER SCIENCE, COMPUTER APPLICATION & INFORMATION TECHNOLOGY

SUBJECT: SCRIPTING LANGUAGE
SEMESTER: IV
SUBJECT CODE: 18ITU404A **CLASS: II B.Sc. IT**

S.No	Lecture Duration (Period)	Topics to be Covered	Support Materials
		Unit – I	
1	1	Introduction	W1
2	1	Embedding VBScript Code in an HTML Document Comments	W1
3	1	Comments-Variables- Operators	W1
4	1	Procedures	W1
5	1	Conditional Statements- Looping Constructs	W1
6	1	Objects and VBScript	W1
7	1	Cookies	W1
8	1	Recapitulation and Discussion of important questions	
		Total No. of Hours Planned for Unit-I	**08**
		Unit – II	
1	1	JavaScript- Introduction, simple programming	S1:192-199, W2
2	1	Obtaining User Input with prompt Dialogs	W2
3	1	Operators (arithmetic, Decision making, assignment, logical, increment and decrement)	S1:204-208, W2
4	1	Functions - program modules in JavaScript, programmer defined functions, function definition	S1:209-212, W2
5	1	Random-number generator, scope rules	W2
6	1	global functions, recursion	W2
7	1	Recapitulation and Discussion of important questions	
		Total No. of Hours Planned for Unit-II	**07**
		Unit – III	
1	1	JavaScript: Arrays	S1:224-228, W2
2	1	Objects - Math Object, String Object, Date Object	S1:231-233, W2
3	1	Boolean & Number Object, document and window Objects	S1:228-231, W2
4	1	Handling event using java script	W2

5	1	Recapitulation and Discussion of important questions	
		Total No. of Hours Planned for Unit-III	**05**
		Unit – IV	
1	1	Evolution of AJAX – AJAX Framework	S1:373-378, W2
2	1	Web applications with AJAX – AJAX with PHP	W2
3	1	AJAX with Databases- Ajax Client Server Architecture	W2
4	1	XML Http Request Object-Call Back Methods	S2:415-418, W2
5	1	Recapitulation and Discussion of important questions	
		Total No. of Hours Planned for Unit-IV	**05**

		Unit – V	
1	1	Servlet Overview – Life cycle of a Servlet	S1: 307-313, W3
2	1	Handling HTTP request and response	S1: 316-322, W3
3	1	Using Cookies – Session tracking	S1: 322-332, W3
4	1	Java Server Pages – Anatomy of JSP	S1: 433-477
5	1	Implicit JSP Objects	S1:447-450
6	1	JDBC- Java Beans – Advantages	S1:457-463
7	1	Enterprise Java Beans – EJB Architecture- Types of Beans – EJB Transactions	W4
8	1	Recapitulation and Discussion of important questions	
9	1	Pervious ESE Question Paper Discussion	
10	1	Pervious ESE Question Paper Discussion	
11	1	Pervious ESE Question Paper Discussion	
		Total No. of Hours Planned for Unit-V	**11**
		Total No Hours	**36**

Suggested Readings

S1: Jeffrey C. Jackson, "Web Technologies--A Computer Science Perspective", Pearson Education, 2006.
S2: Robert. W. Sebesta, "Programming the World Wide Web", Fourth Edition, Pearson Education, 2007 .

Websites

W1: https://www.tutorialspoint.com/vbscript/index.htm
W2: https://www.w3schools.com/js/
W3: https://www.javatpoint.com/servlet-tutorial
W4: https://www.javatpoint.com/ejb-tutorial

Faculty **HoD**

UNIT-I

Introduction To VB script: Introduction- Embedding VBScript Code in an HTML Document Comments-Variables- Operators-Procedures- Conditional Statements- Looping Constructs - Objects and VBScript – Cookies.

1.1. Introduction

Microsoft VBScript (Visual Basic Script) is a general-purpose, lightweight and active scripting language developed that is modeled on Visual Basic. VBScript is developed by Microsoft with the intention of developing dynamic web pages. It is client-side scripting language like JavaScript. VBScript is a light version of Microsoft Visual Basic. The syntax of VBScript is very similar to that of Visual Basic. If you want our webpage to be more lively and interactive, then you can incorporate VBScript in our code.

VBScript is just a scripting language. So, it cannot run its code on its own. It needs a bigger programming language to host it.

1.2. Embedding VBScript Code in an HTML Document

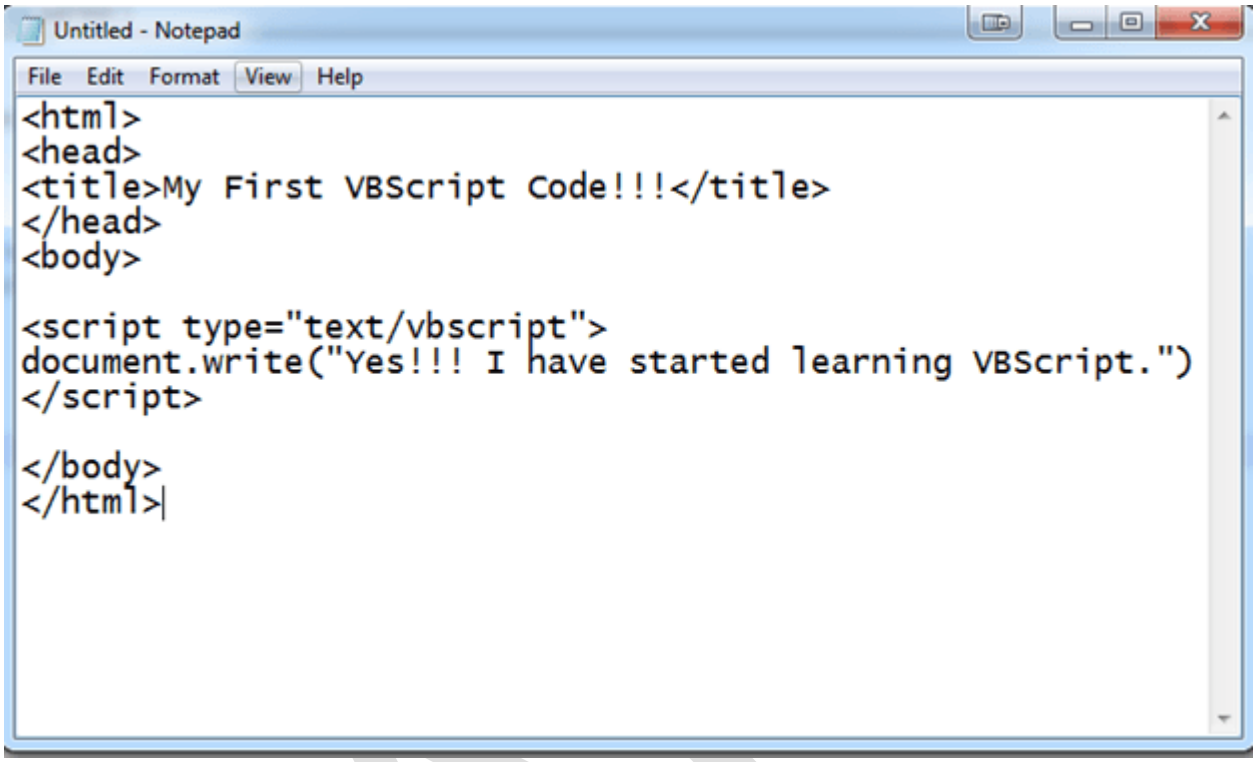
We only need 2 simple tools to create and run VBScript code:

- Web Browser
- Text Editor

In this section, we will embed our VBScript code within a very basic HTML code. This way, we can see VBScript in action by running the particular HTML file on the web browser. Open our text editor (Here, Notepad is used. We can use whichever text editor you want) and add the following lines of code.

```
<html>
<head>
<title>My First VBScript Code!!!</title>
</head>
<body>

<script type="text/vbscript">
document.write("Yes!!! I have started learning VBScript.")
</script>

</body>
</html>
```

Now our text editor will look like this (the appearance and layout could be different based on the text editor you use):



Simple VBscript Program

In this program, the following sections constitute the HTML template.

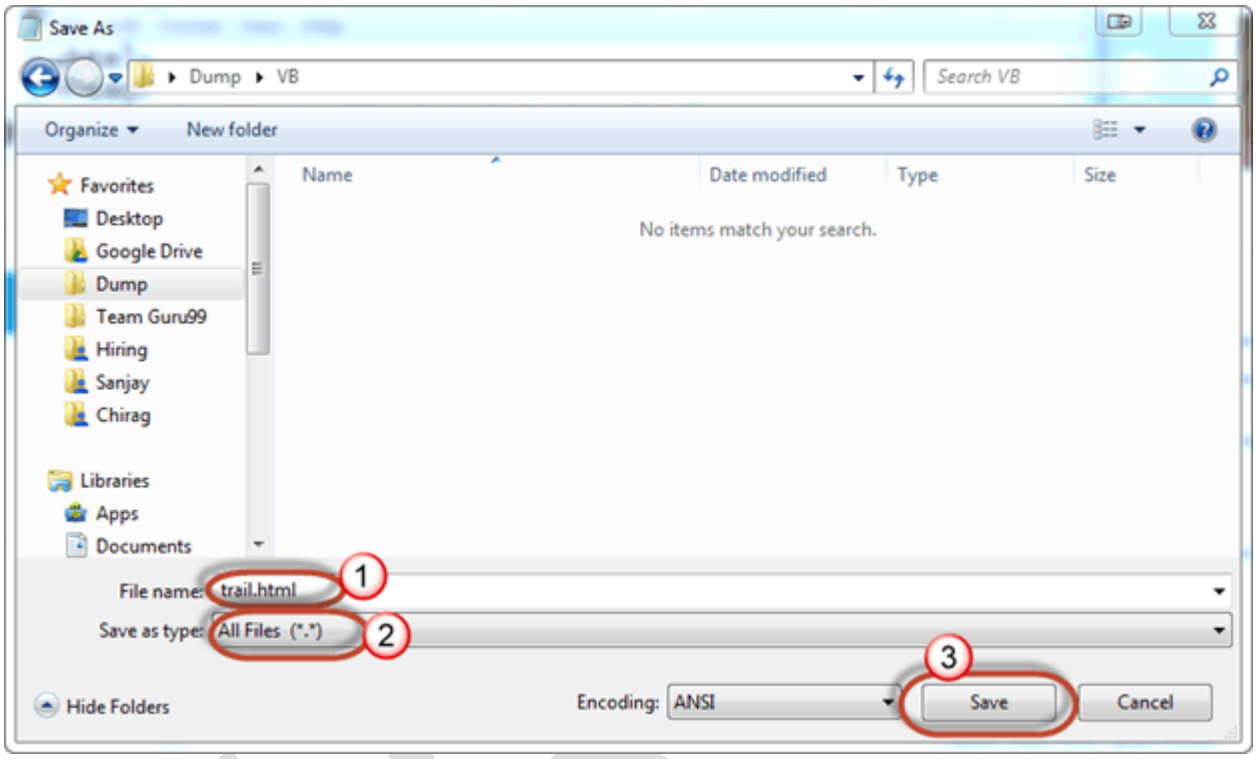
```
<html>
<head>
<title>My First VBScript Code!!!</title>
</head>
<body>

<script type="text/vbscript">
document.write("Yes!!! I have started learning VBScript.")
</script>

</body>
</html>
```

Only the section that starts with <script> comes as part of VB Scripting code.

Whatever string sequence you put in the document.write() will be displayed by browser as page text.

This code will simply output the statement “Yes!!! I have started learning VB Scripting.” on the browser page.

Go to File menu and click” Save” option. Now we will get a window like this:



1. Filename: enter the name as trial.html
2. Save as type: All Files.
3. Click the save button

Click the Save button and we will see the file trial.html in the folder where we have saved our file.

To execute the VB Scripting code we have just created, we need to open the trial.html file in browser.

If we have set any other web browser as our default browser, right-click the file and go to Open With --> Internet Explorer like this:

1.3. Comments

Comments are used to document the program logic and the user information with which other programmers can seamlessly work on the same code in future. It can include information such as developed by, modified by and it can also include incorporated logic. Comments are ignored by the interpreter while execution. Comments in VBScript are denoted by two methods.

1. Any statement that starts with a Single Quote (‘) is treated as comment.

Following is the example −

```
<script language = "vbscript" type = "text/vbscript">
  <!—
    ' This Script is invoked after successful login
    '
    ' Return Value : True / False
  //- >
</script>
```

2. Any statement that starts with the keyword “REM”.

Following is the example −

```
<script language = "vbscript" type = "text/vbscript">
  <!—
    REM This Script is written to Validate the Entered Input
    REM Modified by  : /user2
  //- >
</script>
```

1.4. Variables

A variable is nothing but a space in the computer’s memory that can store certain information. This information is bound to change from time to time. Where the information goes physically is immaterial but when needed, it can be accessed or changed by addressing the name of the variable.

E.g: If there is a statement that we want to run several times, we could use a variable to contain that count. Say X. X is a variable that can be used to store, change and use the space in the memory where we want to keep the count.

All variables are of the data type Variant. Declaring a variable before its use is optional, although it’s a good practice to do so. To make the declaration mandatory there is an “**Option Explicit”** Statement available. To declare variables:

Dim x – This declares x
Dim x, y, z – This declares multiple variables
X=10 – This is how a value is assigned. As a general rule, the variable is the left-hand side component and the right is its value.
X=”Swati” – this is the way a string value is assigned.

To make declarations mandatory this is how the code has to be written:
Option Explicit
Dim x, stri

If Option explicit statement was not used, we could have directly written:
x=100
stri="Swati"
and it would not have thrown an error.

Naming convention: Names must start with an alphabetic character, must be unique, cannot contain an embedded period and cannot exceed 255 chars.

A variable containing a single value is a scalar variable and the one that has more than one is an array. A one dimensional Array can be declared as Dim A(10). All the arrays in VB Script are zero-based that means the array index starts from 0 through the number declared. That means, our array A has 11 elements. Starting from 0 to 10.

To declare a 2-dimensional array simply separate the row count and column count by a comma. Eg: Dim A(5, 3). This means it has 6 rows and 4 columns. The first number is always row and the second a column.

This piece of code shows how we do it. Initially, A is an 11 by 11 array. Then we are resizing it to be an 11 by 21 array and the preserve statement will make sure that the data that is previously contained in the array is not lost.

1.5. Operators

Some of the important operators that are most commonly used are:

Artithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
\	Integer Division (divides two numbers and returns an integer result)
Mod	Modulus (remainder of a division)
^	Exponentiation (raises the number to the power of an exponent)

Assignment Operator

Operator	Description
=	Assign

Comparison Operators

Operator	Description
=	Is equal to
<>	Is not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Logical Operators

Operator	Description
And	Performs a logical conjunction on two expressions (if both expressions evaluate to True, result is True. If either expression evaluates to False, result is False)
Or	Performs a logical disjunction on two expressions (if either or both expressions evaluate to True, result is True).
Not	Performs logical negation on an expression.
Xor	Performs a logical exclusion on two expressions (if one, and only one, of the expressions evaluates to True, result is True. However, if either expression is Null, result is also Null).

Concatenation Operators

Operator	Description
&	Concatenate (join two strings together)
+	Adds two numbers together

The operator precedence rules are:

1. Multiplication or Division take precedence over addition or subtraction
2. If multiplication and division exist in the same expression, then left to right order is considered
3. If Addition and subtraction occur in the same expression, then too, left and right order is taken into consideration.
4. The order can be overridden by using parenthesis. In this case, the expression within the parenthesis is executed first.

5. & operator takes precedence after all arithmetic operators and before all logical operators.

Data Types in VBScript

Unlike other languages, VBScript has only 1 data type called Variant. As this is the only data type that is used in VBScript, it's the only data type that is returned by all the functions in the VBScript.

A variant data type can contain different kinds of information, depending on how it is used. For Example, if we use this data type in String context then this will behave like a String and if we use this in the Numeric context then this will behave like a Number. This is the specialty of a Variant data type.

A Variant data type can contain several subtypes. Now, let's take a look at what all values/data will be returned if a particular subtype is used.

Subtypes include:

#1) Empty: This subtype indicates that the value will be 0 in case of Numeric Variables and "" for String Variables.

#2) Null: This subtype indicates that there is no valid data.

#3) Boolean: This subtype indicates that the resultant value will be either true or false.

#4) Byte: This subtype exhibits that the resultant value will lie in the range between 0 to 255 i.e. the result will be from any value ranging from 0 to 255.

#5) Integer: This subtype shows that the resultant value will lie in the range between -32768 to 32767 i.e. the result will be from any value ranging from -32768 to 32767

#6) Currency: This subtype indicates that the resultant value will lie in the range between -922,337,203,685,477.5808 to 922,337,203,685,477.5807 i.e. the result will be from any value ranging from -327-922,337,203,685,477.5808 to 922,337,203,685,477.5807.

#7) Long: This subtype shows that the resultant value will lie in the range from -2,147,483,648 to 2,147,483,647 i.e. result will be from any value in between -2,147,483,648 to 2,147,483,647.

#8) Single: This subtype exhibits that the resultant value will be from any value in between -3.402823E38 to -1.401298E-45 in case of negative values.

And for positive values, the result will be from any value in between 1.401298E-45 to 3.402823E38.

#9) Double: This subtype indicates that the resultant value will be from any value in between -1.79769313486232E308 to 4.94065645841247E-324 in case of negative values.

And for positive values, the result will be from any value in between 4.94065645841247E-324 to 1.79769313486232E308.

#10) Date (Time): This subtype will return a number which will represent a date value in between January 1, 100 to December 31, 9999

#11) String: This subtype will return a variable-length string value which can approximately be up to 2 billion characters in length.

#12) Object: This subtype will return an object.

#13) Error: This subtype will return an error number.

1.6. Procedures

We have two kinds of procedures: The Sub procedure and the Function procedure.

A Sub procedure:

- is a series of statements, enclosed by the Sub and End Sub statements
- can perform actions, but **does not return** a value
- can take arguments that are passed to it by a calling procedure
- without arguments, must include an empty set of parentheses ()

```
Sub mysub()
 some statements
End Sub

or

Sub mysub(argument1,argument2)
 some statements
End Sub
```

A Function procedure:

- is a series of statements, enclosed by the Function and End Function statements
- can perform actions and **can return** a value
- can take arguments that are passed to it by a calling procedure
- without arguments, must include an empty set of parentheses ()
- returns a value by assigning a value to its name

```
Function myfunction()
 some statements
 myfunction=some value
End Function

or

Function myfunction(argument1,argument2)
 some statements
 myfunction=some value
End Function
```

Call a Sub or Function Procedure

When you call a Function in your code, you do like this:

```
name = findname()
```

Here you call a Function called "findname", the Function returns a value that will be stored in the variable "name".

Or, you can do like this:

```
msgbox "Your name is " & findname()
```

Here you also call a Function called "findname", the Function returns a value that will be displayed in the message box.

When you call a Sub procedure you can use the Call statement, like this:

```
Call MyProc(argument)
```

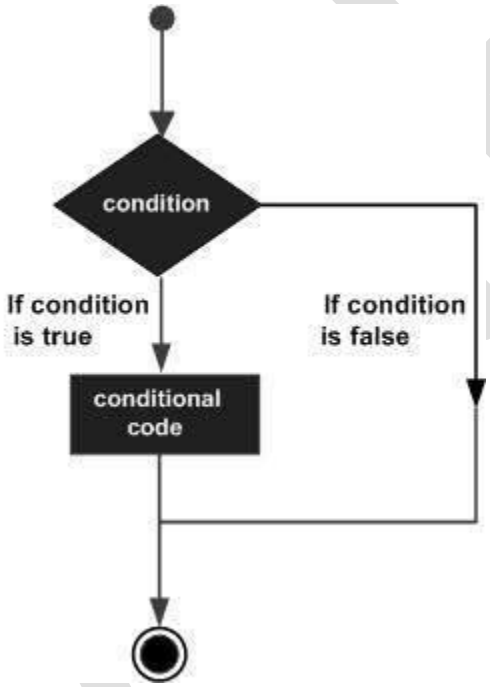
Or, you can omit the Call statement, like this:

MyProc argument

1.7. Conditional Statements

Decision making allows programmers to control the execution flow of a script or one of its sections. The execution is governed by one or more conditional statements.

Following is the general form of a typical decision making structure found in most of the programming languages −



VBScript provides the following types of decision making statements.

Statement	Description
if statement	An **if** statement consists of a Boolean expression followed by one or more statements.
if..else statement	An **if else** statement consists of a Boolean expression followed by one or more statements. If the condition is True, the statements under the **If** statements are executed. If the condition is false, then the **Else** part of the script is Executed
if...elseif..else statement	An **if** statement followed by one or more **ElseIf** Statements, that consists of Boolean expressions and then followed by an

	optional else statement, which executes when all the condition becomes false.
nested if statements	An **if** or **elseif** statement inside another **if** or **elseif** statement(s).
switch statement	A **switch** statement allows a variable to be tested for equality against a list of values.

If....Then.....Else

You should use the If...Then...Else statement if you want to

- execute some code if a condition is true
- select one of two blocks of code to execute

If you want to execute only **one** statement when a condition is true, you can write the code on one line:

```
if i=10 Then msgbox "Hello"
```

There is no ..else.. in this syntax. You just tell the code to perform **one action** if the condition is true (in this case if i=10).

If you want to execute **more than one** statement when a condition is true, you must put each statement on separate lines and end the statement with the keyword "End If":

```
if i=10 Then
  msgbox "Hello"
  i = i+1
end If
```

There is no ..else.. in this syntax either. You just tell the code to perform **multiple actions** if the condition is true.

If you want to execute a statement if a condition is true and execute another statement if the condition is not true, you must add the "Else" keyword:

```
if i=10 then
  msgbox "Hello"
else
  msgbox "Goodbye"
end If
```

The first block of code will be executed if the condition is true, and the other block will be executed otherwise (if i is not equal to 10).

If....Then.....Elseif

You can use the if...then...elseif statement if you want to select one of many blocks of code to execute:

```
if payment="Cash" then
  msgbox "You are going to pay cash!"
elseif payment="Visa" then
  msgbox "You are going to pay with visa."
elseif payment="AmEx" then
  msgbox "You are going to pay with American Express."
else
  msgbox "Unknown method of payment."
end If
```

Select Case

You can also use the SELECT statement if you want to select one of many blocks of code to execute:

```
select case payment
case "Cash"
  msgbox "You are going to pay cash"
case "Visa"
  msgbox "You are going to pay with visa"
case "AmEx"
  msgbox "You are going to pay with American Express"
case Else
  msgbox "Unknown method of payment"
end select
```

This is how it works: First we have a single expression (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each Case in the structure. If there is a match, the block of code associated with that Case is executed.

1.8. Looping Constructs

Very often when you write code, you want to allow the same block of code to run a number of times. You can use looping statements in your code to do this.

In VBScript we have four looping statements:

- **For...Next statement** - runs statements a specified number of times.
- **For Each...Next statement** - runs statements for each item in a collection or each element of an array
- **Do...Loop statement** - loops while or until a condition is true
- **While...Wend statement** - Do not use it - use the Do...Loop statement instead

---

For...Next Loop

You can use a **For...Next** statement to run a block of code, when you know how many repetitions you want.

You can use a counter variable that increases or decreases with each repetition of the loop, like this:

```
For i=1 to 10
  some code
Next
```

The **For** statement specifies the counter variable (**i**) and its start and end values. The **Next** statement increases the counter variable (**i**) by one.

Step Keyword

Using the **Step** keyword, you can increase or decrease the counter variable by the value you specify.

In the example below, the counter variable (**i**) is increased by two each time the loop repeats.

```
For i=2 To 10 Step 2
  some code
Next
```

To decrease the counter variable, you must use a negative **Step** value. You must specify an end value that is less than the start value.

In the example below, the counter variable (**i**) is decreased by two each time the loop repeats.

```
For i=10 To 2 Step -2
  some code
Next
```

Exit a For...Next

You can exit a For...Next statement with the Exit For keyword.

For Each...Next Loop

A **For Each...Next** loop repeats a block of code for each item in a collection, or for each element of an array.

```
dim cars(2)
cars(0)="Volvo"
cars(1)="Saab"
cars(2)="BMW"

For Each x in cars
  document.write(x & "<br />")
Next
```

Do...Loop

You can use Do...Loop statements to run a block of code when you do not know how many repetitions you want. The block of code is repeated while a condition is true or until a condition becomes true.

Repeating Code While a Condition is True

You use the While keyword to check a condition in a Do...Loop statement.

```
Do While i>10
  some code
Loop
```

If **i** equals 9, the code inside the loop above will never be executed.

```
Do
  some code
Loop While i>10
```

The code inside this loop will be executed at least one time, even if **i** is less than 10.

Repeating Code Until a Condition Becomes True

You use the Until keyword to check a condition in a Do...Loop statement.

```
Do Until i=10
  some code
Loop
```

If **i** equals 10, the code inside the loop will never be executed.

```
Do
  some code
Loop Until i=10
```

The code inside this loop will be executed at least one time, even if **i** is equal to 10.

Exit a Do...Loop

You can exit a Do...Loop statement with the Exit Do keyword.

```
Do Until i=10
  i=i-1
  If i<10 Then Exit Do
Loop
```

The code inside this loop will be executed as long as **i** is different from 10, and as long as **i** is greater than 10.

1.9. Objects and VBScript

VBScript runtime objects help us to accomplish various tasks. This section will help you understand how to instantiate an object and work with it.

Syntax

In order to work with objects seamlessly, we need to declare the object and instantiate it using **Set** Keyword.

```
Dim objectname    'Declare the object name
Set objectname = CreateObject(object_type)
```

Example

In the below example, we are creating an object of type **Scripting.Dictionary**.

```
Dim obj
Set obj = CreateObject("Scripting.Dictionary")
```

Destroying the Objects

The significance of destroying the Object is to free the memory and reset the object variable.

Syntax

In order to destroy the objects, we need to use **Set** Keyword followed by the object name and point it to **Nothing**.

```
Set objectname = Nothing 'Destroy the object.
```

Example

In the below example, we are creating an object of type **Scripting.Dictionary**.

```
Dim obj
Set obj = CreateObject("Scripting.Dictionary")
Set obj = Nothing.
```

Object Usage

Please click on each one of the given object types to know more.

Object Type	Description
Class	Class is a container, which holds methods and variables associated with it and accessed by creating an object of Type Class.
Scripting.FileSystemObject	It is the group of objects with which we can work with file system.
Scripting.Dictionary	A Group of objects, which are used for creating the dictionary objects.
Debug	A Global Object with which we can send output to the Microsoft script debugger.

1.10. Cookies

Web Browsers and Servers use HTTP protocol to communicate and HTTP is a stateless protocol. But for a commercial website, it is required to maintain session information among different pages. For example, one user registration ends after completing many pages. But how to maintain user's session information across all the web pages. In many situations, using cookies is the most efficient method of remembering and tracking preferences, purchases, commissions and other information required for better visitor experience or site statistics.

How It Works?

Your server sends some data to the visitor's browser in the form of a cookie. The browser may accept the cookie. If it does, it is stored as a plain text record on the visitor's hard drive. Now, when the visitor arrives at another page on your site, the browser sends the same cookie to the server for retrieval. Once retrieved, your server knows/remembers what was stored earlier. Cookies are a plain text data record of 5 variable-length fields −

- **Expires** − The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.
- **Domain** − The domain name of your site.
- **Path** − The path to the directory or web page that set the cookie. This may be blank if you want to retrieve the cookie from any directory or page.
- **Secure** − If this field contains the word "secure", then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists.
- **Name=Value** − Cookies are set and retrieved in the form of key and value pairs.

Cookies were originally designed for CGI programming and cookies' data is automatically transmitted between the web browser and web server, so CGI scripts on the server can read and write cookie values that are stored on the client.

VBScript can also manipulate cookies using the cookie property of the *Document* object. VBScript can read, create, modify and delete the cookie or cookies that apply to the current web page.

Storing Cookies

The simplest way to create a cookie is to assign a string value to the *document.cookie* object, which looks like this −

Syntax

```
document.cookie = "key1 = value1;key2 = value2;expires = date"
```

Here *expires* attribute is optional. If you provide this attribute with a valid date or time, then cookie will expire at the given date or time and after that cookies' value will not be accessible.

Possible Questions:

(2 Marks)

1. What are comments in VB Script?
2. What is the difference between Sub Procedure and Function Procedure in VB Script?
3. What are Cookies?
4. What is Object in VB Script?

(8 Marks)

1. Explain about variables in VB Script.
2. Describe about various operators .in VB Script.
3. Explain about conditional statements in VB Script.
4. Discuss about looping constructs in VB Script.
5. Explain about procedures and functions in VB Script.
6. Explain about Objects and VB Script.
7. Explain in detail about Cookies

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

SUBJECT : Scripting Language **Unit I**

S No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	Declaring a variable before its use is optional, although it's a good practice to do so and to make the declaration mandatory there is an _____ available	"Exit" Statement	"Exit Do" Statement	"Option Explicit" Statement	"Explicit" Statement	"Option Explicit" Statement
2	Which one is not a Naming Convention in VB Script?	names must start with an alphabetic character	must be unique	cannot contain an embedded period	cannot exceed 145 chars	cannot exceed 145 chars
3	In many situations, using _____ is the most efficient method of remembering and tracking preferences, purchases, commissions and other information required for method of remembering and tracking preferences, purchases, commissions and other information visitor experience or site statistics.	Methods	Functions	Procedures	cookies	cookies

4	Which one is not relevant to VB Script?	VBScript (Visual Basic Script) is a general-purpose, lightweight and active scripting language developed that is modeled on Visual Basic.	It is client-side scripting language like JavaScript.	VBScript is developed by Microsoft with the intention of developing dynamic web pages.	VBScript is just a scripting language and so, it can run its code on its own.	VBScript is just a scripting language and so, it can run its code on its own.
5	All variables in VB Script are of the data type _____.	Static	Dynamic	Variant	Non of a), b) and c)	Variant
6	The following which arew not relevant to a Sub procedure is :	is a series of statements, enclosed by the Sub and End Sub statements	can perform actions, but does return a value	can take arguments that are passed to it by a calling procedure	without arguments, must include an empty set of parentheses ()	can perform actions, but does return a value
7	The following which arew not relevant to a Function procedure is :	is a series of statements, enclosed by the Function and End Function statements	returns a value by assigning a value to its name	without arguments, must include an empty set of parentheses ()	can perform actions, but does not return a value	can perform actions, but does not return a value
8	Conditional statements available in VB Script are	If Else Statement	Switch Statement	Both a) and b)	Non of a) and b)	Both a) and b)
9	We can use a _____ to run a block of code, when we know how many repetitions you want.	For Each...Next statement	Do...Loop statement	While...Wend statement	For...Next statement	For...Next statement
10	We can exit a Do...Loop statement with the_____ .	Exit keyword	Exit Do keyword			

11	Which one is not relevant with cookies?	VBScript cannot read, create, modify and delete the cookie or cookies that apply to the current web page.	VBScript can also manipulate cookies using the cookie property of the Document object.	Cookies were originally designed for CGI programming and cookies' data is automatically transmitted between the web browser and web server, so CGI scripts on the server can read and write cookie values that are stored on the client.	Cookies are a plain text data record of 5 variable-length fields	VBScript cannot read, create, modify and delete the cookie or cookies that apply to the current web page.
12	Which of the following keyword is used to declare a variable in VBScript?	Variant	Var	Dim	Non of the above	Dim
13	Which of the following operator can be used to check if two numbers are equal or not in VBScript?	!=	<>	not	None of the above	<>
14	What is the output of A & B in VBScript if A = 5 and B = 10?	15	510	5	None of these	510
15	Which loop is used to iterate till a condition becomes true?	For Next loop	For Each Next loop	Do While loop	Do Until loop	Do Until loop
16	The command used in VB Script for writing some text on a page is	document.write().	Msgbox	A and B are correct.	None of these	document.write().

17	Why is a scripting language called a scripting language ?	Scripting languages, like JavaScript and VBScript, are designed as an extension to XML.	Scripting languages, like JavaScript and VBScript, are designed as an extension to XHTML.	Scripting languages, like JavaScript and VBScript, are designed as an extension to HTML.	Scripting languages, like JavaScript and VBScript, are designed as an extension to DHTML.	Scripting languages, like JavaScript and VBScript, are designed as an extension to HTML.
18	What does the Option Explicit directive do ?	It forces you to declare all of your variables.	It forces you to call the method.	It forces you to declare all of your static variables	It forces you to declare all of your global variables	It forces you to declare all of your variables.
19	VBScript is developed by	Netscape	Opera.	Sun	Microsoft	Microsoft
20	Where to Put the VBScript code	Head section.	Body Section.	Both body and head section.	None of these.	Both body and head section.
21	Two kinds of procedures in VBScript	Top procedure and bottom procedure	The Sub procedure and the Function procedure.	A and B are correct	None of these	The Sub procedure and the Function procedure.
22	Which of the following function of VBScript converts a given number of any variant subtype to Long?	CDbl	CInt	CLng	CSng	CLng
23	How will you get the absolute value of the given number in VBScript?	Using Abs function	Using Exp function	Using InStr function	Using InStrRev function	Using Abs function
24	How will you trim the spaces on the right of a string using VBScript?	Using Lcase function	Using Ucase function	Using Ltrim function	Using Rtrim function	Using Rtrim function
25	How will you get a combined string from array of string in VBScript?	Using Join function	Using Filter function	Using IsArray function	Using Erase Function	Using Join function
26	How to call a function in VBScript?	Using Call keyword.	Using Function keyword	Using function name	None of the above.	Using Call keyword.

27	Which of the following is correct about classes in VBScript?	VBScript classes are enclosed within Class .... End Class	Classes can contain variables, which can be of private or public.	Variables within classes should follow VBScript naming conventions.	All of the above.	All of the above.
28	The concatanation opeerators in VB Script are	&	+	Both of the above.	None of the above.	Both of the above.
29	VBScript is an	Active Scripting Language	Passive Scripting Language	Both Active and Passive Scripting Language	None of the above.	Active Scripting Language
30	Which function is used to compare two strings in VBScript?	StrCompare	StrComp	CompareStr	Compare	StrComp
31	Which function is used to get a subset of a array in VBScript?	SubString	Subset	Filter	ArraySubset	Filter
32	Which function is used to get the largest subscript of an array in VBScript?	LBound	UBound	SBound	BBound	UBound
33	Which function is used to get the smallest subscript of an array in VBScript?	LBound	UBound	SBound	BBound	LBound
34	What is the use of Trim() function in VBScript?	Remove both leading and trailing spaces from a string	Creates a string with the specified number of spaces	Crops a string	Determine the length of a string	Remove both leading and trailing spaces from a string
35	Which function is used to reverse String in VBscript?	reverse()	rev()	StrReverse()	StrRev()	StrReverse()
36	Which function is used to split a string into an array in VBScript?	ArraySplit	Split	SplitArray	SplitString	Split
37	What is the use of Spaces() function in VBScript?	Remove both leading and trailing spaces from a string	Creates a string with the specified number of spaces	Crops a string	Determine the length of a string	Creates a string with the specified number of spaces

38	What is the use of InStr() function in VBScript?	Returns ANSI Character Code	Returns Character from ANSI Code	Find a string within another	Crops a string from left	Find a string within another
39	VBScript is based on?	Java	Javascript	Visual Basic	Visual c++	Visual Basic
40	What is correct way of creating functions in VBScript?	MyFunction add(){...statements...}	Function add() {...statements...}	MyFunction add() ....statements....End Function	Function add() ....statements....End Function	Function add() ....statements....End Function
41	How many data tyes does VBScript have?	1	4	8	12	1
42	_____indicates whether a particular condition is on or off.	Text Box	Check Box	List Box	Combo Box	Check Box
43	_____statement enables us to trap runtime error.	Error	Runtime Error	On Runtime Error	None of the above	Runtime Error
44	Frame control acts as a _____	Event	Method	Class	Container	Container
45	_____method removes a dialog box from view.	Enabled	Disable	Hide	View	Hide
46	The_____ is a tool used for both input and output purpose.	Label	Text Box	Combo Box	Command Button	Text Box
47	The_____ function returns a string stored in a variant data type.	Chr	Str	Char	None of these	Chr
48	_____property is used to hide the content in text box with some symbols.	Name	Caption	Hidden	Password	Password
49	_____method is used to retrieve the stored text from the clipboard.	Gettext	Gettext	Addtext	Settext	Gettext
50	_____control displays current directory with any sub directories and allows the user to change the directory.	File list box	Drive list box	Directory list box	All of the above	Directory list box
51	_____method is used to forcibly set the CPU focus to a particular control.	Setfocus	Gotfocus	Lostfocus	None of the above	Setfocus

52	The variables that does not change the value during execution of program is_____	Numeric	String	Constant	None of the above	Constant
53	The function procedures are _____ by default.	Public	Private	Protected	None of the above	Public
54	_____is a data type that can be used to declare a text of maximum 10 million characters.	Date	Ulong	Numeric	String	String
55	Variables are named storage locations in memory, the value of which does not change during program _____	Debug	Design	Execution	All of the above	Execution
56	Is scripting languages are case sensitive?	VB Script is case sensitive	Java Script is case insenitive	Both a) and b) are true	Both a) and b) are false	Both a) and b) are true
57	Which VBScript function converts an input string to all lowercase?	LCase	LowerCase	Lower	There is no such function to directly convert to lowercase	LCase
58	VBScript was launched in	1995	1996	1994	1997	1996
59	Which browser has built-in support for executing VBScript?	Internet Explorer	Mozilla Firefox	Opera	None of these	Internet Explorer
60	In the Select Case statement, which case is used for unknown cases?	Else	Default	Unknown	Not	Else
61	How to take request using get method?	Using Request.form	Using Response.write	Using Request.Querystring	Using Request.Servarvariables	Using Request.Querystring

UNIT-II

Introduction To Java Script : JavaScript- Introduction, simple programming, Obtaining User Input with prompt Dialogs, Operators (arithmetic, Decision making, assignment, logical, increment and decrement. Functions - program modules in JavaScript, programmer defined functions, function definition, Random-number generator, scope rules, global functions, recursion.

1.1. Introduction

JavaScript is a very powerful client-side scripting language. JavaScript is used mainly for enhancing the interaction of a user with the webpage. In other words, you can make your webpage more lively and interactive, with the help of JavaScript. JavaScript is also being used widely in game development and Mobile application development.

JavaScript was developed by Brendan Eich in 1995, which appeared in Netscape, a popular browser of that time. The language was initially called LiveScript and was later renamed JavaScript. There are many programmers who think that JavaScript and Java are the same. In fact, JavaScript and Java are very much unrelated. Java is a very complex programming language whereas JavaScript is only a scripting language. The syntax of JavaScript is mostly influenced by the programming language C.

Being a scripting language, JavaScript cannot run on its own. In fact, the browser is responsible for running JavaScript code. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it is up to the browser to execute it. The main advantage of JavaScript is that all modern web browsers support JavaScript. So, you do not have to worry about whether your site visitor uses Internet Explorer, Google Chrome, Firefox or any other browser. JavaScript will be supported. Also, JavaScript runs on any operating system including Windows, Linux or Mac. Thus, JavaScript overcomes the main disadvantages of VBScript (Now deprecated) which is limited to just IE and Windows.

1.2 Simple programming

You should place all your JavaScript code within <script> tags (<script> and </script>) if you are keeping your JavaScript code within the HTML document itself. This helps your browser distinguish your JavaScript code from the rest of the code. As there are other client-side scripting languages (Example: VBScript), it is highly recommended that you specify the scripting language you use. You have to use the type attribute within the <script> tag and set its value to text/javascript like this:

```
<html>
<head>
        <title>My First JavaScript code!!!</title>
        <script type="text/javascript">
                alert("Hello World!");
        </script>
</head>
```

```
<body>
</body>
</html>
```

1.3. Obtaining User Input with prompt Dialogs

Our next script builds on prior scripts to create a dynamic welcome page that obtains the user's name, then displays it on the page. The script uses another predefined dialog box from the window object—a prompt dialog—which allows the user to input a value that the script can use. The program asks the user to input a name, then displays the name in the XHTML document.

Line 12 is a declaration that contains the JavaScript keyword var. Keywords are words that have special meaning in JavaScript. The keyword var at the beginning of the statement indicates that the word name is a variable. A variable is a location in the com-puter's memory where a value can be stored for use by a program. All variables have a name, type and value, and should be declared with a var statement before they are used in

```
<!-- Prompt box used on a welcome screen. -->
 <html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<title>Using Prompt and Alert Boxes</title>
 <script type = "text/javascript">
 <!--
 var name; // string entered by the user
 // read the name from the prompt box as a string
 name = window.prompt( "Please enter your name" );
        document.writeln( "<h1>Hello, " + name +
", welcome to JavaScript programming!</h1>" );
// -->

</script>
```

```
</head>
<body>
<p>Click Refresh (or Reload) to run this script again.</p>
 </body>
</html>
```

Our next script illustrates another use of prompt dialogs to obtain input from the user. Figure 6.9 inputs two integers (whole numbers, such as 7, –11, 0 and 31914) typed by a user at the keyboard, computes the sum of the values and displays the result.

Lines 12–16 declare the variables firstNumber, secondNumber, number1, number2 and sum. Single-line comments state the purpose of each of these variables. Line 19 employs a prompt dialog to allow the user to enter a string representing the first of the two

```
<head>
<title>An Addition Program</title>
 <script type = "text/javascript">
 <!--
 var firstNumber; // first string entered by user
 var secondNumber; // second string entered by user
 var number1; // first number to add
 var number2; // second number to add
 var sum; // sum of number1 and number2
 // read in first number from user as a string
 firstNumber = window.prompt( "Enter first integer" );
 // read in second number from user as a string
 secondNumber = window.prompt( "Enter second integer" );
 // convert numbers from strings to integers
 number1 = parseInt( firstNumber );
 number2 = parseInt( secondNumber );
       sum = number1 + number2; // add the numbers
 // display the results
 document.writeln( "<h1>The sum is " + sum + "</h1>" );

 // -->
 </script>
 </head>
```

```
    <body>
    <p>Click Refresh (or Reload) to run the script again</p>
    </body>
</html>
```

1.4. Operators

JavaScript supports the following types of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Lets have a look on all operators one by one.

Arithmetic Operators

JavaScript supports the following arithmetic operators −

Assume variable A holds 10 and variable B holds 20, then −

Sr.No.	Operator & Description
	+ (Addition)
1	Adds two operands **Ex:** A + B will give 30
	- (Subtraction)
2	Subtracts the second operand from the first **Ex:** A - B will give -10
	*** (Multiplication)**
3	Multiply both operands **Ex:** A * B will give 200
	/ (Division)
4	Divide the numerator by the denominator

Ex: B / A will give 2

% (Modulus)

5 Outputs the remainder of an integer division

Ex: B % A will give 0

++ (Increment)

6 Increases an integer value by one

Ex: A++ will give 11

-- (Decrement)

7 Decreases an integer value by one

Ex: A-- will give 9

Note − Addition operator (+) works for Numeric as well as Strings. e.g. "a" + 10 will give "a10".

Example

The following code shows how to use arithmetic operators in JavaScript.

```
<html>
  <body>

    <script type = "text/javascript">
      <!--
        var a = 33;
        var b = 10;
        var c = "Test";
        var linebreak = "<br />";

        document.write("a + b = ");
        result = a + b;
        document.write(result);
        document.write(linebreak);

        document.write("a - b = ");
        result = a - b;
        document.write(result);
        document.write(linebreak);
```

```
        document.write("a / b = ");
        result = a / b;
        document.write(result);
        document.write(linebreak);

        document.write("a % b = ");
        result = a % b;
        document.write(result);
        document.write(linebreak);

        document.write("a + b + c = ");
        result = a + b + c;
        document.write(result);
        document.write(linebreak);

        a = ++a;
        document.write("++a = ");
        result = ++a;
        document.write(result);
        document.write(linebreak);

        b = --b;
        document.write("--b = ");
        result = --b;
        document.write(result);
        document.write(linebreak);
      //-->
    </script>

    Set the variables to different values and then try...
  </body>
</html>
```

Output

```
a + b = 43
a - b = 23
a / b = 3.3
a % b = 3
a + b + c = 43Test
++a = 35
--b = 8
Set the variables to different values and then try...
```

Comparison Operators

JavaScript supports the following comparison operators −

Assume variable A holds 10 and variable B holds 20, then −

Sr.No.	Operator & Description
	= = (Equal)
1	Checks if the value of two operands are equal or not, if yes, then the condition becomes true. **Ex:** (A == B) is not true.
	!= (Not Equal)
2	Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. **Ex:** (A != B) is true.
	> (Greater than)
3	Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true. **Ex:** (A > B) is not true.
	< (Less than)
4	Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true. **Ex:** (A < B) is true.
	>= (Greater than or Equal to)
5	Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true. **Ex:** (A >= B) is not true.
	<= (Less than or Equal to)
6	Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true. **Ex:** (A <= B) is true.

Example

The following code shows how to use comparison operators in JavaScript.

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var a = 10;
        var b = 20;
        var linebreak = "<br />";

        document.write("(a == b) => ");
        result = (a == b);
        document.write(result);
        document.write(linebreak);

        document.write("(a < b) => ");
        result = (a < b);
        document.write(result);
        document.write(linebreak);

        document.write("(a > b) => ");
        result = (a > b);
        document.write(result);
        document.write(linebreak);

        document.write("(a != b) => ");
        result = (a != b);
        document.write(result);
        document.write(linebreak);

        document.write("(a >= b) => ");
        result = (a >= b);
        document.write(result);
        document.write(linebreak);

        document.write("(a <= b) => ");
        result = (a <= b);
        document.write(result);
        document.write(linebreak);
      //-->
    </script>
    Set the variables to different values and different operators and then try...
```

```
  </body>
</html>
```

Output

```
(a == b) => false
(a < b) => true
(a > b) => false
(a != b) => true
(a >= b) => false
a <= b) => true
```

Set the variables to different values and different operators and then try...

Logical Operators

JavaScript supports the following logical operators −

Assume variable A holds 10 and variable B holds 20, then −

Sr.No.	Operator & Description
	&& (Logical AND)
1	If both the operands are non-zero, then the condition becomes true. **Ex:** (A && B) is true.
	|| (Logical OR)
2	If any of the two operands are non-zero, then the condition becomes true. **Ex:** (A || B) is true.
	! (Logical NOT)
3	Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. **Ex:** ! (A && B) is false.

Example

Try the following code to learn how to implement Logical Operators in JavaScript.

```
<html>
  <body>
```

```
<script type = "text/javascript">
  <!--
     var a = true;
     var b = false;
     var linebreak = "<br />";

     document.write("(a && b) => ");
     result = (a && b);
     document.write(result);
     document.write(linebreak);

     document.write("(a || b) => ");
     result = (a || b);
     document.write(result);
     document.write(linebreak);

     document.write("!(a && b) => ");
     result = (!(a && b));
     document.write(result);
     document.write(linebreak);
  //-->
</script>
<p>Set the variables to different values and different operators and then try...</p>
</body>
</html>
```

Output

(a && b) => false
(a || b) => true
!(a && b) => true
Set the variables to different values and different operators and then try...

Bitwise Operators

JavaScript supports the following bitwise operators −

Assume variable A holds 2 and variable B holds 3, then −

Sr.No.	Operator & Description
1	**& (Bitwise AND)**

It performs a Boolean AND operation on each bit of its integer arguments.

Ex: (A & B) is 2.

| (BitWise OR)

2 It performs a Boolean OR operation on each bit of its integer arguments.

Ex: (A | B) is 3.

^ (Bitwise XOR)

3 It performs a Boolean exclusive OR operation on each bit of its integer arguments. Exclusive OR means that either operand one is true or operand two is true, but not both.

Ex: (A ^ B) is 1.

~ (Bitwise Not)

4 It is a unary operator and operates by reversing all the bits in the operand.

Ex: (~B) is -4.

<< (Left Shift)

5 It moves all the bits in its first operand to the left by the number of places specified in the second operand. New bits are filled with zeros. Shifting a value left by one position is equivalent to multiplying it by 2, shifting two positions is equivalent to multiplying by 4, and so on.

Ex: (A << 1) is 4.

>> (Right Shift)

6 Binary Right Shift Operator. The left operand's value is moved right by the number of bits specified by the right operand.

Ex: (A >> 1) is 1.

>>> (Right shift with Zero)

7 This operator is just like the >> operator, except that the bits shifted in on the left are always zero.

Ex: (A >>> 1) is 1.

Example

Try the following code to implement Bitwise operator in JavaScript.

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var a = 2; // Bit presentation 10
        var b = 3; // Bit presentation 11
        var linebreak = "<br />";

        document.write("(a & b) => ");
        result = (a & b);
        document.write(result);
        document.write(linebreak);

        document.write("(a | b) => ");
        result = (a | b);
        document.write(result);
        document.write(linebreak);

        document.write("(a ^ b) => ");
        result = (a ^ b);
        document.write(result);
        document.write(linebreak);

        document.write("(~b) => ");
        result = (~b);
        document.write(result);
        document.write(linebreak);

        document.write("(a << b) => ");
        result = (a << b);
        document.write(result);
        document.write(linebreak);

        document.write("(a >> b) => ");
        result = (a >> b);
        document.write(result);
        document.write(linebreak);
      //-->
    </script>
    <p>Set the variables to different values and different operators and then try...</p>
  </body>
</html>
```

(a & b) => 2
(a | b) => 3
(a ^ b) => 1
(~b) => -4
(a << b) => 16
(a >> b) => 0
Set the variables to different values and different operators and then try...

Assignment Operators

JavaScript supports the following assignment operators −

Sr.No.	Operator & Description
	= (Simple Assignment)
1	Assigns values from the right side operand to the left side operand **Ex:** C = A + B will assign the value of A + B into C **+= (Add and Assignment)**
2	It adds the right operand to the left operand and assigns the result to the left operand. **Ex:** C += A is equivalent to C = C + A **−= (Subtract and Assignment)**
3	It subtracts the right operand from the left operand and assigns the result to the left operand. **Ex:** C -= A is equivalent to C = C - A ***= (Multiply and Assignment)**
4	It multiplies the right operand with the left operand and assigns the result to the left operand. **Ex:** C *= A is equivalent to C = C * A **/= (Divide and Assignment)**
5	It divides the left operand with the right operand and assigns the result to the left operand. **Ex:** C /= A is equivalent to C = C / A **%= (Modules and Assignment)**
6	It takes modulus using two operands and assigns the result to the left operand.

Ex: C %= A is equivalent to C = C % A

Note − Same logic applies to Bitwise operators so they will become like <<=, >>=, >>=, &=, |= and ^=.

Example

Try the following code to implement assignment operator in JavaScript.

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var a = 33;
        var b = 10;
        var linebreak = "<br />";

        document.write("Value of a => (a = b) => ");
        result = (a = b);
        document.write(result);
        document.write(linebreak);

        document.write("Value of a => (a += b) => ");
        result = (a += b);
        document.write(result);
        document.write(linebreak);

        document.write("Value of a => (a -= b) => ");
        result = (a -= b);
        document.write(result);
        document.write(linebreak);

        document.write("Value of a => (a *= b) => ");
        result = (a *= b);
        document.write(result);
        document.write(linebreak);

        document.write("Value of a => (a /= b) => ");
        result = (a /= b);
        document.write(result);
        document.write(linebreak);

        document.write("Value of a => (a %= b) => ");
        result = (a %= b);
```

```
        document.write(result);
        document.write(linebreak);
      //-->
    </script>
    <p>Set the variables to different values and different operators and then try...</p>
  </body>
</html>
```

Output

```
Value of a => (a = b) => 10
Value of a => (a += b) => 20
Value of a => (a -= b) => 10
Value of a => (a *= b) => 100
Value of a => (a /= b) => 10
Value of a => (a %= b) => 0
Set the variables to different values and different operators and then try...
```

Miscellaneous Operator

We will discuss two operators here that are quite useful in JavaScript: the **conditional operator** (? :) and the **typeof operator**.

Conditional Operator (? :)

The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

Sr.No.	Operator and Description
1	**? : (Conditional)** If Condition is true? Then value X : Otherwise value Y

Example

Try the following code to understand how the Conditional Operator works in JavaScript.

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var a = 10;
        var b = 20;
```

```
        var linebreak = "<br />";

        document.write ("((a > b) ? 100 : 200) => ");
        result = (a > b) ? 100 : 200;
        document.write(result);
        document.write(linebreak);

        document.write ("((a < b) ? 100 : 200) => ");
        result = (a < b) ? 100 : 200;
        document.write(result);
        document.write(linebreak);
      //-->
    </script>
    <p>Set the variables to different values and different operators and then try...</p>
  </body>
</html>
```

Output

((a > b) ? 100 : 200) => 200
((a < b) ? 100 : 200) => 100
Set the variables to different values and different operators and then try...

typeof Operator

The **typeof** operator is a unary operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.

The *typeof* operator evaluates to "number", "string", or "boolean" if its operand is a number, string, or boolean value and returns true or false based on the evaluation.

Here is a list of the return values for the **typeof** Operator.

Type	String Returned by typeof
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"
Function	"function"
Undefined	"undefined"
Null	"object"

Example

The following code shows how to implement **typeof** operator.

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var a = 10;
        var b = "String";
        var linebreak = "<br />";

        result = (typeof b == "string" ? "B is String" : "B is Numeric");
        document.write("Result => ");
        document.write(result);
        document.write(linebreak);

        result = (typeof a == "string" ? "A is String" : "A is Numeric");
        document.write("Result => ");
        document.write(result);
        document.write(linebreak);
      //-->
    </script>
    <p>Set the variables to different values and different operators and then try...</p>
  </body>
</html>
```

Output

Result => B is String
Result => A is Numeric
Set the variables to different values and different operators and then try...

1.5. Functions - program modules in JavaScript, programmer defined functions, function definition

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.

Like any other advanced programming language, JavaScript also supports all the features necessary to write modular code using functions. You must have seen functions like **alert()** and **write()** in the

earlier chapters. We were using these functions again and again, but they had been written in core JavaScript only once.

JavaScript allows us to write our own functions as well. This section explains how to write your own functions in JavaScript.

Function Definition

Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax

The basic syntax is shown here.

```
<script type = "text/javascript">
  <!--
    function functionname(parameter-list) {
      statements
    }
  //-->
</script>
```

Example

Try the following example. It defines a function called sayHello that takes no parameters −

```
<script type = "text/javascript">
  <!--
    function sayHello() {
      alert("Hello there");
    }
  //-->
</script>
```

Calling a Function

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

```
<html>
  <head>
    <script type = "text/javascript">
```

```
      function sayHello() {
        document.write ("Hello there!");
      }
    </script>

  </head>

  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "sayHello()" value = "Say Hello">
    </form>
    <p>Use different text in write method and then try...</p>
  </body>
</html>
```

Output

Function Parameters

Till now, we have seen functions without parameters. But there is a facility to pass different parameters while calling a function. These passed parameters can be captured inside the function and any manipulation can be done over those parameters. A function can take multiple parameters separated by comma.

Example

Try the following example. We have modified our **sayHello** function here. Now it takes two parameters.

```
<html>
  <head>
    <script type = "text/javascript">
      function sayHello(name, age) {
        document.write (name + " is " + age + " years old.");
      }
    </script>
  </head>

  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "sayHello('Zara', 7)" value = "Say Hello">
    </form>
```

```
    <p>Use different parameters inside the function and then try...</p>
  </body>
</html>
```

Output

The return Statement

A JavaScript function can have an optional **return** statement. This is required if you want to return a value from a function. This statement should be the last statement in a function.

For example, you can pass two numbers in a function and then you can expect the function to return their multiplication in your calling program.

Example

Try the following example. It defines a function that takes two parameters and concatenates them before returning the resultant in the calling program.

```
<html>
  <head>
    <script type = "text/javascript">
      function concatenate(first, last) {
        var full;
        full = first + last;
        return full;
      }
      function secondFunction() {
        var result;
        result = concatenate('Zara', 'Ali');
        document.write (result );
      }
    </script>
  </head>

  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "secondFunction()" value = "Call Function">
    </form>
    <p>Use different parameters inside the function and then try...</p>
  </body>
</html>
```

Output

There is a lot to learn about JavaScript functions, however we have covered the most important concepts in this section.

1.6. Random-number generator

Math.random()

The Math.random() function is used to return a floating-point pseudo-random number between range [0,1] , 0 (inclusive) and 1 (exclusive).This random number can then be scaled according to the desired range.

Syntax:

Math.random();

Parameters: This function does not accepts any parameter.

Return Value: The math.random() function returns a floating-point, pseudo-random number between range [0,1) , 0 (inclusive) and 1 (exclusive).

Example

```
Math.random();          // returns a random number
```

Math.random() always returns a number lower than 1.

JavaScript Random Integers

Math.random() used with Math.floor() can be used to return random integers.

Example

```
Math.floor(Math.random() * 10);     // returns a random integer from 0 to 9
```

Example

```
Math.floor(Math.random() * 11);     // returns a random integer from 0 to 10
```

Example

```
Math.floor(Math.random() * 100);     // returns a random integer from 0 to 99
```

Example

```
Math.floor(Math.random() * 101);     // returns a random integer from 0 to 100
```

Example

```
Math.floor(Math.random() * 10) + 1;  // returns a random integer from 1 to 10
```

Example

```
Math.floor(Math.random() * 100) + 1; // returns a random integer from 1 to 100
```

A Proper Random Function

As you can see from the examples above, it might be a good idea to create a proper random function to use for all random integer purposes.

This JavaScript function always returns a random number between min (included) and max (excluded):

Example

```
function getRndInteger(min, max) {
  return Math.floor(Math.random() * (max - min) ) + min;
}
```

This JavaScript function always returns a random number between min and max (both included):

Example

```
function getRndInteger(min, max) {
  return Math.floor(Math.random() * (max - min + 1) ) + min;
```

1.7. Scope rules

Scope determines the accessibility (visibility) of variables.

JavaScript Function Scope

In JavaScript there are two types of scope:

- Local scope
- Global scope

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

Scope determines the accessibility (visibility) of these variables.

Variables defined inside a function are not accessible (visible) from outside the function.

Local JavaScript Variables

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

Local variables have **Function scope**: They 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

}
```

Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.

Local variables are created when a function starts, and deleted when the function is completed.

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.

Example

```
var carName = "Volvo";

// code here can use carName

function myFunction() {

  // code here can also use carName

}
```

JavaScript Variables

In JavaScript, objects and functions are also variables.

Scope determines the accessibility of variables, objects, and functions from different parts of the code.

Automatically Global

If you assign a value to a variable that has not been declared, it will automatically become a **GLOBAL** variable.

This code example will declare a global variable carName, even if the value is assigned inside a function.

Example

```
myFunction();

// code here can use carName

function myFunction() {
  carName = "Volvo";
}
```

Strict Mode

All modern browsers support running JavaScript in "Strict Mode".

You will learn more about how to use strict mode in a later chapter of this unit.

In "Strict Mode", undeclared variables are not automatically global.

Global Variables in HTML

With JavaScript, the global scope is the complete JavaScript environment.

In HTML, the global scope is the window object. All global variables belong to the window object.

Example

```
var carName = "Volvo";

// code here can use window.carName
```

Warning

Do NOT create global variables unless you intend to.

Your global variables (or functions) can overwrite window variables (or functions).
Any function, including the window object, can overwrite your global variables and functions.

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).

Function Arguments

Function arguments (parameters) work as local variables inside functions.

1.8. Global functions

Normally, a function defined would not be accessible to all places of a page. Say a function is mentioned using "function checkCookie()" on a external file loaded first on a page. This function cannot be called by the JavaScript file loaded last. So you have to make it a Global function that can be called from anywhere on a page.

The "window" is a Global Object that has a lot of functions.

```
> typeof window
< "object"
```

This "window" object is accessible to all JavaScript code of a page, even if it's an external file. So, if the "window" object is global, then the functions it contain will also be global. So, if we add a new function to the "window" object, it will be a global function. This is what we are going to do to make a Global function.

Here is a normal way that we use to define a function :

```
function checkCookie(){
 // do whatever you want here
}
```

As you already understood, this function wouldn't be accessible to all JS codes. The error that produce when a function can't be accessible is :

ReferenceError: checkCookie is not defined

To make the "checkCookie" function global, we are going to add the function to the "window" object :

```
window.checkCookie=function(){
 // do whatever you want here
};
```

Note the semi colon (";") at the closing of the function. This semi colon is required when defining a function on a object. As normal you can add parameters to the function.

Another Example :

```
function makeString(string, position){
 // do whatever you want here
}
```

can be made global by :

```
window.makeString=function(string, position){
 // do whatever you want here
};
```

Now, you can call the function directly or call it with window :

```
checkCookie()
window.checkCookie()

makeString("subinsb.com", 5)
window.makeString("subinsb.com", 5)
```

The output of the different ways to call functions will be the same. You should know that jQuery is also doing this to make "$" & "jQuery" functions global.

1.9. Recursion

A recursive function is a function that calls itself. We will take the classic factorial function for the demonstration.

in Mathematics, the factorial of a non-negative integer is the product of all positive integer less than or equal to it. The factorial of the integer n is denoted by n!

For example, the factorial of 5 is calculated as follows

5! = 5 x 4 x 3 x 2 x 1 = 120

it is more readable if you develop the factorial function using the recursion technique. See the following recursive factorial function:

```
var factorial = function pf(n) {

   if (n <= 1) {

      return 1;

   } else {

      return n * pf(n - 1);
```

```
    }

};
```

This is how it works. If n is equal to one or zero, the factorial of n is 1; otherwise, the factorial of n is the product of n and factorial of n - 1.

Possible Questions

(2 Marks)

1. Why Java Script is called as scripting language?
2. What is typeof operator in Java Script?
3. What are global functions?
4. What is mean by recursion?

(8 Marks)

1. Explain in detail about obtaining user input with prompt dialogs in Java Script.
2. Explain operators in Java Script.
3. Describe about Functions in Java Script.
4. Explain about random-number generator in Java Script.
5. Explain about Scope rules

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

SUBJECT : Scripting Language **Unit II**

S No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	Which of the following is true about typeof operator in JavaScript?	The typeof is a unary operator that is placed before its single operand, which can be of any type.	Its value is a string indicating the data type of the operand.	Both of the above.	None of the above.	Both of the above.
2	Which of the following is the correct syntax to create a cookie using JavaScript?	document.cookie = 'key1 = value1; key2 = value2; expires = date';	browser.cookie = 'key1 = value1; key2 = value2; expires = date';	window.cookie = 'key1 = value1; key2 = value2; expires = date';	navigator.cookie = 'key1 = value1; key2 = value2; expires = date';	document.cookie = 'key1 = value1; key2 = value2; expires = date';
3	Which built-in method adds one or more elements to the end of an array and returns the new length of the array in Java Script?	last()	put()	push()	None of the above.	push()
4	Why so JavaScript and Java have similar name?	JavaScript is a stripped-down version of Java	JavaScript's syntax is loosely based on Java's	They both originated on the island of Java	None of the above	JavaScript's syntax is loosely based on Java's
5	When a user views a page containing a JavaScript program, which machine actually executes the script?	The User's machine running a Web browser	The Web server	A central machine deep within Netscape's corporate offices	None of the above	The User's machine running a Web browser

6	_____ JavaScript is also called client-side JavaScript.	Microsoft	Navigator	LiveWire	Native	Navigator
7	_____ JavaScript is also called server-side JavaScript.	Microsoft	Navigator	LiveWire	Native	LiveWire
8	What are variables used for in JavaScript Programs?	Storing numbers, dates, or other values	Varying randomly	Both the above	None of the above	Storing numbers, dates, or other values
9	_____ JavaScript statements embedded in an HTML page can respond to user events such as mouse-clicks, form input, and page navigation.	Client-side	Server-side	Local	Native	Client-side
10	What should appear at the very end of your JavaScript? The <script LANGUAGE="JavaScript">tag	The </script>	The <script>	The END statement	None of the above	The </script>
11	Which of the following can't be done with client-side JavaScript?	Validating a form	Sending a form's contents by email	Storing the form's contents to a database file on the server	None of the above	Storing the form's contents to a database file on the server
12	Which of the following is not a valid JavaScript variable name?	2names	_first_and_last_names	FirstAndLast	None of the above	2names
13	_____ tag is an extension to HTML that can enclose any number of JavaScript statements.	<SCRIPT>	<BODY>	<HEAD>	<TITLE>	<SCRIPT>
14	How does JavaScript store dates in a date object?	The number of milliseconds since January 1st, 1970	The number of days since January 1st, 1900	The number of seconds since Netscape's public stock offering.	None of the above	The number of milliseconds since January 1st, 1970
15	Which of the following attribute can hold the JavaScript version?	LANGUAGE	SCRIPT	VERSION	None of the above	LANGUAGE
16	What is the correct JavaScript syntax to write "Hello World"?	System.out.println("Hello World")	println ("Hello World")	document.write("Hello World")	response.write("Hello World")	document.write("Hello World")

17	What is the correct syntax for referring to an external script called " abc.js"?	<script href=" abc.js">	<script name=" abc.js">	<script src=" abc.js">	None of the above	<script src=" abc.js">
18	Which types of image maps can be used with JavaScript?	Server-side image maps	Client-side image maps	Server-side image maps and Client-side image maps	None of the above	Client-side image maps
19	Which is the correct way to write a JavaScript array?	var txt = new Array(1:"tim",2:"kim",3:"jim")	var txt = new Array:1=("tim")2=("kim")3=("jim")	var txt = new Array("tim","kim","jim")	var txt = new Array="tim","kim",jim"	var txt = new Array("tim","kim","jim")
20	What does the <noscript> tag do?	Enclose text to be displayed by non-JavaScript browsers.	Prevents scripts on the page from executing.	Describes certain low-budget movies.	None of the above	Enclose text to be displayed by non-JavaScript browsers.
21	JavaScript entities start with _____ and end with _____.	Semicolon, colon	Semicolon, Ampersand	Ampersand, colon	Ampersand, semicolon	Ampersand, semicolon
22	Which of the following best describes JavaScript?	a low-level programming language.	a scripting language precompiled in the browser.	a compiled scripting language.	an object-oriented scripting language.	an object-oriented scripting language.
23	Choose the server-side JavaScript object?	FileUpLoad	Function	File	Date	File
24	Choose the client-side JavaScript object?	Database	Cursor	Client	FileUpLoad	FileUpLoad
25	_____method evaluates a string of JavaScript code in the context of the specified object.	Eval	ParseInt	ParseFloat	Efloat	Eval
26	Which of the following event fires when the form element loses the focus: <button>, <input>, <label>, <select>, <textarea>?	onfocus	onblur	onclick	ondblclick	onblur
27	The syntax of Eval is _____	[objectName.]eval(numeriC.	[objectName.]eval(string)	[EvalName.]eval(string)	[EvalName.]eval(numeriC.	[objectName.]eval(string)

28	JavaScript is interpreted by _____	Client	Server	Object	None of the above	Client
29	Using _____ statement is how you test for a specific condition.	Select	If	Switch	For	If
30	Which of the following is the structure of an if statement?	if (conditional expression is true) thenexecute this codeend if	if (conditional expression is true)execute this codeend if	if (conditional expression is true) {then execute this code>->}	if (conditional expression is true) then {execute this code}	if (conditional expression is true) {then execute this code>->}
31	How to create a Date object in JavaScript?	dateObjectName = new Date([parameters])	dateObjectName.new Date([parameters])	dateObjectName := new Date([parameters])	dateObjectName Date([parameters])	dateObjectName = new Date([parameters])
32	The _____ method of an Array object adds and/or removes elements from an array.	Reverse	Shift	Slice	Splice	Splice
33	To set up the window to capture all Click events, we use which of the following statement?	window.captureEvents(Event.CLICK);	window.handleEvents (Event.CLICK);	window.routeEvents (Event.CLICK);	window.raiseEvents(Event.CLICK);	window.captureEvents(Event.CLICK);
34	In JavaScript, _____ is an object of the target language data type that encloses an object of the source language.	a wrapper	a link	a cursor	a form	a wrapper
35	When a JavaScript object is sent to Java, the runtime engine creates a Java wrapper of type _____	ScriptObject	JSObject	JavaObject	Jobject	JSObject
36	_____ class provides an interface for invoking JavaScript methods and examining JavaScript properties.	ScriptObject	JSObject	JavaObject	Jobject	JSObject
37	The syntax of a blur method in a button object is _____	Blur()	Blur(contrast)	Blur(value)	Blur(depth)	Blur()

38	The syntax of capture events method for document object is _____	captureEvents()	captureEvents(args eventType)	captureEvents(event Type)	captureEvents(eve ntVal)	captureEvents(ev entType)
39	The syntax of close method for document object is _____	Close(doC.	Close(object)	Close(val)	Close()	Close()
40	script type="text/javascript"> x=4+"4"; document.write(x); </script> Output-----?	44	8	4	Error output	44
41	What is mean by "this" keyword in javascript?	It refers current object	It referes previous object	It is variable which contains value	None of the above	It refers current object
42	Which best explains getSelection()?	Returns the VALUE of a selected OPTION.	Returns document.URL of the window in focus.	Returns the value of cursor-selected text	Returns the VALUE of a checked radio input.	Returns the value of cursor-selected text
43	Predict the output of the following JavaScript code. <script type="text/javascript" language="javascript"> var x=5; var y=6; var res=eval("x*y"); document.write(res); </script>	“30”	30	5*6	“5*6”	30

44	Predict the output of the following JavaScript code. <script type="text/javascript" language="javascript"> var a = "GeeksforGeeks"; var result = a.substring(4, 5); document.write(result); </script>
	
	<script type="text/javascript" language="javascript">
	
	var a = "GeeksforGeeks";
	var result = a.substring(4, 5);
	document.write(result);
	
	</script>
		sf	ks	s	k	s

45	Which of the following is not a reserved word in JavaScript?	interface	throws	program	short	program
46	Predict the output of the following JavaScript code. <script type="text/javascript"> var a="GeeksforGeeks"; var x=a.lastIndexOf("G"); document.write(x); </script>	8	0	9	Error	8
47	Which of the following is the correct syntax to display “GeeksforGeeks” in an alert box using JavaScript?	alertbox(“GeeksforGeeks”);	msg(“GeeksforGeeks”);	msgbox(“GeeksforGeeks”);	alert(“GeeksforGeeks”);	alert(“GeeksforGeeks”);

48	Choose the correct JavaScript syntax to change the content of the following HTML code. GeeksforGeeks	document.getElement(“geek”).innerHTML=”I am a Geek”;	document.getElementById(“geek”).innerHTML=”I am a Geek”;	document.getId(“geek”)=”I am a Geek”;	document.getElementById(“geek”).innerHTML=I am a Geek;	document.getElementById(“geek”).innerHTML=”I am a Geek”;
49	In JavaScript, we do not have datatypes like integer and float. What is the function that can be used to check if the number is an integer or not?	Integer(value)	ifInteger(value)	isInteger(value)	Non of the above	isInteger(value)
50	Predict the output on the console for the following JavaScript code. <script> let myName = "Geek"; let myCity = "Geekistan"; console.log(`My name is ${myName}. My favorite city is ${myCity}.`) </script>	Compilation Error	My name is Geek. My favorite city is Geekistan.	My name is ${myName}.My favorite city is ${myCity}.	0	My name is Geek. My favorite city is Geekistan.
51	Which of the following is an advantage of using JavaScript?	Increased interactivity.	Less server interaction.	Immediate feedback from the users.	All of the above.	All of the above.
52	Which function of an Array object calls a function for each element in the array?	forEach()	every()	forEvery()	each()	forEach()
53	JavaScript is a _____ Side Scripting Language.	Server	ISP	Browser	None of the above	Browser
54	JavaScript is _____ language.	a compiled	an interpreted	Both a) and b)	Non of the above	an interpreted
55	Which was the first browser to support JavaScript?	Mozilla Firefox	Netscape	Google Chrome	IE	Netscape

56	What is the syntax for creating a function in JavaScript named as Geekfunc?	function = Geekfunc()	function Geekfunc()	function := Geekfunc()	function : Geekfunc()	function Geekfunc()
57	How is the function called in JavaScript?	call Geekfunc();	call function GeekFunc();	Geekfunc();	function Geekfunc();	Geekfunc();
58	What is the correct syntax for adding comments in JavaScript?	<!–This is a comment–&gt	//This is a comment	–This is a comment	**This is a comment**	//This is a comment
59	How to insert a multi-line comment in JavaScript?	<!–This is comment line 1 This is comment line 2–&gt	//This is comment line 1 This is comment line 2//	/*This is comment line 1 This is comment line 2*/	**This is comment line 1 This is comment line 2**	/*This is comment line 1 This is comment line 2*/
60	What is the JavaScript syntax for printing values in Console?	print(5)	console.log(5);	console.print(5);	print.console(5);	console.log(5);
61	How to initialize an array in JavaScript?	var Geeks= “Geek1”, “Geek2”, “Geek3”	var Geeks=(1:Geek1, 2:Geek2, 3:Geek3)	var Geeks=(1=Geek1, 2=Geek2, 3=Geek3)	var Geeks=[“Geek1”, “Geek2”, “Geek3”]	var Geeks=[“Geek1”, “Geek2”, “Geek3”]
62	What will be the output of the following code? <script> document.write(typeof(24.49)); </script>	float	number	integer	double	number
63	What will be the command to print the number of characters in the string “GeeksforGeeks”?	document.write(“GeeksforGeeks”.len);	document.write(sizeof(“GeeksforGeeks”));	document.write(“GeeksforGeeks”.length);	document.write(lenof(“GeeksforGeeks”));	document.write(“GeeksforGeeks”.length);
64	What is the method in JavaScript used to remove the whitespace at the beginning and end of any string ?	strip()	trim()	stripped()	trimmed()	trim()

UNIT-III

Functions, Arrays And Objects : JavaScript: Arrays, Objects - Math Object, String Object, Date Object, Boolean & Number Object, document and window Objects. Handling event using java script

1. JavaScript Arrays

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

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

What is an Array?
An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

var car1 = "Saab";
var car2 = "Volvo";
var car3 = "BMW";
However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

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

Syntax:

var array_name = [item1, item2, ...];
Example
var cars = ["Saab", "Volvo", "BMW"];
Spaces and line breaks are not important. A declaration can span multiple lines:

Example

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

2. Math Object

The math object provides you properties and methods for mathematical constants and functions. Unlike other global objects, Math is not a constructor. All the properties and methods of Math are static and can be called by using Math as an object without creating it.

Thus, you refer to the constant pi as Math.PI and you call the sine function as Math.sin(x), where x is the method's argument.

Syntax

The syntax to call the properties and methods of Math are as follows

```
var pi_val = Math.PI;
var sine_val = Math.sin(30);
```

Math Properties

Here is a list of all the properties of Math and their description.

Sr.No.	Property & Description
1	E \

Euler's constant and the base of natural logarithms, approximately 2.718.

2	LN2

Natural logarithm of 2, approximately 0.693.

3	LN10

Natural logarithm of 10, approximately 2.302.

4	LOG2E

Base 2 logarithm of E, approximately 1.442.

5 LOG10E
Base 10 logarithm of E, approximately 0.434.

6 PI
Ratio of the circumference of a circle to its diameter, approximately 3.14159.

7 SQRT1_2
Square root of 1/2; equivalently, 1 over the square root of 2, approximately 0.707.

8 SQRT2
Square root of 2, approximately 1.414.

In the following sections, we will have a few examples to demonstrate the usage of Math properties.

Math Methods
Here is a list of the methods associated with Math object and their description

Sr.No. Method & Description
1 abs()
Returns the absolute value of a number.

2 acos()
Returns the arccosine (in radians) of a number.

3 asin()
Returns the arcsine (in radians) of a number.

4 atan()
Returns the arctangent (in radians) of a number.

5 atan2()
Returns the arctangent of the quotient of its arguments.

6 ceil()
Returns the smallest integer greater than or equal to a number.

7 cos()

Returns the cosine of a number.

8 exp()
Returns EN, where N is the argument, and E is Euler's constant, the base of the natural logarithm.

9 floor()
Returns the largest integer less than or equal to a number.

10 log()
Returns the natural logarithm (base E) of a number.

11 max()
Returns the largest of zero or more numbers.

12 min()
Returns the smallest of zero or more numbers.

13 pow()
Returns base to the exponent power, that is, base exponent.

14 random()
Returns a pseudo-random number between 0 and 1.

15 round()
Returns the value of a number rounded to the nearest integer.

16 sin()
Returns the sine of a number.

17 sqrt()
Returns the square root of a number.

18 tan()
Returns the tangent of a number.

19 toSource()
Returns the string "Math".

3. String Object

The String object lets you work with a series of characters; it wraps Javascript's string primitive data type with a number of helper methods.

As JavaScript automatically converts between string primitives and String objects, you can call any of the helper methods of the String object on a string primitive.

Syntax
Use the following syntax to create a String object −

var val = new String(string);
The String parameter is a series of characters that has been properly encoded.

String Properties
Here is a list of the properties of String object and their description.

Sr.No.	Property & Description
1	constructor

Returns a reference to the String function that created the object.

2	length

Returns the length of the string.

3	prototype

The prototype property allows you to add properties and methods to an object.

In the following sections, we will have a few examples to demonstrate the usage of String properties.

String Methods
Here is a list of the methods available in String object along with their description.

Sr.No.	Method & Description
1	charAt()

Returns the character at the specified index.

2	charCodeAt()

Returns a number indicating the Unicode value of the character at the given index.

3 concat()
Combines the text of two strings and returns a new string.

4 indexOf()
Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.

5 lastIndexOf()
Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.

6 localeCompare()
Returns a number indicating whether a reference string comes before or after or is the same as the given string in sort order.

7 match()
Used to match a regular expression against a string.

8 replace()
Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.

9 search()
Executes the search for a match between a regular expression and a specified string.

10 slice()
Extracts a section of a string and returns a new string.

11 split()
Splits a String object into an array of strings by separating the string into substrings.

12 substr()
Returns the characters in a string beginning at the specified location through the specified number of characters.

13 substring()
Returns the characters in a string between two indexes into the string.

14 toLocaleLowerCase()
The characters within a string are converted to lower case while respecting the current locale.

15 toLocaleUpperCase()
The characters within a string are converted to upper case while respecting the current locale.

16 toLowerCase()
Returns the calling string value converted to lower case.

17 toString()
Returns a string representing the specified object.

18 toUpperCase()
Returns the calling string value converted to uppercase.

19 valueOf()
Returns the primitive value of the specified object.

4. Date Object

The Date object is a datatype built into the JavaScript language. Date objects are created with the new Date( ) as shown below.

Once a Date object is created, a number of methods allow you to operate on it. Most methods simply allow you to get and set the year, month, day, hour, minute, second, and millisecond fields of the object, using either local time or UTC (universal, or GMT) time.

The ECMAScript standard requires the Date object to be able to represent any date and time, to millisecond precision, within 100 million days before or after 1/1/1970. This is a range of plus or minus 273,785 years, so JavaScript can represent date and time till the year 275755.

Syntax
You can use any of the following syntaxes to create a Date object using Date() constructor.

new Date( )
new Date(milliseconds)
new Date(datestring)
new Date(year,month,date[,hour,minute,second,millisecond])

Note − Parameters in the brackets are always optional.

Here is a description of the parameters −

No Argument − With no arguments, the Date() constructor creates a Date object set to the current date and time.

milliseconds − When one numeric argument is passed, it is taken as the internal numeric representation of the date in milliseconds, as returned by the getTime() method. For example, passing the argument 5000 creates a date that represents five seconds past midnight on 1/1/70.

datestring − When one string argument is passed, it is a string representation of a date, in the format accepted by the Date.parse() method.

7 agruments − To use the last form of the constructor shown above. Here is a description of each argument −

year − Integer value representing the year. For compatibility (in order to avoid the Y2K problem), you should always specify the year in full; use 1998, rather than 98.

month − Integer value representing the month, beginning with 0 for January to 11 for December.

date − Integer value representing the day of the month.

hour − Integer value representing the hour of the day (24-hour scale).

minute − Integer value representing the minute segment of a time reading.

second − Integer value representing the second segment of a time reading.

millisecond − Integer value representing the millisecond segment of a time reading.

Date Properties
Here is a list of the properties of the Date object along with their description.

Sr.No.	Property & Description
1	constructor

Specifies the function that creates an object's prototype.

2 prototype
The prototype property allows you to add properties and methods to an object

In the following sections, we will have a few examples to demonstrate the usage of different Date properties.

Date Methods
Here is a list of the methods used with Date and their description.

Sr.No. Method & Description
1 Date()
Returns today's date and time

2 getDate()
Returns the day of the month for the specified date according to local time.

3 getDay()
Returns the day of the week for the specified date according to local time.

4 getFullYear()
Returns the year of the specified date according to local time.

5 getHours()
Returns the hour in the specified date according to local time.

6 getMilliseconds()
Returns the milliseconds in the specified date according to local time.

7 getMinutes()
Returns the minutes in the specified date according to local time.

8 getMonth()
Returns the month in the specified date according to local time.

9 getSeconds()
Returns the seconds in the specified date according to local time.

10 getTime()
Returns the numeric value of the specified date as the number of milliseconds since January 1, 1970, 00:00:00 UTC.

11 getTimezoneOffset()
Returns the time-zone offset in minutes for the current locale.

12 getUTCDate()
Returns the day (date) of the month in the specified date according to universal time.

13 getUTCDay()
Returns the day of the week in the specified date according to universal time.

14 getUTCFullYear()
Returns the year in the specified date according to universal time.

15 getUTCHours()
Returns the hours in the specified date according to universal time.

16 getUTCMilliseconds()
Returns the milliseconds in the specified date according to universal time.

17 getUTCMinutes()
Returns the minutes in the specified date according to universal time.

18 getUTCMonth()
Returns the month in the specified date according to universal time.

19 getUTCSeconds()
Returns the seconds in the specified date according to universal time.

20 getYear()
Deprecated - Returns the year in the specified date according to local time. Use getFullYear instead.

21 setDate()
Sets the day of the month for a specified date according to local time.

22 setFullYear()

Sets the full year for a specified date according to local time.

23 setHours()
Sets the hours for a specified date according to local time.

24 setMilliseconds()
Sets the milliseconds for a specified date according to local time.

25 setMinutes()
Sets the minutes for a specified date according to local time.

26 setMonth()
Sets the month for a specified date according to local time.

27 setSeconds()
Sets the seconds for a specified date according to local time.

28 setTime()
Sets the Date object to the time represented by a number of milliseconds since January 1, 1970, 00:00:00 UTC.

29 setUTCDate()
Sets the day of the month for a specified date according to universal time.

30 setUTCFullYear()
Sets the full year for a specified date according to universal time.

31 setUTCHours()
Sets the hour for a specified date according to universal time.

32 setUTCMilliseconds()
Sets the milliseconds for a specified date according to universal time.

33 setUTCMinutes()
Sets the minutes for a specified date according to universal time.

34 setUTCMonth()
Sets the month for a specified date according to universal time.

35 setUTCSeconds()
Sets the seconds for a specified date according to universal time.

36 setYear()
Deprecated - Sets the year for a specified date according to local time. Use setFullYear instead.

37 toDateString()
Returns the "date" portion of the Date as a human-readable string.

38 toGMTString()
Deprecated - Converts a date to a string, using the Internet GMT conventions. Use toUTCString instead.

39 toLocaleDateString()
Returns the "date" portion of the Date as a string, using the current locale's conventions.

40 toLocaleFormat()
Converts a date to a string, using a format string.

41 toLocaleString()
Converts a date to a string, using the current locale's conventions.

42 toLocaleTimeString()
Returns the "time" portion of the Date as a string, using the current locale's conventions.

43 toSource()
Returns a string representing the source for an equivalent Date object; you can use this value to create a new object.

44 toString()
Returns a string representing the specified Date object.

45 toTimeString()
Returns the "time" portion of the Date as a human-readable string.

46 toUTCString()
Converts a date to a string, using the universal time convention.

47 valueOf()
Returns the primitive value of a Date object.

5. Boolean Object

The Boolean object represents two values, either "true" or "false". If value parameter is omitted or is 0, -0, null, false, NaN, undefined, or the empty string (""), the object has an initial value of false.

Syntax
Use the following syntax to create a boolean object.

var val = new Boolean(value);
Boolean Properties
Here is a list of the properties of Boolean object −

Sr.No. Property & Description
1 constructor
Returns a reference to the Boolean function that created the object.

2 prototype
The prototype property allows you to add properties and methods to an object.

In the following sections, we will have a few examples to illustrate the properties of Boolean object.

Boolean Methods
Here is a list of the methods of Boolean object and their description.

Sr.No. Method & Description
1 toSource()
Returns a string containing the source of the Boolean object; you can use this string to create an equivalent object.

2 toString()
Returns a string of either "true" or "false" depending upon the value of the object.

3 valueOf()
Returns the primitive value of the Boolean object.

6. Number Object

The Number object represents numerical date, either integers or floating-point numbers. In general, you do not need to worry about Number objects because the browser automatically converts number literals to instances of the number class.

Syntax

The syntax for creating a number object is as follows −

```
var val = new Number(number);
```

In the place of number, if you provide any non-number argument, then the argument cannot be converted into a number, it returns NaN (Not-a-Number).

Number Properties

Here is a list of each property and their description.

Sr.No.	Property & Description
1	MAX_VALUE The largest possible value a number in JavaScript can have 1.7976931348623157E+308
2	MIN_VALUE The smallest possible value a number in JavaScript can have 5E-324
3	NaN Equal to a value that is not a number.
4	NEGATIVE_INFINITY A value that is less than MIN_VALUE.
5	POSITIVE_INFINITY A value that is greater than MAX_VALUE
6	prototype A static property of the Number object. Use the prototype property to assign new properties and methods to the Number object in the current document
7	constructor Returns the function that created this object's instance. By default this is the Number object.

In the following sections, we will take a few examples to demonstrate the properties of Number.

Number Methods
The Number object contains only the default methods that are a part of every object's definition.

Sr.No.	Method & Description
1	toExponential() Forces a number to display in exponential notation, even if the number is in the range in which JavaScript normally uses standard notation.
2	toFixed() Formats a number with a specific number of digits to the right of the decimal.
3	toLocaleString() Returns a string value version of the current number in a format that may vary according to a browser's local settings.
4	toPrecision() Defines how many total digits (including digits to the left and right of the decimal) to display of a number.
5	toString() Returns the string representation of the number's value.
6	valueOf() Returns the number's value.

7. Document Object

The HTML DOM document object is the owner of all other objects in your web page.

The HTML DOM Document Object
The document object represents your web page.

If you want to access any element in an HTML page, you always start with accessing the document object.

Below are some examples of how you can use the document object to access and manipulate HTML.

Finding HTML Elements

Method	**Description**
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Changing HTML Elements

Property	**Description**
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element

Method	Description
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element

Adding and Deleting Elements

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*new*, *old*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

8. Window Object

The window object is supported by all browsers. It represents the browser's window.

All global JavaScript objects, functions, and variables automatically become members of the window object.

Global variables are properties of the window object.

Global functions are methods of the window object.

Even the document object (of the HTML DOM) is a property of the window object:

window.document.getElementById("header");

is the same as:

document.getElementById("header");

Window Size

Two properties can be used to determine the size of the browser window.

Both properties return the sizes in pixels:

- window.innerHeight - the inner height of the browser window (in pixels)
- window.innerWidth - the inner width of the browser window (in pixels)

The browser window (the browser viewport) is NOT including toolbars and scrollbars.

For Internet Explorer 8, 7, 6, 5:

- document.documentElement.clientHeight
- document.documentElement.clientWidth
- or
- document.body.clientHeight
- document.body.clientWidth

A practical JavaScript solution (covering all browsers):

Example

```
var w = window.innerWidth
|| document.documentElement.clientWidth
|| document.body.clientWidth;
```

```
var h = window.innerHeight
|| document.documentElement.clientHeight
|| document.body.clientHeight;
```

The example displays the browser window's height and width: (NOT including toolbars/scrollbars)

Other Window Methods

Some other methods:

- window.open() - open a new window
- window.close() - close the current window
- window.moveTo() - move the current window
- window.resizeTo() - resize the current window

9. Handling event using java script

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.

Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

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

onclick Event Type

This is the most frequently used event type which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type.

Example

```
<html>
  <head>
    <script type = "text/javascript">
      <!--
        function sayHello() {
          alert("Hello World")
        }
      //-->
    </script>
```

```
  </head>

  <body>
    <p>Click the following button and see result</p>
    <form>
      <input type = "button" onclick = "sayHello()" value = "Say Hello" />
    </form>
  </body>
</html>
```

Possible Questions:

(Two Marks)

1. What is mean by event?
2. Give example for events handled using Java Script?
3. What are arrays in Java Script?
4. What is mean by Date object in Java Script?

(Eight Marks)

1. Explain about Math Object in Java Script?
2. Explain String object in Java Script.
3. Explain Boolean & Number object in Java Script.
4. Explain about Document and Window objects in Java Script.
5. Explain about Event Handling in Java Script.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

CLASS : II B.Sc IT

SUBJECT : Scripting Language Unit III

S No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	JavaScript _____ are used to store multiple values in a single variable.	Constants	Variables	Arrays	Class	Arrays
2	_____method of Math object returns the absolute value of a number.	ceil()	abs()	cos()	floor()	abs()
3	_____ method of Math object Returns the largest integer less than or equal to a number.	max()	abs()	ceil()	floor()	floor()
4	Which one is not a HTML event?	openWindow	onchange	onclick	onmouseover	openWindow
5	All global JavaScript objects, functions, and variables automatically become members of the _____ object.	document	window	number	math	window
6	The HTML DOM _____ object is the owner of all other objects in your web page.	document	window	number	math	document
7	_____ method of Number object returns the string representation of the number's value.	valueOf()	toFixed()	toPrecision()	toString()	toString()

8	The _____ object represents two values, either "true" or "false".	Boolean	Number	Document	math	Boolean
9	The _____ property of string object returns the length of the string.	total	Number	length	absolute	length
10	The _____ method of string object returns the character at the specified index	concat()	indexOf()	slice()	charAt()	charAt()
11	In array how many types of class____	1	2	3	0	1
12	In arraylist the elements can _____ size	Growable	sizable	both a and b	no size	both a and b
13	In array elements retrieved with _____loop	for	while	dowhile	none	for
14	Arraylist can be retrieved with loop and ____	operators	iterators	sizable	all of the above	iterators
15	Arraylist can be obtained a ____version	Synchronized	Asynchronized	both a and b	none of the above	Synchronized
16	Array and Arraylist are similar for the ___ and ___operations	get,add	get,insert	insert,del	add,get	get,add
17	_____ indicates the characteristics of data	variable	identifier	datatype	keyword	datatype
18	19.HTTP stands for_____	Hyper text transmission protocol	hyper text transfer protocol	hyper texture transfer protocol	none of this	hyper text transfer protocol
19	In HTTP a client is a _____	web page	web browser	server	all the above	web browser
20	21.A HTML file is a _____	document	reference	source	resource	resource
21	22.HTTP is a _____protocol	statefull	connection-oriented	stateless	connection less	stateless

22	GUI stands for _____	Graphical user interface	Geo-graphical user interface	Graphics user interface	none	Graphical user interface
23	_____ is a example for standalone GUI application	Word excel	Word processor	word document	Microsoft word	Word processor
24	25.URI stands for _____	Uniform Resource Interface	Uniform Relocation Interface	Uniform Resource Identifier	Uniform Relocation Identifier	Uniform Resource Identifier
25	26.URL stands for_____	Uniform Resource Interface	Uniform Resource Locator	Uniform Resource Language	Unified Resource allocator	Uniform Resource Locator
26	Java script is the _____type language means you don't nead to specify type of variable	static	auto	register	dynamic	dynamic
27	_____can store heterogenous data type	array	arraylist	linked list	data link	array
28	Java programs can carry extensive amount of ___time information		compile	execution	none of these	execution
29	which compiler enables high performance	java compiler	java virtual machine	just in out	just in time	just in time
30	The new keyword is followed by a call to a constructor this call_____new object	declaration	initialiation	instantiation	creating	instantiation
31	An array is the basic functionality provided by _____	c++	java script	java	vb.net	java
32	_____represents instance through which can access member	class	object	array	union	object
33	Unintialize variable equal to _____	string	character	null	integer	null
34	Which is not a nonprimitive datatype are as follows	object	class	array	RegExp	class

35	In java script null document.writeln(str==undefined)it returns	FALSE	null	TRUE	empty	FALSE
36	Java script empty variable Boolean contest return_____	FALSE	TRUE	null	char	null
37	Java script as only one _____data type	decimal	float	numeric	string	numeric
38	To instantiate a data object, or any other object use the ____operator.	exist	new	free	create	new
39	New requires a single argument _____method for the object to be created.	constructor	destructor	argument	defaultconstructo	argument
40	Declaration do not _____object	initialization	instantiate	creating	generalization	initialization
41	In array we insert elements using _____operators	conditional	logical	assignment	relational	assignment
42	Array list is the part of _____frame work in java	array	list	collection	none of these	collection
43	Arraylist has set of _____to access and modify them _____	function	class	object	methods	methods
44	Java is designed for the _____environment of the internet	dynamic	distributed	add	replace	distributed
45	We can insert element into the arraylist object using _____method	delete	insert	add	replace	add
46	Length of arraylist is provided by _____method	add	size	create	update	size
47	There are_____types of primitive datatype in javascript	3	5	4	2	5
48	Javascript Boolean type can have_____value	2	1	0	3	2
49	_____initiated java language project	lampsalt	dennis Ritchie	james gosling	bjarne stroustrup	james gosling

50	Javascript symbol datatype it is used for _____unique object property.	variable	identifier	datatype	none of the above	identifier
51	Java bean is an ……	Structure oriented	Object oriented	Both a&b	none	Object oriented
52	Program building blocks are called as……	Components	Elements	Applications	none	Components
53	Java class should follow …… following convention	Constructor	arg constructor	no-arg constructor	Both a&b	no-arg constructor
54	EJB stands for ……	Edision java bean	Element java bean	Enterprise java bean	none	Enterprise java bean
55	EJBs provide infrastructure for developing and deploying ……..	EJB fundamentals	EJB functions	EJB elements	Mission critical	Mission critical
56	UML stands for ……	Unified Modified Language	User Modified Language	Unified Markup Language	User Markup Language	Unified Modified Language
57	The Application Logic layer is also known as …….	Upper tier	Middle tier	Lower tier	Bottom tier	Middle tier
58	Application often need to communicate with other systems is an ……. Manner.	Synchronous	Asynchronous	Authentication	none	Asynchronous
59	There are ….. kinds of EJB.	1	2	3	4	3
60	Session beans are ……..	Stateful	Stateless	Both a&b	none	Both a&b
61	Entity beans are ……	Elements	objects	Attributes	Class	objects
62	JAR stands for …….	Java ARchitechture	Java ARchive	Java Arc	none	Java ARchive
63	JAR files are ……..	Archive files	Auto files	Compress files	De-compress files	Archive files

64	JDK stands for ……	Java Development Kit	Java Details Kit	Java Design Kit	none	Java Development Kit

CLIENT SIDE TECHNOLOGIES - AJAX– Evolution of AJAX – AJAX Framework – Web applications with AJAX – AJAX with PHP – AJAX with Databases- Ajax Client Server Architecture-XML Http Request Object-Call Back Methods.

1. Evolution of AJAX

AJAX is about updating parts of a web page, without reloading the whole page.

AJAX = Asynchronous JavaScript and XML.

AJAX is a technique for creating fast and dynamic web pages.

AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Classic web pages, (which do not use AJAX) must reload the entire page if the content should change.

Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.

AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

- Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
- Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
- XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- AJAX is a web browser technology independent of web server software.
- A user can continue to use the application while the client program requests information from the server in the background.
- Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger.
- Data-driven as opposed to page-driven.

2. AJAX Framework

AJAX cannot work independently. It is used in combination with other technologies to create interactive webpages.

JavaScript

- Loosely typed scripting language.
- JavaScript function is called when an event occurs in a page.
- Glue for the whole AJAX operation.

DOM

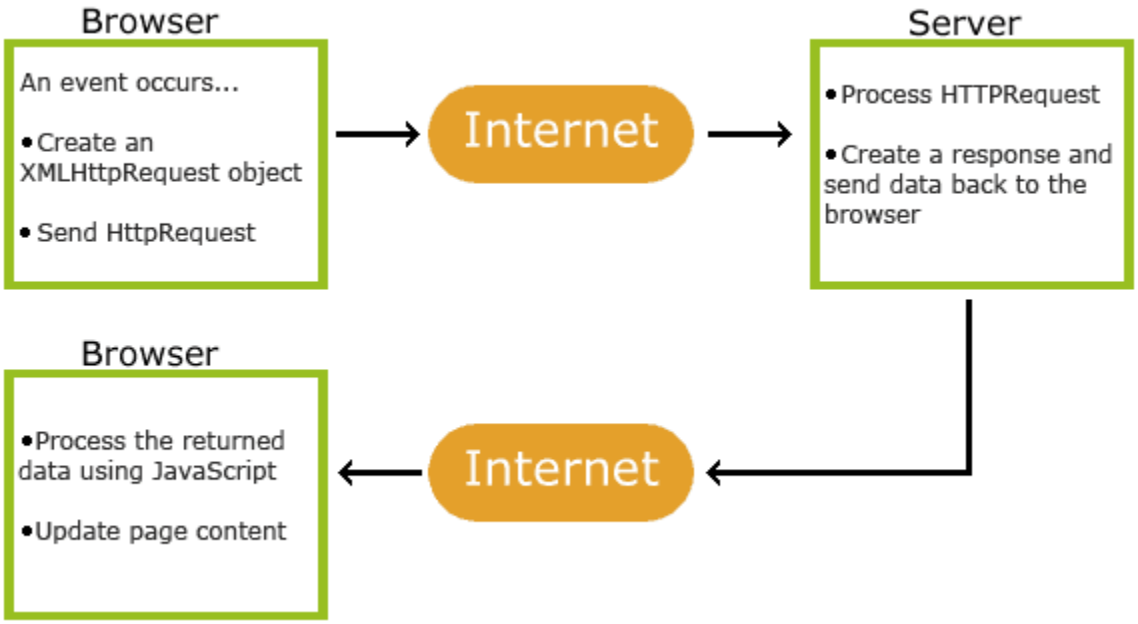
- API for accessing and manipulating structured documents.
- Represents the structure of XML and HTML documents.

CSS

- Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript

XMLHttpRequest

- JavaScript object that performs asynchronous interaction with the server.



AJAX is Based on Internet Standards

AJAX is based on internet standards, and uses a combination of:

- XMLHttpRequest object (to exchange data asynchronously with a server)
- JavaScript/DOM (to display/interact with the information)
- CSS (to style the data)
- XML (often used as the format for transferring data)

AJAX applications are browser- and platform-independent!

Google Suggest

AJAX was made popular in 2005 by Google, with Google Suggest.

Google Suggest is using AJAX to create a very dynamic web interface: When you start typing in Google's search box, a JavaScript sends the letters off to a server and the server returns a list of suggestions.

3. Web applications with AJAX

Here is a list of some famous web applications that make use of AJAX.

Google Maps

A user can drag an entire map by using the mouse, rather than clicking on a button.

Google Suggest

As you type, Google offers suggestions. Use the arrow keys to navigate the results.

Gmail

Gmail is a webmail built on the idea that emails can be more intuitive, efficient, and useful.

Yahoo Maps (new)

Now it's even easier and more fun to get where you're going!

Difference between AJAX and Conventional CGI Program

Try these two examples one by one and you will feel the difference. While trying AJAX example, there is no discontinuity and you get the response very quickly, but when you try the standard GCI example, you would have to wait for the response and your page also gets refreshed.

4. AJAX with PHP

Your client-side script is ready. Now, we have to write our server-side script, which will fetch age, wpm, and sex from the database and will send it back to the client. Put the following code into the file "ajax-example.php".

```
<?php
$dbhost = "localhost";
$dbuser = "dbusername";
$dbpass = "dbpassword";
$dbname = "dbname";

//Connect to MySQL Server
mysql_connect($dbhost, $dbuser, $dbpass);

//Select Database
```

```
mysql_select_db($dbname) or die(mysql_error());

// Retrieve data from Query String
$age = $_GET['age'];
$sex = $_GET['sex'];
$wpm = $_GET['wpm'];

// Escape User Input to help prevent SQL Injection
$age = mysql_real_escape_string($age);
$sex = mysql_real_escape_string($sex);
$wpm = mysql_real_escape_string($wpm);

//build query
$query = "SELECT * FROM ajax_example WHERE sex = '$sex'";

if(is_numeric($age))
  $query .= " AND age <= $age";

if(is_numeric($wpm))
  $query .= " AND wpm <= $wpm";

//Execute query
$qry_result = mysql_query($query) or die(mysql_error());

//Build Result String
$display_string = "<table>";
$display_string .= "<tr>";
$display_string .= "<th>Name</th>";
$display_string .= "<th>Age</th>";
$display_string .= "<th>Sex</th>";
$display_string .= "<th>WPM</th>";
$display_string .= "</tr>";

// Insert a new row in the table for each person returned
while($row = mysql_fetch_array($qry_result)) {
  $display_string .= "<tr>";
  $display_string .= "<td>$row[name]</td>";
  $display_string .= "<td>$row[age]</td>";
  $display_string .= "<td>$row[sex]</td>";
  $display_string .= "<td>$row[wpm]</td>";
  $display_string .= "</tr>";
}

echo "Query: " . $query . "<br />";
```

```
$display_string .= "</table>";

echo $display_string;
?>
```

Now try by entering a valid value (e.g., 120) in *Max Age* or any other box and then click Query MySQL button.

Max Age:

Max WPM:

Sex:

Your result will display here in this section after you have made your entry.

If you have successfully completed this lesson, then you know how to use MySQL, PHP, HTML, and Javascript in tandem to write AJAX applications.

5. AJAX with Databases

To clearly illustrate how easy it is to access information from a database using AJAX, we are going to build MySQL queries on the fly and display the results on "ajax.html". But before we proceed, let us do the ground work. Create a table using the following command.

NOTE − We are assuming you have sufficient privilege to perform the following MySQL operations.

```
CREATE TABLE 'ajax_example' (
  'name' varchar(50) NOT NULL,
  'age' int(11) NOT NULL,
  'sex' varchar(1) NOT NULL,
  'wpm' int(11) NOT NULL,
  PRIMARY KEY  ('name')
)
```

Now dump the following data into this table using the following SQL statements −

```
INSERT INTO 'ajax_example' VALUES ('Jerry', 120, 'm', 20);
INSERT INTO 'ajax_example' VALUES ('Regis', 75, 'm', 44);
INSERT INTO 'ajax_example' VALUES ('Frank', 45, 'm', 87);
INSERT INTO 'ajax_example' VALUES ('Jill', 22, 'f', 72);
```

```
INSERT INTO 'ajax_example' VALUES ('Tracy', 27, 'f', 0);
INSERT INTO 'ajax_example' VALUES ('Julie', 35, 'f', 90);
```

Client Side HTML File

Now let us have our client side HTML file, which is ajax.html, and it will have the following code –

```
<html>
  <body>
    <script language = "javascript" type = "text/javascript">
      <!--
      //Browser Support Code
      function ajaxFunction() {
        var ajaxRequest;  // The variable that makes Ajax possible!

        try {
          // Opera 8.0+, Firefox, Safari
          ajaxRequest = new XMLHttpRequest();
        } catch (e) {

          // Internet Explorer Browsers
          try {
            ajaxRequest = new ActiveXObject("Msxml2.XMLHTTP");
          } catch (e) {

            try {
              ajaxRequest = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {
              // Something went wrong
              alert("Your browser broke!");
              return false;
            }
          }
        }

        // Create a function that will receive data
        // sent from the server and will update
        // div section in the same page.
        ajaxRequest.onreadystatechange = function() {

          if(ajaxRequest.readyState == 4) {
            var ajaxDisplay = document.getElementById('ajaxDiv');
            ajaxDisplay.innerHTML = ajaxRequest.responseText;
```

```
            }
         }

         // Now get the value from user and pass it to
         // server script.
         var age = document.getElementById('age').value;
         var wpm = document.getElementById('wpm').value;
         var sex = document.getElementById('sex').value;
         var queryString = "?age = " + age ;

         queryString +=  "&wpm = " + wpm + "&sex = " + sex;
         ajaxRequest.open("GET", "ajax-example.php" + queryString, true);
         ajaxRequest.send(null);
      }
      //-->
   </script>

   <form name = 'myForm'>
      Max Age: <input type = 'text' id = 'age' /> <br />
      Max WPM: <input type = 'text' id = 'wpm' /> <br />
      Sex:

      <select id = 'sex'>
         <option value = "m">m</option>
         <option value = "f">f</option>
      </select>

      <input type = 'button' onclick = 'ajaxFunction()' value = 'Query MySQL'/>
   </form>

   <div id = 'ajaxDiv'>Your result will display here</div>
</body>
</html>
```

NOTE − The way of passing variables in the Query is according to HTTP standard and have formA.

URL?variable1 = value1;&variable2 = value2;

The above code will give you a screen as given below −

Max Age:

Max WPM:

Sex:

Your result will display here in this section after you have made your entry.

NOTE − This is a dummy screen.

6. Ajax Client Server Architecture

AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

- Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
- Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
- XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- AJAX is a web browser technology independent of web server software.
- A user can continue to use the application while the client program requests information from the server in the background.
- Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger.
- Data-driven as opposed to page-driven.

Rich Internet Application Technology

AJAX is the most viable Rich Internet Application (RIA) technology so far. It is getting tremendous industry momentum and several tool kit and frameworks are emerging. But at the same time, AJAX has browser incompatibility and it is supported by JavaScript, which is hard to maintain and debug.

AJAX is Based on Open Standards

AJAX is based on the following open standards −

- Browser-based presentation using HTML and Cascading Style Sheets (CSS).
- Data is stored in XML format and fetched from the server.

- Behind-the-scenes data fetches using XMLHttpRequest objects in the browser.
- JavaScript to make everything happen.

7. XML Http Request Object

The XMLHttpRequest object is the key to AJAX. It has been available ever since Internet Explorer 5.5 was released in July 2000, but was not fully discovered until AJAX and Web 2.0 in 2005 became popular.

XMLHttpRequest (XHR) is an API that can be used by JavaScript, JScript, VBScript, and other web browser scripting languages to transfer and manipulate XML data to and from a webserver using HTTP, establishing an independent connection channel between a webpage's Client-Side and Server-Side.

The data returned from XMLHttpRequest calls will often be provided by back-end databases. Besides XML, XMLHttpRequest can be used to fetch data in other formats, e.g. JSON or even plain text.

You already have seen a couple of examples on how to create an XMLHttpRequest object.

Listed below are some of the methods and properties that you have to get familiar with.

XMLHttpRequest Methods

- **abort()**

  Cancels the current request.

- **getAllResponseHeaders()**

  Returns the complete set of HTTP headers as a string.

- **getResponseHeader(headerName)**

  Returns the value of the specified HTTP header.

- **open(method, URL)**
- **open(method, URL, async)**
- **open(method, URL, async, userName)**
- **open(method, URL, async, userName, password)**

  Specifies the method, URL, and other optional attributes of a request.

The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods such as "PUT" and "DELETE" (primarily used in REST applications) may be possible.

The "async" parameter specifies whether the request should be handled asynchronously or not. "true" means that the script processing carries on after the send() method without waiting for a response, and "false" means that the script waits for a response before continuing script processing.

- **send(content)**

  Sends the request.

- **setRequestHeader(label, value)**

  Adds a label/value pair to the HTTP header to be sent.

XMLHttpRequest Properties

- **onreadystatechange**

  An event handler for an event that fires at every state change.

- **readyState**

  The readyState property defines the current state of the XMLHttpRequest object.

  The following table provides a list of the possible values for the readyState property −

State	Description
0	The request is not initialized.
1	The request has been set up.
2	The request has been sent.
3	The request is in process.
4	The request is completed.

readyState = 0 After you have created the XMLHttpRequest object, but before you have called the open() method.

readyState = 1 After you have called the open() method, but before you have called send().

readyState = 2 After you have called send().

readyState = 3 After the browser has established a communication with the server, but before the server has completed the response.

readyState = 4 After the request has been completed, and the response data has been completely received from the server.

- **responseText**

  Returns the response as a string.

- **responseXML**

  Returns the response as XML. This property returns an XML document object, which can be examined and parsed using the W3C DOM node tree methods and properties.

- **status**

  Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").

- **statusText**

  Returns the status as a string (e.g., "Not Found" or "OK").

8. Call Back Methods

A callback function is a function passed as a parameter to another function.

If you have more than one AJAX task in a website, you should create one function for executing the XMLHttpRequest object, and one callback function for each AJAX task.

The function call should contain the URL and what function to call when the response is ready.

Example

```
loadDoc("url-1", myFunction1);

loadDoc("url-2", myFunction2);

function loadDoc(url, cFunction) {
  var xhttp;
  xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
```

```
    cFunction(this);
   }
  };
  xhttp.open("GET", url, true);
  xhttp.send();
}

function myFunction1(xhttp) {
  // action goes here
}
function myFunction2(xhttp) {
  // action goes here
}
```

Possible Questions

(2 Marks)

1. What is AJAX?
2. What is the difference between AJAX and CGI?
3. What are Call Back Methods.
4. What are XML Http Request Objects?

(8 Marks)

1. Explain about evolution of AJAX.
2. Explain AJAX Framework.
3. Explain Ajax Client Server Architecture.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

SUBJECT : Scripting Language **Unit IV**

S No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	What are the features of Ajax?	Live data binding	Client-side template rendering	Declarative instantiation of client components	All of the above	All of the above
2	Which of the following are the features of an HTTP request?	URL being requested	Optional request body	Optional set of request headers	All of the mentioned	All of the mentioned
3	AJAX based on _____	JavaScript and XML	VBScript and XML	JavaScript and Java	JavaScript and HTTP requests	JavaScript and XML
4	What are the advantages of Ajax?	Bandwidth utilization	More interactive	Speeder retrieval of data	All of these	All of these
5	What makes Ajax unique?	It works as a stand-alone Web-development tool	It works the same with all Web browsers	It uses C++ as its programming language	It makes data requests asynchronously	It makes data requests asynchronously
6	What are the controls of Ajax?	ScriptManager	ScriptManagerProxy	UpdatePanel	All of these	All of these
7	Which property is used to control the duration of Ajax request.	AsyncPostBackTimeout	AsyncTimeout	Timeout	PostBackTimeout	AsyncPostBackTimeout
8	AJAX was made popular by?	Microsoft	Google	Sun Microsystemwrong	IBM	Google
9	The jQuery AJAX methods .get(), .post(), and .ajax() all require which parameter to be supplied?	method	url	data	headers	url
10	What are all the technologies used by Ajax?	JavaScript	XMLHttpRequest	Document Object Model (DOM)	All of the above	All of the above

11	what is the full form of AJAX ?	Asynchronous Javascript and XML	Another Java and XML Library	Abstract JSON and XML	None of the mentioned	Asynchronous Javascript and XML
12	_____ combination of technologies gives AJAX its name.	ASP and XAML	Atlas and XML	Asynchronous JavaScript and XML	None of the mentioned	Asynchronous JavaScript and XML
13	Which of the following technology is not used by Ajax?	JavaScript	Document Object Model	XMLHttpRequest	Flash	Flash
14	_____is the name of the DLL that contains Ajax control tool kit?	Ajaxtoolkit.dll	Ajaxcontroltoolkit.dll	Ajaxcontrol.dll	control.dll	Ajaxcontroltoolkit.dll
15	What are the disadvantages of Ajax?	Debugging is difficult	Increases size of the requests	Slow and unreliable network connection.	All of the mentioned above	All of the mentioned above
16	_____are the methods used for cross domain Ajax calls?	CORS	JSONP	both 1 & 2	None of the above	both 1 & 2
17	_____ JavaScript is also called client-side JavaScript.	Microsoft	Navigator	LiveWire	Native	Navigator
18	_____ JavaScript is also called server-side JavaScript	Microsoft	Navigator	LiveWire	Native	LiveWire
19	When a user views a page containing a JavaScript program, which machine actually executes the script?	The User's machine running a Web browser	A central machine deep within Netscape's corporate offices	A central machine deep within Netscape's corporate offices	None of the above	The User's machine running a Web browser
20	Why so JavaScript and Java have similar name?	JavaScript is a stripped-down version of Java	JavaScript's syntax is loosely based on Java's	They both originated on the island of Java	None of the above	JavaScript's syntax is loosely based on Java's
21	Using AJAX we can made our web page	more interactive and faster	easy to connect web page with server	more dynamic	None of the above	more interactive and faster
22	AJAX comes in ____.	2003	2005	2004	2006	2005
23	Which Web browser is the least optimized for Microsoft's version of AJAX?	Firefox	Opera	Safari	Internet Explorer	Safari

24	Which one of these legendary Greek mythical figures or places is the code name for Microsoft's version of AJAX ?	Oracle	Atlas	Hercules	Delphi	Atlas
25	What does the XMLHttpRequest object accomplish in Ajax?	It's the programming language used to develop Ajax applications.	It provides a means of exchanging structured data between the Web server and client.	It provides the ability to asynchronously exchange data between Web browsers and a Web server.	It provides the ability to mark up and style the display of Web-page text.	It provides the ability to asynchronously exchange data between Web browsers and a Web server.
26	What is AJAX ?	is a programe	is a country name	is a football club name	None of the above	is a programe
27	By default the execute() method retrieves……	all rows in result set	all columns in result set	limited rows in result set	limited rows in result set	limited rows in result set
28	JDBC is used to……	connectivity between web application to DB	connectivity between program to DB	connectivity between a pplication to DB	none	connectivity between program to DB
29	JNDI stands……	Java Naming & Directory Interface	Java Naming & Dictionary Interface	Java Namespace & Directory Interface	Java Naming & Domain Interface	Java Naming & Domain Interface
30	For user authentication …… method is use	inside the execute()	outside the execute()	bottom the execute()	down the execute()	inside the execute()
31	On the final page of the wizard, review the specification for the new java bean, then click ----- to complete the wizard.	Complete	finish	final	conculsion	final
32	The password will be _____ when you type it in the field .	covered	masked	changed	duplicate	duplicate
33	The data source is defined in an ____	application server	web application	SQL server	database server	SQL server
34	___ is a zip or compression utility tailored to java's needs.	jar	jar extension	pass	applet	applet

35	There are _____ advantages in introspection.	2	3	4	6	4
36	The introspector class provides _____ class.	developer	information	beaninfo	descriptor	descriptor
37	A call to_____ results in the introspection process analyzing the bean classes and super class	setbeaninfo	getbeaninfo	beansetinfo	beangetinfo	getbeaninfo
38	Basically introspection means analysis of _____ capabilities	beans	applets	classes	API	beans
39	Introspection describes how methods , properties , and _____ are discovered in the beans that you write.	event	tool	properties	methods	event
40	Introspection describes how methods, _____ , and event are discovered beans that you write.	event	tool	properties	methods	properties
41	Introspection describes how _____ , properties , and event are discovered beans that you write.	event	tool	properties	methods	methods
42	Variables are named storage locations in memory, the value of which does not change during program _____	Debug	Design	Execution	All of the above	Execution
43	Is scripting languages are case sensitive?	VB Script is case sensitive	Java Script is case insenitive	Both a) and b) are true	Both a) and b) are false	Both a) and b) are true
44	Which VBScript function converts an input string to all lowercase?	LCase	LowerCase	Lower	There is no such function to directly convert to lowercase	LCase
45	VBScript was launched in	1995	1996	1994	1997	1996
46	Which browser has built-in support for executing VBScript?	Internet Explorer	Mozilla Firefox	Opera	None of these	Internet Explorer

47	In the Select Case statement, which case is used for unknown cases?	Else	Default	Unknown	Not	Else
48	How to take request using get method?	Using Request.form	Using Response.write	Using Request.Querystring	Using Request.Servarvariables	Using Request.Querystring
49	What is the syntax for creating a function in JavaScript named as Geekfunc?	function = Geekfunc()	function Geekfunc()	function := Geekfunc()	function : Geekfunc()	function Geekfunc()
50	How is the function called in JavaScript?	call Geekfunc();	call function GeekFunc();	Geekfunc();	function Geekfunc();	Geekfunc();
51	What is the correct syntax for adding comments in JavaScript?	<!–This is a comment–&gt	//This is a comment	–This is a comment	**This is a comment**	//This is a comment
52	How to insert a multi-line comment in JavaScript?	<!–This is comment line 1 This is comment line 2–&gt	//This is comment line 1 This is comment line 2//	/*This is comment line 1 This is comment line 2*/	**This is comment line 1 This is comment line 2**	/*This is comment line 1 This is comment line 2*/
53	What is the JavaScript syntax for printing values in Console?	print(5)	console.log(5);	console.print(5);	print.console(5);	console.log(5);
54	How to initialize an array in JavaScript?	var Geeks= “Geek1”, “Geek2”, “Geek3”	var Geeks=(1:Geek1, 2:Geek2, 3:Geek3)	var Geeks=(1=Geek1, 2=Geek2, 3=Geek3)	var Geeks=[“Geek1”, “Geek2”, “Geek3”]	var Geeks=[“Geek1”, “Geek2”, “Geek3”]
55	What will be the output of the following code? <script> document.write(typeof(24.49)); </script>	float	number	integer	double	number
56	What will be the command to print the number of characters in the string “GeeksforGeeks”?	document.write(“GeeksforGeeks”.len);	document.write(sizeof(“GeeksforGeeks”));	document.write(“GeeksforGeeks”.length);	document.write(lenof(“GeeksforGeeks”));	document.write(“GeeksforGeeks”.length);

57	What is the method in JavaScript used to remove the whitespace at the beginning and end of any string ?	strip()	trim()	stripped()	trimmed()	trim()
58						
59						
60						
61						
62						
63						
64						

Server Side Scripting- JSP : Servlet Overview – Life cycle of a Servlet – Handling HTTP request and response – Using Cookies – Session tracking – Java Server Pages – Anatomy of JSP – Implicit JSP Objects – JDBC – Java Beans – Advantages – Enterprise Java Beans – EJB Architecture – Types of Beans – EJB Transactions

1. Servlet Overview

Java Servlets are programs that run on a Web or Application server and act as a middle layer between a requests coming from a Web browser or other HTTP client and databases or applications on the HTTP server.

Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

Java Servlets often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But Servlets offer several advantages in comparison with the CGI.

- Performance is significantly better.
- Servlets execute within the address space of a Web server. It is not necessary to create a separate process to handle each client request.
- Servlets are platform-independent because they are written in Java.
- Java security manager on the server enforces a set of restrictions to protect the resources on a server machine. So servlets are trusted.
- The full functionality of the Java class libraries is available to a servlet. It can communicate with applets, databases, or other software via the sockets and RMI mechanisms that you have seen already.

Servlets perform the following major tasks −

Read the explicit data sent by the clients (browsers). This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.

Read the implicit HTTP request data sent by the clients (browsers). This includes cookies, media types and compression schemes the browser understands, and so forth.

Process the data and generate the results. This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.

Send the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.

Send the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.

Servlets Packages

Java Servlets are Java classes run by a web server that has an interpreter that supports the Java Servlet specification.

Servlets can be created using the javax.servlet and javax.servlet.http packages, which are a standard part of the Java's enterprise edition, an expanded version of the Java class library that supports large-scale development projects.

These classes implement the Java Servlet and JSP specifications. At the time of writing this tutorial, the versions are Java Servlet 2.5 and JSP 2.1.

Java servlets have been created and compiled just like any other Java class. After you install the servlet packages and add them to your computer's Classpath, you can compile servlets with the JDK's Java compiler or any other current compiler.

2. Life cycle of a Servlet

A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.

- The servlet is initialized by calling the **init()** method.
- The servlet calls **service()** method to process a client's request.
- The servlet is terminated by calling the **destroy()** method.
- Finally, servlet is garbage collected by the garbage collector of the JVM.

Now let us discuss the life cycle methods in detail.

The init() Method

The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards. So, it is used for one-time initializations, just as with the init method of applets.

The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started.

When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. The init() method simply creates or loads some data that will be used throughout the life of the servlet.

The init method definition looks like this −

```
public void init() throws ServletException {
  // Initialization code...
}
```

The service() Method

The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.

Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

Here is the signature of this method −

```
public void service(ServletRequest request, ServletResponse response)
  throws ServletException, IOException {
}
```

The service () method is called by the container and service method invokes doGet, doPost, doPut, doDelete, etc. methods as appropriate. So you have nothing to do with service() method but you override either doGet() or doPost() depending on what type of request you receive from the client.

The doGet() and doPost() are most frequently used methods with in each service request. Here is the signature of these two methods.

The doGet() Method

A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
  // Servlet code
}
```

The doPost() Method

A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
  // Servlet code
}
```

The destroy() Method

The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

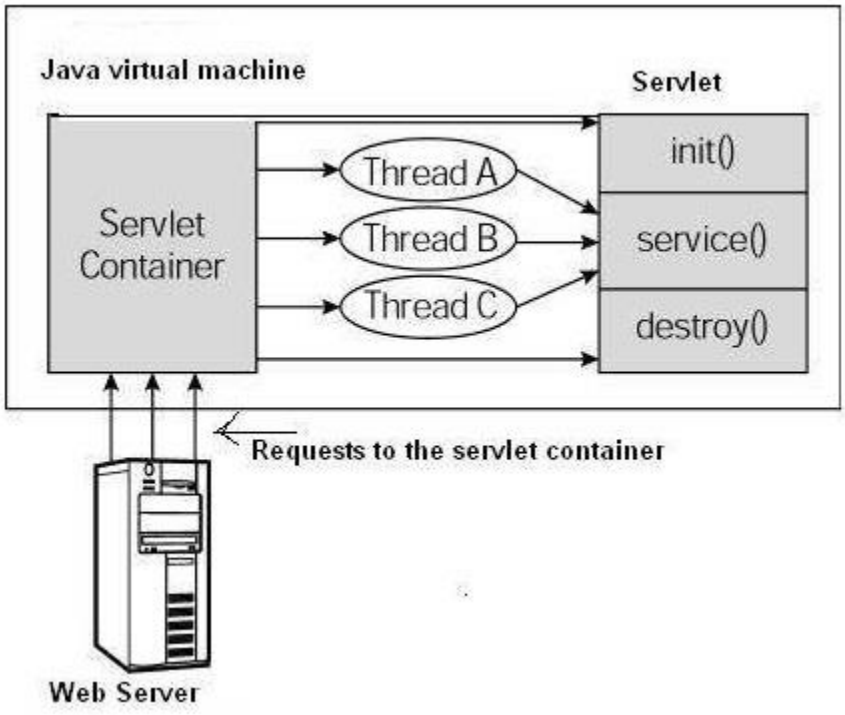
After the destroy() method is called, the servlet object is marked for garbage collection. The destroy method definition looks like this −

```
public void destroy() {
  // Finalization code...
}
```

Architecture Diagram

The following figure depicts a typical servlet life-cycle scenario.

- First the HTTP requests coming to the server are delegated to the servlet container.
- The servlet container loads the servlet before invoking the service() method.
- Then the servlet container handles multiple requests by spawning multiple threads, each thread executing the service() method of a single instance of the servlet.



3. Handling HTTP request and response

HTTP Requests

The request sent by the computer to a web server, contains all sorts of potentially interesting information; it is known as HTTP requests.

The HTTP client sends the request to the server in the form of request message which includes following information:

- The Request-line
- The analysis of source IP address, proxy and port
- The analysis of destination IP address, protocol, port and host
- The Requested URI (Uniform Resource Identifier)
- The Request method and Content
- The User-Agent header
- The Connection control header
- The Cache control header

The HTTP request method indicates the method to be performed on the resource identified by the **Requested URI (Uniform Resource Identifier)**. This method is case-sensitive and should be used in uppercase.

The HttpServlet class provides specialized methods that handle the various types of HTTP requests. A servlet developer typically overrides one of these methods. These methods are doDelete( ), doGet( ), doHead( ), doOptions( ), doPost( ), doPut( ), and doTrace( ). However, the GET and POST requests are commonly used when handling form input.

The HTTP request methods are:

HTTP Request	Description
GET	Asks to get the resource at the requested URL.
POST	Asks the server to accept the body info attached. It is like GET request with extra info sent with the request.
HEAD	Asks for only the header part of whatever a GET would return. Just like GET but with no body.
TRACE	Asks for the loopback of the request message, for testing or troubleshooting.
PUT	Says to put the enclosed info (the body) at the requested URL.
DELETE	Says to delete the resource at the requested URL.
OPTIONS	Asks for a list of the HTTP methods to which the thing at the request URL can respond

4. Using Cookies

Cookies in Servlet

A **cookie** is a small piece of information that is persisted between the multiple client requests.

A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

How Cookie works

By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.

Types of Cookie

There are 2 types of cookies in servlets.

1. Non-persistent cookie
2. Persistent cookie

Non-persistent cookie

It is **valid for single session** only. It is removed each time when user closes the browser.

Persistent cookie

It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

Advantage of Cookies

1. Simplest technique of maintaining the state.
2. Cookies are maintained at client side.

Disadvantage of Cookies

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.

Cookie class

javax.servlet.http.Cookie class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

Constructor of Cookie class

Constructor	Description
Cookie()	constructs a cookie.
Cookie(String name, String value)	constructs a cookie with a specified name and value.

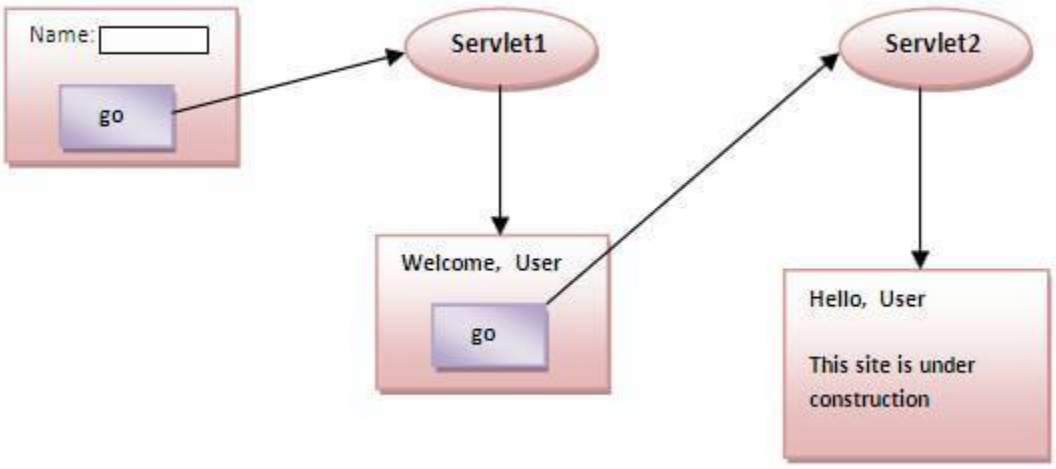
Useful Methods of Cookie class

There are given some commonly used methods of the Cookie class.

Method	Description
public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds.
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.
public String getValue()	Returns the value of the cookie.
public void setName(String name)	changes the name of the cookie.
public void setValue(String value)	changes the value of the cookie.

Simple example of Servlet Cookies

In this example, we are storing the name of the user in the cookie object and accessing it in another servlet. As we know well that session corresponds to the particular user. So if you access it from too many browsers with different values, you will get the different value.



index.html

```
1. <form action="servlet1" method="post">
2. Name:<input type="text" name="userName"/><br/>
3. <input type="submit" value="go"/>
```

4. </form>

FirstServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;



public class FirstServlet extends HttpServlet {


 public void doPost(HttpServletRequest request, HttpServletResponse response){
  try{


  response.setContentType("text/html");
  PrintWriter out = response.getWriter();


  String n=request.getParameter("userName");
  out.print("Welcome "+n);


  Cookie ck=new Cookie("uname",n);//creating cookie object
  response.addCookie(ck);//adding cookie in the response


  //creating submit button
```

```
        out.print("<form action='servlet2'>");
        out.print("<input type='submit' value='go'>");
        out.print("</form>");

        out.close();

        }catch(Exception e){System.out.println(e);}
    }
}
```

SecondServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SecondServlet extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse response){
    try{

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
```

```
        Cookie ck[]=request.getCookies();

        out.print("Hello "+ck[0].getValue());



        out.close();


            }catch(Exception e){System.out.println(e);}

        }

    }
```

5. Session tracking

HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a separate connection to the Web server and the server automatically does not keep any record of previous client request.

Still there are following three ways to maintain session between web client and web server −

Cookies

A webserver can assign a unique session ID as a cookie to each web client and for subsequent requests from the client they can be recognized using the recieved cookie.

This may not be an effective way because many time browser does not support a cookie, so I would not recommend to use this procedure to maintain the sessions.

Hidden Form Fields

A web server can send a hidden HTML form field along with a unique session ID as follows −

<input type = "hidden" name = "sessionid" value = "12345">

This entry means that, when the form is submitted, the specified name and value are automatically included in the GET or POST data. Each time when web browser sends request back, then session_id value can be used to keep the track of different web browsers.

This could be an effective way of keeping track of the session but clicking on a regular (<A HREF...>) hypertext link does not result in a form submission, so hidden form fields also cannot support general session tracking.

URL Rewriting

You can append some extra data on the end of each URL that identifies the session, and the server can associate that session identifier with data it has stored about that session.

For example, with http://tutorialspoint.com/file.htm;sessionid = 12345, the session identifier is attached as sessionid = 12345 which can be accessed at the web server to identify the client.

URL rewriting is a better way to maintain sessions and it works even when browsers don't support cookies. The drawback of URL re-writing is that you would have to generate every URL dynamically to assign a session ID, even in case of a simple static HTML page.

The HttpSession Object

Apart from the above mentioned three ways, servlet provides HttpSession Interface which provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user.

You would get HttpSession object by calling the public method **getSession()** of HttpServletRequest, as below −

HttpSession session = request.getSession();

6. Java Server Pages

Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications. JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases.

Why to Learn JSP?

JavaServer Pages often serve the same purpose as programs implemented using the **Common Gateway Interface (CGI)**. But JSP offers several advantages in comparison with the CGI.

- Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having separate CGI files.

- JSP are always compiled before they are processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.
- JavaServer Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including **JDBC, JNDI, EJB, JAXP,** etc.
- JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

Finally, JSP is an integral part of Java EE, a complete platform for enterprise class applications. This means that JSP can play a part in the simplest applications to the most complex and demanding.

Applications of JSP

As mentioned before, JSP is one of the most widely used language over the web. I'm going to list few of them here:

JSP vs. Active Server Pages (ASP)

The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.

JSP vs. Pure Servlets

It is more convenient to write (and to modify!) regular HTML than to have plenty of println statements that generate the HTML.

JSP vs. Server-Side Includes (SSI)

SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.

JSP vs. JavaScript

JavaScript can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.

JSP vs. Static HTML

Regular HTML, of course, cannot contain dynamic information.

7. Anatomy of JSP

There are many advantages of JSP over the Servlet. They are as follows:

1) Extension to Servlet

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

4) Less code than Servlet

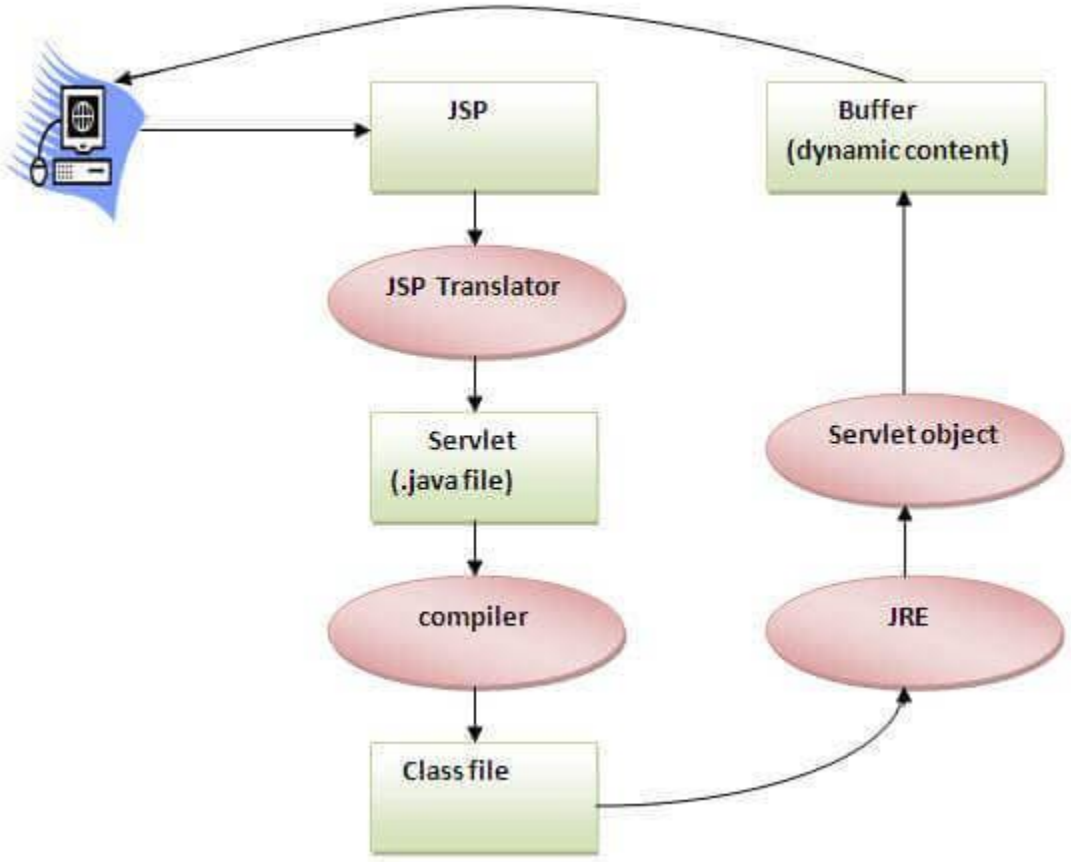
In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

The Lifecycle of a JSP Page

The JSP pages follow these phases:

- Translation of JSP Page
- Compilation of JSP Page
- Classloading (the classloader loads class file)
- Instantiation (Object of the Generated Servlet is created).
- Initialization (the container invokes jspInit() method).
- Request processing (the container invokes _jspService() method).
- Destroy (the container invokes jspDestroy() method).

Note: jspInit(), _jspService() and jspDestroy() are the life cycle methods of JSP.



As depicted in the above diagram, JSP page is translated into Servlet by the help of JSP translator. The JSP translator is a part of the web server which is responsible for translating the JSP page into Servlet. After that, Servlet page is compiled by the compiler and gets converted into the class file. Moreover, all the processes that happen in Servlet are performed on JSP later like initialization, committing response to the browser and destroy.

8. Implicit JSP Objects

These Objects are the Java objects that the JSP Container makes available to the developers in each page and the developer can call them directly without being explicitly declared. JSP Implicit Objects are also called **pre-defined variables**.

Following table lists out the nine Implicit Objects that JSP supports −

S.No.	Object & Description
1	**request**

This is the **HttpServletRequest** object associated with the request.

	response	
2		This is the **HttpServletResponse** object associated with the response to the client.
	out	
3		This is the **PrintWriter** object used to send output to the client.
	session	
4		This is the **HttpSession** object associated with the request.
	application	
5		This is the **ServletContext** object associated with the application context.
	config	
6		This is the **ServletConfig** object associated with the page.
	pageContext	
7		This encapsulates use of server-specific features like higher performance **JspWriters**.
	page	
8		This is simply a synonym for **this**, and is used to call the methods defined by the translated servlet class.
	Exception	
9		The **Exception** object allows the exception data to be accessed by designated JSP.

The request Object

The request object is an instance of a **javax.servlet.http.HttpServletRequest** object. Each time a client requests a page the JSP engine creates a new object to represent that request.

The request object provides methods to get the HTTP header information including form data, cookies, HTTP methods etc.

The response Object

The response object is an instance of a **javax.servlet.http.HttpServletResponse** object. Just as the server creates the request object, it also creates an object to represent the response to the client.

The response object also defines the interfaces that deal with creating new HTTP headers. Through this object the JSP programmer can add new cookies or date stamps, HTTP status codes, etc.

The out Object

The out implicit object is an instance of a **javax.servlet.jsp.JspWriter** object and is used to send content in a response.

The initial JspWriter object is instantiated differently depending on whether the page is buffered or not. Buffering can be easily turned off by using the **buffered = 'false'** attribute of the page directive.

The JspWriter object contains most of the same methods as the **java.io.PrintWriter** class. However, JspWriter has some additional methods designed to deal with buffering. Unlike the PrintWriter object, JspWriter throws **IOExceptions**.

Following table lists out the important methods that we will use to write **boolean char, int, double, object, String**, etc.

S.No.	Method & Description
1	**out.print(dataType dt)** Print a data type value
2	**out.println(dataType dt)** Print a data type value then terminate the line with new line character.
3	**out.flush()** Flush the stream.

The session Object

The session object is an instance of **javax.servlet.http.HttpSession** and behaves exactly the same way that session objects behave under Java Servlets.

The session object is used to track client session between client requests.

The application Object

The application object is direct wrapper around the **ServletContext** object for the generated Servlet and in reality an instance of a **javax.servlet.ServletContext** object.

This object is a representation of the JSP page through its entire lifecycle. This object is created when the JSP page is initialized and will be removed when the JSP page is removed by the **jspDestroy()** method.

By adding an attribute to application, you can ensure that all JSP files that make up your web application have access to it.

The config Object

The config object is an instantiation of **javax.servlet.ServletConfig** and is a direct wrapper around the **ServletConfig** object for the generated servlet.

This object allows the JSP programmer access to the Servlet or JSP engine initialization parameters such as the paths or file locations etc.

The following **config** method is the only one you might ever use, and its usage is trivial −

config.getServletName();

This returns the servlet name, which is the string contained in the **<servlet-name>** element defined in the **WEB-INF\web.xml** file.

The pageContext Object

The pageContext object is an instance of a **javax.servlet.jsp.PageContext** object. The pageContext object is used to represent the entire JSP page.

This object is intended as a means to access information about the page while avoiding most of the implementation details.

This object stores references to the request and response objects for each request. The **application, config, session**, and out objects are derived by accessing attributes of this object.

The pageContext object also contains information about the directives issued to the JSP page, including the buffering information, the errorPageURL, and page scope.

The PageContext class defines several fields, including **PAGE_SCOPE, REQUEST_SCOPE, SESSION_SCOPE,** and **APPLICATION_SCOPE**, which identify the four scopes. It also supports more than 40 methods, about half of which are inherited from the **javax.servlet.jsp.JspContext class**.

One of the important methods is **removeAttribute**. This method accepts either one or two arguments. For example, **pageContext.removeAttribute ("attrName")** removes the attribute from all scopes, while the following code only removes it from the page scope −

pageContext.removeAttribute("attrName", PAGE_SCOPE);

The page Object

This object is an actual reference to the instance of the page. It can be thought of as an object that represents the entire JSP page.

The page object is really a direct synonym for the **this** object.

The exception Object

The exception object is a wrapper containing the exception thrown from the previous page. It is typically used to generate an appropriate response to the error condition.

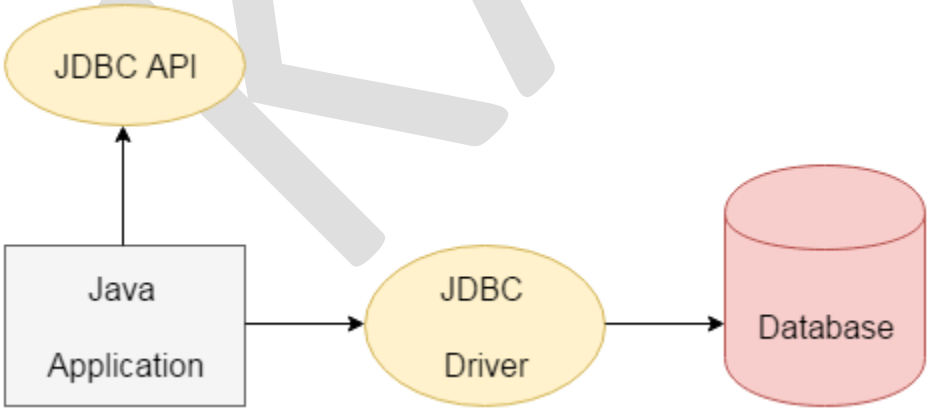
9. JDBC

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

- JDBC-ODBC Bridge Driver,
- Native Driver,
- Network Protocol Driver, and
- Thin Driver

We have discussed the above four drivers in the next chapter.

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.



The current version of JDBC is 4.3. It is the stable release since 21st September, 2017. It is based on the X/Open SQL Call Level Interface. The **java.sql** package contains classes and interfaces for JDBC API. A list of popular *interfaces* of JDBC API are given below:

- Driver interface
- Connection interface
- Statement interface
- PreparedStatement interface
- CallableStatement interface
- ResultSet interface
- ResultSetMetaData interface
- DatabaseMetaData interface
- RowSet interface

A list of popular *classes* of JDBC API are given below:

- DriverManager class
- Blob class
- Clob class
- Types class

Why Should We Use JDBC

Before JDBC, ODBC API was the database API to connect and execute the query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

We can use JDBC API to handle database using Java program and can perform the following activities:

1. Connect to the database
2. Execute queries and update statements to the database
3. Retrieve the result received from the database.

10. Java Beans

A JavaBean is a Java class that should follow the following conventions:

- It should have a no-arg constructor.
- It should be Serializable.
- It should provide methods to set and get the values of the properties, known as getter and setter methods.**Why use JavaBean?**

According to Java white paper, it is a reusable software component. A bean encapsulates many objects into one object so that we can access this object from multiple places. Moreover, it provides easy maintenance.

Simple example of JavaBean class

```
//Employee.java


package mypack;
public class Employee implements java.io.Serializable{
private int id;
private String name;
public Employee(){}
public void setId(int id){this.id=id;}
public int getId(){return id;}
public void setName(String name){this.name=name;}
public String getName(){return name;}
}
```

How to access the JavaBean class?

To access the JavaBean class, we should use getter and setter methods.

```
package mypack;
public class Test{
public static void main(String args[]){
Employee e=new Employee();//object is created
e.setName("Arjun");//setting value to the object
System.out.println(e.getName());
}}
```

Note: There are two ways to provide values to the object. One way is by constructor and second is by setter method.

JavaBean Properties

A JavaBean property is a named feature that can be accessed by the user of the object. The feature can be of any Java data type, containing the classes that you define.

A JavaBean property may be read, write, read-only, or write-only. JavaBean features are accessed through two methods in the JavaBean's implementation class:

1. getPropertyName ()

For example, if the property name is firstName, the method name would be getFirstName() to read that property. This method is called the accessor.

2. setPropertyName ()

For example, if the property name is firstName, the method name would be setFirstName() to write that property. This method is called the mutator.

11. Advantages of JavaBean

The following are the advantages of JavaBean:

- The JavaBean properties and methods can be exposed to another application.
- It provides an easiness to reuse the software components.

Disadvantages of JavaBean

The following are the disadvantages of JavaBean:

- JavaBeans are mutable. So, it can't take advantages of immutable objects.
- Creating the setter and getter method for each property separately may lead to the boilerplate code.

12. Enterprise Java Beans

EJB is an acronym for *enterprise java bean*. It is a specification provided by Sun Microsystems to develop secured, robust and scalable distributed applications.

To run EJB application, you need an *application server* (EJB Container) such as Jboss, Glassfish, Weblogic, Websphere etc. It performs:

a. life cycle management,
b. security,
c. transaction management, and
d. object pooling.

EJB application is deployed on the server, so it is called server side component also.

EJB is like COM (*Component Object Model*) provided by Microsoft. But, it is different from Java Bean, RMI and Web Services.

Enterprise JavaBeans (EJB) is the server-side and platform-independent Java application programming interface (API) for Java Platform, Enterprise Edition (Java EE). EJB is used to simplify the development of large distributed applications.

Enterprise JavaBeans (EJB) are reusable Java components that implement business logic and enable you to develop component-based distributed business applications.

EJBs reside in an EJB container, which provides a standard set of services such as

- Persistence,
- Security,
- Transactions, and
- Concurrency.

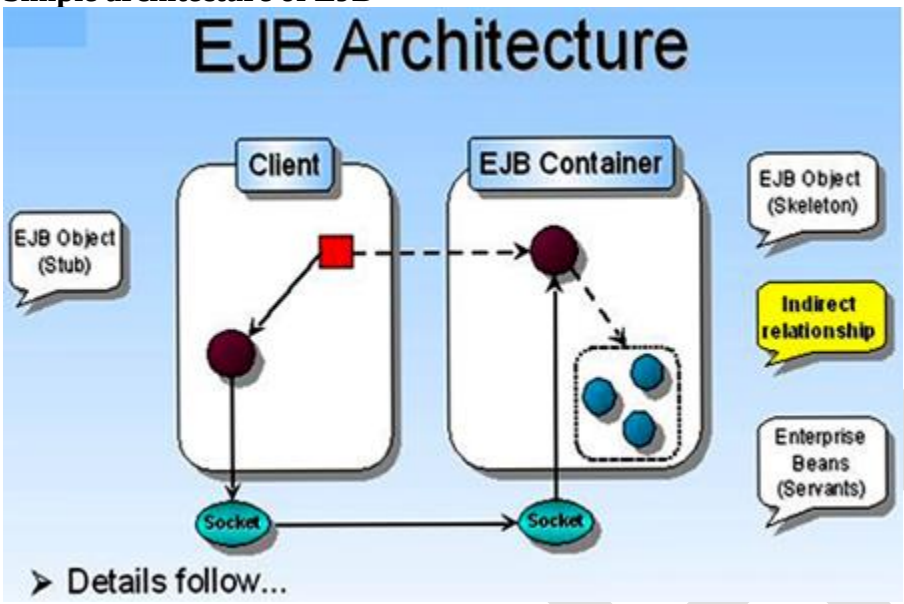
Enterprise JavaBeans are the standard for defining server-side components. WebLogic Server's implementation of the Enterprise JavaBeans component architecture is based on Sun Microsystems EJB specification.

When use Enterprise Java Bean?

1. **Application needs Remote Access**. In other words, it is distributed.
2. **Application needs to be scalable**. EJB applications supports load balancing, clustering and fail-over.
3. **Application needs encapsulated business logic**. EJB application is separated from presentation and persistent layer.

13. EJB Architecture

Simple architecture of EJB



The EJB specification defines the following **four** types of Enterprise JavaBeans:

- **Stateless session.** These non-persistent EJBs provide a service without storing an interaction or conversation state between methods.
- **Stateful session.** These non-persistent EJBs maintain state across methods and transactions. Each instance is associated with a particular client.
- **Entity.** These persistent EJBs represent an object view of the data, usually rows in a database. An entity bean has a primary key as a unique identifier.
- **Message-driven.** These EJBs are integrated with the Java Message Service (JMS) to enable message-driven beans to act as a standard JMS message consumer and perform asynchronous processing between the server and the JMS message producer.

From the EJB sub-node, of the Deployments node of the Administration Console, you can update or configure EJBs deployed on WebLogic Server or monitor the performance.

14. Types of Bean

There are 3 types of enterprise bean in java.

Session Bean

Session bean contains business logic that can be invoked by local, remote or webservice client.

Message Driven Bean

Like Session Bean, it contains the business logic but it is invoked by passing message.

Entity Bean

It encapsulates the state that can be persisted in the database. It is deprecated. Now, it is replaced with JPA (Java Persistent API).

15. EJB Transactions

A transaction is a single unit of work items, which follows the ACID properties. ACID stands for Atomic, Consistent, Isolated, and Durable.

- **Atomic** − If any of the work item fails, the whole unit will be considered failed. Success meant, all items execute successfully.
- **Consistent** − A transaction must keep the system in consistent state.
- **Isolated** − Each transaction executes independent of any other transaction.
- **Durable** − Transaction should survive system failure if it has been executed or committed.

EJB Container/Servers are transaction servers and handles transactions context propagation and distributed transactions. Transactions can be managed by the container or by custom code handling in bean's code.

- **Container Managed Transactions** − In this type, the container manages the transaction states.
- **Bean Managed Transactions** − In this type, the developer manages the life cycle of transaction states.

Container Managed Transactions

EJB 3.0 has specified following attributes of transactions, which EJB containers implement −

- **REQUIRED** − Indicates that business method has to be executed within transaction, otherwise a new transaction will be started for that method.
- **REQUIRES_NEW** − Indicates that a new transaction, is to be started for the business method.
- **SUPPORTS** − Indicates that business method will execute as part of transaction.
- **NOT_SUPPORTED** − Indicates that business method should not be executed as part of transaction.
- **MANDATORY** − Indicates that business method will execute as part of transaction, otherwise exception will be thrown.

- **NEVER** − Indicates if business method executes as part of transaction, then an exception will be thrown.

Bean Managed Transactions

In Bean Managed Transactions, Transactions can be managed by handling exceptions at application level.

Following are the key points to be considered −

- **Start** − When to start a transaction in a business method.
- **Sucess** − Identify success scenario when a transaction is to be committed.
- **Failed** − Identify failure scenario when a transaction is to be rollback.

Possible Questions:

(2 Marks)

1. **What is the difference between Servlet and JSP?**
2. **What is JSP?**
3. **What is JDBC?**
4. **What is EJB?**
5. **List out any four implicit objects in JSP.**
6. **List out types of beans.**

(8 Marks)

1. **Describe the life cycle of servlet.**
2. **Explain about Handling HTTP request and response.**
3. **Explain about session tracking.**
4. **Explain about JSP.**
5. **What are implicit objects in JSP?**
6. **Explain EJB Transactions.**

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

SUBJECT : Scripting Language **Unit V**

S No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	Which page directive should be used in JSP to generate a PDF page?	contentType	generatePdf	typePDF	contentPDF	contentType
2	Which tag should be used to pass information from JSP to included JSP?	Using <%jsp:page> tag	Using <%jsp:param> tag	Using <%jsp:import> tag	Using <%jsp:useBean> tag	Using <%jsp:param> tag
3	Application is instance of which class?	javax.servlet.Application	javax.servlet.HttpContext	javax.servlet.Context	javax.servlet.ServletContext	javax.servlet.ServletContext
4	Which option is true about session scope?	Objects are accessible only from the page in which they are created	Objects are accessible only from the pages which are in same session	Objects are accessible only from the pages which are processing the same request	Objects are accessible only from the pages which reside in same application	Objects are accessible only from the pages which are in same session
5	Which one is the correct order of phases in JSP life cycle?	Initialization, Cleanup, Compilation, Execution	Initialization, Compilation, Cleanup, Execution	Compilation, Initialization, Execution, Cleanup	Cleanup, Compilation, Initialization, Execution	Compilation, Initialization, Execution, Cleanup
6	“request” is instance of which one of the following classes?	Request	HttpRequest	HttpServletRequest	ServletRequest	HttpServletRequest
7	Which is not a directive?	include	page	export	useBean	export
8	Which is mandatory in <jsp:useBean /> tag?	id, class	id, type	type, property	type,id	id, class

9	Which one of the following is correct for directive in JSP?	<%@directive%>	<%!directive%>	<%directive%>	<%=directive%>	<%@directive%>
10	Which of the following action variable is used to include a file in JSP?	jsp:setProperty	jsp:getProperty	jsp:include	jsp:plugin	jsp:include
11	Which attribute uniquely identification element?	ID	Class	Name	Scope	ID
12	“out” is implicit object of which class?	javax.servlet.jsp.PrintWriter	javax.servlet.jsp.SessionWriter	javax.servlet.jsp.SessionPrinter	javax.servlet.jsp.JspWriter	javax.servlet.jsp.JspWriter
13	Which object stores references to the request and response objects?	sessionContext	pageContext	HttpSession	sessionAttribute	pageContext
14	What temporarily redirects response to the browser?	<jsp:forward>	<%@directive%>	response.sendRedirect(URL)	response.setRedirect(URL)	response.sendRedirect(URL)
15	Which tag is used to set a value of a JavaBean?	<c:set>	<c:param>	<c:choose>	<c:forward>	<c:set>
16	Java code is embedded under which tag in JSP?	Declaration	Scriptlet	Expression	Comment	Scriptlet
17	Which of the following is not a directive in JSP?	page directive	include directive	taglib directive	command directive	command directive
18	The difference between Servlets and JSP is the …….	translation	compilation	syntax	Both A and B	syntax
19	Which of the following is true about session bean?	This type of bean stores data of a particular user for a single session.	This is a type of enterprise bean which is invoked by EJB container when it receives a message from queue or topic.	This type of bean represents persistent data storage.	None of the above.	This type of bean stores data of a particular user for a single session.
20	Which of the following bean stores data of a particular user for a single session?	session bean.	entity bean.	message driven bean.	None of the above.	session bean.

21	Which of the following annotation is used to specify or inject a dependency as ejb instance into another ejb?	@javax.ejb.Stateless	@javax.ejb.Stateful	@javax.ejb.MessageDrivenBean	@javax.ejb.EJB	@javax.ejb.EJB
22	Which of the following annotation is used to specify Remote interface(s) of a session bean?	@javax.ejb.Stateless	@javax.ejb.Stateful	@javax.ejb.Remote	@javax.ejb.EJB	@javax.ejb.Remote
23	Which of the following is true about callbacks in EJB?	Callback is a mechanism by which life cycle of an enterprise bean can be intercepted.	EJB Container calls callbacks.	We can define callback methods in the ejb class itself or in a separate class.	All of the above.	All of the above.
24	Which of the following is correct about EJB interceptors?	EJB 3.0 provides specification to intercept business methods calls using methods annotated with @AroundInvoke annotation.	An interceptor method is called by ejbContainer before business method call it is intercepting.	Both of the above.	None of the above.	Both of the above.
25	What ACID stands for?	Atomic, Consistent,Isolated and Durable	Acurate, Correct,Isolated and Durable	Atomic, Consistent, Identical and Done	None of the above.	Atomic, Consistent,Isolated and Durable
26	Which of the following is correct about a MANDATORY attrribute of Container Managed Transactions in EJB?	Indicates that a new transaction is to be started for the business method.	Indicates that a new transaction is to be started for the business method.	Indicates that business method will execute as part of transaction.	Indicates that business method should not be executed as part of transaction.	Indicates that business method will execute as part of transaction otherwise exception will be thrown.

27	Which of the following is true about exceptions handling by EJB Container?	When Application Exception occurs, ejb container intercepts the exception but returns the same to the client as it is.	EJB Container does not roll back the transaction unless it is specified in code by EJBContext.setRollBackOnly() method.	Both of the above	None of the above.	Both of the above
28	Java bean is an ……	Structure oriented	Object oriented	Both a&b	none	Object oriented
29	Program building blocks are called as……	Components	Elements	Applications	none	Components
30	Java class should follow …… following convention	Constructor	arg constructor	no-arg constructor	Both a&b	no-arg constructor
31	EJB stands for ……	Edision java bean	Element java bean	Enterprise java bean	none	Enterprise java bean
32	EJBs provide infrastructure for developing and deploying ……..	EJB fundamentals	EJB functions	EJB elements	Mission critical	Mission critical
33	UML stands for ……	Unified Modified Language	User Modified Language	Unified Markup Language	User Markup Language	Unified Modified Language
34	The Application Logic layer is also known as …….	Upper tier	Middle tier	Lower tier	Bottom tier	Middle tier
35	Application often need to communicate with other systems is an ……. Manner.	Synchronous	Asynchronous	Authentication	none	Asynchronous
36	There are ….. kinds of EJB.	1	2	3	4	3
37	Session beans are ……..	Stateful	Stateless	Both a&b	none	Both a&b
38	Entity beans are ……	Elements	objects	Attributes	Class	objects
39	JAR stands for ……..	Java ARchitechture	Java ARchive	Java Arc	none	Java ARchive

40	JAR files are ……..	Archive files	Auto files	Compress files	De-compress files	Archive files
41	JDK stands for …….	Java Development Kit	Java Details Kit	Java Design Kit	none	Java Development Kit
42	To create a JAR file …….	jar cf jar-file input-file(s)	jar tf jar-file	jar xf jar-file	jar xf jar-file archived-file(s)	jar cf jar-file input-file(s)
43	To extract the contents of a JAR file ……..	jar cf jar-file input-file(s)	jar tf jar-file	jar xf jar-file	jar xf jar-file archived-file(s)	jar xf jar-file
44	To view the content of a JAR file ……..	jar cf jar-file input-file(s)	jar tf jar-file	jar xf jar-file	jar xf jar-file archived-file(s)	jar tf jar-file
45	To extract specific files from a JAR file ……	jar cf jar-file input-file(s)	jar tf jar-file	jar xf jar-file	jar xf jar-file archived-file(s)	jar xf jar-file archived-file(s)
46	To run an application packaged as a JAR file ……	java –jar app.jar	jar tf jar-file	jar xf jar-file	jar xf jar-file archived-file(s)	java –jar app.jar
47	To update a JAR file……..	java –jar app.jar	jar tf jar-file	jar uf jar-file input file(s)	jar xf jar-file archived-file(s)	jar uf jar-file input file(s)
48	…… is the automatic process of analyzing a bean's design patterns to reveal the bean's properties ,events,and methods.	Introspection	Extraspection	Abstraction	none	Introspection
49	Advantages of introspection is ……..	Portability	Reuse	Inheritance	Both a&b	Both a&b
50	Introspection means analysis Of bean ……	Capabilities	Reusability	inheritance	comparability	Capabilities
51	…… is the automatic process	Introspection	Extraspection	Abstraction	none	Introspection
52	which is a java package……..	Beaninfo	infojava	javaifo	none	Beaninfo
53	API expansion is ……	advanced package interface	Application package interface	application programming interface	Advanced Programming interchange	Application package interface
54	IsBound Method comes under …… class	Bean descriptor	method descriptor	property descriptor	none	property descriptor

55	BDK means …….	Button development kit	bean development kit	bean describing kit	none	bean development kit
56	How many steps to be adapted to create a new bean……	7	3	10	6	6
57	Manifest file denoted as……..	.mf	.manif	.manifest	.mfexe	.mf
58	JVM stands for……	java virtual machine	java virtual mechanism	java versibility machine	none	java virtual machine
59						
60						
61						
62						
63						
64						

Reg.No ----------

[18ITU404A & 18CTU404A]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

FIRST INTERNAL EXAMINATION, DECEMBER 2019

Fourth Semester

SCRIPTING LANGUAGE

Class: II B.Sc. IT & II B.Sc. CT **Time : 2 Hours**

Date/Session: 19/12/19 AN **Maximum : 50**

PART-A **[20 * 1 = 20 Marks]**

(Answer all of the following)

1.Which one is not a Naming Convention in VB Script?

a) names must start with an alphabetic character b) must be unique

c) cannot contain an embedded period **d) cannot exceed 145 chars**

2.All variables in VB Script are of the data type _____.

a) Static b) Dynamic

c) Variant d) None of the above

3.Conditional statements available in VB Script are

a) If Else Statement b) Switch Statement

c) Both a) and b) d) Non of a) and b)

4.We can use a _____ to run a block of code, when we know how many repetitions you want.

a) For Each...Next statement b) Do...Loop statement

c) While...Wend statement **d) For...Next statement**

5.We can exit a Do...Loop statement with the_____ .

a) Exit keyword **b) Exit Do keyword** c) Quit keyword d) Quit Do keyword

6.Declaring a variable before its use is optional, although it's a good practice to do so and to make the declaration mandatory there is an _____ available

a) "Explicit" Statement b) "Exit Do" Statement

c) "Option Explicit" Statement d) "Exit" Statement

7.In many situations, using _____ is the most efficient method of remembering and tracking preferences, purchases, commissions and other information required for method of remembering and tracking preferences, purchases, commissions and other information visitor experience or site statistics.

a) Temp memory b) Log memory

c) Session memory **d) Cookies**

8.The following are not relevant to a Function procedure is :

a) is a series of statements, enclosed by the Function and End Function statements

b) returns a value by assigning a value to its name

c) without arguments, must include an empty set of parentheses ()

d) can perform actions, but does not return a value

9.Which of the following keyword is used to declare a variable in VBScript?

a) Variant b) Var **c) Dim** d) None of the above

10.How will you get a combined string from array of string in VBScript?
a) Using Join function b) Using Filter function
c) Using IsArray function d) Using Erase Function
11.How will you trim the spaces on the right of a string using VBScript?
a) Using Lcase function b) Using Ucase function
c) Using Ltrim function **d) Using Rtrim function**
12.How will you get the absolute value of the given number in VBScript?
a) Using Abs function b) Using Exp function
c) Using InStr function d) Using InStrRev function
13.Which of the following function of VBScript converts a given number of any variant subtype to Long?
a) CDbl b) CInt **c) CLng** d) CSng
14.Which built-in method adds one or more elements to the end of an array and returns the new length of the array in Java Script?
a) last() b) put() **c) push()** d) None of the above.
15.Which of the following is the correct syntax to create a cookie using JavaScript?
a) document.cookie = 'key1 = value1; key2 = value2; expires = date';
b) browser.cookie = 'key1 = value1; key2 = value2; expires = date';
c) window.cookie = 'key1 = value1; key2 = value2; expires = date';
d) navigator.cookie = 'key1 = value1; key2 = value2; expires = date';
16.Which of the following is true about typeof operator in JavaScript?
a) The typeof is a unary operator that is placed before its single operand, which can be of any type.
b) Its value is a string indicating the data type of the operand.
c) Both of the above. d) None of the above.
17.Which of the following operator can be used to check if two numbers are equal or not in VBScript?
a) != **b)** <> c) not d) None of the above
18. What is the output of A & B in VBScript if A = 5 and B = 10?
a) 15 **b) 510** c) 5 d) None of these
19.Which loop is used to iterate till a condition becomes true in VB Script?
a) For Next loop b) For Each Next loop
c) Do While loop **d) Do Until loop**
20.The command used in VB Script for writing some text on a page is
a) document.write(). b) Msgbox
c) A and B are correct. d) None of these

PART-B [3 * 2 = 6 Marks]

(Answer all of the following)

21. What are comments in VB Script?
Comments are used to document the program logic and the user information with which other programmers can seamlessly work on the same code in future. It can include information such as developed by, modified by and it can also include incorporated logic. Comments are ignored by the interpreter while execution. Comments in VBScript are denoted by two methods.
22. What is the difference between sub procedure and function procedure?
A Sub procedure:
- is a series of statements, enclosed by the Sub and End Sub statements
- can perform actions, but does not return a value
- can take arguments that are passed to it by a calling procedure
- without arguments, must include an empty set of parentheses ()

A Function procedure:
- is a series of statements, enclosed by the Function and End Function statements
- can perform actions and can return a value
- can take arguments that are passed to it by a calling procedure
- without arguments, must include an empty set of parentheses ()

- returns a value by assigning a value to its name

23. Why Java Script is called as a Scripting Language?
Being a scripting language, JavaScript cannot run on its own. In fact, the browser is responsible for running JavaScript code. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it is up to the browser to execute it.

PART-C **[3* 8 = 24 Marks]**

(Answer ALL of the following)

24. a. Explain about variables in VB Script.

A variable is nothing but a space in the computer's memory that can store certain information. This information is bound to change from time to time. Where the information goes physically is immaterial but when needed, it can be accessed or changed by addressing the name of the variable.

E.g: If there is a statement that we want to run several times, we could use a variable to contain that count. Say X. X is a variable that can be used to store, change and use the space in the memory where we want to keep the count.

All variables are of the data type Variant. Declaring a variable before its use is optional, although it's a good practice to do so. To make the declaration mandatory there is an "Option Explicit" Statement available. To declare variables:

Dim x – This declares x

Dim x, y, z – This declares multiple variables

X=10 – This is how a value is assigned. As a general rule, the variable is the left-hand side component and the right is its value.

X="Swati" – this is the way a string value is assigned.

To make declarations mandatory this is how the code has to be written:

Option Explicit

Dim x, stri

If Option explicit statement was not used, we could have directly written:

x=100

stri="Swati"

and it would not have thrown an error.

Naming convention: Names must start with an alphabetic character, must be unique, cannot contain an embedded period and cannot exceed 255 chars.

A variable containing a single value is a scalar variable and the one that has more than one is an array. A one dimensional Array can be declared as Dim A(10). All the arrays in VB Script are zero-based that means the array index starts from 0 through the number declared. That means, our array A has 11 elements. Starting from 0 to 10.

To declare a 2-dimensional array simply separate the row count and column count by a comma. Eg: Dim A(5, 3). This means it has 6 rows and 4 columns. The first number is always row and the second a column.

This piece of code shows how we do it. Initially, A is an 11 by 11 array. Then we are resizing it to be an 11 by 21 array and the preserve statement will make sure that the data that is previously contained in the array is not lost.

(OR)

b. Describe about various operators .in VB Script.

Some of the important operators that are most commonly used are:

Artithmetic Operators

Operator	Description
+	Addition
-	Subtraction

*	Multiplication
/	Division
\	Integer Division (divides two numbers and returns an integer result)
Mod	Modulus (remainder of a division)
^	Exponentiation (raises the number to the power of an exponent)

Assignment Operator

Operator	Description
=	Assign

Comparison Operators

Operator	Description
=	Is equal to
<>	Is not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Logical Operators

Operator	Description
And	Performs a logical conjunction on two expressions (if both expressions evaluate to True, result is True. If either expression evaluates to False, result is False)
Or	Performs a logical disjunction on two expressions (if either or both expressions evaluate to True, result is True).
Not	Performs logical negation on an expression.
Xor	Performs a logical exclusion on two expressions (if one, and only one, of the expressions evaluates to True, result is True. However, if either expression is Null, result is also Null).

Concatenation Operators

Operator	Description
&	Concatenate (join two strings together)
+	Adds two numbers together

25. a. Explain about conditional statements in VB Script.
    Decision making allows programmers to control the execution flow of a script or one of its sections. The execution is governed by one or more conditional statements.

Following is the general form of a typical decision making structure found in most of the programming languages −

VBScript provides the following types of decision making statements.

Statement	Description
if statement	An if statement consists of a Boolean expression followed by one or more statements.
if..else statement	An if else statement consists of a Boolean expression followed by one or more statements. If the condition is True, the statements under the If statements are executed. If the condition is false, then the Else part of the script is Executed
if...elseif..else statement	An if statement followed by one or more ElseIf Statements, that consists of Boolean expressions and then followed by an optional else statement, which executes when all the condition becomes false.
nested if statements	An if or elseif statement inside another if or elseif statement(s).
switch statement	A switch statement allows a variable to be tested for equality against a list of values.

If....Then.....Else

You should use the If...Then...Else statement if you want to

- execute some code if a condition is true
- select one of two blocks of code to execute

If you want to execute only one statement when a condition is true, you can write the code on one line:

```
if i=10 Then msgbox "Hello"
```

There is no ..else.. in this syntax. You just tell the code to perform one action if the condition is true (in this case if i=10).

If you want to execute more than one statement when a condition is true, you must put each statement on separate lines and end the statement with the keyword "End If":

```
if i=10 Then
  msgbox "Hello"
  i = i+1
end If
```

There is no ..else.. in this syntax either. You just tell the code to perform multiple actions if the condition is true.

If you want to execute a statement if a condition is true and execute another statement if the condition is not true, you must add the "Else" keyword:

```
if i=10 then
  msgbox "Hello"
else
  msgbox "Goodbye"
end If
```

The first block of code will be executed if the condition is true, and the other block will be executed otherwise (if i is not equal to 10).

If....Then.....Elseif

You can use the if...then...elseif statement if you want to select one of many blocks of code to execute:

```
if payment="Cash" then
  msgbox "You are going to pay cash!"
 elseif payment="Visa" then
  msgbox "You are going to pay with visa."
 elseif payment="AmEx" then
  msgbox "You are going to pay with American Express."
 else
  msgbox "Unknown method of payment."
```

end If

Select Case

You can also use the SELECT statement if you want to select one of many blocks of code to execute:

```
select case payment
 case "Cash"
   msgbox "You are going to pay cash"
 case "Visa"
   msgbox "You are going to pay with visa"
 case "AmEx"
   msgbox "You are going to pay with American Express"
 case Else
   msgbox "Unknown method of payment"
end select
```

This is how it works: First we have a single expression (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each Case in the structure. If there is a match, the block of code associated with that Case is executed.

(OR)

b. Discuss about looping constructs in VB Script.

Very often when you write code, you want to allow the same block of code to run a number of times. You can use looping statements in your code to do this.

In VBScript we have four looping statements:

- For...Next statement - runs statements a specified number of times.
- For Each...Next statement - runs statements for each item in a collection or each element of an array
- Do...Loop statement - loops while or until a condition is true
- While...Wend statement - Do not use it - use the Do...Loop statement instead

---

For...Next Loop

You can use a For...Next statement to run a block of code, when you know how many repetitions you want.

You can use a counter variable that increases or decreases with each repetition of the loop, like this:

```
For i=1 to 10
  some code
Next
```

The For statement specifies the counter variable (i) and its start and end values. The Next statement increases the counter variable (i) by one.

Step Keyword

Using the Step keyword, you can increase or decrease the counter variable by the value you specify. In the example below, the counter variable (i) is increased by two each time the loop repeats.

```
For i=2 To 10 Step 2
  some code
Next
```

To decrease the counter variable, you must use a negative Step value. You must specify an end value that is less than the start value.

In the example below, the counter variable (i) is decreased by two each time the loop repeats.

```
For i=10 To 2 Step -2
  some code
Next
```

Exit a For...Next

You can exit a For...Next statement with the Exit For keyword.

For Each...Next Loop

A For Each...Next loop repeats a block of code for each item in a collection, or for each element of an array.

```
dim cars(2)
cars(0)="Volvo"
cars(1)="Saab"
cars(2)="BMW"

For Each x in cars
  document.write(x & "<br />")
Next
```

Do...Loop

You can use Do...Loop statements to run a block of code when you do not know how many repetitions you want. The block of code is repeated while a condition is true or until a condition becomes true.

Repeating Code While a Condition is True

You use the While keyword to check a condition in a Do...Loop statement.

```
Do While i>10
  some code
Loop
```

If i equals 9, the code inside the loop above will never be executed.

```
Do
  some code
Loop While i>10
```

The code inside this loop will be executed at least one time, even if i is less than 10.

Repeating Code Until a Condition Becomes True

You use the Until keyword to check a condition in a Do...Loop statement.

```
Do Until i=10
  some code
Loop
```

If i equals 10, the code inside the loop will never be executed.

```
Do
  some code
Loop Until i=10
```

The code inside this loop will be executed at least one time, even if i is equal to 10.

Exit a Do...Loop

You can exit a Do...Loop statement with the Exit Do keyword.

```
Do Until i=10
  i=i-1
  If i<10 Then Exit Do
Loop
```

The code inside this loop will be executed as long as i is different from 10, and as long as i is greater than 10.

26. a. Explain in detail about Cookies.

    Web Browsers and Servers use HTTP protocol to communicate and HTTP is a stateless protocol. But for a commercial website, it is required to maintain session information among different pages. For example, one user registration ends after completing many pages. But how to maintain user's session information across all the web pages. In many situations, using cookies is the most efficient method of remembering and tracking preferences, purchases, commissions and other information required for better visitor experience or site statistics.

    How It Works?

    Your server sends some data to the visitor's browser in the form of a cookie. The browser may accept the cookie. If it does, it is stored as a plain text record on the visitor's hard drive. Now, when the visitor arrives at another page on your site, the browser sends the same cookie to the

server for retrieval. Once retrieved, your server knows/remembers what was stored earlier.
Cookies are a plain text data record of 5 variable-length fields −
• Expires − The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.
• Domain − The domain name of your site.
• Path − The path to the directory or web page that set the cookie. This may be blank if you want to retrieve the cookie from any directory or page.
• Secure − If this field contains the word "secure", then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists.
• Name=Value − Cookies are set and retrieved in the form of key and value pairs.
Cookies were originally designed for CGI programming and cookies' data is automatically transmitted between the web browser and web server, so CGI scripts on the server can read and write cookie values that are stored on the client.
VBScript can also manipulate cookies using the cookie property of the Document object.
VBScript can read, create, modify and delete the cookie or cookies that apply to the current web page.
Storing Cookies
The simplest way to create a cookie is to assign a string value to the document.cookie object, which looks like this −
Syntax
document.cookie = "key1 = value1;key2 = value2;expires = date"
Here expires attribute is optional. If you provide this attribute with a valid date or time, then cookie will expire at the given date or time and after that cookies' value will not be accessible.

(OR)

b. Explain about various operators in Java Script.

JavaScript supports the following types of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Lets have a look on all operators one by one.

Arithmetic Operators

JavaScript supports the following arithmetic operators −

Assume variable A holds 10 and variable B holds 20, then −

Sr.No.	Operator & Description
1	**+ (Addition)** Adds two operands **Ex:** A + B will give 30
2	**- (Subtraction)**

Subtracts the second operand from the first

Ex: A - B will give -10

*** (Multiplication)**

3 Multiply both operands

Ex: A * B will give 200

/ (Division)

4 Divide the numerator by the denominator

Ex: B / A will give 2

% (Modulus)

5 Outputs the remainder of an integer division

Ex: B % A will give 0

++ (Increment)

6 Increases an integer value by one

Ex: A++ will give 11

-- (Decrement)

7 Decreases an integer value by one

Ex: A-- will give 9

Note − Addition operator (+) works for Numeric as well as Strings. e.g. "a" + 10 will give "a10".

Example

The following code shows how to use arithmetic operators in JavaScript.

```
<html>
   <body>

      <script type = "text/javascript">
         <!--
            var a = 33;
            var b = 10;
            var c = "Test";
            var linebreak = "<br />";

            document.write("a + b = ");
            result = a + b;
            document.write(result);
            document.write(linebreak);
```

```
            document.write("a - b = ");
            result = a - b;
            document.write(result);
            document.write(linebreak);

            document.write("a / b = ");
            result = a / b;
            document.write(result);
            document.write(linebreak);

            document.write("a % b = ");
            result = a % b;
            document.write(result);
            document.write(linebreak);

            document.write("a + b + c = ");
            result = a + b + c;
            document.write(result);
            document.write(linebreak);

            a = ++a;
            document.write("++a = ");
            result = ++a;
            document.write(result);
            document.write(linebreak);

            b = --b;
            document.write("--b = ");
            result = --b;
            document.write(result);
            document.write(linebreak);
         //-->
      </script>

      Set the variables to different values and then try...
   </body>
</html>
```

Output

```
a + b = 43
a - b = 23
a / b = 3.3
a % b = 3
a + b + c = 43Test
++a = 35
--b = 8
Set the variables to different values and then try...
```

Comparison Operators

JavaScript supports the following comparison operators −

Assume variable A holds 10 and variable B holds 20, then −

Sr.No.	Operator & Description

	= = (Equal)
1	Checks if the value of two operands are equal or not, if yes, then the condition becomes true. **Ex:** (A == B) is not true.
	!= (Not Equal)
2	Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. **Ex:** (A != B) is true.
	> (Greater than)
3	Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true. **Ex:** (A > B) is not true.
	< (Less than)
4	Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true. **Ex:** (A < B) is true.
	>= (Greater than or Equal to)
5	Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true. **Ex:** (A >= B) is not true.
	<= (Less than or Equal to)
6	Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true. **Ex:** (A <= B) is true.

Example

The following code shows how to use comparison operators in JavaScript.

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var a = 10;
            var b = 20;
            var linebreak = "<br />";

            document.write("(a == b) => ");
```

```
            result = (a == b);
            document.write(result);
            document.write(linebreak);

            document.write("(a < b) => ");
            result = (a < b);
            document.write(result);
            document.write(linebreak);

            document.write("(a > b) => ");
            result = (a > b);
            document.write(result);
            document.write(linebreak);

            document.write("(a != b) => ");
            result = (a != b);
            document.write(result);
            document.write(linebreak);

            document.write("(a >= b) => ");
            result = (a >= b);
            document.write(result);
            document.write(linebreak);

            document.write("(a <= b) => ");
            result = (a <= b);
            document.write(result);
            document.write(linebreak);
         //-->
      </script>
      Set the variables to different values and different operators and
then try...
   </body>
</html>
```

Output

```
(a == b) => false
(a < b) => true
(a > b) => false
(a != b) => true
(a >= b) => false
a <= b) => true
Set the variables to different values and different operators and then
try...
```

Logical Operators

JavaScript supports the following logical operators −

Assume variable A holds 10 and variable B holds 20, then −

Sr.No.	Operator & Description
1	**&& (Logical AND)**

If both the operands are non-zero, then the condition becomes true.

Ex: (A && B) is true.

|| (Logical OR)

2 If any of the two operands are non-zero, then the condition becomes true.

Ex: (A || B) is true.

! (Logical NOT)

3 Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false.

Ex: ! (A && B) is false.

Example

Try the following code to learn how to implement Logical Operators in JavaScript.

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var a = true;
            var b = false;
            var linebreak = "<br />";

            document.write("(a && b) => ");
            result = (a && b);
            document.write(result);
            document.write(linebreak);

            document.write("(a || b) => ");
            result = (a || b);
            document.write(result);
            document.write(linebreak);

            document.write("!(a && b) => ");
            result = (!(a && b));
            document.write(result);
            document.write(linebreak);
         //-->
      </script>
      <p>Set the variables to different values and different operators and
then try...</p>
   </body>
</html>
```

Output

```
(a && b) => false
(a || b) => true
!(a && b) => true
Set the variables to different values and different operators and then
try...
```

Bitwise Operators

JavaScript supports the following bitwise operators −

Assume variable A holds 2 and variable B holds 3, then −

Sr.No.	Operator & Description
1	**& (Bitwise AND)** It performs a Boolean AND operation on each bit of its integer arguments. **Ex:** (A & B) is 2.
2	**| (BitWise OR)** It performs a Boolean OR operation on each bit of its integer arguments. **Ex:** (A | B) is 3.
3	**^ (Bitwise XOR)** It performs a Boolean exclusive OR operation on each bit of its integer arguments. Exclusive OR means that either operand one is true or operand two is true, but not both. **Ex:** (A ^ B) is 1.
4	**~ (Bitwise Not)** It is a unary operator and operates by reversing all the bits in the operand. **Ex:** (~B) is -4.
5	**<< (Left Shift)** It moves all the bits in its first operand to the left by the number of places specified in the second operand. New bits are filled with zeros. Shifting a value left by one position is equivalent to multiplying it by 2, shifting two positions is equivalent to multiplying by 4, and so on. **Ex:** (A << 1) is 4.
6	**>> (Right Shift)** Binary Right Shift Operator. The left operand's value is moved right by the number of bits specified by the right operand. **Ex:** (A >> 1) is 1.
7	**>>> (Right shift with Zero)** This operator is just like the >> operator, except that the bits shifted in on the left are

always zero.

Ex: (A >>> 1) is 1.

Example

Try the following code to implement Bitwise operator in JavaScript.

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var a = 2; // Bit presentation 10
            var b = 3; // Bit presentation 11
            var linebreak = "<br />";

            document.write("(a & b) => ");
            result = (a & b);
            document.write(result);
            document.write(linebreak);

            document.write("(a | b) => ");
            result = (a | b);
            document.write(result);
            document.write(linebreak);

            document.write("(a ^ b) => ");
            result = (a ^ b);
            document.write(result);
            document.write(linebreak);

            document.write("(~b) => ");
            result = (~b);
            document.write(result);
            document.write(linebreak);

            document.write("(a << b) => ");
            result = (a << b);
            document.write(result);
            document.write(linebreak);

            document.write("(a >> b) => ");
            result = (a >> b);
            document.write(result);
            document.write(linebreak);
         //-->
      </script>
      <p>Set the variables to different values and different operators and
then try...</p>
   </body>
</html>
(a & b) => 2
(a | b) => 3
(a ^ b) => 1
(~b) => -4
(a << b) => 16
(a >> b) => 0
Set the variables to different values and different operators and then
try...
```

Assignment Operators

JavaScript supports the following assignment operators −

Sr.No.	Operator & Description
1	**= (Simple Assignment)** Assigns values from the right side operand to the left side operand **Ex:** C = A + B will assign the value of A + B into C
2	**+= (Add and Assignment)** It adds the right operand to the left operand and assigns the result to the left operand. **Ex:** C += A is equivalent to C = C + A
3	**−= (Subtract and Assignment)** It subtracts the right operand from the left operand and assigns the result to the left operand. **Ex:** C -= A is equivalent to C = C - A
4	***= (Multiply and Assignment)** It multiplies the right operand with the left operand and assigns the result to the left operand. **Ex:** C *= A is equivalent to C = C * A
5	**/= (Divide and Assignment)** It divides the left operand with the right operand and assigns the result to the left operand. **Ex:** C /= A is equivalent to C = C / A
6	**%= (Modules and Assignment)** It takes modulus using two operands and assigns the result to the left operand. **Ex:** C %= A is equivalent to C = C % A

Note − Same logic applies to Bitwise operators so they will become like <<=, >>=, >>=, &=, |= and ^=.

Example

Try the following code to implement assignment operator in JavaScript.

```
<html>
   <body>
```

```
<script type = "text/javascript">
   <!--
      var a = 33;
      var b = 10;
      var linebreak = "<br />";

      document.write("Value of a => (a = b) => ");
      result = (a = b);
      document.write(result);
      document.write(linebreak);

      document.write("Value of a => (a += b) => ");
      result = (a += b);
      document.write(result);
      document.write(linebreak);

      document.write("Value of a => (a -= b) => ");
      result = (a -= b);
      document.write(result);
      document.write(linebreak);
```

Reg.No ----------

[18ITU404A & 18CTU404A]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

FIRST INTERNAL EXAMINATION, DECEMBER 2019

Fourth Semester

SCRIPTING LANGUAGE

Class: II B.Sc. IT & II B.Sc. CT **Time : 2 Hours**

Date/Session: /01/20 AN **Maximum : 50**

PART-A **[20 * 1 = 20 Marks]**

(Answer all of the following)

1. _____ method of Math object Returns the largest integer less than or equal to a number.
   a) max() b) abs()
   c) ceil() **d) floor()**
2. HTTP stands for_____
   a) Hyper text transmission protocol **b) hyper text transfer protocol**
   c) hyper texture transfer protocol d) none of this
3. Javascript Boolean type can have_____value.
   a) 2 b) 1
   c) 0 d) 4
4. The new keyword is followed by a call to a constructor this call _____ new object.
   a) exist b) new
   c) free **d) create**
5. In HTTP a client is a _____
   a) web page **b) web browser** c) server d) all the above
6. GUI stands for _____
   a) Graphical user interface b) Geo-graphical user interface
   c) Graphics user interface d) none
7. All global JavaScript objects, functions, and variables automatically become members of the _____ object.
   a) document **b) window**
   c) number d) math
8. The HTML DOM _____ object is the owner of all other objects in your web page.
   a) document b) window c) number d) math
9. Which one is not a HTML event?
   a) openWindow b) onchange c) onclick d) onmouseover
10. Application often need to communicate with other systems is an ……. Manner.
    a) Synchronous **b) Asynchronous**
    c) Authentication d) none
11. Which of the following event fires when the form element loses the focus: <button>, <input>, <label>, <select>, <textarea>?
    a) onfocus **b) onblur** c) onclick d) ondblclick
12. JavaScript is interpreted by _____
    a) Client b) Server
    c) Object d) None of the above

13. Which of the following attribute can hold the JavaScript version?
a) LANGUAGE b) SCRIPT c) VERSION d) None of the above
14. Which of the following is not a valid JavaScript variable name?
a) 2names b) _first_and_last_names
c) FirstAndLast d) None of the above
15. Which of the following can't be done with client-side JavaScript?
a) Validating a form b) Sending a form's contents by email
c) Storing the form's contents to a database file on the server
d) None of the above
16. When a user views a page containing a JavaScript program, which machine actually executes the script?
a) The User's machine running a Web browser b) The Web server
c) A central machine deep within Netscape's corporate offices d) None of the above.
17. What should appear at the very end of your JavaScript? The <script LANGUAGE="JavaScript">tag
a) The </script> b) The <script> c) The END statement d) None of the above
18. _____ JavaScript statements embedded in an HTML page can respond to user events such as mouse-clicks, form input, and page navigation.
a) Client-side b) Server-side c) Local d) Native
19. Which was the first browser to support JavaScript?
a) Mozilla Firefox **b) Netscape**
c) Google Chrome d) IE
20. How to create a Date object in JavaScript?
a) dateObjectName = new Date([parameters]) b) dateObjectName.new Date([parameters])
c) dateObjectName := new Date([parameters]) d) dateObjectName Date([parameters])

PART-B [3 * 2 = 6 Marks]

(Answer all of the following)

21. What are Arrays in Java Script?
JavaScript arrays are used to store multiple values in a single variable.

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

What is an Array?
An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

var car1 = "Saab";
var car2 = "Volvo";
var car3 = "BMW";
However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?
22. What is mean by recursion?
A recursive function is a function that calls itself. We will take the classic factorial function for the demonstration.
in Mathematics, the factorial of a non-negative integer is the product of all positive integer less than or equal to it. The factorial of the integer n is denoted by n!

For example, the factorial of 5 is calculated as follows

5! = 5 x 4 x 3 x 2 x 1 = 120

it is more readable if you develop the factorial function using the recursion technique. See the following recursive factorial function:

```
var factorial = function pf(n) {
   if (n <= 1) {
      return 1;
   } else {
      return n * pf(n - 1);
   }
};
```

23. What is mean by event in Java Script?

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

PART-C **[3* 8 = 24 Marks]**

(Answer ALL of the following)

24. a. Explain about Functions in Java Script.

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.

Like any other advanced programming language, JavaScript also supports all the features necessary to write modular code using functions. You must have seen functions like alert() and write() in the earlier chapters. We were using these functions again and again, but they had been written in core JavaScript only once.

JavaScript allows us to write our own functions as well. This section explains how to write your own functions in JavaScript.

Function Definition

Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the function keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax

The basic syntax is shown here.

```
<script type = "text/javascript">
  <!--
     function functionname(parameter-list) {
       statements
     }
  //-->
</script>
```

Example

Try the following example. It defines a function called sayHello that takes no parameters −

```
<script type = "text/javascript">
  <!--
     function sayHello() {
       alert("Hello there");
     }
  //-->
</script>
```

Calling a Function

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

```
<html>
  <head>
    <script type = "text/javascript">
      function sayHello() {
        document.write ("Hello there!");
      }
    </script>

  </head>

  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "sayHello()" value = "Say Hello">
    </form>
    <p>Use different text in write method and then try...</p>
  </body>
</html>
```

Output

Function Parameters

Till now, we have seen functions without parameters. But there is a facility to pass different parameters while calling a function. These passed parameters can be captured inside the function and any manipulation can be done over those parameters. A function can take multiple parameters separated by comma.

Example

Try the following example. We have modified our sayHello function here. Now it takes two parameters.

```
<html>
  <head>
    <script type = "text/javascript">
      function sayHello(name, age) {
        document.write (name + " is " + age + " years old.");
      }
    </script>
  </head>

  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "sayHello('Zara', 7)" value = "Say Hello">
    </form>
    <p>Use different parameters inside the function and then try...</p>
  </body>
</html>
```

Output

The return Statement

A JavaScript function can have an optional return statement. This is required if you want to return a value from a function. This statement should be the last statement in a function.

For example, you can pass two numbers in a function and then you can expect the function to return their multiplication in your calling program.

Example

Try the following example. It defines a function that takes two parameters and concatenates them before returning the resultant in the calling program.

```
<html>
  <head>
```

```
<script type = "text/javascript">
  function concatenate(first, last) {
    var full;
    full = first + last;
    return full;
  }
  function secondFunction() {
    var result;
    result = concatenate('Zara', 'Ali');
    document.write (result );
  }
</script>
</head>

<body>
  <p>Click the following button to call the function</p>
  <form>
    <input type = "button" onclick = "secondFunction()" value = "Call Function">
  </form>
  <p>Use different parameters inside the function and then try...</p>
</body>
</html>
```

Output

There is a lot to learn about JavaScript functions, however we have covered the most important concepts in this section.

(OR)

b. What is recursion in Java Script? Explain with example.

A recursive function is a function that calls itself. We will take the classic factorial function for the demonstration.

in Mathematics, the factorial of a non-negative integer is the product of all positive integer less than or equal to it. The factorial of the integer n is denoted by n!

For example, the factorial of 5 is calculated as follows

5! = 5 x 4 x 3 x 2 x 1 = 120

it is more readable if you develop the factorial function using the recursion technique. See the following recursive factorial function:

```
var factorial = function pf(n) {
   if (n <= 1) {
      return 1;
   } else {
      return n * pf(n - 1);
   }
};
```

This is how it works. If n is equal to one or zero, the factorial of n is 1; otherwise, the factorial of n is the product of n and factorial of n - 1.

25. a. Explain about Math Object in detail in Java Script.

The math object provides you properties and methods for mathematical constants and functions. Unlike other global objects, Math is not a constructor. All the properties and methods of Math are static and can be called by using Math as an object without creating it.

Thus, you refer to the constant pi as Math.PI and you call the sine function as Math.sin(x), where x is the method's argument.

Syntax
The syntax to call the properties and methods of Math are as follows

```
var pi_val = Math.PI;
var sine_val = Math.sin(30);
```

Math Properties
Here is a list of all the properties of Math and their description.

Sr.No.	Property & Description
1	E \ Euler's constant and the base of natural logarithms, approximately 2.718.
2	LN2 Natural logarithm of 2, approximately 0.693.
3	LN10 Natural logarithm of 10, approximately 2.302.
4	LOG2E Base 2 logarithm of E, approximately 1.442.
5	LOG10E Base 10 logarithm of E, approximately 0.434.
6	PI Ratio of the circumference of a circle to its diameter, approximately 3.14159.
7	SQRT1_2 Square root of 1/2; equivalently, 1 over the square root of 2, approximately 0.707.
8	SQRT2 Square root of 2, approximately 1.414.

In the following sections, we will have a few examples to demonstrate the usage of Math properties.

Math Methods
Here is a list of the methods associated with Math object and their description

Sr.No.	Method & Description
1	abs() Returns the absolute value of a number.
2	acos() Returns the arccosine (in radians) of a number.
3	asin() Returns the arcsine (in radians) of a number.
4	atan() Returns the arctangent (in radians) of a number.
5	atan2() Returns the arctangent of the quotient of its arguments.

6 ceil()
Returns the smallest integer greater than or equal to a number.

7 cos()
Returns the cosine of a number.

8 exp()
Returns EN, where N is the argument, and E is Euler's constant, the base of the natural logarithm.

9 floor()
Returns the largest integer less than or equal to a number.

10 log()
Returns the natural logarithm (base E) of a number.

11 max()
Returns the largest of zero or more numbers.

12 min()
Returns the smallest of zero or more numbers.

13 pow()
Returns base to the exponent power, that is, base exponent.

14 random()
Returns a pseudo-random number between 0 and 1.

15 round()
Returns the value of a number rounded to the nearest integer.

16 sin()
Returns the sine of a number.

17 sqrt()
Returns the square root of a number.

18 tan()
Returns the tangent of a number.

19 toSource()
Returns the string "Math".

(OR)

b. Discuss about String Object in detail in Java Script.

The String object lets you work with a series of characters; it wraps Javascript's string primitive data type with a number of helper methods.

As JavaScript automatically converts between string primitives and String objects, you can call any of the helper methods of the String object on a string primitive.

Syntax
Use the following syntax to create a String object −

```
var val = new String(string);
```

The String parameter is a series of characters that has been properly encoded.

String Properties
Here is a list of the properties of String object and their description.

Sr.No. Property & Description
1 constructor
Returns a reference to the String function that created the object.

2 length
Returns the length of the string.

3 prototype
The prototype property allows you to add properties and methods to an object.

In the following sections, we will have a few examples to demonstrate the usage of String properties.

String Methods
Here is a list of the methods available in String object along with their description.

Sr.No. Method & Description
1 charAt()
Returns the character at the specified index.

2 charCodeAt()
Returns a number indicating the Unicode value of the character at the given index.

3 concat()
Combines the text of two strings and returns a new string.

4 indexOf()
Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.

5 lastIndexOf()
Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.

6 localeCompare()
Returns a number indicating whether a reference string comes before or after or is the same as the given string in sort order.

7 match()
Used to match a regular expression against a string.

8 replace()
Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.

9 search()
Executes the search for a match between a regular expression and a specified string.

10 slice()
Extracts a section of a string and returns a new string.

11 split()
Splits a String object into an array of strings by separating the string into substrings.

12 substr()
Returns the characters in a string beginning at the specified location through the specified number of characters.

13 substring()
Returns the characters in a string between two indexes into the string.

14 toLocaleLowerCase()
The characters within a string are converted to lower case while respecting the current locale.

15 toLocaleUpperCase()
The characters within a string are converted to upper case while respecting the current locale.

16 toLowerCase()
Returns the calling string value converted to lower case.

17 toString()
Returns a string representing the specified object.

18 toUpperCase()
Returns the calling string value converted to uppercase.

19 valueOf()
Returns the primitive value of the specified object.
26. a. Explain in detail about Date object in Java Script.

The Date object is a datatype built into the JavaScript language. Date objects are created with the new Date( ) as shown below.

Once a Date object is created, a number of methods allow you to operate on it. Most methods simply allow you to get and set the year, month, day, hour, minute, second, and millisecond fields of the object, using either local time or UTC (universal, or GMT) time.

The ECMAScript standard requires the Date object to be able to represent any date and time, to millisecond precision, within 100 million days before or after 1/1/1970. This is a range of plus or minus 273,785 years, so JavaScript can represent date and time till the year 275755.

Syntax
You can use any of the following syntaxes to create a Date object using Date() constructor.

new Date( )
new Date(milliseconds)
new Date(datestring)
new Date(year,month,date[,hour,minute,second,millisecond])
Note − Parameters in the brackets are always optional.

Here is a description of the parameters −

No Argument − With no arguments, the Date() constructor creates a Date object set to the current date and time.

milliseconds − When one numeric argument is passed, it is taken as the internal numeric representation of the date in milliseconds, as returned by the getTime() method. For example, passing the argument 5000 creates a date that represents five seconds past midnight on 1/1/70.

datestring − When one string argument is passed, it is a string representation of a date, in the format accepted by the Date.parse() method.

7 agruments − To use the last form of the constructor shown above. Here is a description of each argument −

year − Integer value representing the year. For compatibility (in order to avoid the Y2K problem), you should always specify the year in full; use 1998, rather than 98.

month − Integer value representing the month, beginning with 0 for January to 11 for December.

date − Integer value representing the day of the month.

hour − Integer value representing the hour of the day (24-hour scale).

minute − Integer value representing the minute segment of a time reading.

second − Integer value representing the second segment of a time reading.

millisecond − Integer value representing the millisecond segment of a time reading.

Date Properties
Here is a list of the properties of the Date object along with their description.

Sr.No. Property & Description
1 constructor
Specifies the function that creates an object's prototype.

2 prototype
The prototype property allows you to add properties and methods to an object

In the following sections, we will have a few examples to demonstrate the usage of different Date properties.

Date Methods
Here is a list of the methods used with Date and their description.

Sr.No. Method & Description
1 Date()
Returns today's date and time

2 getDate()
Returns the day of the month for the specified date according to local time.

3 getDay()
Returns the day of the week for the specified date according to local time.

4 getFullYear()
Returns the year of the specified date according to local time.

5 getHours()

Returns the hour in the specified date according to local time.

6 getMilliseconds()
Returns the milliseconds in the specified date according to local time.

7 getMinutes()
Returns the minutes in the specified date according to local time.

8 getMonth()
Returns the month in the specified date according to local time.

9 getSeconds()
Returns the seconds in the specified date according to local time.

10 getTime()
Returns the numeric value of the specified date as the number of milliseconds since January 1, 1970, 00:00:00 UTC.

11 getTimezoneOffset()
Returns the time-zone offset in minutes for the current locale.

12 getUTCDate()
Returns the day (date) of the month in the specified date according to universal time.

13 getUTCDay()
Returns the day of the week in the specified date according to universal time.

14 getUTCFullYear()
Returns the year in the specified date according to universal time.

15 getUTCHours()
Returns the hours in the specified date according to universal time.

16 getUTCMilliseconds()
Returns the milliseconds in the specified date according to universal time.

17 getUTCMinutes()
Returns the minutes in the specified date according to universal time.

18 getUTCMonth()
Returns the month in the specified date according to universal time.

19 getUTCSeconds()
Returns the seconds in the specified date according to universal time.

20 getYear()
Deprecated - Returns the year in the specified date according to local time. Use getFullYear instead.

21 setDate()
Sets the day of the month for a specified date according to local time.

22 setFullYear()
Sets the full year for a specified date according to local time.

23 setHours()

Sets the hours for a specified date according to local time.

24 setMilliseconds()
Sets the milliseconds for a specified date according to local time.

25 setMinutes()
Sets the minutes for a specified date according to local time.

26 setMonth()
Sets the month for a specified date according to local time.

27 setSeconds()
Sets the seconds for a specified date according to local time.

28 setTime()
Sets the Date object to the time represented by a number of milliseconds since January 1, 1970, 00:00:00 UTC.

29 setUTCDate()
Sets the day of the month for a specified date according to universal time.

30 setUTCFullYear()
Sets the full year for a specified date according to universal time.

31 setUTCHours()
Sets the hour for a specified date according to universal time.

32 setUTCMilliseconds()
Sets the milliseconds for a specified date according to universal time.

33 setUTCMinutes()
Sets the minutes for a specified date according to universal time.

34 setUTCMonth()
Sets the month for a specified date according to universal time.

35 setUTCSeconds()
Sets the seconds for a specified date according to universal time.

36 setYear()
Deprecated - Sets the year for a specified date according to local time. Use setFullYear instead.

37 toDateString()
Returns the "date" portion of the Date as a human-readable string.

38 toGMTString()
Deprecated - Converts a date to a string, using the Internet GMT conventions. Use toUTCString instead.

39 toLocaleDateString()
Returns the "date" portion of the Date as a string, using the current locale's conventions.

40 toLocaleFormat()
Converts a date to a string, using a format string.

41 toLocaleString()
Converts a date to a string, using the current locale's conventions.

42 toLocaleTimeString()
Returns the "time" portion of the Date as a string, using the current locale's conventions.

43 toSource()
Returns a string representing the source for an equivalent Date object; you can use this value to create a new object.

44 toString()
Returns a string representing the specified Date object.

45 toTimeString()
Returns the "time" portion of the Date as a human-readable string.

46 toUTCString()
Converts a date to a string, using the universal time convention.

47 valueOf()
Returns the primitive value of a Date object.

(OR)

b. Explain about handling event using java script.

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.

Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

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

onclick Event Type

This is the most frequently used event type which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type.

Example

```
<html>
  <head>
    <script type = "text/javascript">
      <!--
        function sayHello() {
          alert("Hello World")
        }
      //-->
    </script>
  </head>

  <body>
    <p>Click the following button and see result</p>
    <form>
      <input type = "button" onclick = "sayHello()" value = "Say Hello" />
    </form>
  </body>
</html>
```