14BECS305B **Programming with Java Script**

L	T	P	C
2	0	2	3

COURSE OBJECTIVES:

- Understand and use basic programming syntax using JavaScript
- Understand and use JavaScript to enhance HTML documents.
- Understand the Java Scripts Object Model.
- Understand and use predefined JavaScript objects

LEARNING OUTCOMES:

- Describe the origins of JavaScript and list its key characteristics.
- Communicate with users using JavaScript.
- Define and call JavaScript functions.
- Control program flow.
- Identify and use the JavaScript language objects.
- Use JavaScript with HTML form controls.
- Define and utilize cookies.
- Create custom JavaScript objects.

UNIT I Programming Fundamentals
(9)

What is JavaScript? Brief history-Common use-cases-Runtime environments-Overview of language features-Running JavaScript in the browser and at the command line-Debugging JavaScript in the browser- Authoring and debugging code -The roles and relationships between HTML, CSS and Javascript

UNIT II The Javascript Language
(9)

Basic data types, variables, objects, and mathematical operations- Control structures, conditionals, looping, functions- Data and data structures : Objects -Arrays - Dates and other built-in data objects- More data structures :Functions, objects, and data -JSON - Advanced control structures

UNIT III Javascript and the behavior of Web pages
(9)

Making Web pages behave: manipulating the DOM- Working with Browser Events • Script loading, responding to keyboard input or mouse activity, scrolling- Forms and AJAX- Using

Javascript Libraries for Advanced Behavior :JQuery and others • Animations, AJAX, form and data handling

UNIT IV Javascript Libraries and Advanced Applications (9)

Understanding How Libraries Work -Library Architecture and design patterns -Writing a JQuery plugin- Other kinds of libraries-Media players, layout managers -Writing your own library-Javascript and multimedia

UNIT V Security (9)

Same-origin policy. Cross-site scripting attacks (reflected and persisted). Cookie theft and forgery. Whitelisting and blacklisting.

Total Hours: 45

TEXT BOOKS:

1. Modern Javascript: Develop and Design by Larry Ullman, Peachpit Press,2012

REFERENCES:

1. Javascript Bible, 7th Edition, Danny Goodman Michael Morrison Paul Novitski Tia GustaffRayl, Wiley India Pvt Ltd 2014
2. Web Technologies Black Book: HTML, JavaScript, PHP, Java, JSP, XML and AJAX, Kogent Learning Solutions Inc, Dreamtech Press 2014

WEBSITES:

1. http://proquest.safaribooksonline.com.ezpprod1.hul.harvard.edu/book/programming/javascript/9780132905848

KARPAGAM UNIVERSITY

Faculty of Engineering

Department of Computer Science

Lecture Plan

Name of the Faculty :
Subject code **14BECS305B**
Name of the Subject : Programming with Java Script
Class **: BE CSE**

S.No.	Duration	Topic Name	Teaching Aids	Page no. of Text book
Unit - I - Programming Fundamentals				
1	1	Introduction - What is JavaScript?	BB	T1->3
2	1	Brief history-Common use cases	VIDEO LEC & PPT	T1->19
3	1	use cases-Runtime	PPT	T1->3-4
4	1	*Tutorial : Java Script usage*	-	
5	1	environments-Overview of language features	BB	T1->23
6	1	Running JavaScript in the browser and at the command line	EXE	T1->3
7	1	Debugging JavaScript in the browser-Authoring and debugging code	BB	T1->4
8	1	*Tutorial - Running JavaScript in the browser*	-	
9	1	The roles and relationships between HTML, CSS and Javascript	BB	W3
Unit - II - The Javascript Language				
10	1	Basic data types, variables, objects	BB	T1->29-31
11	1	mathematical operations	BB	T1->33

12	1	Control structures, conditionals, looping	PPT	T1->36
13	1	*Tutorial - data types, variables*	-	
14	1	Data and data structures : Objects -Arrays	PPT	T1->31-32
15	1	Dates and other built-in data objects	PPT	T1->174-175
16	1	More data structures :Functions, objects,	EXE	T1->183-184
17	1	*Tutorial - Objects -Arrays*	-	
18	1	data -JSON - Advanced control structures	PPT	W1,W3
Unit - III - Javascript and the behavior of Web pages				
19	1	Making Web pages behave: manipulating the DOM	PPT	T1->249
20	1	Working with Browser Events	EXE	T2->335
21	1	Script loading	EXE	T2->376
22	1	*Tutorial - Working with Browser Events*	-	
23	1	responding to keyboard input or mouse activity	PPT	W1
24	1	scrolling- Forms	PPT	T2->424
25	1	AJAX- Using Javascript Libraries for Advanced Behavior	EXE	W1
26	1	*Tutorial - keyboard input or mouse activity*	-	
27	1	JQuery and others • Animations, AJAX, form and data handling	BB	W1
Unit - IV -Javascript Libraries and Advanced Applications				
28	1	Understanding How Libraries Work	EXE	T2->490 - 492
29	1	Library Architecture and design patterns	PPT	T2->493-498
30	1	Writing a JQuery plugin	EXE	T2->503-506
31	1	*Tutorial - Architecture and design patterns*	-	
32	1	Other kinds of libraries-Media players	EXE	T2->507-512
33	1	layout managers	EXE	T2->514-519
34	1	Writing your own library	BB	T2->520-521.
35	1	*Tutorial -Writing your own library*	-	
36	1	Javascript and multimedia	EXE	T2->-522-

				525
Unit - V - Input /Output Streams				
37	1	Same-origin policy	BB	W1
38	1	Cross-site scripting attacks - reflected	BB	W1
39	1	Cross-site scripting attacks - persisted	BB	W1
40	1	*Tutorial - Cross-site scripting attacks*	-	
41	1	Using the File Class- Input /Output Exceptions-Creation of Files	BB	W1
42	1	Cookie theft	BB	W1
43	1	forgery	BB	W1
44	1	*Tutorial - Inheritance*	-	
45	1	Whitelisting and blacklisting	BB	W1
46	1	Internal question papers discussion	-	

Total Hours allocated : 46

TEXT BOOKS:

2. Modern Javascript: Develop and Design by Larry Ullman, Peachpit Press,2012

REFERENCES:

3. Javascript Bible, 7th Edition, Danny Goodman Michael Morrison Paul Novitski Tia GustaffRayl, Wiley India Pvt Ltd 2014
4. Web Technologies Black Book: HTML, JavaScript, PHP, Java, JSP, XML and AJAX, Kogent Learning Solutions Inc, Dreamtech Press 2014

WEBSITES:

2. http://proquest.safaribooksonline.com.ezpprod1.hul.harvard.edu/book/programming/javascript/9780132905848

LECTURE NOTES

UNIT-I

JavaScript

History

JavaScript was designed to 'plug a gap' in the techniques available for creating web-pages.

HTML is relatively easy to learn, but it is static. It allows the use of links to load new pages, images, sounds, etc., but it provides very little support for any other type of interactivity.

To create dynamic material it was necessary to use either:

- CGI (Common Gateway Interface) programs
    - Can be used to provide a wide range of interactive features, but...
    - Run on the server, i.e.:
        - A user-action causes a request to be sent over the internet from the client machine to the server.
        - The server runs a CGI program that generates a new page, based on the information supplied by the client.

        - The new page is sent back to the client machine and is loaded in place of the previous page.

      Thus every change requires communication back and forth across the internet.

    - Written in languages such as Perl, which are relatively difficult to learn.

- Java applets
    - Run on the client, so there is no need to send information back and forth over the internet for every change, but...
    - Written in Java, which is relatively difficult to learn.

Netscape Corporation set out to develop a language that:

- Provides dynamic facilities similar to those available using CGI programs and Java applets.
- Runs on the Client.
- Is relatively easy to learn and use.

They came up with ***LiveScript***.

Netscape subsequently teamed-up with Sun Microsystems (the company that developed Java) and produced ***JavaScript***.

Javascript only runs on Netscape browsers (e.g., Netscape Navigator). However, Microsoft soon developed a version of JavaScript for their Internet Explorer browser. It is called ***JScript***. The two languages are almost identical, although there are some minor differences.

Internet browsers such as Internet Explorer and Netscape Navigator provide a range of features that can be controlled using a suitable program. For example, windows can be opened and closed, items can be moved around the page, colours can be changed, information can be read or modified, etc..

However, in order to do this you need to know what items the browser contains, what operations can be carried out on each item, and the format of the necessary commands.

Therefore, in order to program internet browsers, you need to know:

- How to program in a suitable language (e.g., Javascript/JScript)
- The internal structure of the browser.

In this course we will be using JavaScript/JScript to program browsers. However, there are several other languages we could use should we wish to. Therefore, we shall try to distinguish clearly between those aspects of internet programming which are specific to JavaScript/JScript and those which remain the same regardles of which language we choose to use.

We'll start by looking at some of the basic features of the JavaScript language.

Variables & Literals

A variable is a *container* which has a name. We use variables to hold information that may change from one moment to the next while a program is running.

For example, a shopping website might use a variable called *total* to hold the total cost of the goods the customer has selected. The amount stored in this variable may change as the customer adds more goods or discards earlier choices, but the name *total* stays the same. Therefore we can find out the current total cost at any time by asking the program to tell us what is currently stored in *total*.

A literal, by contrast, doesn't have a name - it only has a value.

For example, we might use a literal to store the VAT rate, since this doesn't change very often. The literal would have a value of (e.g.) 0.21. We could then obtain the final cost to the customer in the following way:

VAT is equal to *total* x 0.21

final total is equal to *total* + *VAT*

JavaScript accepts the following types of variables:

Numeric Any numeric value, whether a whole number (an *integer*) or a number that includes a fractional part (a *real*), e.g.,
12
3.14159
etc.

String A group of text characters, e.g.,
Ian
Macintosh G4
etc.

Boolean A value which can only be either True or False, e.g.
completed
married
etc.

We create variables and assign values to them in the following way:

var christianName = "Fred"	(string)
var surname = "Jones"	(string)
var age = 37	(numeric)
var married = false	(Boolean)

Note that:

- When a new variable is created (or *declared*) its name must be preceded by the word `var`
- The type of the variable is determined by the way it is declared:
  - if it is enclosed within quotes, it's a string
  - if it is set to true or false (without quotes) it's a boolean
  - if it is a number (without quotes) it's numeric
- We refer to the equals sign as the *assignment operator* because we use it to assign values to variables;
- Variable names must begin with a letter or an underscore
- Variable names must not include spaces
- JavaScript is case-sensitive

- Reserved words (i.e., words which indicate an action or operation in JavaScript) cannot be used as variable names.

Operators

Operators are a type of command. They perform operations on variables and/or literals and produce a result.

JavaScript understands the following operators:

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

(If you're not sure what a modulus operator does, here are some notes and an example)

These are known as *binary* operators because they require *two* values as input, i.e.:

4 + 3

7 / 2

15 % 4

In addition, JavaScript understands the following operators:

+ +	Increment	Increase value by 1
- -	Decrement	Decrease value by 1
-	Negation	Convert positive to negative, or vice versa

These are known as *unary* operators because they require only *one* value as input, i.e.:

4++	increase 4 by 1 so it becomes 5
7--	decrease 7 by 1 so it becomes 6
-5	negate 5 so it becomes -5

JavaScript operators are used in the following way:

```
var totalStudents = 60
var examPasses = 56
var resits = totalStudents - examPasses
```

Note that by carrying out this operation we have created a new variable - `resits`.

There is no need to declare this variable in advance.

It will be a numeric value because it has been created as a result of an operation performed on two numeric values.

We can also combine these operators (and the assignment operator, =) in certain ways.

For example:

```
total += price
```

performs the same task as:

```
total = total + price
```

Similarly,

```
total *= quantity
```

performs the same task as:

```
total = total * quantity
```

Below is a full list of these 'combined' operators.

+ =	'becomes equal to itself plus'
- =	'becomes equal to itself minus'
* =	'becomes equal to itself multiplied by'
/ =	'becomes equal to itself divided by'
% =	'becomes equal to the amount which is left when it is divided by'

You may find the descriptions helpful when trying to remember what each operator does. For example:

5 * = 3

can be thought of as meaning:

5 becomes equal to itself multiplied by 3

UNIT-II

Functions in JavaScript

In JavaScript, as in other languages, we can create *functions*. A function is a kind of mini-program that forms part of a larger program.

Functions:

- consist of one or more *statements* (i.e., lines of program code that perform some operation)
- are separated in some way from the rest of the program, for example, by being enclosed in curly brackets, {.....}
- are given a unique name, so that they can be *called* from elsewhere in the program.

Functions are used:

- Where the same operation has to be performed many times within a program.

  Rather than writing the same code again and again, it can be written once as a function and used repeatedly. For example:

```
request_confirmation_from_user
```

- To make it easier for someone else to understand your program.

  Rather than writing long, rambling programs in which every single operation is listed in turn, it is usually better to divide programs up into small groups of related operations. For example:

```
set_variables_to_initial_values

welcome_user

obtain_user_input

perform_calculations

display_results
```

In JavaScript, functions are created in the following way:

```
function name()
{
        statement;
        statement;
        statement
}
```

Note that all the statements except the last statement must be followed by semi-colons. The last statement doesn't need one, but if you do put a semi-colon after the last statement it won't cause any problems.

Here is an example of a simple function:

```
function initialiseVariables()
{
        itemsSold = 0;
        nettPrice = 0;
        priceAfterTax = 0
}
```

When called, this function will set the three variables `itemsSold`, `nettPrice` and `priceAfterTax` to zero.

To run this function from somewhere else in a program, we would simply call it by name, e.g.:

```
initialiseVariables();
```

Note that the name must be followed by a pair of brackets. The purpose of these will become clear later.

Functions can be called from within other functions.

For example:

```
function sayGoodbye()
{
        alert("Goodbye!")
}

function sayHello()
{
        alert("Hi, there!");
        sayGoodbye()
}
```

When the function `sayHello()` is called, it first displays an *alert* on the screen. An alert is simply box containing some text and an 'OK' button that the user can press to make the box disappear when the text has been read. In this case the box will contain the words "Hi, there!".

The `sayHello()` function then calls the function `sayGoodbye()`, which posts another alert saying "Goodbye".

Click here to see this example working.

Note that the function `sayGoodbye()` is written first. Browsers interpet JavaScript code line-by-line, starting at the top, and some browsers will report an error if they find a

reference to a function before they find the function itself. Therefore functions should be declared before the point in the program where they are used.

Passing Parameters to Functions

Some functions perform a simple task for which no extra information is needed.

However, it is often necessary to supply information to a function so that it can carry out its task.

For example, if we want to create a function which adds VAT to a price, we would have to tell the function what the price is.

To do this we would pass the price into the function as a *parameter*. Parameters are listed in between the brackets that follow the function name. For example:

```
function name(parameter_1, parameter_2)
{
        statement(s);
}
```

In this case two parameters are used, but it's possible to use more than this if necessary. The additional parameter names would simply be added on to the list of parameters inside the brackets, separated from one another by commas. It's also possible to use just one prameter if that's all that is needed.

Here's an example of a simple function that accepts a single parameter:

```
function addVAT(price)
{
        price *= 1.21;
        alert(price)
}
```

This function accepts a parameter called `price`, multiplies it by 1.21 (i.e., adds an extra 21% to it), and then displays the new value in an alert box.

We would call this function in the following way:

```
addVAT(nettPrice)
```

The parameter `nettPrice` could be either:

- a literal - for example:

```
addVAT(5)
```

- a variable - for example:

```
var nettPrice = 5;
addVAT(nettPrice)
```

Returning values from Functions

Sometimes we also need to get some information back from a function.

For example, we might want to add VAT to a price and then, instead of just displaying the result, pass it back to the user or display it in a table.

To get information back from a function we do the following:

```
function addVAT(price)
{
        price *= 1.21;
        return price
}
```

To call this function we would do the following:

```
var newPrice = addVAT(nettPrice)
```

The value returned by the funtion will be stored in the variable `newPrice`. Therefore this funtion will have the effect of making `newPrice` equal to `nettPrice` multiplied by 1.21.

UNIT-III

JavaScript Comparison Operators

As well as needing to assign values to variables, we sometime need to compare variables or literals.

We do this using *Comparison Operators*.

Comparison Operators compare two values and produce an output which is either true or false.

For example, suppose we have two variables which are both numeric (i.e., they both hold numbers):

`examPasses`

and

`totalStudents`

If we compare them, there are two possible outcomes:

- They have the same value
- They do not have the same value

Therefore, we can make statements like these:

- They are the same
- They are the different
- The first is larger than the second
- The first is smaller than the second

... and then perform a comparison to determine whether the statement is true or false.

The basic comparison operator is:

`==`

(i.e., two equals signs, one after the other with no space in between).

It means 'is equal to'. Compare this with the *assignment operator*, `=`, which means 'becomes equal to'. The assignment operator *makes* two things equal to one another, the comparison operator tests to see if they are *already* equal to one another.

Here's an example showing how the comparison operator might be used:

```
examPasses == totalStudents
```

If `examPasses` and `totalStudents` have the same value, the comparison would return `true` as a result.

If `examPasses` and `totalStudents` have different values, the comparison would return `false` as a result.

Another comparison operator is:

`!=`

(i.e., an exclamation mark followed by an equals sign, with no space in between).

It means 'is NOT equal to'.

For example:

```
examPasses != totalStudents
```

If `examPasses` and `totalStudents` have the same value, the comparison would return `false` as a result.

If `examPasses` and `totalStudents` have different values, the comparison would return `true` as a result.

Two other commonly-used comparison operators are:

<

and

>

< means 'less than'

> means 'greater than'

For example:

```
examPasses < totalStudents
```

If `examPasses` is less than `totalStudents`, the comparison would return `true` as a result.

If `examPasses` is more than `totalStudents`, the comparison would return `false` as a result.

Another example:

```
examPasses > totalStudents
```

If `examPasses` is more than `totalStudents`, the comparison would return `true` as a result.

If `examPasses` is less than `totalStudents`, the comparison would return `false` as a result.

As with some of the other 0perators we have encountered, comparison operators can be combined in various ways.

`<=`

means

'less than or equal to'.

For example:

```
examPasses <= totalStudents
```

If `examPasses` is less than or equal to `totalStudents`, the comparison would return `true` as a result.

If `examPasses` is more than `totalStudents`, the comparison would return `false` as a result.

Also:

`>=`

means

'greater than or equal to'.

For example:

```
examPasses >= totalStudents
```

If `examPasses` is more than or equal to `totalStudents`>, the comparison would return `true` as a result.

If `examPasses` is less than `totalStudents`, the comparison would return `false` as a result.

To summarise, JavaScript understands the following comparison operators:

==	'is equal to'
!=	'is NOT equal to'
<	'is less than'
>	'is greater than'
<=	'is less than or equal to'
>=	'is greater than or equal to'

If-Else Statements in JavaScript

Much of the power of programming languages comes from their ability to respond in different ways depending upon the data they are given.

Thus all programming languages include statements which make 'decisions' based upon data.

One form of decision-making statement is the `If...Else` statement.

It allows us to make decisions such as:

> If I have more than £15 left in my account, I'll go to the cinema.
>
> Otherwise I'll stay at home and watch television.

This might be expressed in logical terms as:

> If (money > 15) go_to_cinema
>
> Else watch_television

The If-Else statement in JavaScript has the following syntax:

```
if (condition)
{
```

```
        statement;
        statement
}
else
{
        statement;
        statement
};
```

The condition is the information on which we are basing the decision. In the example above, the condition would be whether we have more than £15. If the condition is true, the browser will carry out the statements within the `if...` section; if the condition is false it will carry out the statements within the `else...` section.

The `if...` part of the statement can be used on its own if required. For example:

```
if (condition)
{
        statement;
        statement
};
```

Note the positioning of the semi-colons.

If you are using both the `if...` and the `else...` parts of the statement, it is important NOT to put a semi-colon at the end of the `if...` part. If you do, the `else...` part of the statement will never be used.

A semi-colon is normally placed at the very end of the `if...else...` statement, although this is not needed if it is the last or only statement in a function.

A practical If-Else statement in JavaScript might look like this:

```
if (score > 5)
{
        alert("Congratulations!")
}
else
```

```
{
        alert("Shame - better luck next time")
};
```

The `for` Loop

A `for` loop allows you to carry out a particular operation a fixed number of times.

The `for` loop is controlled by setting three values:

- an initial value
- a final value
- an increment

The format of a `for` loop looks like this:

```
for (initial_value; final_value; increment)
{
    statement(s);
}
```

A practical `for` loop might look like this:

```
for (x = 0; x <= 100; x++)
{
    statement(s);
}
```

Note that:

* The loop condition is tested using a variable, `x`, which is initially set to the start value (`0`) and then incremented until it reaches the final value (`100`).

* The variable may be either incremented or decremented.

* The central part of the condition, the part specifying the final value, must remain true throughout the required range. In other words, you could not use `x = 100` in the `for` loop above because then the condition would only be true when x was either 0 or 100, and not for all the values in between. Instead you should use `x <= 100` so that the condition remains true for all the values between 0 and 100.

Here are some practical examples of `for` loops.

The first example is a simple loop in which a value is incremented from 0 to 5, and reported to the screen each time it changes using an alert box. The code for this example is:

```
for (x = 0; x <= 5; x++)
{
    alert('x = ' + x);
}
```

Click here to see this example working.

The second example is the same except that the value is decremented from 5 to 0 rather than incremented from 0 to 5. The code for this example is:

```
for (x = 5; x >= 0; x--)
{
    alert('x = ' + x);
}
```

Click here to see this example working.

The start and finish values for the loop must be known before the loop starts.

However, they need not be written into the program; they can, if necessary, be obtained when the program is run.

For example:

```
var initialValue = prompt("Please enter initial value", "");
var finalValue = prompt("Please enter final value", "");

for (x = initialValue; x <= finalValue; x++)
{
	statement(s);
}
```

In this example, the user is prompted to type-in two numbers which are then assigned to the variables `initialValue` and `finalValue`. The loop then increments from `initialValue` to `finalValue` in exactly the same way as if these values had been written directly into the program.

The `While` Loop

Like the `for` loop, the `while` loop allows you to carry out a particular operation a number of times.

The format of a `while` loop is as follows:

```
while (condition)
{
	statement(s);
}
```

A practical `while` loop might look like this:

```
var x = 500000;

alert("Starting countdown...");

while (x > 0)
{
	x--;
```

```
};
alert("Finished!");
```

In this example, `x` is initially set to a high value (`500,000`). It is then reduced by one each time through the loop using the decrement operator (`x--`). So long as `x` is greater than zero the loop will continue to operate, but as soon as `x` reaches zero the loop condition (`x > 0`) will cease to be true and the loop will end.

The effect of this piece of code is to create a delay which will last for as long as it takes the computer to count down from 500,000 to 0. Before the loop begins, an 'alert' dialog-box is displayed with the message "Starting Countdown...". When the user clicks the 'OK' button on the dialog-box the loop will begin, and as soon as it finishes another dialog-box will be displayed saying "Finished!". The period between the first dialog box disappearing and the second one appearing is the time it takes the computer to count down from 500,000 to 0.

To see this example working, click here.

The principal difference between `for` loops and `while` loops is:

with a `while` loop, the number of times the loop is to be executed need not be known in advance.

`while` loops are normally used where an operation must be carried out repeatedly until a particular situation arises.

For example:

```
var passwordNotVerified = true;

while (passwordNotVerified == true)
{
    var input = prompt("Please enter your password", "");
    if (input == password)
    {
        passwordNotVerified = false;
    }
    else
    {
```

```
            alert("Invalid password - try again")
        }
    }
```

In this example, the variable `passwordNotVerified` is initially set to the Boolean value `true`. The user is then prompted to enter a password, and this password is compared with the correct password stored in the variable called `password`. If the password entered by the user matches the stored password, the variable `passwordNotVerified` is set to `false` and the `while` loop ends. If the password entered by the user does not match the stored password, the variable `passwordNotVerified` remains set to `true` and a warning message is displayed, after which the loop repeats.

To try this piece of code, click here.

PS The password is `CS7000` - and don't forget that the 'CS' must be capitalised.

Testing Boolean Variables

In the `while` loop example above we used the line:

```
var passwordNotVerified = true;
```

and then tested this variable in a conditional statement as follows:

```
while (passwordNotVerified == true)
```

We could also have written the conditional statement like this:

```
while(passwordNotVerified)
```

In other words, if we don't specify `true` or `false` in a conditional statement, the JavaScript interpreter will assume we mean `true` and test the variable accordingly.

This allows us to make our code a little shorter and, more importantly, to make it easier for others to understand. The line:

```
while(passwordNotVerified)
```

is much closer to the way in which such a condition might be expressed in English than:

```
while(passwordNotVerified == true)
```

Logical Operators

We have met a number of operators that can be used when testing conditions, e.g., == , < , > , <= , >= .

Two more operators that are particularly useful with `while` loops are:

`&&`	Logical AND
`||`	Logical OR

These operators are used to combine the results of other conditional tests.

For example:

```
if (x > 0 && x < 100)
```

means...

```
if x is greater than 0 and less than 100...
```

Placing the `&&` between the two conditions means that the `if` statement will only be carried out if BOTH conditions are true. If only one of the conditions is true (e.g., x is greater than 0 but also greater than 100) the condition will return false and the `if` statement won't be carried out.

Similarly:

```
if (x == 0 || x == 1)
```

means...

```
if x is 0 or x is 1...
```

Placing the `||` between the two conditions means that the `if` statement will be executed if EITHER of the conditions are true.

The ability to combine conditions in this way can be very useful when setting the conditions for `while` loops.

For example:

```
var amount = prompt ("Please enter a number between 1 and 9", "");

while (amount < 1 || amount > 9)
{
    alert("Number must be between 1 and 9");
    amount = prompt ("Please enter a number between 1 and 9", "");
}
```

In this example, the variable `amount` is initially set to the value typed-in by the user in response to the 'prompt' dialog-box. If the amount entered by the user is between 1 and 9, the loop condition becomes `false` and the loop ends. If the amount entered by the user is less than 1 or greater than 9, the loop condition remains `true` and a warning is displayed, after which the user is prompted to enter another value.

UNIT-IV

avaScript: Window, Document & Form Objects

JavaScript is an *object-oriented* (or, as some would argue, *object-based*) language.

An object is a set of variables, functions, etc., that are in some way related. They are grouped together and given a name

Objects may have:

Properties

A variable (numeric, string or Boolean) associated with an object. Most properties can be changed by the user.

Example: the title of a document

Methods

Functions associated with an object. Can be called by the user.

Example: the `alert()` method

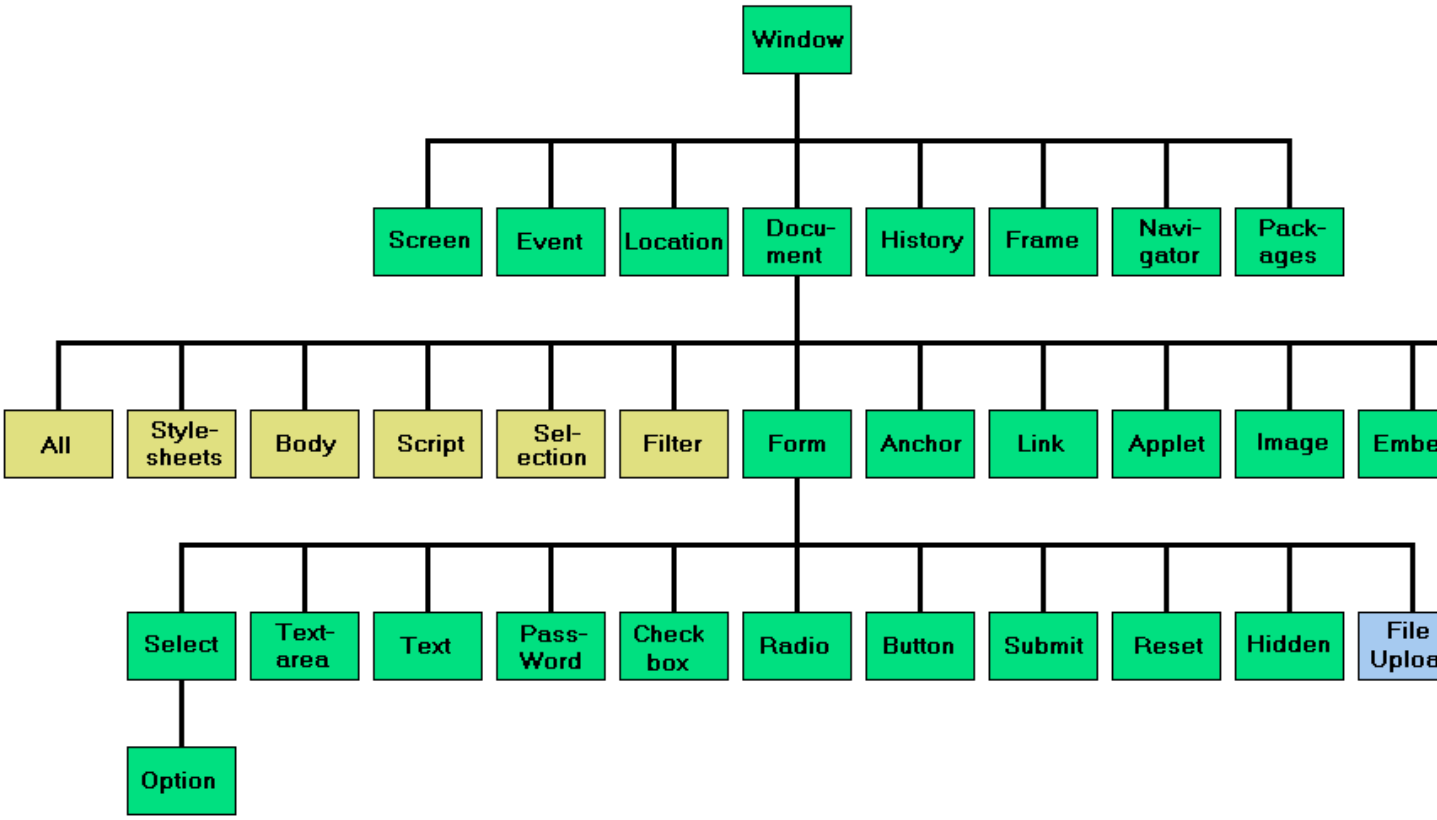
Events

Notification that a particular event has occurred. Can be used by the programmer to trigger responses.

Example: the `onClick()` event.

Internet browsers contain many objects. In the last few years the object structure of internet browsers has become standardised, making programming easier. Prior to this, browsers from different manufacturers had different object structures. Unfortunately, many such browsers are still in use.

The objects are arranged into a hierarchy as shown below:



The Document Object Model. Objects shown in green are common to both Netscape Navigator and Internet Explorer; objects shown in yellow are found only in Internet Explorer while objects shown in blue are found only in Netscape Navigator.

The hierarchy of objects is known as the Document Object Model (DOM).

The Window Object

`Window` is the fundamental object in the browser. It represents the browser window in which the document appears

Its **properties** include:

`status` The contents of the status bar (at the bottom of the browser window). For example:

```
window.status = "Hi, there!";
```

will display the string "Hi, there!" on the status bar.

Click here to see this line of code in operation.

`location` The location and URL of the document currently loaded into the window (as displayed in the location bar). For example:

```
alert(window.location);
```

will display an alert containing the location and URL of this document.

Click here to see this line of code in operation.

`length` The number of frames (if any) into which the current window is divided. For example:

```
alert(window.length);
```

will display an alert indicating the number of frames in the current window.

See under `parent` (below) for an example.

`parent` The parent window, if the current window is a sub-window in a frameset. For example:

```
var parentWindow = window.parent;
alert(parentWindow.length);
```

will place a string representing the parent window into the variable `parentWindow`, then use it to report the number of frames (if any) in the parent window.

Click here to see an example of the use of `parent` and `length`.

`top` The top-level window, of which all other windows are sub-windows. For example:

```
var topWindow = window.top;
alert(topWindow.length);
```

will place a string representing the top-level window into the variable `topWindow`, then use it to report the number of frames (if any)

in the top-level window.

`top` behaves in a very similar way to `parent`. Where there are only two levels of windows, `top` and `parent` will both indicate the same window. However, if there are more than two levels of windows, `parent` will indicate the parent of the current window, which may vary depending upon which window the code is in. However, `top` will always indicate the very top-level window.

Window **methods** include:

`alert()` Displays an 'alert' dialog box, containing text entered by the page designer, and an 'OK' button. For example:

```
alert("Hi, there!");
```

will display a dialog box containing the message "Hi, there!".

Click here to see this line of code in operation.

`confirm()` Displays a 'confirm' dialog box, containing text entered by the user, an 'OK' button, and a 'Cancel' button. Returns `true` or `false`. For example:

```
var response = confirm("Delete File?");
alert(response);
```

will display a dialog box containing the message "Delete File?" along with an 'OK' button and a 'Cancel' button. If the user clicks on 'OK' the variable `response` will contain the Boolean value `true`, and this will appear in the 'alert' dialog-box; If the user clicks on 'Cancel' the variable `response` will contain the Boolean value `false` and this will appear in the 'alert' dialog-box.

Click here to see this code in operation.

`prompt()` Displays a message, a box into which the user can type text, an 'OK' button, and a 'Cancel' button. Returns a text string. The syntax is:

```
prompt(message_string, default_response_string)
```

For example:

```
var fileName = prompt("Select File", "file.txt");
alert(fileName);
```

will display a dialog box containing the message "Select File" along with an 'OK' button, a 'Cancel' button, and an area into which the user can type. This area will contain the string "file.txt", but this can be overwritten with a new name. If the user clicks on 'OK' the variable `fileName` will contain the string "file.txt" or whatever the user entered in its place, and this will be reported using an alert dialog box.

Click here to see this code in operation.

`open()`

Opens a new browser window and loads either an existing page or a new document into it. The syntax is:

```
open(URL_string, name_string, parameter_string)
```

For example:

```
var parameters = "height=100,width=200";
newWindow = open("05_JS4nw.html", "newDocument",
parameters);
```

will open a new window 100 pixels high by 200 pixels wide. An HTML document called '05_JS4nw.html' will be loaded into this window.

Note that the variable `newWindow` is not preceded by the word `var`. This is because `newWindow` is a ***global variable*** which was declared at the start of the script. The reason for this is explained below.

Click here to see this code in operation.

`close()`

Closes a window. If no window is specified, closes the current window. The syntax is:

```
window_name.close()
```

For example:

```
newWindow.close()
```

will close the new window opened by the previous example.

Note that the window name (i.e., `newWindow`) must be declared as a global variable if we want to open the window using one function and close it using another function. If it had been declared as a ***local variable***, it would be lost from the computer's memory as soon as the first function ended, and we would not then be able to use it to close the window.

Click here to see this code in operation.

Window **events** include:

`onLoad()` Message sent each time a document is loaded into a window. Can be used to trigger actions (e.g., calling a function). Usually placed within the `<body>` tag, for example:

```
<body onLoad="displayWelcome()">
```

would cause the function `displayWelcome()` to execute automatically every time the document is loaded or refreshed.

`onUnload()` Message sent each time a document is closed or replaced with another document. Can be used to trigger actions (e.g., calling a function). Usually placed within the `<body>` tag, for example:

```
<body onUnload="displayFarewell()">
```

would cause the function `displayFarewell()` to execute automatically every time the document is closed or refreshed.

The Document Object

The Document object represents the HTML document displayed in a browser window. It has properties, methods and events that allow the programmer to change the way the document is displayed in response to user actions or other events.

Document **properties** include:

`bgColor` The colour of the background. For example:

```
document.bgColor = "lightgreen";
```

would cause the background colour of the document to change to light-green.

Click here to change the background colour of this document.
Click here to change it back again.

`fgColor` The colour of the text. For example:

```
document.fgColor = "blue";
```

will cause the colour of the text in the document to change to blue.

Click here to change the foreground colour of this document.
Click here to change it back again.

(Note that this will not work with all browsers).

`linkColor` The colour used for un-visited links (i.e., those that have not yet been clicked-upon by the user). For example:

```
document.linkColor = "red";
```

will change the colour of all the un-visited links in a document to red.

`alinkColor` The colour used for an active link (i.e., the one that was clicked-upon most recently, or is the process of being clicked). For example:

```
document.alinkColor = "lightred";
```

will change the colour of active links in a document to light-red.

`vlinkColor` The colour used for visited links (i.e., those that have previously been clicked-upon by the user). For example:

```
document.vlinkColor = "darkred";
```

will change the colour of all the visited links in a document to dark-red.

`title` The title of the document, as displayed at the top of the browser window. For example:

```
document.title = "This title has been changed";
```

will replace the existing page title with the text "This title has been changed".

Click here to see this code in operation.

(Note that some browsers do not display a title bar. On such browsers this code will have no effect.)

`forms` An array containing all the forms (if any) in the document. It accepts an index number in the following way:

```
forms[index-number]
```

where `index-number` is the number of a particular form. Forms are automatically numbered from 0, starting at the beginning of the document, so the first form in an HTML document will always have the index-number 0.

An example of the use of the `forms` property is given below, in the section on the `form` object.

Document **methods** include:

`write()` Allows a string of text to be written to the document. Can be used to generate new HTML code in response to user actions. For example:

```
document.write("<h1>Hello</h1> ");
document.write("<p>Welcome to the new page</p>");
document.write("<p>To return to the lecture notes,");
document.write("<a href='05_JS4.html'>click here
</a></p>");
```

will replace the existing page display with the HTML code contained within the brackets of the `document.write()` methods. This code will display the text "Hi, there!" and "Welcome to the new page", followed by a link back to this page.

Note that all the HTML code within the brackets is enclosed within double-quotes. Note too that the link declaration ('05_JS4.html') is enclosed within single quotes. You can use either single or double quotes in both cases, but you must be careful not to mix them up when placing one quoted string inside another.

Click here to see this code in operation.

The Form Object

When you create a form in an HTML document using the `<form>` and `</form>` tags, you automatically create a form object with properties, methods and events that relate to the form itself and to the individual elements within the form (e.g., text boxes, buttons, radio-buttons, etc.). Using JavaScript, you can add behaviour to buttons and other form elements and process the information contained in the form.

Form **properties** include:

`name`

The name of the form, as defined in the HTML `<form>` tag when the form is created, for example:

```
<form name="myForm">
```

This property can be accessed using JavaScript. For example, this paragraph is part of a form that contains the example buttons. It is the third form in the document (the others contain the buttons for the Window and Document object examples). To obtain the name of this form, we could use the following code:

```
alert(document.forms[2].name);
```

This code uses the `document.forms` property described earlier. Since this is the third form in the document it will have the index-number 2 (remember that forms are numbered from 0).

Thus the code above will display the `name` property of the example form, which is simply "formExamples".

Click here to see this code in operation.

`method`

The method used to submit the information in the form, as defined in the HTML `<form>` tag when the form is created, for example:

```
<form method="POST">
```

The `method` property can be set either to `POST` or `GET` (see under 'forms' in any good HTML reference book if you're not sure about the use of the `POST` and `GET` methods).

This property can be accessed using JavaScript. For example, the present form has its `method` attribute set to "POST" (even though it's not actually going to be submitted). So the code:

```
alert(document.forms[2].method);
```

will display the `method` property of the example form, which is "POST".

Click here to see this code in operation.

`action`

The action to be taken when the form is submitted, as defined in the HTML `<form>` tag when the form is created, for example:

```
<form action="mailto:sales@bigco.com">
```

The `action` property specifies either the URL to which the form data should be sent (e.g., for processing by a CGI script) or `mailto:` followed by an email address to which the data should be sent (for manual processing by the recipient). See under 'forms' in any good HTML reference book for more information on the use of the `action` attribute.

This property can be accessed using JavaScript. For example, the present form has its `action` attribute set to "mailto:sales@bigco.com" (even though it's not actually going to be submitted). So the code:

```
alert(document.forms[2].action);
```

will display the `action` property of the example form, which is "mailto:sales@bigco.com".

Click here to see this code in operation.

`length` The number of elements (text-boxes, buttons, etc.) in the form. For example:

```
alert(document.forms[2].length);
```

will display the number of elements in this form (there are 22).
Click here to see this code in operation.

`elements` An array of all the elements in the form. Individual elements are referenced by index-number.
Elements are automatically numbered from 0, starting at the beginning of the form, so the first element in a form will always have the index-number 0. For example:

```
alert(document.forms[2].elements[0].name);
```

will display the name of the first element in this form, which is the button labelled "Get Form Name". It's name is "get_form_name".
Click here to see this code in operation.

Form **methods** include:

`submit()` Submits the form data to the destination specified in the `action` attribute using the method specified in the `method` attribute. As such it performs exactly the same function as a standard 'submit' button, but it allows the programmer greater flexibility. For example, using this method it is possible to create a special-purpose 'submit' button that has more functionality than a standard 'submit' button, perhaps checking the data or performing some other processing before submitting the form.

Form **events** include:

`onSubmit` Message sent each time a form is submitted. Can be used to trigger actions (e.g., calling a function). Usually placed within the `<form>` tags, for example:

```
<form onSubmit="displayFarewell()">
```

would cause the function `displayFarewell()` to execute automatically every time the form is submitted.

Text-boxes and text-areas

Each element within a form is an object in its own right, and each has properties, methods and events that can be accessed using JavaScript.

Text-boxes and text-areas have almost identical sets of properties, methods and events, so they will be considered together.

Text-box and text-area **properties** include:

`name` — The name of the text-box or text-area, as defined in the HTML `<input>` tag when the form is created, for example:

```
<input type=text name="textBox1">
```

The `name` property of a text-box or other form element can be accessed using JavaScript in the manner shown under the section on `document.length`, above.

`value` — Whatever is typed-into a text-box by the user. For example, here is a simple text-box:

This text-box is named `textBox1`. Therefore, we can obtain any text typed into it using the following line of code:

```
alert(document.forms[2].textBox1.value);
```

Type something into the text-box, then click here to see this code in operation.

Text-box and text-area **events** include:

`onFocus` Event signal generated when a user clicks in a text-box or text-area. For example, here is a simple text-box:

This text-box was declared using the following HTML code:

```
<input type=text name="textBox2"
onFocus="alertOnFocus()">
```

The function called `alertOnFocus()` displays an alert box, so clicking in the text-box above should trigger the function and cause the alert to appear.

`onBlur` Event signal generated when a user clicks outside a text-box or text-area having previously clicked inside it. For example, here is a simple text-box:

This text-box was declared using the following HTML code:

```
<input type=text name="textBox3"
onBlur="alertOnBlur()">
```

The function called `alertOnBlur()` displays an alert box, so clicking in the text-box above and then clicking outside it should trigger the function and cause the alert to appear.

Buttons, Radio-buttons and Checkboxes

Buttons, Radio-buttons and Checkboxes have almost identical sets of properties, methods and events, so they will be considered together.

Button, Radio-button and Checkbox **properties** include:

`name`

The name of the button, radio-button or checkbox, as defined in the HTML `<input>` tag when the form is created, for example:

```
<input type=button name="button1">
```

The `name` property of a button, radio-button or checkbox can be accessed using JavaScript in the manner shown under the section on `document.length`, above.

`value`

The value given to the button when it is created. On standard buttons the value is displayed as a label. On radio-buttons and check-boxes the value is not displayed. For example, here is a button:

This button is named `button1` and has the value `Original value, original label`. We can change the value of the button, and hence it's label, using the following code:

```
document.forms[2].button1.value = "New value, new
label";
```

Click here to see this code in operation.

`checked`

This property - which is used with radio-buttons and check-boxes but not standard buttons - indicates whether or not the button has been selected by the user. For example, here is a checkbox:

This checkbox is named `checkbox 1`. We can determine whether it has been selected or not using the following code:

```
if (document.forms[2].checkbox1.checked == true)
{
    alert("Checked");
}
else
{
    alert("Not checked");
};
```

Try clicking on the check-box to select and un-select it, then click here to see this code in operation.

Button, Radio-button and Checkbox **methods** include:

`focus()` Give the button focus (i.e., make it the default button that will be activated if the return key is pressed). For example, here is a button that displays an alert when clicked:

This button is named `button2` and has the value `Hello`. We can give it focus using the following code:

```
document.forms[2].button2.focus();
```

Click here to see this code in operation. A dotted border should appear around the label on the button, indicating that the button now has focus. Pressing the RETURN key should now activate the button, causing it to display the alert just as if it had been clicked.

`blur()` Removes focus from a button. For example, the code:

```
document.forms[2].button2.blur();
```

will remove the focus (indicated by the dotted line) from the button above. Click here to see this code in operation.

`click()` Simulates the effect of clicking the button. For example, below is a button that has the following code:

```
document.forms[2].button2.click();
```

Clicking on this button will have the same effect as clicking directly on the button labelled 'Hello', i.e., it will display the 'Hello to you too' dialog box.

Button, Radio-button and Checkbox **events** include:

`onClick` Signal sent when the button is clicked. Can be used to call a function. Probably the most frequently-used of all the button events (all the example buttons in this document use this method).

For example:

This button was declared using the following code:

```
<input type=button name="button3" value="Click Here"
onClick="alert('onClick event received')">
```

The code `onClick="alert('onClick event received')"` will cause an alert dialog box to appear whenever the button is pressed.

`onFocus` Signal sent when the button receives focus (i.e., when it becomes the default button, the one that is activated by pressing the RETURN key). For example:

This button was declared using the following code:

```
<input type=button name="button4" value="Click Here"
onFocus="alert('This button is now the default')">
```

The first time you click the button it will gain focus, and the alert will appear. However, if you click again, no alert will appear because the button still has focus as a result of the previous click. To make the alert appear again, you will have to remove focus from the button (e.g., by clicking somewhere else on the document) then restore it by clicking the button again.

`onBlur` Signal sent when the button loses focus. For example:

This button was declared using the following code:

```
<input type=button name="button5" value="Click Here"
onBlur="alert('This button is no longer the
default')">
```

Clicking the button will not cause the alert to appear because it will only give the button focus. However, if you remove focus from the button (e.g., by clicking somewhere else on the document) the alert will appear.

The Select Object

Selection-boxes behave in a very similar fashion to radio-buttons: they present several options, of which only one can be selected at a time. They also have a similar set of properties, methods and events.

The principal difference from a programming perspective is that selection-boxes don't have a `checked` property. Instead, to find out which option has been selected, you must use the `SelectedIndex` property.

UNIT-V

Other Objects

In addition to the objects we have already encountered, there are a number of other objects that form part of the JavaScript language. Among the more important and useful of these are the `Date` and `Math` objects.

The `Date` object

The `Date` object allows us to obtain the current date and time, and to perform various timing operations.

In order to use the `Date` object, we must first create a new 'instance' of it. This is done in the following way:

```
var myDateObject = new Date;
```

This creates an object called `myDateObject` that contains information about the date, time, etc., ***at the instant it was created***. The information in `myDateObject` doesn't

change as time passes, so if you want to know the correct time a few minutes later you will have to create a new instance of the `Date` object.

Once you have created an instance of the `Date` object, you can use any of the methods below to obtain information from it:

`getFullYear()` Returns the current year as a four-digit number (e.g., 2000). For example:

```
var myDateObject = new Date;

var currentYear = myDateObject.getFullYear();

alert(currentYear);
```

Click here to see this example working.

`getMonth()` Returns the current month as an integer from 0-11 (e.g., November = 10). For example:

```
var myDateObject = new Date;

var currentMonth = myDateObject.getMonth();

alert(currentMonth);
```

Click here to see this example working.

`getDate()` Returns the day of the month as an integer between 1 and 31. For example:

```
var myDateObject = new Date;
```

```
var currentDate = myDateObject.getDate();

alert(currentDate);
```

Click here to see this example working.

`getDay()` Returns the day of the week as an integer between 0 and 6, starting from Sunday (e.g., Tuesday = 2). For example:

```
var myDateObject = new Date;

var currentDay = myDateObject.getDay();

alert(currentDay);
```

Click here to see this example working.

`getHours()` Returns the hour of the day as an integer between 0 and 23. For example:

```
var myDateObject = new Date;

var currentHour = myDateObject.getHours();

alert(currentHour);
```

Click here to see this example working.

`getMinutes()` Returns the number of minutes since the beginning of the hour as an integer.

For example:

```
var myDateObject = new Date;

var currentMinute = myDateObject.getMinutes();

alert(currentMinute);
```

Click here to see this example working.

`getSeconds()` Returns the number of seconds since the start of the minute as an integer.
For example:

```
var myDateObject = new Date;

var currentSecond = myDateObject.getSeconds();

alert(currentSecond);
```

Click here to see this example working.

In order to use the `Date` object, it is often necessary to convert the data it produces, e.g., to obtain the names of days and months rather than just numbers. To see an example of the `Date` object in use, click here.

The `Date` object also has methods to obtain time intervals as small as milliseconds, to convert between various time systems, to parse dates and times in various formats into individual elements, and to perform various other time-related operations.

The `Math` object

The `Math` object allows us to perform various mathematical operations not provided by the basic operators we have already looked at.

Its methods include the following:

`sqrt(x)` Returns the square root of *x*. For example:

```
var inputValue = prompt("Please enter a value", "");
var squareRoot = Math.sqrt(inputValue);
alert(squareRoot);
```

Click here to see this example working.

`log(x)` Returns the natural logarithm of *x*. For example:

```
var inputValue = prompt("Please enter a value", "");
var logOfX = Math.log(inputValue);
alert(logOfX);
```

Click here to see this example working.

`max(x,y)` Returns whichever is the larger of *x* and *y*. For example:

```
var inputX = prompt("Please enter a value for X", "");
var inputY = prompt("Please enter a value for Y", "");
var largerOfXY = Math.max(inputX, inputY);
alert(largerOfXY);
```

Click here to see this example working.

`min(x,y)` Returns whichever is the smaller of *x* and *y*.

Works in a similar way to `max(x,y)`, above.

`round(x)` Returns the value of *x* rounded to the nearest integer. For example:

```
var inputvalue = prompt("Please enter a value", "");
var roundedValue = Math.round(inputValue);
alert(roundedValue);
```

Click here to see this example working.

`ceil(x)` Returns the absolute value of *x* rounded **up** to the next integer value.

Works in a similar way to `round(x)`, above.

`floor(x)` Returns the absolute value of *x* rounded **down** to the next integer value.

Works in a similar way to `round(x)`, above.

`abs(x)` Returns the absolute value of *x*. For example:

```
var rawValue = prompt("Please enter a value", "")

var absValue = Math.abs(rawValue)

alert(absValue);
```

Click here to see this example working.

`pow(x,y)` Returns the value of *x* raised to the power *y*. For example:

```
var baseValue = prompt ("Please enter base value",
"");

var expValue = prompt ("Please enter exponent", "");

var baseToPower = Math.pow(baseValue, expValue);

alert(baseToPower);
```

ONLINE QUESTIONS

UNIT-I

Questions	opt1	opt2	opt3	opt4	opt	opt	answer

					5	6	
<script type="text/javascript"> x=4+"4"; document.write(x); </script> Output------?	44	8	4	Error output			44
<script type="text/javascript" language="javascript"> var qpt = "Qualiyt Point Technologies "; var result = qpt.split(" "); document.write(result); </script>	Quality	Q,u,a,l,i,t,y,P,o,i,n,t,T,e,c,h,n,o,l,o,g,i,e,s	Qualiyt,Point,Technologies	QualityPointTechnologies			Qualiyt,Point,Technologies
Is it possible to nest functions in JavaScript?	True	FALSE					True

<script> document.write(navigator.appCodeName); </script>	get code name of the browser of a visitor	set code name of the browser of a visitor	None of the above				get code name of the browser of a visitor
Which of the following is true?	If onKeyDown returns false, the key-press event is cancelled	If onKeyPress returns false, the key-down event is cancelled.	If onKeyDown returns false, the key-up event is cancelled.	If onKeyPress returns false, the key-up event is canceled.			If onKeyDown returns false, the key-press event is cancelled
Scripting language are	High Level Programming language	Assembly Level programming language	Machine level programming language				High Level Programming language
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

<script language="javascript"> function x() { var s= "Good 100%"; var pattern = /\D/g; var output= s.match(pattern); document.write(output); } </script>	Good %	1,0,0	G,o,o,d, %	Error			G,o,o,d,%
<script language="javascript"> var qpt="QUALITY POINT TECHNOLOGIES"; alert(qpt.charAt(qpt.length-1)); </script>	P	E	S	Error			S

Choose the client-side JavaScript object:	Database	Cursor	Client	FileUpLoad			FileUpLoad
Are java and javascript the same?	NO	YES					NO
Syntax for creating a RegExp object: (a). var txt=new RegExp(pattern,attributes); (b). var txt=/pattern/attributes; Which of the above mentioned syntax will correct?	(a) only	(b) only	Both (a) and (b)	None			Both (a) and (b)

<script language="javascript"> function x(z,t) { alert(x.length); } </script> output: ?	Error	1	2	3			2
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
In JavaScript, Window.prompt() method return true or false value ?	FALSE	True	None of above				FALSE
Math. round(-20.51)=?	20	-21	19	None			-21

) <script language="javascript"> function x() { var s = "Quality 100%!{[!!"; var pattern = /\w/g; var output = s.match(pattern); document.write(output); } </script>	%,!,{,[,!,!	Q,u,a,l,i,t,y,1,0,0	Quality 100	Error			Q,u,a,l,i,t,y,1,0,0

<script type="text/javascript" language="javascript"> var qpt= new Array(); qpt[0] = "WebDevelopment"; qpt[1]="ApplicationDevelopment" qpt[2]="Testing" qpt[3] = "QualityPointTechnologies"; document.write(qpt[0,1,2,3]); </script>	Error	QualityPointTechnologies	WebDevelopment	WebDevelopmnet,ApplicationDevelopment,Testing,QualityPointTechnologies			QualityPointTechnologies
Choose the server-side JavaScript object:	FileUpLoad	Function	File	Date			File
parseFloat(9+10)=?	19	910	None				None

<script language="javascript"> function x() { document.write(2+5+"8"); } </script>	258	Error	7	78			78
_____ keyword is used to declare variables in javascript.	Var	Dim	String				Var
In Javascript, Which of the following method is used to evaluate the regular expression?	eval(2*(3+5))	evaluate(2*(3+5))	evalu(2*(3+5))	None of the above			eval(2*(3+5))

<script language="javascript"> function x() { var s= "quality 100%"; var pattern = /\d/g; var output= s.match(pattern); document.write(output); } </script>	100	1,0,0	q,u,a,l,i,t,y,%	Error			1,0,0
<script type="text/javascript" language="javascript"> qpt=((45%2)==0)? "hello" : "bye"; document.write(qpt); </script>	hello	bye	Error in string handling	None of the above			bye

)<script language="javascript"> function x() { var qpt = "QualityPoint Technologies "; var pattern = new RegExp("POIiNT","i"); document.write(qpt.match(pattern)); } </script>	Error	POIiNT	Point	null			null
How do you create a new object in JavaScript?	var obj = {};	var obj = Object();	var obj=new {};	None of the above			var obj = {};
In Javascript, What does isNaN function do ?	Return true if the argument is not a number.	Return false if the argument is not a number	Return true if the argument is a number	None of the above			Return true if the argument is not a number.

If x=103 & y=9 then x%=y , what is the value of x after executing x%=y?	4	3	2	5			4
Choose the external object:	Date	Option	Layer	Checkbox			Checkbox
Choose the four symbol pairs that represent RegExp properties lastMatch, lastParent, leftContext, and rightContext, respectively:	$&, $+, $`, $'	$+, $&, $', $`	$&, $~, $`, $'	$+, $&, $`, $'			$&, $+, $`, $'

Which of the following properties hold the values of the pixels of the length of the width and height of the viewer's screen resolution?	screen.width and screen.height	Resolution.width and Resolution.height	screen.pixels.width and screen.pixels.height	ViewerScreen.width and ViewerScreen.height			screen.width and screen.height
ParseInt("15",10)=?	15	10	151	150			15
Which JavaScript feature uses JAR files?	Object signing	Style sheets	Netcaster channels	Image rollovers			Object signing
How to assign a function to a variable with the JavaScript Function contructor ?	var f=Function("x","y", "return x+y");	var f=Function(x,y){ return x+y;}	var f= new Function("x", "y", "return x + y");				var f= new Function(" x", "y", "return x + y");

In JavaScript, Window.alert() is used to allow user to enter something	True	False	None				False

<script language="javascript">
function x()
{
var qpt =
"We are fast growing Software Company located in Chennai, India.";
var pattern = new RegExp("in", "gi");
 	
document.write(pattern.exec(qpt) + " ");
document.write(pattern.exec(qpt) + " ");
document.write(pattern.exec(qpt) + " ");
}
 	
</script>
	in in In	in in in	in in null	in null null			in in In

Is Javascript has any date data type?	Yes	No					No
Math. round(-20.5)=?	-21	20	-20	21			-20
?_name is it valid javascript identifier?	Yes	No					No
<script language="javascript"> function x() { var qpt = "First come, first served"; var pattern = /first/gi; document.write(qpt.match(pattern)[1]); } </script>	first	undefined	First	Error			first

(a). // , /* ..... **/ (b). / , /** ...../ , /* (c). /*.....*/ , // (d). *.....*\ , // In javascript, Which of the above Comments lines are used ?	Only (d)	Only ©	Either (c) or (d)	Only (b)			Only ©

<script language="javascript"> function x() { var s = "Give 100%!{[!!"; var pattern = /\W/g; var output = s.match(pattern); document.write(output); } </script>	,%,!,{,[,!,!	G,i,v,e,1,0,0	Give 100	Error			,%,!,{,[,!,!
Which best describes void?	A method	A function	A statement	An operator			An operator

<script type="text/javascript" language="javascript"> var qpt="Quality PointTechnologies"; var result =qpt.lastIndexOf("l"); document.write(result); </script>	3	18	17	19			18
<script language="javascript"> function x() { var qpt = "First come, first served"; var pattern = /first/g; document.write(qpt.match(pattern)[1]); } </script>	first	First	undefined	None			undefined

<script language="javascript"> function sum(x) { function add(y) { return x+y; } return add; } function callme() { result=sum(5)(5); alert(result); } </script> If you call the function callme(), what will happen ?	10	Error in calling Function	5	None			10
Who invented the javascript programming language?	Tennis Ritchie	James Gosling	Brendan Eich				Brendan Eich

<script type="text/javascript"> document.write(" This is a heading "); document.write(" This is a paragraph. "); document.write(" This is another paragraph. "); </script> Can you write HTML tag inside the javascript ?	No	Yes	Impossible				Yes
Which feature is supported in MSIE 3.x?	split()	document.clear()	join()	charAt()			charAt()
How to speicfy the color of the hypertext links with JavaScript ?	document.linkColor="#00FF00";	document.LColor="#00FF00";	document.LinkC="#00FF00";	document.hyperTextLink="#00FF00":			document.linkColor="#00FF00";

<script language="javascript"> function x() { var qpt = "QualityPoint Technologies "; var pattern = /point/; var output= qpt.search(pattern); document.write("Position: " + output); } </script>	Position-7	Position-1	null	error			Position-1
_____ ___method returns the number of milliseconds in a date string.	B54	setMinutes()	parse()				parse()

_____converts a string to floating point numbers.	eval	ParseInt	ParseFloat	None			ParseFloat
_____ attempts to evaluate a string representing any javascript literals or variables, converting it to a number.	eval	parseFloat	parseInt	None			eval
Which is not an attribute of the cookie property?	path	host	secure	domain			host
How do substring() and substr() differ?	One is not a method of the String object.	substr() takes three arguments, substring() only two.	Only one accepts a desired string length as an argument.	Besides the spelling, nothing.			Only one accepts a desired string length as an argument.
Which is not a reserved word?	interface	short	program	throws			program

In Javascript, Which of the following method is used to find out the character at a position in a string?	charAt()	CharacterAt()	CharPos()	characAt()			charAt()

<script type="text/javascript" language="javascript">
 	
var qpt = "QualityPoint Technologies";
var result =qpt.substring(7,8);
document.write(result);
 	
</script>
	Po	yP	oi	P			P

UNIT-II

Questions	opt1	opt2	opt3	opt4	opt5	opt6	answer
How do you delete an element from an options array?	Set it to false	Set it to null.	Set it to undefined	Set it to -1			Set it to null.
Is javaScript case sensitive ?	Yes	No					Yes
JavaScript RegExp Object has modifier 'i' to _____	Perform case-sensitive matching	Perform case-insensitive matching	Perform both case-sensitive &case-insensitive matching				Perform case-insensitive matching
What are the following looping structures are available in javascripts?	for,forecach	foreach,whileloop	do-while loop,foreach	for , while loop			for , while loop

Which of these is not a method of the Math object?	atan()	atan2()	eval()	acos()			eval()
<script type="text/javascript"> var s = "9123456 or 80000?"; var pattern = /\d{4}/; var output = s.match(pattern); document.write(output); </script>	9123	91234	80000	None			9123
In javascript, RegExp Object Method test() is used to search a string and returns _____	true or false	found value	index	None			true or false

What property would you use to redirect a visitor to another page?	document.URL	window.location.href	.document.location.href	link.href			window.location.href
a.) var qpt="Quality Point Technologies"; b.) var qpt=new String("Quality Point Technologies"); Question: In javascript, which of the above statement can used for string declaration ?	Either (a) or (b)	Only (a)	Neither (a) nor (b)	Only (b)			Either (a) or (b)

<script type="text/javascript" language="javascript"> var qpt = "QualityPointTechnologies"; var result =qpt.indexOf("Tech"); document.write(result); </script>	11	12	15	13			12

<script language="javascript"> function x() { var s = "Eat to live, but do not live to eat"; var pattern = new RegExp("eat$"); document.write(pattern.exec(s)); } </script>	Eat	eat	undefined	Eat eat			eat

<script language="javascript"> function x() { var qpt = "We are fast growing Software Company located in Chennai, India."; var pattern = new RegExp("in","g"); document.write(pattern.exec(qpt) + " "); document.write(pattern.exec(qpt) + " "); document.write(pattern.exec(qpt) + " "); } </script>	in in In	in in in	in in null	in null null			in in null
_____method is used to remove focus from the specified object.	blur()	focus()	None				blur()
parseFloat(“FF2”)=?	152	FF2	NaN	None			NaN

eval((20*4)=?	Nan	204	24	80			80
<script language="javascript"> function x() { var qpt = "QualityPointTechnologies"; var pattern = new RegExp("TECHNOLOGIES","i"); document.write(qpt.match(pattern)); } </script>	null	Technologies	TECHNOLOGIES	Error			Technologies
Javascript is a _____ typed language.	tightly	loosely					loosely

The development environment offers which standard construct for data validation	Super controlled loop constructs	Case sensitivity check	Validation constructs	All of the mentioned			All of the mentioned
The main purpose of a “Live Wire” in NetScape is to	Create linkage between client side and server side	Permit server side, JavaScript code, to connect to RDBMS	Support only non relational database	To interpret JavaScript code			Permit server side, JavaScript code, to connect to RDBMS
The script tag must be placed in	head	head and body	title and head	All of the mentione			head and body
A JavaScript program developed on a Unix Machine	will throw errors and exceptions	must be restricted to a Unix Machine only	will work perfectly well on a Windows Machine	will be displayed as a JavaScript text on the browser			will work perfectly well on a Windows Machine
JavaScript is ideal to	make computations in HTML simpler	minimize storage requirements on the web server	increase the download time for the client	None of the mentioned			minimize storage requirements on the web server

Which attribute is used to specify that the script is executed when the page has finished parsing (only for external scripts)	parse	async	defer	type			async
JavaScript Code can be called by using	RMI	Triggering Event	Preprocessor	Function/ Method			Function/Method
. JavaScript can be written	directly into JS file and included into HTML	directly on the server page	directly into HTML pages	All of the mentioned			directly into JS file and included into HTML
Which of the following Attribute is used to include External JS code inside your HTML Document	. Src	ext	script	link			. Src

A proper scripting language is a	High level programming language	Assembly level programming language	Machine n level program ming language	Low level program ming language			High level programming language
The type of a variable that is volatile is	Volatile variable	Mutable variable	Immutab le variable	Dynamic variable			Mutable variable
A hexadecimal literal begins with	0	0x	0X	Both 0x and 0X			Both 0x and 0X
The generalised syntax for a real number representation is	[digits][.digits][(E|e)[(+|-)]digits]	[digits][+digits][(E|e)[(+|-)]digits]	[digits][(E|e)[(+|-)]digits]	[.digits][digits][(E|e)[(+|-)]digits]			[digits][.digits][(E|e)[(+|-)]digits]
When there is an indefinite or an infinity value during an arithmetic value computation, javascript	Prints an exception error	Prints an overflow error	Displays “Infinity”	Prints the value as such			Displays “Infinity”

Which of the following is not considered as an error in JavaScript?	Syntax error	Missing of semicolons	Division by zero	All of the these			Division by zero
The escape sequence '\f' stands for	Floating numbers	Representation of functions that returns a value	\f is not present in JavaScript	Form feed			Form feed
The snippet that has to be used to check if "a" is not equal to "null" is a. b. c. d.	if(a!=null)	if (!a)	if(a!null)	if(a!==null)			if(a!==null)
The statement a===b refers to	Both a and b are equal in value, type and reference address	Both a and b are equal in value	Both a and b are equal in value and type	There is no such statement			Both a and b are equal in value and type

Assume that we have to convert “false” that is a non-string to string. The command that we use is (without invoking the “new” operator)	false.toString()	String(false)	String newvariable=”false”	Bothfalse.toString()and String(false)			Bothfalse.toString()and String(false)
The functions provide() and require() of Dojo toolkit and Google’s Closure library are used for	declaring and loading modules	loading and declaring modules	Both a and b	None of the mentioned			declaring and loading modules

The maximum number of global symbols a module can define is	2	3	1	4			1
To define each of the set classes as a property of the sets object (namespace) for the module, the statement is	sets = sets.AbstractEnumerableSet.extend();	sets.SingletonSet = sets.AbstractEnumerableSet.extend(...);	sets.SingletonSet = sets.extend(...);	sets = sets.extend(...);			sets.SingletonSet = sets.AbstractEnumerableSet.extend(...);

var Set = sets.Set;
var s = new Set(1,2,3);
What could be the efficiency quotient of the above two statements ?
	The programmer imports at once the frequently used values into the global namespace.	There is no efficiency quotient, the programmer tries to make it inefficient.	The programmer needs to import the Sets everytime he wants to use it.	All of the mentioned			The programmer imports at once the frequently used values into the global namespace.

The scope of a function is also called as	The function's scope	Module function	Modulated function	Private function			Module function
Modules that have more than one item in their API can	Assign itself to a global variable	Invoke another module of the same kind	Return a namespace object	Invoke another module of the same kind			Return a namespace object

The provides() function and the exportsobject are used to	Register the module's API and Store their API	Store the module's API and register their API	Both Register the module's API and Store their API and Store the module's API and register their API	None of the mentioned			Register the module's API and Store their API
Consider the following code snippet var sets = com.davidflanagan.collections.sets; What is the programmer trying to do in the above code snippet?	Importing a single module	Importing a module partially	Importing a namespace	Importing the entire module			Importing the entire module
The properties() method is a	Enumerable method	Non-enumerable method	Operational method	None of the mentioned			Non-enumerable method

What can be done in order to avoid creation of global variables in JavaScript?	To use a method that defines all the variables	To use an object that has the reference to all the variables	To use an object as its namespace	To use global functions			To use an object as its namespace
JavaScript is a _____ ___ language	Object-Oriented	High-level	Assembly-language	Object-Based			Object-Based

The output for the following code snippet would most appropriately be
var a=5 , b=1
var obj = { a : 10 }
with(obj)
{
alert(b)
}
	10	Error	1	5			1

A conditional expression is also called a	Alternate to if-else	Immediate if	If-then-else statement	None of the these			Immediate if

Which is a more efficient code snippet ?
Code 1 :
for(var num=10;num>=1;num--)
{
 	
document.writeln(num);
}
Code 2 :
var num=10;
while(num>=1)
{
 	
document.writeln(num);
num++;
}
	Code 1	Code 2	Both Code 1 and Code2		Cannot Compare		Code 1

A statement block is	conditional block	block that contains a single statement	Both conditional block and block that contains a single statement	block that combines multiple statements into a single compound statement			block that combines multiple statements into a single compound statement

When an empty statement is encountered, a JavaScript interpreter	Ignores the statement	Prompts to complete the statement	Throws an error	Throws an exception			Ignores the statement
The “var” and “function” are	Keywords	Declaration statements	Datatypes	Prototypes			Declaration statements

Consider the following statements
switch(expression)
{
statements
}
In the above switch syntax, the expression is compared with the case labels using which of the following operator(s) ?
	double equals	equals	equal	triple Equals			triple Equals

Consider the following statements
var count = 0;
while (count < 10)
{
 	
console.log(count);
count++;
}
In the above code snippet, what happens?
 	
a. The values of count is logged or stored in
	particular location or storage.	The value of count from 0 to 9 is displayed in the console.	An error is displayed	An exception is thrown			The value of count from 0 to 9 is displayed in the console.

The enumeration order becomes implementation dependent and non-interoperable if :	If the object inherits enumerable properties	The object does not have the properties present in the integer array indices	The delete keyword is never used	Object.defineProperty() is not used			If the object inherits enumerable properties

UNIT-III

Questions	opt1	opt2	opt3	opt4	opt5	opt6	answer

Consider the following code snippet
function printArray(a)
{
var len = a.length, i = 0;
if (len == 0)
 	
console.log("Empty Array");
else
{
do
{
 	
console.log(a[i]);
} while (++i < len);
}
}
What does the above code result?
	Prints the numbers in the array in order	. Prints the numbers in the array in the reverse order	Prints 0 to the length of the array	Prints "Empty Array"			Prints the numbers in the array in order

What are the three important manipulations done in a for loop on a loop variable?	Updation, Incrementation, Initialization	Initialization, Testing, Updation	Testing, Updation, Testing	Initialization, Testing, Incrementation			Initialization, Testing, Updation

Consider the following code snippet **function tail(o)** **{** **for (; o.next; o = o.next) ;** **return o;** **}** **Will the above code snippet work? If not, what will be the error?**	No, this will throw an exception as only numerics can be used in a for loop	No, this will not iterate c.	Yes, this will work	No, this will result in a runtime error with the message "Cannot use Linked List"			Yes, this will work

Consider the following code snippet
for(var p in o)
console.log(o[p]);
The above code is equivalent to which code?
a. for (var i = 0;i <
	length;i++)
	
	console.log(a[i]);
		for (int i = 0;i < a.length;i++)
		
		console.log(a[i]);
			for (var i = 0;i <= a.length;i++)
			
			console.log(a[i]);
				for (var i = 1;i < a.length;i++)
				
				console.log(a[i]);
							length;i++)
							
							console.log(a[i]);

One of the special feature of an interpreter in reference with the for loop is that	Before each iteration, the interpreter evaluates the variable expression and assigns the name of the property	The iterations can be infinite when an interpreter is used	The body of the loop is executed only once	All of the mentioned			Before each iteration, the interpreter evaluates the variable expression and assigns the name of the property

What will happen if the body of a for/in loop deletes a property that has not yet been enumerated?	The property will be stored in a cache	The loop will not run	That property will not be enumerated	All of the mentioned			That property will not be enumerated
What will be the step of the interpreter in a jump statement when an exception is thrown?	The interpreter stops its work	The interpreter throws another exception	The interpreter jumps to the nearest enclosing exception handler	None of the mentioned			The interpreter jumps to the nearest enclosing exception handler
Consider the following code snippet **while (a != 0)** **{** **if (a == 1)** **continue;** **else** **a++;** **}** **What will be the role of the continue keyword in the above code snippet?**	The continue keyword restarts the loop	The continue keyword skips the next iteration	The continue keyword skips the rest of the statements in that iteration	None of the mentioned			The continue keyword skips the rest of the statements in that iteration

Consider the following code snippet **function f(o)** **{** **if (o === undefined) debugger;** **}** **What could be the task of the statement debugger?**	It does nothing but a simple breakpoint	It debugs the error in that statement and restarts the statement's execution	It is used as a keyword that debugs the entire program at once	All of the mentioned			It does nothing but a simple breakpoint
Among the keywords below, which one is not a statement?	debugger	with	if	use strict			use strict
The unordered collection of properties, each of which has a name and a value is called	String	Object	Serialized Object	All of the these			Object
The object has three object attributes namely	Class, parameters, object's extensible flag	Prototype, class, objects' parameters	Prototype, class, object's extensible flag	Native object, Classes and Interfacces and Object's extensible flag			Prototype, class, object's extensible flag

Consider the following code snippet : **var book = {** **"main title": "JavaScript",** **'sub-title': "The Definitive Guide",** **"for": "all audiences",** **author: {** **firstname: "David",** **surname: "Flanagan"** **}** **};** **In the above snippet, firstname and surname are**	properties	property values	property names	objects			property names
A linkage of series of prototype objects is called as :	prototype stack	prototype chain	prototype class	prototypes			**prototype chain**

Consider the below given syntax book[datatype]=assignment_value; In the above syntax, the datatype within the square brackets must be	An integer	A String	An object	None of the mentioned			A String
To determine whether one object is the prototype of (or is part of the prototype chain of) another object, one should use the	isPrototypeOf() method	equals() method	=== operator	None of the mentioned			isPrototypeOf() method
Consider the following code snippet function f() {}; The above prototype represents a	Function f	A custom constructor	Prototype of a function	Not valid			A custom constructor
The purpose of extensible attribute is to	make all of the own properties of that object nonconfigurable	to configure and bring a writable property	"lock down" objects into a known state and prevent	All of the mentioned			"lock down" objects into a known state and prevent outside tampering

			outside tampering				
Identify the process done in the below code snippet **o = {x:1, y:{z:[false,null,""]}};** **s = JSON.stringify(o);** **p = JSON.parse(s);**	Object Encapsulation	Object Serialization	Object Abstraction	Object Encoding			Object Serialization
The basic purpose of the toLocaleString() is to **a. return**	localised object representation	return a parsed string	return a local time in the string format	return a localized string representation of the object			return a localized string representation of the object

UNIT-IV

Question	Option A	Option B	Option C	Option D	Answer
<script type="text/javascript"> x=4+"4"; document.write(x); </script> Output------?	44	8	4	Error output	44

<script type="text/javascript" language="javascript"> var qpt = "Qualiyt Point Technologies"; var result = qpt.split(" "); document.write(result); </script>	Quality	Q,u,a,l,i,t,y,P,o,i,n,t,T,e,c,h,n,o,l,o,g,i,e,s	Qualiyt,Point,Technologies	QualityPointTechnologies	Qualiyt,Point,Technologies
Is it possible to nest functions in JavaScript?	TRUE	FALSE			TRUE
<script> document.write(navigator.appCodeName); </script>	get code name of the browser of a visitor	set code name of the browser of a visitor	None of the above		get code name of the browser of a visitor
Scripting language are	High Level Programming language	Assembly Level programming language	Machine level programming language		High Level Programming language
<script language="javascript"> function x() { var s= "Good 100%"; var pattern = /\D/g; var output= s.match(pattern); document.write(output); } </script>	Good %	1,0,0	G,o,o,d,%	Error	G,o,o,d,%

<script language="javascript"> var qpt="QUALITY POINT TECHNOLOGIES"; alert(qpt.charAt(qpt.length-1)); </script>	P	E	S	Error	S
Choose the client-side JavaScript object:	Database	Cursor	Client	FileUpLoad	FileUpLoad
Are java and javascript the same?	NO	YES			
<script language="javascript"> function x(z,t) { alert(x.length); } </script> output: ?	Error	2	1	3	Error
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 referes previous object
In JavaScript, Window.prompt() method return true or false value ?	FALSE	TRUE	None of above		FALSE
Math. round(-20.51)=?	20	-21	19	None	20

<script language="javascript">
function x()
{
var s = "Quality 100%!{[!!";
var pattern = /\w/g;
var output = s.match(pattern);
document.write(output);
}
</script>
	%,!,{,[,!,!	Q,u,a,l,i,t,y,1,0,0	Quality 100	Error	Q,u,a,l,i,t,y,1,0,0

<script type="text/javascript" language="javascript">
var qpt= new Array();
qpt[0] = "WebDevelopment";
qpt[1]="ApplicationDevelopment"
qpt[2]="Testing"
qpt[3] = "QualityPointTechnologies";
document.write(qpt[0,1,2,3]);
</script>
	Error	QualityPointTechnologies	WebDevelopment	WebDevelopmnet,ApplicationDevelopment,Testing,QualityPointTechnologies	QualityPointTechnologies

Choose the server-side JavaScript object:	FileUpLoad	Function	File	Date	File
parseFloat(9+10) =?	19	910	None		None
Which of the following are the features of jQuery?	Efficient query method for finding the set of document elem	Expressive syntax for referring to elements in the document	All of the mentioned		All of the mentioned

	ents				
Which of the following is a single global function defined in the jQuery library?	jQuery()	Queryanalysis()	None of the mentioned		jQuery()
Which of the following is a factory function?	$()	jQuery()	Queryanalysis()		jQuery()
Which is the code that asks for the set of all div elements in a document?	var divs = $(div);	var divs = jQuery("div");	var divs = $("div");		var divs = $("div");
Which is the method that operates on the return value of $()?	show()	css()	click()	done()	css()
Consider the following code snippet <script src="jquery-1.4.2.min.js"></script> What does the min mean?	Minimised version	Miniature	Minimised parameters		Minimised version
Which of the following is a heavily overloaded function?	jQuery()	$()	script()	Both a and b	Both a and b
Which of the following is an equivalent replacement of $(document).ready(f)?	jQuery(f)	$(f)	None of the mentioned		$(f)
Which of the following is used for parsing JSON text?	jQuery.each()	jQuery.parseJSON()	jQuery.noConflict()		jQuery.parseJSON()

Which of the following is a utility function in jQuery?	jQuery.each()	jQuery.noConflict()	jQuery.parseJSON()		jQuery.noConflict()
Why is the total size of the page important?	Time taken to download	Size of IP packet should be less than 65500	Size of IP packet should be less than 65535	Both a and c	Both a and c
The word “document” mainly refers to	Dynamic Information	Static Information	Both a and b		Static Information
Which identifier is used to represent a web browser window or frame?	frames	window	location		window
The setTimeout() method is used to	Make the event sleep	Register a function to be invoked after a certain time	Invoke an event after a certain time		Register a function to be invoked after a certain time
Which of the following is a global object?	Register	Location	Window		Window

Consider the following code snippet
 	
function printprops(o)
{
for(var p in o)
console.log(p + ": " + o[p] + "\n");
}
 	
What will the above code snippet result ?
	Prints the contents of each property of o	Returns undefined	Both a and b		Returns undefined

When does the function name become optional in JavaScript?	When the function is defined as a looping statement	When the function is defined as expressions	When the function is predefined		When the function is defined as expressions
What is the purpose of a return statement in a function?	Returns the value and stops executing the function	Stops executing the function and returns the value			Stops executing the function and returns the value
What will happen if a return statement does not have an associated expression?	It returns the value 0	It returns the undefined value	None of the mentioned		It returns the undefined value
A function with no return value is called	Procedures	Method	Static function		Procedures

Consider the following code snippet
 	
function hypotenuse(a, b) {
function square(x)
{
return x*x;
}
return Math.sqrt(square(a) + square(b));
}
 	
What does the above code result?
	Sum of square of a and b	Square of sum of a and b	Sum of a and b square		Sum of square of a and b

Which of the following is the correct code for invoking a function without this keyword at all, and also too determine whether the strict mode is in effect?	var strict = (function { return this; });	var strict = (function() { return !this; }());	mode strict = (function { });		var strict = (function() { return !this; }());
Which is an equivalent code to invoke a function m of class o that expects two arguments x and y?	o(x,y);	o.m(x,y);	o.m(x) && o.m(y);		o.m(x,y);

UNIT-V

questions	opt1	opt2	0pt3	opt4	answer
In Servlet Terminology what provides runtime environment for JavaEE (j2ee) applications. It performs many operations that are given below:	Server	Webserver	Container	Application Server	Container
The following example shows the creation of a import java.applet.*; import java.awt.*;	Banner using Applet	Basic Applet	Display clock	None of the above	Basic Applet
An applet can play an audio file represented by the AudioClip interface in the java, applet package Causes the audio clip to replay continually in which method?	public void play()	public void loop()	public void stop()	None of the above	public void loop()
Which are the common security restrictions in applets?	. Applets can't load libraries or define native methods	An applet can't read every system property	Applets can play sounds	Both A & B	Both A & B

From the following statements which is a drawback for Applet?	It works at client side so less response time	Secured	It can be executed by browsers running under many platforms, including Linux, Windows, and Mac Os etc.	Plugin is required at client browser to execute applet	Plugin is required at client browser to execute applet
Applet works at client side so less response time.	True	False	depending upon situation	Not Always true	True
The APPLET tag is used to start an applet from both an HTML document and from an applet viewer.	True	False	depending upon situation	Not Always true	True
Applets cannot make network connection exception to the server host from which it originated.	True	False	depending upon situation	Not Always true	True
What invokes immediately after the start() method and also any time the applet needs to repaint itself in the browser?	stop()	init()	paint()	init()	paint()
Which method is called only once during the run time of your applet?	stop()	paint()	init()	init()	init()
When an applet is terminated which of the following sequence of methods calls take place?		stop(),paint(),destroy()	destroy(),stop(), paint()	. stop(),destroy()	. stop(),destroy()
Applet runs inside the browser and does not works at client side.	. True	False	depending upon situation	Not Always true	False
Which is a special type of program that is embedded in the webpage to generate the dynamic content?	. Package	Applet	Browser	None of the above	Applet
What is used to run an Applet?	An html file	An AppletViewer tool(for testing purpose)	Both A & B	None of the above	None of the above

Which is the correct order of lifecycle in an applet?	. Applet is started,initialized,painted,destroyed,stopped	Applet is painted,started,stopped,initilaized, destroyed	Applet is initialized,started,painted,stopped,destroyed	None of the above	Applet is initialized,started,painted,stopped, destroyed
Java Plug-in software is not responsible to manage the lifecycle of an Applet.	TRUE	False			False
Which method is used to suspend threads that don't need to run when the applet is not visible?	destroy()	paint()	stop()	start()	stop()
All Applets must import java.applet and java.awt.	init(),paint(),start()	Start(),paint(),init()	intit(),start(),paint()	paint(),start(),init()	intit(),start(),paint()
Which method is first Called for any applet when it starts its execution?	void init()	void destroy()	boolean isActive()	None of the above	void init()
In _____ attacks, the attacker manages to get an application to execute an SQL query created by the attacker.	SQL injection	SQL	Direct	Application	SQL injection
A Web site that allows users to enter text, such as a comment or a name, and then stores it and later displays it to other users, is potentially vulnerable to a kind of attack called a _____ attack.	Two-factor authentication	Cross-site request forgery	Cross-site scripting	Cross-site scoring scripting	Cross-site scoring scripting
_____ is an attack which forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated.	Two-factor authentication	Two-factor authentication	Cross-site scripting	Multiple-site scoring scripting	Cross-site scripting
Many applications use _____, where two independent factors are used to identify a user.	Two-factor authentication	Cross-site request forgery	Cross-site scripting	Cross-site scoring scripting	Cross-site scoring scripting
Even with two-factor authentication, users may still be vulnerable to_____attacks.	Radiant	Cross attack	scripting	Man-in-the-middle	Man-in-the-middle

A single _____ further allows the user to be authenticated once, and multiple applications can then verify the user's identity through an authentication service without requiring reauthentication.	OpenID	Sign-on system	Security Assertion Markup Language (SAML)	Virtual Private Database (VPD)	Virtual Private Database (VPD)
The _____ is a standard for exchanging authentication and authorization information between different security domains, to provide cross-organization single sign-on.	OpenID	Sign-on system	Security Assertion Markup Language (SAML)	Virtual Private Database (VPD)	Security Assertion Markup Language (SAML)
The _____ standard is an alternative for single sign-on across organizations, and has seen increasing acceptance in recent years.	OpenID	Single-site system	Security Assertion Markup Language (SAML)	Virtual Private Database (VPD)	OpenID
_____ allows a system administrator to associate a function with a relation; the function returns a predicate that must be added to any query that uses the relation. a) OpenID	Single-site system	Security Assertion Markup Language (SAML)	Virtual Private Database (VPD)		Virtual Private Database (VPD)
. VPD provides authorization at the level of specific tuples, or rows, of a relation, and is therefore said to be a _____ mechanism.	Row-level authorization	Column-level authentication	Row-type authentication	) Authorization security	Row-level authorization
. If a piece of data is stored in two places in the database, then	Storage space is wasted	Changing the data in one spot will cause data inconsistency	In can be more easily accessed	Both a and b	Both a and b

An audit trail _____ .	Is used to make backup copies	Is the recorded history of operations performed on a file	Both a and b	None of the mentioned	Both a and b
Large collection of files are called _____ .	Fields	Records	Database	Sectors	Database