Java Script:

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Note: Java and JavaScript are two completely different languages in both concept and design!

Advantages:

- **JavaScript can interact with events -** A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- **JavaScript can read and write HTML elements -** A JavaScript can read and change the content of an HTML element
- **JavaScript can be used to validate data -** A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
- **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer

Disadvantages

- **Security Issues**
  Javascript snippets, once appended onto web pages execute on client servers immediately and therefore can also be used to exploit the user's system. While a certain restriction is set by modern web standards on browsers, malicious code can still be executed complying with the restrictions set.
- **Javascript rendering varies**

The HTML <script> tag is used to insert a JavaScript into an HTML page.

```
<html>
<body>
 <script type="text/javascript">
document.write("Hello World!");
</script> </body> </html>
```

To insert a JavaScript into an HTML page, we use the <script> tag. Inside the <script> tag we use the type attribute to define the scripting language.

So, the <script type="text/javascript"> and </script> tells where the JavaScript starts and The **document.write** command is a standard JavaScript command for writing output to a page.

By entering the document.write command between the <script> and </script> tags, the browser will recognize it as a JavaScript command and execute the code line. In this case the browser will write Hello World! to the page:

JavaScript's in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.

Identifier:

1. Integer
2. Floating point numbers
3. Strings
4. Booleans

Expression:

<HTML> <Body>
<script>
var a=10;
var b=20;
var c=a+b;
var d="jaipal";
document.write(c+d);
</script>
</BODY> </HTML>
	<HTML> <BODY>
	<script>
	a=10;
	b=20;
	c=a+b;
	d="jaipal";
	document.write(c+d);
	</script>
	</BODY> </HTML>

Java Key Words:

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

JavaScript Assignment Operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

Java Script Operators:

Operator	Description	Example	Result	
+	Addition	x=y+2	x=7	y=5
-	Subtraction	x=y-2	x=3	y=5
*	Multiplication	x=y*2	x=10	y=5
/	Division	x=y/2	x=2.5	y=5
%	Modulus (division remainder)	x=y%2	x=1	y=5
++	Increment	x=++y	x=6	y=6
		x=y++	x=5	y=6
--	Decrement	x=--y	x=4	y=4
		x=y--	x=5	y=4

The + Operator Used on Strings(String Oprator)

The + operator can also be used to add string variables or text values together.

To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.

Given that **x=5**, the table below explains the comparison operators:

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

Logical Operators

Logical operators are used to determine the logic between variables or values.

Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	**and**	**(x < 10 && y > 1) is true**
||	**or**	**(x==5 || y==5) is false**
!	**not**	**!(x==y) is true**

Standard Event Attributes

HTML 4 added the ability to let events trigger actions in a browser, like starting a JavaScript when a user clicks on an element.

<body> and <frameset> Events

The two attributes below can only be used in <body> or <frameset>:

Attribute	Value	Description
onload	script	Script to be run when a document load
onunload	script	Script to be run when a document unload

Form Events

The attributes below can be used in form elements:

Attribute	Value	Description
onblur	script	Script to be run when an element loses focus
onchange	script	Script to be run when an element changes
onfocus	script	Script to be run when an element gets focus
onreset	script	Script to be run when a form is reset
onselect	script	Script to be run when an element is selected
onsubmit	script	Script to be run when a form is submitted

Image Events

The attribute below can be used with the img element:

Attribute	Value	Description
onabort	script	Script to be run when loading of an image is interrupted

Keyboard Events

Valid in all elements except base, bdo, br, frame, frameset, head, html, iframe, meta, param, script, style, and title.

Attribute	Value	Description
onkeydown	script	Script to be run when a key is pressed
onkeypress	script	Script to be run when a key is pressed and released
onkeyup	script	Script to be run when a key is released

Mouse Events

Valid in all elements except base, bdo, br, frame, frameset, head, html, iframe, meta, param, script, style, and title.

Attribute	Value	Description
onclick	script	Script to be run on a mouse click
ondblclick	script	Script to be run on a mouse double-click
onmousedown	script	Script to be run when mouse button is pressed
onmousemove	script	Script to be run when mouse pointer moves
onmouseout	script	Script to be run when mouse pointer moves out of an element
onmouseover	script	Script to be run when mouse pointer moves over an element
onmouseup	script	Script to be run when mouse button is released

Conditional Statements

In JavaScript we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if the condition is true and another code if the condition is false
- **if...else if....else statement** - use this statement to select one of many blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

If Statement

Use the if statement to execute some code only if a specified condition is true.

Syntax

```
if (condition)
  {
  code to be executed if condition is true
  }
```

The JavaScript Switch Statement

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
switch(n)
{
case 1:
  execute code block 1
  break;
case 2:
  execute code block 2
  break;
default:
  code to be executed if n is different from case 1 and 2
```

If...else if...else Statement
Use the if....else if...else statement to select one of several blocks of code to be executed.
Syntax

```
if (condition1)
  {
  code to be executed if condition1 is true
  }
else if (condition2)
  {
  code to be executed if condition2 is true
  }
else
  {
  code to be executed if condition1 and condition2 are not true
  }
```

JavaScript Loops

In JavaScript, there are two different kind of loops:

- **for** - loops through a block of code a specified number of times
- **while** - loops through a block of code while a specified condition is true

The for Loop

The for loop is used when you know in advance how many times the script should run.
Syntax

```
for (variable=startvalue;variable<=endvalue;variable=variable+increment)
{
code to be executed
}
```

The while Loop

The while loop loops through a block of code while a specified condition is true.
Syntax

```
while (variable<=endvalue)
  {
  code to be executed
  }
```

The do...while Loop

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.
Syntax

```
do
  {
  code to be executed
  }
while (variable<=endvalue);
```

JavaScript Popup Boxes

1. Alert Box
2. Confirm Box
3. Prompt Box

Alert Box
An alert box is often used if you want to make sure information comes through to the user.
When an alert box pops up, the user will have to click "OK" to proceed.
Syntax

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

Example

```
<html> <head>
<script type="text/javascript">
function show_alert()
{
alert("I am an alert box!");
}
</script></head> <body> <input type="button" onclick="show_alert()" value="Show alert box" />
```

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

Confirm Box

A confirm box is often used if you want the user to verify or accept something.
When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax

```
confirm("sometext");
```

Example

```
<html> <head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button");
if (r==true)
  {
  document.write("You pressed OK!");
  }
else
  {
  document.write("You pressed Cancel!");
  }
}
</script>
</head> <body>
<input type="button" onclick="show_confirm()" value="Show confirm box" />
</body> </html>
```

Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.
When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax

```
prompt("sometext","defaultvalue");
```

Example

```
<html> <head>
<script type="text/javascript">
function show_prompt()
{
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
  {
  document.write("Hello " + name + "! How are you today?");
  }
}
</script></head> <body>
<input type="button" onclick="show_prompt()" value="Show prompt box" />
 </body> </html>
```

String Object Methods

Method	Description
charAt()	Returns the character at the specified index
charCodeAt()	Returns the Unicode of the character at the specified index
concat()	Joins two or more strings, and returns a copy of the joined strings
fromCharCode()	Converts Unicode values to characters
indexOf()	Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last found occurrence of a specified value in a string
match()	Searches for a match between a regular expression and a string, and returns the matches
replace()	Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring
search()	Searches for a match between a regular expression and a string, and returns the position of the match
slice()	Extracts a part of a string and returns a new string
split()	Splits a string into an array of substrings
substr()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
substring()	Extracts the characters from a string, between two specified indices
toLowerCase()	Converts a string to lowercase letters
toUpperCase()	Converts a string to uppercase letters
valueOf()	Returns the primitive value of a String object

Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y,x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of E^x
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x,y,z,...,n)	Returns the number with the highest value
min(x,y,z,...,n)	Returns the number with the lowest value

pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

Date Object Methods

Method	**Description**
getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year (four digits)
getHours()	Returns the hour (from 0-23)
getMilliseconds()	Returns the milliseconds (from 0-999)
getMinutes()	Returns the minutes (from 0-59)
getMonth()	Returns the month (from 0-11)
getSeconds()	Returns the seconds (from 0-59)
getTime()	Returns the number of milliseconds since midnight Jan 1, 1970
getTimezoneOffset()	Returns the time difference between GMT and local time, in minutes
getUTCDate()	Returns the day of the month, according to universal time (from 1-31)
getUTCDay()	Returns the day of the week, according to universal time (from 0-6)
getUTCFullYear()	Returns the year, according to universal time (four digits)
getUTCHours()	Returns the hour, according to universal time (from 0-23)
getUTCMilliseconds()	Returns the milliseconds, according to universal time (from 0-999)
getUTCMinutes()	Returns the minutes, according to universal time (from 0-59)
getUTCMonth()	Returns the month, according to universal time (from 0-11)
getUTCSeconds()	Returns the seconds, according to universal time (from 0-59)
getYear()	Deprecated. Use the getFullYear() method instead
parse()	Parses a date string and returns the number of milliseconds since midnight of January 1, 1970
setDate()	Sets the day of the month (from 1-31)
setFullYear()	Sets the year (four digits)
setHours()	Sets the hour (from 0-23)
setMilliseconds()	Sets the milliseconds (from 0-999)
setMinutes()	Set the minutes (from 0-59)
setMonth()	Sets the month (from 0-11)
setSeconds()	Sets the seconds (from 0-59)
setTime()	Sets a date and time by adding or subtracting a specified number of milliseconds

	to/from midnight January 1, 1970
setUTCDate()	Sets the day of the month, according to universal time (from 1-31)
setUTCFullYear()	Sets the year, according to universal time (four digits)
setUTCHours()	Sets the hour, according to universal time (from 0-23)
setUTCMilliseconds()	Sets the milliseconds, according to universal time (from 0-999)
setUTCMinutes()	Set the minutes, according to universal time (from 0-59)
setUTCMonth()	Sets the month, according to universal time (from 0-11)
setUTCSeconds()	Set the seconds, according to universal time (from 0-59)
setYear()	Deprecated. Use the setFullYear() method instead
toDateString()	Converts the date portion of a Date object into a readable string
toGMTString()	Deprecated. Use the toUTCString() method instead
toLocaleDateString()	Returns the date portion of a Date object as a string, using locale conventions
toLocaleTimeString()	Returns the time portion of a Date object as a string, using locale conventions
toLocaleString()	Converts a Date object to a string, using locale conventions
toString()	Converts a Date object to a string
toTimeString()	Converts the time portion of a Date object to a string
toUTCString()	Converts a Date object to a string, according to universal time
UTC()	Returns the number of milliseconds in a date string since midnight of January 1, 1970, according to universal time
valueOf()	Returns the primitive value of a Date object

What is an Array?

An array is a special variable, which can hold more than one value, at a time

Create an Array

An array can be defined in three ways.
The following code creates an Array object called myCars:

1: var myCars=new Array(); // regular array (add an optional integer
myCars[0]="Saab"; // argument to control array's size)
myCars[1]="Volvo";
myCars[2]="BMW";
2: var myCars=new Array("Saab","Volvo","BMW"); // condensed array
3: var myCars=["Saab","Volvo","BMW"]; // literal array

Note: If you specify numbers or true/false values inside the array then the variable type will be Number or Boolean, instead of String.

Access an Array

You can refer to a particular element in an array by referring to the name of the array and the index number. The index number starts at 0.
The following code line:
document.write(myCars[0]);
will result in the following output:
Saab

Array Object Methods

Method	Description
concat()	Joins two or more arrays, and returns a copy of the joined arrays
indexOf()	
join()	Joins all elements of an array into a string
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element
slice()	Selects a part of an array, and returns the new array
sort()	Sorts the elements of an array
splice()	Adds/Removes elements from an array
toString()	Converts an array to a string, and returns the result
unshift()	Adds new elements to the beginning of an array, and returns the new length
valueOf()	Returns the primitive value of an array

Date Programes:

```
<HTML>
<BODY>
<script>
var myDate=new Date();
document.write("<h1>"+myDate);
</script>
</BODY></HTML>
```

OutPut: Sat Feb 4 08:44:16 UTC+0530 2012

```
<HTML>
<BODY>
<script>
var myDate=new Date();
document.write("<h1>"+myDate+"<br>"+myDate.
getDay()+"<br>"+myDate.getMonth()+"<br>"+my
Date.getYear()+"<br>"+myDate.getHours()+"<br>
"+myDate.getSeconds()+"<br>"+myDate.getTime(
));
</script>
</BODY>
</HTML>
```

OutPutL:Sat Feb 4 08:48:31 UTC+0530
2012
6
1
2012
8
31
1328325881140

Mathematical Object Programs:

```
<HTML>
<BODY>
<HR>
<H1>Math.abs()</H1>
<script type="text/javascript">
document.write(Math.abs(7.25) + "<br />");
document.write(Math.abs(-7.25) + "<br />");
document.write(Math.abs(null) + "<br />");
document.write(Math.abs("Hello") + "<br />");
document.write(Math.abs(7.25-10));
</script>
<HR><H1>Math.ceil</H1>
<script type="text/javascript">
document.write(Math.ceil(0.60) + "<br />");
document.write(Math.ceil(0.40) + "<br />");
document.write(Math.ceil(5) + "<br />");
document.write(Math.ceil(5.1) + "<br />");
document.write(Math.ceil(-5.1) + "<br />");
document.write(Math.ceil(-5.9));
</script>
<HR><H1>Math.floor</H1>
<script type="text/javascript">
document.write(Math.floor(0.60) + "<br />");
document.write(Math.floor(0.40) + "<br />");
document.write(Math.floor(5) + "<br />");
document.write(Math.floor(5.1) + "<br />");
document.write(Math.floor(-5.1) + "<br />");
document.write(Math.floor(-5.9));
</script>
<HR><H1>Math.sqrt</H1>
<script type="text/javascript">
document.write(Math.sqrt(0) + "<br />");
document.write(Math.sqrt(1) + "<br />");
document.write(Math.sqrt(9) + "<br />");
document.write(Math.sqrt(0.64) + "<br />");
document.write(Math.sqrt(-9));
</script>
<HR><H1>Math.pow</H1>
<script type="text/javascript">
document.write(Math.pow(0,0) + "<br />");
document.write(Math.pow(0,1) + "<br />");
document.write(Math.pow(1,1) + "<br />");
document.write(Math.pow(1,10) + "<br />");
document.write(Math.pow(7,2) + "<br />");
document.write(Math.pow(-7,2) + "<br />");
document.write(Math.pow(2,4));
</script>
<HR><H1>Math.round</H1>
<script type="text/javascript">
document.write(Math.round(0.60) + "<br />");
document.write(Math.round(0.50) + "<br />");
document.write(Math.round(0.49) + "<br />");
document.write(Math.round(-4.40) + "<br />");
document.write(Math.round(-4.60));
</script>
</BODY>
</HTML>
```

OutPut:

```
Math.abs()
7.25
7.25
0
NaN
2.75
Math.ceil
1
1
5
6
-5
-5
Math.floor
0
0
5
5
-6
-6
Math.sqrt
0
1
3
0.8
NaN
Math.pow
1
0
1
1
49
49
16

Math.round
1
1
0
-4
-5
```