Unit -2

CLIENT SIDE PROGRAMMING

2.1 SCRIPTING LANGUAGES

Scripting languages are becoming more popular due to the emergence of web-based applications. The new scripting languages allow users with little or no programming expertise to develop interactive web pages with minimum effort.

A scripting language is a programming language that supports scripts and programs that are written for a special run-time environment. They are interpreted rather than complied.

An interpreter checks the syntax of the code and generates object code one source line at a time.The language is interpreted at run-time so that the instructions are executed immediately. There are two types of scripting languages: Server side scripting languages and Client side scripting languages

Server side scripting Languages

Server side scripting Languages are run on a web server (back end). This environment is known as **Server side scripting environment.**The user issues a request and it is fulfilled by running a script, directly on the web server.The web server will generate dynamic HTML pages as the output according to the script. This HTML is then sent to the client browser. These languages are mostly used in interactive web sites that are connected to databases.

Examples: PERL, ASP (Active Server Pages).

Advantages of Server side scripting Languages

These languages support high customization of the response based on the user's requirements.

Disadvantages of Server side scripting Languages

They impart more load to the web server.They can introduce processing overhead that can decrease performance and force the user to wait for the page to be processed and

recreated. Once the page is posted back to the server, the client must wait for the server to process the request and send the page back to the client.

Client side scripting Languages

They run on a browser (front end). This environment is known as **Client side scripting environment.**The processing of the scripts takes place on the end users computer. The source code of the requested service or web page is transferred from the web server to the end users computer over the internet and run directly in the browser.The scripting language must be enabled on the client computer. They make interactive and dynamic webpages.They also interact with temporary storage and local storage and provide remote services for client-side applications.**Examples:** VB script, JavaScript.

Advantages of Client side scripting Languages

- No load on the server since all the processing is done in the browser.
- These languages are easier than server side scripting languages.

Disadvantages of Client side scripting Languages

- Minimal customization of web pages**.**

Difference between server side and client side scripting languages

Server side Scripting Languages	Client side Scripting Languages
The scripting code is run at the back end (i.e.) at the web server.	The scripting code is run in the back end (i.e) in the end user's browser.
They are less interactive.	They are more interactive.
Any change by these languages will be reflected on the database.	The changes are done only at the client side, so the database will not get affected.
More overhead on the server.	The overhead is on the local browser.
The server side scripts are not visible to the user.	The client side scripts are visible to the user.
They allow a level of privacy and personalization.	The security features are less efficient.

Advantages of scripting languages:

- Any errors in the scripting language will terminate the execution of the source code.They have a simple syntax.They are easy to learn and use.

- This does not require programming expertise.It allows complex tasks to be performed in relatively few steps.It allows simple creation and editing. It could be done in a variety of text editors
- It facilitates the addition of dynamic and interactive activities to web pages. They are portable across various hardware and network platforms and scripts can be embedded in standard text document also.
- Instantaneous error reporting and error correction.The debugging process is also easy.

Disadvantages of scripting languages

- Dubious web sites or unauthorized programs are easily accessed without the user's knowledge because the executable code is run on the end user's browser.
- The above action may harm the end user's system.

Difference between programming languages and scripting languages

Programming Languages	Scripting Languages
They are compiled.	They are interpreted.
They cannot be run directly without compilation.	They can be directly run. No explicit compilation is needed.
They have complete syntax and semantic rules.	They are unstructured subset of programming language.
They are used to build applications.	They are used to control the behavior of an application.

2.2 INTRODUCTION TO JAVASCRIPT

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

Features of JavaScript

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

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

Capabilities of JavaScript

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

Placement of JavaScript in a HTML File:

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

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

Inline JavaScript

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

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

External JavaScript

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

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

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

</script>

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

Advantages of external JavaScript

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

External JavaScript

external.js

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

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

Differences between inline and external javascript

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

Creating a simple web page with JavaScript

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

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

Syntax: <script> JavaScript code</script>

The attributes of the script tag are:

- **Language:**

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

- **Type**

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

  JavaScript code </script>

Simple JavaScript

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


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

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

Comments in JavaScript

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

Data types

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

Reserved words

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

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

JavaScript Variables

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

Syntax:varvariable_name;

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

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

```
{ return “example function” }
```

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

Naming variables

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

Scope of a variable

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

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

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

Special Keywords

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

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

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

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

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

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

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

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

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

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

typeof Operator

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

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

JavaScript Statements

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

Comment Statements

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

Exception Handling Statements

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

Conditional Statements

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

a) Simple if

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

b) if else statement

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

c) if-else ladder

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

If-else ladder

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

d) Switch statement

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

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

Looping Statements

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

a) While loop

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

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

b) do-while loop

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

Do-while


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

Output:

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

c) for loop

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

For loop

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

Break and Continue Statements

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

Functions in JavaScript

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

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

Function functionname(parameter-list)

{ statements}

</script>

Calling a Function:

The syntax to invoke a function is:

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

Functions

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

Output:

This is my first function

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

Alert Dialog box
Dialog boxes

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

Alert dialog box

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

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

```
<head>
<script type="text/javascript">
  alert("Hello there");
</script></head
```

Confirmation Dialog Box:

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

Confirmation dialog box

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

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

Prompt Dialog Box:

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

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

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

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

Prompt dialog box

```
<head>
<script  type="text/javascript">
varretVal = prompt("Enter your name :”);
</script></head>
```

2.6 DHTML WITH JAVASCRIPT

- One of the most popular uses of JavaScript is DHTML (Dynamic HyperText Markup Language).
- DHTML is the combination of HTML and JavaScript.DHTML is using JavaScript to modify the CSS styles of HTML elements.
- DHTML is the combination of several built-in browser features in fourth generation browsers that enable a web page to be more dynamic.
- The HTML document acts as a reference to the DHTML. The DHTML can change the visibility, position, contents, background colour, z-index, clipping, size of the already positioned element. New elements can also be added to the HTML document.

Differences between HTML and DHTML

HTML	DHTML
A plain page without any styles and Scripts called as HTML.	A page with HTML, CSS, DOM and Scripts called as DHTML.
HTML sites will be slow upon client-side technologies.	DHTML sites will be fast enough upon client-side technologies.

HTML stands for only static pages. It is referred as a static HTML and static in nature.	DHTML is Dynamic HTML means HTML+JavaScript. Hence it is referred as a dynamic HTML.

Components of DHTML

Dynamic HTML includes the following components: HTML, Cascading Style Sheets, Scripting and the Document Object Model.

- **HTML**:
  - HTML defines the structure of a Web page, using such basic elements as headings, forms, tables, paragraphs and links.
- **Cascading Style Sheets (CSS)**:
  - A style sheet controls the formatting of HTML elements.
  - Style sheets are used to specify page margins, point sizes and leading.
  - Cascading Style Sheets is a method to determine precedence and to resolve conflicts when multiple styles are used.
- **Scripting**:
  - Scripting provides the mechanisms to interpret user actions and produce client-side changes to a page.
  - DHTML can communicate with several scripting languages but JavaScript is widely used.
- **Document Object Model (DOM)**:
  - The DOM outlines Web page content in a way that makes it possible for HTML elements, style sheets and scripting languages to interact with each other.
  - The W3C defines the DOM as a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure, and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented stage.

Positioning elements in DHTML

- There are two types of positioning: absolute and relative.
- **Absolute positioning** allows to place an element anywhere in relation to the page.
- **Relative positions** the element based on offset values.
- Most DHTML is done with absolutely positioned elements. Relatively positioned elements do not accept the 'clip' style and do not allow their clipping to be changed.

JavaScript and Cascading Style Sheets (CSS)

- Cascading Style Sheets are the standard way to define the presentation of the DHTML pages, from fonts and colors to the complete layout of a page. They are much more efficient than using HTML.
- CSS files are termed "cascading" stylesheets because of two reasons: one style sheet can cascade, or have influence over, multiple pages. Similarly, many CSS files can define a single page.
- CSS is the most important feature of DHTML. The CSS has various style properties.There are 3 ways to implement css commands into the site:

    1. Use one CSS file for all pages.
    2. Integrate CSS commands into the head of each of the documents.
    3. Use the style attribute to put CSS code directly into a HTML element.

Common style properties of CSS

1. **Position** -Specifies how the block should be positioned on the page with respect to other page elements.
    - position:absolute- Block is positioned absolutely within the browser window,relative to <BODY> block.
    - position:relative- Block is positioned relative to its parent block, if any, or else normal flow of page.
    - position:static- Block is positioned according to standard HTML layout rules.
2. **width**-Specifies the width at which the block's contents should wrap. Width may be in measured units (50px), as a percentage of the parent block's width (50%), or auto which wraps the block according to its parent's width.

    Examples: width:50px or width:50%
3. **height**-Specifies the height of the block, measured in units (50px), percentage of the parent block's height (50%), or auto. The height of the block will be forced to the minimum necessary to display its contents.

    Examples: height:50px or height:50%
4. **left**-Specifies the offset of the left edge of the block in accordance with the position attribute. Positive measures (5px) are offset towards the right while negative measures (-5px) are offset towards the left.

    Examples: left:5px or left:-5px

5. **top-** Specified the offset from the top edge of the block in accordance with the position attribute. Positive measures (5px) are offset towards the bottom of the page while negative measures (-5px) are offset towards the top of the page.

   Examples: top:10px or top:-10px

6. **clip-**Specifies a rectangular portion of the block which is visible. The syntax of this property is different for different browsers.

   1. MSIE: clip:rect(top right bottom left)

      Example:clip:rect(0px 30px 50px 0px)

   2. Netscape: clip:rect(left,top,right,bottom)

      Example:clip:rect(0,0,30,50)

7. **visibility-**Specifies whether a block should be visible. If not visible, the block will notappear on the page, although you can make it visible later using JavaScript. The possible values for this property vary between browsers.

   1. MSIE

      visibility:inherit- Block inherits the visibility property of its parent.

      visibility:visible- Block is visible.

      visibility:hidden-Block is hidden or invisible

   2. Netscape:

      visibility:inherit- Block inherits the visibility property of its parent.

      visibility:show- Block is visible.

      visibility:hide- Block is invisible.

8. **z-index-** Specifies the "**stacking order**" of blocks, should they happen to overlap other positioned blocks. A block is assigned a z-index, which is any integer. When blocks overlap, that which has the greater positive z-index appears above a block with a lower z-index. Blocks with an equal z-index value are stacked according to the order in which they appear in the source code (bottom-to-top: first block defined appears on bottom, last block defined appears on top).

   Example: z-index:3

9. **backgroundcolor-**Specifies the background color for the block.

   Example:background-color:green or background-color:FF8F00

10. **backgroundimage-**Specifies a background image for the block.

    Example: background-image:url('images/cat.jpg')

Advantages of CSS

- Pages download faster.Less code and the pages are shorter and neater.
- The look of the site is kept consistent throughout all the pages that work off the same stylesheet.
- Updating the design and general site maintenance are made much easier, and errors caused by editing multiple HTML pages occur far less often.
- The ID field is most important, because id will be used to reference the positioned element. Browsers call these positioned elements as **layers.**

Grouping elements in CSS:

1. : The element is what you could call a neutral element which does not add anything to the document itself. But with CSS, can be used to add visual features to specific parts of text in your documents.
2. :Whereas is used within a block-level element as seen in the previous example, is used to group one or more block-level elements.

Object Referencing

- The simplest way to reference an element in a DHTML document is by using the element's id attribute. The element is represented as an object, and its various XHTML attributes become properties that can be manipulated by scripting.

Changing text using DHTML

```
<html><head><title>Object Model</title>
<script type = "text/javascript">
 function start()
 {alert(  pText.innerText   );
Text.innerText = "Thanks for coming.";}
</script></head>
<body onload = "start()">
<p id = "pText">Welcome to our Web page!</p>
</body></html>
```

- The on load calls JavaScript start function when document loading completes.Start() displays an alert box containing the value of pText.innerText.
- The object pText refers to the p element whose id is set to pText. The **innerText property** of the object refers to the text contained in that element in this example it is Welcome to our Web page!.
- The start() sets the innerText property of pText to a different value. Changing the text displayed on screen is a Dynamic HTML ability called **dynamic content.**

Collections all and children

- **Collections** are arrays of related objects on a page. Collections provide an easy way of referring to any specific element without an id.
- There are several special collections in the object model.The **all collection** contains all the XHTML elements in a document.
- The **length property** of the collection specifies the size of the collection. The **children** collection of a specific element contains that element's child elements. For example, an html element has only two children—the head element and the body element.

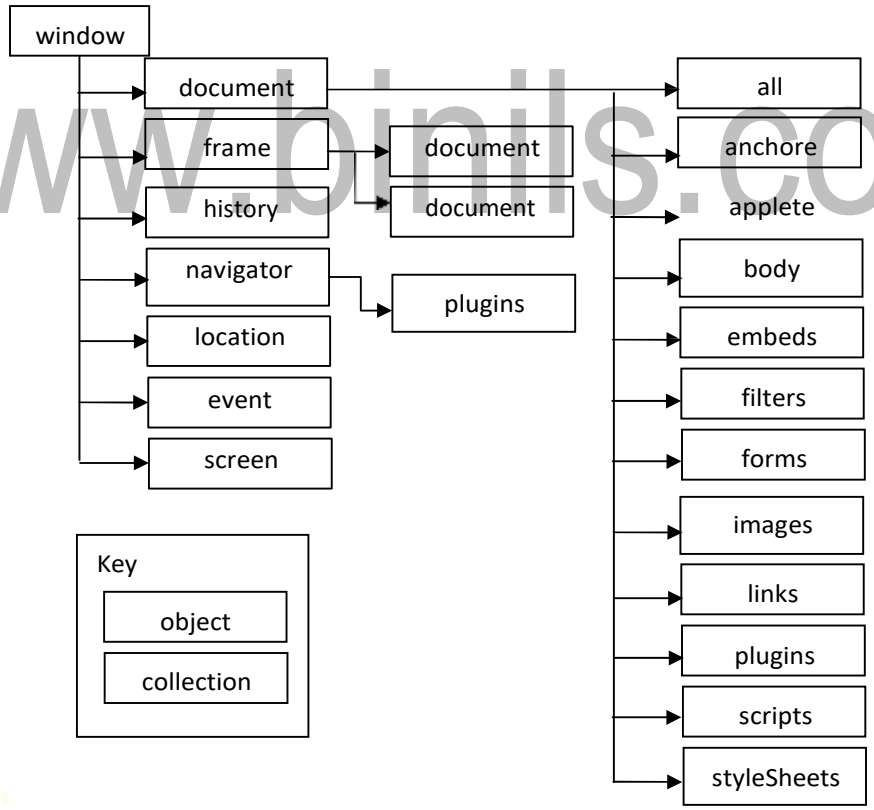


Figure 2.2 Object collections

Object	Description
window	This object represents the browser window and provides access to the document object contained in the window. If the window contains frames, a separate window object is created automatically for each frame, to provide access to the document rendered in that frame. Frames are considered to be subwindows in the browser.
document	This object represents the XHTML document rendered in a window.The document object provides access to every element in the XHTMLdocument and allows dynamic modification of the XHTML document.body This object provides access to the body element of an XHTML document.
history	This object keeps track of the sites visited by the browser user. The object provides a script programmer with the ability to move forward and backward through the visited sites, but for security reasons does not allow the actual site URLs to be manipulated.
navigator	This object contains information about the Web browser, such as the name of the browser, the version of the browser etc.
location	This object contains the URL of the loaded document. When this object is set to a new URL, the browser immediately switches (navigates) to the new location.
Event	This object can be used in an event handler to obtain information about the event that occurred.
screen	The object contains information about the computer screen for the computer on which the browser is running. Information such as the width and height of the screen in pixels can be used to determine the size at which elements should be rendered in a Web page.

Collections	Descriptions
all	Many objects have an all collection that provides access to every element contained in the object. For example, the body object's all collection provides access to every element in the body element of an XHTML document.
anchors	This collection contains all anchor elements (a) that have a name or id attribute. The elements appear in the collection in the order they were defined in the XHTML document.
applets	This collection contains all the applet elements in the XHTML document. Currently, the most common applet elements are Java applets.

embeds	This collection contains all the embed elements in the XHTML document.
forms	This collection contains all the form elements in the XHTML document. The elements appear in the collection in the order they were defined in the XHTML document.
frames	This collection contains window objects that represent each frame in the browser window. Each frame is treated as its own subwindow.

```
<html ><head><title>Object  Model</title>
<style type = "text/css">
    .bigText { font-size: 3em; font-weight: bold }
    . smallText { font-size: .75em }
</style>
<script type = "text/javascript">
function start()
{
varinputClass = prompt("Enter a className for the text " + "(bigText or smallText)", "" );
pText.className = inputClass;}
</script></head>
<body onload = "start()">
<p id = "pText">Welcome to our Web site!</p>
</body></html>
```


The above example contains two classes: .bidText and .smallText. The **class** attribute applies a style class to an element.The class name always follows a period operator (.). Similar properties can be grouped into one class.The property name is followed by a colon (:) and the value of that property. Multiple properties are separated by semicolons (;).

Dynamic Positioning

In dynamic positioning, the XHTML elements can be positioned with scripting. This is done by declaring an element's CSS position property to be either absolute or relative, and then moving the element by manipulating any of the top, left, right or bottom CSS properties.

```
<html><head><title>Dynamic Positioning</title>
<script type = "text/javascript">
var speed = 5;var count = 10; var direction = 1; varfirstLine = "Text growing";
varfontStyle = [ "serif", "sans-serif", "monospace" ];
varfontStylecount = 0;
function start()
 { window.setInterval( "run()", 100 ); }
function run()
{ count +=  speed;
 if ( ( count % 200 ) == 0 )
{ speed *=  -1;
 direction = !direction;
pText.style.color =35 ( speed< 0 ) ? "red"  : "blue"  ;
firstLine = ( speed < 0 ) ? "Text shrinking" : "Text growing";
pText.style.fontFamily = fontStyle[ ++fontStylecount % 3 ];
 }
pText.style.fontSize = count / 3;
pText.style.left = count;
pText.innerHTML = firstLine + "<br /> Font size: " + count + "px";
 }
</script></head>
<body onload = "start()">
<p id = "pText" style = "position: absolute; left: 0; font-family: serif; color: blue">
Welcome!</p></body></html>
```

- In the above example the position of the element is varied on the page by accessing its CSS left attribute, scripting to vary the color, fontFamily and fontSize attributes, and the element's innerHTML property is used to alter the content of the element.
- This function set interval takes two parameters—a function name, and how often to run that function (in this case, every 100 milliseconds). The setTimeout() takes the same parameters but instead waits the specified amount of time before calling the named function only once.

Frame collection

- Frames are seen as collection in DHTML.Usage of frames makes the web page more interactive.

Cross frames

```
<html><head><title>Frames collection</title></head>
<frameset rows = "100, *">
<frame src = "frame.html" name = "upper" />
<frame src = "" name = "lower" /></frameset></html>
```

Frame.html

```
function start()
{ var text = prompt( "What is your name?", "" );
parent.frames( "lower" ).document.write( "<h1>Hello, " + text + "</h1>" );
 }}</script></head><body onload = "start()">
<h1>Cross-frame scripting!</h1>
</body></html>
```

Filters and transitions

Applying filters to text and images causes changes that are persistent.**Transitions** are temporary phenomena. Applying a transition allows to transfer from one page to another with a pleasant visual effect.Filters and transitions do not add content to the pages. They just add

visual effects that work on some event.Filters and transitions are specified with the CSS **filter** property. Transitions give the same kind of graphics capabilities got from presentation software like Microsoft's PowerPoint.Filters are applied in the style attribute. The filter property's value is the name of the filter. Each filter has a property named enabled. If this property is set to true, the filter is applied. If it is set to false, the filter is not applied.

Filter	Description	Syntax
Flipv	Mirrors text vertically	< style = "filter: flipv ">Text</style>
Fliph	Mirrors text horizontally	< style = "filter: fliph ">Text</style>
Chroma	Applies transparency effects dynamically, without using a graphics editor to hard-code transparency into the image. This filter must be set andthen enabled explicitly.	chromaImg.filters("chroma").color = theColor; chromaImg.filters("chroma").enabled = true;
mask	Allows to create an image mask, in which the background of an element is a solid color and the foreground of an element is transparent to the image or color behind it.	< style =”filter: mask(color = CCFFFF)">
invert	Applies a negative image effect—dark areas become light, and light areas become dark.	< style = "filter: invert">
Gray	The gray filter applies a grayscale image effect, in which all color is stripped from the image and all that remains is brightness data.	< style = "filter:gray">
Xray	The xray filter applies an x-ray effect, which basically is an inversion of the grayscale effect.	< style = "filter:xray">
Shadow	This filter creates a shadowing effect that gives the text a three-dimensional appearance. Property direction of the shadowfilter determines in which direction the shadow effect is applied and Property color specifies the color of the shadow that is applied to	<style = “filter: shadow(direction = 0, color = red)>

	the text. The direction is given in angles.	
Alpha	The alpha filter also is used for transparency effects not achievable with the chroma filter. The style parameter takes value of 0 for uniform opacity, 1 for linear gradient, 2 for circular gradient and 3 for rectangular gradient. The opacity and finishopacity properties are both percentages that determineat what percent opacity the specified gradient starts and finishes, respectively. Additional attributes are startX, startY, finishX and finishY specifies at what x-y coordinates the gradient starts and finishes in that element.	<style = “filter: alpha(style = 2, opacity = 100,finishopacity = 0)">
Glow	The glow filter adds an aura of color around text. The color and strength can both be specified as parameters.	< style = “filter: glow(color = red, strength = 5)”>
Blur	The blur filter creates an illusion of motion by blurring text or images in a certain direction and can be applied in any of eight directions, and its strength can vary. The add property, when set to true, adds a copy of the original image over the blurred image, creating a more subtle blurring effect.	<style = "filter: blur(add = 0, direction = 0, strength = 0)">
Wave	The wave filter allows you to apply sine-wave distortions to text and images on your WebPages. The add property, like the blur filter, adds a copy of the text or image underneath the filtered effect. The freq and phase property determines the frequency and phase shift of the wave	<style = " filter: wave(add = 0, freq = 1, phase = 0,strength = 10)”>

	applied respectively. Strength is the amplitude of the sine wave that is applied.	
dropShadow	The dropShadowfilte drops shadow we applied to images.	<style=" filter: dropShadow(offx = 0, offy = 0, color = black) ">
Light	The light filter simulates the effect of a light source shining on the page.	< style = "filter: light">

Filters

```
<html><head><title> filters</title></head><body>
<div style="position:absolute; top:125; left:2; filter:mask">
</div><imgsrc="cat.gif" width=400 height=200 style="filter:invert">
</body></html>
```

Transitions	Description	Syntax
blendTrans	This creates a smooth fade-in/fade-out effect. The duration determines how long the transition takes.	<style=": blendTrans(duration = 3)">
revealTrans	This allows the transition by using professional-style transitions, from box out to random dissolve. The transition paramerter is the index of the element as specified in the transition array. There are 24 tansitions.	<style=" filter: revealTrans(duration = 2, transition = 0)">

Transitions

```
<html><head><script language="text/javascript">
Function blendOut()
{ f.filters("blendTrans").apply();
f.style visibility="hidden";
f.filters("blendTrans").play();
}</script></head>
<body><div id="f" onClick="blendOut()" style="filter:blendTrans(duration=5)">
This is a nice effect</div>
</body></html>
```

Data binding with tabular control

Before the advent of DHTML, the data manipulations were done on the server thus increasing the server load and the network load. With DHTML, these manipulations can be done directly on the client without involving the server and the network.**Data binding** is a process that allows an Internet user to manipulate Web page elements using a Web browser. It employs dynamic HTML and does not require complex scripting or programming.With data binding, data need no longer reside exclusively on the server. The data can be maintained on the client. The data storage is well distinguished from the XHTML markup on the page. In DHTML, larger amount of data will be sent to the client on the first request. Changes done to the data the client side do not propagate back to the server. To bind external data to XHTML elements, Internet Explorer employs software that iscapable of connecting the browser to live data sources. These are known as **Data Source Objects(DSOs).**

Tabular Data Control

The **Tabular Data Control (TDC)** is an ActiveX control that is added to a page with the object element. Data are stored in a separate file and will not be a part of the XHTML document. TDC is one way of accessing data from DSO.

- The object element will have the following properties :
  - Classid-specifies the TDC to be added to the page
  - Param tag-specifies the parameters for the object in the form of 'name-value' pairs.
- The following are the parameters taken by TDC:
  - Data URL: URL of data source.
  - UseHeader: If this value is true, then the first line of the data file has header field.
  - TextQualifier: Qualifiers are characters that are placed at both the ends of a filed.
  - FieldDelim: Characters that act as separaters between different data fields.
- **Record set** is the set of data from the data source. The following are the methods to access the record set:

Methods	Description
Move Next()	Moves the recordset to the next row in the data source.
Move First()	Moves to the first recordset in the file.
MoveLast()	Moves to the last recordset in the file.
Move Previous()	Moves to the previous recordset in the file.

Tabular Data Control

```
<html><head><title>Dynamic Recordset Viewing</title>
<object id = "Colors" classid = "CLSID:333C7BC4-460F-11D0-BC04-0080C7055A83">
<param name = "DataURL" value ="HTMLStandardColors.txt" />
<param name = "UseHeader" value = "TRUE" />
<param name = "TextQualifier" value = "@" />
<param name = "FieldDelim" value = "|" /></object>
<script type = "text/javascript">
varrecordSet = Colors.recordset;
function update()
{ h1Title.style.color = colorRGB.innerText; } function
move( whereTo )
{ switch ( whereTo )
{
case "first": recordSet.MoveFirst(); update(); break; case
"previous":recordSet.MovePrevious();
if ( recordSet.BOF )
recordSet.MoveLast(); update(); break; case
"next": recordSet.MoveNext();
if ( recordSet.EOF ) recordSet.MoveFirst();
update(); break;
case "last": recordSet.MoveLast();update(); break;
} }</script>
<style type = "text/css">
input { background-color: khaki; color: green; font-weight: bold }
</style></head>
<body style = "background-color: darkkhaki">
<h1 style = "color: black" id = "h1Title"> XHTML Color Table</h1>
<span style = "position: absolute; left: 200; width: 270; border-style: groove; text-align: center;
background-color: cornsilk; padding: 10">
```

```
<strong>Color Name: </strong>
<span id = "colorName" style  = "font-family:  monospace"
datasrc = "#Colors" datafld = "ColorName">ABC</span><br />
<strong>Color RGB Value: </strong>
<span id = "colorRGB" style = "font-family: monospace"
datasrc ="#Colors" datafld ="ColorHexRGBValue">ABC
</span><br />
<input type = "button" value = "First"onclick = "move( 'first' );" />
<input type = "button" value = "Previous"onclick = "move( 'previous' );" />
<input type = "button" value = "Next" onclick = "move( 'next' );" />
<input type = "button" value = "Last" onclick = "move( 'last' );" />
</span></body></html>
```

Create a data sorce named HTMLStandardColors.txt that contains colors and its color values.

Binding to an image

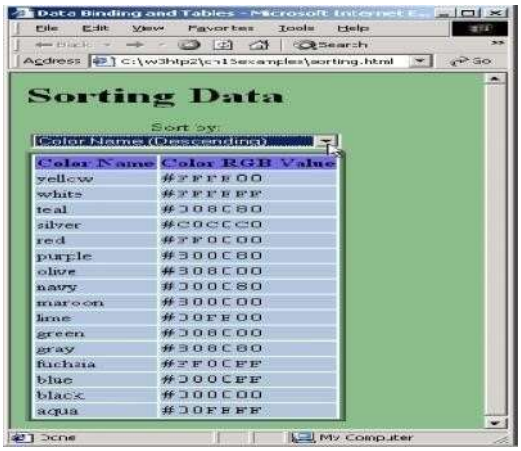
Many different types of XHTML elements for instance, image can be bound to data sources.Images are binded with data set by modifying the src attribute of the image.

<imgdatasrc=”cat.jpeg” datafld=”image”>

The previous example, changes the color based on the recordset. Now by changing the datasrc and datafld attribute as above will make the recordset to traverse through various images (create an image data source). Omit the update();

Binding to a table and sorting the table data: Tables could also be binded to the data source and sorted.

```
<html ><head><title>Data Binding and Tables</title>
<object  id="Colors"  classid="CLSID:333C7BC4-460F-11D0-BC04-0080C7055A83">
<param name = "DataURL" value ="HTMLStandardColors.txt" />
<param name = "UseHeader" value = "TRUE" />
<param name = "TextQualifier" value = "@" />
<param name = "FieldDelim" value = "|" />
</object></head>
<body style = "background-color: darkseagreen">
<h1>Sorting Data</h1>
<table datasrc="#Colors" style="border-style: ridge; border-color: darkseagreen;
background-color: lightcyan">
<caption> Sort by: <select onchange = "Colors.Sort = this.value; Colors.Reset();">
<option value = "ColorName">Color Name (Ascending) </option>
<option value = "-ColorName">Color Name (Descending) </option>
<option value = "ColorHexRGBValue">Color RGB Value (Ascending)</option>
<option value = "-ColorHexRGBValue">Color RGB Value (Descending)</option>
</select></caption>
<thead><tr style = "background-color: mediumslateblue">
<th>Color Name</th><th>Color RGB Value</th></tr></thead>
<tbody><tr style = "background-color: lightsteelblue">
<td><span datafld = "ColorName"></span></td>
<td><span datafld = "ColorHexRGBValue" style = "font-family:
monospace"></span></td>
</tr></tbody></table></body></html>
```



Specify the column by which to sort in the Sort property of the TDC. This example sets property Sort to the value of the selected option tag (this.value) when the onchange event is fired.By default, a column is sorted in ascending order. To sort in descending order, the column name is preceded with a minus sign (-).

Data binding elements

The following elements allow data binding:

Element	Bindable attribute
A	Href
Frame	Href
Iframe	Href
Img	Src
Div	Contained text
Input type=”button”	value
Input type=”chaeckbox”	Checked
Input type=”hidden”	value
Input type=”password”	Value
Input type=”radio”	Checked
Input type=”text”	Value
Marquee	Contained text

Param	Value
Select	select option
Span	Contained text
Table	Contained elememts
Textarea	Contained text

Structures graphics and ActiveX controls

The Structured Graphics Control is an ActiveX control that can be add to the page with an object element. It is easily accessible through scripting.

The Structured Graphics Control is a web interface that is used for visual presentations.The Structured Graphics control facilitates the creation of simple shapes by using functions that can be called via scripting or through param tags inside object elements.

The name attribute of the param tag method determines the order in which the function specified in the value attribute is called. The distortion of shapes like translation, rotation can also be done. To provide interaction with the user, the Structured Graphics Control can process the Dynamic HTML mouse events onmouseup, onmousedown, onmousemove, onmouseover, onmouseout, onclick and ondblclick. By default, the Structured Graphics Control does not capture mouse events, because doing so takes a small amount of processing power.The Structure Graphics Control allows you to keep a set of method calls in a separate source file and to invoke those methods by calling the SourceURL function.The following are the functions available for Structured Graphics Control:

Function	Description
SetLineColor(Rvalue, Gvalue, Bvalue)	sets the color of lines and borders of shapes that are drawn. It takes an RGB triplet in decimal notation as its three parameters.
SetLineStyle(line style, line width)	Draws a line. A value of 1 for line style creates a solid line (the default). A value of 0 does not draw any lines or borders, and a value of 2 creates a dashed line.
SetFillColor(color)	Sets the foreground color with which to fill shapes.
SetFillStyle(color style)	Sets the style in which a shape is filled with color; a value of 1 fills shapes with the solid color declared with the SetFillColor method.
Oval(x, y, height, width, direction)	Places the oval in specifies x-y location. The last parameter specifies the clockwise rotation of the oval relative to the x-axis, expressed in degrees.

Arc(x,y,height,width,starting angle, size, rotation).	Draws an arc with the given parameters.
Pie(x,y,height,width,starting angle, size, rotation)	It fills in the arc with the foreground color, thus creating a pie shape.
Polygon(no of vertices, (x,y) coordinates of the sides of the polygon)	Constructs a polygon. If the no of vertices of the polygon is 3, then 3 pairs of (x,y) co-ordinates must be specified.
Rect(x, y, height, weight, rotation)	Constructs a rectangle.
RoundRect(x, y, height, weight, rotation, height of rounded arc, width of rounded arc)	Constructs a rounded rectangle.
SetFont()	Sets the font style to use when placing text with the Text method.
PolyLine(no of points in the line, x, y of other vertex)	Draws a line with multiple segments
SetTextureFill(x, y, location of texture, value)	Fills a shape with a texture. A last parameter of 0 specifies that the texture should be stretched to fit inside the shape. A last parameterof 1 would instead tile the texture as many times as necessary inside the shape.
Translate(x, y, z)	Moves a shape in coordinate space without deforming it. Its three parametersdetermine the relative distance to move along the x-, y- and z-axes, respectively.
Rotate(x, y, z)	Rotates shapes in three-dimensional space. The three parameters of the Rotate function specify rotation in the x-, y- and z-coordinate planes, respectively.
MouseEventsEnabled()	Turn event capturing on

Path, Sequencer and Sprite ActiveX Controls

The DirectAnimation Path Control allows to control the positions of elements on the page.The Path Control, the Sequencer Control and the Sprite Control allows a Web page designer to add certain multimedia effects to Web pages.This mechanism is more advanced than dynamic CSS positioning, because it allows to define paths that the targeted elements follow. This capacity to define paths gives the ability to create professional presentations, especially when integrated with other Dynamic HTML features such as filters and transitions.

Setting AutoStart attribute of the object to a nonzero value starts the element along a path as soon as the page loads. Setting a zero value prevents it from starting automatically, in which case a script would have to call the Play method to start the path.

- The **Path Control** also allows setting paths for multiple objects present on your page. To set paths for multiple objects, add a separate object tag for each object.
- The z-index of elements that overlap is determined by their order of declaration in the XHTML source (elements declared later in the XHTML file are displayed above elements declared earlier).
- A useful feature of the Path Control is the ability to execute certain actions at any point along an object's path. This capability is implemented with the AddTimeMarker method, which creates a time marker that can be handled with simple JavaScript event handling.
- The **Sequencer Control** provides a simpler interface for calling functions or performing actions at time intervals.The **oninit** event fires when the Sequencer Control has loaded.
- The Item object of the Sequencer Control creates a grouping of events using a common name.The **Sprite Control** allows the displaying animated images composed of individual frames.
- The object tag inserts the Sprite Control. The height and width CSS properties are needed to display the image correctly; they should be equal to the size of one frame in the file. Setting attribute
- Repeat to a nonzero value loops the animation indefinitely. **NumFrames** specifies how many frames are present in the animation source image.
- **Attributes NumFramesAcross** and **Num-FramesDown** specify how many rows and columns of frames there are in the animation file, respectively.**SourceURL** gives a path to the file containing the frames of the animation.
- The animated GIFs are most popular animation formats and are composed frames in the GIF image format. GIF images must be inserted into animated GIF files by using graphics applications such as Adobe PhotoShop elements.

Method	Description
Repeat()	Determines how many times the path will be traversed; setting the value to -1 specifies that the path should loop continuously
Duration()	Specifies the amount of time that it takes to traverse the path, in seconds.
Bounce()	When set to 1, reverses the element's direction on the path when it reaches the end. Setting the value to 0 returns the element to the beginning of the path when the path has been traversed.

PolyLine()	Creates a path with multiple line segments.
Target()	Specifies the id of the element that is targeted by the Path Control. Setting the CSS attribute position to absolute allows the Path Control to move an element around the screen. Otherwise, the element would be static, locked in the position determined by the browser when the page loads.
AddTimeMarker()	The first parameter determines the point at which our time marker is placed along the path, specified in seconds; when this point is reached, the onmarker event is fired. The second parameter gives an identifyingname to the event, which is later passed on to the event handler for the onmarker event. The last parameter specifies whether to fire the onmarker event every time the object's path loops pastthe time marker (value=0) or to fire the event just the first time that the time marker is passed (value= 1).
At(waiting time, action)	This method takes two parameters: How many seconds to wait, and what action to perform when that period of time has expired.
Play()	Starts the targeted element along the path.
Sprite Control()	Controls the rate at which frames are displayed
MouseEventsEnabled()	Enables or disables mouse events.
Stop()	Stops the animation in place

Advantages of Activex Controls: Platform Control, Active X Scripting and Easy To Use and Easy to Find

Disadvantages of Activex Controls: Requires User to Download Something, System Vulnerabilities, Built-in Malware and Spyware and Only Compatible with Microsoft Programs

2.7 JAVASCRIPT OBJECT NOTATION (JSON)

JSON is a text-based data exchange format derived from JavaScript that is used in web services and other connected applications.

JSON Syntax

JSON defines only two data structures: objects and arrays. An object is a set of name-value pairs, and an array is a list of values. JSON defines seven value types: string, number, object, array, true, false, and null**.**

- Objects are enclosed in braces ({}), their name-value pairs are separated by a comma (,), and the name and value in a pair are separated by a colon (:). Names in an object are strings, whereas values may be of any of the seven value types, including another object or an array.

- Arrays are enclosed in brackets ([]), and their values are separated by a comma (,). Each value in an array may be of a different type, including another array or an object.When objects and arrays contain other objects or arrays, the data has a tree-like structure.

Uses of JSON

JSON is often used as a common format to serialize and deserialize data in applications that communicate with each other over the Internet. These applications are created using different programming languages and run in very different environments. JSON is suited to this scenario because it is an open standard, it is easy to read and write, and it is more compact than other representations.

JSON is built on two structures:

A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence

JSON Syntax

JSON syntax is considered as a subset of JavaScript syntax:

- Data is in name/value pairs: JSON data is written as name/value pairs. A name/value pair consists of a field name (in double quotes), followed by a colon.

  Example: “name”:”Cat”

- Data is separated by commas: **Example:** "first_name" : "Sun", "last_name" : "moon",

- Curly braces hold objects. Square brackets hold arrays. JSON keys are on the left side of the colon.They need to be wrapped in double quotation marks,

JSON Values:

In JSON, values must be one of the following data types:string, number, object (JSON object), array, Boolean and null

```
{ "firstName": "J", "lastName": "S", "age": 25, "children": [], "spouse": null,
 "address": {"street": "7504 TVS nagar", "city": "Tambaram","state": "Tamilnadu", "postalCode": "603203" },
 "phoneNumbers": [ {"type": "mobile", "number": "212 555-3346"},
```

```
{ "type": "fax", "number": "646 555-4567" } ]}
```

The first two name value pairs maps a string to another string. The third name value pair maps a string age with a number 25. The fourth pair maps a string children with an empty array []. The fifth pair maps a string spouse with null value. The sixth pair maps a string address with another JSON object. The seventh pair maps a string with array of JSON objects.

Function Files

There is no native support for defining functions in JSON. Commonly used approach is to define function as string and use eval() or new Function() to construct the function. The basic difference between these two are:

- eval() works within the current execution scope. It can access or modify local variables.
- new Function() runs in a separate scope. It cannot access or modify local variables.

HTTP Requests

JSON is most commonly used in asynchronous HTTP requests. This is where an application pulls data from another application via an HTTP request on the web.XMLHttpRequest is an API that provides scripted client functionality for transferring data between a client and a server. It enables to get data from an external URL without having to refresh the page. For example, a user could click a button that results in a small part of the page updating, rather than the whole page.

Artist.txt

```
{ "artists" : [
  { "artistname"  :  "Leonard Cohen", "born" : "1934"      },
  { "artistname" : "Joe Satriani", "born" : "1956" },
  { "artistname" : "Snoop Dogg","born" : "1971" } ]}
```

Below is a sample HTML page that retrieves that JSON data via HTTP, and uses JavaScript to wrap it in HTML tags and output it to the HTML document.

```
<!doctype html>
<title>Example</title><script>
// Store XMLHttpRequest and the JSON file location in variables
varxhr = new XMLHttpRequest();
varurl = "https://www.abc.com/json/Artists.txt";
```

```
// Called whenever the readyState attribute changes
xhr.onreadystatechange = function() {
  if (xhr.readyState == 4 &&xhr.status == 200) { // Check if fetch request is done
varjsonData = JSON.parse(xhr.responseText); // Parse the JSON string
    showArtists(jsonData);// Call the showArtists(), passing in the parsed JSON string
  }};
// Do the HTTP call using the url variable we specified above
xhr.open("GET", url, true);
xhr.send();
// Function that formats the string with HTML tags, then outputs the result
function showArtists(data) {
var output = "<ul>"; // Open  list
var i;    // Loop through the artists, and add them as list items
for (var i in data.artists)  {
      output += "<li>" + data.artists[i].artistname + " (Born: " + data.artists[i].born +
")</li>"; }
output += "</ul>"; // Close list. Output the data to the "artistlist" element
document.getElementById("artistList").innerHTML = output;
}</script><!-- The output appears here -->
<div id="artistList"></div>
```


JSON-SQL

JSON functions in SQL Server enable to analyze and query JSON data, transform JSON to relational format, and export SQL query results as JSON text.

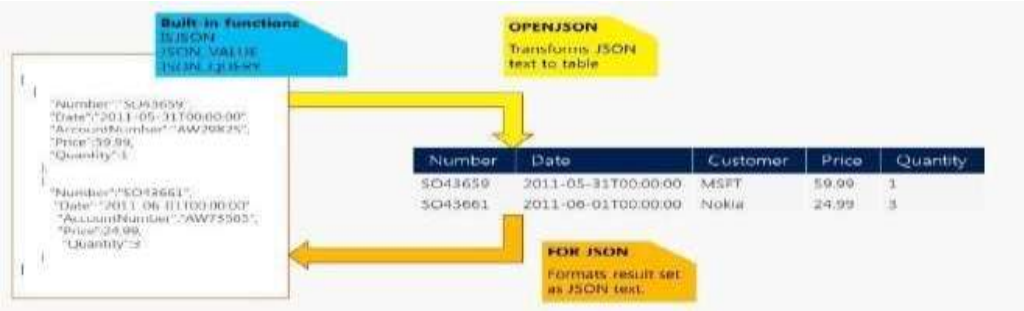


Fig 2.5: JSON with SQL

JSON text can be extracted from JSON or verify that JSON is properly formatted using built-in functions JSON_VALUE, JSON_QUERY, and ISJSON. For more advanced querying and analysis, the OPENJSON function can transform an array of JSON objects into a set of rows. Any SQL query can be executed on the returned result set. Finally, there is the FOR JSON clause that enables to format query results as JSON text.

Transact-SQL code, we will define a text variable to put JSON text:

```
DECLARE @json NVARCHAR(4000)
SET @json = N'{    “info”:{        “type”:1,
    “address”:{ “town”:”Bristol”,“county”:”Avon”, “country”:”England” },
    “tags”:[“Sport”, “Water polo”] },
  “type”:”Basic”}’
```

Extract values and objects from JSON text using the JSON_VALUE and JSON_QUERY functions:

```
SELECT
  JSON_VALUE(@json, ‘$.type’) as type,
  JSON_VALUE(@json, ‘$.info.address.town’) as town,
  JSON_QUERY(@json, ‘$.info.tags’) as tags
```


This query will return “Basic”, “Bristol”, and [“Sport”, “Water polo”] values. The JSON_VALUE function returns one scalar value from JSON text (e.g. strings, numbers, true/false) that is placed on a JSON path specified as the second parameter. JSON_QUERY returns an object or array on the JSON path. JSON built-in functions use JavaScript-like syntax to reference values and objects in JSON text via second parameter.The OPENJSON function enables to reference some array in JSON text and return elements from that array:

```
SELECT valueFROM OPENJSON(@json, ‘$.info.tags’)
```

The string values from the tags array are returned. However, the OPENJSON function can return any complex object.Finally, there is a FOR JSON clause that can format any result set returned by SQL query as JSON text:

```
SELECT object_id, nameFROM sys.tablesFOR JSON PATH
```

2.4 OBJECTS IN JAVASCRIPT

JavaScript is an Object Oriented Programming (OOP) language. A programming language is called object-oriented if it has the following four basic capabilities: encapsulation, aggregation, inheritance and polymorphism. All the objects will have properties and methods.

Object Properties

Object properties can be any of the primitive data types or abstract data types. Object properties are usually variables that are used internally in the object's methods, but can also be globally visible variables that can be used throughout the page.

objectName.objectProperty = propertyValue;

Example: varstr = document.title;

Object Methods

The **methods** are functions that let the object do something or let something be done to it. A function is a standalone unit of statements and a method is attached to an object and can be referenced by the keyword.Methods are useful for everything from displaying the contents of the object to the screen to performing complex mathematical operations on a group of local properties and parameters.

Example: document.write("This is a method of the object document");

User-Defined Objects

Apart from built-in objects, the users can also create their own objects. All user-defined objects and built-in objects are descendants of an object called **Object**. The new operator is used to create a new object.

The new Operator

The new operator is used to create an instance of an object. To create an object, the new operator is followed by the constructor method.

Example:var books = new Array("Thirukural", " Geethai");

In the above example Array() is a built-in object. Books is the instance of the object Array().

The Object() Constructor

A **constructor** is a function that creates and initializes an object. The Object() constructor is used to build an object. The return value of the Object() constructor is assigned to a variable.The variable contains a reference to the new object. The properties assigned to the object are not variables. These properties can be accessed only through objects.

Objectname.property

```
<html><head><script type="text/javascript">
var book = new Object(); // Create the object
 book.name = "PonniyinSelvan"; // Assign properties to the
        object
book.author = "Kalki";
</script></head>
<body><script    type="text/javascript">
document.write("Book name is : " + book.name + "<br>");
document.write("Book author is : " + book.author +  "<br>");
</script></body></html>
```

Book name is : Ponniyin Selvan
Book author is : Kalki

JavaScript Native Objects

JavaScript has several built-in or native objects. These objects are accessible anywhere in the program as the other objects. The following are some of the important JavaScript Native Objects:JavaScript Number Object, JavaScript Boolean Object, JavaScript String Object, JavaScript Array Object, JavaScript Date Object, JavaScript Math Object, JavaScript RegExp Object

2.4.1 JavaScript Number Object

The Number object represents numerical date, either integers or floating-point numbers. The browser automatically converts number literals to instances of the number class.

varval = new Number(number);

If the argument cannot be converted into a number, it returns NaN (Not-a-Number).

Number Properties

Creating an object

Property	Description
MAX_VALUE	The largest possible value a number in JavaScript (1.7976931348623157E+308).
MIN_VALUE	The smallest possible value a number in JavaScript can have (5E-324)
NaN	Equal to a value that is not a number.
NEGATIVE_INFINITY	A value that is less than MIN_VALUE.

POSITIVE_INFINITY	A value that is greater than MAX_VALUE
Prototype	A static property of the Number object. This is used to assign new properties and methods to the Number object in the current document.

Number Methods

Method	Description
Number()	Returns the number object.
toExponential()	Forces a number to display in exponential notation.
toFixed()	Formats a number with a specific number of digits to the right of the decimal.
toLocaleString()	Returns a string value version of the current number in a format that may vary according to a browser's locale settings.
toPrecision()	Defines how many total digits to display of a number (including digits to the before and after the decimal).
toString()	Returns the string representation of the number's value.
valueOf()	Returns the number's value.

2.4.2 JavaScript String object

varval = new String(string); // The string parameter is series of characters.

String Properties

- length -Returns the length of the string.
- Prototype-The prototype property allows you to add properties and methods to an object.

String Methods

Method	Description
String()	Returns the string object.
charAt(i)	Returns the character at the specified index.

charCodeAt(index)	Returns a number indicating the Unicode value of the character at the given index.
concat()	Combines the text of two strings and returns the combined string.
indexOf()	Returns the index of the first occurrence of the String object of or -1 if not found.
lastIndexOf()	Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.
localeCompare()	Returns a number indicating whether a reference string comes before or after or is the same as the given string in sort order.
match()	Used to match a regular expression against a string.
replace()	Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring
search()	Executes the search for a match between a regular expression and a specified string.
slice()	Extracts a section of a string and returns a new string.
split()	Splits a String object into an array of strings by separating the string into substrings.
substr()	Returns the characters in a string beginning at the specified location through the specified number of characters.
substring()	Returns the characters in a string between two indexes into the string.
toLocaleLowerCase()	The characters within a string are converted to lower case while respecting the current locale.
toLocaleUpperCase()	The characters within a string are converted to upper case while respecting the current locale.
toLowerCase()	Returns the calling string value converted to lower case.
toString()	Returns a string representing the specified object.
toUpperCase()	Returns the calling string value converted to uppercase.
valueOf()	Returns the primitive value of the specified object.

String HTML wrappers

The functionalities of these wrappers are similar to the HTML tags.

Method	Description
anchor()	Creates an HTML anchor that is used as a hypertext target.
big()	Creates a string to be displayed in a big font as if it were in a <big> tag.
blink()	Creates a string to blink as if it were in a <blink> tag.
bold()	Creates a string to be displayed as bold as if it were in a <b> tag.
fixed()	Causes a string to be displayed in fixed-pitch font as if it were in a <tt> tag.
fontcolor()	Causes a string to be displayed in the specified color as if it were in a <font color="color"> tag.
fontsize()	Causes a string to be displayed in the specified font size as if it were in a <font size="size"> tag.
italics()	Causes a string to be italic, as if it were in an <i> tag.
link()	Creates an HTML hypertext link that requests another URL.
small()	Causes a string to be displayed in a small font, as if it were in a <small> tag.
strike()	Causes a string to be displayed as struck-out text, as if it were in a <strike> tag.
sub()	Causes a string to be displayed as a subscript, as if it were in a <sub> tag.
sup()	Causes a string to be displayed as a superscript, as if it were in a <sup> tag.

JavaScript Array object

The Array object is used store multiple values in a single variable.

Example:var fruits = new Array("apple", "orange", "mango");

The Array parameter is a list of strings or integers. The maximum length allowed for an array is 4,294,967,295.

Array Properties

- index - The property represents the zero-based index of the match in the string

- input - This property is only present in arrays created by regular expression matches.
- length - Reflects the number of elements in an array.
- Prototype - The prototype property allows you to add properties and methods to an object.

Array Methods

Method	Description
Array()	Returns a reference to the array function that created the object.
concat()	Returns a new array comprised of this array joined with other array(s) and/or value(s).
every()	Returns true if every element in this array satisfies the provided testing function.
filter()	Creates a new array with all of the elements of this array for which the provided filtering function returns true.
forEach()	Calls a function for each element in the array.
indexOf()	Returns the first (least) index of an element within the array equal to the specified value, or
join()	Joins all elements of an array into a string.
lastIndexOf()	Returns the last (greatest) index of an element within the array equal to the specified value
map()	Creates a new array with the results of calling a provided function on every element in this array.
pop()	Removes the last element from an array and returns that element.
push()	Adds one or more elements to the end of an array and returns the new length of the array.
reduce()	Apply a function simultaneously against two values of the array (from left
reduceRight()	Apply a function simultaneously against two values of the array (from right
reverse()	Reverses the order of the elements of an array

shift()	Removes the first element from an array and returns that element.
slice()	Extracts a section of an array and returns a new array.
some()	Returns true if at least one element in this array satisfies the provided testing function.
toSource()	Represents the source code of an object
sort()	Sorts the elements of an array.
splice()	Adds and/or removes elements from an array.
toString()	Returns a string representing the array and its elements.
unshift()	Adds one or more elements to the front of an array and returns the new length of the array.

JavaScript Math Object

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

```
varpi_val = Math.PI; //Math object need not be created
varsine_val = Math.sin(30);
```


Math Properties

Property	Description
E	Euler's constant and the base of natural logarithms, approximately 2.718.
LN2	Natural logarithm of 2, approximately 0.693.
LN10	Natural logarithm of 10, approximately 2.302.
LOG2E	Base 2 logarithm of E, approximately 1.442.
LOG10E	Base 10 logarithm of E, approximately 0.434.
PI	Ratio of the circumference of a circle to its diameter, approximately 3.14159.
SQRT1_2	Square root of 1/2; equivalently, 1 over the square root of 2, approximately 0.707.
SQRT2	Square root of 2, approximately 1.414.

Math Methods

Method	Description
abs()	Returns the absolute value of a number.
acos()	Returns the arccosine (in radians) of a number.
asin()	Returns the arcsine (in radians) of a number.
atan()	Returns the arctangent (in radians) of a number.
atan2()	Returns the arctangent of the quotient of its arguments.
ceil()	Returns the smallest integer greater than or equal to a number.
cos()	Returns the cosine of a number.
exp()	Returns EN, where N is the argument, and E is Euler's constant,the base of the natural logarithm.
floor()	Returns the largest integer less than or equal to a number.
log()	Returns the natural logarithm (base E) of a number.
max()	Returns the largest of zero or more numbers.
abs()	Returns the absolute value of a number.
min()	Returns the smallest of zero or more numbers.
pow()	Returns base to the exponent power, that is, base exponent.
random()	Returns a pseudo
round()	Returns the value of a number rounded to the nearest integer.
sin()	Returns the sine of a number.
sqrt()	Returns the square root of a number.
tan()	Returns the tangent of a number.
toSource()	Returns the string "Math".

Regular expressions and regular objects

A regular expression is an object that describes a pattern of characters. The JavaScript RegExp class represents regular expressions, and both String and RegExp define methods that

use regular expressions to perform powerful pattern-matching and search-and-replace functions on text.

```
var pattern = new RegExp(pattern, attributes);

var pattern = /pattern/attributes;
```

- **pattern**: A string that specifies the pattern of the regular expression or another regular expression.
- **attributes:** An optional string containing any of the "g", "i", and "m" attributes that specify global, case-insensitive, and multiline matches, respectively.

Brackets:

Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

- [...] - Any one character between the brackets.
- [^...] - Any one character not between the brackets.
- [0-9] - It matches any decimal digit from 0 through 9.
- [a-z] - It matches any character from lowercase a through lowercase z.
- [A-Z] - It matches any character from uppercase A through uppercase Z.
- [a-Z] - It matches any character from lowercase a through uppercase Z.

Quantifiers:

The frequency or position of bracketed character sequences and single characters can be denoted by a special character. Each special character have a specific connotation. The +, *, ?, and $ flags all follow a character sequence.

- p+ -It matches any string containing at least one p.
- p*-It matches any string containing zero or more p's.
- p?-It matches any string containing one or more p's.
- p{N} -It matches any string containing a sequence of N p's
- p{2,3}-It matches any string containing a sequence of two or three p's.
- p{2, } -It matches any string containing a sequence of at least two p's.
- p$-It matches any string with p at the end of it.
- ^p-It matches any string with p at the beginning of it.

RegExp Properties

Property	Description
Global	Specifies if the "g" modifier is set.
ignoreCase	Specifies if the "i" modifier is set.
lastIndex	The index at which to start the next match.
Multiline	Specifies if the "m" modifier is set.
Source	The text of the pattern.
Global	Specifies if the "g" modifier is set.

RegExp Methods

Method	Description
exec()	Executes a search for a match in its string parameter.
test()	Tests for a match in its string parameter.
toSource()	Returns an object literal representing the specified object; you can use this value to create a new object.
toString()	Returns a string representing the specified object.

JavaScript Date Object

The Date object is a data type built into the JavaScript language. Once a Date object is created, a number of methods are available to operate onit.

- new Date( ) - This constructor creates a Date object set to the current date and time.
- new Date(milliseconds)- When one numeric argument is passed, it is taken as the internal numeric representation of the date in milliseconds, as returned bythe getTime( ) method. For example, passing the argument 5000 creates a date that represents five seconds past midnight on 1/1/70
- new Date(datestring) - When one string argument is passed, it is a string representation of a date, in the format accepted by the Date.parse( ) method.
- new Date(year,month,date[,hour,minute,second,millisecond]) -The parameters on square brackets are optional . The description of the arguments are listed below:

1. year: Integer value representing the year. For compatibility (in order to avoid the Y2K problem), you should always specify the year in full; use 1998, rather than 98.
2. month: Integer value representing the month, beginning with 0 for January to 11 for December.
3. date: Integer value representing the day of the month.
4. hour: Integer value representing the hour of the day (24-hour scale).
5. minute: Integer value representing the minute segment of a time reading.
6. second: Integer value representing the second segment of a time reading.
7. millisecond: Integer value representing the millisecond segment of a time reading.

Date Properties:

- constructor- Specifies the function that creates an object's prototype.
- Prototype- The prototype property allows you to add properties and methods to an object

Date Methods:

Method	Description
Date()	Returns today's date and time
getDate()	Returns the day of the month for the specified date according to local time.
getDay()	Returns the day of the week for the specified date according to local time.
getFullYear()	Returns the year of the specified date according to local time.
getHours()	Returns the hour in the specified date according to local time.
etMilliseconds()	Returns the milliseconds in the specified date according to local time.
getMinutes()	Returns the minutes in the specified date according to local time.
getMonth()	Returns the month in the specified date according to local time.

getSeconds()	Returns the seconds in the specified date according to local time.
getTime()	Returns the numeric value of the specified date as the number of milliseconds since January 1, 1970, 00:00:00 UTC.
Date()	Returns today's date and time
getDate()	Returns the day of the month for the specified date according to local time.
getDay()	Returns the day of the week for the specified date according to local time.
getFullYear()	Returns the year of the specified date according to local time.
getHours()	Returns the hour in the specified date according to local time.
getTimezoneOffset()	Returns the time
getUTCDate()	Returns the day (date) of the month in the specified date according to universal time.
getUTCDay()	Returns the day of the week in the specified date according to universal time.
getUTCFullYear()	Returns the year in the specified date according to universal time.
getUTCHours()	Returns the hours in the specified date according to universal time.
getUTCMilliseconds()	Returns the milliseconds in the specified date according to universal time.
getUTCMinutes()	Returns the minutes in the specified date according to universal time.
getUTCMonth()	Returns the month in the specified date according to universal time.
getUTCSeconds()	Returns the seconds in the specified date according to universal time.
getYear() Deprecated	Returns the year in the specified date according to local time.
getTimezoneOffset()	Returns the time

getUTCDate()	Returns the day (date) of the month in the specified date according to universal time.
getUTCDay()	Returns the day of the week in the specified date according to universal time.
getUTCFullYear()	Returns the year in the specified date according to universal time.
setDate()	Sets the day of the month for a specified date according to local time.
setFullYear()	Sets the full year for a specified date according to local time.
setHours()	Sets the hours for a specified date according to local time.
setMilliseconds()	Sets the milliseconds for a specified date according to local time.
setMinutes()	Sets the minutes for a specified date according to local time.
setMonth()	Sets the month for a specified date according to local time.
setSeconds()	Sets the seconds for a specified date according to local time.
setTime()	Sets the Date object to the time represented by a number of milliseconds since January 1, 1970, 00:00:00 UTC.
setDate()	Sets the day of the month for a specified date according to local time.
setFullYear()	Sets the full year for a specified date according to local time.
setHours()	Sets the hours for a specified date according to local time.
setMilliseconds()	Sets the milliseconds for a specified date according to local time.
setUTCDate()	Sets the day of the month for a specified date according to universal time. setUTCFullYear()
toLocaleDateString()	Returns the "date" portion of the Date as a string, using the current locale's conventions.
toLocaleFormat()	Converts a date to a string, using a format string.
toLocaleString()	Converts a date to a string, using the current locale's conventions.

toLocaleTimeString()	Returns the "time" portion of the Date as a string, using the current locale's conventions.
toSource()	Returns a string representing the source for an equivalent Date object; you can use this value to create a new object.
toString()	Returns a string representing the specified Date object.
toTimeString()	Returns the "time" portion of the Date as a human
toUTCString()	Converts a date to a string, using the universal time convention.
valueOf()	Returns the primitive value of a Date object.
setUTCDate()	Sets the day of the month for a specified date according to universal time. setUTCFullYear()
toLocaleDateString()	Returns the "date" portion of the Date as a string, using the current locale's conventions.
toLocaleFormat()	Converts a date to a string, using a format string.
toLocaleString()	Converts a date to a string, using the current locale's conventions.
toLocaleTimeString()	Returns the "time" portion of the Date as a string, using the current locale's conventions.
toSource()	Returns a string representing the source for an equivalent Date object; you can use this value to create a new object.

Date objects

```
<html><body><script type="text/javascript">
var d = new Date()
document.write(d.getDate())document.write(".")
document.write(d.getMonth() + 1)document.write(".")
document.write(d.getFullYear())d.setFullYear("1990")document.write(".")
document.write(d.getUTCHours())document.write(".")
document.write(d.getUTCMinutes() + 1)document.write(".")
document.write(d.getUTCSeconds())
varweekday=new
```

```
Array("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
document.write("Today is " + weekday[d.getDay()])
varweekday=new
Array("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
varmonthname=new
Array("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec")
document.write(weekday[d.getDay()] + " ")
document.write(d.getDate() + ". ")
document.write(monthname[d.getMonth()] +" ")
document.write(d.getFullYear())
</body></html>
```

22.12.2014.10.9.11Today is SaturdaySaturday 22. Dec 1990

2.5 VALIDATION and EVENT HANDLING IN JAVASCRIPT

VALIDATION IN JAVASCRIPT

Form validation occurs at the server, after the client had entered all necessary data and then pressed the Submit button. If some of the data that had been entered by the client had been in the wrong form or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information. This is a lengthy process and over burdening server.JavaScript, provides a way to validate form's data on the client's computer before sending it to the web server.Form validation generally performs two functions:

Basic Validation - Checking the form to make sure that the data entered is right.

Data Format Validation – Checking the data entered in the form for right value.

Form validation

```
<html><head>
<title>Form  Validation</title>
<script  type="text/javascript">
</script></head><body>
<form action="/cgi-bin/test.cgi" name="myForm" onsubmit="return(validate());">
<table cellspacing="2" cellpadding="2" border="1">
<tr><td align="right">Name</td>
<td><input type="text" name="Name" /></td></tr>
```

```
<tr><td align="right">EMail</td>
<td><input type="text" name="EMail" /></td></tr>
<tr><td align="right">Zip Code</td>
<td><input type="text" name="Zip" /></td></tr>
<tr><td align="right">Country</td><td>
<select name="Country">
<option value="-1" selected>[choose yours]</option>
<option value="1">USA</option>
<option value="2">UK</option>
<option value="3">INDIA</option>
</select></td></tr>
<tr><td  align="right"></td>
<td><input type="submit" value="Submit" /></td>
</tr></table></form>
<script type="text/javascript">
function validate()
{    if( document.myForm.Name.value == "" )
{alert( "Please provide your name!" );
document.myForm.Name.focus() ;     return false; }
  if( document.myForm.EMail.value == "" )
  {alert( "Please provide your Email!" );document.myForm.EMail.focus() ;
   return false; }
  if( document.myForm.Zip.value == "" ||isNaN( document.myForm.Zip.value ) ||
document.myForm.Zip.value.length != 5 )
{alert( "Please provide a zip in the format #####." );
document.myForm.Zip.focus() ;      return false; }
  if(  document.myForm.Country.value  ==  "-1" )
  {alert( "Please provide your  country!" );      return false; }
  return( true );
}</script></body></html>
```

The validate() in the above example checks whether the user has entered all the fields of the form. In the above case, e-mail field is left blank and when submit button is clicked, the error message is displayed.

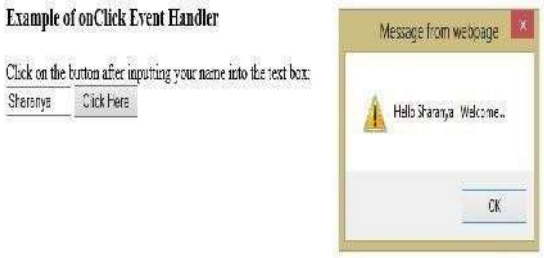
EVENT HANDLING IN JAVASCRIPT

JavaScript's interacts with HTML is through events that occur when the user or browser manipulates a page. When the page loads, that is an event. When the user clicks a button, that click, too, is an event. Events are a part of the Document Object Model (DOM) Level 3 and every HTML element have a certain set of events which triggers the JavaScript Code.

onclick Event Type:

This is the most frequently used event type which occurs when a user clicks mouse left button.


```
<HTML><TITLE>Example of onClick Event Handler</TITLE>
<HEAD><SCRIPT LANGUAGE="JavaScript">
function valid(form){var input=0;
  input=document.myform.data.value;
alert("Hello " + input + " ! Welcome...");
}</SCRIPT></HEAD>
<BODY>
<H3> Example of onClick Event Handler </H3>
Click on the button after inputting your name into the text box:<br>
<form name="myform">
<input type="text" name="data" value=""
```



size=10> *<INPUT TYPE="button" VALUE="Click Here" onClick="valid(this.form)">* *</form></BODY></HTML>*	

Standard Events

Event	Description
onchange	Script runs when the element changes
onsubmit	Script runs when the form is submitted
onreset	Script runs when the form is reset
onselect	Script runs when the element is selected
onblur	Script runs when the element loses focus
onfocus	Script runs when the element gets focus
onkeydown	Script runs when key is pressed
onkeypress	Script runs when key is pressed and released
onclick	Script runs when a mouse click
ondblclick	Script runs when a mouse double
onmousedown	Script runs when mouse button is pressed
onmousemove	Script runs when mouse pointer moves
onmouseout	Script runs when mouse pointer moves out of an element
onmouseover	Script runs when mouse pointer moves over an element
onmouseup	Script runs when mouse button is released
onclick	Script runs when a mouse click
Ondblclick	Script runs when a mouse double

Advantages of JavaScript

- **Speed:** Being client-side scripting, JavaScript is very fast because any code functions can be run immediately instead of having to contact the server and wait for an answer.
- **Simplicity:** JavaScript is relatively simple to learn and implement.
- **Versatility:** JavaScript plays nicely with other languages and can be used in a huge variety of applications. JavaScript can also be used inside scripts written in other languages such as Perl and PHP.
- **Server Load:** Being client-side reduces the demand on the website server.

Disadvantages of JavaScript

- **Security:** Because the code executes on the users' computer, in some cases it can be exploited for malicious purposes. This is one reason some people choose to disable JavaScript.
- **Reliance on End User:** JavaScript is sometimes interpreted differently by different browsers. Whereas server-side scripts will always produce the same output, client-side scripts can be a little unpredictable.