**Smt. Chandaben Mohanbhai Patel Institute of Computer Applications,**

**Charusat - Changa**

**BCA – III**

**CA213 – Programming the Internet - II (PI - II)**

**Unit IV: Java Script Programming ‐ I**

* **Scripting Language**

A scripting language is a programming language that employs a high- level construct to interpret and execute one command at a time. In general, scripting languages are easier to learn and faster to code. it is structured and compiled languages such as C and C++.

* **Difference between client side script and server side script.**

|  |  |
| --- | --- |
| **Client side Script** | **Server side script** |
| The Web Browser executes the client side scripting that resides at the user’s computer. | The Web Server executes the server side scripting that produces the page to be sent to  the browser. |
| Response from a client-side script is faster because the scripts are processed on the local computer. | Response from a server-side script is slower because the scripts are processed remotely. |
| Client side scripting cannot  be used to connect to the databases on the web server. | Server side scripting is used to  connect to the databases that reside on the web server. |
| Client side scripting can’t access the file system that resides at the web server. | Server side scripting can access the file system residing at the web server. |
| The files and settings that are local at the user’s computer can be accessed using Client  side scripting. | The settings that belong to Web server can be accessed using Server side scripting. |
| Client side scripting is possible to be blocked by the user. | Server side scripting can’t be blocked by the user. |
| User can view source code from the web browser. | User can not view the source code from the web browser. |
| Examples of Client side scripting languages: Javascript, VB script, etc. | Examples of Server side scripting languages : PHP, JSP, ASP, ASP.Net, Ruby, Perl and many more. |

* **JavaScript History**
  + JavaScript was first known as LiveScript, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java.
  + JavaScript made its first appearance in Netscape 2.0 in 1995 with the name LiveScript.
  + The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.
  + Sometimes JavaScript is referred to as ECMAscript, ECMA is European Computer Manufacturers Association, it is a private organization that develops standards in information and communication systems.
* **What is JavaScript?**
  + JavaScript is a high-level programming language that is interpreted by another program at runtime rather than compiled by the computer's processor as other programming languages (such as C and C++) are.
  + JavaScript is a lightweight, interpreted programming language.
  + Complementary to and integrated with Java.
  + Complementary to and integrated with HTML.
  + Open and cross-platform

## JavaScript is Case Sensitive

Unlike HTML, JavaScript is case sensitive - therefore watch your capitalization closely when you write JavaScript statements, create or call variables, objects and functions.

* **Advantages of JavaScript**
  + Less server interaction − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
  + Immediate feedback to the visitors − They don't have to wait for a page reload to see if they have forgotten to enter something.
  + Increased interactivity − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
  + Richer interfaces − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.
* **Limitations of JavaScript**
  + We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features –
    - Client-side JavaScript does not allow the reading or writing of files.
    - This has been kept for security reason.
    - JavaScript cannot be used for networking applications because there is no such support available.
    - JavaScript doesn't have any multithreading or multiprocessor capabilities.
* **JavaScript Development Tools:**
  + An interpreter in the browser reads over the JavaScript code, interprets each line, and runs it. More modern browsers use a technology known as Just-In-Time (JIT) compilation, which compiles JavaScript to executable bytecode just as it is about to run.
  + One of major strengths of JavaScript is that it does not require
  + Expensive development tools.
  + You can start with a simple text editor such as Notepad.
* **Embedding JavaScript into HTML document**

The easiest way to test a JavaScript program is by putting it inside an HTML page and loading it in a JavaScript enabled browser.

* + You can integrate javascript code into the HTML file in three ways:
    - Integrating under <head> tag.
    - Integrating under <body> tag
    - Importing the external javascript.
* **Integrating script under the <head> tag:**
  + Javascript in the head section will execute when called i.e javascript in the HTML file will execute immediately while the web page loads into the web browser before anyone uses it.
  + Syntax:

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

--------------

</script>

</head>

<body>

</body>

</html>

* **Integrating script under the <body> tag** 
  + When you place the JavaScript code under the <body> tag, this generates the content of the web page. JavaScript code executes when the web page loads and so in the body section.
  + Syntax:

<!DOCTYPE html>

<html>

<head>

</head>

<body>

<script type="text/javascript">

--------------

</script>

</body>

</html>

You can have any number of scripts:

In the HEAD, scripts are run before the page is displayed

In the BODY, scripts are run as the page is displayed

In the HEAD is the right place to define functions and variables that are used by scripts within the BODY.

* **Importing the External JavaScript**
  + You can import an external javascript file when you want to run the same javascript file on several HTML files without having to write the same javascript code on every HTML file.
  + Save the external javascript file with an extension .js
  + The external javascript file don’t have a <script> tag.
  + Syntax:

<!DOCTYPE html>

<html>

<head>

<style type="text/css" src="first.js" >

</style>

</head>

<body>

</body>

</html>

* **JavaScript Statements**
  + A JavaScript statement is a command to a browser. The purpose of the command is to tell the browser what to do.
  + Example: document.write("Hello");
* **JavaScript Code**
  + JavaScript code (or just JavaScript) is a sequence of JavaScript statements.
  + Each statement is executed by the browser in the sequence they are written.
  + This example will write a heading and two paragraphs to a web page:
  + Example:

<script type="text/javascript">  
document.write("<h1>This is a heading</h1>");  
document.write("<p>This is a paragraph.</p>");  
document.write("<p>This is another paragraph.</p>");  
</script>

* **JavaScript Comments**

Comments can be added to explain the JavaScript, or to make the code more readable. Single line comments start with //. The following example uses single line comments to explain the code:

## Example

<script type="text/javascript">  
// Write a heading  
document.write("<h1>This is a heading</h1>");  
// Write two paragraphs:  
document.write("<p>This is a paragraph.</p>");  
document.write("<p>This is another paragraph.</p>");  
</script>

* **JavaScript Multi-Line Comments**

Multi line comments start with /\* and end with \*/.

The following example uses a multi line comment to explain the code:

## Example

<script type="text/javascript">  
/\*  
The code below will write  
one heading and two paragraphs  
\*/  
document.write("<h1>This is a heading</h1>");  
document.write("<p>This is a paragraph.</p>");  
document.write("<p>This is another paragraph.</p>");  
</script>

* **JavaScript Variables**
  + Variables are declared with the var keyword.
  + JavaScript variables can be used to hold values (x=5) or expressions (z=x+y).
  + Variable can have short names (like x and y) or more descriptive names (age, sum, totalvolume).
  + Variable names must begin with a letter.
  + Variable names can also begin with $ and \_
  + Variable names are case sensitive (y and Y are different variables)
  + Before you use a variable in a JavaScript program, you must declare it.

<script type="text/javascript">

var name;

var age;

</script>

* + You can also declare multiple variables with the same var keyword as follows −

<script type="text/javascript">

var name, age;

</script>

* + Storing a value in a variable is called variable initialization. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

<script type="text/javascript">

var name = “peter";

var age;

age = 20;

</script>

* **JavaScript is untyped language**
  + This means that a JavaScript variable can hold a value of any data type.
  + Unlike many other languages, you don't have to tell JavaScript during variable declaration what type of value the variable will hold.
  + The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

**Data Types in JavaScript**

* + Primitive data types – such as Number, Strings, Boolean
  + Trivial data types – such as Null and Undefined
  + Composite data types – such as Object and Array

JavaScript allows you to work with three **primitive data types** –

* + Numbers, eg. 123, 120.50 etc.

o JavaScript has only one type of numbers. Numbers can be written with, or without decimals:

Example

var a=34.00; // Written with decimals var x2=34;

// Written without decimals

* + Strings of text e.g. "This text string" etc.
* A string is a variable which stores a series of characters like "Peter".
* A string can be any text inside quotes. You can use single or double quotes:

Example

var carname="Volvo C60"; var carname='Volvo XC60';

* + - You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

Example

var answer="It's alright";

var answer="He is called 'Johnny'”;

var answer='He is called Johnny"';

* + Boolean e.g. true or false.
  + Booleans can only have two values: true or false. Example

var x=true; var y=false;

* + Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

JavaScript also defines two trivial data types: null and undefined, each of which defines only a single value.

* + Null
* has only one value in JavaScript: null.
* A variable that contains null contains no valid Number, String, Boolean, Array, or Object. You can erase the contents of a variable (without deleting the variable) by assigning it the null value.

Example

var i;

i=null;

document.write(typeof(i)); // returns object

* + Undefined
  + Value is returned when you use an object property that does not exist, or a variable that has been declared, but has never had a value assigned to it.
  + you can check to see if a variable exists by comparing it to undefined although you can check if its type is undefined by comparing the type of the variable to the string "undefined".
  + The following example shows how to find out the variable has been declared:

Example

var x; document.write(typeof(x)); //returns undefined

if (x == undefined) {

document.write("comparing x to undefined <br/>");

}

if (typeof(x) == "undefined")

{

document.write("comparing the type of x to the string 'undefined'");

}

In addition to these primitive data types, JavaScript supports a composite data type known as object and Array.

* + Object:
    - JavaScript objects are written with curly braces.
    - Object properties are written as name:value pairs, separated by commas.
    - Example

var person={firstname:"John", lastname:"Doe", id:5566};

* The object (person) in the example above has 3 properties: firstname, lastname, and id.
* Spaces and line breaks are not important. Your declaration can span multiple lines:

var person = { firstName : "John", lastName : "Doe", age : 50, eyeColor : "blue"

};

You can address the object properties in two ways:

Example

name=person.lastname;

name=person["lastname"];

* + Array:
    - JavaScript arrays are written with square brackets.
    - Array items are separated by commas.
    - The following code declares (creates) an array called Students, containing three students

Example:

var Student= ["S1", "s2", “s3”];

Accessing Array value through index of an array:

document.write(“Array Value is “ + Student[1] + “<br>”);

Two dimension array:

Example var student = [];

student[0] = ["s1","s2","s3"];

student [1] =["MCA","BCA","MCAL"];

student [2] = [20,23,33];

document.write("<table border=1>"); for(i=0;i<student.length;i++)

{

document.write("<tr>"); for(j=0;j<student[0].length;j++)

{

document.write("<td>" + s[i][j]+ "</td>");

}

document.write("</tr>");

}

* **typeof operator**
* typeof operator is used to find the type of a JavaScript variable.
* Possible values that typeof return: number, "string," "boolean," "object," "function," and "undefined."

Example:

typeof 0 // Returns "number"

typeof "John" // Returns "string"

typeof 314 // Returns "number"

typeof 3.14 // Returns "number"

var a=”20”

document.write("Dataype of variable is " + typeof(b)); // Returns string

* **Types of Operators**

Let us take a simple expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and ‘+’ is called the operator. JavaScript supports the following types of operators.

|  |  |  |
| --- | --- | --- |
| Arithmetic Operators | + (Addition)  - (Subtraction)  \* (Multiplication) | Example:  var a = 33; var b = 10;  document.write("a + b = ");  result = a + b; |

|  |  |  |
| --- | --- | --- |
|  | / (Division)  % (Modulus)  ++ (increment)  -- (Decrement) | document.write(result);  Output:  a + b = 43 |
| Comparison Operators | == (Equal)  !=(Not Equal)  === (Equal: it matches variables value as well as its type)  !== (Not Equal to)  > (Greater Than)  < (Less Than)  >= (Greater Than Equal to)  <= (Less than Equal to) | Example:  var a = 33; var b = 10;  document.write("a > b = ");  result = a > b; document.write(result);  Output: |
| a > b = false |
|  |
| Logical Operator | && (Logical AND)  Ex: (A && B) is true.  || (Logical OR)  Ex: (A || B) is true.  ! (Logical NOT)  Ex: ! (A && B) is false. | Example:  var a = true; var b = false;  document.write("!(a && b) => ");  result = (!(a && b)); document.write(result);  Output: |
| !(a && b) => true |
|  |
| Assignment Operator | = (Simple Assignment )  += (Add and Assignment)  −= (Subtract and Assignment)  \*= (Multiply and Assignment)  /= (Divide and Assignment)  %= (Modules and Assignment) | Example:  var a = 33; var b = 10;  document.write("value of a => (a -= b) => ");  result = (a -= b));  document.write(result);  Output: |
| Value of a => (a -= b) => 23 |
|  |
| Conditional Operator | ? : (Conditional )  If Condition is true? Then value X : Otherwise value Y | Example:  var a = 10; var b = 20;  document.write ("((a > b) ? 100: 200) => ");  result = ((a > b) ? 100 : 200); document.write(result); |
| Output:  ((a > b) ? 100: 200) => 200 |

* **Conditional Statements**

1. **If Statement:**

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

Example:

var age = 20; if( age > 18 ){

document.write("<b>Qualifies for driving</b>");

}

Syntax:

if (expression){

Statement(s) to be executed if expression is true

}

1. **If… Else statement**

The 'if...else' statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.

Here JavaScript expression is evaluated. If the resulting value is true, the given statement(s) in the ‘if’ block, are executed. If the expression is false, then the given statement(s) in the else block are executed.

Example:

var age = 15;

if( age > 18 ){

document.write("<b>Qualifies for driving</b>");

}

else{

document.write("<b>Does not qualify for driving</b>"); }

Syntax:

if (expression){

Statement(s) to be executed if expression is true

}

else{

Statement(s) to be executed if expression is false

}

1. **If else ladder statement**

The if...else if... statement is an advanced form of if…else that allows JavaScript to make a correct decision out of several conditions.

There is nothing special about this code. It is just a series of if statements, where each if is a part of the else clause of the previous statement. Statement(s) are executed based on the true condition, if none of the conditions is true, then the else block is executed.

Example:

var book = "maths";

if(book == "history" ){

document.write("<b>History Book</b>");

}

else if( book == "maths" ){ document.write("<b>Maths Book</b>");

}

else{

document.write("<b>Unknown Book</b>");

}

Syntax:

if (expression 1){

Statement(s) to be executed if expression 1 is true

}

else if (expression 2) {

Statement(s) to be executed if expression 2 is true

}

else{

Statement(s) to be executed if no expression is true

}

1. **Switch case**

The objective of a 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.

Syntax:

switch (expression)

{

case condition 1: statement(s) break;

case condition 2: statement(s) break;

...

case condition n: statement(s) break;

default: statement(s)

}

The break statements indicate the end of a particular case. If they were omitted, the interpreter would continue executing each statement in each of the following cases.

Example:

switch (grade)

{

case 'A': document.write("Good job<br />"); break;

case 'B': document.write("Pretty good<br />"); break;

case 'C': document.write("Passed<br />"); break;

case 'D': document.write("Not so good<br />"); break;

case 'F': document.write("Failed<br />"); break;

default: document.write("Unknown grade<br />")

}

* **Looping Statements**

1. While Loop:

The most basic loop in JavaScript is the while loop .The purpose of a while loop is to execute a statement or code block repeatedly as long as an expression is true. Once the expression becomes false, the loop terminates.

The syntax of while loop in JavaScript is as follows −

while (expression){

Statement(s) to be executed if expression is true

}

Example:

var count = 0;

document.write("Starting Loop ");

while (count < 10){

document.write("Current Count : " + count + "<br />");

count++;

}

1. The 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.

The syntax for do-while loop in JavaScript is as follows −

do{

Statement(s) to be executed;

} while (expression);

Note − Don’t miss the semicolon used at the end of the do...while loop.

Example:

var count = 0;

document.write("Starting Loop" + "<br />"); do{

document.write("Current Count : " + count + "<br />"); count++;

}

while (count < 5); document.write ("Loop stopped!");

1. For Loop:

The 'for' loop is the most compact form of looping. It 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 a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
* The iteration statement where you can increase or decrease your counter.
* The syntax of for loop is JavaScript is as follows −

for (initialization; test condition; iteration statement){ Statement(s) to be executed if test condition is true

}

* Example:

var count;

document.write("Starting Loop" + "<br />");

for(count = 0; count < 10; count++){

document.write("Current Count : " + count );

document.write("<br />");

}

document.write("Loop stopped!");

1. For … in Loop

The for...in loop is used to loop through an object's properties. As we have not discussed Objects yet, you may not feel comfortable with this loop. But once you understand how objects behave in JavaScript, you will find this loop very useful.

Syntax

for (variablename in object){ statement or block to execute

}

In each iteration, one property from object is assigned to variablename and this loop continues till all the properties of the object are exhausted.

Example

var list = ['item1','item2','item3']; for (var i in list) {

document.write(list[i] + “<br>”);

}

* **User defined Function in Javascript**

The most common way to define a function in JavaScript is by using the function keyword, followed by a unique function name, a list of

parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax:

<script type="text/javascript">

function functionname(parameter-list)

{

statements

}

</script>

Example:

<script type="text/javascript"> function sayHello()

{

alert("Hello there");

}

</script>

**Calling function:**

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

Example:

<html>

<head>

<script type="text/javascript"> function sayHello()

{

document.write ("Hello there!");

}

</script>

</head>

<body>

<p>Click the following button to call the function</p>

<form>

<input type="button" onclick="sayHello()" value="Say Hello">

</form>

</body>

</html>

* **Dialog Box**

JavaScript supports three important types of dialog boxes. These dialog boxes can be used to raise and alert, or to get confirmation on any input or to have a kind of input from the users. Here we will discuss each dialog box one by one.

## Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

### Syntax

alert("*sometext*");

## Example

<html>  
<head>  
<script type="text/javascript">  
function show\_alert()  
{  
alert("This is an alert box!");  
}  
</script>  
</head>  
<body>  
  
<input type="button" onclick="show\_alert()" value="Show alert box" />  
  
</body>  
</html>

## Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

### Syntax

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

## Example

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

## Confirm 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 the OK button, the window method confirm() will return true.
* If the user clicks on the Cancel button, then confirm() returns false.
* You can use a confirmation dialog box as follows.

Example:

<html>

<head>

<script type="text/javascript"> function getConfirmation(){

var retVal = confirm("Do you want to continue ?"); if( retVal == true ){

document.write ("User wants to continue!"); return true;

}

else{

document.write ("User does not want to continue!"); return false;

}

}

</script>

</head>

<body>

<form>

<input type="button" value="Click Me" onclick="getConfirmation();"

/>

</form>

</body>

</html>