Web Programming

Javascript

Agenda

1 JavaScript basic elements

Objectives

At the end of this module you will be able to:

- Write JavaScript code using all the basic elements of JavaScript Programs

JavaScript basic elements




DHTML

- DHTML stands for Dynamic Hyper Text Markup Language which helps to add Dynamic content to static HTML pages.
- Dynamic HTML, is a new web technology that enables elements inside your web page to be, well, dynamic.
- Things once considered unchangeable once the page has loaded, such as text, page styles (font color, size etc), element position, etc., can now all be changed dynamically.
- It brings your web pages one step closer to how things look inside your television, where images appear and disappear, text flies in and out, and content moves around freely inside the screen.
- DHTML is any combination of Style Sheets, JavaScript, and Layering.

Introduction to JavaScript

- JavaScript is an easy to use object scripting language.
- It is designed for creating live online applications.
- Code is included as part of a HTML Document Scripts are run by the Web browser.

Introduction to JavaScript: JavaScript Versus Java

- JavaScript can be combined directly with HTML
- The JavaScript language structure is simpler than that of Java
- JavaScript is a purely interpreted language
- The JavaScript interpreter is built into a Web browser

Introduction to JavaScript: Using the SCRIPT Tag

- The <SCRIPT> tag is an extension to HTML that can enclose any number of JavaScript statements as shown here:
  - <SCRIPT>
  - JavaScript statements...
  - </SCRIPT>
- A document can have multiple <SCRIPT> tags, and each can enclose any number of JavaScript statements.
- Unlike HTML, JavaScript is case sensitive.

Introduction to JavaScript: Using the SCRIPT Tag

```
<HTML>
    <HEAD>
        <TITLE>Login Page </TITLE>
     </HEAD>
     <BODY>
          HTML Text goes here.
        <SCRIPT LANGUAGE="JavaScript">
            document.write("WelCome To green Bank")
        </SCRIPT>
     </BODY>
</HTML>
```

Elements of JavaScript Program

Elements of JavaScript Program can be divided into five categories, as follows:

- Variables
- Expressions
- Control Structures
- Functions
- Objects and Arrays

Elements of JavaScript Program: Variables

- Data Types
- Rules for variable names
- Variable Declaration and Scope

Elements of JavaScript Program: Variables

Data Types and Variables

- You can use data in two ways:
  - As a literal or constant value
  - As a variable
- The four fundamental data types used:
  - Numbers
  - Boolean, or logical values
  - Strings
  - The null value/ the object

Elements of JavaScript Program: Variables

Rules for variable names

- Variable names can include alphabets, digits and the underscore
- (_)
- The first character can be either an alphabet or the underscore
- Are case-sensitive
- There is no official limit on the length of variable names, but they must fit within one line.

JavaScript Statements

Conditional statements: if...else and switch

Loop Statements: for, while, do…while, break and continue

Object Manipulation Statements and Operators: new, this and with

Comments: single-line (//) and multi-line (/*...*/)

JavaScript Statements: if else

- **if...else Statement**
- Use the **if statement** to perform certain statements if a logical condition is true; use the optional else clause to perform other statements if the condition is false. An **if statement** looks as follows:

```
if (condition) {
   statements1
}
else {
   statements2
}
```

- The condition can be any JavaScript expression that evaluates to true or false. The statements to be executed can be any JavaScript statements, including further nested **if statements**. If you want to use more than one statement after an if or else statement, you must enclose the statements in curly braces: {}.

What will this code print ?

```
<html>
<body>
<script language="JavaScript">
var a=4, mesg=" ";
if(a==0){
document.write(a+" can't be odd or even"+"<BR>");
}
else{
if(a%2==0)
  mesg="Even";
else
  mesg="Odd";
document.write(x+" is "+ mesg);
}
</script>
</body></html>
```

JavaScript Statements: Switch Statement

- A switch statement allows a program to evaluate an expression and attempt to match the expression's value to a case label
- If a match is found, the program executes the associated statement
- A 'switch' statement looks as follows:

```
switch (expression){
   case label :
      statement;
      break;
default : statement;
}
```

JavaScript Statements: Switch Statement

- The program first looks for a label matching the value of expression and then executes the associated statement. If no matching label is found, the program looks for the optional default statement, and if found, executes the associated statement. If no default statement is found, the program continues execution at the statement following the end of switch.
- The optional break statement associated with each case label ensures that the program breaks out of switch once the matched statement is executed and continues execution at the statement following switch. If break is omitted, the program continues execution at the next statement within the switch statement itself.

What will this code print ?

```
<script language="javascript">
  var choice = 3, Login=" "
  switch(choice){
    case 1: login = "Manager"
            break;
    case 2: login = "Staff"
            break;
    default: login = "Customer";
  }
  document.write("you have Logged in As "+login)
</script>
```

JavaScript Statements: for Statement

When a for loop executes, the following occurs:

Step 1: The initializing expression 'initial-expression', if any, is executed. This expression usually initializes one or more loop counters, but the syntax allows an expression of any degree of complexity.

Step 2: The 'condition' expression is evaluated. If the value of condition is true, the loop statements execute. If the value of condition is false, the for loop terminates.

Step 3: Assuming that the condition is true, the statements execute.

Step 4: Finally, the update expression 'increment-expression' executes and control returns to step 2.

JavaScript Statements: for Statement

- A **for** loop repeats until a specified condition evaluates to false.
- A **'for' statement** looks as follows:

```
for ([initial-expression]; [condition]; [increment-expression])
   {
      statements
   }
```

JavaScript Statements: for Statement

Simple example to understand For Loop and if:

```
<SCRIPT language="javascript">
var loginstatus;
for(loginstatus=1;loginstatus<=3;loginstatus++){
if(loginstatus==1){
document.writeln(" Status=1 You can be Login As Manager");
}
if(loginstatus==2){
document.writeln(" Status=2 You can be Login As Staff");
}
if(loginstatus==3){
document.writeln(" status=3 You can be Login As Customer");
}
}

</SCRIPT >
```

What will this code print ?

```
<html>
<body>
<script language="JavaScript">
var x,y;
for(x=19;x>0;x--){
  for(y=1;y<=x;y++)
    document.write(x+" ");
  document.write("<BR>");
}
</script>
</body>
</html
```

JavaScript Statements: do…while Statement

- The do...while statement repeats until a specified condition evaluates to False
- A 'do...while' statement looks as follows

```
do {
    statement
} while (condition)
```

- Statement executes once before the condition is checked. If condition returns true, the statement executes again. At the end of every execution, the condition is checked. When the condition returns false, execution stops and control passes to the statement following do...while.

What will this code print ?

```
<SCRIPT language="javascript">

    var num = 1234566
    var count = 0
    do {
          count++;
          num /= 10;
        } while (num>1);

document.write("total digits"+count);
</ SCRIPT >
```

JavaScript Statements: while Statement

- A **while** statement executes its statements as long as a specified condition evaluates to true
- If the condition becomes false, the statements within the loop stop executing and control passes to the statement following the loop. The condition test occurs before the statements in the loop are executed. If the condition returns true, the statements are executed and the condition is tested again. If the condition returns false, execution stops and control is passed to the statement following **while**.
- A **'while'** statement looks as follows

```
while (condition) {
   statements
}
```

What will this code print ?

```
<html><body>
<script language="JavaScript">
var x=123;
var y=0;
var z;
while(x>0){
   z=x%10;
   y+=z;
   x=parseInt(x/10);
   document.write(x+"<BR>");
   document.write(y+"<BR>");
   document.write(z+"<BR>");
}
document.write(y);
</script>
</body></html>
```

Functions

- Functions are one of the fundamental building blocks in JavaScript. A function is a JavaScript procedure: a set of statements that performs a specific task. To use a function, you must first define it, then your script can call it.

- A function definition looks as follows:

```
function gcd(m,n) {
     return  n > 0   ?  gcd(n,m%n) : m ;
}
```

Functions(Contd.).

- A function definition consists of the function keyword, followed by the name of the function, a list of arguments to the function, enclosed in parentheses and separated by commas. The JavaScript statements that define the function are enclosed in curly braces: { }. The statements in a function can include calls to other functions defined in the current application. It is good practice to define all your functions in the HEAD of a page so that when a user loads the page, the functions are loaded first.

Functions(Contd.).

```
<script language="javascript">
    function interest(amt,per)
                {
               var m=(per/100)*amt;
           return  m;
                 }

var amount=50000;
var per=6;
 totalbalance= amount+interest(amount,per) ;
 document.write(totalbalance);
</script>
```

What will this code print ?

```
<html>
<body>
<script language="JavaScript">
function gcd(m,n){
  return n > 0 ? gcd(n,m%n):m;
}
document.write(gcd(96,16));
</script>
</body>
</html>
```

Summary

In this module, you were able to:

- Write JavaScript code using all the basic elements of JavaScript Programs

Thank You