

e-PG Pathshala

Subject: Computer Science

Paper: Web Technology

Module: JavaScript Objects

Module No: CS/WT/17

Quadrant 1 – e-text

Learning Objectives

In the last module we have learnt about the basics of JavaScript and to write simple JavaScript programs. Moreover we have discussed about JavaScript Variables, Operators and Control Statements and also about input and output statements.

In this module we will learn about how to create Arrays, Functions and Objects. Moreover we will understand and use built-in Objects in JavaScript and to get an idea about JavaScript DOM.

JavaScript Popup Boxes

Alert Box

An alert box is often used if we want to ensure information to the user. When an alert box pops up, the user will have to click "OK" to proceed.

Example:

```
<script>
alert("Hello World!")
</script>
```

Confirm Box

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

Prompt Box

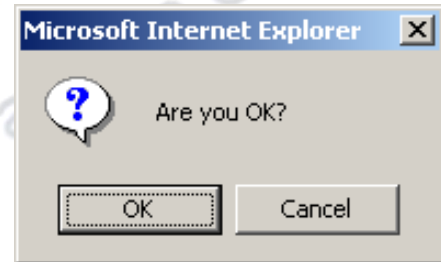
A prompt box is often used if you want the user to input a value before entering a page. When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value. If the user clicks "OK", the box returns the input value. If the user clicks "Cancel", the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

Example:

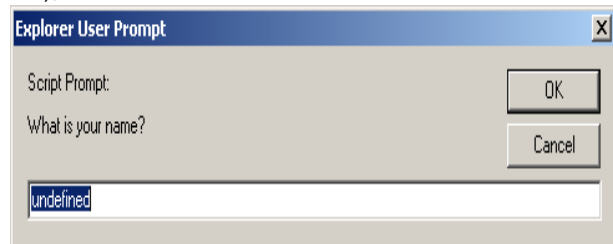
```
<script type="text/javascript">  
  alert("This is an Alert method");
```



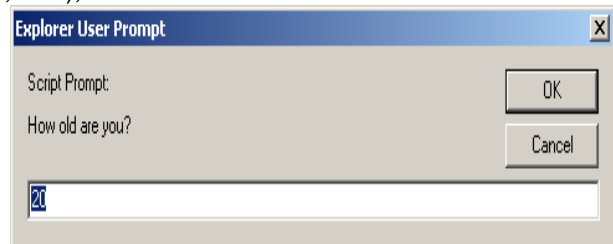
```
confirm("Are you OK?");
```



```
prompt("What is your name?");
```



```
prompt("How old are you?", "20");
```



```
</script>
```

JavaScript - Example

```
<html>
<body>
  <p>Click the button to calculate x.</p>
  <button onclick="myFunction()">Try it</button>
  <br/>
  <br/>Enter first number:
  <input type="text" id="txt1" name="text1">Enter second number:
  <input type="text" id="txt2" name="text2">
  <p id="demo"></p>
  <script>
    function myFunction() {
      var y = parseInt(document.getElementById("txt1").value);
      var z = parseInt(document.getElementById("txt2").value);
      var x = y + z;
      document.getElementById("demo").innerHTML = x;
    }
  </script>
</body>
</html>
```

OUTPUT:

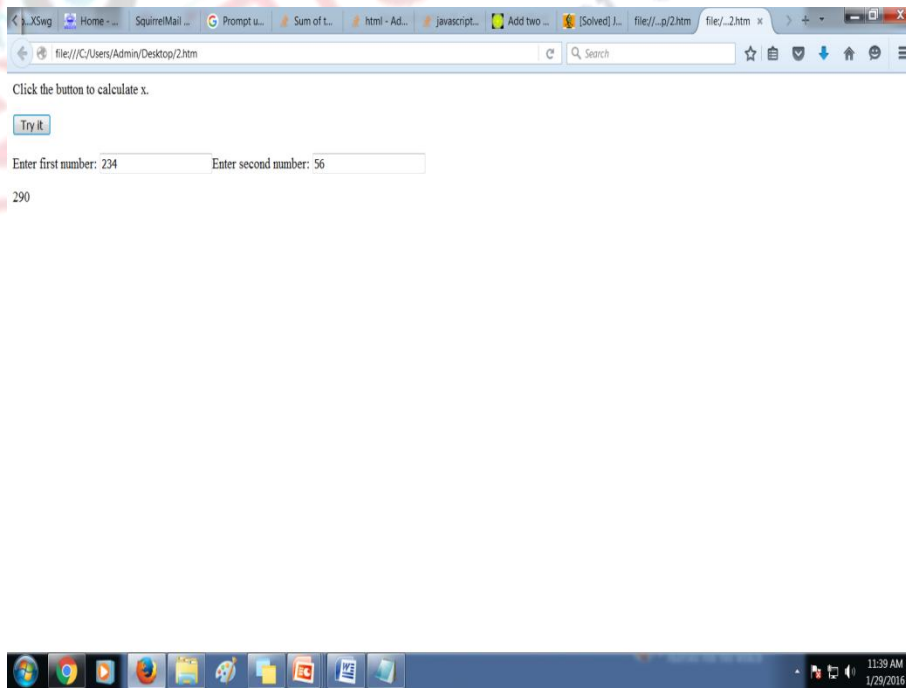


Figure 17.1 Output

Using Arrays

An **array** is an ordered collection of values referenced by a single variable name. The syntax for creating an array variable is:

```
var variable = new Array(size);
```

where *variable* is the name of the array variable and *size* is the number of elements in the array (optional).

```
var variable = new Array(values);
```

where *values* are the array elements enclosed in quotes and separated by commas.

```
var Month=new Array("", "January", "February", "March", "April", "May", "June",  
"July", "August", "September", "October", "November", "December");
```

To populate the array with values, use:

```
variable[i]=value;
```

JavaScript - Functions

We can define a function 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>
```

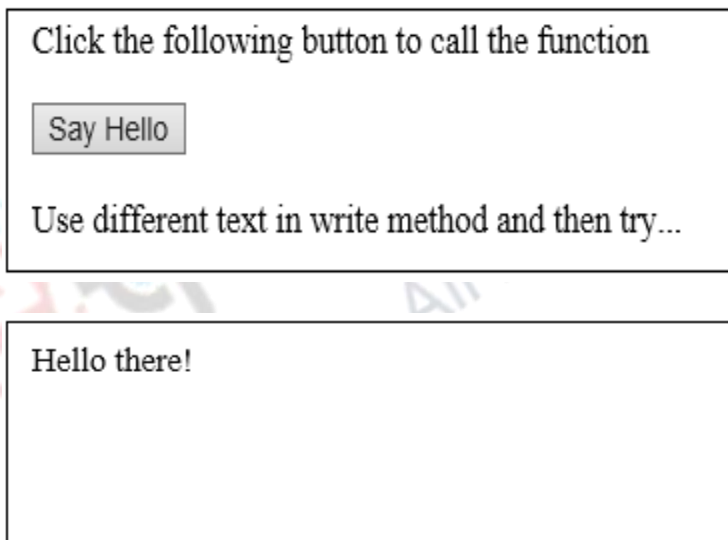
Example:

```
<script type="text/javascript">  
  <!--  
    function sayHello()  
    {  
      alert("Hello there");  
    }  
  //-->  
</script>
```

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">
<p> Use different text in write method and then try... <p>
</form>
</body>
</html>
```

OUTPUT:



The screenshot shows the rendered HTML code. It consists of two rectangular boxes. The top box contains the text "Click the following button to call the function" at the top, a button labeled "Say Hello" in the middle, and the text "Use different text in write method and then try..." at the bottom. The bottom box contains the text "Hello there!".

Function Parameters

It can take multiple parameters separated by comma.

Example:

```
<html>
<head>
<script type="text/javascript">
function sayHello(name, age) {
```

```
document.write (name + " is " + age + " years old.");
}
</script> </head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onClick="sayHello('Zara', 7)" value="Say Hello">
</form>
<p>Use different parameters inside the function and then try...</p>
</body>
</html>
```

OUTPUT:

Click the following button to call the function

Say Hello

Use different parameters inside the function and then try...

Zara is 7 years old.

The return Statement

The Function can have an optional **return** statement. This statement should be the last statement in a function.

Example:

```
<html>
<head>
<script type="text/javascript">
function concatenate(first, last)
{
var full;
full = first + last;
return full; }
function secondFunction()
{
```

```
var result;
result = concatenate('Zara', 'Ali');
document.write (result );
} </script> </head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onClick="secondFunction()" value="Call Function">
</form>
<p>Use different parameters inside the function and then try...</p>
</body>
</html>
```

OUTPUT:

Click the following button to call the function

Use different parameters inside the function and then try...

ZaraAli

JavaScript – Objects

JavaScript is an Object Oriented Programming (OOP) Scripting language . It has the following features,

1. **Encapsulation**
2. **Aggregation**
3. **Inheritance**
4. **Polymorphism**

Object Properties:

Properties are the values associated with a JavaScript object.

```
objectName.objectProperty = propertyValue;
```

Example

```
var str = document.title;
```

Object Methods:

Methods are **actions** that can be

performed on objects. Methods are stored in properties as **function definitions**.

A function is a standalone unit of statements and a method is attached to an object and can be referenced by the **this** keyword.

```
document.write("This is test");
```

User-Defined Objects

Objects are variables too. But objects can contain many values.

var car = {type:"Fiat", model:"500", color:"white"}; where values are written as **name:value** pairs

Methods are stored in properties as function definitions.

All user-defined objects and built-in objects are descendants of an object called **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. The constructor methods are Object(), Array(), and Date(). These constructors are built-in JavaScript functions.

```
var employee = new Object();
var books = new Array("C++", "Perl", "Java");
var day = new Date("August 15, 1947");
```

The Object() Constructor

JavaScript has built-in constructors for native objects like,

```
var x1 = new Object(); // A new Object object
var x2 = new String(); // A new String object
var x3 = new Number(); // A new Number object
var x4 = new Boolean() // A new Boolean object
var x5 = new Array(); // A new Array object
var x6 = new RegExp(); // A new RegExp object
```



```
var x7 = new Function(); // A new Function object
var x8 = new Date();    // A new Date object
```

We can create an "object type" using an built-in constructor function.

Example :

```
<html>
  <head>
    <title>User-defined objects</title>
    <script type="text/javascript">
      var book = new Object(); // Create the object
      book.subject = "JAVA"; // Assign properties to the object
      book.author = "Herbert Schildt";
    </script>
  </head>
  <body>
    <script type="text/javascript">
      document.write("Book name is : " + book.subject + "<br>");
      document.write("Book author is : " + book.author + "<br>");
    </script>
  </body>
</html>
```

OUTPUT:

Book name is : JAVA

Book author is : Herbert Schildt

Working with Object

We can create Object using a function (book) called as an object constructor. Once an object constructor is created, we can create new objects of the same type.

The example shows how we can create an "object type" using an object constructor function.

```
<html>
<head>
<title>User-defined objects</title>
<script type="text/javascript">
  function addPrice(amount)
  {
    this.price = amount;
  }

  function book(title, author)
  {
    this.title = title;
    this.author = author;
    this.addPrice = addPrice;
  }
</script>
</head>
<body>
</body>
</html>
```

```

    }
</script> </head>
<body>
<script type="text/javascript">
var myBook = new book("JAVA", "Herbert Schildt");
myBook.addPrice(100);
document.write("Book title is : " + myBook.title + "<br>");
document.write("Book author is : " + myBook.author + "<br>");
document.write("Book price is : " + myBook.price + "<br>");
</script>
</body>
</html>

```

OUTPUT:

Book title is : JAVA

Book author is : Herbert Schildt

Book price is : 100

The 'with' Keyword

The object specified as an argument '**with**' becomes the default object for the duration of the block that follows. The properties and methods for the object can be used without naming the object.

Syntax :

```

with (object)
{
properties used without the object name and dot
}

```

Example:

```

<html>
<head>
<title>User-defined objects</title>
<script type="text/javascript">
// Define a function which will work as a method
function addPrice(amount){
with(this) {
price = amount;
}
}
function book(title, author){
this.title = title;

```

```

this.author = author;
this.price = 0;
this.addPrice = addPrice; // Assign that method as property.
}
</script>
</head>
<body>
<script type="text/javascript">
var myBook = new book("JAVA", "Herbert Schildt");
myBook.addPrice(100);
document.write("Book title is : " + myBook.title + "<br>");
document.write("Book author is : " + myBook.author + "<br>");
document.write("Book price is : " + myBook.price + "<br>");
</script>
</body>
</html>

```

OUTPUT:

Book title is : JAVA

Book author is : Herbert Schildt

Book price is : 100

JavaScript Native Objects

The following is the list of Built-in or Native objects. They are,

1. JavaScript Number Object
2. JavaScript Boolean Object
3. JavaScript String Object
4. JavaScript Array Object
5. JavaScript Date Object
6. JavaScript Math Object
7. JavaScript RegExp Object

1. JavaScript - The Number Object

The **Number** object represents numerical data, either integers or floating-point numbers.

Syntax:

```
var v1 = new Number(number);
```

Note: In the place of number, if you provide any non-number argument, then the argument cannot be converted into a number, it returns **NaN** (Not-a-Number).

Number Properties:

MAX_VALUE, MIN_VALUE, NaN, NEGATIVE_INFINITY, POSITIVE_INFINITY, prototype,
constructor

Number Methods:

toExponential()

toFixed()

toLocaleString()

toPrecision()

toString()

valueOf()

2. JavaScript - The Boolean Object

The **Boolean** object represents two values, either "true" or "false".

Syntax:

```
var v1 = new Boolean(value);
```

Boolean Properties

Constructor, prototype

Boolean Methods

toSource()

toString()

valueOf()

3. JavaScript - The String Object

The **String** Object help us to work with strings.

Syntax:

```
var str = new String(string);
```

String Properties

constructor , length, prototype

String Methods

charAt(), charCodeAt(), concat(), indexOf(), lastIndexOf(), localeCompare(), match(), replace()

String Object - Example

```
<html >
<head>
<title>Character Processing Methods</title>
<script type = "text/javascript">
var s = "ZEBRA";
var s2 = "AbCdEfG";
document.writeln( "<p>Character at index 0 in '" + s + "' is " + s.charAt( 0 ) );
document.writeln( "<br />Character code at index 0 in '" + s + "' is " +
s.charCodeAt( 0 ) + "</p>" );
document.writeln( "<p>" + String.fromCharCode( 87, 79, 82, 68 ) + " contains
character codes 87, 79, 82 and 68</p>" );
document.writeln( "<p>" + s2 + " in lowercase is '" +s2.toLowerCase() + "'" );
document.writeln( "<br />" + s2 + " in uppercase is '" + s2.toUpperCase() +
'"</p>" );
</script>
</head><body></body></html>
```

OUTPUT:

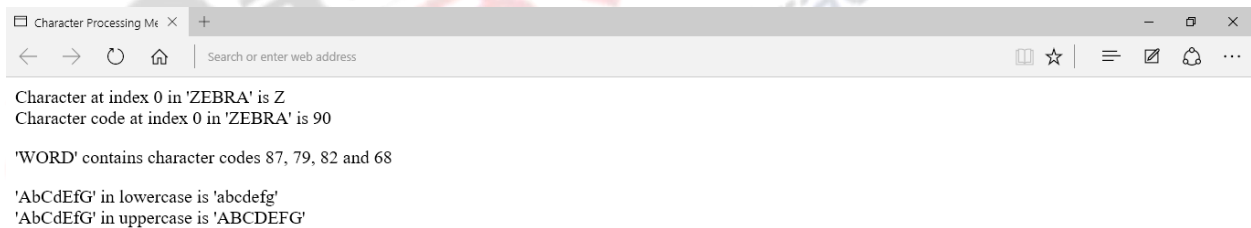


Figure 16.2 Output for String Object Example

4. JavaScript - The Arrays Object

The **Array** object lets you store multiple values in a single variable of same data type. The maximum length allowed for an array is 4,294,967,295.

Syntax:

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

create array by simply assigning values

```
var fruits = [ "apple", "orange", "mango" ];
```

Accessing array elements

fruits[0] is the first element

fruits[1] is the second element

fruits[2] is the third element

Array Properties

Constructor, index, input, length, prototype

Array Methods

concat(), every(), filter(), forEach(), indexOf(), join(), lastIndexOf(), map(), pop(), push(), reduce(), reduceRight(), reverse(), shift(), slice(), some(), toSource(), sort(), splice(), toString(), unshift()

Array Object - Example

```
<html >
  <head>
    <title>Initializing an Array with a Declaration</title>
    <script type = "text/javascript">
      function start()
      {
        var colors = new Array( "cyan", "magenta", "yellow", "black" );
        var integers1 = [ 2, 4, 6, 8 ];
        var integers2 = [ 2, , 8 ];
        outputArray( "Array colors contains", colors );
        outputArray( "Array integers1 contains", integers1 );
        outputArray( "Array integers2 contains", integers2 );
      }
      function outputArray( header, theArray )
      {
        document.writeln( "<h2>" + header + "</h2>" );
        document.writeln( "<table border = \"1\" \"\" + \"width = \"100%\">" );
        document.writeln( "<thead><th width = \"100\" \" \" + \"align = \"left\">Subscript</th>\" + \"<th align = \"left\">Value</th></thead><tbody>" );
```

```

for ( var i = 0; i < theArray.length; i++ )
document.writeln( "<tr><td>" + i + "</td><td>" + theArray[ i ] + "</td></tr>" );
document.writeln( "</tbody></table>" );
}
</script>
</head><body onload = "start()"><a href=index.html>back</a></body>
</html>

```

OUTPUT:

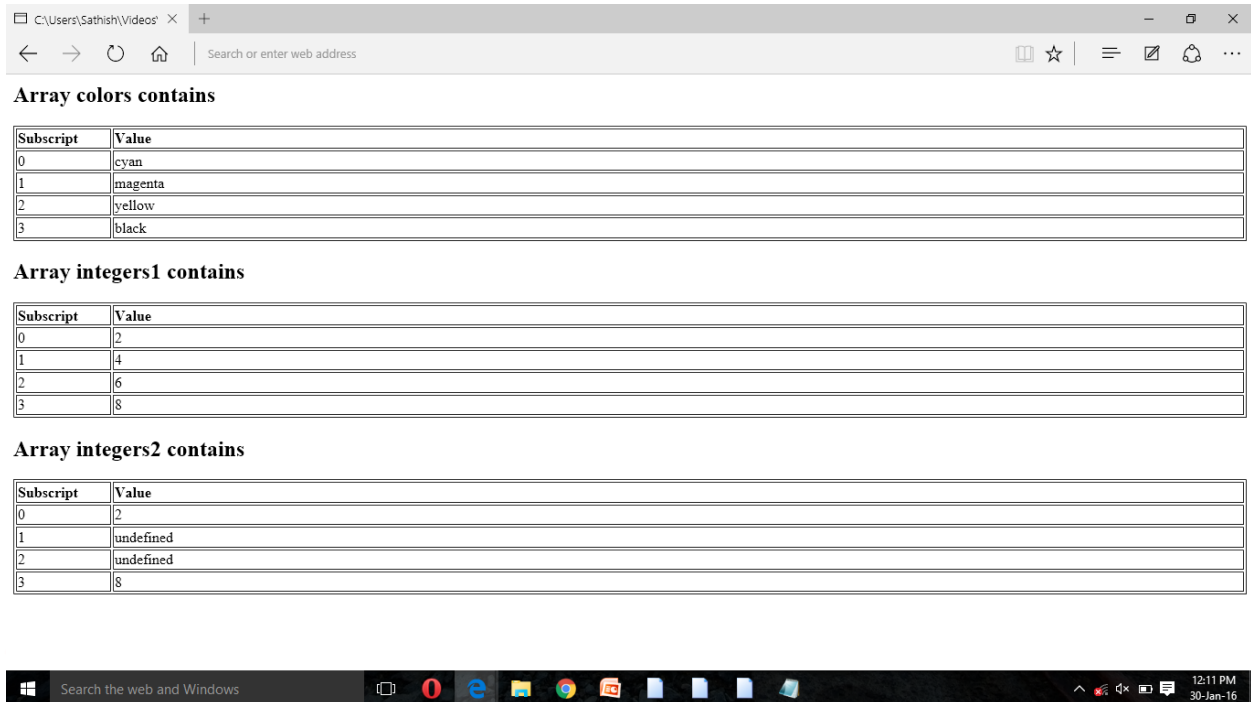


Figure 16.3 Output for Array Example

5. JavaScript - The Date Object

Date objects are created with the **new Date()**. Methods allow you to get and set the year, month, day, hour, minute, second, and millisecond fields of the object, using either local time or UTC (universal, or GMT) time.

Syntax:

- new Date()
- new Date(milliseconds)
- new Date(date string)
- new Date(year,month,date [,hour, minute, second, millisecond])

Date Properties:

constructor, prototype

Date Methods:

Date(), getDate(), getDay(), getFullYear(), getHours(), getMilliseconds(), getMinutes(), getMonth(), getSeconds(), getTime(), getTimezoneOffset(), getUTCDate(), setMinutes(), setMonth(), setSeconds(), setTime()

6. JavaScript - The Math Object

Unlike other global objects, **Math** is not a constructor. All the properties and methods of **Math** are static and can be called by using Math as an object without creating it.

Syntax

```
var pi_val = Math.PI;
var sine_val = Math.sin(30);
```

Math Properties:

E, LN2, LN10, LOG2E, LOG10E, PI, SQRT1_2, SQRT2

Math Methods:

abs(), acos(), asin(), atan(), atan2(), ceil(), cos(), exp(), floor(), log(), max(), min(), pow(), sin(), random(), round(), sqrt(), tan().

Math Object - Example

```
<html >
<head>
<title>Finding the Maximum of Three Values</title>
<script type = "text/javascript">
var input1 =
window.prompt( "Enter first number", "0" );
var input2 =
window.prompt( "Enter second number", "0" );
var input3 =
window.prompt( "Enter third number", "0" );
var value1 = parseFloat( input1 );
var value2 = parseFloat( input2 );
var value3 = parseFloat( input3 );
var maxValue = maximum( value1, value2, value3 );
document.writeln( "First number: " + value1 + "<br />Second number: " + value2
+
\"<br />Third number: " + value3 + "<br />Maximum is: " + maxValue );
function maximum( x, y, z ){
```



```
return Math.max( x, Math.max( y, z ) );
}
</script>
</head>
<body>
<p>Click Refresh (or Reload) to run the script again</p>
<a href=index.html>back</a>
</body>
</html>
```

OUTPUT:

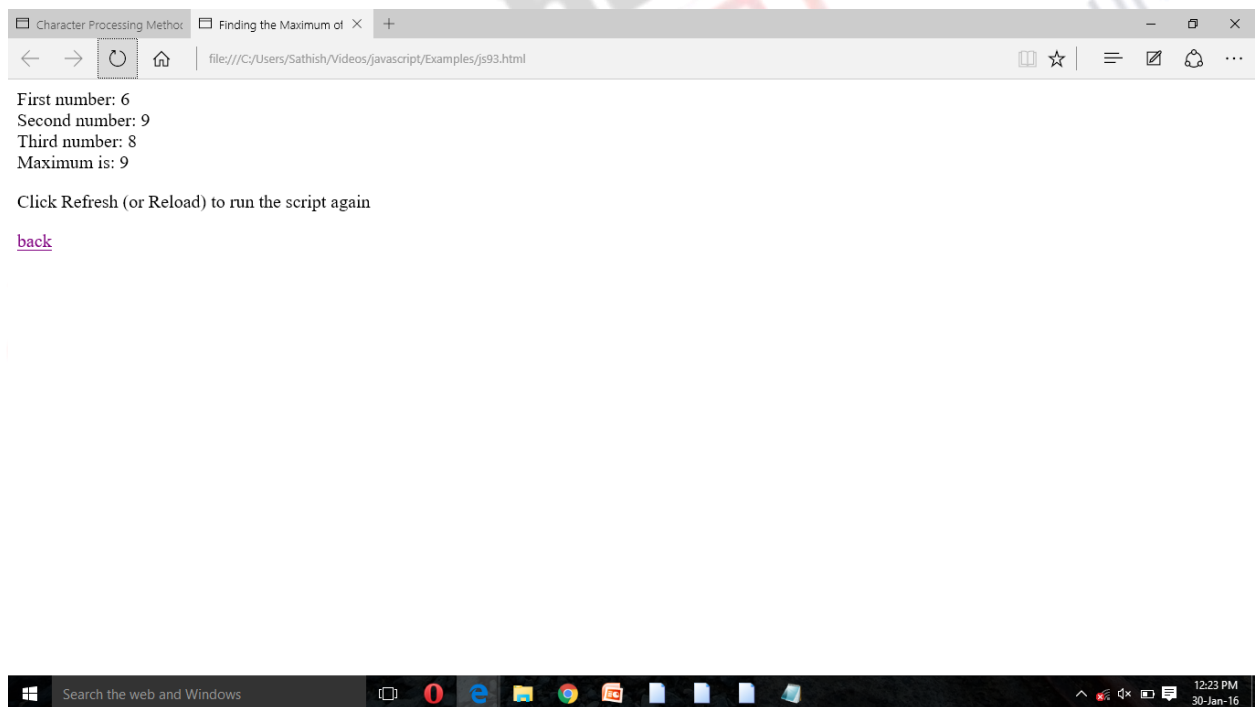


Figure 16.4 Output for Math Object Example

Summary

This module explains about JavaScript Dialog boxes with examples. This module also gives an idea about arrays, functions and objects. Moreover, the module also discusses about Objects that are of two types called native/built-in objects and user defined objects.



A Gateway to All Post Graduate Courses