

JavaScript Programming

A.M.S. Zunaitha Sulthana
Assistant Professor
Department of Computer Science & IT
Jamal Mohamed College (Autonomous)
Tiruchirappalli-620020

Web Programming

20UIT5CC9

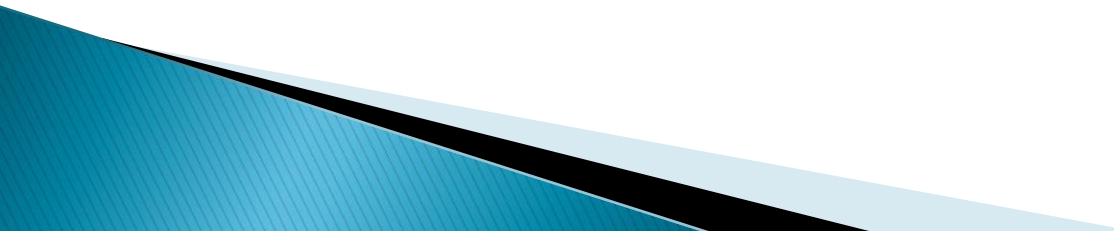
Unit III



Contents

- ▶ **Basic Concepts**
- ▶ **JavaScript Syntax Rules**
- ▶ **Understanding the Document Object Model**
- ▶ **Using Window Objects**
- ▶ **Working with the Document Object**
- ▶ **Working with the Location Object**
- ▶ **Working with DOM Nodes**
- ▶ **Using JavaScript Variables**
- ▶ **Understanding Expressions and Operators**
- ▶ **Data Types in JavaScript**
- ▶ **Using String Objects**
- ▶ **Using Numeric Arrays**
- ▶ **Using String Arrays.**

Introduction

- ▶ JavaScript is a scripting language most often used for client-side web development.
 - ▶ Using JavaScript we can create interactive user interface in a web page.
 - ▶ Eg: menu, pop-up alert, windows etc.
 - ▶ Manipulate web content dynamically.
- 

Example

```
<html>
<body>
<script type="text/javascript">
document.write("JavaScript is a simple language for javatpoint
  learners");
</script>
</body>
</html>
```

- ▶ The **script** tag specifies that we are using JavaScript.
- ▶ The **text/javascript** is the content type that provides information to the browser about the data.
- ▶ The **document.write()** function is used to display dynamic content through JavaScript.

3.1 Basic Concepts

- ▶ **Statements** : A statement is a section of code that performs a single action.
- ▶ A semicolon marks the end of a statement.

Eg:

- ▶ `hours = now.getHours();`
- ▶ `mins = now.getMinutes();`
- ▶ `secs = now.getSeconds();`
- ▶ `var foo = 'hello world ';`

3.1 Basic Concepts

- ▶ **Combining Tasks with Functions.**

A statement that uses a function.

Eg:

- ▶ `document.write("Testing.");`
- ▶ `text = prompt("Enter some text.");`

- ▶ This is an example of a function.
- ▶ Functions provide a simple way to handle a task, such as adding output to a web page.
- ▶ JavaScript includes a wide variety of built-in functions

3.1 Basic Concepts

Variables

- ▶ Variables are containers that can store a number, a string of text, or another value.
- ▶ JavaScript variables can contain numbers, text strings, and other values.

Example: The following statement creates a variable called fred and assigns it the value 27:

- ▶ `var fred = 27;`

3.1 Basic Concepts

Understanding Objects

- ▶ JavaScript also supports objects. Like variables, objects can store data—but they can store two or more pieces of data at once.
- ▶ The items of data stored in an object are called the properties of the object.
- ▶ JavaScript supports three kinds of objects:
 1. *Built-in objects*: Date, Array, String & Math.
 2. *DOM (Document Object Model)*: Represent various components of the browser and the current HTML document
For example ,alert() method of window object.
 3. *Custom Objects* : Objects that you create yourself.
For example, you could create a person object.

3.1 Basic Concepts

Conditionals

- ▶ JavaScript supports conditional statements, which enable you to answer questions like
- ▶ `if (count==1) alert("The countdown has reached 1.");`
- ▶ This compares the variable `count` with the constant `1` and displays an alert message to the user if they are the same.

3.1 Basic Concepts

Loops

- ▶ To create loops, or groups of statements that repeat a certain number of times.
- ▶ For example, these statements display the same alert 10 times, greatly annoying the user:

```
for (i=1; i<=10; i++) {  
  alert(“Yes, it’s yet another alert!”);  
}
```

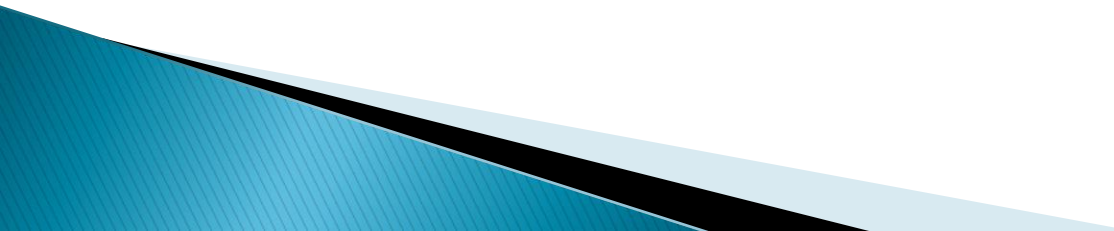
3.1 Basic Concepts

Event Handlers

- ▶ Event handlers are scripts that handle events .
- ▶ They tell the browser what to do when a certain event occurs. They include such events as “When the mouse button clicks” and “When this page is finished loading.”
- ▶ Eg : ``

3.2 JavaScript Syntax Rules.

Variable, Object, and Function Names

- ▶ Names can include uppercase letters, lowercase letters, numbers, and the underscore (_) character.
 - ▶ Names must begin with a letter or underscore.
 - ▶ JavaScript is case sensitive: score, Score, and SCORE would be considered three different variables.
- 

3.2 JavaScript Syntax Rules.

Case Sensitivity

- ▶ JavaScript keywords, such as `for` and `if`, are always lowercase.
- ▶ Built-in objects, such as `Math` and `Date`, are capitalized.
- ▶ DOM object names are lowercase, but their methods are often a combination of capitals and lowercase.

Eg:

- ▶ `toLowerCase` and `getElementById`.

3.2 JavaScript Syntax Rules.

```
<script type="text/javascript">  
<!--  
var name = "Ali";  
var money;  
money = 2000.50;  
//-->  
</script>
```

3.2 JavaScript Syntax Rules.

Reserved Words

- ▶ variable names not be reserved words. These include the words that make up the JavaScript language (such as if and for), DOM object names (such as window and document), and built-in object names (such as Math and Date).

Spacing

- ▶ Blank space (known as whitespace by programmers) is ignored by JavaScript.

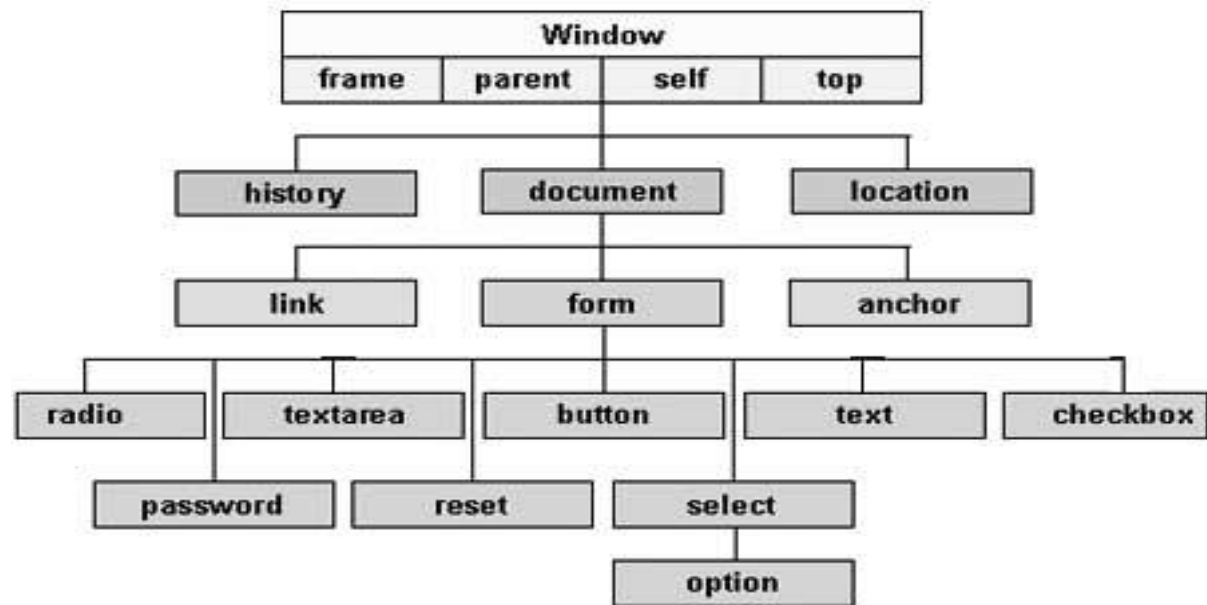
3.3 Understanding the Document Object Model (DOM)

- ▶ The DOM is not part of JavaScript or any other programming language rather, it's an API (Application programming interface) built in to the browser.
- ▶ These objects are organized into a tree-like structure and represent all the content and components of a web document.
- ▶ The objects in the DOM have properties—variables that describe the web page or document
- ▶ methods—functions that enable you to work with parts of the web page.

- ▶ Examples
 - Properties: `document.alinkColor`, `document.URL`, `document.forms[]`, `document.links[]`, `document.anchors[]`
 - Methods: `document.write(document.referrer)`
 - These change the content of the page!

3.4 Using Window Objects

- At the top of the browser object hierarchy is the window object, which represents a browser window.
- The window object is the parent object for all the objects.



3.5 Working with the document object

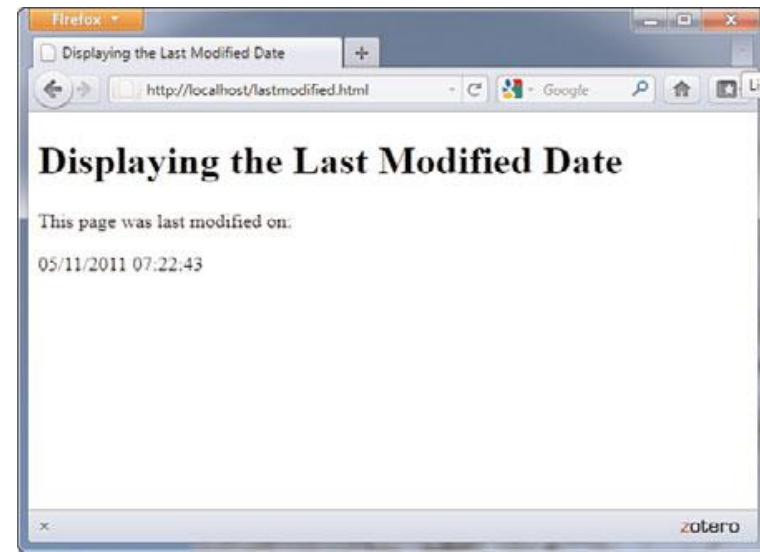
- ▶ The document object represents a web document or page.
- ▶ Web documents are displayed within browser windows, the document object is a child of the window object.
- ▶ Because the window object always represents the current window (the one containing the script) , you can use `window.document` to refer to the current document.

3.5 Working with the document object

- ▶ `document.URL` specifies the document's URL.
- ▶ `document.title` lists the title of the current page, defined by the HTML `<title>` tag.
- ▶ `document.referrer` is the URL of the page the user was viewing prior to the current page—usually, the page with a link to the current page.
- ▶ `document.lastModified` is the date the document was last modified. This date is sent from the server along with the page.
- ▶ `document.cookie` enables you to read or set a cookie for the document.
- ▶ `document.images` returns a collection of images used in the document.

3.5 Working with the document object

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>Displaying the Last Modified Date</title>
</head>
<body>
<h1>Displaying the Last Modified Date</h1>
<p>This page was last modified on:</p>
<script type="text/javascript">
document.write(document.lastModified);
</script>
</body>
</html>
```



3.5 Working with the document object

Writing Text in a Document

- ▶ The `document.write` method prints text as part of the HTML in a document window.
- ▶ `writeln`, also prints text, but it also includes a newline (`\n`) character at the end.
- ▶ The `document.write` method can be used within a `<script>` tag in the body of an HTML document.

3.5 Working with the document object

Using Links and Anchors

- ▶ Another child of the document object is the link object. There can be multiple link objects in a document. Each one includes information about a link to another location or an anchor.
- ▶ You can access link objects with the links array.
- ▶ Each member of the array is one of the link objects in the current page. A property of the array, `document.links.length`, indicates the number of links in the page.

3.5 Working with the document object

Eg:

- ▶ `link1 = links[0].href;`
- ▶ The statement assigns the entire URL of the first link to the variable `link1`.
- ▶ The anchor objects are also children of the document object.
- ▶ Each anchor object represents an anchor in the current document—a particular location that can be jumped to directly.

3.6 Working with the location Object

- ▶ A third child of the window object is the location object. This object stores information about the current URL stored in the window.
- ▶ For example, the following statement loads a URL into the current window:
- ▶ `window.location.href="http://www.google.com";`
- ▶ The `href` property used in this statement contains the entire URL of the window's current location

3.6 Working with the location Object

- ▶ `location.protocol` is the protocol part of the URL (`http:` in the example).
- ▶ `location.hostname` is the host name of the URL (`www.google.com` in the example).
- ▶ `location.port` is the port number of the URL (80 in the example).
- ▶ `location.pathname` is the filename part of the URL (`search` in the example).
- ▶ `location.search` is the query portion of the URL, if any (`q=javascript` in the example).

3.6 Working with the location Object

- ▶ `location.assign()` loads a new document when used as follows:

```
location.assign("http://www.google.com")
```
- ▶ `location.reload()` reloads the current document. This is the same as the Reload button on the browser's toolbar. If you optionally include the `true` parameter, it will ignore the browser's cache and force a reload whether the document has changed or not.
- ▶ `location.replace()` replaces the current location with a new one. This is similar to setting the location object's properties yourself.

3.7 Working with DOM Nodes

Basic Node Properties

- ▶ Each node also has a number of basic properties that you can examine or set.

These include the following:

- ▶ *nodeName* is the name of the node (not the ID). For nodes based on HTML tags, such as `<p>` or `<body>`, the name is the tag name: `p` or `body`.
- ▶ For the document node, the name is a special code: `#document`.
- ▶ Similarly, text nodes have the name `#text`.

3.7 Working with DOM Nodes

- ▶ *nodeType* is an integer describing the node's type: 1 for normal HTML tags, 3 for text nodes, and 9 for the document node.
- ▶ *nodeValue* is the actual text contained within a text node. This property is not valid for other types of nodes.
- ▶ *innerHTML* is the HTML content of any node. You can assign a value including HTML tags to this property and change the DOM child objects for a node dynamically.

3.7 Working with DOM Nodes

Node Relationship Properties

- ▶ The basic properties described previously, each node has a number of properties that describe its relation to other nodes.
- ▶ These include the following:
- ▶ *firstChild* :Is the first child object for a node. For nodes that contain text, such as h1 or p, the text node containing the actual text is the first child.
- ▶ *lastChild* is the node's last child object.
- ▶ *childNodes* is an array that includes all of a node's child nodes. You can use a loop with this array to work with all the nodes under a given node.

3.7 Working with DOM Nodes

Document Methods

- ▶ The document node's methods include the following:
- ▶ *getElementById(id)* returns the element with the specified id attribute.
- ▶ *getElementsByTagName(tag)* returns an array of all of the elements with a specified tag name. You can use the wildcard * to return an array containing all the nodes in the document.
- ▶ *createTextNode(text)* creates a new text node containing the specified text, which you can then add to the document.

3.7 Working with DOM Nodes

Node Methods

- ▶ Each node within a page has a number of methods available. Which of these are valid depends on the node's position in the page and whether it has parent or child nodes.

These include the following:

- ▶ *appendChild(new)* appends the specified new node after all of the object's existing nodes.
- ▶ *insertBefore(new, old)* inserts the specified new child node before the specified old child node, which must already exist.

3.8 Using Variables

Choosing Variable Names

- ▶ Variables are named containers that can store data (for example, a number, a text string, or an object).

There are specific rules you must follow when choosing a variable name:

- ▶ Variable names can include letters of the alphabet, both upper and lowercase.
- ▶ They can also include the digits 0–9 and the underscore (`_`) character.
- ▶ Variable names cannot include spaces or any other punctuation characters.
- ▶ The first character of the variable name must be either a letter or an underscore.
- ▶ Variable names are case sensitive — `totalnum`, `Totalnum`, and `TotalNum` are separate variable names.
- ▶ There is no official limit on the length of variable names, but they must fit within one line.
- ▶ Using these rules, the following are examples of valid variable names:
 - `total_number_of_fish`
 - `LastInvoiceNumber`
 - `a`
 - `_var39`

3.8 Using Variables

Using Local and Global Variables

- ▶ Global variables have the entire script as their scope. They can be used anywhere, even within functions.
- ▶ Local variables have a single function as their scope. They can be used only within the function they are created in.
- ▶ To create a global variable, declare it in the main script, outside any functions.
- ▶ use the var keyword to declare the variable, as in this

example:

```
var students = 25;
```

3.8 Using Variables

- ▶ A local variable belongs to a particular function. Any variable you declare with the `var` keyword in a function is a local variable
- ▶ To create a local variable within a function, you must use the `var` keyword.

3.8 Using Variables

```
<script type="text/javascript">
<!--
var myVar = "global"; // Declare a global variable
function checkscope( ) {
    var myVar = "local"; // Declare a local variable
    document.write(myVar);
}
//-->
</script>
```

It will produce the following result:

Local

3.8 Using Variables

Assigning Values to Variables

The equal sign is used to assign a value to a variable.

Example, statement assigns the value 40 to the variable lines:

```
lines = 40;
```

```
lines = lines + 1;
```

the following shorter version of the preceding example:

```
lines += 1;
```

Similarly,

subtract a number from a variable using the -= operator:

```
lines -= 1;
```

3.8 Using Variables

- ▶ JavaScript also includes the increment and decrement operators, ++ and --.
- ▶ `lines++`;
- ▶ If the operator is after the variable name, the increment or decrement happens after the current expression is evaluated.
- ▶ If the operator is before the variable name, the increment or decrement happens before the current expression is evaluated.
- ▶ The following two statements have different effects:
 - ▶ `alert(lines++)`;
 - ▶ `alert(++lines)`;

3.9 Understanding of Expression and Operators

TABLE 16.1 Common JavaScript Operators

Operator	Description	Example
+	Concatenate (combine) strings	<code>message="this is" + "a test";</code>
+	Add	<code>result = 5 + 7;</code>
-	Subtract	<code>score = score - 1;</code>
*	Multiply	<code>total = quantity * price;</code>
/	Divide	<code>average = sum / 4;</code>
%	Modulo (remainder)	<code>remainder = sum % 4;</code>
++	Increment	<code>tries++;</code>
--	Decrement	<code>total--;</code>

Lists the operators from lowest to highest precedence, and operators with highest precedence are evaluated first.

3.9 Understanding of Expression and Operators

Operator Precedence

- ▶ JavaScript uses rules of operator precedence to decide how to calculate the values.

Example

- ▶ $\text{result} = 4 + 5 * 3;$
- ▶ If you try to calculate this result, there are two ways to do it. You could multiply $5 * 3$ first and then add 4 (result: 19) or add $4 + 5$ first and then multiply by 3 (result: 27).
- ▶ JavaScript solves this dilemma by following the precedence rules: Because multiplication has a higher precedence than addition, it first multiplies $5 * 3$ and then adds 4, producing a result of 19.

3.9 Understanding of Expression and Operators

- ▶ Sometimes operator precedence doesn't produce the result you want. For example, consider this statement:
- ▶ $\text{result} = a + b + c + d / 4;$
- ▶ You can control precedence by using parentheses . To calculate an average:
- ▶ $\text{result} = (a + b + c + d) / 4;$

Operators

Out Put

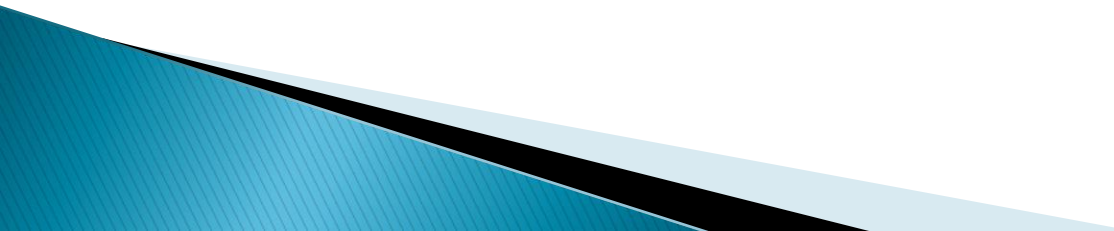
```
<!doctype html>
<html>
<body>
<script>
  var numOne=12, numTwo=10, res;
  res = numOne + numTwo;
  document.write("Add = " + res + "<br/>");
  res = numOne - numTwo;
  document.write("Subtract = " + res + "<br/>");
  res = numOne * numTwo;
  document.write("Multiply = " + res + "<br/>");
</script>
</body>
</html>
```

Add = 22

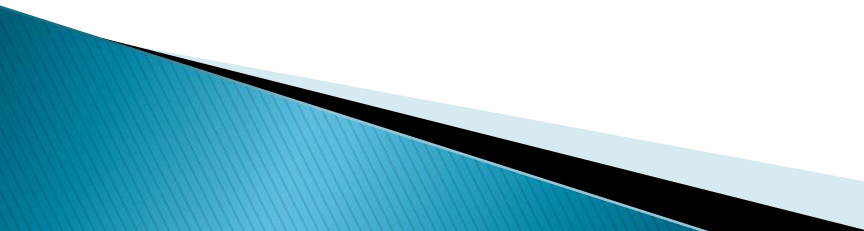
Subtract =2

Multiply = 120

3.10 Data types in javascript

- ▶ In some computer languages, you have to specify the type of data a variable will store, for example, a number or a string.
 - ▶ In JavaScript, you don't need to specify a data type in most cases. However, you should know the types of data JavaScript can deal with.
- 

3.10 Data types in Javascript

- ▶ *Numbers*, such as 3, 25, or 1.4142138—JavaScript supports both integers and floating-point numbers.
 - ▶ *Boolean*, or logical values—These can have one of two values: true or false. These are useful for indicating whether a certain condition is true.
 - ▶ *Strings*, such as “I am a jelly doughnut”—These consist of one or more characters of text.
 - ▶ *The null value, represented by the keyword null*—This is the value of an undefined variable..
- 

3.10 Data types in Javascript

- ▶ JavaScript keeps track of the data type currently stored in each variable, it doesn't restrict you from changing types midstream.
- ▶ For example, suppose you declared a variable by assigning it a value:
 - ▶ `total = 31;`
 - ▶ This statement declares a variable called total and assigns it the value of 31.
 - ▶ This is a numeric variable. Now suppose you changed the value of total:
 - ▶ `total = "albatross";`
 - ▶ This assigns a string value to total, replacing the numeric value.
 - ▶ JavaScript will not display an error

```
<html>
<head>
<title> GfG typeof example </title>
</head>
<body>
  <script type="text/javascript">
    var a = 17;
    var b = "javascript";
    var c = "";
    var d = null;
    document.write("Type of a = " + (typeof a));
    document.write("<br>");
    document.write("Type of b = " + (typeof b));
    document.write("<br>");
    document.write("Type of c = " + (typeof c));
    document.write("<br>");
    document.write("Type of d = " + (typeof d));
    document.write("<br>");
    document.write("Type of e = " + (typeof e));
    document.write("<br>");
  </script>
</body>
</html>
```

Output:

Type of a = number
Type of b = string
Type of c = string
Type of d = object
Type of e = undefined

3.11 Using String Objects

- ▶ Strings store a group of text characters and are named similarly to other variables.
- ▶ `test = "This is a test";`

3.11 Using String Objects

Creating a String Object

- ▶ There are two ways to create a new String object

```
test = "This is a test";
```

```
test = new String("This is a test");
```

- ▶ The second statement uses the new keyword, which is used to create objects.
- ▶ This tells the browser to create a new String object containing the text This is a test and assigns it to the variable test.

3.11 Using String Objects

Assigning a Value

- ▶ You can also assign a value after the string has already been created.
- ▶ For example, the following statement replaces the contents of the test variable with a new string:

```
test = "This is only a test.";
```

- ▶ The concatenation operator (+) to combine the values of two strings.
- ▶ The += operator to add text to a string.
- ▶ Example, this statement adds a period to the current contents of the string sentence:

```
sentence += ".";
```

3.11 Using String Objects

LISTING 16.1 Assigning Values to Strings and Combining Them

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>String Text</title>
  </head>

  <body>
    <h1>String Test</h1>
    <script type="text/javascript">
      test1 = "This is a test. ";
      test2 = "This is only a test.";
      both = test1 + test2;
      alert(both);
    </script>
  </body>
</html>
```



3.11 Using String Objects

Calculating the String's Length

`test.length` refers to the length of the test string.

Example:

- ▶ `test = "This is a test.";`
- ▶ `document.write(test.length);`
- ▶ The second statement displays the length of the string
- ▶ In this Example, 15 characters.
- ▶ The length property is a read-only property

3.11 Using String Objects

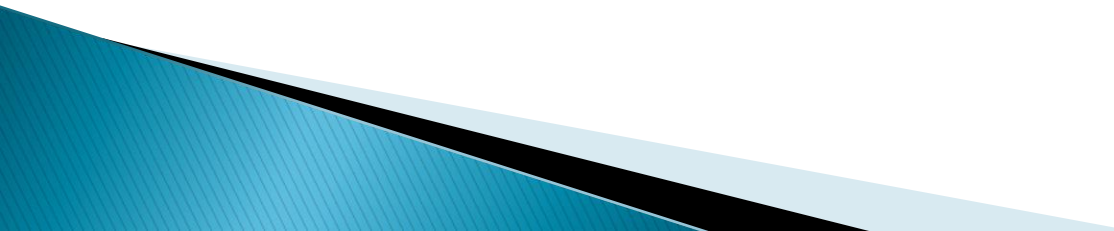
Converting the String's Case

- ▶ Two methods of the String object enable you to convert the contents of a string to all uppercase or all lowercase:
- ▶ `toUpperCase()`—Converts all characters in the string to uppercase
- ▶ `toLowerCase()`—Converts all characters in the string to lowercase

Example:

- ▶ `test = "This Is A Test";`
- ▶ `document.write(test.toLowerCase());`
- ▶ The result would be : this is a test

3.12 Using Numeric Array

- ▶ An array is a numbered group of data items that you can treat as a single unit.
 - ▶ For example, you might use an array called scores to store several scores for a game. Arrays can contain strings, numbers, objects, or other types of data.
 - ▶ Each item in an array is called an element of the array.
- 

3.12 Using Numeric Array

Creating a Numeric Array

Example creates an array with four elements:

```
scores = new Array(4);
```

- ▶ To assign a value to the array, you use an index in brackets . Indexes begin with 0, so the elements of the array in this example would be numbered 0 to 3. These statements assign values to the four elements of the array:
- ▶ `scores[0] = 39;`
- ▶ `scores[1] = 40;`
- ▶ `scores[2] = 100;`
- ▶ `scores[3] = 49;`

3.12 Using Numeric Array

- ▶ creates the same scores array in a single line:
`scores = new Array(39,40,100,49);`
- ▶ alternative way to create the scores array:
`scores = [39,40,100,49];`

3.12 Using Numeric Array

Understanding Array Length

- ▶ This tells you the number of elements in the array. If you specified the length when creating the array, this value becomes the length property's value.
- ▶ Example,

```
scores = new Array(30);  
document.write(scores.length);
```
- ▶ These statements would print the number 30:

3.12 Using Numeric Array

- ▶ You can declare an array without a specific length, and change the length later by assigning values to elements or changing the length property.
- ▶ For example, these statements create a new array and assign values to two of its elements:

```
test = new Array();  
test[0]=21;  
test[5]=22;
```
- ▶ In this example, because the largest index number assigned so far is 5, the array has a length property of 6—remember, elements are numbered starting at 0.

3.13 Using String Arrays

Creating a String Array

- ▶ You declare a string array in the same way as a numeric array.
- ▶ JavaScript does not make a distinction between them:

```
names = new Array(30);
```

- ▶ You can then assign string values to the array elements :

```
names[0] = "Henry J. Tillman";
```

```
names[1] = "Sherlock Holmes";
```

3.13 Using String Arrays

- ▶ As with numeric arrays, you can also specify a string array's contents when you create it.

```
names = new Array("Henry J. Tillman",  
"Sherlock Holmes");
```

```
names = ["Henry J. Tillman", "Sherlock Holmes"];
```

- ▶ You can use string array elements any where you would use a string .
- ▶ For example, the following statement prints the first five characters of the first element of the names array, resulting in Henry:

```
document.write(names[0].substring(0,5));
```

3.13 Using String Arrays

Splitting a String

- ▶ JavaScript includes a string method called `split`, which splits a string into its component parts.
- ▶ To use this method, specify the string to split and a character to divide the parts:

```
test = "John Q. Public";  
parts = test.split(" ");
```

- ▶ In this example, the `test` string contains the name John Q. Public.
- ▶ The `split` method in the second statement splits the name string at each space, resulting in three strings. These are stored in a string array called `parts`.
- ▶ After the example statements execute, the elements of `parts` contain the following:

```
parts[0] = "John" .   parts[1] = "Q." .   parts[2] = "Public"
```

3.13 Using String Arrays

- ▶ JavaScript also includes an array method, `join`, which performs the opposite function. This statement reassembles the parts array into a string:

```
fullname = parts.join(" ");
```

- ▶ The value in the parentheses specifies a character to separate the parts of the array. In this case, a space is used, resulting in the final string John Q.
- ▶ **Public.** If you do not specify a character, commas are used.

3.13 Using String Arrays

Sorting a String Array

- ▶ JavaScript also includes a sort method for arrays, which returns an alphabetically sorted version of the array.
- ▶ For example, the following statements initialize an array of four names and sort them:

```
names[0] = "Public, John Q.";
names[1] = "Doe, Jane";
names[2] = "Duck, Daisy";
names[3] = "Mouse, Mickey";
sortednames = names.sort();
```

- ▶ The last statement sorts the names array and stores the result in a new array, sorted names.

References

- ▶ <https://www.javatpoint.com/javascript-example>
- ▶ <https://codescracker.com/js/program/javascript-add-subtract-multiply-divide.html>
- ▶ https://www.tutorialspoint.com/javascript/javascript_tutorial.pdf
- ▶ **Text Book**
“Sams Teach Yourself HTML, CSS and JavaScript All in One” by Juile C. Meloni