

Java Script

Introduction

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, Chrome, Opera, and Safari.

In this lecture we will introduce JavaScript programming and present examples that illustrate several important features of JavaScript. Each example is carefully analyzed one line at a time

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

The first sample

We begin by considering a simple *script* (or *program*) that displays the text “**Welcome to JavaScript Programming!**” in the body of an XHTML document. The Internet Explorer Web browser contains a *JavaScript interpreter*, which processes the commands written in JavaScript. The JavaScript code and its output are shown in Fig. 3-1.

```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
45
46 <!-- Fig. 7.1: welcome.html -->
47 <!-- Displaying a line of text -->
78
79 <html xmlns = "http://www.w3.org/1999/xhtml">
9 <head>
10 <title>A First Program in JavaScript</title>
11
12 <script type = "text/javascript">
13 <!--
14 document.writeln(
```

```
15 "<h1>Welcome to JavaScript Programming!</h1>" );  
16 // -->  
17 </script>  
18  
19 </head><body></body>  
20 </html>
```

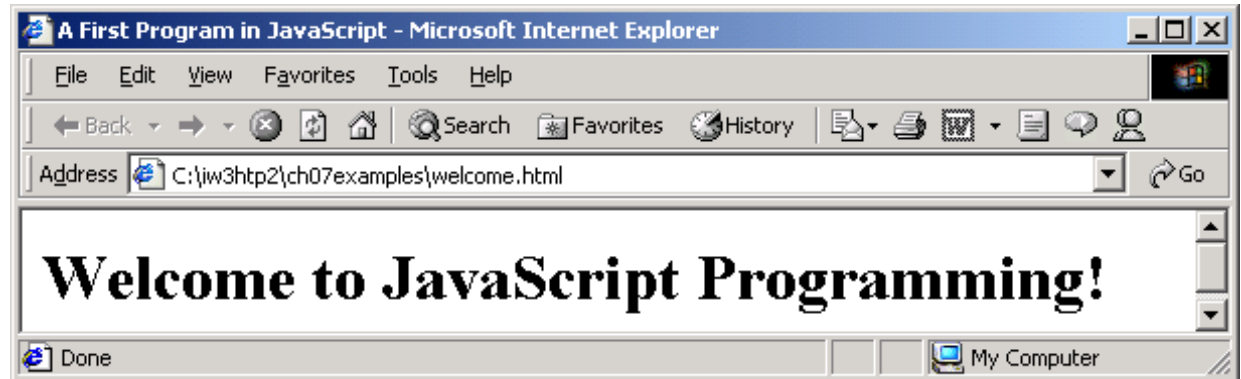


Figure 3-1

This program illustrates several important JavaScript features:

- Line 12 uses the `<script>` tag to indicate to the browser that the text which follows is part of a script. The *type attribute* specifies the type of file as well as the *scripting language* used in the script—in this case, a **text** file written in **javascript**. Both Microsoft Internet Explorer and Netscape Communicator use JavaScript as the default scripting language.
- Lines 14–15 instruct the browser's JavaScript interpreter to perform an *action*, namely to display in the Web page the *string* of characters contained between the *double quotation (") marks*. A string is sometimes called a *character string*, a *message* or a *string literal*. We refer to characters between double quotation marks generically as strings. Individual whitespace characters between words in a string are not ignored by the browser. However, if consecutive spaces appear in a string, browsers condense those spaces to a single space. Also, in most cases, browsers ignore leading whitespace characters (i.e., whitespace at the beginning of a string). use the browser's *document object*, which represents the XHTML document the browser is currently displaying. The **document** object allows a script programmer to specify text to display in the XHTML document. The browser contains a complete set of objects that allow script programmers to access and manipulate every element of an XHTML document.

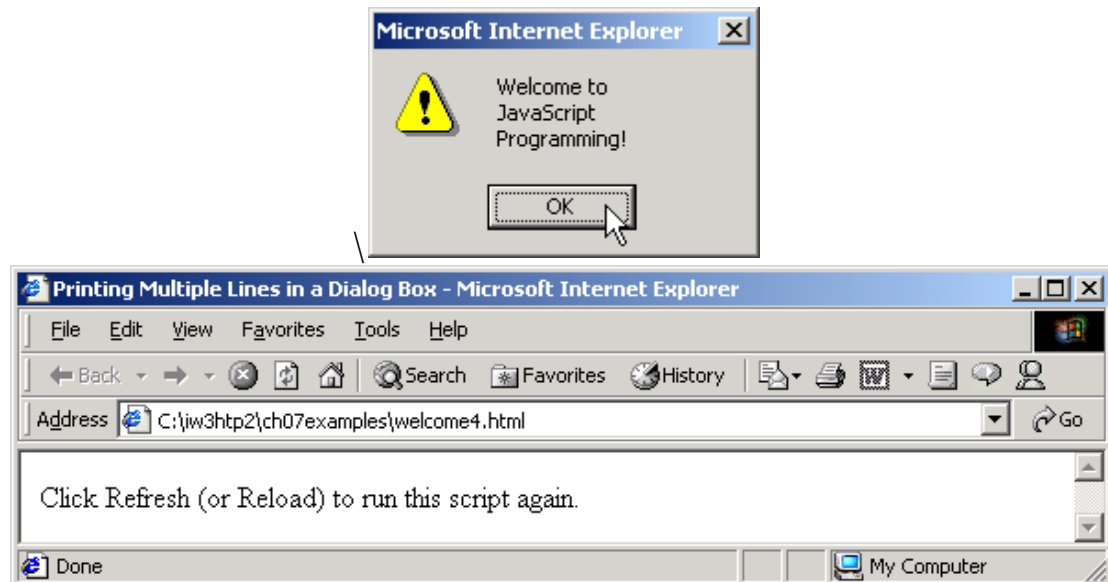
Alert message in JavaScript

Line 13 in the script uses the browser's *window* object to display an alert dialog. The argument to the **window** object's *alert* method is the

string to display. Executing the preceding statement displays the dialog shown in the first window of Fig. 7.4. The *title bar* of the dialog contains the string **Microsoft Internet Explorer**, to indicate that the browser is presenting

a message to the user. The dialog provides an **OK** button that allows the user to *dismiss* (i.e., *hide*) the dialog by clicking the button. To dismiss the dialog position the *mouse cursor* (also called the *mouse pointer*) over the **OK** button and click the mouse.

```
1 <script type = "text/javascript">
2 <!--
3 window.alert( "Welcome to\nJavaScript\nProgramming!" );
4 // -->
5 </script>
```



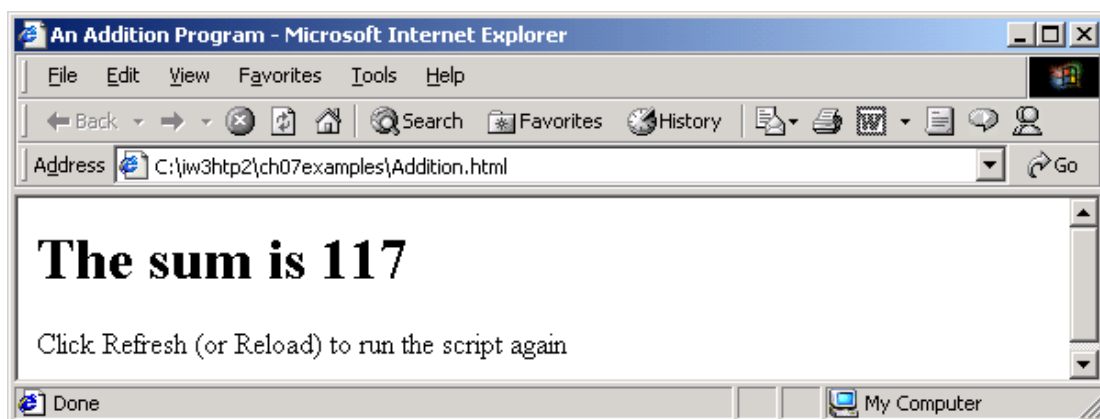
Adding integer to the programs

```
12<script type = "text/javascript">
13 <!--
14 var firstNumber, // first string entered by user
15 secondNumber, // second string entered by user
16 number1, // first number to add
17 number2, // second number to add
18 sum; // sum of number1 and number2
19
20 // read in first number from user as a string
21 firstNumber =
22 window.prompt( "Enter first integer", "0" );
// read in second number from user as a string
25 secondNumber =
```

```

26 window.prompt( "Enter second integer", "0" );
27
28 // convert numbers from strings to integers
29 number1 = parseInt( firstNumber );
30 number2 = parseInt( secondNumber );
31
32 // add the numbers
33 sum = number1 + number2;
34
35 // display the results
36 document.writeln( "<h1>The sum is " + sum + "</h1>" );
37 // -->
38 </script>

```



Lines 14–18 are *declarations*. The keyword **var** at the beginning of the statement indicates that the words **firstNumber**, **secondNumber**, **number1**, **number2** and **sum** are the names of *variables*. A variable is a location in the computer's memory where a value can be stored for use by a program. All variables should be declared with a name in a **var**

statement before they are used in a program. Although using **var** to declare variables is not required.

Comparison in JavaScript

Comparison operators are used in logical statements to determine equality or difference between variables or values let $x=5$.

Operator	Description	Example
==	is equal to	$x==8$ is false
===	is exactly equal to (value and type)	$x===5$ is true $x===\text{"5"}$ is false
!=	is not equal	$x!=8$ is true
>	is greater than	$x>8$ is false
<	is less than	$x<8$ is true
>=	is greater than or equal to	$x>=8$ is false
<=	is less than or equal to	$x<=8$ is true

Logical Operators

Logical operators are used to determine the logic between variables or values. Given that $x=6$ and $y=3$, the table below explains the logical operators:

Operator	Description	Example
&&	and	$(x < 10 \ \&\& \ y > 1)$ is true
	or	$(x==5 \ \ y==5)$ is false
!	not	$!(x==y)$ is true

```

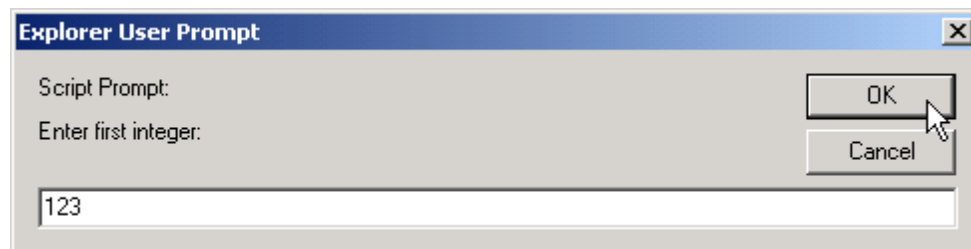
13<script type = "text/javascript">
14 <!--
15 var first, // first string entered by user
16 second, // second string entered by user
17 number1, // first number to compare
18 number2; // second number to compare
19
20 // read first number from user as a string
21 first = window.prompt( "Enter first integer:", "0" );
22
23 // read second number from user as a string
24 second = window.prompt( "Enter second integer:", "0" );
25

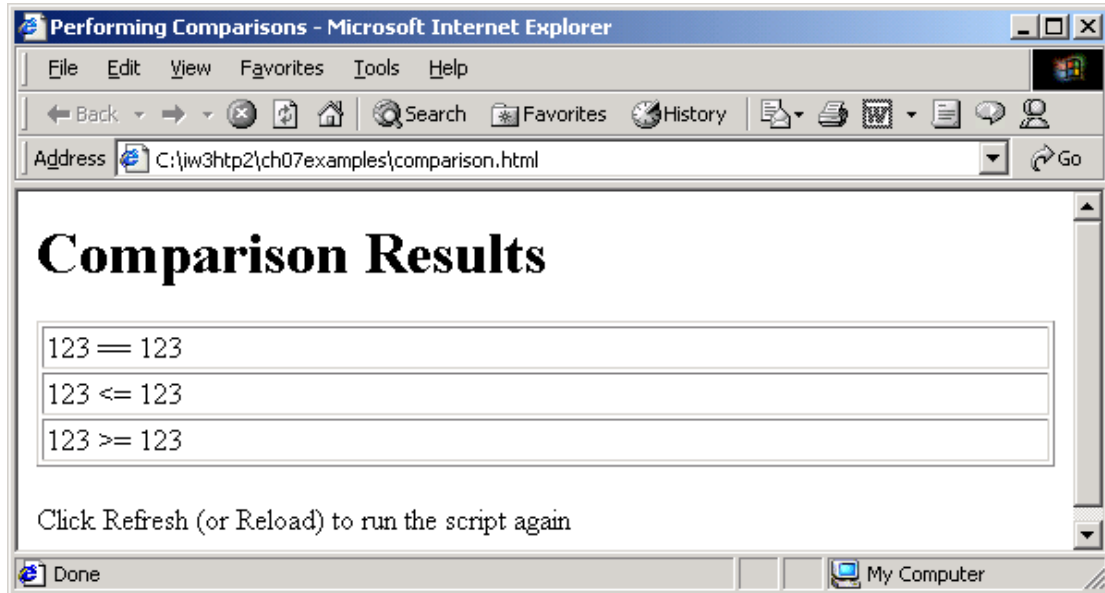
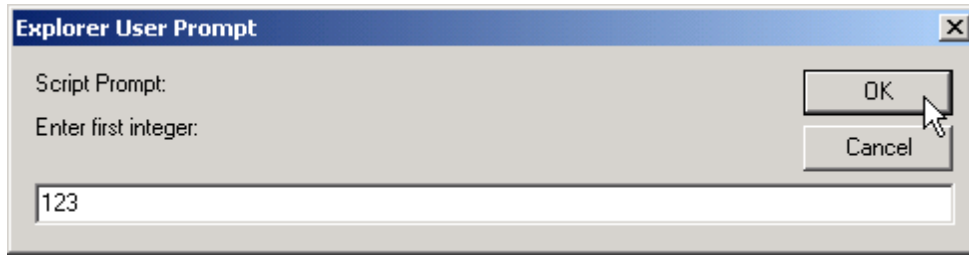
```

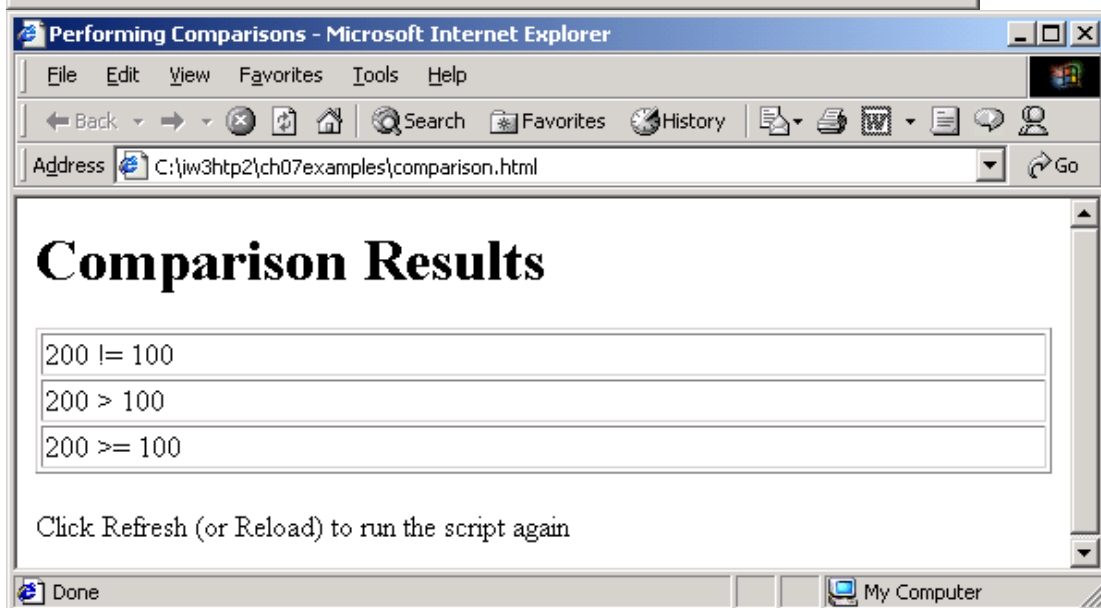
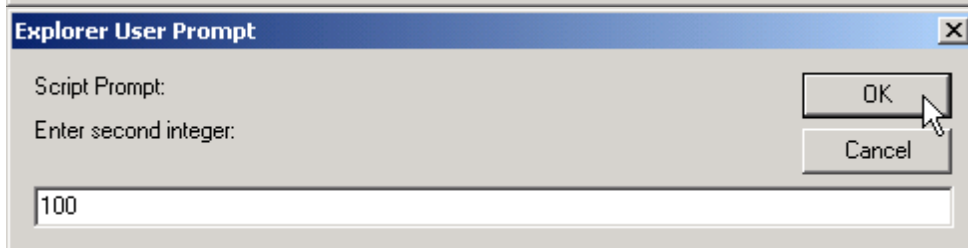
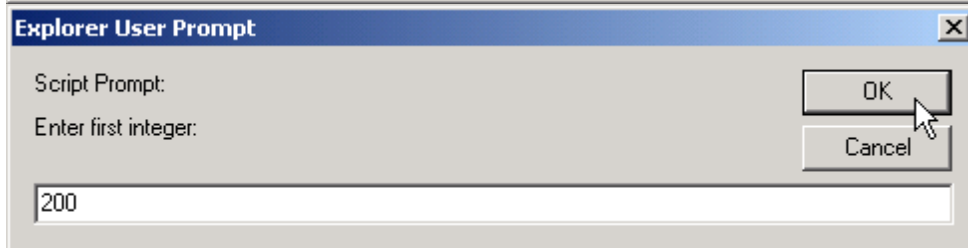
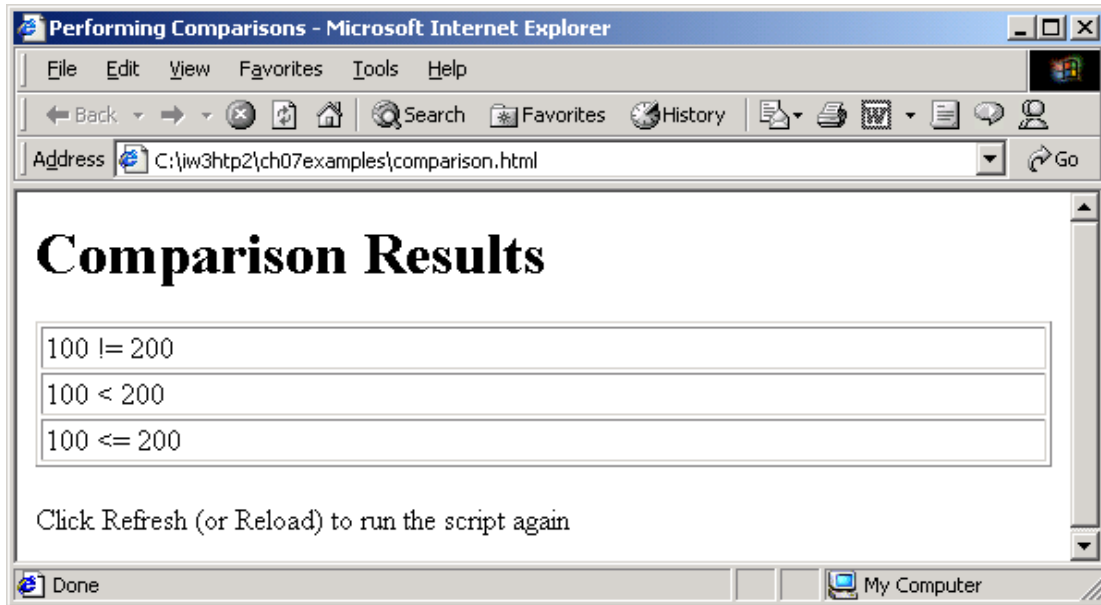
```

26 // convert numbers from strings to integers
27 number1 = parseInt( first );
28 number2 = parseInt( second );
29
30 document.writeln( "<h1>Comparison Results</h1>" );
31 document.writeln(
32 "<table border = \"1\" width = \"100%\">" );
33
34 if ( number1 == number2 )
35 document.writeln( "<tr><td>" + number1 + " == " +
36 number2 + "</td></tr>" );
37
38 if ( number1 != number2 )
39 document.writeln( "<tr><td>" + number1 + " != " +
40 number2 + "</td></TR>" );
41
42 if ( number1 < number2 )
43 document.writeln( "<tr><td>" + number1 + " < " +
44 number2 + "</td></tr>" );
45
46 if ( number1 > number2 )
47 document.writeln( "<tr><td>" + number1 + " > " +
48 number2 + "</td></tr>" );
49
50 if ( number1 <= number2 )
51 document.writeln( "<tr><td>" + number1 + " <= " +
52 number2 + "</td></tr>" );
53
54 if ( number1 >= number2 )
55 document.writeln( "<tr><td>" + number1 + " >= " +
56 number2 + "</td></tr>" );
57
58 // Display results
59 document.writeln( "</table>" );
60 // -->
61 </script>

```







The JavaScript Switch Statement

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
switch(n)
{
case 1:
    execute code block 1
    break;
case 2:
    execute code block 2
    break;
default:
    code to be executed if n is different from case 1 and 2
}
```

This is how it works: First we have a single expression n (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically.

Confirm Box

A confirm box is often used if 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.

Syntax

```
confirm("sometext");
```

Example

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button");
if (r==true)
{
document.write("You pressed OK!");
}
else
{
document.write("You pressed Cancel!");
}
}
</script>
</head>
<body>

<input type="button" onclick="show_confirm()" value="Show confirm
box" />

</body>
</html>
```

JavaScript Functions

To keep the browser from executing a script when the page loads, you can put your script into a function.

A function contains code that will be executed by an event or by a call to the function.

You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file).

Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

How to Define a Function

Syntax

```
function functionname(var1,var2,...,varX)  
{  
  some code  
}
```

The parameters *var1*, *var2*, etc. are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name.

Note: Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

JavaScript Function Example

Example

```
<html>  
<head>  
<script type="text/javascript">  
function displaymessage()  
{  
  alert("Hello World!");  
}
```

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

If the line: `alert("Hello world!!")` in the example above had not been put within a function, it would have been executed as soon as the line was loaded. Now, the script is not executed before a user hits the input button. The function `displaymessage()` will be executed if the input button is clicked.

You will learn more about JavaScript events in the JS Events chapter.

The return Statement

The return statement is used to specify the value that is returned from the function.

So, functions that are going to return a value must use the return statement.

The example below returns the product of two numbers (a and b):

Example

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b;
}
</script>
</head>

<body>
<script type="text/javascript">
document.write(product(5,3));
</script>

</body>
</html>
```

JavaScript Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript, there are two different kind of loops:

- **for** - loops through a block of code a specified number of times
- **while** - loops through a block of code while a specified condition is true

The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
code to be executed
}
```

Example

The example below defines a loop that starts with $i=0$. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs.

Note: The increment parameter could also be negative, and the \leq could be any comparing statement.

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
document.write("The number is " + i);
document.write("<br />");
}
</script>
</body>
</html>
```

The while Loop

The while loop loops through a block of code while a specified condition is true.

Syntax

```
while (var<=endvalue)
{
  code to be executed
}
```

Note: The <= could be any comparing statement.

Example

The example below defines a loop that starts with `i=0`. The loop will continue to run as long as `i` is less than, or equal to 5. `i` will increase by 1 each time the loop runs:

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5)
{
  document.write("The number is " + i);
  document.write("<br />");
  i++;
}
</script>
</body>
</html>
```

The do...while Loop

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

Syntax

```
do
{
  code to be executed
}
while (var<=endvalue);
```

Example

The example below uses a do...while loop. The do...while loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested:

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
do
{
  document.write("The number is " + i);
  document.write("<br />");
  i++;
}
while (i<=5);
</script>
</body>
</html>
```

JavaScript For...In Statement

The for...in statement loops through the elements of an array or through the properties of an object.

Syntax

```
for (variable in object)
{
  code to be executed
}
```

Note: The code in the body of the for...in loop is executed once for each element/property.

Note: The variable argument can be a named variable, an array element, or a property of an object.

Example

Use the for...in statement to loop through an array:

Example

```
<html>
<body>

<script type="text/javascript">
var x;
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";

for (x in mycars)
{
  document.write(mycars[x] + "<br />");
}
</script>
```

```
</body>  
</html>
```

Events

By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.

Every element on a web page has certain events which can trigger a JavaScript. For example, we can use the `onClick` event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags.

Examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input field in an HTML form
- Submitting an HTML form
- A keystroke

Note: Events are normally used in combination with functions, and the function will not be executed before the event occurs!

For a complete reference of the events recognized by JavaScript, go to our complete [JavaScript reference](#).

onLoad and onUnload

The `onLoad` and `onUnload` events are triggered when the user enters or leaves the page.

The `onLoad` event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

Both the `onLoad` and `onUnload` events are also often used to deal with cookies that should be set when a user enters or leaves a page. For

example, you could have a popup asking for the user's name upon his first arrival to your page. The name is then stored in a cookie. Next time the visitor arrives at your page, you could have another popup saying something like: "Welcome John Doe!".

onFocus, onBlur and onChange

The onFocus, onBlur and onChange events are often used in combination with validation of form fields.

Below is an example of how to use the onChange event. The checkEmail() function will be called whenever the user changes the content of the field:

```
<input type="text" size="30" id="email" onchange="checkEmail()">
```

onSubmit

The onSubmit event is used to validate ALL form fields before submitting it.

Below is an example of how to use the onSubmit event. The checkForm() function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be cancelled. The function checkForm() returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:

```
<form method="post" action="xxx.htm" onsubmit="return  
checkForm()">
```

onMouseOver and onMouseOut

onMouseOver and onMouseOut are often used to create "animated" buttons.

Below is an example of an onmouseover event. An alert box appears when an onmouseover event is detected:

```
<a href="http://www.w3schools.com" onmouseover="alert('An  
onmouseover event');return false">
```

JavaScript Objects Introduction

JavaScript is an Object Oriented Programming (OOP) language.

An OOP language allows you to define your own objects and make your own variable types.

Object Oriented Programming

JavaScript is an Object Oriented Programming (OOP) language. An OOP language allows you to define your own objects and make your own variable types.

However, creating your own objects will be explained later, in the Advanced JavaScript section. We will start by looking at the built-in JavaScript objects, and how they are used. The next pages will explain each built-in JavaScript object in detail.

Note that an object is just a special kind of data. An object has properties and methods.

Properties

Properties are the values associated with an object.

In the following example we are using the length property of the String object to return the number of characters in a string:

```
<script type="text/javascript">  
var txt="Hello World!";  
document.write(txt.length);  
</script>
```

The output of the code above will be:

12

Methods

Methods are the actions that can be performed on objects.

In the following example we are using the `toUpperCase()` method of the `String` object to display a text in uppercase letters:

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.toUpperCase());
</script>
```

The output of the code above will be:

HELLO WORLD!

Math Object

The **Math** object's methods allow the programmer to perform many common mathematical calculations.

An object's methods are called by writing the name of the object followed by a dot operator (`.`) and the name of the method.

`document.writeln(Math.sqrt(900.0));`
the output would be 30

Some **Math** object methods are summarized in Figure below

Method	Description	Example
<code>abs(x)</code>	absolute value of x	<code>abs(7.2)</code> is 7.2 <code>abs(0.0)</code> is 0.0 <code>abs(-5.6)</code> is 5.6
<code>ceil(x)</code>	rounds x to the smallest integer not less than x	<code>ceil(9.2)</code> is 10.0 <code>ceil(-9.8)</code> is -9.0
<code>cos(x)</code>	trigonometric cosine of x (x in radians)	<code>cos(0.0)</code> is 1.0
<code>exp(x)</code>	exponential method e^x	<code>exp(1.0)</code> is 2.71828 <code>exp(2.0)</code> is 7.38906
<code>floor(x)</code>	rounds x to the largest integer not greater than x	<code>floor(9.2)</code> is 9.0 <code>floor(-9.8)</code> is -10.0
<code>log(x)</code>	natural logarithm of x (base e)	<code>log(2.718282)</code> is 1.0 <code>log(7.389056)</code> is 2.0
<code>max(x, y)</code>	larger value of x and y	<code>max(2.3, 12.7)</code> is 12.7 <code>max(-2.3, -12.7)</code> is -2.3
<code>min(x, y)</code>	smaller value of x and y	<code>min(2.3, 12.7)</code> is 2.3 <code>min(-2.3, -12.7)</code> is -12.7
<code>pow(x, y)</code>	x raised to power y (x^y)	<code>pow(2.0, 7.0)</code> is 128.0 <code>pow(9.0, .5)</code> is 3.0
<code>round(x)</code>	rounds x to the closest integer	<code>round(9.75)</code> is 10 <code>round(9.25)</code> is 9
<code>sin(x)</code>	trigonometric sine of x (x in radians)	<code>sin(0.0)</code> is 0.0
<code>sqrt(x)</code>	square root of x	<code>sqrt(900.0)</code> is 30.0 <code>sqrt(9.0)</code> is 3.0
<code>tan(x)</code>	trigonometric tangent of x (x in radians)	<code>tan(0.0)</code> is 0.0

String object

The String object is used to manipulate a stored piece of text.

Examples of use:

The following example uses the length property of the String object to find the length of a string:

```
var txt="Hello world!";  
document.write(txt.length);
```

The code above will result in the following output:

```
12
```

The following example uses the toUpperCase() method of the String object to convert a string to uppercase letters:

```
var txt="Hello world!";  
document.write(txt.toUpperCase());
```

The code above will result in the following output:

```
HELLO WORLD!
```

Character Processing Methods

The script of Fig. 12.4 demonstrates some of the String object's character processing methods, including charAt (returns the character at a specific position), charCodeAt (returns the Unicode value of the character at a specific position), fromCharCode (returns a string created from a series of Unicode values), toLowerCase (returns the lowercase version of a string) and toUpperCase (returns the uppercase version of a string).

```
1 <?xml version = "1.0"?>  
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0  
Strict//EN"  
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
45  
<!-- Fig. 12.4: CharacterProcessing.html -->
```

```

6 <!-- Character Processing Methods -->
78
<html xmlns = "http://www.w3.org/1999/xhtml">
9 <head>
10 <title>Character Processing Methods</title>
11
12 <script type = "text/javascript">
13 <!--
14 var s = "ZEBRA";
15 var s2 = "AbCdEfG";
17 document.writeln( "<p>Character at index 0 in '" +
18 s + "' is " + s.charAt( 0 ) );
19 document.writeln( "<br />Character code at index 0 in '"
20 + s + "' is " + s.charCodeAt( 0 ) + "</p>" );
21
22 document.writeln( "<p>" +
23 String.fromCharCode( 87, 79, 82, 68 ) +
24 "' contains character codes 87, 79, 82 and 68</p>" )
25
26 document.writeln( "<p>" + s2 + "' in lowercase is '" +
27 s2.toLowerCase() + "'" );
28 document.writeln( "<br />" + s2 + "' in uppercase is '"
29 + s2.toUpperCase() + "'</p>" );
30 // -->
31 </script>
32
33 </head><body></body>
34 </html>

```

