JAVASCRIPT

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

What can a JavaScript do?

- JavaScript gives HTML designers a programming tool
- JavaScript can put dynamic text into an HTML page
- JavaScript can react to events
- JavaScript can read and write HTML elements
- JavaScript can be used to validate data
- JavaScript can be used to detect the visitor's browser
- JavaScript can be used to create cookies

JavaScript - Client-Side Scripting

- JavaScript gives HTML designers a programming tool
- JavaScript can put dynamic text into an HTML page
- JavaScript can react to events
- JavaScript can change HTML elements
- JavaScript can be used to validate data

Server-Side Scripting

- Dynamically edit, change, or add any content of a Web page
- Respond to user queries and form data
- Access databases and return the result to a browser
- Access files and return the result to a browser
- Transform XML data to HTML data and return the results to a browser
- Customize a Web page to make it more useful for individual users
- Provide security and access control to Web pages
- Tailor your output to different types of browsers
- Minimize network traffic

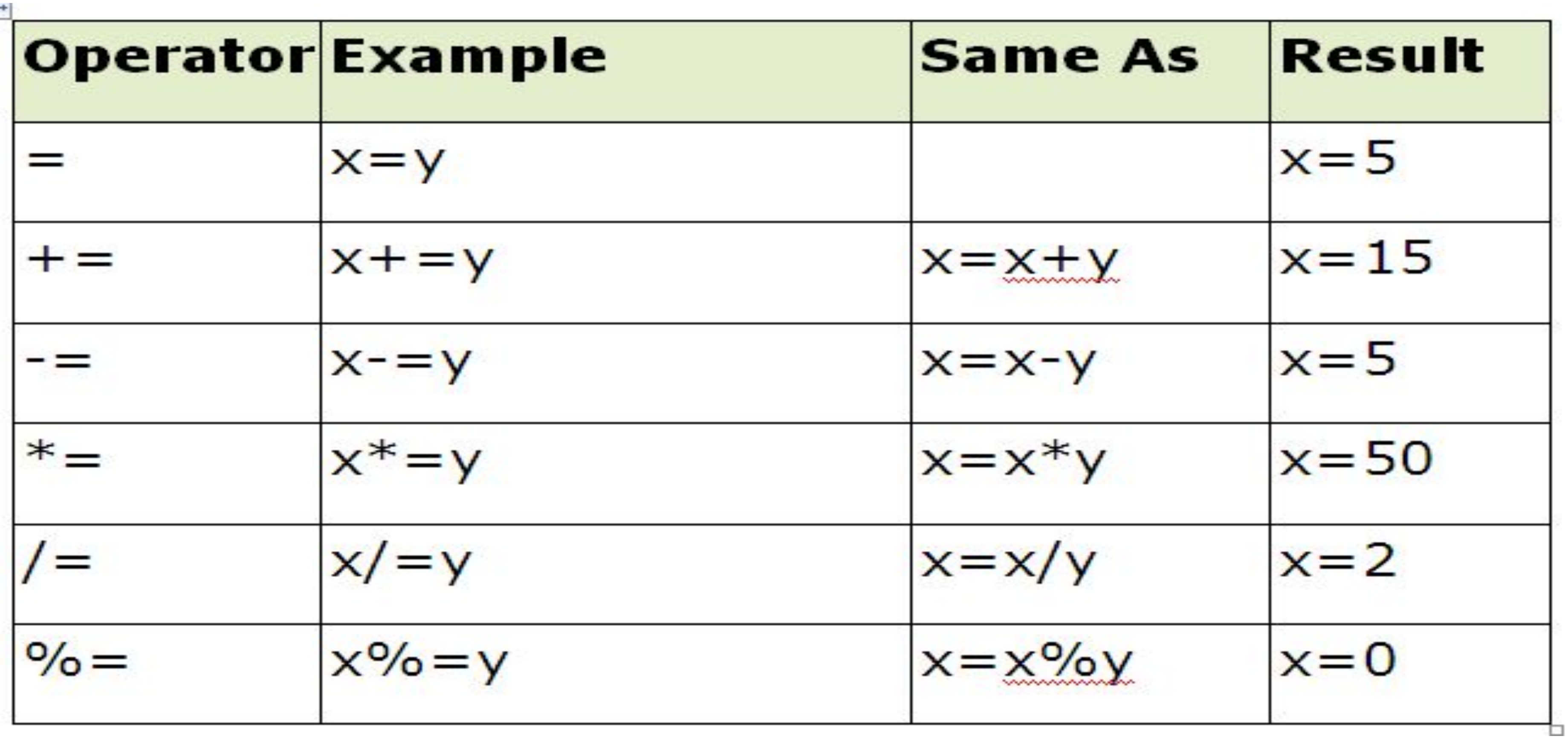
JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values.
Given that y=5, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables.
Given that x=10 and y=5, the table below explains the assignment operators:



Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings

```
txt1="What a very";
txt2="nice day";
txt3=txt1+″ "+txt2;
```

```
What a very nice day
```

5+3	**=8**
“5”+”3”	**=53**
“5”+3	**=53**

Adding Strings and Numbers

The rule is: **If you add a number and a string, the result will be a string!**

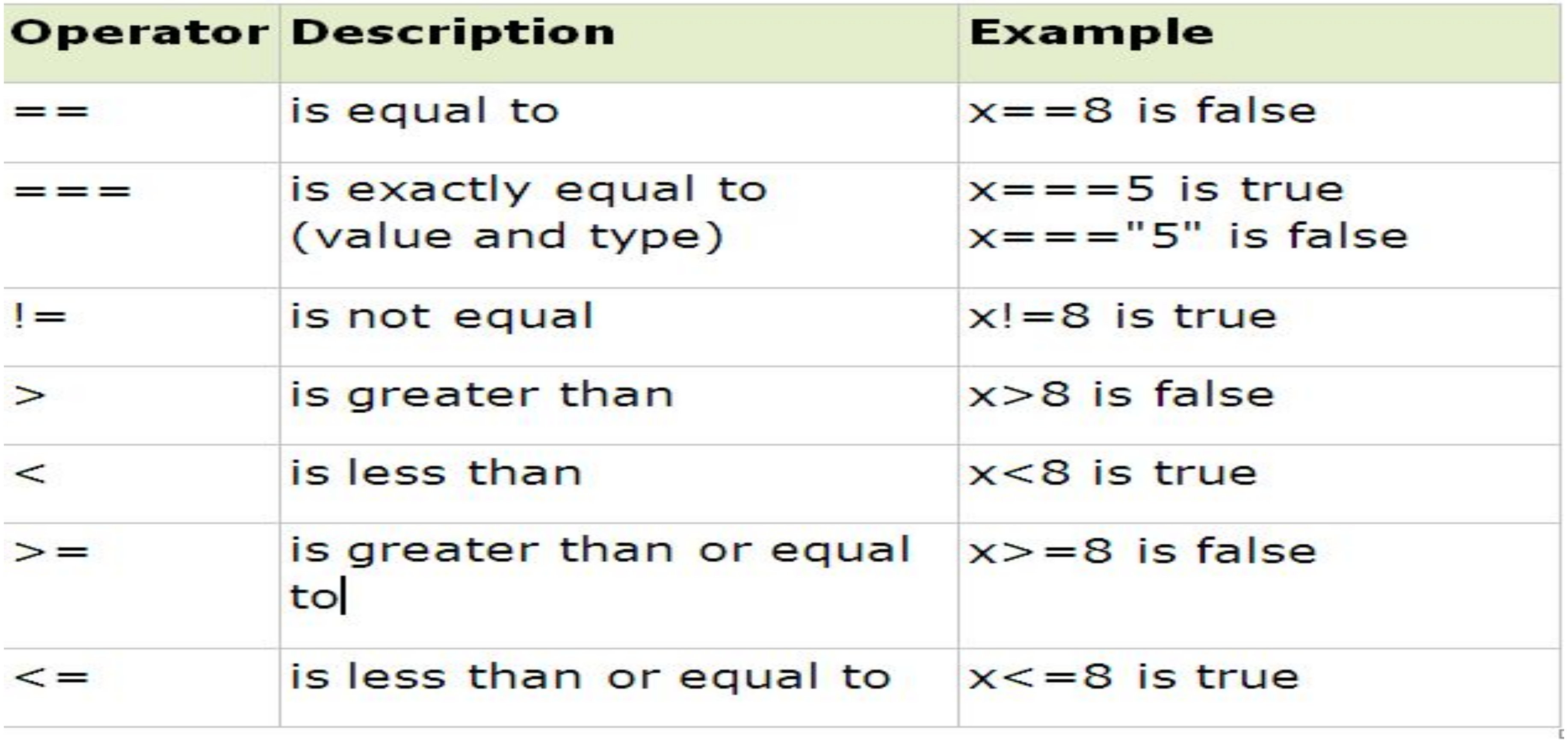
```
x=5+5;
document.write(x);

x="5"+"5";
document.write(x);

x=5+"5";
document.write(x);

x="5"+5;
document.write(x);
```

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.

if (age<18) document.write("Too young");

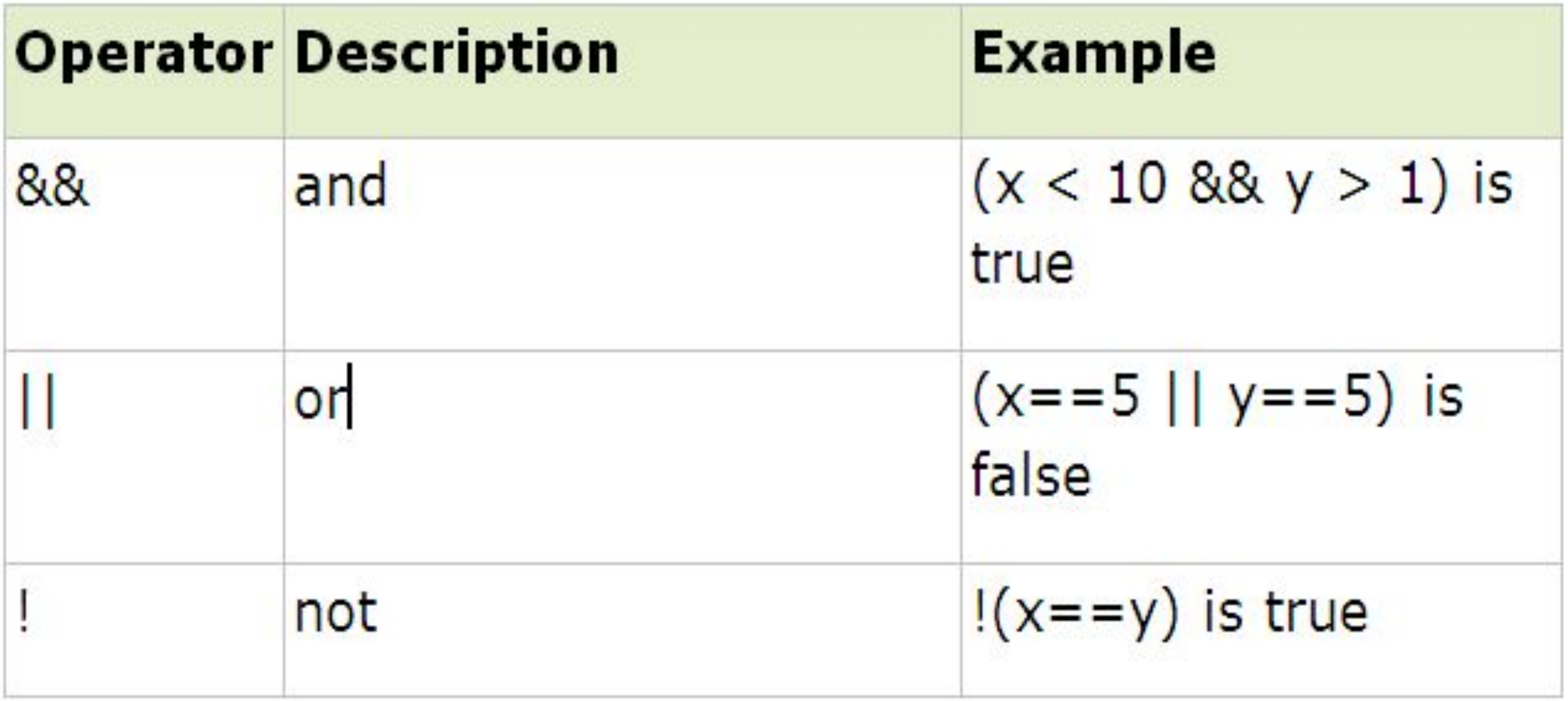
Given that x=5, the table below explains the comparison operators:



Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="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

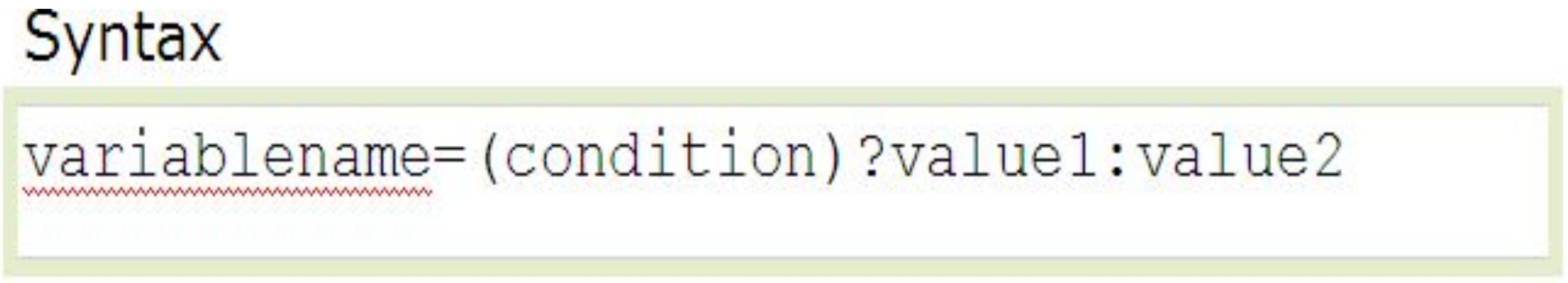
Logical operators are used to determine the logic between variables or values.
Given that x=6 and y=3, the table below explains the logical operators:



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

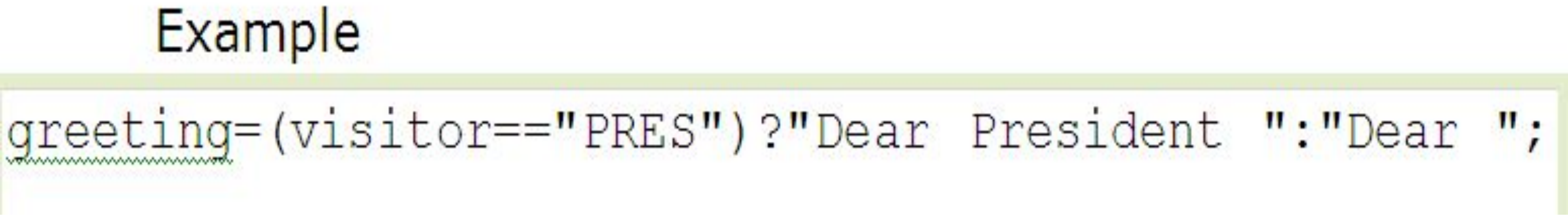
Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.



Syntax

```
variablename=(condition)?value1:value2
```



Example

```
greeting=(visitor=="PRES")?"Dear President ":"Dear ";
```

Conditional Statements

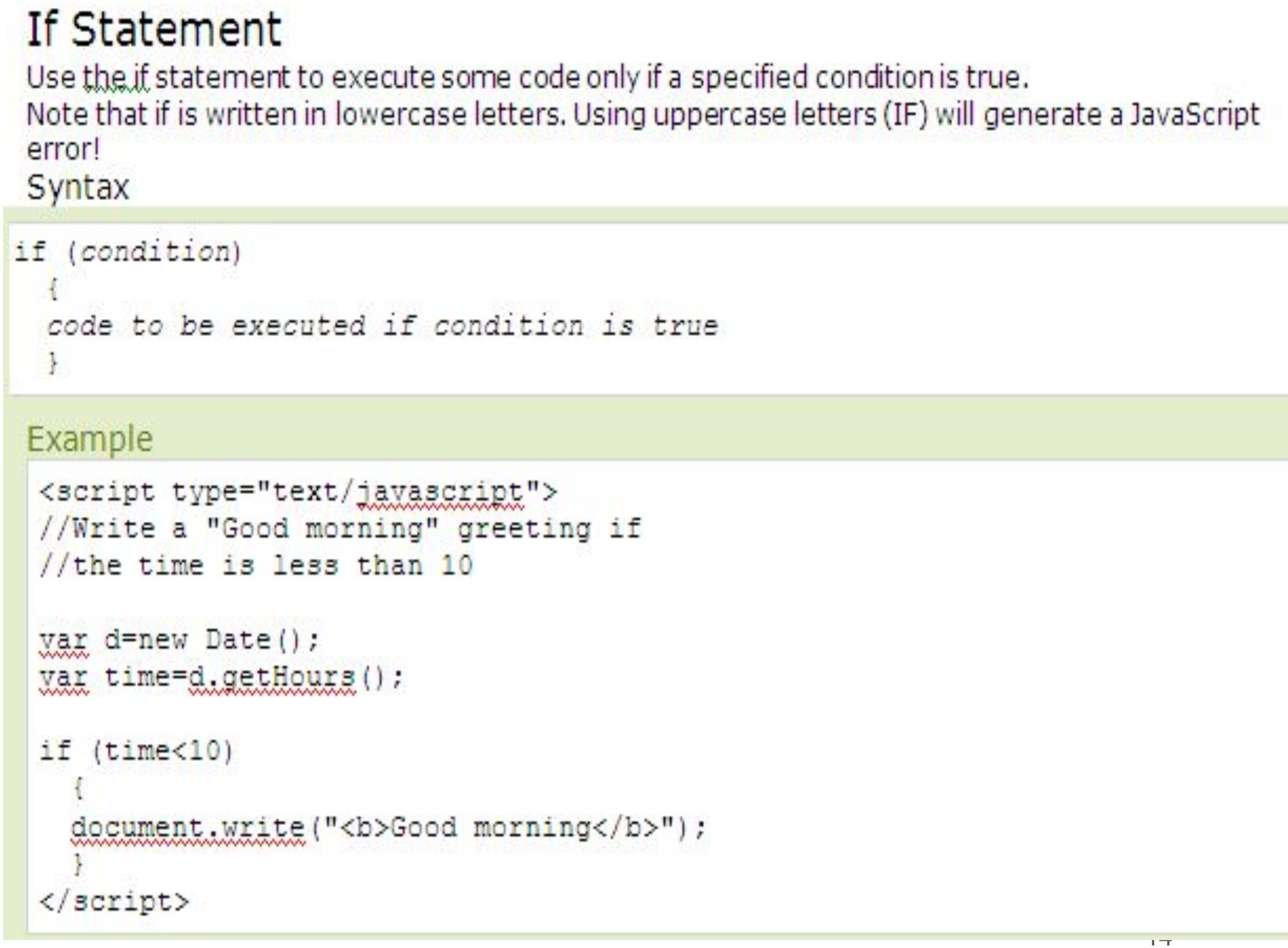
In JavaScript we have the following conditional statements:

if statement - use this statement to execute some code only if a specified condition is true

if...else statement - use this statement to execute some code if the condition is true and another code if the condition is false

if...else if....else statement - use this statement to select one of many blocks of code to be executed

switch statement - use this statement to select one of many blocks of code to be executed



If Statement

Use the if statement to execute some code only if a specified condition is true.
Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

Syntax

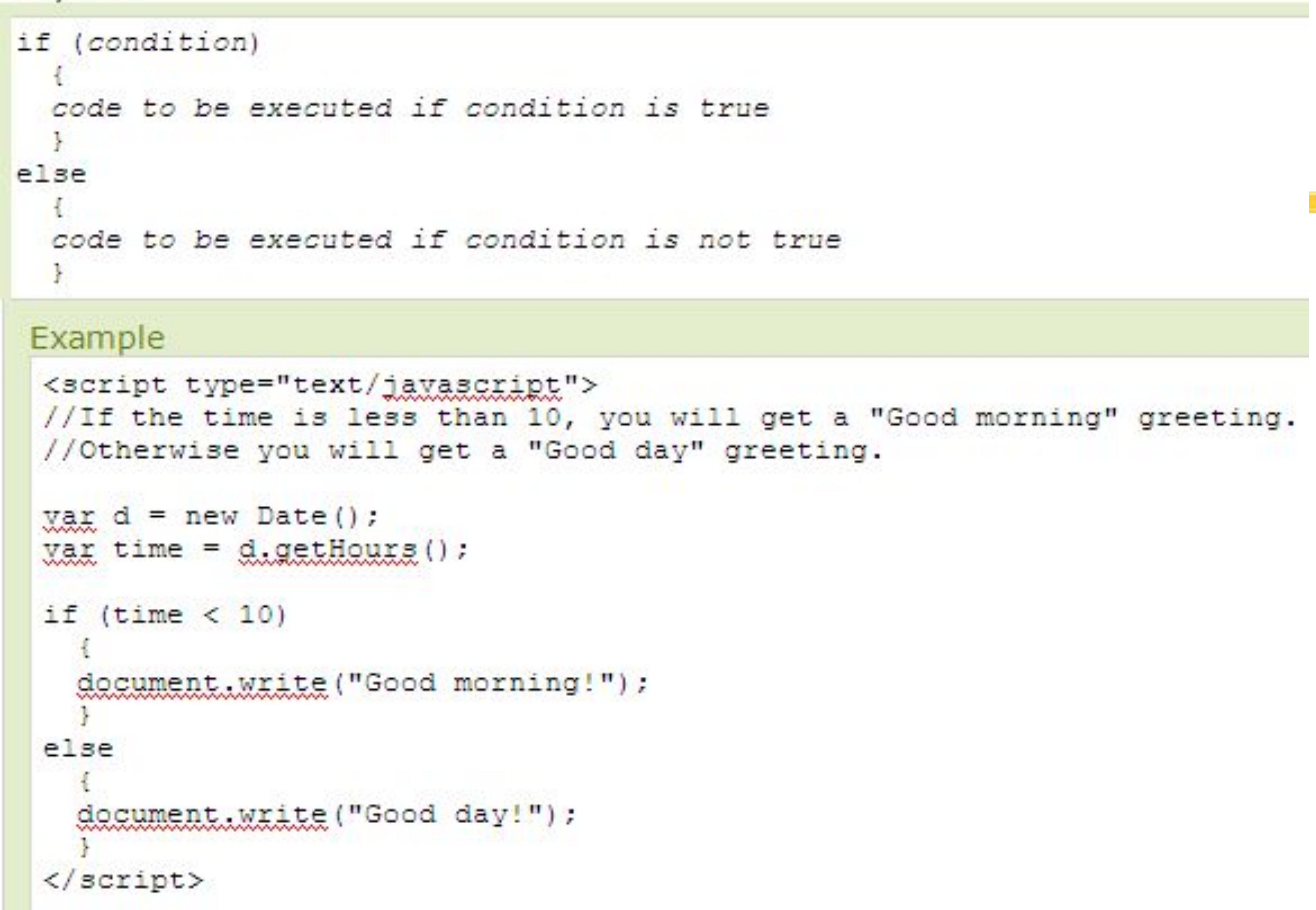
```
if (condition)
  {
  code to be executed if condition is true
  }
```

Example

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10

var d=new Date();
var time=d.getHours();

if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
</script>
```

If...else Statement

Syntax



```
if (condition)
  {
  code to be executed if condition is true
  }
else
  {
  code to be executed if condition is not true
  }
```

Example

```
<script type="text/javascript">
//If the time is less than 10, you will get a "Good morning" greeting.
//Otherwise you will get a "Good day" greeting.

var d = new Date();
var time = d.getHours();

if (time < 10)
  {
  document.write("Good morning!");
  }
else
  {
  document.write("Good day!");
  }
</script>
```

If....else if...else Statement

Syntax

```
if (condition1)
  {
  code to be executed if condition1 is true
  }
else if (condition2)
  {
  code to be executed if condition2 is true
  }
else
  {
  code to be executed if condition1 and condition2 are not true
  }
```

Example

```
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
else if (time>10 && time<16)
  {
  document.write("<b>Good day</b>");
  }
else
  {
  document.write("<b>Hello World!</b>");
  }
</script>
```

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

```
<html><body><p id="demo"> </p><script>
var day;
switch (new Date().getDay()) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case  6:
    day = "Saturday";
}
document.getElementById("demo").innerHTML = "Today is " + day;
</script></body></html>
```

```
Var d=new Date()
Var time=d.getHours()
```

JavaScript Comments

Single line comments start with //.

```
<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:

```
<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>
```

The While Loop

```
while (condition)
 {
 code block to be executed
 }
<html>
<body>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
Var i=0;
while (i<5)
 {
Document.write(i)
Document.write("<br>");
i++;
 }
}
</script>
</body>
</html>
```

The While Loop

```
do
  {
  code block to be executed
  }
while (condition);
```

```
i=0;
do
  {
  x=x + "The number is " + i + "<br>";
  i++;
  }
while (i<5);
Alert(x)
```

The Break Statement

```
for (i=0;i<10;i++)
  {
  if (i==3)
    {
    break;
    }
  x=x + "The number is " + i + "<br>";
  }
```

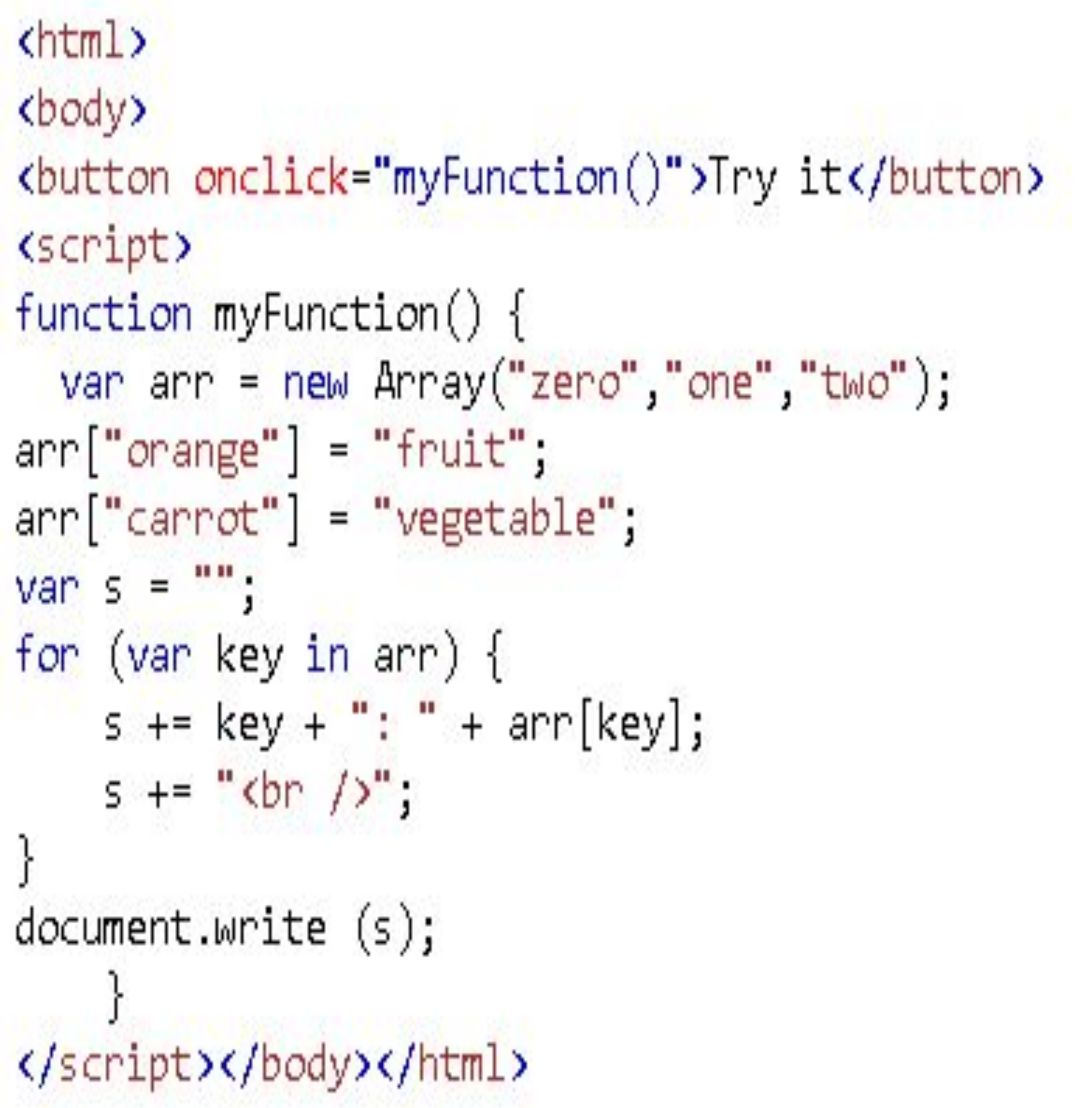
The Continue Statement

```
for (i=0;i<=10;i++)
 {
 if (i==3) continue;
  x=x + "The number is " + i + "<br>";
  }
```

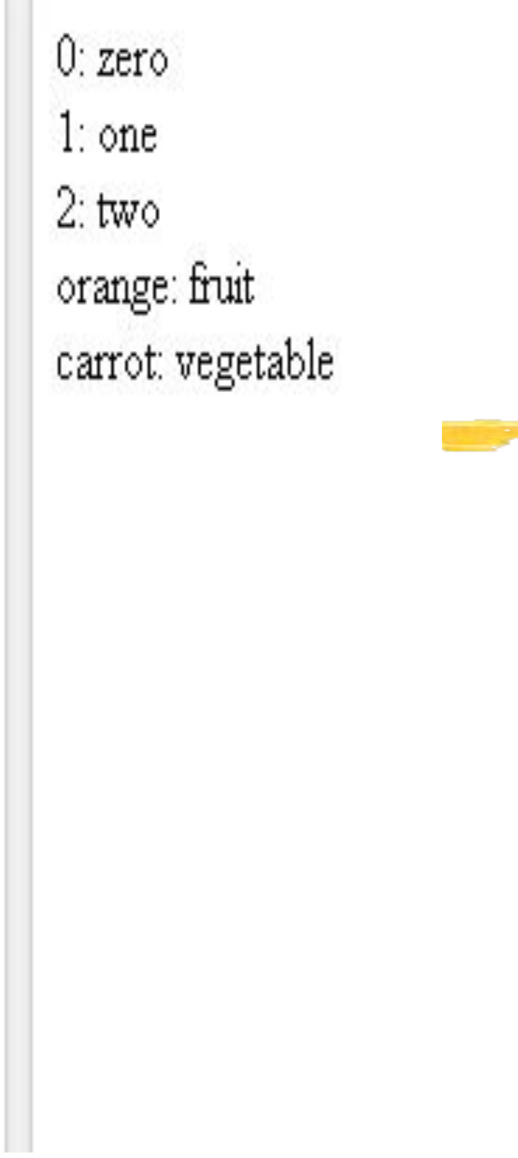
Javascript for…in statement

```
<html>
<body>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
var x;
var txt="";
var person={fname:"John",lname:"Doe",age:25};
for (x in person)
{
txt=txt + person[x];
}
document.write(txt)
}
</script>
</body>
</html>
```

```
var string1 = "";
var object1 = {a: 1, b: 2, c: 3};
for (var property1 in object1)
{
string1 = string1 + object1[property1];
}
document.write(string1);


// expected output: "123"
```



```
<html>
<body>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction() {
  var arr = new Array("zero","one","two");
arr["orange"] = "fruit";
arr["carrot"] = "vegetable";
var s = "";
for (var key in arr) {
    s += key + ": " + arr[key];
    s += "<br />";
}
document.write (s);
    }
</script></body></html>
```



```
0: zero
1: one
2: two
orange: fruit
carrot: vegetable
```

VAR STATEMENT

If it is used in a function,the scope is confined to that function.

If used outside of a function,it can be accessed anywhere on the page

```
<html>
<body>
<script type="text/javascript">
var firstname;
firstname="Hege";
document.write(firstname);
document.write("<br />");
firstname="Tove";
document.write(firstname);
</script>
<p>The script above declares a variable,
assigns a value to it, displays the value, changes the value,
and displays the value again.</p>
</body></html>
```

With Statement

```
With (object){
Code;
}
<html><body>
<script type="text/javascript">
document.write(Math.round(0.60) + "<br />");
document.write(Math.round(0.50) + "<br />");
document.write(Math.round(0.49) + "<br />");
</script></body></html>

<html><body>
<script type="text/javascript">
with (Math)
{
document.write(round(0.60) + "<br />");
document.write(round(0.50) + "<br />");
document.write(round(0.49) + "<br />");
}
</script></body></html>
```

Labelled

Any Javascript identifier that is not a reserved word

Eg:

```
test1: for(var i=0;i<3;i++)
test2: for(var j=0;j<3;j++)
If(i==1 &&j==1){
continue test1;
}
```

Delete

The delete statement an object that was created using the new statement.

delete myobject;

The new statement

The new statement is the way that new objects are created in Javascript

The following is a function to create a house object

```
Function house(rms,stl,yr,garp){
this.room=rms;
this.style=stl;
this..yearBuilt=yr;
this.hasGarage=garp;
}
```

You could then create an instance of a house object by using the new statement

```
Var myhouse=new house(3,’Tenement’,1962,false);
```

The this statement

The this ststement refers to the current object.
Syntax   this.property
Eg. If setSize is a method of the document ,this refers to the specific object whose setSize method is called.

```
Function setSize(x,y)
{
this.horizsize=x;
this.vertSize=y;
}
```

This method sets the size for an object when called as follows:

```
document.setSize(640,480);
```

Comma Operator

The comma allows multiple statements to be executed as one statement
Syntax: ststement1,statement2,statement3

```
<script langusge=”Javascript”>
<!—
X=(y=3,z=9);
Document.write (“z=”,z,”y=”,y);
- >
</script></html>
```

Javascript Objects

- A JavaScript object is a collection of named values

Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

- All cars have the same properties, but the property values differ from car to car

- All cars have the same methods, but the methods are performed at different times

Object Oriented Programming

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

Properties

Properties are the values associated with an object.

```
<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.

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

The output of the code above will be:12

```
HELLO WORLD!
```

Creating JavaScript Objects

Method 1:Creating a Direct Instance

- person=new Object();
  person.firstname="John";
  person.lastname="Doe";
  person.age=50;
  person.eyecolor="blue";

Method 2:Using object literals

- person={firstname:"John",lastname:"Doe",age:50,eyecolor:"blue"};

Method 3:Using an Object Constructor (To create an "object type“)

- function person(firstname,lastname,age,eyecolor)
  {
  this.firstname=firstname;
  this.lastname=lastname;
  this.age=age;
  this.eyecolor=eyecolor;
  }

```
<html><body>
<p>Creating a JavaScript Object.</p>
<p id="demo"></p>
<script>
var person = {
  firstName : "John",
  lastName  : "Doe",
  age       : 50,
  eyeColor  : "blue"
};

document.getElementById("demo").innerHTML = person.firstName
+ " " + person.lastName;

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

```
<html><body><p id="demo"></p>

<script>
var person = new Object();
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";

document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>

</body>
</html>
```

```
<html><body><h2>JavaScript Object Constructors</h2>
<p id="demo"></p>
<script>
// Constructor function for Person objects
function Person(first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eye;
}
// Create 2 Person objects
var myFather = new Person("John", "Doe", 50, "blue");
var myMother = new Person("Sally", "Rally", 48, "green");
// Add a name method to first object
myFather.name = function() {
  return this.firstName + " " + this.lastName;
};
// Display full name
document.getElementById("demo").innerHTML =
"My father is " + myFather.firstName;

document.write ("My father is " +myFather.name());
</script></body></html>
```

```
<html><body>
<h2>JavaScript Object Constructors</h2>
<p id="demo"></p>
<script>
// Constructor function for Person objects
function Person(firstName,lastName,age,eyeColor) {
  this.firstName = firstName;
  this.lastName = lastName;
  this.age = age;
  this.eyeColor = eyeColor;
  this.Name = function () {
    return this.firstName+" " +this.lastName;
  }
}
// Create a Person object
var myMother = new Person("Sally","Rally",48,"green");

// Display last name
document.getElementById("demo").innerHTML =
"My mother's last name is " + myMother.Name();

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

```
<html><body><h2>JavaScript Object Constructors</h2>
<p id="demo"></p>
<script>
// Constructor function for Person objects
function Person(firstName,lastName,age,eyeColor) {
  this.firstName = firstName;
  this.lastName = lastName;
  this.age = age;
  this.eyeColor = eyeColor;
  this.changeName = function (name) {
    this.lastName = name;
  }
}
// Create a Person object
var myMother = new Person("Sally","Rally",48,"green");
// Change last name
myMother.changeName("Doe");
// Display last name
document.getElementById("demo").innerHTML =
"My mother's last name is " + myMother.lastName;
</script></body></html>
```

```
<html><body><h2>JavaScript Object Constructors</h2>
<p id="demo"></p>
<script>
// Constructor function for Person objects
function Person(first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eye;
this.name = function() {
  return this.firstName + " " + this.lastName;
}
this.changeName=function(name1)  {
this.lastName=name1;
}
}
// Create 2 Person objects
var myFather = new Person("John", "Doe", 50, "blue");
var myMother = new Person("Sally", "Rally", 48, "green");

document.write("<br>My mother is " + myMother.firstName);
document.write("<br>My mother is " + myMother.lastName);
document.write ("<br>My mother is " + myMother.name());
myMother.changeName("Doe");
document.write ("<br>My mother is " + myMother.name());

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

Array

1: Regular:

- var myCars=new Array();
  myCars[0]="Saab";
  myCars[1]="Volvo";
  myCars[2]="BMW";

2: Condensed:

var myCars=new Array("Saab","Volvo","BMW");

3: Literal:

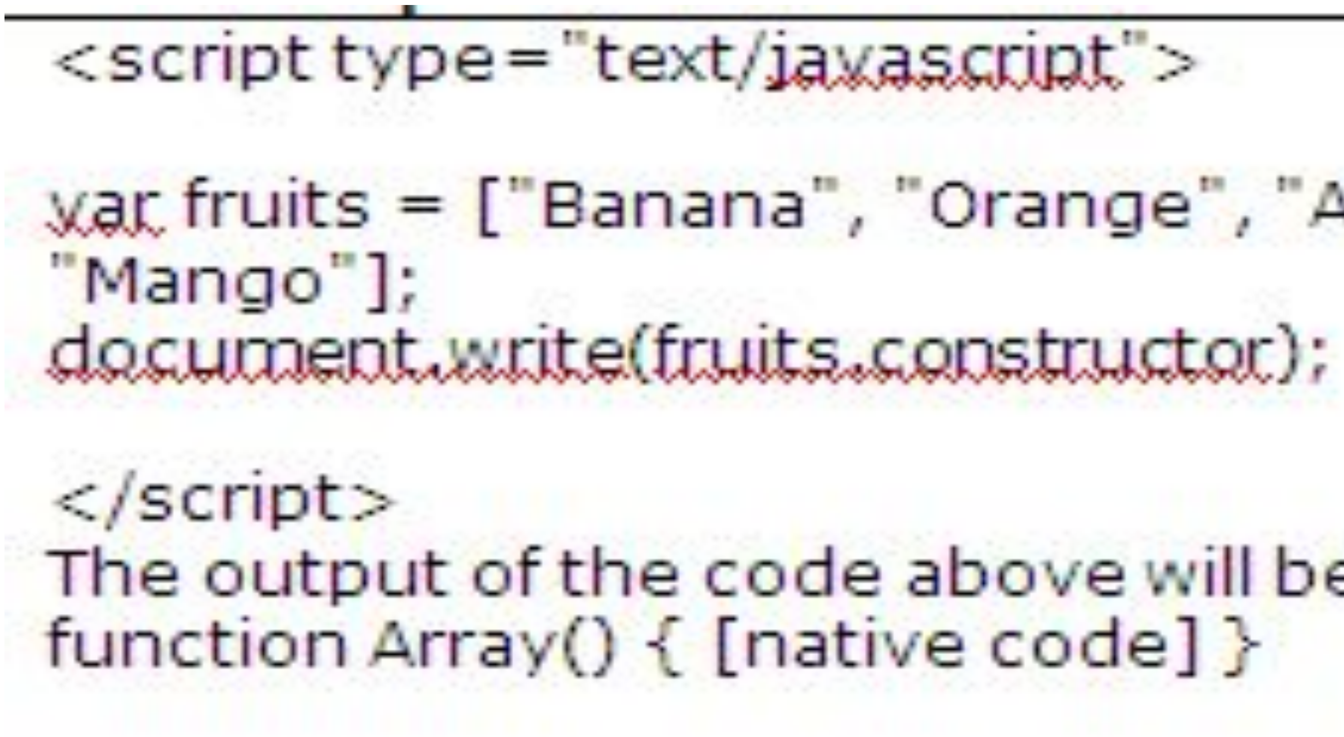
- var myCars=["Saab","Volvo","BMW"];

Array Object Properties

Property	Description
constructor	Returns the function that created the Array object's prototype
length	Sets or returns the number of elements in an array
prototype	Allows you to add properties and methods to an object

Array Object Methods

Method	Description
concat()	Joins two or more arrays, and returns a copy of the joined arrays
join()	Joins all elements of an array into a string
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element
slice()	Selects a part of an array, and returns the new array
sort()	Sorts the elements of an array
splice()	Adds/Removes elements from an array
toString()	Converts an array to a string, and returns the result
unshift()	Adds new elements to the beginning of an array, and returns the new length
valueOf()	Returns the primitive value of an array



```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.write(fruits.constructor);

</script>
```

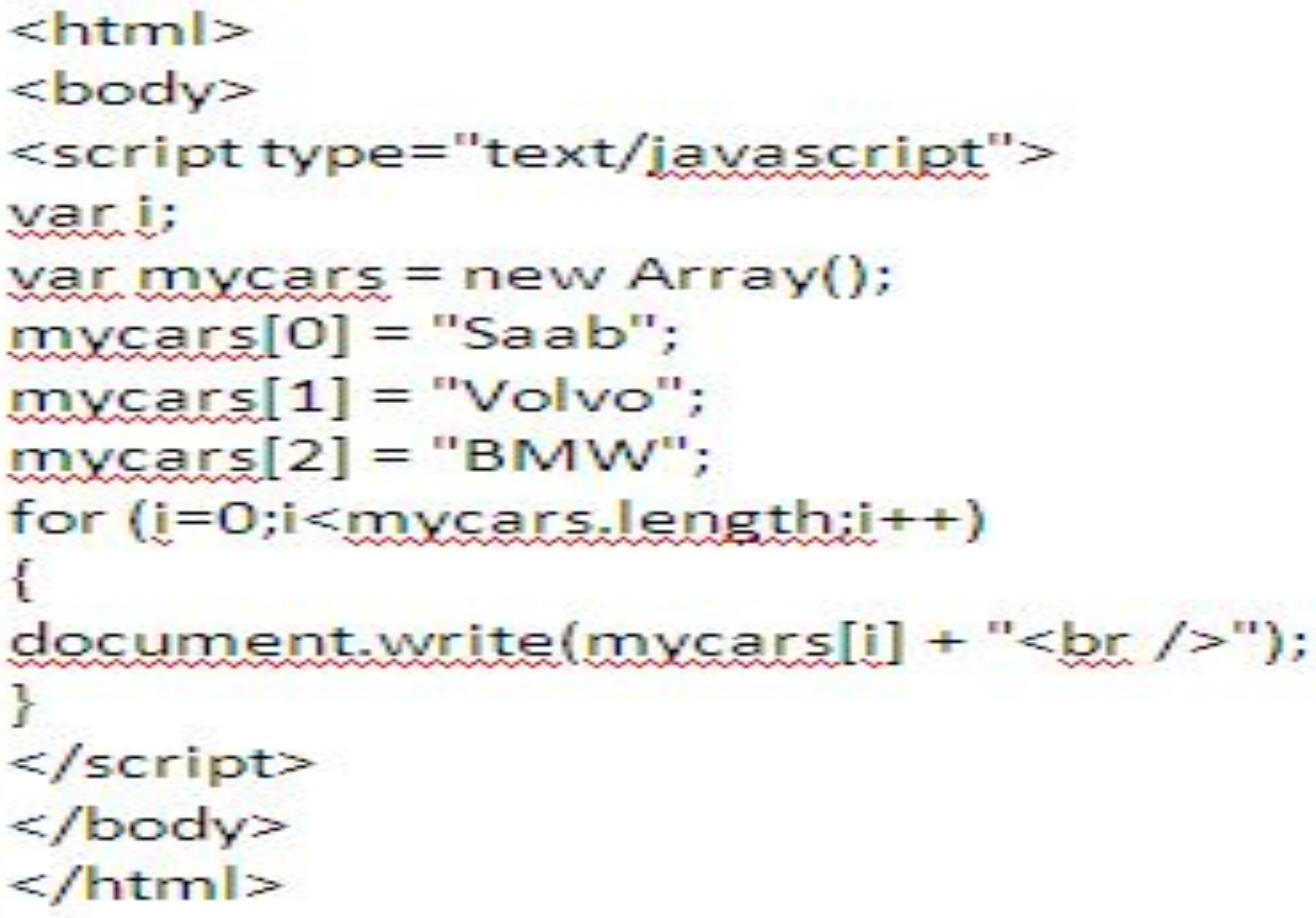
The output of the code above will be:

```
function Array() { [native code] }
```

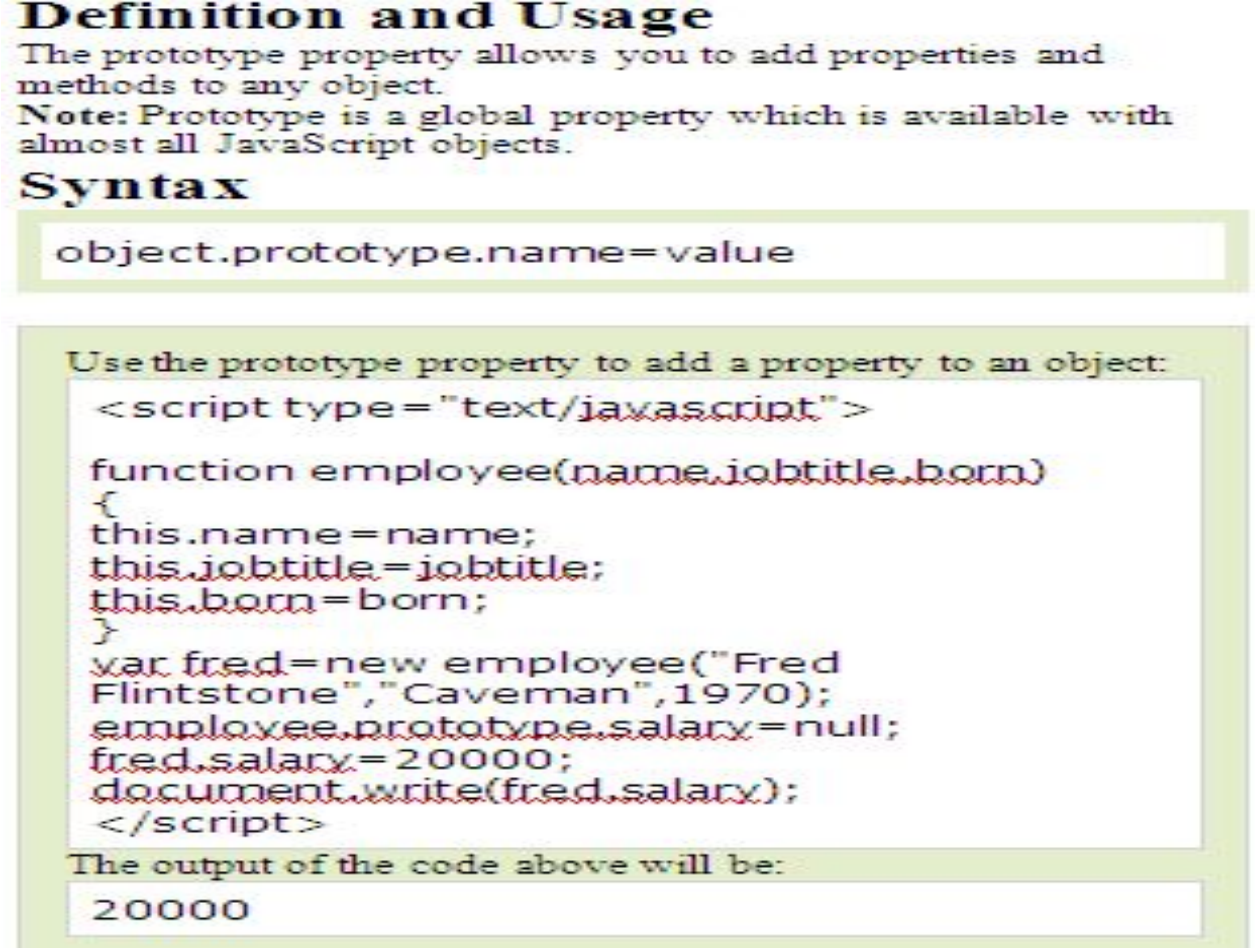
```
<html><body>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
Array.prototype.myUcase = function() {
  var i;
  for (i = 0; i < this.length; i++) {
 this[i] = this[i].toUpperCase();
  }
};


function myFunction() {
  var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.myUcase();
  document.getElementById("demo").innerHTML = fruits;
}
</script></body></html>
```

```
<html><body><h2>JavaScript Arrays</h2>
<p id="demo"></p>
<script>
var fruits, text, fLen, i;
fruits = ["Banana", "Orange", "Apple", "Mango"];
fLen = fruits.length;
text = "<ul>";
for (i = 0; i < fLen; i++) {
  text += "<li>" + fruits[i] + "</li>";
}
text += "</ul>";
document.getElementById("demo").innerHTML = text;
</script></body></html>
```

```
<ul><li>Banana</li><li>Orange</li>     </ul>
```



```
<html>
<body>
<script type="text/javascript">
var i;
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";
for (i=0;i<mycars.length;i++)
{
document.write(mycars[i] + "<br />");
}
</script>
</body>
</html>
```



Definition and Usage

The prototype property allows you to add properties and methods to any object.

Note: Prototype is a global property which is available with almost all JavaScript objects.

Syntax

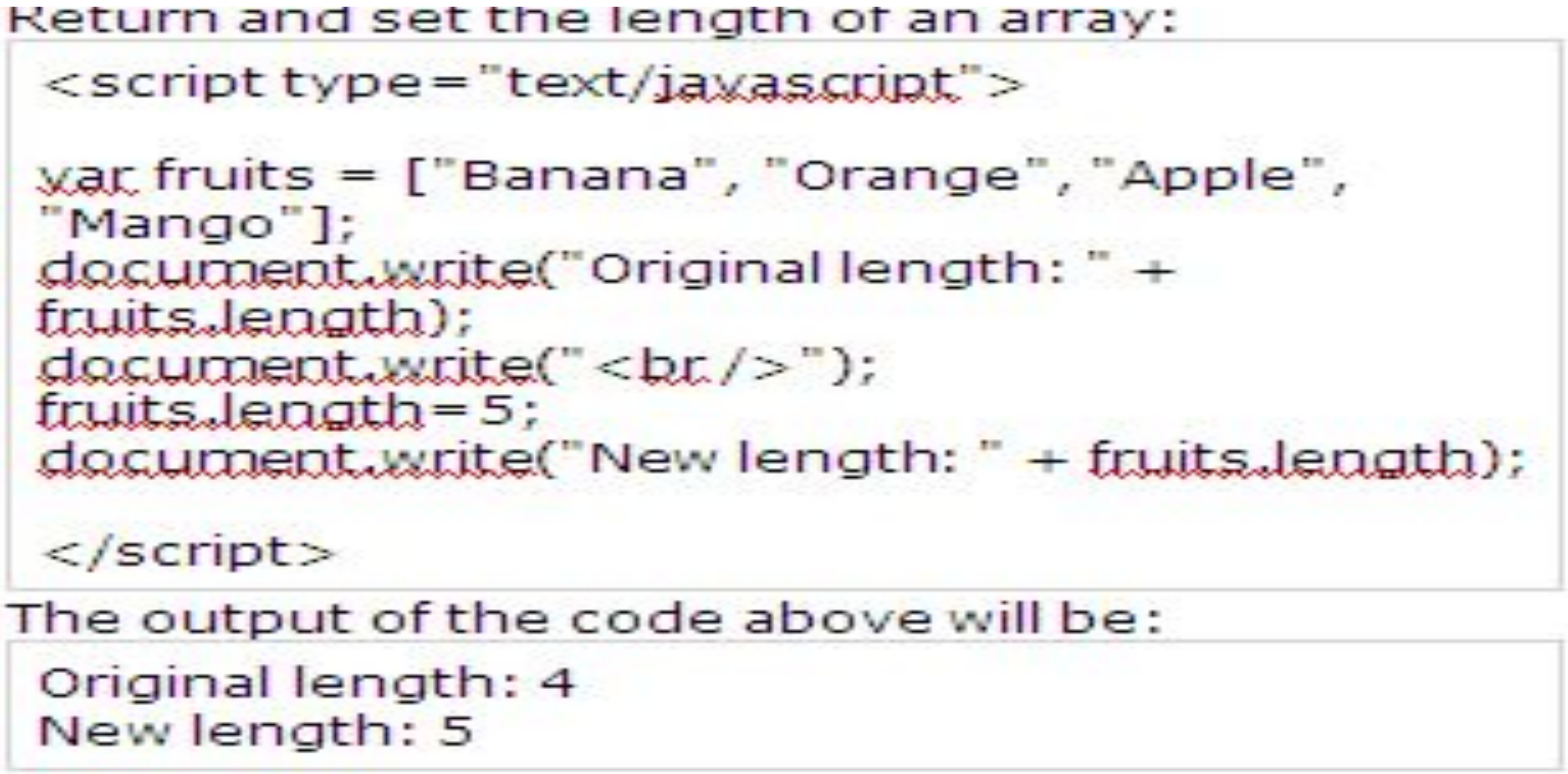
```
object.prototype.name=value
```

Use the prototype property to add a property to an object:

```
<script type="text/javascript">

function employee(name,jobtitle,born)
{
this.name=name;
this.jobtitle=jobtitle;
this.born=born;
}
var fred=new employee("Fred
Flintstone","Caveman",1970);
employee.prototype.salary=null;
fred.salary=20000;
document.write(fred.salary);
</script>
```

The output of the code above will be:

```
20000
```

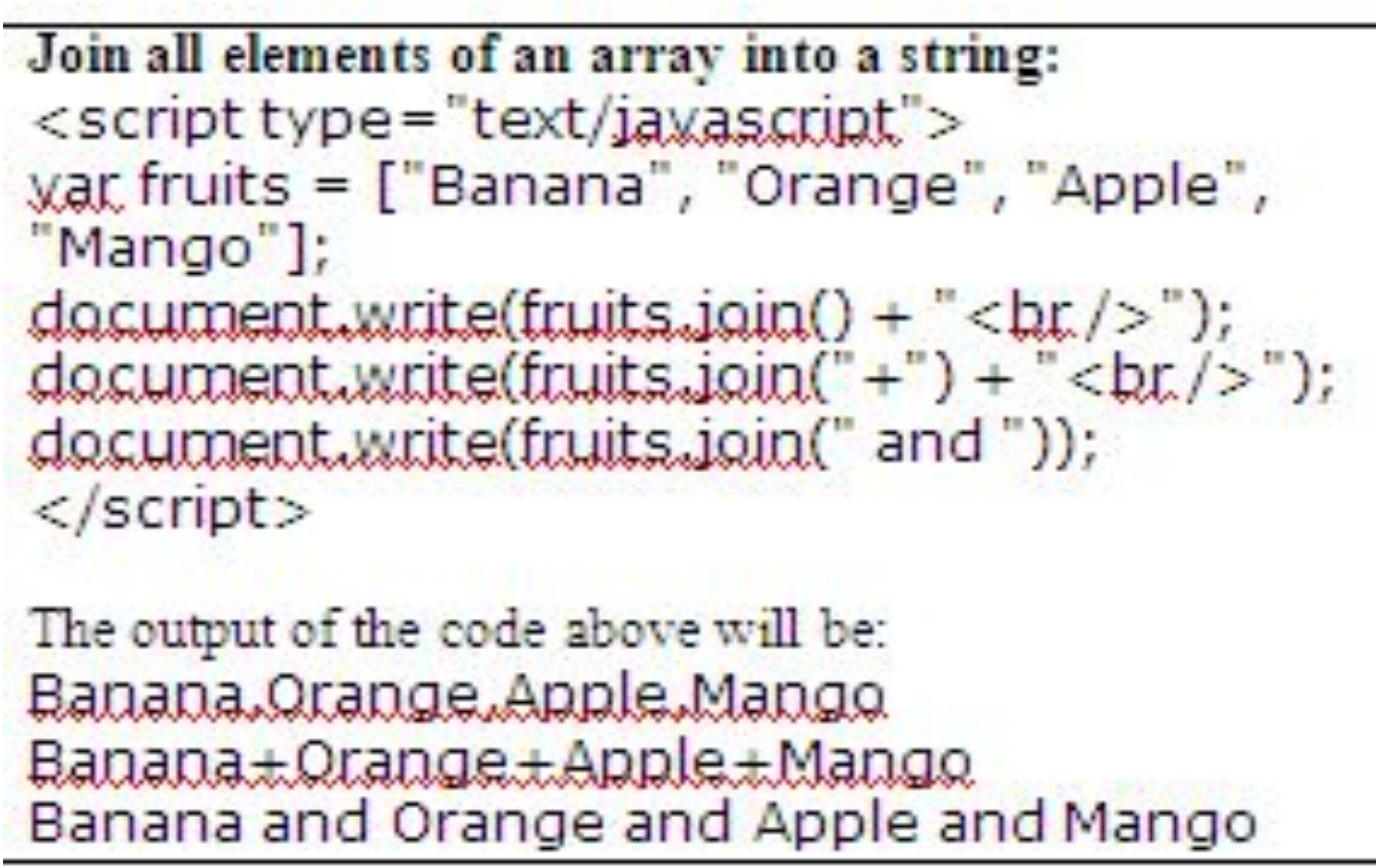


Return and set the length of an array:

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.write("Original length: " +
fruits.length);
document.write("<br />");
fruits.length=5;
document.write("New length: " + fruits.length);

</script>
```

The output of the code above will be:

```
Original length: 4
New length: 5
```

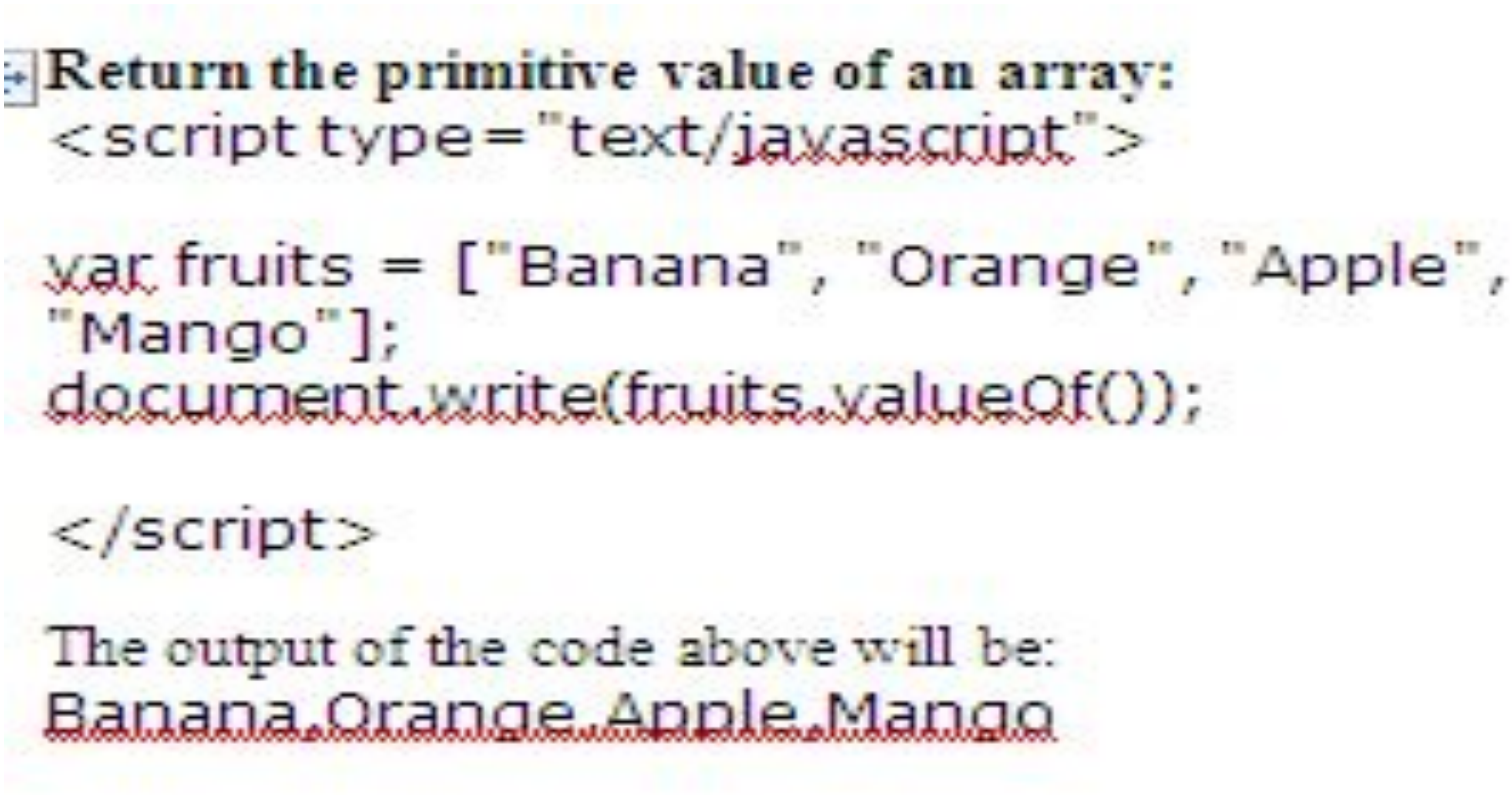


Join all elements of an array into a string:

```
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.write(fruits.join() + "<br />");
document.write(fruits.join("+") + "<br />");
document.write(fruits.join(" and "));
</script>
```

The output of the code above will be:

```
Banana,Orange,Apple,Mango
Banana+Orange+Apple+Mango
Banana and Orange and Apple and Mango
```

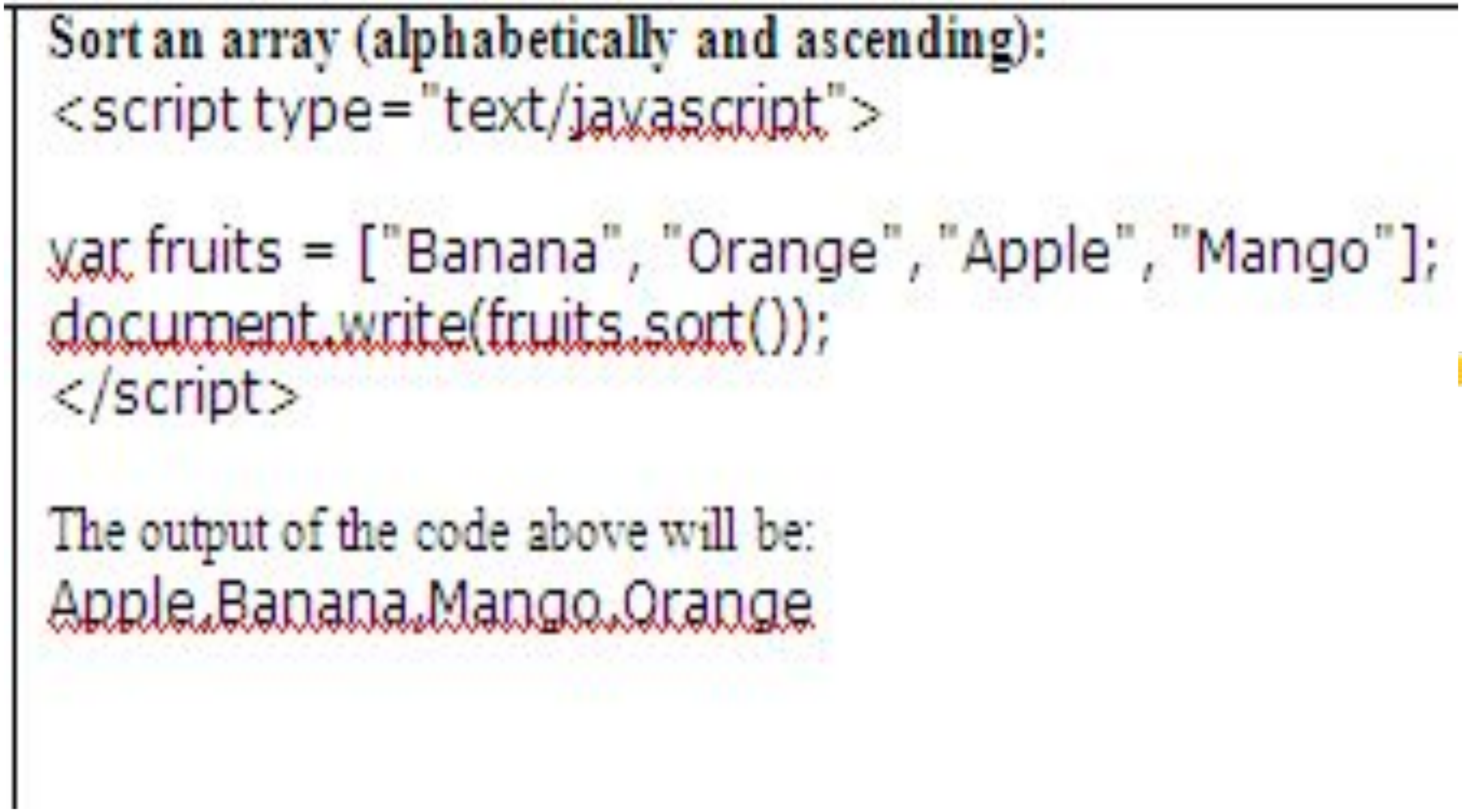


Return the primitive value of an array:

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.write(fruits.valueOf());

</script>
```

The output of the code above will be:

```
Banana,Orange,Apple,Mango
```


Sort an array (alphabetically and ascending):

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.sort());
</script>
```

The output of the code above will be:

```
Apple,Banana,Mango,Orange
```

By default, the sort() method sorts the values as strings in alphabetical and ascending order

"25" is bigger than "100", because "2" is bigger than "1"

array.sort(compareFunction)

```
<html>
<body>
<p id="demo"></p>
<script>
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return a-b});          points.sort((a, b) => a - b);
document.getElementById("demo").innerHTML = points;
</script>
</body>
</html>
```

```
<html><body><p>Descending order.</p>

<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
var points = [40, 100, 1, 5, 25, 10];
document.getElementById("demo").innerHTML = points;

function myFunction() {
  points.sort(function(a, b){return b-a});
  document.getElementById("demo").innerHTML = points;
}

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

The **splice() method** adds and/or removes elements to/from an array, and returns the removed element(s).

array.splice(index,howmany,element1,.....,elementX)

Parameter	Description
index	Required. An integer that specifies at what position to add/remove elements
howmany	Required. The number of elements to be removed. If set to 0, no elements will be removed
element1, ..., elementX	Optional. The new element(s) to be added to the array

Add an element to position 2 in the array:

```
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.write("Removed: " +
fruits.splice(2,0,"Lemon") + "<br />");
document.write(fruits);
</script>
```

The output of the code above will be:

```
Removed:
Banana,Orange,Lemon,Apple,Mango
```

```
<body>
<p>Click the button to add and remove elements.</p>
<button onclick="myFunction()">Try it</button>
<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];


document.write(fruits+"<br>");
 function myFunction() {
    fruits.splice(2, 1, "Lemon", "Kiwi");
    document.write(fruits+"<br>");
    fruits.splice(2, 2);
   document.write(fruits+"<br>");
   fruits.splice(2, 0, "Lemon", "Kiwi");
   document.write(fruits+"<br>");
   fruits.splice(-2, 0, "Banana");
   document.write(fruits+"<br>");
   fruits.splice(-1, 0, "Banana");
   document.write(fruits+"<br>");
 }
</script>
</body>
</html>
```

```
Banana,Orange,Apple,Mango

Banana,Orange,Lemon,Kiwi,Mango

Banana,Orange,Mango

Banana,Orange,Lemon,Kiwi,Mango

Banana,Orange,Lemon,Banana,Kiwi,Mango

Banana,Orange,Lemon,Banana,Kiwi,Banana,Mango
```

```
<html>
<body>
<p id="demo">Click the button to add elements to the array.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.unshift("Lemon","Pineapple");
document.write(fruits)
}
</script>
</body>
</html>
```

```
Lemon,Pineapple,Banana,Orange,Apple,Mango
```

Concat Arrays

```
<html>
<body>

<script type="text/javascript">

var parents = ["Jani", "Tove"];
var children = ["Cecilie", "Lone"];
var family = parents.concat(children);
document.write(family);

</script>

</body>
</html>
```

slice() method

Syntax

***array*.slice(start, end)**

Example

Select elements from an array, and return the new arrays:

```
<html><body><button onclick="myFunction()">Try it</button><script>
function myFunction() {
    var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
 document.write(fruits.slice(0,1) + "<br/>");
 document.write(fruits.slice(1) + "<br/>");
 document.write(fruits.slice(-2) + "<br />");
 document.write(fruits.slice(-3,-1) + "<br />");
  document.write(fruits.slice(-3,-2) + "<br />");
 document.write(fruits);
}
</script></body></html>
```

```
Banana

Orange,Lemon,Apple,Mango

Apple,Mango

Lemon,Apple

Lemon

Banana,Orange,Lemon,Apple,Mango
```

```
<html>
<body>

<p>Click the button to add elements to the beginning of the array.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits;

function myFunction() {
    fruits.unshift("Lemon", "Pineapple");
    document.getElementById("demo").innerHTML = fruits;
}
</script>
```

```
Lemon,Pineapple,Banana,Orange,Apple,Mango
```

JavaScript splice() Method

Syntax

***array*.splice(index,howmany,element1,.....,elementX)**

Example 1

```
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write("Added: " + fruits.splice(2,0,"Lemon") + "<br />");
document.write(fruits);
</script>
```

The output of the code above will be:

```
Added:Banana,Orange,Lemon,Apple,Mango
```

Example 2

Remove one element from position 2, and add a new element to position 2 in the array:

```
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write("Removed: " + fruits.splice(2,1,"Lemon") + "<br />");
document.write(fruits);
</script>
```

The output of the code above will be:

```
Removed: Apple
Banana,Orange,Lemon,Mango
```

JavaScript shift() Method

Syntax

***array*.shift()**

Remove the first element of an array

```
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.shift() + "<br />");
document.write(fruits + "<br />");
document.write(fruits.shift() + "<br />");
document.write(fruits);
</script>
```

The output of the code above will be:

```
Banana
Orange,Apple,Mango
Orange
Apple,Mango
```

```
<html><body><h2>JavaScript Array Methods</h2> <h2>push()</h2>
<p>The push() method appends a new element to an array.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits;
function myFunction() {
    fruits.push("Kiwi");
    fruits.push("Lemon", "Pineapple");
    document.getElementById("demo").innerHTML = fruits;
    }
</script></body></html>
```

```
Banana,Orange,Apple,Mango,Kiwi,Lemon,Pineapple
```

```
<html><body>
<h2>JavaScript Array Methods</h2>
<h2>shift()</h2>
<p>The shift() method returns the element that was shifted out.</p>
<p id="demo1"></p>
<p id="demo2"></p>
<p id="demo3"></p>
<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;
document.getElementById("demo2").innerHTML = fruits.shift();
document.getElementById("demo3").innerHTML = fruits;
</script>
</body>
</html>
```

```
Banana,Orange,Apple,Mango
Banana
Orange,Apple,Mango
```

```
<html>
<body>

<p>Click the button to remove the last element from the array.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits;

function myFunction() {
    fruits.pop();
    document.getElementById("demo").innerHTML = fruits;
}
</script>

</body>
</html>
```

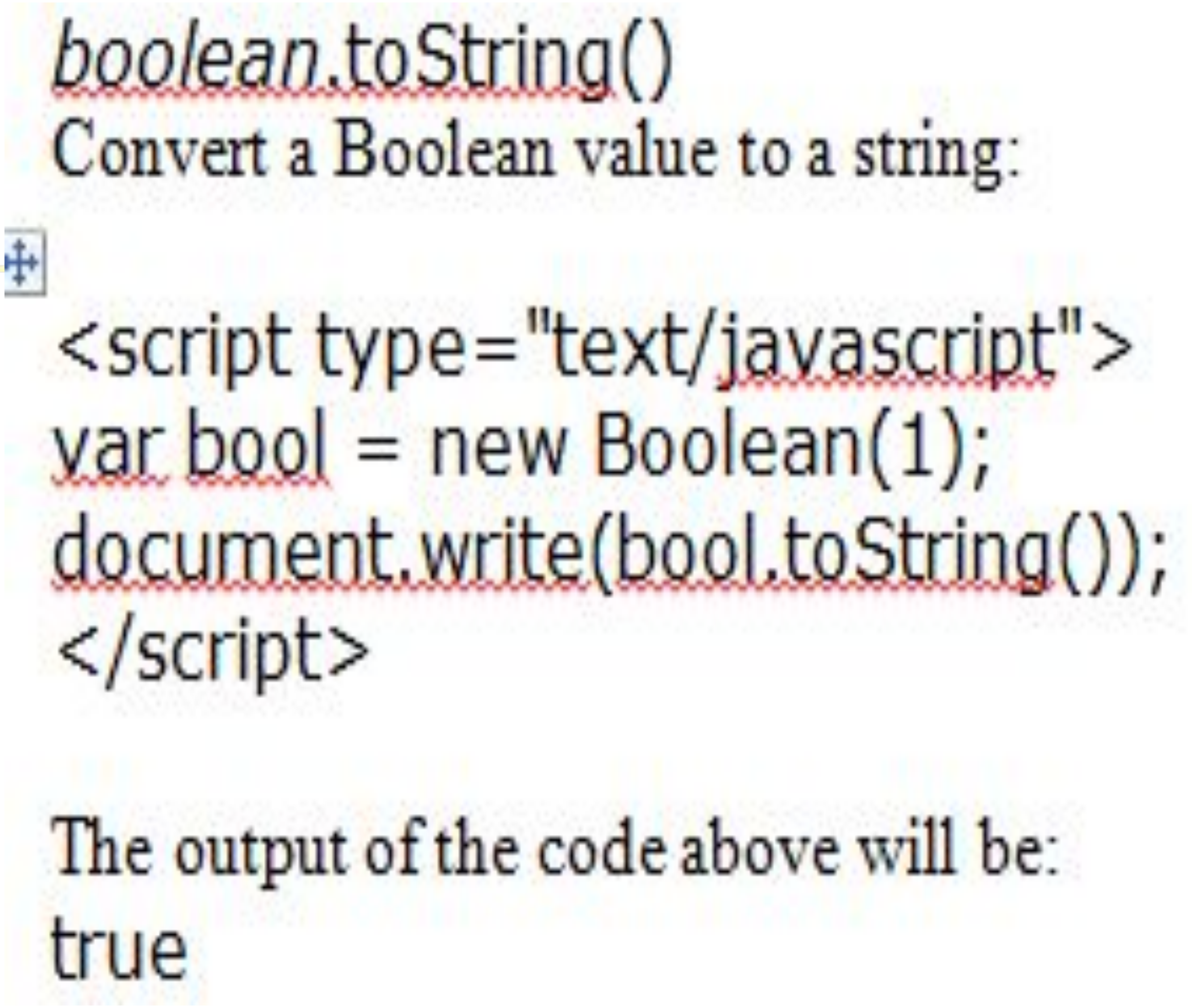
```
 Banana,Orange,Apple
```

Boolean Object Properties

Property	Description
constructor	Returns the function that created the Boolean object's prototype
prototype	Allows you to add properties and methods to an object

Boolean Object Methods

Method	Description
toString()	Converts a Boolean value to a string, and returns the result
valueOf()	Returns the primitive value of a Boolean object



boolean.toString()

Convert a Boolean value to a string:

```
<script type="text/javascript">
var bool = new Boolean(1);
document.write(bool.toString());
</script>
```

The output of the code above will be:

```
true
```

Date Object

There are four ways of instantiating a date:

```
var d = new Date();
var d = new Date(milliseconds);
var d = new Date(dateString);
var d = new Date(year, month, day, hours, minutes, seconds, milliseconds);
```

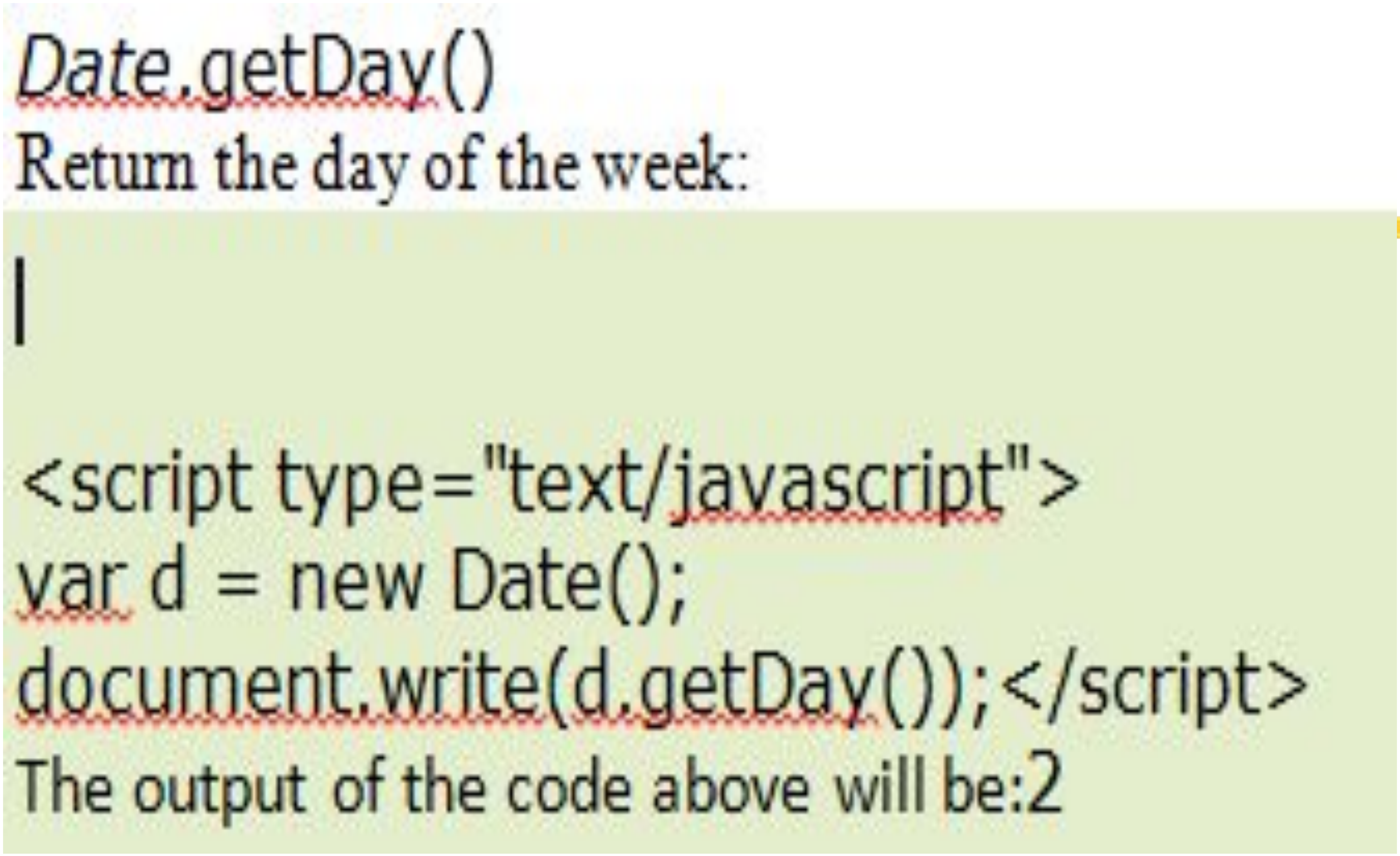
Date Object Properties

Property	Description
constructor	Returns the function that created the Date object's prototype
prototype	Allows you to add properties and methods to an object

```
<html><body>
<script>
var d = new Date();
document.write (d+"<br>");
var d = new Date(2021, 2, 13, 9, 56, 30, 0);
document.write (d+"<br>");
var d = new Date("March 14, 2021 10:15:00");
document.write (d+"<br>");
//the time is: 1615609161734 milliseconds past January 01, 1970

var d = new Date(1615609161734);
document.write (d +"<br>");
</script></body></html>
```

Date Object Methods

Method	Description
getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year (four digits)
getHours()	Returns the hour (from 0-23)
setDate()	Sets the day of the month (from 1-31)
setFullYear()	Sets the year (four digits)
setHours()	Sets the hour (from 0-23)
setMilliseconds()	Sets the milliseconds (from 0-999)
setMinutes()	Set the minutes (from 0-59)
setMonth()	Sets the month (from 0-11)
setSeconds()	Sets the seconds (from 0-59)
setTime()	Sets a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970
valueOf()	Returns the primitive value of a Date object

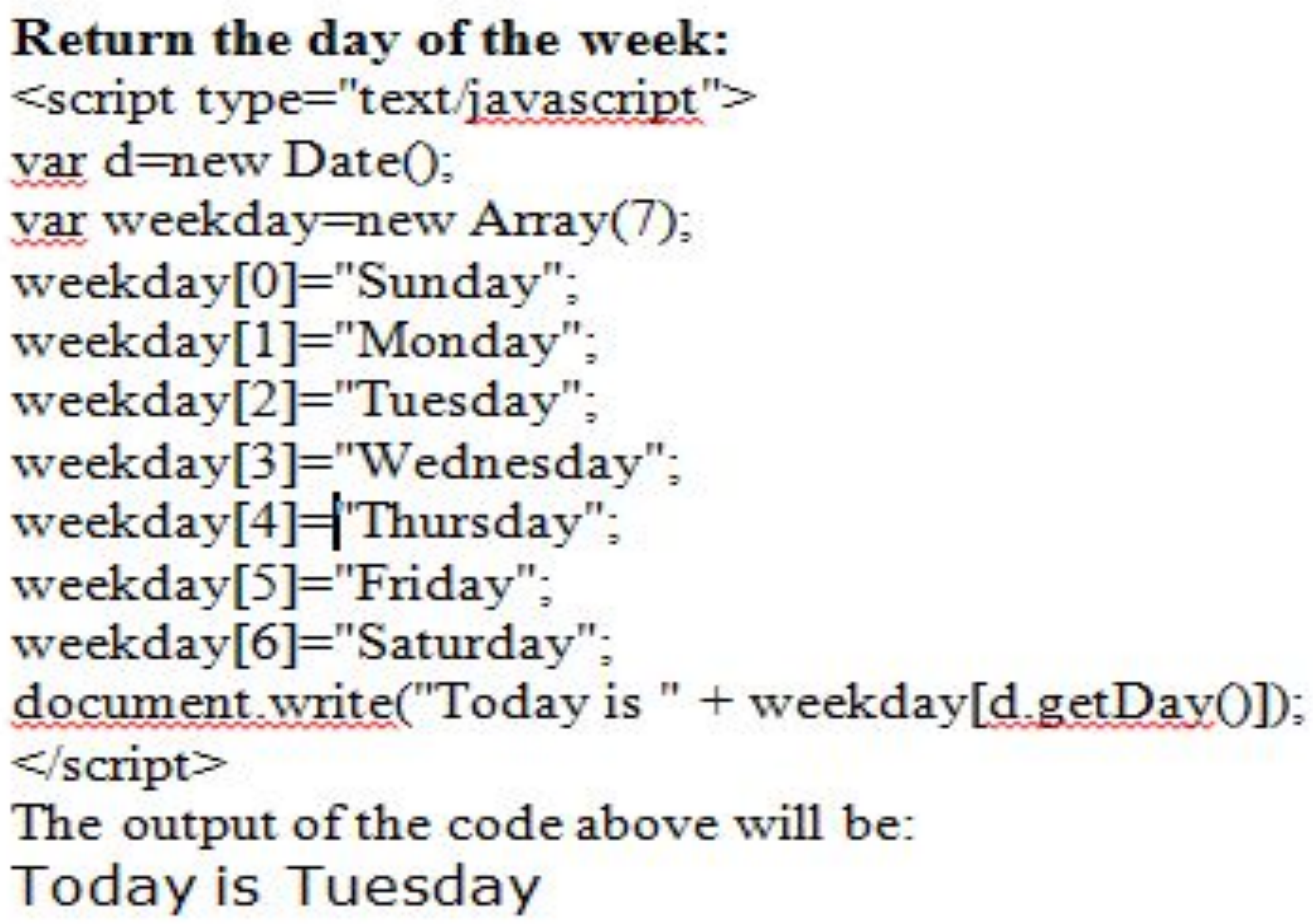


Date.getDay()

Return the day of the week:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getDay());</script>
```

The output of the code above will be:2



Return the day of the week:

```
<script type="text/javascript">
var d=new Date();
var weekday=new Array(7);
weekday[0]="Sunday";
weekday[1]="Monday";
weekday[2]="Tuesday";
weekday[3]="Wednesday";
weekday[4]="Thursday";
weekday[5]="Friday";
weekday[6]="Saturday";
document.write("Today is " + weekday[d.getDay()]);
</script>
```

The output of the code above will be:

```
Today is Tuesday
```

```
<html><body>
<p id="demo"></p>
<script>
var d = new Date();
var months =
["January","February","March","April","May","June","July","August","
September","October","November","December"];
document.getElementById("demo").innerHTML =
months[d.getMonth()];
</script>
</body>
</html>
```

JavaScript getUTCHours() Method

The getUTCHours() method returns the hour (from 0 to 23) of the specified date and time, according to universal time.

Syntax

Date.getUTCHours()

Tip: The Universal Coordinated Time (UTC) is the time set by the World Time Standard.

Example 1

Return the hour, according to universal time:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getUTCHours());
</script>
```

Example 2

Return the UTC hour from a specific date and time:

```
<script type="text/javascript">
var d=new Date("July 21, 1983 01:15:00");
document.write(d.getUTCHours());
</script>
```

JavaScript setHours() Method

The setHours() method sets the hour (from 0 to 23), according to local time.

Syntax

Date.setHours(hour,min,sec,millisec)

Example 2

Set the time to 15:35:01:

```
<script type="text/javascript">
var d = new Date();
d.setHours(15,35,1);
document.write(d);
</script>
```

The output of the code above will be:

```
Tue Jan 1 15:35:01 UTC+0530 2002
```

JavaScript Math Object

The Math object allows you to perform mathematical tasks.
Math is not a constructor. All properties/methods of Math can be called by using Math as an object, without creating it.

Syntax

```
var x = Math.PI; // Returns PI
var y = Math.sqrt(16); // Returns the square root of 16
```

Math Object Properties

Property	Description
E	Returns Euler's number (approx. 2.718)
LN2	Returns the natural logarithm of 2 (approx. 0.693)
LN10	Returns the natural logarithm of 10 (approx. 2.302)
LOG2E	Returns the base-2 logarithm of E (approx. 1.442)
LOG10E	Returns the base-10 logarithm of E (approx. 0.434)
PI	Returns PI (approx. 3.14159)
SQRT1_2	Returns the square root of 1/2 (approx. 0.707)
SQRT2	Returns the square root of 2 (approx. 1.414)

Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y,x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of E^x
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x,y,z,...,n)	Returns the number with the highest value
min(x,y,z,...,n)	Returns the number with the lowest value
pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

JavaScript E Property

The E property returns the Euler's number and the base of natural logarithms, 2.718.

Syntax

```
Math.E
```

Example

Return the Euler's number:

```
<script type="text/javascript">
document.write("Euler's number: " + Math.E);
</script>
```

The output of the code above will be:

```
Euler's number: 2.718281828459045
```

Syntax

```
Math.sin(x)
```

Parameter	Description
x	Required. A number

Example

Return the sine of different numbers:

```
<script type="text/javascript">

document.write(Math.sin(3) + "<br />");
document.write(Math.sin(-3) + "<br />");
document.write(Math.sin(Math.PI/2));

</script>
```

The output of the code above will be:

```
0.1411200080598672
-0.1411200080598672
1
```

JavaScript Number Object

The Number object is an object wrapper for primitive numeric values.

Number objects are created with new Number().

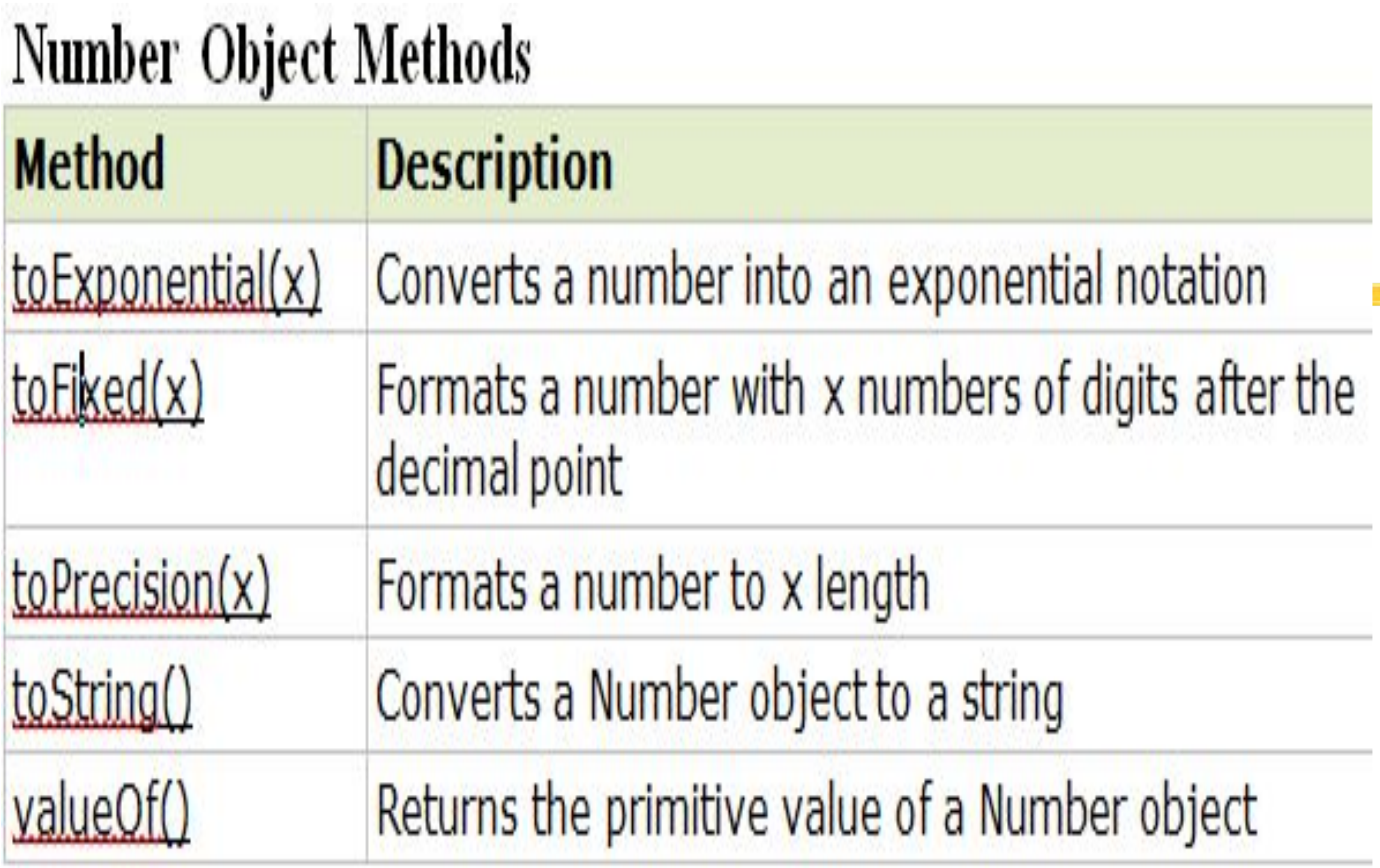
Syntax

```
var num = new Number(value);
```

Note: If the value parameter cannot be converted into a number, it returns NaN (Not-a-Numbe

Number Object Properties

Property	Description
constructor	Returns the function that created the Number object's prototype
MAX_VALUE	Returns the largest number possible in JavaScript
MIN_VALUE	Returns the smallest number possible in JavaScript
NEGATIVE_INFINITY	Represents negative infinity (returned on overflow)
POSITIVE_INFINITY	Represents infinity (returned on overflow)
prototype	Allows you to add properties and methods to an objec



Number Object Methods

Method	Description
toExponential(x)	Converts a number into an exponential notation
toFixed(x)	Formats a number with x numbers of digits after the decimal point
toPrecision(x)	Formats a number to x length
toString()	Converts a Number object to a string
valueOf()	Returns the primitive value of a Number object

JavaScript MAX_VALUE Property

The MAX_VALUE property returns the largest number possible in JavaScript.
This static property has a value of 1.7976931348623157e+308.

Note: Numbers larger than this are represented as infinity.

Syntax

```
Number.MAX_VALUE
```

Example

Return the largest number possible in JavaScript.

```
<script type="text/javascript">
document.write(Number.MAX_VALUE);
</script>
```

The output of the code above will be:

```
1.7976931348623157e+308
```

JavaScript toFixed() Method

The toFixed() method formats a number to use a specified number of trailing decimals.

Syntax

number.toFixed(x)

Parameter	Description
x	Optional. The number of digits after the decimal point. Default is 0 (no digits after the decimal point)

Example

Format a number:

```
<script type="text/javascript">
var num = new Number(13.3714);
document.write(num.toFixed()+"<br />");
document.write(num.toFixed(1)+"<br />");
document.write(num.toFixed(3)+"<br />");
document.write(num.toFixed(10));
</script>
```

The output of the code above will be:

```
13
13.4
13.371
13.3714000000
```

```
function myFunction() {
    var num = 5453465.56789;
    var n = num.toExponential();
    document.getElementById("demo").innerHTML = n;
}
```

```
O/P  5.45346556789e+6
```

```
function myFunction() {
    var num = 1367576.3714;
    document.getElementById("demo").innerHTML = num.toPrecision(5);
    document.getElementById("demo").innerHTML = num.toPrecision(2);
}
```

```
1.3676e+6
1.4e+6
```

String Object

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

String objects are created with new String().

Syntax

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

or more simply:

```
var txt = string;
```

String Object Properties

Property	Description
constructor	Returns the function that created the String object's prototype
length	Returns the length of a string
prototype	Allows you to add properties and methods to an object

String Object Methods

Method	Description
charAt()	Returns the character at the specified index
charCodeAt()	Returns the Unicode of the character at the specified index
concat()	Joins two or more strings, and returns a copy of the joined strings
fromCharCode()	Converts Unicode values to characters
indexOf()	Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last found occurrence of a specified value in a string
match()	Searches for a match between a regular expression and a string, and returns the matches
replace()	Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring
search()	Searches for a match between a regular expression and a string, and returns the position of the match
slice()	Extracts a part of a string and returns a new string
split()	Splits a string into an array of substrings
substr()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
substring()	Extracts the characters from a string, between two specified indices
toLowerCase()	Converts a string to lowercase letters
toUpperCase()	Converts a string to uppercase letters
valueOf()	Returns the primitive value of a String object

JavaScript charAt() Method

Syntax

string.charAt(index)

Parameter	Description
index	Required. An integer between 0 and *string*.length-1

Example

Return the first and last character of a string:

```
<script type="text/javascript">
var str = "Hello world!";
document.write("First character: " + str.charAt(0) + "<br />");
document.write("Last character: " + str.charAt(str.length-1));
</script>
```

The output of the code above will be:

```
First character: H
Last character: !
```

```
<html>
<body>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
var str="Hello world,welcome to the universe.";
var n=str.indexOf("welcome");
document.write(n);
}
</script>
</body>
</html>
```

O/p   12

```
<html>
<body>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
var str="The rain in SPAIN stays mainly in the plain";
var n=str.match(/ain/g);
document.write(n);
}
</script>
</body>
</html>
```

o/p ain,ain,ain

```
<html>
<body>
<p id="demo">Visit Microsoft!</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
var str=document.getElementById("demo").innerHTML;
var n=str.replace("Microsoft","W3Schools");
document.getElementById("demo").innerHTML=n;
}
</script>

</body>
</html>
```

JavaScript search() Method

The search() method searches for a match between a regular expression and a string.

Syntax

string.search(regexp)

Parameter	Description
regexp	Required. A regular expression

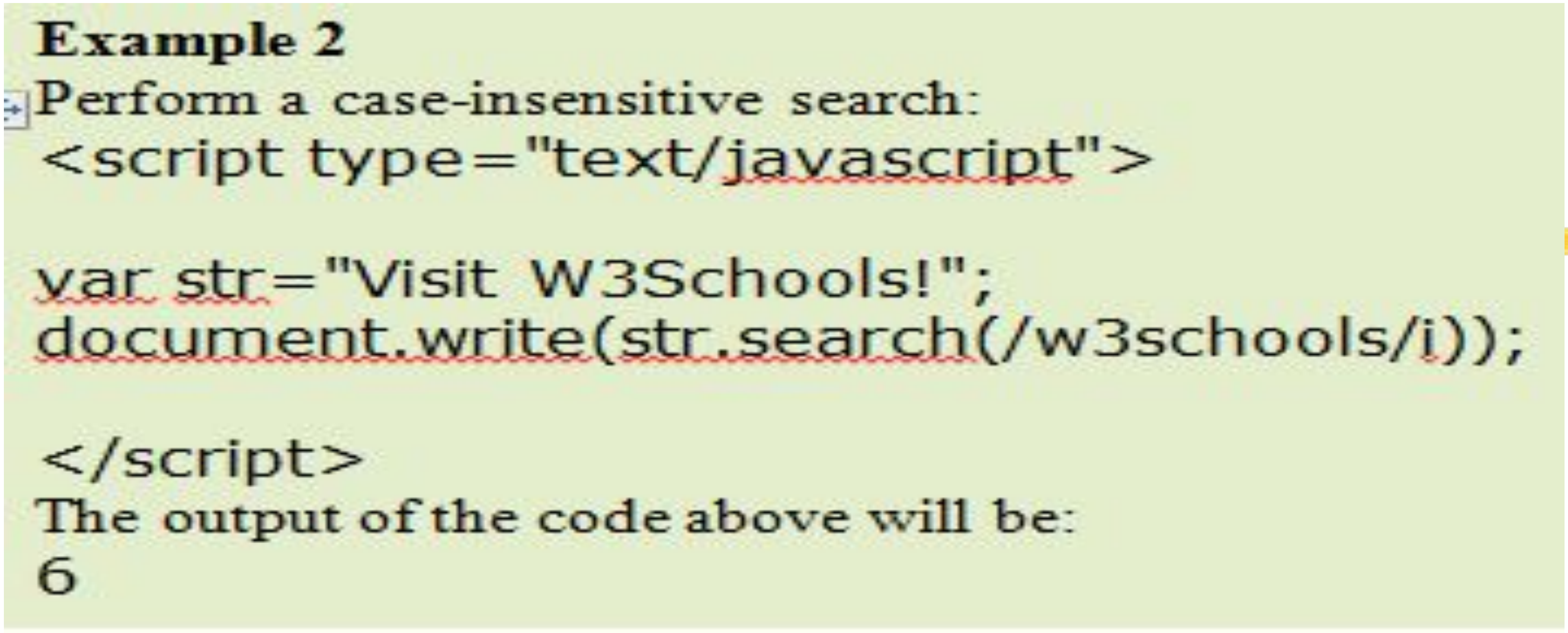
Example 1

Perform a case-sensitive search:

```
<script type="text/javascript">

var str="Visit W3Schools!";
document.write(str.search("W3SCHOOLS"));

</script>
```

The output of the code above will be:

```
-1
```



Example 2

Perform a case-insensitive search:

```
<script type="text/javascript">

var str="Visit W3Schools!";
document.write(str.search(/w3schools/i));

</script>
```

The output of the code above will be:

```
6
```

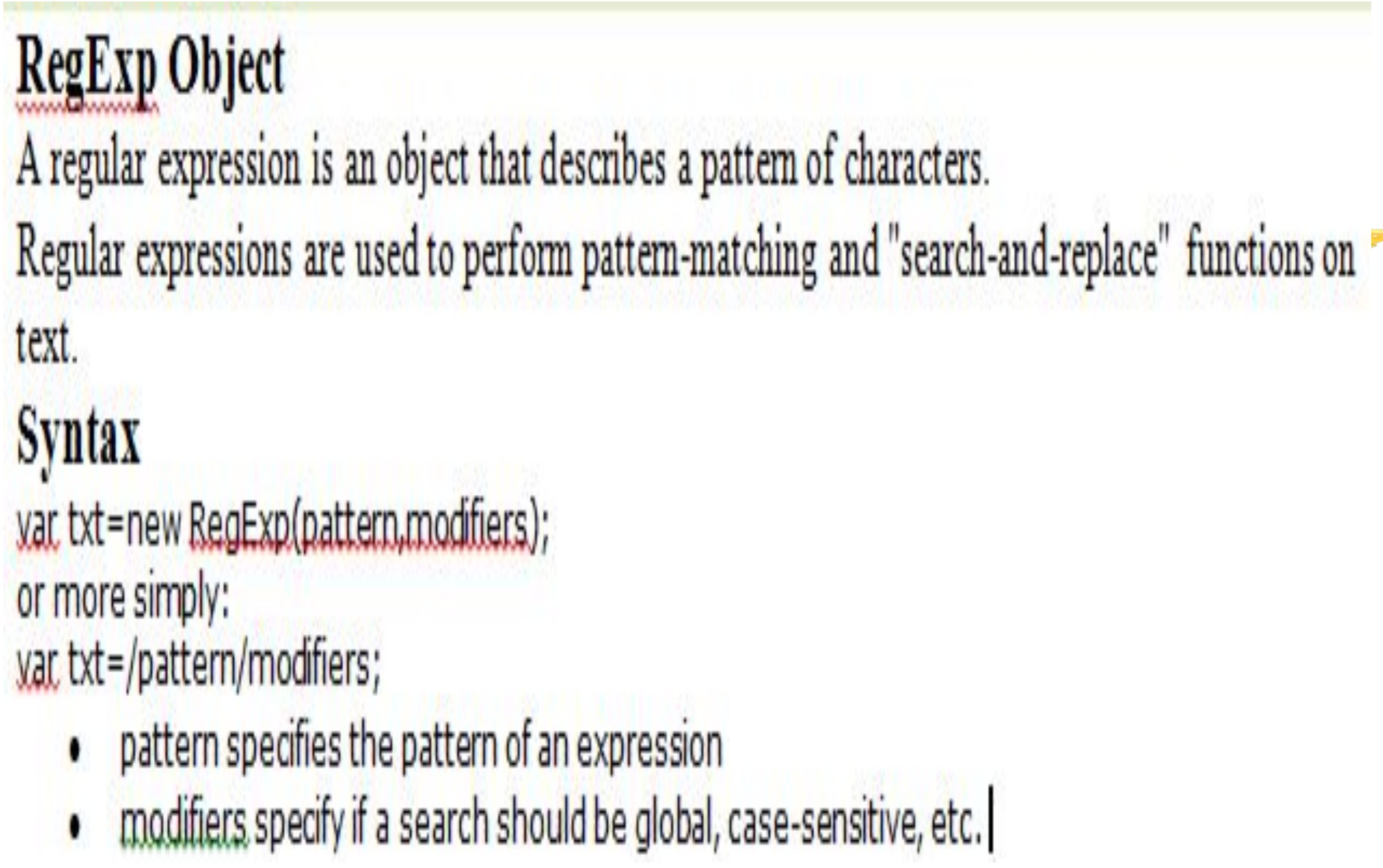
```
<script>
var str = "Apple, Banana, Kiwi";
var res = str.slice(7,13);                                   Banana
document.getElementById("demo").innerHTML = res;
</script>
```

```
var str = "Apple, Banana, Kiwi";                              Banana
var res = str.slice(-12, -6);
```

```
var str = "Apple, Banana, Kiwi";
var res = str.substring(7, 13);                              Banana
```

```
var str = "Apple, Banana, Kiwi";
var res = str.substr(7, 6);                                   Banana
```

```
<p id="demo">Please visit Microsoft!</p>
<script>
function myFunction() {
    var str = document.getElementById("demo").innerHTML;
    var txt = str.replace("Microsoft","W3Schools");
    document.getElementById("demo").innerHTML = txt;
}
</script>
```



RegExp Object

A regular expression is an object that describes a pattern of characters.

Regular expressions are used to perform pattern-matching and "search-and-replace" functions on text.

Syntax

```
var txt=new RegExp(pattern,modifiers);
```

or more simply:

```
var txt=/pattern/modifiers;
```

- pattern specifies the pattern of an expression
- modifiers specify if a search should be global, case-sensitive, etc.

Modifiers

Modifiers are used to perform case-insensitive and global searches:

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

Brackets

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character not between the brackets
[0-9]	Find any digit from 0 to 9
[A-Z]	Find any character from uppercase A to uppercase Z
[a-z]	Find any character from lowercase a to lowercase z
[A-z]	Find any character from uppercase A to lowercase z
[adgk]	Find any character in the given set
[^adgk]	Find any character outside the given set
(red|blue|green)	Find any of the alternatives specified

Metacharacters

Metacharacters are characters with a special meaning:

Metacharacter	Description
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character
\b	Find a match at the beginning/end of a word
\B	Find a match not at the beginning/end of a word
\0	Find a NUL character
\n	Find a new line character
\f	Find a form feed character
\r	Find a carriage return character
\t	Find a tab character
\v	Find a vertical tab character
\xxx	Find the character specified by an octal number xxx
\xdd	Find the character specified by a hexadecimal number dd
\uxxxx	Find the Unicode character specified by a hexadecimal number xxxx

Quantifiers

Quantifier	Description
n+	Matches any string that contains at least one n
n*	Matches any string that contains zero or more occurrences of n
n?	Matches any string that contains zero or one occurrences of n
n{X}	Matches any string that contains a sequence of X n's
n{X,Y}	Matches any string that contains a sequence of X or Y n's
n{X,}	Matches any string that contains a sequence of at least X n's
n$	Matches any string with n at the end of it
^n	Matches any string with n at the beginning of it
?=n	Matches any string that is followed by a specific string n
?!n	Matches any string that is not followed by a specific string n

RegExp Object Properties

Property	Description
global	Specifies if the "g" modifier is set
ignoreCase	Specifies if the "i" modifier is set
lastIndex	The index at which to start the next match
multiline	Specifies if the "m" modifier is set
source	The text of the RegExp pattern

RegExp Object Methods

Method	Description
compile()	Compiles a regular expression
exec()	Tests for a match in a string. Returns the first match
test()	Tests for a match in a string. Returns true or false

```
<html>
<body>

<p>Click the button to do a global search for digits in a string.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "Give 100%!";
  var patt1 = /\d/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

JavaScript RegExp [abc] Expression

The [abc] expression is used to find any character between the brackets.
The characters inside the brackets can be any characters or span of characters.

Syntax

```
new RegExp("[abc]")
```

or simply:

```
/[abc]/
```

Example

Do a global search for the character-span [a-h] in a string:

```
var str="Is this all there is?";
var patt1=/[a-h]/g;
```

The marked text below shows where the expression gets a match:

Is this all there is?

```
<html><body>
<p>Click the button to do a global search for digits in a string.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "Is this all there is?";
  var patt1 = /[a-h]/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body></html>
```

JavaScript RegExp \W Metacharacter

The \W metacharacter is used to find a non-word character.
A word character is a character from a-z, A-Z, 0-9, including the _ (underscore) character.

Syntax

```
new RegExp("\W")
```

or simply:

```
/\W/
```

Example

Do a global search for non-word characters in a string:

```
var str="Give 100%!";
var patt1=/\W/g;
```

The marked text below shows where the expression gets a match:

Give 100%!

JavaScript RegExp \b Metacharacter

The \b metacharacter is used to find a match at the beginning or end of a word.
If no match is found, it returns null.

Syntax

```
new RegExp("\bregexp")
```

or simply:

```
/\bregexp/
```

Example

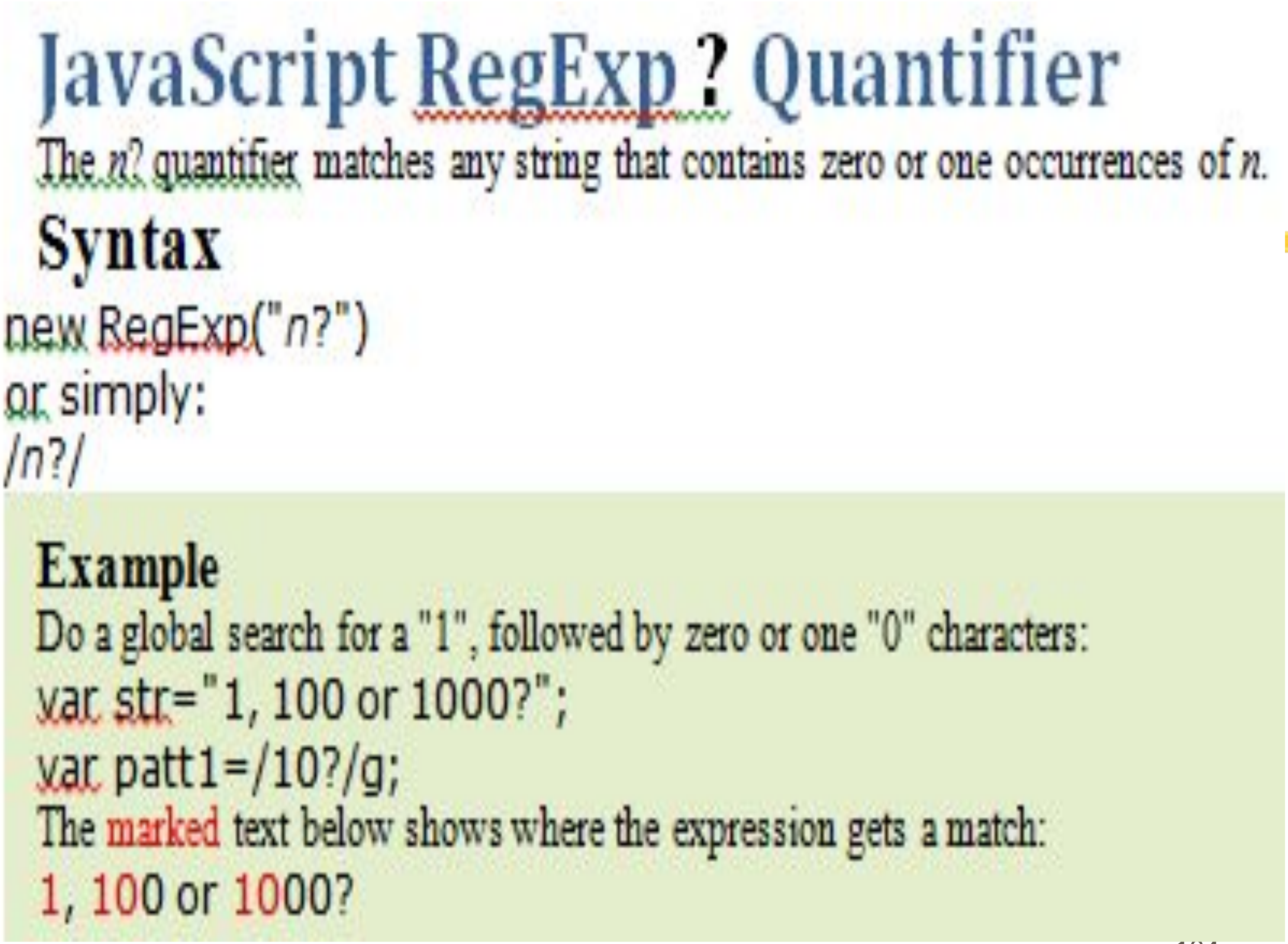
Do a global search for "W3" at the beginning or end of a word in a string:

```
var str="Visit W3Schools";
var patt1=/\bW3/g;
```

The marked text below shows where the expression gets a match:

Visit W3Schools

```
<html><body>
<p>Click the button to do a global search for digits in a string.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "Visit W3Schools";
  var patt1 = /\bW3/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body></html>
```



JavaScript RegExp ? Quantifier

The *n*? quantifier matches any string that contains zero or one occurrences of *n*.

Syntax

```
new RegExp("n?")
```

or simply:

```
/n?/
```

Example

Do a global search for a "1", followed by zero or one "0" characters:

```
var str="1, 100 or 1000?";
var patt1=/10?/g;
```

The marked text below shows where the expression gets a match:

```
1, 100 or 1000?
```

var str = "I Scream For Ice Cream, is that OK?!";
var patt1 = /[^A-e]/g;
var result = str.match(patt1);
	,r,m, ,o,r, , ,r,m,,, ,i,s, ,t,h,t, ,?,!

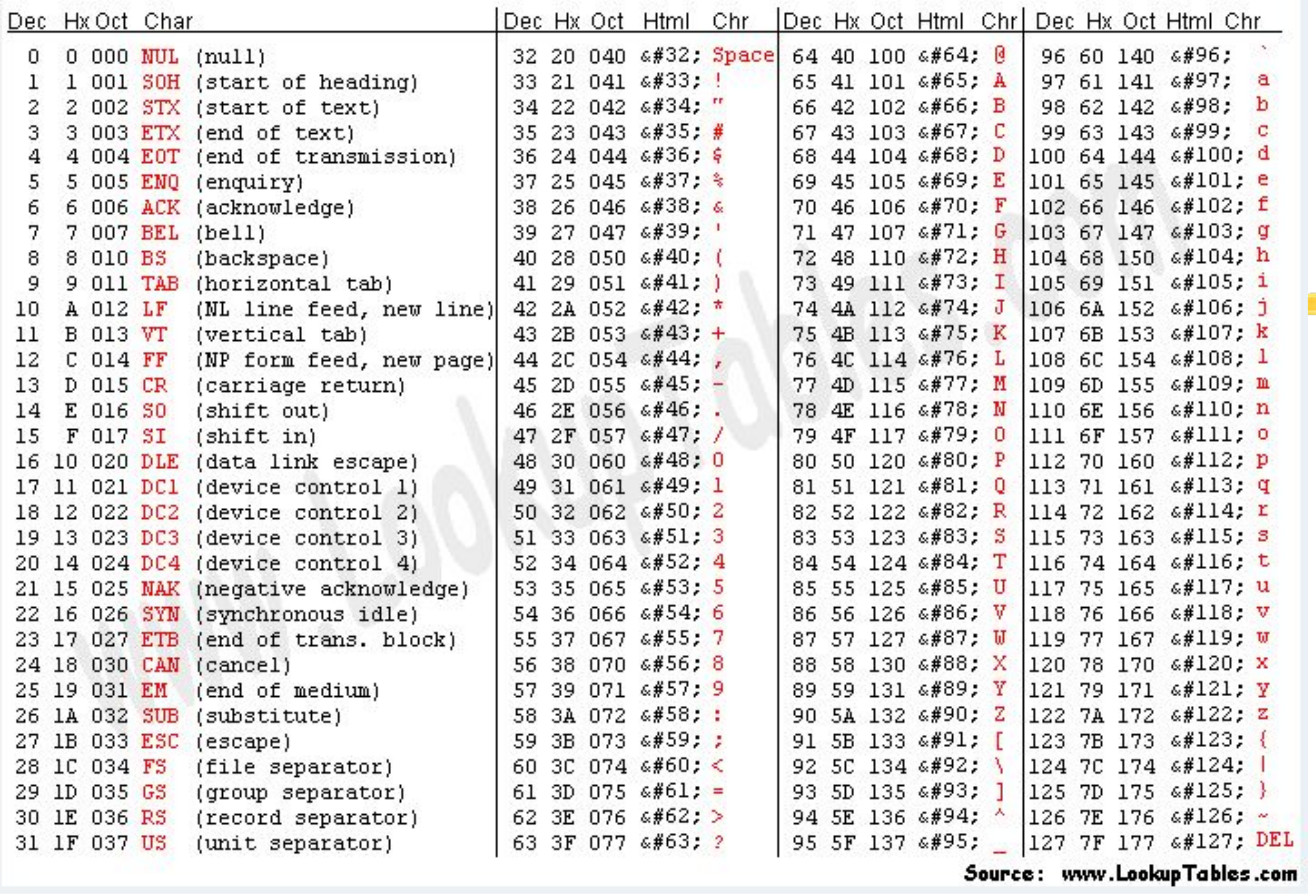
var str = "I Scream For Ice Cream, is that OK?!";
var patt1 = /[^A-e]/gi;
var result = str.match(patt1);
	, , , ,,, , , ,?,!

var str = "That's hot!";
var patt1 = /h.t/g;
var result = str.match(patt1);
	hat,hot

var str = "Give 100%!";
var patt1 = /\d/g;
var result = str.match(patt1);
	1,0,0

var str = "Is this all there is?";
var patt1 = /\s/g;
var result = str.match(patt1);
	, , ,

var str = "Visit W3Schools.\fLearn JavaScript.";
var patt1 = /\f/;
var result = str.search(patt1);
	16



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

var str = "Visit W3Schools. Hello World!";
var patt1 = /\127/g;
var result = str.match(patt1);
	W,W

var str = "Hellooo World! Hello W3Schools!";
var patt1 = /o+/g;
var result = str.match(patt1);
	ooo,o,o,oo

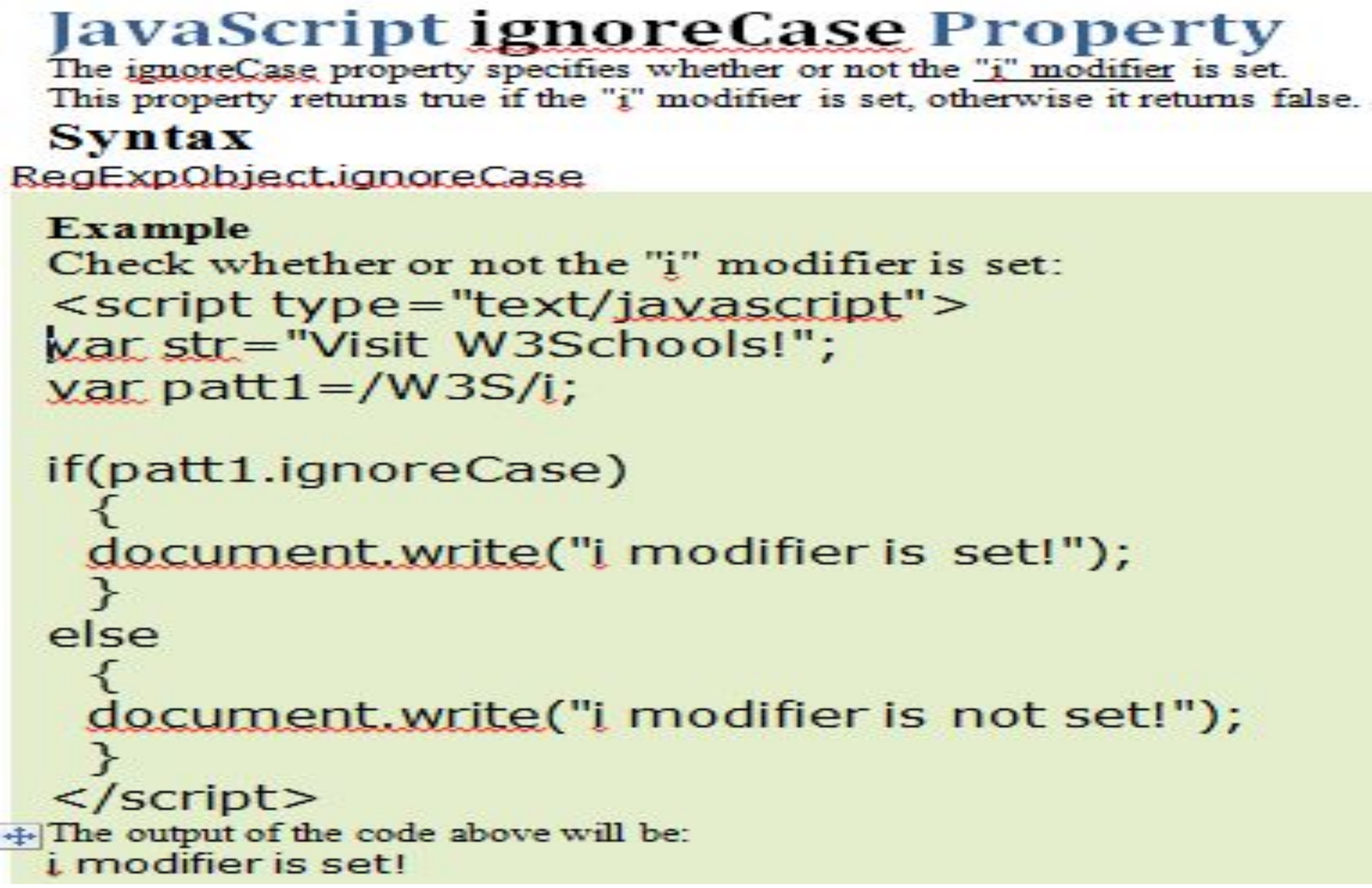
var str = "Hellooo World! Hello W3Schools!";
var patt1 = /lo*/g;
var result = str.match(patt1);
	l,looo,l,l,lo,l

var str = "100, 1000 or 10000?";
var patt1 = /\d{4}/g;
var result = str.match(patt1);
	1000,1000

var str = "Is this his";
var patt1 = /is$/g;
var result = str.match(patt1);
	is

var str = "Is this all there is"; var patt1 = /is(?= all)/; var result = str.match(patt1);	is

var str = "The best things in life are free";
var patt = new RegExp("e");
var res = patt.exec(str);
	e



JavaScript ignoreCase Property

The ignoreCase property specifies whether or not the "i" modifier is set.
This property returns true if the "i" modifier is set, otherwise it returns false.

Syntax

RegExpObject.ignoreCase

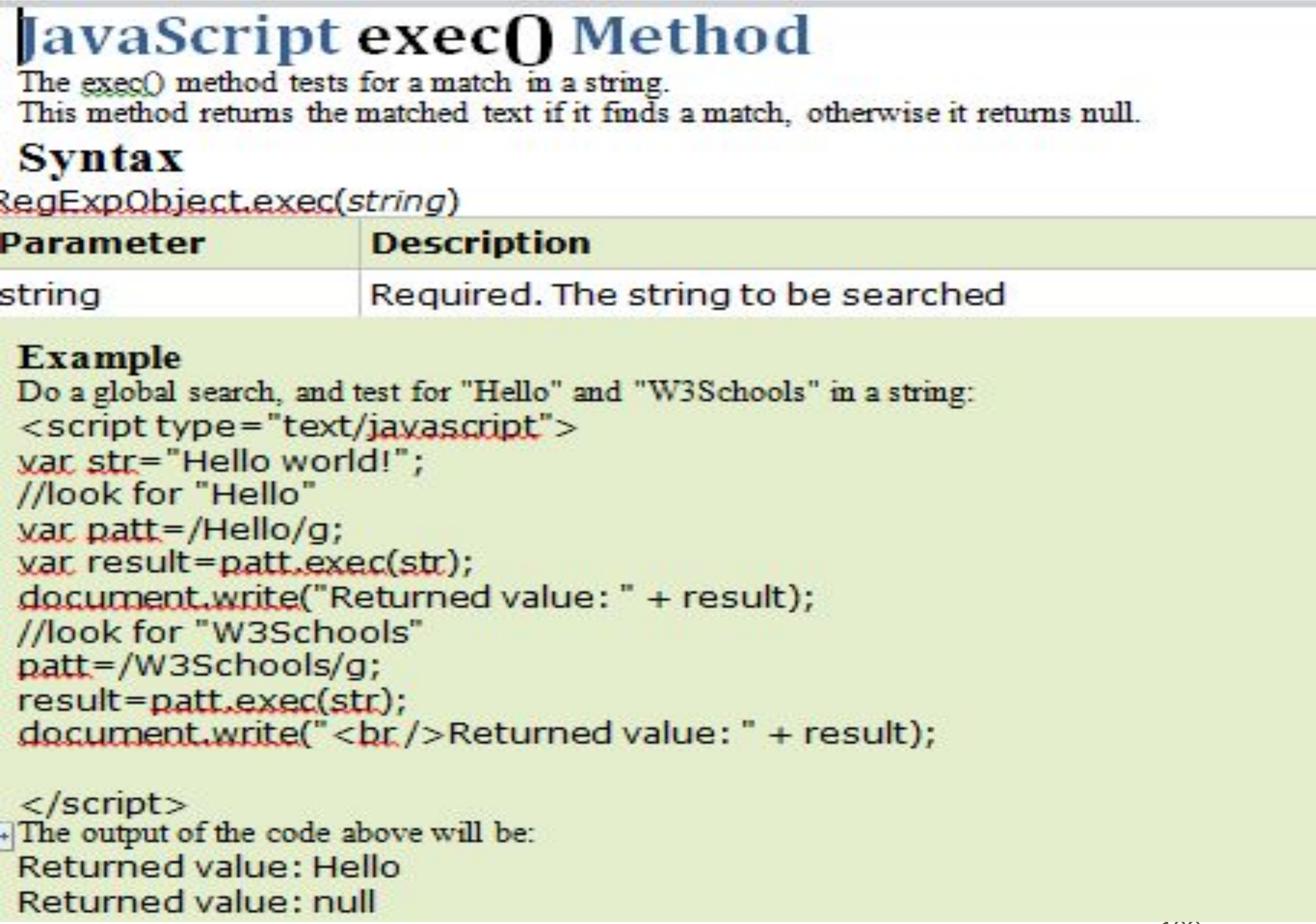
Example

Check whether or not the "i" modifier is set:

```
<script type="text/javascript">
var str="Visit W3Schools!";
var patt1=/W3S/i;

if(patt1.ignoreCase)
  {
  document.write("i modifier is set!");
  }
else
  {
  document.write("i modifier is not set!");
  }
</script>
```

The output of the code above will be:

```
i modifier is set!
```



JavaScript exec() Method

The exec() method tests for a match in a string.
This method returns the matched text if it finds a match, otherwise it returns null.

Syntax

```
RegExpObject.exec(string)
```

Parameter	Description
string	Required. The string to be searched

Example

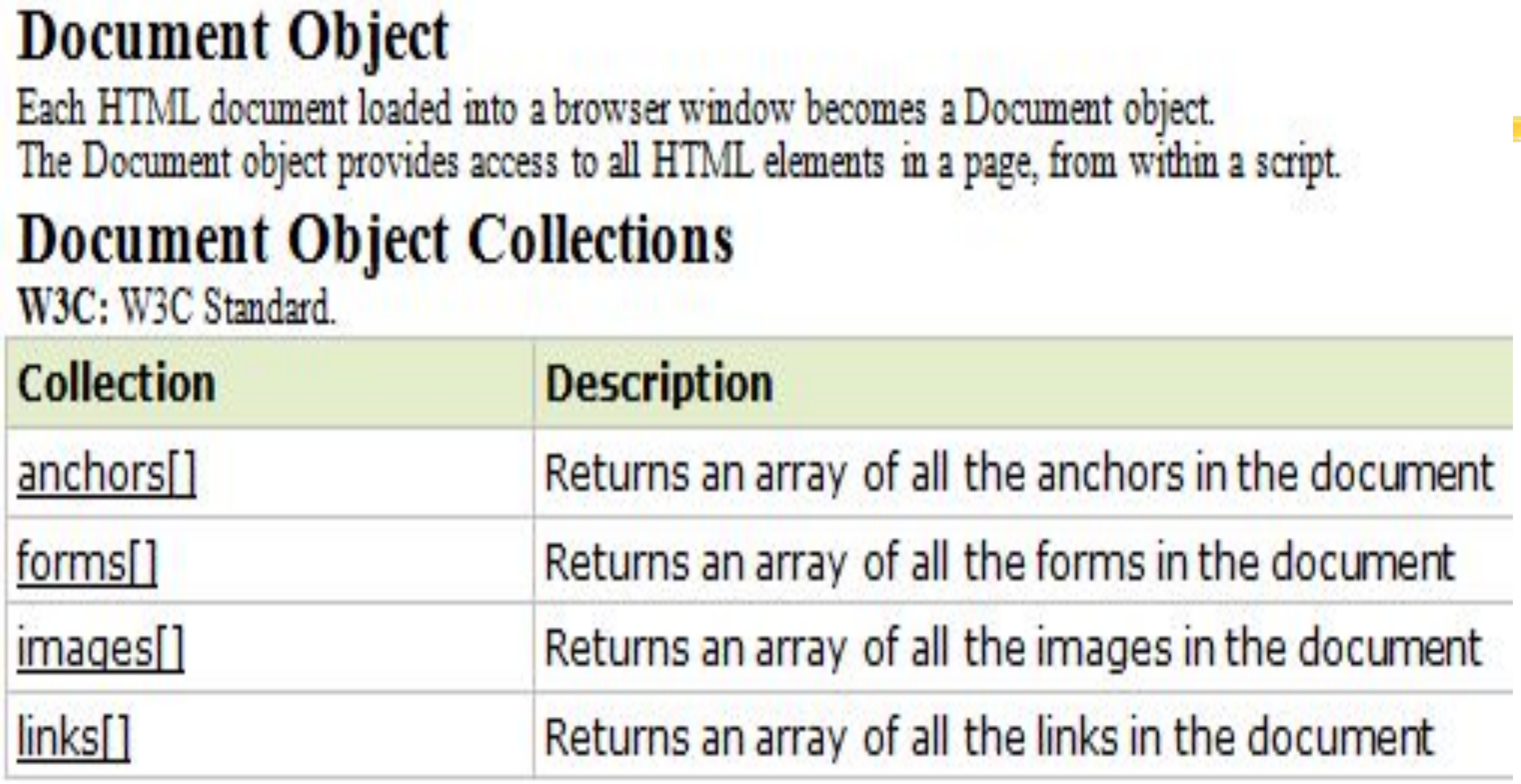
Do a global search, and test for "Hello" and "W3Schools" in a string:

```
<script type="text/javascript">
var str="Hello world!";
//look for "Hello"
var patt=/Hello/g;
var result=patt.exec(str);
document.write("Returned value: " + result);
//look for "W3Schools"
patt=/W3Schools/g;
result=patt.exec(str);
document.write("<br />Returned value: " + result);

</script>
```

The output of the code above will be:

```
Returned value: Hello
Returned value: null
```

Document and its associated objects



Document Object

Each HTML document loaded into a browser window becomes a Document object.
The Document object provides access to all HTML elements in a page, from within a script.

Document Object Collections

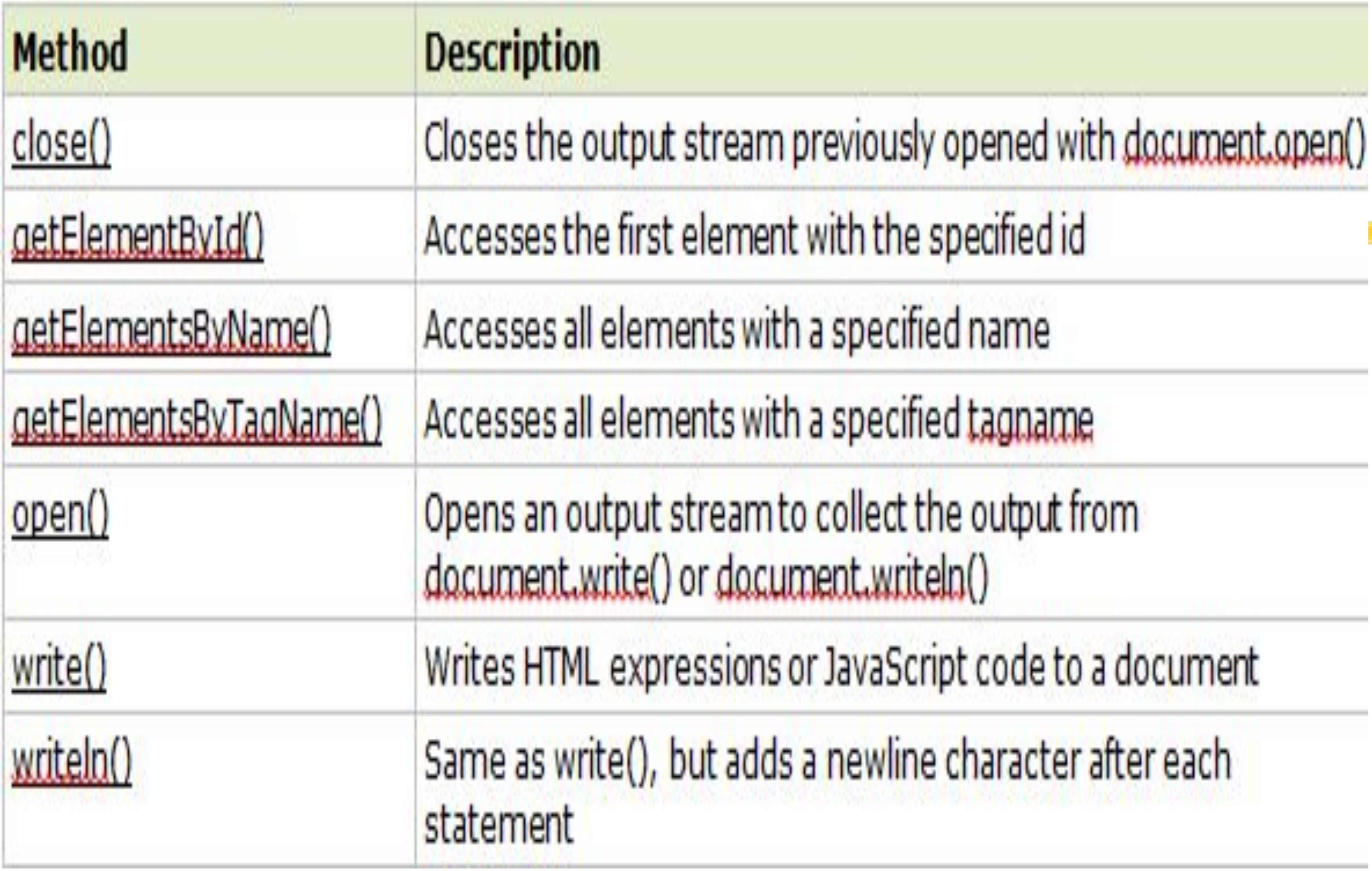
W3C: W3C Standard.

Collection	Description
anchors[]	Returns an array of all the anchors in the document
forms[]	Returns an array of all the forms in the document
images[]	Returns an array of all the images in the document
links[]	Returns an array of all the links in the document

Document Object Properties

Property	Description
cookie	Returns all name/value pairs of cookies in the document
documentMode	Returns the mode used by the browser to render the document
domain	Returns the domain name of the server that loaded the document
lastModified	Returns the date and time the document was last modified
readyState	Returns the (loading) status of the document
referrer	Returns the URL of the document that loaded the current document
title	Sets or returns the title of the document
URL	Returns the full URL of the document

Document Object Methods



Method	Description
close()	Closes the output stream previously opened with document.open()
getElementById()	Accesses the first element with the specified id
getElementsByName()	Accesses all elements with a specified name
getElementsByTagName()	Accesses all elements with a specified tagname
open()	Opens an output stream to collect the output from document.write() or document.writeln()
write()	Writes HTML expressions or JavaScript code to a document
writeln()	Same as write(), but adds a newline character after each statement

Document links Collection

The links collection returns an array of all the links in the current document.

Tip: The links collection counts <a href=""> tags and <area> tags.

Syntax

document.links[].*property*

Example 1

Return the number of links in the document:

```
<html>
<body>

<img src ="planets.gif" width="145" height="126" alt="Planets" usemap ="#planetmap" />
<map name="planetmap">
<area shape="rect" coords="0,0,82,126" href="sun.htm" alt="Sun" />
<area shape="circle" coords="90,58,3" href="mercur.htm" alt="Mercury" />
<area shape="circle" coords="124,58,8" href="venus.htm" alt="Venus" />
</map>
<p><a href="/js/">JavaScript Tutorial</a></p>
<p>Number of areas/links:
<script type="text/javascript">
document.write(document.links.length);
</script></p>
</body></html>
```

The output of the code above will be:

```
Number of areas/links: 4
```

Example 2

Return the id of the first link in the document:

```
<html>
<body>
<img src ="planets.gif" width="145" height="126" alt="Planets" usemap ="#planetmap" />
<map name="planetmap">
<area id="sun" shape="rect" coords="0,0,82,126" href="sun.htm" alt="Sun" />
<area id="mercury" shape="circle" coords="90,58,3" href="mercur.htm" alt="Mercury" />
<area id="venus" shape="circle" coords="124,58,8" href="venus.htm" alt="Venus" />
</map>
<p><a id="javascript" href="/js/">JavaScript Tutorial</a></p>
<p>Id of first area/link:
<script type="text/javascript">
document.write(document.links[0].id);
</script></p></body></html>
```

The output of the code above will be:

```
Id of first area/link: sun
```

Document documentMode Property

The documentMode property returns the mode used by the browser to render the current document.
This property returns one of three values:

- 5 - The page is displayed in IE5 mode
- 7 - The page is displayed in IE7 mode
- 8 - The page is displayed in IE8 mode

Note: If no !DOCTYPE is specified, IE8 renders the page in IE5 mode!

Example

Return the mode used by the browser to render the current document:

```
<html><body>
This document is displayed in:
<script type="text/javascript">
document.write(document.documentMode);
</script></body></html>
```

Document getElementById() Method

The getElementById() method accesses the first element with the specified id.

Syntax

document.getElementById("*id*")

Parameter	Description
id	Required. The id of the element you want to access/manipulate

Example

Alert innerHTML of an element with a specific ID:

```
<html><head>
<script type="text/javascript">
function getValue()
  {
  var x=document.getElementById("myHeader");
  alert(x.innerHTML);
  }
</script></head><body>
<h1 id="myHeader" onclick="getValue()">Click me!</h1>
</body>
</html>
```

o/p: Click me!

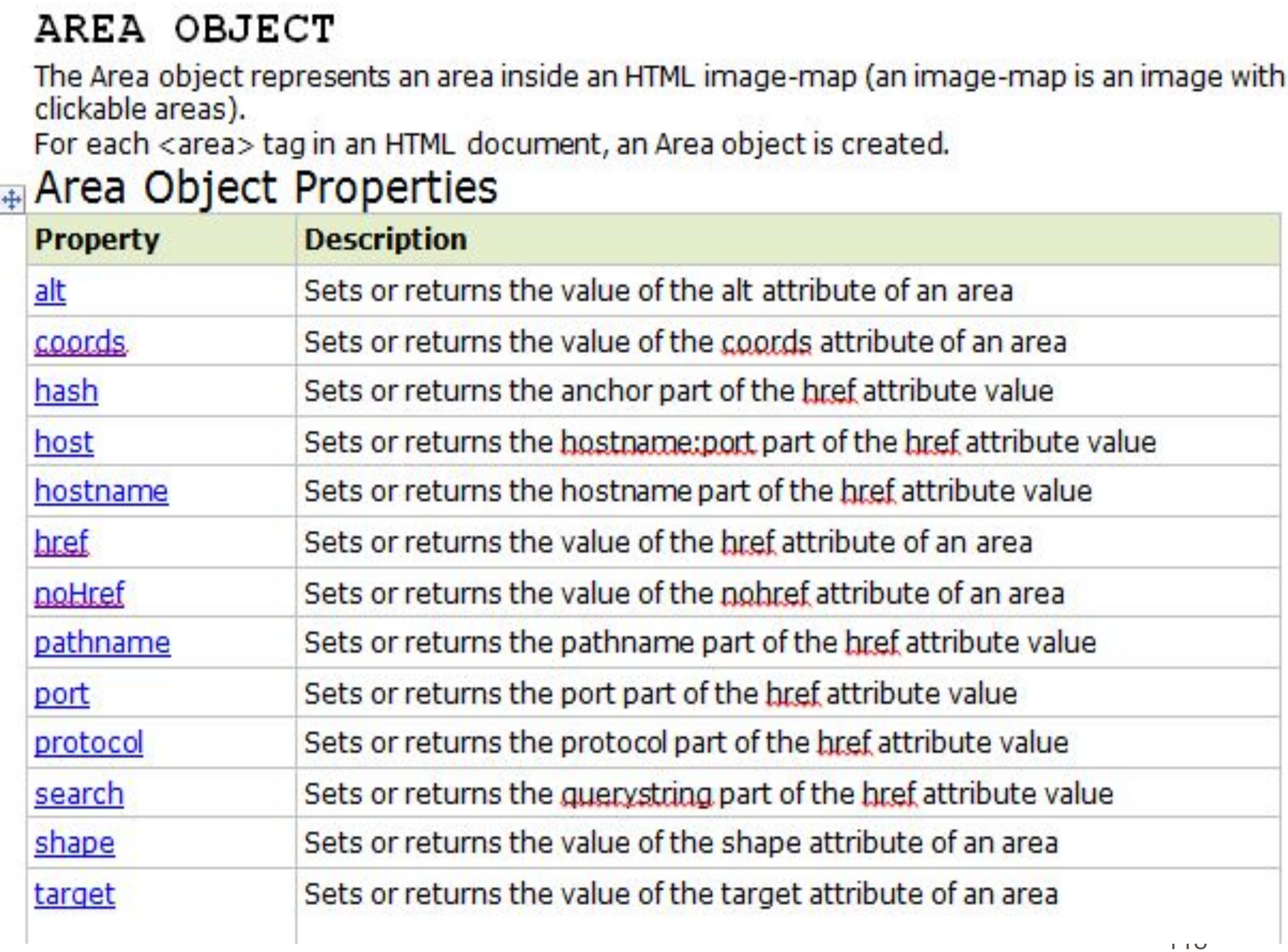
Link Object

The Link object represents an HTML link element.
A link element defines the relationship between two linked documents.
The link element is defined in the head section of an HTML document.

Link Object Properties

Property	Description
charset	Sets or returns the character encoding of the target URL
disabled	Sets or returns whether or not the target URL should be disabled
href	Sets or returns the URL of a linked resource
hreflang	Sets or returns the base language of the target URL
media	Sets or returns on what device the document will be displayed
name	Sets or returns the name of a <link> element
rel	Sets or returns the relationship between the current document and the target URL
rev	Sets or returns the relationship between the target URL and the current document
type	Sets or returns the MIME type of the target URL

```
html><head>
<link rel="stylesheet" type="text/css"
hreflang="us-en" id="style1" href="try_dom_link.css" />
</head><body>
<script type="text/javascript">
var x=document.getElementById("style1");
document.write("Language code=" + x.hreflang);
</script></body></html>
```

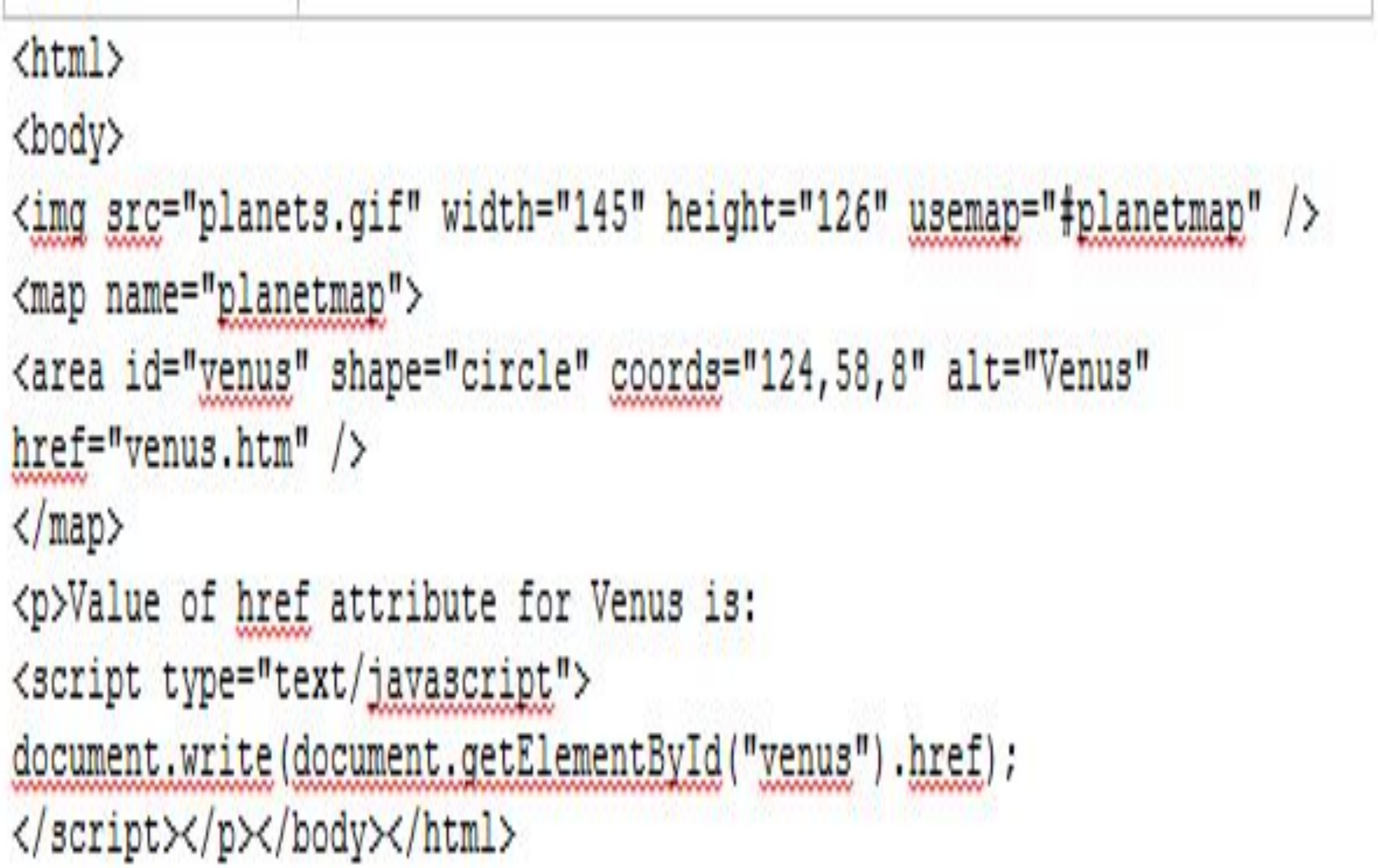


AREA OBJECT

The Area object represents an area inside an HTML image-map (an image-map is an image with clickable areas).
For each <area> tag in an HTML document, an Area object is created.

Area Object Properties

Property	Description
alt	Sets or returns the value of the alt attribute of an area
coords	Sets or returns the value of the coords attribute of an area
hash	Sets or returns the anchor part of the href attribute value
host	Sets or returns the hostname:port part of the href attribute value
hostname	Sets or returns the hostname part of the href attribute value
href	Sets or returns the value of the href attribute of an area
noHref	Sets or returns the value of the nohref attribute of an area
pathname	Sets or returns the pathname part of the href attribute value
port	Sets or returns the port part of the href attribute value
protocol	Sets or returns the protocol part of the href attribute value
search	Sets or returns the querystring part of the href attribute value
shape	Sets or returns the value of the shape attribute of an area
target	Sets or returns the value of the target attribute of an area



```
<html>
<body>
<img src="planets.gif" width="145" height="126" usemap="#planetmap" />
<map name="planetmap">
<area id="venus" shape="circle" coords="124,58,8" alt="Venus"
href="venus.htm" />
</map>
<p>Value of href attribute for Venus is:
<script type="text/javascript">
document.write(document.getElementById("venus").href);
</script></p></body></html>
```

ANCHOR OBJECT

The Anchor object represents an HTML hyperlink.

Anchor Object Properties

Property	Description
charset	Sets or returns the value of the charset attribute of a link
href	Sets or returns the value of the href attribute of a link
hreflang	Sets or returns the value of the hreflang attribute of a link
name	Sets or returns the value of the name attribute of a link
rel	Sets or returns the value of the rel attribute of a link
rev	Sets or returns the value of the rev attribute of a link
target	Sets or returns the value of the target attribute of a link
type	Sets or returns the value of the type attribute of a link

```
<html>
<body>
<p><a id="w3s" href="http://www.w3schools.com/">W3Schools.com</a></p>
<script type="text/javascript">
document.write(document.getElementById("w3s").href);
</script></body></html>
```

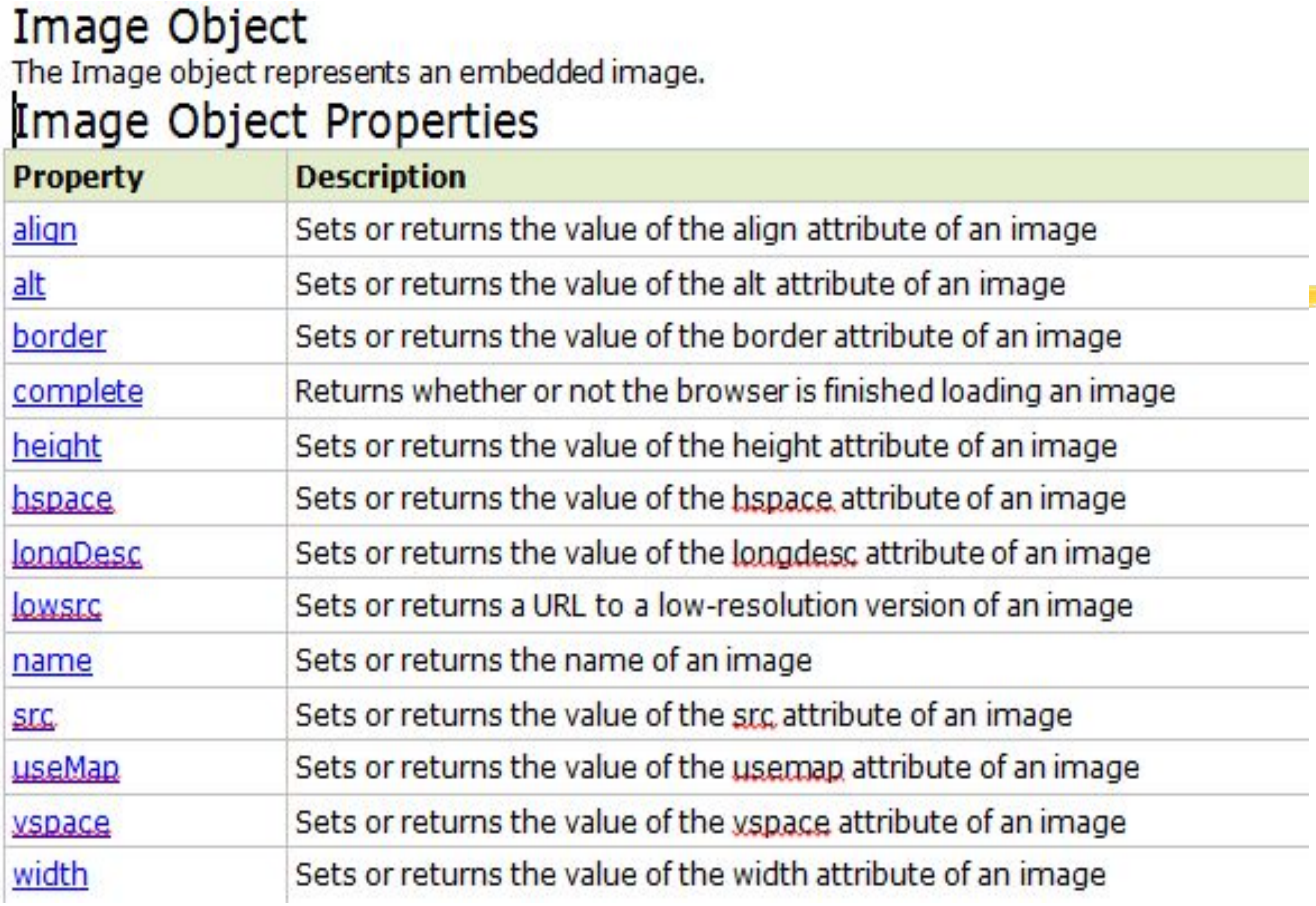


Image Object

The Image object represents an embedded image.

Image Object Properties

Property	Description
align	Sets or returns the value of the align attribute of an image
alt	Sets or returns the value of the alt attribute of an image
border	Sets or returns the value of the border attribute of an image
complete	Returns whether or not the browser is finished loading an image
height	Sets or returns the value of the height attribute of an image
hspace	Sets or returns the value of the hspace attribute of an image
longDesc	Sets or returns the value of the longdesc attribute of an image
lowsrc	Sets or returns a URL to a low-resolution version of an image
name	Sets or returns the name of an image
src	Sets or returns the value of the src attribute of an image
useMap	Sets or returns the value of the usemap attribute of an image
vspace	Sets or returns the value of the vspace attribute of an image
width	Sets or returns the value of the width attribute of an image

```html
<html>
<head>
<script type="text/javascript">
function changeSrc()
  {
  document.getElementById("myImage").src="hackanm.gif";
  }
</script>
</head>
<body>
<img id="myImage" src="compman.gif" width="107" height="98" />
<br /><br />
<input type="button" onclick="changeSrc()" value="Change image"
/>
</body>
</html>
```

<applet> tag

The <applet> tag defines an embedded applet.

Attribute	Value	Description
code	*URL*	Specifies the file name of a Java applet
object	*name*	Specifies a reference to a serialized representation of an applet

Optional Attributes

Attribute	Value	Description
align	Left,right,top bottom,middle baseline	Specifies the alignment of an applet according to surrounding elements
alt	*text*	Specifies an alternate text for an applet
archive	*URL*	Specifies the location of an archive file
codebase	*URL*	Specifies a relative base URL for applets specified in the code attribute
height	*pixels*	Specifies the height of an applet
hspace	*pixels*	Defines the horizontal spacing around an applet
name	*name*	Defines the name for an applet (to use in scripts)
vspace	*pixels*	Defines the vertical spacing around an applet
width	*pixels*	Specifies the width of an applet

```
<applet code="Bubbles.class" width="350" height="350">
Java applet that draws animated bubbles.
</applet>
```

Events

Events are actions that can be detected by JavaScript.

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

Events

- **onLoad and onUnload**
- **onFocus, onBlur and onChange**
  - **<input type="text" size="30" id="email" onchange="checkEmail()">**
- **onSubmit**
  - **<form method="post" action="xxx.htm" onsubmit="return checkForm()">**
- **onMouseOver and onMouseOut**
  - **<a href="http://www.w3schools.com" onmouseover="alert('An onMouseOver event');return false"><img src="w3s.gif" alt="W3Schools" /></a>**

```
<html><body>
<img onmouseover="bigImg(this)" onmouseout="normalImg(this)"
border="0" src="smiley.jpg" alt="Smiley" width="32" height="32">
<p>The function bigImg() is triggered when the user moves the mouse
pointer over the image.</p>
<p>The function normalImg() is triggered when the mouse pointer is
moved out of the image.</p>

<script>
function bigImg(x) {
  x.style.height = "64px";
  x.style.width = "64px";
}

function normalImg(x) {
  x.style.height = "32px";
  x.style.width = "32px";
}
</script>

</body>
</html>
```

Image onabort Event

```
<html>
<head>
<script type="text/javascript">
function abortImage()
{
alert('Error: Loading of the image was aborted!')
}
</script></head><body>
<img src="image_w3default.gif" onabort="abortImage()" />
</body></html>
```

onclick event

```
<html><body>
Field1: <input type="text" id="field1" value="Hello
World!">
<br />
Field2: <input type="text" id="field2">
<br /><br />
Click the button below to copy the content of Field1 to
Field2.
<br />
<button onclick="document.getElementById('field2').value=
document.getElementById('field1').value">Copy
Text</button>
</body></html>
```

onkeydown Event

```
<html><body>
<script type="text/javascript">
function noNumbers(e)
{
var keynum
var keychar
var numcheck
if(window.event) // IE
{
keynum = e.keyCode
}
else if(e.which) // Netscape/Firefox/Opera
{
keynum = e.which
}
keychar = String.fromCharCode(keynum)
numcheck = /\d/
return !numcheck.test(keychar)
}
</script><form>
<input type="text" onkeypress="return noNumbers(event)" />
</form></body></html>
```

onkeyup event

```
<html>
<head>
<script type="text/javascript">
function upperCase(x)
{
var y=document.getElementById(x).value
document.getElementById(x).value=y.toUpperCase()
}
</script></head><body>
Enter your name: <input type="text"
id="fname" onkeyup="upperCase(this.id)">
</body> </html>
```

Mouse events

- **onmousedown Event**
- **onmousemove Event**
- **onmouseup Event**

Events

onselect Event

```
<form>
Select text: <input type="text" value="Hello world!"
onselect="alert('You have selected some of the text.')">
</form>
```

onresize Event

```
<body onresize="alert('You have changed the size of the window')">
```

Form onreset Event

```
<form onreset="alert('The form will be reset')">
Firstname: <input type="text" name="fname" value="Donald" /><br />
Lastname: <input type="text" name="lname" value="Duck" /><br /><br
/>
<input type="reset" value="Reset" />
</form>
```

```
<html>
<head>
<script>
function WhichButton(event)
{
alert("You pressed button: " + event.button)
}
</script>
</head>
<body>

<div onmousedown="WhichButton(event)">Click this text (with one of your mouse-buttons)
<p>
0      Specifies the left mouse-button<br>
1      Specifies the middle mouse-button<br>
2      Specifies the right mouse-button</p>
<p><strong>Note:</strong> Internet Explorer 8, and earlier, returns another result:<br>
1      Specifies the left mouse-button<br>
4      Specifies the middle mouse-button<br>
2      Specifies the right mouse-button</p>

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

```
<html>
<head>
<script>
function myFunction()
{
var x=document.getElementById("fname");
x.value=x.value.toUpperCase();
}
</script>
</head>
<body>
<p>A function is triggered when the user releases a key in the input
field. The function transforms the character to upper case.</p>
Enter your name:
<input type="text" id="fname" onkeyup="myFunction()">
</body>
</html>
```

```
<html>
<head>
<script>
function displayDate()
{
document.getElementById("demo").innerHTML=Date();
}
</script>
</head>
<body>
<h1>My First JavaScript</h1>
<p id="demo">This is a paragraph.</p>
<button type="button" onclick="displayDate()">Display Date</button>
</body>
</html>
```

```
<html>
<head>
<script>
function show_coords(event)
{
var x=event.clientX;
var y=event.clientY;
alert("X coords: " + x + ", Y coords: " + y);
}
</script>
</head>
<body>
<p onmousedown="show_coords(event)">Click this paragraph, and an
alert box will alert the x and y coordinates of the mouse pointer.</p>
</body>
</html>
```

```
<html>
<head><script>
function isKeyPressed(event)
{
if (event.shiftKey==1)
  {
  alert("The shift key was pressed!");
  }
else
  {
  alert("The shift key was NOT pressed!");
  }
}
</script></head>
<body onmousedown="isKeyPressed(event)">
<p>Click somewhere in the document. An alert box will tell you if you
pressed the shift key or not.</p>
</body></html>
```