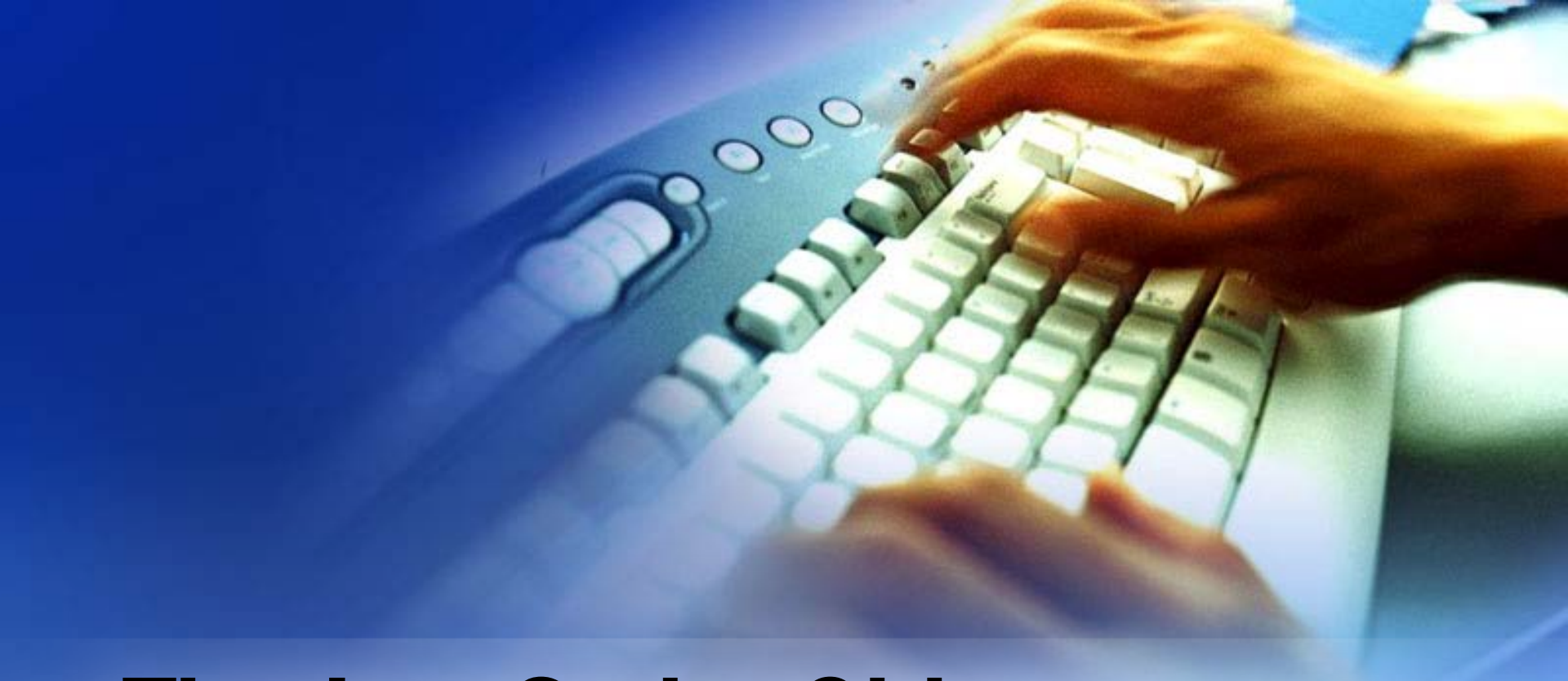


The JavaScript Objects

Bok, Jong Soon
javaexpert@nate.com
www.javaexpert.co.kr

Introduction to JavaScript Objects

- JavaScript's fundamental datatype is the *object*.
- JavaScript Objects
  - Built-in Objects(Basic Objects)
  - Browser Object Model(BOM) Objects
  - Document Object Model(DOM) Objects
  - User-defined Objects

JavaScript Object Overview

- JavaScript basic objects encapsulate the primitive types.
- They provides additional functionality.
- `String` for strings, `Boolean` for booleans, `Number` for numbers.
- Three additional built-in objects : `Math`, `Date`, and `RegExp`.
- JavaScript also has one built-in aggregator object : `Array`.

JavaScript Basic Objects

- Have methods and properties.
- A JavaScript object inherits the properties of another object, known as its *prototype*.
- Can access using the object property operator(.).
- Create implicitly instance.

```
var myName = “Shelley”;
alert(myName.length);
alert(myName.strike());
//returns <strike>Shelley</strike>
```

JavaScript Basic Objects (Cont.)

- Also can explicitly create an object using the **new** keyword.

```
var myName = new String("Shelley");
```

- If omit the **new** keyword, will get a string primitive.

```
var myName = String("Shelley");
```

- If create a **String** object instance, can access the primitive value using **valueOf**.

```
var myName = new String("Shelley");
alert(myName.valueOf());
alert(myName);
```

JavaScript Basic Objects (Cont.)

```
<script type="text/javascript">
        var myName = "Shelley";   //primitive type
        document.write(typeof myName + "<br />");    //print string

        var myName = new String("Shelley");
        document.write(typeof myName + "<br />");    //print object
        document.write(typeof myName.valueOf());     //print string
</script>
```

Creating Objects

- Objects can be create:
  - With object literals.
  - With the `new` keyword.
  - With the `Object.create()` function in ECMAScript 5.

Object Literals

- Is the easiest way.
- Is consist of a comma-separated list of colon-separated `name:value` pairs, enclosed within curly braces(`{}`).
- A property name is a JavaScript *identifier* or a string literal (the empty string is allowed).
- A property value is any JavaScript expression;
  - The value of the expression (it may be a primitive value or an object value) becomes the value of the property.

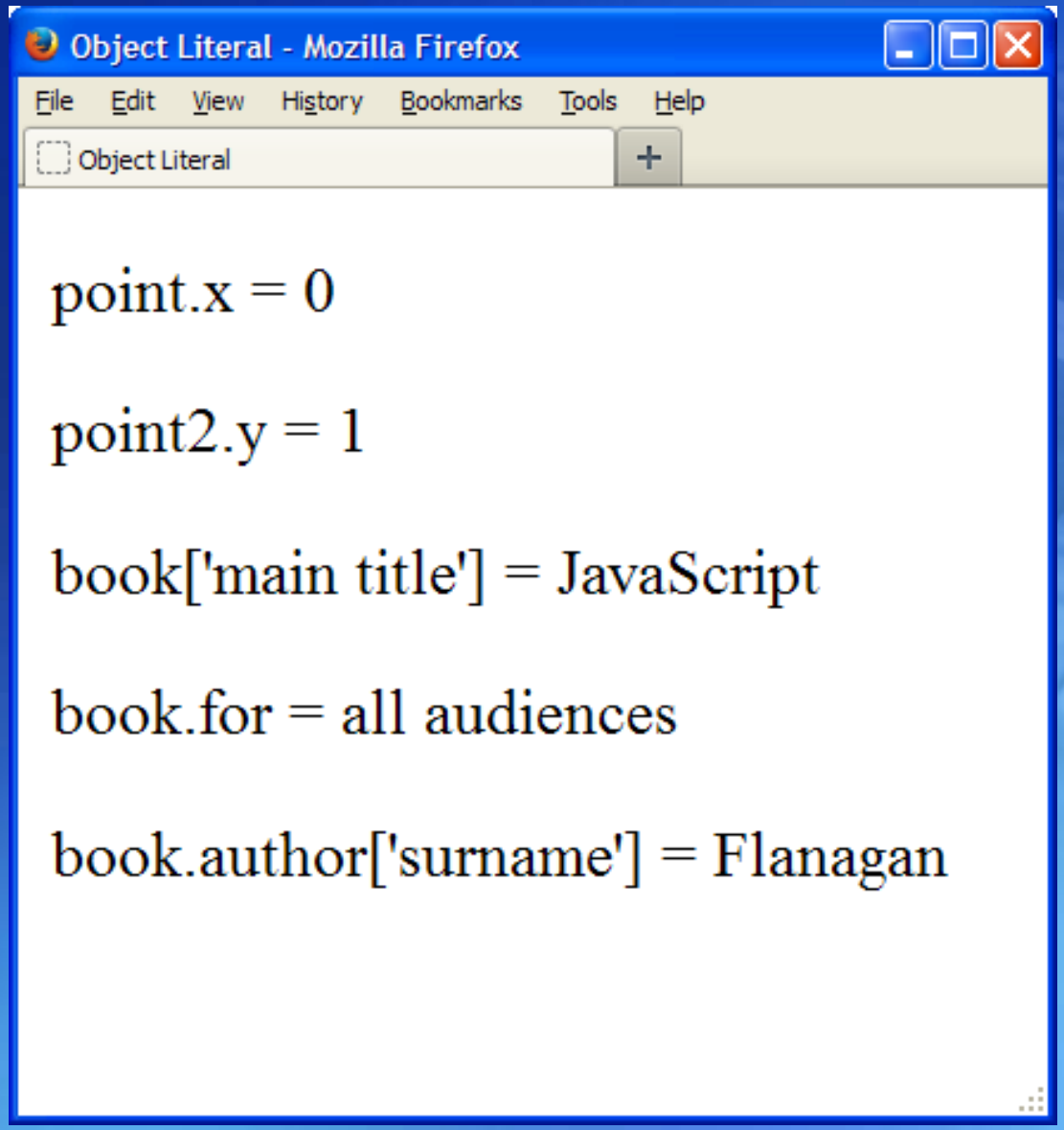
Lab1 : Object Literals

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - objectliteral.html

Lab1 : objectliteral.html

```
4       <title> Object Literal </title>
5       <meta charset="utf-8">
6   </head>
7   <body>
8       <script>
9           var empty = {};                          // An object with no properties
10          var point = { x:0, y:0 };                // Two properties
11          var point2 = { x:point.x, y:point.y+1 }; // More complex values
12          var book = {
13              "main title": "JavaScript",          // Property names include spaces,
14              'sub-title': "The Definitive Guide", // and hyphens, so use string literals
15              "for": "all audiences",              // for is a reserved word, so quote
16              author: {                            // The value of this property is
17                  firstname: "David",              // itself an object. Note that
18                  surname: "Flanagan"              // these property names are unquoted.
19              }
20          };
21          document.write("<p>point.x = " + point.x + "</p>");
22          document.write("<p>point2.y = " + point2.y + "</p>");
23          document.write("<p>book['main title'] = " + book['main title'] + "</p>");
24          document.write("<p>book.for = " + book.for + "</p>");
25          document.write("<p>book.author['surname'] = " + book.author.surname + "</p>");
26      </script>
```

Lab1 : Result


Object Literal - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Object Literal
point.x = 0
point2.y = 1
book['main title'] = JavaScript
book.for = all audiences
book.author['surname'] = Flanagan

Creating Objects with new

- The **new** operator creates and initializes a new object.
- The **new** keyword must be followed by a function invocation.
- A function used in this way is called a *constructor* and serves to initialize a newly created object.

```
var o = new Object();
var a = new Array();
var d = new Date();
var r = new RegExp("js");
```

The `Boolean` Object

- Is used to convert a non-Boolean value to a `Boolean` value(`true` or `false`).
- Is an object wrapper for a boolean value.
- Create a object

```
var myBoolean = new Boolean();
```

- Set to `false`
  - If the `Boolean` object has no initial value.
  - If the passed value is one of the following :
    - `0`, `-0`, `null`, `""`, `false`, `undefined`, `NaN`
- Set to `true`
  - Any other value(even with the string `"false"`).

The Boolean Object - Methods

- **toString()**
  - Converts a **Boolean** value to a string, and returns the result

```
<script type="text/javascript">
  var bool = new Boolean();
  document.write(bool + "<br />");     //print false

  bool = new Boolean(1);
  document.write(bool + "<br />");     //print true
  document.write(bool.toString());     //print true
</script>
```

The Boolean Object – Methods (Cont.)

- **valueOf()**
  - Returns the primitive value of a **Boolean** object

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

  var bool = new Boolean(0);
  document.write(bool.valueOf() + "<br />");          //print false
  document.write(typeof bool.valueOf() + "<br />");   //print boolean

  document.write(bool.toString() + "<br />");         //print false
  document.write(typeof bool.toString() + "<br />");  //print string

  document.write(bool + "<br />");                    //print false
  document.write(typeof bool);                        //print object

</script>
```

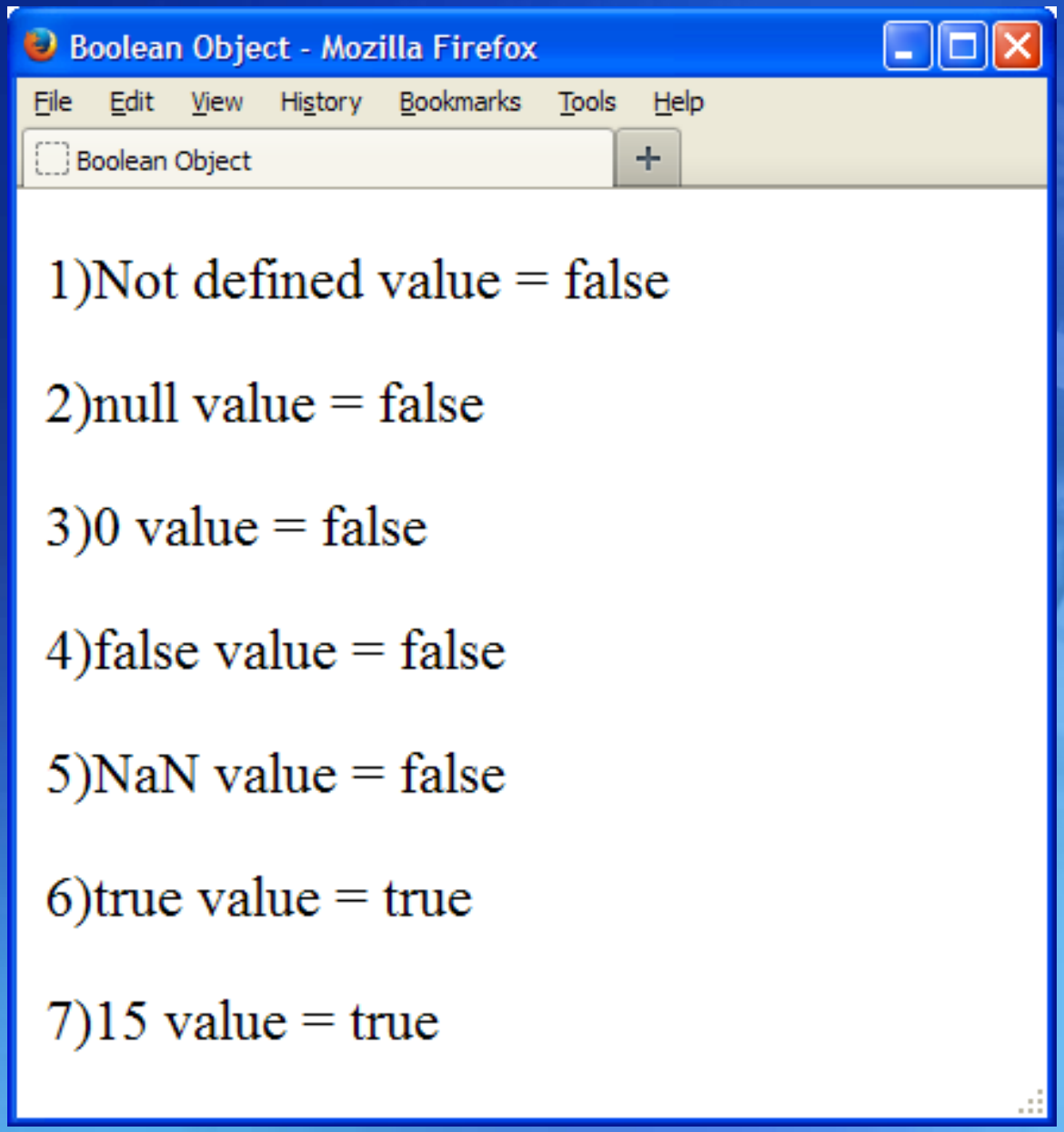
Lab2 : Boolean Object

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - boolean.html

Lab2 : boolean.html

```
8   <script>
9       ex1 = new Boolean();
10      document.write("<p>1)Not defined value = " + ex1 + "</p>");
11
12      ex2 = new Boolean(null);
13      document.write("<p>2)null value = " + ex2 + "</p>");
14
15      ex3 = new Boolean(0);
16      document.write("<p>3)0 value = " + ex3 + "</p>");
17
18      ex4 = new Boolean(false);
19      document.write("<p>4)false value = " + ex4 + "</p>");
20
21      ex5 = new Boolean(NaN);
22      document.write("<p>5)NaN value = " + ex5 + "</p>");
23
24      ex6 = new Boolean(true);
25      document.write("<p>6)true value = " + ex6 + "</p>");
26
27      ex7 = new Boolean(15);
28      document.write("<p>7)15 value = " + ex7 + "</p>");
29
30  </script>
```

Lab2 : Result


Boolean Object - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Boolean Object
1)Not defined value = false
2)null value = false
3)0 value = false
4)false value = false
5)NaN value = false
6)true value = true
7)15 value = true

The Number Object

- Creates a wrapper object to allow to work with numerical values.
- JavaScript numbers can be written with, or without decimals.
- JavaScript does *not* define different types of numbers, like integers, short, long, floating-point etc.
- All numbers are stored as 64-bit(8-Bytes) base 10, floating point numbers.

```
var x = 123e5;          //1230000
var y = 123e-5;         //0.00123
```

The `Number` Object - Properties

- **`MAX_VALUE`**
  - Returns the largest number possible in JavaScript
  - Has a value of approximately *1.79E+308(1.7976931348623157e+308)*.
  - Values larger than **`MAX_VALUE`** are represented as *Infinity*.
  - Is a *static* property.

The `Number` Object – Properties (Cont.)

- `MIN_VALUE`
  - Returns the smallest number possible in JavaScript
  - Has a value of approximately *5e-324*.
  - Values smaller than `MIN_VALUE` are converted to `0`.
  - Is a *static* property.

The `Number` Object – Properties (Cont.)

- **`NaN`**
  - Represents a *Not-a-Number* value
- **`NEGATIVE_INFINITY`**
  - Represents negative *infinity* (returned on overflow)
  - Return value is *-Infinity*.
- **`POSITIVE_INFINITY`**
  - Represents *infinity* (returned on overflow)
  - Return value is *Infinity*.

The Number Object – Properties (Cont.)

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

  document.write(Number.MAX_VALUE + "<br />");  //print 1.7976931348623157e+308
  document.write(Number.MIN_VALUE + "<br />");  //print 5e-324
  document.write(Number.NaN + "<br />");        //print NaN
  document.write(Number.NEGATIVE_INFINITY + "<br />");   //print -Infinity
  document.write(Number.POSITIVE_INFINITY);               //print Infinity

</script>
```

The Number Object – Methods

- **toExponential(x)**
  - Converts a number into an exponential notation.

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

  var num = 77.1234;

  document.write(num.toExponential() + "<br />");       //displays 7.71234e+1

  document.write(num.toExponential(4) + "<br />");      //displays 7.7123e+1

  document.write(num.toExponential(2) + "<br />");      //displays 7.71e+1

  document.write(77.1234.toExponential() + "<br />"); //displays 7.71234e+1

  document.write(77 .toExponential());                  //displays 7.7e+1

</script>
```

The Number Object – Methods (Cont.)

- **toFixed(x)**
  - Formats a number with x numbers of digits after the decimal point.

```
<script type="text/javascript">
  var su = 12345.6789;
  document.write(su.toFixed() + "<br />");            // print 12346
  document.write(su.toFixed(1) + "<br />");           // print 12345.7
  document.write(su.toFixed(6) + "<br />");           // print 12345.678900
  document.write((1.23e+20).toFixed(2) + "<br />");   // print 123000000000000000000.00
  document.write((1.23e-10).toFixed(2) + "<br />");   // print 0.00
  document.write(2.34.toFixed(1) + "<br />");         // print 2.3
  document.write(-2.34.toFixed(1) + "<br />");        // print -2.3
  document.write((-2.34).toFixed(1));                 // print -2.3
</script>
```

The Number Object – Methods (Cont.)

- **toPrecision(x)**
  - Formats a number to x length

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

  var num = 5.123456;

  document.write(num.toPrecision() + "<br />");      //displays 5.123456
  document.write(num.toPrecision(5) + "<br />");     //displays 5.1235
  document.write(num.toPrecision(2) + "<br />");     //displays 5.1
  document.write(num.toPrecision(1));     //displays 5

</script>
```

The Number Object – Methods (Cont.)

- **toString([radix])**
  - Converts a Number object to a string
  - Radix must be an integer between 2 and 36.

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

  var count = 10;
  document.write(count.toString() + "<br />");   // displays "10"
  document.write((17).toString() + "<br />");    // displays "17"

  var x = 6;
  document.write(x.toString(2) + "<br />");      // displays "110"

  document.write((254).toString(16)); // displays "fe"

</script>
```

The `Number` Object – Methods (Cont.)

- **`toLocaleString()`**
  - Convert the number into a string which is suitable for presentation in the given locale.

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

  var number = 3500000;
  document.write(number.toLocaleString());        //print 3,500,000.00

</script>
```

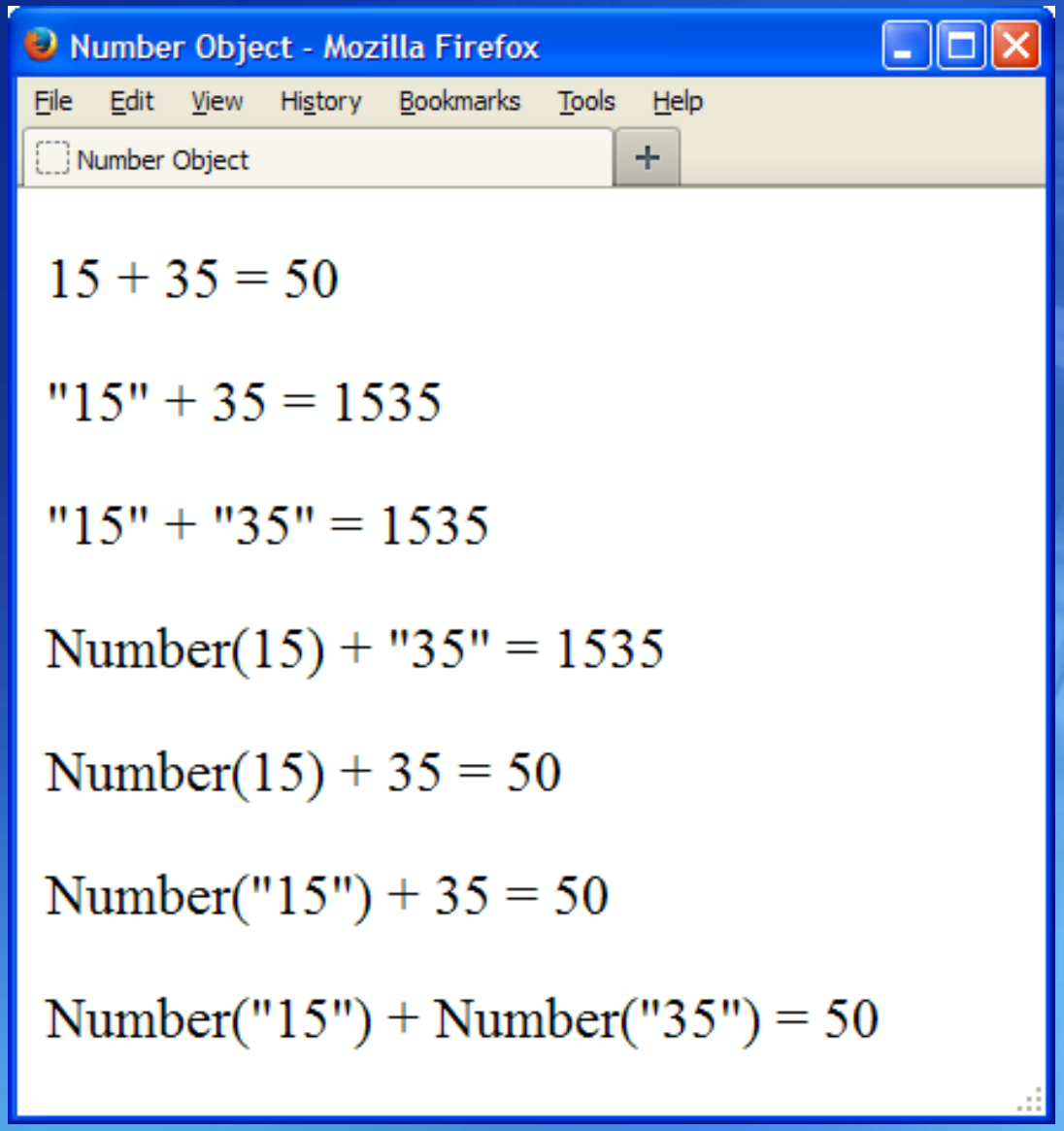
Lab3 : Number Object I

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - number.html

Lab3 : number.html

```
4       <title> Number Object </title>
5       <meta charset="utf-8">
6   </head>
7   <body>
8       <script>
9           document.write("<p>15 + 35 = " + (15 + 35) + "</p>");
10          document.write("<p>\"15\" + 35 = " + ("15" + 35) + "</p>");
11          document.write("<p>\"15\" + \"35\" = " + ("15" + "35") +
            "</p>");
12          document.write("<p>Number(15) + \"35\" = " + (Number(15) +
            "35") + "</p>");
13          document.write("<p>Number(15) + 35 = " + (Number(15) + 35)
             + "</p>");
14          document.write("<p>Number(\"15\") + 35 = " + (Number("15")
             + 35) + "</p>");
15          document.write("<p>Number(\"15\") + Number(\"35\") = " + (
            Number("15") + Number("35")) + "</p>");
16      </script>
17  </body>
```

Lab3 : Result


Number Object - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Number Object
15 + 35 = 50
"15" + 35 = 1535
"15" + "35" = 1535
Number(15) + "35" = 1535
Number(15) + 35 = 50
Number("15") + 35 = 50
Number("15") + Number("35") = 50

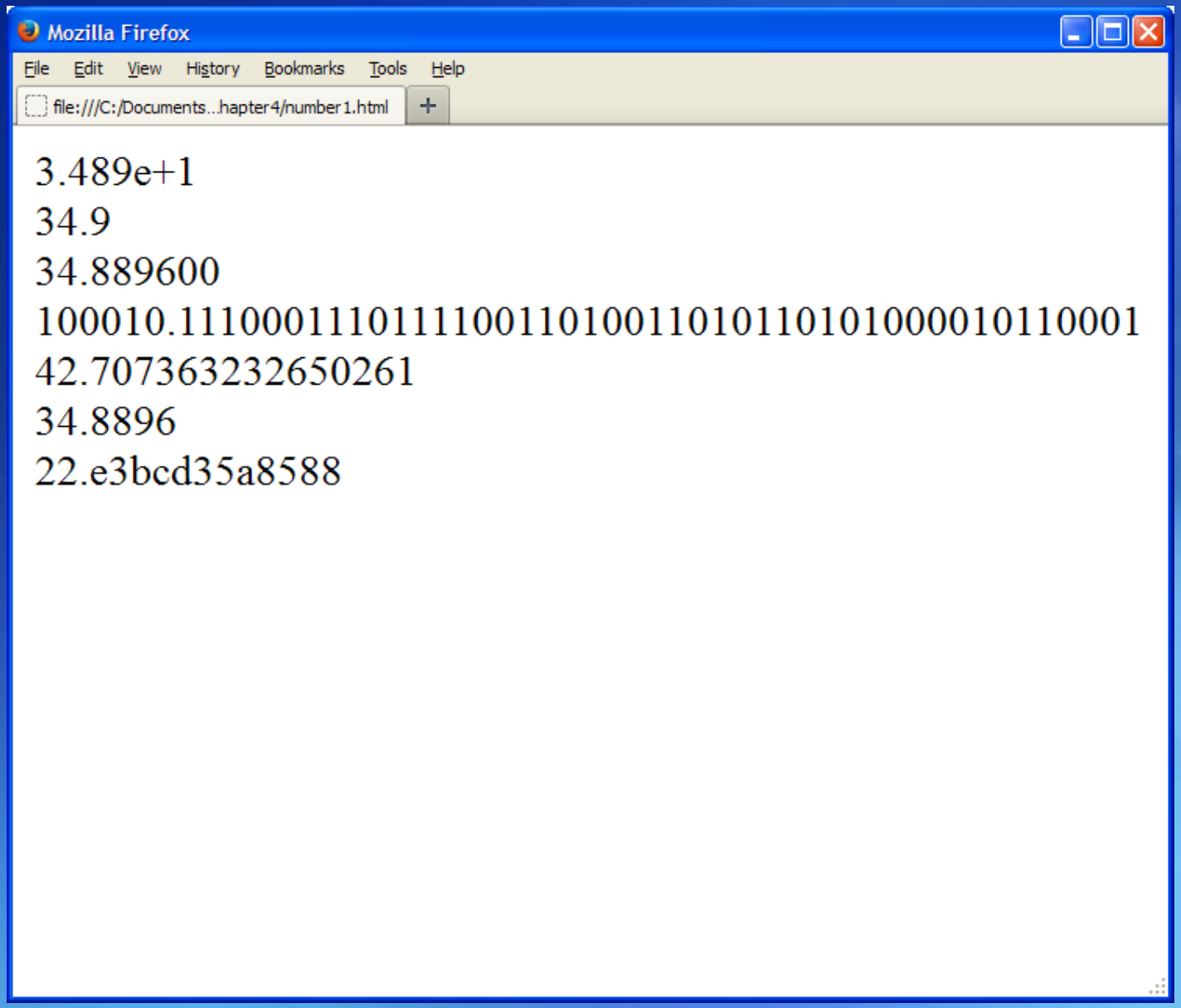
Lab4 : Number Object II

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - number1.html

Lab4 : number1.html

```
9   <script>
10
11      // Number methods
12      var newNumber = new Number(34.8896);
13
14      document.writeln(newNumber.toExponential(3) + "<br />");
15      document.writeln(newNumber.toPrecision(3) + "<br />");
16      document.writeln(newNumber.toFixed(6) + "<br />");
17
18      var newValue = newNumber.valueOf();
19
20      document.writeln(newValue.toString(2) + "<br />");
21      document.writeln(newValue.toString(8) + "<br />");
22      document.writeln(newValue.toString(10) + "<br />");
23      document.writeln(newValue.toString(16) + "<br />");
24
25  </script>
```

Lab4 : Result


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Documents...hapter4/number1.html
3.489e+1
34.9
34.889600
100010.11100011101111001101001101011010100001011000 1
42.707363232650261
34.8896
22.e3bcd35a8588

The `String` Object

- Is used to manipulate a stored piece of text.
- Is a wrapper around the string primitive data type.
- Can explicitly create a `new String` object using the `new String` constructor, passing the literal string as a parameter.

```
var str = new String("Hello, World");
```

- Has several methods, some associated with working with HTML.

The `String` Object - Properties

- **`length`**
  - Returns the length of a string.

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

    var str = "Hello, World!";
    document.write(str.length + "<br />");  //print 13
    str = "안녕하세요";
    document.write(str.length);             //print 5

</script>
```

The `String` Object - Methods

- **`charAt()`**
  - Returns the character at the specified index.
  - Syntax

    `string.charAt( index )`

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

    var anyString = "JavaScript";
    for(var i = 0 ; i < anyString.length ; i++){
       document.write(anyString.charAt(i) + " ");
       //print J a v a S c r i p t
    }

</script>
```

The `String` Object – Methods (Cont.)

- **`charCodeAt()`**
  - Returns the Unicode of the character at the specified index.
  - Syntax

    `string.charCodeAt( index )`

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

    var anyString = "JavaScript";
    for(var i = 0 ; i < anyString.length ; i++){
       document.write(anyString.charCodeAt(i) + " ");
       //print 74 97 118 97 83 99 114 105 112 116
    }

</script>
```

The `String` Object – Methods (Cont.)

- **`concat()`**
  - Joins two or more strings, and returns a copy of the joined strings.
  - Syntax

    `string.concat( string1, string2[, … , stringN )`

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

    var hello = "Hello, ";
    document.write(hello.concat("Kevin", " hava a", " nice day."));
    //Hello, Kevin hava a nice day.

</script>
```

The `String` Object – Methods (Cont.)

- **`fromCharCode()`**
  - Converts Unicode values to characters.
  - Syntax

    `String.fromCharCode( num1, …, numN )`

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

    var n = String.fromCharCode(65);
    document.write("65 = " + n + "<br />");  //print 65 = A
    n = String.fromCharCode(72, 69, 76, 76, 79);
    document.write(n);                       //print HELLO

</script>
```

The `String` Object – Methods (Cont.)

- **`indexOf()`**
  - Returns the position of the first found occurrence of a specified value in a string.
  - Syntax

    `string.indexOf( searchValue[, fromIndex] )`

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

   document.write("Blue Whale".indexOf("Blue") + "<br />");    //print 0
   document.write("Blue Whale".indexOf("Blute") + "<br />");    //print -1
   document.write("Blue Whale".indexOf("Whale", 0) + "<br />");    //print 5
   document.write("Blue Whale".indexOf("Whale", 5) + "<br />");    //print 5
   document.write("Blue Whale".indexOf("", 9) + "<br />");    //print 9
   document.write("Blue Whale".indexOf("", 10) + "<br />");    //print 10
   document.write("Blue Whale".indexOf("", 11));    //print 10

</script>
```

The `String` Object – Methods (Cont.)

- **`lastIndexOf()`**
  - Returns the position of the last found occurrence of a specified value in a string.
  - Syntax

    `string.lastIndexOf( searchValue[, fromIndex] )`

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

    document.write("canal".lastIndexOf("a") + "<br />");     //print 3
    document.write("canal".lastIndexOf("a", 2) + "<br />"); //print 1
    document.write("canal".lastIndexOf("a", 0) + "<br />"); //print -1
    document.write("canal".lastIndexOf("x") + "<br />");     //print -1
    var str = "Blue Whale, Killer Whale";
    document.write(str.lastIndexOf("blue"));                 //print -1

</script>
```

The `String` Object – Methods (Cont.)

- **`match()`**
  - Searches for a match between a regular expression and a string, and returns the matches.
  - Syntax

    `string.match( regexp )`

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

    var str = "The rain in SPAIN stays mainly in the plain";
    var result = str.match(/ain/g);
    document.write(result + "<br />");  //print ain, ain, ain
    var result1 = str.match(/ain/gi);     //print ain, AIN, ain, ain
    document.write(result1);
</script>
```

The `String` Object – Methods (Cont.)

- **`replace()`**
  - Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring.
  - Syntax

    `string.replace( searchValue, newValue )`

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

 var str = "Mr Blue has a blue house and a blue car";
 var newStr = str.replace(/blue/gi, "red");
 document.write(newStr);     //print Mr red has a red house and a red car

</script>
```

The `String` Object – Methods (Cont.)

- **`search()`**
  - Searches for a match between a regular expression and a string, and returns the position of the match.
  - Syntax

    `string.search( regexp )`

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

    var str = "Visit W3Schools!";
    document.write(str.search("W3Schools") + "<br />");   //print 6

    str = "Mr. Blue has a blue house";
    document.write(str.search("blue") + "<br />");        //print 15
    document.write(str.search(/blue/i));                  //print 4

</script>
```

The `String` Object – Methods (Cont.)

- **`slice()`**
  - Extracts a part of a string and returns a new string.
  - Syntax

    `string.slice( beginSlice[, endSlice] )`

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

    var str = "Hello world";
    document.write(str.slice(1, 5) + "<br />");     //print ello
    document.write(str.slice(0) + "<br />");        //print Hello world
    document.write(str.slice(3) + "<br />");        //print lo world
    document.write(str.slice(3, 8) + "<br />");     //print lo wo
    document.write(str.slice(0, 1) + "<br />");     //print H
    document.write(str.slice(-1));                  //print d
</script>
```

The `String` Object – Methods (Cont.)

- **`split()`**
  - Splits a string into an array of substrings.
  - Syntax

    `string.split( [separator][, limit] )`

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

    var str = "How are you doing today?";
    var array = str.split(" ");
    document.write(array + "<br />");     //print How,are,you,doing,today?
    array = str.split("");
    document.write(array + "<br />");
    //print H,o,w, ,a,r,e, ,y,o,u, ,d,o,i,n,g, ,t,o,d,a,y,?

    array = str.split("", 3);
    document.write(array + "<br />");     //H,o,w
    array = str.split("o");
    document.write(array);                //H,w are y,u d,ing t,day?

</script>
```

The `String` Object – Methods (Cont.)

- **`substr()`**
  - Extracts the characters from a string, beginning at a specified start position, and through the specified number of character.
  - Syntax

    `string.substr( start [, length] )`

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

    var str = "Hello world!";
    document.write(str.substr(2,3) + "<br />");     //print llo
    document.write(str.substr(2));                  //print llo world!

</script>
```

The `String` Object – Methods (Cont.)

- **`substring()`**
  - Extracts the characters from a string, between two specified indices.
  - Syntax

    `string.substring( indexA[, indexB] )`

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

    var str = "Hello world!";
    document.write(str.substring(3) + "<br />");  //print lo world!
    document.write(str.substring(3, 7));          //print lo w

</script>
```

The `String` Object – Methods (Cont.)

- **`toLowerCase() / toUpperCase()`**
  - Converts a string to lowercase / uppercase letters.
  - Syntax

```
string.toLowerCase( )
string toUpperCase( )
```

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

    var str = "Hello world!";
    document.write(str.toLowerCase() + "<br />");  //print hello world!
    document.write(str.toUpperCase());             //print HELLO WORLD!

</script>
```

The `String` Object – Methods (Cont.)

- **`valueOf()`**
  - Returns the primitive value of a String object.
  - Syntax
    `string.valueOf( )`

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

    var str = new String("Hello world!");
    document.write(typeof str + "<br />");  //print object
    var str1 = str.valueOf();
    document.write(typeof str1);            //print string

</script>
```

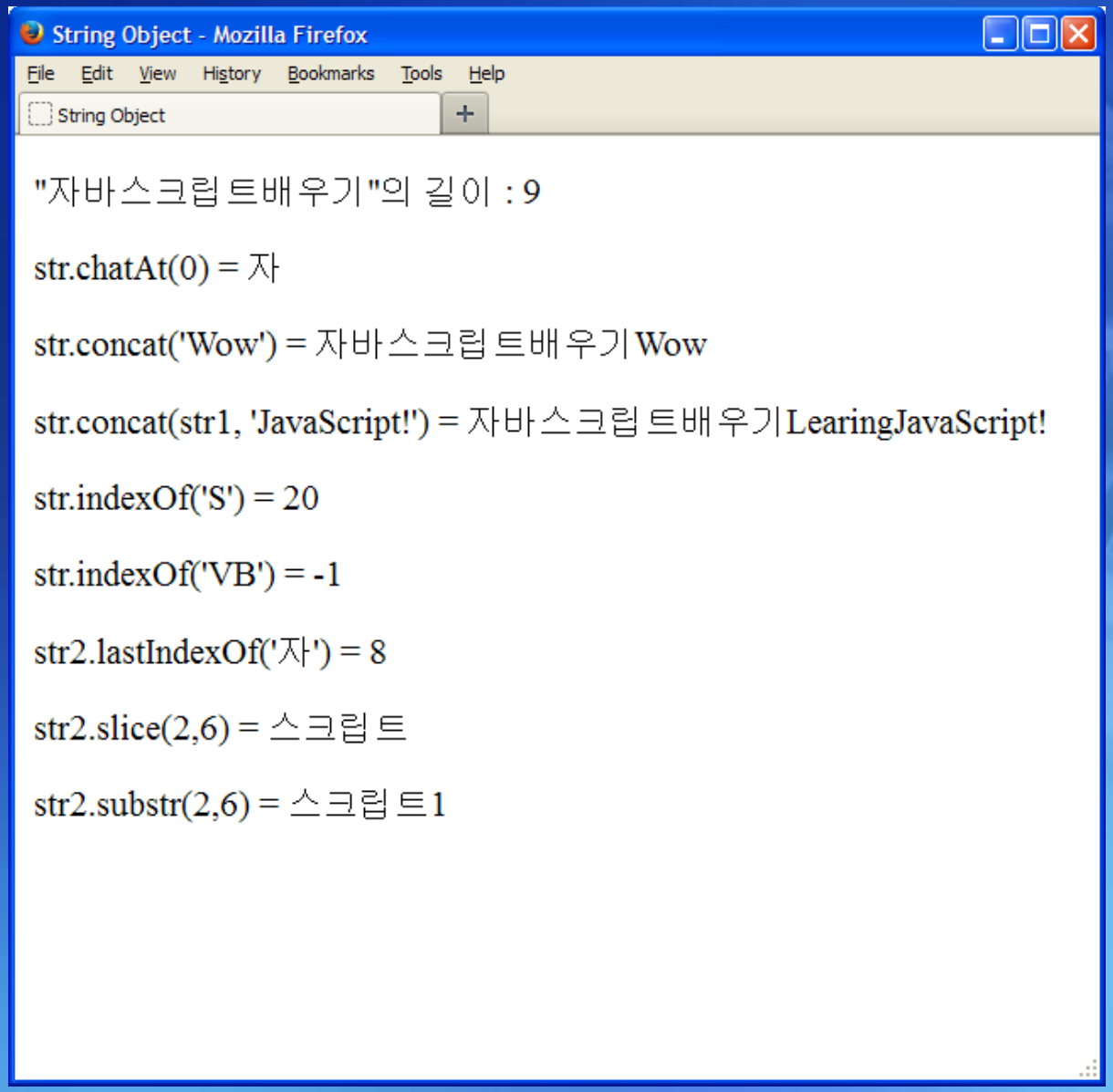
Lab5 : String Object I

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - string.html

Lab5 : string.html

```
8   <script>
9       str = new String("자바스크립트배우기");
10      document.write("<p>\"자바스크립트배우기\"의 길이 : " + str.length + "</p>");
11      document.write("<p>str.chatAt(0) = " + str.charAt(0) + "</p>");
12      document.write("<p>str.concat('Wow') = " + str.concat('Wow') + "</p>");
13
14      str1 = new String("Learing");
15      document.write("<p>str.concat(str1, 'JavaScript!') = " + str.concat(str1,
        "JavaScript!") + "</p>");
16      str = str.concat(str1, 'JavaScript!');
17      document.write("<p>str.indexOf('S') = " + str.indexOf('S') + "</p>");
18      document.write("<p>str.indexOf('VB') = " + str.indexOf('VB') + "</p>");
19
20      str2 = new String("자바스크립트1 자바스크립트2");
21      document.write("<p>str2.lastIndexOf('자') = " + str2.lastIndexOf('자') + "</p>");
22
23      document.write("<p>str2.slice(2,6) = " + str2.slice(2,6) + "</p>");
24      document.write("<p>str2.substr(2,6) = " + str2.substr(2,6) + "</p>");
25  </script>
```

Lab5 : Result


String Object - Mozilla Firefox
File Edit View History Bookmarks Tools Help
String Object
"자바스크립트배우기"의 길이 : 9
str.chatAt(0) = 자
str.concat('Wow') = 자바스크립트배우기Wow
str.concat(str1, 'JavaScript!') = 자바스크립트배우기LearingJavaScript!
str.indexOf('S') = 20
str.indexOf('VB') = -1
str2.lastIndexOf('자') = 8
str2.slice(2,6) = 스크립트
str2.substr(2,6) = 스크립트1

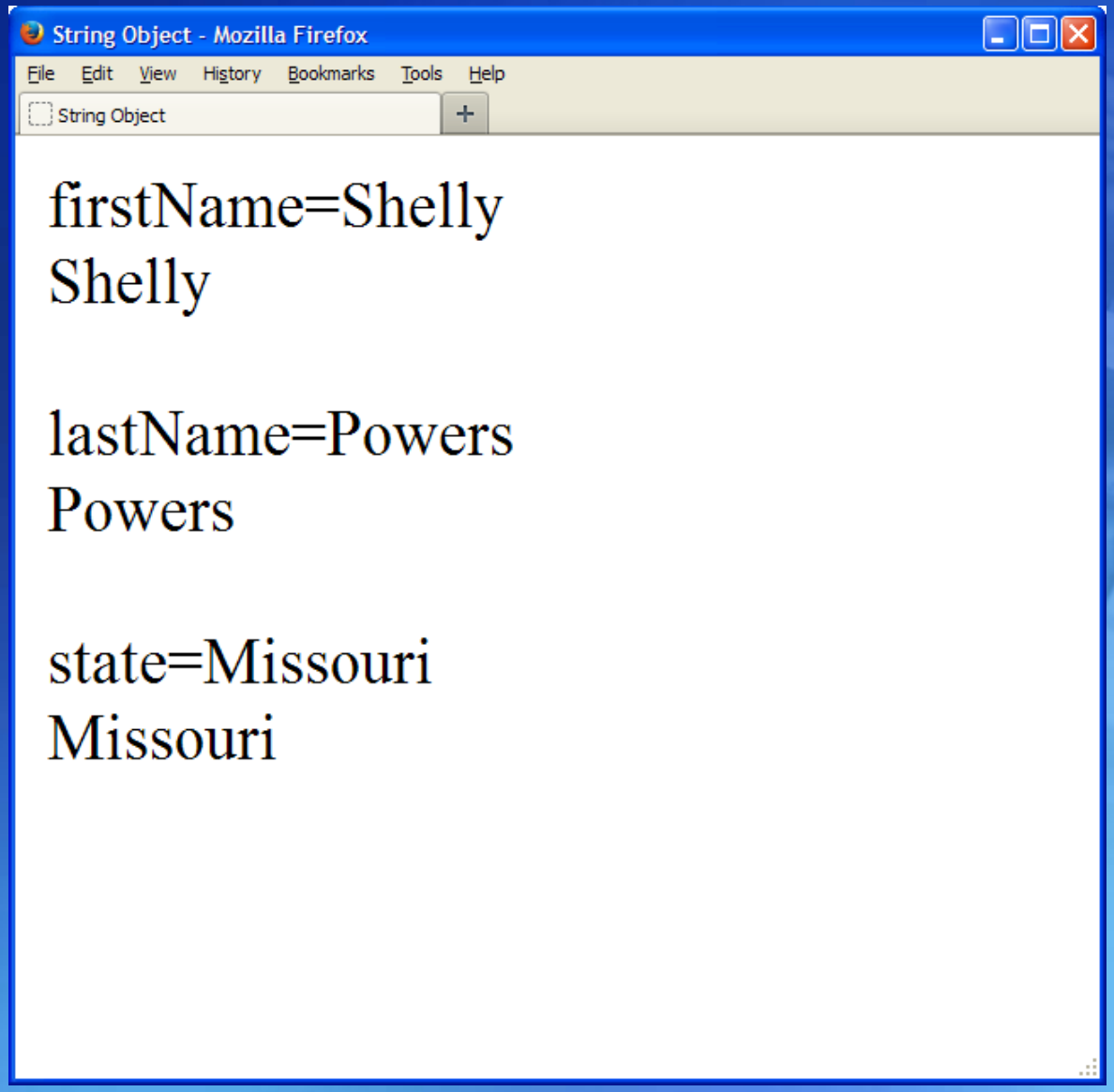
Lab6 : String Object II

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - string1.html

Lab6 : string1.html

```
 9      <script>
10         var inputString =
11            new String(
               'firstName=Shelley,lastName=Powers,state=Missouri,statement="This is a
               test,of split"');
12         var arrayTokens = inputString.split(',',3);
13
14         // process split on commas
15         for (var i = 0; i < arrayTokens.length; i++) {
16            document.writeln(arrayTokens[i] + "<br />");
17
18            // now split on equals and write just value
19            var newTokens = arrayTokens[i].split('=');
20            document.writeln(newTokens[1] + "<br /><br />");
21         }
22      </script>
```

Lab6 : Result


String Object - Mozilla Firefox
File Edit View History Bookmarks Tools Help
String Object
firstName=Shelly
Shelly
lastName=Powers
Powers
state=Missouri
Missouri

The `String` Object – HTML Wrapper Methods

- `anchor()`
- `big()`
- `blink()`
- `bold()`
- `fixed()`
- `fontcolor()`
- `fontsize()`
- `italics()`
- `link()`
- `small()`
- `strike()`
- `sub()`
- `sup()`

The `String` Object – HTML Wrapper Methods (Cont.)

- **`anchor()`**
  - Is used to create an HTML anchor.
  - Returns the string embedded in the *<a>* tag, like this:
    - *string*

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

    var txt = "Hello, World";
    txt.anchor("anchorName");
    alert(txt.anchor("anchorName"));
    //display <a name="anchorName">Hello,World</a>

</script>
```

The `String` Object – HTML Wrapper Methods (Cont.)

- `big()`
  - Displays a string using a big font.
  - `string.big()`
  - *<big>string</big>*
- `blink()`
  - Displays a blinking string.
  - `string.blink()`
  - *<blink>string</blink>*
  - Do not work in IE, Chrome, Safari

The `String` Object – HTML Wrapper Methods (Cont.)

- **`bold()`**
  - Displays a string in bold.
  - `string.bold()`
  - *string*
- **`fixed()`**
  - Displays a string using a fixed-pitch font.
  - `string.fixed()`
  - *<tt>string</tt>*

The `String` Object – HTML Wrapper Methods (Cont.)

- **`fontcolor()`**
  - Displays a string using a specified color.
  - `string.color(color)`
  - *string*
- **`fontsize()`**
  - Displays a string using a specified size.
  - `string.fontsize(size)`
  - *string*

The `String` Object – HTML Wrapper Methods (Cont.)

- `italics()`
  - Displays a string in italic.
  - `string.italics()`
  - *string*
- `link()`
  - Displays a string as a hyperlink.
  - `string.link( url )`
  - *string*

The `String` Object – HTML Wrapper Methods (Cont.)

- `small()`
  - Displays a string using a small font.
  - `string.small()`
  - *<small>string</small>*
- `strike()`
  - Displays a string with a strikethrough.
  - `string.strike()`
  - *<strike>string</strike>*

The `String` Object – HTML Wrapper Methods (Cont.)

- `sub()`
  - Displays a string as subscript text.
  - `string.sub()`
  - *_{string}*
- `sup()`
  - Displays a string as superscript text.
  - `string.sup()`
  - *^{string}*

The `String` Object – HTML Wrapper Methods (Cont.)

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

    var txt = "Hello, World";
    document.write("<p>Big : " + txt.big() + "</p>");
    document.write("<p>Small : " + txt.small() + "</p>");
    document.write("<p>Bold : " + txt.bold() + "</p>");
    document.write("<p>Italic : " + txt.italics() + "</p>");
    document.write("<p>Fixed : " + txt.fixed() + "</p>");
    document.write("<p>Strike : " + txt.strike() + "</p>");
    document.write("<p>Fontcolor : " + txt.fontcolor("green") + "</p>");
    document.write("<p>Fontsize : " + txt.fontsize(6) + "</p>");
    document.write("<p>Subscript : " + txt.sub() + "</p>");
    document.write("<p>Superscript : " + txt.sup() + "</p>");
    document.write("<p>Link : " + txt.link("http://www.javaexpert.co.kr") + "</p>");
    document.write("<p>Blink : " + txt.big() + "</p>");
    //Does not work in IE, Chrome, or Safari

</script>
```

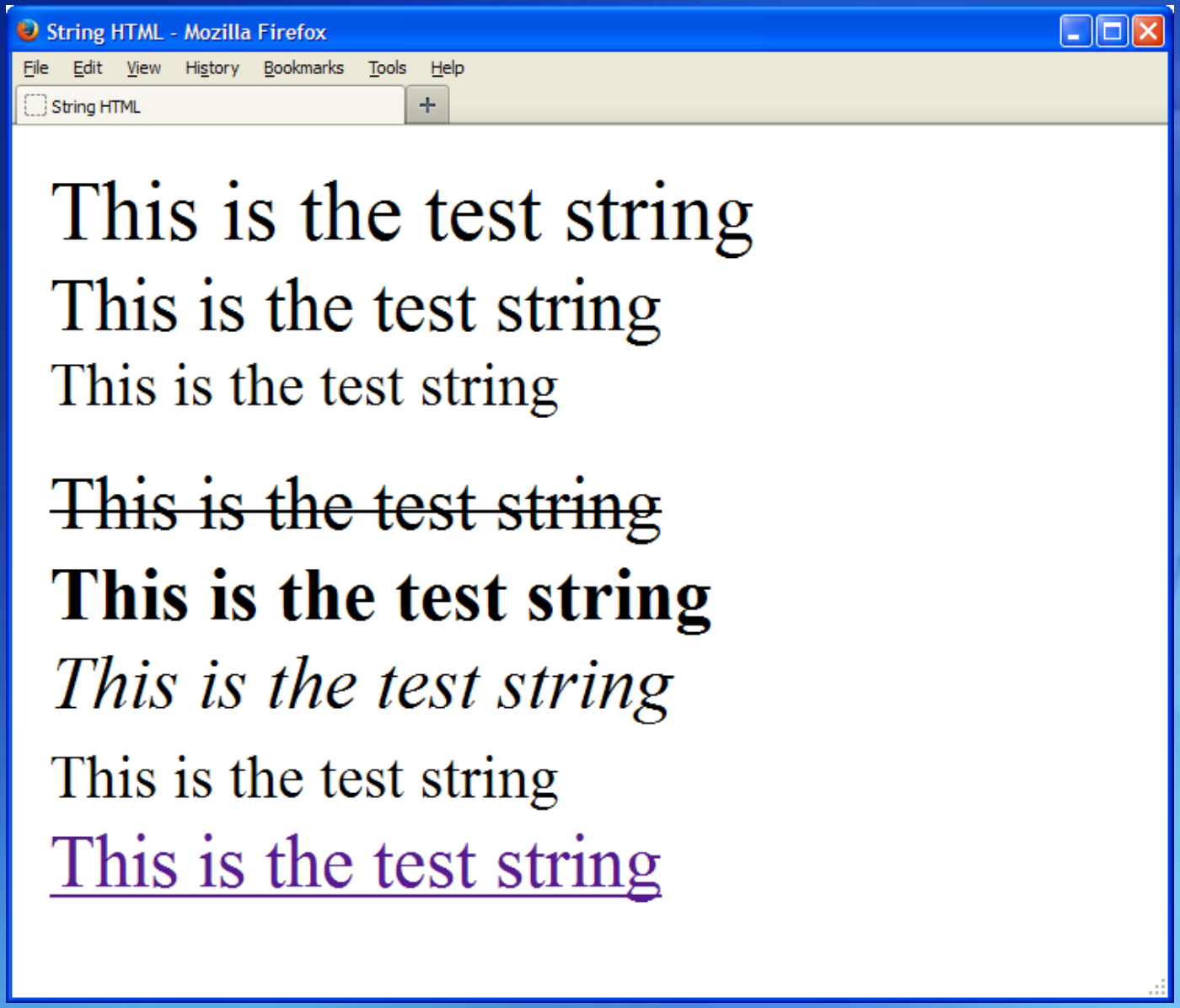
Lab7 : String Object III

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - string2.html

Lab7 : string2.html

```
 9        <script>
10            var someString = new String("This is the test string");
11
12            document.writeln(someString.big() + "<br />");
13            document.writeln(someString.blink() + "<br />");
14            document.writeln(someString.sup() + "<br />");
15            document.writeln(someString.strike() + "<br />");
16            document.writeln(someString.bold() + "<br />");
17            document.writeln(someString.italics() + "<br />");
18            document.writeln(someString.small() + "<br />");
19            document.writeln(someString.link('http://www.oreilly.com'));
20        </script>
```

Lab7 : Result


String HTML - Mozilla Firefox
File Edit View History Bookmarks Tools Help
String HTML
This is the test string
This is the test string
This is the test string
This is the test string
This is the test string
This is the test string
This is the test string
This is the test string

The RegExp Object

- Is short for regular expression.
- Describes a pattern of characters.
- Can use a pattern to describe what you are searching for.
- A simple pattern can be one single character.
- A more complicated pattern can consist of more characters, and can be used for parsing, format checking, substitution and more.
- Is used to perform powerful pattern-matching and "*search-and-replace*" functions on text.

The RegExp Object (Cont.)

- Syntax

```
var patt = new RegExp(pattern, modifiers)
or more simply:
var patt = /pattern/modifiers;
```

```
/ab+c/i;
new RegExp("ab+c", "i");
```

The RegExp Object (Cont.)

- Modifiers
  - Are used to perform case-insensitive and global serarches.
- `i`
  - Perform case-insensitive matching.
- `g`
  - Perform a global match.
  - Find all matches rather than stopping after the first match.
- `m`
  - Perform multiline matching.

The RegExp Object (Cont.)

- 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

The RegExp Object (Cont.)

- 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

The RegExp Object (Cont.)

- Metacharacters
  - Are characters with a special meaning.

Metacharacter	Description
\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

The RegExp Object (Cont.)

- 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 sequences of X n's
n{X,Y}	Matches any string that contains a sequences of X to Y n's
n{X,}	Matches any string that contains a sequences 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 specified string n
?!n	Matches any string that is not followed by a specific string n

The RegExp Object - Properties

- **global**
  - Specifies if the "**g**" modifier is set.
  - Return **true** if the "**g**" modifiers is set, otherwise it returns **false**.
  - Syntax

    RegExpObject.global

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

   var str="Visit javaexpert.com!";
   var patt1=/W3S/g;

   if(patt1.global) {        //print g modifier is set
      document.write("g modifier is set!");
   } else {
      document.write("g modifier is not set!");
   }

</script>
```

The RegExp Object – Properties (Cont.)

- **ignoreCase**
  - Specifies whether or not the "**i**" modifier is set.
  - Return **true** if the "**i**" modifiers is set, otherwise it returns **false**.
  - Syntax

    RegExpObject.ignoreCase

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

   var str="Visit javaexpert.com!";
   var patt1=/W3S/i;

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

</script>
```

The RegExp Object – Properties (Cont.)

- **lastIndex**
  - Specifies the index at which to start the next match.
  - Return an integer that specifies the character position immediately after the last match found by `exec()` or `test()` methods.
  - `exec()` and `test()` reset `lastIndex` to `0` if they do not get a match.
  - Syntax

    ```
    RegExpObject.lastIndex
    ```

The RegExp Object – Properties (Cont.)

- **lastIndex**

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

    var str="The rain in Spain stays mainly in the plain";
    var patt1=/ain/g;

    while (patt1.test(str) == true) {
        document.write("'ain' found. Index now at : " + patt1.lastIndex);
        document.write("<br />");
    }
    //print 'ain' found. Index now at : 8
    //print 'ain' found. Index now at : 17
    //print 'ain' found. Index now at : 28
    //print 'ain' found. Index now at : 43
```

The RegExp Object – Properties (Cont.)

- **multiline**
  - Specifies whether or not the "m" modifier is set.
  - Return **true** if the "m" modifiers is set, otherwise it returns **false**.
  - Syntax

    RegExpObject.multiline

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

   var str="Visit javaexpert.com!";
   var patt1=/W3S/m;

   if(patt1.multiline) {   //print m modifier is set!
      document.write("m modifier is set!");
   } else {
      document.write("m modifier is not set!");
   }

</script>
```

The RegExp Object – Properties (Cont.)

- **source**
  - Returns the text of the RegExp pattern.
  - Syntax

    RegExpObject.source

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

    var str="Visit javaexpert.com!";
    var patt1=/W3S/g;

    document.write(patt1.source);

</script>
```

The RegExp Object – Methods

- `compile()`
  - Compiles a regular expression during executing of a script.
  - Can also be used to change and recompile a regular expression.
  - Syntax

    ```
    obj.compile(regexp, modifier)
    ```

The RegExp Object – Methods

- compile()

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

    var str="Every man in the world! Every woman on earth!";
    var patt=/man/g;
    var str2=str.replace(patt,"person");
    document.write(str2+"<br />");
    //print Every person in the world! Every woperson on earth!

    patt=/(wo)?man/g;
    patt.compile(patt);
    str2=str.replace(patt,"person");
    document.write(str2);
    //print Every person in the world! Every person on earth!

</script>
```

The RegExp Object – Methods (Cont.)

- **exec()**
  - Tests for a match in a string.
  - Returns the first match.
  - Syntax

    obj.exec(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); //print Returned value : Hello

  //look for "W3Schools"
  patt=/W3Schools/g;
  result=patt.exec(str);
  document.write("<br />Returned value: " + result);  //print Returned value : null

</script>
```

The RegExp Object – Methods (Cont.)

- **test()**
  - Tests for a match in a string.
  - Returns **true** if it finds a match, otherwise it returns **false**.
  - Syntax

    obj.test(string)

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

  var str="Hello world!";
  //look for "Hello"
  var patt=/Hello/g;
  var result=patt.test(str);
  document.write("Returned value: " + result); //print Returned value : true

  //look for "W3Schools"
  patt=/W3Schools/g;
  result=patt.test(str);
  document.write("<br />Returned value: " + result);  //print Returned value : false

</script>
```

The RegExp Object – Sample Code

- Using a regular expression to change data format

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

  var pattern = /(\w+)\s(\w+)/;
  var str = "John Smith";
  var newstr = str.replace(pattern, "$2, $1");
  document.write(newstr);   //print Smith, John

</script>
```

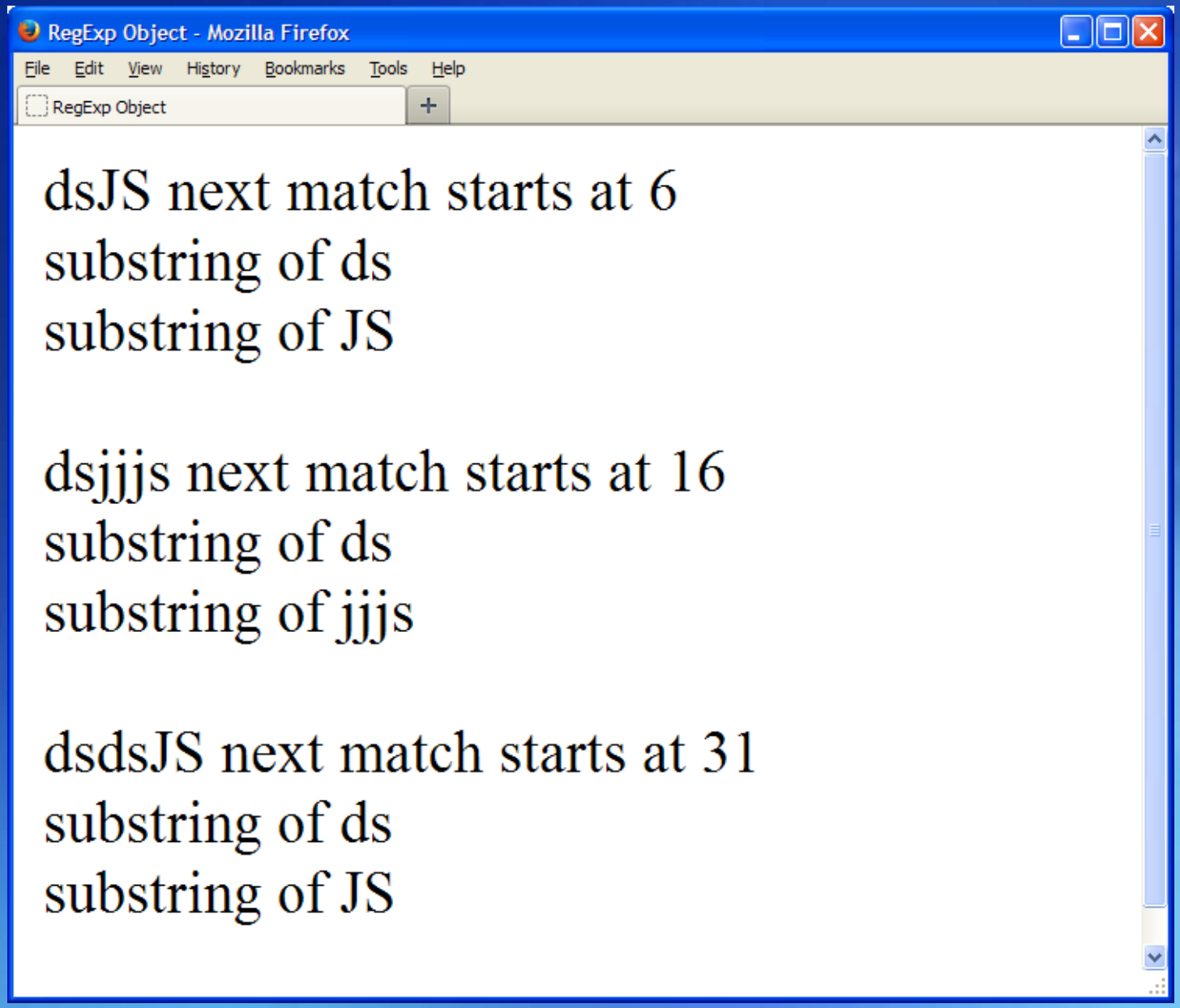
Lab8 : RegExp Object I

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - regexp.html

Lab8 : regexp.html

```
5       <title> RegExp Object </title>
6       <meta charset="utf-8">
7   </head>
8   <body>
9       <script>
10          var re = /(ds)+(j+s)/ig;
11          var str = "cfdsJS *(&dsjjjsYJSjs 888dsdsJS";
12          var resultArray = re.exec(str);
13          while (resultArray) {
14              document.writeln(resultArray[0]);
15              document.writeln(" next match starts at " + re.lastIndex + "<br />");
16              for (var i = 1; i < resultArray.length; i++) {
17                  document.writeln("substring of " + resultArray[i] + "<br />");
18              }
19              document.writeln("<br />");
20              resultArray = re.exec(str);
21          }
22      </script>
23  </body>
```

Lab8 : Result


RegExp Object - Mozilla Firefox
File Edit View History Bookmarks Tools Help
RegExp Object
dsJS next match starts at 6
substring of ds
substring of JS
dsjjjs next match starts at 16
substring of ds
substring of jjjs
dsdsJS next match starts at 31
substring of ds
substring of JS

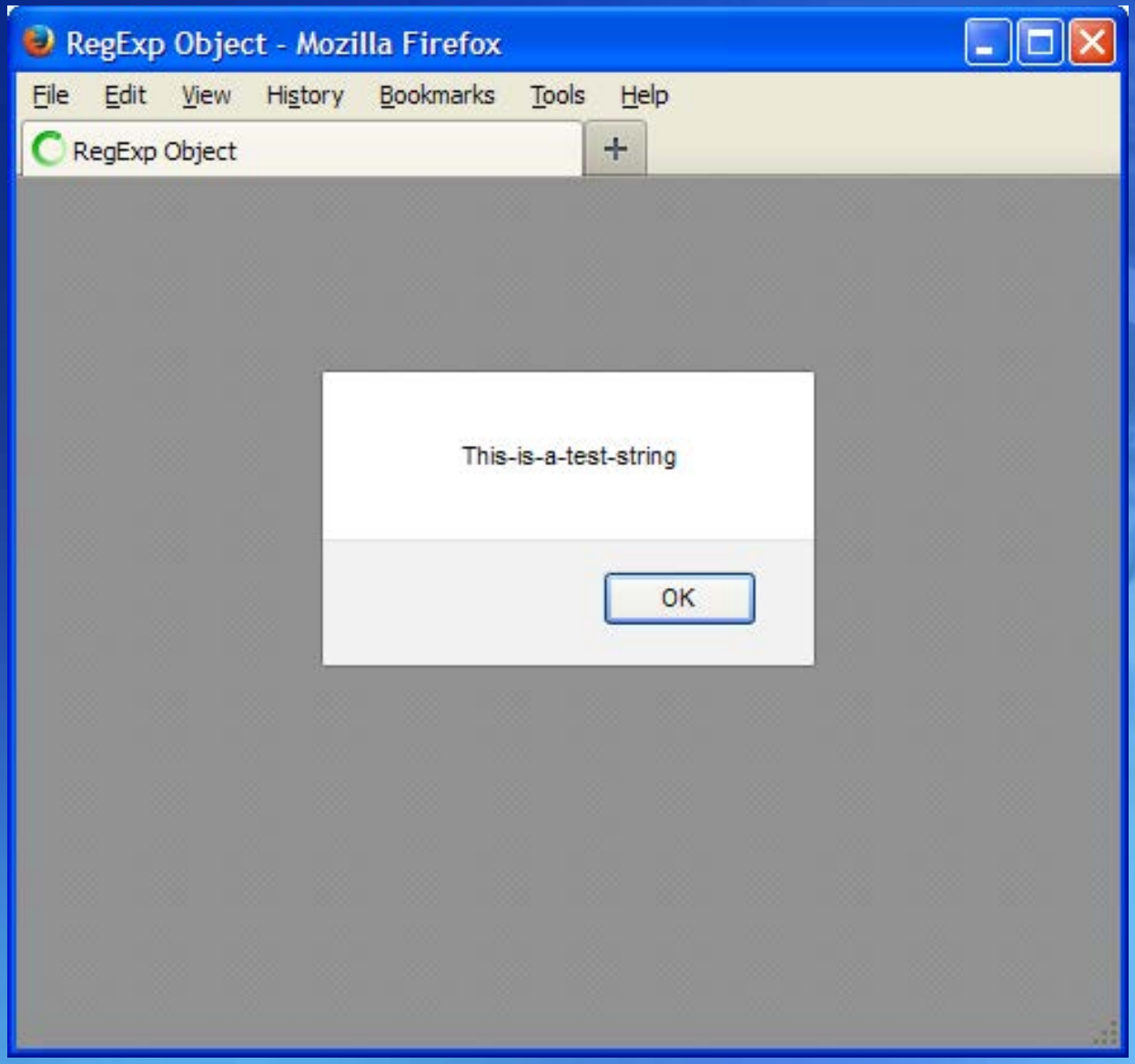
Lab9 : RegExp Object II

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - regexp1.html

Lab9 : regexp1.html

```
3  <html>
4      <head>
5          <title> RegExp Object</title>
6          <meta charset="utf-8">
7      </head>
8      <body>
9          <script>
10             var regExp = /\s\*/g;
11             var str = "This *is *a *test *string";
12             var resultString = str.replace(regExp,'-');
13             alert(resultString);
14         </script>
15     </body>
16 </html>
```

Lab9 : Result


RegExp Object - Mozilla Firefox
File Edit View History Bookmarks Tools Help
RegExp Object
This-is-a-test-string
OK

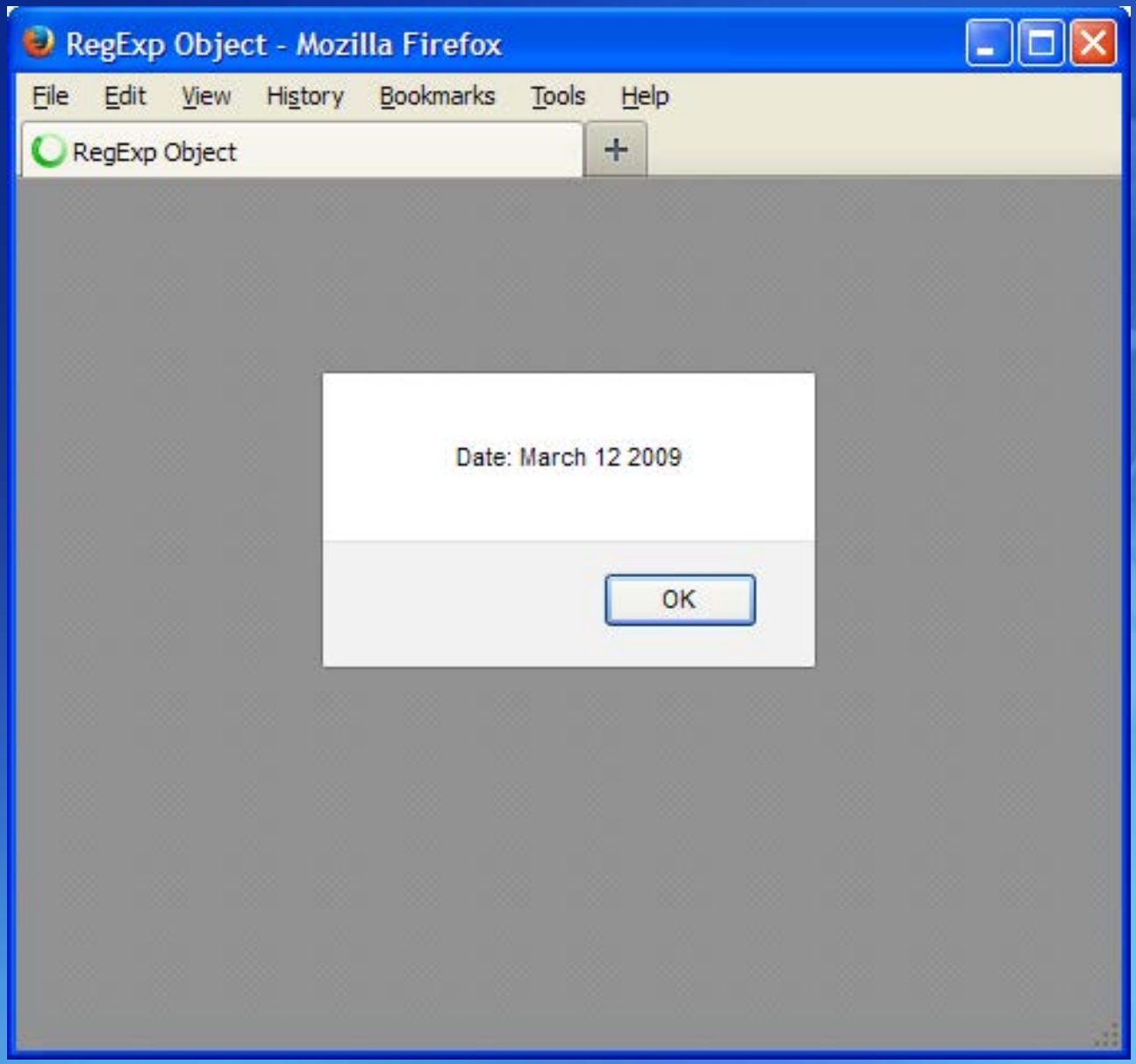
Lab10 : RegExp Object III

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - regexp2.html

Lab10 : regexp2.html

```
2   <!DOCTYPE html>
3   <html>
4       <head>
5           <title> RegExp Object </title>
6           <meta charset="utf-8">
7       </head>
8       <body>
9           <script type="text/javascript">
10              var regExp = /:\D*\s\d+\s\d+/;
11              var str = "This is a date: March 12 2009";
12              var resultString = str.match(regExp);
13              alert("Date" + resultString);
14          </script>
15      </body>
16  </html>
```

Lab10 : Result



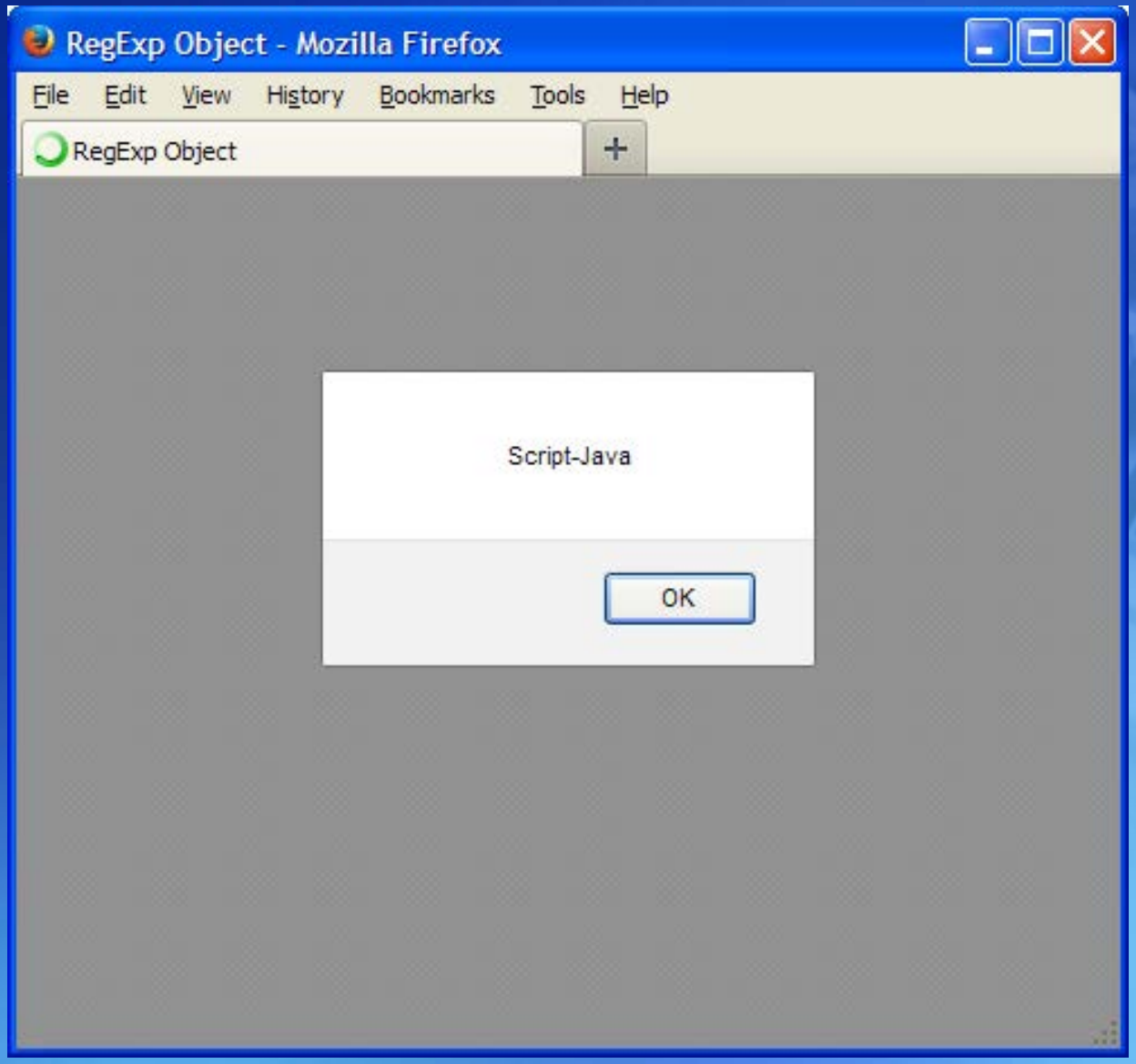
Lab11 : RegExp Object IV

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - regexp3.html

Lab11 : regexp3.html

```
2   <!DOCTYPE html>
3   <html>
4       <head>
5           <title> RegExp Object </title>
6           <meta charset="utf-8">
7       </head>
8       <body>
9           <script>
10              var rgExp = /(\w*)-*(\w*)/
11              var str = "Java---Script";
12              var resultStrng = str.replace(rgExp,"$2-$1");
13              alert(resultStrng);
14          </script>
15      </body>
16  </html>
```

Lab11 : Result



The `Date` Object

- Is used to work with dates and times.
- Handles dates similarly to Java.
- Range is -100,000,000 days to 100,100,100 days relative to 01 January, 1970 UTC.
- Supports a number of UTC(Universal) methods, as well as local time methods.
- UTC, also known as GMT(Greenwich Mean Time), refers to the time as set by the World Time Standard.

The `Date` Object (Cont.)

- Syntax
  - `new Date()`
  - `new Date( value )`
  - `new Date( dateString )`
  - `new Date(year, month, day [, hour, minute, second, millisecond] )`

The Date Object (Cont.)

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

  var today = new Date();     //creates today's date and time.
  document.write(today + "<br />");  //print Mon Oct 15 23:24:23  UTC+0900 2012

  var Xmas95 = new Date("December 25, 1995 13:30:00");
  document.write(Xmas95 + "<br />");  //print Mon Dec 25 13:30:00 UTC+0900 1995

  var Xmas2000 = new Date(2000, 11, 25);
  document.write(Xmas2000 + "<br />"); //print Mon Dec 25 00:00:00 UTC+0900 2000

  var Xmas2012 = new Date(2012, 11, 25, 9, 30, 0);
  document.write(Xmas2012);           //print Tue Dec 25 09:30:00 UTC+0900 2012

</script>
```

The `Date` Object - Methods

- **`parse()`**
  - Parses a date string and returns the number of milliseconds between the date string and midnight of January 1, 1970.
  - Syntax

    `Date.parse( dateString )`

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

  var su = Date.parse("October 15, 2012");
  document.write(su);   //1350226800000

</script>
```

The Date Object - Methods

- **UTC()**
  - Returns the number of milliseconds in a date string since midnight of January 1, 1970, according to universal time.
  - Syntax

```
Date.UTC( year, month, day, hours,
  minutes, seconds, millisec )
```

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

 var date = Date.UTC(2012, 9, 15); //2012-10-15
 document.write(date + "<br />");  //print 1350259200000
 document.write(Date.parse("October, 15, 2012"));
 //print 1350226800000

</script>
```

The `Date` Object – date instance 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)
`getMilliseconds()`	Returns the milliseconds (from 0~999)
`getMinutes()`	Returns the minutes (from 0~59)
`getMonth()`	Returns the month (from 0~11)
`getSeconds()`	Returns the seconds (from 0~59)
`getTime()`	Returns the number of milliseconds since midnight Jan 1, 1970
`getTimezoneOffset()`	Returns the time difference between UTC time and local time, in minutes.

The `Date` Object – date instance Methods (Cont.)

Method	Description
`getUTCDate()`	Returns the day of the month, according to universal time (from 1~31)
`getUTCDay()`	Returns the day of the week, according to universal time (from 0~6)
`getUTCFullYear()`	Returns the year, according to universal time (four digits)
`getUTCHours()`	Returns the hour, according to universal time (from 0~23)
`getUTCMilliseconds()`	Returns the milliseconds, according to universal time (from 0~999)
`getUTCMinutes()`	Returns the minutes, according to universal time (from 0~59)
`getUTCMonth()`	Returns the month, according to universal time (from 0~11)
`getUTCSeconds()`	Returns the seconds, according to universal time (from 0~59)

The `Date` Object – date instance Methods (Cont.)

Method	Description
`setDate()`	Sets the day of the month of a date object
`setFullYear()`	Sets the year (four digits) of a date object
`setHours()`	Sets the hour of a date object
`setMilliseconds()`	Sets the milliseconds of a date object
`setMinutes()`	Sets the minutes of a date object
`setMonth()`	Sets the month of a date object
`setSeconds()`	Sets the seconds of a date object
`setTime()`	Sets a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970

The `Date` Object – date instance Methods (Cont.)

Method	Description
`setUTCDate()`	Sets the day of the month of a date object, according to universal time
`setUTCFullYear()`	Sets the year of a date object, according to universal time
`setUTCHours()`	Sets the hour of a date object, according to universal time
`setUTCMilliseconds()`	Sets the milliseconds of a date object, according to universal time
`setUTCMinutes()`	Sets the minutes of a date object, according to universal time
`setUTCMonth()`	Sets the month of a date object, according to universal time
`setUTCSeconds()`	Sets the seconds of a date object, according to universal time

The `Date` Object – date instance Methods (Cont.)

- `getYear()`
  - Deprecated.
  - Use the `getFullYear()` method instead.
- `setYear()`
  - Deprecated.
  - Use the `setFullYear()` method instead.
- `toGMTString()`
  - Deprecated.
  - Use the `toUTCString()` method instead.

The `Date` Object – date instance Methods (Cont.)

- **`toDateString()`**
  - Converts the date portion of a **`Date`** object into a readable string.
  - Syntax

    `dateObj.toDateString()`

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

 var date = new Date();
 document.write(date.toDateString());
 //print Tue Oct 16 2012

</script>
```

The `Date` Object – date instance Methods (Cont.)

- **`toISOString()`**
  - Returns the date as a string, using the ISO standard.
  - Syntax

    `dateObj.toISOString()`

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

  var date = new Date();
  document.write(date.toISOString());
  //print 2012-10-15T15:39:28.741Z

</script>
```

The `Date` Object – date instance Methods (Cont.)

- **`toJSON()`**
  - Returns the date as a string, formated as a JSON date.
  - Syntax

    `dateObj.toJSON()`

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

  var date = new Date();
  document.write(date.toJSON());
  //print 2012-10-15T15:42:23.656Z

</script>
```

The `Date` Object – date instance Methods (Cont.)

- **`toLocaleDateString()`**
  - Returns the date portion of a **`Date`** object as a string, using locale conventions.
  - Syntax

    `dateObj.toLocaleDateString()`

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

  var date = new Date();
  document.write(date.toLocaleDateString());
  //print Tuesday, October 16, 2012

</script>
```

The `Date` Object – date instance Methods (Cont.)

- **`toLocaleTimeString()`**
  - Returns the time portion of a **`Date`** object as a string, using locale conventions.
  - Syntax

    `dateObj.toLocaleTimeString()`

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

  var date = new Date();
  document.write(date.toLocaleTimeString());
  //print 12:46:50 AM

</script>
```

The `Date` Object – date instance Methods (Cont.)

- **`toLocaleString()`**
  - Converts a `Date` object to a string, using locale conventions.
  - Syntax

    `dateObj.toLocaleString()`

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

  var date = new Date();
  document.write(date.toLocaleString());
  //print Tuesday, October 16, 2012 12:48:48 AM

</script>
```

The `Date` Object – date instance Methods (Cont.)

- **`toString()`**
  - Converts a **`Date`** object to a string.
  - Syntax
    `dateObj.toString()`

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

 var date = new Date();
 document.write(date.toString());
 //print Tue Oct 16 00:50:30 UTC+0900 2012

</script>
```

The `Date` Object – date instance Methods (Cont.)

- **`toTimeString()`**
  - Converts the time portion of a **`Date`** object to a string.
  - Syntax

    `dateObj.toTimeString()`

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

  var date = new Date();
  document.write(date.toTimeString());
  //print 00:52:18 UTC+0900

</script>
```

The `Date` Object – date instance Methods (Cont.)

- **`toUTCString()`**
  - Converts a **`Date`** object to a string, according to universal time.
  - Syntax

    `dateObj.toUTCString()`

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

 var date = new Date();
 document.write(date.toUTCString());
 //print Mon, 15 Oct 2012 15:54:23 UTC

</script>
```

The `Date` Object – date instance Methods (Cont.)

- **`valueOf()`**
  - Returns the primitive value of a **`Date`** object.
  - Syntax
    `dateObj.valueOf()`

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

 var date = new Date();
 document.write(date.valueOf());
 //print 1350316590500

</script>
```

The `Date` Object – Sample Codes

- Set a `Date` object to be 5 days into the future.

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

  var today = new Date();              //Tue Oct 16 UTC+0900 2012
  today.setDate(today.getDate() + 5);
  document.write("After 5 days is " + today);
  //print After 5 days is Sun Oct 21 00:59:57 UTC+0900 2012

</script>
```

The `Date` Object – Sample Codes (Cont.)

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

 var dtNow = new Date();
 // set day, month, year
 dtNow.setDate(18);
 dtNow.setMonth(10);
 dtNow.setYear(1954);
 dtNow.setHours(7);
 dtNow.setMinutes(2);
 // output formatted
 document.writeln(dtNow.toString() + "<br />"); //print Thu Nov 18 07:02:24 UTC+0900 1954
 document.writeln(dtNow.toLocaleString() + "<br />"); //print Thursday, November 18, 1954 7:02:24 AM
 document.writeln(dtNow.toLocaleDateString() + "<br />"); //print Thursday, November 18, 1954
 document.writeln(dtNow.toLocaleTimeString() + "<br />"); //print 7:02:24 AM
 document.writeln(dtNow.toGMTString() + "<br />"); //print Wed, 17 Nov 1954 22:02:24 UTC
 document.writeln(dtNow.toUTCString());  //print Wed, 17 Nov 1954 22:02:24 UTC

</script>
```

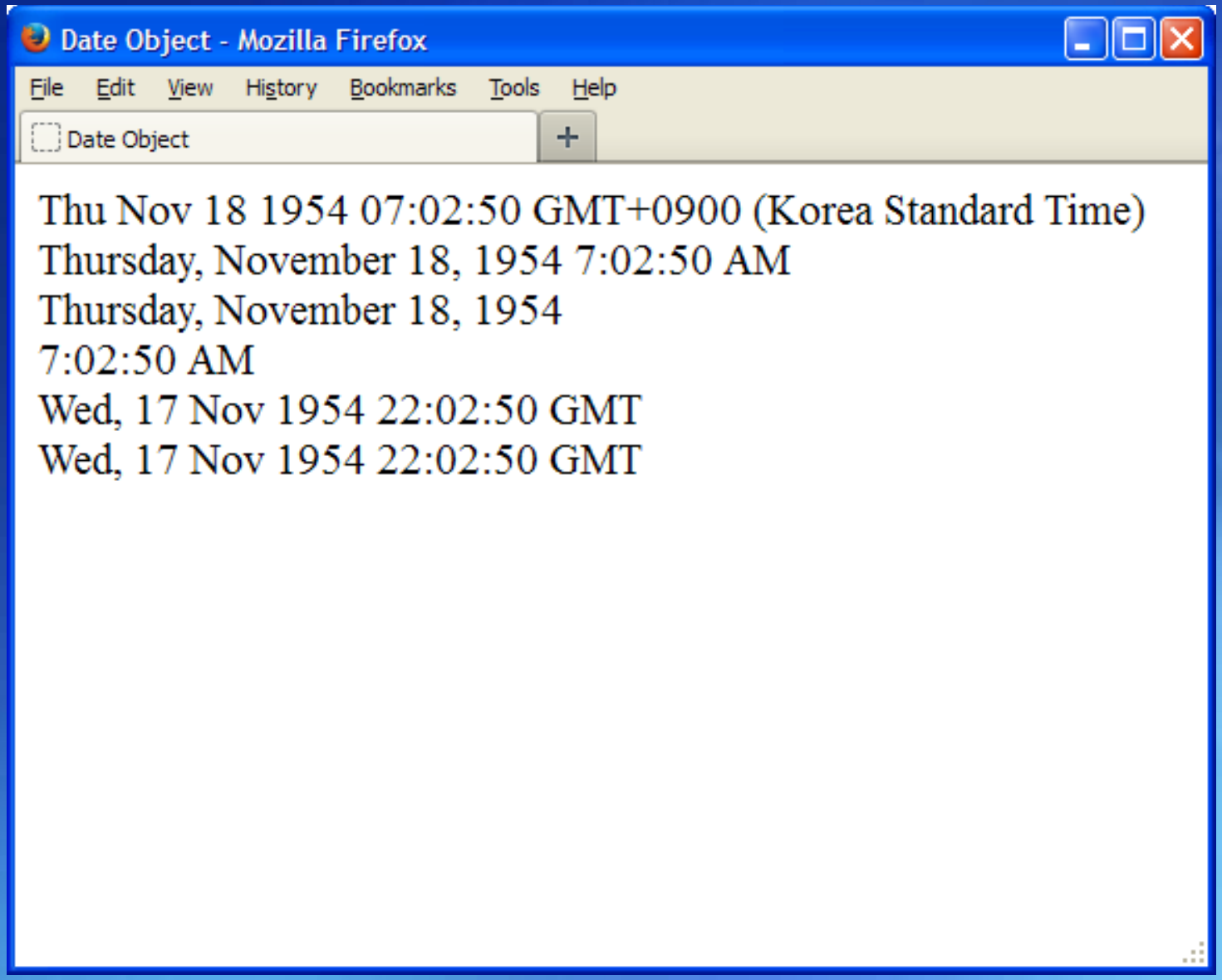
Lab12 : Date Object

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - date.html

Lab12 : date.html

```
9   <script>
10      // new date
11      var dtNow = new Date();
12
13      // set day, month, year
14      dtNow.setDate(18);
15      dtNow.setMonth(10);
16      dtNow.setYear(1954);
17      dtNow.setHours(7);
18      dtNow.setMinutes(2);
19
20      // output formatted
21      document.writeln(dtNow.toString() + "<br />");
22      document.writeln(dtNow.toLocaleString() + "<br />");
23      document.writeln(dtNow.toLocaleDateString() + "<br />");
24      document.writeln(dtNow.toLocaleTimeString() + "<br />");
25      document.writeln(dtNow.toGMTString() + "<br />");
26      document.writeln(dtNow.toUTCString());
27  </script>
```

Lab12 : Result



The Math Object

- Allows you to perform mathematical tasks.
- All properties and methods can be called by using Math as an object *without* creating it.

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

The Math Object (Cont.)

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

 document.write(Math.PI + "<br />");  //print 3.141592653589793
 document.write(Math.ceil(1.4) + "<br />");   //print 2
 document.write(Math.floor(1.4) + "<br />");  //print 1
 document.write(Math.round(1.4) + "<br />"); //print 1
 document.write(Math.random() + "<br />");   //print 0.2941168451669903
 //random number between 0 and 10.
 document.write(Math.floor(Math.random() * 11) + "<br />");
 document.write(Math.pow(4,3));                 //print 64

</script>
```

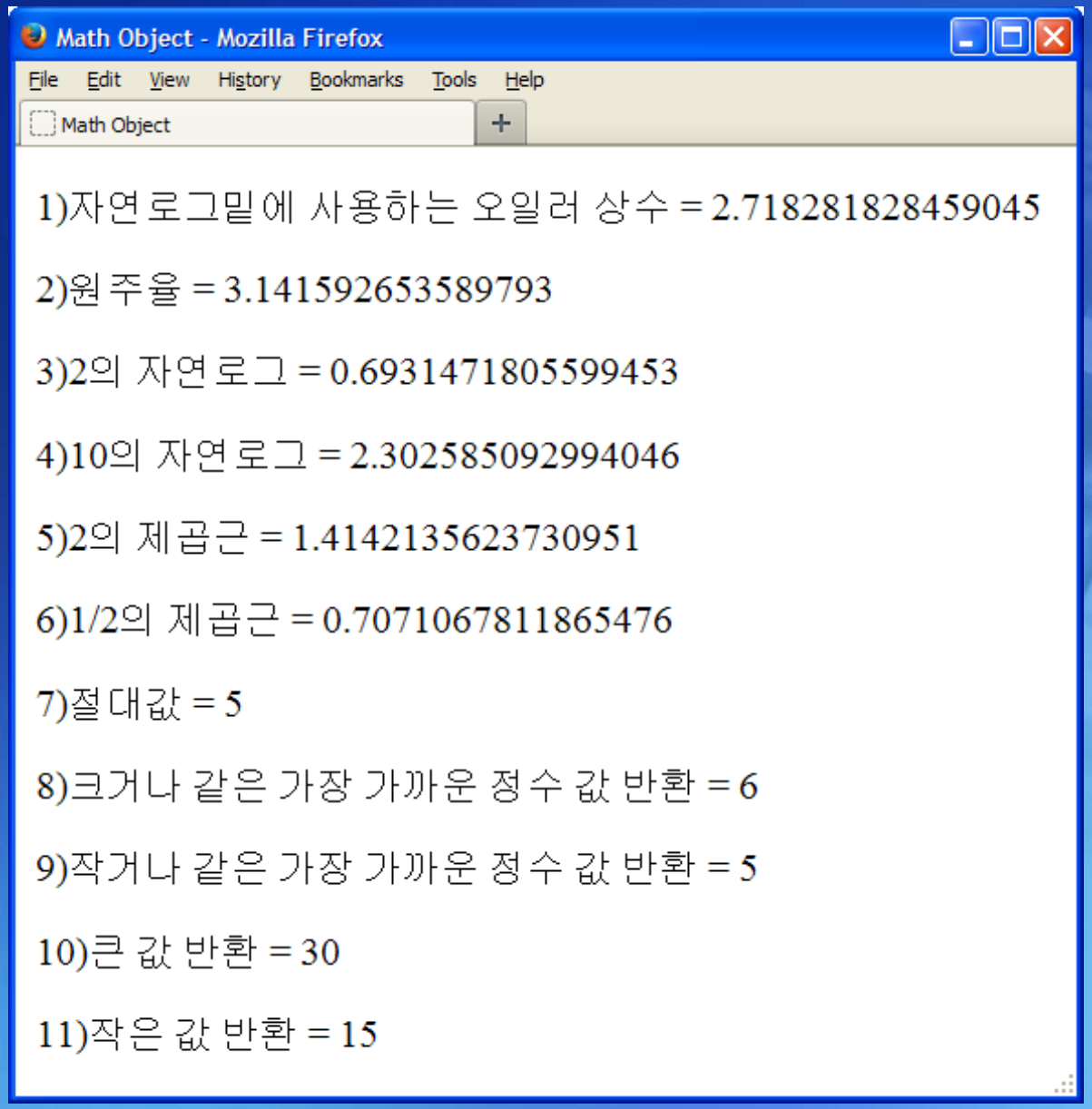
Lab13 : Math Object

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - math.html

Lab13 : math.html

```
7   <body>
8     <script>
9       document.write("<p>1)자연로그밑에 사용하는 오일러 상수 = " + Math.E + "</p>");
10      document.write("<p>2)원주율 = " + Math.PI + "</p>");
11      document.write("<p>3)2의 자연로그 = " + Math.LN2 + "</p>");
12      document.write("<p>4)10의 자연로그 = " + Math.LN10 + "</p>");
13      document.write("<p>5)2의 제곱근 = " + Math.SQRT2 + "</p>");
14      document.write("<p>6)1/2의 제곱근 = " + Math.SQRT1_2 + "</p>");
15      document.write("<p>7)절대값 = " + Math.abs(-5) + "</p>");
16      document.write("<p>8)크거나 같은 가장 가까운 정수 값 반환 = " + Math.ceil(5.3) +
        "</p>");
17      document.write("<p>9)작거나 같은 가장 가까운 정수 값 반환 = " + Math.floor(5.3) +
        "</p>");
18      document.write("<p>10)큰 값 반환 = " + Math.max(15,30) + "</p>");
19      document.write("<p>11)작은 값 반환 = " + Math.min(15,30) + "</p>");
20    </script>
21  </body>
```

Lab13 : Result



The Array Object

- JavaScript does not have an explicit array data type.
- Is used to store multiple values in a single variable.
- Neither the size of a JavaScript array nor the types of its elements are fixed.
- Since an array's size can grow or shrink at any time, JavaScript arrays are not guaranteed to be dense.

The Array Object (Cont.)

- Syntax

```
[element0, element1, … , elementN]
new Array(element0, element1,…, elementN]
new Array(arrayLength)
```

  - arrayLength

    Is an integer between 0 and 2^{32} -1 (inclusive).

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

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

    var myCars1 = new Array("Saab", "Volvo", "BMW");

    var myCars2 = ["Saab", "Volvo", "BMW"];

</script>
```

The Array Object (Cont.)

- All JavaScript variables are objects.
- Then array elements are objects.
- Functions are objects, too.
- JavaScript can have different Objects in one array.

```
myArray[0] = Date.now;
myArray[1] = myFunction();
myArray[2] = myCars;
```

The Array Object - Properties

- **length**
  - Sets or returns the number of elements in an array.
  - Syntax

    `array.length` or `array.length = number`

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

    var cats = ["Dusty", "Misty", "Twiggy"];
    document.write(cats.length + "<br />");  //print 3

    cats.length = 2;
    document.write(cats + "<br />"); //print Dusty, Misty

    cats.length = 0;
    document.write(cats + "<br />"); //print nothing

    cats.length = 3;
    document.write(cats);              //print , ,

</script>
```

The Array Object – Properties (Cont.)

```
<script type="text/javascript">
  function myFunction(){
     var fruits = ['Banana', 'Orange', 'Apple', 'Mango'];
     var obj = document.getElementById("demo");
     obj.innerHTML = fruits.length;
  }
</script>

<body>

  <p id="demo">Click the button to create an array, then display it's length</p>

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

The `Array` Object - Methods

- **`concat()`**
  - Joins two or more arrays, and returns a copy of the joined arrays.
  - Syntax

    `array.concat(array1, array2, …, arrayX)`

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

    var os = ['Windows', 'Linux', 'Unix', 'Mac'];
    var language = ['C-language', 'C++', 'Java', 'C#', 'Delphi'];
    var scripts = ['HTML', 'CSS', 'JavaScript', 'jQuery'];
    var web = ['CGI', 'ASP', 'PHP', 'Servlet/JSP', 'Ruby', 'Python'];
    var obj = os.concat(language, scripts, web);
    document.write(obj);

</script>
```

The `Array` Object – Methods (Cont.)

- **`join()`**
  - Joins all elements of an array into a string.
  - Syntax

    `array.join(separator)`

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

    var fruits = ['Banana', 'Apple', 'Pine Apple', 'Mango', 'Orange'];
    document.write(fruits.join() + "<br />");
    //print Banana, Apple, Pine Apple, Mango, Orange

    document.write(fruits.join(" and "));
    //Banana and Apple and Pine Apple and Mango and Orange

</script>
```

The `Array` Object – Methods (Cont.)

- **`indexOf()`**
  - Search the array for an element and returns it's position.
  - Syntax

    `array.indexOf(item, start)`

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

    var fruits = ['Banana', 'Apple', 'Pine Apple', 'Banana', 'Apple'];
    document.write(fruits.indexOf('Apple') + "<br />");  //print 1

    document.write(fruits.indexOf('Apple', 2));  //print 4

</script>
```

The `Array` Object – Methods (Cont.)

- **`lastIndexOf()`**
  - Search the array for an element, starting at the end, and returns it's position.
  - Syntax

    `array.lastIndexOf(item, start)`

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

  var fruits = ['Banana', 'Apple', 'Pine Apple', 'Banana', 'Apple'];
  document.write(fruits.lastIndexOf('Apple'));  //print 4

</script>
```

The `Array` Object – Methods (Cont.)

- **`push()`**
  - Adds new elements to the end of an array, and returns the new length.
  - Syntax

    `array.lastIndexOf(item1, item2,…, itemN)`

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

    var fruits = ['Banana', 'Apple', 'Pine Apple'];
    fruits.push('Kiwi', 'Lemon', 'Mango');
    document.write(fruits);
    //print Banana, Apple, Pine Apple, Kiwi, Lemon, Mango

</script>
```

The `Array` Object – Methods (Cont.)

- **`pop()`**
  - Removes the last element of an array, and returns that element.
  - Syntax

    `array.pop()`

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

    var fruits = ['Banana', 'Apple', 'Pine Apple'];
    fruits.pop();
    document.write(fruits);          //print Banana, Apple

</script>
```

The `Array` Object – Methods (Cont.)

- **`unshift()`**
  - Adds new elements to the beginning of an array, and returns the new length.
  - Syntax

    `array.unshift(item1, item2,…, itemN)`

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

    var fruits = ['Banana', 'Orange', 'Apple', 'Mango'];
    fruits.unshift('Lemon', 'Pineapple');
    document.write(fruits);
    //print Lemon, Pineapple, Banana, Orange, Apple, Mango

</script>
```

The `Array` Object – Methods (Cont.)

- **`shift()`**
  - Removes the first element of an array, and returns that element.
  - Syntax

    `array.shift()`

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

    var fruits = ['Banana', 'Orange', 'Apple', 'Mango'];
    fruits.shift();
    document.write(fruits);     //print Orange, Apple, Mango

</script>
```

The `Array` Object – Methods (Cont.)

- **`reverse()`**
  - Reverses the order of the elements in a array.
  - Syntax

    `array.reverse()`

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

    var fruits = ['Banana', 'Orange', 'Apple', 'Mango'];
    fruits.reverse();
    document.write(fruits);     //print Mango, Apple, Orange, Banana

</script>
```

The `Array` Object – Methods (Cont.)

- **`slice()`**
  - Selects a part of an array, and returns the new array.
  - Syntax

    `array.slice( start, end)`

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

    var fruits = ['Banana', 'Orange', 'Apple', 'Mango'];
    var newFruits = fruits.slice(1,3);
    document.write(newFruits);    //print Orange, Apple

</script>
```

The Array Object – Methods (Cont.)

- **splice()**
  - Adds / Removes elements from an array.
  - Syntax

    array.splice(index, howmany, item1,…itemX)

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

    var fruits = ['Banana', 'Orange', 'Apple', 'Mango'];
    fruits.splice(2, 0, "Lemon", "Kiwi");
    document.write(fruits);
    //print Banana, Orange, Lemon, Kiwi, Apple, Mango

</script>
```

The `Array` Object – Methods (Cont.)

- **`sort()`**
  - Sorts the elements of an array.
  - Syntax

    `array.sort(sortfunction)`

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

   var fruits = ['Banana', 'Orange', 'Apple', 'Mango'];
   fruits.sort();
   document.write(fruits + "<br />");  //print Apple, Banana, Mango, Orange

   var array = [40, 100, 1,5,25,10];
   array.sort(function(a,b) { return a-b});
   document.write(array);                  //print 1,5,10,25,40,100

</script>
```

The `Array` Object – Methods (Cont.)

- **`toString()`**
  - Converts an array to a string, and returns the result.
  - Syntax

    `array.toString()`

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

    var fruits = ['Banana', 'Orange', 'Apple', 'Mango'];
    document.write(fruits.toString());
    //Banana, Orange, Apple, Mango

</script>
```

The `Array` Object – Methods (Cont.)

- **`valueOf()`**
  - Returns the primitive value of an array.
  - Syntax

    `array.valueOf()`

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

   var fruits = ['Banana', 'Orange', 'Apple', 'Mango'];
   var v = fruits.valueOf();
   document.write(v + "<br />");    //print Banana, Orange, Apple, Mango
   document.write(typeof fruits + "<br />");  //print object
   document.write(typeof v);     //print object

</script>
```

The Array Object – MultiDimensional Arrays

- Arrays can be nested.
- Can contain another array as an element.

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

   var a = new Array(4);
   for(var i = 0 ; i < 4 ; i++){
      a[i] = new Array(4);
      for(var j = 0 ; j < 4 ; j++){
         a[i][j] = "[" + i + ", " + j + "]";
         document.write("a[" + i + "][" + j + "] = " + a[i][j]);
         document.write("   ");
      }
      document.write("<br />");
      /*
      a[0][0] = [0, 0]  a[0][1] = [0, 1]  a[0][2] = [0, 2]  a[0][3] = [0, 3]
      a[1][0] = [1, 0]  a[1][1] = [1, 1]  a[1][2] = [1, 2]  a[1][3] = [1, 3]
      a[2][0] = [2, 0]  a[2][1] = [2, 1]  a[2][2] = [2, 2]  a[2][3] = [2, 3]
      a[3][0] = [3, 0]  a[3][1] = [3, 1]  a[3][2] = [3, 2]  a[3][3] = [3, 3]
      */
   }

</script>
```

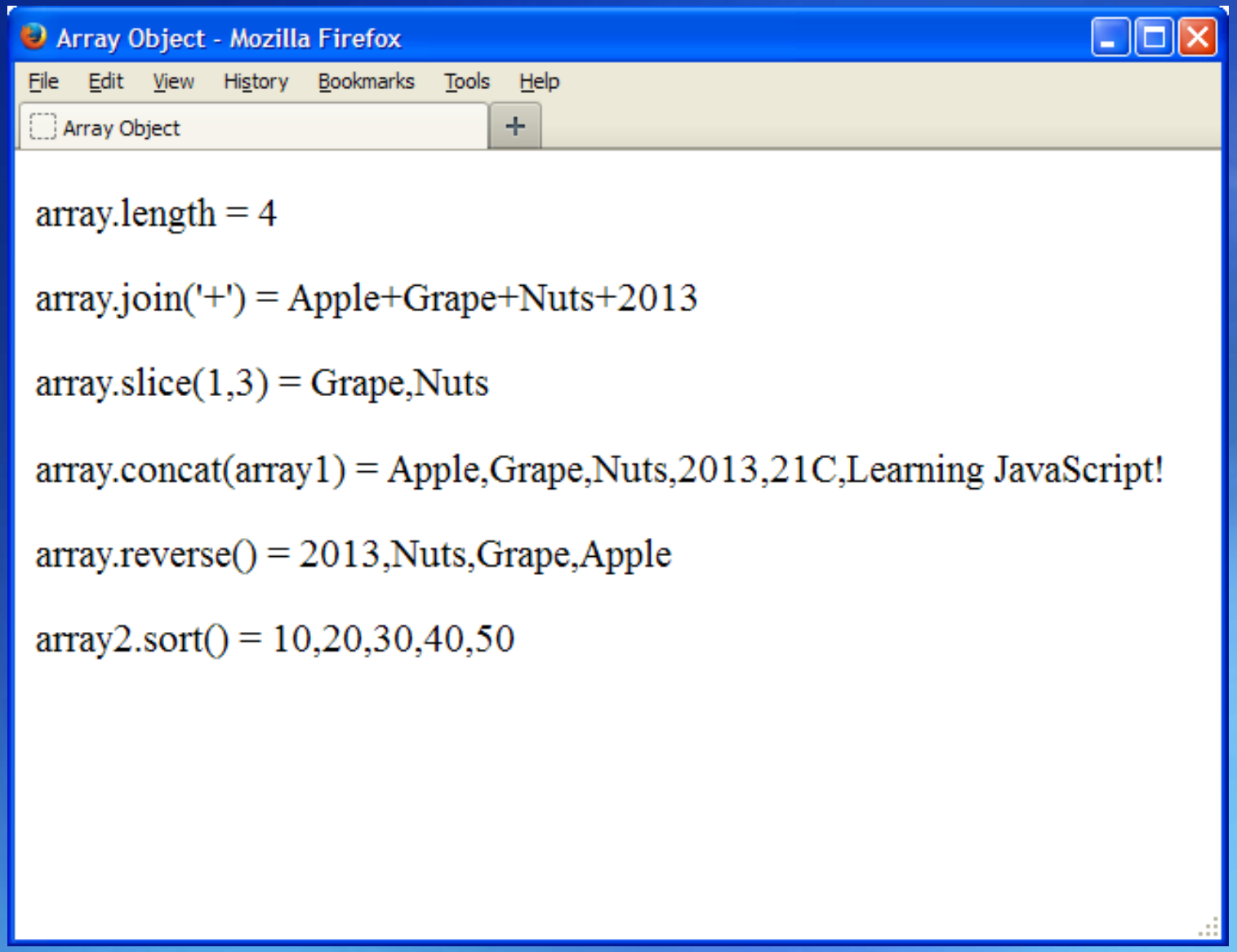
Lab14 : Array Object I

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - array.html

Lab14 : array.html

```
8   <script>
9       array = new Array("Apple", "Grape", "Nuts", 2013);
10      su = array.length;
11      join = array.join('+');
12      slice = array.slice(1,3);
13      document.write("<p>array.length = " + su + "</p>");
14      document.write("<p>array.join('+') = " + join + "</p>");
15      document.write("<p>array.slice(1,3) = " + slice + "</p>");
16
17      array1 = new Array("21C", "Learning JavaScript!");
18      concat = array.concat(array1);
19      reverse = array.reverse();
20      document.write("<p>array.concat(array1) = " + concat + "</p>");
21      document.write("<p>array.reverse() = " + reverse + "</p>");
22
23      array2 = new Array(30,10,50,40,20);
24      sort = array2.sort();
25      document.write("<p>array2.sort() = " + sort + "</p>");
26  </script>
```

Lab14 : Result



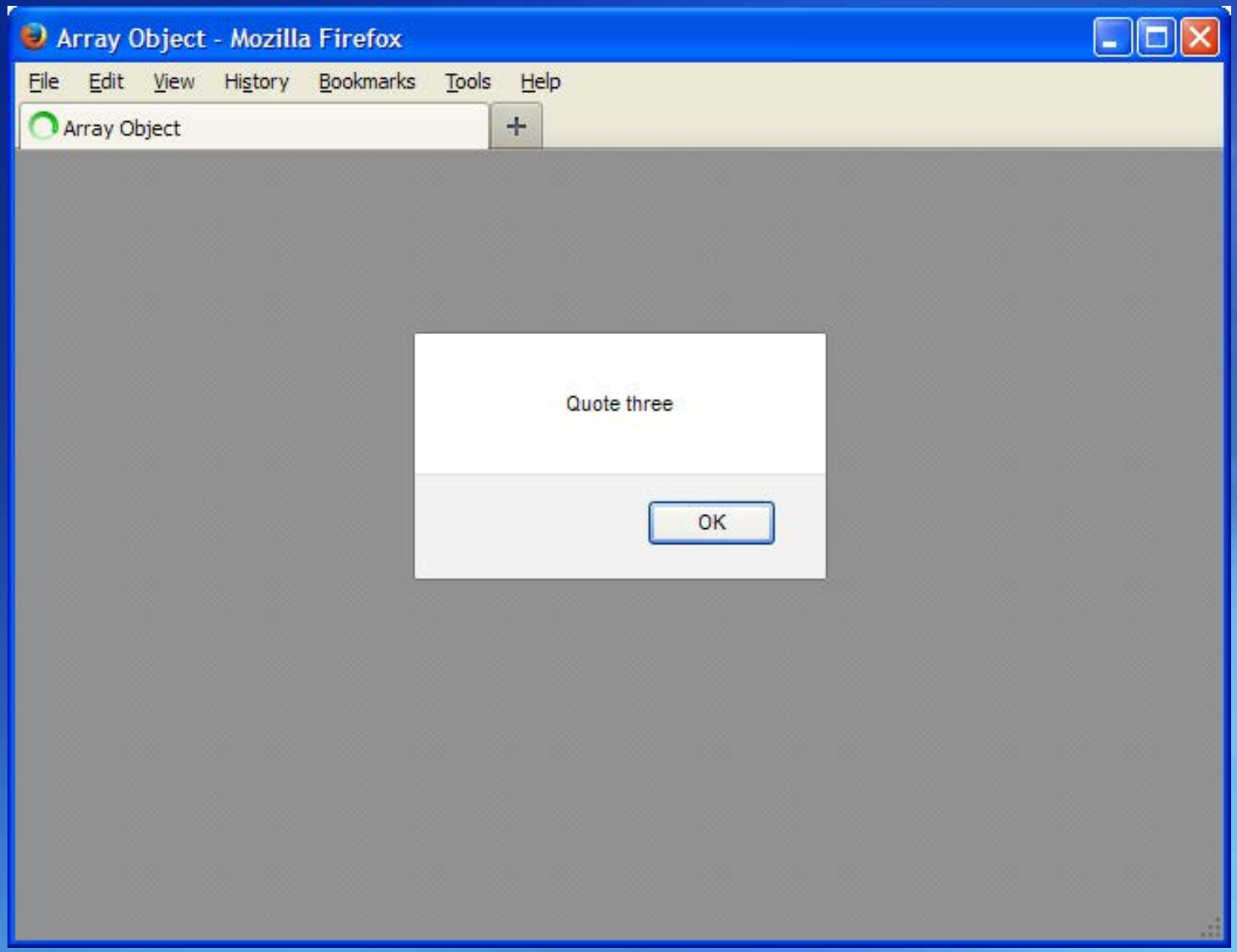
Lab15 : Array Object II

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - array1.html

Lab15 : array1.html

```
9   <script type="text/javascript">
10
11      var quoteArray = new Array(5);
12      quoteArray[0] = "Quote one";
13      quoteArray[1] = "Quote two";
14      quoteArray[2] = "Quote three";
15      quoteArray[3] = "Quote four";
16      quoteArray[4] = "Quote five";
17
18      iValue = Math.random(); // random number between 0 and 1
19      iValue *= 5; // multiply by 5 to move the decimal
20      iValue = Math.floor(iValue); // round to nearest integer
21      alert(quoteArray[iValue]);
22
23  </script>
```

Lab15 : Result



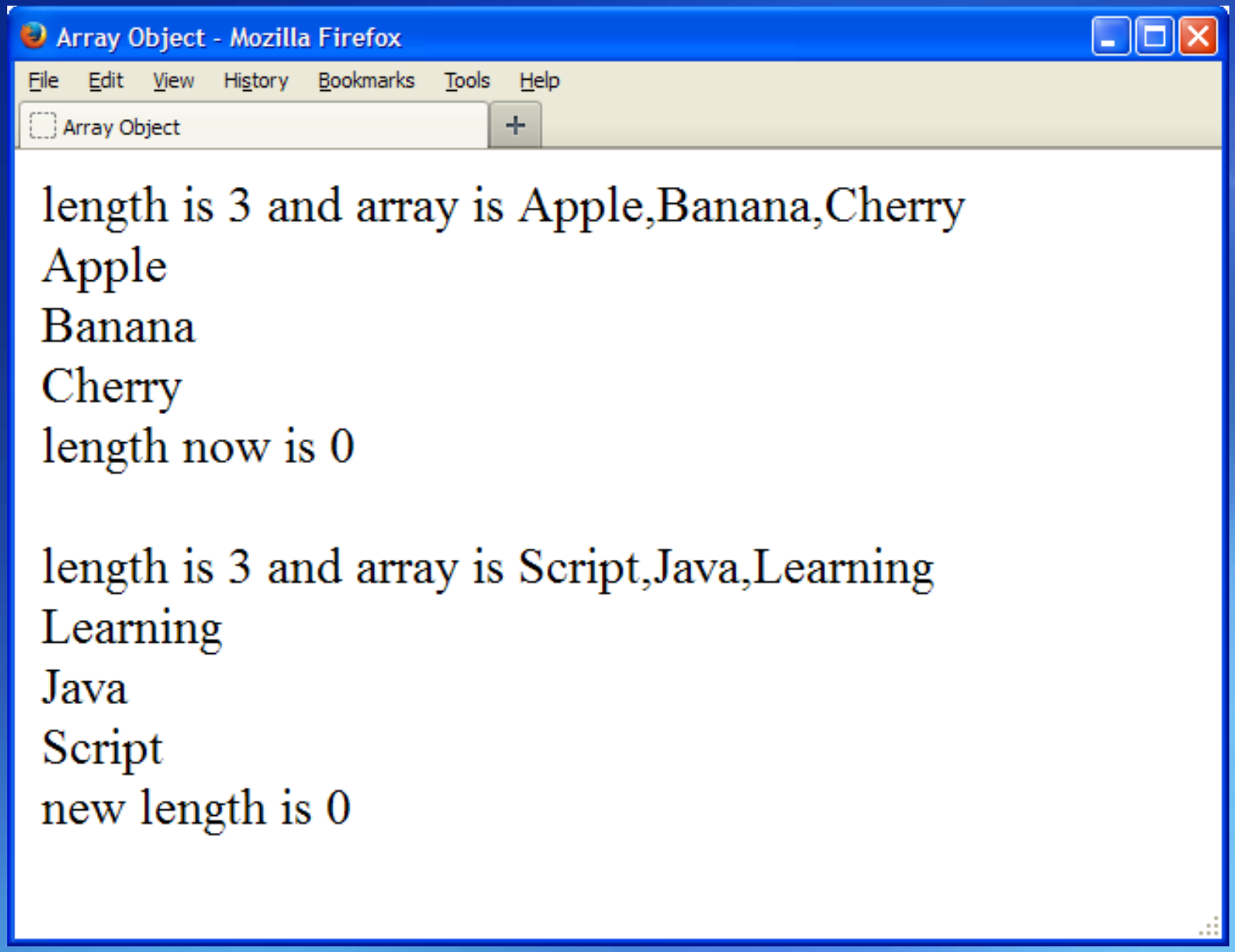
Lab16 : Array Object III

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - array2.html

Lab16 : array2.html

```
9     <script>
10        // create FIFO queue and add items using push
11        var fifoArray = new Array();
12        fifoArray.push("Apple");
13        fifoArray.push("Banana");
14        var ln = fifoArray.push("Cherry");
15        // print out length and array
16        document.writeln("length is " + ln + " and array is " + fifoArray + "<br />");
17        // use shift to shift the items off the array
18        for (var i = 0; i < ln; i++) {
19           document.writeln(fifoArray.shift() + "<br />");
20        }
21        // print out length
22        document.writeln("length now is " + fifoArray.length + "<br /><br />");
23        // now, same with unshift
24        var fifoNewArray = new Array();
25        fifoNewArray.unshift("Learning");
26        fifoNewArray.unshift("Java");
27        ln = fifoNewArray.unshift("Script");
28        document.writeln("length is " + ln + " and array is " + fifoNewArray + "<br/>");
29        // unshift
30        for (var i = 0; i < ln; i++) {
31           document.writeln(fifoNewArray.pop() + "<br />");
32        }
33        document.writeln("new length is " + fifoNewArray.length );
34    </script>
```

Lab16 : Result



Test Your Knowledge : Quiz

1. Comma-separated strings are a common data format. How would you create an array of elements when given a comma-separated string? Write the code for the following string, and then access the third element: `"cats,dogs,birds,horses"`.

Test Your Knowledge : Quiz

1. Comma-separated strings are a common data format. How would you create an array of elements when given a comma-separated string? Write the code for the following string, and then access the third element: "cats,dogs,birds,horses".

```
var animalString = "cats,dogs,birds,horses";
var animalArray = String.split(animalString,",");
alert(animalArray[2]);
```

Test Your Knowledge : Quiz (Cont.)

2. The `\b` special character can define a word boundary, and `\B` matches on a nonword boundary. Define a regular expression that will find all occurrences of the word *fun* in the following string and replace them with power:

```
"The fun of functions is that
they are functional."
```

Test Your Knowledge : Quiz (Cont.)

2. The `\b` special character can define a word boundary, and `\B` matches on a nonword boundary. Define a regular expression that will find all occurrences of the word *fun* in the following string and replace them with power:

   "The fun of functions is that they are functional."

```
var funPattern = /\bfun\b/;
var strToSearch = "The fun of functions …”;
var afterMatch =
        strToSearch.replace(funPattern,"power")
```

Test Your Knowledge : Quiz (Cont.)

3. Create code to get today's date, modify it by a week, and print out the new date.

Test Your Knowledge : Quiz (Cont.)

3. Create code to get today's date, modify it by a week, and print out the new date.

```
var dtNow = new Date();
var hours = dtNow.getHours();
hours+=168;
dtNow.setHours(hours);
document.writeln(dtNow.toString());
```

Test Your Knowledge : Quiz (Cont.)

4. Given a number of 34.44, write code that would round the number down. Do the same to round the number up.

Test Your Knowledge : Quiz (Cont.)

4. Given a number of 34.44, write code that would round the number down. Do the same to round the number up.

```
var baseNum = 34.44;
var numFloor = Math.floor(baseNum);
// returns 34
var numCeil = Math.ceil(baseNum);
// returns 35
```

Test Your Knowledge : Quiz (Cont.)

5. Given a string such as the following, use pattern matching and replace all existing punctuation with commas, and then load it as an array and print out each value:

```
var str =
   "apple.orange-strawberry,lemon-.lime";
```

Test Your Knowledge : Quiz (Cont.)

5. Given a string such as the following, use pattern matching and replace all existing punctuation with commas, and then load it as an array and print out each value:

```
var str =
   "apple.orange-strawberry,lemon-.lime";
```

```
var strToAlter = "apple.orange-strawberry,lemon-.lime";
var puncPattern = /[\.|-]/g;
var afterMatch = strToAlter.replace(puncPattern,",");
var fruits = afterMatch.split(',');
for (var i = 0; i < fruits.length; i++)
    document.writeln(fruits[i]);
```