JavaScript Data Types and Variables

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

JavaScript Syntax

- JavaScript programs are written using the Unicode character set.
- JavaScript is case-sensitive.
- JavaScript ignores *white space*.
- JavaScript have a comments.
- Semicolons are used to delineate expressions in JavaScript .

Case-Sensitive

- Means that language keywords, variables, function names, and other *identifiers* must always be typed with a consistent capitalization of letters.
- Keywords are *lowercase*.
- Variables can be any mix of case.

```
buttonOne
txt1
a
c
```

White Space

- Can use spaces, indenting, or whatever coding standards you prefer to make the JavaScript more readable .

```
function cubeme(incomingNum) {
if (incomingNum == 1) {
return "What are you doing?";
} else {
return Math.pow(incomingNum,3);
}
}
var theNum = 2;
var finalNum = cubeme(theNum);
if (isNaN(finalNum)) {
alert("You should know that 1 to any power is 1.");
} else {
alert("When cubed, " + theNum + " is " + finalNum);
}
```

Comments

- A comment in JavaScript will look familiar to C programming language .
- A multiline comment begins and ends with /* and */ respectively.

```
/* This is a multiline comment in JavaScript
It is just like a C-style comment insofar as it can
span multiple lines before being closed. */
```

- A single-line comment begins with two front slashes (//)

```
// Here is a single line comment.
```

Semicolons

- Officially, putting a semicolon at the end of a statement is optional.
- *However*, leaving off the semicolon makes reading your code more difficult and, in some cases, causes JavaScript errors.

Literals

- To represent values in JavaScript.
- These are fixed values, not variables.
  - `Array` literals
  - `Boolean` literals
  - Floating-point literals
  - Integers
  - `Object` literals
  - `String` literals

Array Literals

- Is a list of zero or more expressions.
- Enclosed in square brackets (`[]`).

```
var coffees = ["French Roast",
"Colombian", "Kona"];

alert(coffees[2]); //returns “Kona”
```

- `Array` literals are also `Array` objects.

`Boolean` Literals

- Has two literal values: `true` and `false`.
- The `Boolean` object is a wrapper around the primitive `Boolean` data type.

Number Literals

- JavaScript has a single number type.
- Internally, it is represented as 64-bit floating point, the same as Java's `double`.
- Is no separate integer type, so `1` and `1.0` are the same value.
- In case an exponent part, then the value of the literal is computed by multiplying the part before the `e` by `10` raised to the power of the part after the `e`.
- `100` → `1e2`

Number Literals (Cont.)

- Negative numbers can be formed by using the – prefix operator.
- The value `NaN` is a number value that is the result of an operation that cannot produce a normal result.
- `NaN` is not equal to any value.
- Can detect `NaN` with the `isNaN(number)` function.
- The value `Infinity` represents all values greater than `1.7976931348623157e+308`.

Integer Literals

- Can be expressed in decimal (base 10), hexadecimal (base 16), and octal (base 8).
- Decimal, base 10

  `0, 117` and `-345`
- Octal, base 8

  `015, 0001` and `-077`
- Hexadecimal, base 16

  `0x1123, 0x00111` and `-0xF1A7`

Floating-point Literals

- A floating-point literal can have the following parts:
  - A decimal integer which can be signed (preceded by "+" or "-"),
  - A decimal point (" . "),
  - A fraction (another decimal number),
  - An exponent.

```
3.1415,    -3.1E12,    .1e12,
2E-12.
.33333333333333333
```

Object Literals

- Is a list of zero or more pairs of property names and associated values.
- Enclosed in curly braces ({ }).

```
<script type="text/javascript">
    var Sales = "Sonata";

    function CarTypes(name) {
       return (name == "Honda") ?  name : "Sorry, we don't sell " + name + "." ;
    }

    var car = { myCar: "Saturn", getCar: CarTypes("Honda"), special: Sales };

    document.write(car.myCar + "<br />");   // Saturn
    document.write(car.getCar + "<br />");  // Honda
    document.write(car.special); // Sonata
</script>
```

Object Literals (Cont.)

- Can use a numeric or string literal for the name of a property or nest an object inside another.

```
<script type="text/javascript">
    var car = { manyCars: {a: "Saab", "b": "Jeep"}, 7: "Mazda" };

    document.write(car.manyCars.b + "<br />"); // Jeep
    document.write(car[7] + "<br />"); // Mazda

    var foo = {a: "alpha", 2: "two"};
    document.write(foo.a + "<br />");    // alpha
    document.write(foo[2] + "<br />");   // two
    //document.write(foo.2);  // Error: missing ) after argument list
    //document.write(foo[a]); // Error: a is not defined
    document.write(foo["a"] + "<br />"); // alpha
    document.write(foo["2"]); // two
</script>
```

String Literals

- Is zero or more characters enclosed in double (") or single (') quotation marks.
- Can contain zero or more characters.
- A string must be delimited by quotation marks of the same type.

```
"foo“    'bar‘    “1234“
"one line \n another line“
"John's cat"
```

`String` Escape Sequences

Character	Meaning
`\b`	Backspace
`\f`	Form feed
`\n`	New line
`\r`	Carriage return
`\t`	Tab
`\v`	Vertical tab
`\'`	Apostrophe or single quote
`\"`	Double quote
`\\`	Backslash character(\).
`\XXX`	Three octal digits *XXX* between 0 and 377. For example, \251 is the octal sequence for the copyright symbol.
`\xXX`	The two hexadecimal digits *XX* between 00 and FF. For example, \xA9 is the hexadecimal sequence for the copyright symbol.
`\uXXXX`	The Unicode character specified by the four hexadecimal digits *XXXX*. For example, \u00A9 is the Unicode sequence for the copyright symbol.

Escaping Characters

- Can insert a quotation mark inside a string by preceding it with a backslash.

```
var quote = "He read \"The Cremation of Sam McGee\" by R.W. Service.";
```

- To include a literal backslash inside a string, you must escape the backslash character.

```
var home = "c:\\temp";
```

Unicode Escape Sequences

- Can use the Unicode escape sequence in string literals, regular expressions, and identifiers.
- The escape sequence consists of six ASCII characters: `\u` and a four-digit hexadecimal number.

```
“A” === “\u0041”
var x = "\u00A9 Netscape Communications";
```

Unicode Values for Special Characters

Category	Unicode Value	Name	Format Name
White space values	\u0009	Tab	<TAB>
	\u000B	Vertical Tab	<VT>
	\u000C	Form Feed	<FF>
	\u0020	Space	<SP>
Line terminator Values	\u000A	Line Feed	<LF>
	\u000D	Carriage Return	<CR>
Additional Unicode escape sequence Values	\u0008	Backspace	<BS>
	\u0009	Horizontal Tab	<HT>
	\u0022	Double Quote	“
	\u0027	Single Quote	‘
	\u005C	Backslash	\

Separators

- Nine ASCII characters delimit program parts.
- ( ), { }, and [ ] are used in pairs.

( ) { } [ ] ; , .

Statements

- Small, logical actions taken on a function.
- Is a single command that performs some activity when executed by the JavaScript interpreter.
- A single line of code terminated by a semicolon(`;`)
- Include expression, empty, block, conditional, iteration, transfer of control, exception handling, variable, labeled, and synchronized.
- Reserved JavaScript words used in statements are `if`, `else`, `switch`, `while`, `do`, `for`, `for/in`, `break`, `continue`, `return`, `synchronized`, `throw`, `try`, `catch`, and `finally`.

```
total = a + b + c + d + e + f;
```

Blocks

- A group of statements is called a block or statement block.
- A block of statements is enclosed in curly braces({ and }).
- In blocks, one statement is interpreted at a time in the order in which it was written or in the order of flow control.
- You can nest block statements.

```
if (sponsored) {
      setPlayerId = n;
      setLastName( );
}
```

Identifier

- Is simply a name.
- Are used to name variables and functions and to provide labels for certain loops code.
- Must start with a letter, underscore (_), or dollar sign ($).
- Subsequent characters can also be digits (0-9).
- Because JavaScript is case sensitive, letters include the characters "A" through "Z" (uppercase) and the characters "a" through "z" (lowercase).

Identifier (Cont.)

- Starting with JavaScript 1.5, you can use ISO 8859-1 or Unicode letters such as å and ü in identifiers.

```
Number_hits
temp99
_name
$str
```

JavaScript Keywords

break	else	new	typeof
case	false	null	var
catch	finally	return	void
continue	for	switch	while
debugger	function	this	with
default	if	throw	
delete	in	true	
do	instanceof	try	

ECMAScript 5 Reserved Words

`class`	`enum`	`extends`	`super`
`const`	`export`	`import`	

ECMA-262 Specification Reserved Words

abstract	export	interface	static
boolean	extends	long	super
byte	final	native	synchronized
char	float	package	throws
class	goto	private	transient
const	implements	protected	volatile
double	import	public	
enum	int	short	

Predefined global identifiers

arguments	Error	Math	RegExp
Array	eval	NaN	String
Boolean	EvalError	Number	SyntaxError
Date	Function	Object	TypeError
decodeURI	Infinity	parseFloat	undefined
decodeURIComponent	isFinite	parseInt	URIError
encodeURI	isNaN	RangeError	
encodeURIComponent	JSON	ReferenceError	

Data Types Overview

- JavaScript is a dynamically typed language.
- Do not have to specify the data type of a variable.
- Data types are converted automatically.

```
var answer = 42;
answer = “Thanks for all the fish…”;
```

Data Types Overview (Cont.)

- In expressions involving numeric and string values with the + operator, JavaScript converts numeric values to strings.

```
//return “The answer is 42”
 x = “The answer is “ + 42;

“37” – 7;  // returns 30
“37” + 7;  // returns “377”
```

Variables

- Can declare a variable in two ways:
  - With the keyword **var**. This syntax can be used to declare both *local* and *global* variables.

    ```
    var x = 42.
    ```

  - By simply assigning it a value. This always declares a *global* variable and generates a strict JavaScript warning. You shouldn't use this variant.

    ```
    x = 42.
    ```

Variables (Cont.)

- After the declaration, the variable is empty (it has no value).
- To assign a value to the variable, use the equal sign:

```
carname = “Volvo”;
```

- However, you can also assign a value to the variable when you declare it:

```
var carname = “Volvo”;
```

Variables (Cont.)

- You can declare many variables in one statement.
- Just start the statement with **var** and separate the variables by comma:

```
var lastname = “Doe”, age = 30,
job = “Carpenter”;
```

- Your declaration can also span multiple lines:

```
var lastname = “Doe”,
age = 30,
job = “Carpenter”;
```

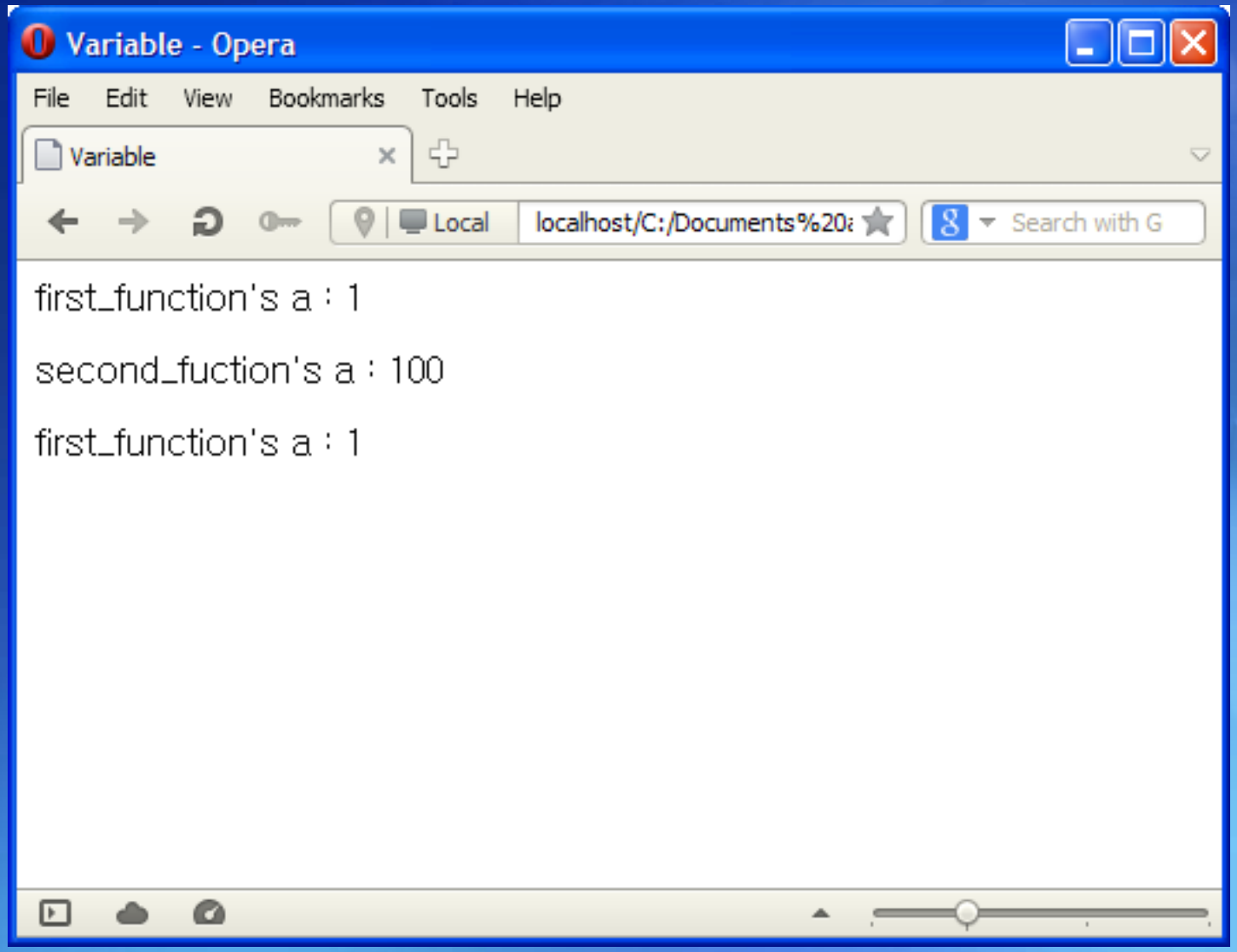
Lab1 : Variable

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

Lab1 : variable.html

```
3   <head>
4       <title> Variable </title>
5       <meta charset="utf-8">
6       <script>
7           function first_function(){
8               a = 1;
9               document.write("first_function's a : " + a + "<br><br>");
10              second_function();
11              document.write("first_function's a : "  + a + "<br><br>");
12          }
13          function second_function(){
14              var a = 100;
15              document.write("second_fuction's a : " + a + "<br><br>");
16          }
17      </script>
18  </head>
19  <body>
20      <script>
21          first_function();
22      </script>
23  </body>
```

Lab1 : Result


Variable - Opera
File Edit View Bookmarks Tools Help
Variable
Local
localhost/C:/Documents%20
Search with G
first_function's a : 1
second_fuction's a : 100
first_function's a : 1

Variable scope

- Declare a variable outside of any function, *global* variable.
- Declare a variable within a function, *local* variable.
- JavaScript does *not* have block statement scope.

```
if(true){
    var x = 5;
}
document.write(x);    //prints 5
```

Lab1 : Variable Scope

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - scope.html
  - global.js
  - global2.js

Lab2 : scope.html

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title> Variable Scope </title>
5          <meta charset="utf-8">
6          <script src="global.js"></script>
7          <script src="global2.js"></script>
8          <script>
9              message = "I'm in the page.";
10             function testScope(){
11                 message += " called in testScope()";
12                 alert(message);
13             }
14         </script>
15     </head>
16     <body onload="testScope();global2Print();globalPrint();">
17         <script>
18             message += " embedded in page.";
19             document.writeln(message);
20         </script>
21     </body>
22 </html>
```

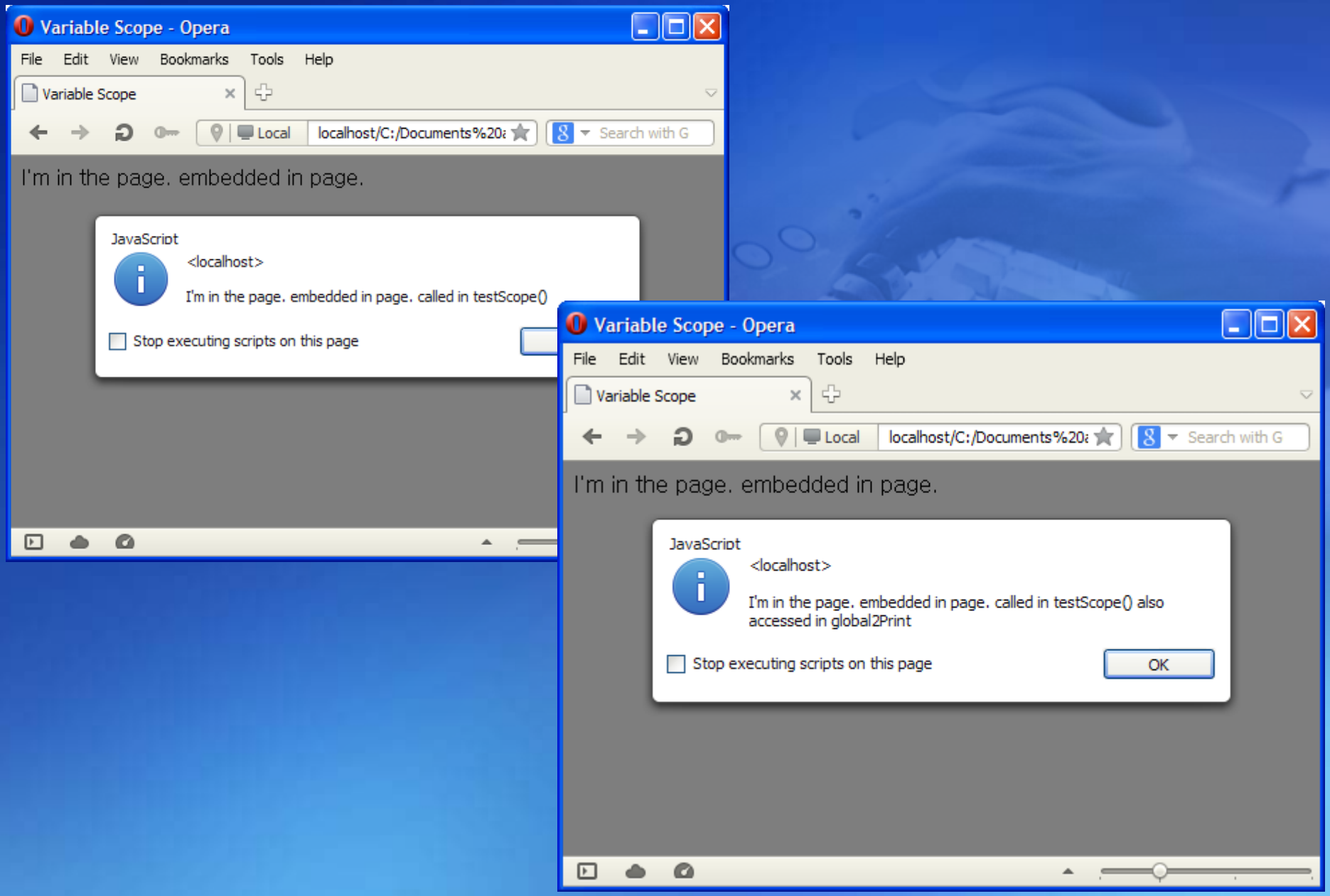
Lab2 : global.js

```
1  message = " globally in globalPrint";
2
3  function globalPrint(){
4      alert(message);
5  }
```

Lab2 : global2.js

```
1  function global2Print(){
2      message += " also accessed in global2Print";
3      alert(message);
4  }
```

Lab2 : Result


Variable Scope - Opera
File Edit View Bookmarks Tools Help
Variable Scope
Local
localhost/C:/Documents%20
Search with G
I'm in the page. embedded in page.
JavaScript
<localhost>
I'm in the page. embedded in page. called in testScope()
Stop executing scripts on this page
Variable Scope - Opera
File Edit View Bookmarks Tools Help
Variable Scope
Local
localhost/C:/Documents%20
Search with G
I'm in the page. embedded in page.
JavaScript
<localhost>
I'm in the page. embedded in page. called in testScope() also accessed in global2Print
Stop executing scripts on this page
OK

Constants

- Can create a read-only, named constant with the const keyword.

```
const PREFIX = '212';
```

- A constant cannot change value through assignment.
- Cannot declare a constant with the same name as a function or variable in the same scope.

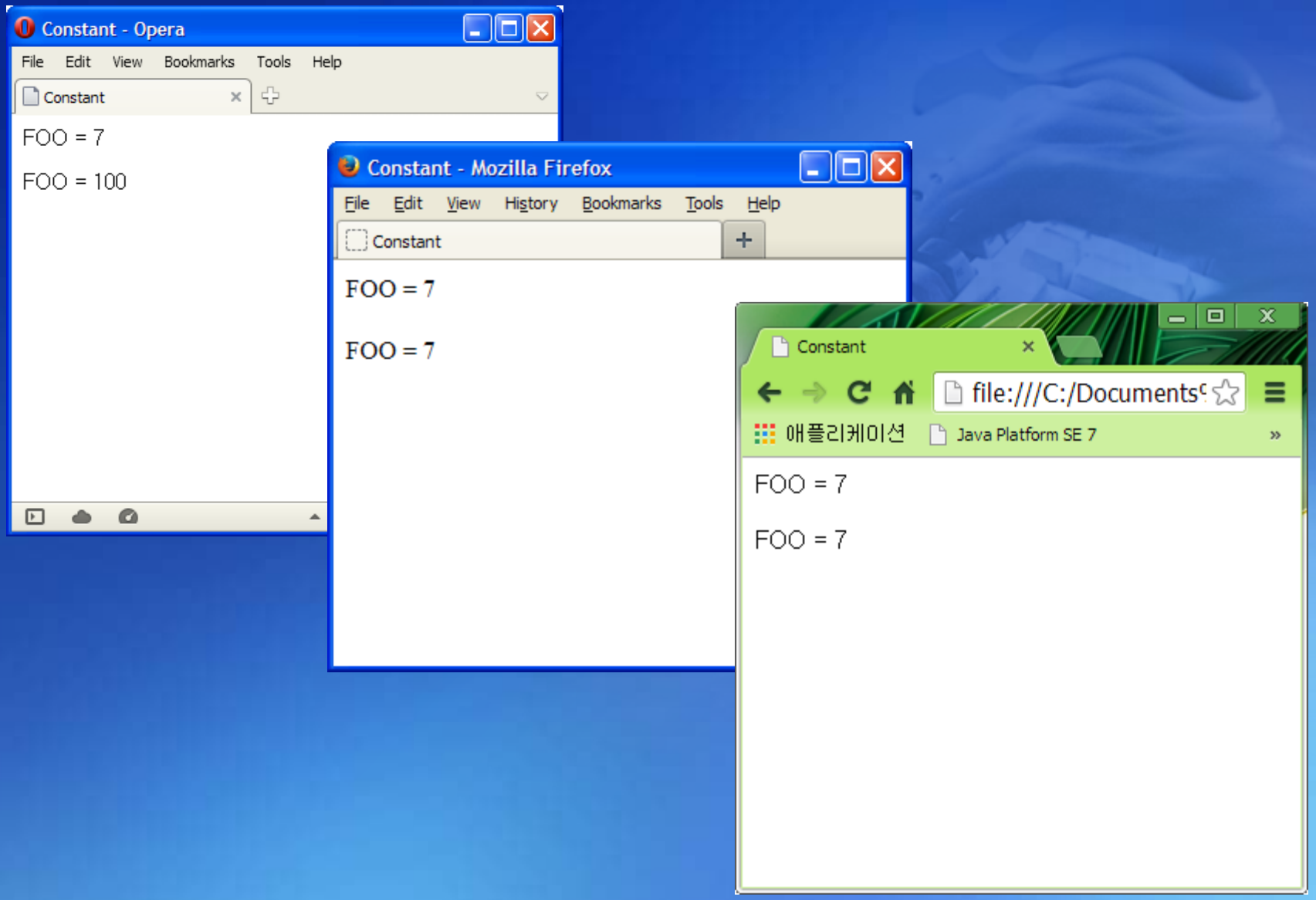
Lab3 : Constant

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

Lab3 : constant.html

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title> Constant </title>
5          <meta charset="utf-8">
6      </head>
7      <body>
8          <script>
9              const FOO = 7;
10             document.write("FOO = " + FOO + "<br><br>");
11
12             FOO = 100;
13             document.write("FOO = " + FOO + "<br><br>");
14         </script>
15     </body>
16 </html>
```

Lab3 : Result


Constant - Opera
File Edit View Bookmarks Tools Help
Constant
FOO = 7
FOO = 100
Constant - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Constant
FOO = 7
FOO = 7
Constant
file:///C:/Documents
애플리케이션
Java Platform SE 7
FOO = 7
FOO = 7

Naming Guidelines

- Use meaningful words.

```
var interestRate = .75;
```

- Use a data type clue as part of the name.

```
var strFirstName = “Shelley”;
```

- Use a plural for collections of items.

```
var customerNames = new Array();
```

- Variables are not capitalized.

```
var firstName = String(“Shelley”);
```

Naming Guidelines

- Functions and variables frequently start with lowercase letters, incorporate a verb representing what the function is doing.

```
function validateNameInRegister(firstName, lastName) { }
```

- Variables and functions have one or more words concatenated into a unique identifier.

```
validateName,    firstName
```

Primitive Types

- Three primitive data types: *string*, *numeric*, and *boolean*.
- Also has built-in objects known as `String`, `Number`, and `Boolean`.
- The two are connected.
- For instance, the `String` object wraps the string primitive.

```
var firstName = "Shelley";
var cappedName =
firstName.toUpperCase();
```

The `String` Data Type

- A string literal is a sequence of characters delimited by single or double quotes.

```
var strString = "This is a string";
var anotherString= 'But this is also a string';
```

- The ending quote character must be the same as the beginning one.
- A string can include quotes.

```
var string_value = "This is a 'string' with a quote.”
var string_value = 'This is a "string" with a quote.'
```

String Escape Sequences

- A string can contain an escape sequence, such as \n for the end-of-line terminator.

```
var str = “These are a apple\n and grape”;
```

```
These are a apple
and grape
```

- You can also use the backslash to denote that the quote in the string is meant to be taken as a literal character, not as an end-of-string terminator:

```
var string_value = "This is a \"string\"
with a quote."
```

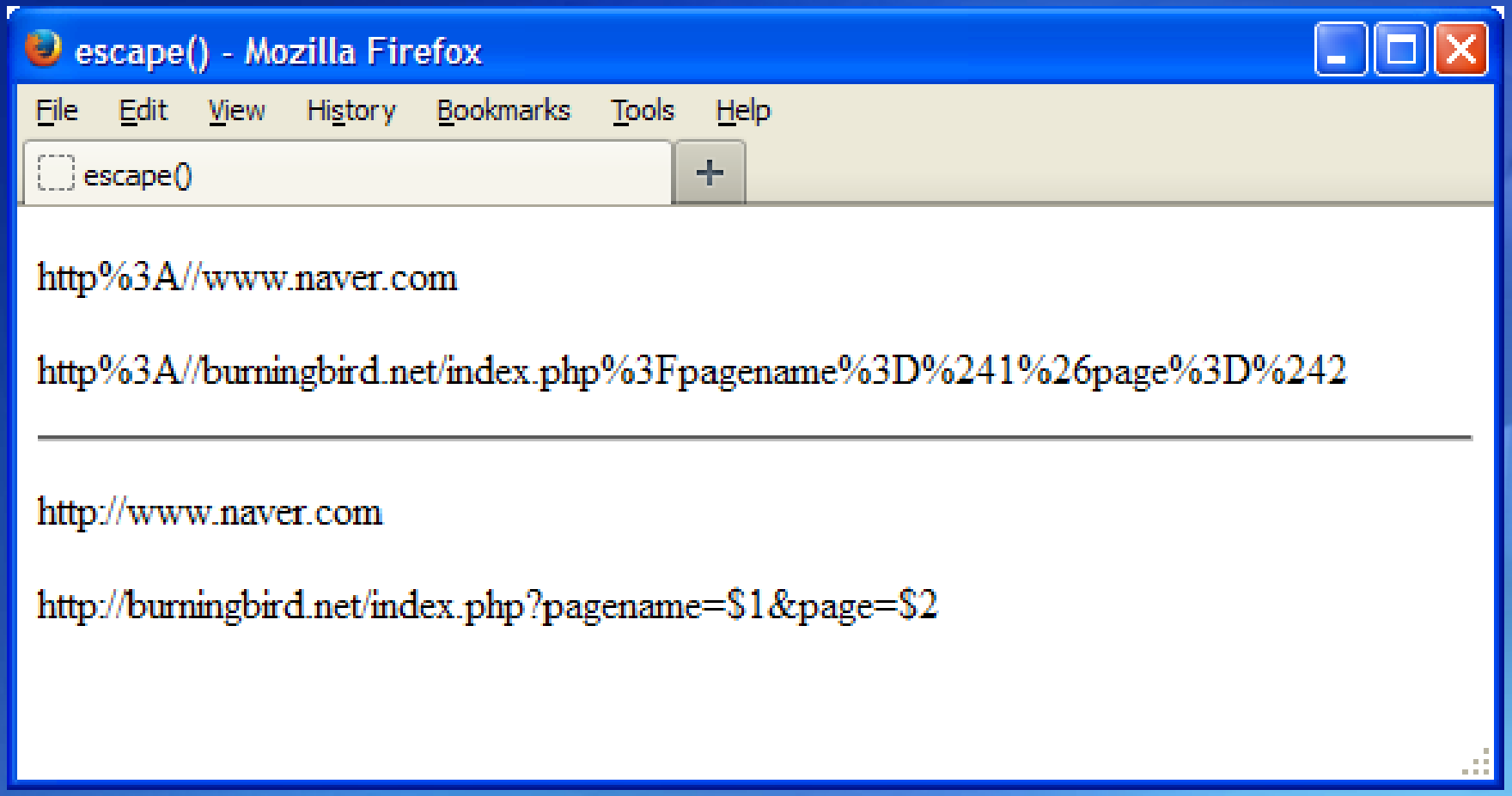
Lab4 : escape(), unescape()

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

Lab4 : escape.html

```
<!DOCTYPE html>
<html>
   <head>
      <title> escape() </title>
      <meta charset="utf-8">
   </head>
   <body>
      <script>
         var str = escape("http://www.naver.com");
         document.write("<p>" + str + "</p>");

         var str1 = escape("http://burningbird.net/index.php?pagename=$1&page=$2");
         document.write("<p>" + str1 + "</p>");

         document.write("<hr>");

         str = unescape("http%3A//www.naver.com");
         document.write("<p>" + str + "</p>");

         str1 =
           unescape("http%3A//burningbird.net/index.php%3Fpagename%3D%241%26page%3D%242");
           document.write("<p>" + str1 + "</p>");

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

Lab4 : Result


escape() - Mozilla Firefox
File Edit View History Bookmarks Tools Help
escape()
http%3A//www.naver.com
http%3A//burningbird.net/index.php%3Fpagename%3D%241%26page%3D%242
http://www.naver.com
http://burningbird.net/index.php?pagename=$1&page=$2

String Encoding

- Encodes a URI by replacing each instance of certain characters by one, two, three, or four escape sequences.
- Use the **`encodeURI`** and **`encodeURIComponent`** methods.
- Can use in links and Ajax applications.

Type	Includes
Reserved Characters	; , / ? : @ & = + $
Unescaped Characters	Alphabetic, decimal digits, - _ . ! ~ * ‘ ( )
Score	#

String Encoding (Cont.)

- **encodeURI(uri)**
  - Encodes special characters, *except* :

    , / ? : @ & = + $ #

```
<script type="text/javascript">
   var uri="my test.asp?name=ståle&car=saab";
   document.write(encodeURI(uri)+ "<br />");
   //var uri="my%20test.asp?name=st%C3%A5le&car=saab";
</script>
```

String Encoding (Cont.)

- **encodeURIComponent(uri)**
  - Encodes special characters.
  - Encodes the following characters.

    , / ? : @ & = + $ #

```
<script type="text/javascript">
   var uri="http://w3schools.com/my test.asp?name=ståle&car=saab";
   document.write(encodeURIComponent(uri));
   //http%3A%2F%2Fw3schools.com%2Fmy%20test.asp%3Fname%3D...
</script>
```

String Encoding (Cont.)

```
function encodeStrings() {
    var uri = "http://burningbird.net/index.php?pagename=$1&page=$2";
    var sOne = encodeURIComponent(uri);
    var uri1 = "http://someapplication.com/?catsname=Zöe&URL=";
    var sTwo = encodeURI(uri1);
    var sOutput = "<p>Link is " + sTwo + sOne + "</p>";
    document.write(sOutput);
    var sOneDecoded = decodeURI(sTwo);
    var sTwoDecoded = decodeURIComponent(sOne);
    var sOutputDecoded = "<p>" + sOneDecoded + "</p>";
    sOutputDecoded += "<p>" + sTwoDecoded + "</p>";
    document.write(sOutputDecoded);
    /*
    Link is
    http://someapplication.com/?catsname=Z%C3%B6e&URL=http%3A%2F%2Fb
    urningbird.net%2Findex.php%3Fpagename%3D%241%26page%3D%242
    http://someapplication.com/?catsname=Zöe&URL=
    http://burningbird.net/index.php?pagename=$1&page=$2
    */
```

Converting to Strings

- `var num_value = 35.00;`
  `alert(num_value); // expects a string`
- `var num_value = 35.00;`
  `var string_value = "This is a number:" + num_value;`
- `var strValue = "4" + 3 + 1; // "431"`
  `var strValueTwo = 4 + 3 + "1"; // 71`
- `var firstResult = "35" - 3; // 32`
- `var secondResult = 30 / "3"; // 10`
- `var thirdResult = "3" * 3; // 9`

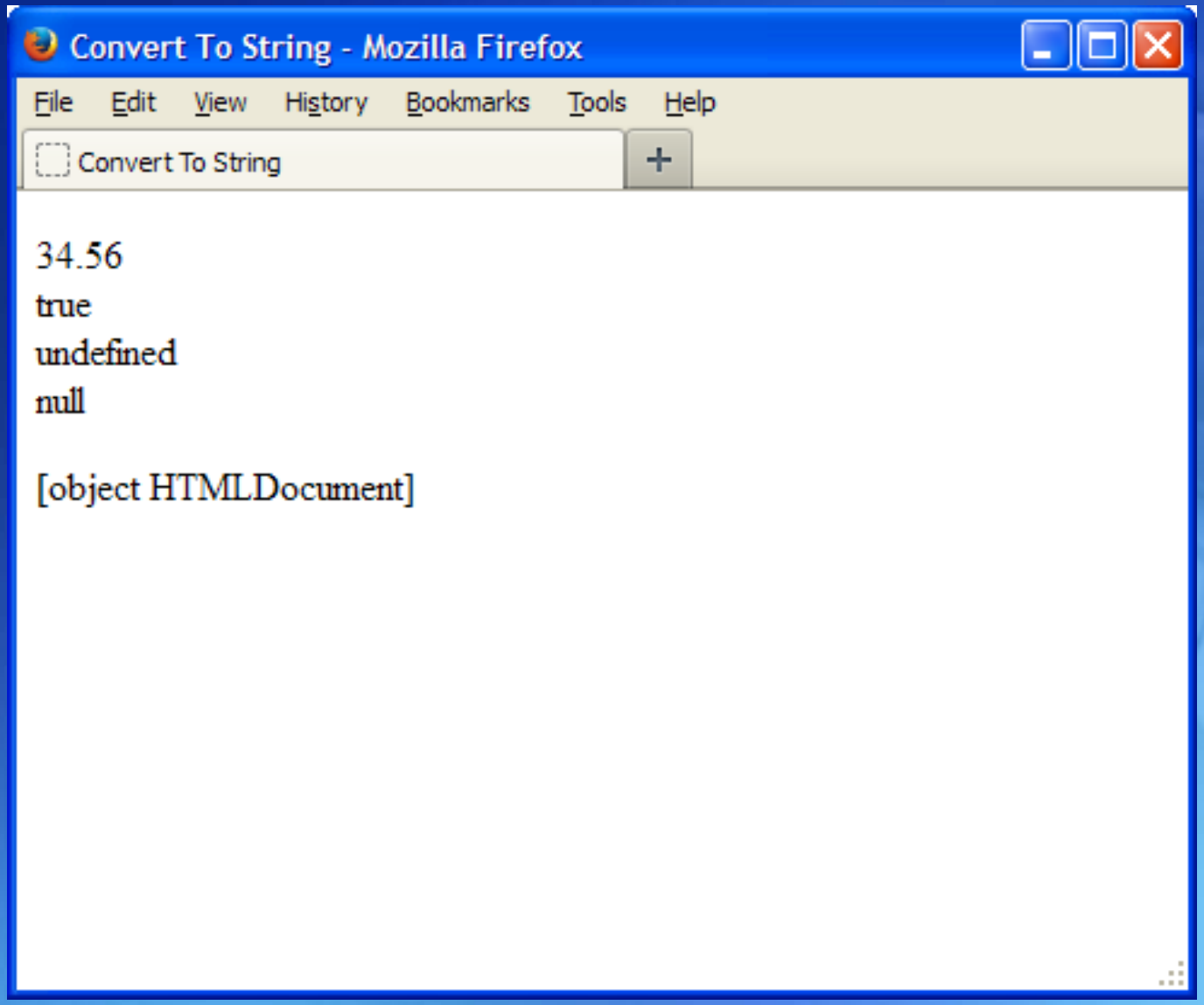
Lab5 : Convert to String

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

Lab5 : convertstring.html

```
7   <body>
8     <script>
9           var newNumber = 34.56;
10          var newBoolean = true;
11          var nothing;
12          var newNull = null;
13          var strNumber = String(newNumber);
14          var strBoolean = String(newBoolean);
15          var strUndefined = String(nothing);
16          var strNull = String(newNull);
17          var strOutput = "<p>" + strNumber + "<br>" + strBoolean + "<br>" + strUndefined +
18            "<br>" + strNull + "</p>";
19          document.writeln(strOutput);
20          var strOutput2 = String(document);
21          document.writeln(strOutput2);
22    </script>
23  </body>
```

Lab5 : Result


Convert To String - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Convert To String
34.56
true
undefined
null
[object HTMLDocument]

The `Boolean` Data Type

- The boolean data type has two possible values: `true` and `false`.
- They are not surrounded by quotes.
- "false" is *not* the same as false.

```
var isMarried = true;
var hasChildren = false;
```

The `Boolean` Data Type (Cont.)

Input	Result
Undefined	`false`
Null	`false`
Boolean	Value of value
Number	Value of `false` if number is 0 or NaN; otherwise, `true`
String	Value of `false` if string is empty; otherwise, `true`
Object	`true`

The `Number` Data Type

- Are floating-point numbers.
- If don't have a decimal point or fractional component, treated as integers.
- base-10 whole numbers in a range of -2^{53} *to* 2^{53}.
- The following are valid integers:
  - `var negativeNumber = -1000;`
  - `var zero = 0;`
  - `var fourDigits = 2534;`

The `Number` Data Type (Cont.)

- Represent the number as an exponent, using scientific notation.
- All of the following are valid floating-point numbers:
  - `var someFloat = 0.3555`
  - `var negDecimal = -2.3;`
  - `var lastNum = 19.5e-2`
- Numbers in a range of `–2e31 to 2e31` (`–2,147,483,648 to 2,147,483,648`).

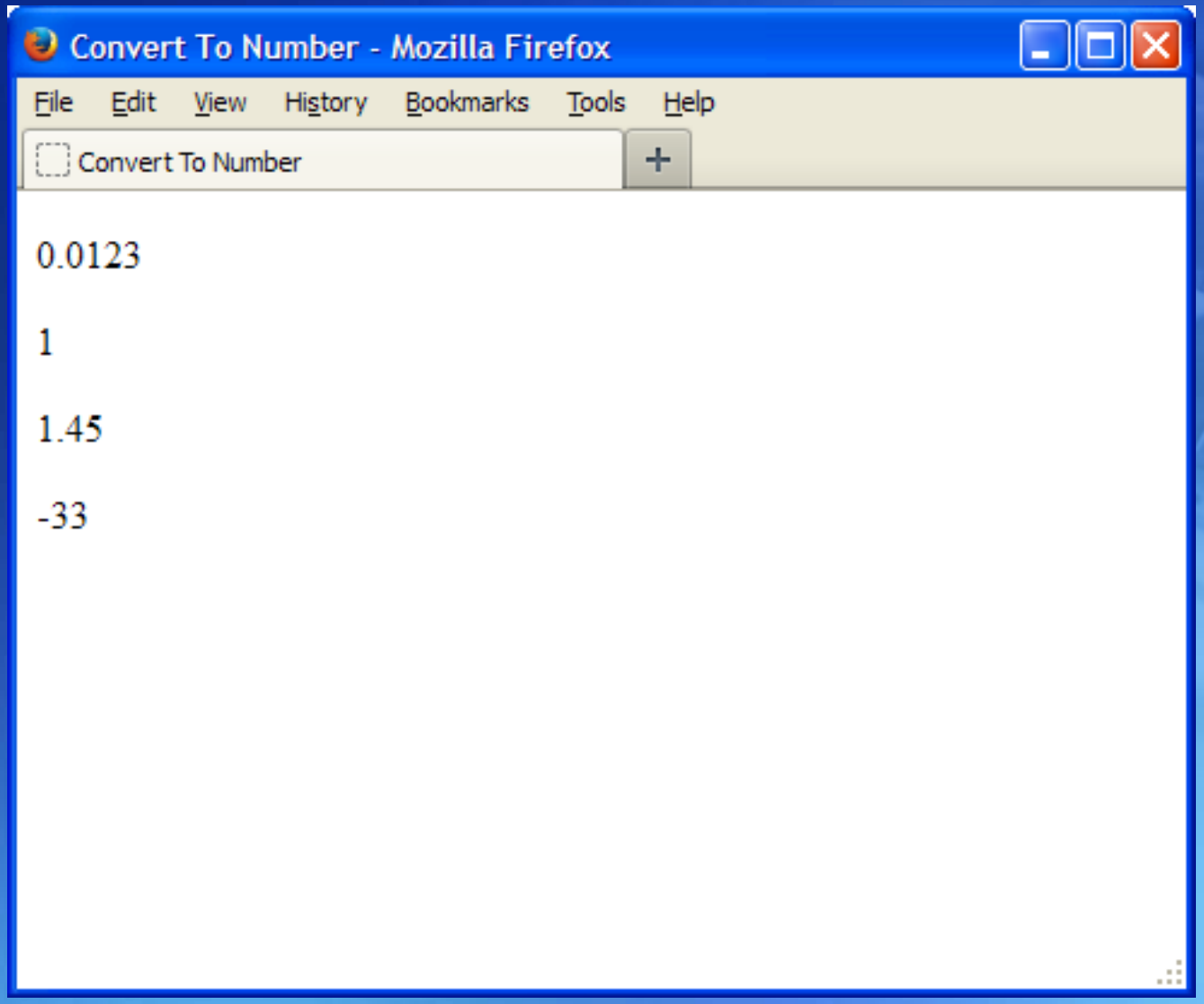
Lab6 : Convert to Number

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

Lab6 : convertnumber.html

```
3   <head>
4       <title> Convert To Number </title>
5       <meta charset="utf-8">
6   </head>
7   <body>
8       <script>
9           var sNum = "1.23e-2";
10          document.writeln("<p>" + parseFloat(sNum) + "</p>");
11          document.writeln("<p>" + parseInt(sNum) + "</p>");
12          var fValue = parseFloat("1.45 inch");
13          document.writeln("<p>" + fValue + "</p>");
14          var iValue = parseInt("-33.50");
15          document.writeln("<p>" + iValue + "</p>");
16      </script>
17  </body>
```

Lab6 : Result


Convert To Number - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Convert To Number
0.0123
1
1.45
-33

The `Number` Data Type (Cont.)

- The `parseInt` function can convert a decimal to an octal or a hexadecimal and back again.
- A second parameter to the function, base, is equivalent to the number *radix*, *and* is 10 or base 10, by default.
- If any other base is specified in a range from 2 to 36, the string is interpreted accordingly. The following JavaScript:

```
var iValue = parseInt("266",16); //550
var iValue = parseInt("55",8);   //45
```

The `null` and `undefined` Variables

- A null variable is assigned *null* as its value.

```
var nullString = null;
```

- If have declared but not initialized the variable.

```
var undefString;
```

- A variable is undefined, even if it is declared, until it is *initialize*d.

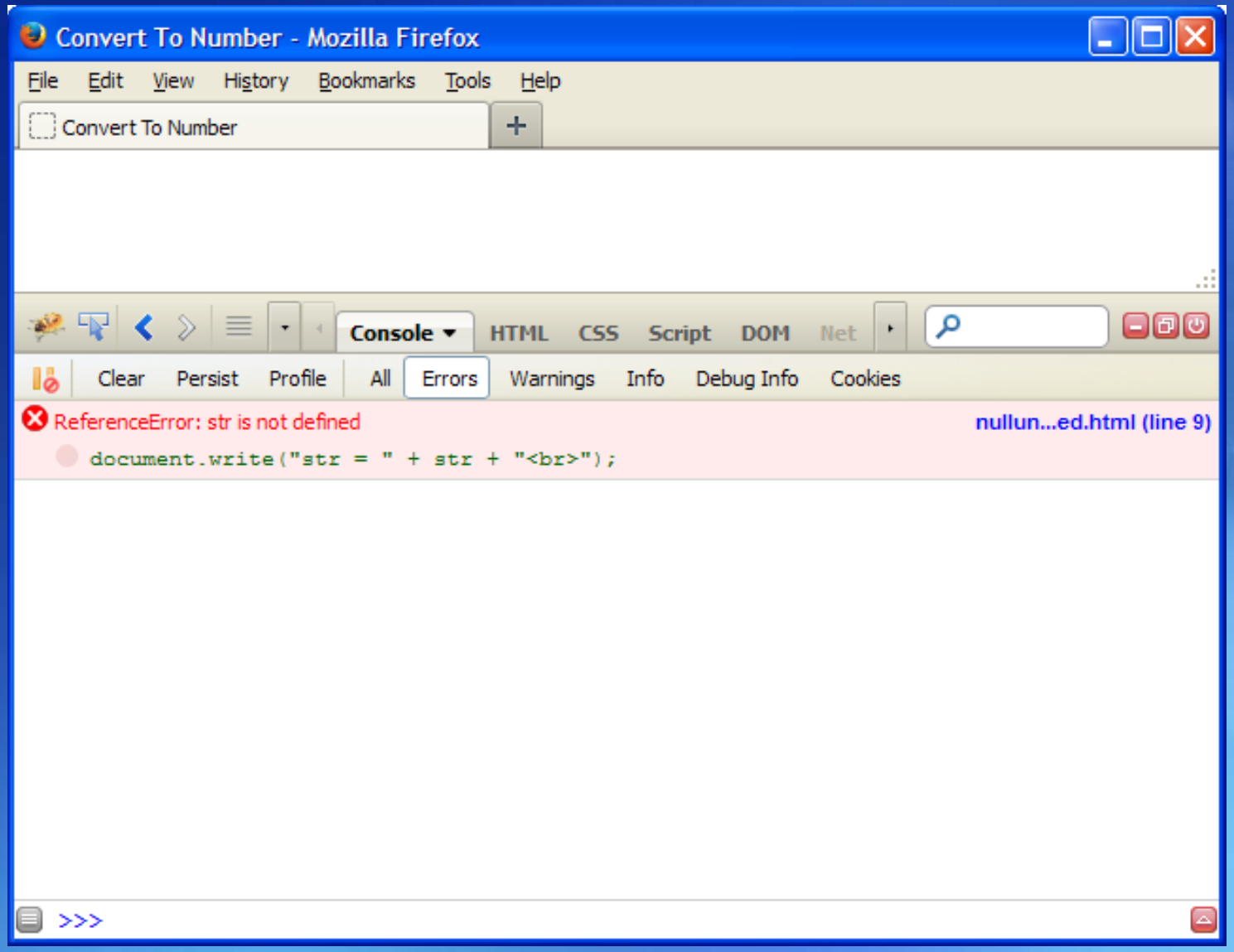
Lab7 : null

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

Lab7 : null.html

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4           <title> null </title>
5           <meta charset="utf-8">
6       </head>
7       <body>
8           <script>
9               document.write("str = " + str + "<br>");
10          </script>
11      </body>
12  </html>
```

Lab7 : Result


Convert To Number - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Convert To Number
Console HTML CSS Script DOM Net
Clear Persist Profile All Errors Warnings Info Debug Info Cookies
ReferenceError: str is not defined
nullun...ed.html (line 9)
document.write("str = " + str + "
");
>>>

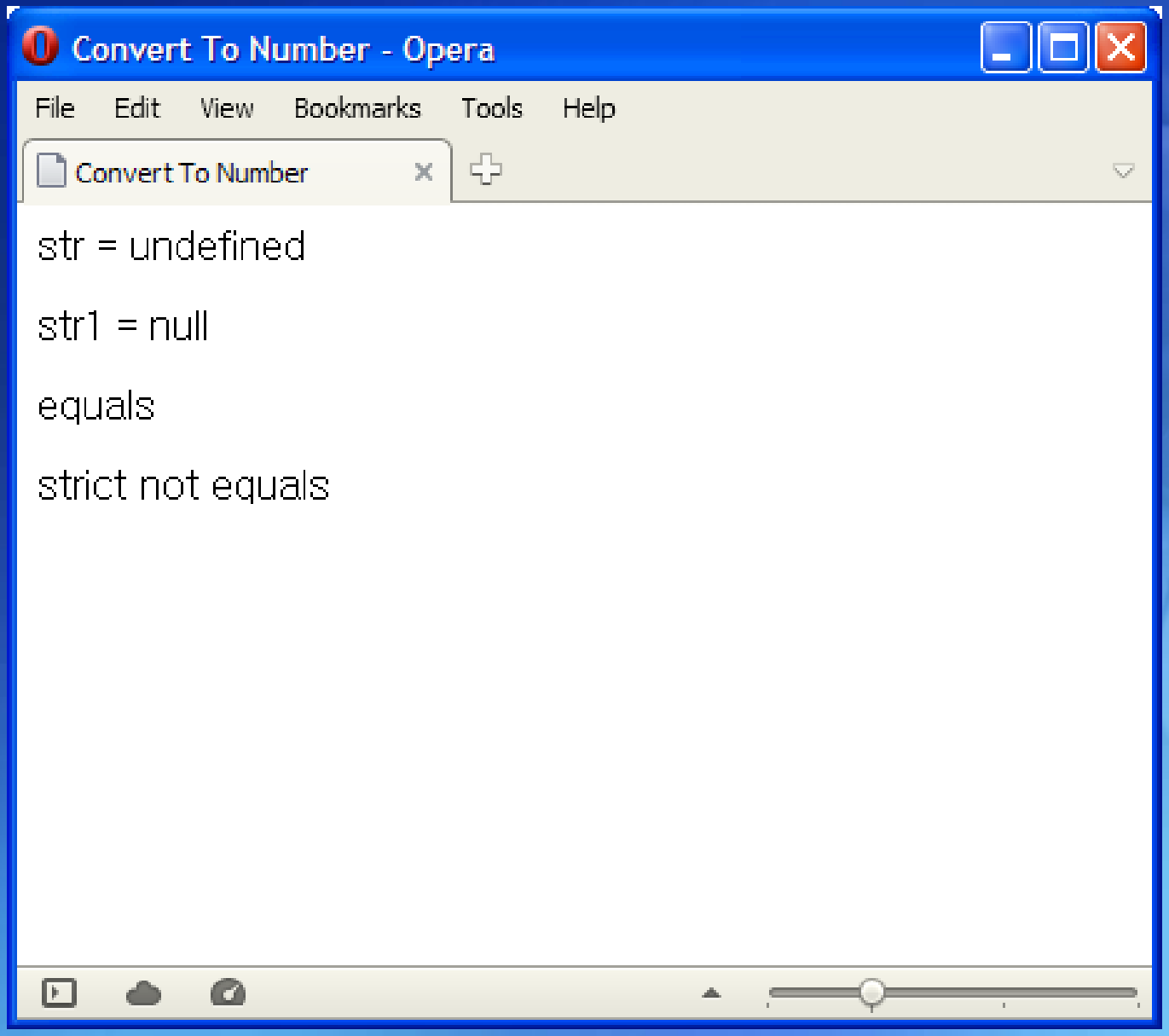
Lab8 : null & undefined

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

Lab8 : nulldefined.html

```
3   <head>
4       <title> null & undefined  </title>
5       <meta charset="utf-8">
6   </head>
7   <body>
8       <script>
9           var str;
10          document.write("str = " + str + "<br><br>");   //undefined
11
12          var str1 = null;
13          document.write("str1 = " + str1 + "<br><br>");   //null
14
15          if(str == null) document.write("equals<br><br>");
16          else document.write("not equals<br><br>");
17
18          if(str === null) document.write("strict equals<br><br>");
19          else document.write("strict not equals<br><br>");
20
21      </script>
22  </body>
```

Lab8 : Result


Convert To Number - Opera
File
Edit
View
Bookmarks
Tools
Help
Convert To Number
str = undefined
str1 = null
equals
strict not equals

Quiz

1. Of the following identifiers, which are valid, which are not, and why?

```
$someVariable
_someVariable
1Variable
some_variable
somevariable
function
some*variable
```

Quiz

1. Of the following identifiers, which are valid, which are not, and why?

```
$someVariable
_someVariable
1Variable
some_variable
somevariable
function
some*variable
```

Quiz (Cont.)

2. Convert the following identifiers to CamelCase notation:

```
var some_month;
function theMonth
   ← function to return current month
current-month ← a constant
var summer_month;
   ← an array of summer months
MyLibrary-afunction
   ← a function from a JavaScript
  package
```

Quiz (Cont.)

2. Convert the following identifiers to CamelCase notation:

```
someMonth;

function getCurrentMonth

CURRENT_MONTH

summerMonths;

myLibraryFunction
```

Quiz (Cont.)

3. Is the following string literal valid? If not, how would you fix it?

```
var someString = 'Who once said,
  "Only two things are infinite,
  the universe and
  human stupidity, and I'm not
  sure about the former."'
```

Quiz (Cont.)

3. Is the following string literal valid? If not, how would you fix it?

```
var someString = "Who once said,
  'Only two things are infinite,
  the universe and
  human stupidity, and I'm not
  sure about the former.'"
```

Quiz (Cont.)

4. Given a number, **432.54**, what JavaScript function(s) returns the integer component of the number, and then finds the hexadecimal and octal conversions?

Quiz (Cont.)

4. Given a number, **432.54**, what JavaScript function(s) returns the integer component of the number, and then finds the hexadecimal and octal conversions?

```
var fltNumber = 432.54;
var intNumber = parseInt(fltNumber);
var octNumber = intNumber.toString(8);
var hexNumber = intNumber.toString(16);
```

Quiz (Cont.)

5. You're creating a JavaScript function in a library that other applications can use.

   A parameter, `someMonth`, is passed to the function. How would you determine whether it's `null` or `undefined`?

Quiz (Cont.)

5. You're creating a JavaScript function in a library that other applications can use.

   A parameter, `someMonth`, is passed to the function. How would you determine whether it's `null` or `undefined`?

```
if(someMonth) {
    …
}
```