

Variables, Operators, Expressions - Lecture 2

1. Review previous input & output options used by JavaScript

2. <Server></Server> Tags

This tells the browser to treat code within the tags as JavaScript. It has no immediate effect on the client's web page but it invokes action on the server.

3. <noscript> </noscript>

JavaScript capable browsers recognize this tag and ignore the content within it. A non-JavaScript capable browser does not recognize the tags and therefore executes the code.

E.g.

```
<Body>
<script language="JavaScript">
<!-- hide
// JavaScript statements
// stop hiding -->
</script>

<noscript>
<H3>You must have an older browser to this text</H3>
</noscript>
</Body>
```

4. Creating Variables

see page 9 Negrino Text

- a) What is a variable?
- b) 6 Standard data types
- c) JavaScript reserved words - see textbook page 184 for list
- d) How variables are assigned a value
- e) Mathematical Operators & Expressions
- f) global and local variables

a) A variable is a user defined storage space in the computer's memory. It is a place in the computer's RAM where you can store a value i.e. a number, text string, True or False etc. Variables allow JavaScript to recall and reuse a piece of data or value.

b) Variable Types "Data Types"

A value is a piece of information called a data type

Standard Variable Types include:

- a) Number 3, 4, -2, 3.141
- b) String "Hello World"
- c) Boolean True or False
- d) Null nothing
- e) Object
- f) Function

Each variable has a property known as its type. Type refers to the kind of information stored in memory.

Numbers

1, -1, 0, 5 integers Note Octal & Hexidecimal numbers
5.2, 3.141 decimals may also be used
6.2 X 10² exponentials

Strings - array of characters

e.g. "Welcome to my website"

e.g. "123" is a string of ASCII characters
note 123 without quotes is stored as binary number

Boolean

True or False
used in decision structures If, else - this is something HTML
can not do

Nulls

nothing or blank - it is a value!
used when necessary to check if a user entered information or left it blank, e.g. confirm & prompt
box

Objects and Functions can also be variables

A literal is the actual value assigned to a variable e.g.

var new=10 10 is the literal value

Declaring Variables - JavaScript is a loosely typed language which means it is not essential to declare variables as it is in other programming languages - JavaScript will attempt to identify what kind of data is being stored.

Nevertheless - it is good programming practice to declare a variable before using it. This is done using the var keyword.

```
var variable1; variables seperated by ;  
var variable2;
```

```
var variable1; var variable 2;
```

c) JavaScript Reserved words and variable names

variable names consist of

- a) letters a-z, A-Z
- b) numbers 0-9 - but cannot be leading character
- c) underscore _ - but no symbols #,\$ etc.
- d) no spaces between text, numbers
- e) can not use one of JavaScripts reserved words - See Negrino page 183-185

Variable names should be descriptive and are case sensitive*

*Note although variable names are case sensitive, I.E. 3.0 does not recognize that "Fred" and "fred" are different, so it is best not use two different variables that differ only by case.

d) Assigning values to variables

Example Scripts

(assume script tags are used and the following scripts are placed within the <Head></Head> tags.)

#1.

```
var price=34.99;  
document.write(price)  
// variable is treated as a number
```

#2

```
var price=$34.99;  
document.write(price)
```

outputs an error! Confused the browser does know if number or string?

#3

```
var price="$34.99"  
document.write(price)  
// outputs $34.99 as a "literal" or text string
```

#4

```
var example="Hello World";  
document.write(example);
```

If you remove the "" quotes in Hellow World you will get an error message.

#5

```
var number1=34
var number2=36
document.write(number1 + number 2)

//output 70
```

#6

```
var number1=34
var number2=36
document.write("number1" + "number 2")

//output number1 number2 no var declared
```

#7

```
document.write(34 + 36)
//output 70
```

#8

```
document.write("34" + "36")
//output 3436
```

#9

```
var number1="34"
var number2="36"
document.write(number1 + number 2)
//output 3436
```

#10 Enter and store a variable using a prompt box

```
var name=prompt("Enter your name", "name")
document.write("Greetings " + name + " Welcome to my website!")
// a prompt by default stores entered variables as text
```

#11 Storing values as Booleans

A Boolean variable has two possible values True or False, because these values are not strings do not surround them with quotation marks.

```
var doYou // declare variable but do not give it a value
```

```
doYou=confirm("Do you like Hot dogs?)
document.write("I think it is ")
document.write(doYou) //output value input as True or False
document.write("that you really like hotdogs.")
```

Output on the screen should be:

I think it is True\False that you really like hotdogs

The scope of a variable refers to whether it is local or global. Global variables can be used by any function and are defined outside the function, whereas a local variable is defined inside a function and only exists while that function is "running". See section on functions.

scope refers to the variables visibility within a program

Expressions & Mathematical Operators

- a) What is an expression
- b) how to assign the result of an expression to a variable
- c) the different types of expressions
- d) standard JavaScript operators
- e) the order or precedence

a) An expression is any operation on data types that evaluates to a single value, such as

```
cartons= 3 + 8
```

- says add 3 and 8 and place the result in memory referenced by the variable cartons

Expressions combine values into a new value

Operators are used to perform operations, in the example above "+" and "=" are operators

Operations on the right of the = sign are always performed first. This is called order of precedence.

Variables may be declared and assigned at the same time, but must be declared before use.

```
var cartons = 3 + 8;  
var eggs = 12 * cartons;
```

This is a legitimate expression, but if the order of the statements were reversed an error would result since eggs relies on the valued stored in cartons for its declaration

```
var broken=broken + 1 // incorrect since broken does not have meaning // until broken itself is declared.
```

JavaScript recognizes 3 Types of Expressions

- 1) Arithmetic
- 2) String
- 3) Logical

Arithmetic E.g. tax = 0.07 * price
String E.g. filename = "win" + ".ini"
Logical E.g. discount = true

Different types of operators

- 1) Arithmetic
- 2) Assignment
- 3) Comparisons
- 4) Logical Operators
- 5) Conditional Operators
- 6) Bitwise operators

1) Arithmetic Operators - Negrino Page 9

X + Y if x, y are numbers = num if x, y are in quotes "X" + "y" = xy

-, +, /, *, %

Modulus remainder when x is divided by y E.g. 5%2=1

X++ same as x=x+1 shorthand

++X x+1=X

X-- x=x-1

--X x-1=x

-X reverse sign

++X adds one before any operation

X++ adds one after any operation

--X subtracts one before any operation

X-- subtracts one after any operation

Screen Output

Test Value of a Value b

```
var a=100 b=a++ 101 100
```

```
var b= a++ b=++a 101 101
```

```
document.write (a); b=a-- 99 100 document.write (b); b=--a 99 99
```

2) Assignment Operators Table 1.4 page 10 Negrino

Putting a value into a variable this usually the final operation performed in an expression by equating a variable to a literal value. The = sign is not the only assignment operator.

Assignment What it does

x = y Sets x = y

x += y x = x+y

x -= y x = x-y

x *= y x = x*y

x /= y x = x/y

x %= y x = x%y

x ^= y x = x^y

x <<= y

x >>= Y

```
x|=y  
x&=y
```

3) Comparison Operators see Table 1.5 Negrino

Comparison operators generate a TRUE or FALSE output based on a logical comparison of the operands on either side of the comparison. There are 6 kinds of comparison operators

```
== set to or equals ( not to be confused with =)  
!= not equals (return true if x is not equal to y)  
< less than (return true if x is less than y)  
> greater than  
<= less than or equal to  
>= greater than or equal to
```

In some languages there is no distinction made between = and ==

```
x == y return true if x and y are equal
```

Logical Comparison

```
x && y return true if x and y are equal  
x || y return true if x or y are equal  
!x return true if x is false
```

4) String operator + concatenates text, or text and number

```
x = cat + 5  
x= cat5
```

This is called CASTING the number is cast into text, using + treats text and a number as a string and casts the 5 into the text string.

To avoid this use

```
parseInt() or parseFloat()
```

```
parseInt("13")  
returns value 13
```

```
parseFloat("45.2")  
returns 45.2
```

These javascript functions convert strings into integers or floating point numbers.

5) Logical Operators work on true or false value

E.g.

```
var snow=true;  
var day="Monday"  
var month="11"
```

These are assignments not comparisons

Operators

&& logical and, returns true if both operands are true, otherwise false;

|| logical or, returns true if either operand is true

! not, returns true if the operand is false and false if the operand is true

6) Conditional operators

Evaluate one of two different values based on a condition

(condition) ? val1 : val2

val1 (true condition)

val2 (false condition)

e.g.

(day == "Saturday") ? "Weekend!" : "Not Saturday!"

Assignment #1 Exercise Ask the user what the value of 10 + 10 is and if he/she is correct return output to the screen Correct or Try again - use the variable and the conditional statement.

Answer

```
<Script language = "JavaScript">
```

```
//define variables
```

```
var question="What is 10 + 10?"
```

```
var answer="20";
```

```
var incorrect="Sorry try again";
```

```
var correct="Yes you are correct!";
```

```
var response=prompt(question, "0") // ask the question
```

```
var output = (response == answer) ? correct : incorrect; //condition
```

```
document.write(output) //output to screen
```

```
</script>
```

7) Bitwise operators

Operate on bits of a variable i.e. 1s and 0s, treats numbers as a 32 bit value

NOT ~
AND &
OR |
NOR ^
>>
>>>
<<

Operators are rarely if ever used in Web-style applications

Assignment #2

#11 Using the prompt box, Query the user to enter two numbers and then output to the screen "The sum of input1 + input 2 is = "calculated sum"

HINT - in order for the prompt box to store the input value as a number you will have to assign variable

`parseInt(number)`

```
var number1=prompt("Enter a number", "number1")
var number1=parseInt(number1)
var number2=prompt("Enter a second number", "number2")
var number2=parseInt(number2) // var can be left out still works
var result=number1 + number2
document.write("<H3>The sum of number1 and number2 is = " + result + "</P>")
```

/ inserting number1 + number 2 results in output the variable as literals ie number1 + number2 rather than summing the values, therefore assign the operation to a variable = result */*

d) Order of precedence refers to the order in which operations are evaluated.

Suppose you wan to calculate how many dozen eggs he has has in total, based on stored eggs minus the number of broken eggs.

```
dozens = total_eggs - broken_eggs/12;
```

In this statement the division occurs first since it takes precedence over the subtraction , so the expression is evaluated to

```
dozens = total eggs - (broken_eggs/12);
```

When what was intended

```
dozens = (total eggs - broken_eggs)/12
```

Operator precedence

1. () []
2. ! ,~, -, ++, -- (negation, increment)

3. * , / , %
4. + -
5. etc.