

Dear Friend,

Hello, & welcome back to **FIREWIRE**, your friendly guide to the exciting world of web-design & web-solutions! In the previous newsletter of Flash 5 Tutorial Series we had discussed about Math Objects, Date & String Function. Now let's Continue with Bitwise Operators, Comparison Operators, Logical Operators, Numeric Operators, String Operators, Operators.

Bitwise Operators

XOR (^)

Using XOR operator, two expressions are converted to 32 bit unsigned integers and return 1 in each bit position where the corresponding bits in expr 1 and expr 2, but not both are one.

Syntax :

expr 1 ^ expr 2

Example :

```
trace (15 ^ 9);
```

The output window shows the value 6.

OR (|)

Used to perform a 32 bit unsigned integers.

Syntax :

expr 1 | expr 2

Example :

```
x = 15;  
y = 9 ;  
trace (x|y);
```

The output window shows a value "15".

Note :

In XOR,

Binary value of 15 - 1111

Binary value of 9 - 1001

(1111 ^ 1001 - 0110) (0110 - 6)

In OR,

Binary subtraction takes place as

(1111 | 1001 - 1111)

(1111 - 15)

Bitwise AND (&)

Using this operator, two expression can be converted to 32 bit unsigned integers and perform a Boolean AND operators.

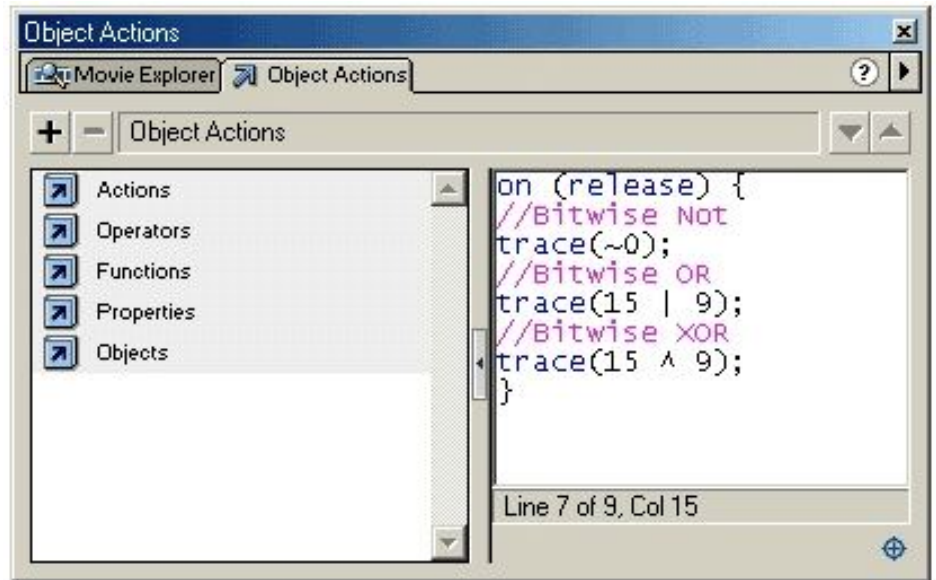
Syntax :

expression 1 & expression 2

Example :

```
If (name == "galaxy" & password == "layout")
{
trace ("welcome");
}
```

The example will check for both the condition and show the output as welcome.



Bitwise Not :

Used to convert expressions to 32 bit unsigned integer, then invert the bits.

To simply explain, changes the sign of the number and subtracts one.

Syntax :

(expression)

Example :

```
trace ( ( 0) );
trace ( ( 1) );
```

The output will be -1 and -2.

Comparison Operators

Using the greater than operator, two expressions can be tested for greater ness.

Syntax :

expr 1 > expr 2

Example :

```
X = 6;
Y = 5;
```

```
trace (x > y);
```

The output window shows "true".

The greater than and equal to is used to compare and greatness of and equality of two expressions.

Syntax :

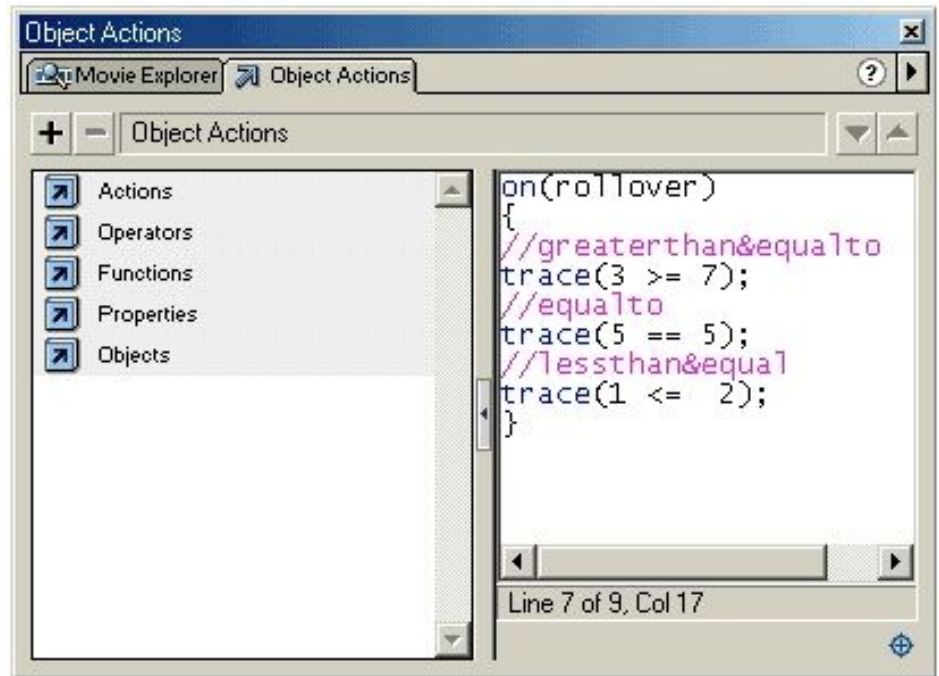
expr 1 >= expr 2

Example :

X = 5

Y = 5

Trace (x>=y);



The output window will show true. Since it is >= & hence x=y (i.e.5=5)

Less , Less Than and Not Equal Operator(<=, <> or <)

These operators are used to compare two expressions. The less than or equal operator checks for whether any one expression is less or equal to the other.

Syntax :

expression 1 <= expression 2

Example :

on (press)

{

x=5 ; y = 10

trace (number(x) <= number (y))

}

The output window will show which number is less(i.e. true).
It will return only Boolean value.

Inequality (<>)

This operator checks exactly opposite for equality.
If the expressions are not equal, then it will return true.

Syntax :

expression 1 <> expression 2

Example :

```
a = 2;  
b = 3;  
trace (a < > b);
```

The output will show (that a < > b) a Boolean value.

Less than (<)

This operator will check whether expression 1 is less than expression 2

Syntax :

```
expr 1 < expr 2
```

Example :

```
a = 2;  
b = 1;  
trace (a > b);
```

The output window shows "false".

Logical Operator

Equal (==)

This operator is used to check for equality.

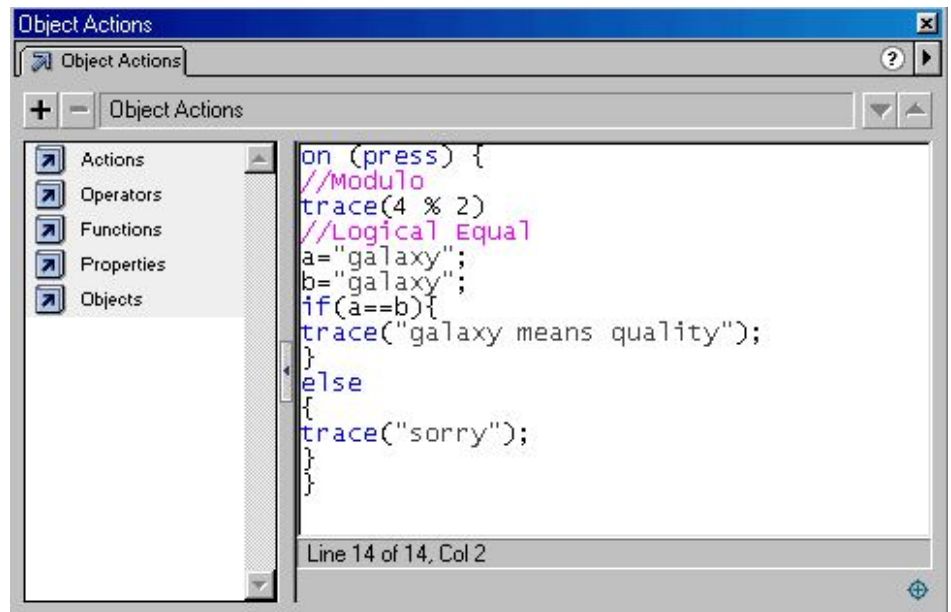
Syntax :

```
expression 1 == expression 2.
```

Example:

```
a = "galaxy"  
b = "galaxy"  
if (a == b)  
{  
  trace ("galaxy means quality");  
}  
else  
{  
  trace ("sorry");  
}
```

In the above example, both a and b are checked for equality and in the output window "galaxy means quality" will be displayed.



Modulo (%)

This operator returns the remainder of expression 1 and expression 2.

Syntax :

expression 1 % expression 2

Example :

```
on (press)
{
x = 4;
y = 2;
trace (x%y);
}
```

From the above example, the output window will show a result "0".

Not (Logical)

The operators invert the Boolean value of a variable or expression.

Syntax :

!(expression)

Example :

```
on (press)
{
galaxy = false;
if (! Galaxy)
{
trace ("don't worry");
}
}
```

On pressing, the button in the output window, the message "Don't worry" will appear.

Not Equal

Using this operator, the opposite of equality operator is tested.

If expression 1 is not equal to expression 2 it returns true.

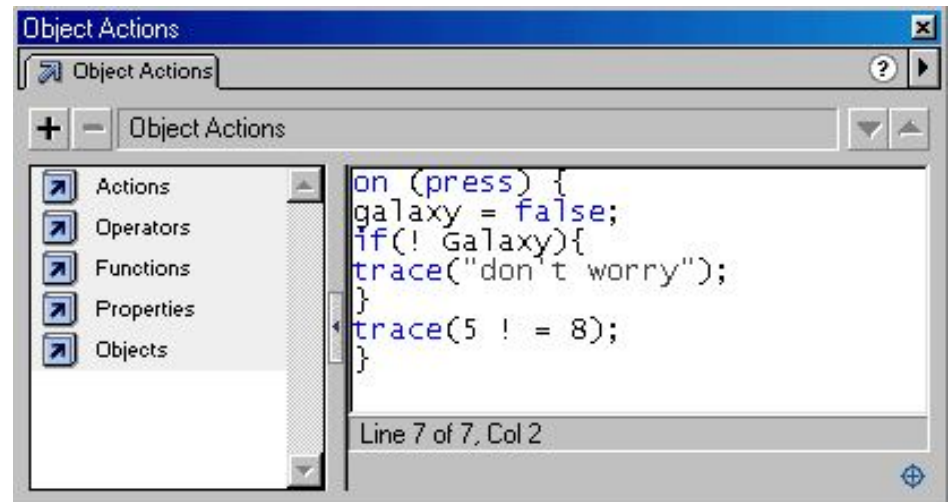
Syntax :

expression 1 != expression 2

Example :

5 != 8 returns true

5 != 5 returns false.



Using this operator, the opposite of equality operator is tested.

If expression 1 is not equal to expression 2 it returns true.

Syntax:

Expression 1 != expression 2

Example:

5 != 8 returns true

5 != 5 returns false.

Numeric Operators

ADD (+)

Using the operator, any two expressions can be added.

Syntax :

expr 1 + expr 2

Example :

x = 2;

y = 2;

z = x + y ;

The output window will show a value "4".

Subtract (-)

Using this operator, any kind of subtraction or negating is possible.

Syntax :

expr 1 - expr 2

Example :

```
on (press)
{
trace -(2+3 )
trace (2-3); trace (3.5-2.3);
}
```

The output window will show as -5, -1, 1.2

Multiplication (X)

Using this operator two numerical expressions can be multiplied. If both are integer, then result will be an integer. If both are floating, then result will be floating integer.

Syntax :

Expr 1 * expr 2

Example :

```
X = 2.0
Y = 3.0
Z = X * Y
Z = 6.0
```

X & Y are floating values and the result is also an floating value.

Division (/)

This operator is used to divide two expressions. The expressions and results are treated as floating numbers.

Syntax :

expr 1 / expr 2

Example :

```
x = 25/5
trace (x);
```

The output will show a value of 5.

String Operators

Using this operator's numbers and strings can be tested. They are similar to comparison operators. The operators of string are

le, lt, gt, ge, eq, ne and add.

Syntax:

expr 1 (operator) expr 2

Example:

On (press)

```
{
trace (4 gt 3);
trace (3 add 4);
trace (4 lt 3);
trace (4 eq 4);
trace (4 ne 3);
trace (4 le 3);
}
```

The output window shows the values as true, 34, false, true, true, false. Hence the string operators can be used with strings and numbers.

And (Logical)

Used to validate two conditions.

Syntax :

expression 1 and expression 2.

Example:

```
if (name == "Galaxy" and password == "layout")
{
go to and stop (5)
}
```

The and operator checks for true in both the condition i.e. only if the two conditions are satisfied, it will execute the rest of the statement.

In the above example, both name and password should be correct to the original value, then rest statements will be executed, otherwise the loop will not execute.

OR (Logical)

If any one condition is true, then the operator will return true.

Syntax:

expression 1 or expression 2

Example :

```
if (name == "Galaxy" or password == "layout")
{
go to and stop (5);
}
```

if any one condition is true, then the statement will be executed. i.e. in the above example, either name or password should be correct and rest of the statements will be executed.

Regards,

Manoj Kotak.

Director - Image Online Pvt. Ltd.

Developer of Layout Galaxy *Ready to use design concept for the web.*

All accompanying logos, brands and product names are trademarks of their respective companies.



Blood for humans comes only from humans : Donate Blood

<http://www.donate-blood.org> E-mail - om@donate-blood.org

What is Layout Galaxy ?

Layout Galaxy is ready to use design concepts in their source format. Different CDs in total contain 300 Photoshop.psd & 100 Flash fla layouts. Select the layout, make necessary changes to suit a customer's need & present demo to the client for approval. It's Select, Set & Go. It saves web designer's time & cost of production. It accelerates web designer's work-pace. It enhances productivity. It is continuous flow of inspiration & wealth of design ideas. Download Free 7 (Photoshop & flash) Layouts to believe it yourself.