**Interactive Development with JavaScript – Module 4**

**Operators**

Operators in JavaScript, as in other languages, are used to achieve arithmetic operations such as addition and subtraction, assign values such as giving a variable a value to hold, and compare values such as equal or not equal. Terms we should understand are “operand”, “unary operator”, and “binary operator”.

* Operand: Items that an operation is applied to. An example is 10 + 3. Here we have one operator (+) and two operands (10, 3), which are sometimes referred to as arguments.
* Binary Operator: Two operands with the operator. As we saw in the previous example the + operator took two operands, 10 and 3.
* Unary Operator: A unary operator has only one operand with the operator. A simple example might be ++var\_01 which will add one to the value of var\_01.

**Operators - Arithmetic**

Arithmetic Operator Specifics

* +
  + Addition
  + 5 + 6 equals 11
* -
  + Subtraction
  + 3 - 9 equals -6
* \*
  + Multiplication
  + 2 \* 4 equals 8
* /
  + Division
  + 4 / 2 equals 2
* %
  + Modulus (Remainder)
  + 10 % 3 equals 1
* \*\*
  + Exponentiation
  + 4 \*\* 2 equals 16
* ++
  + Increment
  + ++var is one greater than original value
* --
  + Decrement
  + --var is one less than original value

**Operators - Assignment**

* =
  + Assignment of the right operand value to the left operand.
  + var\_01 = 42
* +=
  + Adds (sums) the right operand and left operand, then assigns the new value to the left operand.
  + var\_01 += 3
* -=
  + Subtracts the right operand from the left operand, then assigns the new value to the left operand.
  + var\_01 -= 6
* \*=
  + Multiplies the right operand and left operand, then assigns the new value to the left operand.
  + var\_01 \*= 2
* /=
  + Divides the right operand with the left operand, then assigns the new value to the left operand.
  + var\_01 /= 2
* %=
  + Gets the remainder of the left operand divided by the right operand, and then assigns the new value to the left operand.
  + var\_01 %= 3
* \*\*=
  + Raises the first operand to the power of the second operand, and then assigns the new value to the left operand.
  + var\_01 \*\*= 3

**Operators - String**

* +
  + Adds (concatenates) the left String operand (stringVar\_01) and right String operand (stringVar\_02), then assigns the new String value to the left String operand (stringFinal).
  + stringFinal = stringVar\_01 + stringVar\_02
* +=
  + Adds the right String operand and left String operand, then assigns the new concatenates String to the left String operand.
  + stringVar\_01 += "String Literal"
  + stringVar\_01 += stringVar\_02

**Operators - Comparison**

For the following examples the var\_01 variable will have the value of 21.

* ==
  + Equals – value
  + var\_01 == 21 // true
  + var\_01 == 22 // false
* ===
  + Equals - value and type
  + var\_01 === 21 // true
  + var\_01 === "21" // false
* !=
  + Not equal – value
  + var\_01 != 21 // false
  + var\_01 != 22 // true
* !==
  + Not equal – value and type
  + var\_01 !== 21 // false
  + var\_01 !== "21" // true
* >
  + Greater than
  + var\_01 > 20 // true
  + var\_01 > 21 // false
  + var\_01 > 22 // false
* <
  + Less than
  + var\_01 < 20 // false
  + var\_01 < 21 // false
  + var\_01 < 22 // true
* >=
  + Greater than or equal
  + var\_01 >= 20 // true
  + var\_01 >= 21 // true
  + var\_01 >= 22 // false
* <=
  + Less than or equal
  + var\_01 <= 20 // false
  + var\_01 <= 21 // true
  + var\_01 <= 22 // true

**Operators - Logical**

* && (AND)
  + Represented with &&
  + Binary operand operator o Produces true if both operands are true
  + Produces false if either operand is false
  + (true && true) // true
  + (true && false) // false
  + (false && true) // false
  + (false && false) // false
* || (OR)
  + Represented with ||
  + Binary operand operator
  + Produces true if either operand is true
  + Produces false if both operands are false
  + (true && true) // true
  + (true && false) // true
  + (false && true) // true
  + (false && false) // false
* ! (NOT)
  + Represented with !
  + Unary operand operator
  + Produces true or false
  + Reverses the Boolean result of the operand
  + !true // equals false
  + !false // equals true
  + !(3 == 4) // true
  + !(4 == 4) // false
  + !(true && true) // false
  + !(false && false) // true

**Operators -  Precedence**

* Expressions with more than one operator will have an execution order which is based on the operator’s precedence. If two or more operators appear in a single statement with the same precedence, they will execute from left to right. Operators with differing precedence order will execute from the highest level to the lowest level. A common example is with addition and multiplication. Example such as 4 + 5 \* 3 has the same order as 5 \* 3 + 4 as the multiplication is always performed before the addition.
* Parentheses override precedence, so if the goal is to have the addition performed first you would need to enclose the two operands with the addition operator in parentheses, ( 4 + 5 ) \* 3.
* Precedence for some of the JavaScript operators, higher level to a lower level:
  + ( ), [ ]
  + !, ++, --, -
  + \*, /, %
  + +, -
  + <, <=, >, >=
  + ==, !=, ===, !==
  + =
  + +=, -=, \*=, /=, %=