***Java Script***

***JavaScript Variables & Data Types***

***Variables –***

* *In programming, a variable is a value that can be changed depending on the conditions or information being passed to the program.*
* *In java Script variables are case sensitive*

*Ex: var num = 6;*

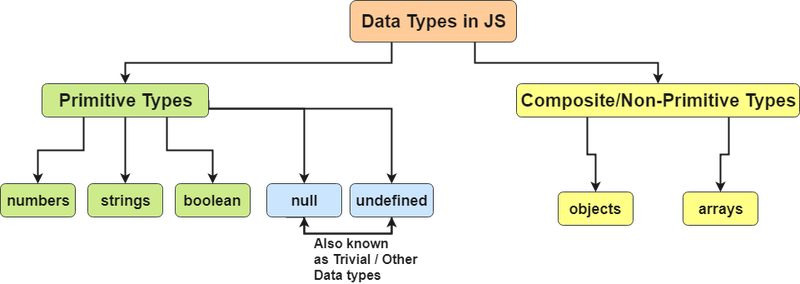
*var Num = 6;*

*Both are different variables*

***Data Types –***

*Datatypes are basically****type of data****that can be used and manipulated in a program. For e.g. numbers (1, 2, 5,5), string (“hello Simple Snippets”, “JS”), Boolean (true, false) etc.*

*Following are the different Data Types in JavaScript –*



*There are two types of data types in JavaScript. (as shown in the diagram above)*

* *Primitive data type*
* *Non-primitive (reference) data type*

***JavaScript Primitive Data Types –***

|  |  |
| --- | --- |
| ***Data Type*** | ***Description*** |
| ***String*** | *represents sequence of characters e.g. “hello”* |
| ***Number*** | *represents numeric values e.g. 100* |
| ***Boolean*** | *represents boolean value either false or true* |
| ***Undefined*** | *represents undefined value* |
| ***Null*** | *represents null i.e. no value at all* |

***JavaScript Strings***

*A string (or a text string) is a series of characters like “Simple Snippets”. Strings are written with quotes. You can use single or double quotes:*

|  |
| --- |
| var carName = "Mercedes";   // Using double quotes  var carName = 'BMW';   // Using single quotes |

|  |
| --- |
| var answer = "It's alright";             // Single quote inside double quotes  var answer = "He is called 'Mark'";    // Single quotes inside double quotes  var answer = 'He is called "Mark"';    // Double quotes inside single quotes |

***JavaScript Numbers***

*JavaScript has only one type of numbers. Numbers can be written with, or without decimals:*

|  |
| --- |
| var x1 = 34.00;     // Written with decimals  var x2 = 34;        // Written without decimals |

*Extra-large or extra small numbers can be written with scientific (exponential) notation:*

|  |
| --- |
| var y = 123e5;      // 12300000  var z = 123e-5;     // 0.00123 |

***JavaScript Non-Primitive/Composite Data Types –***

|  |  |
| --- | --- |
| ***Data Type*** | ***Description*** |
| *Object* | *represents instance through which we can access members* |
| *Array* | *represents group of similar values* |

*Creating Variable in JavaScript*

|  |
| --- |
| var length = 5;   // Number  var lastName = "Simple Snippets";   // String  var flag = true;   // boolean |

* *JavaScript is LOOSELY/WEAKLY TYPED programming language.*
* *JavaScript is DYNAMICALLY TYPED programming language.*

*1.* ***JavaScript is LOOSELY/WEAKLY TYPED:***

*JavaScript is known as untyped/loosely/weakly typed language. This means that we do not have to specify the data type in advance*

*Example –*

|  |
| --- |
| var x;  x = 5; |

*In the above example we can see that we did not have to specify any data type for the variable x in advance.*

***2. JavaScript is DYNAMICALLY TYPED:***

*JavaScript is known as dynamically typed language.*

*This means, that once a variable is created in javascript using the keyword var, we can store any type of value in this variable supported by JavaScript.*

*Example*

|  |
| --- |
| // creating variable to store a number  var num = 5;    // store string in the variable num  num = "Simple Snippets";  or  var a = 5;  var b = 6;  var c = “Sandeep”  var d = 7;  var e = 8  document.write(a+b+c+d+e)  output  13Sandeep78 |

*In this example above, you can see that the data type of the variable num changes from number to string as we pass string data. This it is flexible in nature.*

***Variable Scope in JavaScript***

*Scope of a variable is the part of the program from where the variable may directly be accessible.*

*In JavaScript, there are two types of scopes:*

* ***Global Scope****– Scope outside the outermost function (we will discuss functions in detail in other article) attached to Window.*
* ***Local Scope****– Inside the function being executed.*
* *Let’s look at the code below. We have a global variable defined in first line in global scope. Then we have a local variable defined inside the function fun().*

|  |
| --- |
| <script type = "text/javascript">  var globalVar = "This is a global variable";    function fun() {    var localVar = "This is a local variable";      console.log(globalVar);    console.log(localVar);  }  fun();  console.log(localVar);    < /script> |

***Operators in JavaScript Programming***

*JavaScript operators are symbols which are used to assign values, compare values, perform arithmetic operations, and more.*

* *The variables (operations) are called operands.*
* *The operation (to be performed between the two operands) is defined by an operator.*
* *JavaScript supports the following types of operators.*
* *Arithmetic Operators*
* *Comparison Operators*
* *Logical (or Relational) Operators*
* *Assignment Operators*
* *Conditional (or ternary) Operators*
* *String Operators*
* *Type Operators*
* *Bitwise Operators*

***JavaScript Arithmetic Operators***

|  |  |
| --- | --- |
| ***Operator*** | ***Description*** |
| *+* | *Addition* |
| *–* | *Subtraction* |
| *\** | *Multiplication* |
| */* | *Division* |
| *%* | *Modulus (Remainder)* |
| *++* | *Increment* |
| *—* | *Decrement* |

***Operator Precedence***

*Operator precedence describes the order in which operations are performed in an arithmetic expression.*

*Example –*

*As in traditional school mathematics, the multiplication is done first. Multiplication (\*) and division (/) have higher precedence than addition (+) and subtraction (-).*

|  |  |  |
| --- | --- | --- |
| ***Operator type*** | ***Associativity*** | ***Individual operators*** |
| *Grouping* | *n/a* | *( … )* |
| *Member Access* | *left-to-right* | *… . …* |
| *Computed Member Access* | *left-to-right* | *… [ … ]* |
| *new (with argument list)* | *n/a* | *new … ( … )* |
| *Function Call* | *left-to-right* | *… ( … )* |
| [*new*](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/new)*(without argument list)* | *right-to-left* | *new …* |
| *Postfix Increment* | *n/a* | *… ++* |
| *Postfix Decrement* | *… --* |
| *Logical NOT* | *right-to-left* | *! …* |
| *Bitwise NOT* | *~ …* |
| *Unary Plus* | *+ …* |
| *Unary Negation* | *- …* |
| *Prefix Increment* | *++ …* |
| *Prefix Decrement* | *-- …* |
| *typeof* | *typeof …* |
| *void* | *void …* |
| *delete* | *delete …* |
| *await* | *await …* |
| *Exponentiation* | *right-to-left* | *… \*\* …* |
| *Multiplication* | *left-to-right* | *… \* …* |
| *Division* | *… / …* |
| *Remainder* | *… % …* |
| *Addition* | *left-to-right* | *… + …* |
| *Subtraction* | *… - …* |
| *Bitwise Left Shift* | *left-to-right* | *… << …* |
| *Bitwise Right Shift* | *… >> …* |
| *Bitwise Unsigned Right Shift* | *… >>> …* |
| *Less Than* | *left-to-right* | *… < …* |
| *Less Than Or Equal* | *… <= …* |
| *Greater Than* | *… > …* |
| *Greater Than Or Equal* | *… >= …* |
| *in* | *… in …* |
| *instanceof* | *… instanceof …* |
| *Equality* | *left-to-right* | *… == …* |
| *Inequality* | *… != …* |
| *Strict Equality* | *… === …* |
| *Strict Inequality* | *… !== …* |
| *Bitwise AND* | *left-to-right* | *… & …* |
| *Bitwise XOR* | *left-to-right* | *… ^ …* |
| *Bitwise OR* | *left-to-right* | *… | …* |
| *Logical AND* | *left-to-right* | *… && …* |
| *Logical OR* | *left-to-right* | *… || …* |
| *Conditional* | *right-to-left* | *… ? … : …* |
| *Assignment* | *right-to-left* | *… = …* |
| *… += …* |
| *… -= …* |
| *… \*\*= …* |
| *… \*= …* |
| *… /= …* |
| *… %= …* |
| *… <<= …* |
| *… >>= …* |
| *… >>>= …* |
| *… &= …* |
| *… ^= …* |
| *… |= …* |
| *yield* | *right-to-left* | *yield …* |
| *yield\** | *yield\* …* |
| *Comma / Sequence* | *left-to-right* | *… , …* |

***JavaScript Comparison Operators***

*Comparison and Logical operators are used to test for true or false.  Comparison operators are used in logical statements to determine equality or difference between variables or values.*

*Given that x = 5, the table below explains the comparison operators:*

|  |  |  |  |
| --- | --- | --- | --- |
| ***Operator*** | ***Description*** | ***Comparing*** | ***Returns*** |
| *==* | *equal to* | *x == 8* | *false* |
| *x == 5* | *true* |
| *x == “5”* | *true* |
| *===* | *equal value and equal type* | *x === 5* | *true* |
| *x === “5”* | *false* |
| *!=* | *not equal* | *x != 8* | *true* |
| *!==* | *not equal value or not equal type* | *x !== 5* | *false* |
| *x !== “5”* | *true* |
| *x !== 8* | *true* |
| *>* | *greater than* | *x > 8* | *false* |
| *<* | *less than* | *x < 8* | *true* |
| *>=* | *greater than or equal to* | *x >= 8* | *false* |
| *<=* | *less than or equal to* | *x <= 8* | *true* |

***JavaScript Logical (or Relational) Operators***

*Comparison and Logical operators are used to test for true or false.   Logical operators are used to determine the logic between variables or values.*

*Given that x = 6 and y = 3, the table below explains the logical operators:*

|  |  |  |
| --- | --- | --- |
| ***Operator*** | ***Description*** | ***Example*** |
| *&&* | *and* | *(x < 10 && y > 1) is true* |
| *||* | *or* | *(x == 5 || y == 5) is false* |
| *!* | *not* | *!(x == y) is true* |

***JavaScript Assignment Operators –***

*Assignment operators assign values to JavaScript variables.*

|  |  |  |
| --- | --- | --- |
| ***Operator*** | ***Example*** | ***Same As*** |
| *=* | *x = y* | *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 = 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* |

*The \*\*= operator is an experimental part of the ECMAScript 2016 proposal (ES7). It is not stable across browsers. Do not use it.*

***Adding Strings and Numbers –***

*Adding two numbers, will return the sum, but adding a number and a string will return a string:*

*Example –*

|  |
| --- |
| *var x = 5 + 5;*  *var y = "5" + 5;*  *var z = "Hello" + 5;*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  ***Output***  *10*  *55*  *Hello5* |

***JavaScript Type Operators***

|  |  |
| --- | --- |
| ***Operator*** | ***Description*** |
| *typeof* | *Returns the type of a variable* |
| *instanceof* | *Returns true if an object is an instance of an object type* |

***JavaScript Bitwise Operators***

*Bit operators work on 32 bits numbers. Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Operator*** | ***Description*** | ***Example*** | ***Same as*** | ***Result*** | ***Decimal*** |
| *&* | *AND* | *5 & 1* | *0101 & 0001* | *0001* | *1* |
| *|* | *OR* | *5 | 1* | *0101 | 0001* | *0101* | *5* |
| *~* | *NOT* | *~ 5* | *~0101* | *1010* | *10* |
| *^* | *XOR* | *5 ^ 1* | *0101 ^ 0001* | *0100* | *4* |
| *<<* | *Zero fill left shift* | *5 << 1* | *0101 << 1* | *1010* | *10* |
| *>>* | *Signed right shift* | *5 >> 1* | *0101 >> 1* | *0010* | *2* |
| *>>>* | *Zero fill right shift* | *5 >>> 1* | *0101 >>> 1* | *0010* | *2* |

*Patterns*

|  |  |  |
| --- | --- | --- |
| *Creational* | *Structural* | *Behavioral* |
| *Constructor* | *Decorator* | *Command* |
| *Module* | *Facade* | *Mediator* |
| *Factory* | *Flyweight* | *Observer* |
| *Singleton* |  |  |