Anna Goncharova, Al Mazzoli, Haiqa Kamran, Ashley Park

CMSC 245

Deepak Kumar

October 24, 2020

Assignment 3: Javascript

A data type is considered a value type when it stores a data value within its own memory space, and a reference type when it stores the address of where the value is being stored. In JavaScript, the five primitive types (boolean, null, undefined, string, and number) can be passed by value, while there are three other data types: array, function, and object, which are passed by reference. Before 2015, to declare a JavaSCript variable, the keyword 'var' was commonly used. Moreover, all variables are identified by unique names which are known as identifiers. The general rules for these identifiers names are that they are case sensitive, they must begin with a letter (but can also begin with a $, and an underscore), they cannot use reserved words, and they can have letters, digits, dollar signs, and underscores, and. Moreover, an assignment statement is used to declare and define a variable, which simply says that something on the left has a value on the right. You can declare a variable first and define it later, and also, a variable can be redeclared without losing its value. When defining a variable, the assignment operator typically used in JavaScript is '=' (with other combinations such as +=, *=, etc.). These variables can contain numbers or strings where strings must be passed with double quotations. A typical assignment statement in JavaScript looks like this:

Eg. var cost = 10; //variable cost contains the value 10.

You can also declare multiple variables in one line:

var name = "Adam", age = 37, job = "Software Engineer";

After 2015, a new version of JavaScript allowed the user to use the keywords 'const' and 'let'. Once a variable is defined with 'const', its value cannot be reassigned, and a variable defined by 'let' has a restricted scope.

First it's important to note that variables in JavaScript are not directly associated with any particular value type (their type is inferred), and any variable can be assigned (and re-assigned) values of all types. You can find out the type of a variable by using the typeof operator. Those types can be divided into primitive and structural types. All primitive data types are immutable. Primitives include Boolean(that only has values true and false like in Java), the undefined type(simply a variable that wasn't assigned any value), Number, BigInt, String, and Symbol. The Number type can represent floating point values in the range between -(2^53 − 1) and 2^53 − 1. It also has values of Infinity, -Infinity and NaN(not a number). The + and - Infinity values are used when a programmer is trying to divide by zero for example. The BigInt type can represent integer values beyond the safe integer limits

forNumber. It is similar to the BigInteger class in Java. You can use operators: +, *, -, /,**, and % with both BigInts and Numbers.The String type is used to represent textual information and is stored as a set of 16-bit unsigned integer values. Symbols are the newest primitive in Javascript, they are unique and are typically used as identifiers for object properties.

The structural types include Objects and Functions. As they are aggregate data types, we will not be covering them in this presentation, but it's worth noting that in Javascript they are both considered first class values.

Interestingly, null is considered a primitive value in Javascript, however, because every object in Javascript is derived from null, "typeof null" returns an object.

The for loop is a control structure that performs an action repetitively for a specific number of iterations. There is a variable that acts as a kind of counter, which increments or decrements each time that the action in the body of the loop is executed. Each time the loop iterates, the base condition is checked to see whether or not it is true, and if it is true, the action in the body of the loop executes. The condition is usually dependent on the value of the counter variable - once the counter variable changes in such a way that the condition is no longer true, the loop terminates. In JavaScript there is also a variation called the for-in loop, which performs some action over all applicable properties of an object. There is also a for-of loop, which performs an action on every applicable variable of an object - this loop can often be used in place of the for-in loop. The while loop is a control structure that repeats the action in the body of the loop over and over until a certain condition becomes false. Before the action happens the first time, the condition is tested to see if it's true. If the value is false, the loop doesn't run at all. The do-while loop is basically the same as the for loop except that the statement in the body of the loop is executed before the expression is checked to be true. Therefore, a do-while loop will always happen at least once whether or not the condition is true, whereas a while loop will only happen if the condition is true. The if statement is a control structure based on conditions that result in boolean values which determine whether or not the set of statements in the if statement should be executed. The if-else statement executes the else statement if the if statement's condition was not true, while the if-else-if statement executes the else-if statement's condition (and based on the condition, the statements under the else-if statement) if the first if statement condition was not true. You can also nest if statements to check if another condition is true after checking on a previous condition. The switch statement is a control structure in Javascript that is used the same way as it is in Java. First, an expression is evaluated once, and then the result of the evaluation is compared to each of the cases. Then, when the result fulfills a case, the statements in that specific case are executed or, if it does not fulfill any case, then a default statement is executed. Each case has a break statement as well to jump out of the whole switch statement once a corresponding case has been found for the result. The break statement is a control structure that causes the program to jump out of a loop or a switch statement, while the continue statement moves on to the next iteration of the loop without evaluating the other statements in the loop that come after the continue statement. The labeled statement helps specify to which loop the break or continue statement applies to. For example, if there is an outer and inner loop, and if the break statement is based on a condition in the inner loop but the programmer would like the break statement to apply to the outer loop, then he would just need to identify the outerLoop by writing something like outerLoop: on the line right above the outerLoop and then writing the break statement as break outerLoop;

Citation

https://codeburst.io/explaining-value-vs-reference-in-javascript-647a975e12a0

https://www.w3schools.com/js/js_variables.asp

https://developer.mozilla.org/en-US/docs/Glossary/Primitive

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures

https://www.w3schools.com/js/js_switch.asp

https://www.javascriptinstitute.org/javascript-tutorial/control-structures/