Haiqa Kamran
Anna Goncharova
Al Mazzoli
Ashley Park
CMSC 245
Kumar
December 1st, 2020

Javascript Final Report

Javascript is a dynamic interpreted computer programming language that also has object-oriented capabilities. Its purpose is to allow interaction between the user and the client on web pages, and it is most commonly used alongside HTML and CSS. Javascript has five primitive types that are passed by value: boolean, null, undefined, string, and number. All primitive data types are immutable. The three other data types are array, function, and object, which are passed by reference. The popular JavaScript run-time environment is Node.js. JavaScript has control structures in the form of loops which are for, for-in, for-of, while, and do-while as well as conditional control structures such as if, if-else, if-else-if, switch, break, and labeled statements.

A subroutine is a way that programming languages handle control abstraction. When a subroutine returns a value, it's also called a function, and functions in JavaScript are treated as objects. Basically, when the program gets to the function definition, it removes that part of the code from the program and makes it into an object. Then the function only runs if it is specifically called, unlike the "main" parts of the program, which run automatically. Something that is cool about JavaScript in comparison to other programming languages when it comes to subroutines is that subroutine objects in JavaScript, when they are defined as objects, remember the variables that existed in the same scope environment as the subroutine definition. When the subroutine is executed, then, it will first look at local variables that are passed to it, but if it can't find something it will also draw on the variables that existed in the same scope as the subroutine definition at the moment that the definition was encountered and the subroutine object created. Whatever the function returns is returned to the location where the function was called.

For a function in JavaScript, arguments that are passed by value are primitive data types while the arguments passed by reference are object types (because object references are values). This means that for primitive type variables, changes inside the function do not affect the variable outside the function, however, for objects, changes inside the function are reflected on the object outside the function as well. Moreover, JavaScript parameters are not specified by data types in the function definition, and the maximum number of parameters in the function definition are two-hundred and fifty-five. Furthermore, while passing the arguments inside the function, one must make sure to pass them in the same order as the parameters defined by the function. While the function does not type check the arguments, it also does not check for the total number of arguments being passed. In this way, if there are less arguments passed than the defined parameters, then the default is undefined. However, the default value for parameters can also be set inside the function, and since the launch of ECMAScript 6, default values can be set inside the function definition as well. Moreover, if the total number of arguments passed is more than the defined parameters, then local variables are not generated for the extra arguments. To overcome this, one can make use of the argument object which is a built-in type, and it carries an array containing all the arguments that have been passed.

Runtime errors result in error objects being created and “thrown”. Like many other programming languages, JavaScript’s most common form of exception handling is a try-catch. A try-catch statement “tries” a block of code in which if an error is found, the next block of code under “catch{“ is executed to handle the error. A finally statement can also be attached to the bottom of a try-catch statement, and the block of code under it is executed whether or not an error occurs. JavaScript also utilizes a throw statement to handle user-defined errors which should be caught at some point later in the program. Related to exception handling, there is also a one error method which provides information on the error that was found and programmers are also able to make custom exception objects by extending the "Error" type.

Data abstraction is the programming process of creating a data type, usually a class, that hides the details of the data representation in order to make the data type easier to work with. As was mentioned in the last presentation, in Javascript variable type changes, depending on the value that is assigned to the variable, which makes Javascript a weakly-typed language. Hence, it does not have or need generics. It also does not have the classical built-in support for abstraction like OOP languages, but it has user-defined objects, and OOP functionalities can be achieved by inheritance and object composition(e.g. Linked List). Prototypes and closures can help us accomplish data abstraction.

Prototypes are a Javascript concept, and each object has a property, which holds a link to another object called its prototype. That prototype object has a prototype of its own, and so on until an object is reached with null as its prototype. By definition, null has no prototype, and acts as the final link in this prototype chain. Thus, the prototype chain is the mechanism of inheritance in Javascript. That said, while Javascript does not have interfaces, which enable data abstraction in Java, it is possible to "implement interfaces" in Javascript using prototypes. Another mechanism that allows data abstraction in Javascript uses closures. A closure is the combination of a function bundled together (enclosed) with references to its surrounding state (the lexical environment), giving access to an outer function's scope from an inner function. In JavaScript, closures are created every time a function is created, at function creation time. They are useful because they reduce the need for parameter passing , keeping information contained within one block of code and thus hiding details that are not needed to be revealed. In addition, similar to Java, Javascript has private and public object fields.

Works Cited

https://www.dummies.com/web-design-development/javascript/how-to-pass-and-use-arguments-to-code-with-javascript/
https://medium.com/@csg.riskgame/javascript-abstraction-data-types-and-expressions-b52767bb529e
https://study.com/academy/lesson/data-abstraction-definition-example.html
https://developer.mozilla.org/en-US/
https://www.youtube.com/watch?v=CQqwU2Ixu-U
Forget Everything You Know About Functions: JavaScript Subroutines | SitePen
Microsoft PowerPoint - 6-subroutines (unc.edu)