Julia Lieberman

JavaScript Features, Project part 2 parts a-d

CSCI169 Programming Languages, Spring 2020

JavaScript uses a property called hoisting. This means that by default, declarations (not initializations) of functions and variables are essentially moved to the top. This means that in your code you can reference a variable or function even if they haven't been declared yet. However, this is not the case with variables and constants declared with they keywords *let* and *const*. And to keep good coding practices, it is best to declare your variables at the beginning of your program anyways. Note that JavaScript should be encoded as UTF-16. It also only has floating point numbers, no integers, floats, fixed-points, etc.

Javascript does not require a main function. Functions are function objects, and can be declared in many different ways. If you don't pass in the exact parameters, the missing ones will by default be "undefined." One way to declare a function is using the 'function' keyword:

```
function foo(){...}
```

An important syntactical difference between C++ and JavaScript is that JavaScript uses arrow functions. These are like lambda functions, and allow our functions to have shorter syntax. However, note that they must be defined before they are used. The above function would instead look like:

```
foo = () => {...}
```

You can also write a function that will run without being called with the following syntax:

```
(function () {...})();
```

Loops are very similar to those in C++, however instead of specifying for(int i….) you don't name the type, you just refer to your variable as a 'var' so it would look like

```
for(var i=0; i<10; i++){...}
```

I/O is done very differently than in C/C++. Because JavaScript is made for the browser you usually won't log anything to the console, but you can send an alert to the user and if necessary use the console.log() function. You can use a prompt to receive user input such as

```
var userInput = prompt("your message here")
```

You can also read and write data in a text file using writeFile(path, data, callback). One widely used feature in JavaScript is the concept of a callback function. A callback function is used after another function has finished executing, so in the above example, the callback function which is passed in as a parameter will be executed after writeFile finishes.

JavaScript is an interpreted language, not compiled, and is single-threaded but can use a callback queue. JavaScript uses something called an Engine which functions as an interpreter to execute the code. V8 is a popular one that was written in C++ by Google. The engine has a call stack and a memory heap, and can be implemented as a typical interpreter or JIT: just-in-time, which compiles JavaScript to bytecode in some form.

JavaScipt uses dynamic typing, and allows for duck typing. It is considered a weakly typed language. Instead of type coercion, you can check if two values are equivalent either with == or with ===. The double == will automatically convert the first object to its equivalent value in the type of operand 2. Using the triple === will not allow any conversion. It will return false automatically if the two operands are of different types. It is interpreted, thus types are checked at runtime since there is no compile time. You can add different types of objects together, unlike in Python. Python would throw an error if you tried to run "hi"+10 but in JavaScript this would return "hi10". Return types must either always be the same type, or there should be no return. For example, you can't have an if/else statement like

```
if(a)
  return true
else
  return
```

Instead, both returns would either need to not return anything, or return booleans.

Functions introduce new scope in JavaScript, just like in Python. There are global and local scopes. Global variables (variables not declared within a function) are accessible to all the code. Local variables, declared within a function, are only accessible from within the function and any nested functions. To declare a global variable within a function, you must use the window object. You can access an existing global variable or declare a new one in the same way:

```
var x;
```

```
function foo()
{
    window.x=5;
}
```

All global variables declared outside of functions are automatically added to the window object.

JavaScript has no concept of mutable and immutable.