## Explain differences between Java and JavaScript

Java is a compiled language whereas JavaScript is a scripted language. This means that all Java code has to go through a compiler before it can be run. This checks for all checked exceptions that would otherwise be thrown during runtime in a Javascript application, all unchecked exceptions works the same though. This also makes it easier to write the actual code you want to write in Java, as if you misspell one of your variables it can’t be used later, whereas JavaScript just creates a new variable on the global scope.

Java code is written through an Intergraded Development Environment (IDE), which turns the code into bytecode. This bytecode is not readable by the human eye and needs a Java Virtual Machine (JVM) to run it. JavaScript is executed in the same language that it’s written by a JavaScript engine. The compiler turns Java code into bytecode and then a JVM turns the bytecode into the native code of your computer.

There is some similarities between Java and JavaScript but in general, they are as different as apples and cats. Both are Object Oriented Languages, and both needs an interpreter to run. Java uses a JVM, and JavaScript uses a browsers JavaScript engine.

Code completion is a lot better in Java.

JavaScript is more loosely typed than Java. No type checking as an example.

## Explain the following features in JavaScript

### Function Closures and the JavaScript Module Pattern

A function Closure is an inner function that has access to the outer functions variables scope chain. A closure has three scope chains:

* Its own scope
* The outer functions scope
* Global scope

A closure does not only have access to the variables in an outer function but also to the parameters set in the outer function.

A closure has access to the outer functions variables and parameter even after the outer function returns

### JavaScript Prototyping

Prototyping is JavaScripts way of handling inheritance. As mentioned earlier JavaScript is an Object Oriented Scripting language, which means that we work with objects in JavaScript. Every Object in JavaScript has a prototype by default as prototypes themselves are objects. The only exception to this is when we get to the top of a prototype chain.

Prototype is a help when multible instances of an object share a common prototype. Just like with Java Classes properties of the prototype object are defined only one place, but inherited by all instances. This makes code a lot easier to maintain.

The simplest way to make a prototype is by making a constructer function. You can add .prototype property to a constructer which will be assigned as the prototype to all instances created when the constructer function is called.

When searching for a variable JavaScript will work its way up the prototype chain to find a match and if none is found it will return undefined.

### User defined Callback Functions

In JavaScript functions are first-class objects, so they can be used in a first-class manner just like any other object (String, Number, etc.) as they are objects themselves. This means that functions can be stored in variables, passed as arguments, created within other functions and be returned from functions.

Passing a function as a parameter into another function is the essence of a callback function. When a callback has been parsed as a parameter, it can then be executed inside the other function. The callback we parse is not executed immediately but is instead “called back” at some specified point inside the containing functions body.

It’s possible both to use named or anonymous functions as callbacks. A lot of the time an anonymous functions is going to be fine, but if we end up in a chain of callbacks, it’s often a good idea to name the functions and then parse them into the function as a callback, to make reading of the code easier.

### Explain generally about node.js and NPM

### Provide examples of user defined reusable modules implemented in Node.js