The **JavaScript** language has three components: core, client-side and server-side.

* **Core JavaScript** is the base part of the JavaScript language that is supported on both the client and server side.
* [Client-side JavaScript (CSJS)](http://help.dottoro.com/ljswgnnf.php) contains the core JavaScript elements and has several additional objects, properties and methods that are supported by browsers.
* **Server-side JavaScript (SSJS)** also contains the core JavaScript elements and it has additional features supported by server-side products.

**JavaScript** is a single threaded, non-blocking, asynchronous, concurrent language. **A**lso it is a high-level, garbage-collected, prototype-based, multi-paradigm, dynamic language.

1. **High-level:**
   * Level (vs. low-level) refers to the level of abstraction from the computer's type. While the code is more programmer-friendly to read and write, that code will need to be transformed into machine-readable code.
2. **Garbage-collected:**
   * JavaScript has a background process to free up memory based on reachability. This is based on whether an object is referenced by or tied to any other object
3. **Prototype-based:**
   * JavaScript is prototype-based, (rather than class-based). All JavaScript objects have a prototype that they can inherit methods from. So not only does your object you create have access to the methods you explicitly define, it will also have access to the properties of its prototype going back to the Object. Prototype.
4. **Multi-paradigm:**
   * While JavaScript is an object-oriented language in the sense it's based on objects with properties and methods, JavaScript supports both imperative and functional programming paradigms.
5. **Dynamic:**
   * Dynamic refers to typing (e.g. number, string, boolean). Instead of specifying the type when we write the code and checking type at compilation, JavaScript does the type checking at runtime.

Classes:-

[class JSContext](https://developer.apple.com/documentation/javascriptcore/jscontext)

A JSContext object represents a JavaScript execution environment. You create and use JavaScript contexts to evaluate JavaScript scripts from Objective-C or Swift code, to access values defined in or calculated in JavaScript, and to make native objects, methods, or functions accessible to JavaScript.

[class JSManagedValue](https://developer.apple.com/documentation/javascriptcore/jsmanagedvalue)

A JSManagedValue object wraps a [JSValue](https://developer.apple.com/documentation/javascriptcore/jsvalue) object, adding “conditional retain” behavior to provide automatic memory management of values. The primary use case for a managed value is to store a JavaScript value in an Objective-C or Swift object that is itself exported to JavaScript.

[class JSValue](https://developer.apple.com/documentation/javascriptcore/jsvalue)

A JSValue instance is a reference to a JavaScript value. You use the JSValue class to convert basic values (such as numbers and strings) between JavaScript and Objective-C or Swift representations in order to pass data between native code and JavaScript code. You can also use this class to create JavaScript objects that wrap native objects of custom classes or JavaScript functions whose implementations are provided by native methods or blocks.

[class JSVirtualMachine](https://developer.apple.com/documentation/javascriptcore/jsvirtualmachine)

A JSVirtualMachine instance represents a self-contained environment for JavaScript execution. We use this class for two main purposes: to support concurrent JavaScript execution, and to manage memory for objects bridged between JavaScript and Objective-C or Swift.

Protocols:-

[protocol JSExport](https://developer.apple.com/documentation/javascriptcore/jsexport)

The protocol we implement to export Objective-C classes and their instance methods, class methods, and properties to JavaScript code.

### Legacy JavaScriptCore APIs:-

[JSBase.h](https://developer.apple.com/documentation/javascriptcore/jsbase_h)- Defines the JavaScriptCore interface engine.

[JSContextRef.h](https://developer.apple.com/documentation/javascriptcore/jscontextref_h)

[JSObjectRef.h](https://developer.apple.com/documentation/javascriptcore/jsobjectref_h)- A JavaScript Object.

[JSStringRef.h](https://developer.apple.com/documentation/javascriptcore/jsstringref_h) - A UTF16 character buffer that is the fundamental string representation in JavaScript.

[JSStringRefCF.h](https://developer.apple.com/documentation/javascriptcore/jsstringrefcf_h) - Contains CFString convenience methods.

[JSValueRef.h](https://developer.apple.com/documentation/javascriptcore/jsvalueref_h)- A JavaScript value that is the base type for all JavaScript values and the polymorphic functions on them.

### Structures:-

[struct JSTypedArrayType](https://developer.apple.com/documentation/javascriptcore/jstypedarraytype)

A constant identifying the Typed Array type of a JSObjectRef.

Objects

* [ActiveXObject](http://help.dottoro.com/ljiujjib.php)
* [arguments](http://help.dottoro.com/ljbeppcp.php)
* [Array](http://help.dottoro.com/ljnkadsn.php)
* [Boolean](http://help.dottoro.com/ljkrbhhs.php)
* [Debug](http://help.dottoro.com/ljjctxwj.php)
* [Date](http://help.dottoro.com/ljlecttk.php)
* [Enumerator](http://help.dottoro.com/ljpppcjg.php)
* [Error](http://help.dottoro.com/ljfhismo.php)
* [Function](http://help.dottoro.com/ljgecwtw.php)
* [Math](http://help.dottoro.com/ljoupdvs.php)
* [Number](http://help.dottoro.com/ljvcpdus.php)
* [Object](http://help.dottoro.com/ljufnnvc.php)
* [RegExp](http://help.dottoro.com/ljarccdp.php)
* [String](http://help.dottoro.com/ljetnrhv.php)
* [VBArray](http://help.dottoro.com/ljfnfcop.php)

## Difference between Java vs JavaScript

### 1) Execution Environment

First difference between Java and JavaScript is that Java is compiled + interpreted language, Java code is fist compiled into [class files](http://javarevisited.blogspot.sg/2012/05/10-points-about-class-file-in-java.html) containing byte code and than executed by JVM, on the other hand JavaScript code is directly executed by browser. One more difference which comes form this fact is that, Java is run inside JVM and needs [JDK or JRE](http://java67.blogspot.sg/2013/02/difference-between-jdk-and-jre-in-java.html) for running, on there other hand JavaScript runs inside browser and almost every modern browser supports JavaScript.

### 2) Static vs Dynamic Typed language

Another key difference between JavaScript and Java is that, JavaScript is a dynamic typed language, while Java is a statically typed language. Which means, variables are declared with type at compile time, and can only accept values permitted for that type, other hand variables are declared using vary keyword in JavaScript, and can accept different kinds of value e.g. String, numeric and boolean etc. When one variable or value is compared to other using == operator, JavaScript performs type coercion. Though it also provides === operator to perform strict equality check, which checks for type as well.

### 3) Support of Closures

JavaScript supports closures, in form of anonymous function. In simple words, you can pass a function as an argument to another function. Java doesn't treat method as first class citizen and only way to simulate closure is by using anonymous class. By the  way Java 8 has brought real closure support in Java in form of [lambda expression](http://javarevisited.blogspot.sg/2014/02/10-example-of-lambda-expressions-in-java8.html) and this has made things much easier. It's very easy to write expressive code without much clutter in Java 8.

### 4) OOP

Java is an Object Oriented Programming language, and though JavaScript also supports [class and object](http://javarevisited.blogspot.sg/2012/12/difference-between-class-and-object-in-oops-java.html), it's more like an object oriented  scripting language. It's much easier to structure code of large enterprise application in Java then JavaScript. Java provides packages to group related class together, provides much better deployment control using JAR, WAR and EAR as well.

### 5) Right Once Run Anywhere

Java uses byte code to achieve [platform independence](http://java67.blogspot.sg/2012/08/how-java-achieves-platform-independence.html), JavaScript directly runs on browser, but code written in JavaScript is subject to browser compatibility issue i.e. certain code which work in Mozilla Firefox, may not work in Internet Explorer 7 or 8. This is because of browse based implementation of JavaScript. This was really bad until jQuery comes. Its a JavaScript library which helps to free web developers from this browser compatibility issues. This is why it is prefer to write code using jQuery rather than using plain old JavaScript code, even if its as simple as calling getElementById() or getElementByName() methods to retrieve DOM elements.

### 7) Block vs Function based Scoping

Java mainly uses block based scoping i.e. a variable goes out of scope as soon as control comes out of the block, unless until its not a [instance or class variable](http://javarevisited.blogspot.sg/2012/02/difference-between-instance-class-and.html). On the other hand JavaScript mainly uses function based scoping, a variable is accessible in the function they are declared. If you have a global variable and local variable with same name, local will take precedence in JavaScript.

### 8) Constructors

Java has concept of constructors, which has some special properties e.g. [constructor chaining](http://javarevisited.blogspot.sg/2012/12/constructor-chaining-in-java-calling-another-constructor.html) and ensuring that super class constructor runs before sub class, on the other hand JavaScript constructors are just another function. There is no special rules for constructors in JavaScript e.g. they cannot have return type or their name must be same as class.

### 9) NullPointerException

JavaScript is much more forgiving than Java, you don't have [NullPointerException](http://javarevisited.blogspot.sg/2012/06/common-cause-of-javalangnullpointerexce.html) in JavaScript, your variable can accept different kinds of data because of JavaScript is dynamically typed language.

### 10) Applicability

Last but not the least, JavaScript has it's own space, sitting cozy along with HTML and CSS in Web development, while Java is everywhere. Though both has good number of open source libraries to kick start development, but jQuery has certainly brings JavaScript on fore front.