JavaScript is one of the most hated programming language amongst the developer community. There is infinite number of articles on the internet which highlights bad parts of the language. But the fact is that it’s still the most powerful language, thanks to the web. Though primarily introduced as client side programming language, JavaScript found its footprint on the server side as well with framework like node.js.

If you’re not familiar with how JavaScript differs from programming languages like C++, Java and C#, I recommend you to read my article on Top 10 JavaScript traps for C# programmer.

Functions are at the core of language and there are different ways [Function declaration, Function Expression, Anonymous Functions etc.] to define it in your JavaScript program. However each of these options has some pros and cons associated with it. Thus, a series of short posts explaining JavaScript functions.

Let’s start with Function Declaration.

**Function Declaration**

Also known as **Function Statement**; is the oldest way of defining a function in JavaScript. It follows similar function declaration syntax as other programming languages like C, C++, C#, Java etc.

Code listing shown below defines a logMessage function, which logs the value of input parameter message to console.

function logMessage(message){

console.log(message);

}

logMessage("JavaScript is my favorite language ");

In the function declaration syntax, you must provide function name [logMessage in this case] during its declaration. If you skip function name, it becomes an **anonymous function**, which will cover later in this series.

**Trap 1 – Hoisting**

In traditional programming languages like C# or Java, you need to define a function first before you can call it. JavaScript however works differently. Let’s update our code listing and call the logMessage function before its declaration.

logMessage("JavaScript is my favorite language ");

function logMessage(message){

console.log(message);

}

If you execute the code, program will not throw any exception and will simply log the message to the console. Now, how is that even possible? Enter function hoisting.

Function hoisting is the process in which JavaScript runtime hoists all the functions declared using function declaration syntax at the top of JavaScript file. So internally, even though you have called logMessage function before its declaration, JavaScript engine internally hoisted it at the top, so call to logMessage function gets executed without any error.

**Trap 2 – Readability**

Function declaration syntax and dynamic nature of JavaScript language can sometimes impacts program readability. Again, let the code do the talking.

function print(input) {

console.log("Print was called with value " + input);

}

print(10);

function print(input) {

console.log("Redefining print with value " + input);

}

print(20);

If you read above program, you would think that it will log messages as shown below



Unfortunately that is not the case, and again it’s related with function hoisting trap we discussed earlier. All the functions declared using function declaration syntax gets hoisted, so second print function [which contains Redefining statement] overwrites first print function, so when you call print function with values 10 and 20 respectively, it will always get the overridden function and display the output as shown below.



So keep in mind, how you read and think is not what you always get in JavaScript ☺. Declaring functions using Function Expression syntax helps us to resolve this issue. We will cover Function Expression in next article.

Thanks.