Welcome to JavaScript Function series. This is the third article in JavaScript function series focusing on **Anonymous Functions**. Follow the links mentioned below for earlier articles in the series.

JavaScript Function Series Part 1 – Function Declaration

JavaScript Function Series Part 2 – Function Expression

**Anonymous Functions**

As the name suggests anonymous functions are functions without any given name. They are declared similar as function expression syntax, except that the function is declared without any name. In either case, once you assign a function to a variable [log as shown in below example], you have to use the assigned variable to call the function.

var log = function (message){

console.log(message);

}

log("This is a function expression");

Function hoisting rules for function expression and anonymous functions are same. That means; JavaScript engine hoist the function variable instead of complete function definition. So JavaScript runtime re-implements the code as shown in below code snippet

var log;

log = function(message){

console.log(message);

}

log("This is a function expression");

Note again, it doesn’t hoist function definition completely. It just hoists the assigned variable and initializes it where function was initially defined. So in this case, you cannot call the function unless it is explicitly defined earlier. Doing so will result into a runtime exception undefined is not a function since we are trying to call a function which has not been defined.

var log;

log("JavaScript is my favorite language"); // throws exception

log = function (message){

console.log(message);

}

**Anonymous Functions In Object Literal**

// in an object

var person = {

sayHello: function(){

console.log("hello");

}

};

person.sayHello();

**Anonymous Functions In jQuery**

// event handler

$("p").click = function(){

alert("hello");

};

**Anonymous Functions as callback functions**

Anonymous functions are commonly used as callback functions. In below code example, we have defined an object literal *Contest* which contains a function *askQuestion* which takes function as argument and calls it during its execution. During *askQuestion* function execution, we can simply pass an anonymous function as an argument, which returns the answer to the console.

var Contest = {

ans: "Red",

askQuestion: function (answered) {

console.log("Your fav color :" + this.ans);

answered(this.ans);

}

}

Contest.askQuestion(function (answer) {

console.log("Answer is " + answer);

});

**No Readability Traps**

Let’s take a look at readability trap we discussed in function declaration example in earlier post and redefine *print* functions using anonymous function syntax.

var print = function (input) {

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

}

print(10);

var print = function (input) {

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

}

print(20);

In this case, after calling *print* function *print(10)*, we are overriding *print* function definition, hence both the calls to *print* function will display output as shown below.

C:\$\Code - Git\Blog\2014\08\JavaScript Functions Part 1 - Function Declaration\assets\Output1.PNG

**Limitations**