**Understanding Closures in JavaScript**

(Bronx Coffee and Code, 6 April 2017)

Kem Crimmins

email: kem.crimmins@gmail.com

twitter: @kemcrimmins

github: https://github.com/kemcrimmins

**Quotations from smart people**

“Like objects, closures are a mechanism for containing state. In JavaScript, a closure is created whenever a function accesses a variable defined outside the immediate function scope. It’s easy to create closures: Simply define a function inside another function, and expose the inner function, either by returning it, or passing it into another function. **The variables used by the inner function will be available to it, even after the outer function has finished running.**” (Eric Elliott, “The Two Pillars of JavaScript—Pt 2: Functional Programming”, emphasis added)

“A closure is the bundling of a function with its lexical environment. Closures are created at function creation time. When a function is defined inside another function, it has access to the variable bindings in the outer function, **even after the outer function exits**.” (Eric Elliott, “Why Learn Functional Programming in JavaScript? (Composing Software, emphasis added)

“Closure is when a function is able to remember and access its lexical scope even when that function is executing outside its lexical scope.” (Kyle Simpson, *Scope & Closures* [*You Don’t Know JS*])

**copy and paste (or type) the following function into console**

function outer (a) {

console.log("outer executes" + a);

function inner(b) {

console.log("inner executes with " +a+ ", " +b);

}

return inner;

}

**proof of concept: type in console after copying above to console**

var test = new outer(1);

test(2);

**Analysis**

function outer (a) {

**// the parameter a is a local variable for function outer**

console.log("outer executes" + a);

function inner(b) {

**// the parameter b is a local variable for function inner, which includes**

**// variable a (from function outer) as part of its scope**

console.log("inner executes with " +a+ ", " +b);

}

return inner; **// function outer exits with this return statement**

}

var test = new outer(1);

**// what is the value of test? Its value is what is returned by outer(1):**

var a = 1;  
function inner(b) {

console.log(“inner executes with “ +a+ “, “ +b);

}

test(2);

// **what is executed when test(2) is called? The following code is executed:**

var a = 1;

function inner(b) {

// var b = 1;

console.log(“inner executes with “ +a+ “, “ +b);

}