<https://2ality.com/2011/09/currying-vs-part-eval.html>

Currying vs. Partial Application

Currying – taking each argument that would be passed to a single function and turning it into function inside a function inside a function, etc.

Connect is one of these, closures are this

Takes a binary function and returns a unary function that returns a unary function

Partial application is similar

Takes a function and a fixed value for the first argument to produce a new function

It fills in one of the parameters – creates binding to a particular value for the remainder of the function

So where we would have a curried function look like:

function (x) {

return function(y) {

return x+y;

}

}

We would have a partial application function analogous to this curried function look like:

function(y) {

return 5 + y;

}

Where the first argument of function is bound to 5 – we are simply filling in the first argument of the function

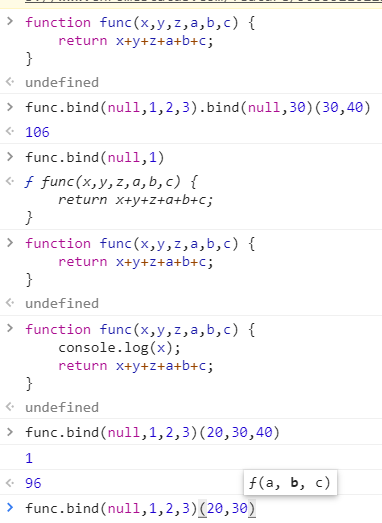
The .bind method is an application of partial application

function func(x,y,z,a,b,c) {

return x+y+z+a+b+c;

}

func.bind(null,1,2,3) will have x, y and z coming from the bind function => this binding is called partial application



As we can see, this changes the function signature of the resulting function

Now only a, b and c are parameters, and x, y and z are preset with the values from bind

As we can see from the console.log of x, x is given the value of 1

This is partial application

This is very different from currying

Currying always produces nested unary (1-ary) functions. The transformed function is still largely the same as the original.

Partial application produces functions of arbitrary arity. The transformed function is different from the original – it needs less arguments.

Interestingly, with a curried function and curried invocation, it is easy to achieve the same effect as binding one argument (performing this operation several times yields the general case): To bind the first argument to a value, you simply apply the outer-most of the nested functions to that value, which returns the desired new function.