Closure – What Is It and How Does Haskell Utilize It

# Fair warning

if you don’t like JavaScript, you’re out of luck. The examples in this article are in Haskell and JavaScript as we did not employ statically nested functions in our previous courses with C and Java. I am not sorry Matt.

# Review on scoping

Thinking back to System Software for those who have taken it at this point, we can recall that (for the example PM0 code) to access a variable inside a statically nested function the variable being accessed must be in the immediately availiable function scope or must be a member of a static parent function. In Laymen’s terms, you cannot access a variable which is not in your currently executing function unless it was mentioned in one of parent functions up the stack.

Let’s take for example the following JavaScript code:

**Here, we notice the following:**

// Function A is the static parent of B

function A() {

// Some members of A

int a, b, c;

// This is statically nested within the function A

function B() {

// Some members of B

int d, e;

// Be can access members of A

a = c + d;

function D() { }

}

// B is invoked within A

B();

}­

* **A** is a static parent of **B**
* **A**’s members are: **a**, **b**, **c**
* **B** has access to **a**, **b**, **c**, **d**, **e**
* **A** does not have access to **d**, **e**
* **B** cannot be invoked outside the scope of **A**
* If a function **C** existed within **A**, but not within **B**, it would have access to **B**, as well as **a**, **b**, **c**; albeit not **d**, **e**.
* **A** cannot access **D**, but **B** can access **D**
* **D** can access **a**, **b**, **c**

## What does scoping look like in Haskell?

If you haven’t already taken a look over my article about variable scope in Haskell, please take a moment to read it. It’s very short and helps give a little bit of background so you’ll better understand closures.

<insert link to scope article>

This concludes my references to the basics of statically nested functions. From here on out I will be talking about closures. To understand closure, you need to be able to understand statically-nested functions. If you are having trouble understanding what is going on, then I'd recommend this as a reference: <https://en.wikipedia.org/wiki/Nested_function>

# What is closure?

In programming terms, closure is a means of passing around a function as a first-class citizen – in other words, languages which support this feature allows functions to be passed as arguments.

## JavaScript

In JavaScript, this creates a reference to a function object, and the reference can then be passed around as a variable. This object is called a closure. It is a an actual object which The function For example:

const a = function A() {

return 1;

}

// Print out the function reference  
console.log(a);

// prints: function A() {

// return 1;

// }

// Print out the function value

// Notice the use of the parenthesis “()”  
console.log(a());

// prints: 1

## Haskell

## What does that mean?

Being that functions can be passed around

## How does this work?

When a function is statically nested within another function, a closer is created. Lets take a look at an example of

// Function A is the static parent of B

function A() {

// Some members of A

int a, b, c;

// This is statically nested within the function A

function B() {

// Some members of B

int d, e;

// Be can access members of A

a = c + d;

}

// B is invoked within A

B();

}­

Notice how function **B** is statically nested within **A**. In JavaScript, this creates a closure for the function **A**. A closure is an enclosed structure which has reference to the function which contains statically-nested child functions. This reference knows about all of the members of A, and

## When are closures created?

# Why is it useful?

## Lazy Evaluation

The concept of lazy evaluation pretty much covers why closure is important. Lazy evaluation is enabled by the ability to pass around a function by reference. This way, we can talk about the function without executing it. Let’s talk about constructing the set integers from Simply put, there no way to compute this and pass the result of such a computation around in a program. Closures allow us to pass the function by reference and evaluate the result later.

# References

<https://en.wikipedia.org/wiki/Closure_(computer_programming)>  
- 2/4/2017 | <http://stackoverflow.com/questions/36636/what-is-a-closure>  
- 2/4/2017 | <http://stackoverflow.com/questions/111102/how-do-javascript-closures-work>  
<https://en.wikipedia.org/wiki/Free_variables_and_bound_variables>

# Ideas

* Free Variables – variables which are not lexically scoped to the current context.