TECHNICAL BLOG – “Powerful Ruby Features: Blocks, Procs and Lambdas”

Introduction

Blocks, procs and lambdas are referred to “closures” in computer science As the title of this blog suggests, these particular features, or closures, are very powerful but, also, very misunderstood and many developers aren’t fully aware of the subtle difference between them. The gist of what these closures do is they allow a developer to pass code to a method and then execute that code later on.

Blocks

Starting first with blocks, this is the type of code which is implicitly passed to a method using “do…end” or “{…}”. The general practice is to use the curly brackets for one line blocks and the do/end syntax for multiple line blocks. Here are some examples:

An important and some might say magical, component of a black is the use of the key word “yield” which defers the calling methods execution to allow the block to be evaluated. If there is a result produced by the block, which there may not be, this result is evaluated by any remaining code in the method. Something important to note is that yield can take parameters which are passed and evaluated within the block. [NEED A BETTER EXPLANATION OF EXACTLY WHAT YIELD DOES AND WHEN TO USE IT]

\*\*\*lamen explanation: Basically, you can use yield almost as a place holder and then input some block of code at a later time. [use example from first link below] So you can set up a loop of some kind but not decide on what the code will specifically do but when you actually pass that new instance method to an object, you can add any type of block code (following the same components) as you’d like. \*\*\*\*

RESOURCES:

<http://www.reactive.io/tips/2008/12/21/understanding-ruby-blocks-procs-and-lambdas/>

<http://rubymonk.com/learning/books/4-ruby-primer-ascent/chapters/18-blocks/lessons/64-blocks-procs-lambdas>

<http://code.tutsplus.com/tutorials/ruby-on-rails-study-guide-blocks-procs-and-lambdas--net-29811>