**Functions Invocation, Briefly**

**Ruby**

In Ruby, 'ruby string'.upcase is just syntactic sugar for 'ruby string'.upcase(). Both of these examples will call, or invoke, the upcase method on 'ruby string' and return "RUBY STRING".

**JavaScript**

In JavaScript, 'javascript string'.toUpperCase (without parentheses) **is not the same** as 'javascript string'.toUpperCase() (with parentheses). The distinction is very important.

To illustrate:

// accessing a property that is a function:

'javascript string'.toUpperCase // => function toUpperCase() { [native code] }

// invoking the accessed function:

'javascript string'.toUpperCase() // => "JAVASCRIPT STRING"

**Retrieving a property**

In JavaScript, when we say someObject.someMethod without parentheses, we are just retrieving the property stored on someObject (or an object higher up in someObject's inheritance chain) at the key someMethod. It's the same as saying someObject['someMethod']. The function object that is found at the key someMethod will be returned to us un-invoked.

The ability to retrieve a function object in this way can be quite useful. We will often grab a reference to an object's method in order to pass that method as a callback to another function, to be executed at a later time.

**Invoking a function**

If we want to actually invoke (i.e. execute) a function or method in JavaScript, we must signal that by adding parentheses. When we write 'javascript string'.toUpperCase(), the function is retrieved as discussed above, and, in addition, the () cause the retrieved function to actually be executed, returning "JAVASCRIPT STRING"

Í