Programming with Closures

A MUUG Presentation

About Trevor Cordes

- UNIX-head since 1992 (SunOS > AIX > RH > Fedora)
- Fedora, PHP & Perl fan (wanna fight?)
- MUUG Vice-President
- STUG Past-President (defunct Atari ST club)
- Owner, Tecnopolis Enterprises, est. 1999

What’s a Function?

- a.php

What's a First-class Function?

- Where functions are "first-class citizens"
- Functions can be passed as arguments to other functions
- Functions can be assigned to variables
- Functions can be returned as results from other functions
- From the world of functional programming languages
- From the mid-60's (Lisp/Scheme are early examples)

Anonymous Functions

- Any function that isn’t named

First-class Function

- b.php

Changing Function Behaviour with Flags

- The simplest technique
- Pass a flag to a function to change its behaviour
- c.php

Why Not Pass The Behaviour?

- First-class functions allow us to pass behaviours instead of flags
- Sometimes advantageous to have the logic at the caller
- … or elsewhere
- … rather than in the function
- d.php

What Does This Remind You Of?

- ?

OO (Object Oriented) Programming

- Changing behaviours with code instead of flags smells like
- … OO subclasses and method overriding
- … a bit, at least

Closures

- Wikipedia:
- “a technique for implementing lexically scoped name binding in a language with first-class functions”
- Uh...

Closures

- Wikipedia:
- “a record storing a function together with an environment”
- Better!

Without Closures

- What if we want to write a function that when called
- … increments a counter and then outputs its value
- … remembering the counter between calls
- Without OO, we need a global variable
- … in addition to the the global functions
- e.php

With Closures

- Here’s the magic
- f.php

IncClosure Notes

- The counter variable is local to closure-generating function
- It is made accessible inside the closure in php with “use”
- Must be used “by reference” with php’s “&”
- The counter acts like a global in that it survives the scope of the function
- But unlike a global it is protected and hidden globally
- Unless accessed via the closure

Multiple Independent Counters

- Each call to the closure generator creates a new environment
- And a new counter
- Completely distinct and separate from any other counters
- g.php

Always Be Terse

- Eliminate the $inc var in the generator
- Return the anonymous function directly
- h.php

More Features

- How about decrementing?
- i.php
- Return two closures
- Both have access to the same “hidden global”

Another Caller Syntax

- How about call via a list
- Only one closure-calling variable
- Could allow access to many closure functions
- j.php

What Does This Remind You Of (Redux)?

- ?

OO Again

- Data encapsulation
- Data protection
- The data and the code become one
- But without the idiosyncrasies and syntax of OO
- One isn’t “better”
- OO does offer more than just this though
- Another tool in your toolbox

From the Horse's Mouth

- Wikipedia:
- "Constructs such as objects […] can thus be implemented with closures."

OO Similarities

- The list-calling syntax method looks a lot like object syntax:
- object.method()
- $object→method() in scripting languages
- vs
- $object[method]()

What's It Good For?

- Neat! But why?
- Often used with callbacks
- Event handlers (js)
- "to hide state" (Wikipedia)
- Closures "delay evaluation", only "doing" something when they are called
- "all of Smalltalk's standard control structures […] are defined using objects whose methods accept closures" (Wikipedia)

And…

- It’s just darn super cool!

Language Support

- Most modern scripting (Perl, PHP, Python...)
- js
- Java: 8 adds lambda expressions
- C is a special case, with only GCC allowing “nested functions”?
- clang
- C++, Objective-C 2.0 “blocks”

References

- https://en.wikipedia.org/wiki/First-class_function
- https://en.wikipedia.org/wiki/Closure_(computer_science)