

Events

Creating an event

What's that timmy? You're exposing your API to someone else, and they want convenient handy-dandy events?! No problem! As with everything in Boo, it's **dead simple** to get up and running within moments with the event keyword.

```
import System

class Clicker:
    event Clicked as EventHandler
    def RaiseClick():
        Clicked(self, null)
```

Aww, yeah! Now you can use it like any other event!

```
p = Clicker()
p.Clicked += def(sender, args):
    print "Tah CLICKED!"
p.RaiseClick()
```

(it prints out "Tah CLICKED!" if you're curious)

By now you're probably wondering what "EventHandler" is. In .NET, they are commonly referred to as delegates, a type of event that can be subscribed to by multiple functions, as long as each function has the same method signature as the delegate. An "event" is a special kind of delegate that has some rules:

- It can't be called from outside its declaring class.

This keeps sneaky coders from invoking an event from somewhere else and potentially messing things up. Leaky boats are bad!

Suppose, though, you want a ham sandwich on rye, or you want to expose a unique kind of event that has its own unique arguments. Here's another code sample to whet your appetite:

```

import System

class Sandwich:
    //object - sandwich eaten.
    //bool - if there are leftovers.
    event Eaten as callable(object, bool)
    //The two codes of line below are
    //the equivalent of the one line of code
above!
    event Eating as EatingEvent
    //object -- sandwich being eaten.
    //string -- the kind of sammich being
eaten.
    callable EatingEvent(sammich as object,
type as string)

    def Eat():
        Eating(self, "Turkey sammich.")
        Eaten(self, false)

turkeyAndSwiss = Sandwich()
turkeyAndSwiss.Eating += def(obj, sammich):
    print "You're eating a $sammich! It must be
good."
turkeyAndSwiss.Eaten += def(obj, leftovers):
    print "You", ("didn't leave me
anything?!","left leftovers! How
sweet!")[leftovers]
turkeyAndSwiss.Eat()

```

See Also

[Callable Types](#), [Functions](#), [Parameters](#), [Events](#)