CallOverloading

November 12, 2018

0.1 Call Overloading

Call overloading is a fairly new feature to Julia. The idea is that a function is also just a type: it's a type which is "callable", i.e. the instance of the type acts like a function. These functions compile and thus are as fast as any other function. As of v0.5, anonymous functions are now instances of callable types which are generated as needed.

To make a callable type, first make a type:

```
In [1]: mutable struct CallTest
            a::Float64
        end
```

Then define its call as follows:

```
In [2]: (c::CallTest)(b::Float64) = c.a*b
```

Note that the fields of the `CallTest` from which the function is defined are available from within the function (this can be used to store model parameters, see ParameterizedFunctions.jl). Now if you make an instance of `CallTest`, it will both act like your type an the function you described. For example:

```
In [3]: c = CallTest(2.0)
        c(3.0)

Out[3]: 6.0
```

You can add extra dispatches to the function on your type:

```
In [4]: (c::CallTest)(b::Int) = c.a/b

In [5]: c(2)

Out[5]: 1.0
```