Ruby master - Feature #12625

TypeError.assert, ArgumentError.assert

07/24/2016 11:36 PM - eike.rb (Eike Dierks)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

I am well aware that ruby is not typed (and so for a good reason)

But sometimes it makes sense to check the types (or values) of arguments upfront.
(It actually helps to narrow down your arguments to the known)

I'd like to suggest an extension to all the Error types.
(I should obviously come up with a gem first, but you get me)

I frequently write (in the prologue)
raise TypeError.new(...) unless SomeClass === arg

I suggest to extend TypeError so that I could write instead:
TypeError.assert(msg)(SomeClass, arg)
and i'd expect that to raise the exception in the calling frame (which can be done via the binding?)

We might want to do the same for ArgumentError:
ArgumentError.assert(msg){|arg| arg.nil?}
ArgumentError.assert('too low'){|arg| arg<0}

But obviously, this api is just a first shot.

I'm with you: we can't check args at compile time.
But checking args at runtime
should be done via well known verses.

added bonus:
we might want to make that a domain language.
We could attach the argument checking rules to the method.

This might help in generating automated testing rules.

No one wants types.
but annotating types helps a lot.

It helps in documenting the code.
It helps in making the code more foolproof.

```
def foo(arg)
    TypeError.assert(String, arg) # should raise in the foo frame if arg is not of type String
    ...
end
```

History

#1 - 07/25/2016 01:55 AM - shyouhei (Shyouhei Urabe)

I'm not sure wether I should be for this one or !==, but at least I can agree that raise TypeError unless SomeClass === arg is an idiom. I also experienced this before.

#2 - 07/25/2016 06:04 AM - akhramov (Artem Khramov)

Hi. It sounds like design-by-contract programming for me.

There are a few gems you might want to take a look at.

#3 - 08/10/2016 04:36 AM - shyouhei (Shyouhei Urabe)

See also [#12624](#), Matz do not want assertion by class.