Class

Binding < Object

Objects of class Binding encapsulate the execution context at some particular place in the code and retain this context for future use. The variables, methods, value of self, and possibly an iterator block accessible in this context are all retained. Binding objects can be created using Kernel#binding and are made available to the callback of Kernel#set_trace_func.

These binding objects can be passed as the second argument of the Kernel#eval method, establishing an environment for the evaluation.

```
class Demo
  def initialize(n)
    @secret = n
  end
  def get_binding
    return binding()
  end
end

k1 = Demo.new(99)
b1 = k1.get_binding
k2 = Demo.new(-3)
b2 = k2.get_binding

# Pass to eval...
eval("@secret", b1)   # =>   99
# Or eval via binding...
b2.eval("@secret")    # =>   -3

eval("@secret")       # =>   nil
```

Instance methods

eval

bind.eval(*string* ⟨ , *file* ⟨ , *line* ⟩ ⟩) → *obj*

1.9

Evaluates the Ruby code in *string* using the context of *bind*. Equivalent to calling Kernel#eval with a second argument of *bind*. See the start of this section for an example.