Ruby - Feature #8923

Frozen nil/true/false

09/19/2013 11:21 AM - ko1 (Koichi Sasada)

Status:	Closed
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	2.2.0

Description

Related to [Feature #8906]

We already froze Integer, Float. Symbols soon (now working).

How about to freeze nil, true and false, too?

frozen ruby? "Ruby"f?

Related issues:

Related to Ruby - Feature #8579: Frozen string syntax	**Closed**	**06/29/2013**

Associated revisions

Revision dd9a92417d0e7d9fd3545feda6240140de50fac8 - 09/11/2014 06:13 AM - nobu (Nobuyoshi Nakada)

test_object.rb: add assertions

- test/ruby/test_object.rb (test_freeze_immediate): assertions for [Feature #8923].

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@47525 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision dd9a9241 - 09/11/2014 06:13 AM - nobu (Nobuyoshi Nakada)

test_object.rb: add assertions

- test/ruby/test_object.rb (test_freeze_immediate): assertions for [Feature #8923].

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@47525 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 152d36a79eacce66c65e154a9df28422f69d7f29 - 06/03/2015 08:53 PM - Eric Wong

variable.c: remove generic ivar support for special constants

Special constants are all frozen since [Feature #8923] and cannot
support ivars. Remove some unused code we had for supporting them.

- variable.c (special_generic_ivar): remove flag
  (givar_i, rb_mark_generic_ivar_tbl): remove functions
  (rb_free_generic_ivar, rb_ivar_lookup, rb_ivar_delete,
  generic_ivar_set, rb_ivar_set, rb_ivar_defined,
  rb_copy_generic_ivar, rb_ivar_foreach, rb_ivar_count,
  rb_obj_remove_instance_variable):
  adjust for lack of ivar support in special constants
- test/ruby/test_variable.rb: test ivars for special consts
- internal.h: remove rb_mark_generic_ivar_tbl decl
- gc.c (gc_mark_roots): remove rb_mark_generic_ivar_tbl call

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@50758 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 152d36a7 - 06/03/2015 08:53 PM - Eric Wong

variable.c: remove generic ivar support for special constants

Special constants are all frozen since [Feature #8923] and cannot
support ivars. Remove some unused code we had for supporting them.

- variable.c (special_generic_ivar): remove flag

(givar_i, rb_mark_generic_ivar_tbl): remove functions
(rb_free_generic_ivar, rb_ivar_lookup, rb_ivar_delete,
generic_ivar_set, rb_ivar_set, rb_ivar_defined,
rb_copy_generic_ivar, rb_ivar_foreach, rb_ivar_count,
rb_obj_remove_instance_variable):
adjust for lack of ivar support in special constants

- test/ruby/test_variable.rb: test ivars for special consts
- internal.h: remove rb_mark_generic_ivar_tbl decl
- gc.c (gc_mark_roots): remove rb_mark_generic_ivar_tbl call

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@50758 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 09/19/2013 03:43 PM - Hanmac (Hans Mackowiak)

they are a bit different with methods, because defining singleton methods on them redirects to the curresponding Class

but yes, instance variables on them doesnt make much sense

#2 - 09/19/2013 11:00 PM - headius (Charles Nutter)

I obviously support this :-)

#3 - 01/30/2014 06:17 AM - hsbt (Hiroshi SHIBATA)

- Target version changed from 2.1.0 to 2.2.0

#4 - 08/31/2014 05:26 AM - ko1 (Koichi Sasada)

Matz, what do you think about it?

#5 - 09/04/2014 11:27 AM - matz (Yukihiro Matsumoto)

I agreed with making those values, if no significant comparability problem happens.
Please experiment.

Matz.

#6 - 09/05/2014 10:31 PM - nobu (Nobuyoshi Nakada)

- Description updated

#7 - 09/08/2014 07:17 AM - Hanmac (Hans Mackowiak)

i am unsure about toally freeze them ... some might extend/include some "Boolean" module into true/false or the TrueClass/FalseClass to specially check if a value is only true or false

i dont know if freezing this objects would break such a thing ...

#8 - 09/10/2014 08:28 AM - ko1 (Koichi Sasada)

The following patch makes test-all green.

```
Index: gc.c
===================================================================
--- gc.c (revision 47513)
+++ gc.c (working copy)
@@ -2286,11 +2286,11 @@ should_be_callable(VALUE block)
 static void
 should_be_finalizable(VALUE obj)
 {
-    rb_check_frozen(obj);
     if (!FL_ABLE(obj)) {
   rb_raise(rb_eArgError, "cannot define finalizer for %s",
     rb_obj_classname(obj));
     }
+    rb_check_frozen(obj);
 }

 /*
Index: include/ruby/ruby.h
===================================================================
--- include/ruby/ruby.h (revision 47513)
+++ include/ruby/ruby.h (working copy)
@@ -1126,7 +1126,7 @@ struct RStruct {
```

```
     (OBJ_TAINTABLE(x) && FL_ABLE(s)) ? \
     RBASIC(x)->flags |= RBASIC(s)->flags & FL_TAINT : 0)

-#define OBJ_FROZEN(x) (!!(FL_ABLE(x)?(RBASIC(x)->flags&(FL_FREEZE)):(FIXNUM_P(x)||FLONUM_P(x)||STATIC_SYM_P(x
))))
+#define OBJ_FROZEN(x) (!!(FL_ABLE(x)?(RBASIC(x)->flags&(FL_FREEZE)):1))
 #define OBJ_FREEZE(x) FL_SET((x), FL_FREEZE)

 #if USE_RGENGC
Index: object.c
===================================================================
--- object.c (revision 47513)
+++ object.c (working copy)
@@ -980,7 +980,7 @@ rb_obj_tainted(VALUE obj)
 VALUE
 rb_obj_taint(VALUE obj)
 {
-    if (!OBJ_TAINTED(obj)) {
+    if (!OBJ_TAINTED(obj) && OBJ_TAINTABLE(obj)) {
   rb_check_frozen(obj);
   OBJ_TAINT(obj);
     }
@@ -1057,8 +1057,6 @@ rb_obj_infect(VALUE obj1, VALUE obj2)
     OBJ_INFECT(obj1, obj2);
 }

-static st_table *immediate_frozen_tbl = 0;
-
 /*
  *  call-seq:
  *     obj.freeze    -> obj
@@ -1089,10 +1087,7 @@ rb_obj_freeze(VALUE obj)
     if (!OBJ_FROZEN(obj)) {
   OBJ_FREEZE(obj);
   if (SPECIAL_CONST_P(obj)) {
-     if (!immediate_frozen_tbl) {
-  immediate_frozen_tbl = st_init_numtable();
-     }
-     st_insert(immediate_frozen_tbl, obj, (st_data_t)Qtrue);
+     rb_bug("special consts should be frozen.");
   }
     }
     return obj;
@@ -1112,12 +1107,7 @@ rb_obj_freeze(VALUE obj)
 VALUE
 rb_obj_frozen_p(VALUE obj)
 {
-    if (OBJ_FROZEN(obj)) return Qtrue;
-    if (SPECIAL_CONST_P(obj)) {
- if (!immediate_frozen_tbl) return Qfalse;
- if (st_lookup(immediate_frozen_tbl, obj, 0)) return Qtrue;
-    }
-    return Qfalse;
+    return OBJ_FROZEN(obj) ? Qtrue : Qfalse;
 }


Index: test/ruby/test_eval.rb
===================================================================
--- test/ruby/test_eval.rb (revision 47513)
+++ test/ruby/test_eval.rb (working copy)
@@ -130,7 +130,7 @@ class TestEval < Test::Unit::TestCase
   def forall_TYPE
     objects = [Object.new, [], nil, true, false] # TODO: check
     objects.each do |obj|
-      obj.instance_variable_set :@ivar, 12
+      obj.instance_variable_set :@ivar, 12 unless obj.frozen?
       yield obj
     end
   end
@@ -145,7 +145,7 @@ class TestEval < Test::Unit::TestCase
       assert_equal :sym,  o.instance_eval(":sym")

       assert_equal 11,    o.instance_eval("11")
-      assert_equal 12,    o.instance_eval("@ivar")
```

```
+       assert_equal 12,    o.instance_eval("@ivar") unless o.frozen?
        assert_equal 13,    o.instance_eval("@@cvar")
        assert_equal 14,    o.instance_eval("$gvar__eval")
        assert_equal 15,    o.instance_eval("Const")
@@ -155,7 +155,7 @@ class TestEval < Test::Unit::TestCase
        assert_equal "19",  o.instance_eval(%q("1#{9}"))

        1.times {
-         assert_equal 12,  o.instance_eval("@ivar")
+         assert_equal 12,  o.instance_eval("@ivar") unless o.frozen?
          assert_equal 13,  o.instance_eval("@@cvar")
          assert_equal 14,  o.instance_eval("$gvar__eval")
          assert_equal 15,  o.instance_eval("Const")
@@ -173,7 +173,7 @@ class TestEval < Test::Unit::TestCase
        assert_equal :sym,  o.instance_eval { :sym }

        assert_equal 11,    o.instance_eval { 11 }
-       assert_equal 12,    o.instance_eval { @ivar }
+       assert_equal 12,    o.instance_eval { @ivar } unless o.frozen?
        assert_equal 13,    o.instance_eval { @@cvar }
        assert_equal 14,    o.instance_eval { $gvar__eval }
        assert_equal 15,    o.instance_eval { Const }
@@ -183,7 +183,7 @@ class TestEval < Test::Unit::TestCase
        assert_equal "19",  o.instance_eval { "1#{9}" }

        1.times {
-         assert_equal 12,  o.instance_eval { @ivar }
+         assert_equal 12,  o.instance_eval { @ivar } unless o.frozen?
          assert_equal 13,  o.instance_eval { @@cvar }
          assert_equal 14,  o.instance_eval { $gvar__eval }
          assert_equal 15,  o.instance_eval { Const }
Index: test/ruby/test_weakmap.rb
===================================================================
--- test/ruby/test_weakmap.rb (revision 47513)
+++ test/ruby/test_weakmap.rb (working copy)
@@ -19,13 +19,13 @@ class TestWeakMap < Test::Unit::TestCase
     assert_raise(ArgumentError) {@wm[true] = x}
     assert_raise(ArgumentError) {@wm[false] = x}
     assert_raise(ArgumentError) {@wm[nil] = x}
-    assert_raise(RuntimeError) {@wm[42] = x}
-    assert_raise(RuntimeError) {@wm[:foo] = x}
+    assert_raise(ArgumentError) {@wm[42] = x}
+    assert_raise(ArgumentError) {@wm[:foo] = x}
     assert_raise(ArgumentError) {@wm[x] = true}
     assert_raise(ArgumentError) {@wm[x] = false}
     assert_raise(ArgumentError) {@wm[x] = nil}
-    assert_raise(RuntimeError) {@wm[x] = 42}
-    assert_raise(RuntimeError) {@wm[x] = :foo}
+    assert_raise(ArgumentError) {@wm[x] = 42}
+    assert_raise(ArgumentError) {@wm[x] = :foo}
   end

   def test_include?
```

#9 - 09/10/2014 08:30 AM - ko1 (Koichi Sasada)

Hans Mackowiak wrote:

> i am unsure about toally freeze them ... some might extend/include some "Boolean" module into true/false or the TrueClass/FalseClass to specially check if a value is only true or false
>
> i dont know if freezing this objects would break such a thing ...

```
p true.frozen? #=> true
module Boolean; end
class TrueClass
  include Boolean
end
p true.kind_of?(Boolean) #=> true
```

Not freeze TrueClass, but freeze true object.

#10 - 09/10/2014 10:32 AM - fxn (Xavier Noria)

Hans Mackowiak wrote:

> i am unsure about toally freeze them ... some might extend/include some "Boolean" module into true/false or the TrueClass/FalseClass to specially check if a value is only true or false

Can you explain that use case a little further?

#11 - 09/11/2014 05:24 AM - matz (Yukihiro Matsumoto)

@ko1 (Koichi Sasada) you have proven there's no significant issue. Go ahead.
Of course, we might have to revert it if something happens outside of test suites.

Matz.

#12 - 09/11/2014 06:14 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

- % Done changed from 0 to 100

Applied in changeset r47525.

---

test_object.rb: add assertions

- test/ruby/test_object.rb (test_freeze_immediate): assertions for
  [Feature #8923].