Cheatography

Extending Ruby with C - Part 2 Cheat Sheet

by Ryan Johnson (CITguy) via cheatography.com/138/cs/249/

Ruby C - Common Methods

int **rb_respond_to**(VALUE self, ID method) => 0|nonzero

VALUE **rb_thread_create**(VALUE (*func)(), void *data)
> Runs *func* in new thread, passing *data* as an arg.

VALUE **rb_obj_is_instance_of**(VALUE obj, VALUE klass) => Qtrue|Qfalse

VALUE **rb_obj_is_kind_of**(VALUE obj, VALUE klass)
> Returns Qtrue if *klass* is superclass of *obj* class.

Ruby C - Exceptions

void **rb_raise**(V exception, const char *fmt, ...)
> Raises *exception*. *fmt* and args used like in printf.

void **rb_fatal**(const char *fmt, ...)
> Raises Fatal exception, terminating process. No rescue blocks called, but ensure blocks will be called. *fmt* and args used like in printf.

void **rb_bug**(const char *fmt, ...)
> Terminates process immediately--no handlers of any sort called. *fmt* and args are interpreted like printf. *Call only if a fatal bug has been exposed.*

void **rb_sys_fail** (const char *msg)
> Raises a platform-specific exception corresponding to last known system error, with the given *msg*.

V **rb_rescue**(V (*body)(), V args, V (*rescue)(), V rargs)
> Executes body with given *args*. If StandardError exception raised, execute *rescue* with given *rargs*.

V **rb_ensure**(V (*body)(), V args, V (*rescue)(), V eargs)
> Executes body with given *args*. Whether or not an exception is raised, execute *ensure* with given *rargs* after *body* has completed.

V **rb_protect**(V (*body)(), V args, int *result)
> Executes *body* with given *args* and returns nonzero in *result* if any exception raised.

void **rb_notimplement**()
> Raises NotImpError exception to indicate enclosed function is NYI, or not available on platform.

void **rb_exit**(int status)
> Exits Ruby with given *status*. Raises SystemExit exception and calls registered exit functions/final izers.

void **rb_warn**(const char *fmt, ...)
> Unconditionally issues warning message to standard error. *fmt* and args used like in printf.

void **rb_warning**(const char *fmt, ...)
> Conditionally issues a warning message to standard error if Ruby was invoked with the -w flag. *fmt* and args used like in printf.

V = VALUE

Ruby C - Array Methods

VALUE **rb_ary_new**()
> Returns new Array with default size.

VALUE **rb_ary_new2**(long length)
> Returns new Array of given *length*.

VALUE **rb_ary_new3**(long length, ...)
> Returns new Array of given *length* and populated with remaining arguments.

VALUE **rb_ary_new4**(long length, VALUE *values)
> Returns new Array of given *length* and populated with C array *values*.

void **rb_ary_store**(VALUE self, long index, VALUE value)
> Stores *value* at *index* in array *self*.

VALUE **rb_ary_push**(VALUE self, VALUE value)

VALUE **rb_ary_pop**(VALUE self)

VALUE **rb_ary_shift**(VALUE self)

VALUE **rb_ary_unshift**(VALUE self, VALUE value)

VALUE **rb_ary_entry**(VALUE self, long index)
> Returns array *self*'s element at *index*.

Ruby C - Iterators

void **rb_iter_break**()
> Breaks out of enclosing iterator block.

VALUE **rb_each**(VALUE obj)
> Invokes 'each' method of the given *obj*.

VALUE **rb_yield**(VALUE arg)
> Transfers execution to iterator block in the current context, passing *arg* as an argument. Multiple values may be passed in an array.

int **rb_block_given_p**()
> Nonzero if yield would execute a block in current context--that is, if a code block was passed to current method and is available to be called.

VALUE **rb_iterate**(VALUE (*method)(), VALUE args, VALUE (*block)(), VALUE arg2)
> Invokes *method* with *args* and block *block*. Yield from that method will invoke *block* with arg given to yield and second arg *arg2*.

VALUE **rb_catch**(const char *tag, VALUE (*proc)(), VALUE value)
> Equivalent to Ruby catch.

void **rb_throw**(const char *tag, VALUE value)
> Equivalent to Ruby throw.

Ruby C - Hash Methods

VALUE **rb_hash_new**()

VALUE **rb_hash_aref**(VALUE self, VALUE key)
> Returns element corresponding to *key* in *self*.

VALUE **rb_hash_aset**(VALUE self, VALUE key, VALUE value)
> Sets value for *key* to *value* in *self*. Returns *self*.

Ruby C - Accessing Variables

V **rb_iv_get**(V obj, char *name)
> Returns instance var *name* (must specify "@" prefix) from given *obj*.

V **rb_ivar_get**(V obj, ID name)
> Returns instance var *name* from given *obj*.

V **rb_iv_set**(V obj, char *name, V value) => value
> Sets instance var *name* (must specify "@" prefix) in given *obj* to *value*.

V **rb_ivar_set**(V obj, ID name, V value)
> Sets instance var *name* in *obj* to *value*.

V **rb_gv_set**(const char *name, V value) => value
> Sets global var *name* ("$" prefix optional) to *value*.

V **rb_gv_get**(const char *name)
> Returns global var *name* ("$" prefix optional).

void **rb_cvar_set**(V class, ID name, V val)
> Sets class var *name* in *class* to *value*.

V **rb_cvar_get**(V class, ID name)
> Returns class var *name* from given *class*.

int **rb_cvar_defined**(V class, ID name)
> Qtrue if class var *name* has been defined for *class*.

void **rb_cv_set**(V class, const char *name, V val)
> Sets class var *name* (must specify "@@" prefix) in given *class* to *value*.

V **rb_cv_get**(V class, const char *name)
> Returns class var *name* (must specify a "@@" prefix) from given *class*.

V = VALUE

Ruby C - String Methods

VALUE **rb_str_new**(const char *src, long length)=> String
> Initialized with *length* chars from *src*.

VALUE **rb_str_new2**(const char *src) => String
> Initialized with null-terminated C string *src*.

VALUE **rb_str_dup**(VALUE str) => String
> Duplicated from *str*.

VALUE **rb_str_cat**(VALUE self, const char *src, long length) => self
> Concatenates *length* chars from *src* onto *self*.

VALUE **rb_str_concat**(VALUE self, VALUE other) => self
> Concatenates *other* onto String *self*.

VALUE **rb_str_split**(VALUE self, const char *delim)
> Returns array of String objects created by splitting *self* on *delim*.

Cheatographer

Ryan Johnson (CITguy)
cheatography.com/citguy/

Cheat Sheet

This cheat sheet was published on 15th February, 2012 and was last updated on 17th February, 2012.

Sponsor