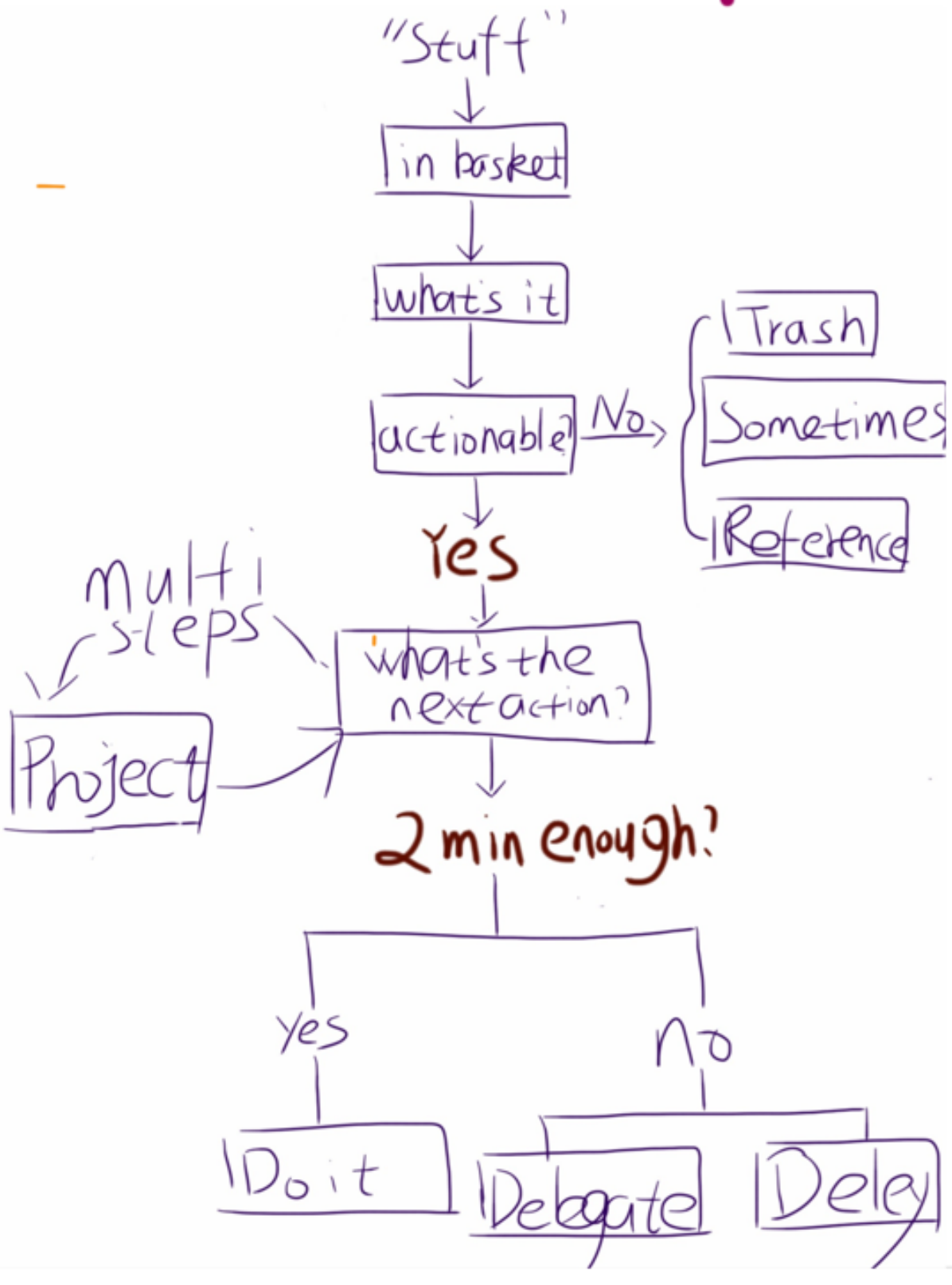
GTD — Getting Things Done

Focus On "Next Action"
Clear all things out of your mind
少于2分钟的事情立刻估文

Two key principles
1. Capturing all things
2. Disciplining yourself.

Anxiety comes from unmanaged committment, so you need to clear it out of your mind

GTD Workflow



认知与设计

——理解UI设计准则

一.我们感知自己的期望

人的感官经常只能感受到自己最先设定的期望目标

基于这个原理.UI设计有以下三个基本准则:

1. 消除歧义
2. 保持一致
3. 识别用户的目标.

二:视觉的工作原理

根无括来说.人眼视觉是整体性的,以下7条描述性的准则说明人眼如何识别出整体:

1. 接近性原理

   相互接近的物体属于一个整体

2. 相似性原理

   相似的物体属于一个整体

3. 连续性原理

人眼倾向于感受连续的物体，而不是离散的碎片

4. 封闭性原理

人眼倾向于感受封闭的物体. 它会自动尝试将尚未开的图形关闭起来

5. 对称性原理

人眼倾向于按照对称性，分解复杂的场景

6. 主体/背景原理

人眼倾向于将场景分为主体与背景，主体占据主要注意力.

7. 共同命运原理

一起运动的物体属于一个整体.

Ruby

Primitives: Data Type

number, +, -, *, /, %, **

string

- single quoted string, span multiple lines, \
  only apply to ' \.
- double quoted string, span multiple lines, \
  apply to a lot more characters, support interpolation #
- arbitrary delimiters for string literals.
  %q for single quoted rule
  %Q, or %, for double quoted rule
- here documents.
  <<Delimiter_string
  <<- Delimiter_string

so you can use strings as delimiters.

- String is mutable
- character literal: ?a
  integer in ruby 1.8.
  length 1 string in ruby 1.9
- +, *, <<

array

- literal: [ ]
- arbitrary delimiter:
  %w for single quote string
  %W for double quote string.
- slice and negative index.
- +, <<, 1.8

  + return new array, << modify current array.

Seperate words in string

hash

- literal: {key1 => value1, key2 => value2, ...}

range

- literal

  start..end for inclusive range

  Start...end for exclusive range.

- <=> for member comparision, succ method for next

  object. === for membership testing

Symbol

- literal :str
- arbitrary delimiter %s{str}

ture, false, nil

Primitives: Object.

should be BaseObject in Ruby 1.9

Base class of all objects.

- Object Creation

  class method new: allocate memory, and call initialize.

- Object Identification

  object_id method.

- Object Class

  class method to get Class object.

  instance_of? is equal to x.class = Class

  kind_of?, or is_a?, is used to test Is-A relation.

- Object Equality

  equal? to test reference.

  == would be modified to test value.

  eql? would be modified to a no-conversion ==

  === is used in case implictly.

  =~ would be modified to pattern matching.

- Object Conversion
  to_a, to_s method.
  Array, Float, Integer, String method.
- Object Copying
  clone and dup for shallow copy.
  initialize_copy is a copy constructor for deep copy used by dup/clone.
  difference between dup and clone: clone reserve freeze and taint status, and singleton method; dup don't
- Object Freezing
  freeze method to make object immutable

Primitives : Method

def to define method, undef to delete it.

Singleton method.

```
def o.method
  ...
end.
```

def method to current class

operator method

operator as method name

but +@ and -@ for unary plus and minus

method alias

alias new exist

Variable length argument list

```
def m(a1, *argument_as_array)
```

block argument

Block can have local variables

- every method can have a block argument, invoked by yield.
- block argument is a syntactic form, not object

- you can use &argument to get a Proc object created for the block.

  def m(&p) #P is a Proc object for block

- * and & used in method invoktion

  * means unpack array as seperate arguments

  & means using Proc object code for block.

- code in block structure can be considered to a closure

method object.

- method itself is only a syntactic form, not object. but you can get a Method object for a specific method, by meta-programming machenism.
- Method is not a closure.

hash argument form

key1 ⇒ value1, key2 ⇒ value2, ··· can be used in argument position; it'll become a hash object.

Primitives : Proc

Two types of Proc, both are closure

- proc
- Lambda

Creation Method

Proc.new {block} for proc

Kernel::lambda {block} for lambda

Applying Method

Proc::call

Difference between proc and lambda

- return in proc returns invokation method. as break, next, redo
  return in lambda only returns lambda itself.
- proc use yield semantics for argument. while lambda use method invocation semantics.
  basicly, number of arguments has to be exact for lambda calling.

Scope

- Scope is an evaluation environment, includes:

1. current object, as self
2. local variables
3. Constants
4. global variables. 5. current class

- Scope Gate is a way to enter new scope

1. class keyword
2. module keyword
3. def keyword

- closure will save scope.

eval family

- eval

  evaluate string in current scope.

- instance_eval and class_eval

  change self as calling obj, current class as eigenclass or calling object (is itself a class)

Constant is packaged like files in directory.

- class or module is like directory.
- Constant is like files.
- Constant is globally accessible, as long as you use full path.

Hook method

you can listen to events happening within object model, like which class includes which module.

Combination

backtick command

- platform dependent
- `` `str` ``: str is a double quoted string, passed to Kernel.` method, the value of this expression is the return value of Kernel.` .

Variable names by lexical scope and Constant Property.

- Global Variable

  $var
- Class Variable

  @@ var
- Instance Variable

  @ var
- Local Variable

  var, or _var

- Constant Variable
  Var

parentheses can be used to group expressions
false and nil can be considered false in
  Boolean Context.

Conditional

- if
  else and elsif clause
  modifier syntax
  newline and then are delimiters
- unless
  else clause
  modifier syntax
  newline and then are delimiters
- case
  else clause
  many when clauses, then and newline are delimiters

=== used in when implicitly.

Loop

- while/until
  newline and do are delimiters
  modifier syntax
- for
  can be used in Enumerable object
  invoke each implictly
  newline and do are delimiters.

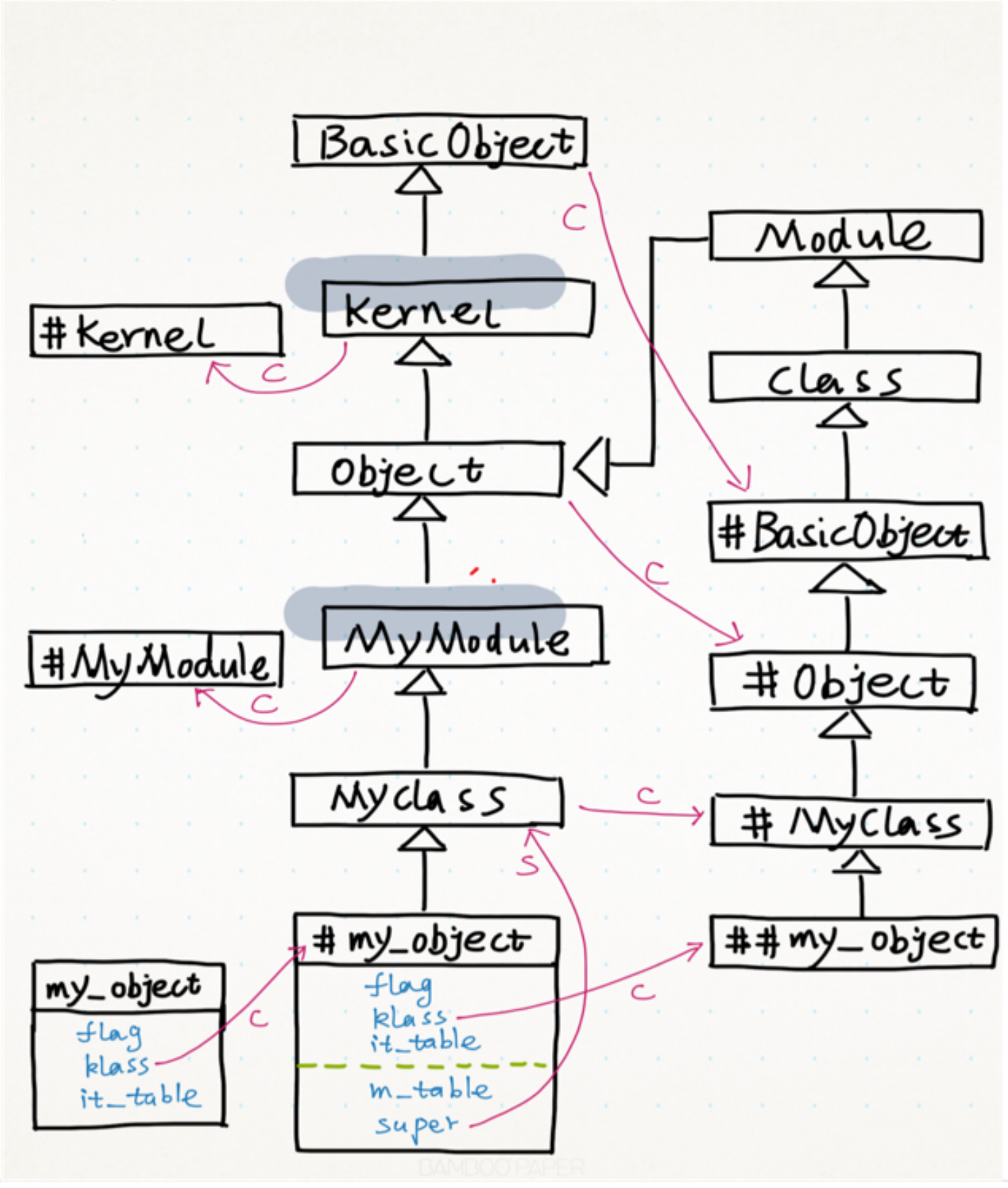
yield is used to invoke block argument.

break/next/redo

- can only be used whthin loop or block
- when used in block, it applys to the invoking method.

Abstraction

Ruby Object Model



Unified Theory of Ruby Object Model:

1. Only 2 kinds of object: regular object or class object.
2. included module will put a proxy in the ancestor chain. to support module sharing. however, module eigenclass will not be put into the "right" chain
3. every object has a eigenclass as its real class; and obj.superall.eigenclass is equal to obj.eigenclass.superclass. However, module seems to be an exception.
4. Method lookup rule: right and up.

When do Ruby create local variable?

Ruby解释器在看到赋值语句的时候，才会创建Local Variable.

换句话说，如果某个名称的Local Variable不存在，那么对其的第一次赋值会导致该变量被创建.

注意以下的例子：

```
class A
  def f1
    @var
  end
  def f2
    f1=1
  end
end
```

在以上的例子中，f2中使用的f1是指新创建的变量，而不是指方法f1.

A funny feature : Named Capturing In Regexp

Name Capturing can be used to back-reference.

(?<name>regexp-pattern), use \k<name> to back reference.

can NOT be used spontaneouly with numbered capturing

=~ operator create local variables according to name capturing

For Example:

/(?<keyword>\w+)/ =~ "abc"

create local variable keyword. whose value is string "abc", in current scope.

Ruby Development Environment

Install Ruby 1.9.3 in Win7

need to install DevKit, in order to add native extension gem (like debugger)

Ruby Debugger

- default debugger usage: ruby -r debug file.rb
  too slow.
- debugger gem
  fast debugger implementation, usage: rdebug file.rb
  better and newer than ruby-debug gem. easily to install, just need to gem install debugger.

Ruby IDE

RubyMine seems good. can be used as front end of debugger gem. GUI debugger is very cool

Module

Can be used as Namespace, or mixin

Namespace : Module.func

Mixin : include module

implementation

proxy object (for sharing) in ancestor chain. However. the eigenclass of this proxy will not in the right chain.

Module # module_definition

a convinient way to fulfill two purpose (namespace & mixin). it will create class method. and private instance method.

mixin

include, and extend only accept Module type object. an Module object can not be instantiated, module inheritance hierarchy can not exist, However. it can include other Module.

mixin can be seen as restricted multiple inheritance

inheritance hierarchy in ruby is tree like, mixin is multiple inheritance in essence, in a restricted way.

eval environment & eval

eval environment, also called scope

data included: current object (self),
current class, local variables.

Scope gate create new scope

class/module keyword

def keyword

Specific scope

→ Top-Level

current class = Object

current object = main

→ Top-Level in Loaded Module

current class = Object

current object = main

same object

→ Inside class/module, outside def

current class = defined class

current object = defined class

→ Inside def

current class = defined class

current object = calling instance

eval family (do not create new scope)

→ Kernel#eval (str, scope)

→ BasicObject#instance_eval

accept str, or block

reset current object to the calling object.

reset current class to the eigenclass of calling object.

→ Module#class_eval , Module#module_eval

accept str, or block.

reset current object to the calling object.

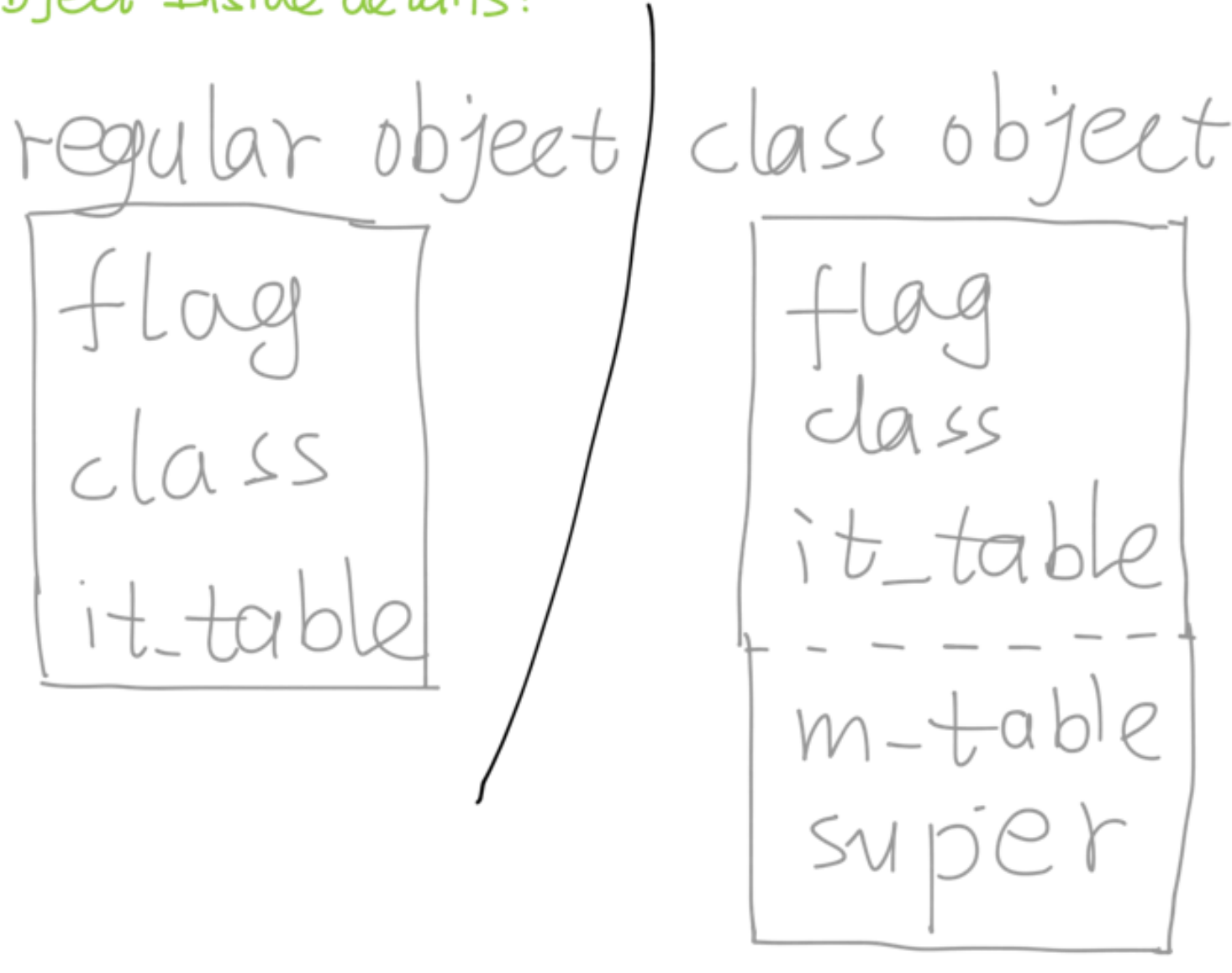
reset current class to the calling object.

Reflection

access and modify program runtime information.

basicly, these functions are defined in Kernel, Object Module, and Class.

Object Inside details:



Enumerator

Internal Iterator

→ Encapslate iterate process, easy to use and implement, difficult to customize iteration process

→ Visitor Pattern

→ Inconenient to use in language that absent closure

External Iterator

→ client can control iteration process, difficult to implement.

→ Iterator Pattern

→ Kernel#loop is used fo enumerator. since it stop iteration when StopIteration exception raised.

Character Set and Encoding

Character Set

A set of characters, in which each character maps to a distinct number. This number is called Code Point

Character Encoding Scheme

Binary representation of Code Point.

Unicode

→ Unicode is a Character Set.

→ There are some Character Encoding Schemes for Unicode

UTF-8

UTF-16LE

UTF-16BE

UTF-32LE

UTF-32BE

UTF-8 scheme is variant-length, compatible with ASCII, does not contain NULL(0) byte, and does not

have LE/BE issue.

UCS and CSI.

→ 2 ways programming language manipulate string

→ UCS (Universal Character Set) style use internal encoding for string. During Input/Output, Conversion may be needed.

→ CSIC (Character Set Independent) style do not change encoding ever.

→ Ruby 1.9 use CSI style, encoding is stored in each string

Local Variable Definition in eval

→ eval("var = 1", binding)会在当前binding中增加局部变量var吗?

Yes. But it can not be accessed outside of eval.

Because local variables are created in compile time. According to Mats

C++ Template

Function Template

- using typename as class parameter keyword
- compiler support argument deduction
- 2 kinds of template parameter:
  - class parameter
  - nontype parameter
- function template can be overloaded.

Class Template

- use typename as class parameter keyword
  class parameter can be used as a ordinary type inside class declaration.
- class type should be ClassName<TypeName> when you need it. if you only need class name (class constructor, desturctor), you only use ClassName.

- member function of class template looks like a function template, except that template type declaration should be identical to class template type declaration.
- class template specification.
  need to do all member function specification
  template<> .....
- class template partial specification.
  specify some arguments.
- default template arguments.
  previous parameter could be used.

Nontype template parameters.
can only be integer number, enum, and pointers to external linkage object. or reference

Member Templates
could be nested class template, or member function template. has multiple parameter clause

member of class template is not member template independent

template template parameter

- class template could also be template parameter.
  template<template<typename T> class C> ...
- parameters musted be exactly matched, default value will not be considered.

zero initialization

- for basic primitive types T, you can use T() to get "zero" value.

member of class template is Not member function template.

member function template can not be virtual, since you don't know the numbers

template usually is external linked, except static. can not use C linkage convention.

primary template is normal template, partial specification is non-primary template

type parameter 中的 type，不一定是 class 也不一定有名字。

---

function template 具现化的 function 很不一样

qualified name and dependent name

以scope resolution运算符
或者member access运算符
指明的名称

名称不依
赖于template
parameter

name
looking
up process

① Ordinary
look up:
对于qualified name, 在其
符号指定命名空间中找.

对于unqualified name
逐渐往上层的enclosing namespace中找；如果处于方法定义中，还要去类和基类的成员中找.

②ADL：Argument-dependent Lookup
只针对一种情况：函数调用函数名称为一个unqualified name.
优先在参数相关Namespace中找.

Friend Name Injection

如果某个friend function的
声明(注一声明位于class内部的friend
声明中，当某次对function的
调用(还没)ADL，且class是与
参数相关的class，那么这个
function的名称可以认为在
class所属的enclosing namespace
中可见

Class Name Injection

在类声明内部，class name
均为一个unqualified name，表
示类本身.

对于class template来说，
下来template parameter的类
名表示使用template parameter
的是，某具体argument的
数据类型是谁，

使用::class_name表示模
板本身，但要注意，
它是下的另一种
写法