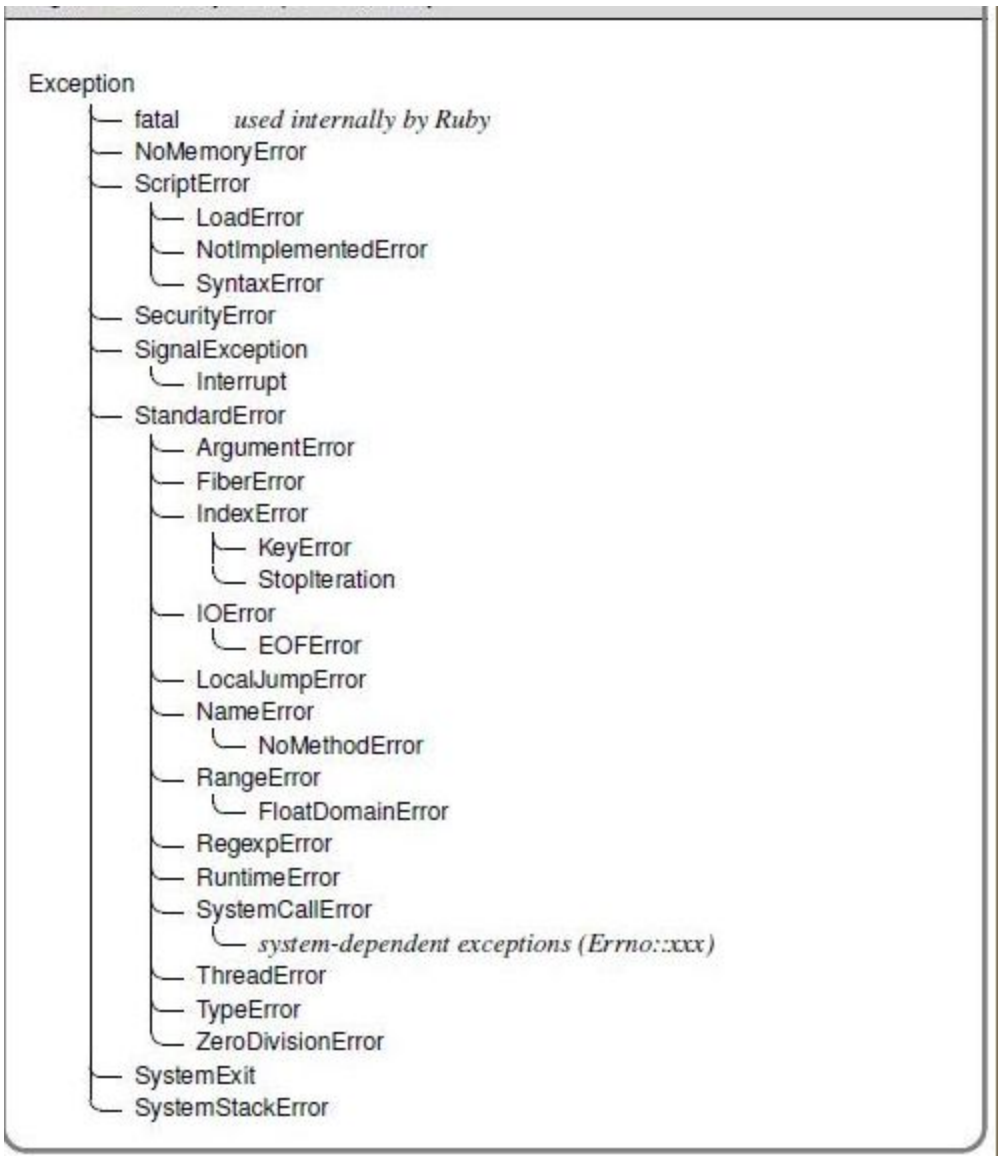
Ruby Exceptions

An exception is a special kind of object, an instance of the class Exception or a descendant of that class that represents some kind of exceptional condition; it indicates that something has gone wrong. When this occurs, an exception is raised. By default, Ruby programs terminate when an exception occurs.

Exception Handlers

An exception handler is a block of code that is executed if an exception occurs during the execution of some block of code. Raising an exception means stopping normal execution of the program and transferring the flow-of-control to the exception handling code where you either deal with the problem that's been encountered or exit the program completely.

Exception Hierarchy



The following method raises an exception whenever it's called:

```
def raise_exception
  puts 'Called before exception is raised'
  raise 'An exception is raised'
  puts 'Called after exception is raised'  #never printed
end
```

```
raise_exception

Called before exception is raised
RuntimeError: An exception is raised
      from (irb):3:in `raise_exception'
      from (irb):6
```

The raise method is from the Kernel module. By default, raise creates an exception of the RuntimeError class. To raise an exception of a specific class, you can pass in the class name as an argument to raise.

```
def twice(x)
  raise ArgumentError, 'Argument is not numeric' unless x.is_a?
Numeric
  x * 2
end

irb(main):011:0> puts twice(2)
4
=> nil

puts twice('String')
ArgumentError: Argument is not numeric
      from (irb):8:in `twice'
      from (irb):12
```