Cheat Sheet for comprehensive Ruby

Basic Syntax and Structure

- Comments

- Single-line: `# This is a comment`
- Multi-line: `=begin` and `=end`

- Variables

- Local: `variable_name`
- Instance: `@variable_name`
- Class: `@@variable_name`
- Global: `$variable_name`

- Constants

- Defined with uppercase: `CONSTANT_NAME`

- Operators

- Arithmetic: `+`, `-`, `*`, `/`, `%`, `**`

- Comparison: `==`, `!=`, `>`, `<`, `>=`, `<=`, `<=>`
- Logical: `&&`, `||`, `!`

- Assignment: `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `**=`

Control Structures

- Conditional Statements

- `if` statement:

```
if condition
  # code
elsif another_condition
  # code
else
  # code
end
```

- `unless` statement:

```
unless condition
  # code
end
```

- Ternary operator: `condition ? true_expression : false_expression`

- **Loops**

- `while` loop:

```
while condition
  # code
end
```

- `until` loop:

```
until condition
  # code
end
```

- `for` loop:

```
for i in 0..5
  # code
end
```

- `each` iterator:

```
(0..5).each do |i|
  # code
end
```

- `times` iterator:

```
5.times do |i|
  # code
end
```

Methods and Blocks

- Defining Methods

- Basic method:

```
def method_name(parameter)
  # code
end
```

- Default parameters:

```
def method_name(parameter = default_value)
  # code
end
```

- Variable number of arguments:

```
def method_name(*args)
  # code
end
```

- Blocks

- Single-line block:

```
array.each { |element| puts element }
```

- Multi-line block:

```
array.each do |element|
  puts element
end
```

- Lambdas and Procs

- Lambda:

```
my_lambda = lambda { |x| x * 2 }
```

- Proc:

```
my_proc = Proc.new { |x| x * 2 }
```

Classes and Objects

- Defining Classes

- Basic class:

```
class MyClass
  def initialize(parameter)
    @instance_variable = parameter
  end
end
```

- Accessor methods:

```
class MyClass
  attr_accessor :variable_name
end
```

- Inheritance

- Basic inheritance:

```
class ChildClass < ParentClass
  # code
end
```

- Modules

- Basic module:

```
module MyModule
  def my_method
    # code
  end
end
```

- Including a module:

```
class MyClass
  include MyModule
end
```

Collections

- **Arrays**

- Creating an array: `array = [1, 2, 3]`
- Accessing elements: `array[index]`
- Adding elements: `array.push(element)` or `array << element`
- Removing elements: `array.delete(element)`

- **Hashes**

- Creating a hash: `hash = { key1: value1, key2: value2 }`
- Accessing values: `hash[:key]`
- Adding key-value pairs: `hash[:new_key] = new_value`
- Removing key-value pairs: `hash.delete(:key)`

- **Ranges**

- Creating a range: `range = (1..5)`
- Iterating over a range:

```
(1..5).each do |i|
  # code
end
```

Strings and Symbols

- **Strings**

- Creating a string: `string = "Hello, World!"`
- String interpolation: `"#{variable_name}"`
- Common methods:
- `string.length`
- `string.upcase`
- `string.downcase`
- `string.include?("substring")`

- **Symbols**

- Creating a symbol: `:symbol_name`
- Symbols are immutable and reusable: `:symbol_name.object_id`

Error Handling

- Exceptions

- Basic exception handling:

```
begin
  # code
rescue StandardError => e
  puts e.message
end
```

- Raising exceptions:

```
raise "Error message"
```

File I/O

- Reading Files

- Basic file read:

```
File.read("filename.txt")
```

- Reading line by line:

```
File.foreach("filename.txt") do |line|
  # code
end
```

- Writing Files

- Basic file write:

```
File.write("filename.txt", "content")
```

- Appending to a file:

```
File.open("filename.txt", "a") do |file|
  file.write("content")
end
```

Regular Expressions

- Basic Regex

- Creating a regex: `regex = /pattern/`
- Matching: `string =~ regex`
- Common methods:
- `string.match(regex)`
- `string.scan(regex)`

Metaprogramming

- Dynamic Method Definition

- `define_method`:

```
define_method(:method_name) do |args|
  # code
end
```

- `send`:

```
object.send(:method_name, args)
```

- Singleton Methods

- Defining a singleton method:

```
def object.method_name
  # code
end
```

Tips and Tricks

- Shortcuts

- `array.map(&:method_name)` is equivalent to `array.map { |element| element.method_name }`
- `array.select(&:method_name)` is equivalent to `array.select { |element| element.method_name }`

- Debugging

- Use `puts` or `p` for quick debugging:

```
puts variable_name
p variable_name
```

- Use `binding.pry` for interactive debugging:

```
require 'pry'
binding.pry
```

- Performance

- Use `Benchmark` for performance testing:

```
require 'benchmark'
Benchmark.bm do |x|
  x.report { # code }
end
```

Advanced Topics

- Concurrency

- Threads:

```
thread = Thread.new { # code }
thread.join
```

- Fibers:

```
fiber = Fiber.new do
  # code
end
fiber.resume
```

- C Extensions

- Basic C extension:

```
VALUE method_name(VALUE self) {
  // code
}
```

- Compiling:

```
ruby extconf.rb
make
```

Useful Libraries

- Standard Library

- `csv`: For handling CSV files
- `json`: For JSON parsing and generation
- `net/http`: For HTTP requests
- `date`: For date manipulation

- Gems

- `rails`: Full-stack web framework
- `sinatra`: Lightweight web framework
- `rspec`: Testing framework
- `pry`: Interactive debugging

Resources

- Documentation

- [Ruby Documentation](https://ruby-doc.org/)
- [Ruby Core API](https://ruby-doc.org/core-X.X.X/)
- [Ruby Standard Library](https://ruby-doc.org/stdlib-X.X.X/)

- Community

- [Ruby Forum](https://www.ruby-forum.com/)
- [Stack Overflow](https://stackoverflow.com/questions/tagged/ruby)
- [RubyGems](https://rubygems.org/)

Conclusion

This cheat sheet covers the essential aspects of Ruby, from basic syntax to advanced topics. Use it as a quick reference to enhance your Ruby programming skills.

By Ahmed Baheeg Khorshid

ver 1.0