Ruby Cheatsheet

Basic Syntax and Data Types

Ruby is an interpreted, high-level, general-purpose programming language emphasizing simplicity and productivity.

- Variable Declaration:
  - `local_var`: Local variable.
  - `@instance_var`: Instance variable.
  - `$global_var`: Global variable.
  - `CONSTANT`: Constants (start with uppercase).
- Basic Types: `Integer`, `Float`, `String`, `Symbol`, `Boolean`, `NilClass`.
- Symbol: Immutable, reused strings. `:my_symbol`.
- String Interpolation: `"Hello, #{name}"`.

Control Flow

- `if-elsif-else`:

```ruby
if condition
  # ...
elsif other_condition
  # ...
else
  # ...
end
```

- `unless`: Inverse of `if`.
- `case-when`:

```ruby
case status
when "value1"
  # ...
when "value2"
  # ...
else
  # ...
end
```

- Loops:
  - `while condition ... end`
  - `until condition ... end`
  - `for item in collection ... end` (Rarely used)

Methods

- Definition:

```ruby
def my_method(param1, param2="default")
  # Body
  param1 + param2 # Last expression is implicitly returned
end
```

- `return`: Use for explicit return.
- Conventions: `?` suffix for boolean returns; `!` suffix for destructive or dangerous actions.

Blocks, Procs, and Lambdas

Ruby's powerful closure features.

- Block: `do ... end` or `{ ... }`.

```ruby
[1, 2, 3].each do |number|
  puts number
end
```

- `yield`: Execute the block passed to a method.
- Proc: A block turned into an object. `my_proc = Proc.new { |x| puts x }`.
- Lambda: Similar to Proc, but strict on arguments and handled `return` differently. `my_lambda = ->(x) { puts x }`.

Classes and Objects

Everything is an object.

- Class Definition:

```ruby
class MyClass
  def initialize(name)
    @name = name # Instance variable
  end

  def greet
    "Hello, #{@name}!"
  end
end
```

- Accessors:
  - `attr_reader :name` (getter)
  - `attr_writer :name` (setter)
  - `attr_accessor :name` (getter & setter)
- Inheritance: `class Derived < Base ... end`.
- Modules: Used for namespacing and mixins (simulating multiple inheritance).

```ruby
module MyModule
  def fly; puts "I'm flying!"; end
end
class Bird; include MyModule; end
```

Major Collections

Array

```ruby
arr = [1, "apple", 3.5]
arr << "new_item" # Add
first = arr[0] # Access
arr.each { |item| puts item } # Iterate
```

Hash

Key-value pairs (Dictionaries).

```ruby
my_hash = { "name" => "Jules", "age" => 30 }
my_hash = { name: "Jules", age: 30 } # Symbol keys (Ruby 1.9+)
```

Enumerable Module

Collection of iteration methods included in Array, Hash, etc.

- `map` / `collect`: Returns new array with block applied to each element.
- `select` / `filter`: Returns new array with elements where block is true.
- `reject`: Opposite of select.
- `reduce` / `inject`: Cumulative operation.

RubyGems and Bundler

- RubyGems: Package manager for Ruby.
  - `gem install <name>`: Install gem.
- Bundler: Manages project dependencies.
  - `Gemfile`: Specify gems.
  - `bundle install`: Install specified gems.
  - `bundle exec <cmd>`: Run command in the project context.

Exception Handling

```ruby
begin
  # Risky code
rescue SomeError => e
  puts "Error: #{e.message}"
rescue
  # Catch all others
ensure
  # Always executed
end
```