LESSON 7

app.rb

```ruby
module Greetable
  def greet = "Salaam"
end
class T; include Greetable; end
```

Ruby

Modules & Mixins

Learn it all @MyWebUniversity



A FREE MYWEBUNIVERSITY COURSE

RUBY PROGRAMMING

LESSON 7

Inheritance, Modules and Mixins

Subclasses, super, modules as namespaces, include, extend and duck typing

Inheritance

animals.rb

```ruby
class Animal
  def initialize(name) = @name = name
  def speak = "..."
  def to_s = "#{@name} says #{speak}"
end
class Dog < Animal
  def speak = "Woof"
end
class Puppy < Dog
  def speak = super + "!"
end
puts Puppy.new("Rex")      # Rex says Woof!
```

- A class inherits from another with the less-than sign. The subclass gets all the parent's ...
- super calls the parent's version of the same method. With no parentheses, it passes along ...

01

Duck Typing

duck.rb

```ruby
class Robot
  def speak = "Beep"
end

[Dog.new("Rex"), Robot.new].each do |thing|
  puts thing.speak             # no common parent needed
end

Robot.new.respond_to?(:speak)  # true
```

- Ruby does not care about an object's class, only whether it responds to the method you ...
- respond_to? asks whether an object has a method.

02

Modules as Namespaces

geometry.rb

```ruby
module Geometry
  PI2 = Math::PI * 2

  def self.circumference(r) = PI2 * r

  class Circle
    def initialize(r) = @r = r
  end
end
Geometry.circumference(1)     # 6.283...
Geometry::Circle.new(2)
```

- A module groups related methods and constants under one name, so different libraries do not …
- Use two colons to reach inside a module, like Math colon colon PI.

03

Mixins with include

mixins.rb

```ruby
module Greetable
  def greet = "Salaam, I am #{name}"
end
module Printable
  def print_card = puts("[#{self}]")
end

class Teacher
  include Greetable, Printable
  attr_reader :name
  def initialize(name) = @name = name
end
Teacher.new("Ahmad").greet
```

- Ruby has single inheritance, but a class can include many modules. Their methods become ...
- extend adds a module's methods to one object or to the class itself.

04

Include Enumerable

classroom.rb

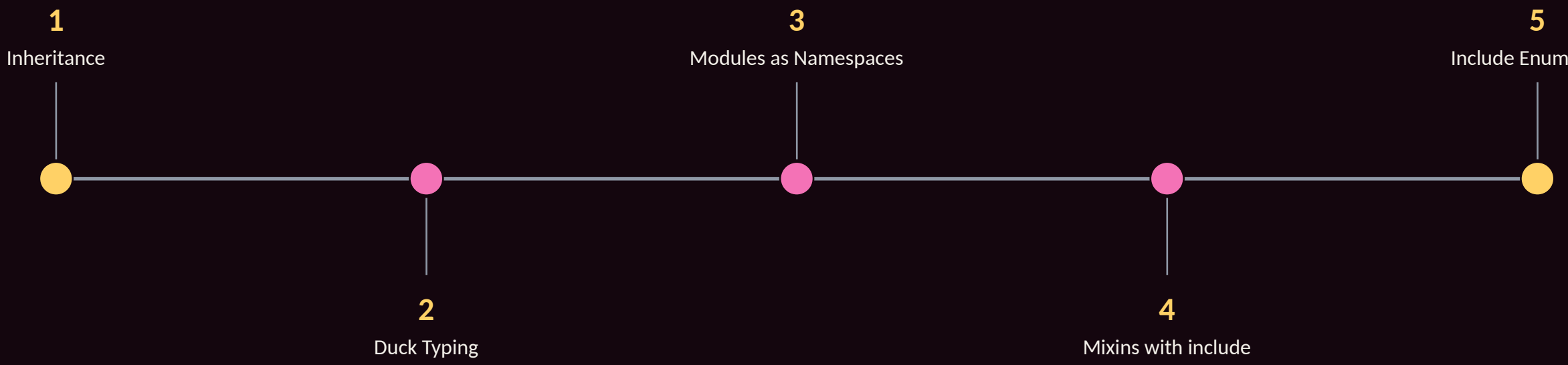
```ruby
class Classroom
  include Enumerable
  def initialize(*students) = @students = students
  def each(&block) = @students.each(&block)
end

room = Classroom.new("Zahra", "Ahmad", "Mariam")
room.sort            # ["Ahmad", "Mariam", "Zahra"]
room.select { |n| n.start_with?("M") }
Classroom.ancestors  # [Classroom, Enumerable, Object, ...]
```

- The most powerful mixin is Enumerable. Define just one method, each, include Enumerable, ...
- ancestors shows the full lookup chain of classes and modules.

05

Lesson Map


1
Inheritance
2
Duck Typing
3
Modules as Namespaces
4
Mixins with include
5
Include Enumerable

Sources and Further Reading

animals.rb

```ruby
class Animal
  def initialize(name) = @name = name
  def speak = "..."
  def to_s = "#{@name} says #{speak}"
end
class Dog < Animal
  def speak = "Woof"
end
class Puppy < Dog
  def speak = super + "!"
end
puts Puppy.new("Rex")      # Rex says Woof!
```

RUBY PROGRAMMING · 7

geometry.rb

```ruby
module Geometry
  PI2 = Math::PI * 2

  def self.circumference(r) = PI2 * r

  class Circle
    def initialize(r) = @r = r
  end
end
Geometry.circumference(1)     # 6.283...
Geometry::Circle.new(2)
```

Thank you for watching

MyWebUniversity.com

Free technical education since 2000 — English and Dari

@MyWebUniversity · @MyWebUniversity-Dari