Inheritance and Composition

Extending Behavior

Author: Torey Hickman

Phase 1: Day 7

Inheritance and Composition

- Why
- How it works

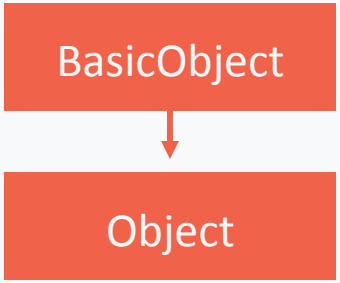
Why Inherit and Composition

- Don’t repeat yourself
- Gain functionality while maintaining a separation of concerns

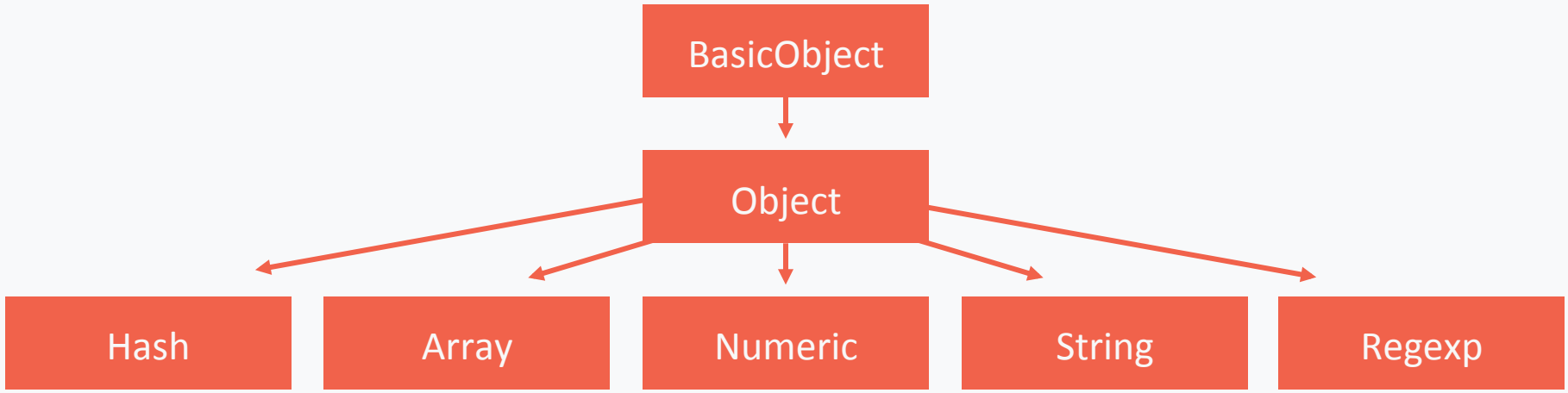
Hierarchy of Classes, Abridged

BasicObject

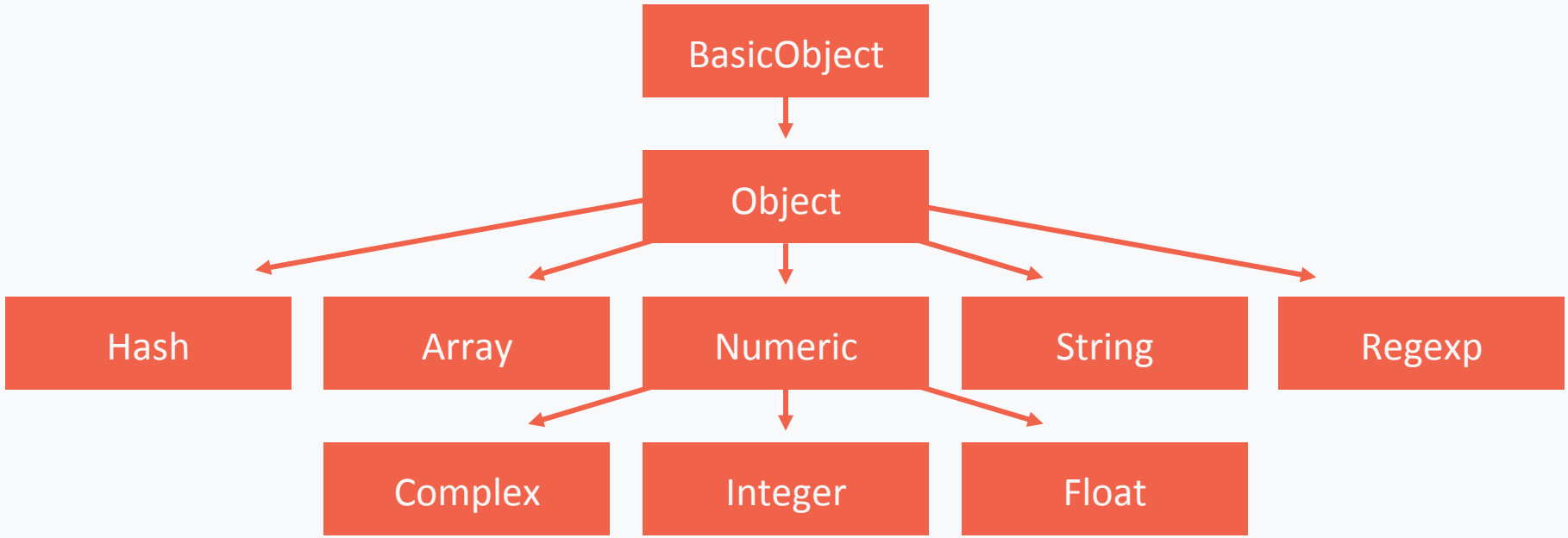
Hierarchy of Classes, Abridged



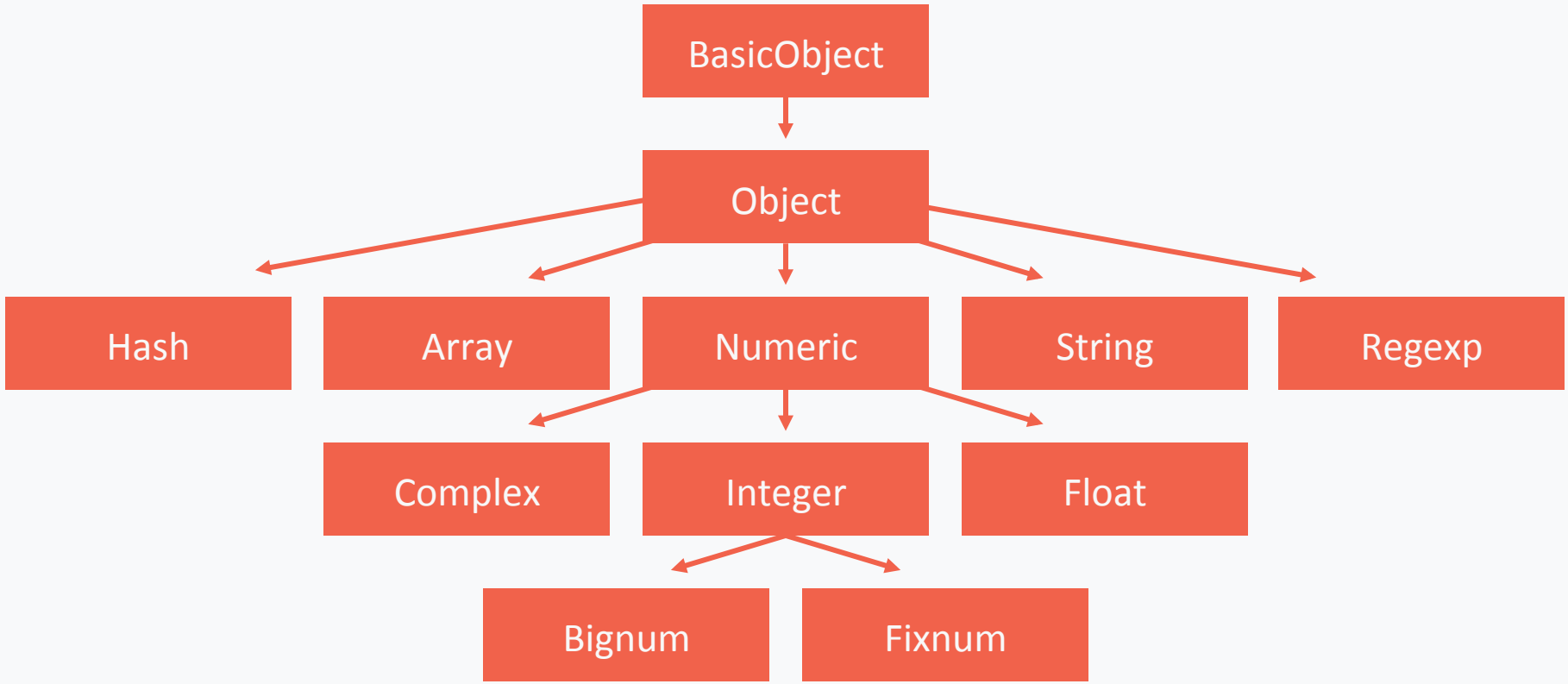
Hierarchy of Classes, Abridged



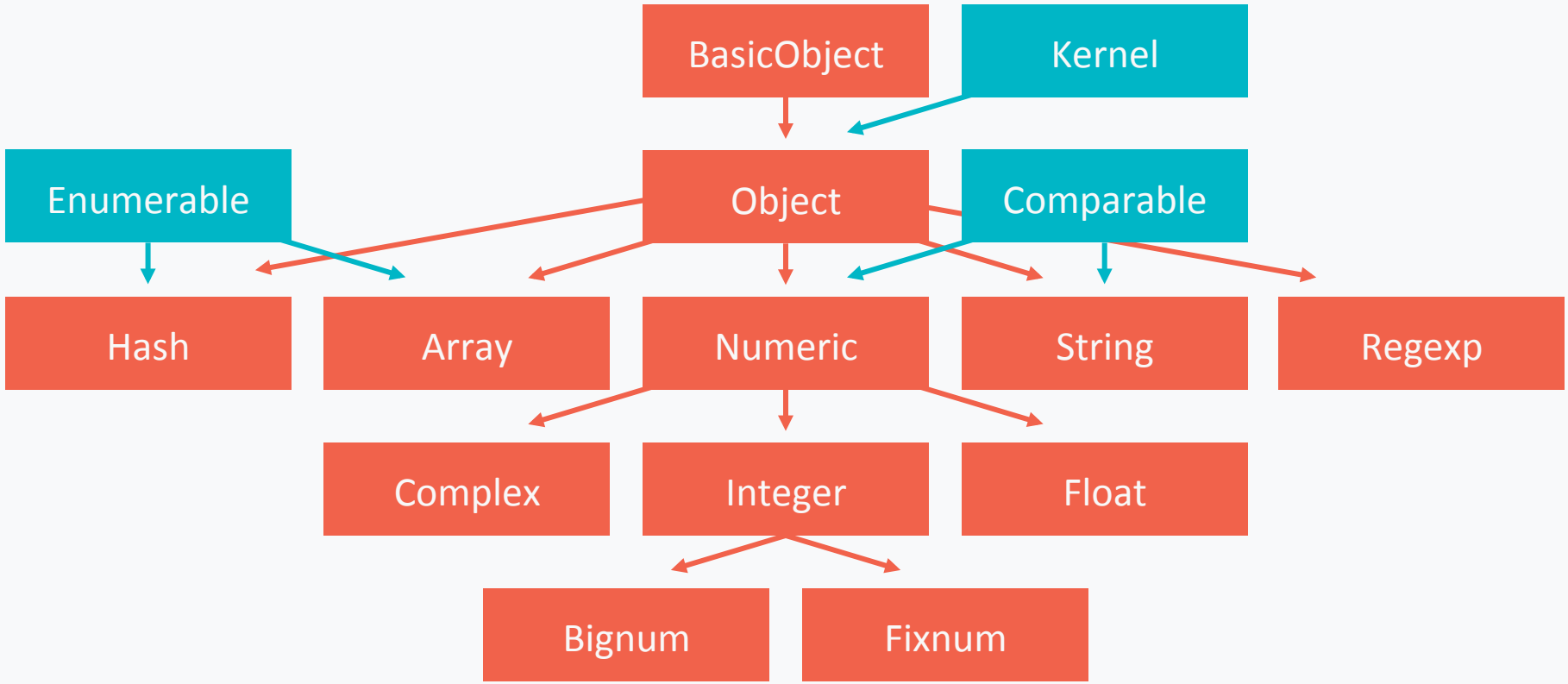
Hierarchy of Classes, Abridged



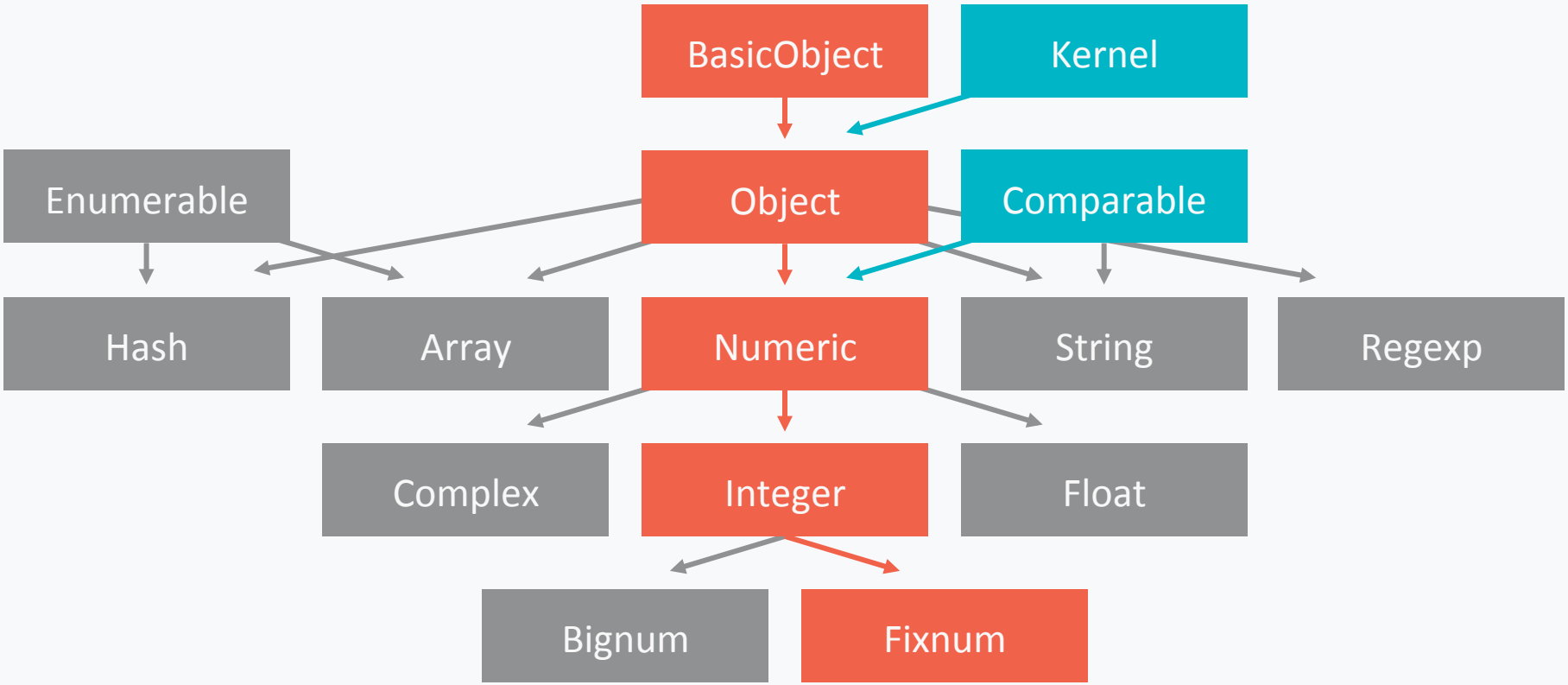
Hierarchy of Classes, Abridged



Hierarchy with Modules



Ancestry of Fixnum



Ancestry of Fixnum

```
Fixnum.ancestors
```

Ancestry of Fixnum

```
Fixnum.ancestors
# => [Fixnum,
      Integer,
      Numeric,
      Comparable,
      Object,
      Kernel,
      BasicObject]
```

Fixnum and #object_id

```
my_fixnum = 5
my_fixnum.object_id
# => 11
```

Fixnum and #object_id

```
my_fixnum = 5
my_fixnum.object_id
# => 11

my_fixnum.class
# => Fixnum
```

Fixnum and #object_id

```
my_fixnum = 5
my_fixnum.object_id
# => 11

my_fixnum.class
# => Fixnum

Fixnum.instance_methods(false)
```

Fixnum and #object_id

```
my_fixnum = 5
my_fixnum.object_id
# => 11

my_fixnum.class
# => Fixnum

Fixnum.instance_methods(false)
# => [:to_s, :-@, :+, :-, :*, :/, :div, :%, :modulo, :divmod,
      :fdiv, :**, :abs, :magnitude, :==, :===, :<=>, :>, :>=, :<,
      :<=, :~, :&, :|, :^, :[], :<<, :>>, :to_f, :size, :zero?,
      :odd?, :even?, :succ]
```

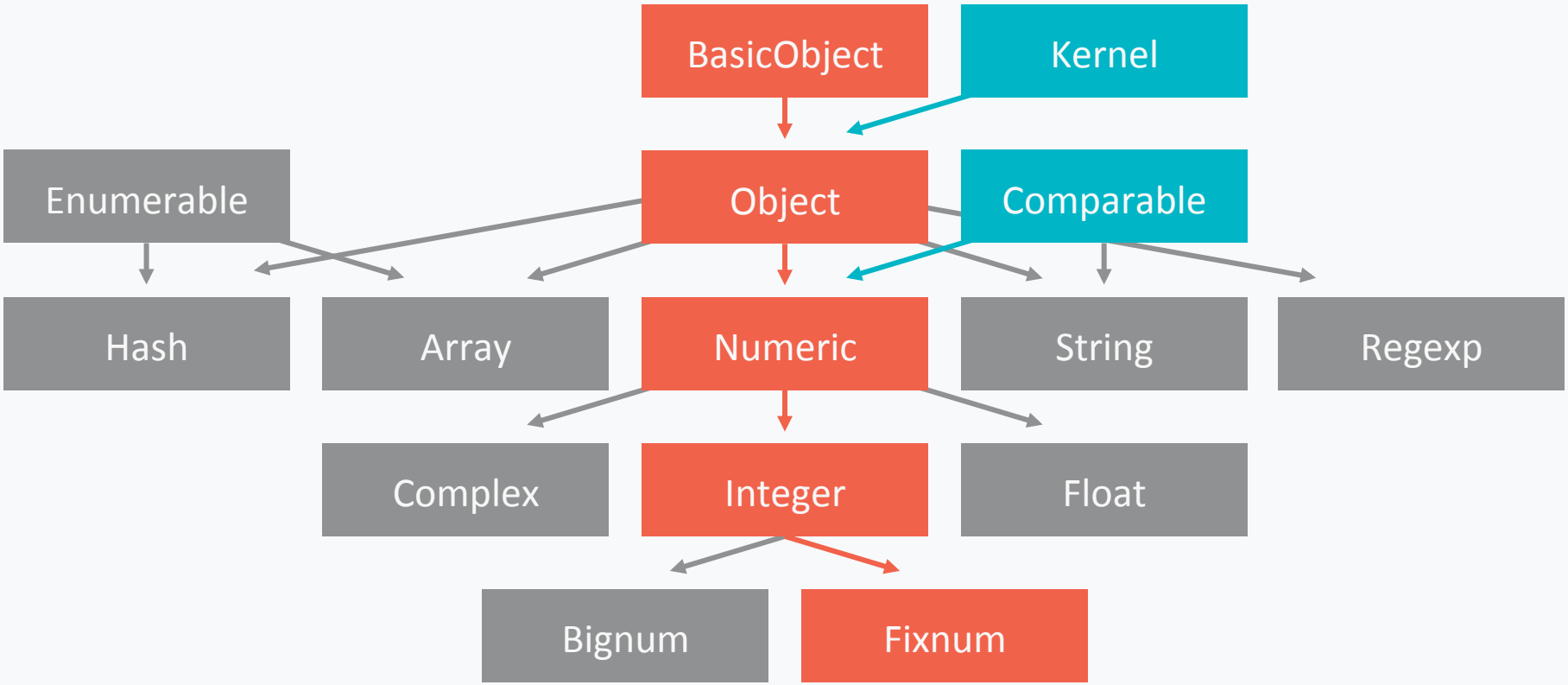
Fixnum and #object_id

```
my_fixnum = 5
my_fixnum.object_id
# => 11

Fixnum.ancestors.find do |ancestor_class|
  ancestor_class.instance_methods(false).include?(:object_id)
end
```

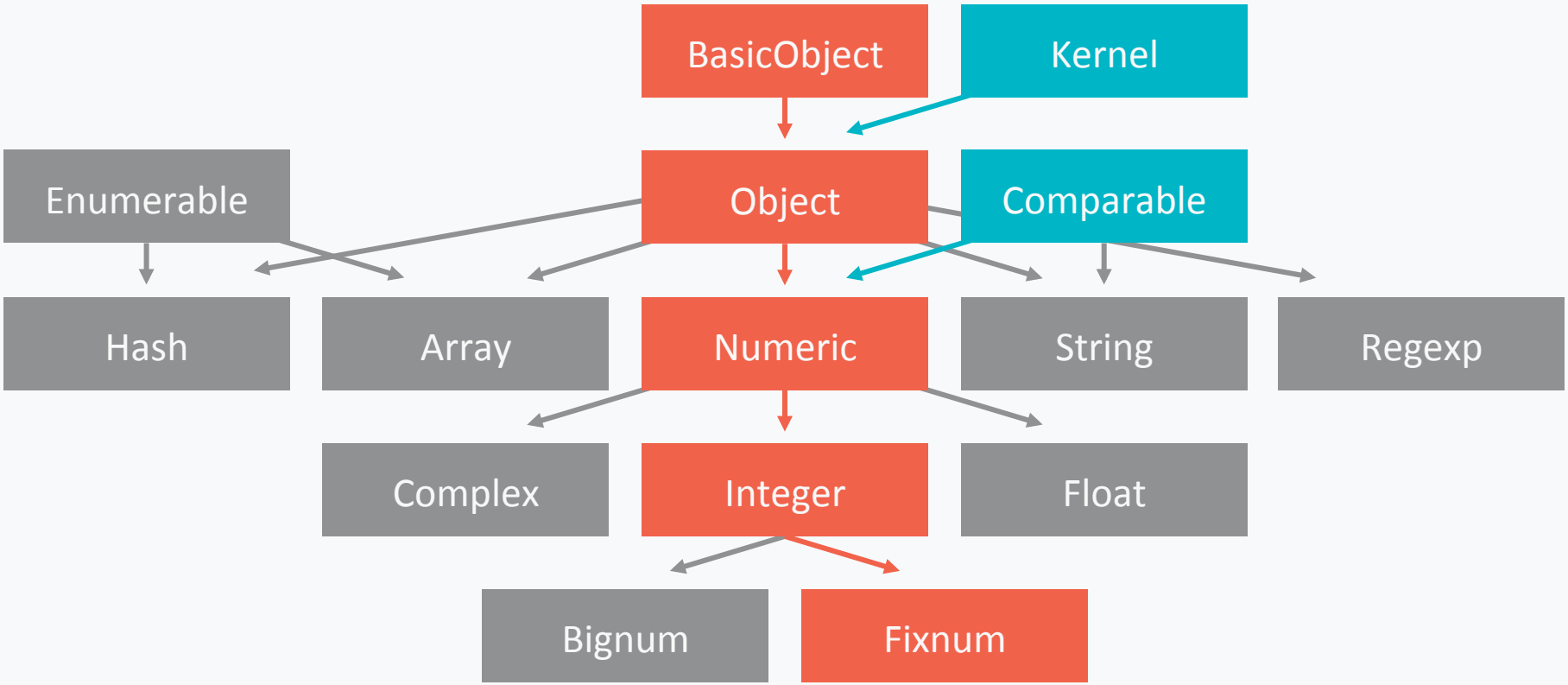
Fixnum and #object_id

```
my_fixnum = 5
my_fixnum.object_id
# => 11

Fixnum.ancestors.find do |ancestor_class|
  ancestor_class.instance_methods(false).include?(:object_id)
end
# => Kernel
```

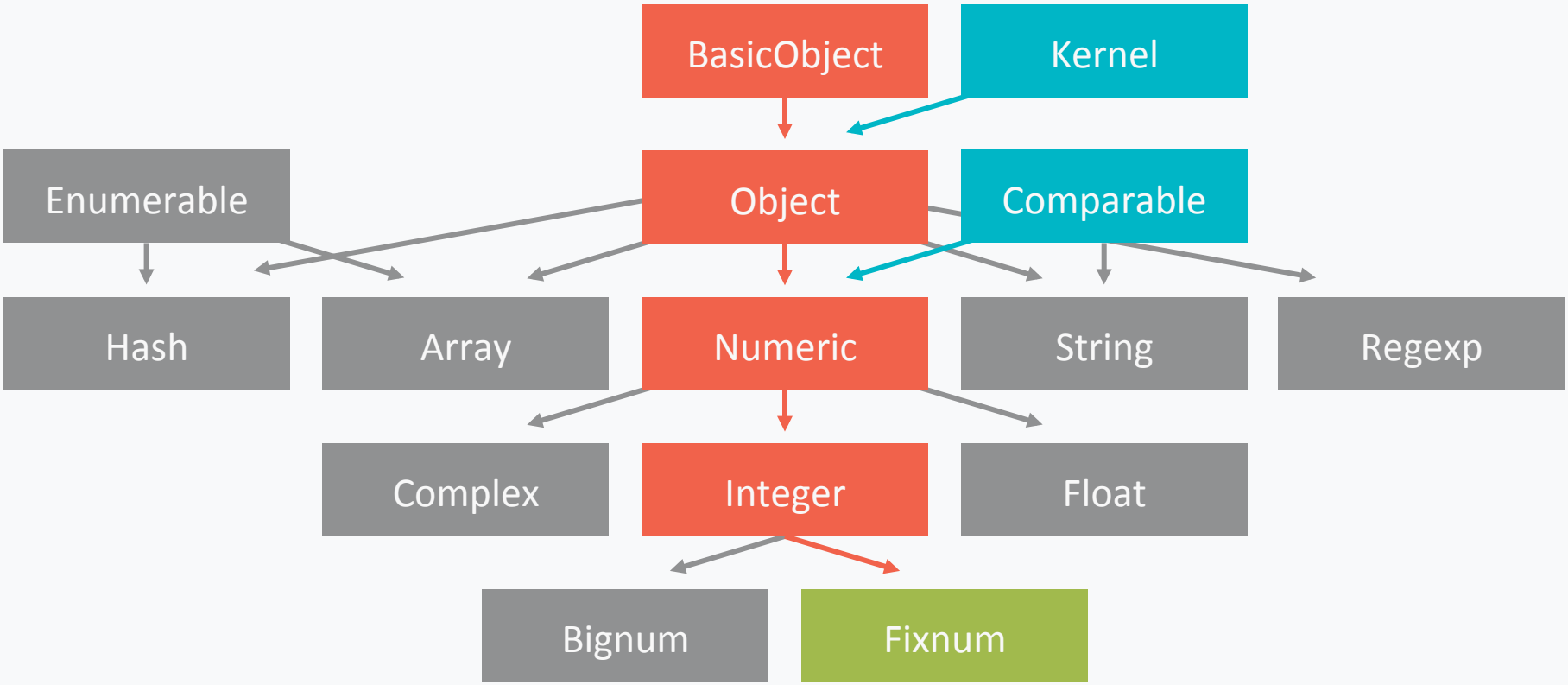
Fixnum and #object_id



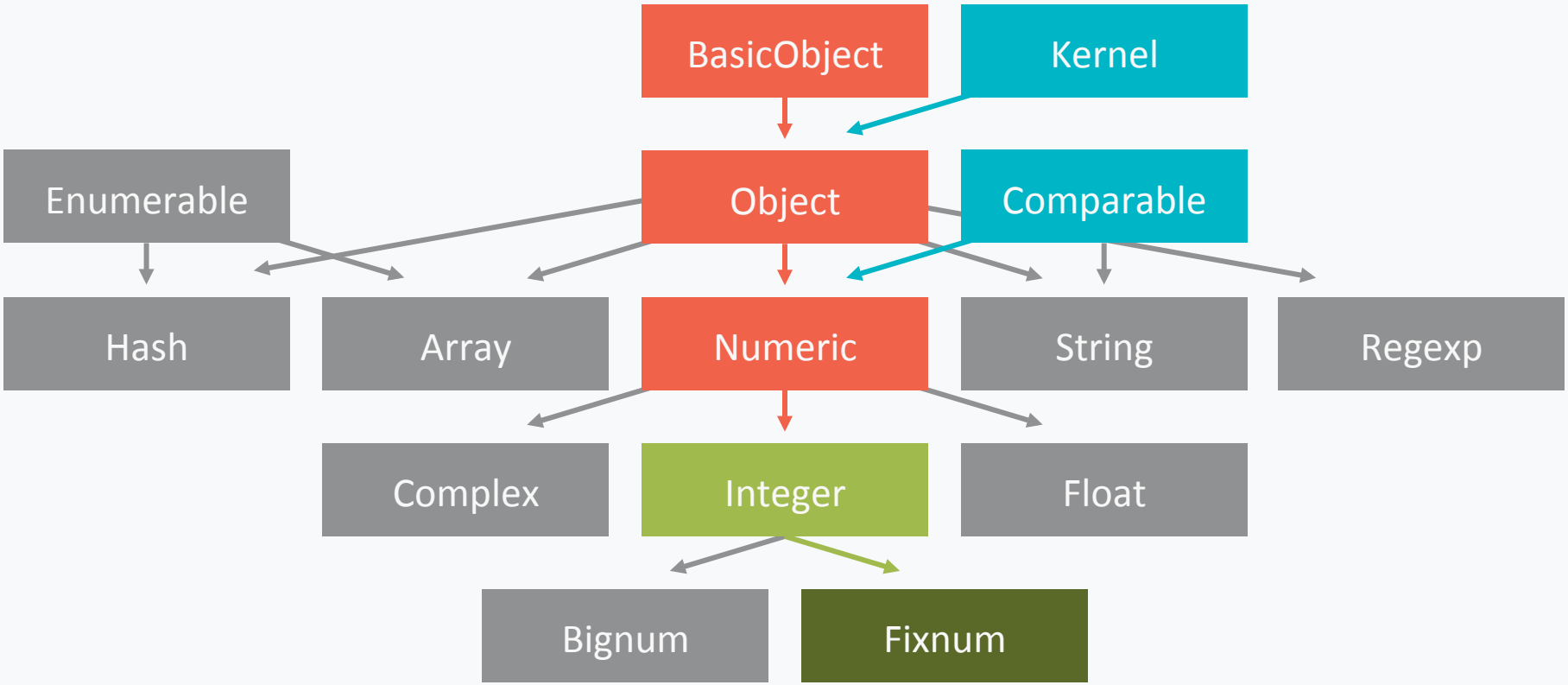
Fixnum and #object_id



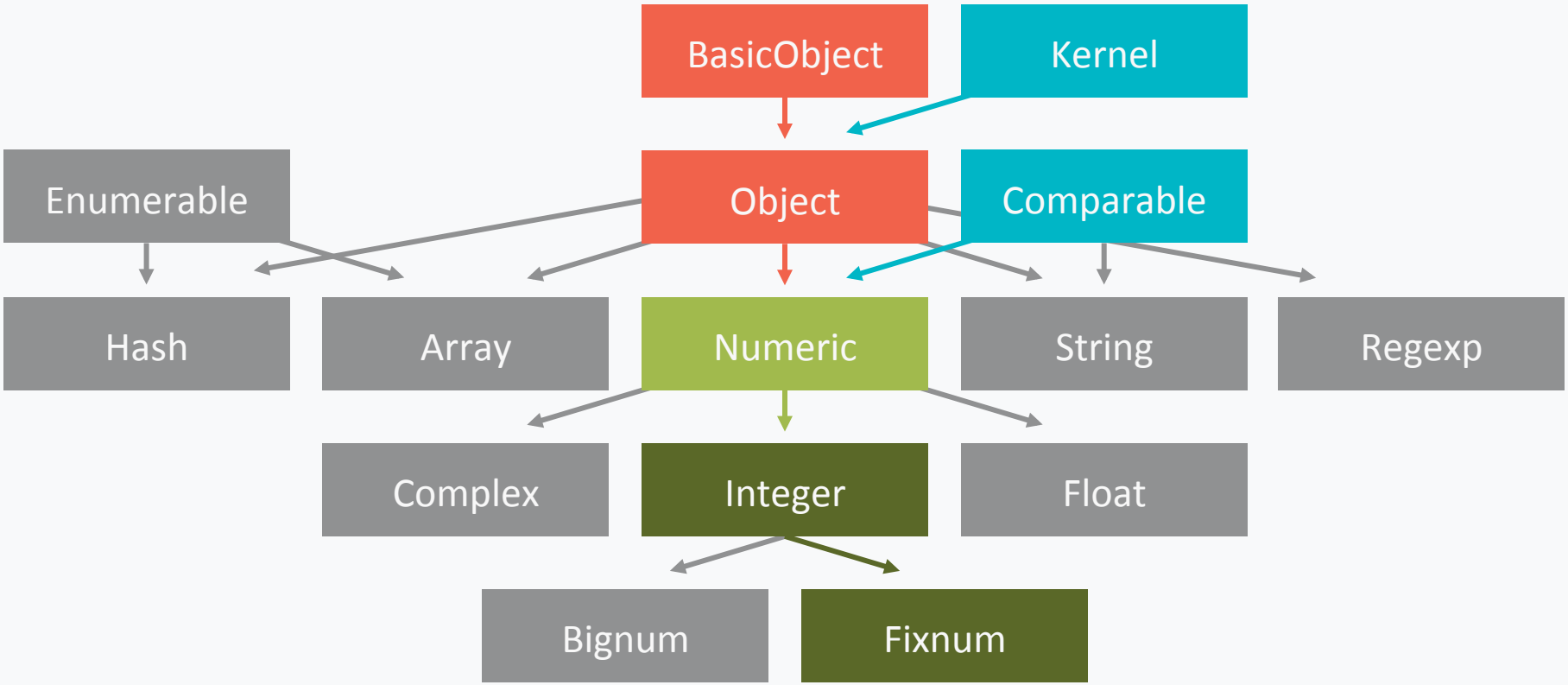
Fixnum and #object_id



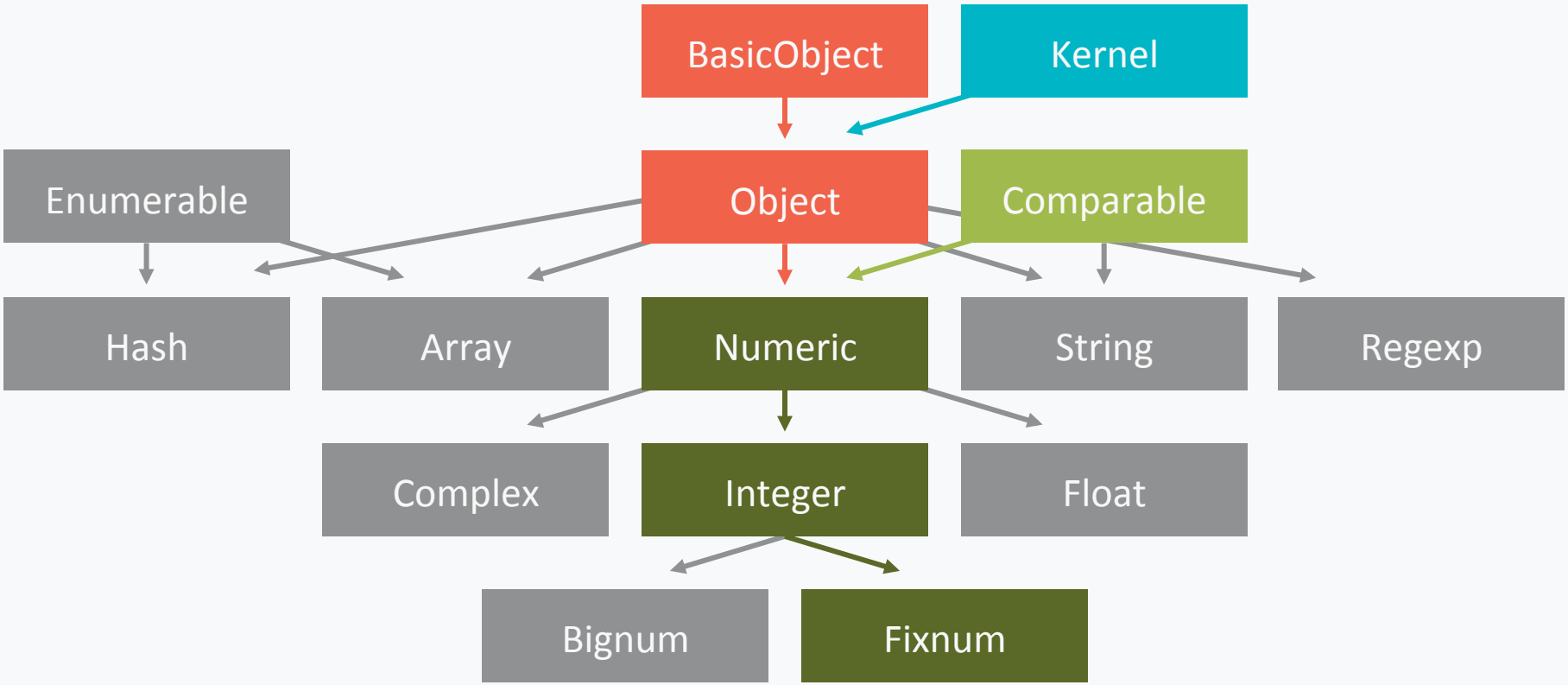
Fixnum and #object_id



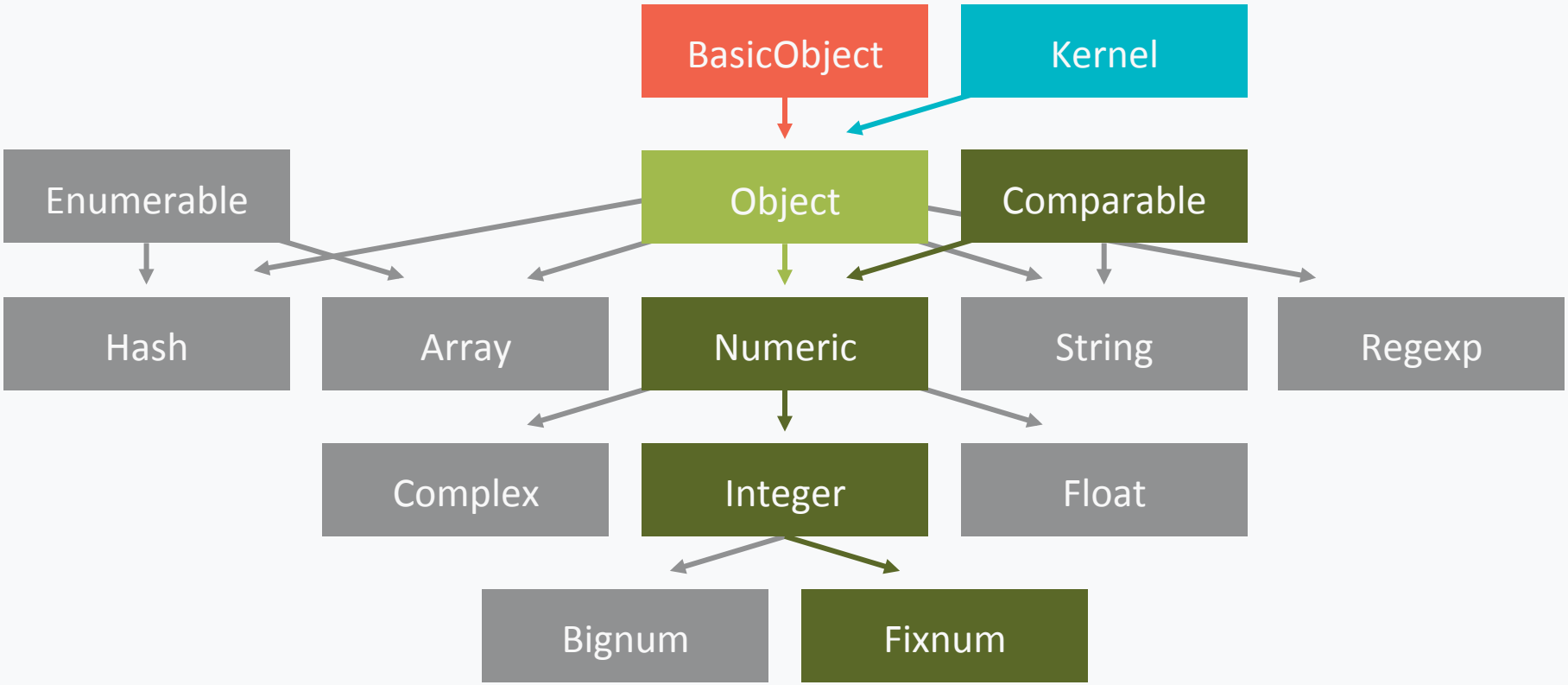
Fixnum and #object_id



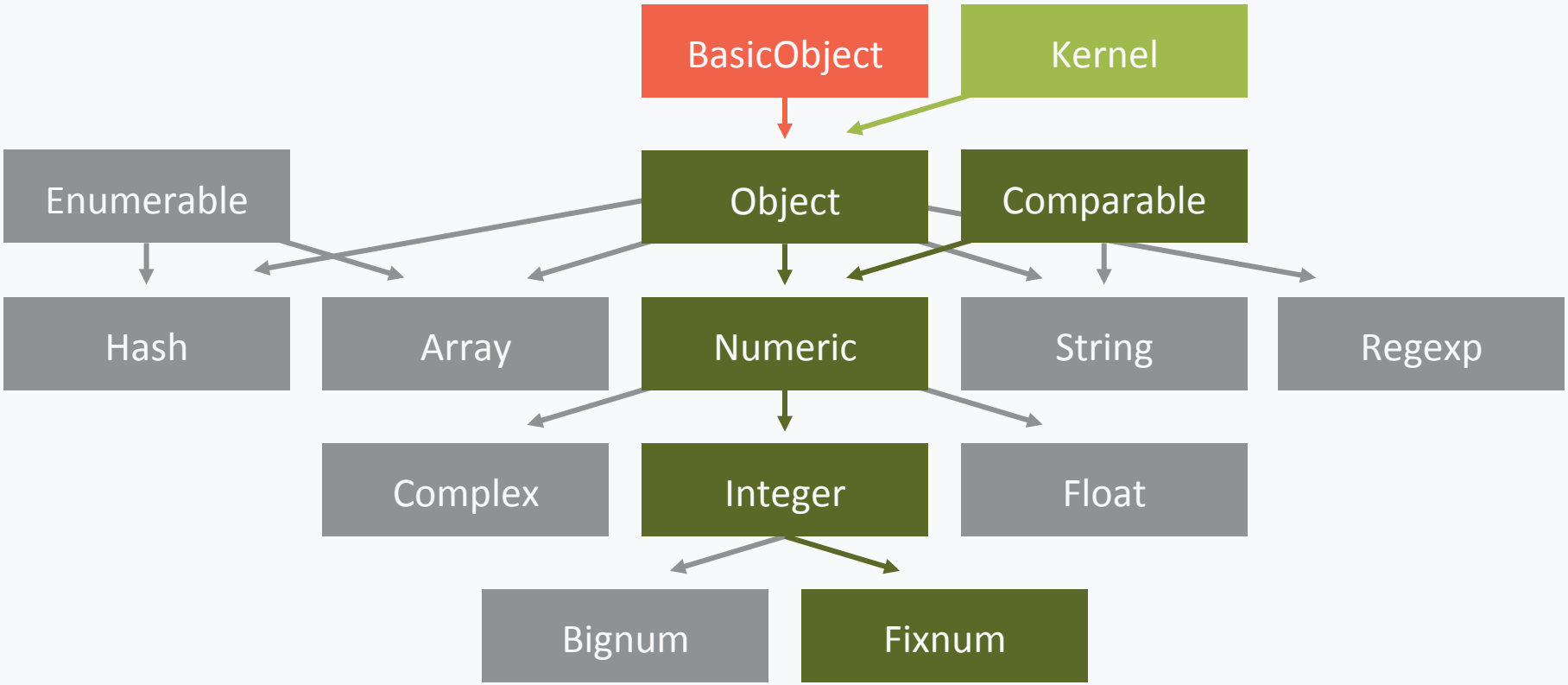
Fixnum and #object_id



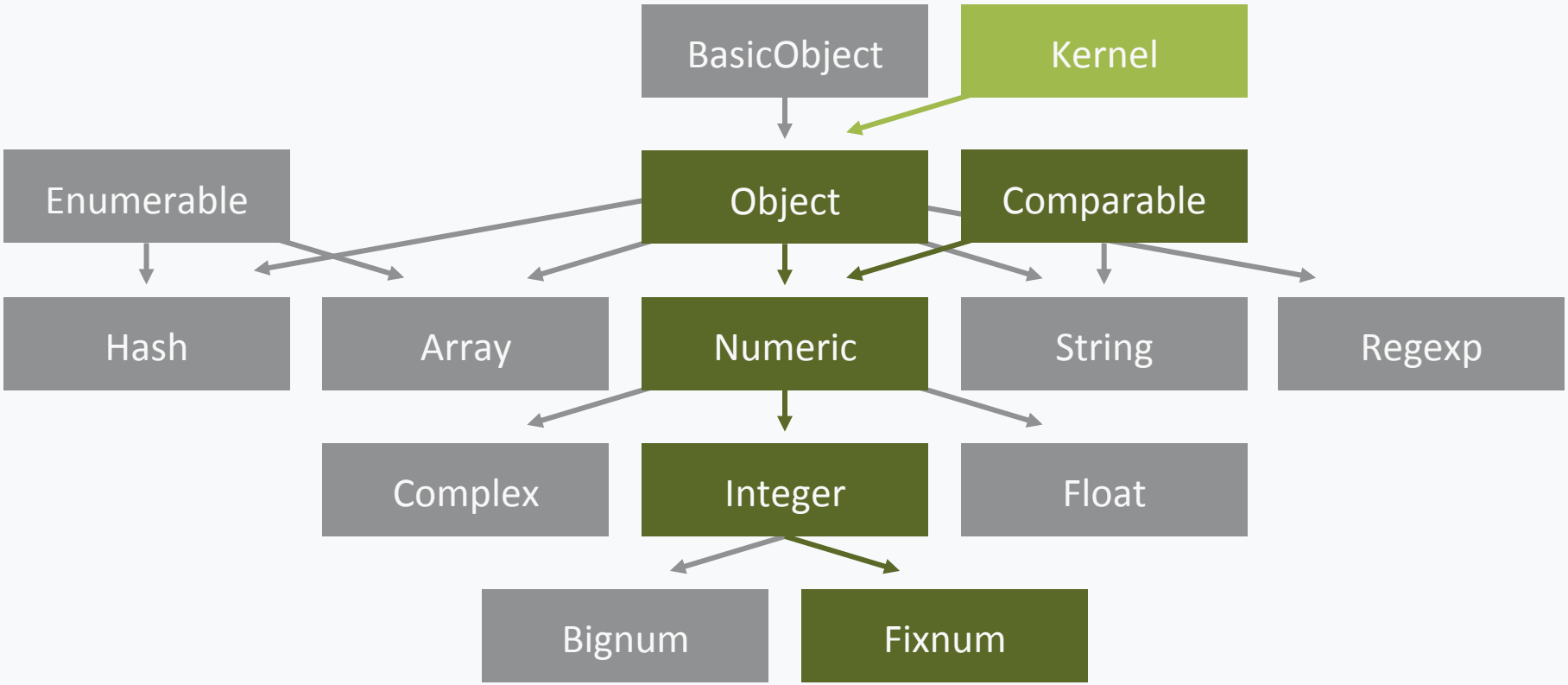
Fixnum and #object_id



Fixnum and #object_id



Fixnum and #object_id



NoMethodError

```
my_fixnum = 5

my_fixnum.some_other_method
```

NoMethodError

```
my_fixnum = 5

my_fixnum.some_other_method
# => NoMethodError:
     undefined method `some_other_methos' for 5:Fixnum
```

NoMethodError

- If no ancestor classes have the instance method, call #method_missing …

NoMethodError

- If no ancestor classes have the instance method, call #method_missing …
- and start back up the ancestor chain

Custom Classes and Modules

```
module LifeCycle
  def birthday
    self.age = age + 1
  end
end
```

```
class Animal
  include LifeCycle

  attr_accessor :age
  def initialize
    @age = 0
  end
end

class Mammal < Animal
end

class Dog < Mammal
end
```

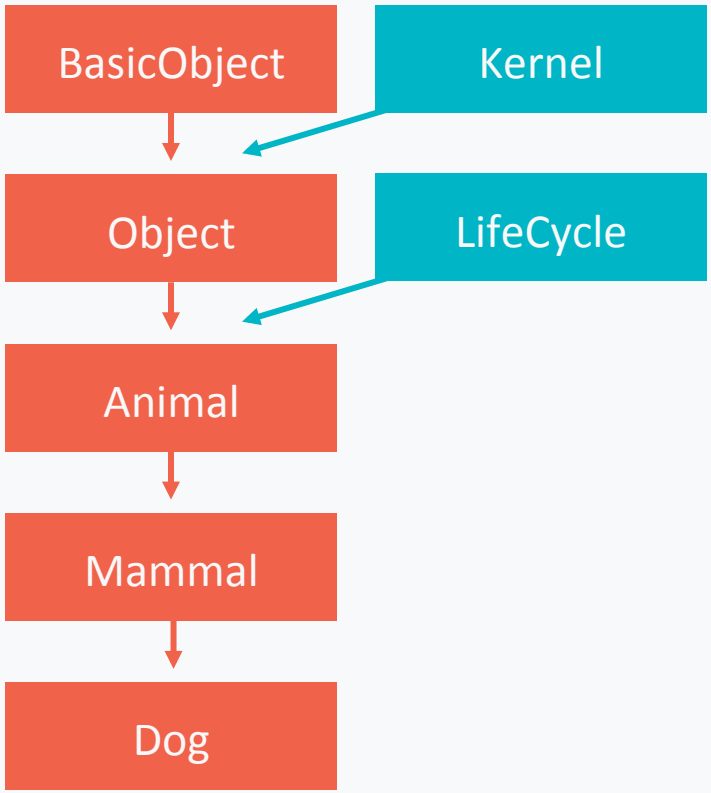
Custom Classes and Modules

```
Dog.ancestors
```

Custom Classes and Modules

```
Dog.ancestors
# => [Dog,
      Mammal,
      Animal,
      LifeCycle,
      Object,
      Kernel,
      BasicObject]
```

Custom Classes and Modules



Custom Classes and Modules

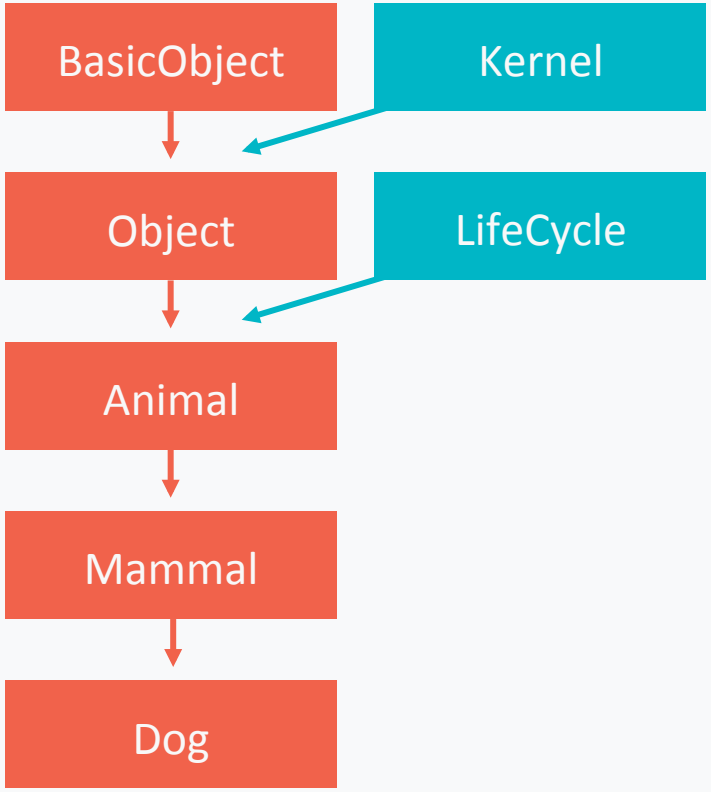
```
tenley = Dog.new
```

Custom Classes and Modules

```
tenley = Dog.new
# => #<Dog:0x007f9db226bef0 @age=0>
```

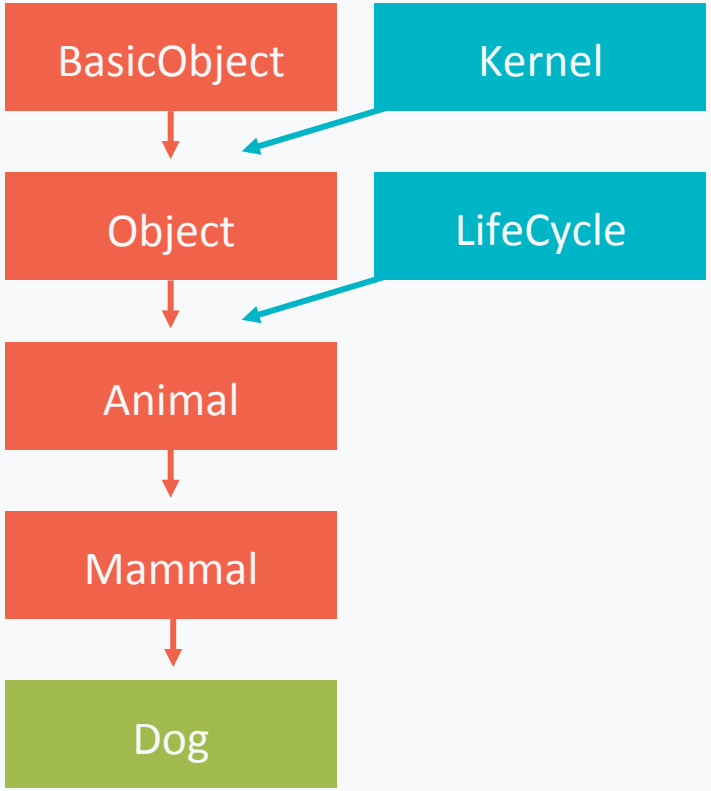
Where is #initialize

```
class Animal
  include LifeCycle

  attr_accessor :age
  def initialize
    @age = 0
  end
end

class Mammal < Animal
end

class Dog < Mammal
end
```



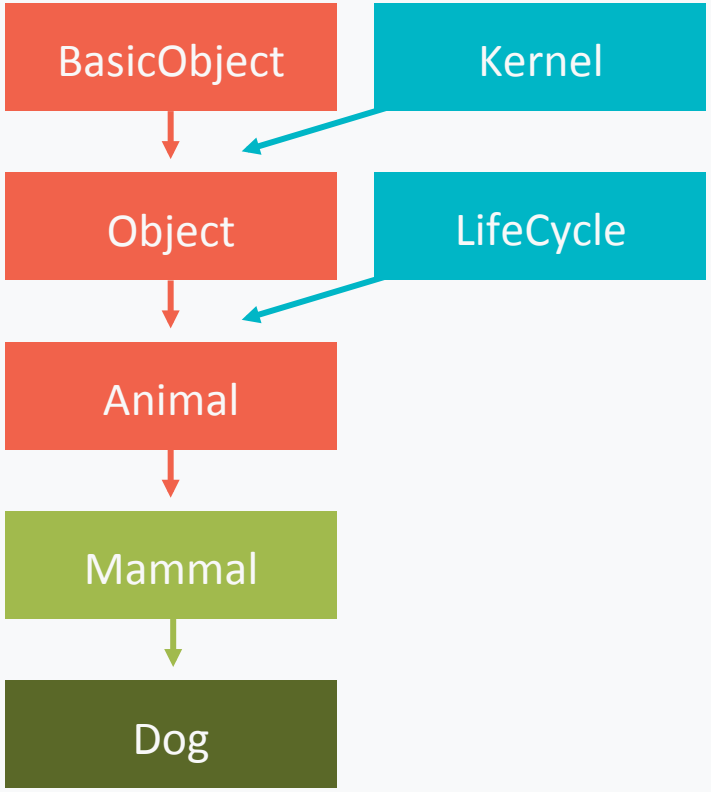
Where is #initialize

```
class Animal
  include LifeCycle

  attr_accessor :age
  def initialize
    @age = 0
  end
end

class Mammal < Animal
end

class Dog < Mammal
end
```



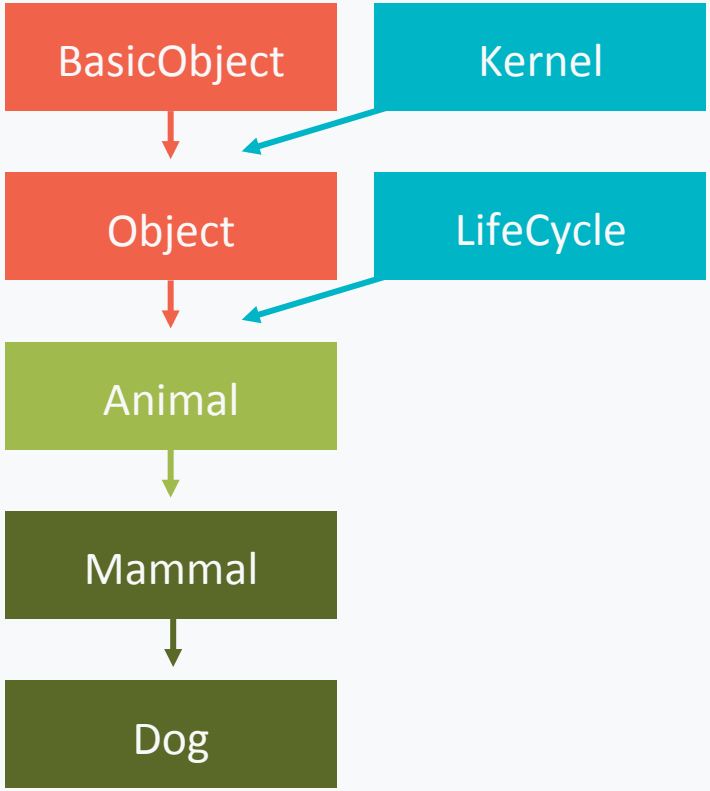
Where is #initialize

```
class Animal
  include LifeCycle

  attr_accessor :age
  def initialize
    @age = 0
  end
end

class Mammal < Animal
end

class Dog < Mammal
end
```



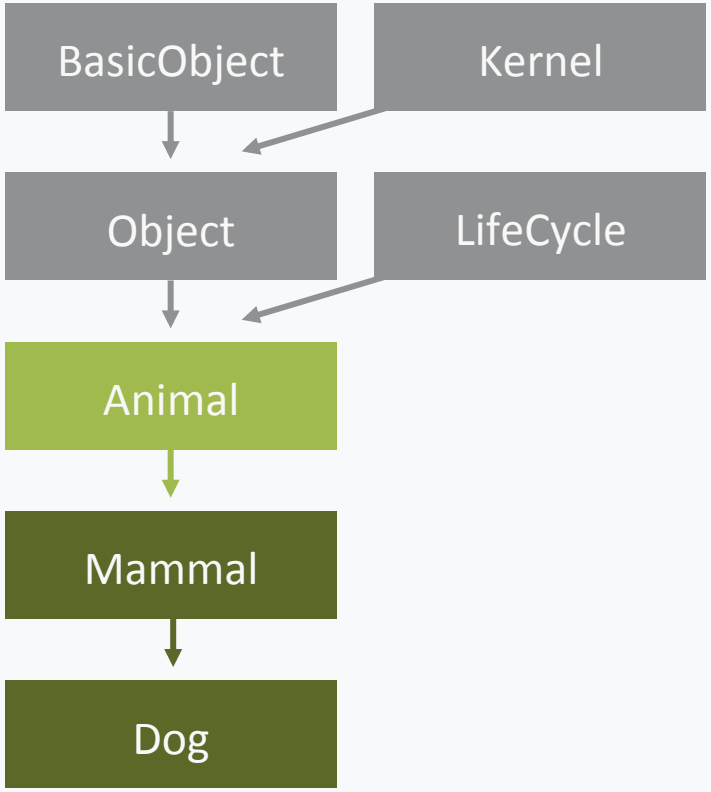
Where is #initialize

```
class Animal
  include LifeCycle

  attr_accessor :age
  def initialize
    @age = 0
  end
end

class Mammal < Animal
end

class Dog < Mammal
end
```



Where is #initialize

```
class Animal
  include LifeCycle

  attr_accessor :age
  def initialize
    @age = 0
  end
end

class Mammal < Animal
end

class Dog < Mammal
end
```

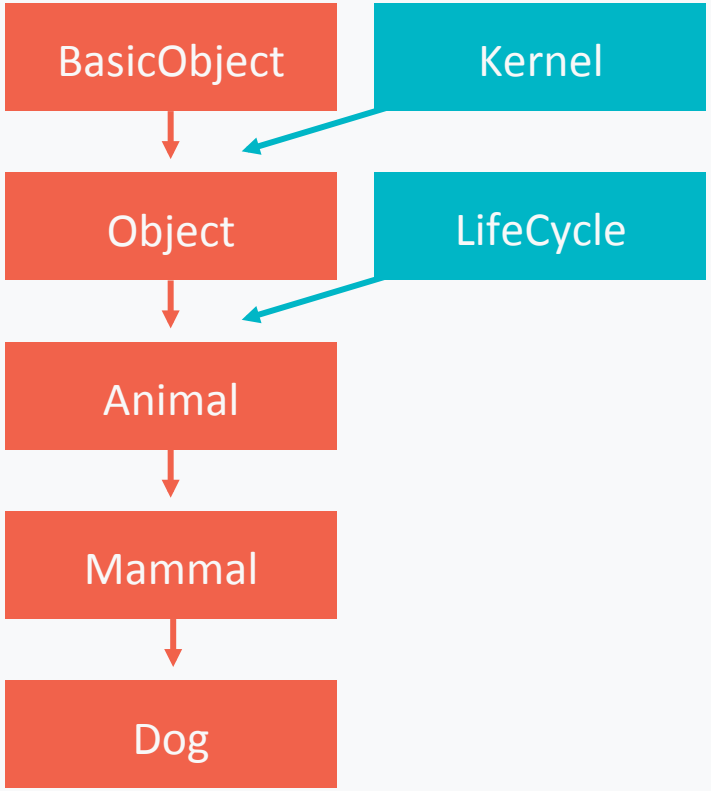


Custom Classes and Modules

```
tenley = Dog.new
# => #<Dog:0x007f9db226bef0 @age=0>

tenly.birthday
# => 1

tenley
# => #<Dog:0x007f9db226bef0 @age=1>
```

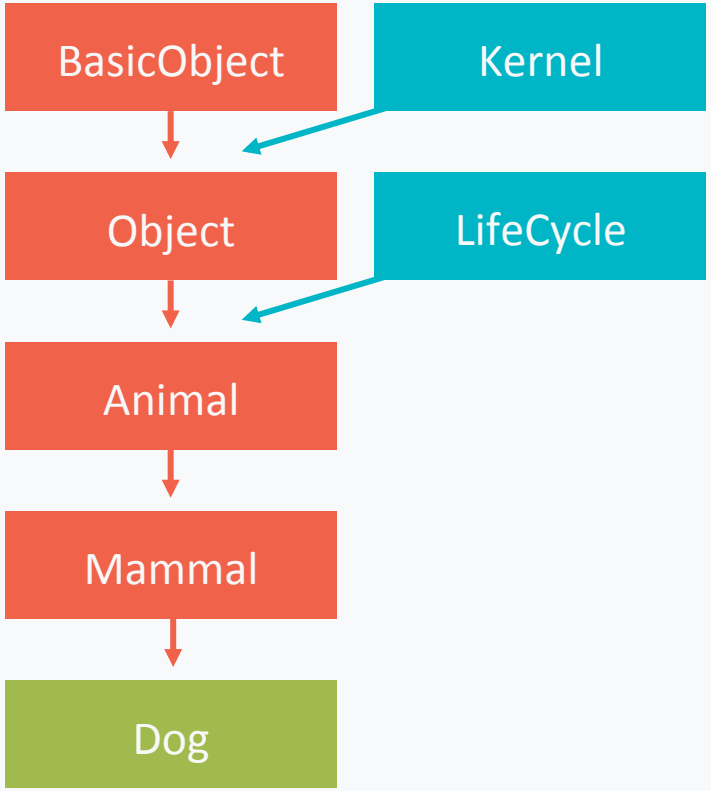
Where is #birthday

```
module LifeCycle
  def birthday
    self.age = age + 1
  end
end

class Animal
  include LifeCycle

  # ...
end

class Mammal < Animal; end

class Dog < Mammal; end
```



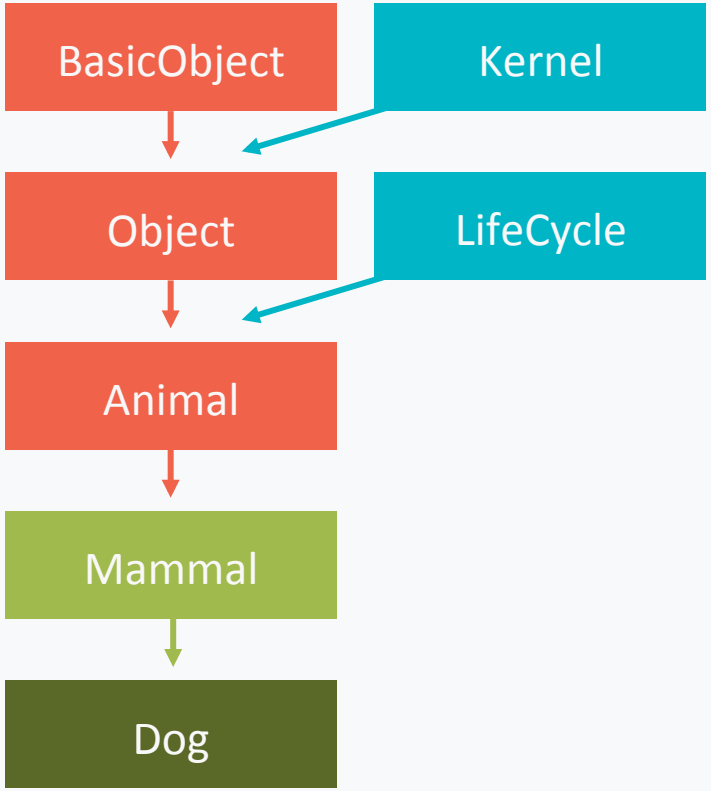
Where is #birthday

```
module LifeCycle
  def birthday
    self.age = age + 1
  end
end

class Animal
  include LifeCycle

  # ...
end

class Mammal < Animal; end

class Dog < Mammal; end
```



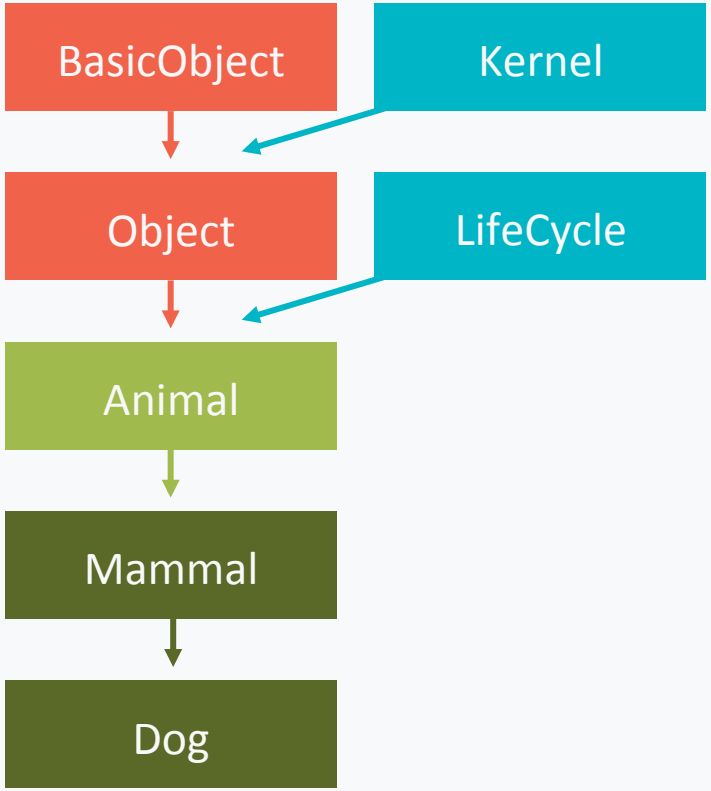
Where is #birthday

```
module LifeCycle
  def birthday
    self.age = age + 1
  end
end

class Animal
  include LifeCycle

  # ...
end

class Mammal < Animal; end

class Dog < Mammal; end
```



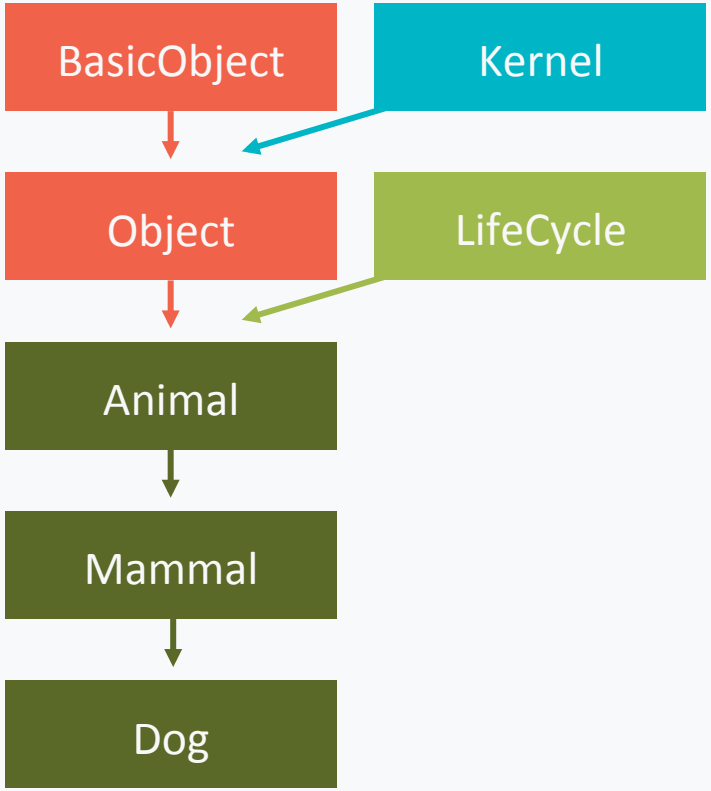
Where is #birthday

```
module LifeCycle
  def birthday
    self.age = age + 1
  end
end

class Animal
  include LifeCycle

  # ...
end

class Mammal < Animal; end

class Dog < Mammal; end
```



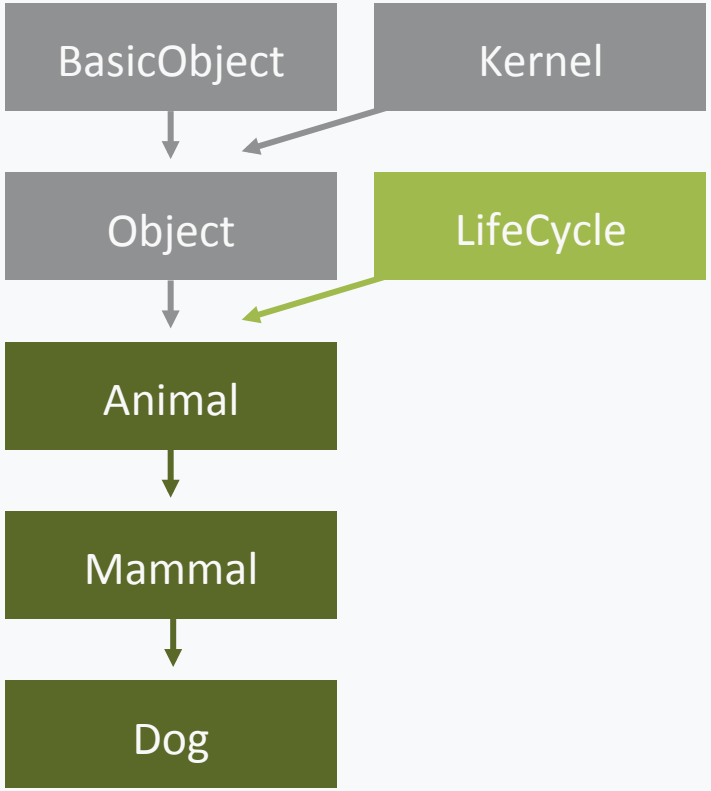
Where is #birthday

```
module LifeCycle
  def birthday
    self.age = age + 1
  end
end

class Animal
  include LifeCycle

  # ...
end

class Mammal < Animal; end

class Dog < Mammal; end
```



Where is #birthday

```
module LifeCycle
  def birthday
    self.age = age + 1
  end
end

class Animal
  include LifeCycle

  # ...
end

class Mammal < Animal; end

class Dog < Mammal; end
```



Inheritance vs Composition

- Both add behaviors and DRY code

Favor Composition

- Favor including modules to extend behavior
  - more flexible
  - test modules in isolation
  - include as many as wanted
  - workaround for single inheritance

Use Inheritance

- A “type of” relationship
- You have to