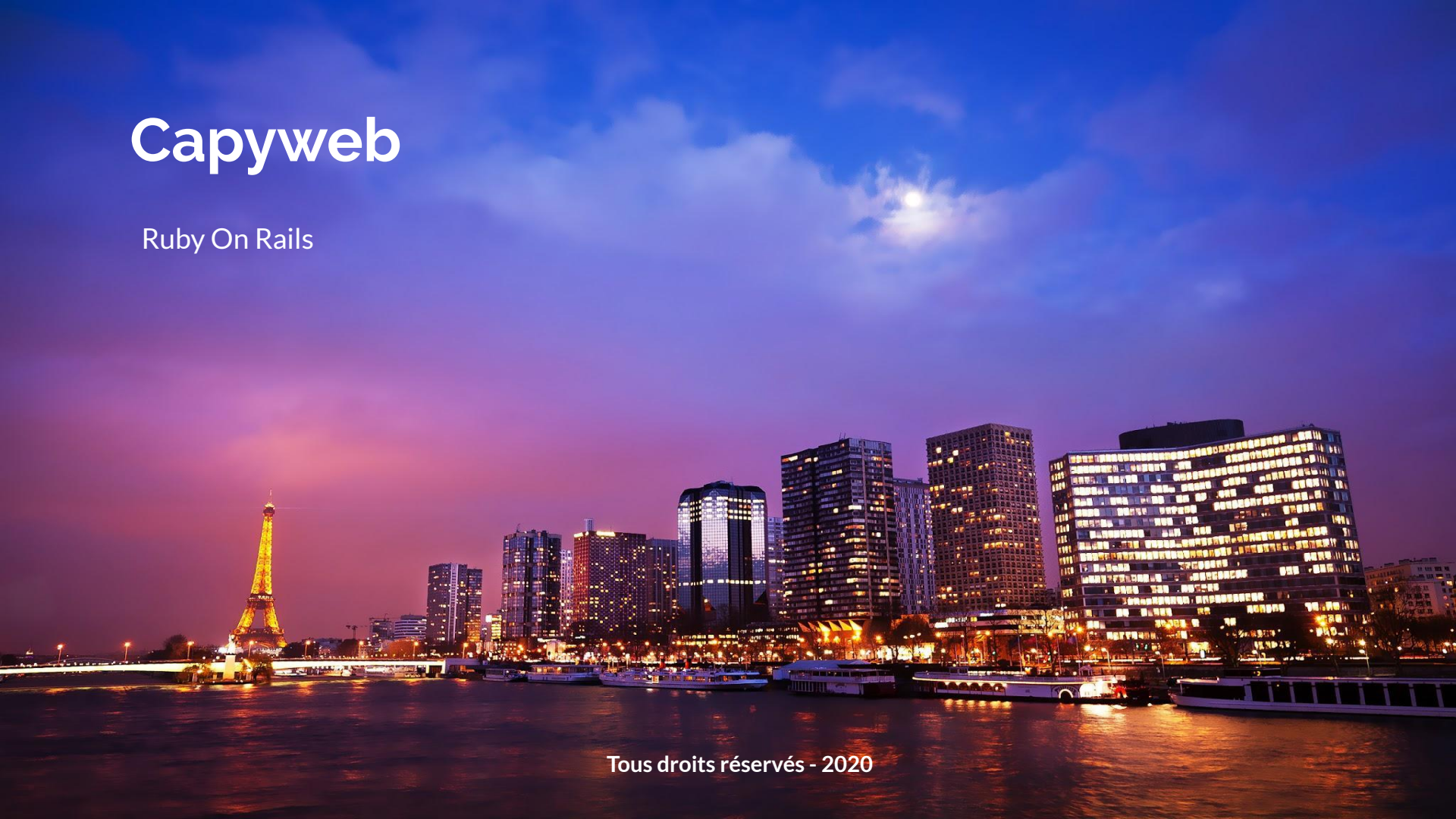
Capyweb

Ruby On Rails

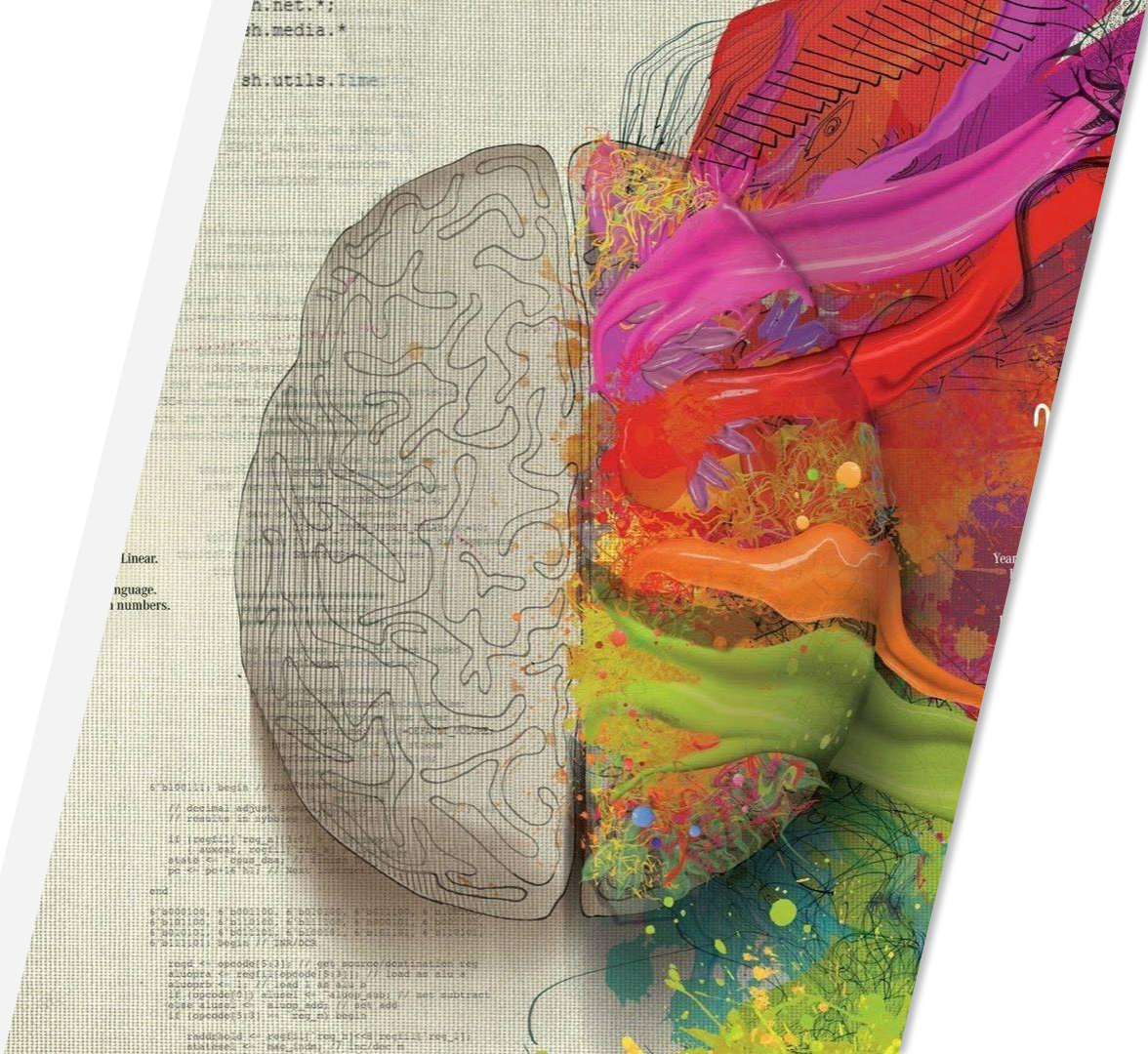


Capyweb

Une équipe expérimentée

Une agence née de liens tissés depuis 3 années entre chacun de ses membres.

Des projets déjà menés à ses termes avec expertise.



UN TRAVAIL D'ÉQUIPE AVEC

tous nos partenaires et contacts

- ➔ Développement de site, application hybride et outils sur mesure en: PHP, JS, Ruby.
- ➔ Notre entité donne des formations dans les écoles techs.
- ➔ Gestion du club GESEntrepreneurs (Facebook, LinkedIn) avec des afterworks et création de liens entre les entrepreneurs.
- ➔ Aides bénévole sur le concours EngrainaGES (concours d'entrepreneuriat pour les projets early-stage).

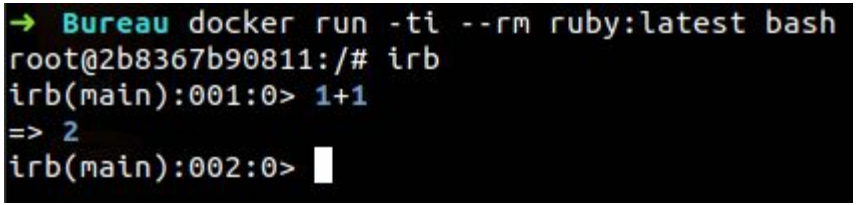
Ruby

Qu’est-ce que Ruby ?

- ➔ Créé en 1993 par Yukihiro Matsumoto
- ➔ Version actuel : 2.7.1
- ➔ Langage interprété
- ➔ Orienté objet
- ➔ Typage fort et dynamique
- ➔ Flexibilité : réécrire à la volée des fonction du langage
- ➔ Mots clé simples en anglais

Installation de Ruby (Docker)

docker run -ti --rm ruby:latest bash



Ou sinon:

https://www.ruby-lang.org/fr/documentation/installation/

Syntaxe

```ruby
age = 19

if age >= 18
  puts "Vous êtes majeur"
end
```

- ➔ Pas d’accolade
- ➔ Pas de point-virgule
- ➔ Pas de parenthèse

Method chaining

```
1  [1,2,2,3,3,3].uniq.sort.reverse
2  # = [3, 2, 1]
```

Predicate & Bang methods (? & !)

```ruby
cours = "Ruby"
cours.downcase!
puts cours
# => ruby
```

```ruby
5.odd?
# => true

2.odd?
# => false
```

Les opérateurs arithmétiques

```ruby
1+1
# => 2

2 - 1
# => 1

5 * 3
# => 15

4 / 2
# => 2

5 % 2
# => 1

2 ** 8
# => 256
```

Les opérateurs de comparaison

```
==
!=
>
<
>=
<=
===
```

```ruby
(1..10) === 4
# => true
(1..10) === 15
# => false

Integer === 10
# => true
Integer === 'ten'
# => false

/ruby/ === 'ruby on rails'
# => true
/ruby/ === 'javascript'
# => false
```

Les opérateurs d’assignation

```
a = b

a += b
# => a = a + b

a -= b
# => a = a - b

a *= b
# => a = a * b

a /= b
# => a = a / b

a %= b
# => a = a % b

a **= b
# => a = a ** b
```

Les opérateurs logiques

```ruby
&& = and
|| = or
! = not

majeur = true
puts majeur ? "Vous êtes majeur" : "Vous êtes mineur"
majeur ? puts("Vous êtes majeur") : puts ("Vous êtes mineur")
```

Les conditions

```ruby
if age >= 18
  # ...
else
  # ...
end



unless role == "admin"
  # ...
end



puts "Vous êtes majeur" unless age <= 18
```

L’opérateur Lonely

```ruby
if user && user.name && user.email
  # ...
end

if user&.name&.email
  # ...
end
```

Les boucles

```ruby
eleves = ['Michel', 'Sarah', 'Tom']

for eleve in eleves
  # ...
end

i = 0
while i <= 5
  # ...
  i += 1
end
```

```ruby
i = 0
until i == 5
  # ...
  i += 1
end

i = 0
loop do
  i += 1
  # ...
  if i == 10
    break
  end
end
```

Les tableaux (déclaration)

```
1  eleves = ['Michel', 'Sarah', 'Tom']
2
3  eleves = %W(Michel Sarah Tom)
4
5  profil = ['Michel', 23, false, :male]
```

Les tableaux (méthode)

```ruby
array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

array.first # => 1
array.last # =>10
array.empty? # => false
array.include?(5) # => true
array.reverse.first # => 10
```

Les symboles

```
order.status = "canceled"
order.status = "confirmed"

order.status = :canceled
order.status = :confirmed


car1 = "red"
car2 = "red"
car3 = "red"
car4 = "black"

car1 = :red
car2 = :red
car3 = :red
car4 = :black
```

Les hashs (déclaration)

```ruby
{ "un" => "one", "deux" => "two", "trois" => "three" }

{ :un => "one", :deux => "two", :trois => "three" }

{ 1 => "un", 2 => "deux", 3 => "trois" }

{ "francais" => ["bonjour", "mardi"], "english" => ["notebook", "chair"] }

{ :trad => { :un => "one", :deux => "two", :trois => "three" } }
```

Les hashs (méthode)

```ruby
hash = { "un" => "one", "deux" => "two", "trois" => "three" }

hash.empty? # => false
hash.has_key? 'toto' # => false
hash.has_value? 'two' # => true
hash.keys # => ["un", "deux", "trois"]
```

Les blocks

```ruby
def color(car)
  if car == "bmw"
    yield "black"
  elsif car == 'citroen'
    yield "red"
  else
    yield unknown
  end
end

car1 = "bmw"
color(car1) { |color| puts "The car is #{color}" }
```

```ruby
[1, 2, 3, 4, 5].reduce { |a, b| a + b } # => 15

[1, 2, 3, 4, 5].reduce do |a, b|
  a + b
end
# => 15
```

Les classes

```ruby
class Adresse
  def initialize(rue)
    @rue = rue
  end

  def rue
    @rue
  end
end

adresse1 = Adresse.new("20 rue de la paix")
adresse1.rue
```

attr_reader

```ruby
class Adresse
  def initialize(rue)
    @rue = rue
  end

  def rue
    @rue
  end
end

adresse1 = Adresse.new("20 rue de la paix")
adresse1.rue
```

```ruby
class Adresse
  attr_reader :rue

  def initialize(rue)
    @rue = rue
  end
end

adresse1 = Adresse.new("20 rue de la paix")
adresse1.rue
```

attr_writer

```ruby
class Adresse
  attr_reader :rue

  def initialize(rue)
    @rue = rue
  end

  def rue=(une_rue)
    @rue = une_rue
  end
end

adresse1 = Adresse.new("20 rue de la paix")
adresse1.rue

adresse1.rue = "1 rue du paradis"
adresse1.rue
```

```ruby
class Adresse
  attr_reader :rue
  attr_writer :rue

  def initialize(rue)
    @rue = rue
  end
end

adresse1 = Adresse.new("20 rue de la paix")
adresse1.rue

adresse1.rue = "1 rue du paradis"
adresse1.rue
```

attr_accessor

```ruby
class Adresse
  attr_reader :rue
  attr_writer :rue

  def initialize(rue)
    @rue = rue
  end
end

adresse1 = Adresse.new("20 rue de la paix")
adresse1.rue

adresse1.rue = "1 rue du paradis"
adresse1.rue
```

```ruby
class Adresse
  attr_accessor :rue

  def initialize(rue)
    @rue = rue
  end
end

adresse1 = Adresse.new("20 rue de la paix")
adresse1.rue

adresse1.rue = "1 rue du paradis"
adresse1.rue
```

Méthode privé

```ruby
class Adresse
  attr_accessor :rue

  def initialize(rue)
    @rue = rue
    self.add_country
  end

  private

  def add_country
    @rue += ", France"
  end


end

adresse1 = Adresse.new("20 rue de la paix")
adresse1.rue
# => 20 rue de la paix , France
adresse1.rue = "1 rue du paradis"
adresse1.rue
# => 1 rue du paradis

adresse1.add_country
# => Error
```

Pratique

Executer un script ruby (Docker)

test.rb

```ruby
puts "Votre nom ?"
nom = gets.chomp!

puts "Votre nom : #{nom}"
```

docker run -ti --rm -v ${PWD}:/home/exo -w /home/exo ruby:latest ruby ***test.rb***

Exercice 1

- Écrire un programme demandant à l'utilisateur de saisir deux chaînes de caractères au clavier.

  Voir si la première chaîne est deux fois plus grande que la deuxième et afficher un message approprié.

Exercice 2

- Ecrire un programme permettant de calculer la somme des nombres compris entre 1 et un entier demandé à l'utilisateur

- Exemple si l'utilisateur entre 10 : 1+2+3+4+5+6+7+8+9+10 --> 55

Exercice 3

- Ecrire un programme qui demande à l'utilisateur de saisir 5 entiers stockés dans un tableau.

- Le programme doit ensuite afficher l'indice du plus grand élément, puis la moyenne des entiers.

Merci pour votre attention !

Me contacter:

Jacques Atacan
jacques.atacan@capyweb.fr

https://capyweb.fr/