

Tête première dans Phoenix LiveView

par Marc-Olivier Fiset





Votre feedback est le bienvenu!



Au menu

- Courte présentation
- Elixir crash course speedrun any%
- Phoenix LiveView, c'est quoi?
- Séance de *Live coding*
- Conclusion
- Questions



Salut! 🖐️

Moi c'est Marco

Directeur des technologies
chez Rum&Code



Rum&Code, c'est qui?

Rum&Code est une agence de **développement sur mesure**, basée en Mauricie.

Plus de **20 projets actifs** dans plusieurs de domaines d'affaires différents:

- Finances
- Éducation
- IoT
- Agriculture
- et plusieurs autres

On utilise **Elixir et Phoenix depuis 2020** pour tous nos démarrages de projets.



Pourquoi apprendre un nouveau langage?



Un peu de notoriété



MIREGO

Et plusieurs autres:

- Pepsico
- Pinterest
- Adobe
- Bet365
- TheScore
- DNSimple
- Farmbot
- Motorola
- Moz
- Whatsapp*



Elixir Crash Course speedrun any%



defmodule Example do

def sum_of_squares(numbers) do

squares = Enum.map(numbers, fn x -> x * x end)

sum = Enum.reduce(squares, fn a, b -> a + b end)

sum

end

end

```
defmodule Example do
  def sum_of_squares(numbers) do
    squares = Enum.map(numbers, fn x -> x * x end)
    sum = Enum.reduce(squares, fn a, b -> a + b end)

    sum
  end
end
```

```
defmodule Example do
  def sum_of_squares(numbers) do
    squares = Enum.map(numbers, fn x -> x * x end)
    sum = Enum.reduce(squares, fn a, b -> a + b end)

    sum
  end
end
```

```
defmodule Example do
  def sum_of_squares(numbers) do
    squares = Enum.map(numbers, fn x -> x * x end)
    sum = Enum.reduce(squares, fn a, b -> a + b end)
```

sum

end

end

```
defmodule Example do
  def sum_of_squares(numbers) do
    squares = Enum.map(numbers, fn x -> x * x end)
    sum = Enum.reduce(squares, fn a, b -> a + b end)

    sum
  end
end
```

```
defmodule Example do
  def sum_of_squares(numbers) do
    squares = Enum.map(numbers, fn x -> x * x end)
    sum = Enum.reduce(squares, fn a, b -> a + b end)

    sum
  end
end
```

```
defmodule Example do
  def sum_of_squares(numbers) do
    numbers
    |> Enum.map(fn x -> x * x end)
    |> Enum.reduce(fn a, b -> a + b end)
  end
end
```

```
defmodule Example do
  def sum_of_squares(numbers) do
    squares = Enum.map(numbers, fn x -> x * x end)
    sum = Enum.reduce(squares, fn a, b -> a + b end)

    sum
  end
end
```

```
defmodule Example do
  def sum_of_squares(numbers) do
    numbers
    |> Enum.map(fn x -> x * x end)
    |> Enum.reduce(fn a, b -> a + b end)
  end
end
```

Example.sum_of_squares(1..10) # => 385

```
defmodule Example do
  def fib(0), do: 0
  def fib(1), do: 1
  def fib(n) when n > 1, do: fib(n - 2) + fib(n - 1)
  def fib(n), do: nil
end
```



```
defmodule Example do
  def fib(0), do: 0
  def fib(1), do: 1
  def fib(n) when n > 1, do: fib(n - 2) + fib(n - 1)
  def fib(n), do: nil
end
```

```
defmodule Example do
  def fib(0), do: 0
  def fib(1), do: 1
  def fib(n) when n > 1, do: fib(n - 2) + fib(n - 1)
  def fib(n), do: nil
end
```

```
defmodule Example do
  def fib(0), do: 0
  def fib(1), do: 1
  def fib(n) when n > 1, do: fib(n - 2) + fib(n - 1)
  def fib(n), do: nil
end
```

Example.fib(9)

=> 34

```
defmodule Example do
  def fib(0), do: 0
  def fib(1), do: 1
  def fib(n) when n > 1, do: fib(n - 2) + fib(n - 1)
  def fib(n), do: nil
end
```

Example.fib(9)

=> 34

Example.fib(-1)

=> nil

```
defmodule Circle do
  defstruct [:radius]
end
```

```
defmodule Rectangle do
  defstruct [:width, :height]
end
```

```
rectangle = %Rectangle{
  width: 20,
  height: 30
}
```

```
circle = %Circle{
  radius: 15
}
```

```
defmodule Circle do
  defstruct [:radius]
end
```

```
defmodule Rectangle do
  defstruct [:width, :height]
end
```

```
rectangle = %Rectangle{
  width: 20,
  height: 30
}
```

```
circle = %Circle{
  radius: 15
}
```

```
defmodule Circle do
  defstruct [:radius]
end
```

```
defmodule Rectangle do
  defstruct [:width, :height]
end
```

```
rectangle = %Rectangle{
  width: 20,
  height: 30
}
```

```
circle = %Circle{
  radius: 15
}
```

```
defmodule Circle do
  defstruct [:radius]
end
```

```
defmodule Rectangle do
  defstruct [:width, :height]
end
```

```
rectangle = %Rectangle{
  width: 20,
  height: 30
}
```

```
circle = %Circle{
  radius: 15
}
```



```
defmodule Geometry do
```

```
  def area(%Circle{radius: r}) do  
    3.14159 * r * r  
  end
```

```
  def area(%Rectangle{width: w, height: h}) do  
    w * h  
  end
```

```
end
```

```
Geometry.area(circle)
```

```
# => 706.85775
```

```
Geometry.area(rectangle)
```

```
# => 600
```

```
defmodule Geometry do
  def area(%Circle{radius: r}) do
    3.14159 * r * r
  end

  def area(%Rectangle{width: w, height: h}) do
    w * h
  end
end
```

```
Geometry.area(circle)
# => 706.85775
```

```
Geometry.area(rectangle)
# => 600
```

```
defmodule Geometry do
  def area(%Circle{radius: r}) do
    3.14159 * r * r
  end

  def area(%Rectangle{width: w, height: h}) do
    w * h
  end
end
```

Geometry.area(circle)

=> 706.85775

Geometry.area(rectangle)

=> 600



Elixir – Autres choses à savoir

Langage fonctionnel

Plutôt que orienté objet

Dynamique*

Plutôt que fortement typé

*Système de types graduel en cours de développement

Bâti sur Erlang

Plusieurs décennies de robustesse éprouvée dans des systèmes temps réels (téléphonie, communications, etc)

Runtime distribué

On peut *scaler* en ajoutant plus de *nodes* et en les connectant entre elles

Arbre de processus supervisés

Isolation du *state*

Gestion de la concurrence

Résilience en cas d'erreur



Phoenix LiveView, c'est quoi?





Phoenix LiveView, c'est quoi?

Un framework web moderne, à la Ruby on Rails, Laravel, etc



Phoenix LiveView, c'est quoi?

Une extension à Phoenix qui ajoute le concept de
« pages connectées »



Assez parlé, montre-moi plus de code!







Phoenix LiveView, c'est quoi?

Server-side rendering

Un backend pour les gouverner tous

Components purement en Elixir

Toute la logique est dans le même langage

Communication bi-directionnelle par WebSocket

De l'interactivité en temps réel

Système de hooks JavaScript

Parce qu'on peut pas toujours s'en sauver

Navigation "live"

Rendus partiels et transitions instantanées

Tout simplement la meilleure expérience de développement web



La meilleure expérience de développement web



« *Complexity is the enemy* »



D'autres frameworks/librairies qui offrent une expérience similaire

- Livewire (Laravel)
- La suite Hotwire (Rails, mais *backend-agnostic*)
- HTMX (beaucoup plus *barebones*)

Mon conseil? Abandonnez vos *single page apps* et retrouvez la simplicité du *server-side rendering*



Merci! ❤️

Des questions?



Votre feedback est le bienvenu!

