WHAT IS PATTERN MATCHING

and How to Use the Debugger

PETER MUELLER

- @felix_starman
- felix-starman
- slides at: gh/felix-starman/what_is_pattern_(...)

PATTERN MATCHING

FIRST LET'S TALK ABOUT "ASSIGNMENT"

```
// javascript
let some_var = ["menuItem", 37, "Crunchy Frog"]

# ruby
some_var = [:menu_item, 37, "Crunchy Frog"]

# elixir
some_var = [:menu_item, 37, "Crunchy Frog"]
```

PATTERN MATCHING

THERE IS EVEN *multiple-assignment* IN SOME LANGUAGES

```
// javascript: destructuring assignment
[type, id, title] = ["menuItem", 37, "Crunchy Frog"]

# ruby: multiple-assignment
[type, id, title]  = [:menu_item, 37, "Crunchy Frog"]

# elixir: matching "assignment"
[type, id, title]  = [:menu_item, 37, "Crunchy Frog"]
```

PATTERN MATCHING

THERE IS EVEN *multiple-assignment* IN SOME LANGUAGES

```
// javascript: destructuring assignment
[type, id, title] = ["menuItem", 37, "Crunchy Frog"]

# ruby: multiple-assignment
[type, id, title]  = [:menu_item, 37, "Crunchy Frog"]

# elixir: matching "assignment"
[type, id, title]  = [:menu_item, 37, "Crunchy Frog"]
```

PATTERN MATCHING

THERE IS EVEN *multiple-assignment* IN SOME LANGUAGES

```
// javascript: destructuring assignment
[type, id, title] = ["menuItem", 37, "Crunchy Frog"]

# ruby: multiple-assignment
[type, id, title]  = [:menu_item, 37, "Crunchy Frog"]

# elixir: matching "assignment"
[type, id, title]  = [:menu_item, 37, "Crunchy Frog"]
```

PATTERN MATCHING

BUT "Matching" IS NOT QUITE assignment

```elixir
# elixir
fav_fruit = %{apples: ["Honey Crisp", "Granny Smith"]}
%{apples: apples} = fav_fruit
apples # ["Honey Crisp", "Granny Smith"]

%{durian: true} = fav_fruit
# (MatchError) no match of right hand side value: %{apples: ["Honey Crisp", "Granny Smith"]}
```

PATTERN MATCHING

BUT "Matching" IS NOT QUITE assignment

```elixir
# elixir
fav_fruit = %{apples: ["Honey Crisp", "Granny Smith"]}
%{apples: apples} = fav_fruit
apples # ["Honey Crisp", "Granny Smith"]

%{durian: true} = fav_fruit
# (MatchError) no match of right hand side value: %{apples: ["Honey Crisp", "Granny Smith"]}
```

PATTERN MATCHING

BUT "Matching" IS NOT QUITE assignment

```elixir
# elixir
fav_fruit = %{apples: ["Honey Crisp", "Granny Smith"]}
%{apples: apples} = fav_fruit
apples # ["Honey Crisp", "Granny Smith"]

%{durian: true} = fav_fruit
# (MatchError) no match of right hand side value: %{apples: ["Honey Crisp", "Granny Smith"]}
```

PATTERN MATCHING

BUT "*Matching*" IS NOT QUITE *assignment*

```elixir
# elixir
fav_fruit = %{apples: ["Honey Crisp", "Granny Smith"]}
%{apples: apples} = fav_fruit
apples # ["Honey Crisp", "Granny Smith"]

%{durian: true} = fav_fruit
# (MatchError) no match of right hand side value: %{apples: ["Honey Crisp", "Granny Smith"]}
```

PATTERN MATCHING

BUT "Matching" IS NOT QUITE assignment

```elixir
# elixir
fav_fruit = %{apples: ["Honey Crisp", "Granny Smith"]}
%{apples: apples} = fav_fruit
apples # ["Honey Crisp", "Granny Smith"]

%{durian: true} = fav_fruit
# (MatchError) no match of right hand side value: %{apples: ["Honey Crisp", "Granny Smith"]}
```

PATTERN MATCHING

YOU CAN MATCH ON THE SHAPE, BUT ASSIGN *nothing*

```
# Maps
%{username: _} = user_session
%{username: _, admin: true} = user_session
%{username: "demosthenes"} = user_session

%{author: ^current_user} = article
```

PATTERN MATCHING

YOU CAN MATCH ON THE SHAPE, BUT ASSIGN *nothing*

```
# Maps
%{username: _} = user_session
%{username: _, admin: true} = user_session
%{username: "demosthenes"} = user_session

%{author: ^current_user} = article
```

PATTERN MATCHING

YOU CAN MATCH ON THE SHAPE, BUT ASSIGN *nothing*

```
# Maps
%{username: _} = user_session
%{username: _, admin: true} = user_session
%{username: "demosthenes"} = user_session

%{author: ^current_user} = article
```

PATTERN MATCHING

YOU CAN MATCH ON THE SHAPE, BUT ASSIGN *nothing*

```
# Maps
%{username: _} = user_session
%{username: _, admin: true} = user_session
%{username: "demosthenes"} = user_session

%{author: ^current_user} = article
```

MATCHING ON OTHER DATA STRUCTURES

```
# Strings
"NAME=" <> name = commandline_args # String matching

# Lists
[un, pw] = String.split("my_user:password1", ":")

# Tuples
{:ok, result} = VeryImportantThing.do_it()
```

MATCHING ON OTHER DATA STRUCTURES

```
# Strings
"NAME=" <> name = commandline_args # String matching

# Lists
[un, pw] = String.split("my_user:password1", ":")

# Tuples
{:ok, result} = VeryImportantThing.do_it()
```

MATCHING ON OTHER DATA STRUCTURES

```
# Strings
"NAME=" <> name = commandline_args # String matching

# Lists
[un, pw] = String.split("my_user:password1", ":")

# Tuples
{:ok, result} = VeryImportantThing.do_it()
```

MATCHING IN FUNCTION DEFINITIONS

LITTLE UGLY

```
def authorized?(conn) do
  if conn.admin do
    true
  else
    if conn.current_user == conn.owner_of_thing do
      true
    else
      false
    end
  end
end
```

MATCHING IN FUNCTION DEFINITIONS

LITTLE UGLY

```
def authorized?(conn) do
  if conn.admin do
    true
  else
    if conn.current_user == conn.owner_of_thing do
      true
    else
      false
    end
  end
end
```

MATCHING IN FUNCTION DEFINITIONS

LITTLE UGLY

```
def authorized?(conn) do
  if conn.admin do
    true
  else
    if conn.current_user == conn.owner_of_thing do
      true
    else
      false
    end
  end
end
```

MATCHING IN FUNCTION DEFINITIONS

LITTLE UGLY

```
def authorized?(conn) do
  if conn.admin do
    true
  else
    if conn.current_user == conn.owner_of_thing do
      true
    else
      false
    end
  end
end
```

MATCHING IN FUNCTION DEFINITIONS

LITTLE UGLY

```
def authorized?(conn) do
  if conn.admin do
    true
  else
    if conn.current_user == conn.owner_of_thing do
      true
    else
      false
    end
  end
end
```

MATCHING IN FUNCTION DEFINITIONS

LITTLE UGLY

```
def authorized?(conn) do
  if conn.admin do
    true
  else
    if conn.current_user == conn.owner_of_thing do
      true
    else
      false
    end
  end
end
```

MATCHING IN FUNCTION DEFINITIONS

LITTLE UGLY

```
def authorized?(conn) do
  if conn.admin do
    true
  else
    if conn.current_user == conn.owner_of_thing do
      true
    else
      false
    end
  end
end
```

MATCHING IN FUNCTION DEFINITIONS

LITTLE UGLY

```
def authorized?(conn) do
  if conn.admin do
    true
  else
    if conn.current_user == conn.owner_of_thing do
      true
    else
      false
    end
  end
end
```

MATCHING IN FUNCTION DEFINITIONS

LITTLE UGLY

```
def authorized?(conn) do
  if conn.admin do
    true
  else
    if conn.current_user == conn.owner_of_thing do
      true
    else
      false
    end
  end
end
```

MATCHING IN FUNCTION DEFINITIONS

AWWWWWWW YEAAAAH

```elixir
def authorized?(%{admin: true}), do: true
def authorized?(%{owner: user, current_user: user}), do: true
def authorized?(_), do: false

# These are "definitions" for ONE function (`authorized?/1`)
# Patterns are evaluated in the order they are defined
```

MATCHING IN FUNCTION DEFINITIONS

AWWWWWWW YEAAAAH

```elixir
def authorized?(%{admin: true}), do: true
def authorized?(%{owner: user, current_user: user}), do: true
def authorized?(_), do: false

# These are "definitions" for ONE function (`authorized?/1`)
# Patterns are evaluated in the order they are defined
```

MATCHING IN FUNCTION DEFINITIONS

AWWWWWWW YEAAAAH

```
def authorized?(%{admin: true}), do: true
def authorized?(%{owner: user, current_user: user}), do: true
def authorized?(_), do: false

# These are "definitions" for ONE function (`authorized?/1`)
# Patterns are evaluated in the order they are defined
```

MATCHING IN FUNCTION DEFINITIONS

AWWWWWWW YEAAAAH

```elixir
def authorized?(%{admin: true}), do: true
def authorized?(%{owner: user, current_user: user}), do: true
def authorized?(_), do: false

# These are "definitions" for ONE function (`authorized?/1`)
# Patterns are evaluated in the order they are defined
```

MATCHING IN FUNCTION DEFINITIONS

AWWWWWWW YEAAAAH

```elixir
def authorized?(%{admin: true}), do: true
def authorized?(%{owner: user, current_user: user}), do: true
def authorized?(_), do: false

# These are "definitions" for ONE function (`authorized?/1`)
# Patterns are evaluated in the order they are defined
```

DEBUGGER