Examples of Variants

Let's take a look at a few more instances of variants in action!

WE'LL COVER THE FOLLOWING

The `shape` Variant

We'll create a `shape` variant which contains the `Square`, `Circle`, and `Triangle` constructors. Each constructor will have arguments of the `float` type as its dimensions.

Whenever one of the constructors is called, the area for that shape will be calculated through the `getArea()` method.

Implementation

```
type shape =
  | Square(float)
  | Circle(float)
  | Triangle(float, float);

let ci = Circle(7.0);
let sq = Square(8.5);
let tr = Triangle(15.1, 2.5);

let getArea = (myShape) =>
  switch (myShape) {
    | Square(length) => "Area of square: " ++ string_of_float(length *. length)
    | Circle(radius) => "Area of circle: " ++ string_of_float(radius *. radius *. 3.142)
    | Triangle(length, width) => "Area of triangle: " ++ string_of_float(width *. length *. 0
  };

Js.log(getArea(ci));
Js.log(getArea(sq));
Js.log(getArea(tr));
```

Tennis Players

In this example, we have created two variants, `player` and `turn`. The tennis match has two players, and each player can either serve or receive based on the `turn` variant.

For this purpose, `turn` will use the `player` variant in its constructors. This is an example of variants using other variants as data types.

The rest is just pattern matching through `switch` expressions:

```
type player =
  | Player1
  | Player2;

type turn =
  | Serve(player) /* Argument of the player type */
  | Receive(player);

let p2Serve: turn = Serve(Player2);
let p1Receive: turn = Receive(Player1);

let tennisServe = (playerTurn) =>
  switch(playerTurn) {
    | Serve(Player1)  => "Player1 is serving"
    | Serve(Player2) => "Player2 is serving"
    | _ => "" /* Ignoring all other cases */
  };

let tennisReceive = (playerTurn) =>
  switch(playerTurn) {
    | Receive(Player1) => "Player1 is receiving"
    | Receive(Player2) => "Player2 is receiving"
    | _ => "" /* Ignoring all other cases */
  };

Js.log(tennisServe(p2Serve));
Js.log(tennisReceive(p1Receive));
```

In the next lesson, we'll explore the built-in `optional` variant data type.