Pixi 5 Destructuring with pattern matching:

Elm:

Tuple:

```
myTuple = ("A", "B", "C")
myNestedTuple = ("A", "B", "C", ("X", "Y", "Z"))

let
  (a,b,c) = myTuple
in
  a ++ b ++ c
-- "ABC" : String

let
  (a,b,c,(x,y,z)) = myNestedTuple
in
  a ++ b ++ c ++ x ++ y ++ z
-- "ABCXYZ" : String
```

```
-- with no destructuring
width = 200
height = 100

-- with destructuring
(width, height) = (200, 100)
```

List:

```
myList = ["a", "b", "c"]

first list =
  case list of
    f::_ -> Just f
    [] -> Nothing

first myList
-- Just "a"
```

Record:

```
myRecord = { x = 3, y = 4 }

sum record =
  let
    {x,y} = record
  in
    x + y

sum myRecord
-- 7
```

Cleaner:

```
sum {x,y} =
  x + y
```

Dropping one value:

```
onlyX {x} =
  x

onlyX myRecord
-- 3 : number
```

Custom type:

```
type MyThing
  = AString String
  | AnInt Int
  | ATuple (String, Int)

unionFn : MyThing -> String
unionFn thing =
  case thing of
    AString s -> "It was a string: " ++ s
    AnInt i -> "It was an int: " ++ toString i
    ATuple (s, i) -> "It was a string and an int: " ++ s ++ " and " ++ toString i
```