An aggregate data type that consists of labels mapped to arbitrary values.

```
type name =
    { name1 : type1;
      ...
      nameN : typeN
    }
```

```
let name =
   { name1 = val1;
     ...
     nameN = valN;
   }
```

- With projection: `record.label`
- With pattern matching

```
let { name1 = var1; name2 = var2 } = expr
```

Not all fields must be matched.

```
let name = { someTuple with field1 = val1; ... }
```

The fields must be taken from `someTuple`.

In the declaration prefix one or more of the field names with the `mutable` keyword.

```
mutableRecord.field <- newVal
```

... shared between all toplevel records.

Values of records with fields of the same name overwrite values of fields in previous records.

The same as lists, except for the pipe next to the brackets:

```
[| val1 ; ... ; valN |]
```

- Use pattern matching.
- Access directly with `arr.(index)`.

```
arr.(index) <- newVal
```

- Using the literal syntax.
- Using `Array.create len initVal`