How does OCaml support parametric polymorphism?

How are type variables written?

Add type annotations to a function declaration.

What is the *value restriction*?

What is the type of an expression that is a composition of truly polymorphic parts?

Unless otherwise stated *values* are ...

What is the purpose of the value restriction?

Why can't function application result in a polymorphic type?

How can one circumvent the value restriction?

Why doesn't OCaml provide overloading?

How are tuples written?

And their types?

What is *destructuring*?

Demonstrate on tuples.

Access the elements of a tuple.

How are lists written?

How is the *cons* operator used?

How are parameterized types written?

What is an association list?

How is it queried?

What is *tail recursion*?

Why is it desirable?

List accumulator values in recursive functions are best built ...