OCaml Functional Programming

FP Concepts, OCaml Language, OCaml Tools, OCaml Library, Dune, Opam, Projects Using OCaml

" OCaml is an industrial strength programming language supporting functional, imperative and object-oriented styles " [https://OCaml.org/]. OCaml is best known as a functional programming language and that is what we focus on in this intensive course. OCaml competes with Haskell to be the leading functional language. For a number of reasons, we prefer OCaml. It has a number of advanced features, a richer type system and a more extensive system library. It is also used on many cutting edge projects that interest us. Examples of practical uses of OCaml include: the experimental redtt (based on cubical type theory) and the well established Coq proof assistants, samples in the important TAPL book, the WebAssembly spec interpreter, the mirageOS unikernel and Jane Street.

Functional programming is different from regular object-oriented programming. So we start by looking at FP for non-FP programmers. Then we explore all aspects of programming with OCaml – the language, tooling and system library. We also explore add-on libraries. Our goal is to ensure all attendees are up to speed with OCaml programming and immediately after this course can be productive as OCaml developers.

Target Audience

Experienced software developers who wish to start functional programming using the OCaml language.

Prerequisites

Attendees need prior programming experience in one of the mainstream object-oriented programming languages.

Important: This course assumes attendees have no prior functional programming experience, so in addition to teaching OCaml, we also cover general aspects of functional programming.

Contents of One-Day Training Course

FP For non FP Programmers

What is functional programming?
Mathematics is important for modern code
What is a variable (immutability)
Pattern matching
Handling change
Pure functions
Building more reliable code
Many FP ideas are seeping into other types of programming – why and how?

OCaml Tour

Language basics (control flow, etc.)
Structure source trees and individual units
`main` function-conventional/not necessary
Creating functions
Events and callbacks
Type inferencing

OCaml Language

Modules (submodules)
.mli interface definition files
More detailed look at functions
Visibility
Objects and object types
Classes / polymorphism / class types
Type mismatch and other errors

OCaml Runtime

OCaml runtime architecture
How code executes (e.g. as native code)
Dynamic linking -what's involved in dynamically loading a library
The GC
Foreign function interface - interacting with C code (and other languages)

OCaml Library

Structure of standard library
Tour of main functionality areas
Common collections
Text handling
Threads library (modules: thread, mutex, condition, event, ..)
Async and deferred computation

Tooling

Debugging
Testing
Lint
Pre-processor
Compilation tools – (parsing, etc.)
Abstract syntax tree

Dune

Dune is OCaml's composable build system
Defining steps needed for build
Configuration
S-Expression syntax
Compilation flags

Opam

Opam is the OCaml package manager
Package repository
Extensive collection of pre-built packages
Managing locally installed packages
Packaging definition file – creating and populating for a custom project

Projects

Exploring usage of OCaml in a variety of open source projects that have shipped, to see how everything fits together in a larger production setting