

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.

Contents of One-Day Training Course	
<p>Target Audience Experienced software developers who wish to start functional programming using the OCaml language.</p> <p>Prerequisites Attendees need prior programming experience in one of the mainstream object-oriented programming languages.</p> <p>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.</p>	<div style="display: flex; justify-content: space-between;"> <div style="width: 48%;"> <p>FP For non FP Programmers</p> <p>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?</p> <p style="text-align: center;">OCaml Tour</p> <p>Language basics (control flow, etc.) Structure source trees and individual units main function-conventional/not necessary Creating functions Events and callbacks Type inferencing</p> <p style="text-align: center;">OCaml Language</p> <p>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</p> <p style="text-align: center;">OCaml Runtime</p> <p>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)</p> </div> <div style="width: 48%;"> <p style="text-align: center;">OCaml Library</p> <p>Structure of standard library Tour of main functionality areas Common collections Text handling Threads library (modules: thread, mutex, condition, event, ..) Async and deferred computation</p> <p style="text-align: center;">Tooling</p> <p>Debugging Testing Lint Pre-processor Compilation tools – (parsing, etc.) Abstract syntax tree</p> <p style="text-align: center;">Dune</p> <p>Dune is OCaml’s composable build system Defining steps needed for build Configuration S-Expression syntax Compilation flags</p> <p style="text-align: center;">Opam</p> <p>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</p> <p style="text-align: center;">Projects</p> <p>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</p> </div> </div>