Videos and Reading Notes

Help

Having trouble viewing lectures? Try changing players. Your current player format is html5. Change to flash.

Section 0

- Course Introduction: About the Course (12:23)
- Course Introduction: About Programming Languages (12:04)
- Grading Policy (12:59)
- Software Installation Introduction (1:22)
- Emacs installation (2:27)
- SML installation (1:47)
- SML Mode installation (3:52)

Section 1

- Section 1 Reading Notes
- ML Variable Bindings and Expressions (14:32)
- Rules for Expressions (9:13)
- The REPL and Errors (12:10)
- Shadowing (6:49)
- Functions Informally (7:37)
- Functions Formally (8:56)
- Pairs and Other Tuples (9:16)

- ✔ Introducing Lists (10:34)
- ✔ List Functions (11:21)
- ✔ Let Expressions (7:51)
- ✔ Nested Functions (7:38)
- ✔ Let and Efficiency (10:10)
- ✔ Options (9:09)
- ✔ Booleans and Comparison Operations (7:00)
- ✔ Benefits of No Mutation (7:24)
- ✔ Optional: Java Mutation (4:44)
- ✔ Pieces of a Language (6:02)

Section 2

- ✔ Section 2 Reading Notes
- ✔ Building Compound Types (5:34)
- ✔ Records (8:25)
- ✔ Tuples as Syntactic Sugar (6:06)
- ✔ Datatype Bindings (8:39)
- ✔ Case Expressions (10:50)

- ✔ Useful Datatypes (11:22)
- ✔ Pattern Matching So Far (4:37)
- ✔ Another Expression Example (8:01)
- ✔ Type Synonyms (6:19)
- ✔ Lists and Options are Datatypes (9:10)
- ✔ Polymorphic Datatypes (8:11)
- ✔ Each of Pattern Matching / Truth About Functions (14:30)
- ✔ A Little Type Inference (6:28)
- ✔ Polymorphic and Equality Types (8:00)
- ✔ Nested Patterns (10:32)
- ✔ More Nested Patterns (10:43)
- ✔ Nested Patterns Precisely (6:35)
- ✔ Optional: Function Patterns (3:40)
- ✔ Exceptions (7:47)
- ✔ Tail Recursion (9:44)

- ✔ Accumulators for Tail Recursion (7:39)
- ✔ Perspective on Tail Recursion (5:33)

Section 3

- ✔ Section 3 Reading Notes
- ✔ Introduction to First-Class Functions (7:50)
- ✔ Functions as Arguments (8:40)
- ✔ Polymorphic Types and Functions as Arguments (7:57)
- ✔ Anonymous Functions (9:17)
- ✔ Unnecessary Function Wrapping (5:05)
- ✔ Map and Filter (8:56)
- ✔ Generalizing Prior Topics (9:01)
- ✔ Lexical Scope (7:22)
- ✔ Lexical Scope and Higher-Order Functions (7:37)
- ✔ Why Lexical Scope (11:44)
- ✔ Closures and Recomputation (6:30)

- ✔ Fold and More Closures (11:28)
- ✔ Closure Idiom: Combining Functions (9:18)
- ✔ Closure Idiom: Currying (10:32)
- ✔ Partial Application (9:41)
- ✔ Currying Wrapup (6:40)
- ✔ Mutable References (8:07)
- ✔ Closure Idiom: Callbacks (8:25)
- ✔ Standard-Library Documentation (7:00)
- ✔ Optional: Abstract Data Types With Closures (11:14)
- ✔ Optional: Closure Idioms Without Closures (4:48)
- ✔ Optional: Java Without Closures (12:08)
- ✔ Optional: C Without Closures (10:56)

⌄ Optional: Course Motivation

- ✔ Optional: Course-Motivation Introduction (5:56)
- ✔ Optional: Why Study General PL Concepts? (10:30)
- ✔ Optional: Are All PLs the Same? (6:51)
- ✔ Optional: Why Functional Languages? (11:15)

Optional: Why ML, Racket, and Ruby? (12:34)

Section 4

Section 4 Reading Notes

- ✔ Section Introduction (1:45)
- ✔ What is Type Inference (5:37)
- ✔ ML Type Inference (6:09)
- ✔ Type Inference Examples (10:27)
- ✔ Polymorphic Examples (10:52)
- ✔ Optional: The Value Restriction and Other Type-Inference Challenges (9:53)
- ✔ Mutual Recursion (9:45)
- ✔ Modules for Namespace Management (6:25)
- ✔ Signatures and Hiding Things (7:02)
- ✔ A Module Example (11:06)
- ✔ Signatures for Our Example (11:03)
- ✔ Signature Matching (4:03)
- ✔ An Equivalent Structure (6:38)

- ✔ Another Equivalent Structure (9:01)
- ✔ Different Modules Define Different Types (3:32)
- ✔ Equivalent Functions (8:41)
- ✔ Standard Equivalences (10:01)
- ✔ Equivalence Versus Performance (6:00)

Section 5

- ✔ Section 5 Reading Notes
- ✔ Introduction to Racket (8:23)
- ✔ Racket Definitions, Functions, Conditionals (10:14)
- ✔ Racket Lists (9:07)
- ✔ Syntax and Parentheses (8:36)
- ✔ Parentheses Matter! (Debugging Practice) (10:50)
- ✔ Dynamic Typing (9:36)
- ✔ Cond (8:52)
- ✔ Local Bindings (13:11)
- ✔ Toplevel Bindings (4:54)
- ✔ Mutation with set! (8:28)

- The Truth About Cons (7:46)
- mcons For Mutable Pairs (7:38)
- Delayed Evaluation and Thunks (10:12)
- Avoiding Unnecessary Computations (6:48)
- Delay and Force (9:18)
- Using Streams (10:10)
- Defining Streams (10:49)
- Memoization (10:17)
- Macros: The Key Points (8:52)
- Optional: Tokenization, Parenthesization, and Scope (7:30)
- Optional: Racket Macros With define-syntax (9:25)
- Optional: Variables, Macros, and Hygiene (10:52)
- Optional: More Macros Examples (9:12)

Section 6

Section 6 Reading Notes

- ✔ Section Introduction (2:08)
- ✔ Datatype-Programming in Racket Without Structs (13:21)
- ✔ Datatype-Programming in Racket With Structs (9:34)
- ✔ Advantages of Structs (8:13)
- ✔ Implementing Programming Languages (10:07)
- ✔ What Your Interpreter Can and Cannot Assume (13:47)
- ✔ Implementing Variables and Environments (6:29)
- ✔ Implementing Closures (6:32)
- ✔ Optional: Are Closures Efficient? (9:03)
- ✔ Racket Functions As “Macros” For Interpreted Language (9:15)
- ✔ ML Versus Racket (8:53)
- ✔ What is Static Checking? (10:00)
- ✔ Soundness and Completeness (9:46)
- ✔ Weak Typing (9:39)
- ✔ Static Versus Dynamic Typing, Part One (9:40)
- ✔ Static Versus Dynamic Typing, Part Two (13:47)
- ✔ Optional: eval and quote (7:58)

✔ Section 7

- Section 7 Reading Notes
- Introduction to Ruby (10:08)

Classes and Objects (12:37)

Object State (13:14)

Visibility (8:39)

A Longer Example (12:19)

Everything is an Object (8:15)

Class Definitions are Dynamic (7:42)

Duck Typing (7:28)

Arrays (11:13)

Blocks (11:34)

Using Blocks (6:24)

Procs (7:08)

Hashes and Ranges (8:59)

Subclassing (10:11)

Why Use Subclassing? (7:16)

Overriding and Dynamic Dispatch (10:56)

Method-Lookup Rules, Precisely (11:31)

Dynamic Dispatch Versus Closures (9:38)

Optional: Dynamic Dispatch Manually in Racket (15:56)

Section 8

Section 8 Reading Notes

OOP Versus Functional Decomposition (12:51)

Adding Operations or Variants (11:08)

Binary Methods with Functional Decomposition (7:18)

Double Dispatch (14:46)

Optional: Multimethods (6:36)

Multiple Inheritance (10:18)

Mixins (11:52)

Interfaces (7:33)

Optional: Abstract Methods (8:51)

Subtyping From the Beginning (10:43)

The Subtype Relation (8:20)

Depth Subtyping (8:50)

Optional: Java/C# Arrays (9:11)

Function Subtyping (11:17)

Subtyping for OOP (11:44)

Generics Versus Subtyping (8:16)

Bounded Polymorphism (8:05)

Course Wrapup

Optional: Course Content Wrapup (9:43)

Optional: MOOC Wrapup (9:39)