FP Overview

Paradigms overview:

- Imperative
    - OOP
    - Structural
    - Procedural
- Non-Imperative
    - Logical
    - Functional
    - Declarative

Intro:

```
x = x + 1
```

- absurd
- syntax: literals, place for signs
- semantic: plus operation, assignment placement

History:

McCarthy, Church, Haskel Curry

λ calc vs Turing machine

λ Calculus overview

abstraction (fn construction)

```
λx.t (bind x to t)
```

application (fn call or result)

```
t(a)   (invoke t with binding)
```

reduction (redex, substitution)

```
a -> 5 -> t(5)
```

Applicative order vs normal order

```
macro system
side effects
```

Lambda functions & Closures

- function literal
- function value
- Functional type is First class citizen
- Higher order functions
- Currying

Recursion

- Tail recursion
- Y Combinator

GC

- Closures
- Free variables
- Bound variables

Type system

- implicit
- explicit
- strong
- weak

Pros and Cons

Pros

- Immutable - no side effects, parallel, optimization
- Lazy - speed, infinite data structures
- Declarative - literate programming

- Provable - it can be verified with mathematically methods
- REPL - speedup development
- Runtime evaluator & ast modification - Metaprogramming, Hot code swap

Cons

- Slooow, Memory consumed
- Too complicated
- Solid background
- IDE
- Debuggers
- FP Weather

Popular FPL

- Erlang
- Lisps (CL, Scheme, Clojure, Arc)
- Haskel
- F#

Books

- SICP
- TAPL
- PCL
- Little Schemer
- Concurrent programming in Erlang
- F# for C# developers

Conclusion

```
Lisp programmers know the value of everything and the cost of nothing
```