Reason for the Course

Interest in and use of functional programming has been growing for a number of years. Part of the reason for this growth is that functional programming is better at handling concurrency than imperative or object-oriented programming. For example Google adopted the standard functions reduce and map from functional programming to process search requests quickly using parallelism. Some people are promoting functional programming as better paradigm for producing software than object-oriented programming.

Facebook uses OCaml a functional language internally. Whats App backend servers that handle 10 billion messages a day are written in the functional language Erlang. A number of major banks in this country are using functional languages Clojure, Scala and Haskell. LinkedIn claims to have reduced their code base from 60,000 lines of code to 2,000 lines of code by switching from Ruby on Rails to using node.js which utilizes functional language features. The major new addition to Java 8, lambdas, comes from the functional world. The head of the Java development team has stated that they will continue to add more functional features to Java. So Java programmers will be using aspects of functional programming in the future. Like Google you will find uses of functional programming practices in non-functional languages.

What is Functional Programming

There is no single agreed on definition of functional programming. However there are features that are common among some functional programming languages. First functions are different. In an imperative (and object-oriented) language like Java functions are usually a sequence of assignment statements. In a function programming language functions are more like mathematical functions, they just compute a value and the value computed only depends on the function's input. Assignments statements are rarely used. Second functions can be assigned to variables and passed as arguments to other functions. In functional programming functions that operate on other functions are common. Lambdas, a functional feature, are just functions with no name that are usually passed as arguments. Another common feature of functional programming languages is immutable data. That is like Java's strings once you create a data structure you can not modify it. This is very useful in concurrent programming as you no longer have to worry about one thread modifying a value while another thread is reading the same value. This leads to a different style of programs than you see in imperative or object-oriented languages.
About the Course

As indicated above the structure and style of functional programs is different than imperative and object-oriented programs. The primary goal of this course will be to learn how to write good software in the functional paradigm. You will be writing be writing a number of programs in this course to give you experience in using functional style of programming. The course does not assume any prior experience with functional programming. We will use the Clojure programming language. No prior knowledge of Clojure is needed. The first third of the course will be spent on the basic syntax, libraries and basic practices. Then we will move on to more advanced features like concurrency and web programming. The end of the course will be spent on function programming style and patterns. A early rough draft of the course outline is:

Introduction Imperative verses Functional programming
  Clojure Basics
    Scalar Data Types, Composite types, Lazy evaluation
    Functional forms, recursion
    Lambdas, Closures, Macros
    Namespaces, Multi-methods
  Testing, Clojure - Java interoperation
Concurrence & Mutation
  Software Transactional memory
    Vars, Refs, Agents, Atoms, promises, futures
  Functional programming, Web Apps, Databases
  Functional programming patterns and practices

Course Prerequisites

CS 310 and CS 320.

Catalog Description

Basic concepts of functional programming; imperative verses functional functions, lambdas, first-class functions, higher order functions, immutability & side effects, recursion, currying, functional polymorphism, list compressions, lazy evaluation. Functional programming style, patterns & idioms. Functional concurrency. Programs & project using Clojure. No prior functional programming experience or knowledge of Clojure needed.

About Clojure

Rich Hickey spent two years designing Clojure. His goal was to produce a practical functional programming language. The first version of Clojure appeared in 2007. Clojure compiles to Java byte code, so it runs on the Java Virtual Machine (JVM). Clojure programs have full access to all Java libraries. Java programs can call Clojure code.
Clojure programs are packaged in Java jar files. Anywhere you use Java you can use Clojure and you can extend existing Java code using Clojure. ClojureScript is a version of Clojure that compiles to JavaScript. So Clojure can be used to anyplace that JavaScript is used. People are working on a version of Clojure for .net. There is a very active and growing Clojure community. Clojure is being used in places from small startup companies to large international banks.

Text and IDE

The course will use the text *Clojure Programming* by Chas Emerick, Brian Carper and Christophe Grand and *Functional Thinking: Paradigm Over Syntax* by Neal Ford. If you wish to start learning Clojure before the class starts there are Clojure plugins for Eclipse and IntelliJ. However the IDE Light Table is a very interactive IDE for Clojure which has some very useful features. Highly recommended and it is free. Download at: [http://www.lighttable.com](http://www.lighttable.com)