CSE 341
Lecture 29 b

Course wrap-up

slides created by Marty Stepp

http://www.cs.washington.edu/341/

One view of languages

	functional	**object-oriented**
statically typed	ML	Java
dynamically typed	Scheme	JavaScript

A broader view

Not all languages are functional or OO!

- logic languages (e.g., Prolog)
- scripting languages (Perl, Python, Lua)
- query languages (SQL)
- purely functional languages (Haskell; no ref or set!)
- visual languages, spreadsheet languages, GUI-builders, text-formatters, hardware-synthesis, ...
- languages with heavy support for parallel programming

Why did we do this?

- the time needed to "pick up" a new language will drop dramatically (though you have to learn its libraries, too)
- use mutation for what it's good for; not to create brittle programs with unseen dependencies
- syntax matters, but it's not everything
- apply idioms in languages besides where you saw them
- recognize that language-design is hard; semantics should not be treated lightly; more syntax is not always better

Big ideas

- code runs in environments; scope/resolution matters
- recursive data is processed with recursive functions
- without mutation, copying vs. aliasing is indistinguishable
- closures have many powerful uses
- (dis-) advantages of static typing (and what is checked)
- when evaluation occurs is important (thunks/macros)
- OO vs. FP: many similarities and a couple big differences
- parametric polymorphism vs. subtyping
- can embed a language in another via interpreters/macros

Big picture questions

- Which language we learned is your favorite? Why?
  - Least favorite?
- What are the pros and cons of static/dynamic typing?
- What are some benefits of coding in a functional style?
- How does a functional language handle extensibility and reusable code, as opposed to how OO languages do it?

What next?

- learn more about the languages we covered
  - be careful/honest when listing them on your resume...!
- learn a language similar to / inspired by ones we saw
  - **Scala**: functional/OO mixture that runs on Java VM
  - **F#**: Microsoft's ML clone; can interact with C# code
  - **C#**: Microsoft's Java clone
  - **Clojure**: Scheme/Lisp dialect that runs on Java VM
  - **Scala**/**Ruby**/**Lua**: dynamic and high-level, like JavaScript
- take **CSE 401** (Compilers)
  - learn much more about how compilers/interpreters work