COMP302 Lecture 8 28 September 2016

Late binding
- x = 2 in foo

Early binding
- x = 1 in foo

Closures and continuations
see 8c302
To execute the code, we need the code, and we need the environment. If we kept this around, we can execute this at any time. Could package up code + environment = "thunk".
If we package a function + environment = "closure".

e.g. *see 8c302-1*
When executing:
1. Evaluating the arguments

I have a global environment with z and foo inside.
When calling foo, I am creating a new environment, which I attach to the parent. Contains a, b, i.
I then execute the code with the environment and its parents.
➔ I kept the code and environment together, can execute the code later.

I can also package the code and environment after the call. *see 8c302-2*

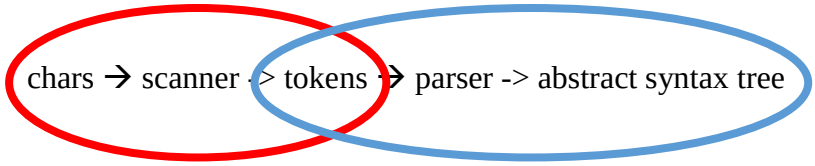
How and why would we do this?
In JS, we do get closures when we return functions. *see 8c302.js*
N.B. we do create environments in calls.
- We can "capture" variables this way.
  - We can implement "objects" !
- This is also how you can hide data

We can use let to fix this
We can also fix this with another function into helloMaker

Constructing languages
- Need to define our language
  - very formal way
  - maps to an implementation easily
- Syntax: "what is allowed? What can I express?"
- Semantics: "what does it mean/do?"

Starting with syntax
- Stream of characters.
- Characters → compiler/Virtual Machine/exec → executable/displayed values
- Dividing the second part into two pieces:
  - chars → Front End → Back End → exec.

- Syntax understanding is in FE. FE also split in pieces



1
2