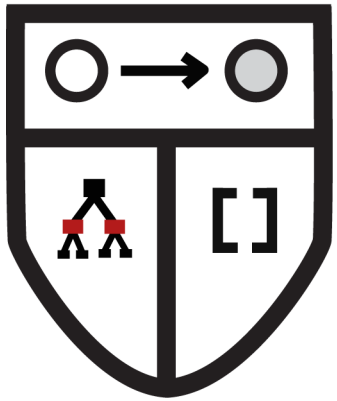

CS 3110

Functors

Nate Foster

Spring 2019

Please try to sit with your team & have your iClicker out and ready.

Today's music: "Uptown Funk" by Mark Ronson feat. Bruno Mars

Review

Previously in 3110:

- modules, structures, signatures, abstract types
- aspects of modularity: namespaces, abstraction

Today:

- code reuse: functors and includes

Review

Encapsulation: hide parts of module from clients

```
module type Stack = sig
  type 'a t
  val push : 'a -> 'a t -> 'a t
end

module ListStack : Stack = struct
  type 'a t = 'a list
  let push x s = x::s
end
```

type constructor **t** is *abstract*: clients of this signature know the type exists but not what it is

Review

Encapsulation: hide parts of module from clients

```
module type Stack = sig
  type 'a t
  val push : 'a -> 'a t -> 'a t
end

module ListStack : Stack = struct
  type 'a t = 'a list
  let push x s = x::s
end
```

module is *sealed*: all definitions in it except those given in signature **`Stack`** are hidden from clients

FUNCTORS

(funk you up?)

Cornell (CS) funk you up:
https://www.youtube.com/watch?v=Au56Ah92Ulk

Functors are "functions" on structures

Demo

Matching

A structure **`Struct`** matches a signature **`Sig`** if:

1. **`Struct`** defines every declaration in **`Sig`**

2. The type of each definition in **`Struct`** is the same as or more general than the declaration in **`Sig`**

Re-using code

PARAMETERIZED MODULE: TEST SUITE

Demo

Re-using code

PARAMETERIZED MODULE: MAP

https://www.cs.cornell.edu/courses/cs3110/2018fa/manual-4.6/libref/Map.html

Demo

INCLUDES

Demo

Code reuse from includes

- Interface inheritance
- Implementation inheritance

Upcoming events

- [Today] Foster OH in Gates 432
- [Tonight] Level up!

This is higher-order funk.

THIS IS 3110