



CS 3110

Functors

Nate Foster

Spring 2020

Today's music: Uptown Funk
by Mark Ronson, featuring Bruno Mars

CLICKER QUESTION 1

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

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

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

Review

Encapsulation: hide parts of module from clients

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

```
module ListStack : StackSig = 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
StackSig are hidden from clients

CLICKER QUESTION 2



FUNCTORS

(funk you up?)

Cornell (CS) funk you up:

<https://www.youtube.com/watch?v=Au56Ah92Ulk>

Functors are
“functions”
on structures

Matching (review + more)

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 modules

TEST SUITE BUILDER

Demo

Re-using code: Parameterized module

STANDARD LIBRARY MAP

<http://caml.inria.fr/pub/docs/manual-ocaml/libref/Map.html>

Demo

Re-using code: Interface and implementation inheritance

INCLUDES

Demo

Upcoming events

- [last night] A1 due
- [today] A2 out
- [tomorrow night] R3 due

This is higher-order funk.

THIS IS 3110