Elm

* Statically typed
* Inspired by Haskell (Haskell for webpages)
* Compiled (to Javascript)
* Immutable data structures
* Type inferencing
* Partial application, currying

Primitive types

* Int
* Float
* String
* Char
* Bool

Functions

* Optional typing, partial application, composition

Homogenous, iterable data types

* List, arrays, set, and dictionaries

About

* Type functional programming for the web
* Aims to compete with javascript, a language that would be basically compiled down to Javascipt, HTML and CSS.
* The goal of ELM is to bring approachability to functional languages and bring maintainability to Javascript programs—bring functional programming to frontend.

Why choose ELM?

* Purity
  + Every function in ELM is a pure function, it operates on inputs and returns outputs.
* Immutabiity
  + All data in ELM is immutable
  + In adding new data to a list, it’s basically creating a whole new list back
  + Data is always design

**Description of languages**

**Language A**

* Language features
* Purpose (Intended use) and Motivations
* History (Authors, Revisions, Adoption)

**Language B**

* Language features
* Purpose (Intended use) and Motivations
* History (Authors, Revisions, Adoption)

**Feature 1**

**Description of feature in Language A not present in B**

Include code examples for A and (if useful) B

**Advantages in having the feature**

**Advantages in not having the feature**

**How to implement similar functionality in B**

**Feature 2**

**Description of feature in Language A not present in B**

Include code examples for A and (if useful) B

**Advantages in having the feature**

**Advantages in not having the feature**

**How to implement similar functionality in B**

2 aspects discussed in class (e.g. numbers, scope, functions, lambdas, closures, types)

**Functions**

Elm supports two kind of functions, anonymous and named.