```
data FBAE where
        Num :: Int -> FBAE
        Plus :: FBAE -> FBAE -> FBAE
        Minus :: FBAE -> FBAE -> FBAE
        Bind :: String -> FBAE -> FBAE -> FBAE
        Id :: String -> FBAE
        App :: FBAE -> FBAE -> FBAE
        Lambda :: String -> FBAE -> FBAE
        Closure :: String -> FBAE -> Env -> FBAE


t ::= Num
        | True
        | False
        | t+t
        | t-t
        | bind id = t in t
        | id
        | if t then t else t
        | t <= t
        | t && t
        | isZero t

v ::= Num
        | True
        | False


eval :: AE -> Maybe Int
eval e (Lambda i b) = return (ClosureV i b e)
eval e (App f a) = do { (ClosureV i b j) <- eval e f;
                        v <- eval e a;
                        eval (i,v):j b}

elab (BindX i v b) = (App (Lambda i (elab b)) (elab v))



typeof :: ABE -> Maybe TABE
typeof (Plus l r) = do {   TNum <- typeof l;
                           TNum <- typeof r;
                           return TNum}
```

First order: not in arguments. Not in returns
Higher order: arguments, returns,
First class: + function pointers
Lambda expression - function value
Application - call lambda with actuals
Domain - types of formals
Range - type of return

```
bind inc = (lambda x in x+1) in
        (inc)(1)
```

Dynamic scope - depends on when evaluated
Static scope - depends on where defined (closures)

Enrichment - adding features to a new language
Extension - adding new features by modifying the evaluation function
Derived form - defines new features in terms of existing expressions
Elaboration - translating one abstract syntax into another
External language - language that is input to an elaborator (bind)
Internal language - language that is output by an elaborator (lambda)

Dynamic:

```
bind f = lambda x in if x=0 then 1 else x*(f)(x-1) in
        (f)(2)
```

Static:

```
eval e (Fix f) = do {(ClosureV i b e') <- eval e f;
                                        eval e' (subst i (Fix (Lambda i b)) b)}
```

bind f = fix(lambda g in (lambda x in if x=0 then 1 else x*(g)(x-1)))