

UMass Boston Computer Science  
**CS450 High Level Languages** (section 2)

# **Variables and Environments**

Wednesday, November 15, 2023

# *Logistics*

- HW 7 out
  - due: Sun 11/19 11:59 pm EST
  - Really due: Wed 11/22 11:59 pm EST
  - (no hw over Thanksgiving)
- Reminder: **Pass/Fail & Course Withdraw Deadline**
  - tomorrow Thurs 11/16

Last Time

# Introducing: The “CS450js” Programming Lang!

Programmer writes:



```
;; A 450jsExpr is one of:  
;; - Number  
;; - String  
;; - (list '+ 450jsExpr 450jsExpr)  
;; - (list '- 450jsExpr 450jsExpr)
```

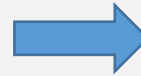


“eval450js”

```
;; A 450jsResult is one of:  
;; - Number  
;; - String  
;; - NaN
```

“meaning” of the program

parse450js



```
;; A 450jsAST is one of:  
;; - (num Number)  
;; - (str String)  
;; - (add 450jsAST 450jsAST)  
;; - (sub 450jsAST 450jsAST)
```

```
(struct num [val])  
(struct str [val])  
(struct add [lft rgt])  
(struct sub [lft rgt])
```



run450js

(JS semantics)

# “CS450js” Examples

Programmer writes:



```
;; A 450jsExpr is one of:  
;; - Number  
;; - String  
;; - (list '+ 450jsExpr 450jsExpr)  
;; - (list '- 450jsExpr 450jsExpr)
```



“eval450js”

```
;; A 450jsResult is one of:  
;; - Number  
;; - String  
;; - NaN
```

```
(check-equal? (eval450js 100) 100)
```

```
(check-equal? (eval450js “one”) “one”)
```

```
(check-equal? (eval450js ‘(+ 100 200)) 300)
```

```
(check-equal? (eval450js ‘(+ “cs” 450)) “cs450”)
```

```
(check-equal? (eval450js ‘(+ 1 2 3 4)) ??? )
```

```
match: no matching clause for ‘(+ 1 2 3 4)
```

# Dynamic Errors (e.g, Exceptions)

When: a function argument:

1. Comes from arbitrary users
2. Has a sufficiently complex data definition
  - (So that signature and contracts are not enough)

Then: **dynamic errors** may be needed

# Parsing: “CS450js” Programs

```
;; parse450js: 450jsExpr -> 450jsAST
;; Converts a CS450js Lang surface program to its AST
```

```
;; A 450jsExpr is one of:
;; - Number
;; - String
;; - (list '+ 450jsExpr 450jsExpr)
;; - (list '- 450jsExpr 450jsExpr)
```

```
(define (450jsexpr? s)
  (or (number? s)
      (string? s)
      (cons? s)))
```

```
(define/contract (parse450js s)
  (-> 450jsexpr? 450jsAST?)
  (match s
    [(? number?) (num s)]
    [(? string?) (str s)]
    [ `( + ,x ,y) (add (parse450js x) (parse450js y))]
    [ `( - ,x ,y) (sub (parse450js x) (parse450js y))]
  )))
```

???

```
;; A 450jsAST is one of:  
;; - (num Number)  
;; - (str String)  
;; - (add 450jsAST 450jsAST)  
;; - (sub 450jsAST 450jsAST)
```

```
(struct num [val])  
(struct str [val])  
(struct add [lft rgt])  
(struct sub [lft rgt])
```

???

```
(define (450jsAST? s)  
  (or (num? s) (str? s)  
      (add? s) (sub? s)))
```

# Interlude: Inheritance and “Super” Structs

```
;; A 450jsAST is one of:  
;; - (num Number)  
;; - (str String)  
;; - (add 450jsAST 450jsAST)  
;; - (sub 450jsAST 450jsAST)
```

```
(struct num [val])  
(struct str [val])  
(struct add [lft rgt])  
(struct sub [lft rgt])
```



```
;; A 450jsAST is one of  
;; - (num Number)  
;; - (str String)  
;; - (add 450jsAST 450jsAST)  
;; - (sub 450jsAST 450jsAST)
```

```
(struct 450jsAST [])  
(struct num 450jsAST [val])  
(struct str 450jsAST [val])  
(struct add 450jsAST [lft rgt])  
(struct sub 450jsAST [lft rgt])
```

“abstract” struct  
(implicitly defines  
450jsAST? predicate)

```
(define (450jsAST? s)  
  (or (num? s) (str? s)  
      (add? s) (sub? s)))
```

Alternatively ...

“super” struct declaration

e.g., if  $p = (\text{sub } (\text{num } 1) (\text{num } 2))$  then both  
 $(\text{sub? } p) = \text{true}$  and  
 $(450jsAST? p) = \text{true}$



# Interlude: Inheritance and “Super” Structs

This kind of “polymorphic” “abstract” data definition is what we’ve been creating all semester!

“super” structs are just a convenience for the same thing (when all itemizations are structs)

WAIT, I thought “Inheritance is bad”???

NO, accepted OO principles says:

Inheritance of implementations is bad ❌

Interfaces and abstract classes are ok ✅

```
;; A 450jsAST is one of
;; - (num Number)
;; - (str String)
;; - (add 450jsAST 450jsAST)
;; - (sub 450jsAST 450jsAST)
(struct 450jsAST [])
(struct num 450jsAST [val])
(struct str 450jsAST [val])
(struct add 450jsAST [lft rgt])
(struct sub 450jsAST [lft rgt])
```

“abstract” struct  
(implicitly defines  
450jsAST? predicate)

# Parsing: “CS450js” Programs

```
;; parse450js: 450jsExpr -> 450jsAST  
;; Converts a CS450js Lang surface program to its AST
```

```
;; A 450jsExpr is one of:  
;; - Number  
;; - String  
;; - (list '+ 450jsExpr 450jsExpr)  
;; - (list '- 450jsExpr 450jsExpr)
```

```
(define/contract (parse450js s)  
  (-> 450jsExpr? 450jsAST?)  
  (match s  
    [(? number?) (num s)]  
    [(? string?) (str s)]  
    [`(+ ,x ,y) (add (parse450js x) (parse450js y))]  
    [`(- ,x ,y) (sub (parse450js x) (parse450js y))]  
    [_ (error ... )]))
```

function argument:

1. Comes from arbitrary users
2. Has sufficiently complex data definition where contracts are insufficient

# Interlude: Racket exceptions

Exceptions are just special structs

Super struct (enables using exception API)

```
(struct exn:fail:syntax:cs450js exn:fail:syntax [])
```

```
(define/contract (parse450js s)
  (-> 450jsexpr? 450jsAST?)
  (match s
    [(? number?) (num s)]
    [(? string?) (str s)]
    [ `( + ,x ,y) (add (parse450js x) (parse450js y))]
    [ `( - ,x ,y) (sub (parse450js x) (parse450js y))]
    [ _ (raise-syntax-error
         'parse450js "not a valid CS450js program" s
         #:exn exn:fail:syntax:cs450js))]))
```

# “CS450js” Examples

Programmer writes:



```
;; A 450jsExpr is one of:  
;; - Number  
;; - String  
;; - (list '+ 450jsExpr 450jsExpr)  
;; - (list '- 450jsExpr 450jsExpr)
```



“eval450js”

```
;; A 450jsResult is one of:  
;; - Number  
;; - String  
;; - NaN
```

```
(check-equal? (eval450js '(+ 1 2 3 4)) ??? )
```

```
match: no matching clause for '(+ 1 2 3 4)
```

```
parse450js: not a valid CS450js program in: (+ 1 2 3 4)
```

```
(check-exn exn:fail:syntax:cs450js?  
  (λ () (eval450js '(+ 1 2 3 4))))
```

# HW7: Let's Get Started Early

- Add some boolean constructs:
  - true / false literal values
  - “loose equality” comparator (look it up)
  - If-then-else, with “truthy” true / false (look it up)

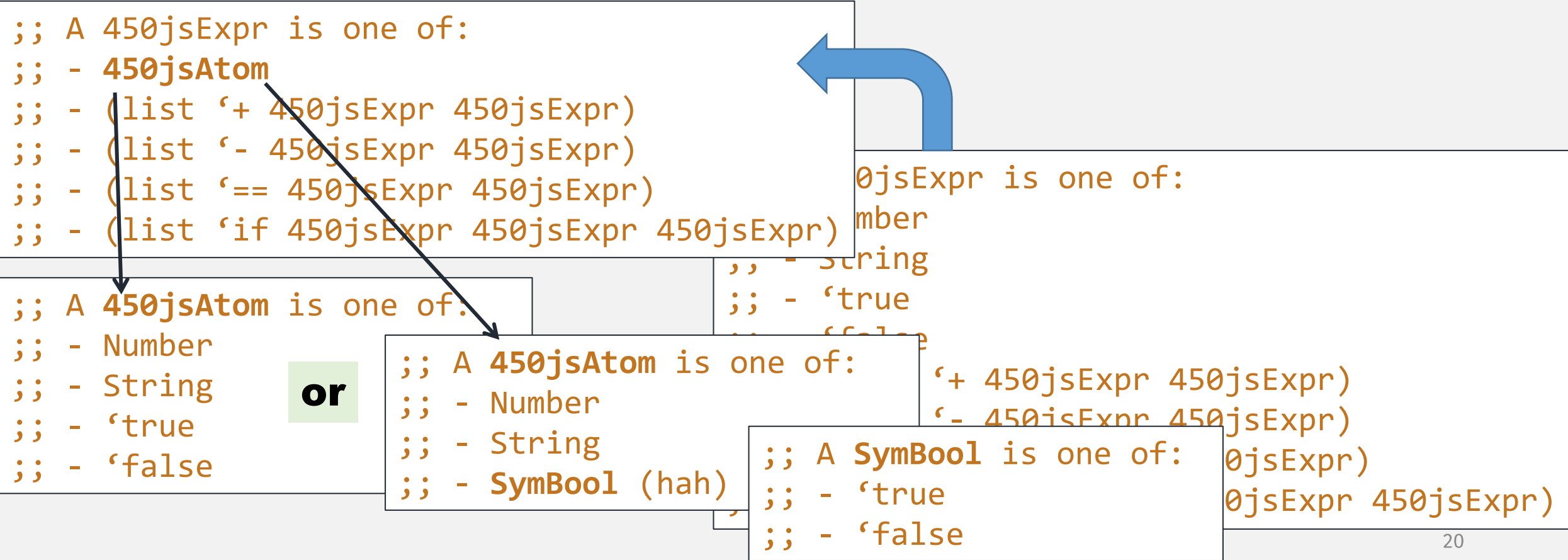
```
;; A 450jsExpr is one of:  
;; - Number  
;; - String  
;; - (list '+ 450jsExpr 450jsExpr)  
;; - (list '- 450jsExpr 450jsExpr)
```



```
;; A 450jsExpr is one of:  
;; - Number  
;; - String  
;; - 'true  
;; - 'false  
;; - (list '+ 450jsExpr 450jsExpr)  
;; - (list '- 450jsExpr 450jsExpr)  
;; - (list '== 450jsExpr 450jsExpr)  
;; - (list 'if 450jsExpr 450jsExpr 450jsExpr)
```

# HW7: Let's Get Started Early

- Refactor data definitions? (your design choice)



- Repo: [cs450f23/lecture20-inclass](#)
- File: [cs450js-tests-<your last name>.rkt](#)

# In-class Coding 11/20: write hw7 tests

```
;; parse450js: 450jsExpr -> 450jsAST
;; Parses "CS450js Lang" program to AST
```

```
;; run450js: 450jsAST -> 450jsResult
;; Computes result of running CS450js AST
```

```
(define eval450js (compose run450js parse450js))
```

```
;; A 450jsExpr is one of:
;; - Number
;; - String
;; - 'true
;; - 'false
;; - (list '+ 450jsExpr 450jsExpr)
;; - (list '- 450jsExpr 450jsExpr)
;; - (list '== 450jsExpr 450jsExpr)
;; - (list 'if 450jsExpr 450jsExpr 450jsExpr)
```

```
;; "adding" bools
(check-equal? (eval450js '(+ true false)) 1)
```

```
;; weird ==
(check-true (eval450js '(= (+ 10 90) (+ "10" "0"))))
```

```
;; js "truthy true"
(check-equal? (eval450js '(if 10 100 200)) 100)
;; js "truthy false"
(check-equal? (eval450js '(if (- 100 100) "a" "b")) "b")
```

- For `==`: Look into JavaScript "loose equality"
- For `if`: look into JavaScript "truthy" values
- I will collect so all can use for hw (even if it's wrong!)

```
;; - Number
;; - String
;; - Boolean
;; - NaN
```

# Introducing: The "CS450js" Programming Lang!

Programmer writes:

Next Feature: Variables?

```
;; A 450jsExpr is one of:
;; - Number
;; - String
;; - (list '+ 450jsExpr 450jsExpr)
;; - (list '- 450jsExpr 450jsExpr)
```

"eval450js"

```
;; A 450jsResult is one of:
;; - Number
;; - String
;; - NaN
```

parse450js

```
;; A 450jsAST is one of:
;; - (num Number)
;; - (str String)
;; - (add 450jsAST 450jsAST)
;; - (sub 450jsAST 450jsAST)

(struct num [val])
(struct str [val])
(struct add [lft rgt])
(struct sub [lft rgt])
```

run450js

(JS semantics)





# Adding Variables

;; A Variable is a Symbol

;; A 450jsExpr is one of:  
;; - Number  
;; - String  
;; - Variable  
;; - (list '+ 450jsExpr 450jsExpr)  
;; - (list '- 450jsExpr 450jsExpr)

parse450js

;; A 450jsAST is one of:  
(num Number)  
(str String)  
(var Symbol)  
(add 450jsAST 450jsAST)  
(sub 450jsAST 450jsAST)

**Q<sub>1</sub>**: What is the “meaning” of a variable?

**A<sub>1</sub>**: Whatever “value” it is bound to

**Q<sub>2</sub>**: Where do these “values” come from?

**A<sub>2</sub>**: Other parts of the program

;; A 450jsResult is one of:  
;; - Number  
;; - String  
;; - NaN

run450js

struct num [val])  
struct str [val])  
(struct var [name])  
(struct add [lft rgt])

The run function needs to “remember” these values (with an **accumulator!**)

# run450js, with an accumulator

```
;; run: 450jsAST -> 450jsResult  
;; Computes result of running CS450js AST
```

```
(define (run p)  
  ;; accumulator acc : Environment  
  ;; invariant: Contains variable+result pairs that are currently in-scope  
  (define (run/acc p acc)  
    (match p  
      [(num n) n]  
      [(add x y) (450+ (run x) (run y))]))  
  (run/acc p ??? ))
```

# Environments

- A data structure that “associates” two things together
  - E.g., maps, hashes, etc
  - For simplicity, let’s use list-of-pairs

```
;; An Environment is one of:  
;; - empty  
;; - (cons (list Var 450jsResult) Environment)  
;; interpretation: a runtime environment for  
;; (ie gives meaning to) cs450js-lang variables  
;; if there are duplicates,  
;; vars at front of list shadow those in back
```

# Environments

- A data structure that “associates” two things together
  - E.g., maps, hashes, etc
  - For simplicity, let’s use list-of-pairs
- Needed operations:
  - `add : Env Var Result -> Env`
  - `Lookup : Env Var -> Result`

# run450js, with an Environment

## TODO:

- When are variables “added” to environment
- Initial environment?

```
;; run: 450jsAST -> 450jsResult  
;; Computes result of running CS450js AST
```

```
(define (run p)  
  ;; accumulator env : Environment  
  ;; invariant: Contains variable+result pairs that are in-scope  
  (define (run/acc p env)  
    (match p  
      [(num n) n]  
      [(var x) (lookup env x)]  
      [(add x y) (450+ (run x env) (run y env))]))  
  (run/acc p ??? ))
```

# Adding Variables

```
;; A 450jsExpr is one of:  
;; - Number  
;; - String  
;; - Variable  
;; - `(bind [x ,450jsExpr] ,450jsExpr)  
;; - (list '+ 450jsExpr 450jsExpr)  
;; - (list '- 450jsExpr 450jsExpr)
```

parse450js



```
;; A 450jsAST is one of:  
;; - (num Number)  
;; - (str String)  
;; - (var Symbol)  
;; - (bind Symbol 450jsAST 450jsAST)  
;; - (add 450jsAST 450jsAST)  
;; - (sub 450jsAST 450jsAST)
```

```
(struct num [val])  
(struct str [val])  
(struct var [name])  
(struct bind [var expr body])  
(struct add [lft rgt])  
(struct sub [lft rgt])
```

# run450js, with an Environment

```
;; run: 450jsAST -> 450jsResult
```

```
(define (run p)
  ;; accumulator env : Environment
  ;; invariant: Contains variable names that are in scope
  (define (run/acc p env)
    (match p
      [(num n) n]
      [(var x) (lookup env x)]
      [(bind x e body) (run body (add env x (run e env)))]
      [(add x y) (450+ (run x env) (run y env))]))
    (run/acc p ??? ))
```

1. Compute 450jsResult that variable x represents

2. add variable x to environment

3. run body with that new environment

• Repo: `cs450f23/lecture20-inclass`

• File: `env-<your last name>.rkt`

# In-class Coding 11/20: write env ops

- Needed operations:

- `add : Env Var Result -> Env`
- `Lookup : Env Var -> Result`

```
;; An Environment is one of:
```

```
;; - empty
```

```
;; - (cons (list Var 450jsResult) Environment)
```

```
;; interpretation: a runtime environment for
```

```
;; (ie gives meaning to) cs450js-lang variables
```

```
;; if there are duplicates,
```

```
;; vars at front of list shadow those in back
```

Think about examples where this happens!



# No More Quizzes!

but push your in-class work to:

Repo: `cs450f23/lecture20-inclass`