

## Computer Science I

Professor Tom Ellman  
Lecture 13

## Types of Values in Scheme

- Numbers: 1, 2, 17, 3.14159.
- Symbols: bill, hillary, al, tipper.
- Lists: (bill hillary al tipper), (bill hillary chelsea).
- Booleans: #t, #f.
- Strings: "To be or not to be.", "Vassar College".
- Procedures: #<PROCEDURE +>, #<PROCEDURE \*>.

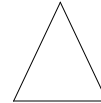
## There are More Things in Heaven and Earth ...

- Scheme provides only a few types of data.
- Our programs need to reason about many more kinds of objects: People, places, accounts, courses, computers, etc.
- Scheme allow the programmer to construct more complex types of data.
- The possibilities are endless.

## A Simple Graphics Application

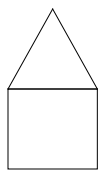


Rectangle



Triangle

...and anything you  
can build from them ...

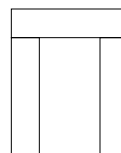


House

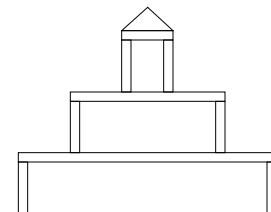


Tower

...and anything else you  
can build from them ...



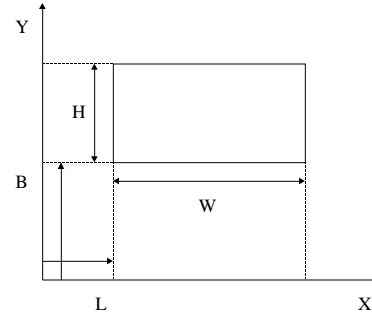
Arch



Pyramid

## What is a Rectangle?

- It depends on what you want to do with it.
- Let's suppose we want to draw it.
- What information do we need?
  - The height of the rectangle.
  - The width of the rectangle.
  - The location of the rectangle.



## Using Lists to Implement the Rectangle Data Type

```
Welcome to Dr. Scheme.  
> (define rect (list 10 10 20 40))  
> (draw-rectangle rect)  
#t  
  
> ...Do other work...  
  
> (draw-rectangle rect)  
#t  
...Etc...
```

## What Have We Gained?

- We have “glued” together several pieces of data to create a composite data object.
- We have given the data a name.
- We can refer to the composite data object as a whole.
- We can use the composite data object over and over simply by referring to its name.

## A Procedure to Draw a Rectangle

```
(define draw-rectangle  
  (lambda (r)  
    (begin (goto (car r) (cadr r))  
           (drawto (car r) (+ (cadr r) (caddr r)))  
           (drawto (+ (car r) (caddr r)) (+ (cadr r) (caddr r)))  
           (drawto (+ (car r) (caddr r)) (cadr r))  
           (drawto (car r) (cadr r))))))
```

## Problems with this Implementation

- What if the programmer forgets the order of the (L B W H) data on the list?
- What if the programmer forgets whether the data in the variable RECT represents a rectangle or something else?

## A Better Implementation of the Rectangle Data Type

```
Welcome to Dr. Scheme.
>>> (define rect (make-rectangle 10 10 20 40))
>>> rect
(rectangle 10 10 20 40)
>>> (draw-rectangle rect)
#t

>>> ...Do other work...

>>> (draw-rectangle rect)
#t

...Etc...
```

## Constructor Function

```
(define make-rectangle
  (lambda (left bottom width height)
    (list 'rectangle left bottom width height)))
```

## Selector Functions

```
(define rectangle-left (lambda (r) (cadr r)))

(define rectangle-bottom (lambda (r) (caddr r)))

(define rectangle-width (lambda (r) (caddrdr r)))

(define rectangle-height (lambda (r) (car (cddddr r))))
```

Now the programmer can forget the order of the (L B W H) data on the list that represents the rectangle.

## More Selector Functions

```
(define rectangle-right
  (lambda (r) (+ (rectangle-left r) (rectangle-width r))))

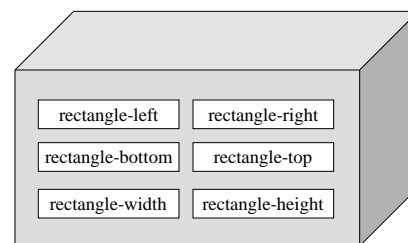
(define rectangle-top
  (lambda (r) (+ (rectangle-bottom r) (rectangle-height r))))
```

Now the programmer the programmer can forget whether the top and right sides of the rectangle are stored on the list or are computed on the fly.

## A Procedure to Draw a Rectangle

```
(define draw-rectangle
  (lambda (r)
    (begin (goto (rectangle-left r) (rectangle-bottom r))
           (drawto (rectangle-left r) (rectangle-top r))
           (drawto (rectangle-right r) (rectangle-top r))
           (drawto (rectangle-right r) (rectangle-bottom r))
           (drawto (rectangle-left r) (rectangle-bottom r)))))
```

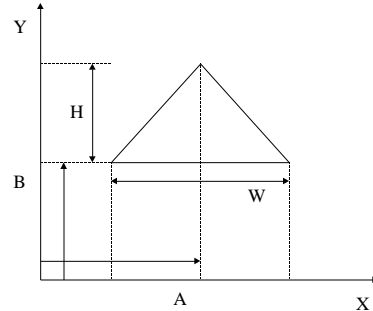
## Rectangle Data Abstraction



The constructor and selector functions let us treat a rectangle as a *black box*. The data is inside the box. We access the data through the interface provided on the outside of the box. We can forget about what is actually inside the box.

## What is a Triangle?

- Let's restrict ourselves to isosceles triangles, i.e. triangles with two equal length sides.
- Let's suppose we want to draw it.
- What information do we need?
  - Location of the base midpoint
  - Width of the base.
  - Height of the base.



## Constructor Function

```
(define make-triangle
  (lambda (apex bottom width height)
    (list 'triangle apex bottom width height)))
```

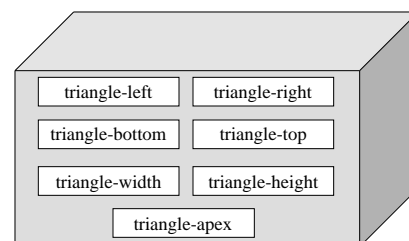
## Selector Functions

```
(define triangle-apex (lambda (t) (cadr t)))
(define triangle-bottom (lambda (t) (caddr t)))
(define triangle-width (lambda (t) (caddr t)))
(define triangle-height (lambda (t) (car (cddddr t))))
(define triangle-left
  (lambda (t) (- (triangle-apex t)
                 (/ (triangle-width t)2))))
(define triangle-right
  (lambda (t) (+ (triangle-apex t)
                 (/ (triangle-width t)2))))
(define triangle-top
  (lambda (t) (+ (triangle-bottom t)
                 (triangle-height t))))
```

## A Procedure to Draw a Triangle

```
(define draw-triangle
  (lambda (t)
    (begin (goto (triangle-left t) (triangle-bottom t))
            (drawto (triangle-apex t) (triangle-top t))
            (drawto (triangle-right t) (triangle-bottom t))
            (drawto (triangle-left t) (triangle-bottom t))))
```

## Triangle Data Abstraction



The constructor and selector functions let us treat a triangle as a *black box*. The data is inside the box. We access the data through the interface provided on the outside of the box. We can forget about what is actually inside the box.

## Type Predicates

```
(define rectangle?  
  (lambda (r) (and (pair? r)  
                    (equal? (car r) 'rectangle))))  
  
(define triangle?  
  (lambda (t) (and (pair? t)  
                   (equal? (car t) triangle))))
```

## A Procedure to Draw Objects

```
(define draw-object  
  (lambda (x)  
    (cond ((rectangle? x) (draw-rectangle x))  
          ((triangle? x) (draw-triangle x))  
          (else #f))))
```

## Data Abstraction

- Building composite data structures out of simple ones.
- Give the composite data structure a name.
- Treat the composite data structure as a single object.
- Write constructor and selector functions that hide the representation of the data structure.
- Treat the object as a black box: We use the selector functions to access the data and forget about what's actually inside the box.