

# Metaprogramming with Macros

Eugene Burmako

École Polytechnique Fédérale de Lausanne  
<http://scalamacros.org/>

10 September 2012

# Macros

# Macros

Macros realize the notion of textual abstraction.

# Macros

Macros realize the notion of textual abstraction.

Textual abstraction:

- ▶ Recognize pieces of text that match a specification
- ▶ Replace them according to a procedure

## Example

```
(let (x 42) (print x))
```

## Example

```
(let (x 42) (print x))
```

```
((lambda (x) (print x)) 42)
```

## Step 1. Recognize pieces of text

```
(let (x 42) (print x))
```

```
(defmacro let args
```

```
((lambda (x) (print x)) 42)
```

## Step 1. Recognize pieces of text

```
(let (x 42) (print x))
```

```
(defmacro let args
```

```
((lambda (x) (print x)) 42)
```



## Step 2. Replace them according to a procedure

```
(let (x 42) (print x))
```

```
(defmacro let args  
  (cons  
    (cons 'lambda  
          (cons (list (caar args))  
                  (cdr args))))  
    (cdar args)))
```

```
((lambda (x) (print x)) 42)
```

## Step 2. Replace them according to a procedure

```
(let (x 42) (print x))
```

```
(defmacro let args  
  (cons  
    (cons 'lambda  
          (cons (list (caar args))  
                  (cdr args)))  
    (cdar args)))
```

```
((lambda (x) (print x)) 42)
```

# The essence of macros

- ▶ Recognize pieces of text that match a specification
- ▶ Replace them according to a procedure

# Why macros?

- ▶ Deeply embedded DSLs (database access, testing)
- ▶ Optimization (programmable inlining, fusion)
- ▶ Analysis (integrated proof-checker)
- ▶ Effects (effect containment and propagation)
- ▶ ...

# Today's talk

Macrology is vast:

- ▶ Notation
- ▶ Variable capture
- ▶ Typechecking meta-programs
- ▶ Syntax extensibility
- ▶ ...

Surveyed papers are versatile as well.

# Today's talk

Going into all the details would be a genuine pleasure.

But instead let me tell you a story.

# Outline

The prelude of macros

The tale of bindings

The trilogy of tongues

The vision of the days to come

## Anaphoric if

```
(aif (calculate)
      (print it)
      (error "does not compute"))
```



## Anaphoric if

```
(aif (calculate)
      (print it)
      (error "does not compute"))
```

```
(let* ((temp (calculate))
        (it temp))
      (if temp
          (print it)
          (error "does not compute")))
```

## The aif macro

```
(aif (calculate)
     (print it)
     (error "does not compute"))
```

```
(defmacro aif args
```

```
(let* ((temp (calculate))
       (it temp))
  (if temp
      (print it)
      (error "does not compute"))))
```

## Start with a notation

```
(aif (calculate)
      (print it)
      (error "does not compute"))
```

```
(defmacro aif args
  (let* ((temp (car args))
         (it temp))
    (if temp
        (cadr args)
        (caddr args))))
```

```
(let* ((temp (calculate))
       (it temp))
  (if temp
      (print it)
      (error "does not compute")))
```

## Surround it with parentheses

```
(aif (calculate)
  (print it)
  (error "does not compute"))

(defmacro aif args
  (list 'let* (list (list 'temp (car args))
                    (list 'it 'temp))
    (list 'if 'temp
          (cadr args)
          (caddr args))))

(let* ((temp (calculate))
      (it temp))
  (if temp
    (print it)
    (error "does not compute")))
```

## Quasiquote

```
(aif (calculate)
      (print it)
      (error "does not compute"))
```

```
(defmacro aif args
  ' (let* ((temp .....))
        (it temp))
      (if temp
          .....
          .....)))
```

```
(let* ((temp (calculate))
        (it temp))
  (if temp
      (print it)
      (error "does not compute")))
```

## Unquote

```
(aif (calculate)
  (print it)
  (error "does not compute"))
```

```
(defmacro aif args
  '(let* ((temp ,(car args))
          (it temp))
    (if temp
      ,(cadr args)
      ,(caddr args))))
```

```
(let* ((temp (calculate))
      (it temp))
  (if temp
    (print it)
    (error "does not compute")))
```

## Unquote

```
(aif (calculate)
  (print it)
  (error "does not compute"))

(defmacro aif args

  '(let* ((temp ,(car args))
          (it temp))
    (if temp
      ,(cadr args)
      ,(caddr args))))

(let* ((temp (calculate))
      (it temp))
  (if temp
    (print it)
    (error "does not compute")))
```

## Macro by example (MBE)

```
(aif (calculate)
  (print it)
  (error "does not compute"))

(defmacro+ aif
  (aif cond then else)
  (let* ((temp cond)
        (it temp))
    (if temp
        then
        else)))

(let* ((temp (calculate))
      (it temp))
  (if temp
    (print it)
    (error "does not compute")))
```



# Interlude

- ▶ Macros are regular functions that happen to work with syntax objects
- ▶ Quasiquotes = static templates + dynamic holes

# Outline

The prelude of macros

The tale of bindings

The trilogy of tongues

The vision of the days to come

## Anaphoric if

```
(defmacro+ aif
  (aif cond then else)
  (let* ((temp cond)
        (it temp))
    (if temp then else)))
```

- ▶ So far macros are simple: define a function, recognize pieces of text and replace them with a template
- ▶ This is so immediately useful, that we could wrap up right now

But actually

The aif macro has two bugs

## What's wrong?

```
(defmacro+ aif
  (aif cond then else)
  (let* ((temp cond)
        (it temp))
    (if temp then else)))
```

## What's wrong?

```
(aif (calculate)
      (print it)
      (error "does not compute"))
```

```
(defmacro+ aif
  (aif cond then else)
  (let* ((temp cond)
         (it temp))
    (if temp then else)))
```

## What's wrong?

```
(aif (calculate)
      (print it)
      (error "does not compute"))
```

```
(defmacro+ aif
  (aif cond then else)
  (let* ((temp cond)
         (it temp))
    (if temp then else)))
```

## What's wrong?

```
(aif (calculate)
  (print it)
  (error "does not compute"))

(defmacro+ aif
  (aif cond then else)
  (let* ((temp cond)
        (it temp))
    (if temp then else)))

(let* ((temp (calculate))
      (it temp))
  (if temp
    (print it)
    (error "does not compute")))
```



## Bug #1: Violation of hygiene

```
(let ((temp 451°F))  
  (aif (calculate)  
    (print it)  
    (print temp)))
```

```
(defmacro+ aif  
  (aif cond then else)  
  (let* ((temp cond)  
    (it temp))  
    (if temp then else)))
```

```
(let ((temp 451°F))  
  (let* ((temp (calculate))  
    (it temp))  
    (if temp  
      (print it)  
      (print temp))))
```

## Bug #2: Violation of referential transparency

```
(let ((if hijacked))
  (aif (calculate)
    (print it)
    (error "does not compute")))
```

```
(defmacro+ aif
  (aif cond then else)
  (let* ((temp cond)
        (it temp))
    (if temp then else)))
```

```
(let ((if hijacked))
  (let* ((temp (calculate))
        (it temp))
    (if temp
      (print it)
      (error "does not compute"))))
```

## Old school

```
(defmacro+ aif
  (aif cond then else)

  (let* ((temp cond)
        (it temp))
    (if temp then else)))
```

## Old school

```
(defmacro+ aif
  (aif cond then else)
  (let ((temp (gensym)))
    (let* ((temp cond)
           (it temp))
      (if temp then else))))
```

And please don't rename core forms

# Interlude

- ▶ Cross-pollination of scopes can lead to inadvertent variable capture
- ▶ Violation of hygiene = def site harms call site
- ▶ Violation of referential transparency = call site harms def site

# Outline

The prelude of macros

The tale of bindings

The trilogy of tongues

The vision of the days to come

# Outline

The prelude of macros

The tale of bindings

The trilogy of tongues

The vision of the days to come