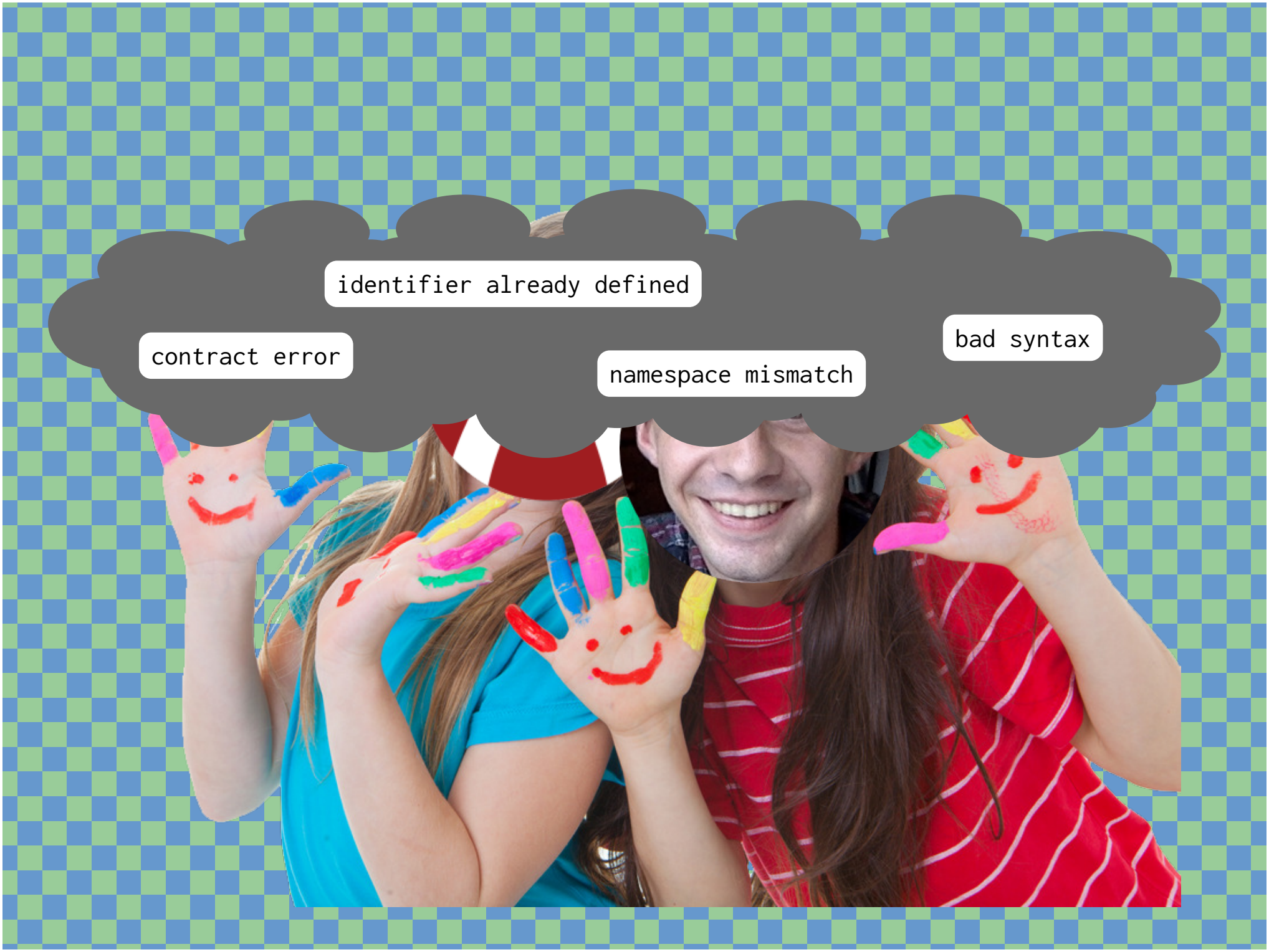
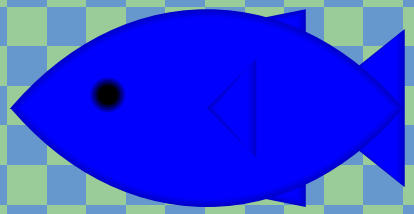
Fun and Games 2

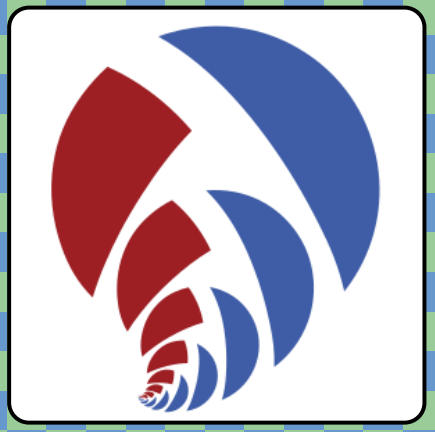
Ben Greenman

2021 11 07


identifier already defined
contract error
namespace mismatch
bad syntax

Reminder: Programming is fun!

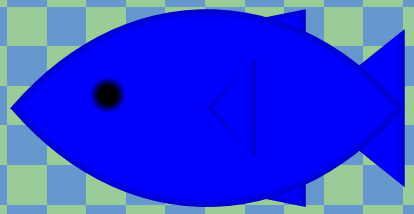


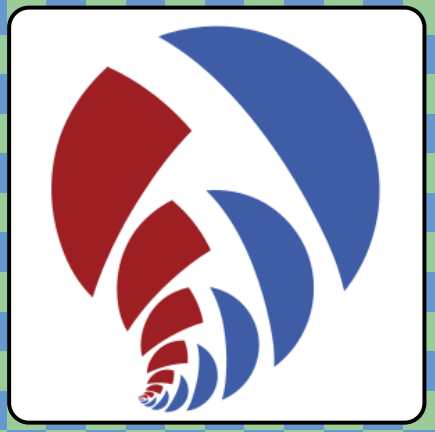


Standard Fish
2019

Quickscript
2020

Reminder: Programming is fun!





Standard Fish
2019

Quickscript
2020

Syntax Parse
2021

Extensible Functions

```
(interp0 '(+ 1 (+ 5 6)))
(interp1 '(+ 1 (* 5 6)))
```

Extensible Functions

```
(interp0 '(+ 1 (+ 5 6)))
(interp1 '(+ 1 (* 5 6)))
```

```
(define (fix f)
 ((λ (x) (f (λ (g) ((x x) g))))
  (λ (x) (f (λ (g) ((x x) g))))))

(define ((interp0 recur) e)
  (match e
    ....))

(define ((interp1 recur) e)
  (match e
    ....))
```

Extensible Functions

```
((fix interp0) '(+ 1 (+ 5 6)))
((fix interp1) '(+ 1 (* 5 6)))
```

```
(interp0 '(+ 1 (+ 5 6)))
(interp1 '(+ 1 (* 5 6)))
```

```
(define (fix f)
 ((λ (x) (f (λ (g) ((x x) g))))
  (λ (x) (f (λ (g) ((x x) g))))))

(define ((interp0 recur) e)
  (match e
    ....))

(define ((interp1 recur) e)
  (match e
    ....))
```

Extensible Functions

```
((fix interp0) '(+ 1 (+ 5 6)))
((fix interp1) '(+ 1 (* 5 6)))
```

```
(interp0 '(+ 1 (+ 5 6)))
(interp1 '(+ 1 (* 5 6)))
```

```
(define (fix f)
 ((λ (x) (f (λ (g) ((x x) g))))
  (λ (x) (f (λ (g) ((x x) g))))))

(define ((interp0 recur) e)
  (match e
    ....))

(define ((interp1 recur) e)
  (match e
    ....))
```

```
(define-extend (interp0 e)
  (match e
    [`(+ ,x ,y)
     (+ (interp0 x) (interp0 y))]
    [(? number?) e]))

(define-extend (interp1 e)
  #:extend interp0
  (match e
    [`(* ,x ,y)
     (* (interp1 x) (interp1 y))]
    [_ (interp0 e)]))
```

Dot and Underscore Notation

```
(define h
  (new Horse [breed "Danish Pony"]))

(send h run "Bella")



(define fibs (vector 42 1 2 3 5 8))

(vector-ref fibs 0)

(vector-set! fibs
             0
             (+ 1
                (vector-ref fibs 0)))
```

Dot and Underscore Notation

```
(define h
  (new Horse [breed "Danish Pony"]))

(send h run "Bella")


(define fibs (vector 42 1 2 3 5 8))

(vector-ref fibs 0)

(vector-set! fibs
             0
             (+ 1
                (vector-ref fibs 0)))
```

```
(define h
  (new Horse [breed "Danish Pony"]))

(h.run "Bella")


(define fibs (vector 42 1 2 3 5 8))

fibs_0

(:= fibs_0 (+ 1 fibs_0))
```

JS Dict Notation

```
(define tbl (hash 'a 1 'b 2 'c 3))

(define a (hash-ref tbl 'a))

(define b (hash-ref tbl 'b))

(define tbl2
  (hash-remove
    (hash-remove tbl 'a)
    'b))
```

JS Dict Notation

```
(define tbl (hash 'a 1 'b 2 'c 3))

(define a (hash-ref tbl 'a))

(define b (hash-ref tbl 'b))

(define tbl2
  (hash-remove
    (hash-remove tbl 'a)
    'b))
```

```
(define tbl (hash 'a 1 'b 2 'c 3))

(js-extract {a b #:rest tbl2} tbl)

;; a = 1
;; b = 2
;; tbl2 = (hash 'c 3)
```

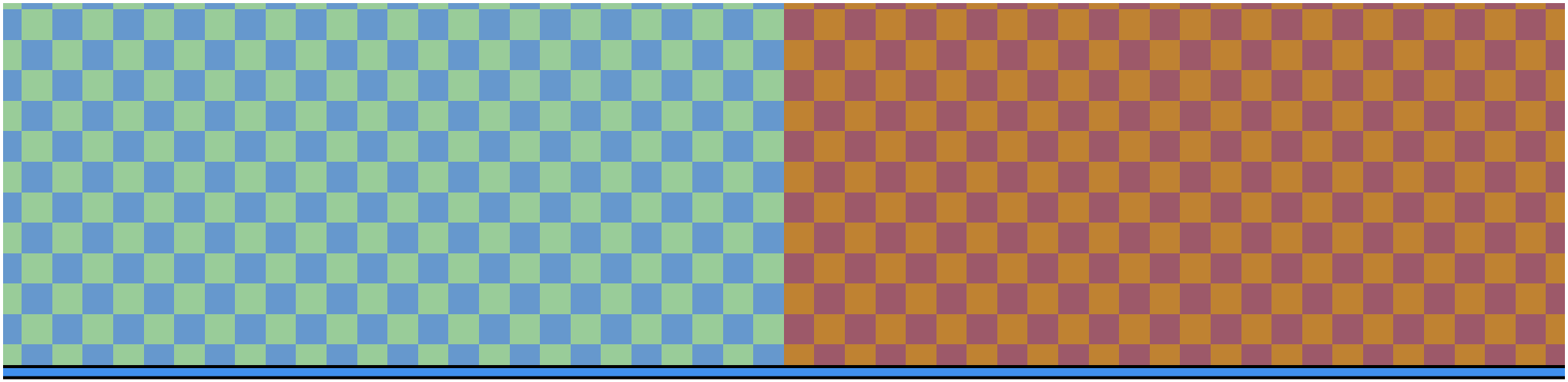
Pyret for

```
(andmap (λ (thing quantity)
          ....)
        things
        quantities)
```

Pyret for

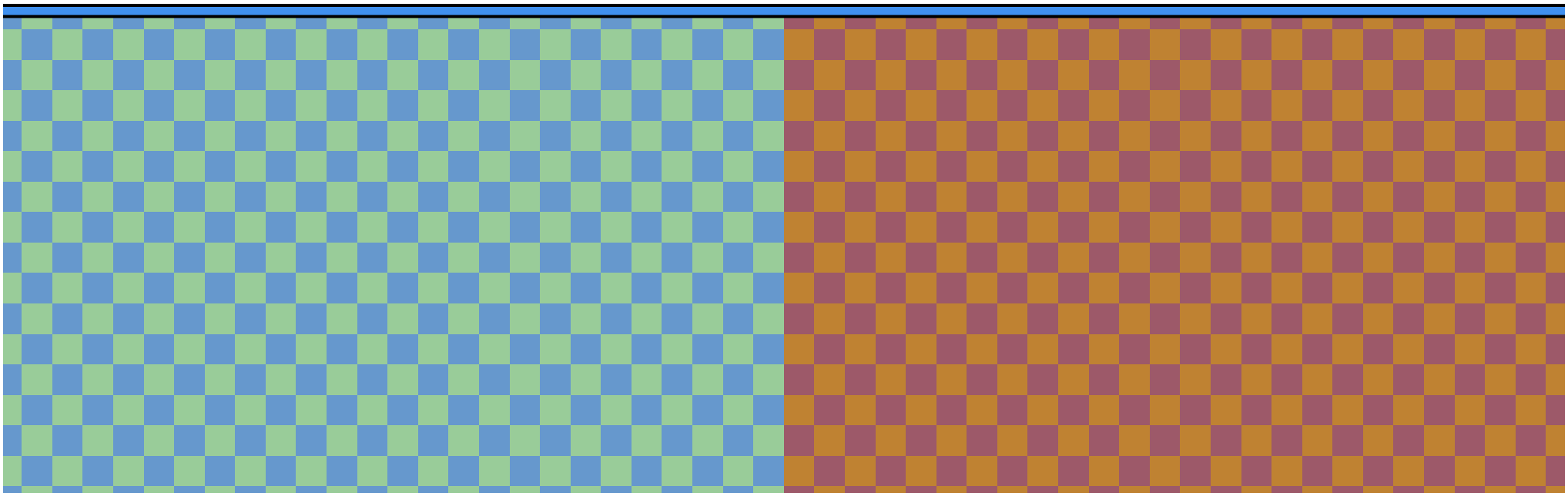
```
(andmap (λ (thing quantity)
          ....)
        things
        quantities)
```

```
(pyret-for andmap
           ([thing things]
            [quantity quantities])
  ....)
```



```
(~? A B)

(~@ A ...)
```



Sequencing definitions with ~?

```
(define x 4)
(define y 18)
(define-values (quot rem)
  (quotient/remainder x y))
```

```
(define*
  [x 4]
  [y 18]
  [(quot rem)
   (quotient/remainder x y)])
```

Sequencing definitions with ~?

```
(define x 4)
(define y 18)
(define-values (quot rem)
  (quotient/remainder x y))
```

```
(define*
  [x 4]
  [y 18]
  [(quot rem)
   (quotient/remainder x y)])
```

```
(define-syntax-parser define*
  [(_ (~or [id:id expr:expr]
           [(idv:id ...+) expr:expr])
      ...+)
   #'(begin
       (~? (define id expr)
           (define-values (idv ...) expr)) ...)])
```

Splicing keyword args with ~@

```
(string-trim " abc "
             #:left? #t
             #:right? #t)
```

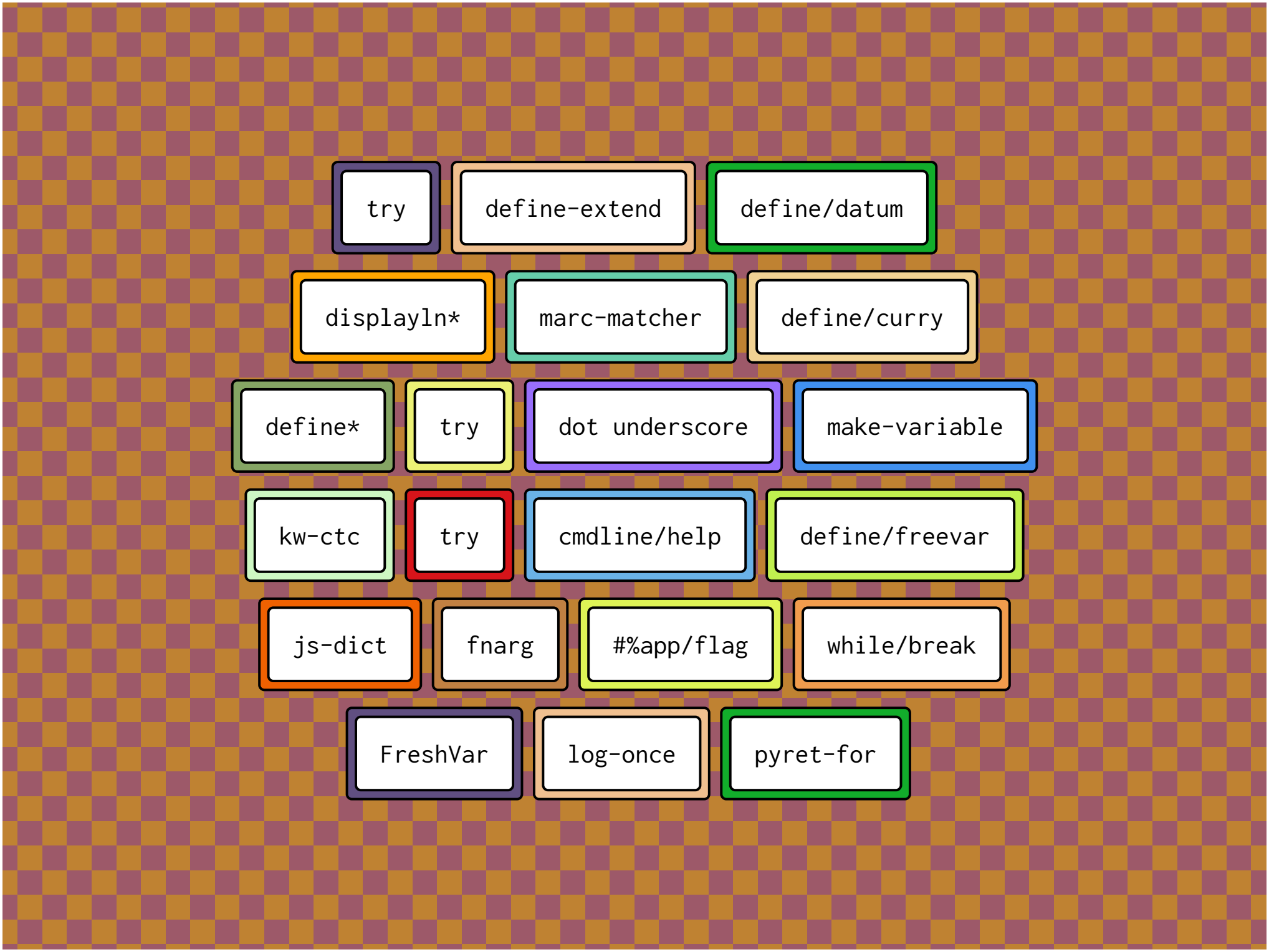
```
(string-trim " abc "
             #:left?
             #:right?)
```

Splicing keyword args with ~@

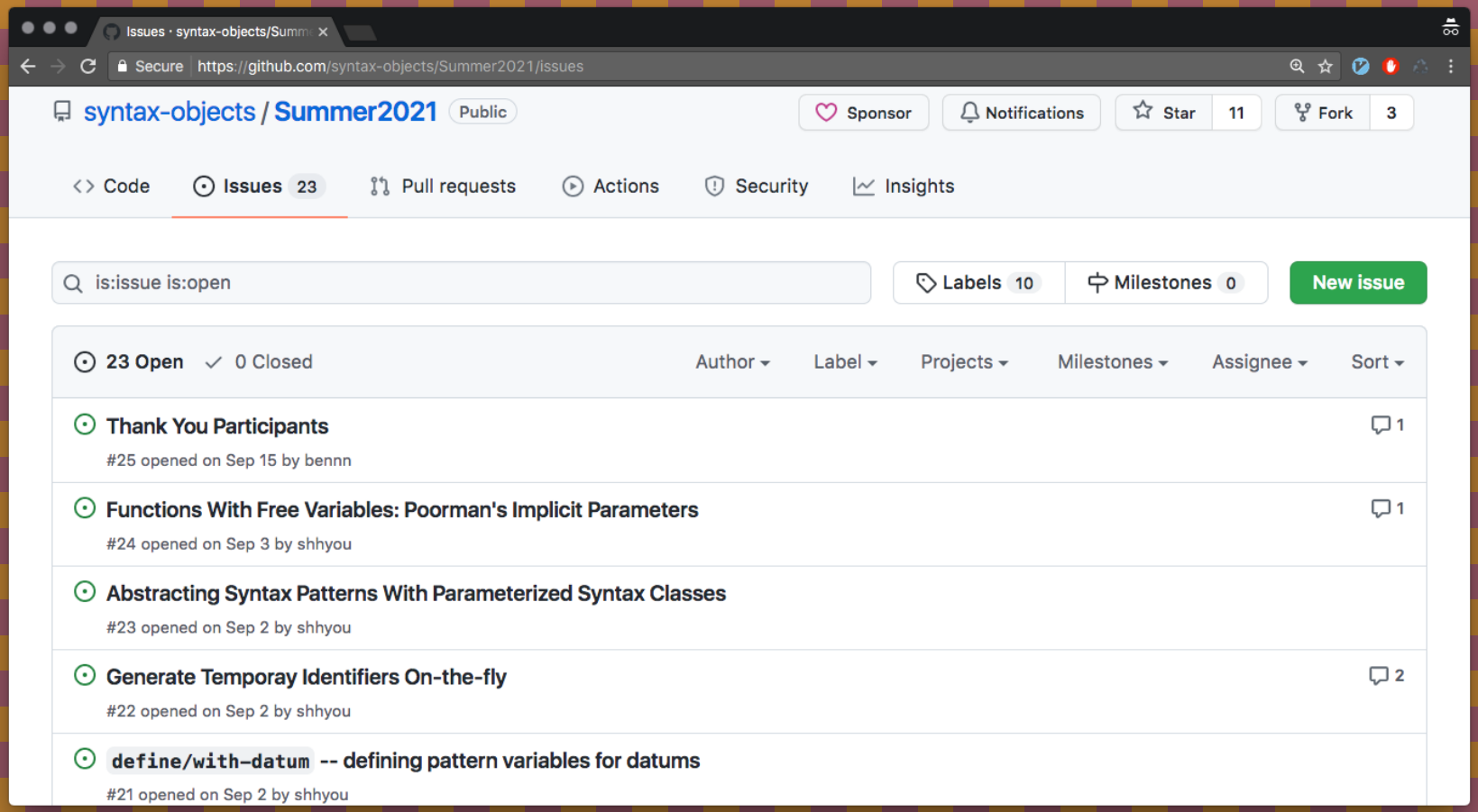
```
(string-trim " abc "
             #:left? #t
             #:right? #t)
```

```
(string-trim " abc "
             #:left?
             #:right?)
```

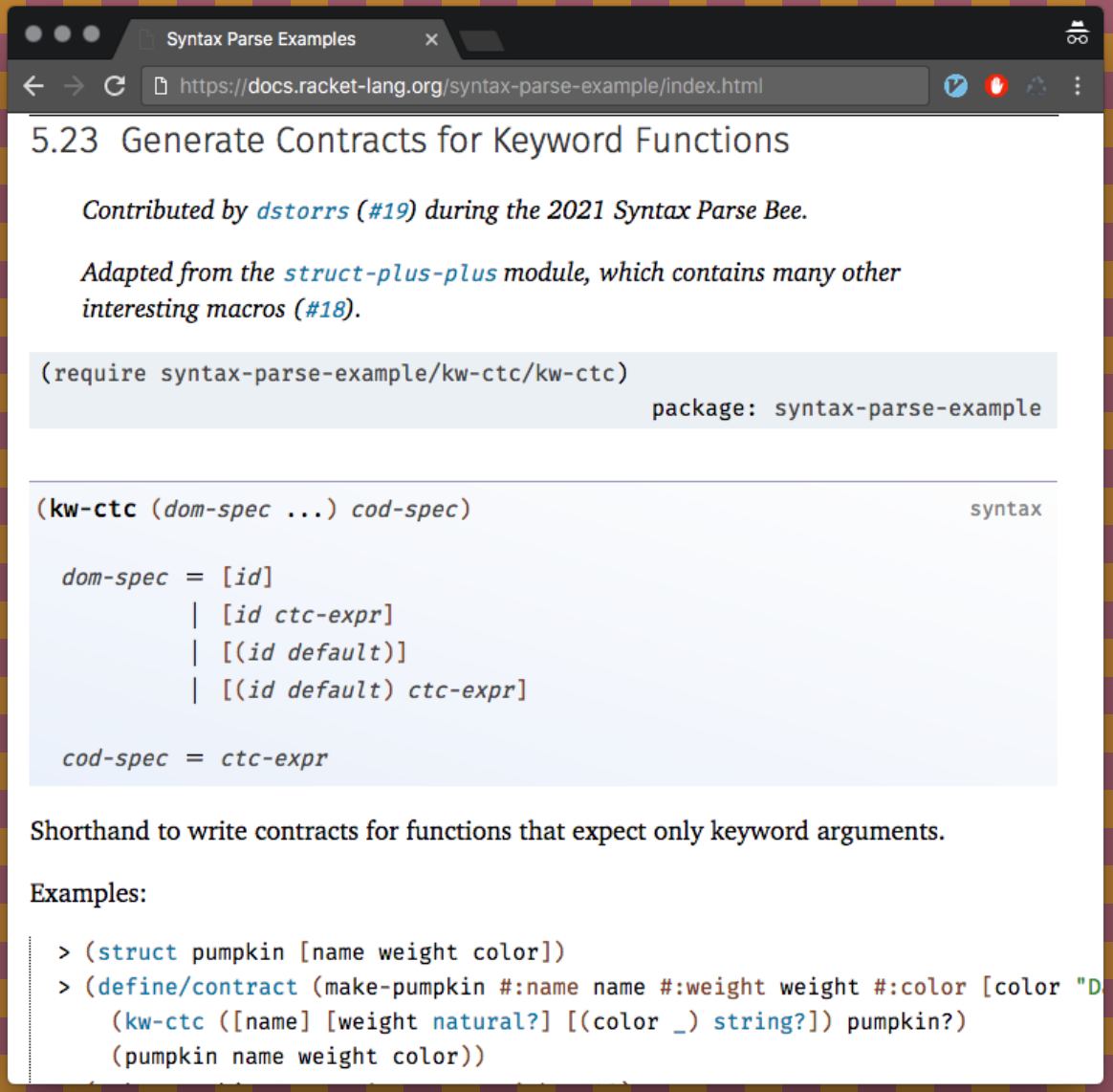
```
(define-syntax-parser #%app/flag
  [(_ f no-kw:expr ...
      kw:arg/kw ...)
   #'(#%app f no-kw ... {~@ kw.k kw.v} ...)])
```


try
define-extend
define/datum
displayln*
marc-matcher
define/curry
define*
try
dot underscore
make-variable
kw-ctc
try
cmdline/help
define/freevar
js-dict
fnarg
#%app/flag
while/break
FreshVar
log-once
pyret-for

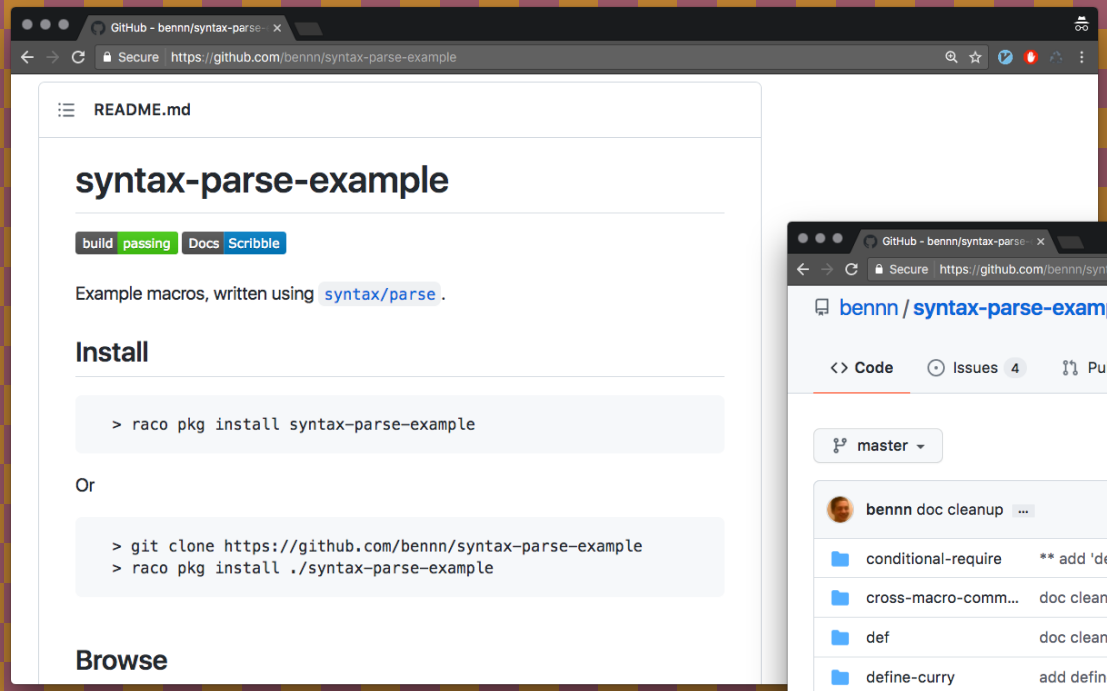
Submissions github.com/syntax-objects/Summer2021

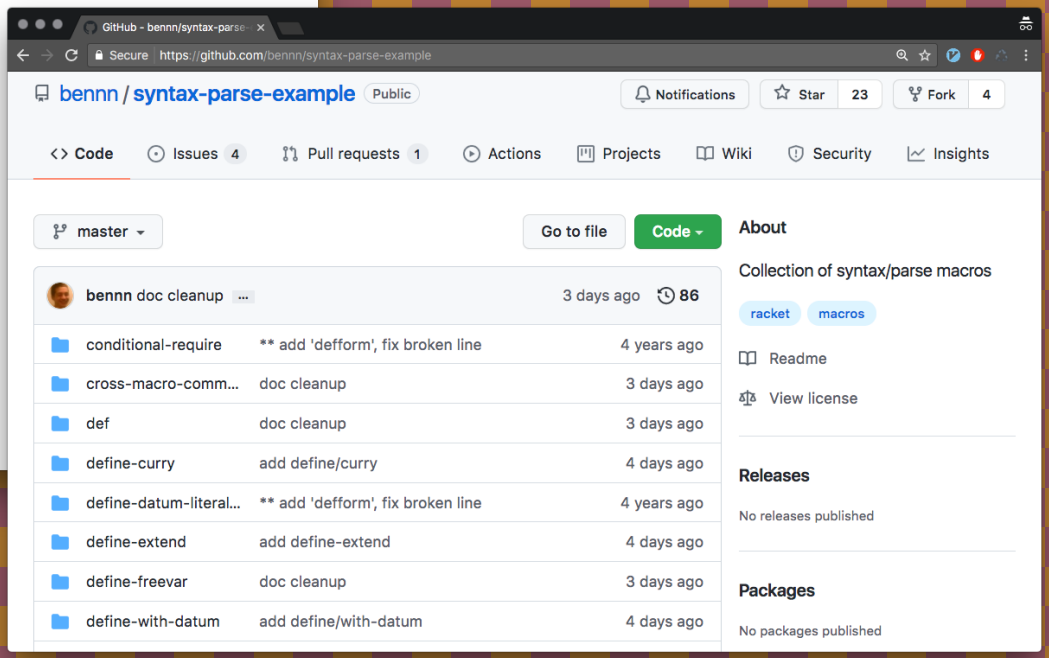


Community Resource docs.racket-lang.org/syntax-parse-example



Contribute! github.com/syntax-objects/syntax-parse-example





Legend of the Macro Pillow

ccs.neu.edu/home/pnkfelix/prl-tmp/gallery/index.html

Community Resource docs.racket-lang.org/syntax-parse-example

Submissions github.com/syntax-objects/Summer2021

Contribute! github.com/syntax-objects/syntax-parse-example

Thank You!

Stephen


Lazerbeak12345

hzafar

agj

Fictitious-Rotor

xgqt

benknoble

AlexKnauth


sorawee

eutro

camoy

soegaard

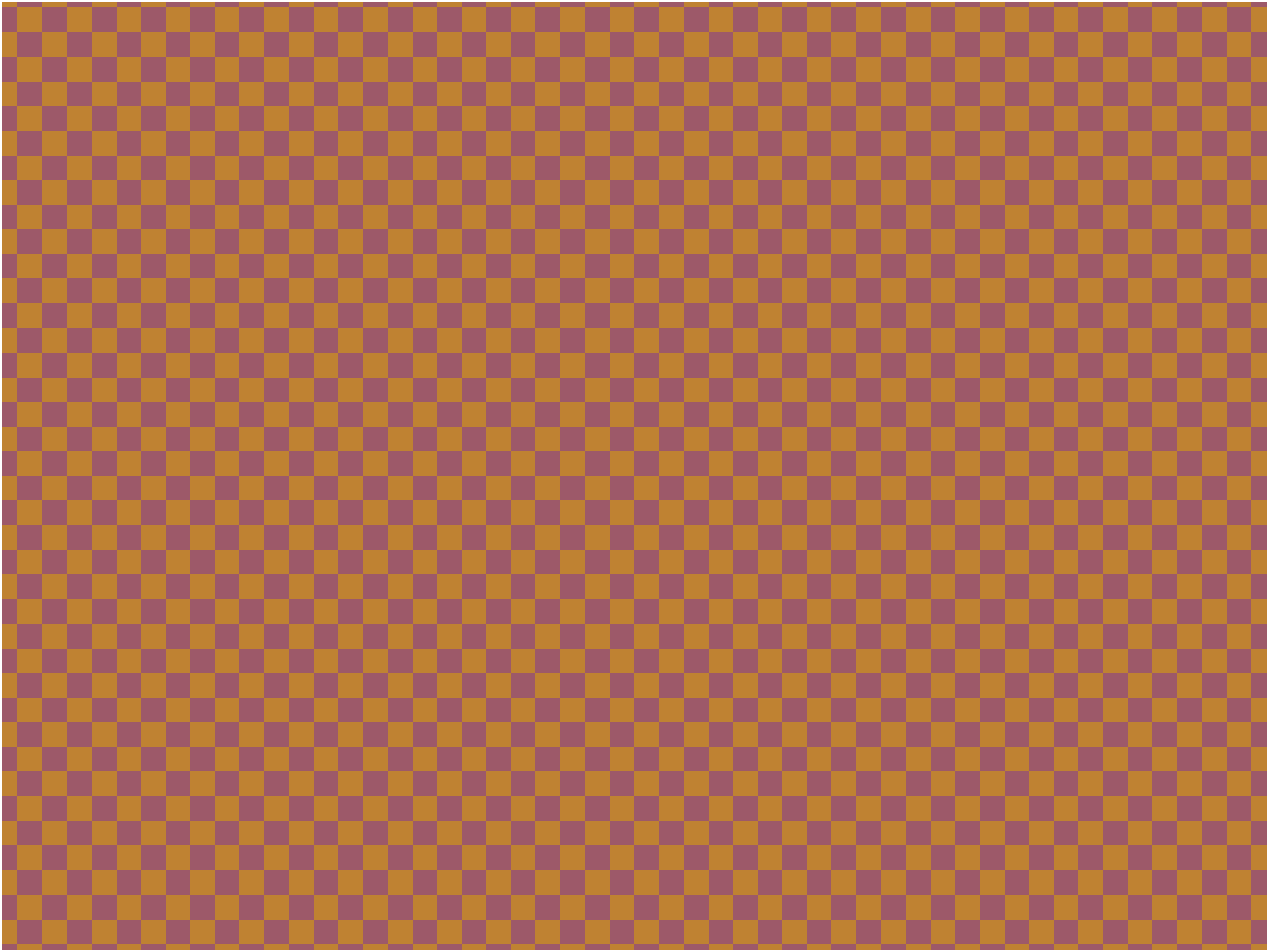
Metaxal

dstorrs

countvajhula

shhyou

Contributors



Community Resource docs.racket-lang.org/syntax-parse-example

Submissions github.com/syntax-objects/Summer2021

Contribute! github.com/syntax-objects/syntax-parse-example