Implementing General Contract Boundaries

T. Stephen Strickland Matthias Felleisen

sstrickl@ccs.neu.edu matthias@ccs.neu.edu

Northeastern University

Boston, MA, USA

Contracts and Contract Boundaries

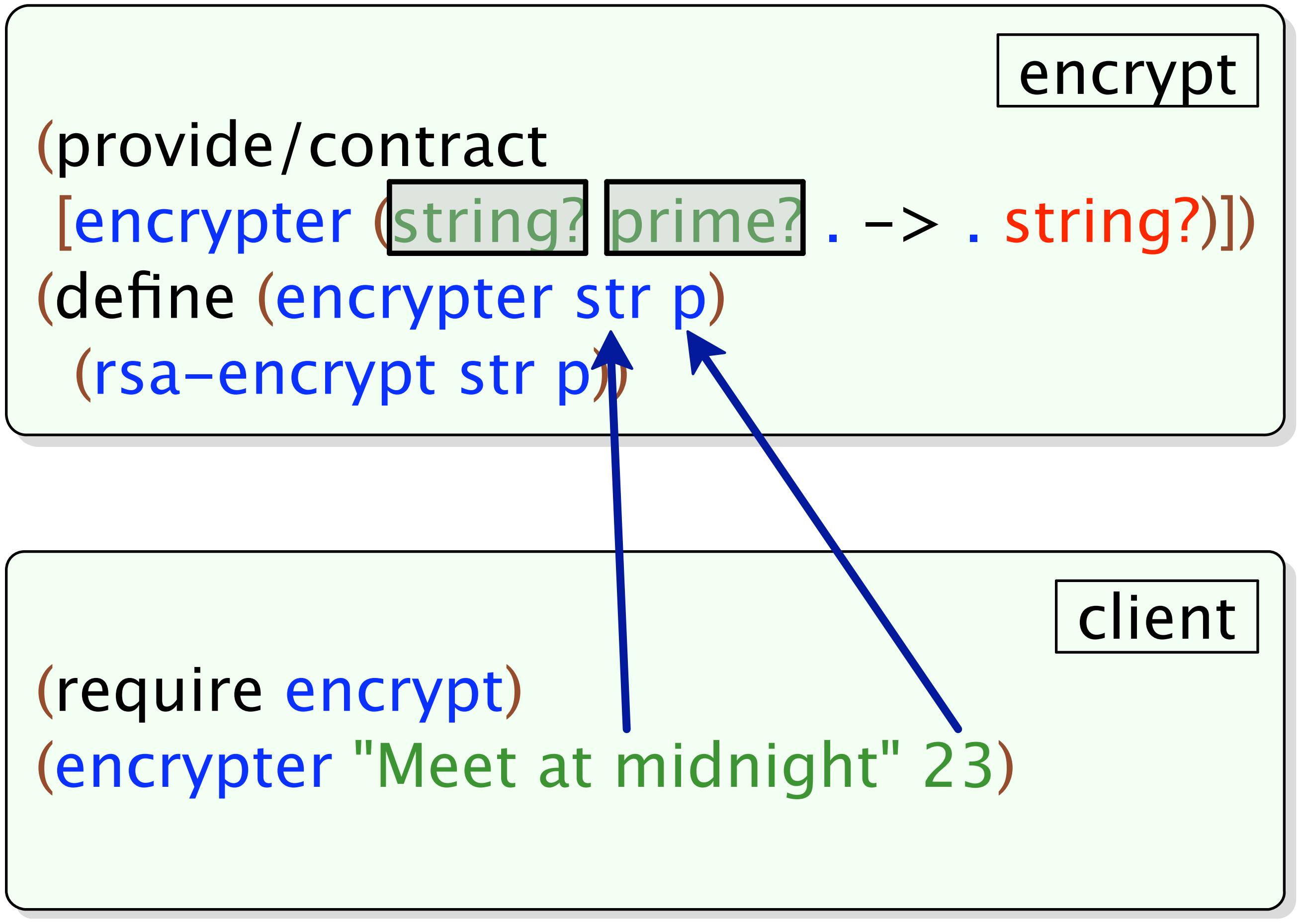
A contract is a specification and an agreement.

encrypt

```
(provide/contract
 [encrypter (string? prime? . -> . string?)])
(define (encrypter str p)
  (rsa-encrypt str p))
```

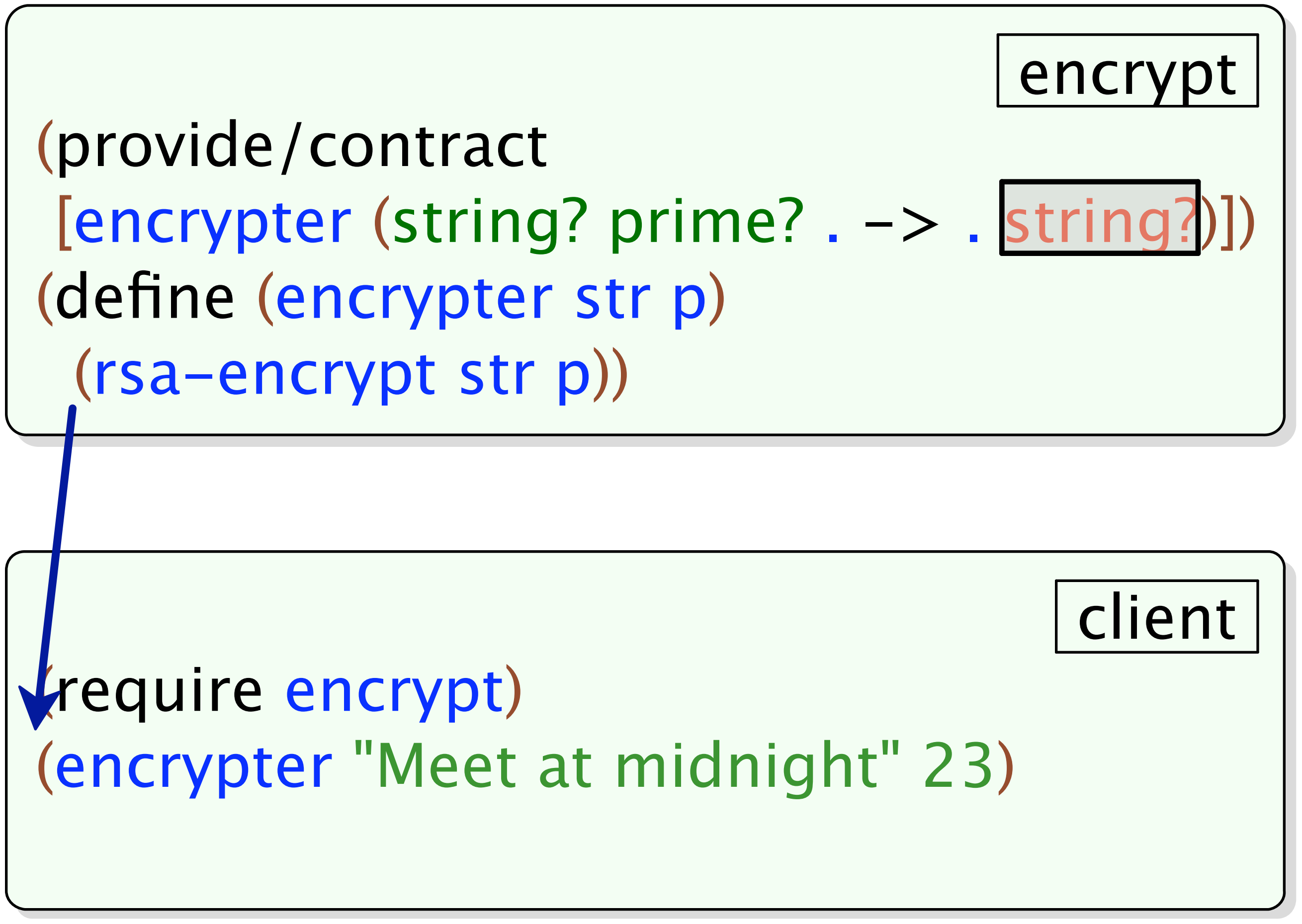
client

```
(require encrypt)
(encrypter "Meet at midnight" 23)
```



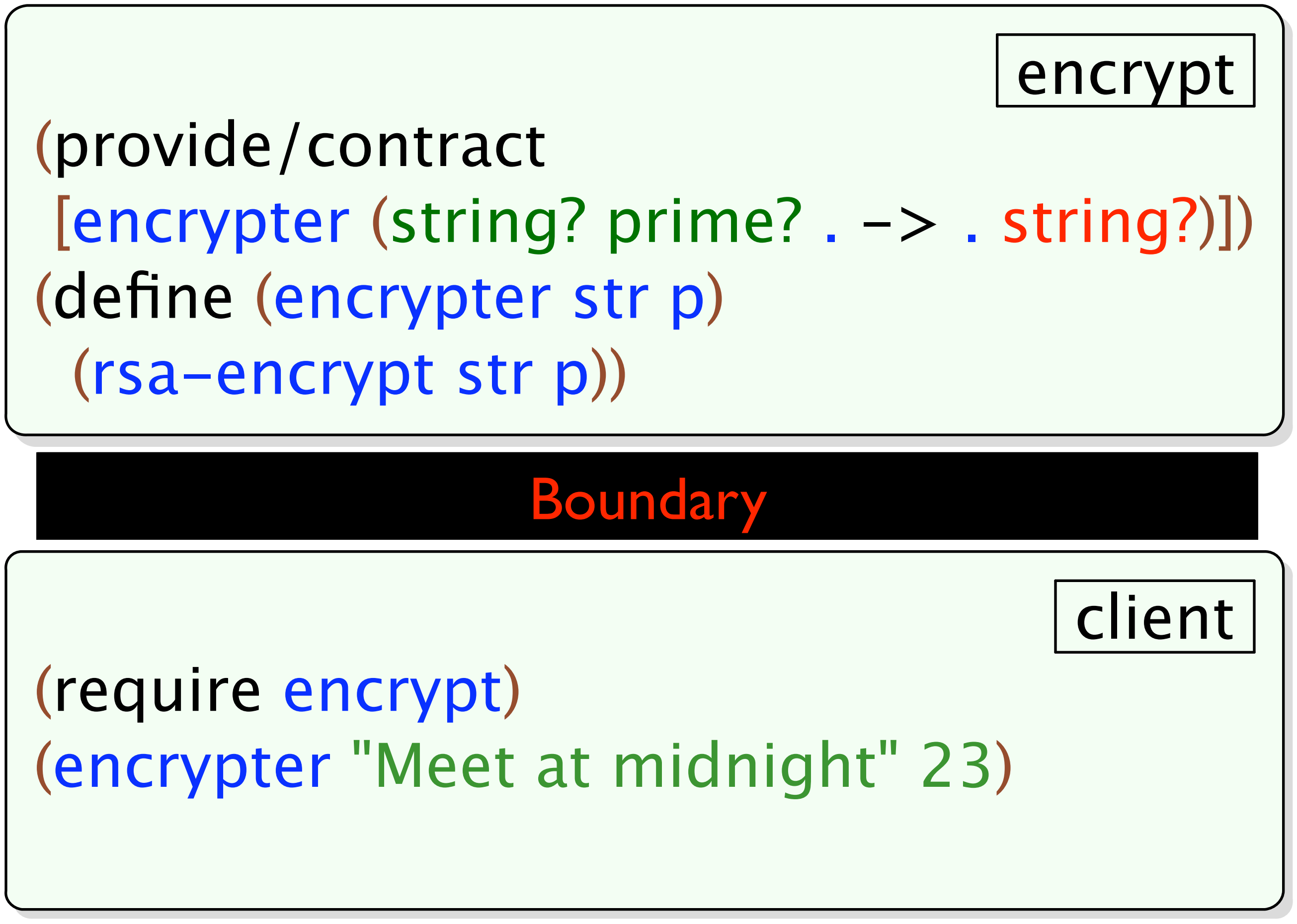
```
(provide/contract
 [encrypter (string? prime? . -> . string?)])
(define (encrypter str p)
  (rsa-encrypt str p))
```

```
(require encrypt)
(encrypter "Meet at midnight" 23)
```



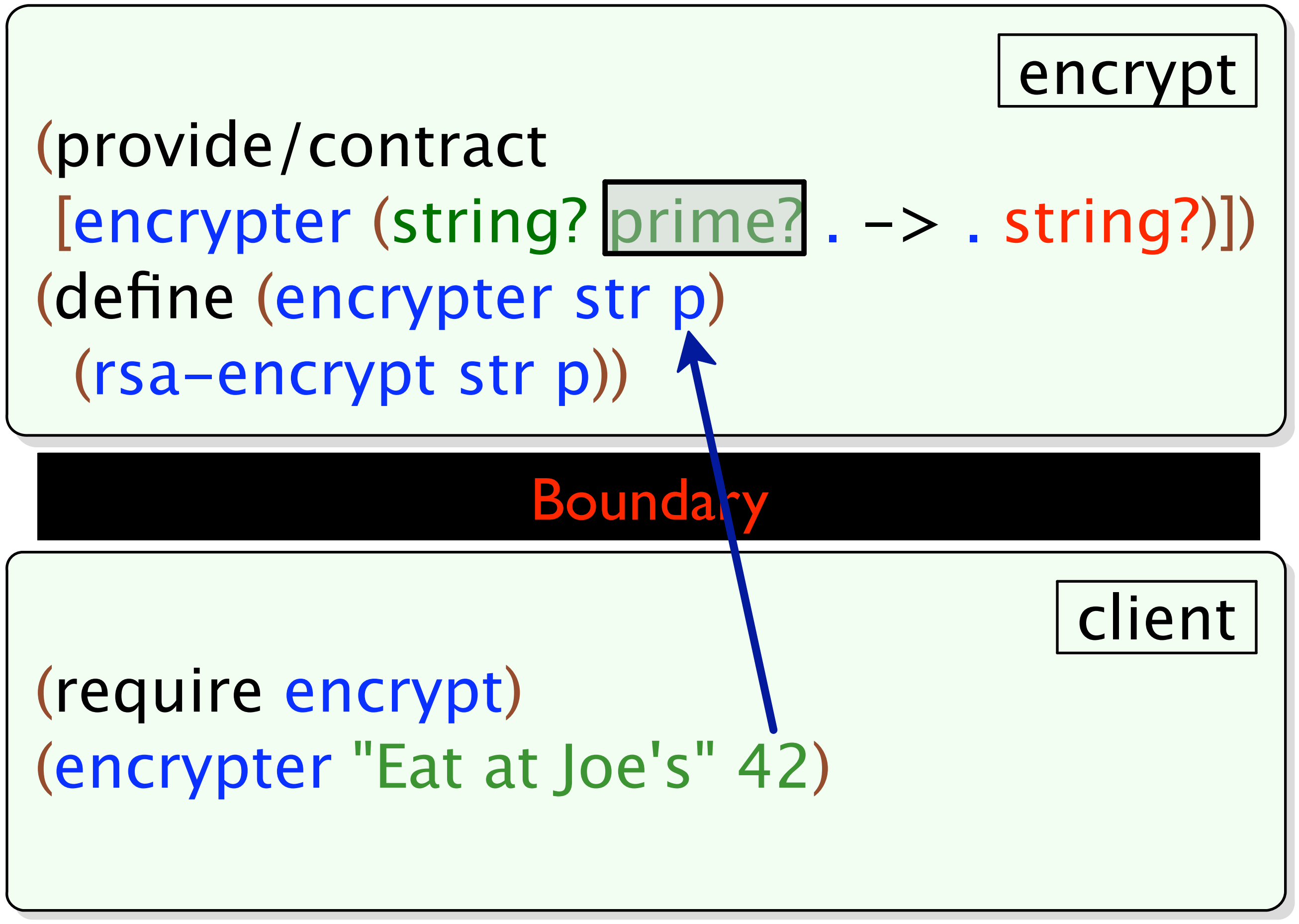
```
(provide/contract
 [encrypter (string? prime? . -> . string?)])
(define (encrypter str p)
  (rsa-encrypt str p))
```

```
(require encrypt)
(encrypter "Meet at midnight" 23)
```



```
(provide/contract
 [encrypter (string? prime? . -> . string?)])
(define (encrypter str p)
  (rsa-encrypt str p))
```

```
(require encrypt)
(encrypter "Meet at midnight" 23)
```



```
(provide/contract
 [encrypter (string? prime? . -> . string?)])
(define (encrypter str p)
  (rsa-encrypt str p))
```

```
(require encrypt)
(encrypter "Eat at Joe's" 42)
```

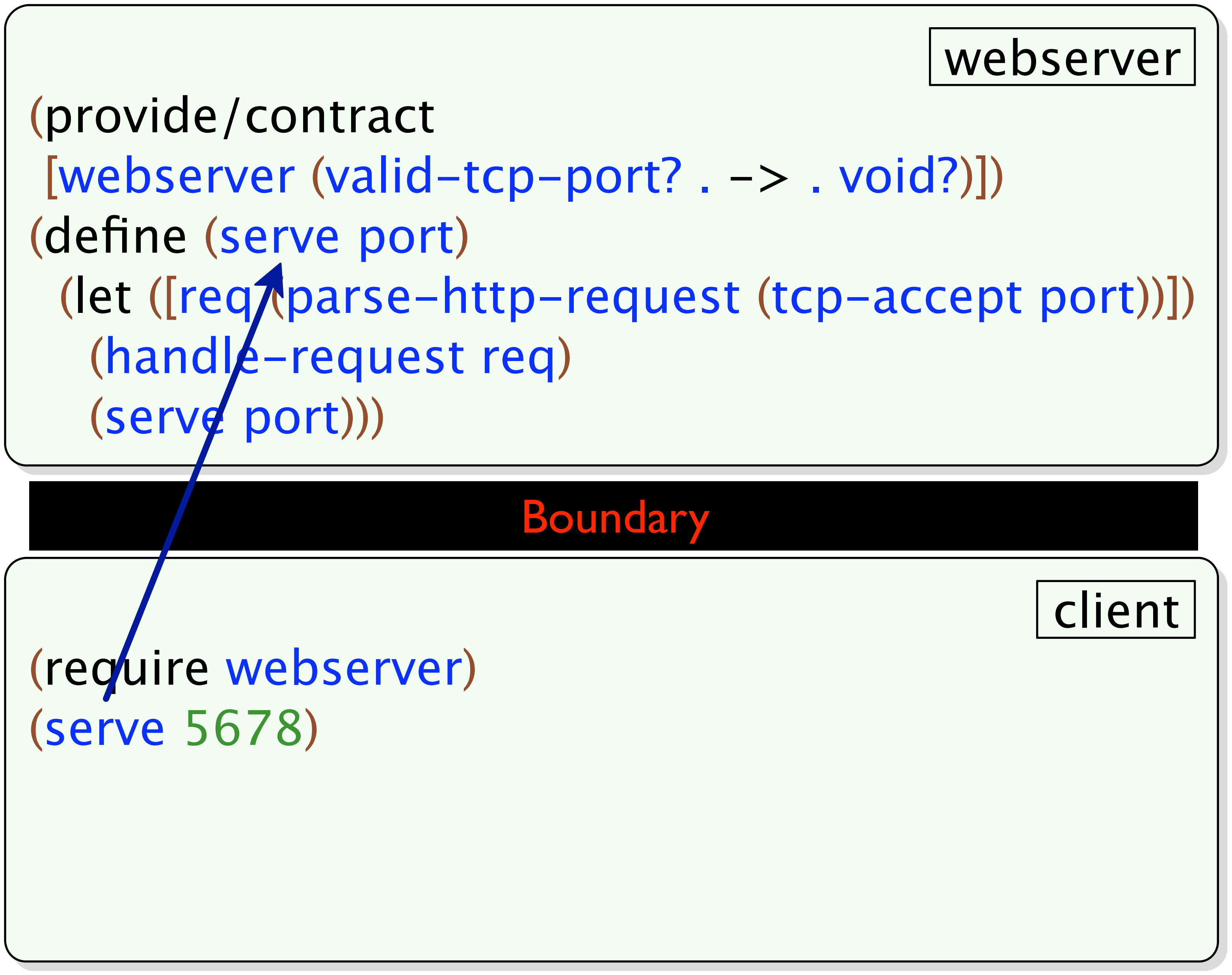
client broke the contract (string? prime? . -> . string?) on encrypter; expected <prime?>, given: 42 webserver

```
(provide/contract
 [webserver (valid-tcp-port? . -> . void?)])
(define (serve port)
  (let ([req (parse-http-request (tcp-accept port))])
    (handle-request req)
    (serve port)))
```

Boundary

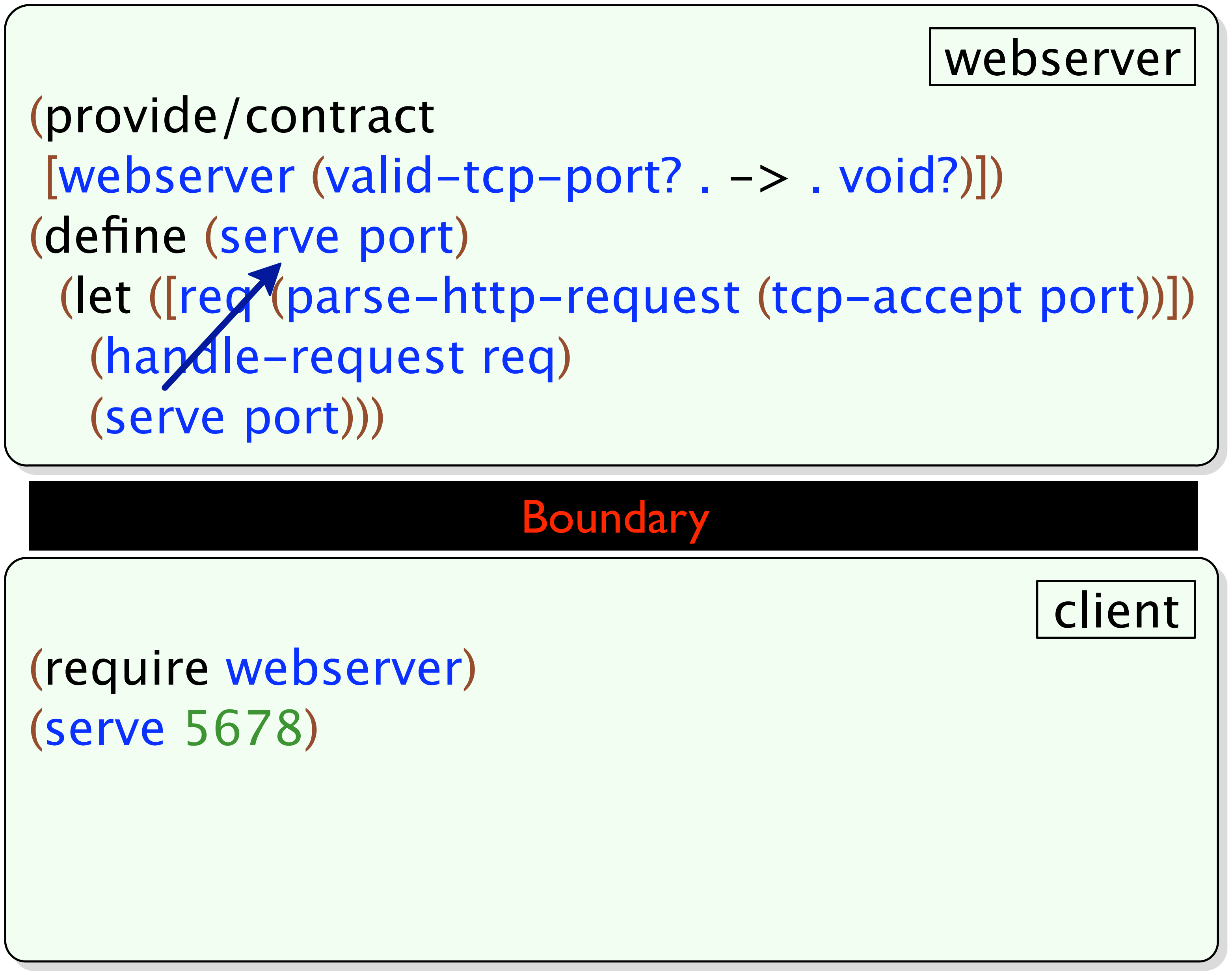
client

```
(require webserver)
(serve 5678)
```



```
(provide/contract
 [webserver (valid-tcp-port? . -> . void?)])
(define (serve port)
  (let ([req (parse-http-request (tcp-accept port))])
    (handle-request req)
    (serve port)))
```

```
(require webserver)
(serve 5678)
```



webserver

```
(provide/contract
 [webserver (valid-tcp-port? . -> . void?)])
(define (serve port)
  (let ([req (parse-http-request (tcp-accept port))])
    (handle-request req)
    (serve port)))
```

Boundary

client

```
(require webserver)
(serve 5678)
```

Static vs. Dynamic

The two parties agreeing to static contract boundaries can be determined at compile-time.

The two parties agreeing to dynamic contract boundaries are only determined at run-time.

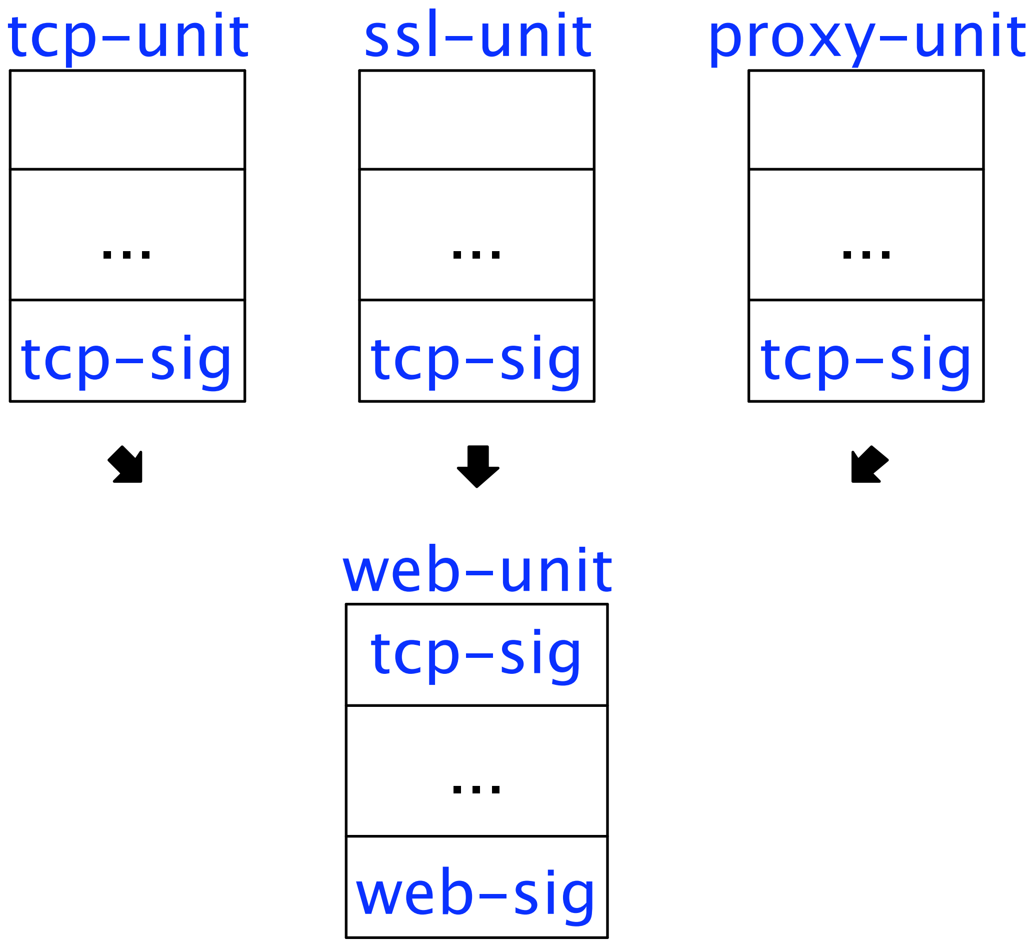
PLT Scheme units are first-class, dynamically linked modules.

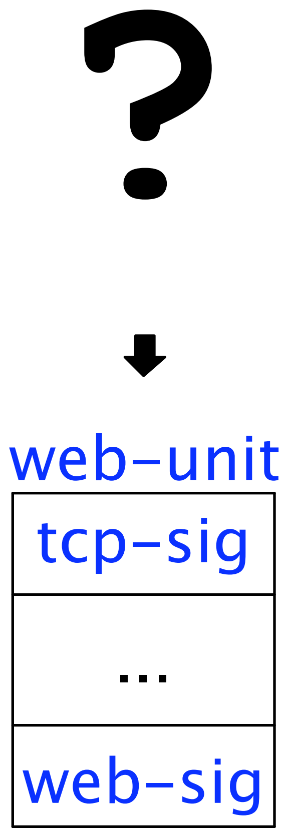
PLT Scheme units are first-class, dynamically linked modules.

```
(define–signature tcp–sig (accept listen close ...))

(define–signature web–sig (serve))

(define web–unit
  (unit (import tcp–sig) (export web–sig)
        (define (serve port) ...)))
```

PLT Scheme units are first-class, dynamically linked modules.

```
(define tcp–unit
  (unit (import) (export tcp–sig) ...))

(define web–unit
            (unit (import tcp–sig) (export web–sig)
                     (define (serve port) ...)))

(compound–unit (import) (export web–sig)
                              (link tcp–unit web–unit))
```


tcp–unit
ssl–unit
proxy–unit
...
tcp–sig
web–unit
tcp–sig
...
web–sig


?
web–unit
tcp–sig
...
web–sig

Implementing Contracts for Units

Signatures contain contracts.

```
(define-signature web-sig
  ((contracted [serve (valid-tcp-port? . -> . void?)])))
```

Units handle uncontracted and contracted names differently.

```
(define-signature http-request-sig
 ((contracted [parse-http-request
                    (input-port? . -> . valid-http-req?)])
  handle-req))

                 handle-req

              parse-http-request
```

Units handle uncontracted and contracted names differently.

```
(define-signature http-request-sig
  ((contracted [parse-http-request
                 (input-port? . -> . valid-http-req?)])
   handle-req))
```

handle-req $\boxed{\text{value}}$

parse-http-request

Units handle uncontracted and contracted names differently.

```
(define-signature http-request-sig
 ((contracted [parse-http-request
                    (input-port? . -> . valid-http-req?)])
  handle-req))
```

handle-req	value	
parse-http-request	value	blame

Contract Regions

auth

```
(define (user-info user) ...)
(define (authenticate user passwd)
  ...
  (string=? passwd (hash-ref (user-info user) 'passwd))
  ...)
```

auth

```
(require user–info)
(define (authenticate user passwd)
  ...
  (string=? passwd (hash–ref (user–info user) 'passwd))
  ...)
```

user–info

```
(define (user–info user) ...)
(provide/contract
 [user–info (–> string? (hash/c symbol? string?))])
```

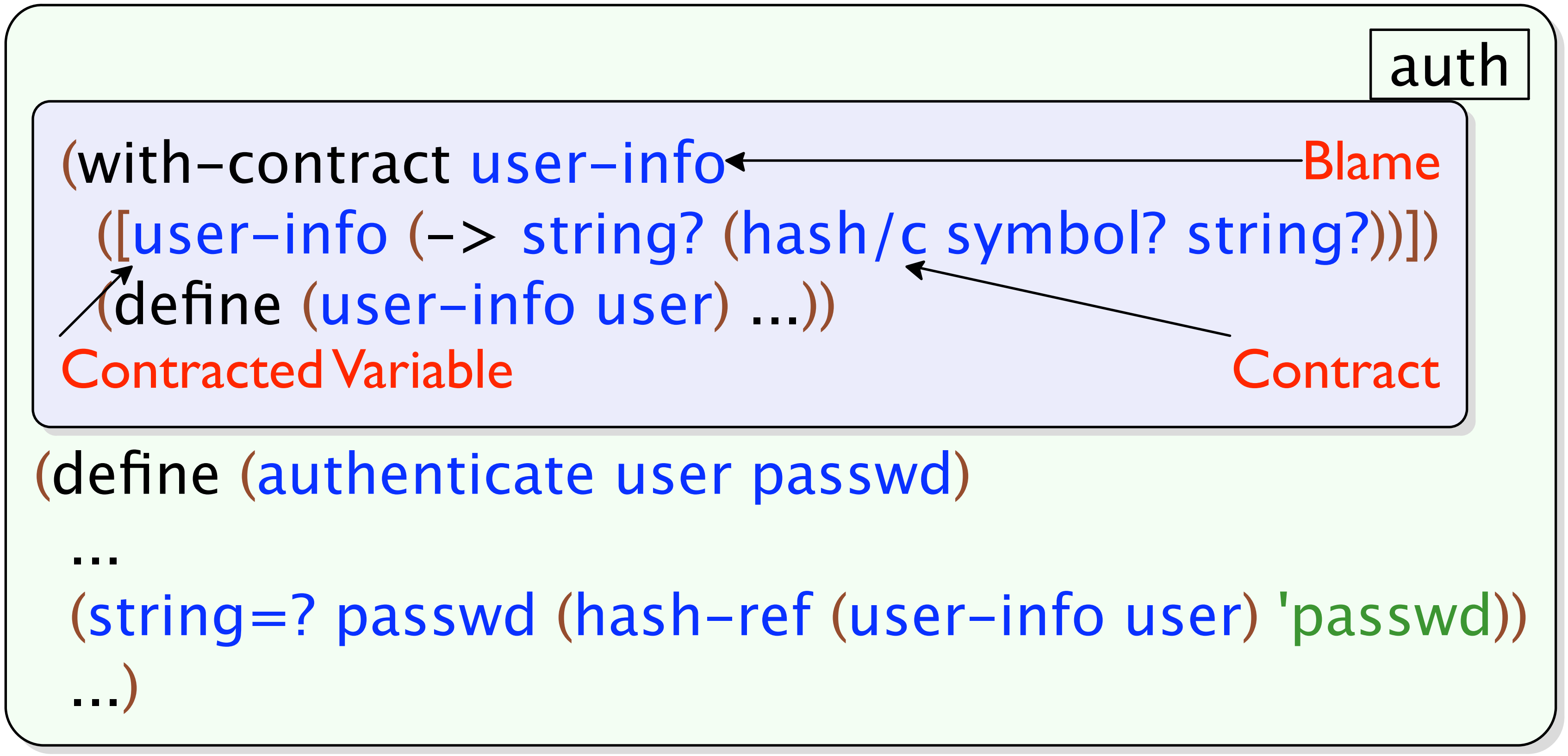
auth

```
(define (user-info user) ...)
```

```
(define (authenticate user passwd)
  ...
  (string=? passwd (hash-ref (user-info user) 'passwd))
  ...)
```

auth

```
(with-contract user-info
  ([user-info (-> string? (hash/c symbol? string?))])
  (define (user-info user) ...))

(define (authenticate user passwd)
  ...
  (string=? passwd (hash-ref (user-info user) 'passwd))
  ...)
```



```
(with-contract user-info
  ([user-info (-> string? (hash/c symbol? string?))])
  (define (user-info user) ...))
(define (authenticate user passwd)
  ...
  (string=? passwd (hash-ref (user-info user) 'passwd))
  ...)
```

server

```
(define (handle-request req) ...)
(define (add-choice s)
  (handle-request
   (make-special-request ...)))
```

```
... (add-choice "Newark") ...
(define (serve port)
  (let ([req (parse-http-request (tcp-accept port))])
    (handle-request req))
  (serve port))
```

server

```
(with-contract handle-request
  ([handle-request (-> valid-http-req? void?)])
  (define (handle-request req) ...)
  (define (add-choice s)
    (handle-request
     (make-special-request ...))))
... (add-choice "Newark") ...
(define (serve port)
  (let ([req (parse-http-request (tcp-accept port))])
    (handle-request req))
  (serve port))
```

server

```
(with-contract handle-request
  ([handle-request (-> valid-http-req? void?)])
  (define (handle-request req) ...)
  (define (add-choice s)
    (handle-request
     (make-special-request ...))))
... (add-choice "Newark") ...
(define (serve port)
  (let ([req (parse-http-request (tcp-accept port))])
    (handle-request req))
  (serve port))
```

Conclusion

General contract boundaries:

- static vs. dynamic
- unit contracts
- contract regions

`http://www.plt-scheme.org`