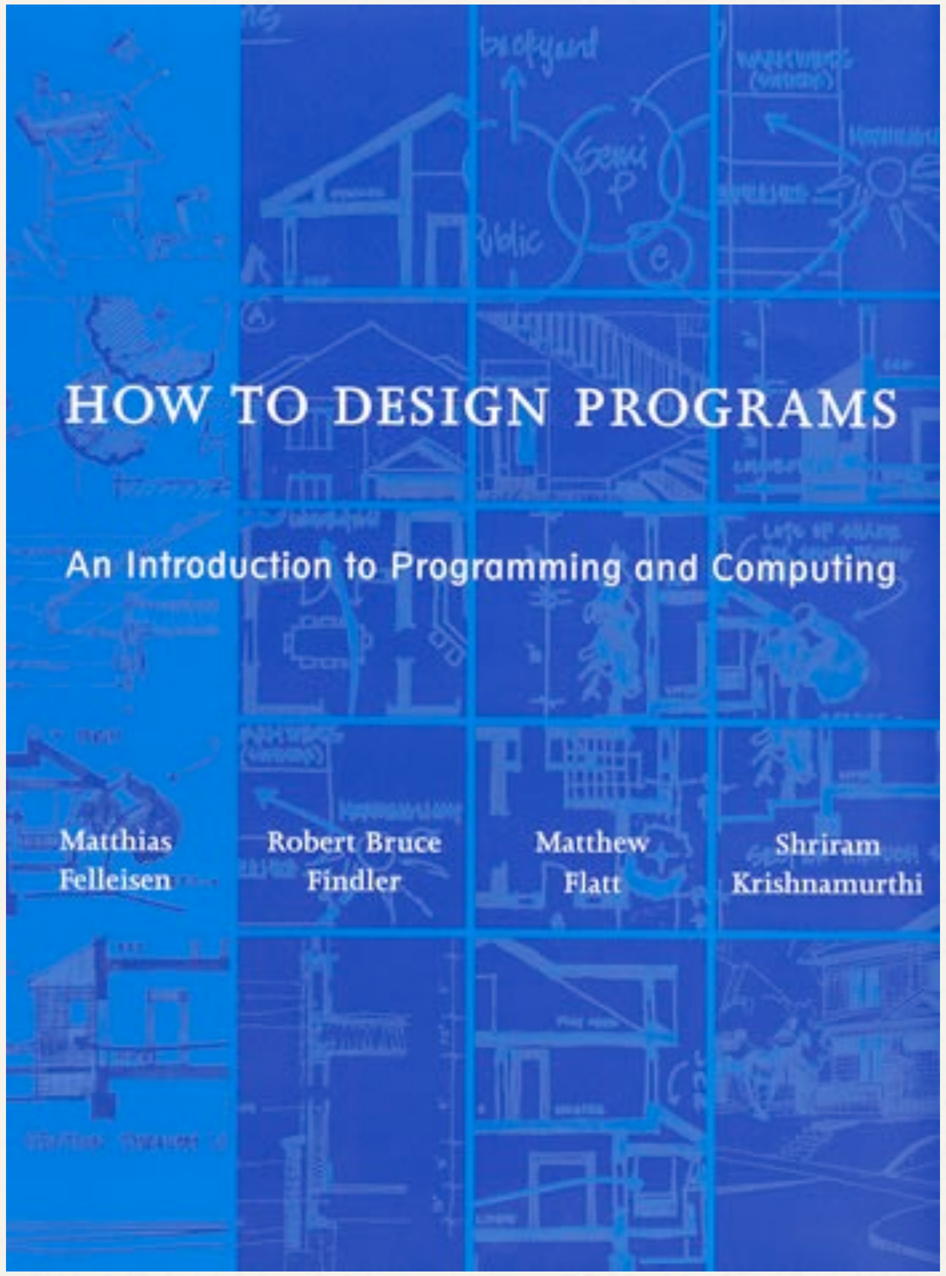
What's Wrong with *How to Design Programs*; What's New in *How to Design Programs 2e*

Matthias Felleisen

Content

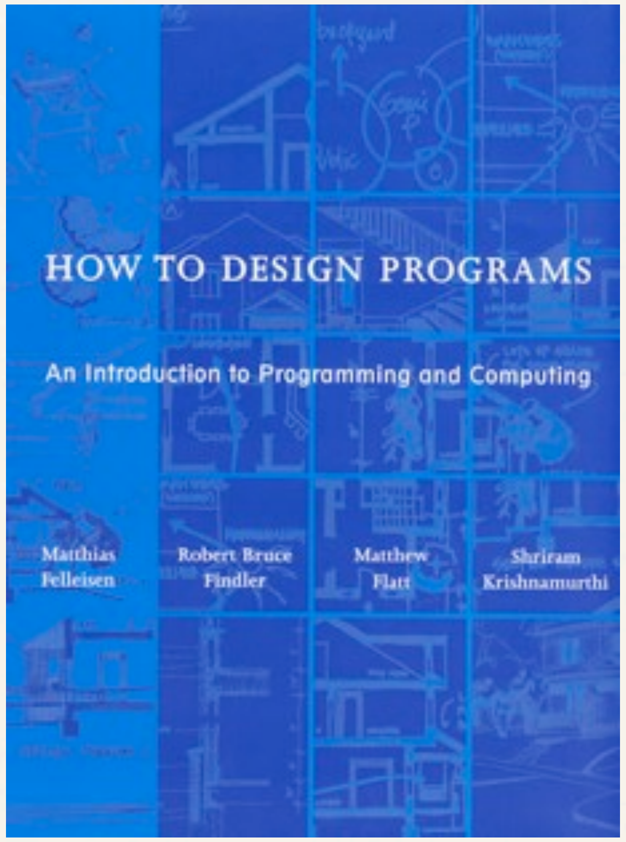

HOW TO DESIGN PROGRAMS

An Introduction to Programming and Computing

Matthias Felleisen
Robert Bruce Findler
Matthew Flatt
Shriram Krishnamurthi

Context

Content



HOW TO DESIGN PROGRAMS

An Introduction to Programming and Computing

Matthias Felleisen

Robert Bruce Findler

Matthew Flatt

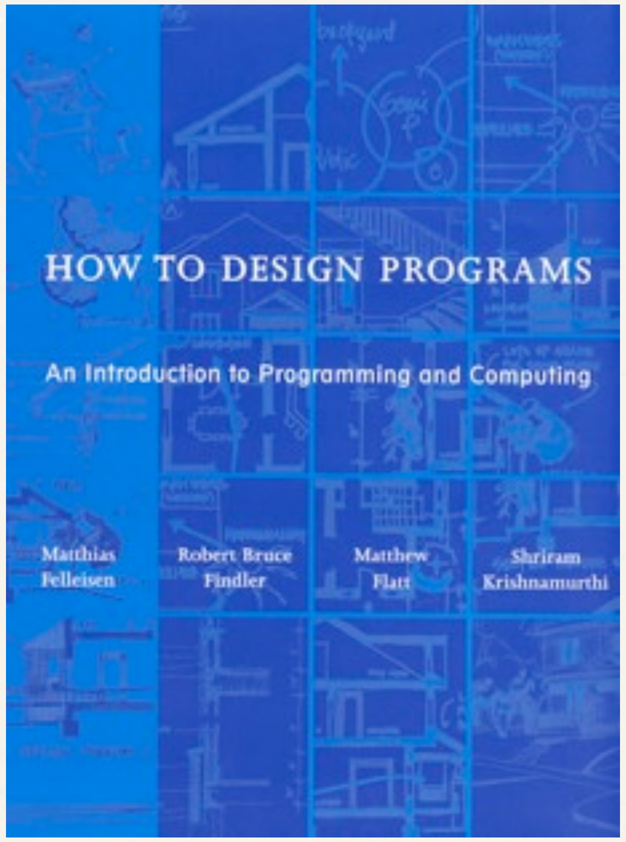
Shriram Krishnamurthi

Outside Context

Academic Context

Outside
Context

Content



HOW TO DESIGN PROGRAMS

An Introduction to Programming and Computing

Matthias Felleisen

Robert Bruce Findler

Matthew Flatt

Shriram Krishnamurthi

Academic
Context

Outside Context

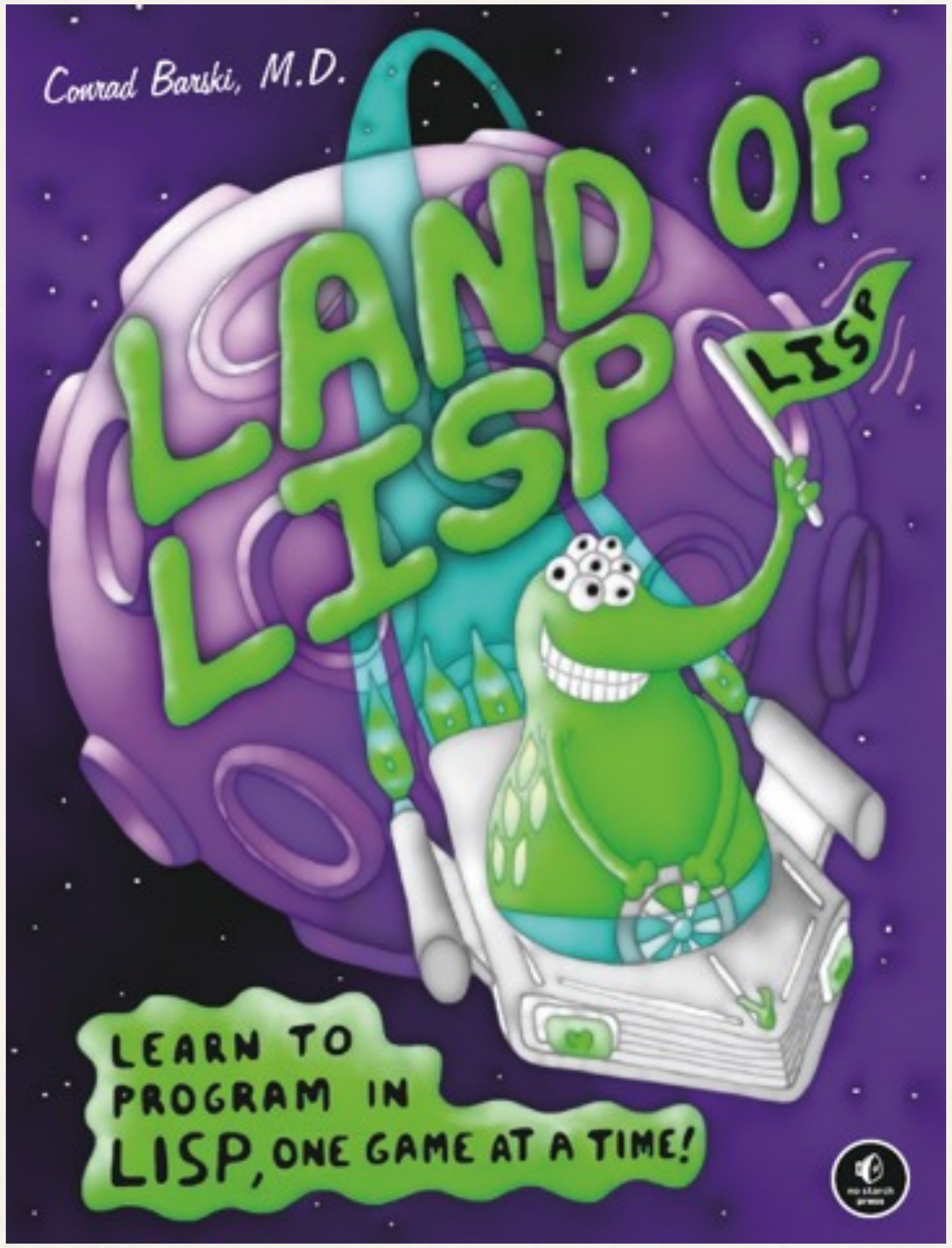
What is a student to do
when s/he reaches the end of HtDP?

Outside Context

What is a student to do
when s/he reaches the end of HtDP?

What is a student to do
who doesn’t see HtDP in his/her first year?

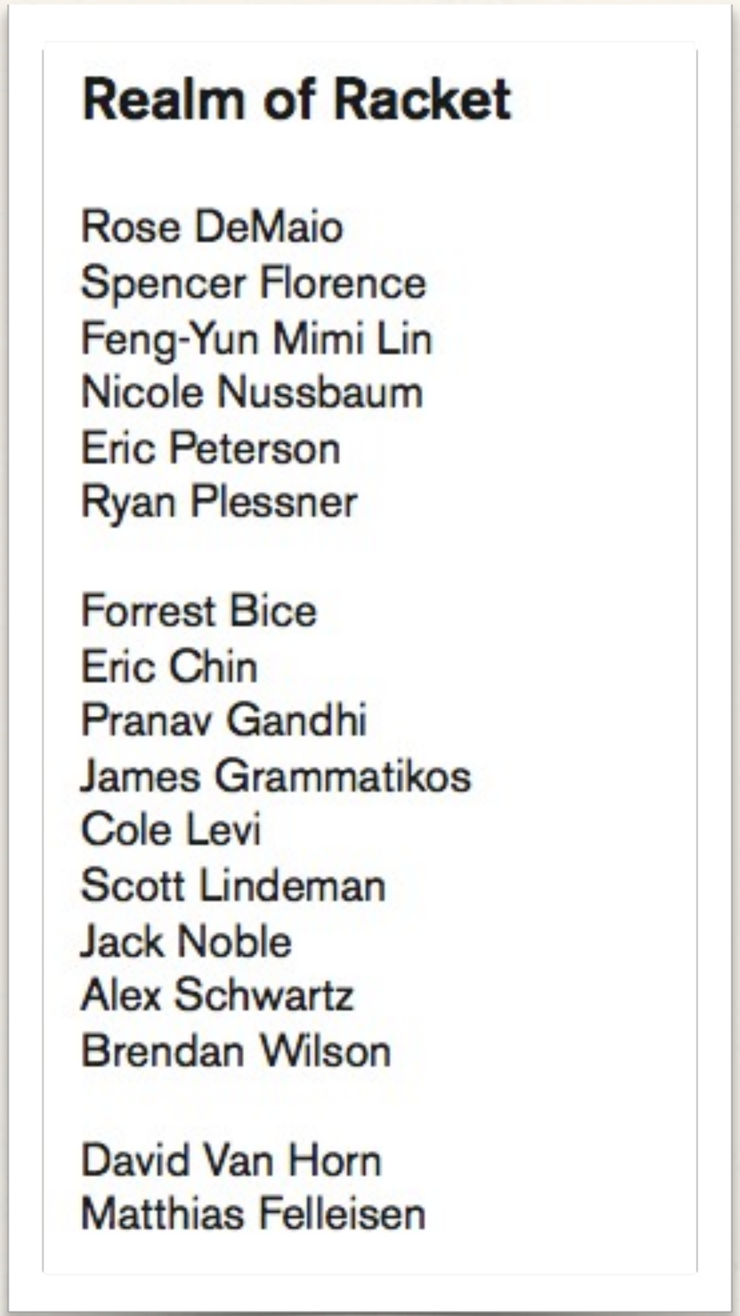
Outside Context



Lisp
Fun
Games
Learning
Programming
Stories with Comics
Web Sites, Songs, and Videos

video

ROAR: Realm of Racket



Realm of Racket

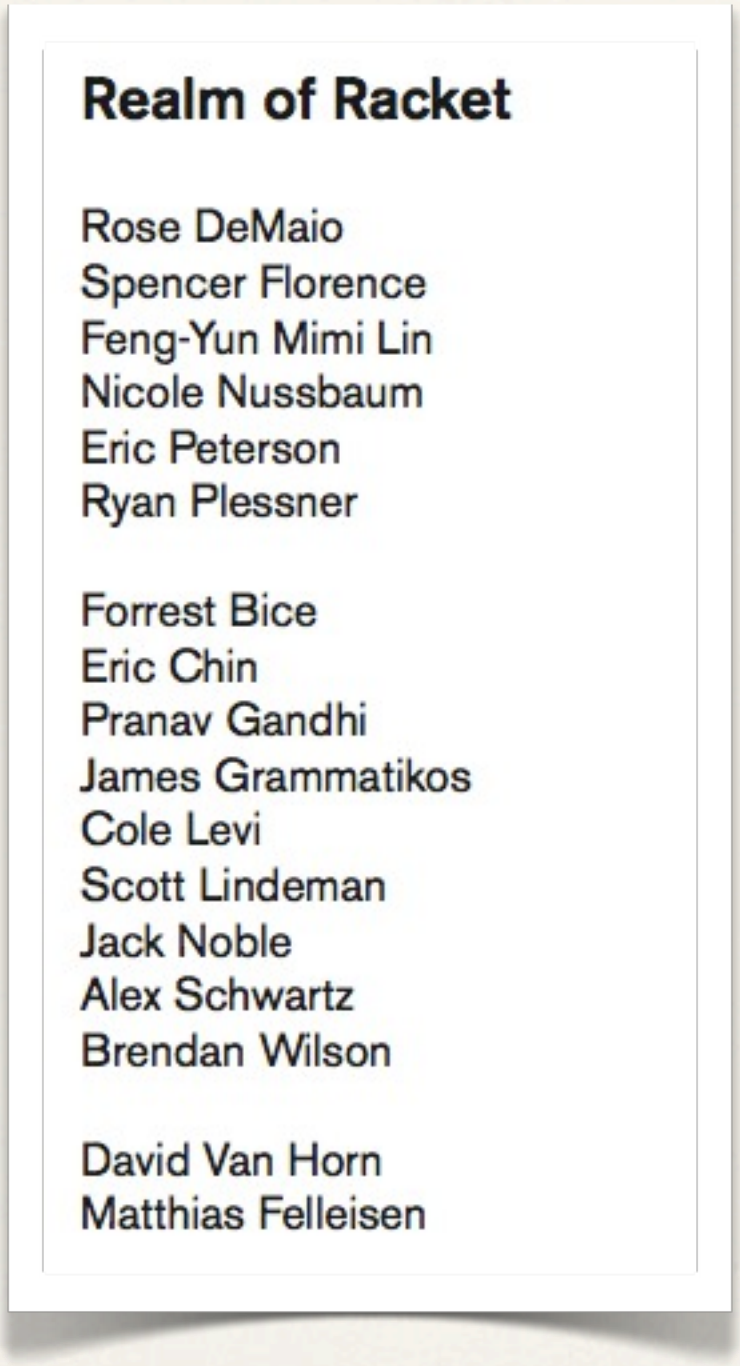
Rose DeMaio
Spencer Florence
Feng-Yun Mimi Lin
Nicole Nussbaum
Eric Peterson
Ryan Plessner

Forrest Bice
Eric Chin
Pranav Gandhi
James Grammatikos
Cole Levi
Scott Lindeman
Jack Noble
Alex Schwartz
Brendan Wilson

David Van Horn
Matthias Felleisen

Lisp
Fun
Games
Learning
Programming
Stories with Comics
Web Sites, Songs, and Videos

ROAR: Realm of Racket



Realm of Racket

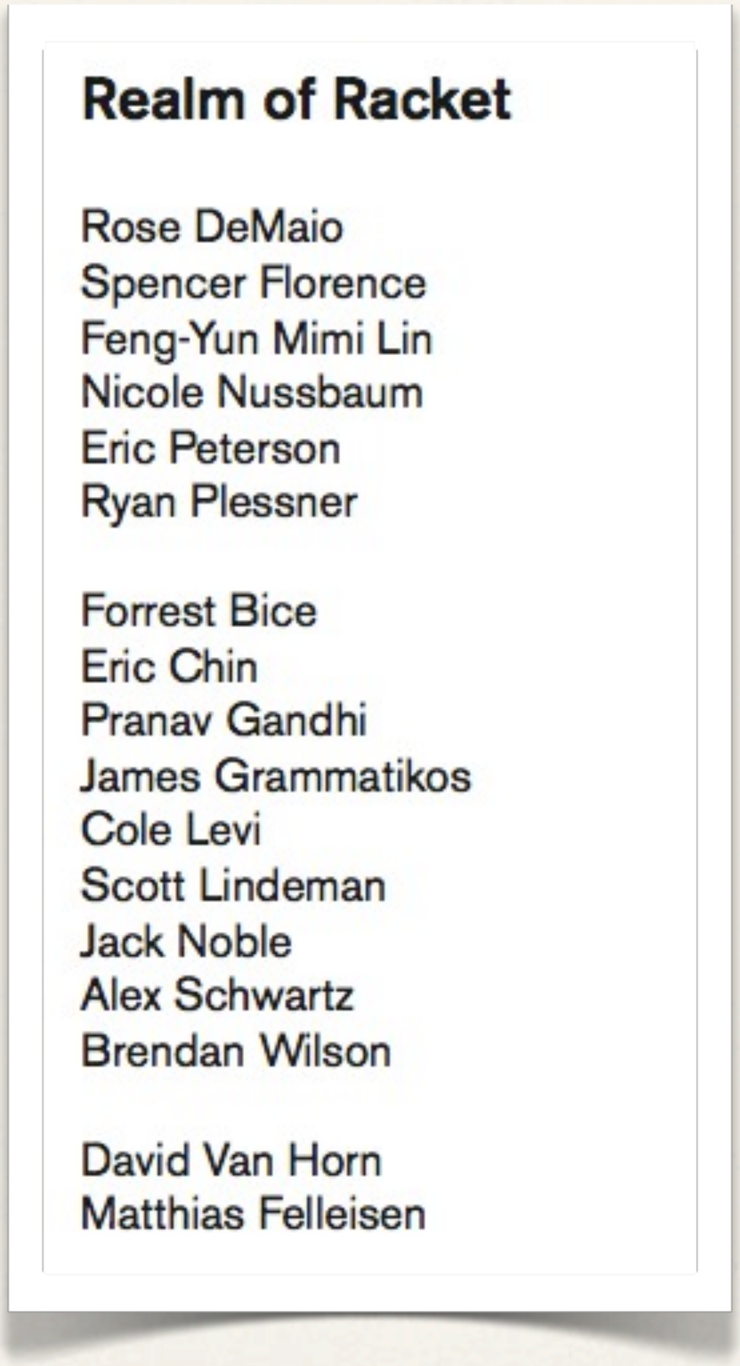
Rose DeMaio
Spencer Florence
Feng-Yun Mimi Lin
Nicole Nussbaum
Eric Peterson
Ryan Plessner

Forrest Bice
Eric Chin
Pranav Gandhi
James Grammatikos
Cole Levi
Scott Lindeman
Jack Noble
Alex Schwartz
Brendan Wilson

David Van Horn
Matthias Felleisen

~~Lisp~~
Fun
Games
Learning
Programming
Stories with Comics
Web Sites, Songs, and Videos

ROAR: Realm of Racket



Realm of Racket

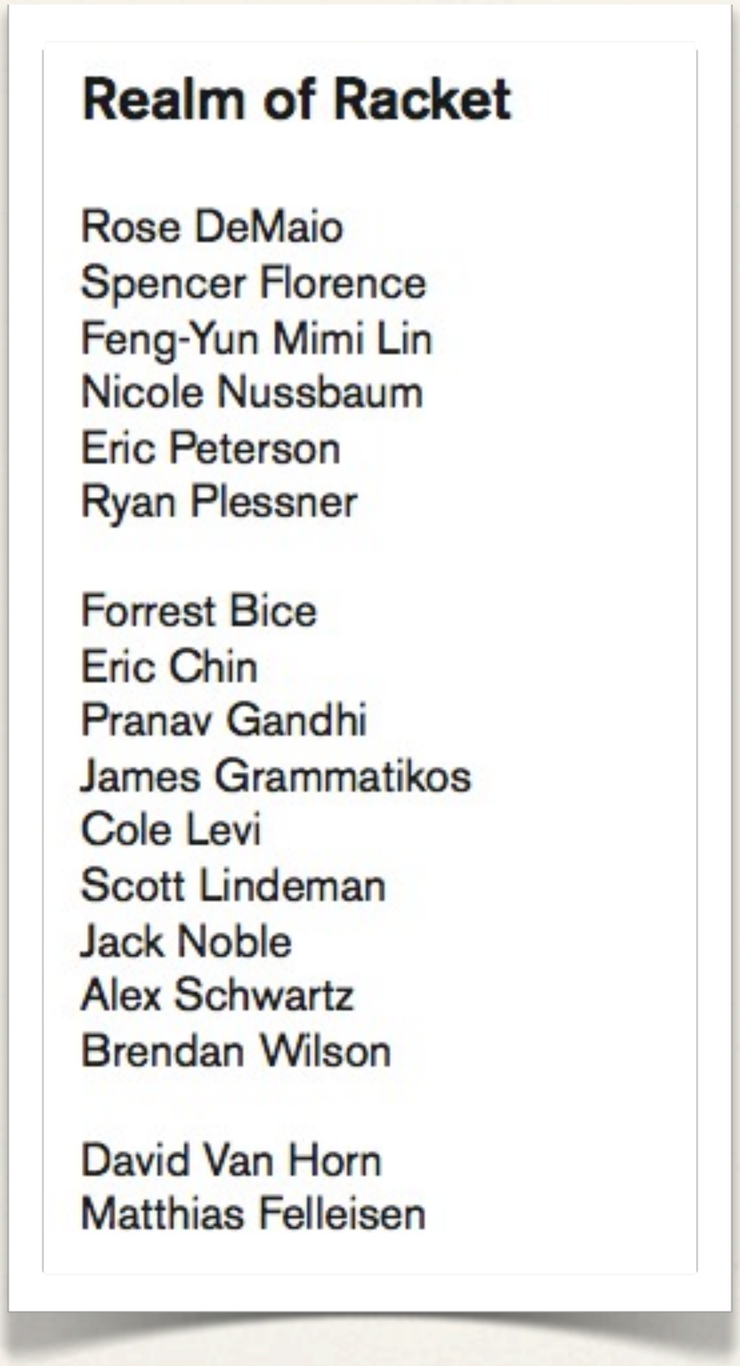
Rose DeMaio
Spencer Florence
Feng-Yun Mimi Lin
Nicole Nussbaum
Eric Peterson
Ryan Plessner

Forrest Bice
Eric Chin
Pranav Gandhi
James Grammatikos
Cole Levi
Scott Lindeman
Jack Noble
Alex Schwartz
Brendan Wilson

David Van Horn
Matthias Felleisen

Fun
Games
Learning
Programming
Stories with Comics
Web Sites, Songs, and Videos

ROAR: Realm of Racket



Realm of Racket

Rose DeMaio
Spencer Florence
Feng-Yun Mimi Lin
Nicole Nussbaum
Eric Peterson
Ryan Plessner

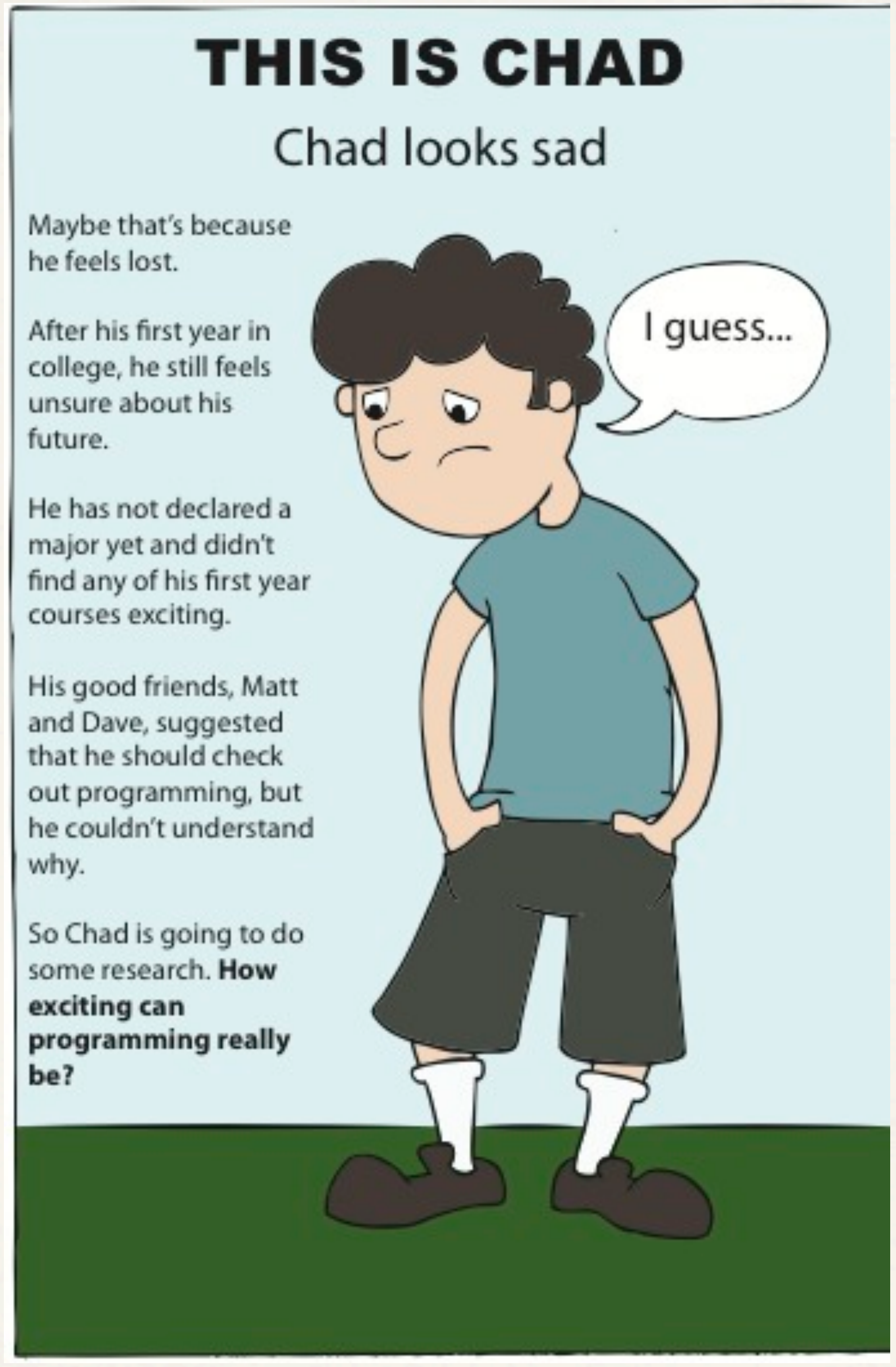
Forrest Bice
Eric Chin
Pranav Gandhi
James Grammatikos
Cole Levi
Scott Lindeman
Jack Noble
Alex Schwartz
Brendan Wilson

David Van Horn
Matthias Felleisen

Racket

Fun
Games
Learning
Programming
Stories with Comics
Web Sites, Songs, and Videos

Inofficial Launch



Maybe that's because he feels lost.

After his first year in college, he still feels unsure about his future.

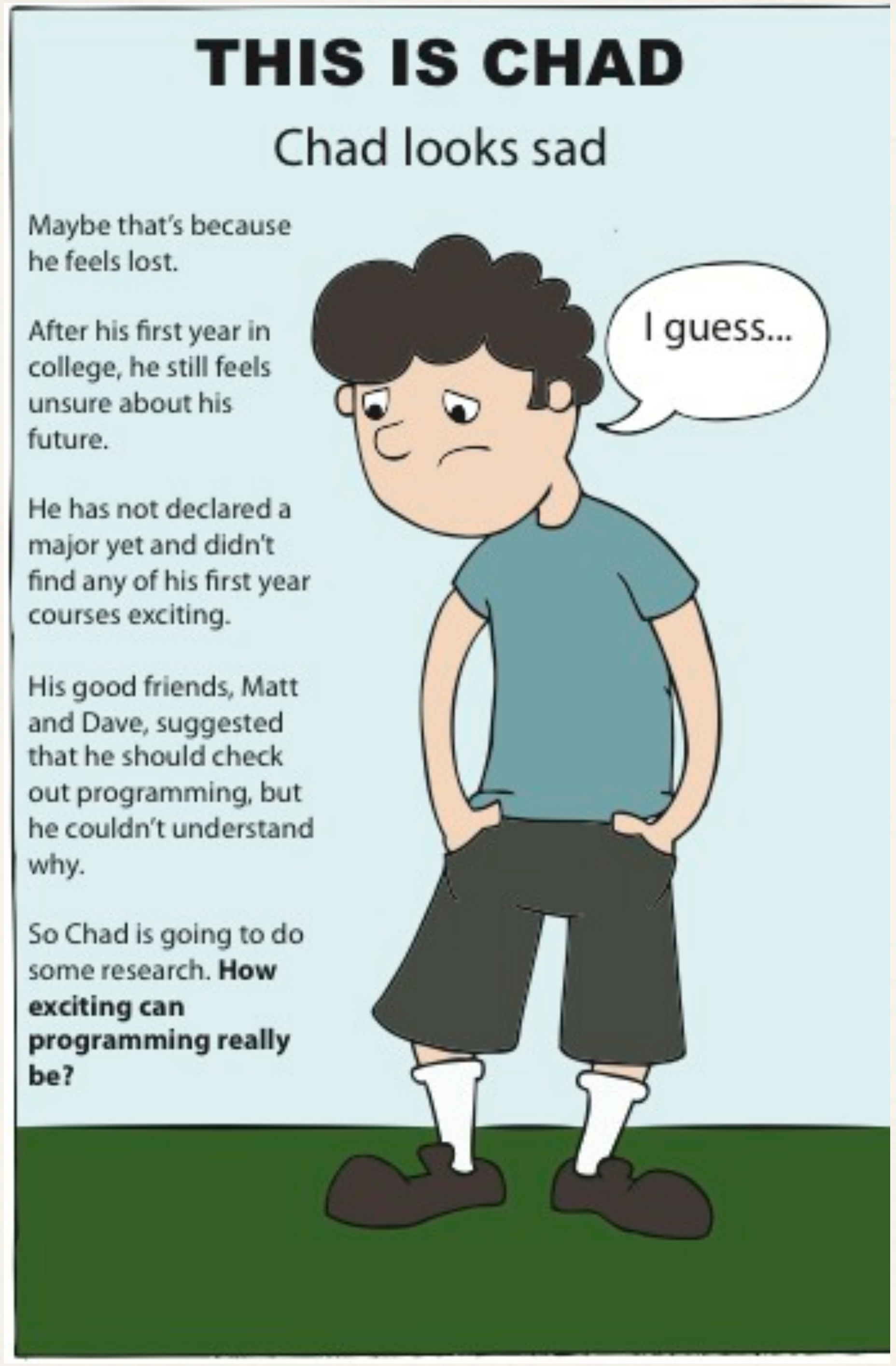
He has not declared a major yet and didn't find any of his first year courses exciting.

His good friends, Matt and Dave, suggested that he should check out programming, but he couldn't understand why.

So Chad is going to do some research. **How exciting can programming really be?**

by freshmen,
for freshmen

Inofficial Launch


THIS IS CHAD
Chad looks sad
Maybe that's because he feels lost.
After his first year in college, he still feels unsure about his future.
He has not declared a major yet and didn't find any of his first year courses exciting.
His good friends, Matt and Dave, suggested that he should check out programming, but he couldn't understand why.
So Chad is going to do some research. **How exciting can programming really be?**
I guess...

by freshmen, for freshmen

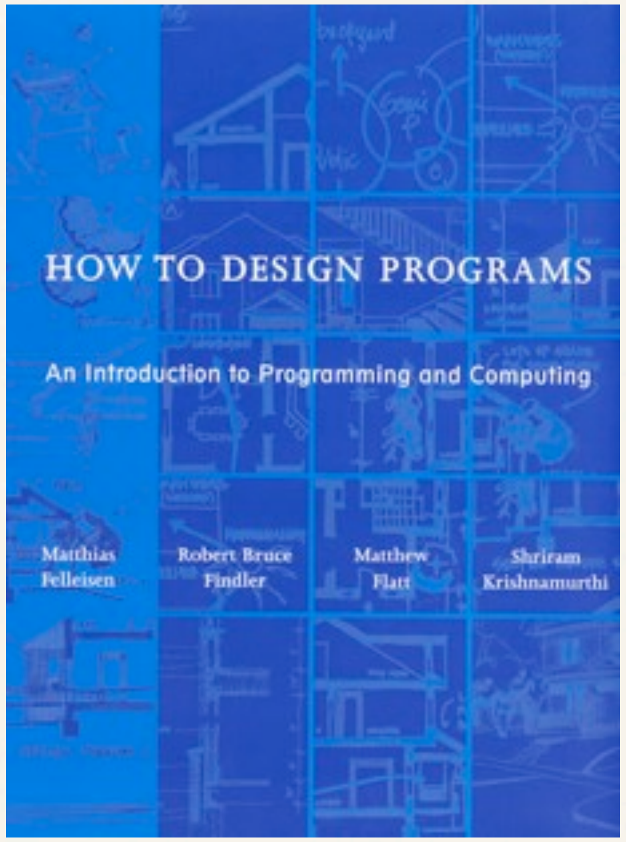
David van Horn

Mimi Lin
Nicole Nussbaum
Spencer Florence
Pranav Gandhi

We need your help.
When we launch,
please spread the word.
Watch users@racket-lang.org
for announcements.

Outside
Context

Content

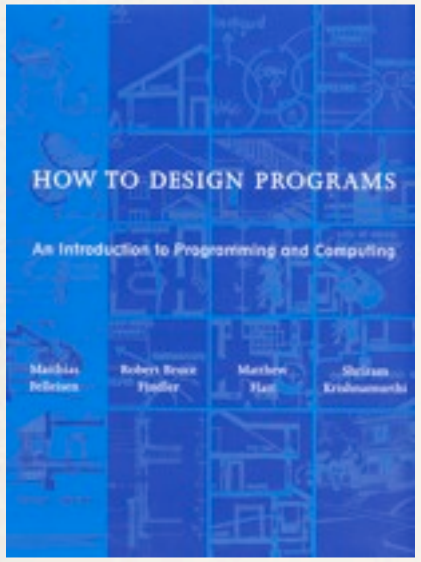

HOW TO DESIGN PROGRAMS

An Introduction to Programming and Computing

Matthias Felleisen
Robert Bruce Findler
Matthew Flatt
Shriram Krishnamurthi

Academic
Context

Academic Context

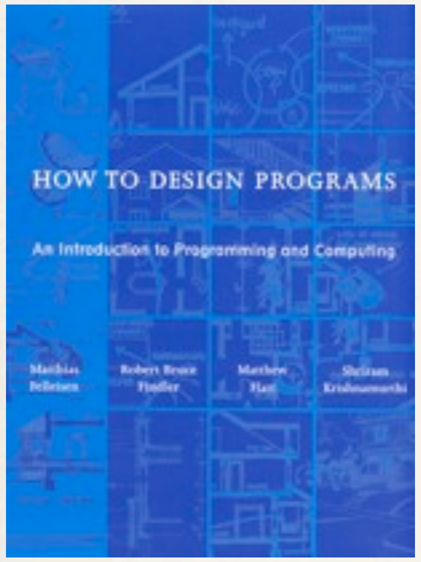


How to Design **Programs**
How to Design **Components**
How to Design **Systems**

How to Prove **Programs**

$\}$ Racket

Academic Context



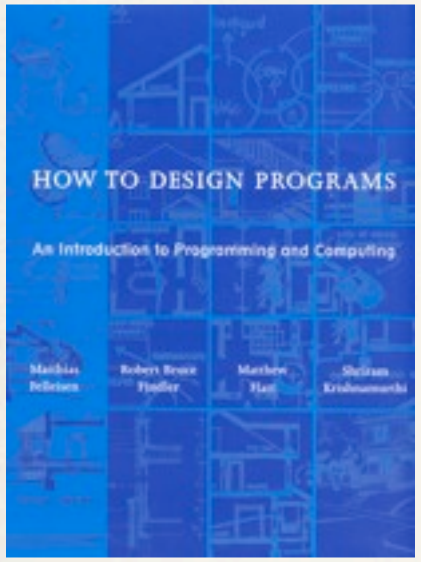
Sam Tobin-Hochstadt
David van Horn

How to Design **Programs**
How to Design **Components**
How to Design **Systems**

How to Prove **Programs**

} Racket

Academic Context



Sam Tobin-Hochstadt
David van Horn

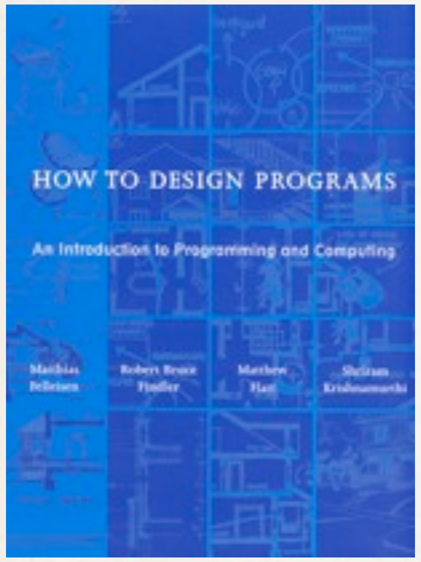
How to Design **Programs**
How to Design **Components**
How to Design **Systems**

How to Prove **Programs**

} Racket

Carl Eastlund
Daniel Friedman

Academic Context



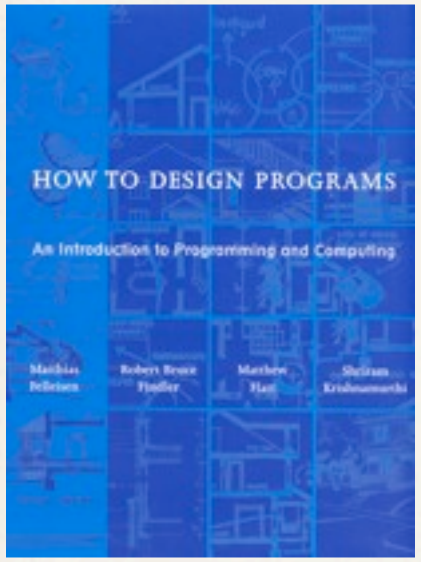
transition to 'regular' programming:

arrays

for loops

types

Academic Context



HtDP/2e

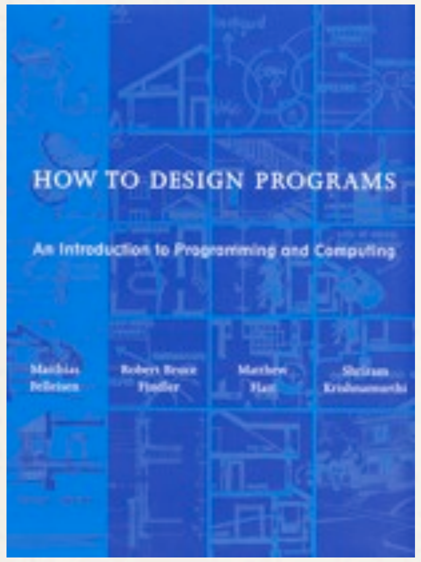
transition to ‘regular’ programming:

arrays

for loops

types

Academic Context



transition to 'regular' programming:

arrays

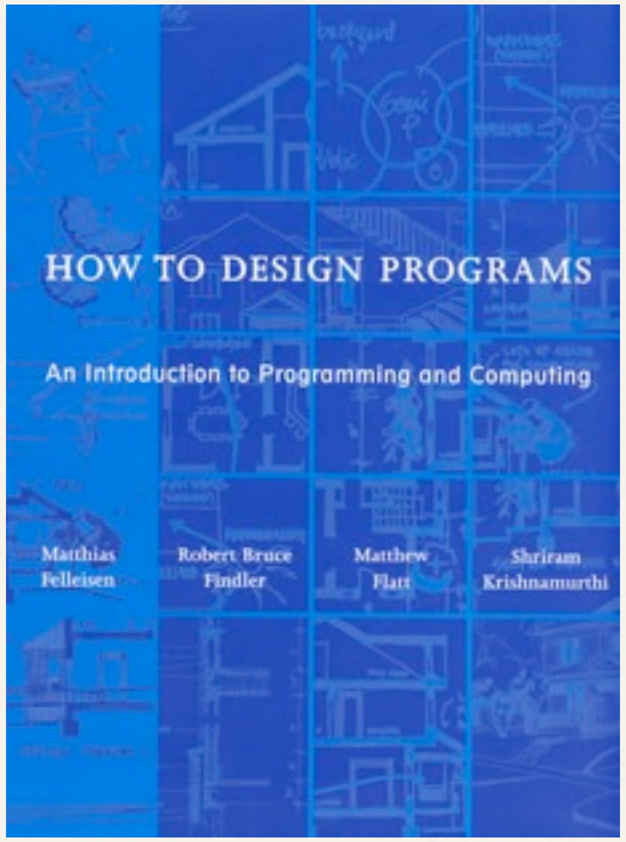
for loops

types

HtDP/2e

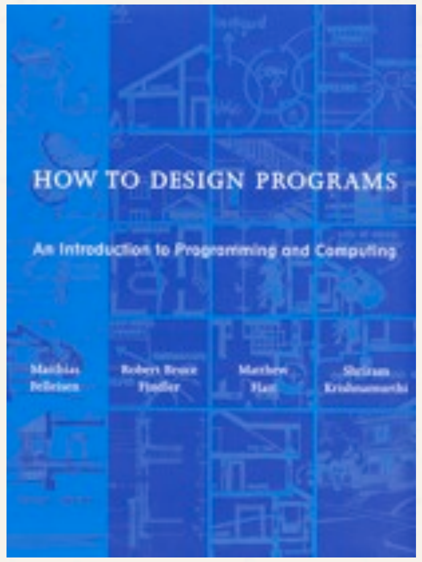
HtDP/3e:
signatures
types
contracts

Content: HtDP/2e



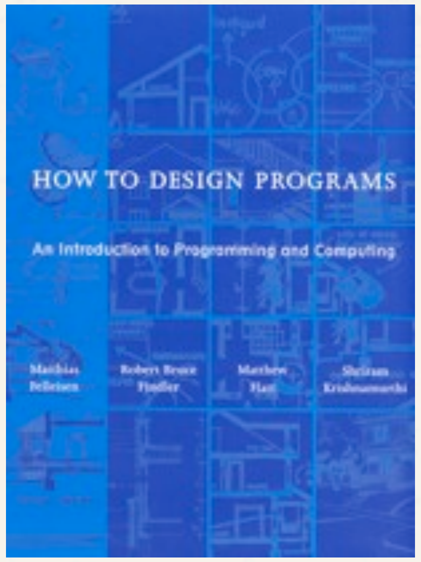
Context

Content HtDP/2e



design recipes
design guidelines
topics, order of (mostly)

Content
HtDP/2e



design recipes
design guidelines
topics, order of (mostly)

algorithmic trade-offs for design
animation, games (context)
modularity plus ADTs
real-world data (context)
vectors and iterators

HtDP/2e order of topics

finite data
simple recursive data
functional abstraction
complex recursive data
generative recursion
design with accumulators
modules and abstract data
  functional data representations
loops and iterators

HtDP / 2e order of topics

finite data
simple recursive data
functional abstraction
complex recursive data
generative recursion
design with accumulators
modules and abstract data
  functional data representations
loops and iterators

missing:
- mutable variables
- mutable structures

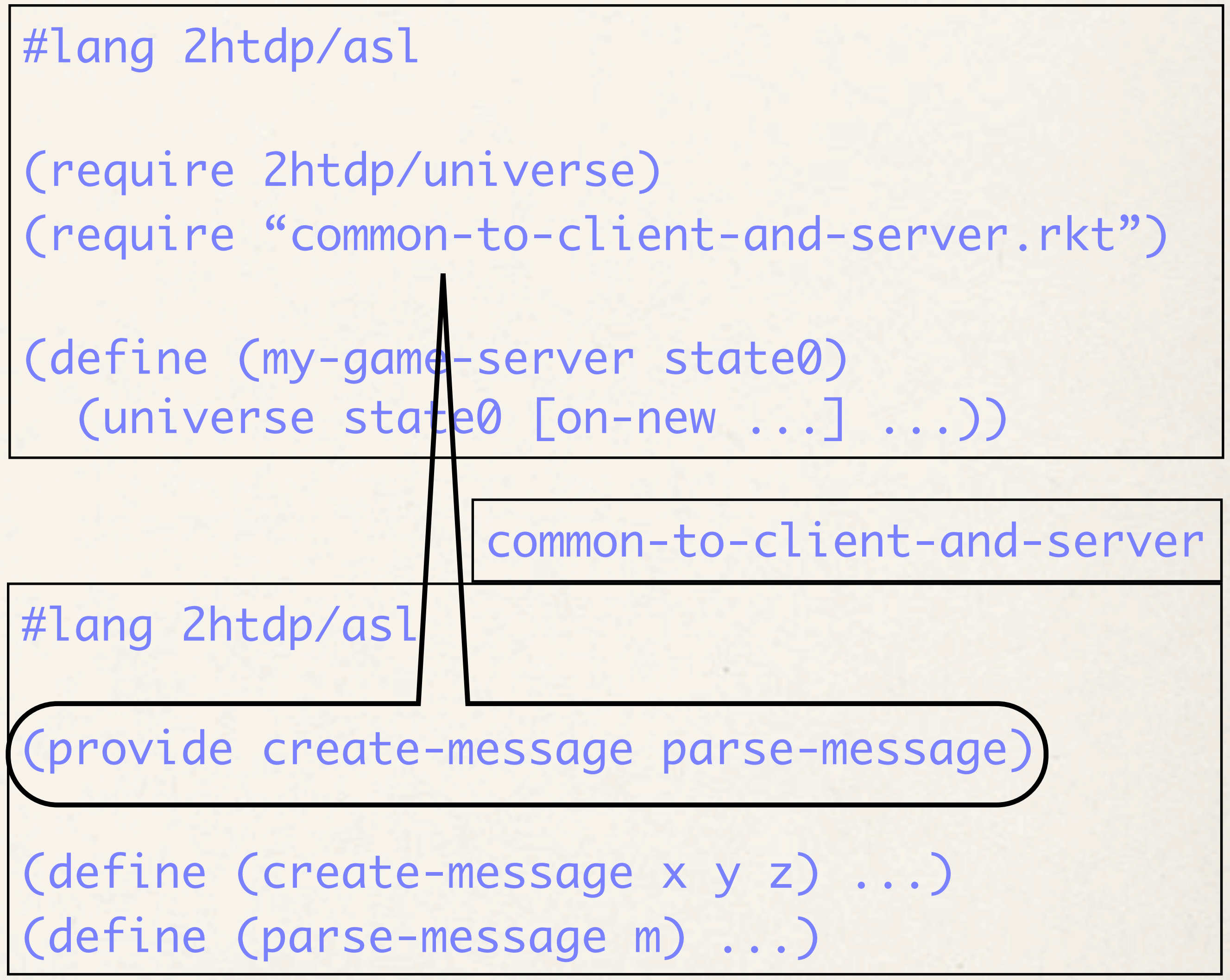
HtDP / 2e
[order of] topics

finite data
simple recursive data
functional abstraction
complex recursive data
generative recursion
design with accumulators
modules and abstract data
  functional data representations
loops and iterators

```
#lang 2htdp/asl

(require 2htdp/universe)
(require “common-to-client-and-server.rkt”)

(define (my-game-server state0)
  (universe state0 [on-new ...] ...))
```

HtDP / 2e [order of] topics

finite data
simple recursive data
functional abstraction
complex recursive data
generative recursion
design with accumulators
modules and abstract data
    functional data representations
loops and iterators



```
#lang 2htdp/asl

(require 2htdp/universe)
(require “common-to-client-and-server.rkt”)

(define (my-game-server state0)
  (universe state0 [on-new ...] ...))
```

```
common-to-client-and-server
```

```
#lang 2htdp/asl

(provide create-message parse-message)

(define (create-message x y z) ...)
(define (parse-message m) ...)
```

HtDP/2e
[order of] topics

finite data
simple recursive data
functional abstraction
complex recursive data
generative recursion
design with accumulators
modules and data abstraction
  functional data representations
loops and iterators

HtDP/2e
[order of] topics

finite data
simple recursive data
functional abstraction
complex recursive data
generative recursion
design with accumulators
modules and data abstraction
  functional data representations
loops and iterators

examples:
+ finite sets
+ dictionary/hashes
+ infinite sets

HtDP/2e
[order of] topics

finite data
simple recursive data
functional abstraction
complex recursive data
generative recursion
design with accumulators
modules and data abstraction
  functional data representations
loops and iterators

examples:
+ finite sets
+ dictionary/hashes
+ infinite sets

```
#lang 2htdp/isl
;; Set = [Any -> Boolean]

;; Set Set -> Set

(check-expect
  (element-of (union odd? even?)
              (random 100000))
   true)

(define (union s t)
  (lambda (x)
    (or (s x) (t x))))
```

HtDP / 2e
[order of] topics

finite data
simple recursive data
functional abstraction
complex recursive data
generative recursion
design with accumulators
modules and abstract data
  functional data representations
loops and iterators

```
#lang 2htdp/asl

;; [Vectorof Number] -> Number

(check-within (norm (vector 1 1 1)) (sqrt 3)
              .0001)

(define (norm v)
  (sqrt
   (for/fold ((sum 0)) ((x v))
     (+ sum (sqr x)))))
```

HtDP / 2e [order of] topics

finite data
simple recursive data
functional abstraction
complex recursive data
generative recursion
design with accumulators
modules and abstract data
  functional data representations
loops and iterators

```
#lang 2htdp/asl

;; [Vectorof Number] -> Number

(check-within (norm (vector 1 1 1)) (sqrt 3)
              .0001)

(define (norm v)
  (sqrt
   (for/fold ((sum 0)) ((x v))
     (+ sum (sqr x)))))
```

```
#lang 2htdp/asl

;; Number [Vectorof Number] -> [Vectorof Number]

(check-expect (scalar* 3 (vector 0 -4 2))
              (vector 0 -12 6))

(define (scalar* a v)
  (for/vector ((x v))
    (* a x)))
```

HtDP/2e
algorithmic trade-off

...

functional abstraction

intermezzo: O(...), running time, vectors

complex recursive data:

  lookup in lists vs BSTs

  measurements

generative recursion:

  insertion sort vs quicksort,

  graph traversals based on lists, vectors, links

design with accumulators:

  more data accumulators (invariants)

  tree structures w/ accumulators

...

HtDP/2e context: animation & games & real data

HtDP / 2e

~~context: animation & games & real data~~

HtDP/2e

~~context: animation & games & real data~~

Input/Output:

HtDP/2e

~~context: animation & games & real data~~

Input/Output:

$\boxed{\text{interactive I/O}}$

HtDP/2e

~~context: animation & games & real data~~

Input/Output:

interactive I/O	batch (file, net) I/O

HtDP/2e

~~context: animation & games & real data~~

Input/Output:

how to build a *complete* application	
interactive I/O	batch (file, net) I/O

HtDP/2e
context: real data

finite data
simple recursive data
functional abstraction
complex recursive data
generative recursion
design with accumulators
modules and abstract data
   functional data representations
loops and iterators

```
#lang 2htdp/bsl

(require 2htdp/batch-io)
(require 2htdp/itunes)

;; String -> [Listof iTuneRecords]
(define (retriev-database file-name)
   (list->iTune-Record
     (read-file-as-list file-name)))

... process titles, singers, ...
```

HtDP/2e context: real data

finite data
simple recursive data
functional abstraction
complex recursive data
generative recursion
design with accumulators
modules and abstract data
  functional data representations
loops and iterators

```
#lang 2htdp/isl

(require 2htdp/batch-io)

;; String -> [Listof iTuneRecords]
(define (retriev-database file-name)
   (write-as-csv-file
     (add-row-to-spread-sheet
       (read-file-as-csv file-name
                         process-cell))))

;; Cell -> ...
(define (process-cell c)
   ...)
```

HtDP/2e
context: real data

finite data
simple recursive data
functional abstraction
complex recursive data
generative recursion
design with accumulators
modules and abstract data
  functional data representations
loops and iterators

```
#lang 2htdp/isl

(require 2htdp/universe)
(require 2htdp/batch-io)
(require "google-yahoo-credentials.rkt")

(define (main s)
  (big-bang (retrieve-maps
              (retrieve-coordinates s))
            [to-draw draw-first-map]
            [on-key rotate-maps]))

;; Address -> Coordinates
(define (retrieve-coordinates address)
   (read-url YAHOO-GEO-SERVICE ...))

;; Coordinates -> [Listof Image]
(define (retrieve-maps coordinates)
   (read-url GOOGLE-MAPS ...))
```

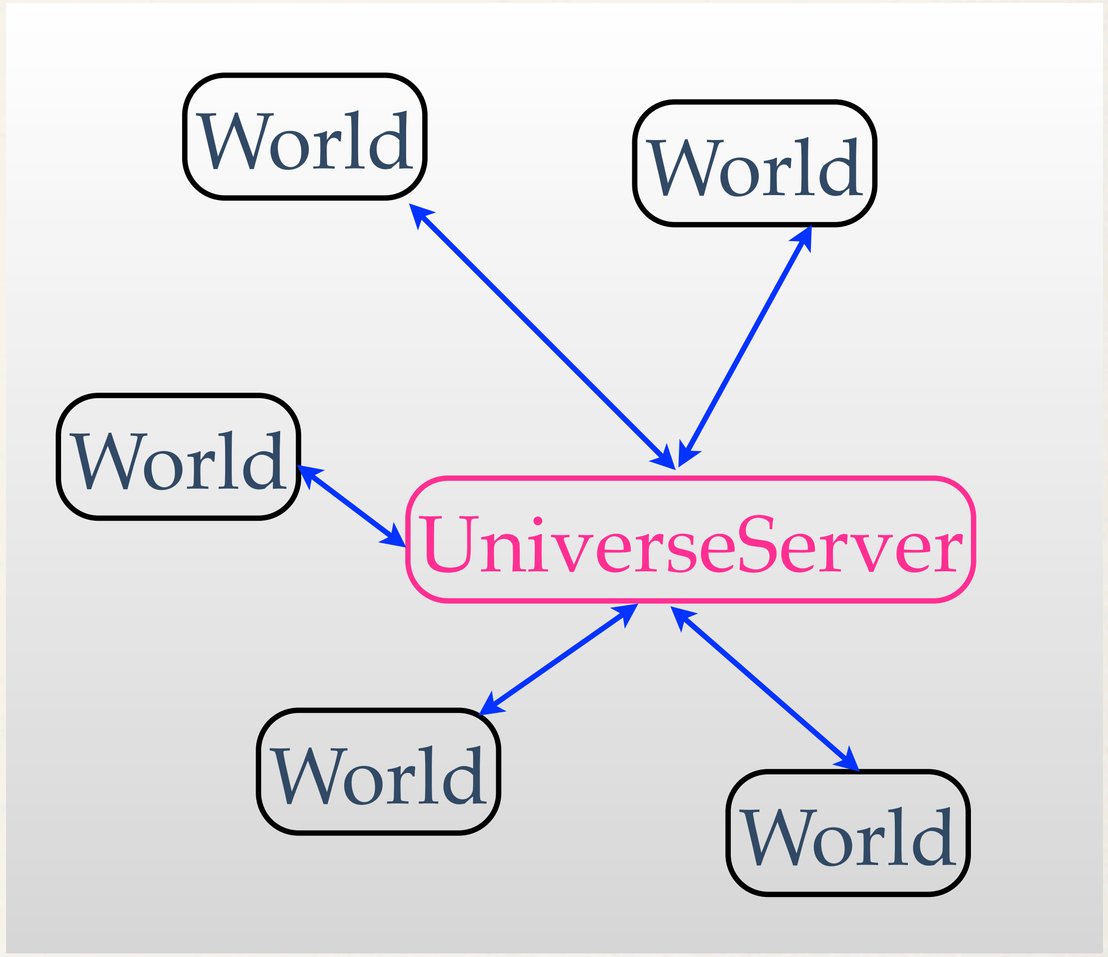
HtDP / 2e
context: beyond big-bang

finite data
simple recursive data
functional abstraction
complex recursive data
generative recursion
design with accumulators
modules and abstract data
   functional data representations
loops and iterators

```
#lang 2htdp/isl

(require 2htdp/universe)
(require 2htdp/image)

;; Nat -> OrcWorld
(define (launch-orc-battle s)
  (big-bang (create-orcs-and-fighter s)
            [to-draw render-orc-game]
            [on-key fight-orcs]
            [stop-when win-or-lose?]
            [on-tick counting-down]))


...
```

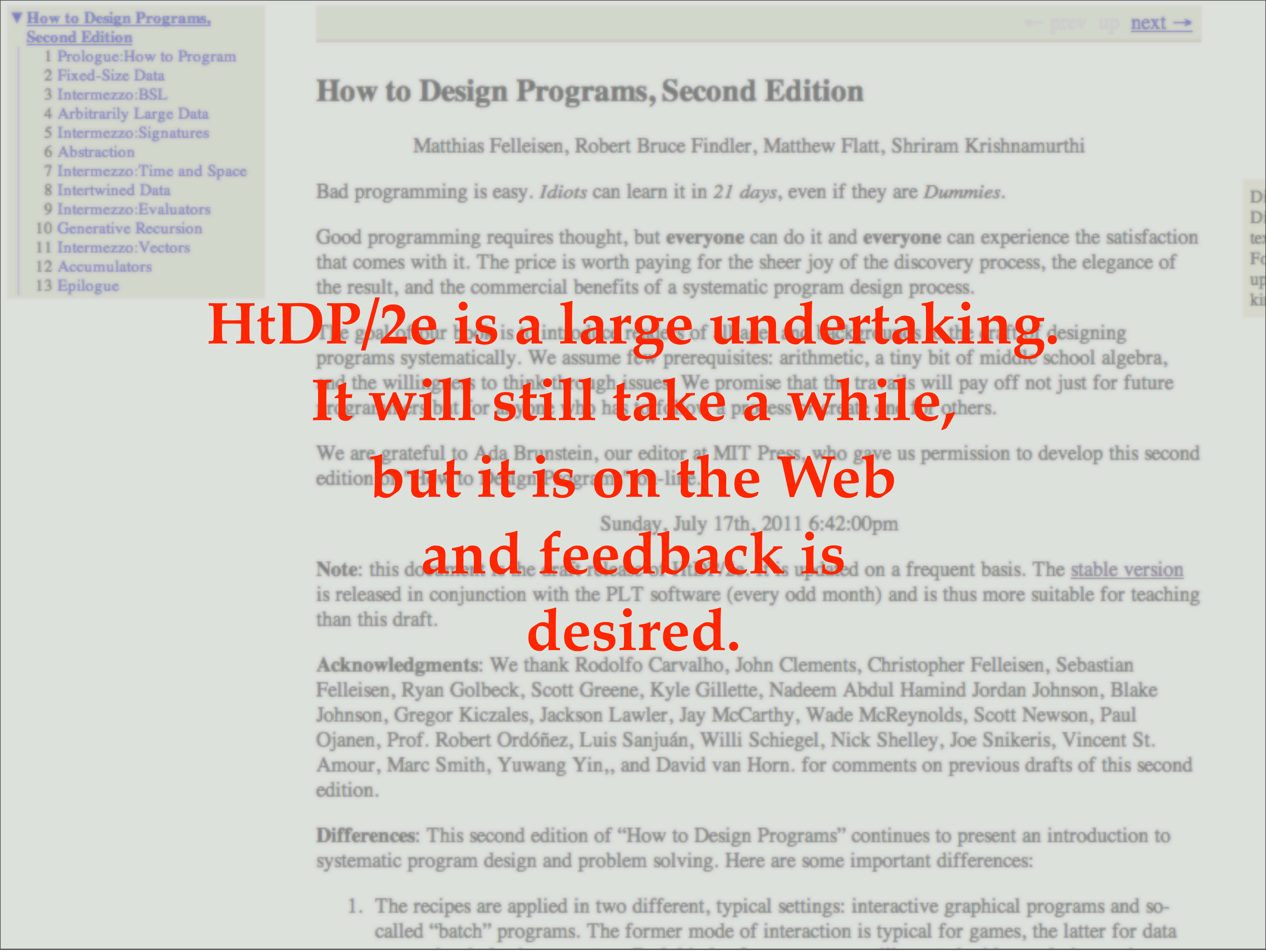
HtDP / 2e
context: beyond big-bang

finite data
simple recursive data
functional abstraction
complex recursive data
generative recursion
design with accumulators
modules and abstract data
  functional data representations
loops and iterators



stop

HtDP/2e is a large undertaking.
It will still take a while,
but it is on the Web
and feedback is
desired.



How to Design Programs, Second Edition

Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, Shriram Krishnamurthi

Bad programming is easy. *Idiots* can learn it in *21 days*, even if they are *Dummies*.

Good programming requires thought, but **everyone** can do it and **everyone** can experience the satisfaction that comes with it. The price is worth paying for the sheer joy of the discovery process, the elegance of the result, and the commercial benefits of a systematic program design process.

The goal of our book is to introduce readers of all ages and backgrounds to the craft of designing programs systematically. We assume few prerequisites: arithmetic, a tiny bit of middle school algebra, and the willingness to think through issues. We promise that the travails will pay off not just for future programmers but for anyone who has to follow a process or create one for others.

We are grateful to Ada Brunstein, our editor at MIT Press, who gave us permission to develop this second edition of How to Design Programs on-line.

Sunday, July 17th, 2011 6:42:00pm

Note: this document is the draft release of HtDP/2e. It is updated on a frequent basis. The stable version is released in conjunction with the PLT software (every odd month) and is thus more suitable for teaching than this draft.

Acknowledgments: We thank Rodolfo Carvalho, John Clements, Christopher Felleisen, Sebastian Felleisen, Ryan Golbeck, Scott Greene, Kyle Gillette, Nadeem Abdul Hamind Jordan Johnson, Blake Johnson, Gregor Kiczales, Jackson Lawler, Jay McCarthy, Wade McReynolds, Scott Newson, Paul Ojanen, Prof. Robert Ordóñez, Luis Sanjuán, Willi Schiegel, Nick Shelley, Joe Snikeris, Vincent St. Amour, Marc Smith, Yuwang Yin,, and David van Horn. for comments on previous drafts of this second edition.

Differences: This second edition of "How to Design Programs" continues to present an introduction to systematic program design and problem solving. Here are some important differences:

1. The recipes are applied in two different, typical settings: interactive graphical programs and so-called "batch" programs. The former mode of interaction is typical for games, the latter for data