6.1040: Software Design

Towards Concepts

Arvind Satyanarayan & Max Goldman

with material by Daniel Jackson

Fall ’24

Modularity

Specifications

Dependencies

Modularity

functions
abstract data types (encapsulation of data + operations, information hiding)
exports, access control

Specifications

preconditions and postconditions
pure functions and mutating functions (with abstract state)
grammars

Dependencies

representation independence

Modularity

New kind of module: *concepts*

Specifications

New tools for specs: *state machines* with *relational state*

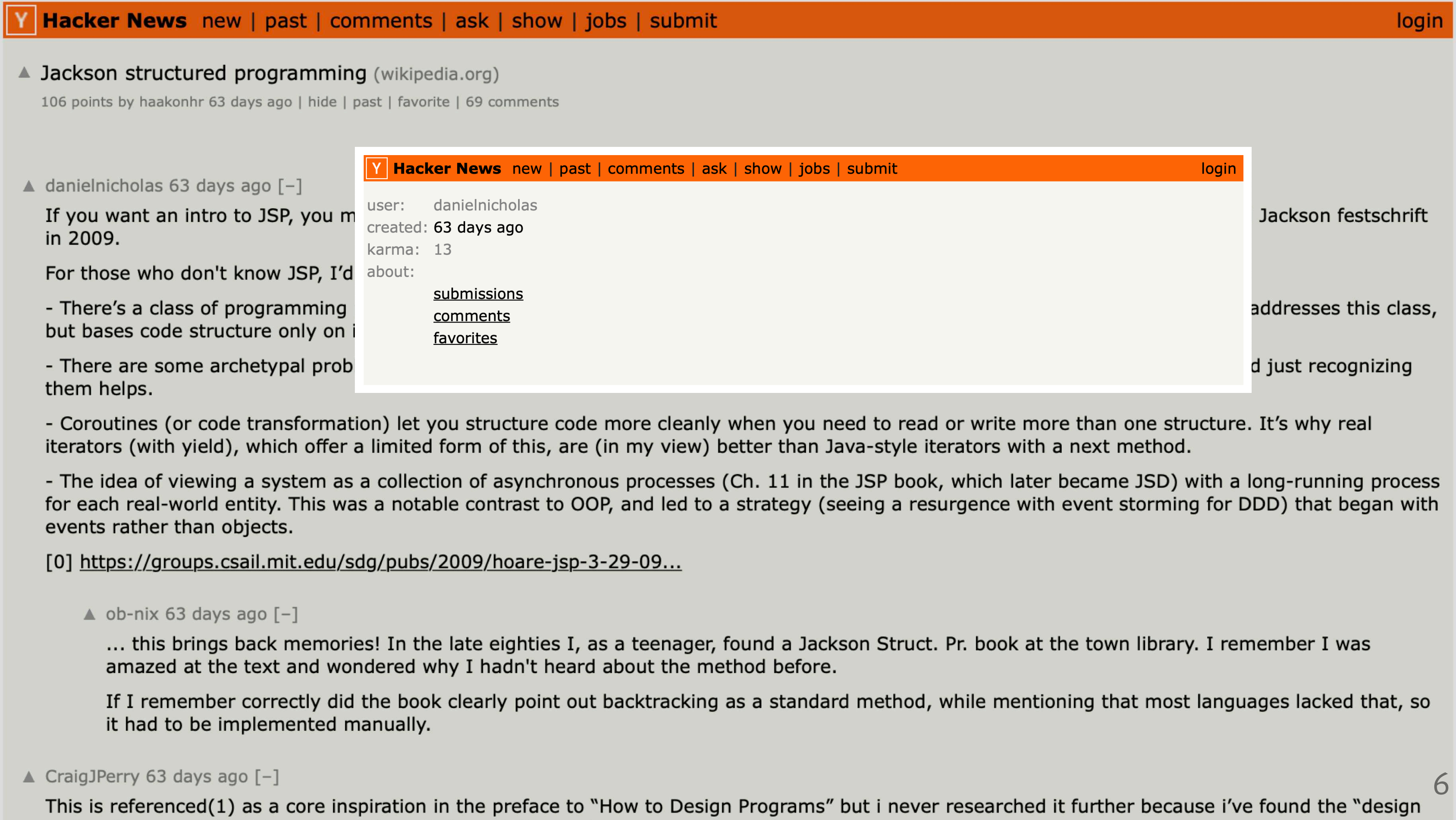
Dependencies

New focus on dependence between modules

Software designers

> When you go to design a house you talk to an architect first, not an engineer. Why is this?
>
> Because the criteria for what makes a good building fall outside the domain of engineering.
>
> Similarly, in computer programs, the selection of the various components must be driven by the conditions of use.
>
> How is this to be done? By software designers.

Mitchell Kapor, *A Software Design Manifesto* (1996)



Y **Hacker News** new | past | comments | ask | show | jobs | submit login

▲ Jackson structured programming (wikipedia.org)

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments

▲ danielnicholas 63 days ago [–]

If you want an intro to JSP, you m Jackson festschrift in 2009.

For those who don't know JSP, I’d

- There’s a class of programming addresses this class, but bases code structure only on i

- There are some archetypal prob d just recognizing them helps.

- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It’s why real iterators (with yield), which offer a limited form of this, are (in my view) better than Java-style iterators with a next method.

- The idea of viewing a system as a collection of asynchronous processes (Ch. 11 in the JSP book, which later became JSD) with a long-running process for each real-world entity. This was a notable contrast to OOP, and led to a strategy (seeing a resurgence with event storming for DDD) that began with events rather than objects.

[0] https://groups.csail.mit.edu/sdg/pubs/2009/hoare-jsp-3-29-09...

Y **Hacker News** new | past | comments | ask | show | jobs | submit login

user:	danielnicholas
created:	63 days ago
karma:	13
about:	
	submissions
	comments
	favorites

▲ ob-nix 63 days ago [–]

... this brings back memories! In the late eighties I, as a teenager, found a Jackson Struct. Pr. book at the town library. I remember I was amazed at the text and wondered why I hadn't heard about the method before.

If I remember correctly did the book clearly point out backtracking as a standard method, while mentioning that most languages lacked that, so it had to be implemented manually.

▲ CraigJPerry 63 days ago [–]

This is referenced(1) as a core inspiration in the preface to “How to Design Programs” but i never researched it further because i’ve found the “design

Software concepts

What is a concept?

Semantic

↳ about underlying behavior users experience

Not internals, user-facing

Not UI, but underlying function

Not just structure, behavior

Purposive

↳ fulfills an entire user need

Included for a reason

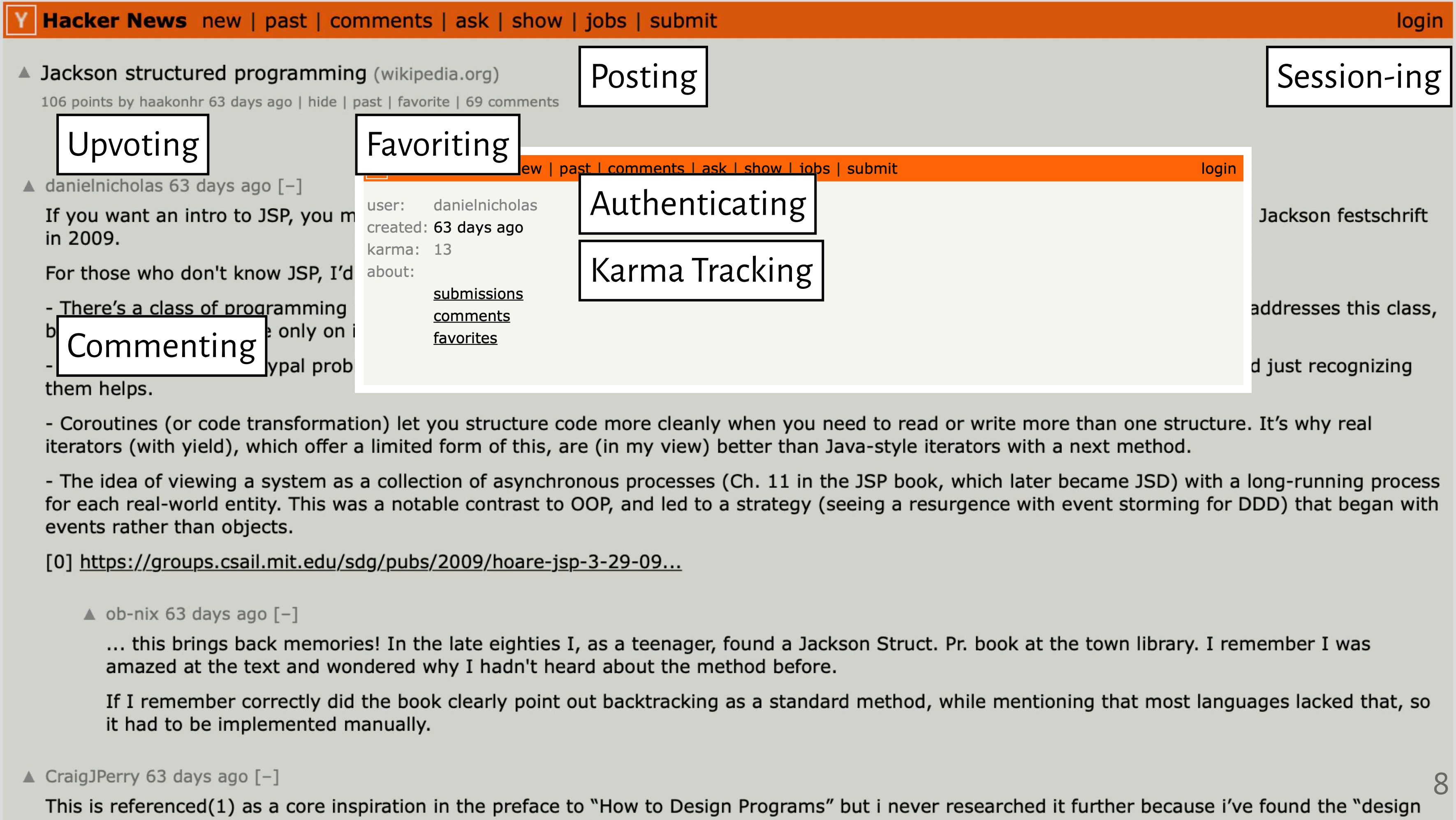
End-to-end, not just a fragment

Modular

↳ mutually independent

Generic (using polymorphic parameters)

Reusable within and across apps



Hacker News new | past | comments | ask | show | jobs | submit login

▲ Jackson structured programming (wikipedia.org)

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments

▲ danielnicholas 63 days ago [–]

If you want an intro to JSP, you m Jackson festschrift in 2009.

For those who don't know JSP, I'd

- There's a class of programming addresses this class, b only on i

- ypal prob d just recognizing them helps.

- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It's why real iterators (with yield), which offer a limited form of this, are (in my view) better than Java-style iterators with a next method.

- The idea of viewing a system as a collection of asynchronous processes (Ch. 11 in the JSP book, which later became JSD) with a long-running process for each real-world entity. This was a notable contrast to OOP, and led to a strategy (seeing a resurgence with event storming for DDD) that began with events rather than objects.

[0] https://groups.csail.mit.edu/sdg/pubs/2009/hoare-jsp-3-29-09...

▲ ob-nix 63 days ago [–]

... this brings back memories! In the late eighties I, as a teenager, found a Jackson Struct. Pr. book at the town library. I remember I was amazed at the text and wondered why I hadn't heard about the method before.

If I remember correctly did the book clearly point out backtracking as a standard method, while mentioning that most languages lacked that, so it had to be implemented manually.

▲ CraigJPerry 63 days ago [–]

This is referenced(1) as a core inspiration in the preface to "How to Design Programs" but i never researched it further because i've found the "design

ew | past | comments | ask | show | jobs | submit login

user: danielnicholas

created: 63 days ago

karma: 13

about:

submissions

comments

favorites

Posting

Session-ing

Upvoting

Favoriting

Authenticating

Karma Tracking

Commenting

What is a concept?

Semantic Purposive Modular

Explaining a concept

Name

The two hardest problems in computer science are:
(i) cache invalidation, (ii) naming things, and (iii) off by one errors

The two hardest problems in computer science are:
(i) people, (ii), convincing computer scientists that the hardest problem in CS is people, and (iii) off by one errors
-Jeff Bigham

What is a concept?

Semantic Purposive Modular

Explaining a concept

Name

Today's convention: _______ing, to distinguish from related entities or objects or types
Note that *The Essence of Software*, tutorial resources, *etc.* do **not** use this convention

Purpose

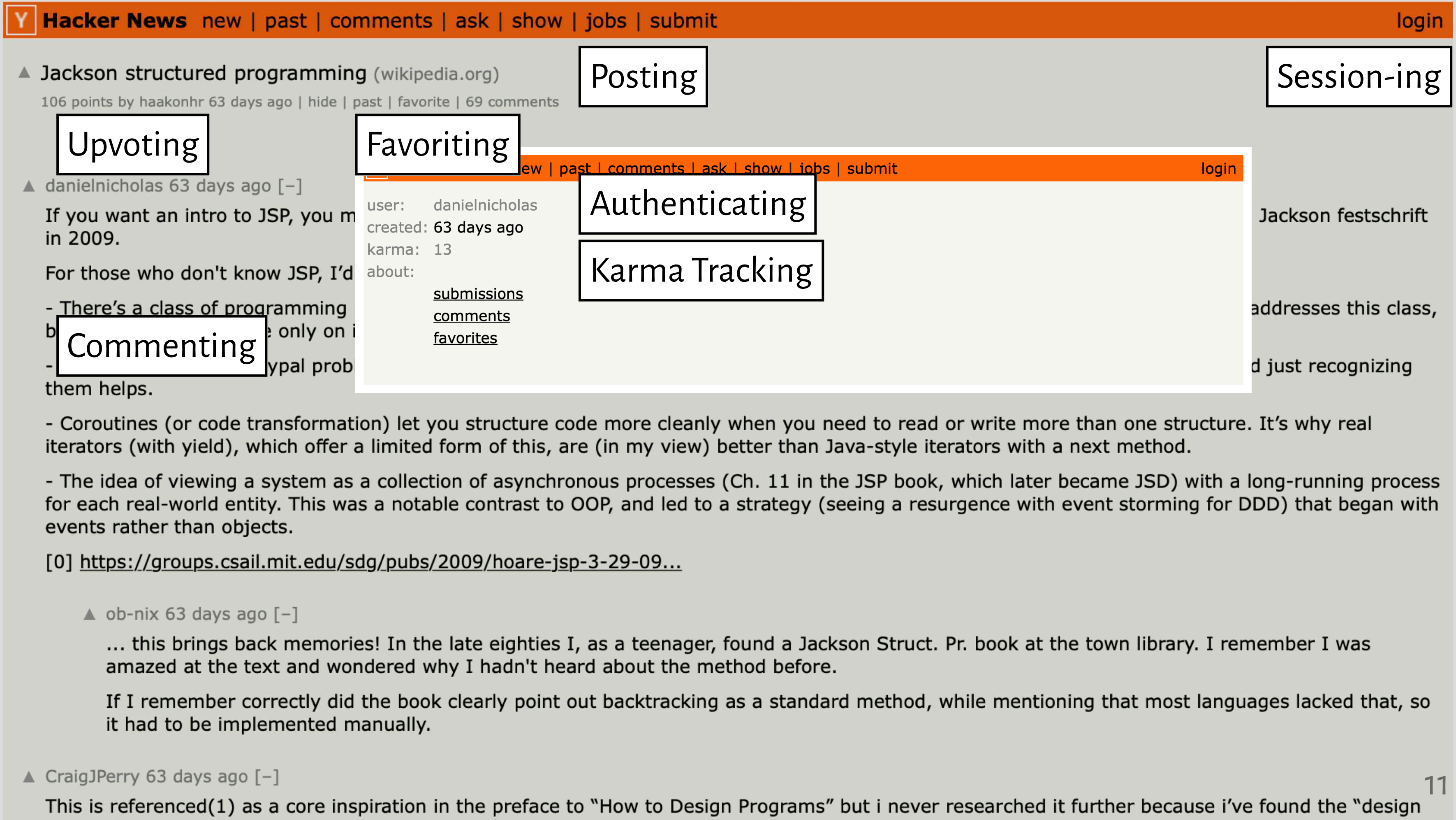
What is it for?

Operational principle

A small story that explains how it works
Idea due to Michael Polanyi

e.g. Reserving a Table

"after you make a reservation for time *t*,
and don't cancel it, when you arrive at
time *t*, there is a table for you"



Y **Hacker News** new | past | comments | ask | show | jobs | submit login

▲ Jackson structured programming (wikipedia.org)

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments

Posting

Session-ing

Upvoting

Favoriting

▲ danielnicholas 63 days ago [–]

If you want an intro to JSP, you m … Jackson festschrift in 2009.

For those who don't know JSP, I'd …

- There's a class of programming … addresses this class, b … only on i …

- … ypal prob … d just recognizing them helps.

Commenting

ew | past | comments | ask | show | jobs | submit login

user: danielnicholas

created: 63 days ago

karma: 13

about:

submissions

comments

favorites

Authenticating

Karma Tracking

- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It's why real iterators (with yield), which offer a limited form of this, are (in my view) better than Java-style iterators with a next method.

- The idea of viewing a system as a collection of asynchronous processes (Ch. 11 in the JSP book, which later became JSD) with a long-running process for each real-world entity. This was a notable contrast to OOP, and led to a strategy (seeing a resurgence with event storming for DDD) that began with events rather than objects.

[0] https://groups.csail.mit.edu/sdg/pubs/2009/hoare-jsp-3-29-09...

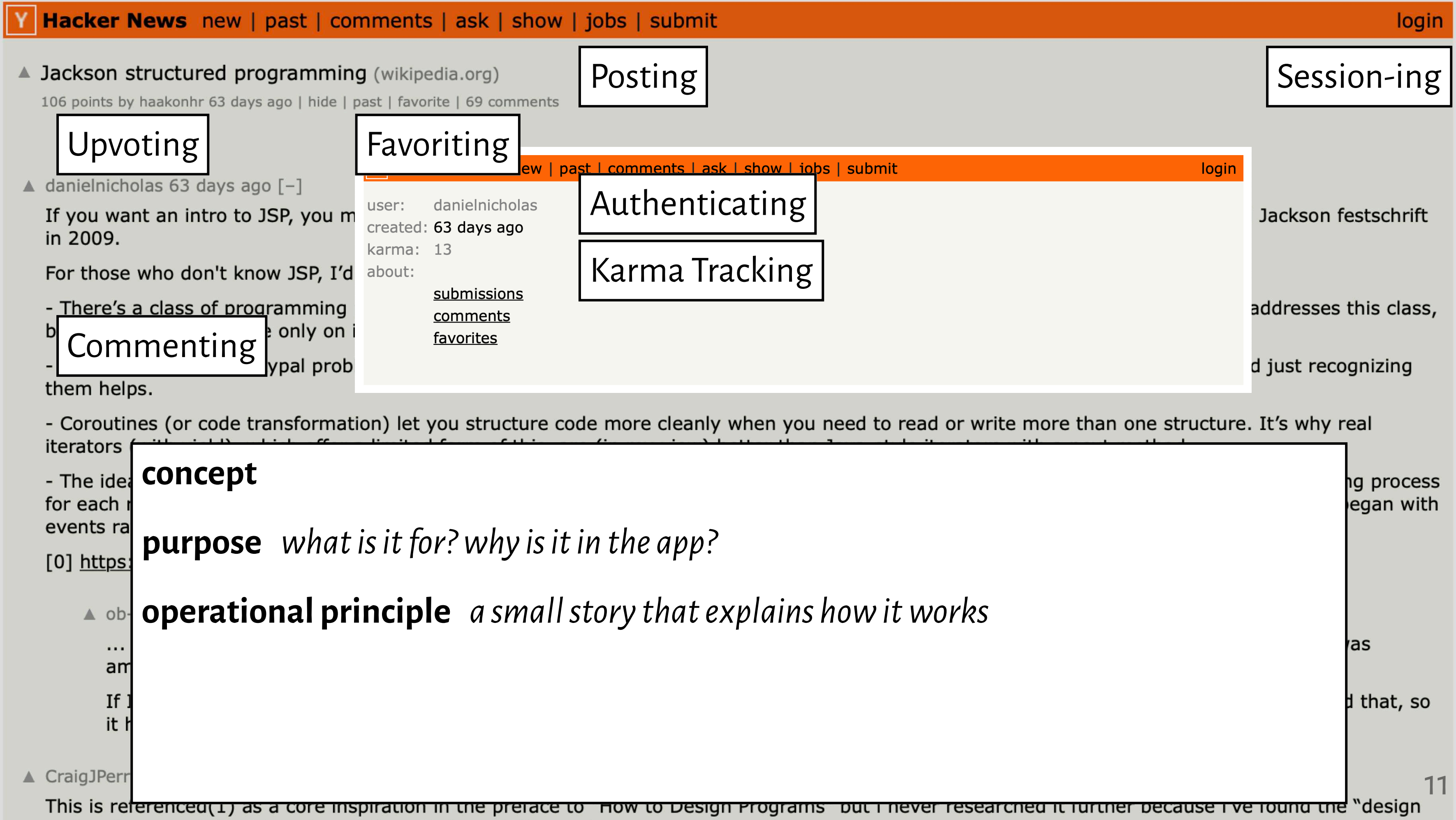
▲ ob-nix 63 days ago [–]

... this brings back memories! In the late eighties I, as a teenager, found a Jackson Struct. Pr. book at the town library. I remember I was amazed at the text and wondered why I hadn't heard about the method before.

If I remember correctly did the book clearly point out backtracking as a standard method, while mentioning that most languages lacked that, so it had to be implemented manually.

▲ CraigJPerry 63 days ago [–]

This is referenced(1) as a core inspiration in the preface to "How to Design Programs" but i never researched it further because i've found the "design



Hacker News new | past | comments | ask | show | jobs | submit login

▲ Jackson structured programming (wikipedia.org)

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments

▲ danielnicholas 63 days ago [–]

If you want an intro to JSP, you m … Jackson festschrift in 2009.

For those who don't know JSP, I'd

- There's a class of programming … addresses this class, b… only on i…

- … ypal prob … d just recognizing them helps.

- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It's why real iterators …

- The idea … ng process for each … egan with events ra…

[0] https:…

▲ ob-…

… as

am…

If I … that, so it h…

▲ CraigJPerr…

This is referenced(1) as a core inspiration in the preface to "How to Design Programs" but I never researched it further because I've found the "design

…ew | past | comments | ask | show | jobs | submit login

user: danielnicholas

created: 63 days ago

karma: 13

about:

submissions

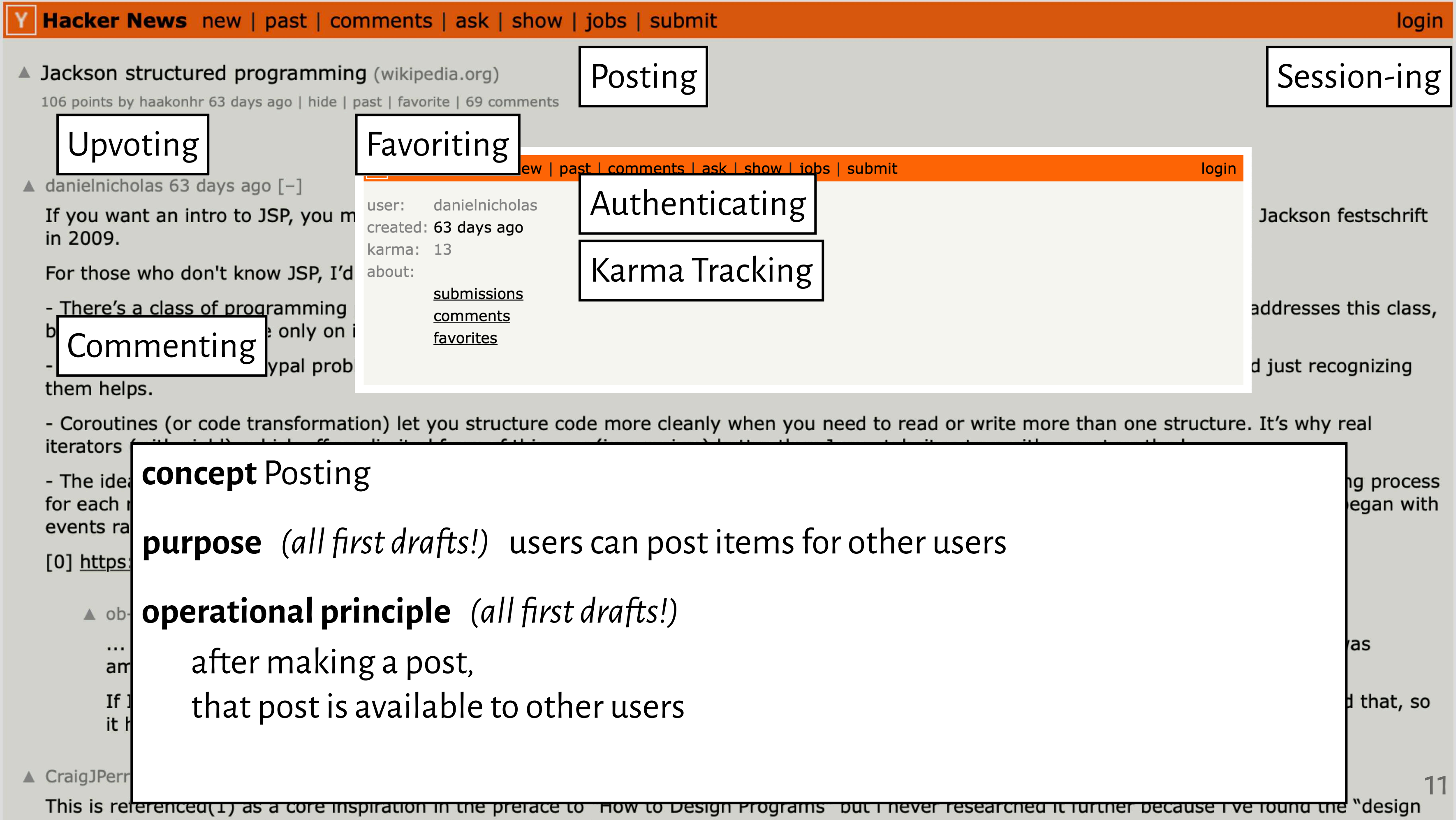
comments

favorites

concept

purpose *what is it for? why is it in the app?*

operational principle *a small story that explains how it works*



Hacker News new | past | comments | ask | show | jobs | submit login

▲ Jackson structured programming (wikipedia.org)

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments

▲ danielnicholas 63 days ago [–]

If you want an intro to JSP, you m Jackson festschrift in 2009.

For those who don't know JSP, I'd

- There's a class of programming addresses this class, b only on i

- ypal prob d just recognizing them helps.

- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It's why real iterators

- The idea ng process for each egan with events ra

[0] https:

▲ ob-

am as

If I that, so it h

▲ CraigJPerr

This is referenced(1) as a core inspiration in the preface to "How to Design Programs" but I never researched it further because I've found the "design

ew | past | comments | ask | show | jobs | submit login

user: danielnicholas
created: 63 days ago
karma: 13
about:

submissions
comments
favorites

Posting

Session-ing

Upvoting

Favoriting

Authenticating

Karma Tracking

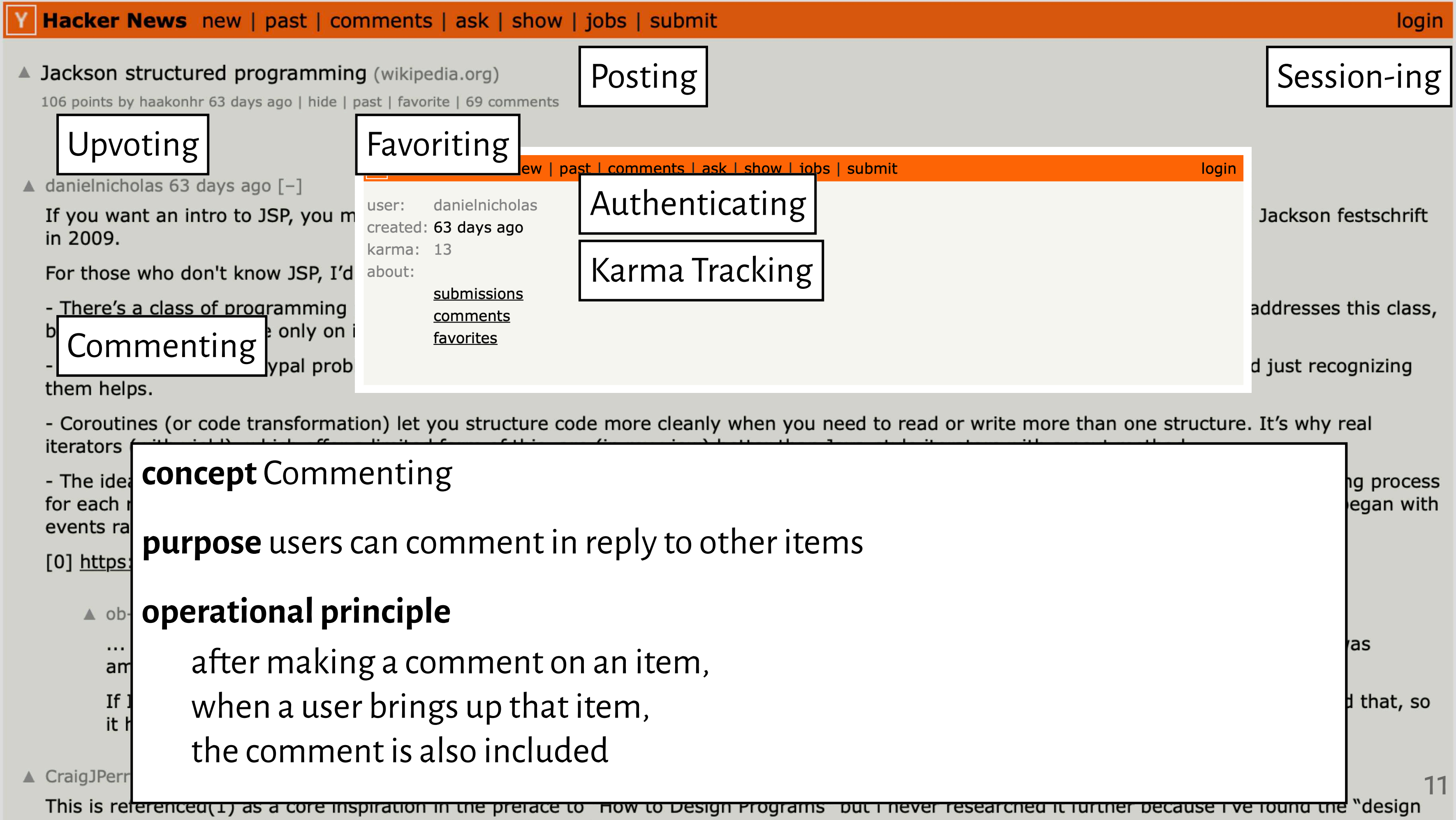
Commenting

concept Posting

purpose *(all first drafts!)* users can post items for other users

operational principle *(all first drafts!)*

after making a post,
that post is available to other users



Y **Hacker News** new | past | comments | ask | show | jobs | submit login

▲ Jackson structured programming (wikipedia.org)

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments

▲ danielnicholas 63 days ago [–]

If you want an intro to JSP, you m … Jackson festschrift in 2009.

For those who don't know JSP, I'd

- There's a class of programming … addresses this class, b … only on i

- … ypal prob … d just recognizing them helps.

- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It's why real iterators

- The idea … ng process for each … egan with events ra

[0] https:

▲ ob-

…
am … as

If I … d that, so it h

▲ CraigJPerr

This is referenced(1) as a core inspiration in the preface to "How to Design Programs" but I never researched it further because I've found the "design

ew | past | comments | ask | show | jobs | submit login

user: danielnicholas
created: 63 days ago
karma: 13
about:

submissions
comments
favorites

Posting

Session-ing

Upvoting

Favoriting

Authenticating

Karma Tracking

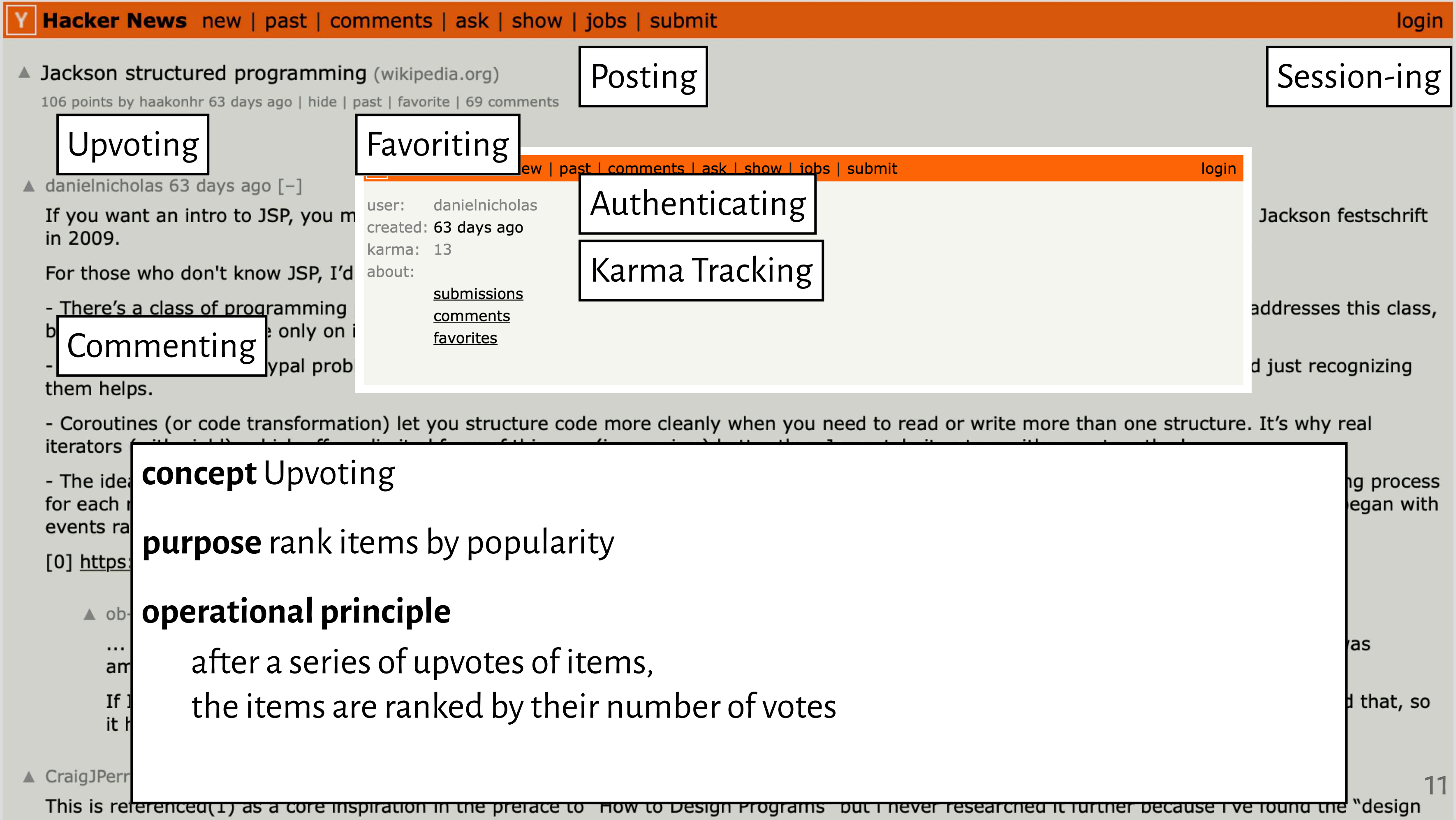
Commenting

concept Commenting

purpose users can comment in reply to other items

operational principle

after making a comment on an item,
when a user brings up that item,
the comment is also included



Y **Hacker News** new | past | comments | ask | show | jobs | submit login

▲ Jackson structured programming (wikipedia.org)

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments

▲ danielnicholas 63 days ago [–]

If you want an intro to JSP, you m … Jackson festschrift in 2009.

For those who don't know JSP, I'd

- There's a class of programming … addresses this class, b … only on i

- … ypal prob … d just recognizing them helps.

- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It's why real iterators

- The ide … ng process for each … egan with events ra

[0] https:

▲ ob-

…

am … as

If I … d that, so it h

▲ CraigJPerr

This is referenced(1) as a core inspiration in the preface to "How to Design Programs" but I never researched it further because I've found the "design

ew | past | comments | ask | show | jobs | submit login

user: danielnicholas

created: 63 days ago

karma: 13

about:

submissions

comments

favorites

Posting

Session-ing

Upvoting

Favoriting

Authenticating

Karma Tracking

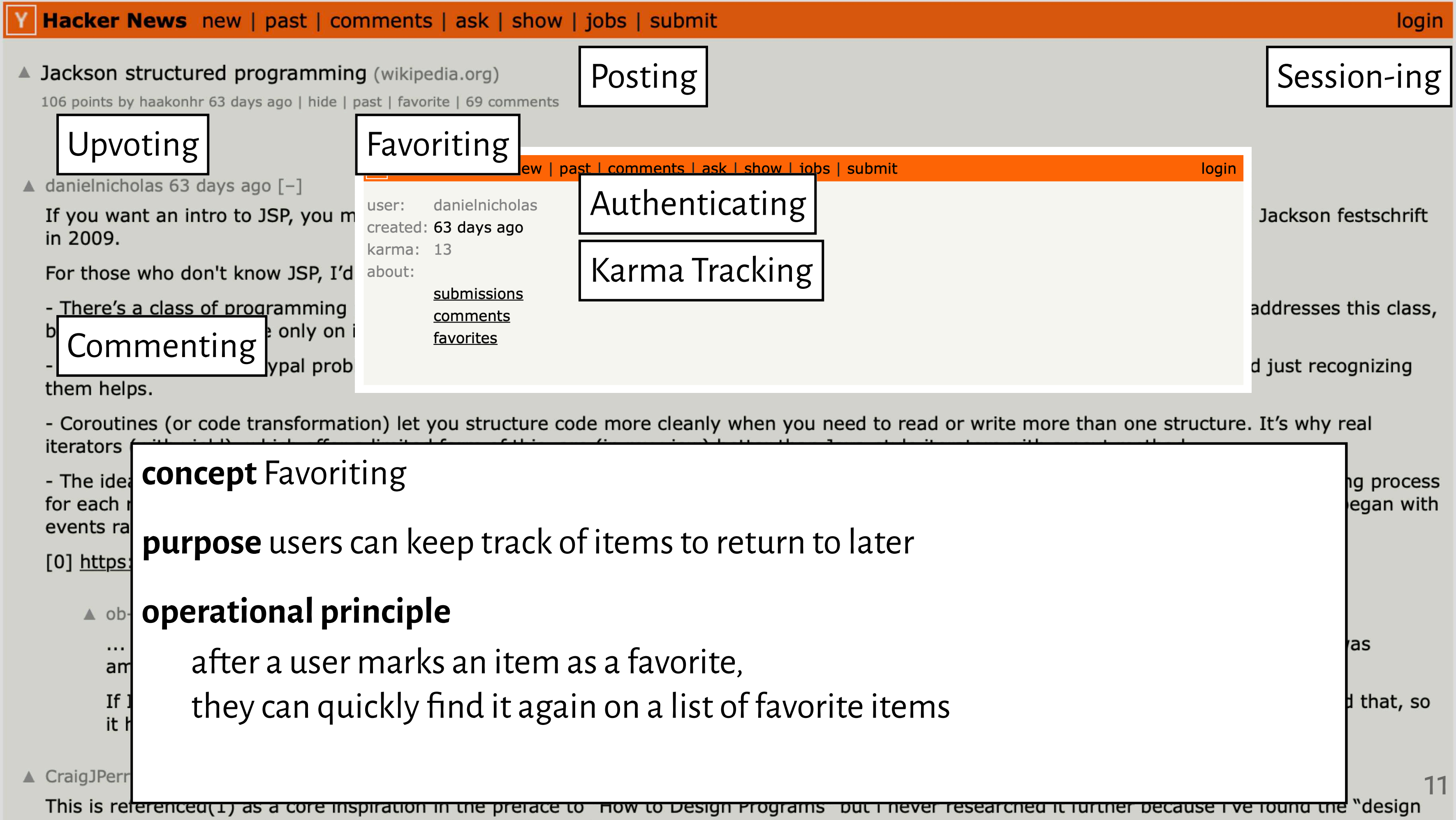
Commenting

concept Upvoting

purpose rank items by popularity

operational principle

after a series of upvotes of items,
the items are ranked by their number of votes



Y **Hacker News** new | past | comments | ask | show | jobs | submit login

▲ Jackson structured programming (wikipedia.org)

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments

▲ danielnicholas 63 days ago [–]

If you want an intro to JSP, you m Jackson festschrift in 2009.

For those who don't know JSP, I'd

- There's a class of programming addresses this class, b only on i

- ypal prob d just recognizing them helps.

- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It's why real iterators

- The idea ng process for each egan with events ra

[0] https:

▲ ob

...

am as

If I d that, so

it h

▲ CraigJPerr

This is referenced(1) as a core inspiration in the preface to "How to Design Programs" but I never researched it further because I've found the "design

ew | past | comments | ask | show | jobs | submit login

user: danielnicholas

created: 63 days ago

karma: 13

about:

submissions

comments

favorites

Posting

Session-ing

Upvoting

Favoriting

Authenticating

Karma Tracking

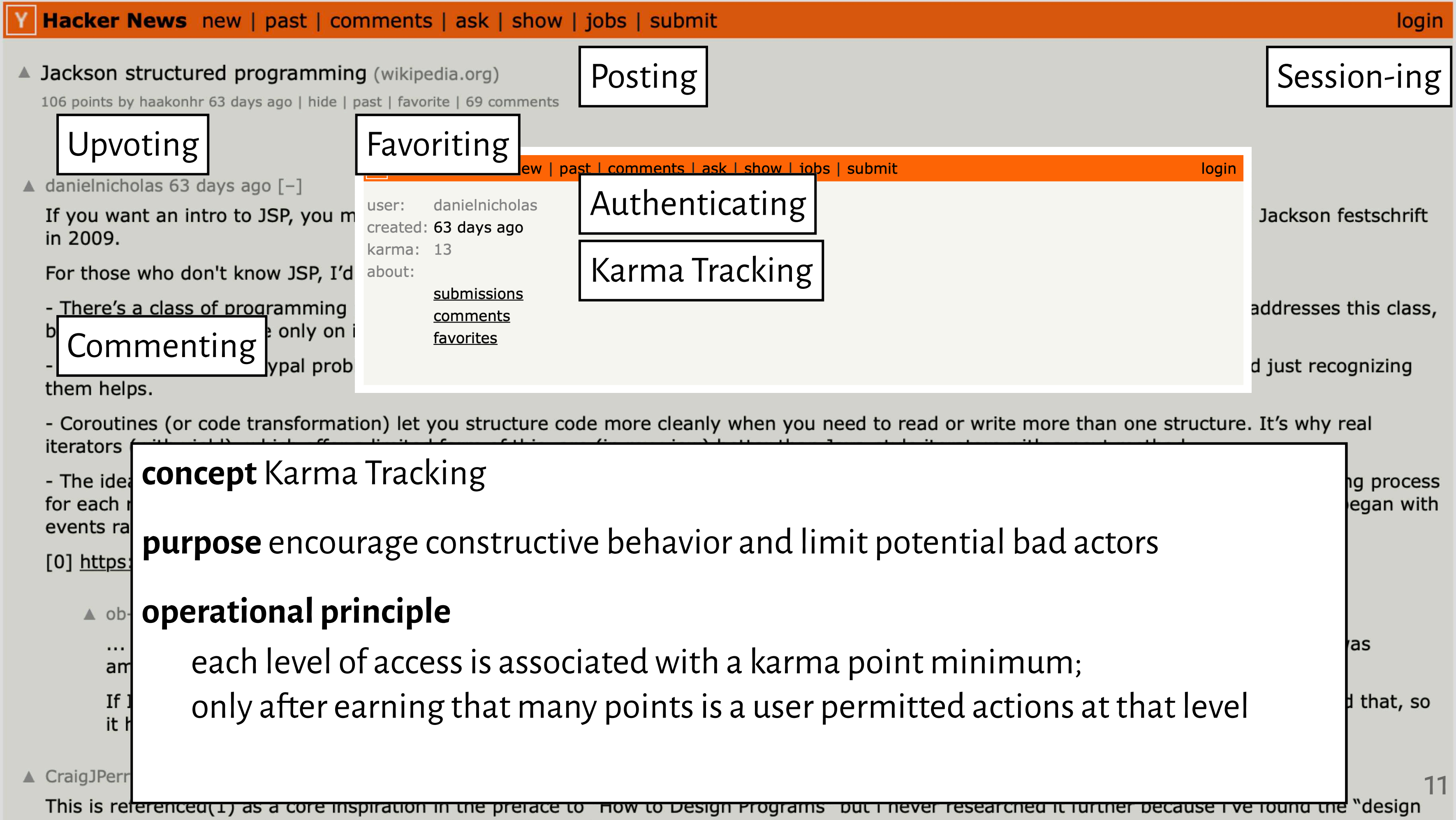
Commenting

concept Favoriting

purpose users can keep track of items to return to later

operational principle

after a user marks an item as a favorite,
they can quickly find it again on a list of favorite items



Hacker News new | past | comments | ask | show | jobs | submit login

▲ Jackson structured programming (wikipedia.org)

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments

▲ danielnicholas 63 days ago [–]

If you want an intro to JSP, you m Jackson festschrift in 2009.

For those who don't know JSP, I'd

- There's a class of programming addresses this class, b only on i

- ypal prob d just recognizing them helps.

- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It's why real iterators

- The ide ng process for each r egan with events ra

[0] https:

▲ ob

...

am as

If I d that, so it h

▲ CraigJPerr

This is referenced(1) as a core inspiration in the preface to "How to Design Programs" but I never researched it further because I've found the "design

ew | past | comments | ask | show | jobs | submit login

user: danielnicholas
created: 63 days ago
karma: 13
about:

submissions
comments
favorites

Posting

Session-ing

Upvoting

Favoriting

Authenticating

Karma Tracking

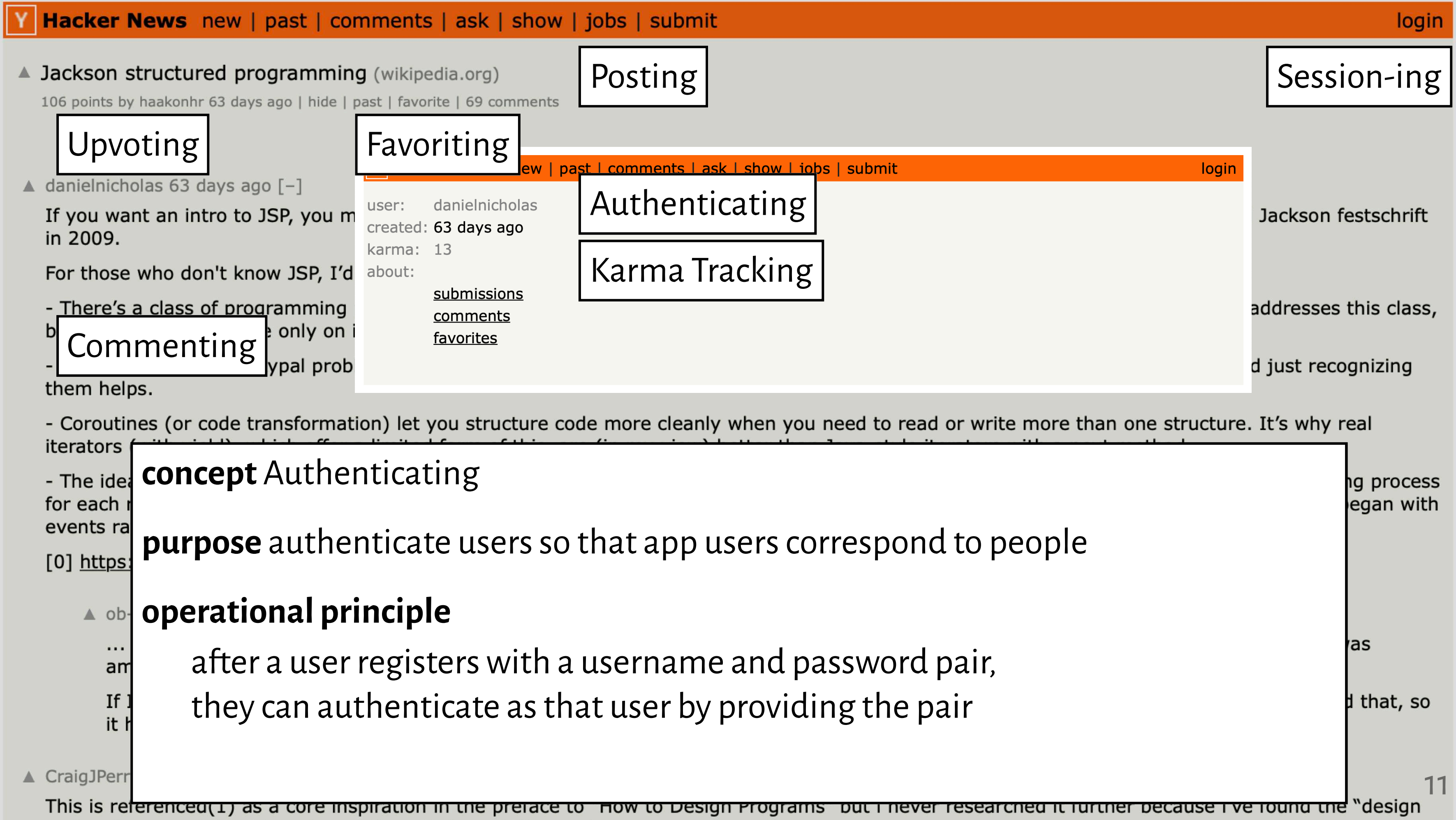
Commenting

concept Karma Tracking

purpose encourage constructive behavior and limit potential bad actors

operational principle

each level of access is associated with a karma point minimum;
only after earning that many points is a user permitted actions at that level



Hacker News new | past | comments | ask | show | jobs | submit login

▲ Jackson structured programming (wikipedia.org)

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments

▲ danielnicholas 63 days ago [–]

If you want an intro to JSP, you m … Jackson festschrift in 2009.

For those who don't know JSP, I'd

- There's a class of programming … addresses this class, b … e only on i

- … ypal prob … d just recognizing them helps.

- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It's why real iterators

- The ide … ng process for each … egan with events ra

[0] https:

▲ ob

...

am … as

If I … d that, so it h

▲ CraigJPerr

This is referenced(1) as a core inspiration in the preface to "How to Design Programs" but I never researched it further because I've found the "design

ew | past | comments | ask | show | jobs | submit login

user: danielnicholas
created: 63 days ago
karma: 13
about:

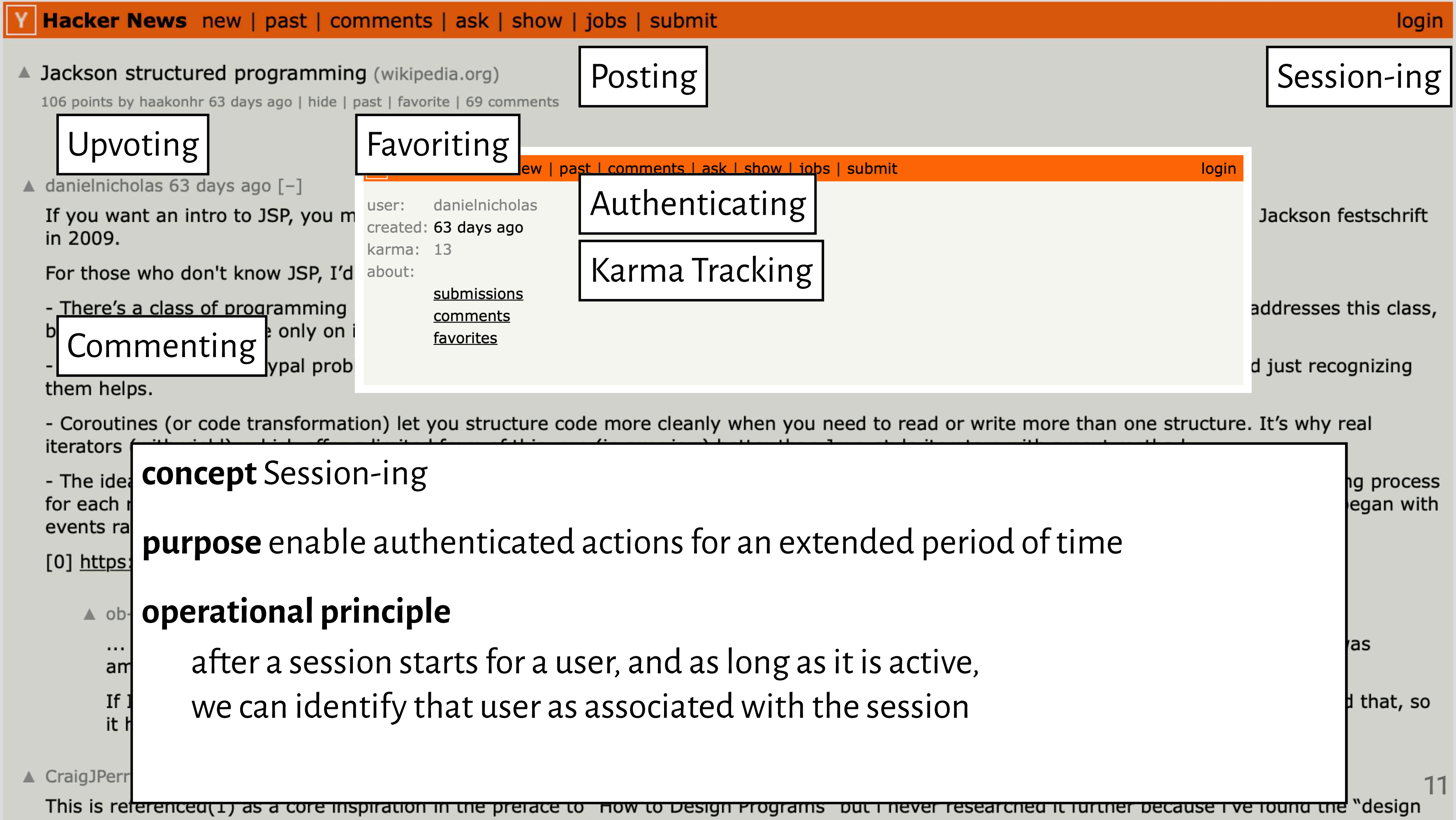
submissions
comments
favorites

concept Authenticating

purpose authenticate users so that app users correspond to people

operational principle

after a user registers with a username and password pair,
they can authenticate as that user by providing the pair



Y **Hacker News** new | past | comments | ask | show | jobs | submit login

▲ Jackson structured programming (wikipedia.org)

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments

▲ danielnicholas 63 days ago [–]

If you want an intro to JSP, you m … Jackson festschrift in 2009.

For those who don't know JSP, I'd

- There's a class of programming … addresses this class, b … only on i

- … ypal prob … d just recognizing them helps.

- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It's why real iterators

- The ide … ng process for each … egan with events ra

[0] https:

▲ ob-

…

am … as

If I … d that, so it h

▲ CraigJPerr

This is referenced(1) as a core inspiration in the preface to "How to Design Programs" but I never researched it further because I've found the "design

ew | past | comments | ask | show | jobs | submit login

user: danielnicholas

created: 63 days ago

karma: 13

about:

submissions

comments

favorites

Posting

Session-ing

Upvoting

Favoriting

Authenticating

Karma Tracking

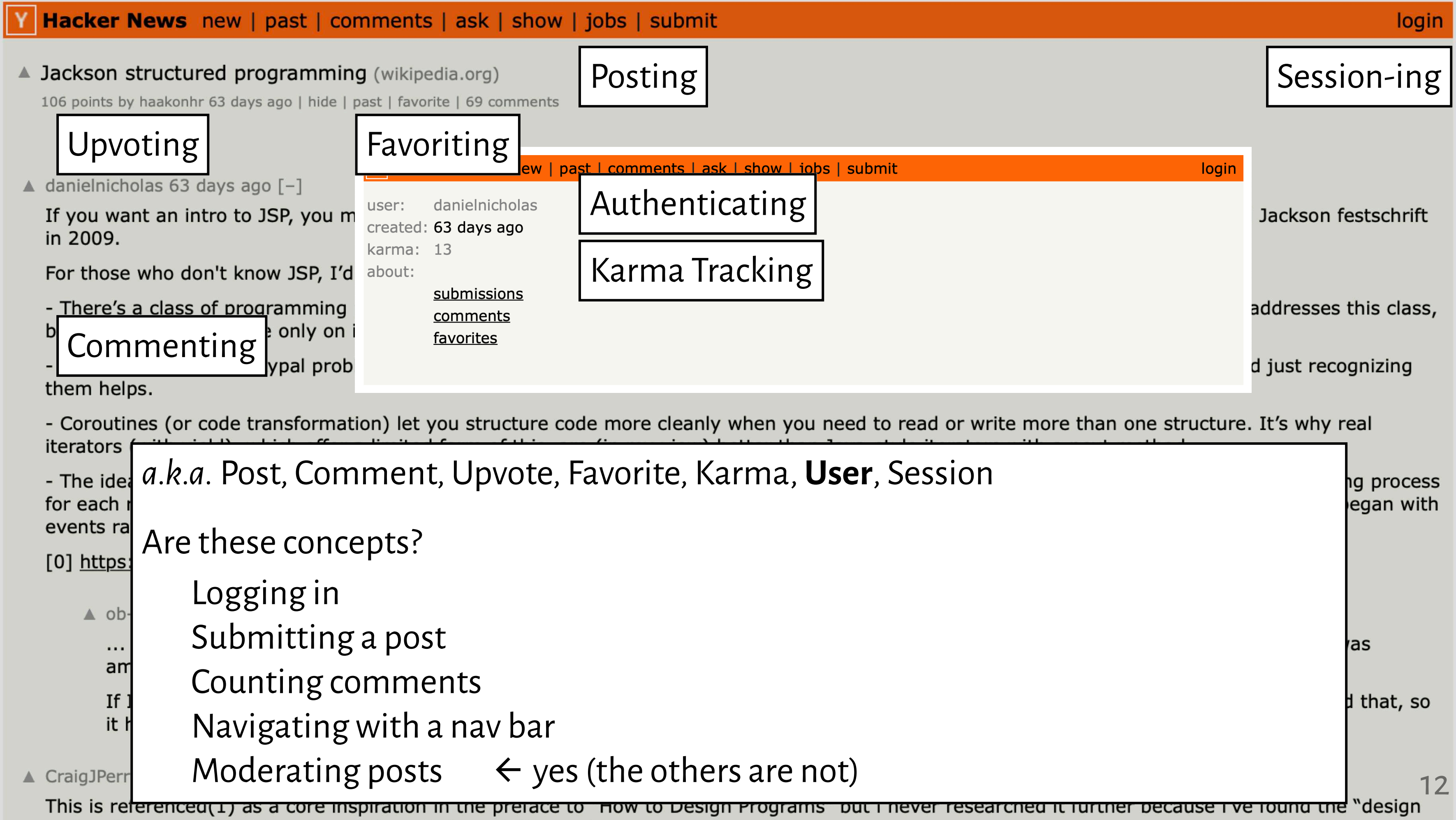
Commenting

concept Session-ing

purpose enable authenticated actions for an extended period of time

operational principle

after a session starts for a user, and as long as it is active,
we can identify that user as associated with the session



Hacker News new | past | comments | ask | show | jobs | submit login

▲ Jackson structured programming (wikipedia.org)

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comments

▲ danielnicholas 63 days ago [–]

If you want an intro to JSP, you m … Jackson festschrift in 2009.

For those who don't know JSP, I'd

- There's a class of programming … addresses this class, b … only on i

- … ypal prob … d just recognizing them helps.

- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It's why real iterators

- The idea … ng process for each … egan with events ra

[0] https:

▲ ob-

…
am … as

If I … that, so it h

▲ CraigJPerr

This is referenced(1) as a core inspiration in the preface to "How to Design Programs" but I never researched it further because I've found the "design

ew | past | comments | ask | show | jobs | submit login

user: danielnicholas
created: 63 days ago
karma: 13
about:

submissions
comments
favorites

Posting

Session-ing

Upvoting

Favoriting

Authenticating

Karma Tracking

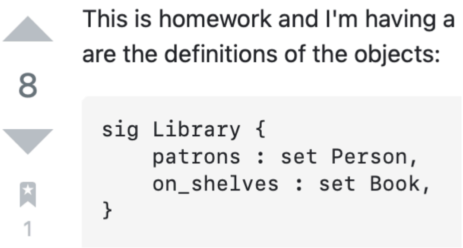
Commenting

a.k.a. Post, Comment, Upvote, Favorite, Karma, **User**, Session

Are these concepts?

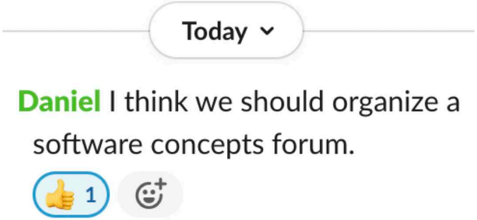
Logging in
Submitting a post
Counting comments
Navigating with a nav bar
Moderating posts ← yes (the others are not)

Similar user interfaces, different concepts

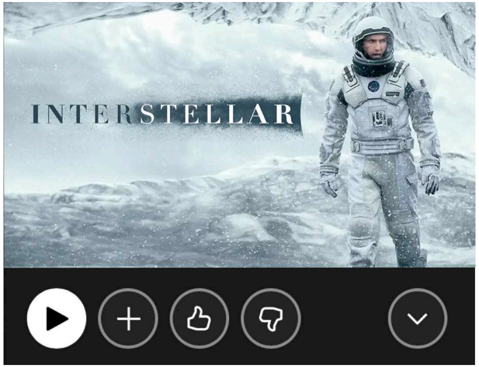


This is homework and I'm having a are the definitions of the objects:

```
sig Library {
     patrons : set Person,
     on_shelves : set Book,
}
```



Daniel I think we should organize a software concepts forum.



concept Upvoting
purpose rank by popularity
principle after a series of upvotes of items, the items are ranked by their number of votes

concept Reacting
purpose broadcast reactions to the author and others
principle when one user selects a reaction, it's shown to others (often aggregated)

concept Recommending
purpose use likes to suggest
principle user's likes lead to a ranking of kinds of items, determining which items are recommended

Choosing abstractions

A common pair: Authenticating + Session-ing

Authenticating *without* sessions?

Session-ing *without* authentication?

Can we build a compound concept of AuthenticatedSession-ing out of these two sub-concepts?

- We can! But let your default be one collection of concepts in an app, not a hierarchy

Intrinsic dependencies

```
class Post {
    readonly comments: Comment[];
}

class Post {
    public addComment(comment: Comment) {
        // ... TODO ...
    }
}

class Comment {
    readonly post: Post;
    readonly parent: Comment|undefined;
}
```

No intrinsic dependencies

Concepts are mutually independent

There are no intrinsic dependencies between concepts

posting.ts

```
class Post {
    readonly comments: Comment[]; // no!
}
```

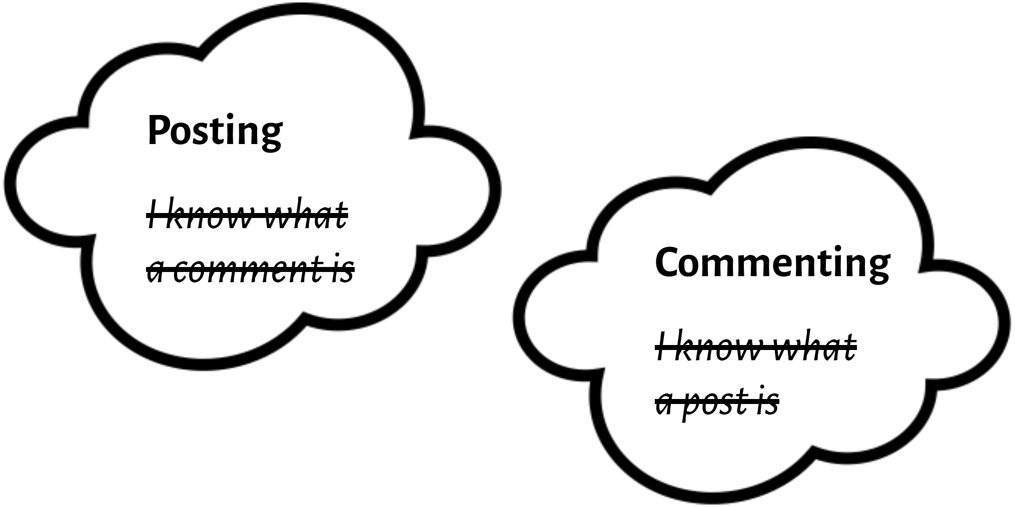
commenting.ts

```
class Comment {
    readonly post: Post; // no!
}
```

No intrinsic dependencies

Concepts are mutually independent

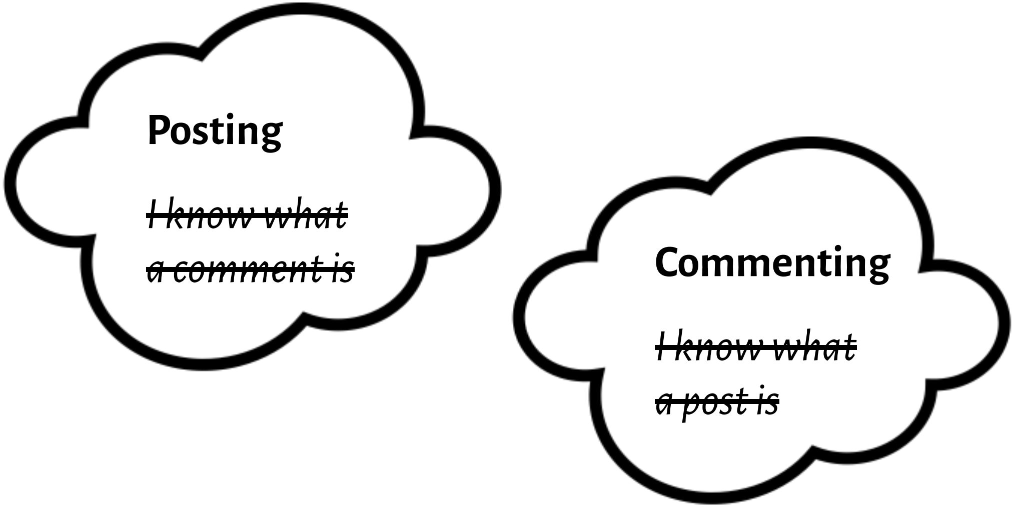
There are no intrinsic dependencies between concepts



No intrinsic dependencies

Concepts are mutually independent

There are no intrinsic dependencies between concepts



Posting

~~I know what a comment is~~

Commenting

~~I know what a post is~~

concept Posting

purpose users can post items for other users

operational principle

after making a post,
that post is available to other users

concept Commenting

purpose users can comment in reply to other items

operational principle

after making a comment on an item,
when a user brings up that item,
the comment is also included

No intrinsic dependencies

Concepts are mutually independent

There are no intrinsic dependencies between concepts

(1) Polymorphic parameters allow references to data from other concepts without creating a dependency

Posting

Posting is life!

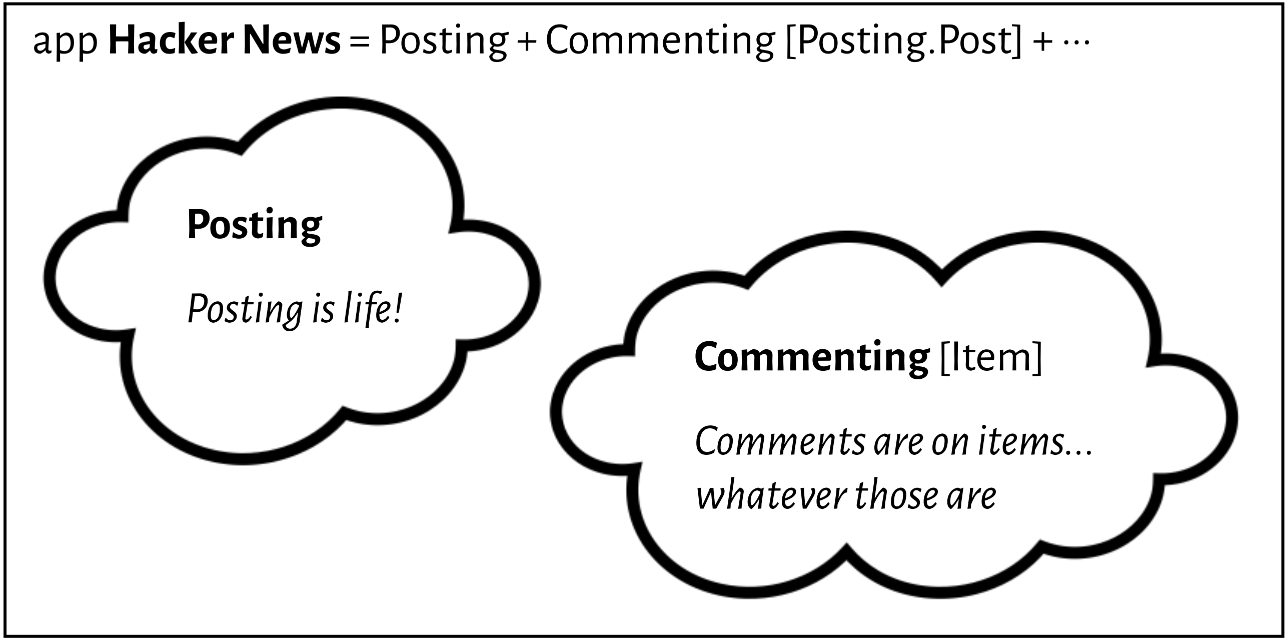
Commenting [Item]

Comments are on items…
whatever those are

(2) Discuss next time: the app coordinates concepts with *synchronizations*

Extrinsic dependencies

Dependencies in the context of the application



app **Hacker News** = Posting + Commenting [Posting.Post] + ⋯

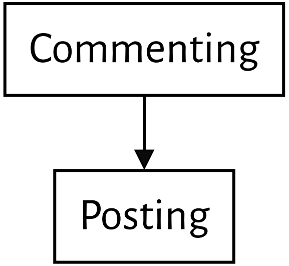
Posting

Posting is life!

Commenting [Item]

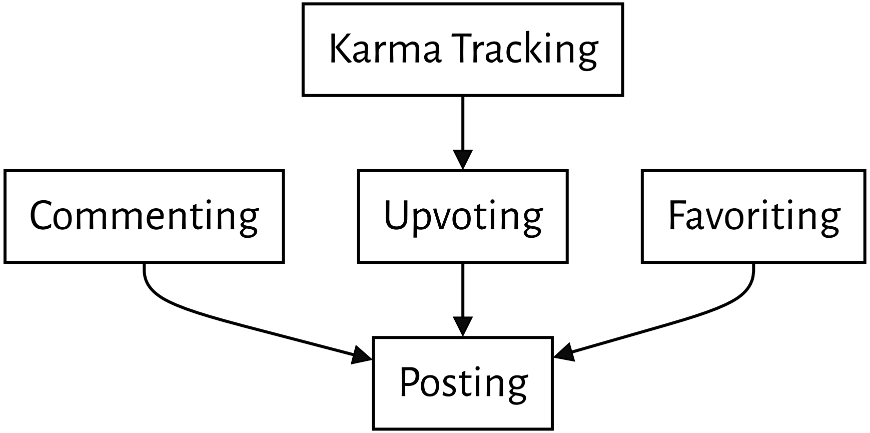
Comments are on items… whatever those are

In this app, Commenting only makes sense if we also have Posting



Extrinsic dependencies

Concept dependency graph for Hacker News

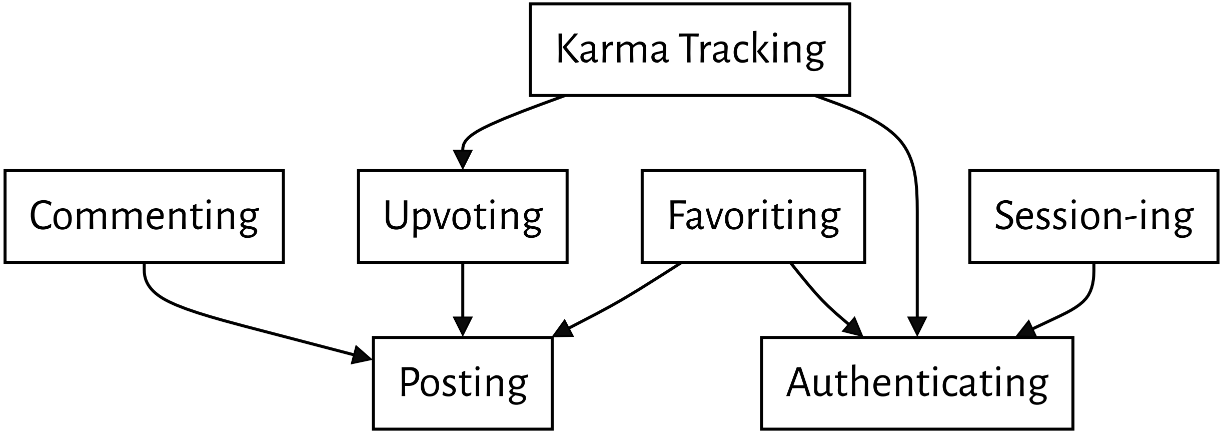


A → B means: *an app that includes A must also include B*

And now we can talk about **subsets** of the application that makes sense

Extrinsic dependencies

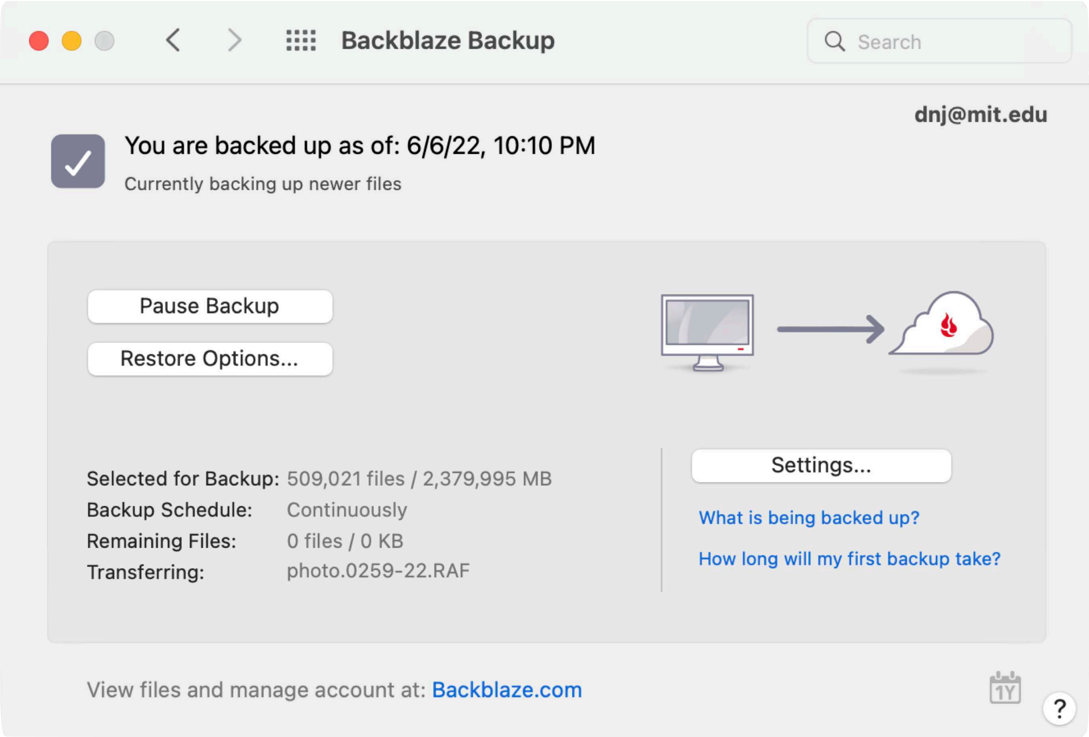
Concept dependency graph for Hacker News, with users and sessions



A → B means: *an app that includes A must also include B*

(do Upvoting, Commenting, and Posting also depend on Authenticating? maybe!)

Conceptual models



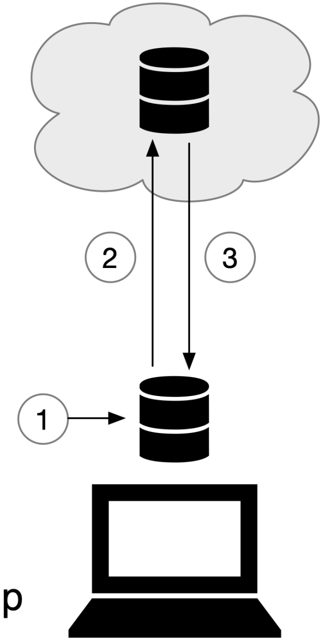
☑ You are backed up as of: 6/6/22, 10:10 PM

Currently backing up newer files

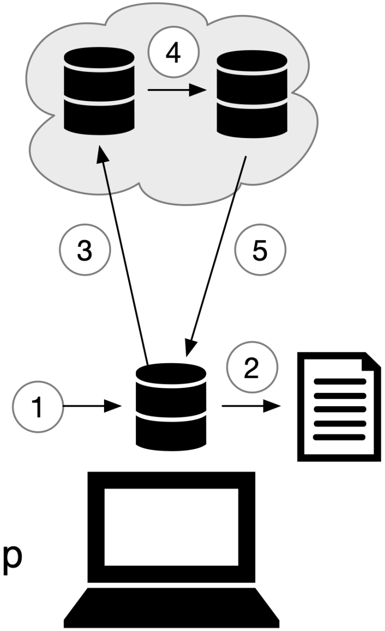
Selected for Backup:	509,021 files / 2,379,995 MB
Backup Schedule:	Continuously
Remaining Files:	0 files / 0 KB
Transferring:	photo.0259-22.RAF

View files and manage account at: Backblaze.com

Conceptual models

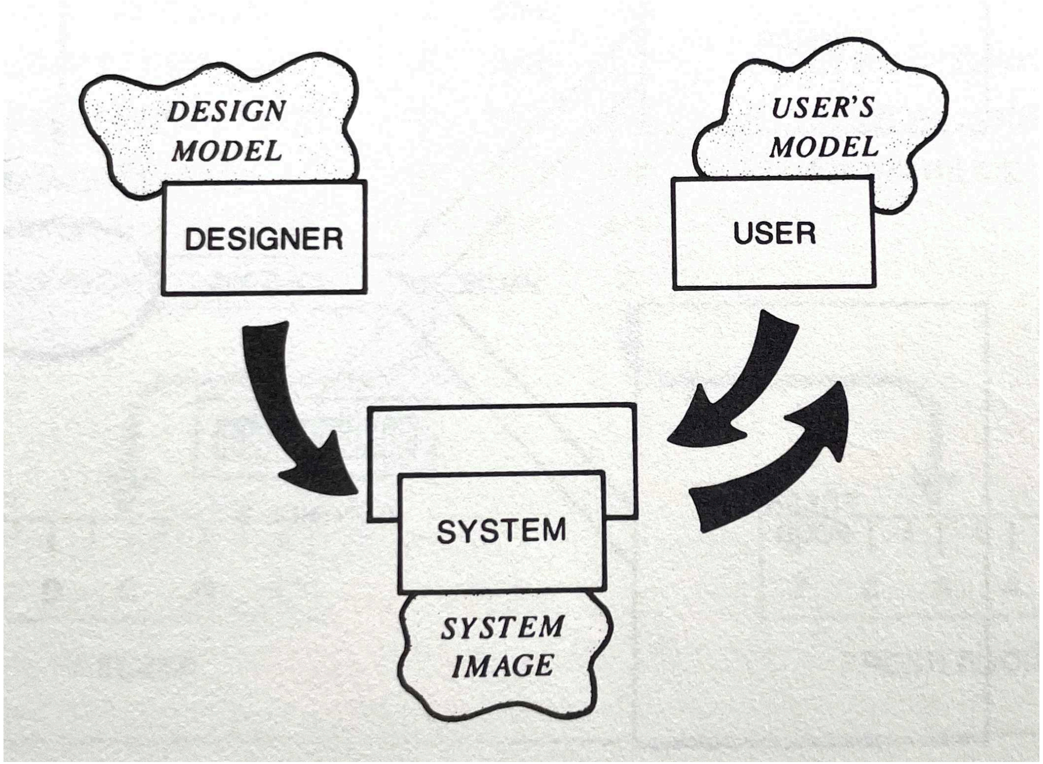


1. file modified
2. file backed up
3. file restored

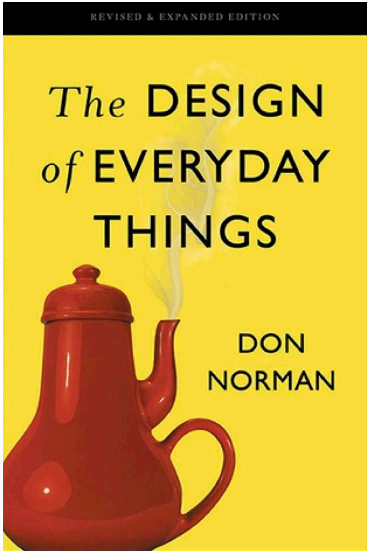


1. file modified
2. list created
3. files backed up
4. files available
5. file restored

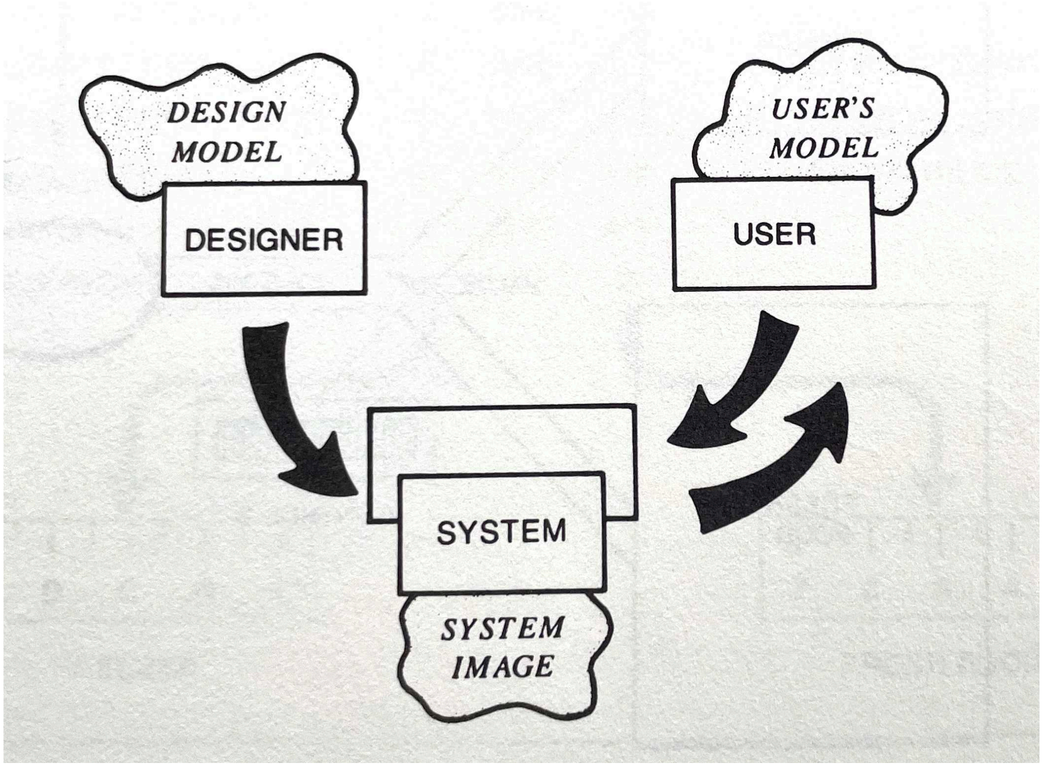
Conceptual models

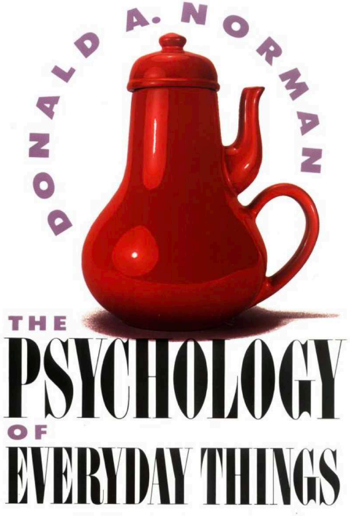


From *The Design of Everyday Things*



Conceptual models





From *The Design of Everyday Things*

Conceptual models

User-centered design

(as developed starting in the 1980s, perhaps being a bit unfair...)
Concepts are a byproduct of design
Designer's job: shape the user interface to project concepts
Concepts are psychological

Concept-based design

(what you are practicing in this class!)
Concepts are the essence of design
Designer's job: shape the concepts
Concepts are computational

DONALD A. NORMAN
THE PSYCHOLOGY OF EVERYDAY THINGS
THE ESSENCE OF SOFTWARE
WHY CONCEPTS MATTER FOR GREAT DESIGN
DANIEL JACKSON

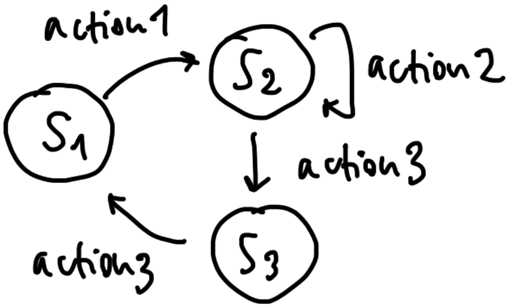
State machines

🛒 Shopping Carts

purpose users can gather a collection of items to buy in one transaction

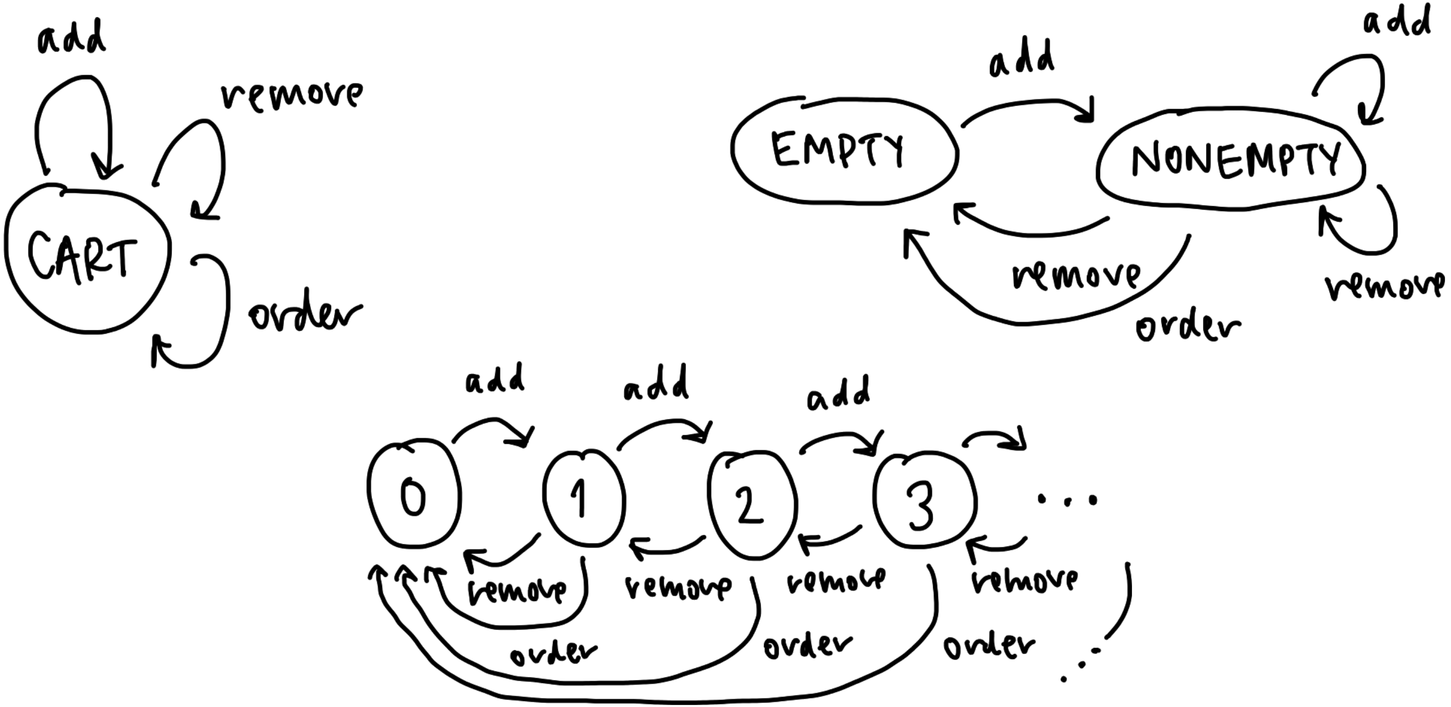
operational principle after adding items to the cart (and potentially removing some), the user places an order for those items, and the cart becomes empty

Let's start by thinking about **one** shopping cart
Try drawing a diagram to explain the operation of the **one** shopping cart
Use labeled nodes for states and labeled edges for actions



State machines

🛒 One shopping cart



Relational state

🛒 One shopping cart

What state do we need to store?

📄 cart.ts

```ts
// RI: values are integers
items: Map<Item, number>;
```

```ts
// RI: keys of items are item IDs, values are integers
items: Map<string, number>;
```

```ts
// AF: cart where count of x is # of times x appears in items
// RI: elements of items are item IDs
items: Array<string>;
```

This is not our goal right now!

Relational state

🛒 One shopping cart

What state do we need to store?

A set of pairs, *e.g.*:

{ (LEGO set #31208, 1),
(Blueberry muffin Lärabar, 42),
(*The Essence of Software*, 5) }

A binary relation from items to integers,

count: Item → Integer

Invariants

🛒 One shopping cart

Refining the state specification

More specific about the types...

Item → { i: Integer | i > 0 } *// only positive counts*

More specific about the multiplicities...

Item → Integer *// ... means...*
Item → **set** Integer *// items have zero or more counts, why is that not helpful for the shopping cart?*

Item → **one** Integer *// items have exactly one count, where could this be useful?*

Item → **opt** Integer *// items have zero or one count* ✓

Concepts as state machines

Let your default be that concepts manage sets of things, *e.g.*:

Posting: Posts
Commenting: Comments
Upvoting: Upvotes
Authenticating: Username + password pairs

🛒 Shopping Carts

purpose users can gather a collection of items to buy in one transaction
operational principle after adding items to the cart (and potentially removing some), the user places an order for those items, and the cart becomes empty
actions

add (cart: Cart, item: Item)
remove (cart: Cart, item: Item)
order (cart: Cart)

Concepts as state machines

🛒 Shopping Carts

actions

add (cart: Cart, item: Item)
remove (cart: Cart, item: Item)
order (cart: Cart)

state

What state do I need? As a ternary relation:

added: Cart → Item → **one** Integer

But let your default be binary relations:

added: Cart → LineItem
item: LineItem → **one** Item
count: LineItem → **one** Integer

Once we explain how the add, remove, and order actions update this state, we will have a complete* understanding of Shopping Carts

* except for all the stuff we haven't thought about yet like where do new carts come from? and does every user have a cart? or maybe more than one cart? and maybe you can put things in a cart before you even log in? and then what happens when you do log in? and...

Today

Concepts for structuring functionality

- Semantic: about underlying behavior users experience
- Purposive: fulfill an entire user need
- Modular: mutually independent

Dependencies

No intrinsic dependencies between concepts

State machines and **relational state**

Looking ahead

Specify concepts as state machines with relational state
Specify apps that combine several concepts and synchronize their behavior
Use the framework of concepts and synchronizations to move around the design space