designing concepts

Daniel Jackson · Autodesk, Boston · March 17-18, 2025

process for designing a concept

pick a name
specific to function
but for general use

describe purpose
why design or use it?
value to stakeholders

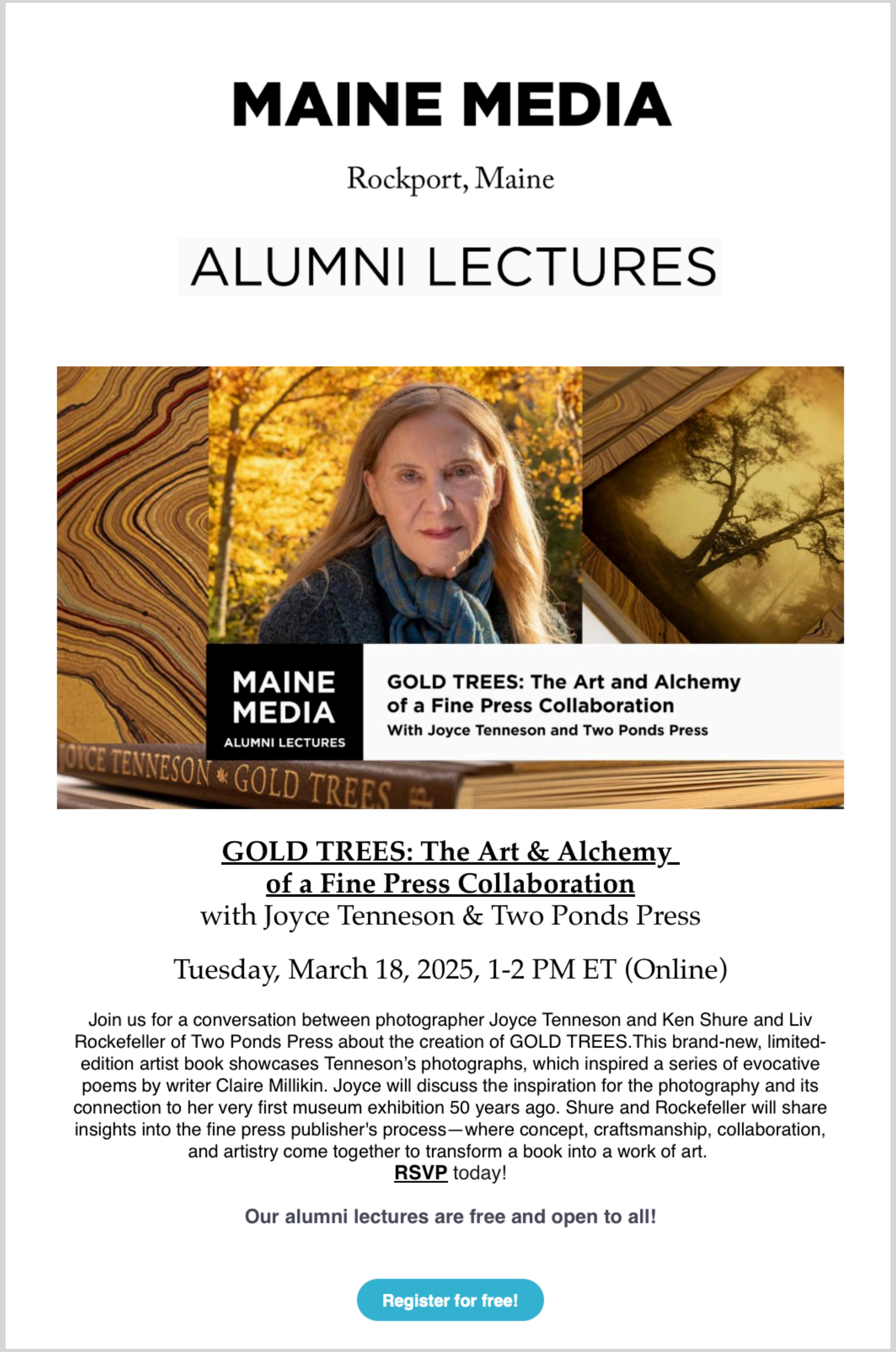
tell story
a simple scenario
of how it's used
including setup

list actions
by user or system
key steps, not UI

specify state
what's remembered
enough for actions

example:
EventBrite

event is announced



MAINE MEDIA

Rockport, Maine

ALUMNI LECTURES

MAINE MEDIA
ALUMNI LECTURES
GOLD TREES: The Art and Alchemy of a Fine Press Collaboration
With Joyce Tenneson and Two Ponds Press
JOYCE TENNESON * GOLD TREES

GOLD TREES: The Art & Alchemy of a Fine Press Collaboration
with Joyce Tenneson & Two Ponds Press

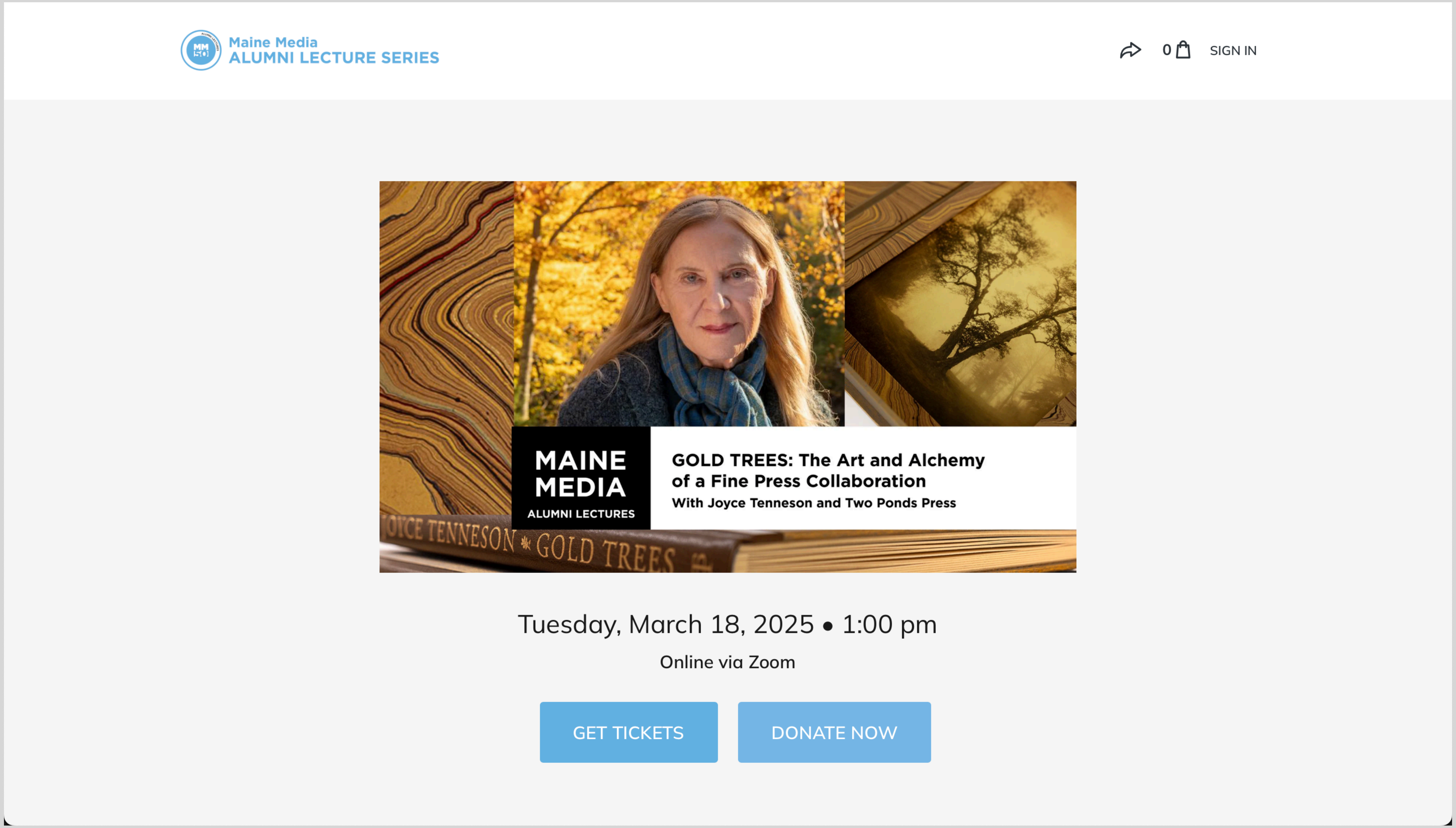
Tuesday, March 18, 2025, 1-2 PM ET (Online)

Join us for a conversation between photographer Joyce Tenneson and Ken Shure and Liv Rockefeller of Two Ponds Press about the creation of GOLD TREES.This brand-new, limited-edition artist book showcases Tenneson's photographs, which inspired a series of evocative poems by writer Claire Millikin. Joyce will discuss the inspiration for the photography and its connection to her very first museum exhibition 50 years ago. Shure and Rockefeller will share insights into the fine press publisher's process—where concept, craftsmanship, collaboration, and artistry come together to transform a book into a work of art.
RSVP today!

Our alumni lectures are free and open to all!

Register for free!

opening link to website



Maine Media
ALUMNI LECTURE SERIES

0 SIGN IN

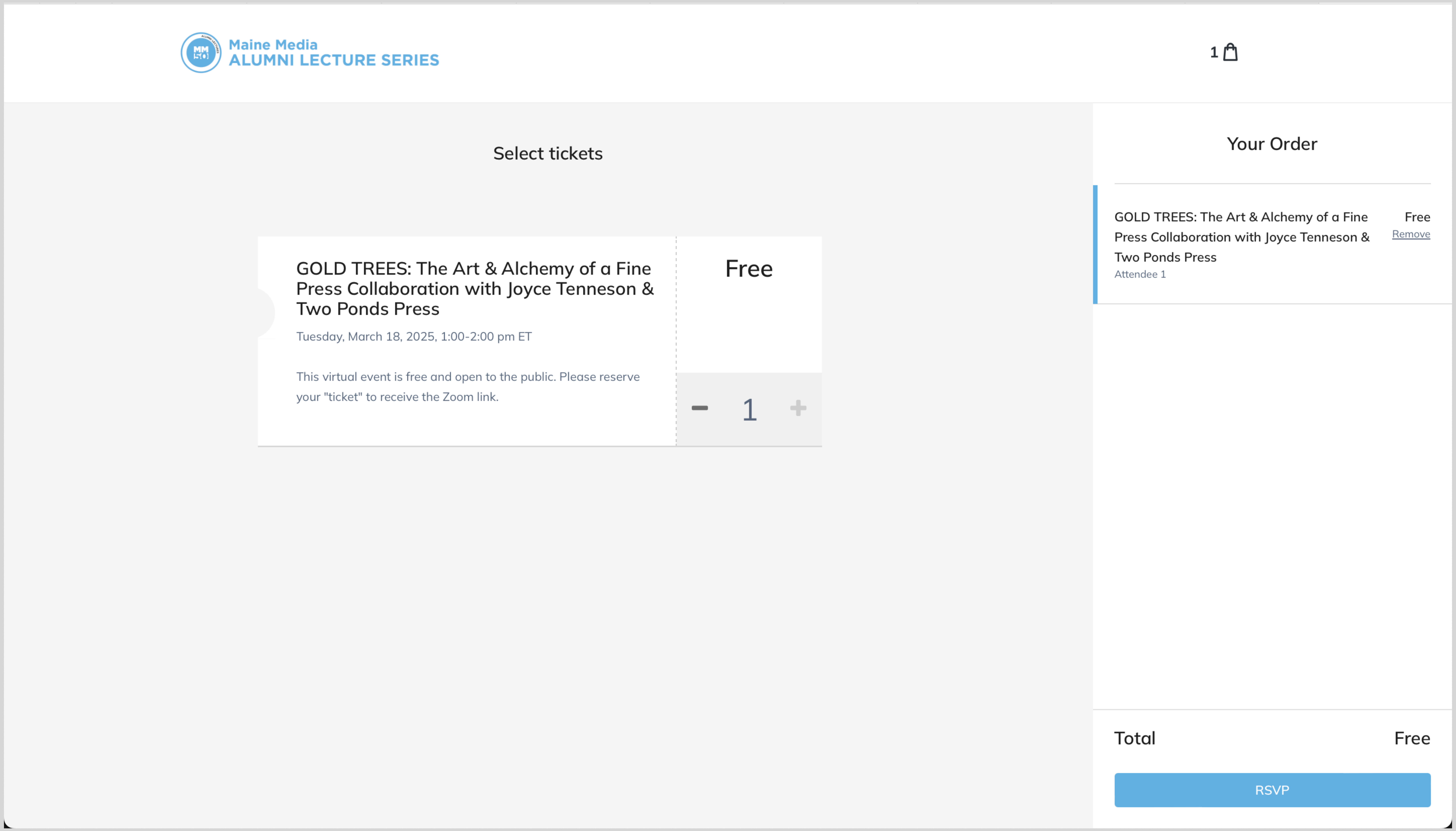
MAINE MEDIA
ALUMNI LECTURES

GOLD TREES: The Art and Alchemy of a Fine Press Collaboration
With Joyce Tenneson and Two Ponds Press

Tuesday, March 18, 2025 • 1:00 pm

Online via Zoom

selecting number of tickets


Maine Media
ALUMNI LECTURE SERIES

1

Select tickets

GOLD TREES: The Art & Alchemy of a Fine Press Collaboration with Joyce Tenneson & Two Ponds Press

Tuesday, March 18, 2025, 1:00-2:00 pm ET

This virtual event is free and open to the public. Please reserve your "ticket" to receive the Zoom link.

Free

− 1 +

Your Order

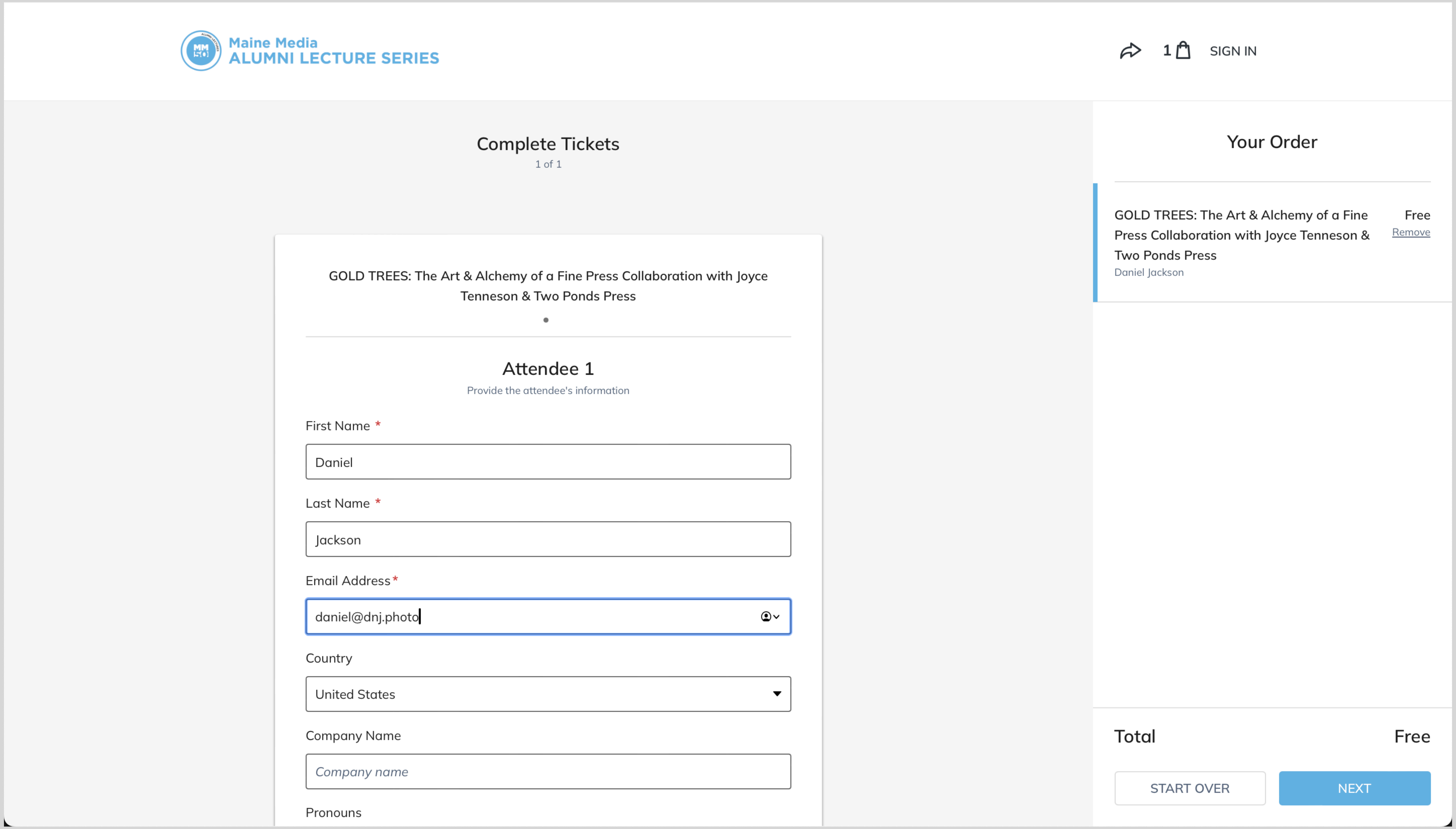
GOLD TREES: The Art & Alchemy of a Fine Press Collaboration with Joyce Tenneson & Two Ponds Press
Attendee 1

Free
Remove

Total Free

RSVP

entering name and email


Maine Media
ALUMNI LECTURE SERIES

1 SIGN IN

Complete Tickets
1 of 1

GOLD TREES: The Art & Alchemy of a Fine Press Collaboration with Joyce Tenneson & Two Ponds Press

Attendee 1
Provide the attendee's information

First Name *	Daniel
Last Name *	Jackson
Email Address*	daniel@dnj.photo
Country	United States
Company Name	Company name
Pronouns	

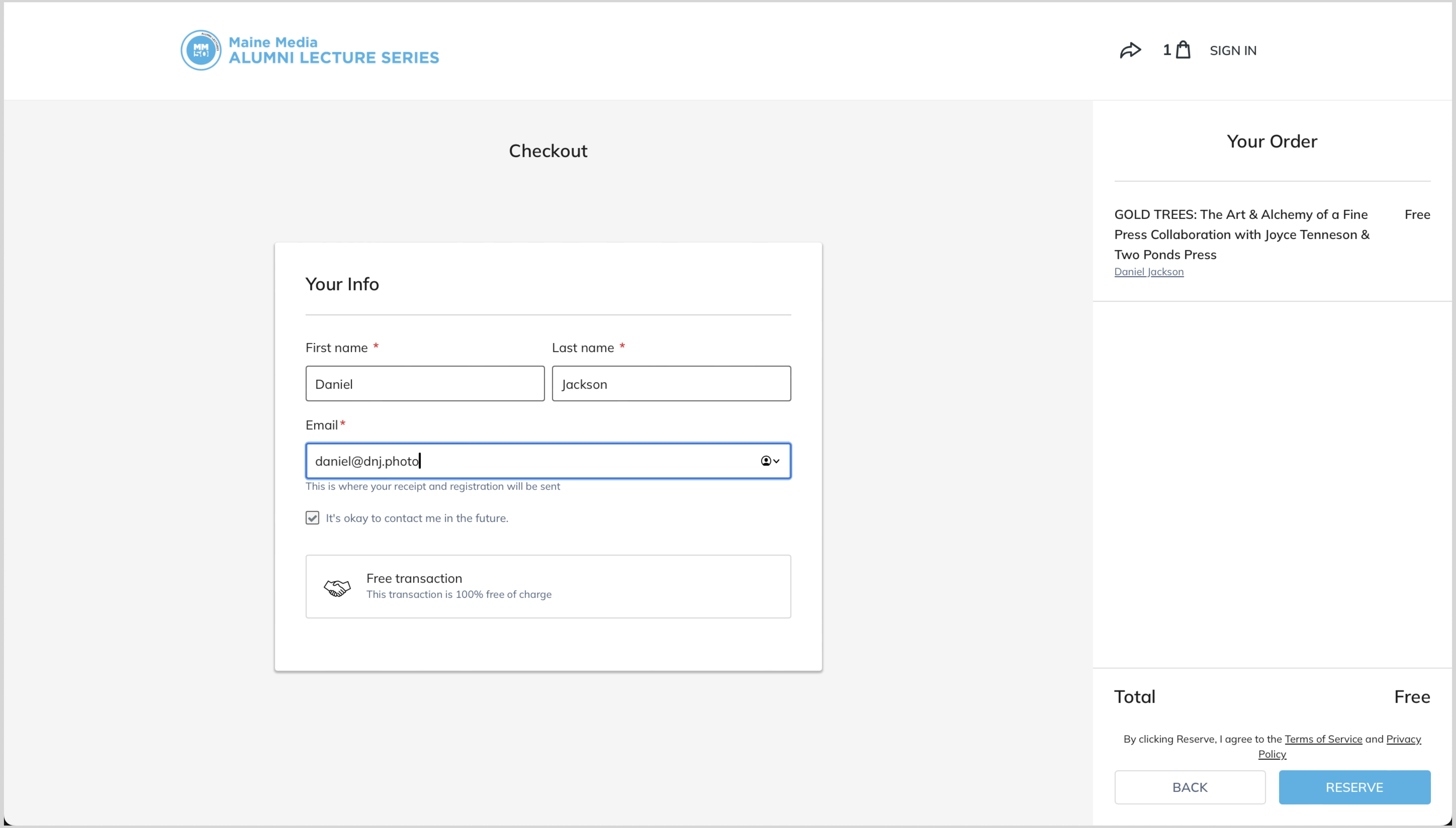
Your Order

GOLD TREES: The Art & Alchemy of a Fine Press Collaboration with Joyce Tenneson & Two Ponds Press — Free
Daniel Jackson
Remove

Total — Free

START OVER NEXT

entering name and email (again)



Maine Media
ALUMNI LECTURE SERIES

1 SIGN IN

Checkout

Your Info

First name *: Daniel

Last name *: Jackson

Email*: daniel@dnj.photo

This is where your receipt and registration will be sent

☑ It's okay to contact me in the future.

Free transaction
This transaction is 100% free of charge

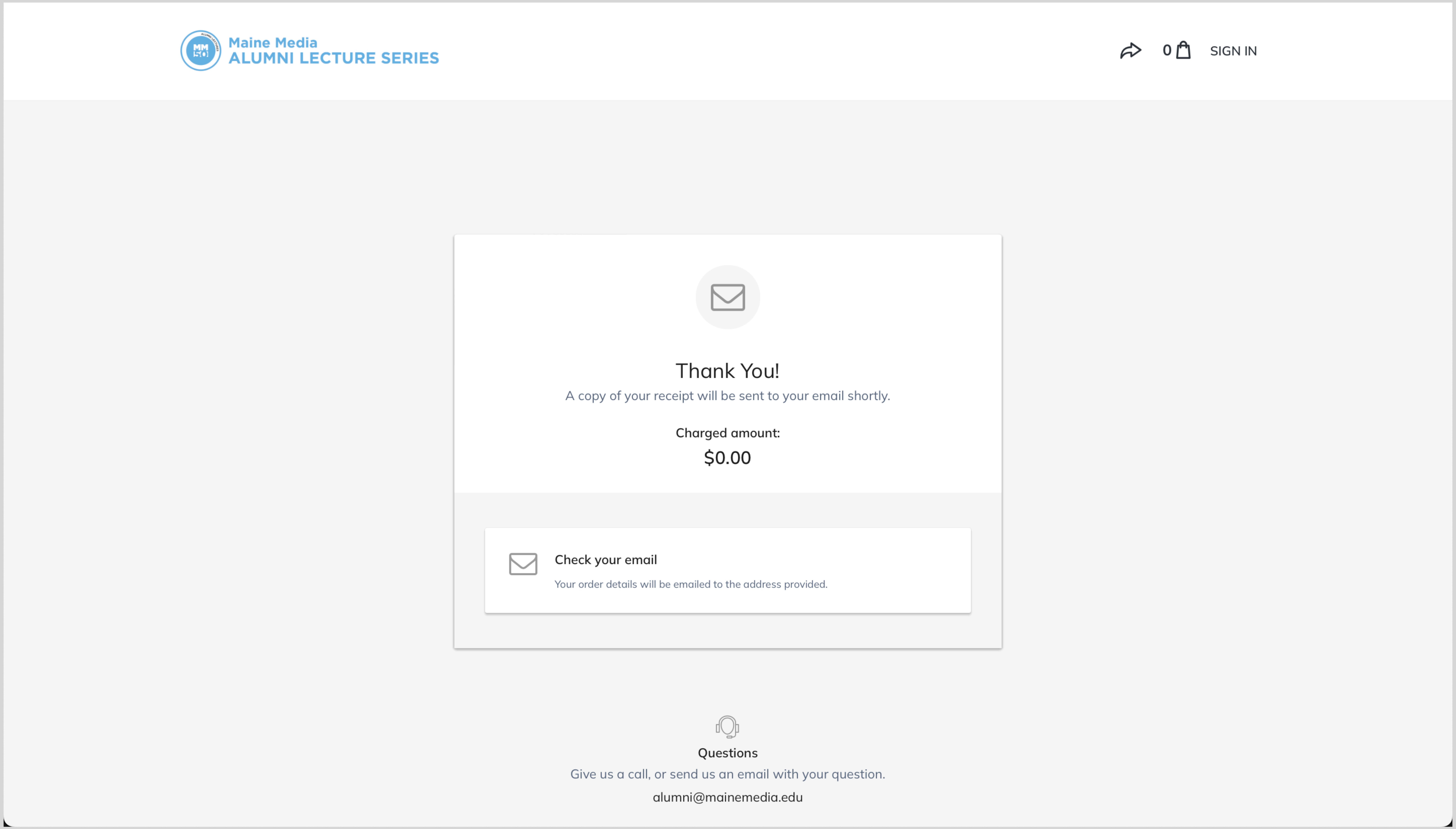
Your Order

GOLD TREES: The Art & Alchemy of a Fine Press Collaboration with Joyce Tenneson & Two Ponds Press Daniel Jackson	Free
Total	Free

By clicking Reserve, I agree to the Terms of Service and Privacy Policy

BACK RESERVE

success!



Thank You!

A copy of your receipt will be sent to your email shortly.

Charged amount:

$0.00

Check your email

Your order details will be emailed to the address provided.

Questions

Give us a call, or send us an email with your question.

alumni@mainemedia.edu

designing the core concept

process for designing a concept

pick a name
specific to function
but for general use

describe purpose
why design or use it?
value to stakeholders

tell story
a simple scenario
of how it's used
including setup

list actions
by user or system
key steps, not UI

specify state
what's remembered
enough for actions

picking a name

pick a name
specific to function
but for general use

Event

EventTicket ✓

EventTicketing

Ticket

describing a purpose

describe purpose
why design or use it?
value to stakeholders

organizing events

raising money for events

issuing tickets for events

managing event attendance

telling the story

an event organizer creates an event and announces it or invites people to it; they can then register, and the organizer can see who registered; eventually the people who registered can attend the event

tell story
a simple scenario of how it's used including setup

listing actions

getEmailInvitation
openInvitation
selectCount
enterAttendee
enterContact
clickReserve

no! these are
all low-level
UI interactions

registerForEvent

this one action
is enough to cover
the entire website
interaction!

but crucial actions are
missing: how did the
event appear in the first
place? what happens
after registration?

list actions
by user or system
key steps, not UI

listing actions

let's return to our story
for hints about the actions

an event organizer creates an event and announces it or invites people to it; they can then register, and the organizer can see who registered; eventually the people who registered can attend the event

create event

announce event

register for event

view registrations

attend event

list actions
by user or system
key steps, not UI

separation of concerns
always in back of mind:
does this belong to another concept? (eg: announce)

formalizing actions

create (by: User, on: Date): Event

register (e: Event, u: User)

attend (e: Event, u: User)

separation of concerns
always in back of mind:
does this belong to another concept? (eg: event details)

list actions
by user or system
key steps, not UI

specifying state

informally

a set of events
for each event
  a date/time
  an organizer
  a set of registrants

in a programming/spec notation

events: **set** Event
date: Event -> **one** Date
organizer: Event -> **one** User
registrants: Event -> **set** User

specify state
what's remembered
enough for actions

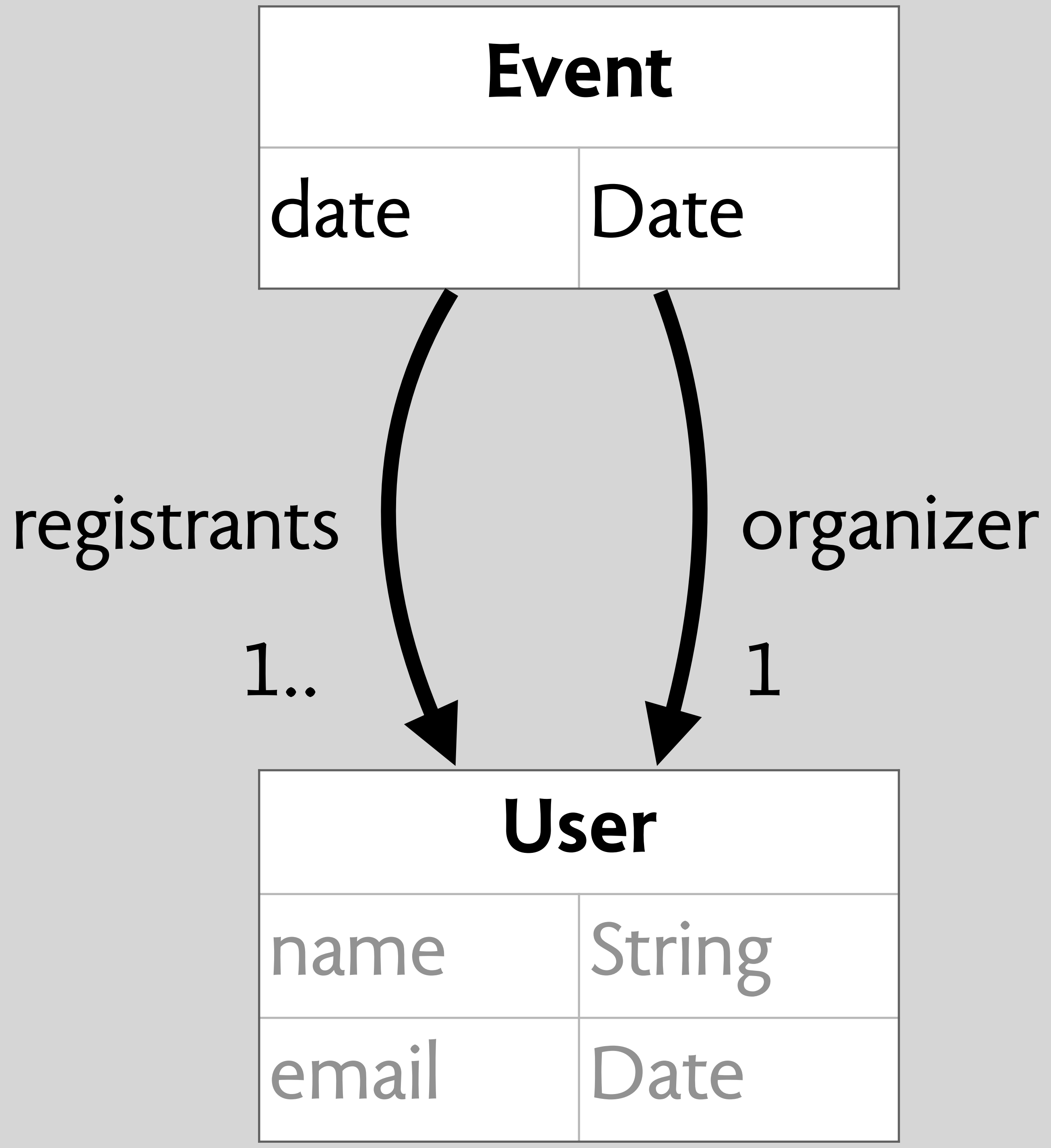
separation of concerns
always in back of mind:
does this belong to another concept? (eg: registrant name and email)

another way to define state

in a programming/spec notation

```
events: set Event
date: Event -> one Date
organizer: Event -> one User
registrants: Event -> set User
```

as a graphical data model



specify state
what's remembered
enough for actions

putting it all together

concept EventTicket [User]

User is a generic type

purpose managing event attendance

principle an event organizer creates an event (and announces it or invites people to it); they can then register, and the organizer can see who registered; eventually the people who registered can attend the event

state
  events: **set** Event
  date: Event -> **one** Date
  organizer: Event -> **one** User
  registrants: Event -> **set** User

actions
create (by: User, on: Date): Event
register (e: Event, u: User)
attend (e: Event, u: User)

specifying the actions

```
state
  events: set Event
  date: Event -> one Date
  organizer: Event -> one User
  registrants: Event -> set User
```

```
actions
```

```
create (by: User, on: Date): Event
// create a fresh event e not in events
// set e.date to on
// set e.organizer to by
// return e
```

```
register (e: Event, u: User)
// add u to e.registrants
```

```
attend (e: Event, u: User)
// ???
```

specifying the actions, take two

```
state
  events: set Event
  date: Event -> one Date
  organizer: Event -> one User
  registrants: Event -> set User
  attendees: Event -> set User
```

```
actions
```

```
create (by: User, on: Date): Event
// create a fresh event e not in events
// set e.date to on
// set e.organizer to by
// return e
```

```
register (e: Event, u: User)
// add u to e.registrants
```

```
attend (e: Event, u: User)
// add u to e.attendees
```

our final concept

concept EventTicket [User]

purpose managing event attendance

principle an event organizer creates an event (and announces it or invites people to it); they can then register, and the organizer can see who registered; eventually the people who registered can attend the event

state
  events: **set** Event
  date: Event -> **one** Date
  organizer: Event -> **one** User
  registrants: Event -> **set** User
  attendees: Event -> **set** User

actions
create (by: User, on: Date): Event
register (e: Event, u: User)
attend (e: Event, u: User)

completing the design

some supporting concepts

concept EventTicket [User]

purpose managing event attendance

principle an event organizer creates an event (and announces it or invites people to it); they can then register, and the organizer can see who registered; eventually the people who registered can attend the event

state
events: **set** Event
date: Event -> **one** Date
organizer: Event -> **one** User
registrants: Event -> **set** User
attendees: Event -> **set** User

actions
create (by: User, on: Date): Event
register (e: Event, u: User)
attend (e: Event, u: User)

concept UserProfile

purpose track user details

principle after a profile is created, you can find the user by email address

state
user: **set** User
first, last: User -> **one** String
email: User -> **one** Email

actions
create (fst, lst: String, e: Email): User
find (e: Email): User

concept EventCatalog

purpose share event descriptions

principle after an event is created, invitees can read about the details

state
events: **set** Event
title: Event -> **one** String
organizer: Event -> **one** User

actions
create (title: String, u: User): Event
get_details (e: Event): String

sample synchronizations

concept EventTicket [User]

actions
create (by: User, on: Date): Event
register (e: Event, u: User)
attend (e: Event, u: User)

concept UserProfile

actions
create (fst, lst: String, e: Email): User
find (e: Email): User

concept EventCatalog

actions
create (title: String, u: User): Event
get_details (e: Event): String

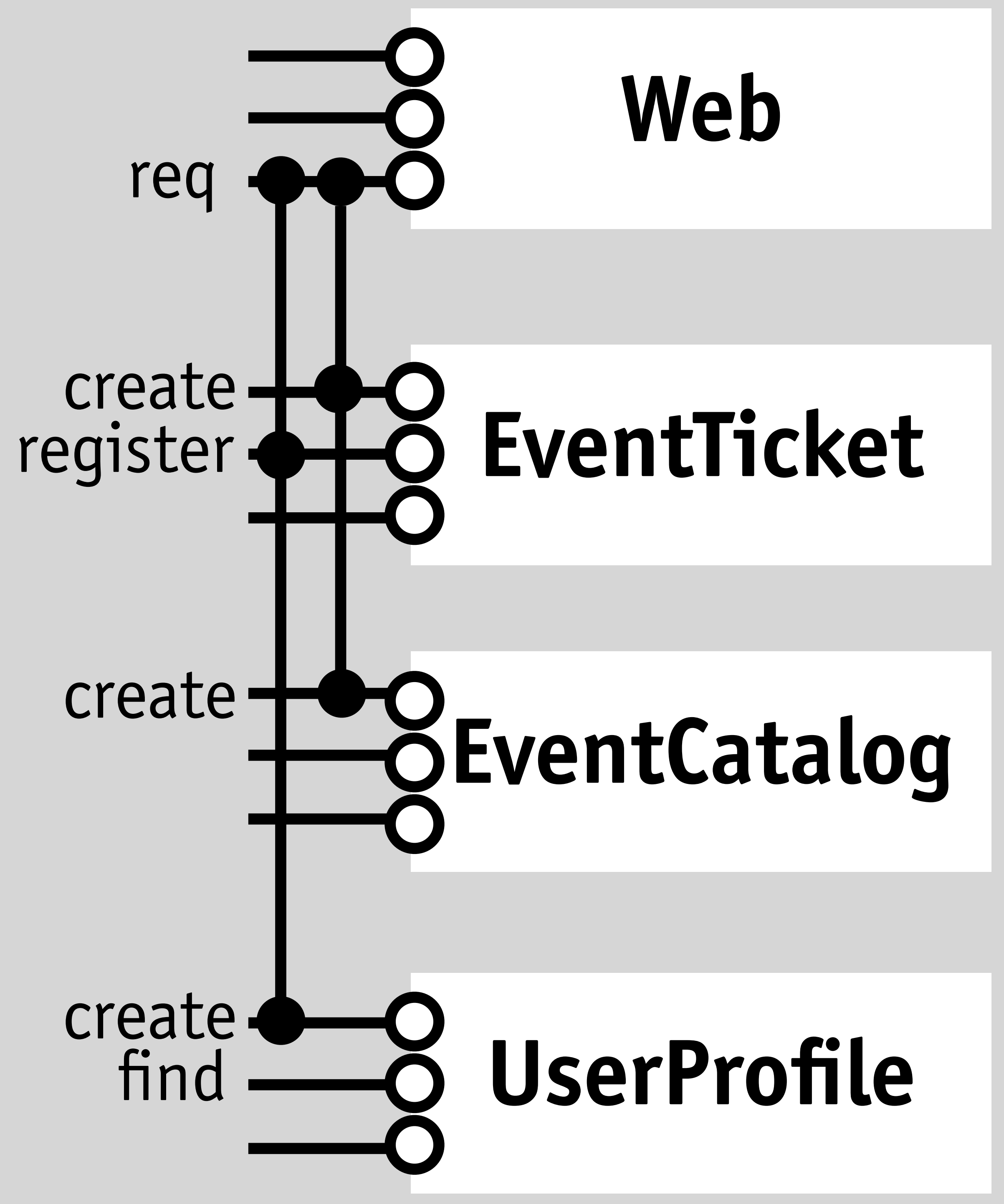
when Web.req (*create*, by, on, title) **then**
 ec = EventCatalog.create (title, by)
 et = EventTicket.create (by, on)
 et.catalog = ec

when Web.req (*view*, event) **then**
 ec = event.catalog
 s = EventCatalog.get_details (ec)
 Web.response (s)

when Web.req (*register*, email, fst, lst, event) **then**
 u = UserProfile.create (fst, lst, email)
 EventTicket.register (event, u)

when Web.req (*register*, email, event)
 u = UserProfile.find (email)
then EventTicket.register (event, u)

synchronizations



```
when Web.req (create, by, on, title) then
  ec = EventCatalog.create (title, by)
  et = EventTicket.create (by, on)
  et.catalog = ec
```

```
when Web.req (view, event) then
  ec = event.catalog
  s = EventCatalog.get_details (ec)
  Web.response (s)
```

```
when Web.req (register, email, fst, lst, event) then
  u = UserProfile.create (fst, lst, email)
  EventTicket.register (event,  u)
```

```
when Web.req (register, email, event)
  u = UserProfile.find (email)
then EventTicket.register (event,  u)
```

runtime coupling
but no design coupling
or code coupling

app-specific behaviors
often in syncs alone
so concepts stay pure

your turn:
design issues

pick some design issues, discuss & report back

registrant canceling their registration
registrant changing first name
registrant changing email address
organizer changing event time
organizer changing event description
organizer canceling event
limiting capacity for event
requiring payment for registration
notifying registrant of registration by email
reminding registrants of upcoming meeting
requiring ticket to be obtained after registering

function extensions

eve maliciously registers alice
eve maliciously cancels alice's registration
eve cancels event

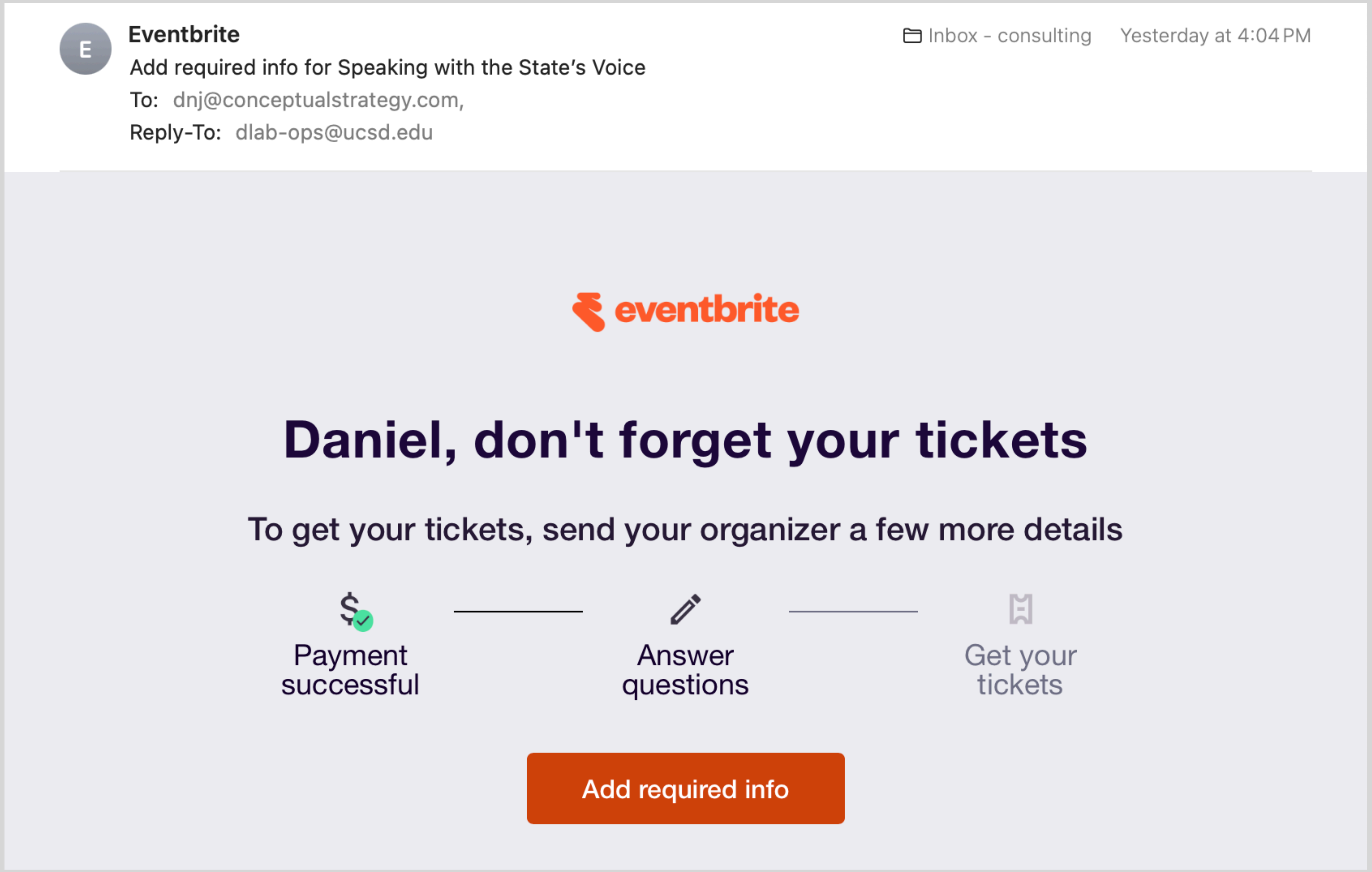
security threats

how might you adjust the design?
change existing concepts?
change existing syncs?
add concepts or syncs?

are there more consequences?
is this function desirable?
knock-on effects?
implications for the future?

separate tickets: good or bad?

what's going on here?



E

Eventbrite

Add required info for Speaking with the State's Voice

To: dnj@conceptualstrategy.com,

Reply-To: dlab-ops@ucsd.edu

Inbox - consulting

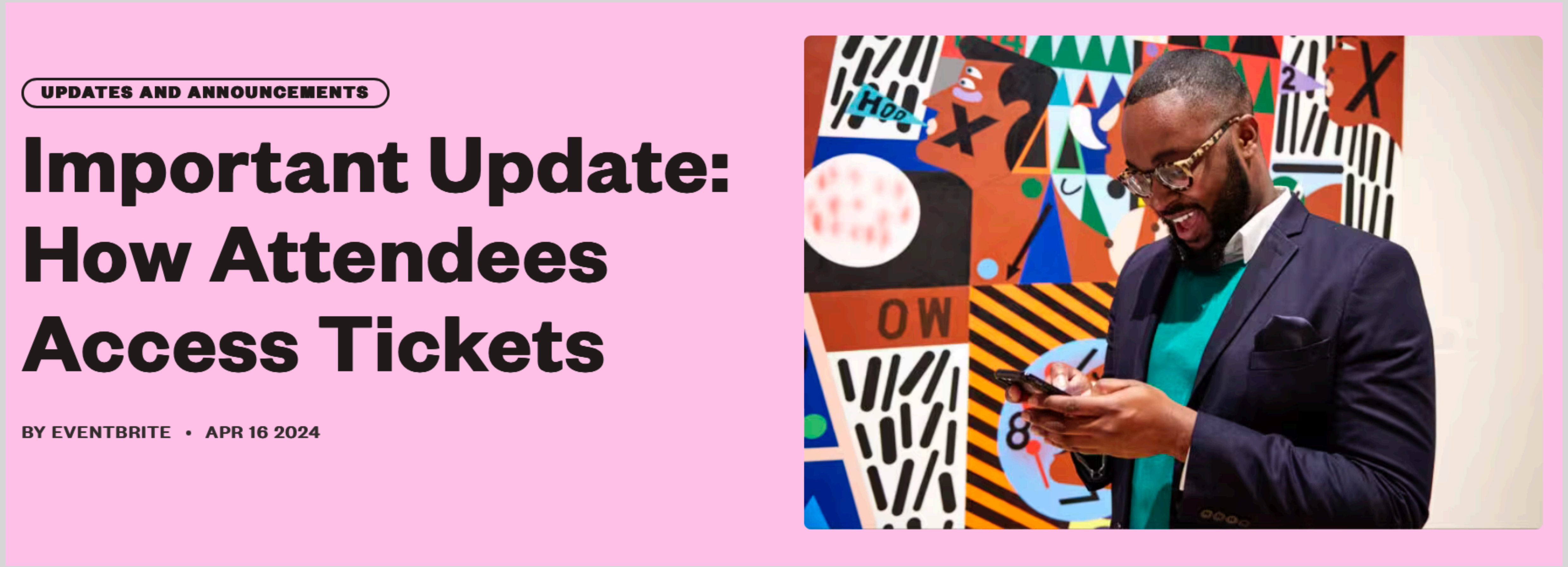
Yesterday at 4:04 PM

eventbrite

Daniel, don't forget your tickets

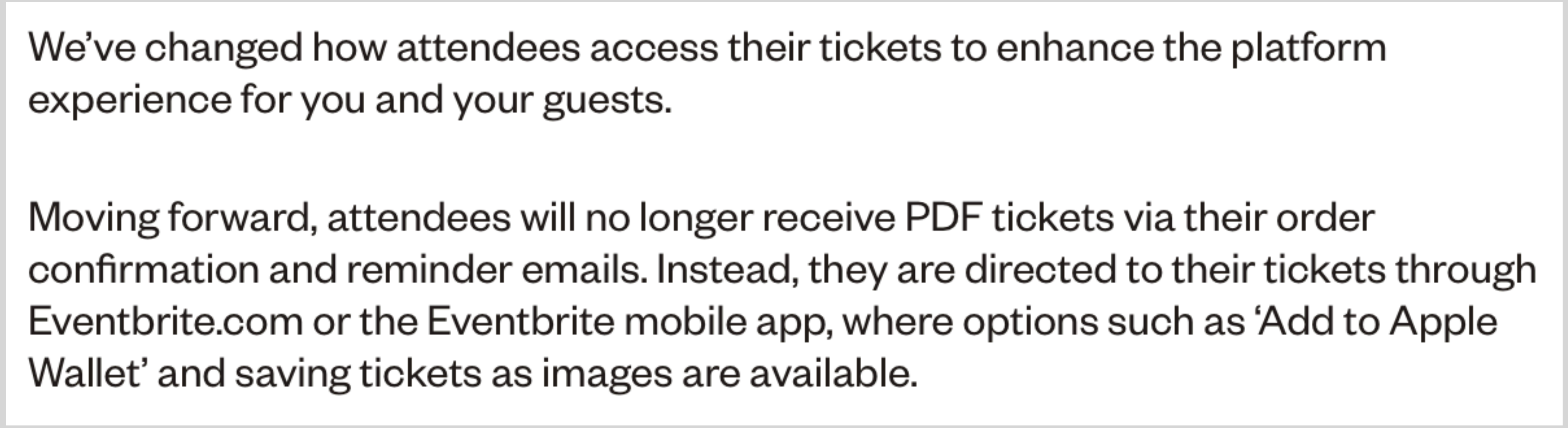
To get your tickets, send your organizer a few more details

is this honest, or a dark design?

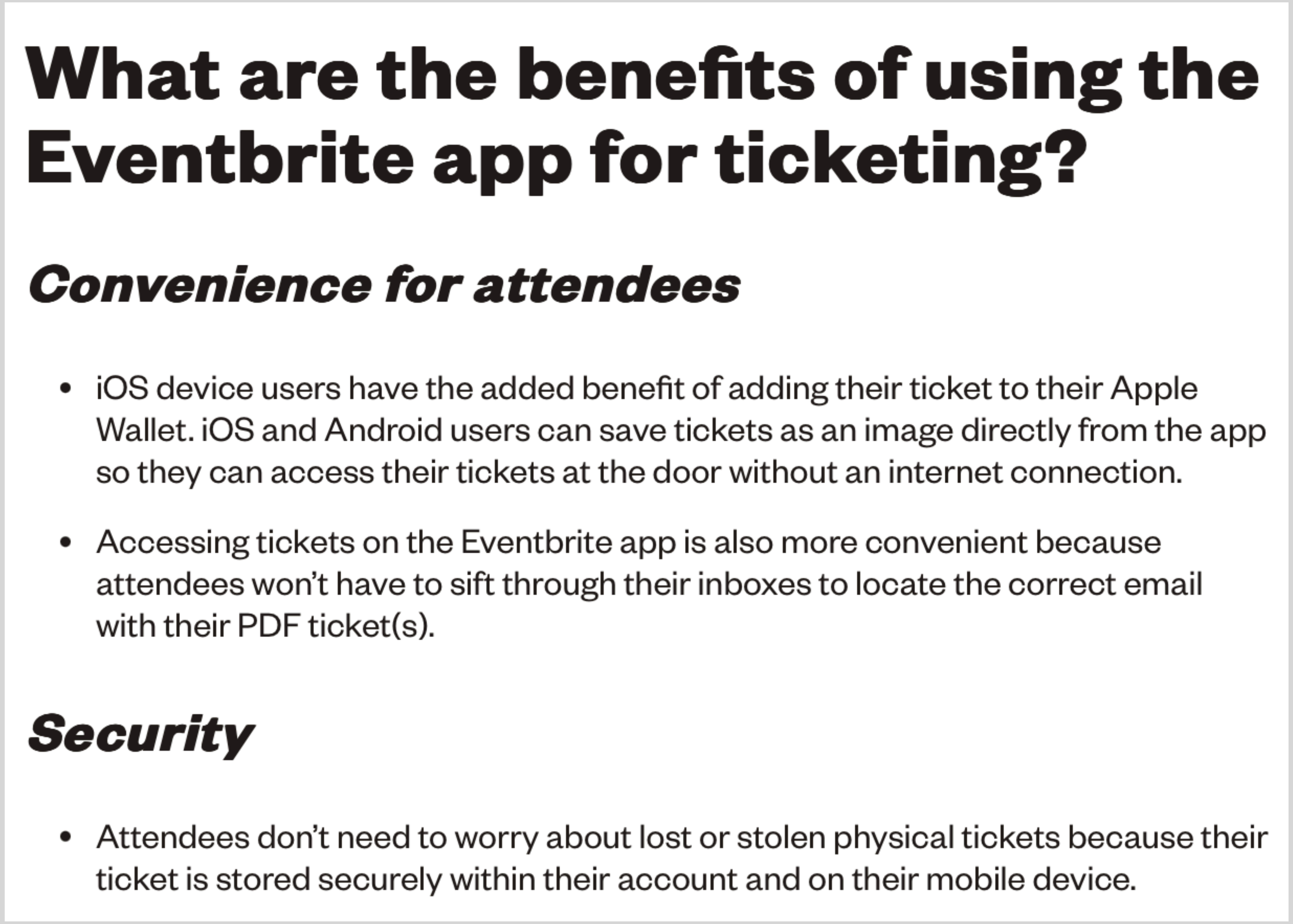

UPDATES AND ANNOUNCEMENTS

Important Update: How Attendees Access Tickets

BY EVENTBRITE • APR 16 2024


We've changed how attendees access their tickets to enhance the platform experience for you and your guests.

Moving forward, attendees will no longer receive PDF tickets via their order confirmation and reminder emails. Instead, they are directed to their tickets through Eventbrite.com or the Eventbrite mobile app, where options such as 'Add to Apple Wallet' and saving tickets as images are available.


What are the benefits of using the Eventbrite app for ticketing?

Convenience for attendees

- iOS device users have the added benefit of adding their ticket to their Apple Wallet. iOS and Android users can save tickets as an image directly from the app so they can access their tickets at the door without an internet connection.
- Accessing tickets on the Eventbrite app is also more convenient because attendees won't have to sift through their inboxes to locate the correct email with their PDF ticket(s).

Security

- Attendees don't need to worry about lost or stolen physical tickets because their ticket is stored securely within their account and on their mobile device.

can you explain this design in concept lingo?

what are the UX implications?

does it have the claimed benefits? who really gains?

takeaways

process & concept elements

pick a name
specific to function
but for general use

concept EventTicket [User]

describe purpose
why design or use it?
value to stakeholders

purpose managing
event attendance

tell story
a simple scenario
of how it's used
including setup

principle an event organizer creates an event (and announces it or invites people to it); they can then register, and the organizer can see who registered; eventually the people who registered can attend the event

list actions
by user or system
key steps, not UI

actions
create (by: User, on: Date): Event
register (e: Event, u: User)
attend (e: Event, u: User)

specify state
what's remembered
enough for actions

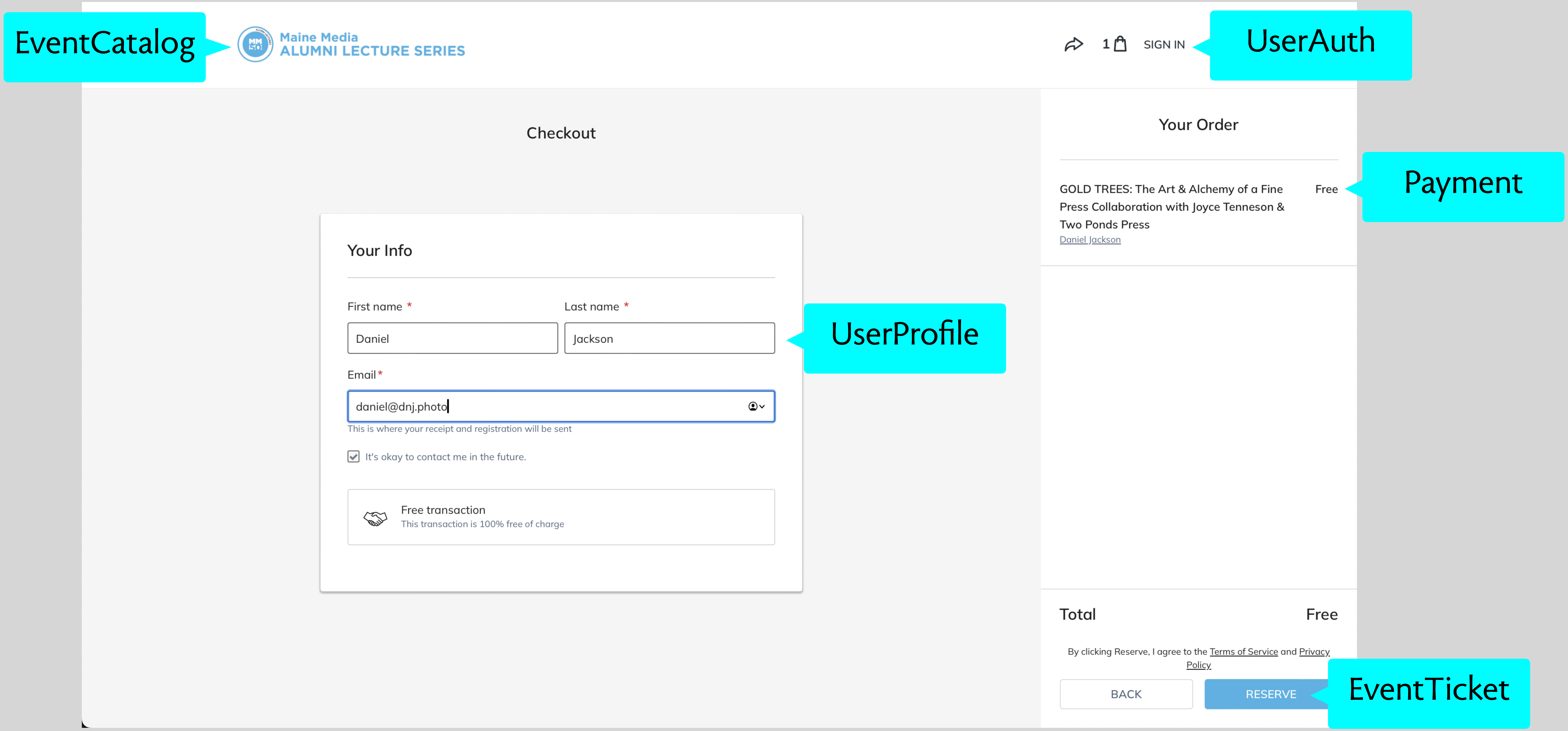
state
events: **set** Event
date: Event -> **one** Date
organizer: Event -> **one** User
registrants: Event -> **set** User
attendees: Event -> **set** User

two sides of a concept

users' perspective
a behavioral protocol

software perspective
a "nanoservice"

one page, but many concepts



Maine Media
ALUMNI LECTURE SERIES

1 SIGN IN

Checkout

Your Info

First name * Daniel

Last name * Jackson

Email * daniel@dnj.photo

This is where your receipt and registration will be sent

☑ It's okay to contact me in the future.

Free transaction
This transaction is 100% free of charge

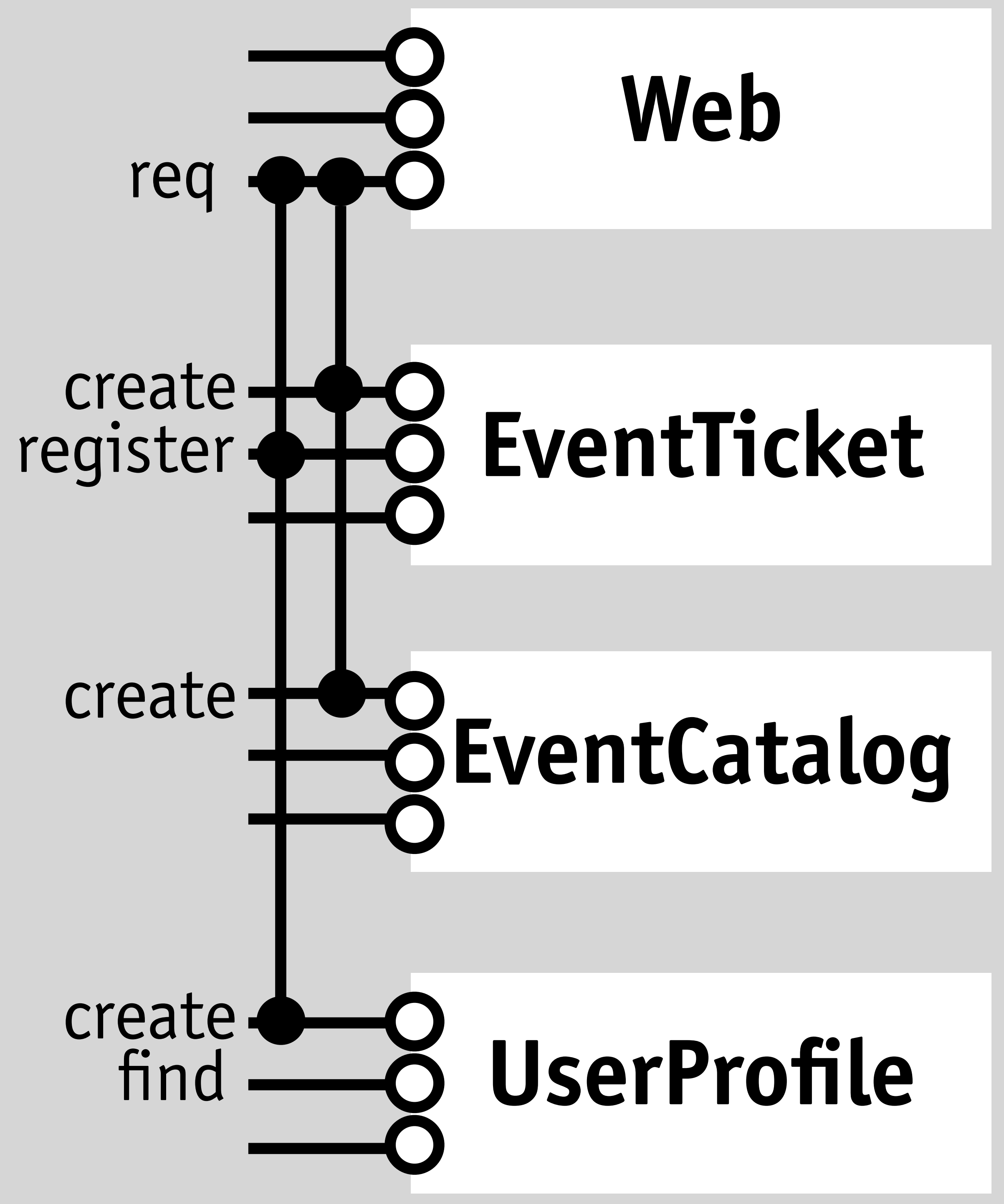
Your Order

GOLD TREES: The Art & Alchemy of a Fine Press Collaboration with Joyce Tenneson & Two Ponds Press Daniel Jackson	Free
Total	Free

By clicking Reserve, I agree to the Terms of Service and Privacy Policy

BACK RESERVE

synchronizations



```
when Web.req (create, by, on, title) then
  ec = EventCatalog.create (title, by)
  et = EventTicket.create (by, on)
  et.catalog = ec
```

```
when Web.req (view, event) then
  ec = event.catalog
  s = EventCatalog.get_details (ec)
  Web.response (s)
```

```
when Web.req (register, email, fst, lst, event) then
  u = UserProfile.create (fst, lst, email)
  EventTicket.register (event,  u)
```

```
when Web.req (register, email, event)
  u = UserProfile.find (email)
then EventTicket.register (event,  u)
```

runtime coupling
but no design coupling
or code coupling

app-specific behaviors
often in syncs alone
so concepts stay pure

what’s next?

what’s next?

a design exercise
you’ll design a concept similar to EventTicket
hands-on experience, always trickier than it seems
but also always more interesting…