RED space

Webinar Series

Better Living Through Composition

Let's talk — sarah.padovani@redspace.com / redspace.com

June 10, 2026

RED space

Webinar Series

Sarah Padovani

Web Team Lead

RED space

Webinar
Series

Disclaimers

- This is NOT an inheritance vs composition talk
- This talk is about designing for composability
- React examples, global concepts

RED space

Webinar
Series

Composition is...

The act of breaking a complex problem down into smaller problems, and composing simple solutions to form a complete solution.

RED space

Webinar
Series

Software Development is...

The act of breaking a complex problem down into smaller problems, and composing simple solutions to form a complete solution.

RED space

Webinar
Series

How do I do it well?

RED space

Webinar
Series

How to compose software

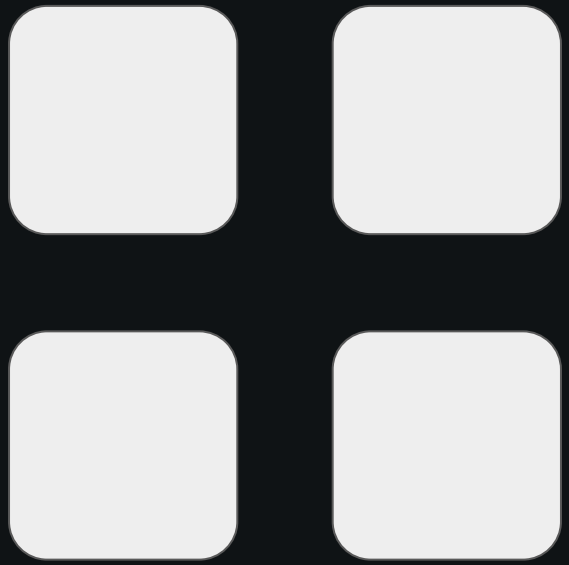
Step 1: Split responsibilities

RED space

Webinar Series

RED space

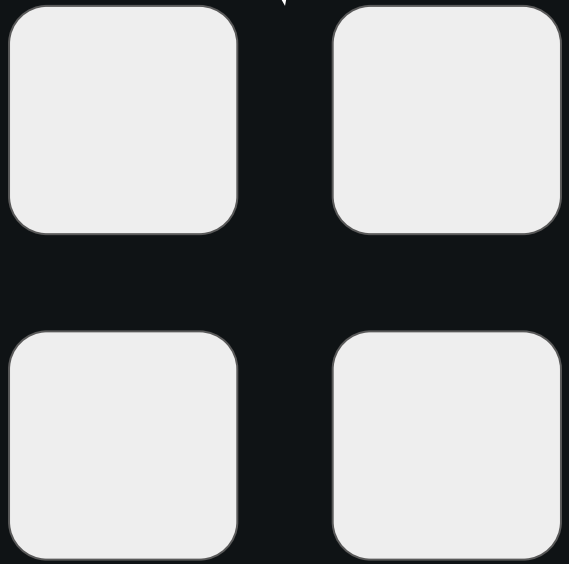
Webinar Series



RED space

Webinar Series

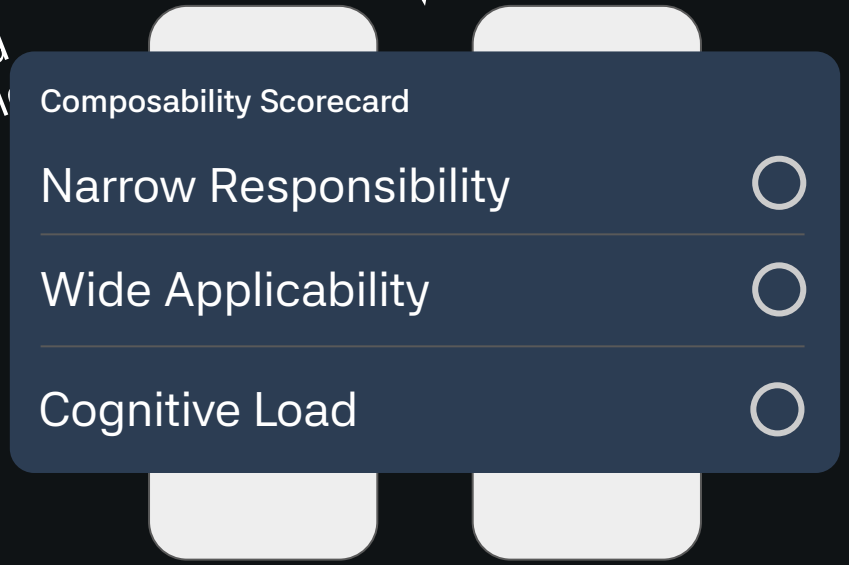
How do I find these boundaries?



RED space

Webinar
Series

How do I find these boundari



Where to draw boundaries

Look for reuse

Don't Repeat Yourself (DRY)

- Often interpreted as sharing code that looks the same
- Things that look the same are not necessarily LOGICALLY the same
- The purpose of DRY isn't to reduce bundle size, it's to avoid redundant logic and bugs
- If you DRY something that looks the same, you're going to need to unDRY them later

RED space

Webinar Series

RegistrationForm.tsx

```tsx
export const RegistrationForm = () => {
  const [username, setUsername] = useState("");
  const [address, setAddress] = useState("");

  const handleSubmit = () => {
    register(username, address)
  };

  return (
    <form onSubmit={handleSubmit}>
      <input placeholder="Name" onChange={setUsername} />
      <input placeholder="Address" onChange={setAddress} />
      <button>Submit</button>
    </form>
  )
}

```

ProfileForm.tsx

```tsx
export const ProfileForm = ({ name, address }) => {
  const [username, setUsername] = useState(name);
  const [address, setAddress] = useState(address);

  const handleSubmit = () => {
    save(username, address)
  };

  return (
    <form onSubmit={handleSubmit}>
      <input placeholder="Name" onChange={setUsername} />
      <input placeholder="Address" onChange={setAddress} />
      <button>Submit</button>
    </form>
  )
}

```

RED space

Webinar Series

CustomerForm.tsx

```tsx
export const CustomerForm: React.FC<CustomerFormProps> = ({
  username: initUsername,
  address: initAddress,
  onSubmit,
}) => {
  const [username, setUsername] = useState(initUsername);
  const [address, setAddress] = useState(initAddress);

  const handleSubmit = () => {
    onSubmit(username, address);
  };

  return (
    <form onSubmit={handleSubmit}>
      <input placeholder="Name" onChange={setUsername} />
      <input placeholder="Address" onChange={setAddress} />
      <button>Submit</button>
    </form>
  );
}

```

RED space

Webinar Series

RegistrationForm.tsx

```tsx
export const RegistrationForm = () => {
  const handleSubmit = (username, address) => {
    register(username, address)
  };

  return (
    <CustomerForm onSubmit={handleSubmit} />
  )
}

```

```tsx
  return (
    <form onSubmit={handleSubmi
      <input placeholder="Name'
      <input placeholder="Addre
      <button>Submit</button>
    </form>
  );
}

```

ProfileForm.tsx

```tsx
export const ProfileForm = ({ user }) => {
  const handleSubmit = (username, address) => {
    save(username, address)
  };

  return (
    <CustomerForm
      username={user.name}
      address={user.address}
      onSubmit={handleSubmit}
    />
  )
}

```

RED space

Webinar Series

RegistrationForm.tsx

```tsx
export const RegistrationForm = () => {
  const handleSubmit = (username, address) => {
    register(username, address)
  };

  return (
    <CustomerForm onSubmit={handleSubmit} />
  )
}

```

```tsx
  return (
    <form onSubmit={handleSubmi
      <input placeholder="Name'
      <input placeholder="Addre
      <button>Submit</button>
    </form>
  );
}

```

ProfileForm.tsx

```tsx
export const ProfileForm = ({ user }) => {
  const handleSubmit = (username, address) => {
    save(username, address)
  };

  return (
    <CustomerForm
      username={user.name}      // readonly
      address={user.address}
      onSubmit={handleSubmit}
    />
  )
}

```

RED space

Webinar Series

CustomerForm.tsx

```tsx
export const CustomerForm = ({
  mode,
  onSubmit,
}) => {
  const [username, setUsername] = useState("");
  const [address, setAddress] = useState("");

  const handleSubmit = () => {
    onSubmit(username, address);
  };

  return (
    <form onSubmit={handleSubmit}>
      <input placeholder="Name" onChange={setUsername} disabled={mode === "edit"} />
      <input placeholder="Address" onChange={setAddress} />
      <button>Submit</button>
    </form>
  );
}

```

RED space

Webinar Series

CustomerForm.tsx

```tsx
export const CustomerForm = ({
  mode,
  onSubmit,
}) => {
  const [username, setUsername] = useState("");
  const [address, setAddress] = useState("");

  const handleSubmit = () => {
    onSubmit(username, address);
  };

  return (
    <form onSubmit={handleSubmit}>
      <input placeholder="Name" onChange={setUserr
      <input placeholder="Address" onChange={setAd
      <button>Submit</button>
    </form>
  );
}
```

Composability Scorecard

- Narrow Responsibility ✗
- Wide Applicability ✗
- Cognitive Load ✗

RED space

Webinar Series

utils.ts

```ts
export function formatApi(id: string) {
  return `https://api.com/${id}`
}

export function formatLocalStorage(videoId: string, progress: string) {
  return `video-progress_${videoId}--${progress}`
}

```

RED space

Webinar Series

utils.ts

```ts
export function formatApi(id: string) {
  return `https://api.com/${id}`
}

export function formatLocalStorage(videoId: strir
  return `video-progress_${videoId}--${progress}`
}
```

Composability Scorecard

Narrow Responsibility	✅
Wide Applicability	❌
Cognitive Load	✅

RED space

Webinar Series

utils.ts

```ts
parseUri("https://api.com/{id}", { id: "param" })
parseUri("video-progress_{videoId}--{progress}", { videoId: "video", progress: "2" })

export const parseUri = (template: string, context: { [key: string]: string }): string => {
  return template.replace(/{([^{}]+)\}/g, function (_, expression: string) {
    return context[expression];
  });
};

```

RED space

Webinar Series

utils.ts

```ts
parseUri(Endpoints.Thing, { id: "param" })
parseUri(LocalStorageKeys.VideoProgress, { videoId: "video", progress: "2" })

export const parseUri = (template: string, context: { [key: string]: string }): string => {
  return template.replace(/{([^{}]+)\}/g, function (_, expression: string) {
    return context[expression];
  });
};

```

RED space

Webinar Series

utils.ts

```ts
parseUri(Endpoints.Thing, { id: "param" })
parseUri(LocalStorageKeys.VideoProgress, { videoI

export const parseUri = (template: string, contex
  return template.replace(/{([^{}]+)\}/g, functic
    return context[expression];
  });
};

```

Composability Scorecard

- Narrow Responsibility ✓
- Wide Applicability ✓
- Cognitive Load ✓

Where to draw boundaries

Look for separate concerns

- Do one thing and do it well
- If you’re doing multiple things, it might make sense to split them up

RED space

Webinar Series

OrderProcessor.ts

```ts
export function OrderProcessor() {
  return (order: Order) => {
    const total = order.items.reduce(totalPrice, 0)
    const receipt = { orderId: order.id, total, processedAt: new Date() };

    const message = `Order ${receipt.orderId}: $${receipt.total}`;
    nodemailer.sendMail({ to: order.customerEmail, subject: "Receipt", text: message });
  }
}

const process = OrderProcessor()
process(order)

function totalPrice(item, accum) {
  return accum + (item.price * item.qty)
}

```

RED space

Webinar Series

OrderProcessor.ts

```ts
export function OrderProcessor() {
  return (order: Order) => {
    const total = order.items.reduce(totalPrice, 0)
    const receipt = { orderId: order.id, total, pr
    const message = `Order ${receipt.orderId}: $${
    nodemailer.sendMail({ to: order.customerEmail,
  }
}

const process = OrderProcessor()
process(order)

function totalPrice(item, accum) {
  return accum + (item.price * item.qty)
}
```

Composability Scorecard

- Narrow Responsibility
- Wide Applicability
- Cognitive Load

RED space

Webinar Series

OrderProcessor.ts

```ts
export function OrderProcessor(notifier: Notifier) {
  return (order: Order) => {
    const total = order.items.reduce(totalPrice, 0)
    const receipt = { orderId: order.id, total, processedAt: new Date() };

    notifier.send(receipt)
  }
}

OrderProcessor(EmailNotifier)
OrderProcessor(SlackNotifier)
OrderProcessor(NullNotifier)
OrderProcessor(UserPreferences.notifier)
```

RED space

Webinar Series

OrderProcessor.ts

```ts
export function OrderProcessor(notifier: Notifier) {
  return (order: Order) => {
    const total = order.items.reduce(totalPrice,
    const receipt = { orderId: order.id, total, p

    notifier.send(receipt)
  }
}

OrderProcessor(EmailNotifier)
OrderProcessor(SlackNotifier)
OrderProcessor(NullNotifier)
OrderProcessor(UserPreferences.notifier)
```

Composability Scorecard

- Narrow Responsibility ✓
- Wide Applicability ✓
- Cognitive Load ✓

RED space

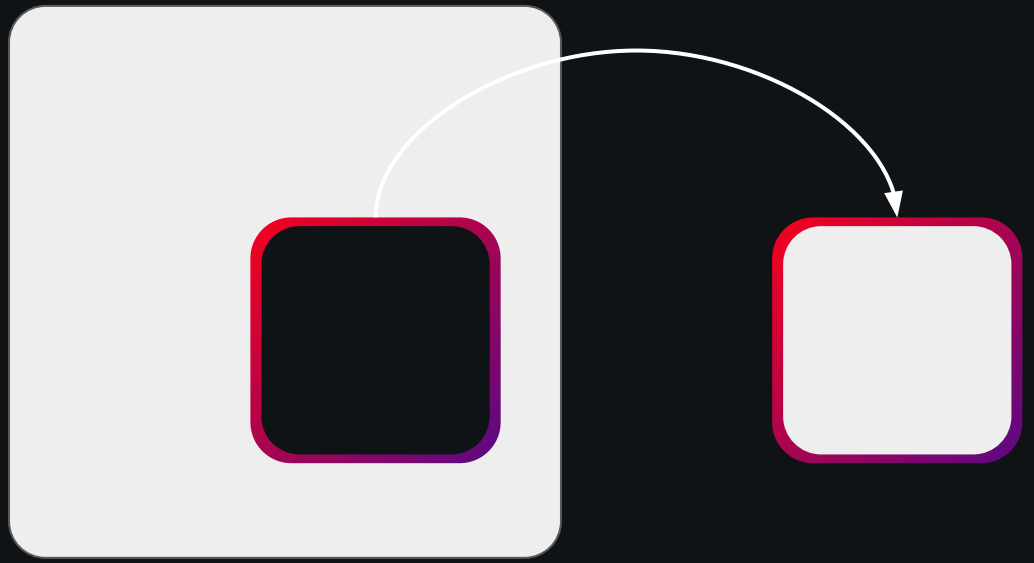
Webinar Series

RED space

Webinar Series

RED space

Webinar Series



RED space

Webinar Series

Where to draw boundaries

Look for noise

- Signal to noise ratio
- Noisy code has a high cognitive debt
- "What is important?"
- Remove what isn't important

RED space

Webinar Series

Dashboard.tsx

```tsx
const Dashboard = ({ auditCycles, isProcessing }) => (
  <div className="flex w-full flex-col gap-6">
    {auditCycles.map((cycle) => {
      if (cycle.source === "client") {
        const receipts = cycle.payload.docs
          .filter((p) => p.data.type === "receipt")
          .map((p) => ({ id: p.id, ...p.data as ReceiptData, status: "ready" as const }))

        return (
          <div key={cycle.id} className="flex flex-col items-end gap-2">
            {receipts.length > 0 && <ReceiptPreview files={receipts} />}
            {cycle.notes && <div className="bg-primary_hover">{cycle.notes}</div>}
          </div>
        )
      }

      const { logs, lastIdx } = cycle
      const isLastCycle = isProcessing && lastIdx === history.length - 1

      const items = logs.flatMap((m) => m.docs).sort((a, b) => a.order - b.order)
      const displayCost = logs.some((m) => m.riskCost != null)
        ? logs.reduce((sum, m) => sum + (m.riskCost ?? 0), 0)
        : logs.reduce((sum, m) => sum + (m.cost ?? 0), 0) || null

      const summary = logs.map((m) => m.summary).filter(Boolean).join("\n\n")

      return (
        <div key={logs[0].id} className="max-w-[584px]">
          {displayCost && <RiskBlock cost={displayCost} isProcessing={isLastCycle} />}

          {items
            .filter((p) => p.data.type === "gateway")
            .map((p) => {
              const data = p.data as GatewayData
              return (
                <ClearanceBlock
                  key={p.id}
                  gateway={data.gateway}
```

RED space

Webinar Series

Dashboard.tsx

```tsx
const Dashboard = ({ auditCycles, isProcessing }) => (
  <div className="flex w-full flex-col gap-6">
    {auditCycles.map((cycle) => {
      if (cycle.source === "client") {
        const receipts = cycle.payload.docs
          .filter((p) => p.data.type === "receipt")
          .map((p) => ({ id: p.id, ...p.data as ReceiptData, status: "ready" as const }))

        return (
          <div key={cycle.id} className="flex flex
            {receipts.length > 0 && <ReceiptPrevie
            {cycle.notes && <div className="bg-pri
          </div>
        )
      }

      const { logs, lastIdx } = cycle
      const isLastCycle = isProcessing && lastIdx

      const items = logs.flatMap((m) => m.docs).sc
      const displayCost = logs.some((m) => m.riskC
        ? logs.reduce((sum, m) => sum + (m.riskCos
        : logs.reduce((sum, m) => sum + (m.cost ??

      const summary = logs.map((m) => m.summary).f

      return (
        <div key={logs[0].id} className="max-w-[584px]">
          {displayCost && <RiskBlock cost={displayCost} isProcessing={isLastCycle} />}

          {items
            .filter((p) => p.data.type === "gateway")
            .map((p) => {
              const data = p.data as GatewayData
              return (
                <ClearanceBlock
                  key={p.id}
                  gateway={data.gateway}
```

Composability Scorecard

Narrow Responsibility ✗

Widely Applicable ✗

Cognitive Load ✗

RED space

Webinar Series

Dashboard.tsx

```tsx
const Dashboard = ({ auditCycles, isProcessing }) => {
  const { processedCycles, resolveTx } = useLedgerData(auditCycles, isProcessing)

  return (
    <div className="flex w-full flex-col gap-6">
      {processedCycles.map((cycle) => {
        if (cycle.type === "client") {
          return (
            <div key={cycle.id} className="flex flex-col items-end gap-2">
              {cycle.receipts.length > 0 && <ReceiptPreview files={cycle.receipts} />}
              {cycle.notes && <div className="bg-primary_hover">{cycle.notes}</div>}
            </div>
          )
        }

        return (
          <div key={cycle.id} className="max-w-[584px]">
            {cycle.displayCost && <RiskBlock cost={cycle.displayCost} isProcessing={cycle.isLast

            {cycle.gateways.map((g) => (
              <ClearanceBlock
                key={g.id}
                gateway={g.gateway}
                onApprove={(id) => resolveTx(id, "clear")}
                onReject={(id, rsn) => resolveTx(id, "bounce", rsn)}
              />
            ))}

            {cycle.summary && <Markdown className="prose text-primary">{cycle.summary}</Markdowr
            {cycle.isLastCycle && !cycle.summary && <ClearingIndicator />}
          </div>
        )
      })}
    </div>
  )
}

```

RED space

Webinar Series

Dashboard.tsx

```tsx
const Dashboard = ({ auditCycles, isProcessing }) => {
  const { processedCycles, resolveTx } = useLedgerData(auditCycles, isProcessing)

  return (
    <div className="flex w-full flex-col gap-6">
      {processedCycles.map((cycle) => {
        if (cycle.type === "client") {
          return (
            <div key={cycle.id} className="flex flex-col items-end gap-2">
              {cycle.receipts.length > 0 && <ReceiptPreview files={cycle.receipts} />}
              {cycle.notes && <div className="bg-primary_hover">{cycle.notes}</div>}
            </div>
          )
        }

        return (
          <div key={cycle.id} className="max-w-[584px]">
            {cycle.displayCost && <RiskBlock cost={cycle.displayCost} isProcess

            {cycle.gateways.map((g) => (
              <ClearanceBlock
                key={g.id}
                gateway={g.gateway}
                onApprove={(id) => resolveTx(id, "clear")}
                onReject={(id, rsn) => resolveTx(id, "bounce", rsn)}
              />
            ))}

            {cycle.summary && <Markdown className="prose text-primary">{cycle.summary}</Markdowr
            {cycle.isLastCycle && !cycle.summary && <ClearingIndicator />}
          </div>
        )
      })}
    </div>
  )
}
```

Dashboard

RED space

Webinar Series

Dashboard.tsx

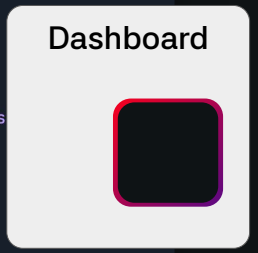
```tsx
const Dashboard = ({ auditCycles, isProcessing }) => {
  const { processedCycles, resolveTx } = useLedgerData(auditCycles, isProcessing)

  return (
    <div className="flex w-full flex-col gap-6">
      {processedCycles.map((cycle) => {
        if (cycle.type === "client") {
          return (
            <div key={cycle.id} className="flex flex-col items-end gap-2">
              {cycle.receipts.length > 0 && <ReceiptPreview files={cycle.receipts} />}
              {cycle.notes && <div className="bg-primary_hover">{cycle.notes}</div>}
            </div>
          )
        }

        return (
          <div key={cycle.id} className="max-w-[584px]">
            {cycle.displayCost && <RiskBlock cost={cycle.displayCost} isProcess

            {cycle.gateways.map((g) => (
              <ClearanceBlock
                key={g.id}
                gateway={g.gateway}
                onApprove={(id) => resolveTx(id, "clear")}
                onReject={(id, rsn) => resolveTx(id, "bounce", rsn)}
              />
            ))}

            {cycle.summary && <Markdown className="prose text-primary">{cycle.summary}</Markdowr
            {cycle.isLastCycle && !cycle.summary && <ClearingIndicator />}
          </div>
        )
      })}
    </div>
  )
}
```



RED space

Webinar Series

Dashboard.tsx

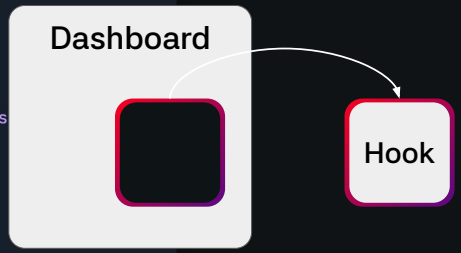
```tsx
const Dashboard = ({ auditCycles, isProcessing }) => {
  const { processedCycles, resolveTx } = useLedgerData(auditCycles, isProcessing)

  return (
    <div className="flex w-full flex-col gap-6">
      {processedCycles.map((cycle) => {
        if (cycle.type === "client") {
          return (
            <div key={cycle.id} className="flex flex-col items-end gap-2">
              {cycle.receipts.length > 0 && <ReceiptPreview files={cycle.receipts} />}
              {cycle.notes && <div className="bg-primary_hover">{cycle.notes}</div>}
            </div>
          )
        }

        return (
          <div key={cycle.id} className="max-w-[584px]">
            {cycle.displayCost && <RiskBlock cost={cycle.displayCost} isProcess

            {cycle.gateways.map((g) => (
              <ClearanceBlock
                key={g.id}
                gateway={g.gateway}
                onApprove={(id) => resolveTx(id, "clear")}
                onReject={(id, rsn) => resolveTx(id, "bounce", rsn)}
              />
            ))}

            {cycle.summary && <Markdown className="prose text-primary">{cycle.summary}</Markdowr
            {cycle.isLastCycle && !cycle.summary && <ClearingIndicator />}
          </div>
        )
      })}
    </div>
  )
}
```



RED space

Webinar Series

Dashboard.tsx

```tsx
const Dashboard = ({ auditCycles, isProcessing }) => {
  const { processedCycles, resolveTx } = useLedgerData(auditCycles, isProcessing)

  return (
    <div className="flex w-full flex-col gap-6">
      {processedCycles.map((cycle) => {
        if (cycle.type === "client") {
          return (
            <div key={cycle.id} className="flex flex-col items-end gap-2">
              {cycle.receipts.length > 0 && <Recei
              {cycle.notes && <div className="bg-p
            </div>
          )
        }

        return (
          <div key={cycle.id} className="max-w-[58
            {cycle.displayCost && <RiskBlock cost=

            {cycle.gateways.map((g) => (
              <ClearanceBlock
                key={g.id}
                gateway={g.gateway}
                onApprove={(id) => resolveTx(id, "
                onReject={(id, rsn) => resolveTx(i
              />
            ))}

            {cycle.summary && <Markdown className="prose text-primary">{cycle.summary}</Markdowr
            {cycle.isLastCycle && !cycle.summary && <ClearingIndicator />}
          </div>
        )
      })}
    </div>
  )
}

```

Composability Scorecard

- Narrow Responsibility ✓
- Wide Applicability ✓
- Cognitive Load ✓

RED space

Webinar Series

utils.ts

```ts
export async function executePrimaryProcessAction(): Promise<{ error: string | null; processId:
  try {
    ...
    const permittedResources = await determinePermittedResourcesForEntity({
      groupId: auth.entity.groupId,
      auth,
    })
    ...
}

```

RED space

Webinar Series

utils.ts

```ts
export async function determinePermittedResourcesForEntity({
  groupId,
  auth,
}: {
  groupId: string
  auth: SecurityEvaluator
}): Promise<GroupResourceProvider[]> {
  const availableResources = new Set((await fetchGroupItems(groupId)).map((item) ⇒ item.provide
  ...
}

```

RED space

Webinar Series

utils.ts

```ts
export const fetchGroupItems = async (groupId: string): Promise<GroupItemSummary[]> => {
    const summaryMap = await fetchGroupItemSummaryMap(groupId);
    ...
};

```

RED space

Webinar Series

utils.ts

```ts
const fetchGroupItemSummaryMap = async (groupId: string) => {
  ...
  const provider = extractGroupProviderFromItemName({ groupId })
  ...
}

```

RED space

Webinar Series

utils.ts

```ts
export const extractGroupProviderFromItemName = ({
  groupId,
  itemName,
  delimiter = ".",
}: {
  groupId: string
  itemName: string
  delimiter?: string
}): GroupResourceProvider | undefined => {
  let route: readonly string[]

  try {
    route = parseItemRoute(itemName, { delimiter })
  } catch {
    return undefined
  }
  ...
}
```

RED space

Webinar Series

utils.ts

```ts
export const parseItemRoute = (rawRoute: string, config?: ParseItemRouteConfig): ItemRoute => {
  const delimiter = config?.delimiter ?? DEFAULT_ITEM_ROUTE_DELIMITER
  const blocks = rawRoute.split(delimiter)

  return buildItemRoute(...asItemRoute(blocks))
}

```

RED space

Webinar Series

utils.ts

```ts
export const buildItemRoute = (...blocks: ItemRoute): ItemRoute =>
  asItemRoute(blocks.map((block, index) => standardizeItemRouteBlock(block, index)))

```

RED space

Webinar Series

utils.ts

```ts
const asSecretPath = (segments: string[]): SecretPath => {
  const [firstSegment, ...remainingSegments] = segments

  if (!firstSegment) {
    throw new Error("Secret path must contain at least one segment.")
  }

  return [firstSegment, ...remainingSegments]
}

```

RED space

Webinar Series

utils.ts

```
1  const asSecretPath = (segments: string
2    const [firstSegment, ...remainingSe
3
4    if (!firstSegment) {
5      throw new Error("Secret path must cont
6    }
7
8    return [firstSegment, ...remainingSegme
9  }
10
```

RED space

Webinar Series

utils.ts

```
1  const asSecretPath = (segments: string
2    const [firstSegment, ...remainingSe ts] =
3
4    if (!firstSegment) {
5      throw new Error("Secret path must cont at
6    }
7
8    return [firstSegment, ...remainingSegme
9  }
10
```

Composability Scorecard

- Narrow Responsibility ✅
- Wide Applicability ✅
- Cognitive Load ❌

RED space

Webinar Series

Where to draw boundaries

If you don’t need to… Don’t!

- Do you need to reuse this?
- Do you need to test this?
- Is the current responsibility too big?
- Don’t predict the future
- Remember your cognitive budget and spend it wisely

RED space

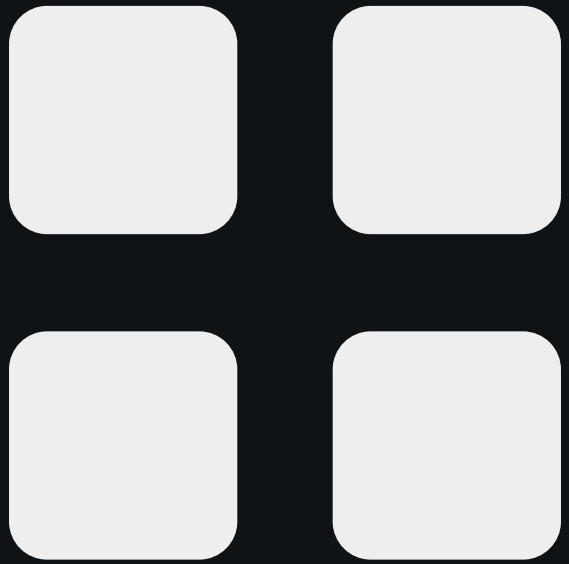
Webinar Series

How to compose software

Step 2: Strengthen Responsibilities

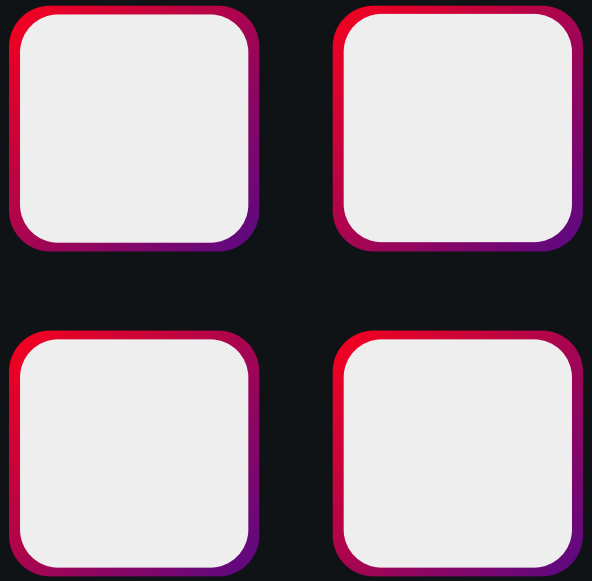
RED space

Webinar Series



RED space

Webinar Series



RED space

Webinar Series

useIntersecting.ts

```ts
function useIntersecting(ref: RefObject<Element>): boolean {
  const [isIntersecting, setIsIntersecting] = useState(false)

  useEffect(() => {
    const observer = new IntersectionObserver(([entry]) => {
      setIsIntersecting(entry.isIntersecting)
    })

    if (ref.current) observer.observe(ref.current)
    return () => observer.disconnect()
  }, [])

  return isIntersecting
}

const isIntersecting = useIntersecting(ref)

```

RED space

Webinar Series

useIntersecting.ts

```ts
function useIntersecting(ref: RefObject<Element>): boolean {
  const [isIntersecting, setIsIntersecting] = use

  useEffect(() => {
    const observer = new IntersectionObserver(([e
      setIsIntersecting(entry.isIntersecting)
    })

    if (ref.current) observer.observe(ref.current
    return () => observer.disconnect()
  }, [])

  return isIntersecting
}

const isIntersecting = useIntersecting(ref)
```

Composability Scorecard

- Narrow Responsibility ✅
- Wide Applicability ❌
- Cognitive Load ✅

RED space

Webinar Series

useIntersecting.ts

```ts
function useIntersection(ref: RefObject<Element>): IntersectionObserverEntry | null {
  const [entry, setEntry] = useState<IntersectionObserverEntry | null>(null);

  useEffect(() => {
    const observer = new IntersectionObserver(([entry]) => {
      setEntry(entry);
    });

    if (ref.current) observer.observe(ref.current);
    return () => observer.disconnect();
  }, []);

  return entry;
}

const { isIntersecting } = useIntersection(ref)

```

RED space

Webinar Series

useIntersecting.ts

```ts
function useIntersection(ref: RefObject<Element>): IntersectionObserverEntry | null {
  const [entry, setEntry] = useState<IntersectionO
  useEffect(() => {
    const observer = new IntersectionObserver(([e
      setEntry(entry);
    });
    if (ref.current) observer.observe(ref.current
    return () => observer.disconnect();
  }, []);
  return entry;
}
const { isIntersecting } = useIntersection(ref)
```

Composability Scorecard

- Narrow Responsibility ✓
- Wide Applicability ✓
- Cognitive Load ✓

RED space

Webinar Series

Button.tsx

```tsx
type ButtonProps = {
  label: string;
  onClick: () => void;
  disabled?: boolean;
  variant?: "primary" | "secondary";
}

export const Button: React.FC<ButtonProps> = ({
  label,
  onClick,
  disabled,
  variant = "primary"
}) => {
  return (
    <button
      onClick={onClick}
      disabled={disabled}
      className={`btn btn--${variant}`}
    >
      {label}
    </button>
  );
}

```

RED space

Webinar Series

Button.tsx

```tsx
type ButtonProps = {
  label: string;
  onClick: () => void;
  disabled?: boolean;
  variant?: "primary" | "secondary";
}

export const Button: React.FC<ButtonProps> = ({
  label,
  onClick,
  disabled,
  variant = "primary"
}) => {
  return (
    <button
      onClick={onClick}
      disabled={disabled}
      className={`btn btn--${variant}`}
    >
      {label}
    </button>
  );
}

```

Button.tsx

```tsx
<Button
  label={<SearchIcon />}      // label is typed as string
  onClick={handleSearch}
/>

```

RED space

Webinar Series

Button.tsx

```tsx
type ButtonProps = {
  label: string;
  onClick: () => void;
  disabled?: boolean;
  variant?: "primary" | "secondary";
}

export const Button: React.FC<ButtonProps> = ({
  label,
  onClick,
  disabled,
  variant = "primary"
}) => {
  return (
    <button
      onClick={onClick}
      disabled={disabled}
      className={`btn btn--${variant}`}
    >
      {label}
    </button>
  );
}

```

Button.tsx

```tsx
<Button
  label={<SearchIcon />}      // label is typed as string
  onClick={handleSearch}
/>

<Button
  label="Export"
  onClick={handleExport}
  onMouseEnter={showTooltip}  // not in ButtonProps
  onMouseLeave={hideTooltip}  // not in ButtonProps
/>

```

RED space

Webinar Series

Button.tsx

```tsx
type ButtonProps = {
  label: string;
  onClick: () => void;
  disabled?: boolean;
  variant?: "primary" | "secondary";
}

export const Button: React.FC<ButtonProps> = ({
  label,
  onClick,
  disabled,
  variant = "primary"
}) => {
  return (
    <button
      onClick={onClick}
      disabled={disabled}
      className={`btn btn--${variant}`}
    >
      {label}
    </button>
  );
}

```

Button.tsx

```tsx
<Button
  label={<SearchIcon />}      // label is typed as string
  onClick={handleSearch}
/>

<Button
  label="Export"
  onClick={handleExport}
  onMouseEnter={showTooltip}  // not in ButtonProps
  onMouseLeave={hideTooltip}  // not in ButtonProps
/>

<Button
  label="Save"
  type="submit"               // not in ButtonProps
/>

```

RED space

Webinar Series

Button.tsx

```tsx
type ButtonProps = {
  label: string;
  onClick: () => void;
  disabled?: boolean;
  variant?: "primary" | "secondary";
}

export const Button: React.FC<ButtonProps> = ({
  label,
  onClick,
  disabled,
  variant = "primary"
}) => {
  return (
    <button
      onClick={onClick}
      disabled={disabled}
      className={`btn btn--${variant}`}
    >
      {label}
    </button>
  );
}
```

Button.tsx

```tsx
<Button
  label={<SearchIcon />}      // label is typed as string
  onClick={handleSearch}
/>
```

Composability Scorecard

Narrow Responsibility	✅
Wide Applicability	❌
Cognitive Load	✅

RED space

Webinar Series

Button.tsx

```tsx
type ButtonProps = React.ComponentPropsWithoutRef<"button"> & {
  variant?: "primary" | "secondary";
};

export const Button: React.FC<ButtonProps> = ({
  variant = "primary",
  children,
  className,
  ...buttonProps
}) => {
  return (
    <button
      className={cn(`btn btn--${variant}`, className)}
      {...buttonProps}
    >
      {children}
    </button>
  );
}

```

Button.tsx

```tsx
<Button
  label={<SearchIcon />}      // label is typed as string
  onClick={handleSearch}
/>

<Button
  label="Export"
  onClick={handleExport}
  onMouseEnter={showTooltip}  // not in ButtonProps
  onMouseLeave={hideTooltip}  // not in ButtonProps
/>

<Button
  label="Save"
  type="submit"               // not in ButtonProps
/>

```

RED space

Webinar Series

Button.tsx

```tsx
type ButtonProps = React.ComponentPropsWithoutRef<"button"> & {
  variant?: "primary" | "secondary";
};

export const Button: React.FC<ButtonProps> = ({
  variant = "primary",
  children,
  className,
  ...buttonProps
}) => {
  return (
    <button
      className={cn(`btn btn--${variant}`, className)}
      {...buttonProps}
    >
      {children}
    </button>
  );
}
```

Button.tsx

```tsx
<Button
  label={<SearchIcon />}      // label is typed as string
  onClick={handleSearch}
/>
```

Composability Scorecard

- Narrow Responsibility ✓
- Wide Applicability ✓
- Cognitive Load ✓

RED space

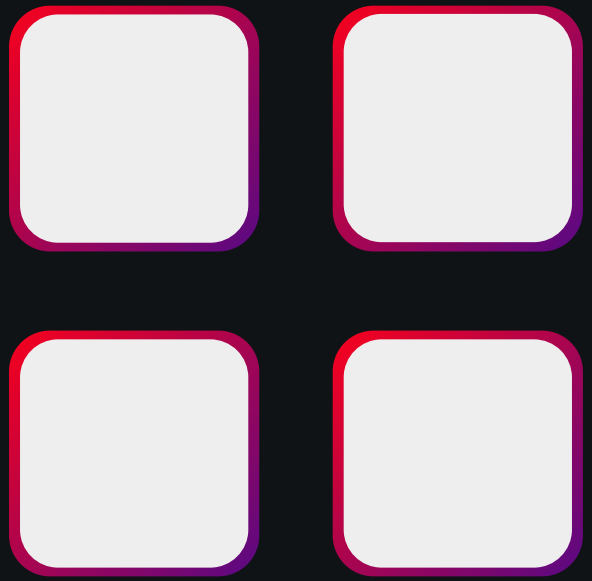
Webinar Series

How to compose software

Step 3: Connect responsibilities

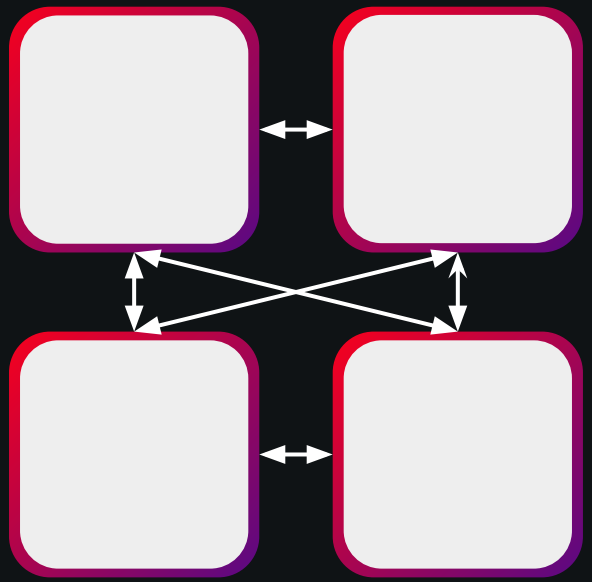
RED space

Webinar Series



RED space

Webinar Series



RED space

Webinar Series

Avatar.tsx

```tsx
<Avatar
  src="/john.png"
  name="John Doe"
  badge={true}
/>

```

Avatar.tsx

```tsx
type AvatarProps = {
  src: string
  name: string
  badge: boolean
}

const Avatar: React.FC<AvatarProps> = ({
  src,
  name,
  badge
}) => {
  return (
    <div>
      <img src={src} alt={name} />
      <span>{name}</span>

      {badge && <Badge color="green" />}
    </div>
  )
}

```

RED space

Webinar Series

Avatar.tsx

```tsx
<Avatar
  src="/john.png"
  name="John Doe"
  badge={true}
/>
```

Avatar.tsx

```tsx
type AvatarProps = {
  src: string
  name: string
  badge: boolean
}
```

Composability Scorecard

- Narrow Responsibility ✓
- Wide Applicability ✓
- Cognitive Load ✓

RED space

Webinar Series

Avatar.tsx

```tsx
<Avatar
  src="/john.png"
  name="John Doe"
  badgeColor="blue"
/>

```

Avatar.tsx

```tsx
type AvatarProps = {
  src: string
  name: string
  badgeColor: string
}

const Avatar: React.FC<AvatarProps> = ({
  src,
  name,
  badgeColor
}) => {
  return (
    <div>
      <img src={src} alt={name} />
      <span>{name}</span>

      {badgeColor && <Badge color={badgeColor} />}
    </div>
  )
}

```

RED space

Webinar Series

Avatar.tsx

```tsx
<Avatar
  src="/john.png"
  name="John Doe"
  badgeColor="blue"
/>
```

Avatar.tsx

```tsx
type AvatarProps = {
  src: string
  name: string
  badgeColor: string
}
```

Composability Scorecard

- Narrow Responsibility ✅
- Wide Applicability ❌
- Cognitive Load ✅

RED space

Webinar Series

Avatar.tsx

```tsx
<Avatar
  src="/john.png"
  name="John Doe"
  badge={<Badge color="blue" />}
/>

<Avatar
  src="/john.png"
  name="John Doe"
  badge={<Pill label="2" />}
/>

```

Avatar.tsx

```tsx
type AvatarProps = {
  src: string
  name: string
  badge: React.ReactNode
}

const Avatar: React.FC<AvatarProps> = ({
  src,
  name,
  badge
}) => {
  return (
    <div>
      <img src={src} alt={name} />
      <span>{name}</span>

      {badge}
    </div>
  )
}

```

RED space

Webinar Series

Avatar.tsx

```tsx
<Avatar
  src="/john.png"
  name="John Doe"
  badge={<Badge color="blue" />}
/>

<Avatar
  src="/john.png"
  name="John Doe"
  badge={<Pill label="2" />}
/>
```

Avatar.tsx

```tsx
type AvatarProps = {
  src: string
  name: string
  badge: React.ReactNode
}
```

Composability Scorecard

- Narrow Responsibility ✓
- Wide Applicability ✓
- Cognitive Load ✓

RED space

Webinar Series

Avatar.tsx

```tsx
<a href="/john">
  <Avatar
    src="/john.png"
    name="John Doe"
    badge={<Badge color="blue" />}
  />
</a>

<button onClick={handleClick}>
  <Avatar
    src="/john.png"
    name="John Doe"
    badge={<Pill label="2" />}
  />
</button>

```

RED space

Webinar Series

Avatar.tsx

```tsx
<Avatar
  src="/john.png"
  name="John Doe"
  component="a"
  href="/john"
/>

<Avatar
  src="/john.png"
  name="John Doe"
  component="button"
  onClick={handleClick}
/>
```

Avatar.tsx

```tsx
export type AvatarProps<C extends React.ElementType> = {
  component?: C
  src: string
  name: string
  badge: React.ReactNode
} & React.ComponentPropsWithoutRef<C>

const Avatar = <C extends React.ElementType>({
  src,
  name,
  badge,
  component,
  ...componentProps
}: AvatarProps<C>) => {
  const Comp = component || "div"

  return (
    <Comp {...componentProps}>
      <img src={src} alt={name} />
      <span>{name}</span>

      {badge}
    </Comp>
  )
}
```

RED space

Webinar Series

Avatar.tsx

```tsx
<Avatar
  src="/john.png"
  name="John Doe"
  component="a"
  href="/john"
/>

<Avatar
  src="/john.png"
  name="John Doe"
  component="button"
  onClick={handleClick}
/>
```

Avatar.tsx

```tsx
export type AvatarProps<C extends React.ElementType> = {
  component?: C
  src: string
  name: string
  badge: React.ReactNode
} & React.ComponentPropsWithoutRef<C>

      {badge}
    </Comp>
  )
}
```

Composability Scorecard

- Narrow Responsibility ✅
- Wide Applicability ✅
- Cognitive Load ✅

RED space

Webinar Series

Avatar.tsx

```tsx
const user = useUser("user_1")

<Avatar
  src={user.profileUrl}
  name={`${user.firstName} ${user.lastname}`}
/>

```

RED space

Webinar Series

Avatar.tsx

```tsx
function useUserAvatar(id: string): AvatarProps {
  const data = useFetch(parseUri(Endpoints.Profile, { id }))

  return {
    src: data?.profileUrl,
    name: `${data.firstName} ${data.lastName}`,
  }
}

const john = useUserAvatar("user_1")
const jane = useUserAvatar("user_2")

<Avatar {...john} />
<Avatar {...jane} />

```

RED space

Webinar Series

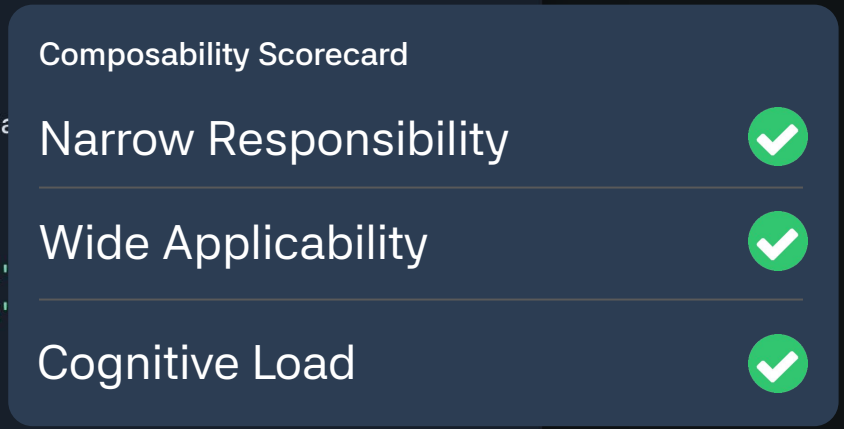
Avatar.tsx

```tsx
function useUserAvatar(id: string): AvatarProps {
  const data = useFetch(parseUri(Endpoints.Profile, { id }))

  return {
    src: data?.profileUrl,
    name: `${data.firstName} ${data
  }
}

const john = useUserAvatar("user_1"
const jane = useUserAvatar("user_2"

<Avatar {...john} />
<Avatar {...jane} />
```



RED space

Webinar Series

App.tsx

```tsx
<Video mode="autoplay" src="https://example.video" />
<Video mode="playlist" srcs={["https://example.video", "https://another.video"]} />

```

Video.tsx

```tsx
interface VideoProps {
  mode: "autoplay" | "playlist";
  src?: string;          // autoplay only
  srcs?: string[];       // playlist only
}

function Video({ mode, src, srcs }: VideoProps) {
  const [isPlaying, setIsPlaying] = useState(false);
  const [error, setError] = useState<MediaError | null>(null);
  const [retryCount, setRetryCount] = useState(0);
  const videoRef = useRef<HTMLVideoElement>(null);
  const MAX_RETRIES = 3;

  const [index, setIndex] = useState(0);
  const [progress, setProgress] = useState(0);

  useEffect(() => {
    const video = videoRef.current;
    if (!video) return;

    const onPlay  = () => setIsPlaying(true);
    const onPause = () => setIsPlaying(false);

    const onError = () => {
      setError(video.error);
      if (mode === "autoplay") {
        if (retryCount < MAX_RETRIES) {
          setTimeout(() => {
            video.load();
            video.play();
            setRetryCount(c => c + 1);
          }, 2000 * retryCount);
        }
      } else {
        if (index < (srcs?.length ?? 0) - 1) {
          setIndex(i => i + 1);
          setRetryCount(0);
        } else if (retryCount < MAX_RETRIES) {
          setTimeout(() => {
            video.load();
            video.play();
            setRetryCount(c => c + 1);
          }, 1000);
        }
      }
    };

```

RED space

Webinar Series

App.tsx

```tsx
<Video mode="autoplay" src="https://example.video" />
<Video mode="playlist" srcs={["https://example.video", "https://another.video"]} />

```

Video.tsx

```tsx
interface VideoProps {
  mode: "autoplay" | "playlist";
  src?: string;          // autoplay only
  srcs?: string[];       // playlist only
}

function Video({ mode, src, srcs }: VideoProps) {
  const [isPlaying, setIsPlaying] = useState(false);
  const [error, setError] = useState<MediaError | null>(null);
  const [retryCount, setRetryCount] = useState(0);
  const videoRef = useRef<HTMLVideoElement>(null);
  const MAX_RETRIES = 3;

  const [index, setIndex] = useState(0);
  const [progress, setProgress] = useState(0);

  useEffect(() => {
    const video = videoRef.current;
    if (!video) return;

    const onPlay  = () => setIsPlaying(true);
    const onPause = () => setIsPlaying(false);

    const onError = () => {
      setError(video.error);
      if (mode === "autoplay") {
        if (retryCount < MAX_RETRIES) {
          setTimeout(() => {
            video.load();
            video.play();
            setRetryCount(c => c + 1);
          }, 2000 * retryCount);
        }
      } else {
        if (index < (srcs?.length ?? 0) - 1) {
          setIndex(i => i + 1);
          setRetryCount(0);
        } else if (retryCount < MAX_RETRIES) {
          setTimeout(() => {
            video.load();
            video.play();
            setRetryCount(c => c + 1);
          }, 1000);
        }
      }
    };

```

RED space

Webinar Series

App.tsx

```tsx
<Video mode="autoplay" src="https://example.video" />
<Video mode="playlist" srcs={["https://example.video", "https://another.video"]} />
```

Video.tsx

```tsx
interface VideoProps {
  mode: "autoplay" | "playlist";
  src?: string;          // autoplay only
  srcs?: string[];       // playlist only
}

function Video({ mode, src, srcs }: VideoProps) {
  const [isPlaying, setIsPlaying] = useState(false);
  const [error, setError] = useState<MediaError | null>(null);
  const [retryCount, setRetryCount] = useState(0);
  const videoRef = useRef<HTMLVideoElement>(null);
  const MAX_RETRIES = 3;

  const [index, setIndex] = useState(0);
```

```tsx
    } else {
      if (index < (srcs?.length ?? 0) - 1) {
        setIndex(i => i + 1);
        setRetryCount(0);
      } else if (retryCount < MAX_RETRIES) {
        setTimeout(() => {
          video.load();
          video.play();
          setRetryCount(c => c + 1);
        }, 1000);
      }
    }
  };
```

Composability Scorecard

- Narrow Responsibility ✗
- Wide Applicability ✗
- Cognitive Load ✗

RED space

Webinar Series

Avatar.tsx

```
1 const autoPlayerProps = useAutoPlayer()
2 const playlistPlayer = usePlaylistPlayer()
3
4 <Video {...playlistPlayer} />
5
```

RED space

Webinar Series

Avatar.tsx

```
1 const autoPlayerProps = useAut
2 const playlistPlayer = usePlay
3
4 <Video {...playlistPlayer} />
5
```

Composability Scorecard

- Narrow Responsibility ✓
- Wide Applicability ✓
- Cognitive Load ✓

RED space

Webinar
Series

Why compose software?

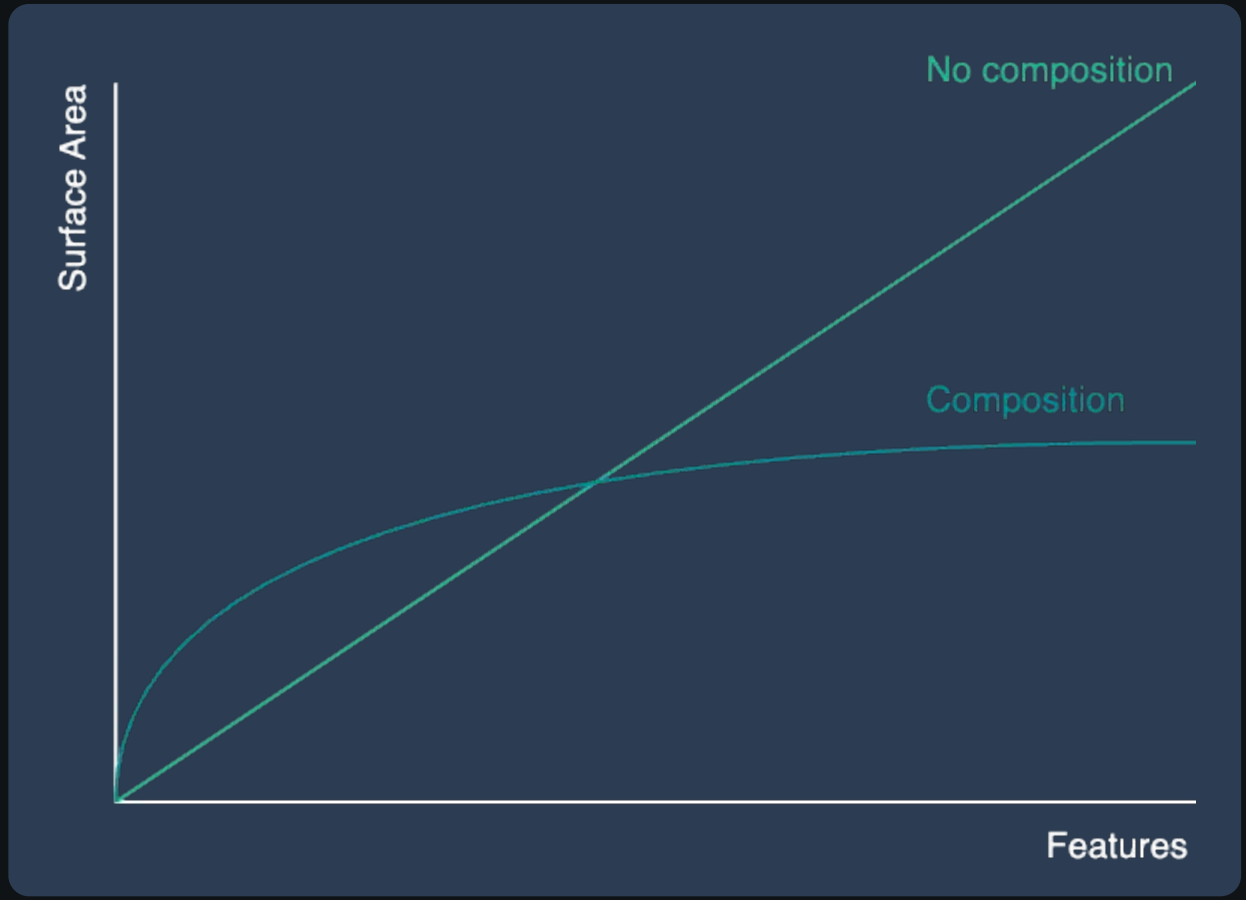
Improved Readability

- Reading code is harder than writing code.
- High cognitive cost makes your code harder to read.
- Well composed code should fit within your cognitive budget
    - Noisy code (not enough abstraction) buries the signal, making it hard to find what you need
    - Foggy code (too much abstraction) makes it easy to get lost
- All code has a cognitive cost. Make sure you're spending your budget wisely.

RED space

Webinar
Series

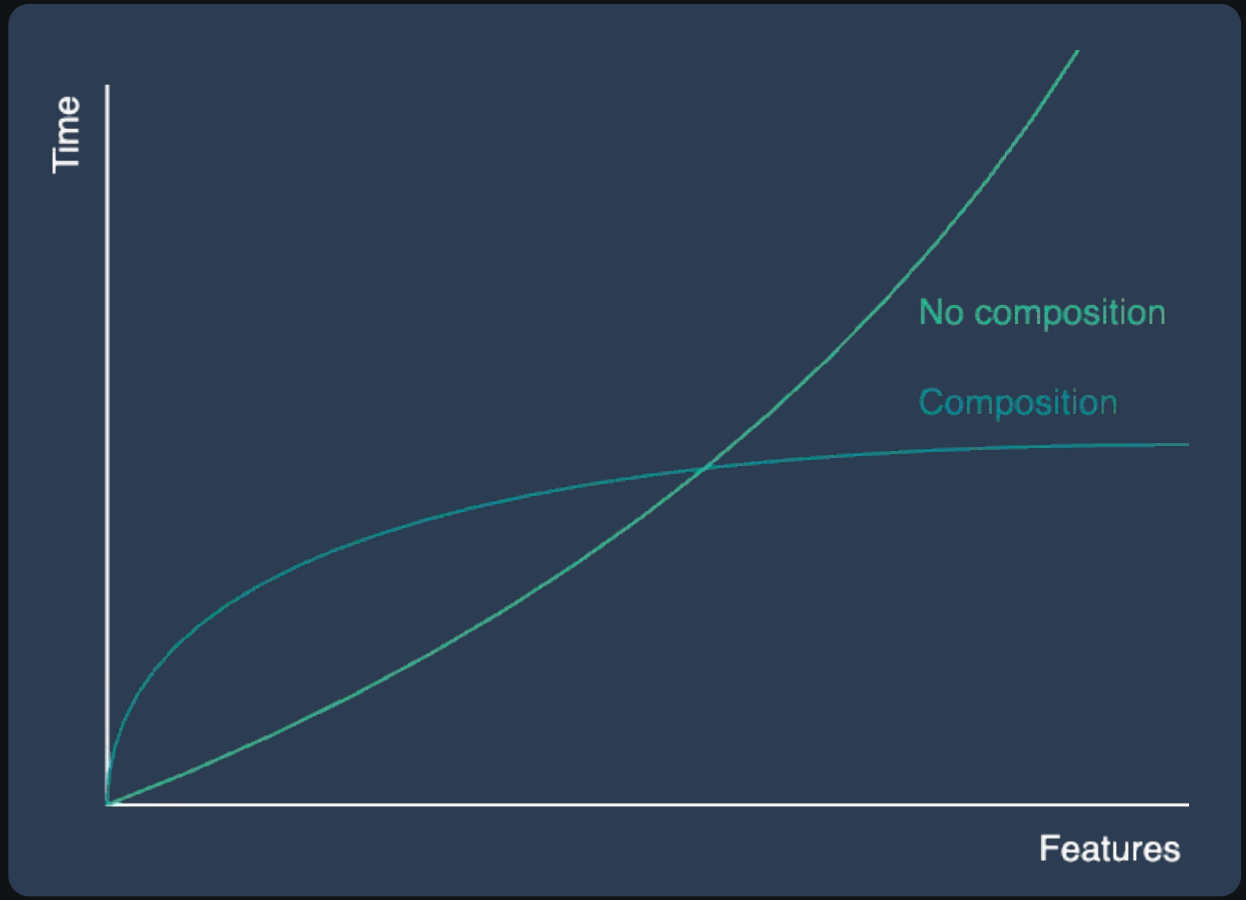
Less code



RED space

Webinar
Series

Faster development



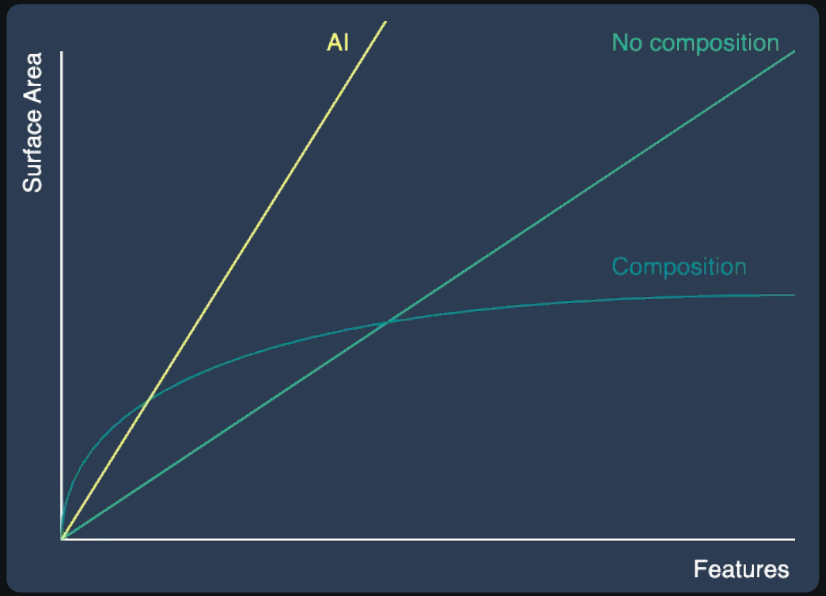
RED space

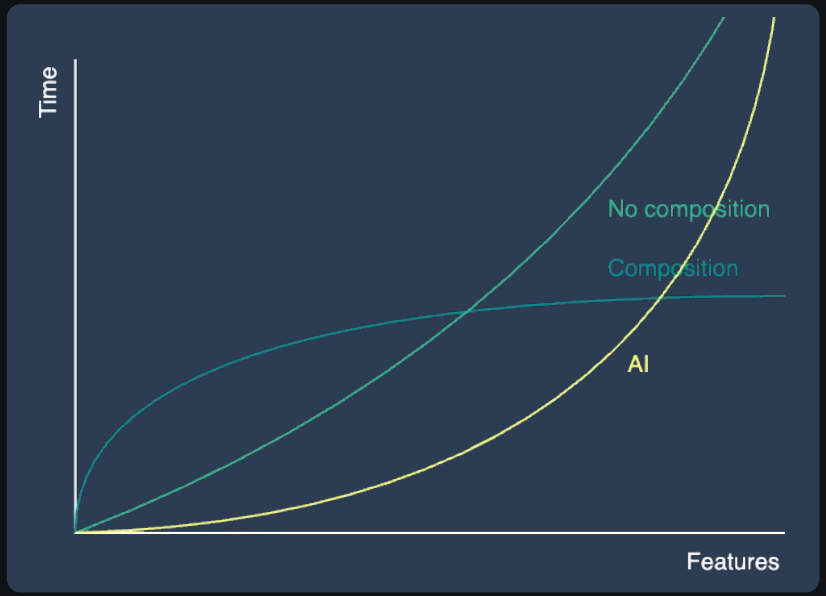
Webinar Series

But Sarah! What about AI?

RED space

Webinar Series





The cost of doing it poorly

- High cognitive debt
- Low output
- Confused developers
- Confused developers write bugs

RED space

Webinar Series

You’re already doing it, so do it well

RED space

Webinar Series

Thanks!

Reading

- Composing Software - An Introduction - Eric Elliott
- Compositional Software Design - Jakob Jenkov