

Design Systems

SESSION 15
10.19.26

What a design system is, where it came from, who in a company uses one, and how it gets better.

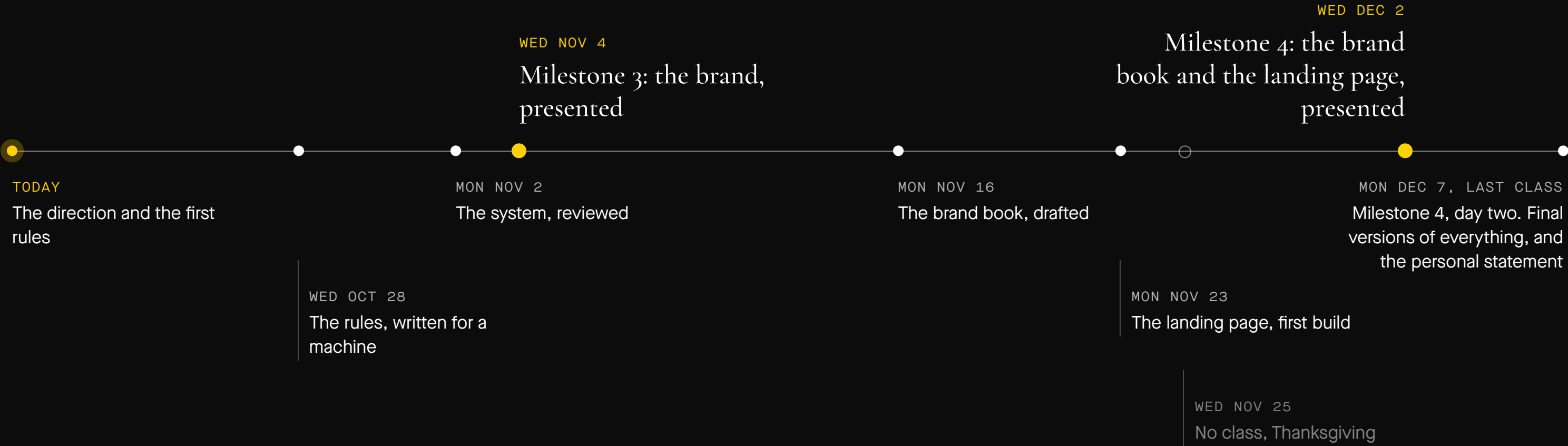
Workhorse Collective

ART 418 · Fall 2026

Name Abraham Garcia
Email abe@wrkhrs.co

Meets Mon + Wed
Today Session 15 · 10.19.2026

What is due, from today to the last class.



MILESTONES ARE PRESENTED TO THE ROOM. CHECKPOINTS ARE THE SMALLER STEPS ON THE WAY TO EACH ONE.

Today.

PART	WHAT	YOU LEAVE WITH
1	What a design system is. Its history, from IBM and NASA to atomic design and the files an AI tool reads today.	The parts of a system, and where they came from
2	How it is built today. Tailwind and shadcn/ui, the tools most websites and AI tools use now.	Semantic names, and a system an AI tool can build with
3	Who uses it. Marketing, sales, product, support, hiring, legal.	Why a business pays for a system
4	Made in loops. Iterative design, sketches, and prototypes.	How a system gets better each time it is used
5	The brand repository. The starter repository, read as a design system for one brand.	Brand first, and a brand an AI tool can follow
6	Your turn. Catch up your repository, choose one direction, and write your first rules.	The checkpoint: your direction and at least three rules, pushed

What a design system is

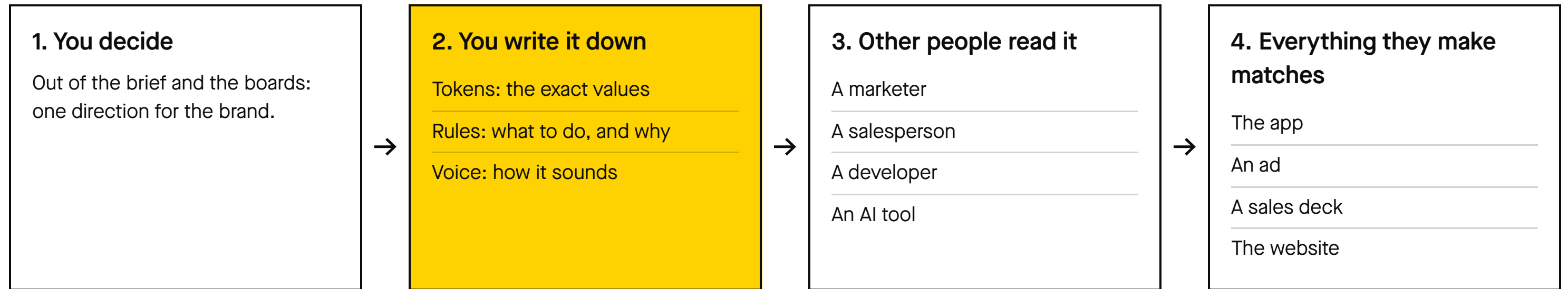
From the first printed manuals to the files an AI tool reads today.

A design system is a brand's decisions, recorded so that hundreds of people can make work that looks and sounds like one brand.

Up to now you have made one brand by hand: a pitch, a brief, boards. A company needs hundreds of people to make that brand every day, on screens and in places you will never see. The system is how they do it without you in the room.

If you have never heard the words before, that is normal. You have used dozens of design systems anyway: every Google app, every national park brochure, every bank's website.

How a design system works, in one picture.



You are in step 1 today, and you start step 2 before you leave. The yellow box is what the starter repository holds.

Words you will hear today.

WORD	WHAT IT MEANS HERE
Design system	A brand's decisions, written down so other people can make work that belongs to it
Token	One exact value with a name. <code>color.action</code> is the blue used for buttons.
Component	A part built once and used again: a button, a card, a menu
Rule	What to do, and why, for a call someone else will have to make. You write your first ones today.
Direction	The one brand you choose out of your moodboards
Repository	A project folder that keeps every saved version of every file. Yours lives on GitHub.
Commit, push	Save a version with a note about what changed, then send it up to GitHub
AI tool, skill	A program that can make things under your rules, and a written list of steps it follows

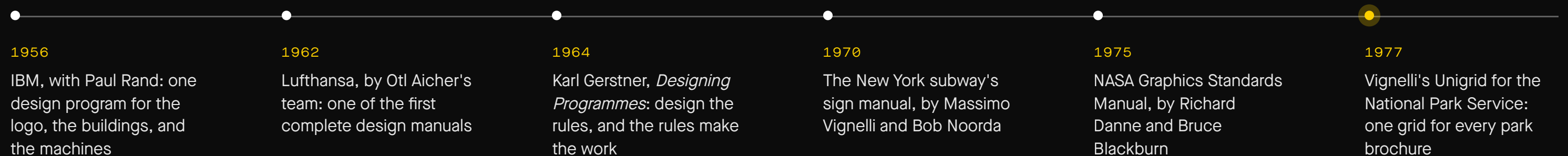
The parts of a system.

Today you start the rules and the voice. Tokens are Wednesday. Components and patterns come with the landing page in November.

PART	WHAT IT IS	ON THE INVENTED BUDGETING APP
Tokens	The exact values: colors, type sizes, spacing	Paper white, ink black, and one red
Components	The pieces things are built from	A button, a balance line, a weekly card
Patterns	How the pieces go together for a task	Setting up a budget in three screens
Rules	What to do, and why, for the judgment calls	Red only means overspent
Voice	How the brand sounds in writing	Plain sentences, never cheering
Who changes it	How a change gets proposed, decided, and recorded	The designer decides, and the change gets a date and a reason

Where design systems came from: printed manuals

It started with companies that had to look the same in a hundred places at once.



NO NEED TO MEMORIZE THESE DATES. EVERY ONE OF THESE WAS A BINDER, AND WHEN A RULE CHANGED, THE PAGE WAS SWAPPED OUT.

"Instead of solutions for problems, programmes for solutions."

KARL GERSTNER, *DESIGNING PROGRAMMES*, 1964

Gerstner was a Swiss designer. He argued that a designer should write the rules that produce the answers, so other people can use them to make the work. **That is the idea of a design system, sixty years ago.**

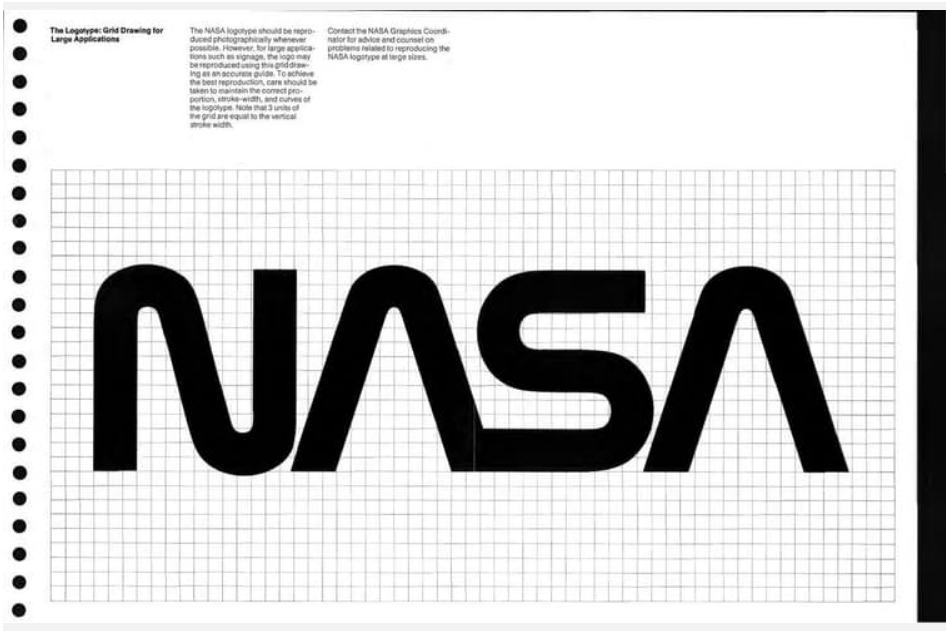
Your rules file is a program in his sense: someone who never met you follows it and makes something that belongs to your brand.

You saw this manual on Sept 30. Look at what kinds of rules it holds.



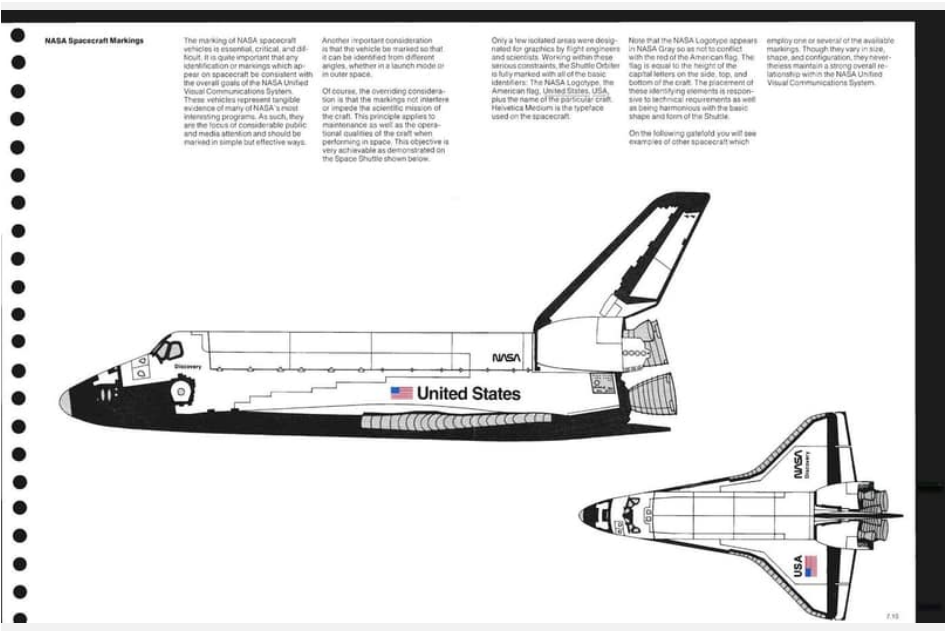
Exact values

Two colors, matched in any medium. Today we would call these tokens.



A component, drawn exactly

The logotype on a grid, so a sign painter can draw it at any size.



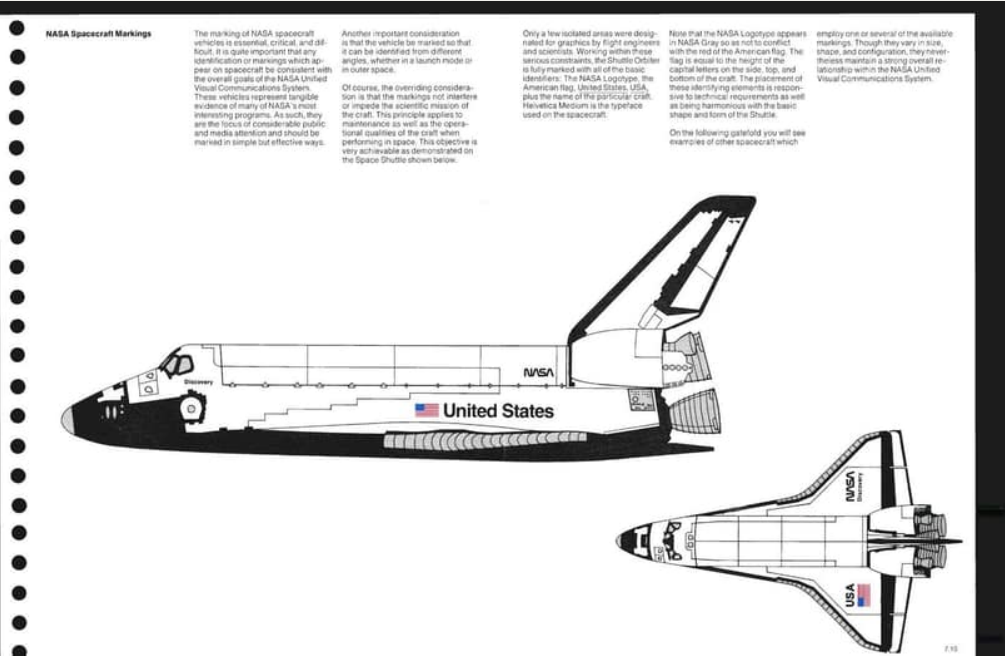
A judgment, with its reason

Where the markings go on the Space Shuttle, and why.

A rule says what to do, and it says why.

"Note that the NASA Logotype appears in NASA Gray so as not to conflict with the red of the American flag."

NASA SPACECRAFT MARKINGS, PAGE 7.10



NASA Graphics Standards Manual, 1975, page 7.10: NASA Spacecraft Markings. Richard Danne and Bruce Blackburn. Public domain.

WHAT TO DO

On the Shuttle, the logotype is gray instead of red.

WHY

The flag is right beside it, and two reds would fight.

WHAT IT LETS YOU DO

Someone who never met the designers knows what to do on the next spacecraft, and when the rule applies.

1977: Christopher Alexander describes design as patterns, each one a problem with its answer.

An architect, he and his colleagues wrote 253 patterns for towns and buildings, from a whole region down to a window seat, in a book called *A Pattern Language*. Software designers borrowed the idea in 1994, and design systems still call their parts patterns.

EACH OF ALEXANDER'S PATTERNS HAS

A name

The problem, and why it matters

The answer, said as an instruction

A photograph of it working

EACH RULE IN YOUR REPOSITORY HAS

A heading

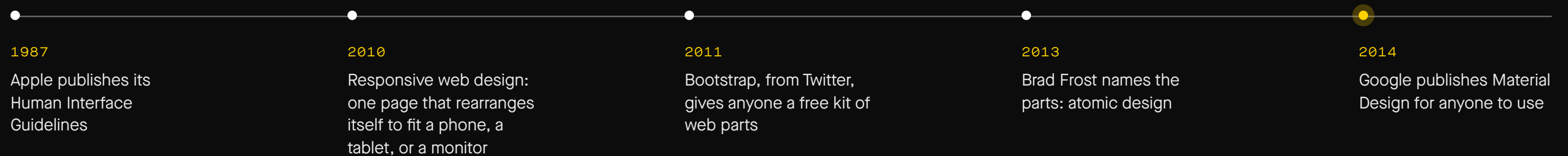
Because: the decision or the brief section it came from

Do: what to do, in one sentence

Right and **Wrong:** one example of each

Where design systems came from: screens

Then the brand moved onto screens, and the parts had to be built in code.



ONCE A PAGE HAD TO FIT EVERY SCREEN, NOBODY COULD DRAW IT JUST ONCE, SO DESIGNERS STARTED DRAWING THE PARTS.

Why atomic design happened.

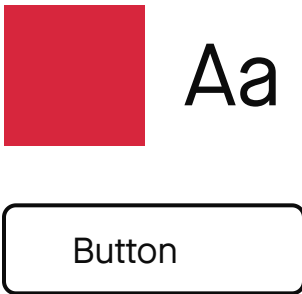
Before 2010	A designer drew each web page as one fixed picture, and a developer copied it.
2010	Responsive design meant the same page had to work on a phone, a tablet, and a wide monitor. There was no longer one picture to draw.
2012	Designers drew the parts instead. Samantha Warren's style tiles put a brand's colors, type, and buttons on one sheet before any page existed. Dan Mall's element collages did the same with pieces of real layouts.
2013	Brad Frost, a web designer, gave the parts names borrowed from chemistry. His book <i>Atomic Design</i> followed in 2016.

Brad Frost's idea: build pages from small parts, the way chemistry builds things from atoms.

Fix a part once and every page that uses it gets the fix.

Atoms	Molecules	Organisms	Templates	Pages
The smallest pieces: a color, a typeface, a button. The red for overspent	A few atoms that do one job together. A balance line: label, number, red when over	Molecules joined into a section. The weekly card: five balance lines and a total	The layout of a whole screen, with empty slots. The home screen, before any numbers	The template with real content in it. One student's home screen, with this week's spending

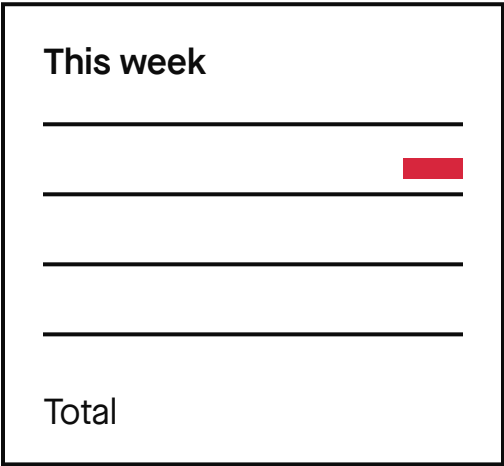
The same five levels, drawn for the budgeting app.



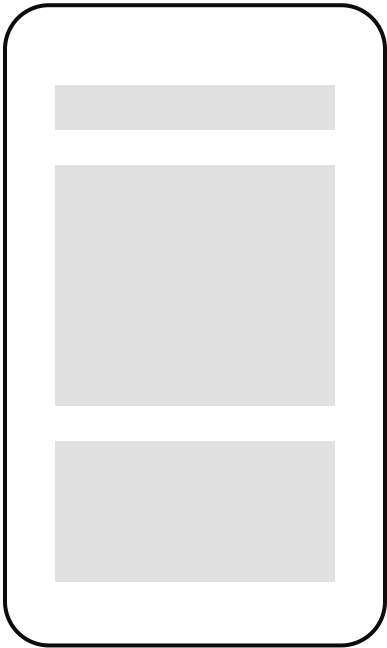
Atoms



Molecule



Organism



Template

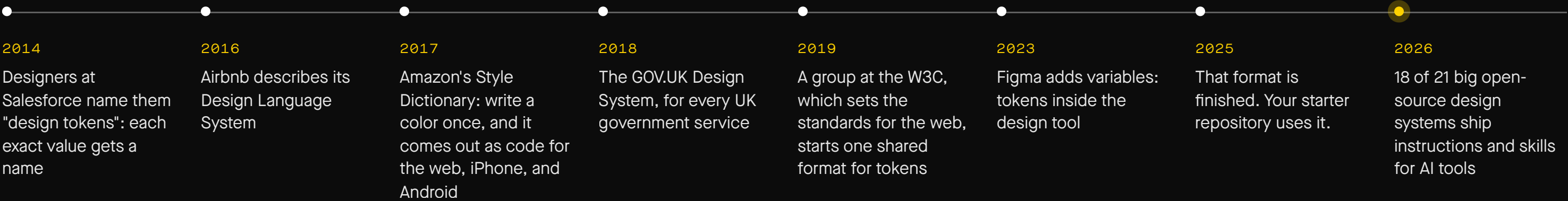


Page

Each level is made of the one before it. Change the red atom, and the balance line, the card, and every screen change with it.

Where design systems came from: tokens and machines

Then companies ran their systems as products, and the exact values became data.

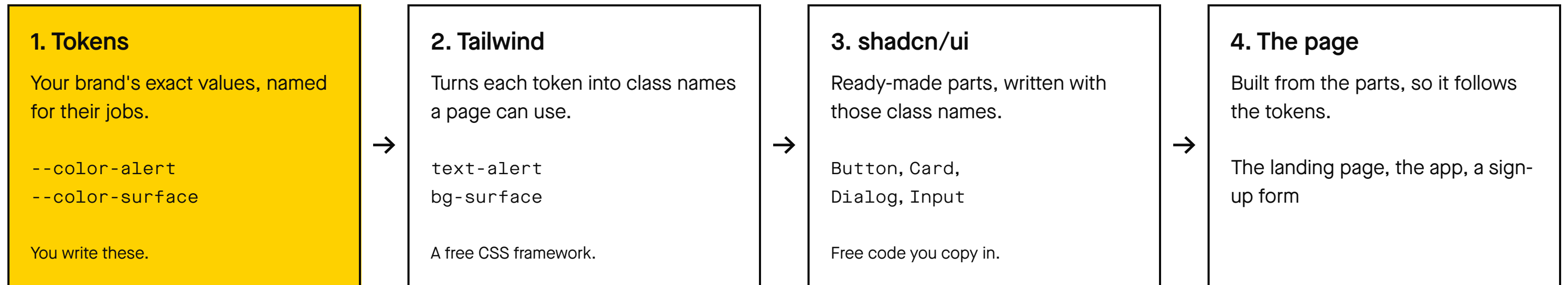


THE JOB STAYED THE SAME. WHO READS THE MANUAL CHANGED: A SIGN PAINTER, THEN A DEVELOPER, NOW A MACHINE AS WELL.

How it is built today

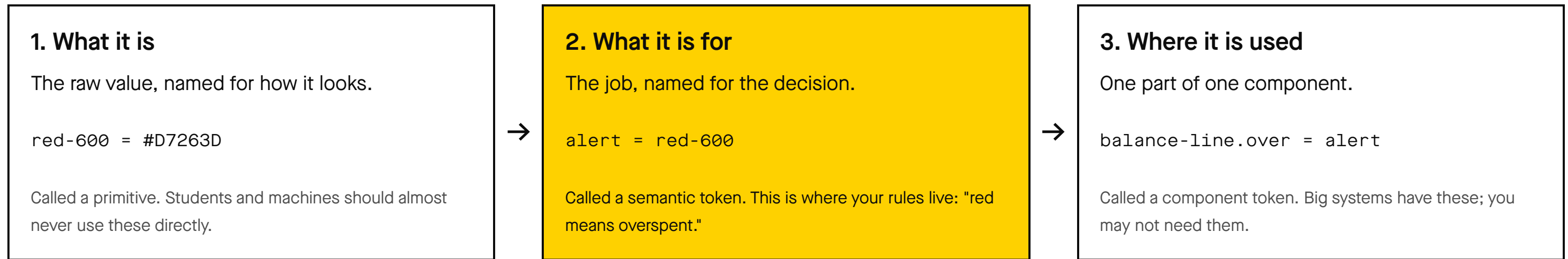
Tailwind and shadcn/ui: the tools most websites, and most AI tools, use to turn a design system into a real page.

From your decisions to a page: the four layers of a website today.



An AI tool works at every layer: it reads the tokens, writes Tailwind classes, and adds shadcn/ui parts. If layer 1 is empty, it fills it with defaults, and the page looks like everyone else's.

Every value gets three names: what it is, what it is for, and where it is used.



Change the brand's red: edit layer 1. Decide warnings should be orange: point layer 2 somewhere else.
Nothing in layer 3 or on any page has to change.

Tailwind: the way most websites, and most AI tools, write style today.

Tailwind is a free CSS framework. Instead of writing a separate style sheet, you put short class names on the HTML, and each class does one small thing.

```
<p class="text-lg font-semibold text-red-600">  
  $12 over  
</p>
```

CLASS	WHAT IT DOES
text-lg	Large type size
font-semibold	Semibold weight
text-red-600	Text in Tailwind's red, shade 600

Each class is an atom from atomic design. When you ask an AI tool for a web page, it usually writes it this way, and if nobody gives it your colors, it uses Tailwind's built-in ones. That is part of why so many AI-made sites look alike.

In Tailwind 4, your tokens go in one block called `@theme`.

Every variable you write there becomes a class you can use. Name them for their job, and the page says what it means.

THE TOKENS, IN YOUR CSS FILE

```
@theme {  
  /* 1. what it is */  
  --color-red-600: #D7263D;  
  --color-black: #0C0C0C;  
  --color-paper: #FFFFFF;  
  
  /* 2. what it is for */  
  --color-alert: var(--color-red-600);  
  --color-ink: var(--color-black);  
  --color-surface: var(--color-paper);  
}
```

THE PAGE, USING THEM

```
<!-- Wrong: says how it looks -->  
<span class="text-red-600">$12 over</span>  
  
<!-- Right: says what it means -->  
<span class="text-alert">$12 over</span>  
<main class="bg-surface text-ink">
```

Popular kits built on Tailwind, like shadcn/ui, use the same idea with names such as `bg-background`, `bg-primary`, and `text-destructive`.

shadcn/ui: parts you copy into your project, not a library you install.

Started in 2023 by a developer who goes by shadcn. You run one command, and the part's code lands in your own folder, where you can read it and change it.

```
$ npx shadcn@latest add button
```

```
✓ Created components/ui/button.tsx
```

EACH PART COMES WITH	WHAT THAT GIVES YOU
Behavior	Works with a keyboard and a screen reader, from a library called Radix
Style	Written in Tailwind classes you can read
Semantic colors	Every color is a job name, like primary, never a raw value
Variants	A short list of the versions allowed, and no others

shadcn/ui names every color for its job, in pairs.

THE VARIABLE	ITS JOB
background / foreground	The page, and the text on it
primary / primary-foreground	The main action, like Save, and the text on that button
secondary / secondary-foreground	A second, quieter action
muted / muted-foreground	Quiet areas, and less important text
accent / accent-foreground	Hover and highlight
destructive	Delete, and errors
border, input, ring	Lines, form fields, and the outline that shows keyboard focus
radius	How round the corners are

Every surface color has a -foreground partner, the color of the text that sits on it. A readable-text rule is built into the names.

A variant is a rule the code enforces.

This is the button file shadcn/ui puts in your project, shortened. It lists every kind of button allowed. There is no seventh kind: nobody makes a purple button without editing this file.

```
const buttonVariants = cva("rounded-md text-sm font-medium", {
  variants: {
    variant: {
      default: "bg-primary text-primary-foreground",
      destructive: "bg-destructive text-white",
      outline: "border bg-background",
      secondary: "bg-secondary text-secondary-foreground",
      ghost: "hover:bg-accent",
      link: "text-primary underline",
    },
    size: { default: "h-9 px-4", sm: "h-8 px-3", lg: "h-10 px-6", icon: "size-9" },
  },
})
```

READ IT AS RULES

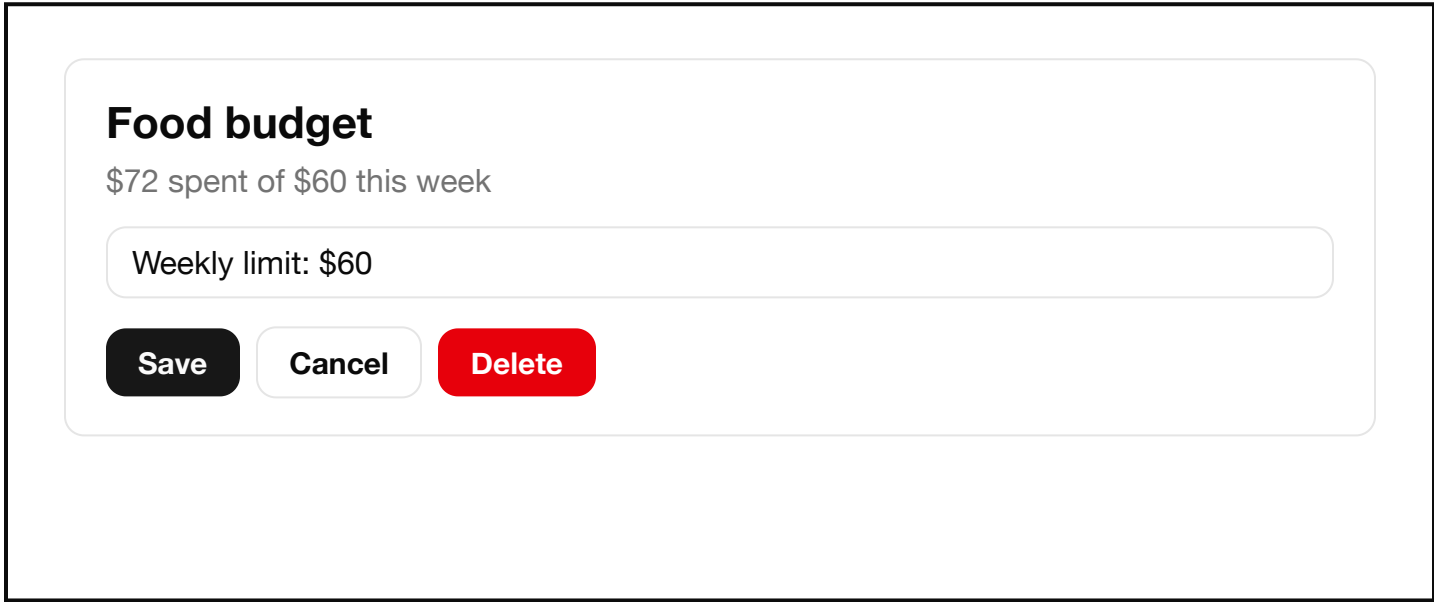
Six kinds of button, four sizes, and nothing else

Every color is a job name: primary, never blue-600

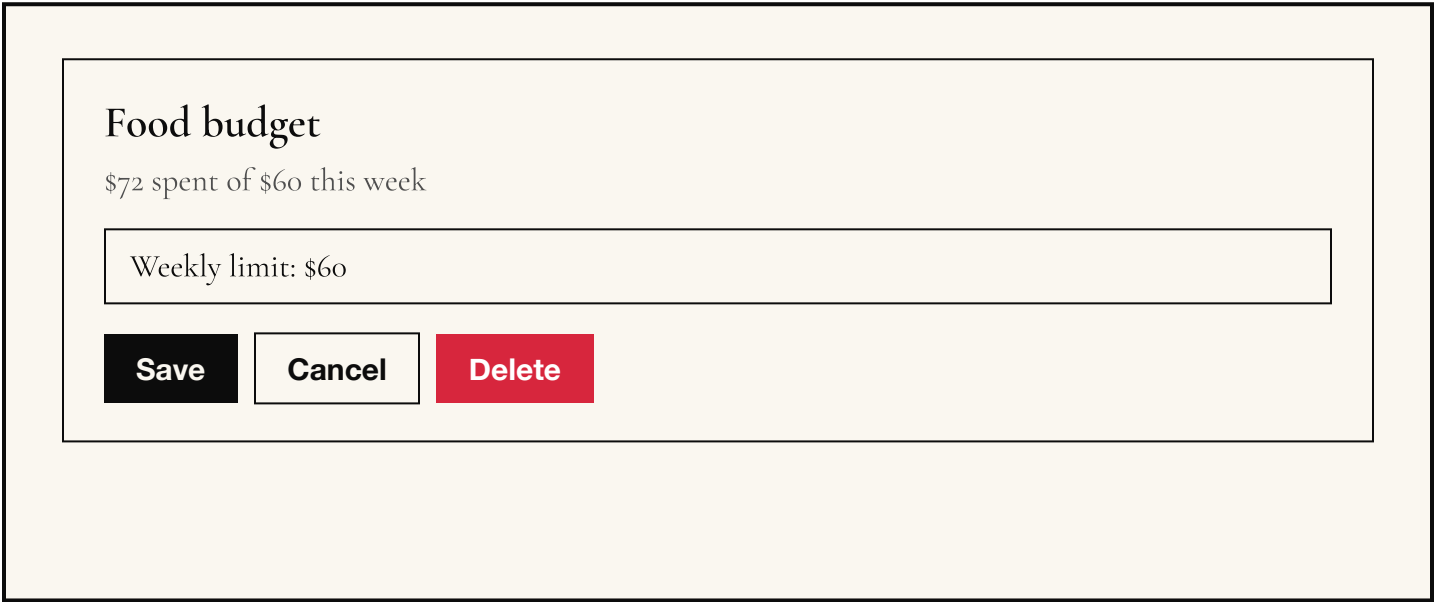
Your rules file says when each one is used: "destructive only for deleting"

Change the variables, and every part changes with them.

The same card, field, and three buttons, built from the same code. Only the variables are different.



shadcn/ui's default theme. Gray, rounded, Helvetica-like. Every new project starts here.



The budgeting app's theme. Paper, ink, square corners, the serif, and its red only for Delete.

How an AI tool builds with shadcn/ui today.

1 · READS

`components.json`
where parts go, and which style
the project uses

2 · ASKS

the registry what parts exist,
through shadcn's own MCP
server or its command line

3 · ADDS

the real code with `npx`
`shadcn add`, instead of
writing a button from memory

4 · STYLES

with the project's job names,
`bg-primary`, never raw colors

5 · CHECKS

that the project still builds, and
fixes what fails

"NEVER fetch raw files from GitHub manually."

From the instructions shadcn/ui ships to AI tools. It forces the tool to use the real parts instead of guessing at them: the "forbid it" and "check it" moves from the survey.

The catch: the same parts make the same websites.

Bootstrap made the web of 2014 look alike. Untouched shadcn/ui is doing it again, and AI tools reach for it by default.

Acme AI
Ship faster with AI.
[Get started](#)

Nimbus
The platform for modern teams.
[Get started](#)

Ledgerly
Budgeting, reimagined.
[Get started](#)

Orbit
All your work in one place.
[Get started](#)

shadcn/ui gives everyone the same parts. The only thing that makes them yours is layer 1: your tokens and your rules. That is what you start today and finish on Wednesday.

The same jobs, in your starter repository.

You do not need Tailwind or shadcn/ui for this class. The starter does each job in a plain file, so you can see it.

IN TAILWIND AND SHADCN/UI

IN YOUR STARTER REPOSITORY

Tokens in @theme, with job names

05-tokens/, with job names, starting Wednesday

Variants listed in the code

The **Do** line of each rule

components.json and the registry

AGENTS.md and the folder map in the README

shadcn/ui's skill for AI tools

The five skills in .claude/skills

The build check

node scripts/check.mjs

Who uses it

Most of the people who use a system work in marketing, sales, support, and the rest of the company.

Every part of a business makes something with the brand.

WHO	WHAT THEY MAKE	WHAT THE SYSTEM GIVES THEM
Marketing	Ads, social posts, email, the website	Campaigns that look like one brand, made in days instead of weeks
Sales	Decks, proposals, one-page summaries	On-brand work without waiting for a designer
Product and engineering	The app and the site	Parts built once and reused, so new screens cost less
Support	Help pages and replies to upset customers	Voice rules for the hard moments, when tone matters most
Hiring	Job posts and the first week for new people	The same voice inside the company as outside it
Legal	Trademarks and partner agreements	Rules for using the mark, which help the company protect it
Leadership	Announcements, reports to investors, the company's plans	A brand that stays recognizable as the company grows

Why a business pays for a system.

Speed

Nobody redesigns the button. A new page is made from parts that already work, so people spend their time on the problem in front of them.

Recognition

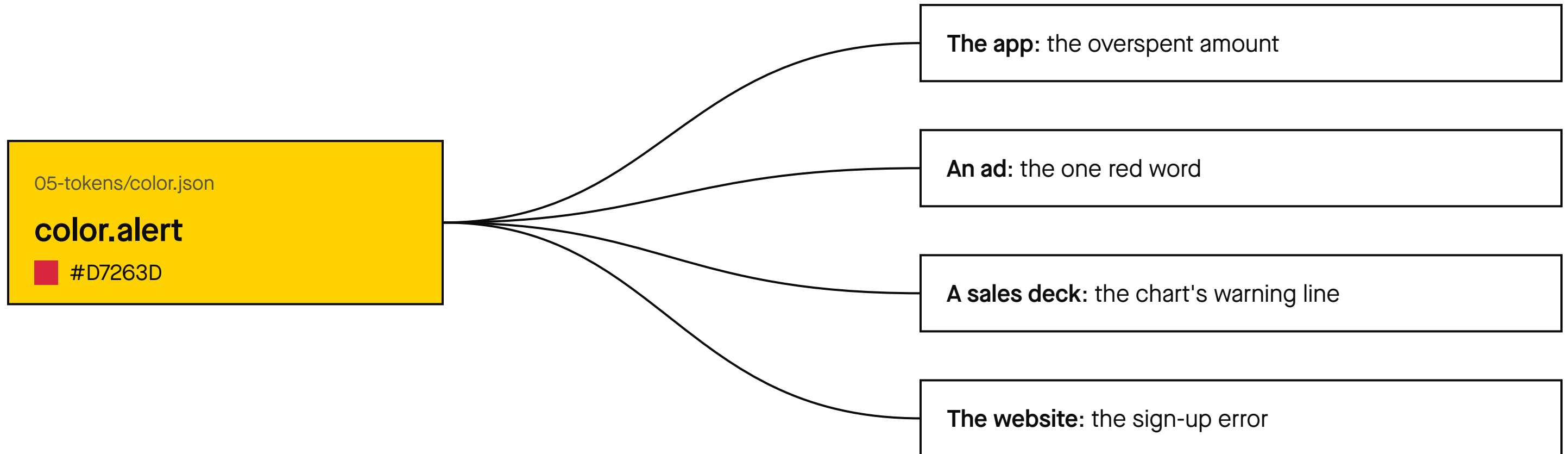
Neumeier says a brand is a person's gut feeling about a product. That feeling builds up each time the brand shows up the same way.

Change

When the red changes, it changes in one file, and every screen, ad, and deck that uses the red changes with it.

Neumeier's brand gap is the space between the strategy in the brief and the work people actually make. A system is how the strategy reaches everyone who makes something.

Change it in one place, and it changes everywhere.



The value lives in one file, and everything else points at its name. Wednesday you make this file for your own brand.

For a studio, the system is part of what the client buys.

A logo is one file. A system is what lets the client's own people keep using the brand after the studio leaves. That is why your Milestone 3 is the brand presented as a system, and Milestone 4 adds the brand book and a page built from it.

Nov 4

The brand, presented: the direction, the rules, the voice, the exact values.

Nov 16

The brand book, drafted: the system written for a person to read.

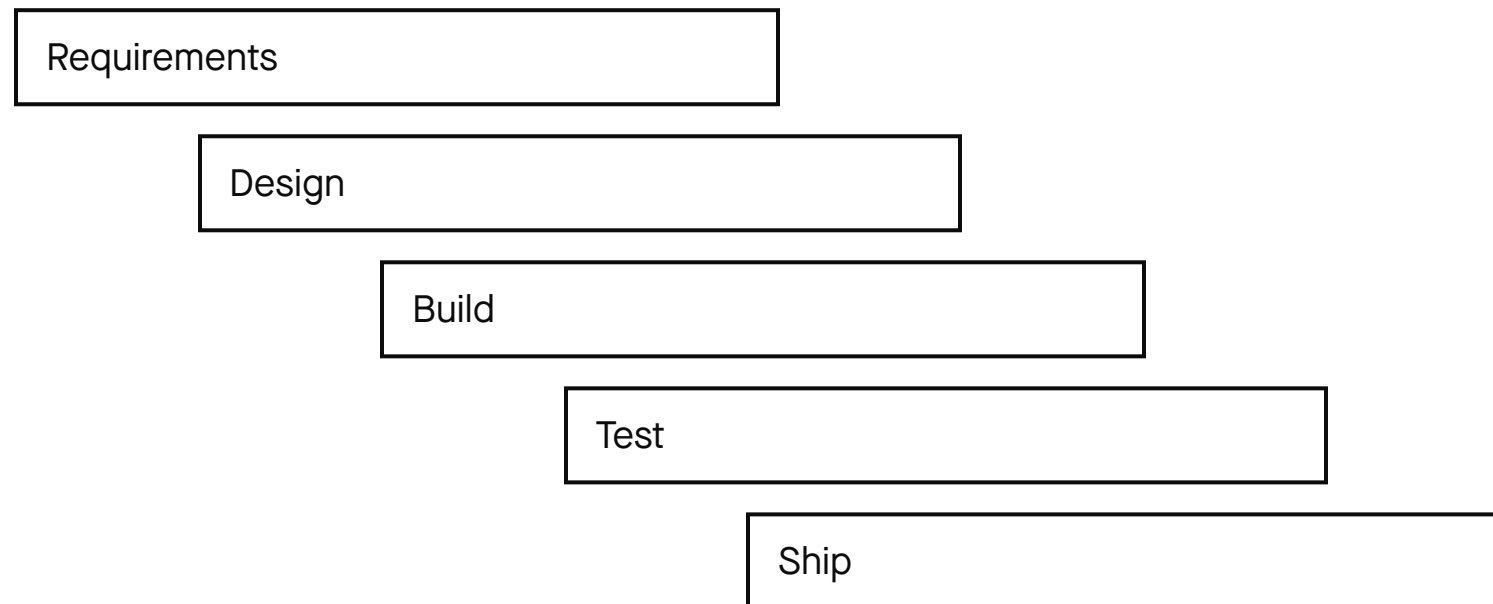
Dec 2

The landing page, built from the system, and presented with the book.

Made in loops

Teams make a brand in rounds, and show each round to someone before they start the next.

1970: the plan that only goes one way.



Winston Royce, an engineer, drew this chart in a 1970 paper on building large software. Each step finishes before the next one starts. People later called it the waterfall.

"I believe in this concept, but the implementation described above is risky and invites failure."

Royce's point was that you learn the most at the testing step, when it is the most expensive to go back. He drew arrows going backward.

Iterative design works in small loops, and each loop teaches you something before the next one.

Make

Something small enough to finish this week.

Show

Put it in front of people: the room, the client, a user, or a machine.

Learn

Write down what came back, including what surprised you.

Change

Fix the work, or fix the rule that produced it, and go again.

In 2001 a group of software developers wrote this down as the Agile Manifesto. One line of it:
"Responding to change over following a plan."

Bill Buxton separates sketches from prototypes.

Buxton is a designer and researcher. Sketches are quick and many, and you make them to explore. Prototypes are fewer and more finished, and each one tests a single idea.

HOW FINISHED	WHAT IT IS	WHAT IT CAN TELL YOU
A sketch	Pencil on paper, ten in an hour	Which ideas are worth one more hour
A wireframe	Boxes and words, no color	Whether people can find things
A clickable mockup	Screens linked together in Figma	Whether the steps make sense in order
A coded page	A real page built from the tokens	Whether the system holds up in the real thing

You have been working in loops all semester.

WHEN	WHAT YOU MADE	WHO IT WAS SHOWN TO, AND WHAT CAME BACK
Sept 21	The pitch, recorded	Me: comments on the problem and the audience
Sept 30	The creative brief	You, filling it in: the questions you could not answer yet
Oct 7 to 14	Moodboards and the updated deck	The room: what people asked, and what nobody asked
Today	One direction and the first rules	Me, at your desk
Oct 28	The rules, written for a machine	An AI model: whatever it gets wrong is your to-do list
Nov 4	The brand, presented	The room again
Nov 23 to Dec 2	The landing page and the book	The room, and anyone with the link

A system that changes needs a record of every change.

Teams give the system a version number, write down each change with its date and reason, and keep the old versions. Then anyone can see what changed, and why, and undo it.

A version number

Like 0.1.0. It goes up each time the system changes. The starter kit went to 0.2.0 this morning.

A decision record

What you chose, what else you looked at, and which part of the brief it answers. Your direction file, written today, is your first one.

A log

Two lines for each class you work: what you decided, and what the machine did. It goes in `LOG.md`.

The brand repository

The starter repository, read as a design system for one brand.

The starter repository is a design system kit for one brand, and it is public.

Workhorse made it for this class and released it on GitHub, so anyone can use it for a brand project. The folders, templates, and instructions can be reused by anyone who credits them. The work you put in your copy belongs to you.

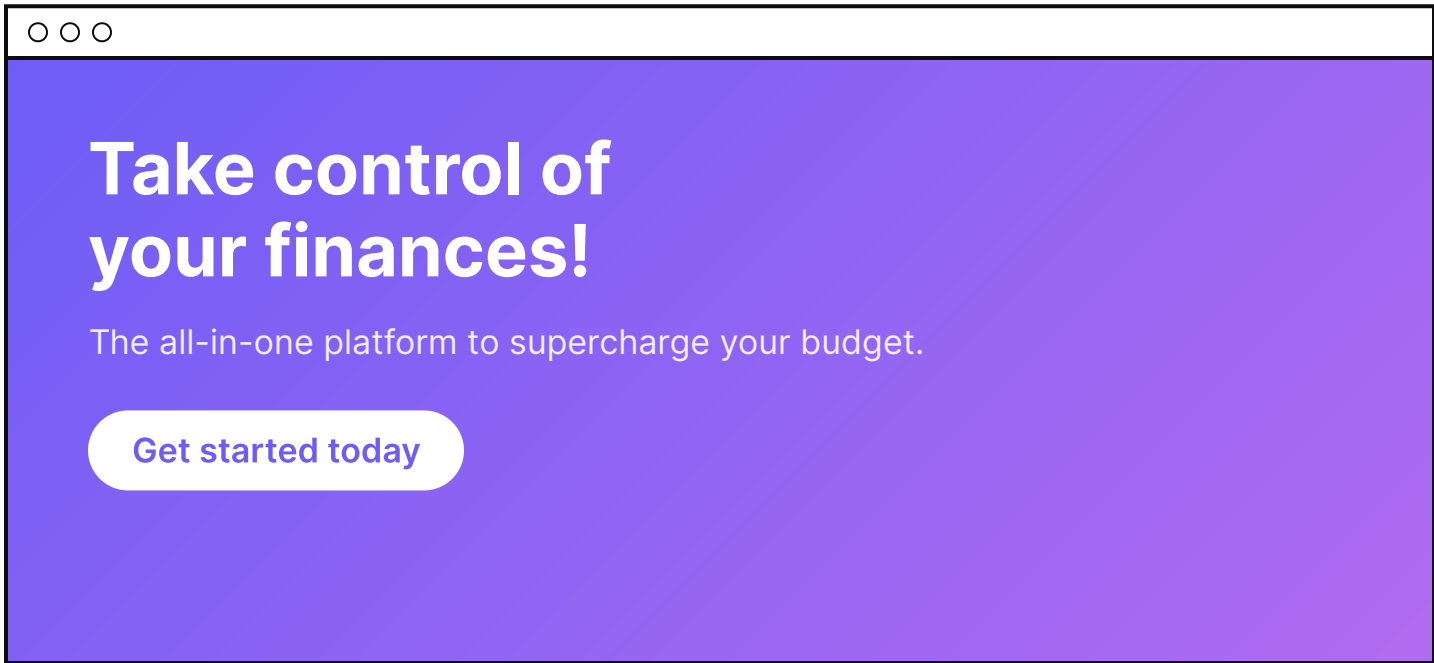
Read it the way you would read someone else's system: what does it ask you to decide, in what order, and who is it written for?

Brand first: the folders follow the order of the decisions.

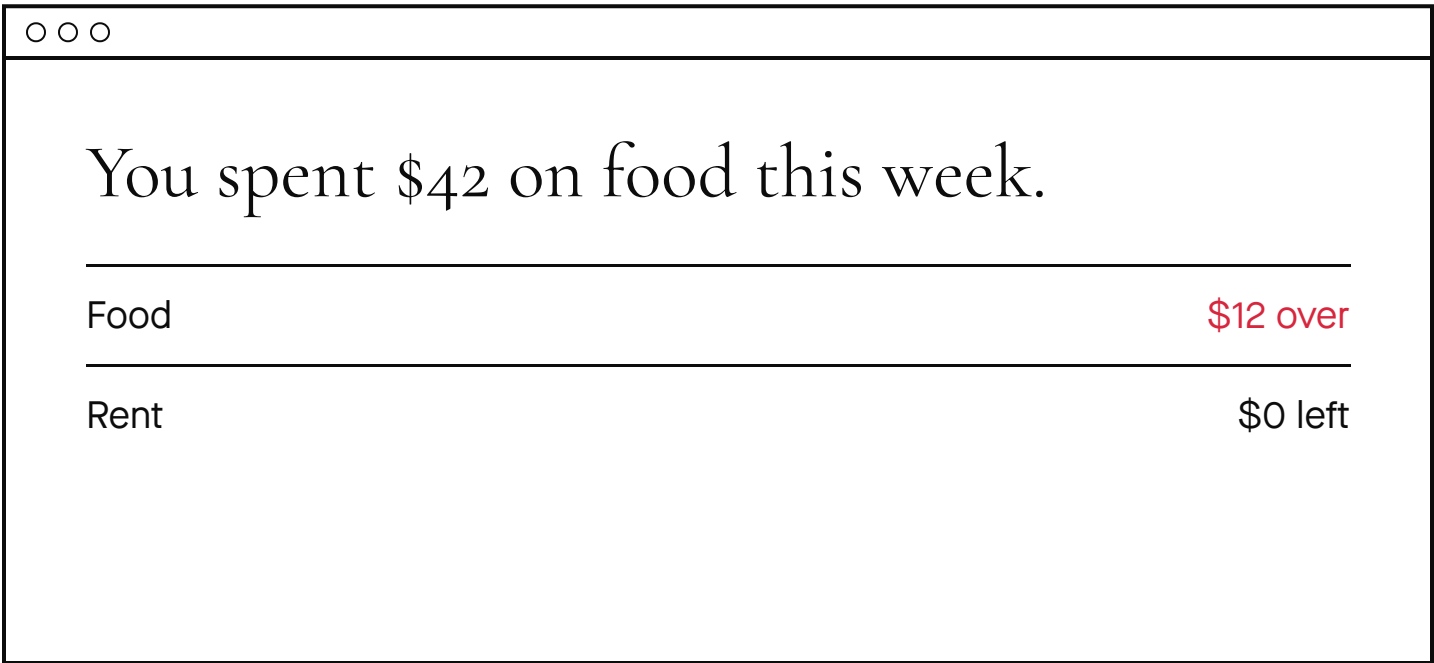
FOLDER	THE DECISION	WHEN
01-brief	Who the brand is for, and what it should make them think and feel	Sept 30
02-research	What is true about the audience and the competition	All semester
03-explorations	The possible brands: boards, themes, and language	Oct 7, Oct 14
04-brand	The one direction, the rules, the voice, and the mark	Today to Nov 4
05-tokens	The exact values	Oct 21
06-book	The system written for a person to read	Nov 16
07-site	A page built from all of the above	Nov 18 to Dec 2
decks	Not a decision: every PDF you present or turn in, named with the date first	Every crit

Without your rules, a machine makes the average.

AI models learn from millions of websites, so with nothing else to go on they choose what is most common. Anthropic calls this distributional convergence. Your rules are what move it off the average.



No rules. The most common choices: the Inter typeface, a purple gradient, a cheering headline.



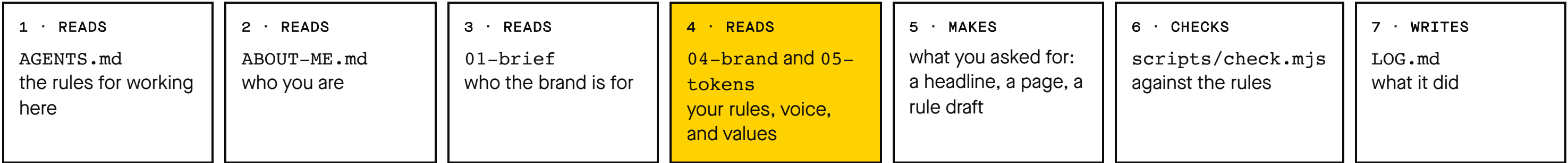
The budgeting app's rules attached. Black and paper white, one red for overspent, one plain sentence.

The repository is also written for an AI tool, so the tool follows your brand.

You don't need an AI tool today. If you use one (Claude Code, Cursor, Codex, Copilot, or Gemini), open the repository folder in it. Before it does anything, it reads **AGENTS.md**, its instructions, then **ABOUT-ME.md**, who you are, then your brief.

THE TOOL'S INSTRUCTIONS SAY	WHAT THAT PROTECTS
The brief is the source.	Every decision names the part of the brief it answers.
Do not decide for the student.	No invented research, quotes, customers, or sources.
Exact values live in the tokens.	No color gets typed into a page by hand.
Write in the brand's voice.	The tool reads the voice file before it writes a word.
Say what the machine did.	Every session the tool worked in ends with a log entry.

What an AI tool does in your repository, in order.



Step 4 is the part you write, starting today. If a rule is missing or vague, the tool falls back to the average: the purple gradient and the cheering headline.

The tool can suggest a change to the brand. Only you can make it.

When you correct the tool's writing the same way twice, it adds a line to `04-brand/voice-queue.md`. You read that file and decide what becomes a rule.

DATE	PROPOSED RULE	WHY	SOURCE
Oct 28	Never call the user "budgeter"; say "you"	Corrected twice in the welcome screens	The tool
Oct 29	Adopted into the voice rules, revision 2	The student agreed	The student

Five skills: steps the tool follows, written down once.

A skill is a short text file of instructions. In Claude Code you ask for one by name, or just describe the job: "help me write a rule for my color."

SKILL	WHAT IT DOES
Write rules	Asks you about one decision and writes your answers into the four parts of a rule. It never fills a part with its own guess.
Write in voice	Reads the voice rules before writing any copy for the brand
Check	Reads a draft for the habits that make writing sound machine-made, and flags them
Trace to the brief	Lists every decision in the work and the words in the brief it answers, and flags what answers nothing
Log the run	Writes the log entry: what was decided, and what the machine made, kept, and threw out

Four ways real design systems keep a machine on-brand.

WAY	WHAT IT MEANS	A REAL SYSTEM THAT DOES IT	IN YOUR REPOSITORY
Tell it	Instructions the tool reads before it starts	Ant Design keeps one instruction file that every AI tool reads	AGENTS.md
Show it	Examples of the right answer	Cloudscape, Amazon's system, publishes 181 example pages for tools to copy	The Right line in every rule
Forbid it	Say what never to do	HeroUI's instructions open: "STOP. What you remember is WRONG."	The Wrong line, and "never"
Check it	Test the result and send back what fails	Ant Design runs its checker after every change the tool makes	scripts/check.mjs, and you

The survey's finding: checking beats forbidding. A tool can ignore a "never." It cannot ignore a check that fails.

Name a value for its job, and the look can change later.

A preview of Wednesday. A token is one exact value with a name, kept in the **05–tokens** folder. Figma can read the same file.

color.action

The one call to action on a screen. If it changes from blue to red, the name still fits.

color.blue

Says what it looks like. The day it turns red, every page that uses it is wrong.

A description

Every token says what it is for: "The one call to action per view."

The repository checks itself.

One command reads the whole system and lists what breaks its own rules. You don't need it today: we run it together on Wednesday, and after that before every commit.

```
node scripts/check.mjs
```

Contrast

```
  ok    color.ink on color.surface: 17.40 to 1
```

Site

```
  FAIL  site.css types values by hand: #fff, 12px
```

Rules

```
  FAIL  "Red means overspent" points at
        color.alert, which is not in 05-tokens
```

WHAT IT CHECKS

Every token says what it is for

The CSS says the same thing as the token files

Text is readable on its ground

The site types no colors or sizes by hand

Every rule has Do, Because, Right, and Wrong

What makes a repository easy for someone else to pick up.

HABIT

IN THE STARTER REPOSITORY

Tell people where to start

The README has the steps and a map of every folder

Say what others may do with it

The license: reuse the kit with credit; your work stays yours

Use plain text

Every file is text or JSON, so any person or tool can open it

Keep one home for every value

Colors live only in the tokens, and the page reads them

Commit small, and say why

One change per commit, with a message that says what and why

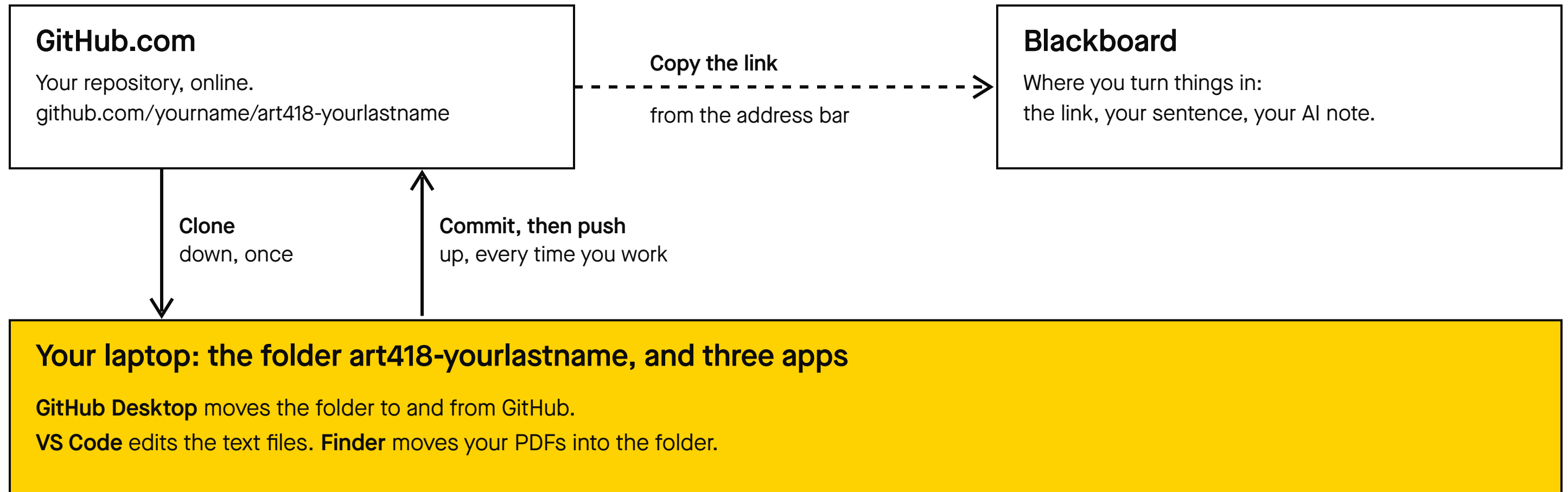
Keep receipts

The log, the direction file, and the voice queue

Your turn

Catch up your repository, choose one direction, and write your first rules.

Your work lives in three places. Here is how it moves.



Seven steps, one slide each. Do each step before the next slide, and raise your hand if what you see does not match the box on the right.

ON GITHUB.COM

Make your own copy of the starter

- 1 Sign in at **github.com**.
- 2 Go to **github.com/workhorse/brand-project-starter**
- 3 Click the green **Use this template** button, at the top right, next to **Code**. Choose **Create a new repository**.
- 4 Under **Repository name**, type art418- and your last name, all lowercase: art418-garcia
- 5 Click **Private**, then the green **Create repository** button at the bottom.

YOU SHOULD SEE

A page with **your** GitHub name at the top left:
yourname / art418-garcia

A **Private** label, and the folders 01-brief to 07-site.

IF IT GOES WRONG

No **Use this template** button? You are not signed in. Sign in and reload the page.

ON GITHUB.COM

Add me, so I can see it

- 1 On your new repository's page, click **Settings**, the last tab, with a gear.
- 2 In the left column, click **Collaborators**. GitHub may ask for your password.
- 3 Click the green **Add people** button.
- 4 Type thehonestape and click it in the list.
- 5 Click **Add thehonestape to this repository**.

YOU SHOULD SEE

thehonestape in the list, marked **Pending invite**.

That is right. I accept it on my end.

IN GITHUB DESKTOP

Put the repository on your laptop

- 1 Open **GitHub Desktop**. If it asks, sign in with your GitHub account.
- 2 In the menu bar at the top of the screen: **File › Clone Repository...**
- 3 Click the **GitHub.com** tab, then click `art418-garcia` in the list.
- 4 Look at **Local Path** and leave it as it is. It is usually `Documents/GitHub/art418-garcia`.
- 5 Click the blue **Clone** button.

YOU SHOULD SEE

No local changes in the middle of the window.

At the top left: **Current repository: art418-garcia**.

IF IT GOES WRONG

Your repository is not in the list? Click the small refresh arrow at the top of the list, or check you are signed in to the same account.

IN GITHUB DESKTOP

Open the folder

- 1 Menu bar: **Repository** › **Show in Finder**. A Finder window opens. Keep it open; you need it in step 5.
- 2 Menu bar: **Repository** › **Open in Visual Studio Code**.

YOU SHOULD SEE

In VS Code, the left column lists the same folders as GitHub: 01-brief, 02-research, and on down to decks.

IF IT GOES WRONG

No VS Code? Get it free at code.visualstudio.com.
Or open any .md file with TextEdit, then **Format** › **Make Plain Text**.

IN FINDER

Put your PDFs in the decks folder

- ① In the Finder window from step 4, double-click the decks folder.
- ② Drag in three PDFs: your creative brief, your moodboards, and your October 14 deck.
- ③ Rename each one, date first. Click the name once, press `Return`, type, and press `Return` again.

YOU SHOULD SEE

Three PDFs in decks, sorted by date, plus the README.md that was already there.

2026-09-30-creative-brief.pdf

2026-10-07-moodboards.pdf

2026-10-14-explorations.pdf

IN VS CODE

Paste your brief in as text

- 1 In VS Code's left column, click 01-brief, then `creative-brief.md`.
- 2 Open your brief beside it: the Google Doc or the PDF.
- 3 Under each heading in the file, paste your answer from your brief. Leave the headings as they are.
- 4 Save: `Cmd` + `S`.

YOU SHOULD SEE

Your words under each heading.

No white dot on the file's tab at the top. A dot means it is not saved yet.

IN GITHUB DESKTOP

Save a version, and send it up

- 1 Go back to GitHub Desktop. The left column lists every file you changed.
- 2 At the bottom left, in the **Summary** box, type: `Move work into the starter repository`
- 3 Click the blue **Commit to main** button.
- 4 Click **Push origin** at the top of the window.

YOU SHOULD SEE

GitHub Desktop says **No local changes** again.

On github.com, reload your repository page: your PDFs are in decks.

IF IT GOES WRONG

The button says **Publish branch** or **Fetch origin** instead? Click it; it does the same job.

Choose one direction: hold each board up to your brief.

- 1 Reread your brief: who the brand is for, and what they should think and feel.
- 2 Hold each board up to those words. Which one answers the most of them?
- 3 Read your notes from Oct 14. What did the room ask, and what did it miss?
- 4 Pick one, and write it in one sentence.
- 5 For each board you did not pick, write down what it would have cost you.

The invented budgeting app, choosing between three boards.

BOARD	WHAT IT LOOKS LIKE	HELD UP TO THE BRIEF
The coach	Bright green, progress rings, and messages that cheer you on	It looks like every other budgeting app, and the brief says that tone makes students feel judged.
The ledger	Black and paper white, plain numbers, and no cheering	It answers "calm" and "plain" in the brief, and it talks like a friend who is good with money.
The arcade	Game colors, points, and levels for saving	Fun the first week, but it makes money feel like a game, and the brief asks for trust.

The direction in one sentence: a budgeting app in black and paper white that tells students where their money went, plainly, and never cheers.

Then turn it into rules. Every rule has four parts.

Today, write all your rules under Judgment rules in `04-brand/rules.md`, in words. On Wednesday your color and type rules get a token and move up to Exact rules.

PART	WHAT IT SAYS	THE INVENTED BUDGETING APP
Do	What to do, in one sentence	Use black and paper white on every screen. The only other color is one red, for money you have overspent.
Because	The decision or the part of the brief it came from	The direction: calm and plain. The brief: students feel judged by cheerful green apps.
Right	One example that follows it	A balance in black on paper white, with the overspent amount in red.
Wrong	One example that breaks it	A green progress ring with confetti when you save.

"Keep it friendly" is still a wish.

Nobody could tell you whether they followed it. Here is the same idea written so someone could.

Do	Say what happened in one plain sentence. Never use an exclamation mark.
Because	The direction: it talks like a friend who is good with money.
Right	"You spent \$42 on food this week."
Wrong	"Whoa! Big week for snacks!"

Three questions for every rule: could someone who never met you follow it, could you tell if they broke it, and does it name where it came from?

IN VS CODE · 04-BRAND/DIRECTION.MD

Write your direction in the file

- 1 In the left column, click 04-brand, then `direction.md`.
- 2 Replace every part in parentheses with your answer, and delete the parentheses. The next slide shows a finished one.
- 3 Under each heading with a dash, write one line after the dash. Add a dash for each extra line.
- 4 In the table, write one row for each board you did not choose.
- 5 Save: `Cmd` + `S`.

YOU SHOULD SEE

No parentheses left in the file.

Your direction in one sentence, the brief's words it answers, and one row per board you left behind.

04-BRAND/DIRECTION.MD

Before, and after it is filled in

WHAT YOU FIND IN THE FILE

```
**Chosen on:** (date)

**The direction in one sentence:** (if it takes
more than one breath to say, it is two directions)

**Which board or theme it comes from:** (its name)

## Which words of the brief it answers best
-

## The alternatives, and what they would have cost
| Board or theme | What it would have cost |
|---|---|
|   |   |
```

THE INVENTED BUDGETING APP, FILLED IN

```
**Chosen on:** 2026-10-19

**The direction in one sentence:** A budgeting
app in black and paper white that tells students
where their money went, plainly, and never cheers.

**Which board or theme it comes from:** The ledger

## Which words of the brief it answers best
- Brief 5, Insight: students feel judged by
cheerful, colorful budgeting apps.

## The alternatives, and what they would have cost
| Board or theme | What it would have cost |
|---|---|
| The coach | Looks like every other app |
| The arcade | Makes money a game; the brief asks for trust |
```

IN VS CODE · 04-BRAND/RULES.MD

Write three rules in the file

- 1 Click 04-brand, then rules.md. Scroll down to the heading `## Judgment rules`.
- 2 Select the five lines from `### (Rule name)` down to `- **Wrong:**`. Copy them, and paste them twice under the first, so you have three empty rules.
- 3 Replace `(Rule name)` with a short name, and write after each colon. Start with color, type, and voice.
- 4 Leave `## Exact rules` empty today. Your color and type rules move there Wednesday, when they get a token.
- 5 Save: `Cmd` + `S`.

YOU SHOULD SEE

Three rules under `## Judgment rules`, each with Do, Because, Right, and Wrong filled in.

The Examples at the bottom of the file show two finished rules. Read them, then delete that section.

04-BRAND/RULES.MD

One empty rule, and the same rule filled in

WHAT YOU COPY

```
### (Rule name)
```

- **Do:**
- **Because:**
- **Right:**
- **Wrong:**

THE INVENTED BUDGETING APP, FILLED IN

```
### Black and white, and one red
```

- **Do:** Use black and paper white on every screen. The only other color is one red, for money you have overspent.
- **Because:** The direction: calm and plain. Brief 5: students feel judged by cheerful green apps.
- **Right:** A balance in black on paper white, with the overspent amount in red.
- **Wrong:** A green progress ring with confetti when you save.

IN VS CODE · LOG.MD

Write today in the log

- 1 Click LOG.md, near the bottom of the left column.
- 2 Scroll to the line --- near the top. Click on the empty line under it.
- 3 Type these three lines, with your own words after the colons:

```
## 2026-10-19  
Decided: my direction is the ledger, plain and calm.  
Machine: none.
```
- 4 If you used an AI tool, the Machine line says which tool, what it made, and what you kept.
- 5 Save: `Cmd` + `S`.

YOU SHOULD SEE

Today's date as a heading, a **Decided** line, and a **Machine** line, under the ---.

Next time, the newest entry goes on top.

IN GITHUB DESKTOP

Save a version, and send it up again

- 1 In GitHub Desktop, check the left column lists `direction.md`, `rules.md`, and `LOG.md`.
- 2 In the **Summary** box, type: `Direction` and `first rules`
- 3 Click **Commit to main**.
- 4 Click **Push origin**.

YOU SHOULD SEE

No local changes.

On github.com, open `04-brand/rules.md`: your three rules are there.

ON BLACKBOARD

Turn in today's checkpoint

- 1 On github.com, on your repository's page, copy the address from the browser's address bar.
- 2 On Blackboard, open ART 418, then **Course Content**, then **Checkpoint: The direction and the first rules**.
- 3 In the text box, paste three things: the address, your direction sentence, and your AI note.
The AI note: which tool you used and for what, or "I did not use AI today."
- 4 Click **Submit** by 2:50.

YOU SHOULD SEE

Blackboard shows the checkpoint as **Submitted**.

Today's checkpoint: what I look for.

IN YOUR REPOSITORY

WHAT IT HOLDS

decks/	The brief, the moodboards, and the Oct 14 deck as PDFs, date first: 2026-10-14-explorations.pdf
01-brief/creative-brief.md	Your brief's answers as text
04-brand/direction.md	Your direction in one sentence, the brief's words it answers, and what the others would have cost
04-brand/rules.md	At least three rules under Judgment rules, each with Do, Because, Right, and Wrong
LOG.md	An entry for today: Decided, and Machine

Blackboard	"Checkpoint: The direction and the first rules": your repository link and your direction sentence in the text box, and the AI note, by 2:50 today
-------------------	---

Before Wednesday.

WHAT	WHAT TO DO	DUE
Read	Davis, <i>Creative Strategy and the Business of Design</i> , chapters 9 and 11: leading the client, and aligning the target, the channel, and the message. About 20 pages.	Oct 21
Watch	Brad Frost, <i>Atomic Design</i> , about 30 minutes, at vimeo.com/109130093 : today's atoms and molecules, from the person who named them.	Oct 21
Bring	Your laptop and your repository. On Wednesday we take a brand apart into exact values for color, type, and spacing, and you do the same to yours.	Oct 21

The next three weeks, up to Milestone 3.

CLASS	WHAT WE DO	READ OR WATCH BEFORE CLASS	DUE
Wed Oct 21	Tokens and rules. We take a brand apart into exact values for color, type, and spacing, and you do the same to yours.	Davis, chapters 9 and 11, about 20 pages. Brad Frost, <i>Atomic Design</i> , video, 30 minutes.	Nothing turned in
Mon Oct 26	Voice and language. How the brand sounds, what it is called, and the rules for its writing.	Eli Altman, <i>Don't Call It That</i> , the opening sections through the first naming exercise, 24 pages.	Nothing turned in
Wed Oct 28	Testing the system with AI. You give your rules to an AI model with nothing else attached, and read what comes back.	Ted Chiang, "Why A.I. Isn't Going to Make Art," <i>The New Yorker</i> , about 25 minutes.	Checkpoint: the rules, written for a machine
Mon Nov 2	System review. The whole system read as one thing, then studio time on the brand.	Ian Silber on <i>Lenny's Podcast</i> , video, 55 minutes. John Berger, <i>Ways of Seeing</i> , episode 4, 30 minutes.	Checkpoint: the system, reviewed, in class, with one skill you wrote
Wed Nov 4	Milestone 3: the brand, presented. Boards up before class starts, then three minutes each.	Nothing new	A mark, a palette, the type, the voice, and three applications, with your deck rebuilt

Close

Choose one brand.
Then write down how to make it.

"The NASA Logotype appears in NASA Gray so as not to conflict with the red of the American flag." NASA, 1975.