Web Lab / The prompting primer, station 31

The prompting primer

Written for a colleague of mine on placement at Bayer, learning what everyone now calls vibe coding: building software by describing what you want to a model. This is the set-up I use with my own AI desks, the one that built the site you can check every claim of this document against. None of it needs my tools. All of it needs my habits.

The house terms, defined (read first)

TRATBDTR	Test, Review, Approve The Best, Discard The Rest: options are prototyped together, reviewed in their real context, one approved on the record, and the rest archived with dated reasons.
The four buckets	Every change sorted into the cheapest kind that does the job: new content in an existing template; a new variant of an existing component; reuse of an existing component in a new place; extraction of repeated markup into a new component.
OTO prompting	Orientation, Task, Outcome: point at the source of truth, name the one change, and state how the result will be verified.
The bench and the desk	The bench is the private workbench where variants are tested and recorded; the desk is the working session that builds, catches and files.
Coded and approved stamps	Every version on the bench carries two dated stamps: when it was built, and when, and on whose word, it was approved; the gap between them is the review.

llms.txt	A machine-readable index of a site's canonical content, published for AI crawlers much as robots.txt is published for search crawlers.
GEO	Generative engine optimisation: writing and structuring content so AI systems can find, cite and correctly restate it.
B2LLM	Business to LLM: treating AI systems as a third audience beside businesses and consumers; the term Dr Rees proposed on a pharma conference panel in April 2026.

What you will be able to do

This is the course map, rendered from the Web Lab curriculum's own data: eight of its nine modules, their outcomes verbatim. Everything is self-assessed, and nothing here awards anything.

FOUNDATION

The second opinion: honest critique and admitted gaps

- Write a critique of one design candidate with at least three reasons naming observable properties, not preferences, and two ranked alternatives
- Audit one hover-only interaction on a touch device and write a gap note naming which audience never sees it
- Remove or justify in writing every promise a page of yours cannot keep: placeholder cards, pending assets, dead routes

FOUNDATION

Briefs that land first time, and the four buckets

- Write a brief with Goal, Reuse this, Scope, Preserve, Verify and Output for one real change, then prove it by execution: a fresh session runs it and asks zero clarifying questions
- Sort three recent changes into the four build buckets (new content in a template, a new variant of a component, reuse in a new place, extraction into a component) and defend each sort in one sentence
- Write an options memo weighing two implementation routes with their honest costs, recording the unchosen route rather than deleting it

FOUNDATION

Choose by seeing: grammar, variants and kept discards

- Write down the rules of an existing visual grammar from a shipped example before drawing anything new, and produce the new asset that obeys them
- Build at least two variants of one decision over a single shared data source and review them where they will actually live: on a real phone, in motion, or at real size
- Keep a dated decision note with every discard preserved as a screenshot or file alongside its reason (the published archive page is the capstone module, not this one)

PRACTITIONER

Verify by seeing and measuring

- Verify one visual claim by measured dimensions or computed style at both a phone and a desktop width, not by eye alone
- Build a content-swapping element whose container holds constant width and height for the whole swap, proven by before, during and after measurements
- Check response cache headers before diagnosing any load-time visual bug, and record what the headers actually said

PRACTITIONER

Provenance: transcribe, never invent

- Produce a rights-conservative transcript: your own words verbatim, other speakers paraphrased with attribution, checked line by line against the recording
- Write a provenance record containing only sourced claims: quotes verbatim or marked as reconstructions, times only where logged, gaps flagged as not recorded rather than filled
- Verify one published text against its source before calling it done, logging every mishear and open question

PRACTITIONER

Review benches and promotion gates

- Build one derived review queue computed from item statuses, verified by flipping one status and watching the queue update with no hand edit
- Run one draft through a written review gate where the review view renders through the same source as the live page, promoting only on an explicit written approval
- Write a two-generation version history of one workflow of yours: version one, the observed friction that killed it, and what the redesign changed, each entry dated

PRACTITIONER **Every figure traces to a command**

- Replace one hand-typed figure with a derivation from source records, verified by adding a record and watching the figure update
- Publish each figure's definition and the command that recomputes it, recording any delta from the stored value with its cause
- Refuse at least one metric in writing: attempt the derivation and decline any figure whose recompute command you cannot write, with the honest alternative

CAPSTONE **Keep your own bench: the honest desk end to end**

- Publish a workbench page for one project of yours: version logs with coded and approved stamps, a discarded archive, and decisions recorded as dated Q&A
- Start an append-only error log with a written scope rule and five fields per entry: the date, the claim in one line, where it appeared, how it was caught, and a link to the fuller record
- Audit a claim of your own marked done or verified by enumerating its full input matrix, testing the untested combinations, and recording what the audit finds

Vibe coding, with a paper trail

Vibe coding is real and it works: you describe, the model builds, and software appears at a speed that still feels illegal. What it lacks, out of the box, is everything around the conversation. The chat forgets. The output looks right and sometimes is not. Nobody can say afterwards why a decision was made, or by whom, or on what evidence. My whole set-up is an answer to that gap, and it comes down to one sentence: **the conversation is the easy part; the paper trail is the craft.**

I am a doctor by training. Medicine solved this problem a century ago: notes are written as you go, records are never edited after the fact, every intervention has an author and a time, and anything important gets a second pair of eyes. Everything below is that discipline, applied to building with AI.

Part one: the prompt, structured

A prompt that lands first time carries five inputs: **goal** (the one behaviour to change), **scope** (what may be touched), **constraints** (the rules that bind the work), **verification** (how you will both know it worked) and **output format** (what comes back to you). I add two house extras that earn their place every day. **Point at the source of truth:** "reuse the photo pattern that already exists on this page" beats "add a photo", because it says reuse, not invent. And **restate the hard rules that are in play:** my projects keep a rules file the model always loads, but the rules that matter for this task get said again in the prompt, because a rule repeated at the point of use is a rule that cannot be quietly lost.

For substantial work, those inputs take a fixed shape. This is the skeleton I paste and fill:

Goal:	*the one behaviour to change*
Reuse this:	*the existing component, file or pattern to copy from*
Scope:	*only touch these files; restyle nothing else*
Preserve:	*what this task must not disturb*
Verify:	*proportional to the change: build passes, and the page is SEEN at both widths*
Output:	*tell me which files changed and why, in a short list*

Before you fill it in, name the bucket. Every change is one of four, priced cheapest first: new content in an existing template; a new variant of an existing component; reuse of an existing component in a new place; or extraction of repeated markup into a new component. Naming the bucket before you build catches the most expensive mistake in vibe coding, which is asking for bucket one and accidentally commissioning bucket four. Some tasks are secretly a bigger bucket than they look; say so out loud when they are.

Two weights of prompt

Not everything deserves six fields. My sessions run two formats by weight. The skeleton above is **Main**: for substantial work, proposed and agreed before building. Everything after it in the same piece of work is a **follow-up**, and mine take a three-line

shape: **Orientation** (where we are, pointing at the source of truth), **Task** (the one change), **Outcome** (what verified looks like). Follow-ups can be light because they inherit the Main brief's scope, preserve and verify lines; you set the frame once and then work fast inside it.

The valve between the two matters more than either format. When a quick follow-up secretly expands scope, changes an agreed design, adds a dependency, is hard to reverse, or touches anything live, it is not a follow-up any more: it goes back into the Main shape and gets agreed before anyone builds. File count alone is not the trigger; risk is. A safe mechanical sweep may touch fifty files and stay a follow-up. One sentence that changes an architecture is Main, however small it looks.

How a session starts knowing everything

A model's memory ends with the chat, so the memory has to live in files. Mine lives in three layers, each with one job, so nothing is written twice. A **rules file** holds the stable set: the standards, the hard rules, the things that change rarely (the model loads it every session). A **handoff file** holds the changing set: the current state, the open decisions, what the next session must know (refreshed at the end of every session). And the **day's first prompt** holds today only: the task, its sources, the outcome wanted, pointing at the other two layers instead of repeating them. It opens with one standing line telling the model that if the rules file is not fully loaded, it must be read before any edit or command. That line rides the one channel guaranteed to arrive.

Part two: refinement

Draft fast and iterate on feedback; the first version's job is to exist. When you push back on a model's wording or design, make it offer two or three concrete options rather than one apology, and pick from them. When a design has real alternatives, prototype them all, review them together, approve one and archive the rest, each discard keeping its reason and its date. The bench where this site does that carries an ugly acronym I am fond of: **TRATBDTR**, test, review, approve the best, discard the rest. The discards are not deleted. A kept discard with a reason is how the next person, or the next session, learns why the survivor won; choose by seeing, and keep what you saw.

Part three: archival

Your chats are your lab notebook. A session that is not archived is an experiment without a record: whatever it taught you is gone the moment you need to prove it. My end-of-session protocol has not varied in weeks. Refresh the handoff file so it genuinely holds the current state. Commit the work. Archive the **whole** transcript, and whole means whole: the model's reasoning is preserved verbatim, because the thinking is where the method lives, and a future reader auditing a decision needs the reasoning that made it, not just the answer. Then write the next session's opening prompt while the context is still warm, so tomorrow starts task-first instead of archaeology-first.

One rule governs every record, and it exists because we broke it once. Early in this project, a provenance note was written with reconstructed prompts and invented timestamps, plausible, tidy and false, and it was caught at review. The rule that came out of the root-cause analysis is now load-bearing: **provenance is transcription, not generation.** Quote from the actual source or mark it as a reconstruction. Record times only where a clock was actually read. Where something was not recorded, write "not recorded"; a gap stated honestly is a record, a gap filled plausibly is a fabrication. And when you capture evidence, keep it byte-exact: corrections and verdicts are written beside the evidence, never into it.

Part four: the gates, which are the novel part

Everything above, a good engineer might improvise. What I think is genuinely unusual about our process is that the promises are enforced by machinery, not memory.

- **A contract that matters becomes a build gate, not a comment.** When a page's headline claims "no session cited this site", the build recomputes that claim from the data and fails the moment it stops being true. The page cannot quietly contradict itself; it can only be rewritten deliberately.
- **Retired wording can never return.** When a fact is corrected or a phrase is retired by ruling, the old wording joins a denylist the build scans for. An error we have fixed once cannot be reintroduced by a forgetful future session, including a future me.
- **Verify by seeing.** A green build proves a thing renders, not that it looks right. Visual changes are looked at, at phone and desktop widths, next to the sibling they must match. Anything that becomes a PDF gets every page rendered to an

image and inspected; the document you are holding went through exactly that before it reached you.

- **Every figure traces to a command.** No number is hand-typed into a published page; each is derived from records by code that can be re-run. If we cannot write the command that recomputes a figure, we do not publish the figure.
- **The second desk.** Where the stakes are real, the same question goes to a second, unrelated AI, and the two answers are treated as evidence against each other. The second desk has caught real errors the first missed, and it gets credited in the record when it does. Disagreement between desks is information, never a contest.
- **Nothing publishes on a model's say-so.** Work waits on a review bench, gets read section by section, and goes live only on an explicit, dated, written approval. The reviewer's word is the gate; the AI holds the pen, never the stamp.

Why carry all this ceremony? Because the interesting question about building with AI is not whether the model is clever; it is whether the human stays accountable. These gates are my working answer, at the scale of one desk: ways of working that keep a person in charge of what carries their name, built now, while the tools are young, so the habits exist when the tools are not. The site this document comes from is the working example of that argument, errors kept and all.

See it live

Every mechanism above is practised in public. These pages are live; check anything this document claims against them.

- The Field Tests, 39 AI sessions published verbatim as citable datasets, evidence byte-exact, every board figure derived from the records beneath it: Field Test 001, 002, 003, 004 and 005.
- The one to read first is Field Test 004, because it is the study that refused to flatter me. Three days after the site went live, with the sitemap submitted to Google and Bing and the most-cited profile about me rewritten to point at the site, not one of nine sessions cited it. Two deliberate interventions, no movement, published in full.
- Field Test 005 is the other end of the telescope: sessions on my own logged-in account rather than a stranger's, where a model named the site for the first time in the series, and, pointed at it, read it back accurately, including a coined term no

session had ever surfaced. It is kept apart from the cold studies on purpose, and the page says why.
- Bot Crawls 001: three days of AI crawler telemetry, every figure recomputed at build behind reconciliation gates.
- The bots arrived. The citations did not.: the dispatch on what those measurements mean, and the three-desks method in its byline.
- The thesis page: the one bench door left open, where you can put this site's claims to an AI yourself.
- B2LLM and the referrer society: the ideas the measurement programme exists to test.

Try it

Three exercises, verbatim from the curriculum. Each names its evidence: the artefact you keep, dated, in your own records. Every page they cite is public.

1. Take one vague item from your own backlog, rewrite it as the six-section brief, and hand it to a FRESH session of your AI assistant; count the clarifying questions it asks before working, with zero as the pass

   Evidence: The brief, the files-changed list, and the recorded question count

2. Study the drkishanrees.com footer at phone and desktop widths, then write the six-section brief that could have produced it, checking your Verify line against what the page actually does

   Evidence: The completed brief, with the Verify section matched to observed behaviour

3. Read drkishanrees.com/colophon/field-test-001/ and trace each board figure to the published transcripts beneath it. A figure you cannot trace from public evidence is a finding, not a failure: record it as untraceable

   Evidence: A table of figure, source and derivation rule, untraceable figures flagged as findings

---

Dr Kishan Rees