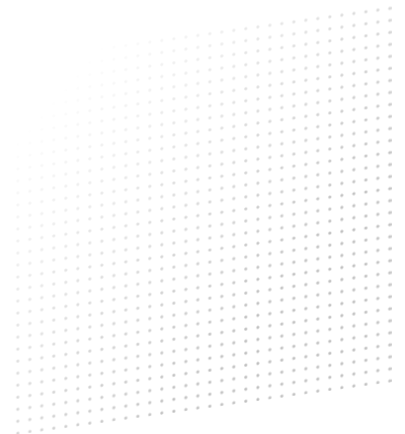

A RESOURCE FOR DESIGN SYSTEM AND DEVELOPER EXPERIENCE TEAMS

Time to first component benchmark.



Watch five new developers try your system. Time it properly.

Same tasks. Same clock. Fix the top three.

25

checks, each with its evidence

02

tasks: hello component, first pull request

05

participants per run, new to the system

The first half hour decides adoption.

API teams have measured "time to first hello world" for more than a decade: how long a developer takes, from nothing, to get working code out of an API [S01]. Postman calls time to first call the most important API metric [S02]. The essay behind this guide makes the same case for design systems: if a new developer ships a first component in under 30 minutes they keep using the system, and if it takes hours they build custom UI.

A benchmark only helps if the next run is comparable with the last. This guide fixes the parts that drift: what counts as start and stop, who takes part, the environment, what the facilitator may say, how unfinished sessions count, and which statistics to report on five people.

The kit: the protocol, two task definitions, a timing sheet, eleven friction codes and a zero-dependency script that validates a sheet, reports each task and compares two runs. The two sample runs are synthetic; every number quoted is the script's output on them.

Version 1.0 / Sources checked 24 September 2026

Field Guide 17 of the Design × AI series. Pairs with Field Guides 16 (AARRR funnel, where this is the activation diagnostic) and 18 (component usage Slack bot); all three use Acme and its `@acme/ui` library. Verified 24 September 2026 with Node 22.22 and the SQLite 3.51.0 CLI.

Practical guidance, not a standard. Five participants give a direction and a list of friction, not a precise estimate. The sample runs are invented to show the method. The 30-minute target comes from the essay Growth loops for design system adoption. Prepared with AI assistance and edited by hand.

START HERE

Two tasks, one clock.

T1 isolates finding, installing and configuring. T2 is the essay's task. Together they separate setup friction from build friction.

	T1 Hello component	T2 Login form PR
API analogue [S01]	Time to first hello world	Time to first useful app
Card	Show an <code>@acme/ui</code> Button labelled Save on the home page.	Build a login form with the system and open a pull request.
Start	Card read; participant says start or presses Start	Same
Stop	The Button renders, no errors in terminal or console	Pull request opened, lint and typecheck passing locally
Timebox	60 minutes	120 minutes, the essay's broken-onboarding line
Target	None until the first baseline	Median of 30 minutes (essay)

Full definitions and done criteria in `tasks.json`; the session rules in `protocol.md`.

First run

Sections 01 to 03. Book 5 participants and a clean container; print the timing sheet.

Analysing a run

Section 04. Run `ttfc.mjs` on the sheet and report the table as printed.

Proving a fix worked

Section 05. Same protocol version, new participants, `--baseline` on the old sheet.

Already tracking the funnel

Field Guide 16's ACT-3 is field time in days; this is task time in minutes. Report both.

Label	Meaning
DATA	A field in a kit file: <code>tasks.json</code> , the timing sheet or <code>friction-codes.json</code> .
SCRIPT	Computed or enforced by <code>ttfc.mjs</code> .
PROTOCOL	How a session is run, so runs stay comparable.
PRIVACY	Protects participants. The benchmark tests the system, not the person.
LOOP	Feeds the result back into the next run or the funnel.
PRACTICE	A working method with a review signal, not a hard gate.

SECTION 01

Decide what "first component" means.

Most arguments about onboarding speed are arguments about when the clock started. Write the events down once.

Suggested owners: Design system engineering lead + DX lead

☐ Two tasks: a hello world and a first real change

T01 **DATA**

One task mixes setup and build friction together. T1 ends at a rendered component; T2 ends at a pull request, the way API teams pair hello world with a first useful app [S01].

Evidence: Two tasks in `tasks.json`, each with its own timebox and done criteria.

S01. Essay task for T2.

☐ Start and stop are events someone can see

T02 **PROTOCOL** **DATA**

Start when the participant has read the card and says start. Stop at an observable state: the component on screen, the pull request opened. Never at "I think I'm done".

Evidence: `start` and `stop` fields per task; the facilitator or the recording confirms both.

S02: time from a defined point to the first successful call.

☐ T2 stops at an opened pull request, not a merge

T03 **PROTOCOL**

Merge time measures reviewers. The essay stops the clock when the participant submits the pull request; lint and typecheck must pass locally first.

Evidence: T2 `done` criteria in `tasks.json`.

Essay benchmark.

☐ Changing a task changes the protocol version

T04 **SCRIPT** **DATA**

A new card, stop event or timebox makes old times incomparable. Bump `protocolVersion` and restart the baseline.

Evidence: `ttfc.mjs` refuses a sheet mixing versions and warns when comparing across them.

Kit.

☐ Lab minutes are not field days

T05 **PROTOCOL** **LOOP**

The benchmark times a task in a session. Field Guide 16's ACT-3 counts calendar days from onboarding to the first merged pull request. DX Core 4 measures organisation-wide throughput [S05]; this is a narrower, task-level signal.

Evidence: Both numbers appear in the weekly funnel report, on separate lines.

S05. Field Guide 16.

If two people would start the clock at different moments, you do not have a benchmark yet.

SECTION 02

Same kind of people, same machine.

Most variance in five sessions comes from who sat down and what their laptop already had. Control both.

Suggested owners: DX lead + research lead

☐ **Five participants who have never used the system**

T06 **PROTOCOL** **SCRIPT**

The essay recommends 3 to 5 new engineers. Take 5 so one outlier does not decide the median. Never the system team.

Evidence: `ttfc.mjs` notes any task with fewer than 5 sessions as directional.

Essay. Kit threshold.

☐ **Voluntary, pseudonymous, reported per task**

T07 **PRIVACY**

Say in the invitation that the system is being tested, not the person. Record `p01`, not a name. A single timed task says nothing about someone's productivity [S06].

Evidence: No names in the sheet; the key is held by the facilitator and deleted after the run.

S06.

☐ **A clean container with no package cache**

T08 **PROTOCOL**

A warm cache or a registry token from last year hides the install friction T1 exists to find. Use a fresh dev container or virtual machine.

Evidence: `environment` column is `clean-container` for every session.

Kit protocol, section 3.

☐ **The same starter commit for the whole run**

T09 **PROTOCOL**

Pin the starter app to one commit per run and record it. A dependency bump mid-run makes the last sessions a different test.

Evidence: The commit hash in the run notes.

Kit protocol.

☐ **One session setup per run**

T10 **PROTOCOL** **SCRIPT**

Moderated and silent is the default. Think-aloud gives reasons but changes times. Unmoderated sessions scale across time zones but cannot clarify a task [S04]. Pick one per protocol version.

Evidence: `ttfc.mjs` refuses a run that mixes `mode` or `think_aloud` values.

S04.

Control the room, or you are timing the laptops.

SECTION 03

Run the session the same way every time.

The facilitator's job is to watch and write, not to rescue. Every exception to that is recorded.

Suggested owners: Facilitator

☐ One assist after five stuck minutes, and it is counted

T11 **PROTOCOL** **DATA**

Help changes the time. If a participant is stuck for 5 minutes, give one pointer, then write the minute and the pointer in the notes. The assist is itself a finding.

Evidence: `assists` column and a note. The Q2 sample has 4 assisted sessions.

Kit protocol.

☐ Interruptions are subtracted, not ignored

T12 **DATA** **SCRIPT**

A fire drill or a call is not task time. Record it in seconds and the script subtracts it.

Evidence: `paused_seconds`: the Q2 sample's 67-minute T2 session with 180 paused seconds counts as 64.

Kit.

☐ Timeouts and give-ups are recorded, never dropped

T13 **PROTOCOL** **SCRIPT**

Stop at the timebox and record `timeout`; a participant who stops trying records `abandoned`. Dropping them makes a slow system look fast.

Evidence: The validator rejects a `success` longer than the timebox. Q2 T2 has 1 timeout and 1 abandoned of 5.

Kit.

☐ Friction is coded live from a fixed list

T14 **DATA** **SCRIPT**

Code every stall over a minute and every question with `friction-codes.json`. Use `env-setup` for problems outside the system, so they are recorded but not fixed by the wrong team.

Evidence: The validator rejects unknown codes; 11 codes ship, including `other`.

Kit.

☐ One question at the end

T15 **PROTOCOL**

"What nearly made you give up?" The answer names the friction the timings cannot.

Evidence: The answer in `notes` for every session.

Kit protocol.

The facilitator watches. The sheet remembers.

SECTION 04

Report what five sessions can support.

Task times are skewed: a few long sessions pull the mean up. Report robust statistics, name the method, and show what did not finish.

Suggested owners: DX lead

☐ **Median and p75, with the method named**

T16 **SCRIPT**

Set targets on the median; watch p75 for the slow tail. Percentiles interpolate linearly between ranks, the same as Excel PERCENTILE.INC and SQLite's `percentile()` [S07].

Evidence: Q3 sample: T1 median 12.0, p75 14.0 minutes; T2 median 31.0, p75 38.0.

S07. Kit output.

☐ **Unfinished sessions count at the timebox**

T17 **SCRIPT**

A timeout is at least the timebox, so it enters the percentiles at that value. A percentile that lands on one is a lower bound and prints with `>=`.

Evidence: Q2 sample, T2: median 88.0, p75 `>=120.0` because 2 of 5 sessions did not finish.

Kit.

☐ **Geometric mean for the typical time on small samples**

T18 **SCRIPT**

Below 25 sessions, the sample median is a biased estimate; the geometric mean of completed times is more accurate [S03]. Report it beside the median.

Evidence: Q3 sample, T1: geometric mean 11.5 against a median of 12.0.

S03, Sauro and Lewis, CHI 2010.

☐ **Completion rate sits beside every time**

T19 **SCRIPT**

A fast median with 60% completion is not a fast system. Print both on the same line.

Evidence: Q2 sample: T1 80%, T2 60% completion.

Kit output.

☐ **The top three friction codes, counted**

T20 **SCRIPT**

Count task sessions per code across both tasks. The list is the work plan for the next quarter.

Evidence: Q2 sample: `unclear-example` (5), `registry-auth` (3), `a11y-lint` (2).

Kit output.

Five sessions give you a direction and a work list. Do not dress them up as precision.

SECTION 05

Compare like with like, then fix three things.

The benchmark is worth running twice. The second run is where it proves or disproves the fixes.

Suggested owners: Design system lead + DX lead

☐ **Compare only runs on the same protocol**

T21 **SCRIPT** **PROTOCOL**

Same protocol version, same setup, new participants. Anything else is a different experiment.

Evidence: `ttfc.mjs --baseline` warns when protocol versions differ.

Kit.

☐ **Report change in minutes and percent, bounds called out**

T22 **SCRIPT**

When the baseline percentile was a lower bound, the real improvement is at least the one shown. Say so.

Evidence: Q2 to Q3, T2: median 88.0 to 31.0 (-57 min, -64.8%); p75 `>=120.0` to 38.0, flagged as a lower bound.

Kit output.

☐ **A target is met, or it is not**

T23 **PRACTICE**

The essay's target is a median under 30 minutes. 31 is not met. Do not round, and remember it is the median of five people.

Evidence: Q3 sample, T2: median 31.0, reported `not met`.

Essay target. Kit output.

☐ **Fix the top three before the next run**

T24 **PRACTICE** **LOOP**

The essay's rule. Name an owner for each and link the fix from the run notes, so the next run tests something.

Evidence: Q2's top code, `unclear-example`, drops to 1 task session in Q3.

Essay. Kit output.

☐ **The result feeds the funnel report**

T25 **LOOP**

Activation is the stage the benchmark diagnoses. Put the latest median and p75 on the benchmark line of Field Guide 16's weekly report, and rerun each quarter.

Evidence: The benchmark line of `weekly-funnel.md` is filled with the latest run.

Field Guide 16.

The benchmark is done when the next run is faster for a reason you can name.

APPENDIX A

One row per session and task.

Timestamps, not typed minutes: the script computes durations, so nobody does arithmetic under time pressure.

Column	Holds	Checked by the script
<code>run_id</code> , <code>protocol_version</code>	Which run, which protocol	One version per sheet
<code>session_id</code> , <code>participant_id</code>	Pseudonymous ids	No duplicate session and task
<code>mode</code> , <code>think_aloud</code> , <code>environment</code>	Session setup	One setup per run
<code>task_id</code>	T1 or T2	Known task
<code>start_ts</code> , <code>stop_ts</code>	ISO 8601 UTC	Stop after start
<code>paused_seconds</code>	Interruptions	Subtracted from duration
<code>outcome</code>	success, fail, timeout, abandoned	Success within the timebox
<code>assists</code> , <code>friction_codes</code> , <code>notes</code>	What happened	Codes in <code>friction-codes.json</code>

data/run-2026-q2.sample.csv (two rows)

```
run_id,protocol_version,session_id,participant_id,mode,think_aloud,environment,task_id,
start_ts,stop_ts,paused_seconds,outcome,assists,friction_codes,notes
2026-q2,1.0,s01,p01,moderated,no,clean-container,T1,
2026-06-16T09:00:00Z,2026-06-16T09:34:00Z,0,success,0,registry-auth;install-peer-deps,
2026-q2,1.0,s04,p04,moderated,no,clean-container,T2,
2026-06-16T14:10:00Z,2026-06-16T15:45:00Z,0,abandoned,0,unclear-example;ally-lint,
Gave up: could not map error text to a fix
```

Rows wrapped to fit the page; each is one line in the file.

APPENDIX B

Two runs, compared.

Q2 is the baseline, Q3 follows fixes to registry setup, the starter template and examples. Both synthetic, five moderated sessions each.

terminal

```
$ node scripts/ttfc.mjs data/run-2026-q2.sample.csv
Time to first component, run 2026-q2 (protocol 1.0), minutes
Task                n  done  median    p75  geomean    range  target
T1 Hello component   5   80%   34.0    41.0   30.2   22.0-41.0  none
T2 Login form PR     5   60%   88.0  >=120.0   74.0   64.0-88.0  median 30: not met
Top friction: unclear-example (5), registry-auth (3), a11y-lint (2)

$ node scripts/ttfc.mjs data/run-2026-q3.sample.csv --baseline data/run-2026-q2.sample.csv
Time to first component, run 2026-q3 (protocol 1.0), minutes
Task                n  done  median    p75  geomean    range  target
T1 Hello component   5  100%   12.0    14.0   11.5    7.0-19.0  none
T2 Login form PR     5  100%   31.0    38.0   33.0   26.0-44.0  median 30: not met
Top friction: a11y-lint (2), types (2), docs-not-found (1)

Compared with 2026-q2:
  T1 median 34.0 to 12.0 (-22 min, -64.7%); p75 41.0 to 14.0 (-27 min);
    completion 80% to 100%
  T2 median 88.0 to 31.0 (-57 min, -64.8%); p75 >=120.0 to 38.0 (-82 min);
    completion 60% to 100%
  note: T2: the baseline p75 is a lower bound (sessions hit the timebox),
    so the real improvement is at least the one shown
```

Output lightly trimmed (legend lines) and wrapped. The Q3 friction list is the next quarter's work: accessibility lint messages and component types.

KEEP WITH THE RUN

Leave a run record.

One record per run, so the next facilitator can reproduce it and the next reader can trust the comparison.

Run id / protocol version	Dates / facilitator
Starter app commit	Session setup (mode, think-aloud)
Participants (count, how recruited)	Median / p75 per task
Completion per task	Top three friction codes
Fixes committed (owner, link)	Next run date

Delete the key.

Once the run is analysed, delete the list that links participant codes to names. The findings do not need it.

SOURCES / MAINTENANCE

Keep the guide current.

Sources checked 24 September 2026. Vendor statements are cited as vendor views. No external benchmark figures are reused: the only targets are the essay's.

S01 / Red Hat Developer, Building great APIs part II: simplicity, flexibility and TTFHW

Guerrero and Willmott define time to first hello world and its follow-ons (November 2012).

<https://developers.redhat.com/blog/2012/11/09/building-great-apis-part-ii-simplicity-flexibility-and-ttfhw>

S02 / Postman, The most important API metric is time to first call

Time from a point of discovery to the first successful call; usability testing as one way to measure it.

<https://blog.postman.com/the-most-important-api-metric-is-time-to-first-call/>

S03 / MeasuringU, Average task times in usability tests: what to report?

Sauro and Lewis (CHI 2010): geometric mean for samples under 25, median above.

<https://measuringu.com/average-times/>

S04 / Nielsen Norman Group, Remote usability tests: moderated and unmoderated

Trade-offs between facilitated and self-run sessions (Schade, 2013).

<https://www.nngroup.com/articles/remote-usability-tests/>

S05 / DX, Guide to the DX Core 4

Speed, effectiveness, quality and impact as organisation-level productivity measures.

<https://docs.getdx.com/dx-core-4/>

S06 / Forsgren et al., The SPACE of Developer Productivity

ACM Queue, 2021: productivity is not captured by one metric or one dimension.

<https://www.microsoft.com/en-us/research/publication/the-space-of-developer-productivity-theres-more-to-it-than-you-think/>

S07 / SQLite, the percentile extension

`percentile()` interpolates linearly at $P*(N-1)/100$, the method `ttfc.mjs` uses.

<https://sqlite.org/percentile.html>

Maintenance: review the protocol after each run; any change to tasks, stop events, timeboxes or session setup is a new protocol version. Update the PDF, HTML, Markdown and JSON together.