F

is for

FAST

(Simplified Guide)

JOHNATHAN HAIR

Preface

I began this project in 2023, when generative AI tools were flooding the market with low-effort books. That noise almost stopped me. If AI could generate endless guides, did another manual really matter?

As it turns out, yes. When anyone can build software with AI, the difference between shipping successfully and getting caught in endless rework comes down to process making clear what should be delivered and what success looks like for the customer. The right framework eliminates friction before it starts.

This iteration is deliberately concise: no fluff, just the principles you need to build efficiently.

Fast Is Not Urgency

Startups want to be responsive. They want to recognise an opportunity, make a decision and deliver something valuable before the moment has passed. They want to learn quickly without becoming trapped in unnecessary process.

Those are reasonable ambitions.

Urgency is often what appears when the organisation struggles to achieve them. Priorities exceed capacity. Decisions arrive late. Commitments are made before the work is understood. Important context is missing, risks are discovered too late and several initiatives compete for the same people.

The pressure reaches the team at the end of the chain. People compensate by starting before they understand the problem, reading only part of the specification, bypassing peer review, weakening tests or leaving important decisions undocumented.

The work may move sooner, but it also comes back.

Some situations are genuinely urgent. A service failure, a serious customer problem or a closing opportunity may demand an immediate response. But when urgency becomes the normal operating condition, it is no longer an exceptional response. It is evidence that something in the way work is organised needs attention.

Fast is the ability to respond and deliver valuable outcomes without repeatedly depending on urgency. It comes from clear facts, deliberate focus, reusable judgment, smooth flow, early feedback and reliable follow-through.

Framework

Building without friction requires a repeatable system. The Fast Framework provides this structure, taking your work from a vague idea to a valuable outcome. The core process relies on six steps: establishing Facts, choosing Focus, applying Frameworks, improving Flow, gathering Feedback, and ensuring Follow-Through.

Once this foundation is set, the framework leverages Force Multipliers to extend what you can do, turning each completed cycle into a self-reinforcing Flywheel.

The sequence is:

Facts → Focus → Frameworks → Flow → Feedback → Follow-Through → Force Multipliers → Flywheel

In plain language:

See → Choose → Decide → Move → Learn → Change → Multiply → Compound

The first six F's form the operating pipeline. They turn uncertainty into a useful outcome and an informed next action. Force Multipliers increase the reach of that pipeline. The Flywheel appears when every cycle improves the conditions for the next.

This is not a rigid process. You may discover during Flow that the Facts are weak, or learn during Feedback that the Focus was misplaced. Returning to an earlier discipline is not failure. It is the framework doing its job before more effort is lost.

Use the framework as a diagnostic. When important work feels slow, ask where it is waiting. When activity is intense but producing little value, ask which discipline has been skipped.

When urgency becomes constant, ask what weakness in the system it is concealing. When a new tool promises acceleration, ask what part of the system it will multiply.

The purpose is not to make everything immediate. Some work deserves care, evidence and patience. The purpose is to reduce the distance between uncertainty and a responsible, valuable result.

That word responsible matters. A shortcut is not fast when it makes later work slower. A review is not waste when it prevents expensive rework. A process is not valuable merely because it exists, but removing it without understanding the risk it controls is haste, not improvement.

Step 1. Facts: Face Reality

Every initiative begins with a story about what is happening and what should happen next. Stories create direction, but they become dangerous when assumptions are presented as facts.

Facts establish contact with reality. They include observed behaviour, direct evidence, known constraints and measurable conditions. They do not eliminate uncertainty. They show where uncertainty exists so that the next action can be designed intelligently.

Separate three kinds of statements:

- **Facts**: what has been observed
- **Assumptions**: what is currently believed but not sufficiently tested
- **Bets**: what you have chosen to do despite uncertainty

This distinction makes reasoning inspectable. A bet can be bold and still be honest. An assumption can be sensible and still need testing. A fact can be accurate and still be incomplete.

The failure pattern

Urgency compresses attention. Teams promote weak signals into strong conclusions because there appears to be no time to investigate. Interest becomes demand. Activity becomes value. A confident opinion becomes customer truth. A partial reading of the specification becomes shared understanding. Once the story hardens, substantial work accumulates behind it.

The cost appears late: a solution is delivered well, but the original problem was misunderstood.

Put Facts into practice

Before committing meaningful effort, write three lists:

1. What we have observed
2. What we currently assume
3. What we are choosing to bet

Identify the assumption that could invalidate the most work. Look for the smallest responsible way to learn more about it.

Facts should also have a route into ordinary work. Conversations, operating signals, support requests, financial behaviour, failures and observed workarounds all contain evidence. Collection is not enough; relevant facts must be available when decisions are made.

The discipline of Facts is not about waiting for certainty. It is about preventing avoidable misguided confidence.

Fast principle: Fast begins with reality. Haste begins with an assumption.

Step 2. Focus: Choose What Matters

Most organisations have more plausible work than they can complete. Requests, risks, opportunities and improvements all compete for the same attention. When everything remains important, value moves slowly even though activity is everywhere.

Focus is not identifying good ideas. It is choosing which good idea will move now—and protecting it from the arrival of the next one.

A priority becomes real only when it resolves a conflict. If two important initiatives require the same person, platform or decision, Focus determines which one moves first.

Focus on an outcome

A list of features or tasks describes activity. An outcome describes the change the work should create.

Complete this sentence:

> For [specific person or group], we want [observable change], because [valuable consequence].

The statement creates a boundary around value. It allows you to consider several ways of producing the outcome instead of committing too early to the most obvious solution.

Then find the smallest complete promise: the narrowest experience or intervention that allows someone to achieve a worthwhile result. Small should not mean fragmented or useless. Complete should not mean comprehensive.

The failure pattern

Work expands when the user is vague, the outcome is implicit and exclusions are never stated. Urgency makes this worse by labelling every request important and every deadline immovable. Teams add reasonable capabilities until a small question becomes a large programme. The eventual result may be impressive but arrive too late to guide the original decision.

Put Focus into practice

For the next initiative, define:

1. The person whose outcome matters
2. The observable change you want
3. The smallest complete promise
4. What will deliberately remain outside the work

The fourth decision makes Focus credible.

Fast principle: Fast makes a clear choice. Urgency makes everything a priority

Step 3. Frameworks: Make Judgment Repeatable

Recurring decisions consume attention when their reasoning must be rebuilt every time. People ask the same questions, reopen settled trade-offs and wait for someone who carries the necessary context.

A framework is a compact representation of the questions, principles, defaults and trade-offs that improve a recurring decision. It does not eliminate judgment. It gives judgment a better starting point.

For example, a framework for evaluating a request might ask:

- Who is experiencing the problem?
- What are they unable to accomplish?
- What evidence shows the consequence matters?
- Does solving it support the chosen direction?
- What is the smallest way to test the value?

The framework does not decide. It prevents the conversation from being governed entirely by urgency, hierarchy or the attractiveness of a proposed solution.

Good process serves the same purpose. A specification, peer review, test or release check should protect an outcome or control a known risk. If a safeguard repeatedly delays work, improve how it operates: clarify ownership, reduce the batch, bring review earlier or automate the mechanical parts. Removing the safeguard without understanding what it protects merely moves risk downstream.

The failure pattern

Frameworks fail when they become decorative diagrams, lengthy governance documents or rules applied without context. A framework that is difficult to remember, hard to find or disconnected from the decision is documentation, not leverage. Yet abandoning every framework under pressure creates the opposite failure: consequential decisions are made from incomplete context and must later be reconstructed.

They also fail when they never change. A framework captures current judgment. Feedback should improve it when assumptions, conditions or risks change.

Put Frameworks into practice

Choose a decision that has occurred at least three times. Capture:

1. The outcome the decision should protect
2. The questions a sound decision must answer
3. The default when evidence is weak
4. The evidence that justifies an exception
5. Where the framework will appear when the decision recurs

Keep it close to the work. The best framework is not the most complete one; it is the one people actually use to make better decisions.

Fast principle: Keep the safeguards that protect value. Improve the way they flow.

Step 4. Flow: Shorten the Distance to Value

Work can be active without moving.

An initiative may spend a small amount of time being changed and most of its life waiting for information, decisions, specialist capacity, review or release. Everyone remains busy while the valuable outcome remains unfinished.

Flow follows the work rather than the organisational chart. It asks how an idea, request or problem travels through the whole system and where momentum is lost.

Look for waiting

The largest opportunities are often found between activities:

- Work waiting for a decision
- Context lost during a handover
- Too many initiatives competing for the same capacity
- Large batches waiting for review or release
- Rework caused by late discovery

Optimising the fastest activity will not improve the whole if the work still waits at the constraint.

Reduce work in progress. Starting creates activity; finishing creates value. When important work is blocked, the first response should be to help it move rather than begin something new.

Make work small enough to travel through the complete system. A narrow end-to-end slice creates a path for value and feedback. Once the path is real, it can be widened with evidence.

Flow is not haste

Flow removes avoidable waiting and makes responsible work easier to finish. Haste removes steps without understanding their contribution.

Skipping the full specification may make implementation begin earlier while causing clarification and rework later. Bypassing peer review may move a change toward release while moving defects toward customers. Weakening development quality may shorten one task while making every change to the same area harder.

These are not improvements to Flow. They are transfers of work and risk into a later queue, where they are usually more expensive and less visible.

The failure pattern

Local efficiency can damage overall Flow. One function produces work faster than the next can absorb it. Automation increases output at a non-critical step. People are rewarded for remaining busy rather than helping the shared outcome finish. Under constant urgency, the fastest-looking local action wins even when the whole outcome becomes slower.

Fast principle: Fast makes a clear choice. Urgency makes everything a priority

1. Time actively changing the work
2. Time spent waiting
3. Points where context changed hands
4. Places where work returned for clarification
5. The first point at which reality could respond

Begin with the longest wait or the latest discovery.

Fast principle: Remove waiting and rework, not understanding and care.

Step 5. Feedback: Let Reality Answer

Delivery makes an idea available to reality. Feedback is how the system discovers what happened.

Feedback includes behaviour, results, conversations, failures and operational signals. These forms of evidence are not interchangeable. A metric may show a pattern; a conversation may help explain it. A stated preference may be useful, but observed behaviour is usually closer to the outcome.

The goal is not to collect the largest quantity of feedback. It is to decide what question needs an answer and create a trustworthy way for reality to answer it.

Design feedback with the work

Before acting, complete two sentences:

> We believe this change will cause...
>
> We will reconsider if we observe...

The first gives the work a testable purpose. The second protects the team from interpreting every result as confirmation.

Useful feedback is close enough to the desired outcome to guide a decision. Activity is not automatically evidence of value. A completed task, a release, an opened message or a recorded interaction may say little about whether the intended change occurred.

Peer review, automated checks and small releases are also feedback mechanisms. They allow reality to answer while the cost of correction is still low. Treating them only as delays confuses the absence of challenge with progress.

The failure pattern

Feedback becomes passive when it has no audience, owner or decision attached. Dashboards accumulate, conversations are summarised and requests enter a backlog, but nothing is scheduled to interpret the evidence. In urgent environments, uncomfortable feedback is especially easy to defer because acknowledging it appears to threaten the deadline.

Feedback can also create noise when every reaction is treated as an instruction. A request is evidence that something matters. It is not always the correct specification for what should happen next.

Put Feedback into practice

For a meaningful change, identify:

1. The assumption being exposed to reality
2. The behaviour or result that would support it
3. The evidence that would weaken it
4. Who will interpret the result

Fast principle: The earlier reality can answer, the less haste can hide.

Step 6. Follow-Through: Make Learning Change the System

Feedback creates knowledge. Follow-Through converts knowledge into capability.

Organisations frequently collect evidence, discuss lessons and then preserve the conditions that produced the same result. The finding is recorded, but future behaviour remains unchanged.

Constant urgency is one reason. The team repairs the immediate symptom, promises to return to the underlying cause and moves directly to the next urgent demand. The workaround becomes normal. The weakness that created the incident remains in place, ready to generate the next one.

In systems-thinking terms, Feedback is the sensor. Follow-Through is the actuator. Both are required to close the loop.

Turn evidence into a response

An insight should produce an explicit disposition:

- Act now because it affects an important outcome
- Run a defined next test because uncertainty remains
- Change a framework, default or guardrail
- Preserve it for a named future condition
- Decline action and record the reasoning

“Not now” can be a sound decision. “Somewhere in the backlog” often avoids one.

Give the learning an owner and a trigger. A trigger may be a date, but it can also be a condition: review after enough cases occur, reconsider when manual effort crosses a threshold, or act when a repeated signal appears.

Change more than the output

The obvious response is often to change the immediate product, service or deliverable. The more valuable response may be to change the system that created it.

Follow-Through can update:

- The work itself
- A process or ownership boundary
- A technical or operational control
- A framework or default
- The information made visible to future decisions

It can also retire a process that no longer protects anything, or strengthen one that has repeatedly prevented harm. The objective is neither maximum process nor minimum process. It is the least costly system that reliably protects the intended outcome.

Preserve why the decision was made, not only what was decided. Without the reason, a sensible constraint can survive too long or be removed only for the organisation to relearn the same lesson.

The failure pattern

Feedback without Follow-Through creates informed stagnation. Follow-Through without Feedback

creates reactive change. The pair must remain adjacent. Otherwise urgency repeatedly wins the current cycle while the system repeatedly loses the next.

Fast principle: Finish the learning, not only the task.

Force Multipliers: Extend Capability

A force multiplier increases the effect of an existing capability. It does not determine whether that capability is useful.

Force multipliers include reusable assets, automation, distribution, specialist knowledge and well-designed tools. AI agents are a particularly flexible modern example: they can monitor information, apply defined reasoning, use tools and carry routine actions through a workflow.

The order matters. A multiplier applied to a capable system increases useful reach. Applied to a confused or hurried system, it can produce more noise, more complexity and more work to verify. AI agents can act much faster than a person, which makes clear context, evaluation and boundaries more important rather than less.

Multiply the whole pipeline

AI agents can support every earlier discipline:

- Facts: monitor signals, organise evidence and retrieve relevant context
- Focus: summarise competing needs and expose unresolved assumptions
- Frameworks: apply recurring decision criteria consistently
- Flow: reduce translation, implementation and coordination toil
- Feedback: classify results, identify patterns and surface exceptions
- Follow-Through: prepare actions, update knowledge and monitor commitments

This is broader than generating output. The opportunity is to reduce the cost of keeping the entire learning loop moving.

Keep judgment accountable

Break a candidate workflow into four parts:

1. Observe: what information enters?
2. Interpret: what reasoning or classification is required?
3. Decide: which judgment must remain accountable to a person?
4. Act: what can be completed safely after the decision?

Use a multiplier where context can be supplied, quality can be evaluated, the work recurs and mistakes can be contained. Measure the improvement in the whole Flow, not only the time saved on one task.

The failure pattern

The most common mistake is accelerating activity that should be removed, simplified or challenged. The second is surrendering consequential judgment because an output appears confident. Under urgency, both mistakes become tempting: automation is used to compensate for unclear thinking, and verification is treated as expendable.

Fast principle: A multiplier makes discipline more powerful—and haste more expensive.

Flywheel: Make Every Cycle Better

Completing work produces an outcome. A flywheel also improves how the next outcome will be produced.

A repeated process is not automatically a flywheel. If the same decisions are reconstructed, the same mistakes recur and the same manual effort is required, the system is cycling without gaining momentum.

A flywheel retains value from each pass through the pipeline.

The opposite is an urgency loop. A shortcut creates a defect, unclear decision or fragile component. That consequence produces unplanned work. Unplanned work creates more urgency, which justifies another shortcut. The organisation appears to be moving constantly while its capacity is consumed by consequences of earlier haste.

Call this urgency debt: future attention already committed by today's shortcuts. Like other forms of debt, it can occasionally be taken deliberately. It becomes dangerous when nobody records the obligation, prices the consequence or creates a credible route to repayment.

What should accumulate

Each cycle can leave behind:

- Knowledge: evidence, customer language, operating lessons and failed assumptions
- Judgment: frameworks, principles, defaults and examples
- Assets: reusable components, templates, data and automation
- Capability: simpler workflows, better tools and reduced toil
- Trust: visible reasoning, reliable follow-through and demonstrated results

The next cycle begins with better Facts, sharper Focus, stronger Frameworks and a clearer path to Flow. Feedback becomes easier to interpret. Follow-Through becomes less manual. Force Multipliers operate with better context.

The pipeline delivers the work. The Flywheel improves the pipeline.

Retain selectively

Not everything should become a reusable asset. Documenting every conversation, generalising every component and automating every task can create a new form of toil.

Retain something when recurrence is likely, the cost is meaningful and the retained form will be available where future work occurs. The test is whether it reduces future effort without creating greater maintenance than it saves.

The failure pattern

Flywheels lose momentum when learning remains trapped in people or tools, when urgent work repeatedly overrides improvement, or when local optimisation damages the whole. Heroic recovery may finish the current job while preserving dependence on the hero. A healthy flywheel replaces recurring urgency with retained knowledge, clearer decisions and safer defaults.

Put Flywheel into practice

At the end of meaningful work, ask:

1. What did we learn that changes a future decision?
2. What did we create that can be reused?
3. What recurring toil did we remove?
4. What trust did we earn?
5. Where will each advantage enter the next cycle?

Fast principle: Leave the next cycle calmer, clearer and more capable.

Run the Fast Diagnostic

Use the framework when work feels slow, confused or unnecessarily effortful. Score each discipline from 1 to 5:

- 1 — Fragile: success depends on heroics, memory or luck
- 2 — Inconsistent: useful practices exist but are applied irregularly
- 3 — Functional: the discipline usually supports the work
- 4 — Strong: it is explicit, repeatable and measured
- 5 — Compounding: it improves itself and strengthens other disciplines

First ask: are we fast or merely urgent?

Before scoring the framework, look for signs that activity is being mistaken for Fast:

- Work begins before the problem or specification is understood
- Quality checks, tests or peer review are routinely bypassed
- New work is opened faster than valuable work is finished
- The same shortcuts create repeated clarification, defects or incidents
- Heroics are celebrated while the conditions requiring them remain unchanged

One exceptional event may justify urgency. A repeated pattern points to a system that needs redesign.

Eight diagnostic questions

1. Facts: What do we know, and how do we know it?

Focus: What matters now, and what will we leave undone?

2. Frameworks: Which reasoning are we repeatedly paying to reconstruct?
3. Flow: Where does valuable work wait?
4. Feedback: What did reality tell us?
5. Follow-Through: What will change because of what we learned?
6. Force Multipliers: What useful capability are we multiplying?
7. Flywheel: What became permanently easier?

Do not begin by improving every score. Find the weakest discipline that constrains the important outcome. A low score may be acceptable when it does not affect current work. A single constraint often matters more than several average weaknesses.

Choose one intervention that can be observed within a short cycle. Define the change you expect, who owns it and what evidence will show whether it helped. Then run the pipeline again.

The framework is working when it helps you replace pressure with design:

- Better evidence instead of louder certainty
- Clearer choices instead of more priorities
- Retained judgment instead of repeated debate
- Movement of value instead of visible busyness
- Useful learning instead of passive measurement
- Changed behaviour instead of recorded insight
- Multiplied capability instead of multiplied noise
- Compounding improvement instead of recurring toil

Fast should feel increasingly deliberate. As the system improves, less energy is spent recovering

context, chasing decisions, correcting avoidable defects and coordinating around preventable surprises. Important work can still be intense. It no longer needs permanent urgency to move.

> See clearly. Choose deliberately. Decide consistently. Move value. Learn from reality. Change the system. Multiply what works. Compound the advantage.

About Johnathan Hair

About Johnathan Hair

Johnathan Hair is a product-minded systems thinker who helps people and organisations turn uncertain ideas into valuable outcomes quickly, reliably and with less toil.

His work connects evidence, product judgment, flow, feedback systems and practical AI agents.

Rather than treating Fast as pressure applied to individuals, he focuses on improving the whole system through which ideas become decisions, outcomes and compounding capability.

https://johnathan.hair

https://f-is-for-fast.com