

A2A Protocol Simply Explained: Here are 3 key differences to MCP!



ZACHARY HUANG

APR 13, 2025



19



1

SI



Imagine a world where your AI Agents don't just work FOR you, but work WITH each other! Google's Agent-to-Agent (A2A) protocol promises to turn solo AI performers into collaborative team players. But how does it stack up against Anthropic's Model Context Protocol (MCP)? Time to break down the buzz!

Introduction

Looks like an article worth saving!

Option



Hover over the brain icon or use hotkeys to save with Memex.

Think about how you
information, ask que

Remind me later

Hide Forever

301

That's exactly what Google is aiming for with their new [Agent-to-Agent \(A2A\) protocol](#), released on April 9, 2025. Instead of each AI Agent being a lone worker, turns them into team players. Your research Agent could seamlessly pass findings your writing Agent, or your travel planner could check with your finance Agent to ensure hotel options fit your budget—all without you having to play middleman. The developer community is clearly excited—[over 7,000 GitHub stars](#) in just days after launch says it all. But this isn't the first attempt at getting AI systems to work together. Anthropic introduced their [Model Context Protocol \(MCP\)](#) with similar not long ago.

So what's a developer to make of these options? Is A2A just MCP with a different name? Should you invest time learning one over the other? Or do they serve different purposes entirely?

- Explain both protocols in plain language anyone can understand
- Highlight the 3 technical differences that impact how you'll use them
- Reveal how these protocols might actually complement rather than compete v each other

No technical jargon overload—just clear insights into how these protocols are reshaping AI collaboration. **Looks like an article worth saving!** Option Q

Let's dive in! Hover over the brain icon or use hotkeys to save with Memex.

MCP vs. A2A

Hide Forever

Imagine you're planning a dream vacation to Hawaii. There's so much to figure out. You need to:

- Check weather forecasts to pick the best month
- Find flights within your budget
- Plan activities based on local recommendations
- Convert your dollars to understand local prices

That's a lot of different tasks requiring different expertise! Now, suppose you have an AI Agent like Claude to help you. You ask, "Claude, what's the weather like in Maui next week?"

But here's the problem: **Claude was trained on older data and doesn't know today's weather or next week's forecast.** It's like asking your smart friend about the weather in a city they haven't visited recently – they simply don't know! This is where our most important standards come in to save the day.

MCP: Giving AI Agents Super Powers

Model Context Protocol (MCP) is like giving any AI Agent the ability to use specialized tools when needed.

Before MCP, the conversation might go like this:

You: "What's the weather in Maui next week?"

Claude: "I don't have access to real-time weather data or forecasts. You'd need to check a weather service for that information."

Pretty disappointing. **Introducing MCP:**

You: "What's the weather in Maui next week?"

Looks like an article worth saving!

Option



Hover over the brain icon or use hotkeys to save with Memex.

Claude: *[internally]*

Remind me later

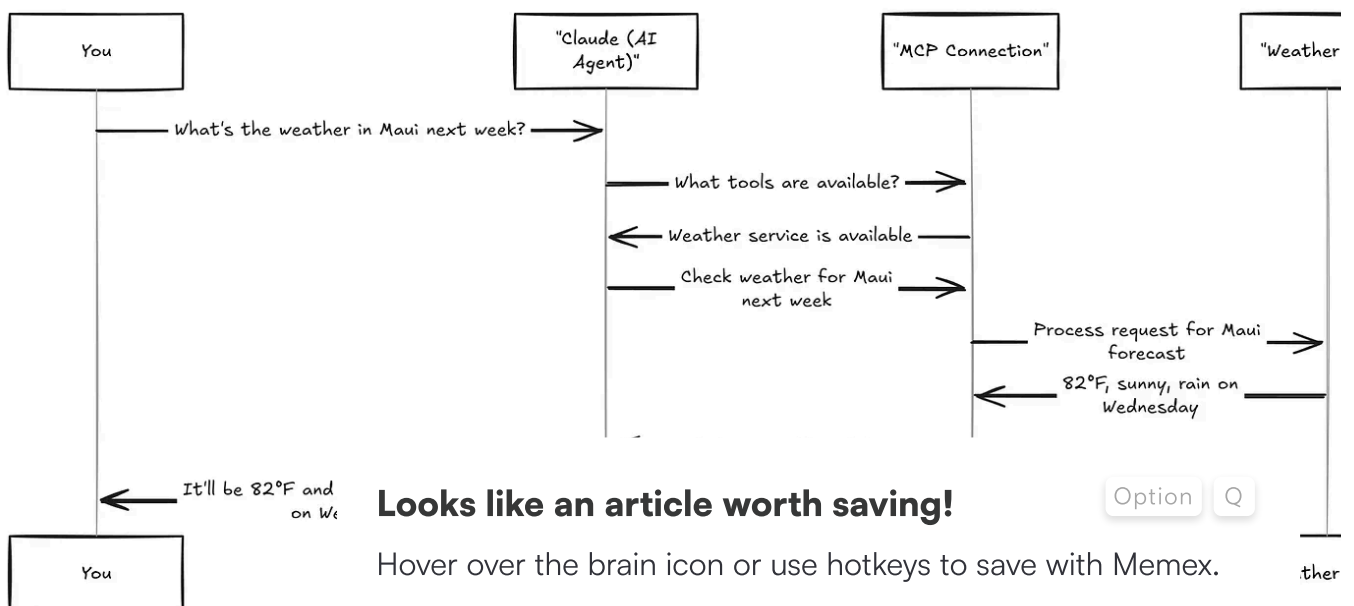
Hide Forever

later

forecast, Maui will be sunny with temperatures around 82°F next week, with a shower expected on Wednesday afternoon."

What's actually happening behind the scenes is like a magic trick with four simple steps:

1. **Discover Available Superpowers:** Claude peeks into its toolbox asking, "What cool gadgets do I have today?" and discovers there's a weather service ready to use!
(Code Reference: Uses [list_tools method](#))
2. **Make the Perfect Request:** Claude crafts a crystal-clear message: "Hey Weather Tool, what's the forecast for Maui next week?" (but in computer-speak)
(Code Reference: Uses [call_tool](#))
3. **Get the Expert Answer:** The weather service does its weather-magic and replies: "Maui: 82°F, sunny, with a sprinkle on Wednesday!"
(Code Reference: Uses [Tool.run](#))
4. **Translate to Human-Friendly:** Claude takes this weather-speak and turns it into a friendly, conversational response just for you
(Code Reference: Gets result from [call_tool](#))



Remind me later

Hide Forever

The magic here is that MCP provides a standard way for ANY AI Agent to connect ANY tool. It's like creating a universal adapter that works with all appliances and power outlets.

For tool builders, this means they only need to build their weather API, calculator email sender once following the MCP standard, and it will work with Claude, GPT, Gemini, or any other MCP-compatible Agent.

For users like you and grandma, it means your AI can suddenly do all sorts of things it couldn't do before - check real-time weather, send actual emails, book real appointments, all without being explicitly programmed for each task.

A2A: Assembling Your AI Dream Team

But what about that complex Hawaii trip planning? You need more than just weather data - you need specialized expertise in flights, hotels, activities, budgeting, and more. This is where Agent-to-Agent (A2A) protocol shines. While MCP connects an AI to tools, A2A connects specialized AI Agents to each other.

Imagine having a travel dream team:

- A flight-finding specialist who knows every airline and route
- A hotel expert who can match accommodations to your preferences
- A local guide specialist familiar with Hawaiian activities and culture
- A budget manager who helps keep spending in check

With A2A, your personal Agent can delegate specific parts of your vacation planning to these specialists, then collect and organize their expert answers.

Before A2A, the conversation would look like this:

You: "Plan my 5-day trip to Hawaii"

Personal Agent: "I'd be happy to help you plan your 5-day trip to Hawaii. Here are some suggestions: 1. Flight: Hawaiian Airlines offers direct flights from LAX to HNL. 2. Hotel: The Hale Koa Hotel is a great choice for a luxury stay. 3. Activities: Visit Pearl Harbor, hike Diamond Head, and enjoy the views from the Waikiki Beach. 4. Food: Try the local cuisine at the Hale Koa Hotel. 5. Transportation: Rent a car or use taxis. Let me know if you need more details!"

Looks like an article worth saving!

Option Q

Hover over the brain icon or use hotkeys to save with Memex.

Remind me later

Hide Forever

1aV

access to current flight prices or hotel availability. For activities, I can suggest popular options, but can't confirm if they'll fit your specific dates or budget."

Not very helpful for actually booking your dream vacation! But with A2A:

You: "Plan my 5-day trip to Hawaii in June with a \$3000 budget."

Personal Agent: *[internally delegates to specialist agents]* "I've prepared a complete 5-day Hawaii itinerary for June! Your round-trip flights from LAX (\$650) plus the beachfront hotel in Maui (\$1200) leave \$1150 for activities. I've scheduled snorkel at Molokini Crater on day 2 (\$145/person), a traditional luau on day 3 (\$120/person) and the Road to Hana tour on day 4 (\$210/person). That leaves about \$675 for meals and souvenirs. Would you like me to book any of these options?"

Behind the scenes, your personal Agent uses A2A to orchestrate this vacation symphony:

1. **Find the Right Experts:** Your Personal Agent flips through its digital Rolodex "business cards" ([AgentCard](#)) to see which specialist agents can help with travel activities, and budgeting. (Code Reference: Uses [A2ACardResolver](#) to fetch the cards)
2. **Divide and Conquer:** The Personal Agent splits your vacation dream into manageable pieces:
 - Job 1: Find flights/hotels → Travel Agent
 - Job 2: Suggest activities → Local Guide Agent
 - Job 3: Check budget → Budget Agent
3. **Assign the Flight Mission:**
 - The Personal Agent wraps Job 1 into a neat digital package ([Task](#)) with a tracking number "12345678901234567890"
 - It ships this package to the Travel Agent (Code Reference: [A2ATaskSender](#))
 - The Travel Agent receives the package and starts working on it.

Looks like an article worth saving!

Option Q

Hover over the brain icon or use hotkeys to save with Memex.

Remind me later

Hide Forever

- **Personal Agent → Travel Agent:** "Find affordable flights to Hawaii from [location] and hotel options in Maui for 5 days in June, for a couple with a total budget of \$3000."
- **Travel Agent → Personal Agent:** "Found round-trip LAX to OGG (Maui) for \$650/person on Hawaiian Airlines. The Aloha Beach Resort has rooms for \$240/night, totaling \$1200 for 5 nights. Both options include free cancellation."

4. Repeat for Activities and Budget:

- The Personal Agent does the same package-and-send dance for the other giving each specialist their own mission and tracking number.
- **Personal Agent → Local Guide:** "What are the best activities for a couple in Maui for 5 days in June with about \$1150 left after flights and hotel?"
- **Local Guide → Personal Agent:** "Recommended activities: Molokini Crater snorkeling (\$145/pp), traditional luau at the Old Lahaina Luau (\$120/pp), and Road to Hana guided tour (\$210/pp). These are highly rated and available in June."
- **Personal Agent → Budget Agent:** "Does this plan fit within \$3000: \$650 flights, \$1200 hotel, \$475 for main activities, plus food and souvenirs?"
- **Budget Agent → Personal Agent:** "The essentials total \$2325, leaving \$675 for meals and souvenirs. This is approximately \$135/day for food and extras, which is reasonable for Hawaii if you mix restaurant meals with more affordable options."

5. Play the Waiting Game:

- Just like tracking a delivery, the Personal Agent periodically asks, "Is task done yet?" using the `tasks/get` command (*Code Reference: Uses [A2AClient.get_task](#)*)

- When an agent grabs

Looks like an article worth saving!

Option Q

ers

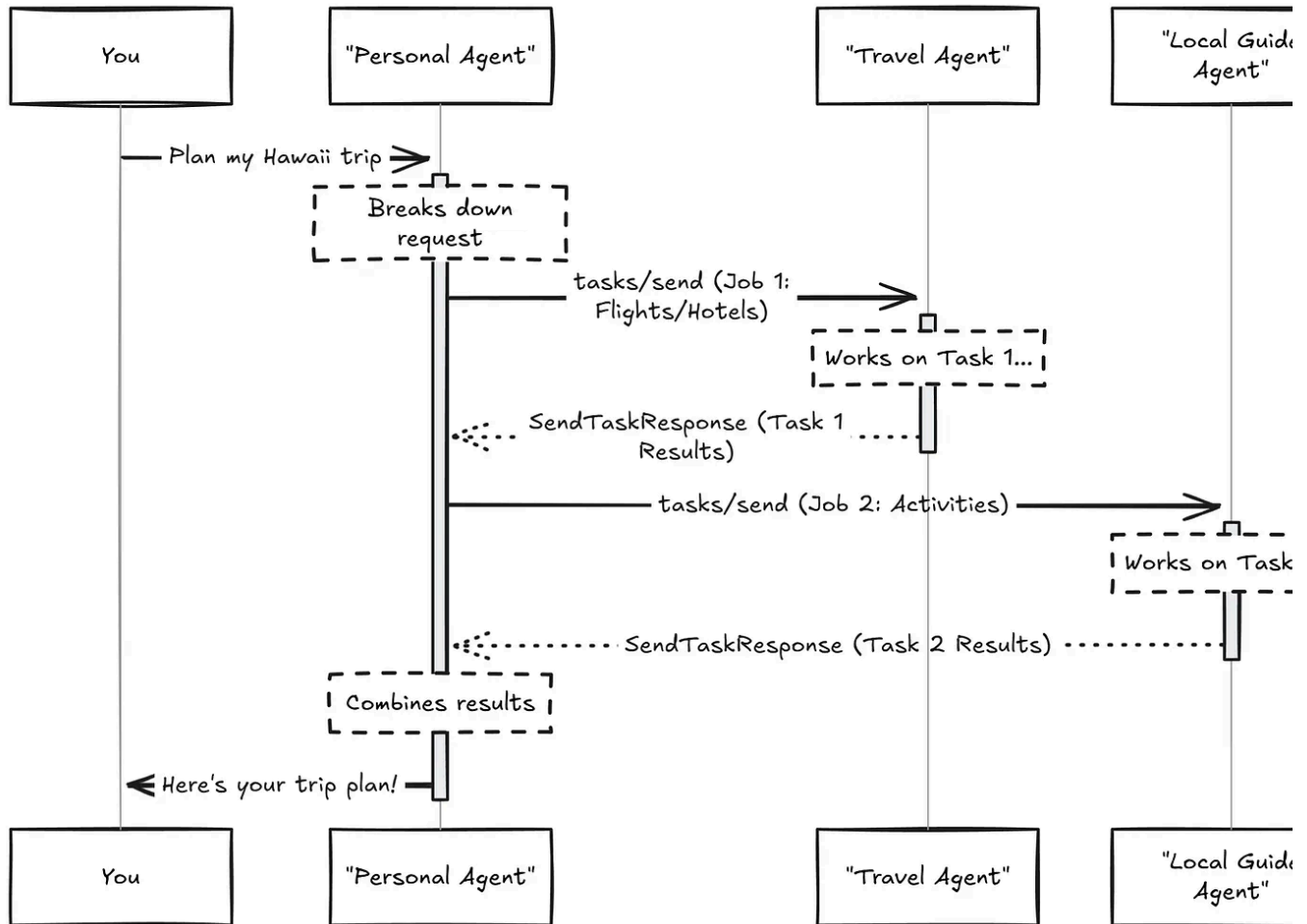
Hover over the brain icon or use hotkeys to save with Memex.

6. Create Vacation recommendation

Remind me later

Hide Forever

you



Your personal Agent then compiles all this specialist knowledge into one cohesive plan. The entire process is seamless to you - you just get a complete, expert-crafted vacation itinerary!

The Beautiful Combination

The real magic happens when MCP and A2A work together:

1. Your personal Agent uses A2A to connect with specialized AI agents
2. Each specialized agent uses MCP to connect with specific tools they need
3. The result is a new, seamless experience for you

working together **Looks like an article worth saving!**

Option Q

Hover over the brain icon or use hotkeys to save with Memex.

It's like having both : their own specialized that no single AI could handle alone.

Remind me later

Hide Forever

Ita

x t

The key difference is who's talking to whom:

- **MCP:** AI Agents talking to tools (Claude + weather service)
- **A2A:** AI Agents talking to other AI Agents (Trip planner + hotel expert)

And the best part? Build once, connect anywhere. These standards ensure that new tools and new Agents can join the ecosystem without requiring custom code for every possible combination.

To see minimal working code examples of these protocols in action:

- For a simple MCP implementation, check out the [PocketFlow MCP Example](#)
- For a basic A2A implementation, visit the [PocketFlow A2A Example](#)

Beyond Tools vs. Agents: The Real Technical Differences

That whole "MCP talks to tools, A2A talks to agents" explanation? It's a good start but if you're hungry for more, you might be thinking, "Wait, isn't that too simplistic?"

You'd be right! The boundary between "tools" and "agents" is actually pretty fuzzy in fact, as [OpenAI's documentation shows](#), you can even use entire agents as tools! Imagine asking your AI Agent to use another AI Agent as a tool - it gets meta pretty quickly.

If you're not satisfied with the surface-level explanation and want to dig deeper, you're in the right place. Let's peel back the layers and look at the actual code-level differences that make these protocols fundamentally distinct - beyond just who's talking to whom!

Quick disclaimer: We **Look like an article worth saving!** based on working with it. Hover over the brain icon or use hotkeys to save with Memex. If you spot something off, click the flag icon.

Remind me later

Hide Forever

Difference 1: Natural Language vs. Structured Schema

With A2A, you simply say "How much is 100 USD in Canadian dollars?" With MCP you need to match exactly what the currency tool expects - like `{"currency_from": "USD", "currency_to": "CAD", "amount": 100}`. It's a completely different approach to communication!

A2A's Human-Like Communication

A2A embraces natural language just like we humans talk to each other:

```
# A2A Client sending a task
user_message = Message(
    role="user",
    parts=[TextPart(text="How much is 100 USD in CAD?")]
)
```

The receiving agent interprets this request however it's able to - just like when you ask a friend for help. There's no strict format required.

MCP's Precise Parameter Requirements

MCP, on the other hand, demands exact parameters that match a predefined schema.

```
# MCP Client calling a tool
tool_name = "get_exchange_rate"
# Must match EXACTLY what the tool expects
arguments = {"currency_from": "USD", "currency_to": "CAD"}
```

If you don't provide exactly what the tool expects (maybe it wants "from_currency" instead of "currency_from"), the call fails. No wiggle room here!

Why This Matters

Looks like an article worth saving!

Option Q

Think about calling a currency tool. A2A is flexible, while MCP is rigid. Hover over the brain icon or use hotkeys to save with Memex.

representative who can handle various categories. With MCP, you need to be precise in specific fields.

Remind me later

Hide Forever

o
in

The natural language approach of A2A means it can handle "out of the box" requests and adapt to new situations - like asking "Is \$100 USD enough for dinner for two in Toronto?" A natural language agent can figure out this requires currency knowledge plus local restaurant pricing. Meanwhile, MCP's strict schema approach means reliable, predictable results - perfect when you need exact answers to standard questions.

Difference 2: Task Lifecycle vs. Function Calls

A2A treats work as complete tasks with a lifecycle, while MCP treats them as individual function calls. It's like the difference between a project manager (A2A) and a calculator (MCP).

A2A's Multi-Stage Task Management

A2A is built around the concept of a "Task" that has a complete lifecycle with multiple states:

```
# A2A Task has explicit states in its lifecycle
{
  "id": "task123",
  "status": {
    "state": "running", # Can be: pending → running →
    completed/failed
    "startTime": "2025-04-12T10:30:00Z"
  },
  "artifacts": [...] # Partial results as they're created
}
```

Tasks can flow naturally through different states - they can start, require more input to produce partial results, and eventually complete or fail. The protocol itself manages this lifecycle, making

Looks like an article worth saving!

Option Q

Hover over the brain icon or use hotkeys to save with Memex.

MCP's Single-Stage

MCP operations are

Remind me later

Hide Forever

```
# MCP has no built-in "in-progress" or "needs more input" states
result = client.call_tool("get_exchange_rate", {"from": "USD", "to":
"CAD"})

# If arguments are invalid or missing, it simply fails with an error
# There's no protocol-level "input-required" state
```

If a tool needs more information, it can't request it within the protocol - it must either return an error or the client application must handle creating a new, separate request. Progress updates are optional add-ons, not core to the operation's lifecycle.

Why This Matters

Think about renovating a house vs. using individual tools. A2A is like hiring a contractor who manages the entire renovation project - they'll handle all the stages, give you updates, and work through unexpected issues. MCP is like calling individual specialists - the plumber does one thing, the electrician another, with no one tracking the overall project.

A2A shines for complex, multi-stage work where there's uncertainty and a need for adaptability. Research projects, creative work, and problem-solving all fit this model.

MCP excels when you need precise, individual operations with predictable inputs and outputs - calculations, data retrievals, and specific transformations.

Difference 3: High-Level Skills vs. Specific Functions

A2A describes what agents can do in general terms, while MCP spells out exactly which functions are available and how to call them.

A2A's Capability Descriptions

A2A describes an agent's capabilities like this:

Looks like an article worth saving! Option Q

Hover over the brain icon or use hotkeys to save with Memex.

```
# A2A AgentCard {
  agent_skill = AgentSkill {
    id="research_capability",
    ...
  }
}
```

Remind me later Hide Forever

```

    name="Research and Analysis",
    description="Can research topics and provide analysis.",
    examples=[
        "Research quantum computing advances",
        "Analyze market trends for electric vehicles"
    ]
    # Notice: no schema for how to invoke these skills!
)

```

It's like a resume that tells you what someone is good at, not exactly how to assign them work.

MCP's Function Specifications

MCP precisely defines each available function:

```

# MCP tool definition
{
  "name": "get_exchange_rate",
  "description": "Get exchange rate between currencies",
  "inputSchema": {
    "type": "object",
    "properties": {
      "currency_from": {"type": "string"},
      "currency_to": {"type": "string"}
    },
    "required": ["currency_from", "currency_to"]
  }
}

```

It's like an instruction manual that leaves nothing to interpretation.

Why This Matters

Think about delegating the research portion of your expertise. MCP is like a database using these

Looks like an article worth saving!

Hover over the brain icon or use hotkeys to save with Memex.

Remind me later

Hide Forever

Option

Q

rg
r
e

A2A's approach allows for creativity, initiative, and handling unexpected situations. The agent can adapt its approach based on what it discovers along the way.

MCP's approach ensures consistency, predictability, and precise control over exactly what happens and how.

The Philosophy Difference

These three technical differences reflect fundamentally different visions of AI collaboration:

- **MCP's Philosophy:** "Tell me EXACTLY what you want done and how to do it"
- **A2A's Philosophy:** "Tell me WHAT you want accomplished, I'll figure out HOW"

Think of MCP as building with Lego pieces - precise, predictable, combinable in countless ways. A2A is more like clay - flexible, adaptable, able to take forms you might not initially imagine.

The beauty is that you don't have to choose! Many sophisticated systems will implement both: MCP for precise tool execution and A2A for collaborative problem solving between specialized agents.

Conclusion: The Future is Collaborative AI

We've unpacked the key differences between A2A and MCP, and here's the bottom line: these aren't competing standards—they're complementary approaches solving different parts of the same puzzle.

Think of MCP as building precise tools and A2A as assembling skilled teams. The magic happens when you combine them:

- **MCP gives individual connections**
- Looks like an article worth saving!** Option Q
- Hover over the brain icon or use hotkeys to save with Memex.

- **A2A enables the**

Remind me later

Hide Forever

- **Together, they form an ecosystem** where specialized expertise and capabilities can flow seamlessly

For developers, this means you don't have to choose sides in some imaginary protocol war. Instead, ask yourself: "Do I need precise tool execution (MCP), agent collaboration (A2A), or both?" The answer depends entirely on your use case.

Where do we go from here? Expect to see increasing integration between these protocols as the AI ecosystem matures. The companies behind them understand that interoperability is the future—silos of AI Agents will become a thing of the past. The most powerful applications won't just use one Agent with a few tools; they'll orchestrate entire teams of specialized agents working together. As a developer, positioning yourself at this intersection means you'll be building the next generation of truly collaborative AI systems.

Looking to implement these protocols in your projects? Check out the minimal examples [MCP](#) and [A2A](#) on GitHub!

Thanks for reading Pocket Flow! Subscribe for free to receive new posts and support my work.



19 Likes • 1 Restack

← Previous

Next →

Looks like an article worth saving!

Option



Discussion about this article

Hover over the brain icon or use hotkeys to save with Memex.

Comments

Restacks

Remind me later

Hide Forever



Write a comment...

© 2025 Zachary Huang • [Privacy](#) • [Terms](#) • [Collection notice](#)
[Substack](#) is the home for great culture

Looks like an article worth saving!

Option



Hover over the brain icon or use hotkeys to save with Memex.

Remind me later

Hide Forever