

Build Simple AI Agent using Typescript

Presented by Paco Technical Lead/Deputy
Hive Master

19 October 2024



Who am I?



PACO Sarin Suriyakoon

Technical Lead/Deputy Hive Master at PALO IT

- Super app for Enterprise
- LLM for Enterprise
- Linkedin: "Sarin Suriyakoon"
- Linkedin QR Code



Who is PALO IT?



PALO^{IT}

A **global tech innovation consultancy** dedicated to achieving your business goals through next-level product-centric software delivery.

We provide expertise across the full product journey from ideation to launch and scale, using Sustainable IT practices as a commitment to crafting tech as a force for good.

50
Nationalities

18
Offices

10
Countries



25% Organic Growth

\$70m Turnover

700+ Technology experts
from across the globe

100% Independently-owned

0 Debt



PALO IT SERVICES



PALO^{IT}

 **Microsoft**
Gold Partner

GitHub Verified Partner

PRODUCT CONSULTING

1

New Digital Product Innovation

- Validate Product-Market Fit
- Formulate Implementation Plan
- Minimize Development Cost

2

Existing Product Optimization

- Increase Product Adoption
- Identify Product Improvements
- Improve Viability and Profitability



Contact Us

ORGANIZATION CONSULTING

Delivery Acceleration

3

- Reduce Time to Market
- Enable Value-Based Delivery
- Modernize Tech Infrastructure
- Align Business and Product Delivery
- Staff Empowerment

Sustainability For Positive Impact

4

- Sustainability Fundamentals
- ESG Data Management
- Enabling An Impact Mindset
- Tech For Good



Contact Us



Introduction

Motivation and Background



Introduction

- This project created to study about ReAct prompt engineering technique.
- Start out as an assistant that can operate the laptop. Inspired by OS Copilot and Mobile Agent paper
- It is designed to be a bot that execute shell command directly in the laptop.
- The task that aims to do are DevOps, Automation, Coding(Execute it too) combine that together we control the laptop with just human commands





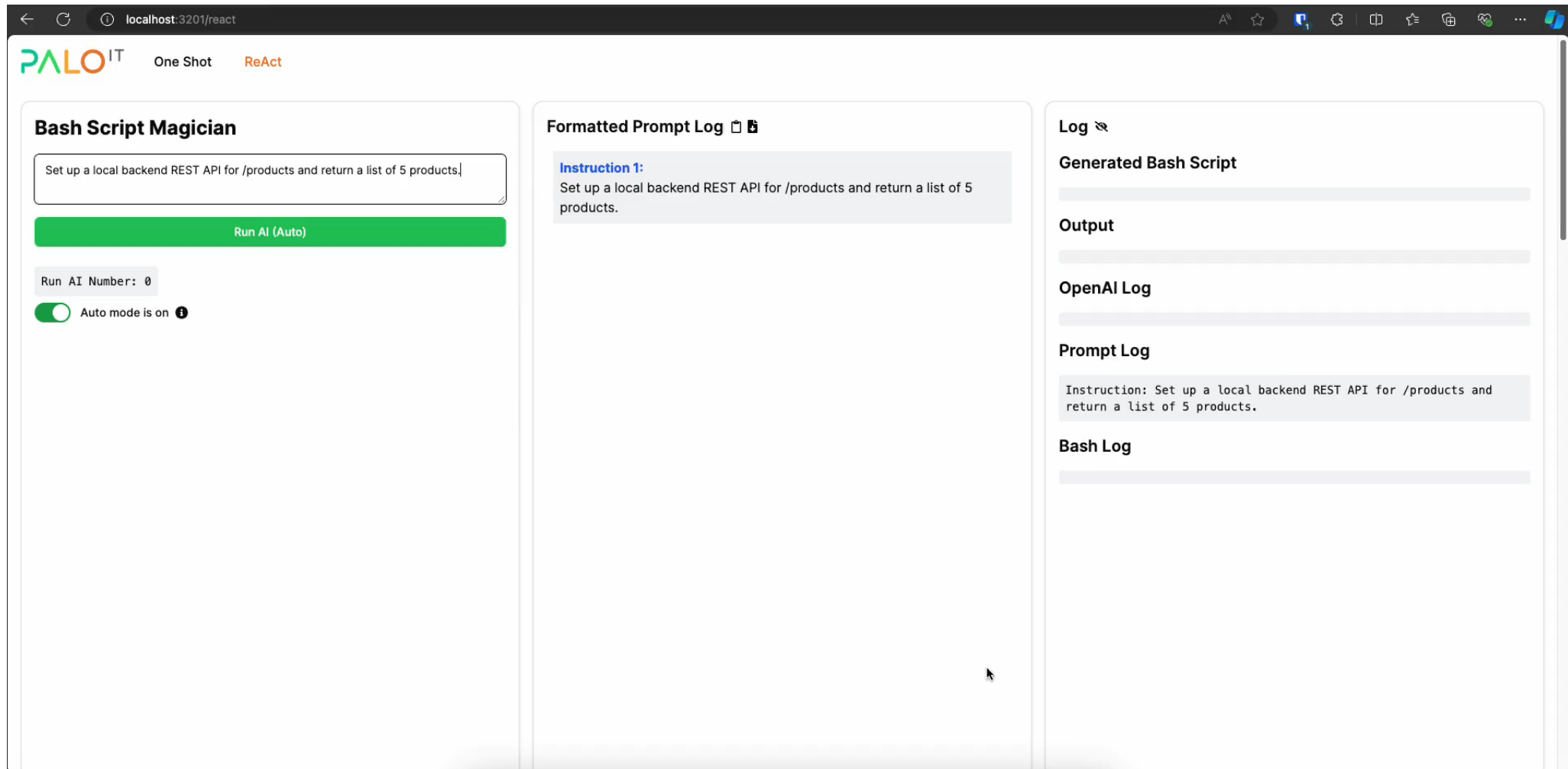
Shell Agent Showcase

Demo



Introduce Shell Agent with NextJS

- With the need to quickly build POC that help visualize how ReAct work we have...



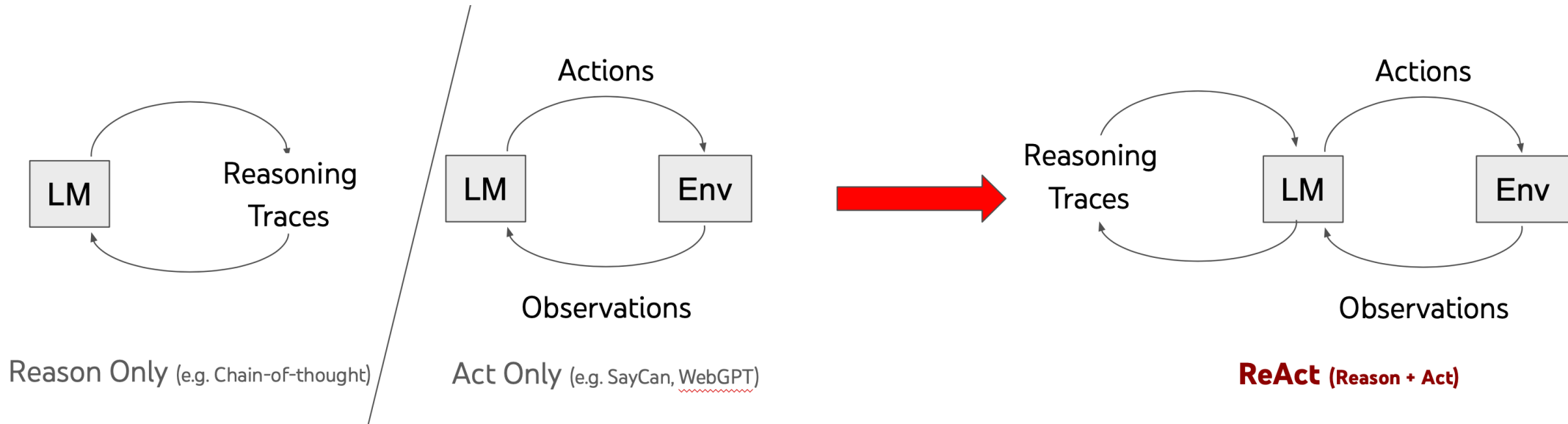
Introduce Shell Agent with NextJS

- Go to Shell Agent menu then input instruction that you want in the input box
- The instruction should be task that you need the agent to execute in your local laptop
- Auto Mode Off: Click "Run AI" alternating with "Run Bash Script"
- Auto Mode On: Click "Run AI(Auto)" and see the magic happen



What is ReAct?

- It came from a pioneer paper that describe agent. LLM that interact with real world.
- ReAct is Reasoning and Action(<https://react-lm.github.io/>)



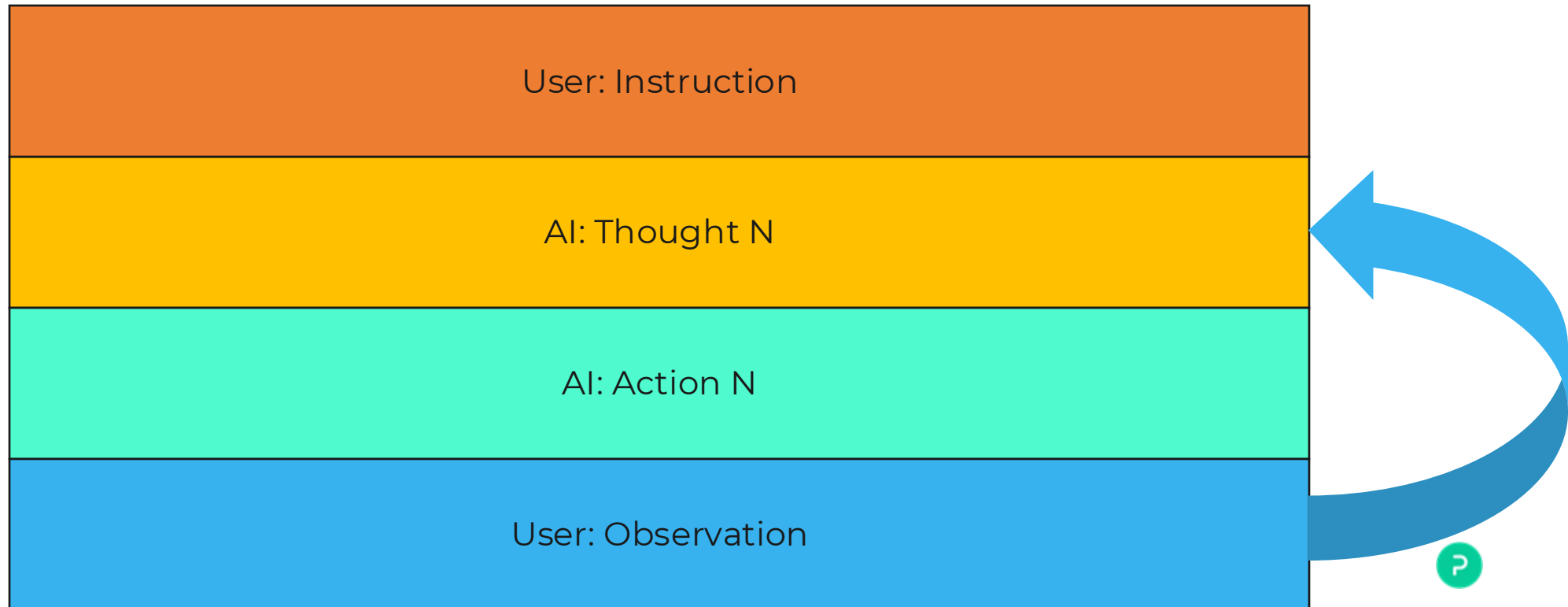
How original Paper manipulate result string?

- Let's take a quick look at <https://github.com/ysymyth/ReAct>
- <https://github.com/ysymyth/ReAct/blob/master/hotpotqa.ipynb>
- At `obs, r, done, info = step(env, action[0].lower() + action[1:])`

```
def webthink(idx=None, prompt=webthink_prompt, to_print=True):
    question = env.reset(idx=idx)
    if to_print:
        print(idx, question)
    prompt += question + "\n"
    n_calls, n_badcalls = 0, 0
    for i in range(1, 8):
        n_calls += 1
        thought_action = llm(prompt + f"Thought {i}:", stop=[f"\nObservation {i}:"])
        try:
            thought, action = thought_action.strip().split(f"\nAction {i}: ")
        except:
            print('ohh...', thought_action)
            n_badcalls += 1
            n_calls += 1
            thought = thought_action.strip().split('\n')[0]
            action = llm(prompt + f"Thought {i}: {thought}\nAction {i}:", stop=[f"\n"]).strip()
        obs, r, done, info = step(env, action[0].lower() + action[1:])
        obs = obs.replace('\n', '')
        step_str = f"Thought {i}: {thought}\nAction {i}: {action}\nObservation {i}: {obs}\n"
        prompt += step_str
        if to_print:
            print(step_str)
        if done:
            break
    if not done:
        obs, r, done, info = step(env, "finish[]")
    if to_print:
        print(info, '\n')
    info.update({'n_calls': n_calls, 'n_badcalls': n_badcalls, 'traj': prompt})
    return r, info
```

Conclusion of ReAct

- It's stacking reasoning and action and send to LLM and attach the result back.
- Thought-1, Action-1, Observation-1 ...Thought-N,Action-N,Observation-N
- Until the task is done



Example(Let's check with playground)

- <https://github.com/palo-it-th/PALO-LLM-Playground/blob/main/examples/basic.md>
- <https://github.com/palo-it-th/PALO-LLM-Playground/blob/main/examples/prompt.md>

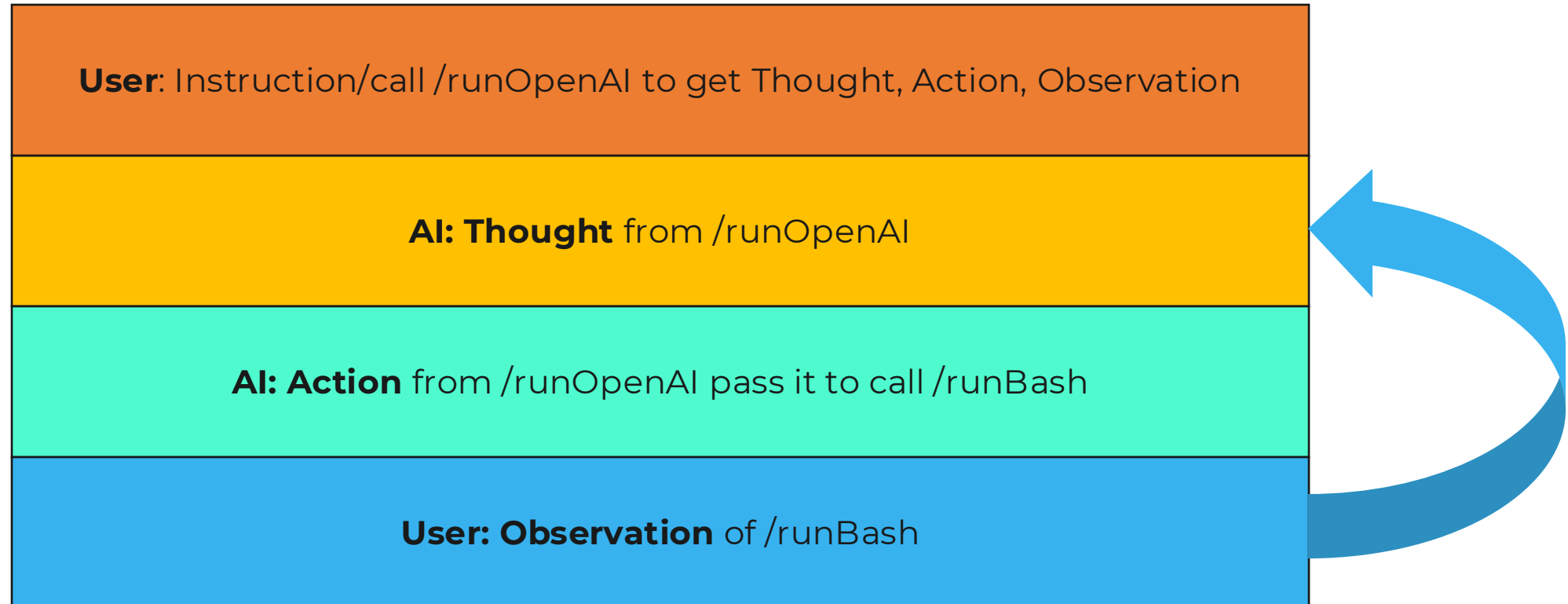


Implementation of Shell Agent

- Fullstack POC build with NextJS
- API 1 /runOpenAI use for calling OpenAI
- API 2 /runBash use for executing Bash command
- FE is responsible for handling and coordinate between /runOpenAI and /runBash and show the working status of the prompt



Prompt Structure



OpenAI SDK with Typescript explained

Chat with OpenAI 🗨️

Explain what is LLM like I am 5?

Okay! Imagine you have a magic box that knows a lot of stories and facts. When you ask it a question or tell it to make up a story, it uses all the things it knows to give you an answer or create a new story. This magic box is called an LLM, which stands for "Large Language Model." It's like a super-smart friend who has read a lot of books and can talk to you about almost anything!

OK explain openai sdk messages structure like I am 5 too!

Alright! Imagine you have a toy robot that can talk to you. To tell the robot what to say or do, you give it special cards with instructions. Each card has a few parts:

Role: This tells the robot who is talking. It could be you, the robot, or someone else. It's like saying, "This is from me!" or "This is from the robot!"

Content: This is the message itself. It's what you want to say or ask. Like, "What's your favorite color?" or "Tell me a story about a dragon."

Name (optional): Sometimes, you might want to give a name to the person talking, like "Mom" or "Dad," to make it clear who's speaking.

When you give these cards to the robot in the right order, it knows how to respond and keep the conversation going. This is how the OpenAI SDK messages work, helping computers understand and talk to us!

Structure Output with ReAct+Verification Step By Step

PALOIT

Shell Agent

Chat

Chat with Tools

RAG Chunking Raw Text

RAG Qdrant

Bash Script Magician

Create json-server that has mock claims api and show me the openapi spec

3+

Stop

Status: Prompt saved successfully to
./data/success/success_create_json_server_that_has_mock_claims_api_and

Run AI Number: 11

Working Directory \$HOME/workspace_react_1

workspace_react_1

Auto mode is on ⓘ

Example Instructions

Set up a local backend REST API for /products and return a list of 5 products.

Say Hello with Current Time

Formatted Prompt Log

system-0:

[SEE IN RAW FORMAT]

user-1:

Instruction: Create json-server that has mock claims api and show me the openapi spec

assistant-2:

Thought: I need to check if json-server is installed.

ActionType: Execute

ChildProcess: exec

Status: In Progress

Action: npm list -g json-server

user-3:

Observation: Error executing command in /Users/sarinsuriyakoon/workspace_react_1: Command failed: npm list -g json-server

assistant-4:

Information Classification: CONFIDENTIAL (sensitive business information, the level of protection is dictated by legal agreements)

System Prompt Review

- Let's check out the code!
- <https://github.com/palo-it-th/PALO-LLM-Playground/blob/main/src/app/api/runOpenAI/systemPromptJSON.ts>



Quick Demo!



Thank you! Let's do Q&A

PALO IT SERVICES



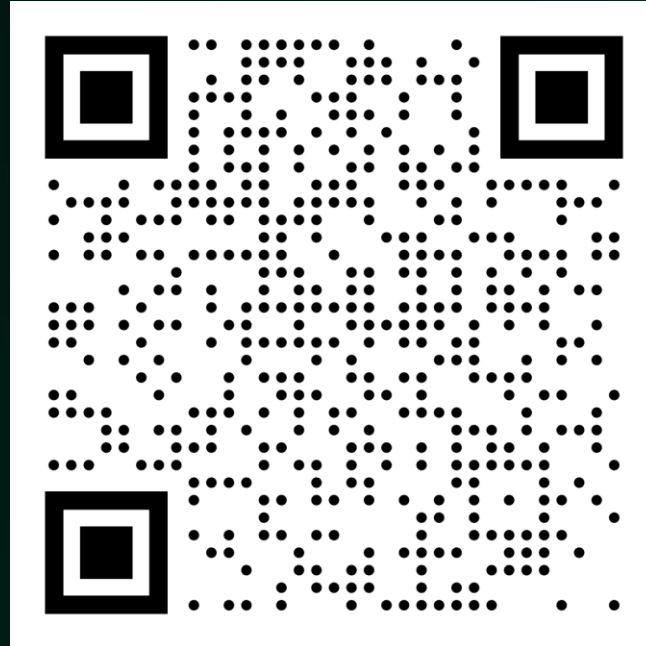
PALO^{IT}

 **Microsoft**
Gold Partner

GitHub Verified Partner

DOWNLOAD PRESENTATION

Build Simple AI Agent using Typescript



CONTACT US



02-180-6121



th-marketing@palo-it.com