Adding AI Capabilities to React Apps with Vercel AI SDK

6/26/2024, 5:57:15 PM

1/42 Adding AI Capabilities to React Apps with Vercel AI SDK

Welcome everyone! I'm excited to talk about adding AI capabilities to your React apps using the Vercel AI SDK. AI is revolutionizing how we build applications, and today we'll explore practical ways to incorporate it into your projects.

2/42 TOC

Let's take a look at what we'll cover today. We'll start with the basics of AI integration, move on to practical implementations, and finish with some advanced techniques.

3/42 About me

Before we dive in, let me introduce myself. I'm Nir Tamir, a senior frontend developer passionate about open source and tooling. I work with early-stage startups, helping them leverage cutting-edge technologies like AI.

4/42 You don't need to be a Data Scientist to be an AI Engineer

Now, I want to start with an important message: You don't need to be a data scientist to work with AI. As developers, we can leverage AI through APIs and SDKs, making it accessible for us to build intelligent applications.

5/42 AI Engineer

Let's talk about what an AI Engineer does. As this image shows, AI Engineers work on the application side of AI, using APIs and SDKs to integrate AI capabilities into software. We're not training models, but rather using them to solve real-world problems.

6/42 AI Engineer Skills

These are some of the key skills for AI Engineers. Notice how they blend traditional software engineering with AI-specific knowledge. We'll touch on many of these skills throughout our discussion today.

7/42 My journey

I'd like to share a personal anecdote that illustrates the power of AI. During a recent cruise where I had no internet access, I used open-source AI models running locally on my computer to help plan my trip. This experience showed me how AI can be a powerful tool even in offline scenarios.

8/42 Ollama

One of the tools that made this possible was Ollama. It's a fantastic platform for running open-source AI models locally. Let's dive into what makes Ollama special and how it can be useful for AI developers.

9/42 Ollama

Ollama offers several advantages. It's open-source, works offline, and provides a CLI interface. It supports vision models and has integrations with various development tools. Most importantly for us, it exposes a REST API that we can use in our applications.

10/42 Calling Ollama via REST API

Here's an example of how to call Ollama's API. Notice how similar it is to OpenAI's API structure. This compatibility makes it easier to switch between different AI providers in your applications.

11/42 Ollama is compatible with OpenAI interface

Here we are calling OpenAI API SDK, but changing the baseURL to point to Ollama's API. Because it's the same API - it should work the same way. Notice that some of the things are not supported by Ollama, like tools.

12/42 Calling Ollama via SDK

Ollama also provides an SDK, which simplifies the process of interacting with the AI models. Here's an example of how to use it for a simple chat interaction. We can also enable streaming for real-time responses, enhancing the user experience.

13/42 Calling Ollama via SDK

Let's see a practical example of using the Ollama SDK. This code snippet demonstrates how to send a simple question to the AI model and receive a response.

14/42 Calling Ollama via SDK Streaming

Here's how we can use streaming with the Ollama SDK. This approach allows us to receive and process the AI's response in real-time, piece by piece, which can significantly improve the perceived responsiveness of your application.

15/42 AI SDK

Now, let's shift our focus to the Vercel AI SDK. This toolkit provides a higher level of abstraction for working with AI models, making it even easier to integrate AI capabilities into your applications.

16/42

Here's the documentation for the Vercel AI SDK. It's a comprehensive resource that we'll be referring to throughout our discussion. I encourage you to explore it further after this presentation.

17/42 Getting started with Google Gemini AI

Let's start with a practical example using Google's Gemini AI. First, we'll install the necessary package and set up our environment variables. This setup process is crucial for securely connecting to the AI service.

18/42 AI SDK example

Here's a basic example of using the AI SDK with Google's Gemini model. We're setting up the AI, sending a prompt, and logging the response. This demonstrates how straightforward it is to interact with an AI model using the SDK.

19/42 AI SDK Switch models easily

Here is an example of how easy it is to change the AI provider in the SDK. One line of code to just change the model field

---

20/42 AI SDK Image example

The AI SDK also supports working with images. In this example, we're asking the AI to describe an image. This capability opens up many possibilities for applications involving visual content analysis.

21/42 AI SDK Object example

The AI SDK also allows us to generate structured objects. This is particularly useful when you need specific data formats from the AI's response. Here, we're using Zod to define a schema for a recipe, which the AI will then generate.

22/42 AI SDK Tools example

The AI SDK also supports the use of tools, which allow the AI to interact with external data or functions. In this example, we're creating a weather tool that the AI can use to get weather information for a specific location.

23/42 Zod auto infer types

We get automatic type inference from Zod

24/42 AI SDK Tools example with roundtrips

This example demonstrates a more complex interaction using tools and roundtrips. The AI can make multiple calls to the weather tool, allowing for more sophisticated and context-aware responses. This is particularly useful for scenarios where the AI needs to gather multiple pieces of information to formulate a complete answer.

25/42 Think about agents, tasks, real-time data, connect to external APIs...

The examples we've seen so far are just the beginning. As you delve deeper into AI integration, you can explore more advanced concepts like creating agents, defining complex tasks, incorporating real-time data, and connecting to various external APIs. The possibilities are vast and exciting.

26/42 AI SDK UI

Now, let's shift our focus to the UI aspects of AI integration. The AI SDK provides components and hooks that make it easy to create interactive AI-powered interfaces in your React applications. We'll explore how to build chat interfaces, completion components, and other AI-driven UI elements. It supported in multiple UI frameworks, like React, Solid, Vue, and Svelte. Notice that at this time most tools functionality is available only in React. https://sdk.vercel.ai/docs/ai-sdk-ui/overview#ui-framework-support

---

27/42 Chat UI with useChat

ai sdk exposes useChat that can be used to create chat UI. It returns the messages, and input + handleInputChange to control the input value, and handleSubmit that sends a post request to `` `/api/chat` `` with the message list

---

28/42 Chat Route

Here we define a POST route that receives the message list, calls the ai model and returns a stream of messages

---

29/42

useChate use swr internally to call this the chat route

30/42 **AI SDK RSC**

React Server Components (RSC) offer new possibilities for AI integration. With AI SDK RSC, we can stream AI-generated content directly from the server to the client, enabling highly dynamic and responsive AI-powered interfaces. This approach can significantly improve performance and user experience in AI-heavy applications.

31/42 **JavaScript generator functions**

We need to introduce the concept of generator functions because it's used when we want to create generativeUI. We define a generator with function*.

Here's how it works:

Normally, when you write a function in JavaScript, it runs from start to finish each time you call it. But a generator function is special because it can pause its execution and then pick up where it left off later.

Yield Statement: Within a generator function, the `yield` keyword is used to pause the function and return a value. When the function is called again, it resumes execution right after the `yield`.

Iterator Object: When you call a generator function, it doesn't execute immediately. Instead, it returns an iterator object. You interact with this object to control the execution of the generator.

Explanation

Calling `countToThree`: When you call `countToThree`, it returns an iterator object and doesn't start executing the function.

- `iterator.next()`: Calling `next()` starts the generator. It runs until it hits the first `yield`, returning the value after `yield`. The done property indicates if the generator has finished running.
- Resuming Execution: Each subsequent call to `next()` resumes execution from where it left off, running until the next `yield`.
- Completion: When the generator function has no more `yield` statements to execute, it returns `{ value: undefined, done: true }`.

Use Cases

Generator functions are particularly useful for:

- **Managing asynchronous code:** You can pause and resume functions to handle asynchronous operations more cleanly.
- **Lazy evaluation:** Generate values on the fly rather than computing everything upfront, which can be useful for handling large datasets.
- **Custom Iterators:** Create complex iteration logic that can be paused and resumed.

32/42 Stream UI - server action

First of all - we have a server action here.

Server Actions are asynchronous functions that are executed on the server. They can be used in Server and Client Components to handle form submissions and data mutations in Next.js applications.

It's like having a fetch client / trpc without the validations. We define them with `actions.tsx` and and use top-level `use server`;

Here we call `streamUI` which is like `generateText`. The text value renders the value of the generated tokens from the model response. Rendering JSX from the server is done by using a tool (like before) - which the model decide to call. As a reminder - a tool is just a function call. But now we can define the parameters as props, and return JSX instead. Every time we want to `await` - we don't want to block the stream - so we yield the value - in this case we yield the `<LoadingComponent>` then we fetch the weather data and return it when it's ready.

33/42 Stream UI - client component

In the client side - we just calll the server action, set the state and render it

---

35/42 Chrome 127 Experimental Built In AI Provider

An exciting development in the AI space is Chrome"s experimental built-in AI provider. This feature, available in Chrome 127, allows developers to access AI capabilities directly through the browser. Let's look at some early reactions and potential implications of this technology.

36/42 Chrome 127 Experimental Built In AI Provider

Here is a demo runnong ai in the browser. window.ai. 2 lines of code.

37/42 Chrome 127 Experimental Built In AI Provider

Let's look at a practical example of using Chrome"s built-in AI provider. This code demonstrates how to generate text using the Chrome AI, showcasing the simplicity and power of browser-based AI capabilities.

38/42 WebGPU enables to run more AI models in the browser

WebGPU is a game-changer for running AI models in the browser. It provides the necessary performance boost to execute complex AI tasks client-side, opening up new possibilities for responsive and privacy-preserving AI applications.

39/42 References

Throughout this presentation, we've covered a lot of ground. Here's a tweet that teach you a little bit more about the magic of AI. It's not related to the coding, but I'm highly recommend to watch it in order to unserstand the power of AI.

41/42 Thank you!

Thank you all for your attention! I hope this presentation has given you insights into how to add AI capabilities to your React apps using the Vercel AI SDK. Remember, the field of AI is rapidly evolving, and there's always more to learn. Feel free to reach out if you have any questions or want to discuss AI integration further.

42/42 Glossary

Retrieval-Augmented Generation (RAG) is the process of optimizing the output of a large language model, so it references an authoritative knowledge base outside of its training data sources before generating a response.