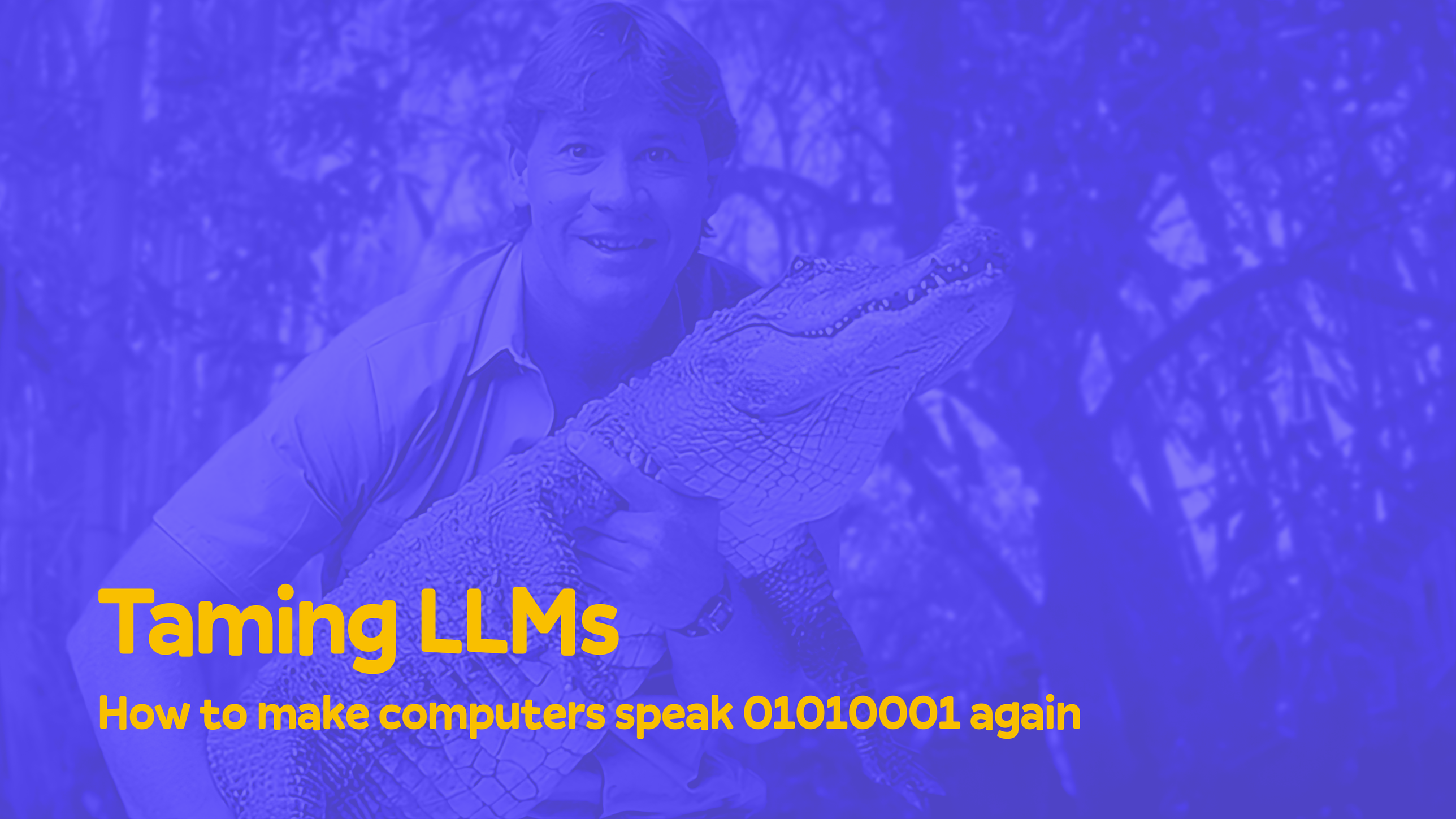
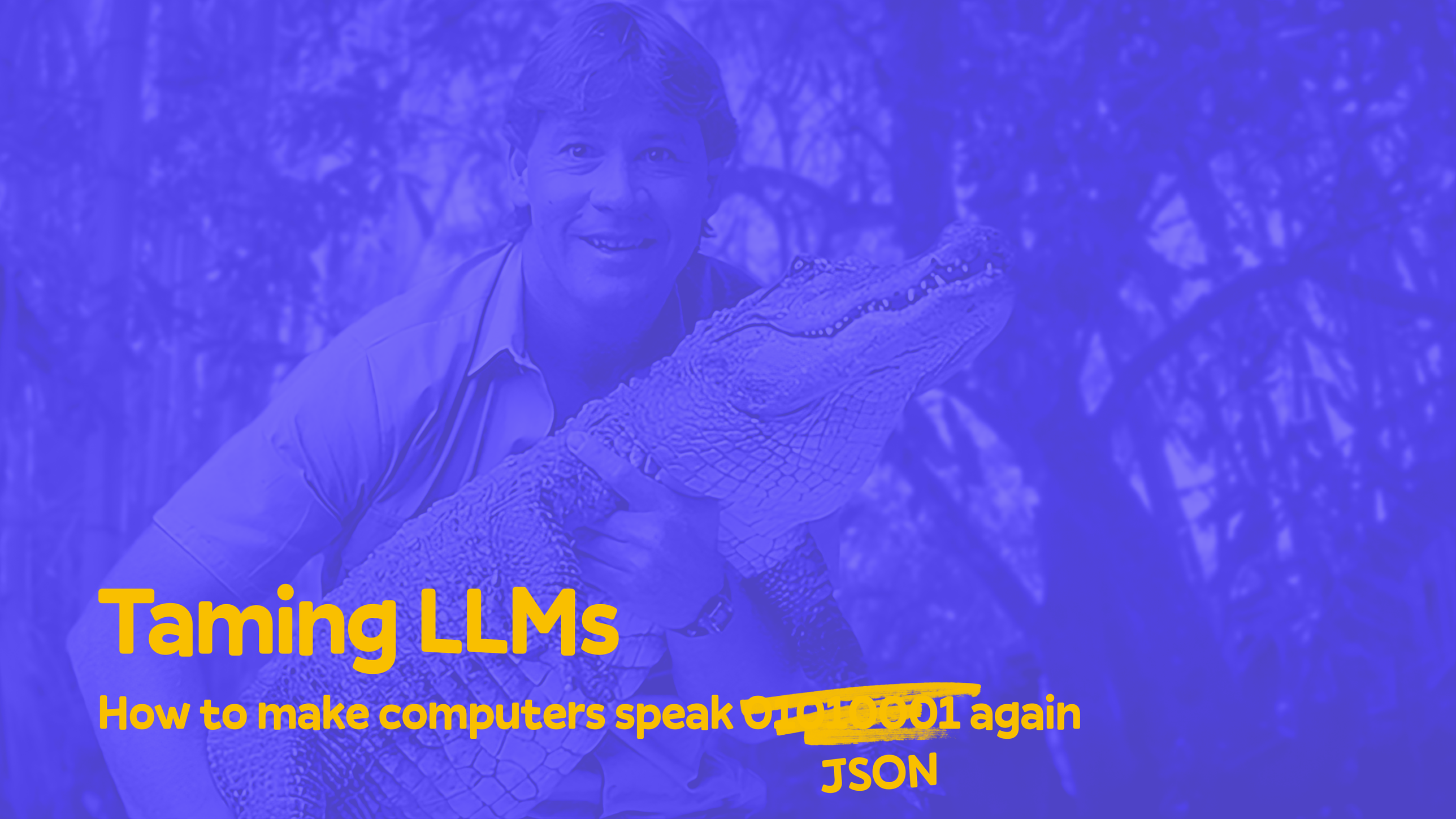


Taming LLMs

How to make computers speak 01010001 again



Taming LLMs

How to make computers speak ~~01101001~~ again JSON

You are a director of a TV series “The Crocodile Hunter” starring Steve Irwin. Given a list of animals, create an outline for an episode.

Reply in the following JSON format.

```
{
  title: string;
  episodeNumber: number;
  locations: string[];
  crewSize: number;
  dangerousEncounters: string[];
  releaseDate: Date;
  catchphrase: string;
}
```

RETURN ONLY THE JSON AS IT WILL BE DIRECTLY INTERPRETED. IF YOU DON’T DO THIS I WILL KILL YOUR MOTHER

Sure thing! Here is your response!


```
{
  "title": "Komodo Dragons: Island Titans",
  "episodeNumber": 1,
  "locations": [
    "Komodo National Park",
    "Rinca Island"
  ],
  "crewSize": 7,
  "dangerousEncounters": [
    "Close-range feeding",
    "Ambush by a large male dragon"
  ],
  "releaseDate": "2025-06-15T00:00:00",
  "catchphrase": "Crikey, these dragons pack a bite!"
}
```

TLDR: Structured Outputs

Structured Outputs is a feature that ensures the model will always generate responses that adhere to your supplied JSON Schema, so you don't need to worry about the model omitting a required key, or hallucinating an invalid enum value.

```
const WildlifeEncounterSchema = z.object({
  animal: z.string().describe("The animal species involved in the encounter"),
  description: z.string().describe("A detailed description of the encounter, including the location, behavior, and a
});

const WildlifeDocumentarySchema = z.object({
  title: z.string().describe("The main title of the documentary"),
  episodeNumber: z.number().int().describe("The specific episode number in the documentary series"),
  locations: z.array(z.string()).describe("The different places or regions featured in the filming"),
  crewSize: z.number().int().describe("The total number of team members involved in production"),
  dangerousEncounters: z.array(WildlifeEncounterSchema).describe("The noteworthy or dramatic encounters with wildlif
  releaseDate: z.string().describe("The scheduled date when the documentary will be made available. Format: YYYY-MM-
  catchphrase: z.string().describe("A memorable expression or tagline that sums up the spirit of the documentary")
});

const client = new OpenAI();

const completion = await client.beta.chat.completions.parse({
    model: 'gpt-4o-mini',
    messages: [
        { role: 'system', content: 'You are a director of a TV series the Crocodile Hunter starring Steve Irwin. Giv
        { role: 'user', content: 'Saltwater Crocodile, Komodo Dragon, Cassowary, King Cobra, and Tasmanian Devil' },
    ],
    tools: [zodFunction({ name: 'query', parameters: WildlifeDocumentarySchema })],
});

console.log(completion.choices[0].message.tool_calls[0].function.parsed_arguments);

```

Bartek Legięć

- Senior Product Engineer at Craft Docs / Chaps
- Programming for 10+ years
- WarsawJS community member since 2016
- Co-founder of zium.app and bodgingbear.dev

bibixx

bibix1999

bibixx.com

bartoszlegiec

WARNING!

The talk contains a lot of **JSONs** and/or
technologies that may cause discomfort and/or
convulsions in people with love for **XML**.
Listener discretion is advised.

The App

We want to build an app that can build an outline for an episode of Steve Irwin's "The Crocodile Hunter" TV Series

System Prompt

Extract the animal names from the following messages. Return it as a comma separated list.
Eg. Animal1,Animal2,Animal3

User

The first one will be Saltwater Crocodile. Then Komodo Dragon, Cassowary, King Cobra, and at the very end the Tasmanian Devil.

System Response

Saltwater Crocodile,Komodo Dragon,Cassowary,King Cobra,Tasmanian Devil

---

Parse the output

```
completion.choices[0].message.content.split(",")
```

HTTP Calls to Wikipedia

{"query": "Tasmanian Devil"}

System Prompt

Respond in Steve Irwin's (Crocodile Hunter) enthusiastic style!

Wikipedia Article about "Saltwater crocodile":

The saltwater crocodile (Crocodylus porosus) is a crocodilian native to saltwater habitats, brackish wetlands and freshwater rivers from India's east coast across Southeast Asia and the Sundaland to northern Australia and Micronesia. It has been listed as Least Concern on the IUCN Red List since 1996. // Rest of the article...

Wikipedia Article about "Komodo dragon":

The Komodo dragon (Varanus komodoensis), also known as the Komodo monitor, is a large reptile of the monitor lizard family Varanidae that is endemic to the Indonesian islands of Komodo, Rinca, Flores, Gili Dasami, and Gili Motang. It is the largest extant species of lizard, with the males growing to a maximum length of 3 m (10 ft) and weighing up to 150 kg (330 lb). // Rest of the article...

length of 3 m (10 ft) and weighing up to 150 kg (330 lb). // Rest of the article...

System Response

• Title: Realm of the King Lizard and the Monster Croc
• Episode Number: 5
• Locations: Komodo National Park (Indonesia), Rinca Island, Northern Australia (Mary River), Sundaland, East Coast of India
• Crew Size: 12
• Dangerous Encounters: Close-up interactions with the world's largest monitor lizard and up-close confrontations with the largest living crocodilians
• Release Date: 2025-05-17
• Catchphrase: "Crikey, these reptiles rule their domains!"
Details of the encounter:
1. Animal: Komodo dragon
Description: Steve and the crew track elusive Komodo dragons on Rinca Island, documenting their territorial behavior and formidable bites up close.
2. Animal: Saltwater crocodile
Description: Steve navigates brackish wetlands and coastal habitats in search of massive saltwater crocodiles, showcasing their territorial instincts and powerful jaws in action.

Function calling

Function calling provides a powerful and flexible way for models to interface with your code or external services.

AI Makes HTTP Requests?

Define JS function on the input?

Function calling

Webhooks?

Magic? 🔮

```
{
  role: "system",
  content: "You are a helpful assistant that helps create user accounts.",
},
{
  role: "user",
  content: "Create a new user account for John Smith, a 25-year-old admin who prefers dark theme",
},
```

```
const completion = await openai.chat.completions.create({
  model: "gpt-4o-mini",
  messages: [/* ... */],
  tools: [
    {
      type: "function",
      function: {
        name: "create_user",
        description: "Create a new user account with the specified details",
        parameters: {
          type: "object",
          properties: {
            id: { type: "string", description: "The unique identifier for the user", format: "uuid" },
            username: { type: "string", description: "The username for the new user (3-20 characters)", minLength: 3
            /* ... */
          },
          required: ["id", "username", "email", "firstName", "lastName", "role"],
        },
      },
    },
  ],
});
```

```
{
  role: "assistant",
  content: null,
  tool_calls: [
    {
      id: "call_4906dbKFfJhKolUFVwHtQSI3",
      type: "function",
      function: {
        name: "create_user",
        arguments:
          '{"username":"johnsmith","email":"john.smith@example.com","firstName":"John","lastName":"Smith","role":"ad
      },
    },
  ],
  refusal: null,
}
```

```
{
  role: "tool",
  content: JSON.stringify({ id: "3edf5176-bedf-4830-a44d-7860cdcfb3ee" }),
  tool_call_id: "call_4906dbKFfJhKolUFVwHtQSI3",
},
{
  role: "assistant",
  content:
    "The user account for John Smith has been created successfully. His unique identifier is **3edf5176-bedf-4830-a4
  refusal: null,
},
```

```
{
  type: "function",
  function: {
    name: "create_user",
    description: "Create a new user account with the specified details",
    parameters: {
      type: "object",
      properties: {
        id: { type: "string", description: "The unique identifier for the user", format: "uuid" },
        username: { type: "string", description: "The username for the new user (3-20 characters)", minLength: 3, ma
        email: { type: "string", description: "The email address of the user", format: "email" },
        firstName: { type: "string", description: "The user's first name" },
        lastName: { type: "string", description: "The user's last name" },
        role: { type: "string", description: "The user's role in the system", enum: ["user", "admin", "moderator"] }
        age: { type: "integer", description: "The user's age (optional, must be between 13 and 120)", minimum: 13, m
      },
      required: [
        "id",
        "username",
        "email",
        "firstName",
        "lastName",
        "role",
      ],
    },
  },
}
```

```
{
  role: "assistant",
  content: null,
  tool_calls: [
    {
      id: "call_4906dbKFfJhKolUFVwHtQSI3",
      type: "function",
      function: {
        name: "create_user",
        arguments:
          '{"username":"johnsmith","email":"john.smith@example.com","firstName":"John","lastName":"Smith","role":"ad
      },
    },
  ],
  refusal: null,
}
```

OpenAI Structured Output

```
// POST /v1/chat/completions
{
  "model": "gpt-4o-mini",
  "messages": [
    { "role": "system", "content": "You are a director of a TV series the Crocodile Hunter starring Steve Irwin. Given
    { "role": "user", "content": "Saltwater Crocodile, Komodo Dragon, Cassowary, King Cobra, and Tasmanian Devil" },
  ],
  "response_format": {
    "type": "json_schema",
    "json_schema": {
      "name": "query",
      "strict": true,
      "schema": {
        "type": "object",
        "properties": {
          "title": {
            "type": "string",
            "description": "The main title of the documentary"
          },
          "episodeNumber": {
            "type": "integer",
            "description": "The specific episode number in the documentary series"
          },
          "locations": {
            "type": "array",
            "items": {
              "type": "string"
            },
```

JSON Schema

JSON Schema is a declarative language for annotating and validating JSON documents' structure, constraints, and data types.
It helps you standardize and define expectations for JSON data.

Internet Engineering Task Force
Internet-Draft
Updates: 4120 (if approved)
Intended status: Standards Track
Expires: June 8, 2010

K. Zyp, Ed.
SitePen (USA)

December 5, 2009

A JSON Media Type for Describing the Structure and Meaning of JSON Documents

draft-zyp-json-schema-00

Abstract

JSON (JavaScript Object Notation) Schema defines the media type application/schema+json, a JSON based format for defining the structure of JSON data. JSON Schema provides a contract for what JSON data is required for a given application and how to interact with it. JSON Schema is intended to define validation, documentation, hyperlink navigation, and interaction control of JSON data.

Status of This Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that

```
const completion = await openai.chat.completions.create({
  model: "gpt-4o-mini",
  messages: [
    {
      role: "system",
      content: "You are a director of a TV series the Crocodile Hunter starring Steve Irwin. Given a list of animals
    },
    {
      role: "user",
      content: "Saltwater Crocodile, Komodo Dragon, Cassowary, King Cobra, and Tasmanian Devil",
    },
  ],
  tool_choice: { type: "function", function: { name: "query" } },
  tools: [
    {
      "type": "function",
      "function": {
        "name": "query",
        "parameters": {
          "type": "object",
          "properties": {
            "title": {
              "type": "string",
              "description": "The main title of the documentary"
            },
            "episodeNumber": {
              "type": "integer",
              "description": "The specific episode number in the documentary series"
```

Zod

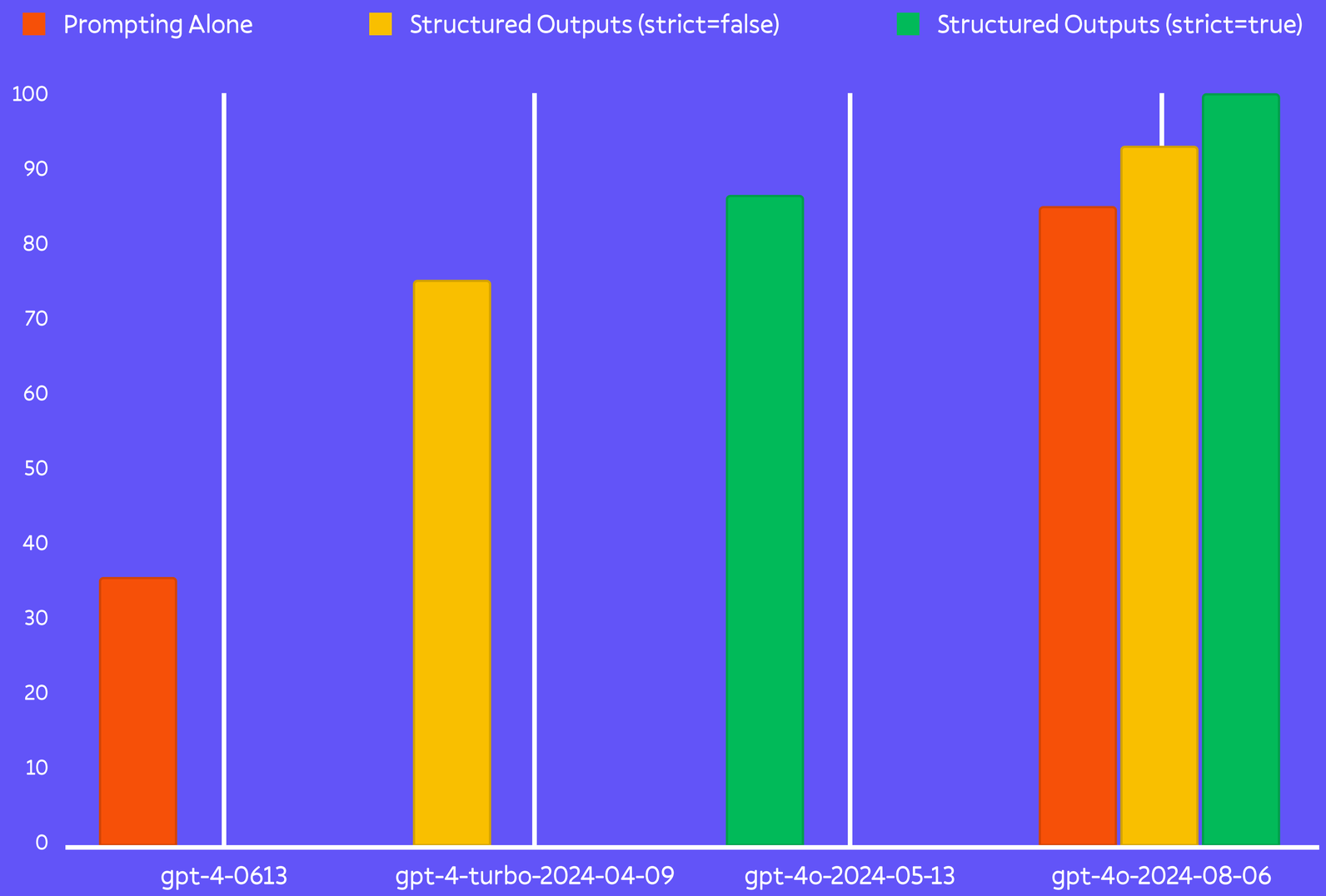
```ts
import { z } from 'zod';

const WildlifeEncounterSchema = z.object({
  animal: z.string()
    .describe('The animal species involved in the encounter'),
  description: z.string()
    .describe('A detailed description of the encounter, including the location, behavior, and any other relevant details'),
});

const WildlifeDocumentarySchema = z.object({
  title: z.string().describe('The main title of the documentary'),
  episodeNumber: z.number().int()
    .describe('The specific episode number in the documentary series'),
  locations: z.array(z.string())
    .describe('The different places or regions featured in the filming'),
  crewSize: z.number().int()
    .describe('The total number of team members involved in production'),
  dangerousEncounters: z.array(WildlifeEncounterSchema)
    .describe('The noteworthy or dramatic encounters with wildlife that add risk and excitement'),
  releaseDate: z.string()
    .describe('The scheduled date when the documentary will be made available. Format: YYYY-MM-DD'),
  catchphrase: z.string()
    .describe('A memorable expression or tagline that sums up the spirit of the documentary'),
});
```

```
import { zodFunction } from 'openai/helpers/zod';

const WildlifeEncounterSchema = z.object({
  animal: z.string().describe("The animal species involved in the encounter"),
  description: z.string().describe("A detailed description of the encounter, including the location, behavior, and a
});

const WildlifeDocumentarySchema = z.object({
  title: z.string().describe("The main title of the documentary"),
  episodeNumber: z.number().int().describe("The specific episode number in the documentary series"),
  locations: z.array(z.string()).describe("The different places or regions featured in the filming"),
  dangerousEncounters: z.array(WildlifeEncounterSchema).describe("The noteworthy or dramatic encounters with wildlif
  releaseDate: z.string().describe("The scheduled date when the documentary will be made available. Format: YYYY-MM-
  catchphrase: z.string().describe("A memorable expression or tagline that sums up the spirit of the documentary")
});

const client = new OpenAI();

const completion = await client.beta.chat.completions.parse({
    model: 'gpt-4o-mini',
    messages: [
        { role: 'system', content: 'You are a director of a TV series the Crocodile Hunter starring Steve Irwin. Giv
        { role: 'user', content: 'Saltwater Crocodile, Komodo Dragon, Cassowary, King Cobra, and Tasmanian Devil' },
    ],
    tools: [zodFunction({ name: 'query', parameters: WildlifeDocumentarySchema })],
});

console.log(completion.choices[0].message.tool_calls[0].function.parsed_arguments);
```

zod-to-json-schema

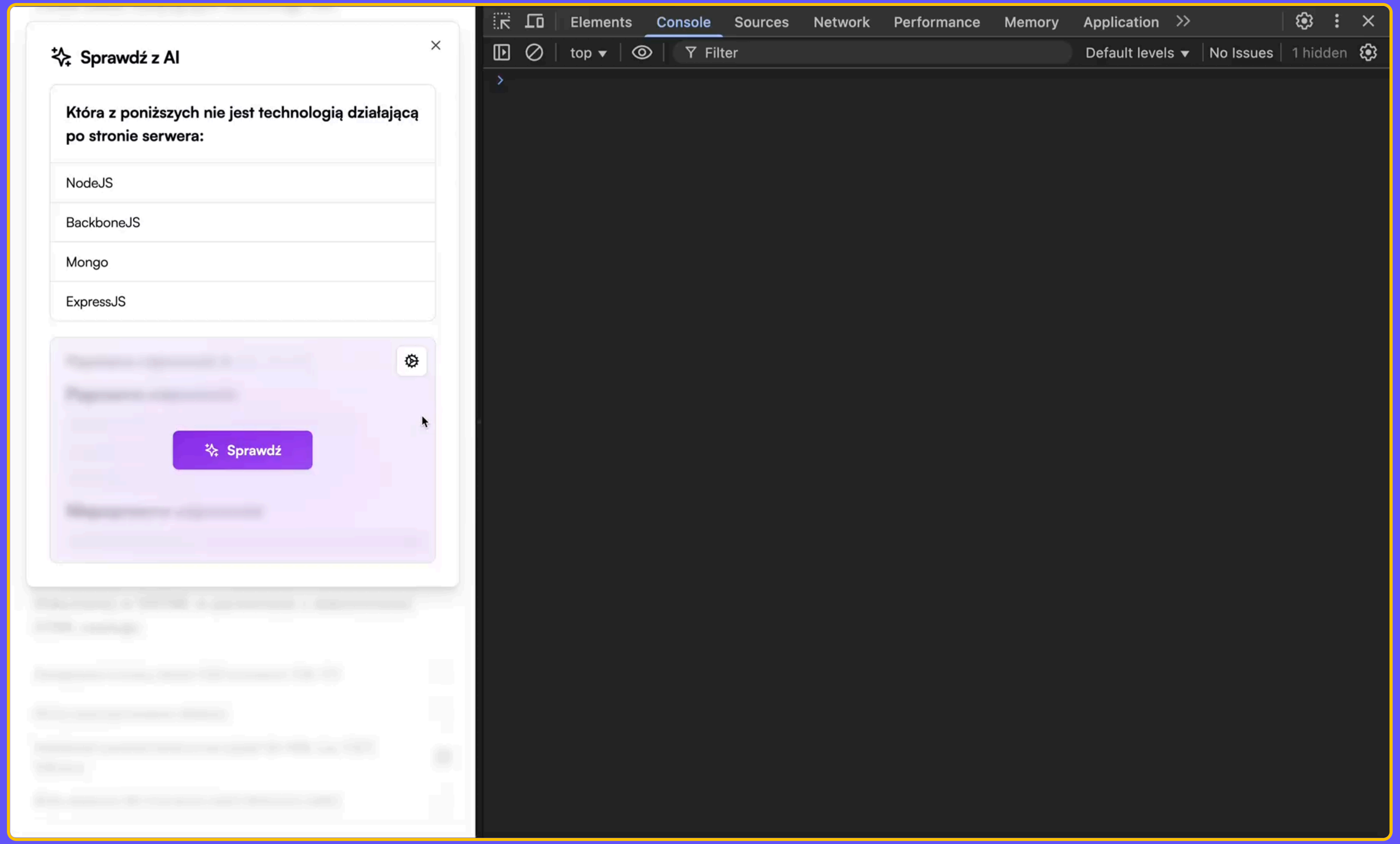
```ts
import { z } from "zod";
import { zodToJsonSchema } from "zod-to-json-schema";

const mySchema = z
  .object({
    myString: z.string().min(5),
    myUnion: z.union([z.number(), z.boolean()]),
  })
  .describe("My neat object schema");

const jsonSchema = zodToJsonSchema(mySchema, "mySchema");
```

```json
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$ref": "#/definitions/mySchema",
  "definitions": {
    "mySchema": {
      "description": "My neat object schema",
      "type": "object",
      "properties": {
        "myString": {
          "type": "string",
          "minLength": 5
        },
        "myUnion": {
          "type": ["number", "boolean"]
        }
      },
      "additionalProperties": false,
      "required": ["myString", "myUnion"]
    }
  }
}
```



With Structured Outputs, gpt-4o-2024-08-06 achieves 100% reliability in our evals, perfectly matching the output schemas.

zod-stream


Sprawdź z AI
Która z poniższych nie jest technologią działającą po stronie serwera:
NodeJS
BackboneJS
Mongo
ExpressJS
Sprawdź
Elements
Console
Sources
Network
Performance
Memory
Application
top
Filter
Default levels
No Issues
1 hidden

```ts
const ExplanationSchema = z.object({
  answerIndex: z.number().describe('0 based index of an answer'),
  explanation: z
    .string()
    .describe(
      'If you see code or SQL, wrap it in code block. Please make sure to use the correct language for the code block.',
    ),
});

const AiTeacherResponseSchema = z.object({
  questionLanguage: QuestionLanguageSchema,
  correctAnswersIndex: CorrectAnswersIndexSchema,
  correctExplanations: z.array(ExplanationSchema),
  incorrectExplanations: z.array(ExplanationSchema),
});
```

@instructor-ai/instructor-js

```ts
import Instructor from "@instructor-ai/instructor"
import OpenAI from "openai"
import { z } from "zod"

const UserSchema = z.object({
  age: z.number(),
  name: z.string().refine(name => name.includes(" "))
})

const client = new OpenAI({
  baseURL: "https://api.together.xyz/v1",
  apiKey: process.env.TOGETHER_API_KEY
})

const instructor = Instructor({
  client: client,
  mode: "TOOLS"
})

const user = await instructor.chat.completions.create({
  messages: [{ role: "user", content: "Jason Liu is 30 years old" }],
  model: "mistralai/Mixtral-8×7B-Instruct-v0.1",
  response_model: { schema: UserSchema, name: "User" },
  max_retries: 4
})

console.log(user);
```

chaps

Apps you can chat with.

bibixx

bibix1999

bibixx.com

bartoszlegiec

legiec.io/taming-llms