Copy page

Developer quickstart

Take your first steps with the OpenAI API.

The OpenAI API provides a simple interface to state-of-the-art AI models for text generation, natural language processing, computer vision, and more. This example generates text output from a prompt, as you might using ChatGPT.

Generate text from a model — javascript

```javascript
import OpenAI from "openai";
const client = new OpenAI();

const response = await client.responses.create({
    model: "gpt-5",
    input: "Write a one-sentence bedtime story about a unicorn."
});

console.log(response.output_text);
```

Configure your development environment

Install and configure an official OpenAI SDK to run the code above.

Responses starter app

Start building with the Responses API.

Text generation and prompting

Learn more about prompting, message roles, and building conversational apps.

Analyze images and files

Send image URLs, uploaded files, or PDF documents directly to the model to extract text, classify content, or detect visual elements.

Image URL | File URL | Upload file

Analyze the content of an image

javascript

```javascript
import OpenAI from "openai";
const client = new OpenAI();

const response = await client.responses.create({
    model: "gpt-5",
    input: [
        {
            role: "user",
            content: [
                {
                    type: "input_text",
                    text: "What is in this image?",
                },
                {
                    type: "input_image",
                    image_url: "https://openai-documentation.vercel.app/images/
                },
            ],
        },
    ],
});

console.log(response.output_text);
```

Image inputs guide

Learn to use image inputs to the model and extract meaning from images.

File inputs guide

Learn to use file inputs to the model and extract meaning from documents.

Extend the model with tools

Give the model access to external data and functions by attaching tools. Use built-in tools like web search or file search, or define your own for calling APIs, running code, or integrating with third-party systems.

Web search | File search | Function calling | Remote MCP

Use web search in a response

javascript

```javascript
import OpenAI from "openai";
const client = new OpenAI();

const response = await client.responses.create({
    model: "gpt-5",
    tools: [
        { type: "web_search" },
    ],
    input: "What was a positive news story from today?",
});

console.log(response.output_text);
```

Use built-in tools

Learn about powerful built-in tools like web search and file search.

Function calling guide

Learn to enable the model to call your own custom code.

Stream responses and build realtime apps

Use server-sent streaming events to show results as they're generated, or the Realtime API for interactive voice and multimodal apps.

Stream server-sent events from the API

javascript

```javascript
import { OpenAI } from "openai";
const client = new OpenAI();

const stream = await client.responses.create({
    model: "gpt-5",
    input: [
        {
            role: "user",
            content: "Say 'double bubble bath' ten times fast.",
        },
    ],
    stream: true,
});

for await (const event of stream) {
    console.log(event);
}
```

Use streaming events

Use server-sent events to stream model responses to users fast.

Get started with the Realtime API

Use WebRTC or WebSockets for super fast speech-to-speech AI apps.

Build agents

Use the OpenAI platform to build agents capable of taking action—like controlling computers—on behalf of your users. Use the Agents SDK for Python or TypeScript to create orchestration logic on the backend.

Build a language triage agent

javascript

```javascript
import { Agent, run } from '@openai/agents';

const spanishAgent = new Agent({
    name: 'Spanish agent',
    instructions: 'You only speak Spanish.',
});

const englishAgent = new Agent({
    name: 'English agent',
    instructions: 'You only speak English',
});

const triageAgent = new Agent({
    name: 'Triage agent',
    instructions:
        'Handoff to the appropriate agent based on the language of the request.
    handoffs: [spanishAgent, englishAgent],
});

const result = await run(triageAgent, 'Hola, ¿cómo estás?');
console.log(result.finalOutput);
```

Build agents that can take action

Learn how to use the OpenAI platform to build powerful, capable AI agents.