

[Changelog](#)

[Aug 15th, 2024](#)

[Aug 6st, 2024](#)

[Aug 1st, 2024](#)

[Jul 24th, 2024](#)

[Jul 23rd, 2024](#)

[Jul 18th, 2024](#)

[Jul 17th, 2024](#)

[Jun 6th, 2024](#)

[Jun 3rd, 2024](#)

[May 15th, 2024](#)

[May 13th, 2024](#)

[May 9th, 2024](#)

[May 7th, 2024](#)

[May 6th, 2024](#)

[May 2nd, 2024](#)

[Apr 29th, 2024](#)

[Apr 17th, 2024](#)

[Apr 16th, 2024](#)

[Apr 15th, 2024](#)

[Apr 9th, 2024](#)

[Apr 4th, 2024](#)

[Apr 1st, 2024](#)

[Mar 29th, 2024](#)

[Mar 14th, 2024](#)

[Feb 9th, 2024](#)

[Feb 1st, 2024](#)

[Jan 25th, 2024](#)

[Dec 20th, 2023](#)

[Dec 15th, 2023](#)

[Dec 14th, 2023](#)

[Nov 30th, 2023](#)

[Nov 6th, 2023](#)

[Oct 16th, 2023](#)

[Oct 6th, 2023](#)

[Text generation models](#)

[Quickstart](#)

[Python](#)

[Javascript - NodeJS](#)

[Curl](#)

[Prompt engineering](#)

[FAQ](#)

[Which model should I use?](#)

[How should I set the temperature parameter?](#)

[Is fine-tuning available for the latest models?](#)

[Do you store the data that is passed into the API?](#)

[How can I make my application more safe?](#)

[Should I use ChatGPT or the API?](#)

[Vision](#)

[Introduction](#)

[Quickstart](#)

[Python](#)

[Curl](#)

[Node Javascript](#)

[Uploading base 64 encoded images](#)

[Python](#)

[Multiple image inputs](#)

[Python](#)

[Node - Javascript](#)

[Curl](#)

[Low or high fidelity image understanding](#)

[Python](#)

[Node - Javascript](#)

[Curl](#)

[Managing images](#)

[Limitations](#)

[Calculating costs](#)

[FAQ](#)

[Can I fine-tune the image capabilities in gpt-4?](#)

[Can I use gpt-4 to generate images?](#)

[What type of files can I upload?](#)

[Is there a limit to the size of the image I can upload?](#)

[Can I delete an image I uploaded?](#)

[Where can I learn more about the considerations of GPT-4 with Vision?](#)

[How do rate limits for GPT-4 with Vision work?](#)

[Can GPT-4 with Vision understand image metadata?](#)

[What happens if my image is unclear?](#)

[Function calling](#)

[Introduction](#)

[Example use cases](#)

[The lifecycle of a function call](#)

[How to use function calling](#)

[Step 1: Pick a function in your codebase that the model should be able to call](#)

[Python](#)

[NodeJS - Javascript](#)

[Step 2: Describe your function to the model so it knows how to call it](#)

[Step 3: Pass your function definitions as available “tools” to the model, along with the messages](#)

[Python](#)

[NodeJS - Javascript](#)

[Step 4: Receive and handle the model response](#)

[If the model decides that no function should be called](#)

[Python](#)

[Javascript](#)

[Javascript - Node](#)

[If the model generated a function call](#)

[Python](#)

[Javascript - NodeJS](#)

[Handling the model response indicating that a function should be called](#)

[Python](#)

[NodeJS - Javascript](#)

[Step 5: Provide the function call result back to the model](#)

[Python](#)

[Javascript](#)

[Handling edge cases](#)

[Python](#)

[Javascript - Node](#)

[Function calling with Structured Outputs](#)

[Curl](#)

[Python](#)

[NodeJS - Javascript](#)

[Supported schemas](#)

[Customizing function calling behavior](#)

[Configuring parallel function calling](#)

[Python](#)

[Javascript - NodeJS](#)

[Python](#)

[Javascript - NodeJS](#)

[Parallel function calling and Structured Outputs](#)

[Configuring function calling behavior using the tool_choice parameter](#)

[Python](#)

[NodeJS - Javascript](#)

[Understanding token usage](#)

[Tips and best practices](#)

[Turn on Structured Outputs by setting strict: "true"](#)

[Name functions intuitively, with detailed descriptions](#)

[Name function parameters intuitively, with detailed descriptions](#)

[Consider providing additional information about how and when to call functions in your system message](#)

[Use enums for function arguments when possible](#)

[Keep the number of functions low for higher accuracy](#)

[Set up evals to act as an aid in prompt engineering your function definitions and system messages](#)

[Fine-tuning may help improve accuracy for function calling](#)

[FAQ](#)

[How do functions differ from tools?](#)

[Should I include function call instructions in the tool specification or in the system prompt?](#)

[Which models support function calling?](#)

[What are some example functions?](#)

[Can the model execute functions itself?](#)

[What are Structured Outputs?](#)

[Why might I not want to turn on Structured Outputs?](#)

[How do I ensure the model calls the correct function?](#)

[What does Structured Outputs mean for Zero Data Retention?](#)

[Resources](#)

[Structured Outputs](#)

[Introduction](#)

[Python](#)

[Javascript - Nodejs](#)

[When to use Structured Outputs via function calling vs via response_format](#)

[Structured Outputs vs JSON mode](#)

[Examples](#)

[Chain of thought](#)

[Python](#)

[Javascript - NodeJS](#)

[Curl](#)

[Example response](#)

[Structured data extraction](#)

[Python](#)

[Javascript-Nodejs](#)

[Curl](#)

[Example response](#)

[UI Generation](#)

[Python](#)

[Javascript-Nodejs](#)

[Curl](#)

[Example response](#)

[Moderation](#)

[Python](#)

[Javascript - NodeJS](#)

[Curl](#)

[Example response](#)

[How to use Structured Outputs with response_format](#)

[SDK Objects](#)

[Step 1: Define your schema](#)

[Tips for your JSON Schema](#)

[Step 2: Supply your schema in the API call](#)

[Python](#)

[Javascript - NodeJS](#)

[Curl](#)

[Step 3: Handle edge cases](#)

[Python](#)

[Javascript - Nodejs](#)

[Step 4: Use the generated structured data in a type-safe way](#)

[Python](#)

[Refusals with Structured Outputs](#)

[Python](#)

[Javascript - Nodejs](#)

[Tips and best practices](#)

[Handling user-generated input](#)

[Handling mistakes](#)

[Avoid JSON schema divergence](#)

[Supported schemas](#)

[Supported types](#)

[All fields must be required](#)

[Objects have limitations on nesting depth and size](#)

[additionalProperties: false must always be set in objects](#)

[Key ordering](#)

[Some type-specific keywords are not yet supported](#)

[For anyOf, the nested schemas must each be a valid JSON Schema per this subset](#)

[Definitions are supported](#)

[Recursive schemas are supported](#)

[JSON mode](#)

[Handling edge cases](#)

[Python](#)

[Javascript- NodeJS](#)

[Resources](#)

[Advanced Usage](#)

[Reproducible outputs](#)

[Beta](#)

[Managing tokens](#)

[Parameter details](#)

[Frequency and presence penalties](#)

[Token log probabilities](#)

[Other parameters](#)

[Chat Completions](#)

[Overview](#)

[Message roles](#)

[Getting started](#)

[Python](#)

[Javascript - NodeJS](#)

[Curl](#)

[Chat Completions response format](#)

[Python](#)

[Javascript - NodeJS](#)

[Next steps](#)

[Fine-tuning](#)

[Introduction](#)

[Which models can be fine-tuned?](#)

[When to use fine-tuning](#)

[Common use cases](#)

[Preparing your dataset](#)

[Example format](#)

[Multi-turn chat examples](#)

[Crafting prompts](#)

[Example count recommendations](#)

[Train and test splits](#)

[Token limits](#)

[Estimate costs](#)

[Check data formatting](#)

[Upload a training file](#)

[Python](#)

[Javascript-NodeJS](#)

[Curl](#)

[Create a fine-tuned model](#)

[Python](#)

[Javascript- NodeJS](#)

[Python](#)

[Javascript - NodeJS](#)

[Use a fine-tuned model](#)

[Python](#)

[Javascript - NodeJS](#)

[Use a checkpointed model](#)

[Analyzing your fine-tuned model](#)

[Iterating on data quality](#)

[Iterating on data quantity](#)

[Iterating on hyperparameters](#)

[Fine-tuning examples](#)

[Fine-tuning Integrations](#)

[Weights and Biases Integration](#)

[Authenticate your Weights and Biases account with OpenAI](#)

[Enable the Weights and Biases integration](#)

[View your fine-tuning job in Weights and Biases](#)

[FAQ](#)

[When should I use fine-tuning vs embeddings / retrieval augmented generation?](#)

[Can I fine-tune GPT-4o, GPT-4 Turbo or GPT-4?](#)

[How do I know if my fine-tuned model is actually better than the base model?](#)

[Can I continue fine-tuning a model that has already been fine-tuned?](#)

[How can I estimate the cost of fine-tuning a model?](#)

[How many fine-tuning jobs can I have running at once?](#)

[How do rate limits work on fine-tuned models?](#)

[Can I use the /v1/fine-tunes endpoint?](#)

Changelog

Keep track of changes to the OpenAI API. You can also track changes via our [public OpenAPI specification](#) which is used to generate our SDKs, documentation, and more. This changelog is maintained in a best effort fashion and may not reflect all changes being made.

Aug 15th, 2024

Released [dynamic model for chatgpt-4o-latest](#)—this model will point to the latest GPT-4o model used by ChatGPT.

Aug 6st, 2024

Launched [Structured Outputs](#)—model outputs now reliabilty adhere to developer supplied JSON Schemas.

Released [gpt-4o-2024-08-06](#), our newest model in the gpt-4o series.

Aug 1st, 2024

Launched [Admin and Audit Log APIs](#), allowing customers to programmatically administer their organization and monitor changes using the audit logs. Audit logging must be enabled within [settings](#).

Jul 24th, 2024

Launched [self-serve SSO configuration](#), allowing Enterprise customers on custom and unlimited billing to set up authentication against their desired IDP.

Jul 23rd, 2024

Launched [fine-tuning for GPT-4o mini](#), enabling even higher performance for specific use cases.

Jul 18th, 2024

Released **GPT-4o mini**, our affordable and intelligent small model for fast, lightweight tasks.

Jul 17th, 2024

Released **Uploads** to upload large files in multiple parts.

Jun 6th, 2024

Parallel function calling can be disabled in Chat Completions and the Assistants API by passing `parallel_tool_calls=false`.

.NET SDK launched in Beta.

Jun 3rd, 2024

Added support for **file search customizations**.

May 15th, 2024

Added support for **archiving projects**. Only organization owners can access this functionality.

Added support for **setting cost limits** on a per-project basis for pay as you go customers.

May 13th, 2024

Released **GPT-4o** in the API. GPT-4o is our fastest and most affordable flagship model.

May 9th, 2024

Added support for **image inputs** to the Assistants API.

May 7th, 2024

Added support for **fine-tuned models** to the Batch API.

May 6th, 2024

Added ``stream_options: {"include_usage": true}`` parameter to the Chat Completions and Completions APIs. Setting this gives developers access to usage stats when using streaming.

May 2nd, 2024

Added a new endpoint to delete a message from a thread in the Assistants API.

Apr 29th, 2024

Added a new function calling option ``tool_choice: "required"`` to the Chat Completions and Assistants APIs.

Added a guide for the Batch API and Batch API support for embeddings models

Apr 17th, 2024

Introduced a series of updates to the Assistants API, including a new file search tool allowing up to 10,000 files per assistant, new token controls, and support for tool choice.

Apr 16th, 2024

Introduced project based hierarchy for organizing work by projects, including the ability to create

API keys and manage rate and cost limits on a per-project basis (cost limits available only for Enterprise customers).

Apr 15th, 2024

Released Batch API

Apr 9th, 2024

Released GPT-4 Turbo with Vision in general availability in the API

Apr 4th, 2024

Added support for `seed` in the fine-tuning API

Added support for `checkpoints` in the fine-tuning API

Added support for adding Messages when creating a Run in the Assistants API

Apr 1st, 2024

Added support for filtering Messages by `run_id` in the Assistants API

Mar 29th, 2024

Added support for `temperature` and `assistant message creation` in the Assistants API

Mar 14th, 2024

Added support for `streaming` in the Assistants API

Feb 9th, 2024

Added `timestamp_granularities` parameter to the Audio API

Feb 1st, 2024

Released `gpt-3.5-turbo-0125`, an updated GPT-3.5 Turbo model

Jan 25th, 2024

Released `embedding V3` models and an updated GPT-4 Turbo preview

Added `dimensions` parameter to the Embeddings API

Dec 20th, 2023

Added `additional_instructions` parameter to run creation in the Assistants API

Dec 15th, 2023

Added `logprobs` and `top_logprobs` parameters to the Chat Completions API

Dec 14th, 2023

Changed `function parameters` argument on a tool call to be optional

Nov 30th, 2023

Released OpenAI Deno SDK

Nov 6th, 2023

Released GPT-4 Turbo Preview, updated GPT-3.5 Turbo, GPT-4 Turbo with Vision, Assistants API, DALL·E 3 in the API, and text-to-speech API

Deprecated the Chat Completions `functions` parameter in favor of `tools`

Released OpenAI Python SDK V1.0

Oct 16th, 2023

Added `encoding_format` parameter to the Embeddings API

Added `max_tokens` to the Moderation models

Oct 6th, 2023

Added `function calling` support to the Fine-tuning API