OpenAI Platform

Web search

Copy page

Allow models to search the web for the latest information before generating a response.

Web search allows models to access up-to-date information from the internet and provide answers with sourced citations. To enable this, use the web search tool in the Responses API or, in some cases, Chat Completions.

There are three main types of web search available with OpenAI models:

1. Non-reasoning web search: The non-reasoning model sends the user's query to the web search tool, which returns the response based on top results. There's no internal planning and the model simply passes along the search tool's responses. This method is fast and ideal for quick lookups.
2. Agentic search with reasoning models is an approach where the model actively manages the search process. It can perform web searches as part of its chain of thought, analyze results, and decide whether to keep searching. This flexibility makes agentic search well suited to complex workflows, but it also means searches take longer than quick lookups. For example, you can adjust GPT-5's reasoning level to change both the depth and latency of the search.
3. Deep research is a specialized, agent-driven method for in-depth, extended investigations by reasoning models. The model conducts web searches as part of its chain of thought, often tapping into hundreds of sources. Deep research can run for several minutes and is best used with background mode. These tasks typically use models like `o3-deep-research`, `o4-mini-deep-research`, or `gpt-5` with reasoning level set to `high`.

Using the Responses API, you can enable web search by configuring it in the `tools` array in an API request to generate content. Like any other tool, the model can choose to search the web or not based on the content of the input prompt.

Web search tool example — 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);
```

Output and citations

Model responses that use the web search tool will include two parts:

- A `web_search_call` output item with the ID of the search call, along with the action taken in `web_search_call.action`. The action is one of:
  - `search`, which represents a web search. It will usually (but not always) includes the search `query` and `domains` which were searched. Search actions incur a tool call cost (see pricing).
  - `open_page`, which represents a page being opened. Supported in reasoning models.
  - `find_in_page`, which represents searching within a page. Supported in reasoning models.
- A `message` output item containing:
  - The text result in `message.content[0].text`
  - Annotations `message.content[0].annotations` for the cited URLs

By default, the model's response will include inline citations for URLs found in the web search results. In addition to this, the `url_citation` annotation object will contain the URL, title and location of the cited source.

When displaying web results or information contained in web results to end users, inline citations must be made clearly visible and clickable in your user interface.

```
[
    {
        "type": "web_search_call",
        "id": "ws_67c9fa0502748190b7dd390736892e100be649c1a5ff9609",
        "status": "completed"
    },
    {
        "id": "msg_67c9fa077e288190af08fdffda2e34f20be649c1a5ff9609",
        "type": "message",
        "status": "completed",
        "role": "assistant",
        "content": [
            {
                "type": "output_text",
                "text": "On March 6, 2025, several news...",
                "annotations": [
                    {
                        "type": "url_citation",
                        "start_index": 2606,
```

```
                    "end_index": 2758,
                    "url": "https://...",
                    "title": "Title..."
                }
            ]
        }
    ]
}
]
```

Domain filtering

Domain filtering in web search lets you limit results to a specific set of domains. With the `filters` parameter you can set an allow-list of up to 20 URLs. When formatting URLs, omit the HTTP or HTTPS prefix. For example, use `openai.com` instead of `https://openai.com/`. This approach also includes subdomains in the search. Note that domain filtering is only available in the Responses API with the `web_search` tool.

Sources

To view all URLs retrieved during a web search, use the `sources` field. Unlike inline citations, which show only the most relevant references, sources returns the complete list of URLs the model consulted when forming its response. The number of sources is often greater than the number of citations. Real-time third-party feeds are also surfaced here and are labeled as `oai-sports`, `oai-weather`, or `oai-finance`. The sources field is available with both the `web_search` and `web_search_preview` tools.

List sources — curl

```curl
curl "https://api.openai.com/v1/responses" \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $OPENAI_API_KEY" \
-d '{
  "model": "gpt-5",
  "reasoning": { "effort": "low" },
  "tools": [
    {
      "type": "web_search",
      "filters": {
        "allowed_domains": [
          "pubmed.ncbi.nlm.nih.gov",
          "clinicaltrials.gov",
          "www.who.int",
          "www.cdc.gov",
          "www.fda.gov"
        ]
      }
    }
  ],
  "tool_choice": "auto",
  "include": ["web_search_call.action.sources"],
  "input": "Please perform a web search on how semaglutide is used in the treatı
}'
```

User location

To refine search results based on geography, you can specify an approximate user location using country, city, region, and/or timezone.

- The `city` and `region` fields are free text strings, like `Minneapolis` and `Minnesota` respectively.
- The `country` field is a two-letter ISO country code, like `US`.
- The `timezone` field is an IANA timezone like `America/Chicago`.

Note that user location is not supported for deep research models using web search.

Customizing user location — javascript

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

const response = await openai.responses.create({
    model: "o4-mini",
    tools: [{
        type: "web_search",
        user_location: {
            type: "approximate",
            country: "GB",
            city: "London",
            region: "London"
        }
    }],
    input: "What are the best restaurants near me?",
});
console.log(response.output_text);
```

Live internet access

Control whether the web search tool fetches live content or uses only cached/indexed results in the Responses API.

- Set `external_web_access: false` on the `web_search` tool to run in offline/cache-only mode.
- Default is `true` (live access) if you do not set it.

Preview variants (`web_search_preview`) ignore this parameter and behave as if `external_web_access` is `true` .

Control live internet access

curl

```curl
curl "https://api.openai.com/v1/responses" -H "Content-Type: application/json" -
  "model": "gpt-5",
  "tools": [
    { "type": "web_search", "external_web_access": false }
  ],
  "tool_choice": "auto",
  "input": "Find the sunrise time in Paris today and cite the source."
}'
```

API compatibility

Web search is available in the Responses API as the generally available version of the tool, `web_search` , as well as the earlier tool version, `web_search_preview` . To use web search in the Chat Completions API, use the specialized web search models `gpt-5-search-api` , `gpt-4o-search-preview` and `gpt-4o-mini-search-preview` .

Limitations

Web search is currently not supported in `gpt-5` with `minimal` reasoning, and `gpt-4.1-nano` .

When used as a tool in the Responses API, web search has the same tiered rate limits as the models above.

Web search is limited to a context window size of 128000 (even with `gpt-4.1` and `gpt-4.1-mini` models).

Usage notes

API AVAILABILITY	RATE LIMITS	NOTES
✓ Responses ✓ Chat Completions ⊗ Assistants	Same as tiered rate limits for underlying model used with the tool.	Pricing ZDR and data residency