OpenAI Platform

Web search

Copy page

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

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
    model: "gpt-5",
    tools: [
        { type: "web_search_preview" },
    ],
    input: "What was a positive news st
});

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

Web search tool versions

You can also force the use of the `web_search_preview` tool by using the `tool_choice` parameter, and setting it to `{type: "web_search_preview"}` - this can help ensure lower latency and more consistent results.

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. Only emitted by Deep Research models.
  - `find_in_page`, which represents searching within a page. Only emitted by Deep Research 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_67c9fa0502748190b7dd3
        "status": "completed"
    },
    {
        "id": "msg_67c9fa077e288190af08
        "type": "message",
        "status": "completed",
        "role": "assistant",
        "content": [
            {
                "type": "output_text",
                "text": "On March 6, 20
                "annotations": [
                    {
                        "type": "url_ci
                        "start_index":
                        "end_index": 27
                        "url": "https:/
                        "title": "Title
                    }
                ]
            }
        ]
    }
]
```

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
    model: "o4-mini",
    tools: [{
        type: "web_search_preview",
        user_location: {
            type: "approximate",
            country: "GB",
            city: "London",
            region: "London"
        }
    }],
    input: "What are the best restaurar
});
console.log(response.output_text);
```

Search context size

When using this tool, the `search_context_size` parameter controls how much context is retrieved from the web to help the tool formulate a response. The tokens used by the search tool do **not** affect the context window of the main model specified in the `model` parameter in your response creation request. These tokens are also **not** carried over from one turn to another — they're simply used to formulate the tool response and then discarded.

Choosing a context size impacts:

- **Cost**: Search content tokens are free for some models, but may be billed at a model's text token rates for others. Refer to pricing for details.
- **Quality**: Higher search context sizes generally provide richer context, resulting in more accurate, comprehensive answers.
- **Latency**: Higher context sizes require processing more tokens, which can slow down the tool's response time.

Available values:

- `high`: Most comprehensive context, slower response.
- `medium` (default): Balanced context and latency.
- `low`: Least context, fastest response, but potentially lower answer quality.

ⓘ Context size configuration is not supported for o3, o3-pro, o4-mini, and deep research models.

Customizing search context… javascript

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

const response = await openai.responses
    model: "gpt-4.1",
    tools: [{
        type: "web_search_preview",
        search_context_size: "low",
    }],
    input: "What movie won best picture
});
console.log(response.output_text);
```

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

Limitations

Web search is currently not supported in the `gpt-4.1-nano` model.

The `gpt-4o-search-preview` and `gpt-4o-mini-search-preview` models used in Chat Completions only support a subset of API parameters - view their model data pages for specific information on rate limits and feature support.

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).

Refer to this guide for data handling, residency, and retention information.