

[Documentation](#)

*search*

Search

- [rocket launch](#)

[Get started](#)

- [Installation](#)  
*add*
- [Fundamentals](#)  
*add*
- [First steps](#)  
*add*
- [code](#)

[Develop](#)

- [Concepts](#)  
*add*
- [API reference](#)  
*remove*
  - PAGE ELEMENTS

---
  - [Write and magic](#)  
*remove*
    - [st.write](#)
    - [st.write stream](#)
    - [magic](#)
  - [Text elements](#)  
*add*
  - [Data elements](#)  
*add*
  - [Chart elements](#)  
*add*
  - [Input widgets](#)  
*add*
  - [Media elements](#)  
*add*
  - [Layouts and containers](#)  
*add*
  - [Chat elements](#)  
*add*
  - [Status elements](#)  
*add*
  - [Third-party components](#)*open in new*
  - APPLICATION LOGIC

---
  - [Navigation and pages](#)  
*add*
  - [Execution flow](#)  
*add*
  - [Caching and state](#)  
*add*

- [Connections and secrets](#)  
*add*
  - [Custom components](#)  
*add*
  - [Utilities](#)  
*add*
  - [Configuration](#)  
*add*
  - TOOLS
- 
- [App testing](#)  
*add*
  - [Command line](#)  
*add*
  - [Tutorials](#)  
*add*
  - [Quick reference](#)  
*add*
  - [web\\_asset](#)

## [Deploy](#)

- [Concepts](#)  
*add*
- [Streamlit Community Cloud](#)  
*add*
- [Snowflake](#)
- [Other platforms](#)  
*add*
- [school](#)

## [Knowledge base](#)

- [FAQ](#)
- [Installing dependencies](#)
- [Deployment issues](#)
- [Home/](#)
- [Develop/](#)
- [API reference/](#)
- [Write and magic/](#)
- [st.write\\_stream](#)

# st.write\_stream



Streamlit Version  ▼

Stream a generator, iterable, or stream-like sequence to the app.

`st.write_stream` iterates through the given sequences and writes all chunks to the app. String chunks will be written using a typewriter effect. Other data types will be written using `st.write`.

## `st.write_stream(stream)`

### Parameters

The generator or iterable to stream.

`stream` (Callable, Generator, Iterable, OpenAI Stream, or LangChain Stream)

If you pass an async generator, Streamlit will internally convert it to a sync generator.

Note

To use additional LLM libraries, you can create a wrapper to manually define a generator function and include custom output parsing.

### Returns

(str or list)

The full response. If the streamed output only contains text, this is a string. Otherwise, this is a list of all the streamed objects. The return value is fully compatible as input for `st.write`.

### Example

You can pass an OpenAI stream as shown in our tutorial, [Build a basic LLM chat app](#). Alternatively, you can pass a generic generator function as input:

```
import time
import numpy as np
import pandas as pd
import streamlit as st

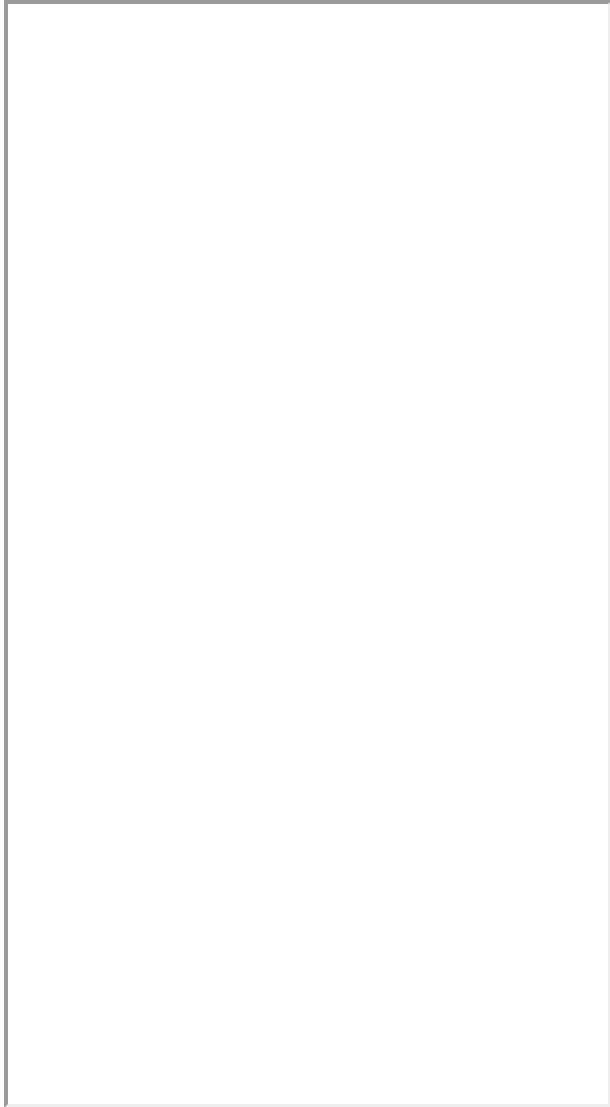
_LOREM_IPSUM = """
Lorem ipsum dolor sit amet, **consectetur adipiscing** elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
"""


def stream_data():
    for word in _LOREM_IPSUM.split(" "):
        yield word + " "
        time.sleep(0.02)

    yield pd.DataFrame(
        np.random.randn(5, 10),
        columns=["a", "b", "c", "d", "e", "f", "g", "h", "i", "j"],
    )

    for word in _LOREM_IPSUM.split(" "):
        yield word + " "
        time.sleep(0.02)

if st.button("Stream data"):
    st.write_stream(stream_data)
```



[Built with Streamlit](#)   
[Fullscreen open in new](#)

*star*

**Tip**

If your stream object is not compatible with `st.write_stream`, define a wrapper around your stream object to create a compatible generator function.

```
for chunk in unsupported_stream: yield preprocess(chunk)
```

For an example, see how we use [Replicate](#) with [Snowflake Arctic](#) in [this code](#).

[←Previous: st.write](#)[Next: magic→](#)

*forum*

**Still have questions?**

Our [forums](#) are full of helpful information and Streamlit experts.

[Home](#)[Contact Us](#)[Community](#)

