ChatGPT

Streamlit API 参考汇总（中文）

Streamlit 的 API 参考按类别组织。每个类别下都列出应用开发者可以调用的 Streamlit 函数、类或对象。下表概述了各类别包含的 API 项目数量，表中只统计顶层的 Streamlit 命令/类，不包括返回对象的方法（例如 `st.area_chart().add_rows`）。

API 类别	示例（简短说明）	不同 API 项目的数量
写入与魔法	通用输出函数 `st.write`（显示任意对象），流式输出支持 `st.write_stream`，以及魔法命令 [1]	**3**
文本元素	格式化函数，如渲染 Markdown 的 `st.markdown`、标题/副标题 (`st.title`、`st.header`、`st.subheader`)、徽章 `st.badge`、说明文字、代码块、数学公式等 [2] [3]	**13**
数据元素	显示数据的函数：交互表格 (`st.dataframe`、`st.data_editor`)、列配置类、静态表格 `st.table`、指标卡 `st.metric`、JSON 显示 `st.json` [4] [5]	**6** 个函数和 **16** 个列配置类
图表元素	简单图表（面积图、柱状图、折线图、散点图和地图）以及 Altair、Plotly、Graphviz、Vega-Lite、PyDeck 和 Matplotlib 的高级封装 [6]	**12**
输入组件	按钮、表单提交按钮、文件上传器以及选择类组件（复选框、单选框、多选、滑块、日期时间输入、聊天输入、摄像头/音频输入等） [7]	**25**
媒体元素	在应用中嵌入图片、Logo、PDF、音频和视频 [8]	**5**
布局与容器	用于组织布局的容器：列、容器、对话框、可展开面板、侧边栏、弹出按钮和标签页 [9]	**8**
聊天元素	构建聊天界面的函数——消息容器 `st.chat_message`、输入框 `st.chat_input` 和可更新的状态容器 `st.status` [10]	**3**
状态元素	进度条、加载旋转、通用状态容器、Toast 消息、气球和雪花动画，以及成功、信息、警告、错误、异常等提示 [11]	**11**
认证与用户信息	原生认证：`st.login` 发起认证流程，`st.logout` 清除身份，`st.user` 获取当前用户信息 [12]	**3**
导航与页面	构建多页面应用的函数：`st.navigation`、`st.Page`、`st.page_link` 以及程序化跳转的 `st.switch_page` [13]	**4**

API 类别	示例（简短说明）	不同 API 项目的数量
执行流程与表单	创建模态对话框（`st.dialog`）、表单容器和提交按钮（`st.form`、`st.form_submit_button`）、独立片段（`st.fragment`），以及控制脚本重跑（`st.rerun`）或停止执行（`st.stop`） [14]	**6**
缓存与状态	数据缓存装饰器 `st.cache_data`、资源缓存 `st.cache_resource`，以及持久化会话状态、请求上下文和查询参数的对象 [15] [16]	**5**
连接与密钥	数据库连接 `st.connection` 及其连接类（如 `SQLConnection`、`SnowflakeConnection`）和读取密钥的 `st.secrets` [17]	**5**
自定义组件	声明自定义组件、展示原始 HTML 或加载远程网页（`st.components.v1.declare_component`、`st.components.v1.html`、`st.components.v1.iframe`） [18]	**3**
配置	读取/设置配置选项（`st.get_option`、`st.set_option`），设置页面配置 `st.set_page_config` 以及 `config.toml` 文件 [19]	**4**
测试	使用 `st.testing.v1.AppTest` 进行无头测试 [20]	**1**
命令行接口	`streamlit run`、`streamlit cache clear`、`streamlit config show`、`streamlit docs`、`streamlit hello`、`streamlit help`、`streamlit init`、`streamlit version` 等 CLI 命令 [21]	**8**

全部类别共计约 **124 个顶层 API 项目**（函数、类和 CLI 命令）。下文详细阐述各类别 API 的功能及使用方法。

写入与魔法

- **`st.write`**：用于显示任意类型的数据。它根据参数类型选择合适的显示方式，例如字符串会调用 `st.markdown`，数据框会调用 `st.dataframe`，异常对象会显示异常 [1]。
- **`st.write_stream`**：接收生成器或可迭代对象，逐块输出并返回完整字符串（或列表）。它通过 `st.write` 输出每个块，实现打字机效果 [22]。
- **魔法命令**：在主脚本中直接书写变量或表达式时，Streamlit 自动调用 `st.write` 进行显示，这种隐式调用称为“魔法” [23]。

文本元素

文本元素用于格式化和展示文本内容：

- **`st.markdown`**：渲染 GitHub 风格的 Markdown，支持表情、彩色文字、LaTeX、Google Material 图标等 [2]。
- **标题函数**：`st.title`、`st.header` 和 `st.subheader` 用于显示标题、页眉和副标题，可指定锚点和宽度 [3] [24]。
- **`st.badge`**：显示带图标的彩色徽章，例如“Beta”、“开源”标识 [25]。
- **`st.caption`**：显示较小的说明文字或脚注，支持 Markdown [26]。
- **`st.code`**：显示代码块，可设置语言、高亮、自动换行、容器高度和宽度 [27]。
- **`st.divider`**：渲染水平分隔线，相当于 Markdown 的 `---` [28]。

- **`st.echo`**：用作上下文管理器，在块内既显示代码又执行代码，可通过 `code_location` 调整代码的位置 [29]。
- **`st.latex`**：显示 LaTeX 数学公式，接受字符串或 SymPy 表达式 [30]。
- **`st.text`**：显示原始文本，不解析 Markdown 或 HTML [31]。
- **`st.help`**：展示任意 Python 对象的文档和函数签名 [32]。
- **`st.html`**：在应用中插入原始 HTML，使用 DOMPurify 进行安全过滤 [33]。

数据元素

- **`st.dataframe`**：以交互式表格显示数据结构（pandas、pyarrow、polars、Snowpark 等）。支持设置宽高、隐藏索引、列顺序、列配置、行选择方式、回调函数等 [4]。通过返回对象的 `add_rows` 方法可以动态追加数据 [34]。
- **`st.data_editor`**：允许用户编辑数据，与 `st.dataframe` 支持相同的数据类型，并提供行数控制、`on_change` 回调、列配置等功能，提交后返回编辑后的数据 [35]。
- **`st.column_config`**：用于自定义列的显示方式。包含多个类：`Column`（基类）、`TextColumn`、`NumberColumn`、`CheckboxColumn`、`SelectboxColumn`、`DatetimeColumn`、`DateColumn`、`TimeColumn`、`JSONColumn`、`ListColumn`、`LinkColumn`、`ImageColumn`、`AreaChartColumn`、`LineChartColumn`、`BarChartColumn`、`ProgressColumn` [36]。
- **`st.table`**：显示静态表格，适用于小规模数据或不需要交互的表格，可在单元格中渲染 Markdown [37]。
- **`st.metric`**：展示指标值，配有粗体字体和可选的增减变化指示。参数包括 `label`、`value`、`delta`、`delta_color`、`border`、`width`、`height`，可在列中并排展示多个指标 [5]。
- **`st.json`**：以可展开的树状结构美化显示对象或 JSON 字符串，通过 `expanded` 控制是否默认展开 [38]。

图表元素

Streamlit 提供简单图表和高级图表封装 [6]：

- **简单图表**：`st.area_chart`、`st.bar_chart`、`st.line_chart`、`st.scatter_chart` 和 `st.map`。这些函数接受数据（DataFrame、pyarrow Table 或 numpy 数组），自动生成相应图表，并返回一个可以调用 `add_rows` 追加数据的对象 [39]。
- **高级图表**：`st.altair_chart`（Altair/Vega-Lite 图）、`st.bokeh_chart`（已弃用）、`st.graphviz_chart`（Graphviz 图）、`st.plotly_chart`（Plotly 图）、`st.pydeck_chart`（PyDeck 地图）、`st.pyplot`（Matplotlib 图）、`st.vega_lite_chart`（直接使用 Vega-Lite 规范）。这些函数一般接受第三方库生成的对象，并提供诸如 `use_container_width` 等选项。

输入组件

输入组件用于收集用户输入 [7]：

- **按钮**：`st.button` 显示普通按钮；`st.download_button` 用于下载文件；`st.link_button` 打开外部链接；`st.page_link` 跳转至其他页面；`st.form_submit_button` 用于表单提交 [40]。
- **选择类组件**：`st.checkbox`（复选框）、`st.toggle`（开关）、`st.radio`（单选按钮）、`st.pills`（胶囊式单选）、`st.segmented_control`（水平分段选择）、`st.selectbox`（下拉选择）、`st.multiselect`（多选）、`st.select_slider`（选择区间）、`st.slider`（数值滑块）、`st.number_input`（数值输入框）、`st.color_picker`（取色器）、`st.feedback`（收集用户反馈）。
- **日期/时间输入**：`st.date_input`、`st.time_input` 收集日期或时间。

- **文本输入**：`st.text_input` 用于单行文本；`st.text_area` 收集多行文本；`st.chat_input` 用于聊天应用的输入 [10]。
- **媒体输入**：`st.file_uploader` 上传文件；`st.audio_input` 录制音频；`st.camera_input` 拍摄图片或视频；`st.data_editor` 用于编辑表格（同时属于数据元素）。

所有组件都支持常见参数：`label`（标签）、`value`（默认值）、`key`（唯一键）、`help`（提示）、`disabled`、`on_change`（回调）以及传递回调参数的 `args` / `kwargs`。部分组件（例如 `st.slider`、`st.date_input`）还有 `min_value`、`max_value`、`step` 和 `format` 等参数。

媒体元素

- **`st.image`**：显示图片，可接受文件、URL 或 numpy 数组；
- **`st.logo`**：渲染应用的 Logo；
- **`st.pdf`**：嵌入 PDF 文档；
- **`st.audio`**：播放音频文件或数组；
- **`st.video`**：嵌入视频文件或 URL [8]。每个函数都接受数据源、格式选项和媒体开始时间等参数。

布局与容器

- **`st.columns(n, gap)`**：创建 `n` 列，返回多个容器，可在其中写入元素。
- **`st.container()`**：返回通用容器，用于将元素组合在一起。
- **`st.dialog()`**：创建模态对话框，可放置部件并独立运行逻辑，可设置 `key`、`title` 和图标等。
- **`st.empty()`**：保留一个空区域，可后续填充任何元素。
- **`st.expander(label, expanded)`**：添加可折叠面板，在展开后显示内容。
- **`st.popover()`**：显示一个小的弹出框锚定在按钮上。
- **`st.sidebar`**：返回侧边栏容器；在 `st.sidebar` 下调用函数会将元素放到侧边栏。
- **`st.tabs()`**：创建标签页并返回标签容器 [9]。所有容器支持使用 `with` 语法进行上下文管理。

聊天元素

- **`st.chat_message(role, avatar)`**：向聊天会话中添加消息，可在容器内写入文本、图表等 [10]。
- **`st.chat_input(placeholder)`**：在页面底部显示输入框，返回用户输入的消息。
- **`st.status(label)`**：创建可持续更新的状态容器，可用返回对象的 `update()` 方法更改状态（加载、成功、错误等）[41]。

状态元素

用于显示进度和通知 [11]：

- **`st.progress(value)`**：绘制横向进度条，接受 0.0–1.0 或整数。返回对象可更新进度。
- **`st.spinner(text)`**：在长运算期间显示旋转动画。
- **`st.status`**：同上，用于多步状态更新 [41]。
- **`st.toast(message)`**：显示短暂的 Toast 消息。
- **动画效果**：`st.balloons()` 放飞彩色气球；`st.snow()` 触发雪花动画。
- **提示函数**：`st.success`、`st.info`、`st.warning`、`st.error` 和 `st.exception` 显示不同风格的提示 [11]。

认证与用户信息

Streamlit 支持原生用户认证。调用 `st.login()` 发起认证流程；`st.logout()` 清除会话；`st.user` 返回当前用户的 ID、姓名和邮箱等信息 [12]。启用认证需要在 `config.toml` 中开启对应功能。

导航与页面

用于构建多页面应用 [13]：

- **`st.navigation`**：公开一个注册表来配置页面路径和元数据；
- **`st.Page`**：表示一个页面，定义其标题、图标和路径；
- **`st.page_link()`**：创建一个跳转按钮链接到其他页面；
- **`st.switch_page(target)`**：在代码中程序化跳转到指定页面。所有页面函数可设定 `label`、`icon`、`disabled` 等参数。

执行流程与表单

Streamlit 提供控制脚本执行和组织用户输入的命令 [14]：

- **`st.dialog()`**：打开模态对话框，其内代码与主应用独立运行，可指定是否打开。
- **`st.form(key, clear_on_submit, enter_to_submit, border, width, height)`**：创建表单容器，该容器包含内置的提交按钮，只有当用户点击提交按钮时表单内的所有值才会发送到服务器 [42]。表单不能嵌套，也不能包含普通按钮 [43]。
- **`st.form_submit_button`**：在表单内显示提交按钮，可设置标签、帮助信息、回调函数、图标等 [44]。
- **`st.fragment(func, run_every)`**：将函数装饰为独立片段，在输入变化时只重跑该片段，减少全局重跑 [45]。可使用 `run_every` 指定自动重跑间隔，并在片段内调用 `st.rerun(scope="fragment")` 触发片段重跑 [46]。
- **`st.rerun(scope="app"|"fragment")`**：立即停止当前运行并调度重跑，`scope` 为 `app` 时重跑整个应用，为 `fragment` 时重跑片段 [47]。频繁使用可能导致无限循环 [48]。
- **`st.stop()`**：立即停止执行，不再运行后续语句，经常用于在缺少必需输入时终止脚本 [49]。

缓存与状态

为提升性能和维护状态，Streamlit 提供以下工具：

- **`st.cache_data`**：基于函数参数缓存数据；适用于数据加载或计算 [15]。
- **`st.cache_resource`**：缓存全局资源，例如数据库连接或模型 [15]。
- **`st.session_state`**：类字典对象，持久化用户状态（例如组件值）到下一次重跑 [16]。
- **`st.context`**：暴露请求级上下文，可访问 `headers` 和 `cookies` [16]。
- **`st.query_params`**：读取或设置 URL 查询参数，方便持久化页面状态或生成分享链接 [16]。

连接与密钥

内置连接器用于访问外部数据源 [17]：

- **`st.connection(name, type, options)`**：返回连接对象。内置类型包括 `SQLConnection`（通用 SQL 数据库）、`SnowflakeConnection`（Snowflake 数据库）；基类 `BaseConnection` 定义通用接口。连接对象通常提供 `query` 方法执行 SQL 并获取数据。

- `st.secrets`：读取 `.streamlit/secrets.toml` 中的密钥并返回字典，用于存储 API 密钥或数据库凭据 [17]。

自定义组件

- `st.components.v1.declare_component`：注册自定义 JavaScript 组件，需提供前端代码和唯一名称。
- `st.components.v1.html`：在 iframe 中显示任意 HTML，可嵌入自定义 HTML、CSS 或 JavaScript。
- `st.components.v1.iframe`：嵌入远程网页 [18]。

配置

Streamlit 的配置可通过 `config.toml` 文件或编程方式设置 [19]：

- `st.get_option(option_name)`：获取配置选项当前值；
- `st.set_option(option_name, value)`：在运行时设置配置选项；
- `st.set_page_config`：设置页面标题、图标、布局、侧边栏展开状态和初始主题；
- **配置文件**：`~/.streamlit/config.toml` 用于存储持久的配置。

测试

使用 `st.testing.v1.AppTest` 可无头测试 Streamlit 应用。`AppTest` 能在测试环境中运行应用、操作组件并断言输出，适合自动化测试交互应用 [20]。

命令行接口

Streamlit 命令行工具提供以下命令 [21]：

- `streamlit run`：运行 Streamlit 应用；
- `streamlit cache clear`：清除所有缓存的数据和资源；
- `streamlit config show`：显示当前配置；
- `streamlit docs`：打开文档；
- `streamlit hello`：启动内置示例应用；
- `streamlit help`：显示 CLI 使用说明；
- `streamlit init`：创建样板项目；
- `streamlit version`：显示安装版本号。

结论

Streamlit 提供全面的 API，用于以纯 Python 构建交互式 Web 应用。文档按类别组织，以帮助开发者快速找到需要的函数：无论是展示数据、构建表单、嵌入媒体、管理导航、缓存耗时操作、连接数据库还是扩展自定义组件。通过组合这些 API，开发者可以快速构建数据驱动的仪表板、聊天界面或多页面应用。

---

[1] st.write - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/write-magic/st.write

[2] st.markdown - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/text/st.markdown

3 st.title - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/text/st.title

4 st.dataframe - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/data/st.dataframe

5 st.metric - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/data/st.metric

6 Chart elements - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/charts

7 Input widgets - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/widgets

8 Media elements - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/media

9 Layouts and Containers - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/layout

10 Chat elements - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/chat

11 Display progress and status - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/status

12 Authentication and user info - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/user

13 Navigation and pages - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/navigation

14 Execution flow - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/execution-flow

15 16 Caching and state - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/caching-and-state

17 Connections and databases - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/connections

18 Custom components - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/custom-components

19 Configuration - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/configuration

20 App testing - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/app-testing

21 Command-line options - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/cli

22 st.write_stream - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/write-magic/st.write_stream

23 Magic - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/write-magic/magic

24 st.header - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/text/st.header

25 st.badge - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/text/st.badge

26 st.caption - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/text/st.caption

27 st.code - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/text/st.code

28 st.divider - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/text/st.divider

29 st.echo - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/text/st.echo

30 st.latex - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/text/st.latex

31 st.text - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/text/st.text

32 st.help - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/text/st.help

33 st.html - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/text/st.html

34 39 st.area_chart - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/charts/st.area_chart

35 st.data_editor - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/data/st.data_editor

36 st.column_config - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/data/st.column_config

37 st.table - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/data/st.table

38 st.json - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/data/st.json

40 44 st.form_submit_button - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/execution-flow/st.form_submit_button

41 st.status - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/status/st.status

42 43 st.form - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/execution-flow/st.form

45 46 st.fragment - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/execution-flow/st.fragment

47 48 st.rerun - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/execution-flow/st.rerun

49 st.stop - Streamlit Docs
https://docs.streamlit.io/develop/api-reference/execution-flow/st.stop