Streamlit API cheat sheet

This is a summary of the docs for the latest version of Streamlit, v1.49.0.

Install & Import

```
pip install streamlit

streamlit run first_app.py

# Import convention
>>> import streamlit as st
```

Pre-release features

```
pip uninstall streamlit
pip install streamlit-nightly --upgrade
```

Learn more about experimental features

Command line

```
streamlit cache clear
streamlit config show
streamlit docs
streamlit hello
streamlit help
streamlit init
streamlit run streamlit_app.py
streamlit version
```

Magic commands

```
# Magic commands implicitly
# call st.write().
"_This_ is some **Markdown**"
my_variable
"dataframe:", my_data_frame
```

Display text

```
st.write("Most objects") # df, err, func, keras!
st.write(["st", "is <", 3])
st.write_stream(my_generator)
st.write_stream(my_llm_stream)

st.text("Fixed width text")
st.markdown("_Markdown_")
st.latex(r""" e^{i\pi} + 1 = 0 """)
st.title("My title")
st.header("My header")
st.subheader("My sub")
st.code("for i in range(8): foo()")
st.badge("New")
st.html("<p>Hi!</p>")
```

Display data

```
st.dataframe(my_dataframe)
st.table(data.iloc[0:10])
st.json({"foo":"bar","fu":"ba"})
st.metric("My metric", 42, 2)
```

Display media

```
st.image("./header.png")
st.logo("logo.jpg")
st.pdf("my_document.pdf")
st.audio(data)
st.video(data)
st.video(data, subtitles="./subs.vtt")
```

Display charts

```
st.area_chart(df)
st.bar_chart(df)
st.bar_chart(df, horizontal=True)
st.line_chart(df)
st.map(df)
st.scatter_chart(df)

st.altair_chart(chart)
st.graphviz_chart(fig)
st.plotly_chart(fig)
st.pydeck_chart(chart)
st.pyplot(fig)
st.vega_lite_chart(df, spec)

# Work with user selections
event = st.plotly_chart(
    df,
    on_select="rerun"
)
event = st.altair_chart(
    chart,
    on_select="rerun"
)
event = st.vega_lite_chart(
    df,
```

```
    spec,
    on_select="rerun"
)
```

To use Bokeh, see our custom component `streamlit-bokeh`.

Add elements to sidebar

```
# Just add it after st.sidebar:
a = st.sidebar.radio("Select one:", [1, 2])

# Or use "with" notation:
with st.sidebar:
    st.radio("Select one:", [1, 2])
```

Columns

```
# Two equal columns:
col1, col2 = st.columns(2)
col1.write("This is column 1")
col2.write("This is column 2")

# Three different columns:
col1, col2, col3 = st.columns([3, 1, 1])
# col1 is larger.

# Bottom-aligned columns
col1, col2 = st.columns(2, vertical_alignment="bottom")

# You can also use "with" notation:
with col1:
    st.radio("Select one:", [1, 2])
```

Tabs

```
# Insert containers separated into tabs:
tab1, tab2 = st.tabs(["Tab 1", "Tab2"])
tab1.write("this is tab 1")
tab2.write("this is tab 2")

# You can also use "with" notation:
with tab1:
    st.radio("Select one:", [1, 2])
```

Expandable containers

```
expand = st.expander("My label", icon=":material/info:")
expand.write("Inside the expander.")
pop = st.popover("Button label")
pop.checkbox("Show all")

# You can also use "with" notation:
with expand:
    st.radio("Select one:", [1, 2])
```

Control flow

```
# Stop execution immediately:
st.stop()
# Rerun script immediately:
st.rerun()
# Navigate to another page:
st.switch_page("pages/my_page.py")

# Define a navigation widget in your entrypoint file
pg = st.navigation(
    st.Page("page1.py", title="Home", url_path="home", default=True)
    st.Page("page2.py", title="Preferences", url_path="settings")
)
pg.run()

# Group multiple widgets:
with st.form(key="my_form"):
    username = st.text_input("Username")
    password = st.text_input("Password")
    st.form_submit_button("Login")

# Define a dialog function
@st.dialog("Welcome!")
def modal_dialog():
    st.write("Hello")
```

```
modal_dialog()

# Define a fragment
@st.fragment
def fragment_function():
    df = get_data()
    st.line_chart(df)
    st.button("Update")

fragment_function()
```

Display interactive widgets

```
st.button("Click me")
st.download_button("Download file", data)
st.link_button("Go to gallery", url)
st.page_link("app.py", label="Home")
st.data_editor("Edit data", data)
st.checkbox("I agree")
st.feedback("thumbs")
st.pills("Tags", ["Sports", "Politics"])
st.radio("Pick one", ["cats", "dogs"])
st.segmented_control("Filter", ["Open", "Closed"])
st.toggle("Enable")
st.selectbox("Pick one", ["cats", "dogs"])
st.multiselect("Buy", ["milk", "apples", "potatoes"])
st.slider("Pick a number", 0, 100)
st.select_slider("Pick a size", ["S", "M", "L"])
st.text_input("First name")
st.number_input("Pick a number", 0, 10)
st.text_area("Text to translate")
st.date_input("Your birthday")
st.time_input("Meeting time")
st.file_uploader("Upload a CSV")
st.audio_input("Record a voice message")
st.camera_input("Take a picture")
st.color_picker("Pick a color")
```

```
# Use widgets' returned values in variables:
for i in range(int(st.number_input("Num:"))):
    foo()
if st.sidebar.selectbox("I:",["f"]) == "f":
    b()
my_slider_val = st.slider("Quinn Mallory", 1, 88)
st.write(slider_val)

# Disable widgets to remove interactivity:
st.slider("Pick a number", 0, 100, disabled=True)
```

Build chat-based apps

```
# Insert a chat message container.
with st.chat_message("user"):
    st.write("Hello 👋")
    st.line_chart(np.random.randn(30, 3))

# Display a chat input widget at the bottom of the app.
st.chat_input("Say something")

# Display a chat input widget inline.
with st.container():
    st.chat_input("Say something")
```

Learn how to Build a basic LLM chat app

Mutate data

```
# Add rows to a dataframe after
# showing it.
element = st.dataframe(df1)
element.add_rows(df2)

# Add rows to a chart after
# showing it.
element = st.line_chart(df1)
element.add_rows(df2)
```

Display code

```
with st.echo():
    st.write("Code will be executed and printed")
```

Placeholders, help, and options

```
# Replace any single element.
element = st.empty()
element.line_chart(...)
element.text_input(...)  # Replaces previous.

# Insert out of order.
elements = st.container()
elements.line_chart(...)
st.write("Hello")
elements.text_input(...)  # Appears above "Hello".

# Horizontal flex
flex = st.container(horizontal=True)
flex.button("A")
flex.button("B")

st.help(pandas.DataFrame)
st.get_option(key)
st.set_option(key, value)
st.set_page_config(layout="wide")
st.query_params[key]
st.query_params.from_dict(params_dict)
st.query_params.get_all(key)
```

```
st.query_params.clear()
st.html("<p>Hi!</p>")
```

Connect to data sources

```
st.connection("pets_db", type="sql")
conn = st.connection("sql")
conn = st.connection("snowflake")

class MyConnection(BaseConnection[myconn.MyConnection]):
    def _connect(self, **kwargs) -> MyConnection:
        return myconn.connect(**self._secrets, **kwargs)
    def query(self, query):
        return self._instance.query(query)
```

Optimize performance

Cache data objects

```
# E.g. Dataframe computation, storing downloaded data, etc.
@st.cache_data
def foo(bar):
    # Do something expensive and return data
    return data
# Executes foo
d1 = foo(ref1)
# Does not execute foo
# Returns cached item by value, d1 == d2
d2 = foo(ref1)
# Different arg, so function foo executes
d3 = foo(ref2)
# Clear the cached value for foo(ref1)
foo.clear(ref1)
# Clear all cached entries for this function
foo.clear()
# Clear values from *all* in-memory or on-disk cached functions
st.cache_data.clear()
```

Cache global resources

```
# E.g. TensorFlow session, database connection, etc.
@st.cache_resource
def foo(bar):
    # Create and return a non-data object
    return session
# Executes foo
s1 = foo(ref1)
# Does not execute foo
# Returns cached item by reference, s1 == s2
s2 = foo(ref1)
# Different arg, so function foo executes
s3 = foo(ref2)
# Clear the cached value for foo(ref1)
foo.clear(ref1)
# Clear all cached entries for this function
foo.clear()
# Clear all global resources from cache
st.cache_resource.clear()
```

Display progress and status

```
# Show a spinner during a process
with st.spinner(text="In progress"):
    time.sleep(3)
    st.success("Done")

# Show and update progress bar
bar = st.progress(50)
time.sleep(3)
bar.progress(100)

with st.status("Authenticating...") as s:
    time.sleep(2)
    st.write("Some long response.")
    s.update(label="Response")

st.balloons()
st.snow()
st.toast("Warming up...")
st.error("Error message")
st.warning("Warning message")
st.info("Info message")
st.success("Success message")
st.exception(e)
```

Personalize apps for users

```
# Authenticate users
if not st.user.is_logged_in:
    st.login("my_provider")
f"Hi, {st.user.name}"
st.logout()

# Get dictionaries of cookies, headers, locale, and browser data
st.context.cookies
st.context.headers
st.context.ip_address
st.context.is_embedded
st.context.locale
st.context.theme.type
st.context.timezone
st.context.timezone_offset
st.context.url
```

← **Previous:** Quick reference

Next: Release notes →

Still have questions?

Our forums are full of helpful information and Streamlit experts.