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
      add
    - 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

App testing

Streamlit app testing framework enables developers to build and run headless tests that execute their app code directly, simulate user input, and inspect rendered outputs for correctness.

The provided class, AppTest, simulates a running app and provides methods to set up, manipulate, and inspect the app contents via API instead of a browser UI. It can be used to write automated tests of an app in various scenarios. These can then be run using a tool like pytest. A typical pattern is to build a suite of tests for an app that ensure consistent functionality as the app evolves, and run the tests locally and/or in a CI environment like Github Actions.

The AppTest class

st.testing.v1.AppTest

st.testing.v1.AppTest simulates a running Streamlit app for testing.

```
from streamlit.testing.v1 import AppTest at = AppTest.from_file("streamlit_app.py") at.secrets["WORD"] = "Foobar" at.run() assert not at.exception at.text_input("word").input("Bazbat").run() assert at.warning[0].value == "Try again."
```

AppTest.from_file

st.testing.v1.AppTest.from_file initializes a simulated app from a file.

```
from streamlit.testing.v1 import AppTest at = AppTest.from_file("streamlit_app.py") at.secrets["WORD"] = "Foobar" at.run() assert not at.exception
```

AppTest.from_string

st.testing.v1.AppTest.from_string initializes a simulated app from a string.

```
from streamlit.testing.v1 import AppTest app_script = """ import streamlit as st word_of_the_day = st.text_input("What's the word of the day?", key="word") if word_of_the_day == st.secrets["WORD"]: st.success("That's right!") elif word_of_the_day and word_of_the_day != st.secrets["WORD"]: st.warn("Try again.") """ at = AppTest.from_string(app_script) at.secrets["WORD"] = "Foobar" at.run() assert not at.exception
```

AppTest.from_function

st.testing.v1.AppTest.from_function initializes a simulated app from a function.

```
from streamlit.testing.v1 import AppTest def app_script (): import streamlit as st word_of_the_day = st.text_input("What's the word of the day?", key="word") if word_of_the_day == st.secrets["WORD"]: st.success("That's right!") elif word_of_the_day and word_of_the_day != st.secrets["WORD"]: st.warn("Try again.") at = AppTest.from_function(app_script) at.secrets["WORD"] = "Foobar" at.run() assert not at.exception
```

Testing-element classes

Block

A representation of container elements, including:

- st.chat_message
- st.columns
- st.sidebar
- st.tabs
- The main body of the app.

```
# at.sidebar returns a Block at.sidebar.button[0].click().run() assert not at.exception
```

Element

The base class for representation of all elements, including:

- st.title
- st.header
- st.markdown
- st.dataframe

```
# at.title returns a sequence of Title # Title inherits from Element assert at.title[0].value == "My awesome app"
```

Button

A representation of `st.button` and `st.form_submit_button`.

```
at.button[0].click().run()
```

ChatInput

A representation of `st.chat_input`.

```
at.chat_input[0].set_value("What is Streamlit?").run()
```

Checkbox

A representation of `st.checkbox`.

```
at.checkbox[0].check().run()
```

ColorPicker

A representation of `st.color_picker`.

```
at.color_picker[0].pick("#FF4B4B").run()
```

DateInput

A representation of `st.date_input`.

```
release_date = datetime.date(2023, 10, 26) at.date_input[0].set_value(release_date).run()
```

Multiselect

A representation of `st.multiselect`.

```
at.multiselect[0].select("New York").run()
```

NumberInput

A representation of `st.number_input`.

```
at.number_input[0].increment().run()
```

Radio

A representation of `st.radio`.

```
at.radio[0].set_value("New York").run()
```

SelectSlider

A representation of `st.select_slider`.

```
at.select_slider[0].set_range("A","C").run()
```

Selectbox

A representation of `st.selectbox`.

```
at.selectbox[0].select("New York").run()
```

Slider

A representation of `st.slider`.

```
at.slider[0].set_range(2,5).run()
```

TextArea

A representation of `st.text_area`.

```
at.text_area[0].input("Streamlit is awesome!").run()
```

TextInput

A representation of `st.text_input`.

```
at.text_input[0].input("Streamlit").run()
```

TimeInput

A representation of `st.time_input`.

```
at.time_input[0].increment().run()
```

Toggle

A representation of `st.toggle`.

```
at.toggle[0].set_value("True").run()
```

←Previous: ConfigurationNext: st.testing.v1.AppTest→

forum

Still have questions?

Our forums are full of helpful information and Streamlit experts.
