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
      remove
      - BUTTONS

      - st.button
      - st.download_button
      - st.form_submit_button*link*
      - st.link_button
      - st.page_link
      - SELECTIONS

      - st.checkbox
      - st.color_picker
      - st.feedback
      - st.multiselect
      - st.pills
      - st.radio
      - st.segmented_control
      - st.selectbox
      - st.select_slider
      - st.toggle
      - NUMERIC

            - st.number_input
            - st.slider
            - DATE & TIME
            - st.date_input
            - st.time_input
            - TEXT
            - st.chat_input*link*
            - st.text_area
            - st.text_input
            - MEDIA & FILES
            - st.audio_input
            - st.camera_input
            - st.data_editor*link*
            - st.file_uploader
        - 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/
- Input widgets/
- st.camera_input

st.camera_input

Streamlit Version Version 1.41.0

Display a widget that returns pictures from the user's webcam.

Function signature[source]

st.camera_input(label, key=None, help=None, on_change=None, args=None, kwargs=None, *, disabled=False, label_visibility="visible")

Parameters	
label (str)	A short label explaining to the user what this widget is used for. The label can optionally contain GitHub-flavored Markdown of the following types: Bold, Italics, Strikethroughs, Inline Code, Links, and Images. Images display like icons, with a max height equal to the font height. Unsupported Markdown elements are unwrapped so only their children (text contents) render. Display unsupported elements as literal characters by backslash-escaping them. E.g., `"1\. Not an ordered list"`. See the `body` parameter of `st.markdown` for additional, supported Markdown directives. For accessibility reasons, you should never set an empty label, but you can hide it with `label_visibility` if needed. In the future, we may disallow empty labels by raising an exception.
key (str or int)	An optional string or integer to use as the unique key for the widget. If this is omitted, a key will be generated for the widget based on its content. No two widgets may have the same key.

Returns	
(None or UploadedFile)	The UploadedFile class is a subclass of BytesIO, and therefore is "file-like". This means you can pass an instance of it anywhere a file is expected.

Function signature[source]

st.camera_input(label, key=None, help=None, on_change=None, args=None, kwargs=None, *, disabled=False, label_visibility="visible")

help (str)	An optional tooltip that gets displayed next to the widget label. Streamlit only displays the tooltip when `label_visibility="visible"`.
on_change (callable)	An optional callback invoked when this camera_input's value changes.
args (tuple)	An optional tuple of args to pass to the callback.
kwargs (dict)	An optional dict of kwargs to pass to the callback.
disabled (bool)	An optional boolean that disables the camera input if set to `True`. Default is `False`.
label_visibility ("visible", "hidden", or "collapsed")	The visibility of the label. The default is `"visible"`. If this is `"hidden"`, Streamlit displays an empty spacer instead of the label, which can help keep the widget alligned with other widgets. If this is `"collapsed"`, Streamlit displays no label or spacer.
Returns	
(None or UploadedFile)	The UploadedFile class is a subclass of BytesIO, and therefore is "file-like". This means you can pass an instance of it anywhere a file is expected.

Examples

```
import streamlit as st

enable = st.checkbox("Enable camera")
picture = st.camera_input("Take a picture", disabled=not enable)

if picture:
    st.image(picture)
```

[Built with Streamlit 🎈](#)
[Fullscreen *open_in_new*](#)

To read the image file buffer as bytes, you can use `getvalue()` on the `UploadedFile` object.

```
import streamlit as st img_file_buffer = st.camera_input("Take a picture") if img_file_buffer is not
None: # To read image file buffer as bytes: bytes_data = img_file_buffer.getvalue() # Check the type
of bytes_data: # Should output: <class 'bytes'> st.write(type(bytes_data))
```
priority_high

Important

`st.camera_input` returns an object of the `UploadedFile` class, which a subclass of BytesIO. Therefore it is a "file-like" object. This means you can pass it anywhere where a file is expected, similar to `st.file_uploader`.

Image processing examples

You can use the output of `st.camera_input` for various downstream tasks, including image processing. Below, we demonstrate how to use the `st.camera_input` widget with popular image and data processing libraries such as [Pillow](#), [NumPy](#), [OpenCV](#), [TensorFlow](#), [torchvision](#), and [PyTorch](#).

While we provide examples for the most popular use-cases and libraries, you are welcome to adapt these examples to your own needs and favorite libraries.

Pillow (PIL) and NumPy

Ensure you have installed Pillow and NumPy.

To read the image file buffer as a PIL Image and convert it to a NumPy array:

```
import streamlit as st from PIL import Image import numpy as np img_file_buffer =
st.camera_input("Take a picture") if img_file_buffer is not None: # To read image file buffer as a PIL
Image: img = Image.open(img_file_buffer) # To convert PIL Image to numpy array: img_array =
np.array(img) # Check the type of img_array: # Should output: <class 'numpy.ndarray'>
st.write(type(img_array)) # Check the shape of img_array: # Should output shape: (height, width,
channels) st.write(img_array.shape)
```

OpenCV (cv2)

Ensure you have installed OpenCV and NumPy.

To read the image file buffer with OpenCV:

```
import streamlit as st import cv2 import numpy as np img_file_buffer = st.camera_input("Take a
picture") if img_file_buffer is not None: # To read image file buffer with OpenCV: bytes_data =
img_file_buffer.getvalue() cv2_img = cv2.imdecode(np.frombuffer(bytes_data, np.uint8),
cv2.IMREAD_COLOR) # Check the type of cv2_img: # Should output: <class 'numpy.ndarray'>
st.write(type(cv2_img)) # Check the shape of cv2_img: # Should output shape: (height, width, channels)
st.write(cv2_img.shape)
```

TensorFlow

Ensure you have installed TensorFlow.

To read the image file buffer as a 3 dimensional uint8 tensor with TensorFlow:

```
import streamlit as st import tensorflow as tf img_file_buffer = st.camera_input("Take a picture") if
img_file_buffer is not None: # To read image file buffer as a 3D uint8 tensor with TensorFlow:
bytes_data = img_file_buffer.getvalue() img_tensor = tf.io.decode_image(bytes_data, channels=3) #
Check the type of img_tensor: # Should output: <class 'tensorflow.python.framework.ops.EagerTensor'>
st.write(type(img_tensor)) # Check the shape of img_tensor: # Should output shape: (height, width,
channels) st.write(img_tensor.shape)
```

Torchvision

Ensure you have installed Torchvision (it is not bundled with PyTorch) and PyTorch.

To read the image file buffer as a 3 dimensional uint8 tensor with `torchvision.io`:

```
import streamlit as st import torch import torchvision img_file_buffer = st.camera_input("Take a
picture") if img_file_buffer is not None: # To read image file buffer as a 3D uint8 tensor with
`torchvision.io`: bytes_data = img_file_buffer.getvalue() torch_img = torchvision.io.decode_image(
torch.frombuffer(bytes_data, dtype=torch.uint8) ) # Check the type of torch_img: # Should output:
```

```
<class 'torch.Tensor'> st.write(type(torch_img)) # Check the shape of torch_img: # Should output shape: torch.Size([channels, height, width]) st.write(torch_img.shape)
```

PyTorch

Ensure you have installed PyTorch and NumPy.

To read the image file buffer as a 3 dimensional uint8 tensor with PyTorch:

```
import streamlit as st import torch import numpy as np img_file_buffer = st.camera_input("Take a picture") if img_file_buffer is not None: # To read image file buffer as a 3D uint8 tensor with PyTorch: bytes_data = img_file_buffer.getvalue() torch_img = torch.ops.image.decode_image( torch.from_numpy(np.frombuffer(bytes_data, np.uint8)), 3 ) # Check the type of torch_img: # Should output: <class 'torch.Tensor'> st.write(type(torch_img)) # Check the shape of torch_img: # Should output shape: torch.Size([channels, height, width]) st.write(torch_img.shape)
```

←Previous: st.audio_inputNext: st.data_editor→

forum

Still have questions?

Our forums are full of helpful information and Streamlit experts.

Cookie policy

forum Ask AI