← Textbox Video →

UploadButton

```
gradio.UploadButton(···)
```

Description

Used to create an upload button, when clicked allows a user to upload files that satisfy the specified file type or generic files (if file_type not set).

Behavior

As input: passes the uploaded file as a `file-object` or `List[file-object]` depending on `file_count` (or a `bytes`/`List[bytes]` depending on `type`)

As output: expects function to return a `str` path to a file, or `List[str]` consisting of paths to files.

Initialization

Parameter	Description
`label` *str* **default: "Upload a File"**	Text to display on the button. Defaults to "Upload a File".
`value` *str | list[str] | Callable | None* **default: None**	File or list of files to upload by default.
`every` *float | None* **default: None**	If `value` is a callable, run the function 'every' number of seconds while the client connection is open. Has no effect otherwise. Queue must be enabled. The event can be accessed (e.g. to cancel it) via this component's .load_event attribute.

ımeter	Description
`variant` *Literal[('primary', 'secondary', 'stop')]* **default: "secondary"**	'primary' for main call-to-action, 'secondary' for a more subdued style, 'stop' for a stop button.
`visible` *bool* **default: True**	If False, component will be hidden.
`size` *Literal[('sm', 'lg')] | None* **default: None**	Size of the button. Can be "sm" or "lg".
`icon` *str | None* **default: None**	
`scale` *int | None* **default: None**	relative width compared to adjacent Components in a Row. For example, if Component A has scale=2, and Component B has scale=1, A will be twice as wide as B. Should be an integer.
`min_width` *int | None* **default: None**	minimum pixel width, will wrap if not sufficient screen space to satisfy this value. If a certain scale value results in this Component being narrower than min_width, the min_width parameter will be respected first.
`interactive` *bool* **default: True**	If False, the UploadButton will be in a disabled state.
`elem_id` *str | None* **default: None**	An optional string that is assigned as the id of this component in the HTML DOM. Can be used for targeting CSS styles.

ımeter	Description
`elem_classes` *list[str] | str | None* **default: None**	An optional list of strings that are assigned as the classes of this component in the HTML DOM. Can be used for targeting CSS styles.
`render` *bool* **default: True**	If False, component will not render be rendered in the Blocks context. Should be used if the intention is to assign event listeners now but render the component later.
`type` *Literal[('filepath', 'bytes')]* **default: "filepath"**	Type of value to be returned by component. "file" returns a temporary file object with the same base name as the uploaded file, whose full path can be retrieved by file_obj.name, "binary" returns an bytes object.
`file_count` *Literal[('single', 'multiple', 'directory')]* **default: "single"**	if single, allows user to upload one file. If "multiple", user uploads multiple files. If "directory", user uploads all files in selected directory. Return type will be list for each file in case of "multiple" or "directory".
`file_types` *list[str] | None* **default: None**	List of type of files to be uploaded. "file" allows any file to be uploaded, "image" allows only image files to be uploaded, "audio" allows only audio files to be uploaded, "video" allows only video files to be uploaded, "text" allows only text files to be uploaded.

Shortcuts

Class	Interface String Shortcut	Initialization
`gradio.UploadButton`	"uploadbutton"	Uses default values

Demos

upload_button

```
import gradio as gr
```

```
ef upload_file(files):
    file_paths = [file.name for file in files]
    return file_paths

with gr.Blocks() as demo:
    file_output = gr.File()
    upload_button = gr.UploadButton("Click to Upload a File", file_types=["image", "video"]
, file_count="multiple")
    upload_button.upload(upload_file, upload_button, file_output)
```

Event Listeners

Description

Event listeners allow you to capture and respond to user interactions with the UI components you've defined in a Gradio Blocks app. When a user interacts with an element, such as changing a slider value or uploading an image, a function is called.

Supported Event Listeners

The `UploadButton` component supports the following event listeners. Each event listener takes the same parameters, which are listed in the Event Arguments table below.

Listener	Description
`gradio.UploadButton.click(fn, ···)`	Triggered when the UploadButton is clicked.
`gradio.UploadButton.upload(fn, ···)`	This listener is triggered when the user uploads a file into the UploadButton.

Event Arguments

Parameter	Description
`fn` *Callable | None | Literal['decorator']* **default: "decorator"**	the function to call when this event is triggered. Often a machine learning model's prediction function. Each parameter of the function corresponds to one input component, and the function should return a single value or a tuple of values, with each element in the tuple corresponding to one output component.

Parameter	Description
`inputs` *Component | list[Component] | set[Component] | None* **default: None**	List of gradio.components to use as inputs. If the function takes no inputs, this should be an empty list.
`outputs` *Component | list[Component] | None* **default: None**	List of gradio.components to use as outputs. If the function returns no outputs, this should be an empty list.
`api_name` *str | None | Literal[False]* **default: None**	defines how the endpoint appears in the API docs. Can be a string, None, or False. If set to a string, the endpoint will be exposed in the API docs with the given name. If None (default), the name of the function will be used as the API endpoint. If False, the endpoint will not be exposed in the API docs and downstream apps (including those that `gr.load` this app) will not be able to use this event.
`scroll_to_output` *bool* **default: False**	If True, will scroll to output component on completion
`show_progress` *Literal[('full', 'minimal', 'hidden')]* **default: "full"**	If True, will show progress animation while pending
`queue` *bool | None* **default: None**	If True, will place the request on the queue, if the queue has been enabled. If False, will not put this event on the queue, even if the queue has been enabled. If None, will use the queue setting of the gradio app.

Parameter	Description
`batch` *bool* **default: False**	If True, then the function should process a batch of inputs, meaning that it should accept a list of input values for each parameter. The lists should be of equal length (and be up to length `max_batch_size`). The function is then ***required*** to return a tuple of lists (even if there is only 1 output component), with each list in the tuple corresponding to one output component.
`max_batch_size` *int* **default: 4**	Maximum number of inputs to batch together if this is called from the queue (only relevant if batch=True)
`preprocess` *bool* **default: True**	If False, will not run preprocessing of component data before running 'fn' (e.g. leaving it as a base64 string if this method is called with the `Image` component).
`postprocess` *bool* **default: True**	If False, will not run postprocessing of component data before returning 'fn' output to the browser.
`cancels` *dict[str, Any] | list[dict[str, Any]] | None* **default: None**	A list of other events to cancel when this listener is triggered. For example, setting cancels=[click_event] will cancel the click_event, where click_event is the return value of another components .click method. Functions that have not yet run (or generators that are iterating) will be cancelled, but functions that are currently running will be allowed to finish.
`every` *float | None* **default: None**	Run this event 'every' number of seconds while the client connection is open. Interpreted in seconds. Queue must be enabled.

Parameter	Description
`trigger_mode` *Literal['once', 'multiple', 'always_last'] | None* **default: None**	If "once" (default for all events except `.change()`) would not allow any submissions while an event is pending. If set to "multiple", unlimited submissions are allowed while pending, and "always_last" (default for `.change()` event) would allow a second submission after the pending event is complete.
`js` *str | None* **default: None**	Optional frontend js method to run before running 'fn'. Input arguments for js method are values of 'inputs' and 'outputs', return should be a list of values for output components.
`concurrency_limit` *int | None | Literal['default']* **default: "default"**	If set, this is the maximum number of this event that can be running simultaneously. Can be set to None to mean no concurrency_limit (any number of this event can be running simultaneously). Set to "default" to use the default concurrency limit (defined by the `default_concurrency_limit` parameter in `Blocks.queue()`, which itself is 1 by default).
`concurrency_id` *str | None* **default: None**	If set, this is the id of the concurrency group. Events with the same concurrency_id will be limited by the lowest set concurrency_limit.
`show_api` *bool* **default: True**	whether to show this event in the "view API" page of the Gradio app, or in the ".view_api()" method of the Gradio clients. Unlike setting api_name to False, setting show_api to False will still allow downstream apps to use this event. If fn is None, show_api will automatically be set to False.

← Textbox

Video →