LoginButton

`gradio.LoginButton(···)`

Description

Button that redirects the user to Sign with Hugging Face using OAuth.

Behavior

As input: undefined

As output: undefined

Initialization

Parameter	Description
`value` *str* **default: "Sign in with Hugging Face"**	
`logout_value` *str* **default: "Logout ({})"**	The text to display when the user is signed in. The string should contain a placeholder for the username with a call-to-action to logout, e.g. "Logout ( )".
`every` *float | None* **default: None**	

meter	Description
`variant` *Literal[('primary', 'secondary', 'stop')]* **default: "secondary"**	
`size` *Literal[('sm', 'lg')] | None* **default: None**	
`icon` *str | None* **default: "https://huggingface.co/front/assets/huggingface_logo-noborder.svg"**	
`link` *str | None* **default: None**	
`visible` *bool* **default: True**	
`interactive` *bool* **default: True**	
`elem_id` *str | None* **default: None**	
`elem_classes` *list[str] | str | None* **default: None**	

meter	Description
`render` *bool* **default: True**	
`scale` *int | None* **default: 0**	
`min_width` *int | None* **default: None**	
`signed_in_value` *str* **default: "Signed in as {}"**	

Shortcuts

Class	Interface String Shortcut	Initialization
`gradio.LoginButton`	"loginbutton"	Uses default values

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 `LoginButton` 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.LoginButton.click(fn, ...)`	Triggered when the Button is clicked.

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.
`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

Parameter	Description
`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.
`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.

Parameter	Description
`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.
`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.

LinePlot

LogoutButton →

Parameter	Description
`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.