

Gradio를 활용한 나만의 웹앱 구축



Contents

01 Gradio 소개

02 Gradio: Interface

03 Gradio: Blocks And Event

04 Gradio: etc.

05 Gradio Docs



01. Gradio 소개

01 Gradio가 뭔가요?



- ML모델, 함수 또는 기타 Python 코드를 위한 사용자 인터페이스를 손쉽게 구축할 수 있게 해주는 오픈 소스 Python 라이브러리
- 간편한 사용성, 다양한 입출력 컴포넌트 지원, 쉬운 공유 등 다양한 장점
- 활용 사례
 - 이미지 분류기
 - 자연어처리 어플리케이션
 - 오디오 분석 도구 등

01 Gradio 소개

02 Gradio 활용 예시 - LLM 어플리케이션

The screenshot shows a Gradio-based LLM application interface. At the top, there's a logo for 'bLOSSOM' featuring a stylized flower icon and the word 'bLOSSOM' in blue. Below the logo, a navigation bar includes a 'Chatbot' button and a trash bin icon. A search bar on the right contains the Korean text '지드래곤에 대해 설명해줘' (Describe about Jimin).

The main content area consists of two message boxes. The first message box, from '지드래곤' (Jimin), contains the following text:

지드래곤(본명: 권지용, 1988년 8월 18일 출생)은 대한민국의 보이 그룹 빅뱅의 리더로, 음악인, 작곡가, 남성 솔로 가수입니다. 그는 프로듀싱 능력이 뛰어나 빅뱅의 모든 히트곡을 포함한 많은 곡을 제작했습니다. 2024년, 7년 4개월 만에 솔로 활동을 재개하고, 신곡 '파워'를 전 세계에 공개했습니다. '파워'는 지드래곤의 첫 번째 컴백 곡으로, 2017년 미니앨범 EP '권지용' 이후 처음으로 나온 작품입니다. 지드래곤은 빅뱅 내에서 리더로 활동하며 그룹의 중심을 이루고 있으며, 그의 행보는 항상 가요계에 큰 화제를 일으키고 있습니다.

The second message box, from 'MAMA' (Mnet Asian Music Awards), contains the following text:

MAMA(Mnet Asian Music Awards) 2024에서 여러 아티스트들이 무대를 선보였으며, 지드래곤은 7년 4개월 만에 솔로 활동을 재개하고, 신곡 '파워'를 전 세계에 공개한 바 있습니다. MAMA 2024는 'BIG BLUR : What is Real?'라는 테마를 가지고 진행되었으며, 박보검이 영어 진행으로 무대에 나섰습니다. 또한, 세븐틴은 로봇과 함께하는 '마에스트로' 무대를 선보이며 팬들을 열광시켰습니다. 이 외에도 다양한 K-팝 그룹들이 무대에 올라 전 세계에 신곡을 선보였습니다.

At the bottom left, there's a text input field with the placeholder 'Type here and press enter, 질문을 입력하고 enter 버튼을 눌러주세요.' (Type here and press enter, please enter your question). At the bottom right, there's a 'Clear history' button.

02 Gradio 활용 예시 - 시각-언어 모델 어플리케이션

X-LLaVA: Large Language and Vision Assistant

[Project Page] [Code] [Model]]

X-LLaVA

Image

이미지를 끌어 놓으세요
- 또는 -
클릭해서 업로드하기

X-LLaVA Chatbot



이 이미지를 보고 느껴지는 분위기에 대해 설명해줘

Examples



Describe what is interesting about this image.



Please explain the atmosphere you feel when you see the image.

Parameters

Enter text and press ENTER

Send

Upvote

Downvote

Flag

Regenerate

Clear

X-LLaVA

License

The service is a research preview intended for non-commercial use only, subject to the model [License](#) of LLaMA, [Terms of Use](#) of the data generated by OpenAI.

02. Gradio: Interface

00 환경설정

```
1 !pip install -U gradio
→ Collecting gradio
  Downloading gradio-5.6.0-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<24.0,>=22.0 (from gradio)
  Downloading aiofiles-23.2.1-py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.7.1)
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.5-py3-none-any.whl.metadata (27 kB)
Collecting ffdpy (from gradio)
  Downloading ffdpy-0.4.0-py3-none-any.whl.metadata (2.9 kB)
Collecting gradio-client==1.4.3 (from gradio)
  Downloading gradio_client-1.4.3-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.27.2)
Requirement already satisfied: huggingface-hub>=0.25.1 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.26.2)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.1.4)
Collecting markupsafe~2.0 (from gradio)
  Downloading MarkupSafe-2.1.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.0 kB)
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (1.26.4)
Requirement already satisfied: orjson>=3.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.10.11)
```

- 셀에 !pip install -U gradio 를 통해 gradio 최신 버전 설치

01 Interface

```

1 import gradio as gr
2
3 def greet(name, intensity):
4     return "Hello, " + name + "!" * int(intensity)
5
6 demo = gr.Interface(
7     fn=greet,
8     inputs=["text", "slider"],
9     outputs=["text"],
10)
11
12 demo.launch()

```

Running Gradio in a Colab notebook requires sharing enabled. Automatically setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly).

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()

* Running on public URL: <https://3a697d83bc7781d52a.gradio.live>

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (<https://huggingface.co/spaces>)

The screenshot shows a Gradio interface with the following components:

- name:** Input field containing "임현석".
- intensity:** Slider control set to 8.
- output:** Text area displaying "Hello, 임현석!!!!!!".
- Flag:** A large gray button labeled "Flag".
- Buttons:** Two buttons at the bottom left: "Clear" (gray) and "Submit" (orange).

- Interface는 Gradio의 가장 핵심적인 클래스.
- 해당 클래스를 통해 사용자로부터 입력을 받아 함수를 실행하고, 결과를 출력하는 웹 인터페이스를 생성.

01 Interface

```
1 import gradio as gr
2
3 def greet(name, intensity):
4     return "Hello, " + name + "!" * int(intensity)
5
6 demo = gr.Interface(
7     fn=greet,
8     inputs=["text", "slider"],
9     outputs=["text"],
10 )
11
12 demo.launch()
```

- fn: 사용자 인터페이스(UI)를 감싸는 기능
- inputs: 입력에 사용될 Gradio 구성 요소로 구성 요소의 수는 함수의 인자 수와 일치해야 함.
- outputs: 출력에 사용될 Gradio 구성요소로 구성 요소의 수는 함수의 반환 값 수와 일치해야 함.

02 Components

```
1 import gradio as gr
2
3 def greet(name, intensity):
4     return "Hello, " + name + "!" * int(intensity)
5
6 demo = gr.Interface(
7     fn=greet,
8     inputs=["text", "slider"],
9     outputs=["text"],
10 )
11
12 demo.launch()
```

- 데모에서 입력 또는 출력으로 사용할 수 있는 30개 이상의 사전 구축된 Components가 포함되어 있음.
- 위의 코드에서는 기본 버전의 gr.Textbox, gr.Slider을 활용

02 Components

```

1 import gradio as gr
2
3 def greet(name, intensity):
4     return "Hello, " + name + "!" * intensity
5
6 demo = gr.Interface(
7     fn=greet,
8     inputs=["text", gr.Slider(value=2, minimum=1, maximum=10, step=1)],
9     outputs=[gr.Textbox(label="greeting", lines=3)],
10)
11
12 demo.launch()

```

Running Gradio in a Colab notebook requires sharing enabled. Automatically setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly).

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()

* Running on public URL: <https://aff1ebf2c081b884cd.gradio.live>

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (<https://huggingface.co/spaces>)

The screenshot shows a Gradio interface with the following components:

- A text input field labeled "name".
- An "intensity" slider component. It consists of a horizontal slider bar with a value of 2, a text input field showing "2", and a dropdown menu with options "2" and "5". Below the slider is a numerical range from 1 to 10.
- A text output field labeled "greeting".
- Two buttons at the bottom: "Clear" (gray) and "Submit" (orange).
- A "Flag" button located below the "greeting" field.

- gr.Slider와 gr.Textbox의 내부 인자들을 수정하여 Components를 커스터마이징

02 Components

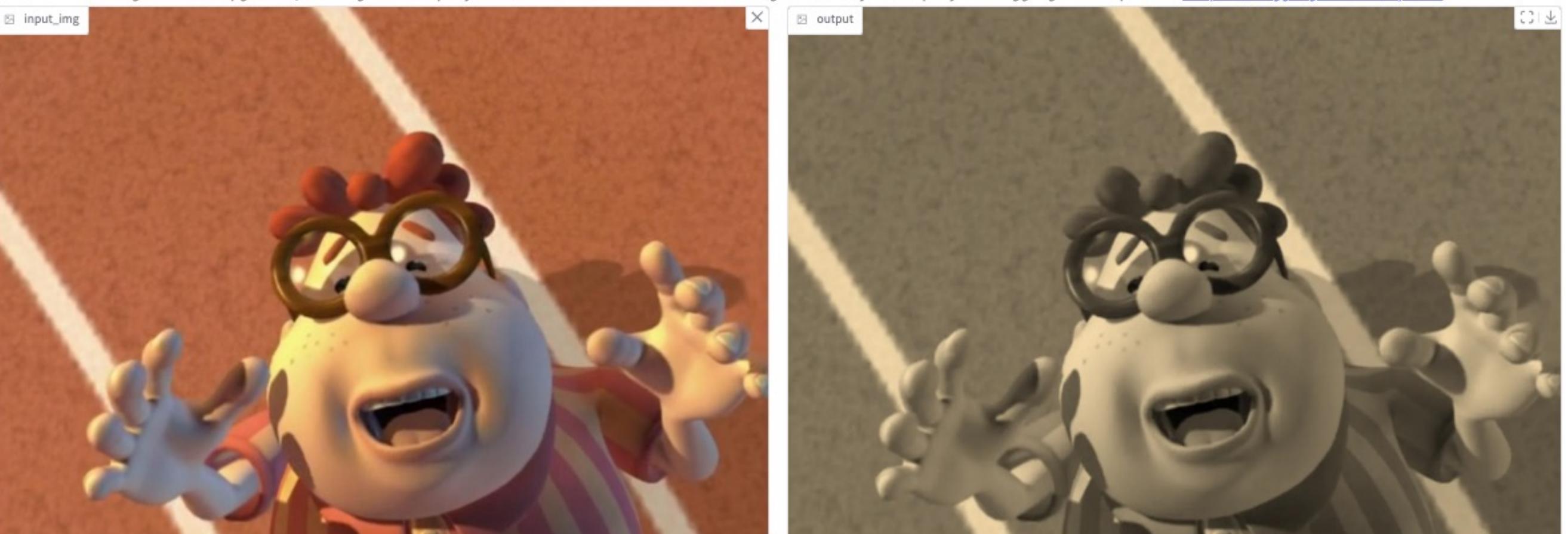
```
1 import numpy as np
2 import gradio as gr
3
4 def sepia(input_img):
5     sepia_filter = np.array([
6         [0.393, 0.769, 0.189],
7         [0.349, 0.686, 0.168],
8         [0.272, 0.534, 0.131]
9     ])
10    sepia_img = input_img.dot(sepia_filter.T)
11    sepia_img /= sepia_img.max()
12    return sepia_img
13
14 demo = gr.Interface(sepia, gr.Image(), "image")
15 demo.launch()
16
```

→ Running Gradio in a Colab notebook requires sharing enabled. Automatically setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly).

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()

* Running on public URL: <https://d447dc413efe7662af.gradio.live>

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (<https://huggingface.co/spaces>)



02 Components

입력 컴포넌트

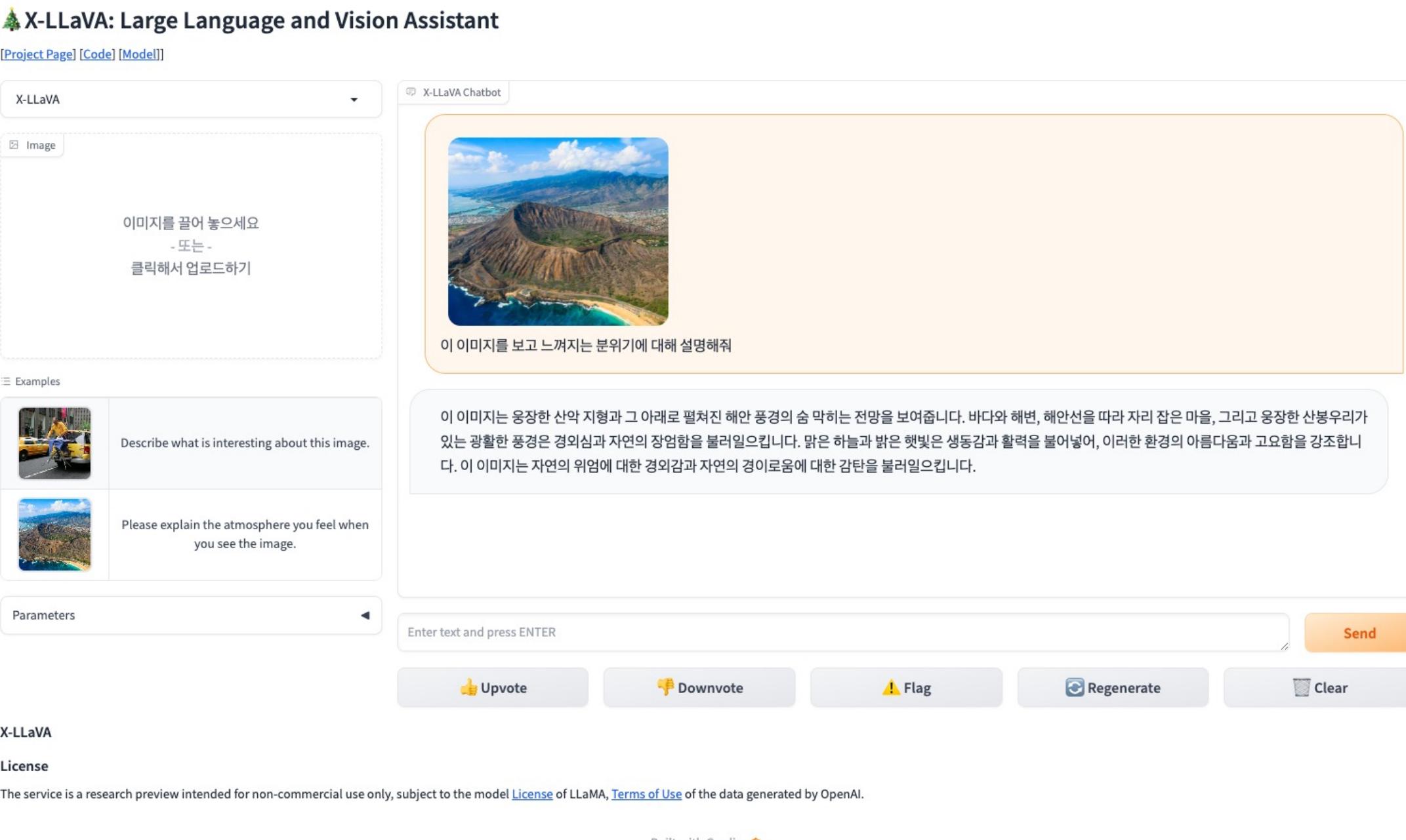
- TextBox: 텍스트 입력
- Number: 숫자 입력
- Image: 이미지 업로드 or 웹캠을 통한 입력
- Audio: 오디오 파일 업로드 or 녹음 후 입력
- Video: 비디오 파일 업로드 or 웹캠을 통한 입력
- Checkbox: boolean 값 입력
- Dropdown: 미리 정의된 옵션 중 하나 선택
- Slider: 범위 내의 숫자 값 선택
- File: 파일 업로드

출력 컴포넌트

- TextBox: 텍스트를 출력
- Image: 이미지를 출력
- Audio: 오디오를 재생
- Video: 비디오를 재생
- Label: 분류 결과나 레이블을 출력
- Plot: 그래프나 차트를 출력
- Dataframe: 표 형태의 데이터를 출력
- HTML: HTML 코드를 렌더링

03. Gradio: Blocks and Event

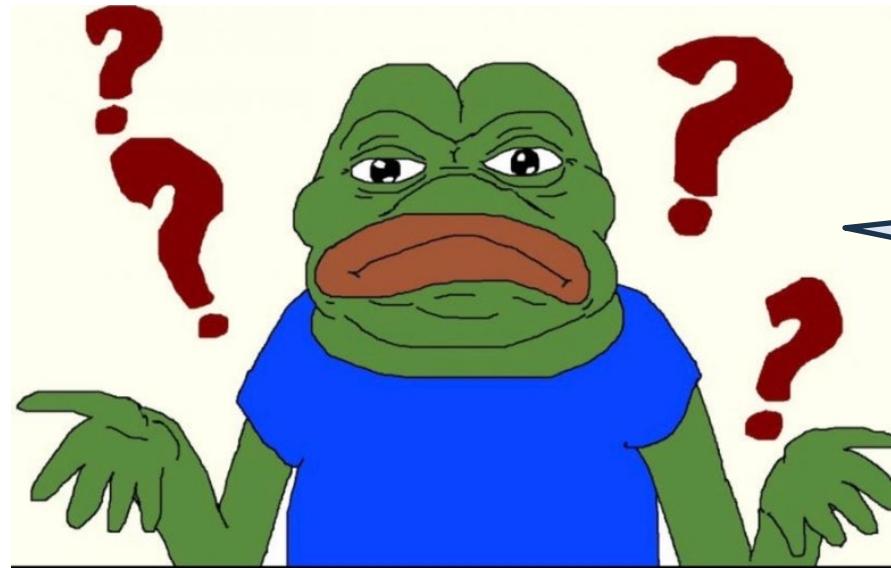
01 Blocks Structure



Blocks란?

- 복잡한 웹 어플리케이션이나 대화형 인터페이스를 구축하기 위한 강력한 레이아웃 및 워크플로우 구성 도구
- 기본적으로 Blocks는 Gradio의 컴포넌트들을 유연하게 배치하고 조합하여 복잡한 인터페이스를 만들도록 해줌.

01 Blocks Structure



엥 그러면 Interface랑 다른게 뭐임ㅋ
배우기 힘든데 그거 왜씀ㅋ

(Interface만 사용해본 사람)



주먹 쥐게 만드네.
Interface는 간단한 입력과 출력, 함수로 빠르게 웹 인터페이스 생성! 하지만 Static함.
내 맘대로 유연한 레이아웃이랑 여러 단계의 상호작용에는 Blocks가 필요.

(고수)

01 Blocks Structure

```
1 import gradio as gr
2
3
4 def greet(name):
5     return "Hello " + name + "!"
6
7
8 with gr.Blocks() as demo:
9     name = gr.Textbox(label="Name")
10    output = gr.Textbox(label="Output Box")
11    greet_btn = gr.Button("Greet")
12    greet_btn.click(fn=greet, inputs=name, outputs=output, api_name="greet")
13
14 demo.launch()
```

- demo로 정의한 Blocks를 불러옴.
- 인터페이스의 레이아웃과 동작을 세부적으로 구성할 수 있게 해주는 관리자.

01 Blocks Structure

```
1 import gradio as gr
2
3
4 def greet(name):
5     return "Hello " + name + "!"
6
7
8 with gr.Blocks() as demo:
9     name = gr.Textbox(label="Name")
10    output = gr.Textbox(label="Output Box")
11    greet_btn = gr.Button("Greet")
12    greet_btn.click(fn=greet, inputs=name, outputs=output, api_name="greet")
13
14 demo.launch()
```

- Components는 Interface에서 사용하던 것들과 동일
- 하지만 Interface에서는 내부 파라미터를 통해 전달하지만, with 구문 내에서 생성되는 즉시 Blocks에 자동 추가

02 Event Listener

```
1 import gradio as gr
2
3
4 def greet(name):
5     return "Hello " + name + "!"
6
7
8 with gr.Blocks() as demo:
9     name = gr.Textbox(label="Name")
10    output = gr.Textbox(label="Output Box")
11    greet_btn = gr.Button("Greet")
12    greet_btn.click(fn=greet, inputs=name, outputs=output, api_name="greet")
13
14 demo.launch()
```

- 이벤트 리스너는 컴포넌트 간의 데이터 흐름을 정의함
- 위의 예시에서는 greet_btn 버튼이 클릭 될 때, greet 함수를 실행하도록 설정.

02 Event Listener

```

1 import gradio as gr
2
3 def welcome(name):
4     return f"안녕하세요, {name}님!"
5
6 with gr.Blocks() as demo:
7     inp = gr.Textbox(label="이름 입력")
8     out = gr.Textbox(label="환영 메시지")
9
10    # Textbox의 값이 변경될 때마다 welcome 함수를 호출
11    inp.change(fn=welcome, inputs=inp, outputs=out)
12
13 demo.launch()
14

```

Running Gradio in a Colab notebook requires sharing enabled. Automatically setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly).

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
 * Running on public URL: <https://cd9cdf51251c098fd0.gradio.live>

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (<https://huggingface.co/spaces>)

- 기존의 이벤트 리스너와 달리 change() 리스너는 타이핑이 이루어질 때마다 함수가 실행됨.

04. Gradio: etc.

01 Controlling Layout

```
1 with gr.Blocks() as demo:  
2     with gr.Row():  
3         btn1 = gr.Button("Button 1")  
4         btn2 = gr.Button("Button 2")  
5  
6 demo.launch()
```

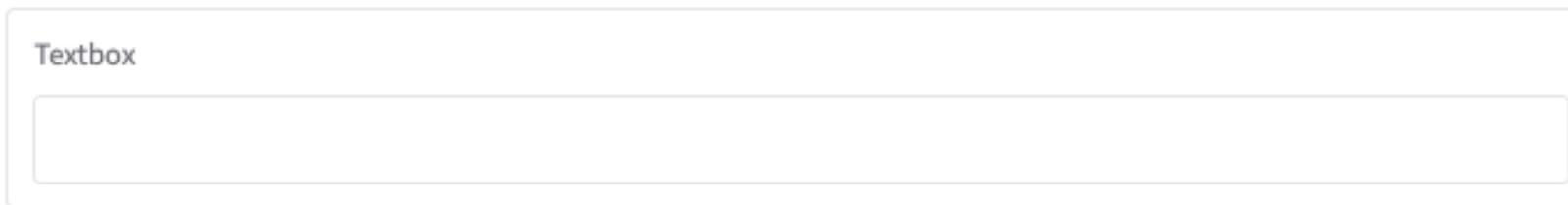
Button 1

Button 2

- `with gr.Row()`는 수평으로, `with gr.Column()`은 수직으로 배치됨.

01 Controlling Layout

```
1 import gradio as gr
2
3 with gr.Blocks() as demo:
4     with gr.Row(equal_height=True):
5         textbox = gr.Textbox()
6         btn2 = gr.Button("Button 2")
7
8 demo.launch()
```



equal_height를 하지 않았을 때,



equal_height를 설정하였을 때,

- equal_height는 Row의 모든 요소를 같은 높이로 설정
- Column에 equal_width는 존재하지 않음

01 Controlling Layout

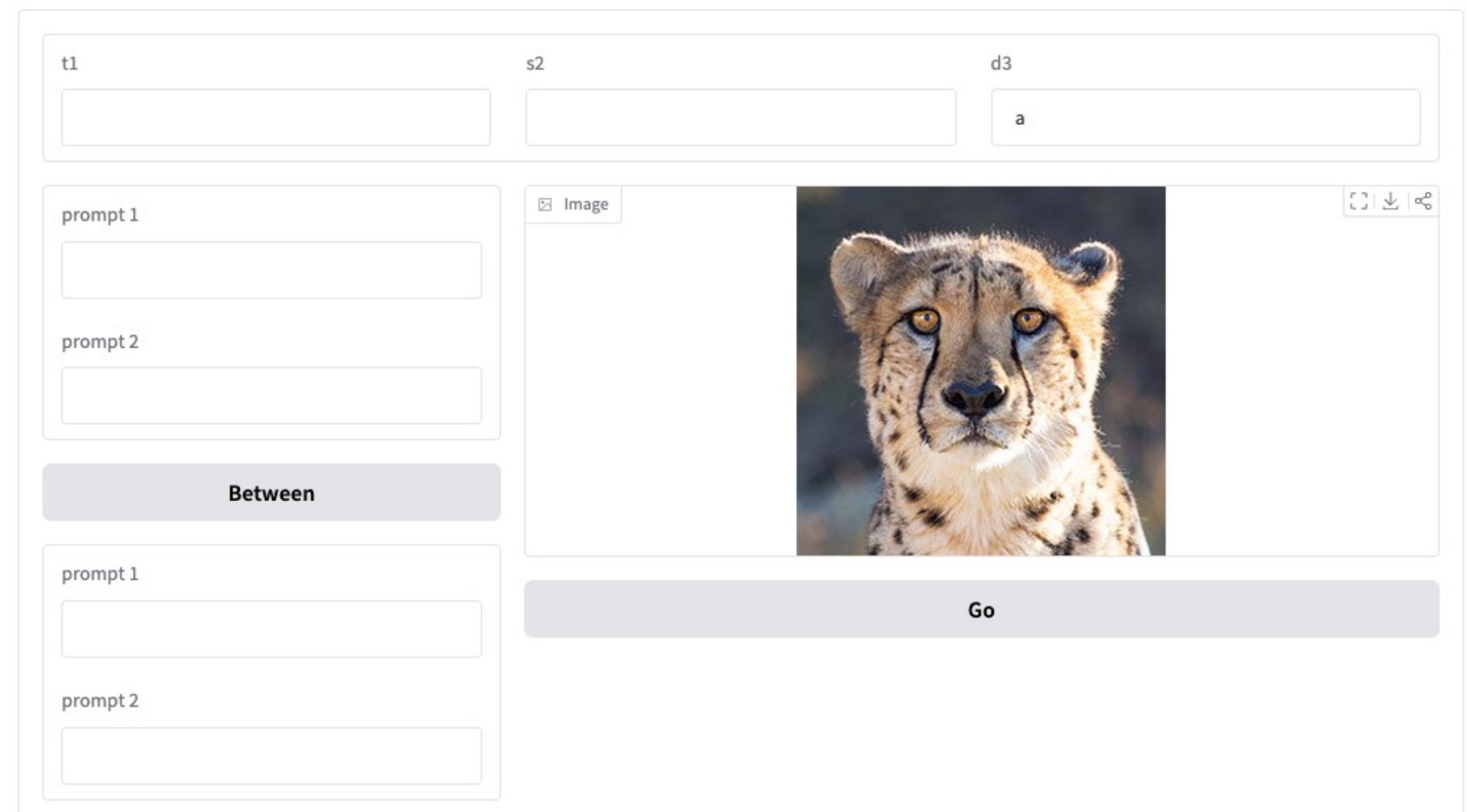
```

import gradio as gr

with gr.Blocks() as demo:
    with gr.Row():
        text1 = gr.Textbox(label="t1")
        slider2 = gr.Textbox(label="s2")
        drop3 = gr.Dropdown(["a", "b", "c"], label="d3")
    with gr.Row():
        with gr.Column(scale=1, min_width=300):
            text1 = gr.Textbox(label="prompt 1")
            text2 = gr.Textbox(label="prompt 2")
            inbtw = gr.Button("Between")
            text4 = gr.Textbox(label="prompt 1")
            text5 = gr.Textbox(label="prompt 2")
        with gr.Column(scale=2, min_width=300):
            img1 = gr.Image("images/cheetah.jpg")
            btn = gr.Button("Go")

demo.launch()

```



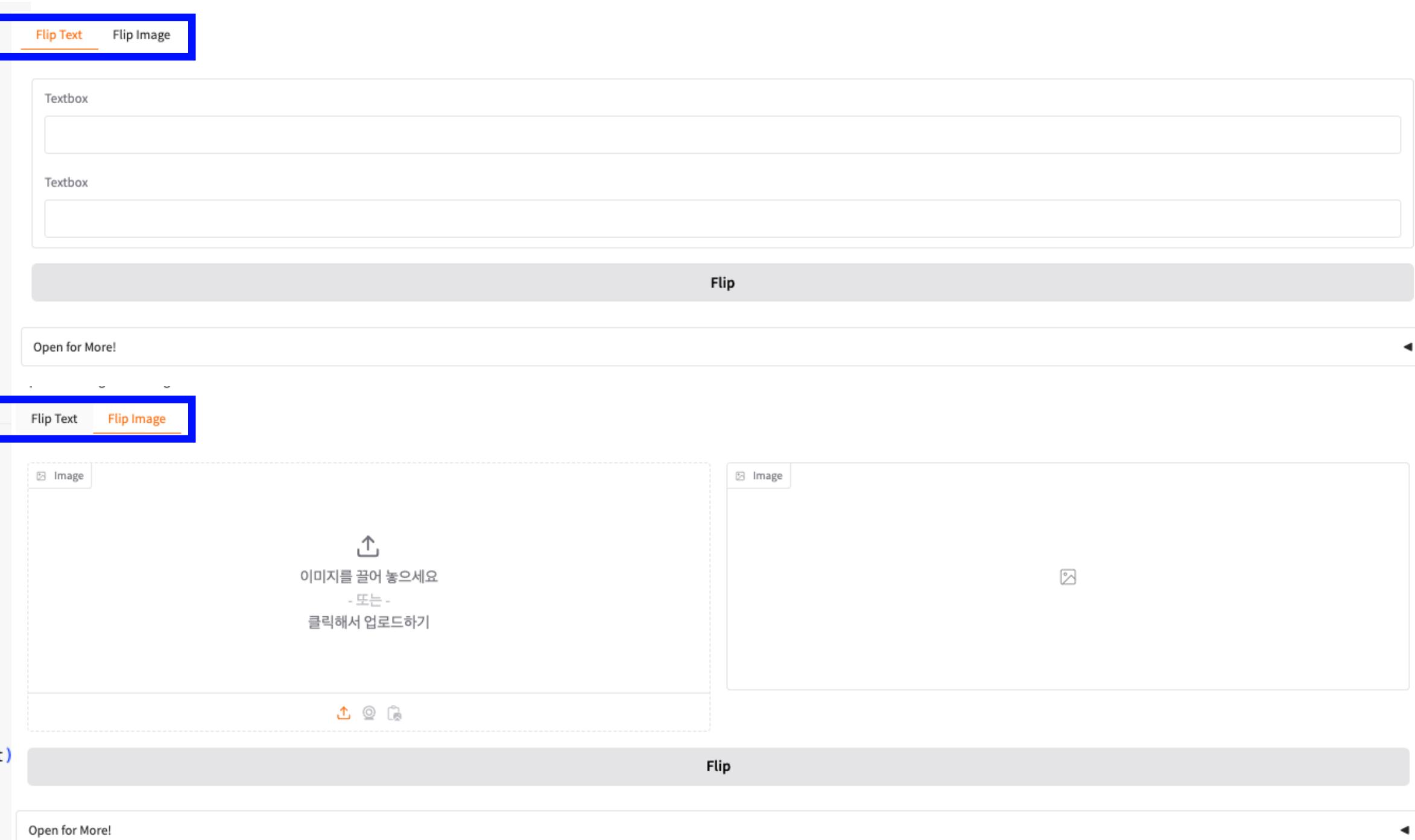
- min_width는 최소 너비를 뜻하며, scale은 배수를 뜻함.
- 한 개의 Row, Column마다 하나의 박스로 구성되어 있는 것을 볼 수 있음.

01 Controlling Layout

```

1 import numpy as np
2 import gradio as gr
3
4 def flip_text(x):
5     return x[::-1]
6
7 def flip_image(x):
8     return np.fliplr(x)
9
10 with gr.Blocks() as demo:
11     gr.Markdown("Flip text or image files using this demo.")
12     with gr.Tab("Flip Text"):
13         text_input = gr.Textbox()
14         text_output = gr.Textbox()
15         text_button = gr.Button("Flip")
16     with gr.Tab("Flip Image"):
17         with gr.Row():
18             image_input = gr.Image()
19             image_output = gr.Image()
20             image_button = gr.Button("Flip")
21
22     with gr.Accordion("Open for More!", open=False):
23         gr.Markdown("Look at me...")
24         temp_slider = gr.Slider(
25             0, 1,
26             value=0.1,
27             step=0.1,
28             interactive=True,
29             label="Slide me",
30         )
31
32     text_button.click(flip_text, inputs=text_input, outputs=text_output)
33     image_button.click(flip_image, inputs=image_input, outputs=image_output)
34
35 demo.launch()
36

```



- Tab은 크롬의 탭 창과 같이 하나를 새로 생성할 수 있음.
- 각 Tab의 컨텍스트 내에 있는 Components만 표시.

02 State

```

1 import gradio as gr
2
3 with gr.Blocks() as demo:
4     cart = gr.State([])
5     items_to_add = gr.CheckboxGroup(["Cereal", "Milk", "Orange Juice", "Water"])
6
7     def add_items(new_items, previous_cart):
8         print(new_items)
9         cart = previous_cart + new_items
10        return cart
11
12    gr.Button("Add Items").click(add_items, [items_to_add, cart], cart)
13
14    cart_size = gr.Number(label="Cart Size")
15    cart.change(lambda cart: len(cart), cart, cart_size)
16
17 demo.launch()

```

Running Gradio in a Colab notebook requires sharing enabled. Automatically setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly).

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
 * Running on public URL: <https://faee66b76b4371a4e5.gradio.live>

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (<https://huggingface.co/spaces>)

The screenshot shows a Gradio application interface. At the top, there is a 'Checkbox Group' containing four items: Cereal, Milk, Orange Juice, and Water, all of which are checked. Below this is a large grey button labeled 'Add Items'. Underneath the button is a 'Cart Size' input field containing the number '6'.

- gr.State를 활용하여 사용자별로 상태를 유지해야 하는 경우 사용함

03 Visible

```

1 import gradio as gr
2
3 with gr.Blocks() as demo:
4     name_box = gr.Textbox(label="Name")
5     age_box = gr.Number(label="Age", minimum=0, maximum=100)
6     symptoms_box = gr.CheckboxGroup(["Cough", "Fever", "Runny Nose"])
7     submit_btn = gr.Button("Submit")
8
9     with gr.Column(visible=False) as output_col:
10         diagnosis_box = gr.Textbox(label="Diagnosis")
11         patient_summary_box = gr.Textbox(label="Patient Summary")
12
13     def submit(name, age, symptoms):
14         return {
15             "submit_btn": gr.Button(visible=False),
16             "output_col": gr.Column(visible=True),
17             "diagnosis_box": "covid" if "Cough" in symptoms else "flu",
18             "patient_summary_box": f"{name}, {age} y/o",
19         }
20
21     submit_btn.click(
22         submit,
23         [name_box, age_box, symptoms_box],
24         [submit_btn, diagnosis_box, patient_summary_box, output_col],
25     )
26
27 demo.launch()

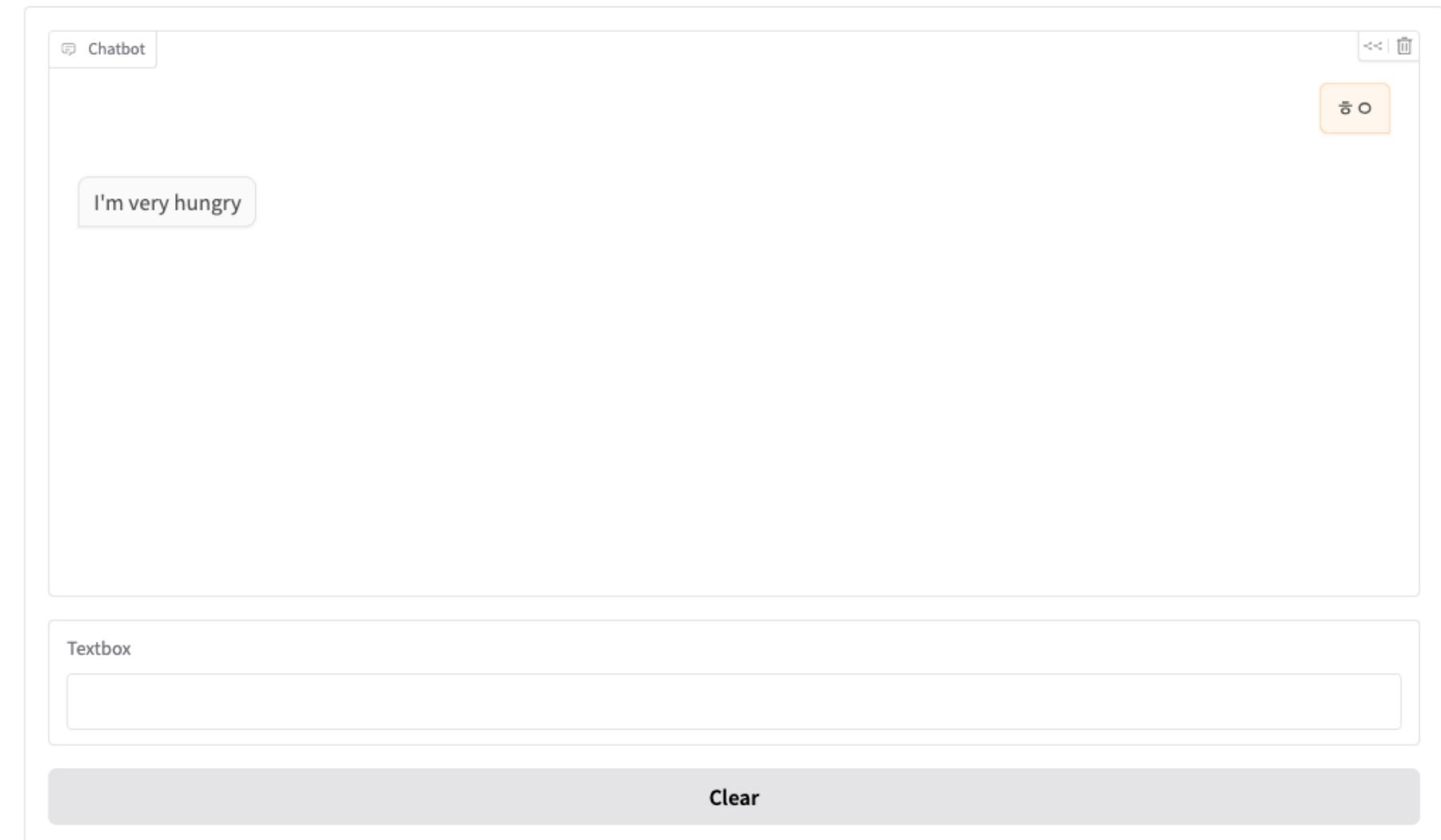
```

The image shows two screenshots of a Gradio application interface. In the top screenshot, the 'output_col' section is hidden (visible=False), so the 'Diagnosis' and 'Patient Summary' boxes are not visible. In the bottom screenshot, after a click, the 'output_col' section is shown (visible=True), revealing the 'Diagnosis' box containing 'flu' and the 'Patient Summary' box containing '임현석, 66 y/o'. A large black arrow points from the top state to the bottom state, indicating the change in visibility.

- Column, Row 혹은 Components에 visible 인자를 통하여 시각화 유무를 결정할 수 있음.

04 ChatBot

```
1 import gradio as gr
2 import random
3 import time
4
5 with gr.Blocks() as demo:
6     chatbot = gr.Chatbot(type="messages")
7     msg = gr.Textbox()
8     clear = gr.ClearButton([msg, chatbot])
9
10    def respond(message, chat_history):
11        bot_message = random.choice(["How are you?", "Today is a great day", "I'm very hungry"])
12        chat_history.append({"role": "user", "content": message})
13        chat_history.append({"role": "assistant", "content": bot_message})
14        time.sleep(2)
15        return "", chat_history
16
17    msg.submit(respond, [msg, chatbot], [msg, chatbot])
18
19 demo.launch()
```



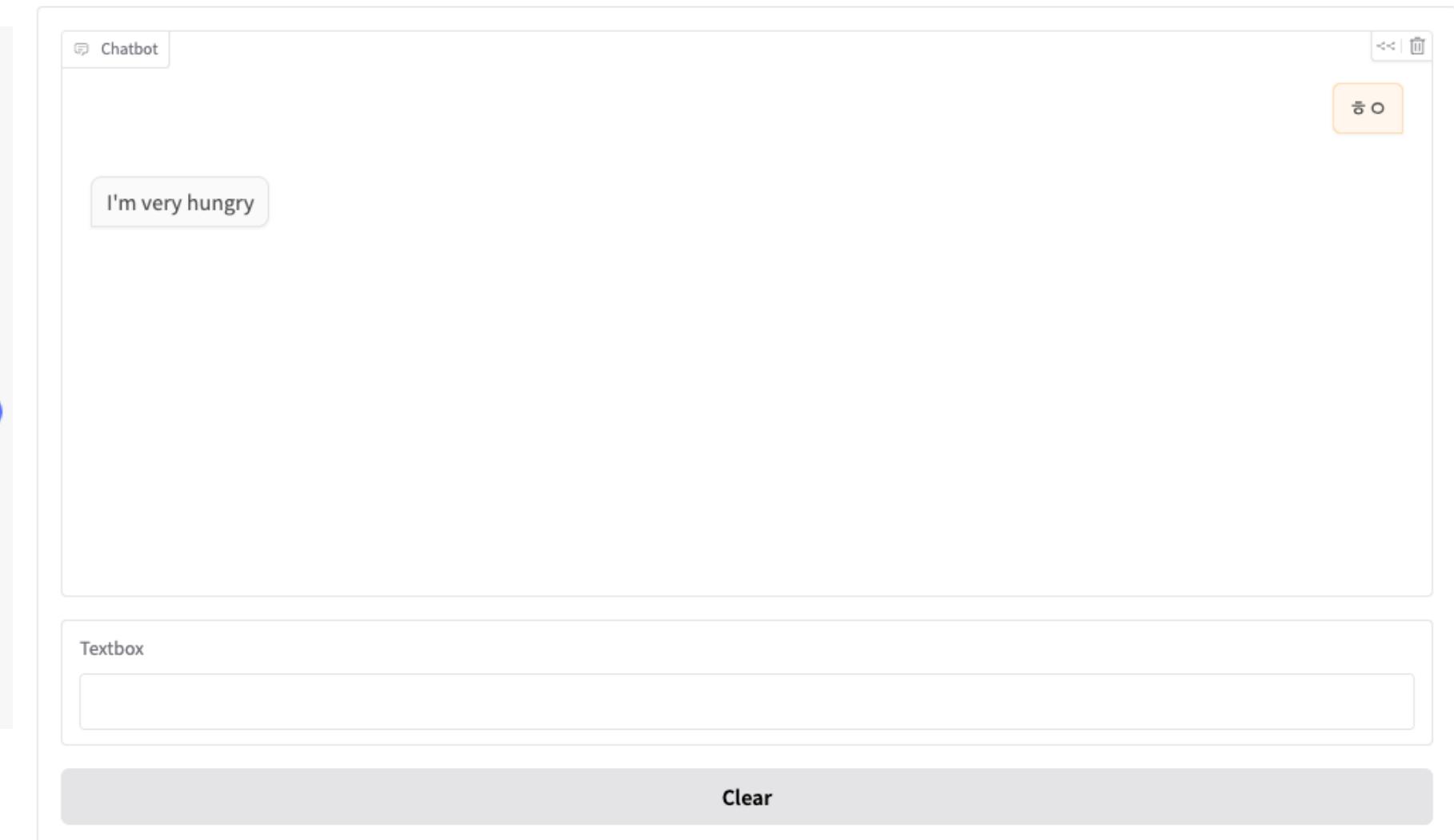
- Gradio에서는 챗봇을 위한 Components 또한 제공하고 있음.

04 ChatBot

```

1 import gradio as gr
2 import random
3 import time
4
5 with gr.Blocks() as demo:
6     chatbot = gr.Chatbot(type="messages")
7     msg = gr.Textbox()
8     clear = gr.ClearButton([msg, chatbot])
9
10    def respond(message, chat_history):
11        bot_message = random.choice(["How are you?", "Today is a great day", "I'm very hungry"])
12        chat_history.append({"role": "user", "content": message})
13        chat_history.append({"role": "assistant", "content": bot_message})
14        time.sleep(2)
15        return "", chat_history
16
17    msg.submit(respond, [msg, chatbot], [msg, chatbot])
18
19 demo.launch()

```



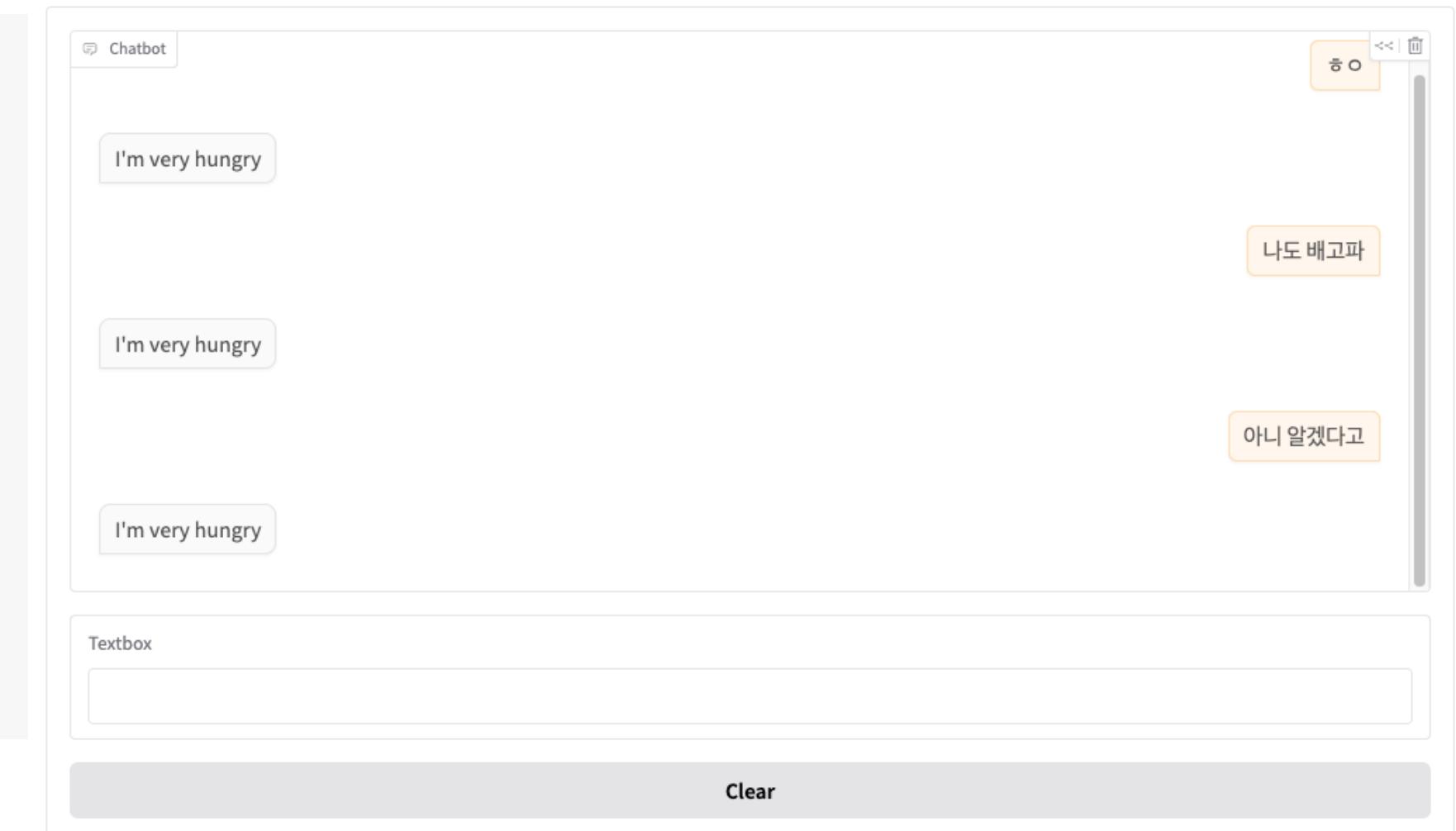
- Chatbot: 대화의 전체 기록을 저장하는 값으로 사용자와 봇 간의 응답 쌍 목록
- Textbox: 사용자가 메시지를 입력한 다음 Enter 버튼을 통해 질문을 제공하고 응답을 제공받음
- ClearButton: Textbox와 전체 챗봇 기록을 지움(Clear history)

04 ChatBot

```

1 import gradio as gr
2 import random
3 import time
4
5 with gr.Blocks() as demo:
6     chatbot = gr.Chatbot(type="messages")
7     msg = gr.Textbox()
8     clear = gr.ClearButton([msg, chatbot])
9
10    def respond(message, chat_history):
11        bot_message = random.choice(["How are you?", "Today is a great day", "I'm very hungry"])
12        chat_history.append({"role": "user", "content": message})
13        chat_history.append({"role": "assistant", "content": bot_message})
14        time.sleep(2)
15        return "", chat_history
16
17    msg.submit(respond, [msg, chatbot], [msg, chatbot])
18
19 demo.launch()

```



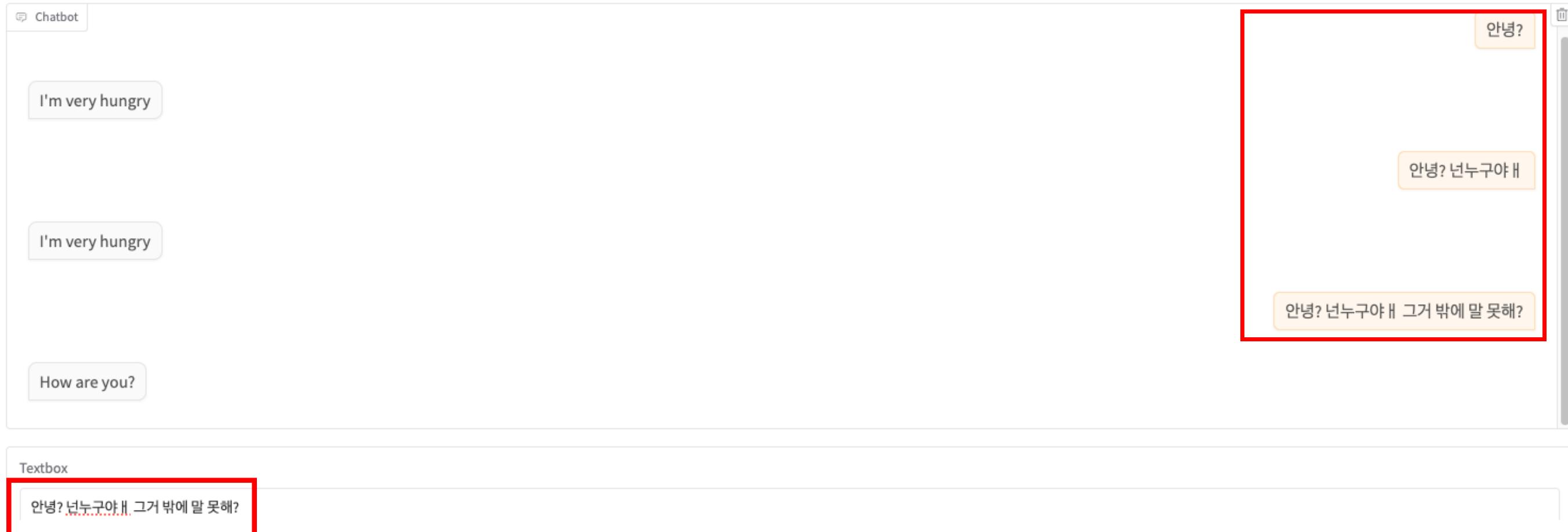
- `respond` 함수에서 `return`으로 ""을 제공하는 이유는 다음 대화를 위해 `Textbox`(질문이 들어가는 곳)초기화해주는 것
- 이미 대화 내용은 `chat_history`에 저장되어 있음.

04 ChatBot

```

1 import gradio as gr
2 import random
3 import time
4
5 with gr.Blocks() as demo:
6     chatbot = gr.Chatbot(type="messages")
7     msg = gr.Textbox()
8     clear = gr.ClearButton([msg, chatbot])
9
10    def respond(message, chat_history):
11        bot_message = random.choice(["How are you?", "Today is a great day", "I'm very hungry"])
12        chat_history.append({"role": "user", "content": message})
13        chat_history.append({"role": "assistant", "content": bot_message})
14        time.sleep(2)
15        return message, chat_history
16
17    msg.submit(respond, [msg, chatbot], [msg, chatbot])
18
19 demo.launch()

```



- “”을 return하지 않고 messages를 return하였을 때, 아래의

그림과 같이 그 전의 질문이 계속 이어지는 현상을 볼 수 있음.

05. Gradio Docs

01 수업 자료 활용

gradio

Quickstart Docs Playground Custom Components Community Search

Getting Started 5.6.0 The Interface Class →

Quickstart

- Installation
- Building Your First Demo
- Sharing Your Demo
- An Overview of Gradio
- Custom Demos with gr.Blocks
- Chatbots with gr.ChatInterface
- The Gradio Python & JavaScript Ecosystem
- What's Next?

Building Interfaces

- The Interface Class
- More On Examples
- Flagging
- Interface State
- Reactive Interfaces
- Four Kinds Of Interfaces

Building With Blocks

- Blocks And Event Listeners
- Controlling Layout
- State In Blocks
- Dynamic Apps With Render Decorator

Quickstart

Gradio is an open-source Python package that allows you to quickly **build** a demo or web application for your machine learning model, API, or any arbitrary Python function. You can then **share** a link to your demo or web application in just a few seconds using Gradio's built-in sharing features. *No JavaScript, CSS, or web hosting experience needed!*

Users > abidlabs > Desktop > app.py > ...

```

1  from model import generate_image
2
3  import gradio as gr
4
5  with gr.Blocks() as demo:
6      gr.Markdown("# Your Fancy Image Generator")
7      with gr.Row(equal_height=True):
8          textbox = gr.Textbox(lines=1, show_label=False, p)
9          button = gr.Button("Generate", variant="primary")
10         image = gr.Image(height=250)
11
12         button.click(
13             generate_image,
14             inputs=textbox,
15             outputs=image,
16         )
17
18     demo.launch(share=True)
19     # Creates a public link like:
20     # https://b25f924d951331d187.gradio.live

```

Your Fancy Image Generator

- <https://www.gradio.app/guides/quickstart>

02 각 클래스별 자세한 내용

The screenshot shows the Gradio documentation website with the URL <https://www.gradio.app/docs/gradio/interface>. The page title is "Interface". The left sidebar shows a navigation menu with sections like "Building Demos", "Interface", "Blocks", "Blocks Layout", "Components", and "Demos". The "Interface" section is currently selected. The main content area contains the following sections:

- New to Gradio? Start here: Getting Started**
- See the Release History**
- ChatInterface →**
- Description**: A detailed description of the Interface class, stating it's the main high-level class for creating web-based GUIs around machine learning models.
- Example Usage**: A code snippet demonstrating how to create a demo using the Interface class:

```
import gradio as gr

def image_classifier(inp):
    return {'cat': 0.3, 'dog': 0.7}

demo = gr.Interface(fn=image_classifier, inputs="image", outputs="label")
demo.launch()
```

- Initialization**

The right side of the interface lists various methods and guides:

- Interface: Description, Example Usage, Initialization, Demos, Methods, launch, load, from_pipeline, integrate, queue, Guides.

- <https://www.gradio.app/docs/gradio/interface>

감사합니다.