Gradio를 활용한 나만의 웹앱 구축

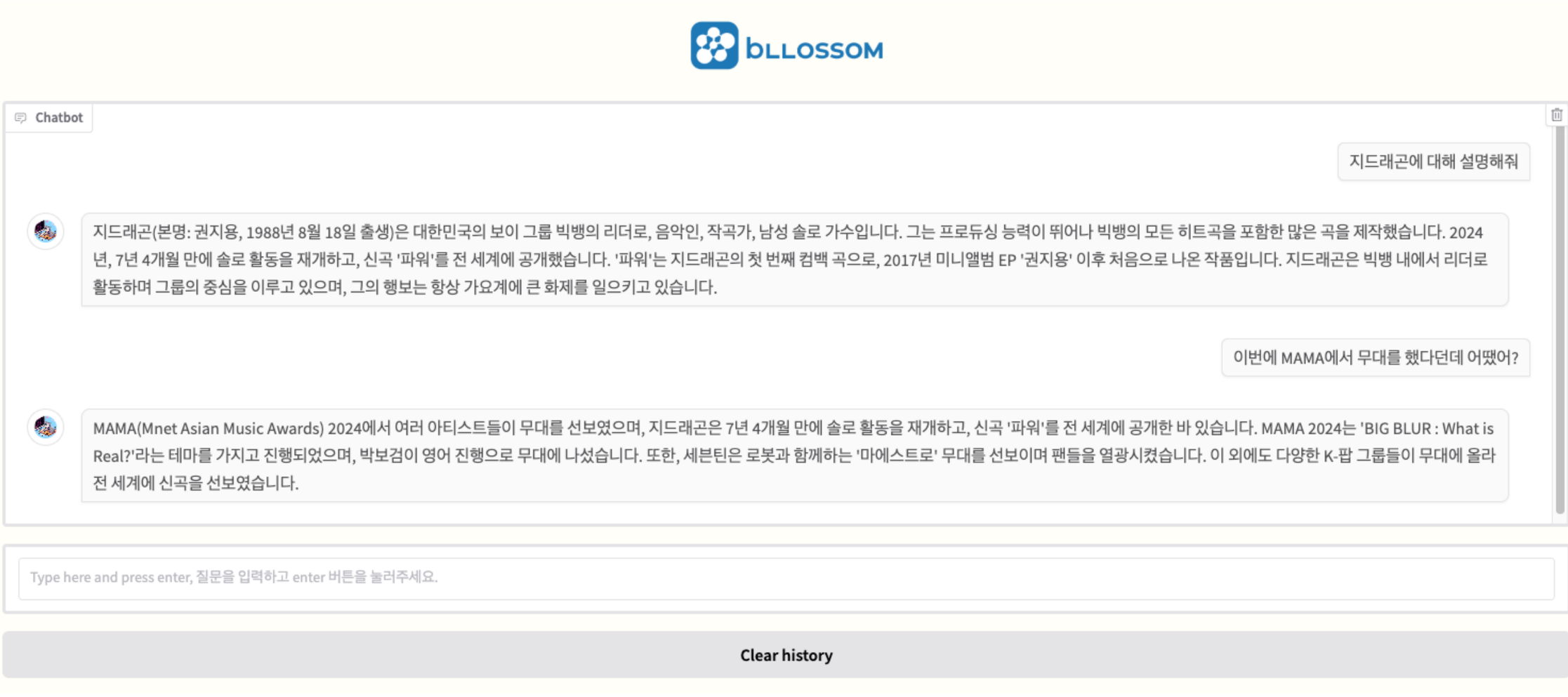
Contents

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

02 Gradio 활용 예시 - LLM 어플리케이션



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

Examples

Describe what is interesting about this image.

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

Parameters

X-LLaVA Chatbot

이 이미지를 보고 느껴지는 분위기에 대해 설명해줘

이 이미지는 웅장한 산악 지형과 그 아래로 펼쳐진 해안 풍경의 숨 막히는 전망을 보여줍니다. 바다와 해변, 해안선을 따라 자리 잡은 마을, 그리고 웅장한 산봉우리가 있는 광활한 풍경은 경외심과 자연의 장엄함을 불러일으킵니다. 맑은 하늘과 밝은 햇빛은 생동감과 활력을 불어넣어, 이러한 환경의 아름다움과 고요함을 강조합니다. 이 이미지는 자연의 위엄에 대한 경외감과 자연의 경이로움에 대한 감탄을 불러일으킵니다.

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.

Built with Gradio

02. Gradio: Interface

00 환경설정

```
1 !pip install -U gradio
```

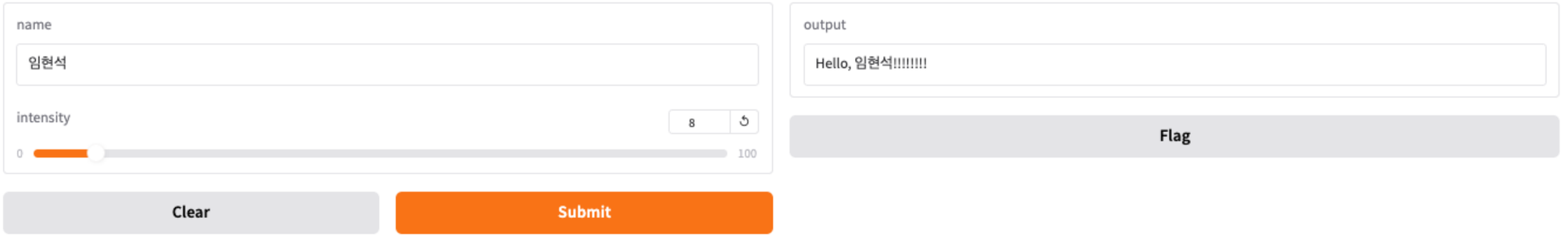
```
Collecting gradio
  Downloading gradio-5.6.0-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<24.0,>=22.0 (from gradio)
  Downloading aiofiles-23.2.1-py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.7.1)
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.5-py3-none-any.whl.metadata (27 kB)
Collecting ffmpy (from gradio)
  Downloading ffmpy-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)
```

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

01 Interface

```python
import gradio as gr

def greet(name, intensity):
    return "Hello, " + name + "!" * int(intensity)

demo = gr.Interface(
    fn=greet,
    inputs=["text", "slider"],
    outputs=["text"],
)

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)
```



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

01 Interface

```python
import gradio as gr

def greet(name, intensity):
    return "Hello, " + name + "!" * int(intensity)

demo = gr.Interface(
    fn=greet,
    inputs=["text", "slider"],
    outputs=["text"],
)

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

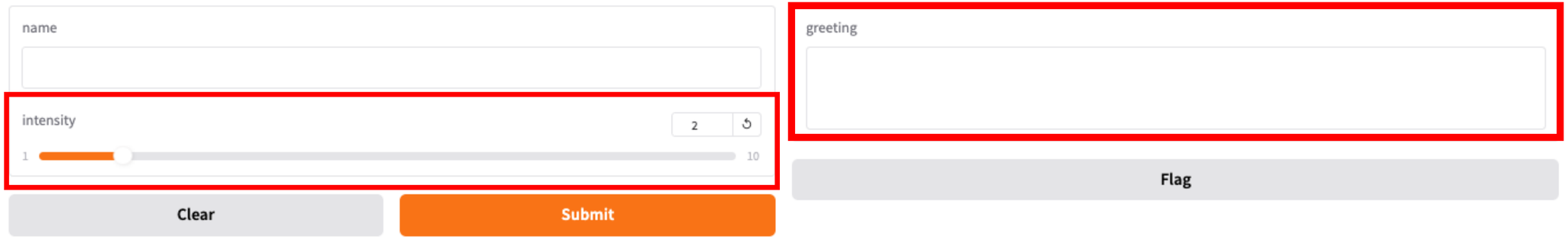
```python
import gradio as gr

def greet(name, intensity):
    return "Hello, " + name + "!" * int(intensity)

demo = gr.Interface(
    fn=greet,
    inputs=["text", "slider"],
    outputs=["text"],
)

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

```python
import gradio as gr

def greet(name, intensity):
    return "Hello, " + name + "!" * intensity

demo = gr.Interface(
    fn=greet,
    inputs=["text", gr.Slider(value=2, minimum=1, maximum=10, step=1)],
    outputs=[gr.Textbox(label="greeting", lines=3)],
)

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)
```



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

02 Components

```python
import numpy as np
import gradio as gr

def sepia(input_img):
    sepia_filter = np.array([
        [0.393, 0.769, 0.189],
        [0.349, 0.686, 0.168],
        [0.272, 0.534, 0.131]
    ])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

demo = gr.Interface(sepia, gr.Image(), "image")
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://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

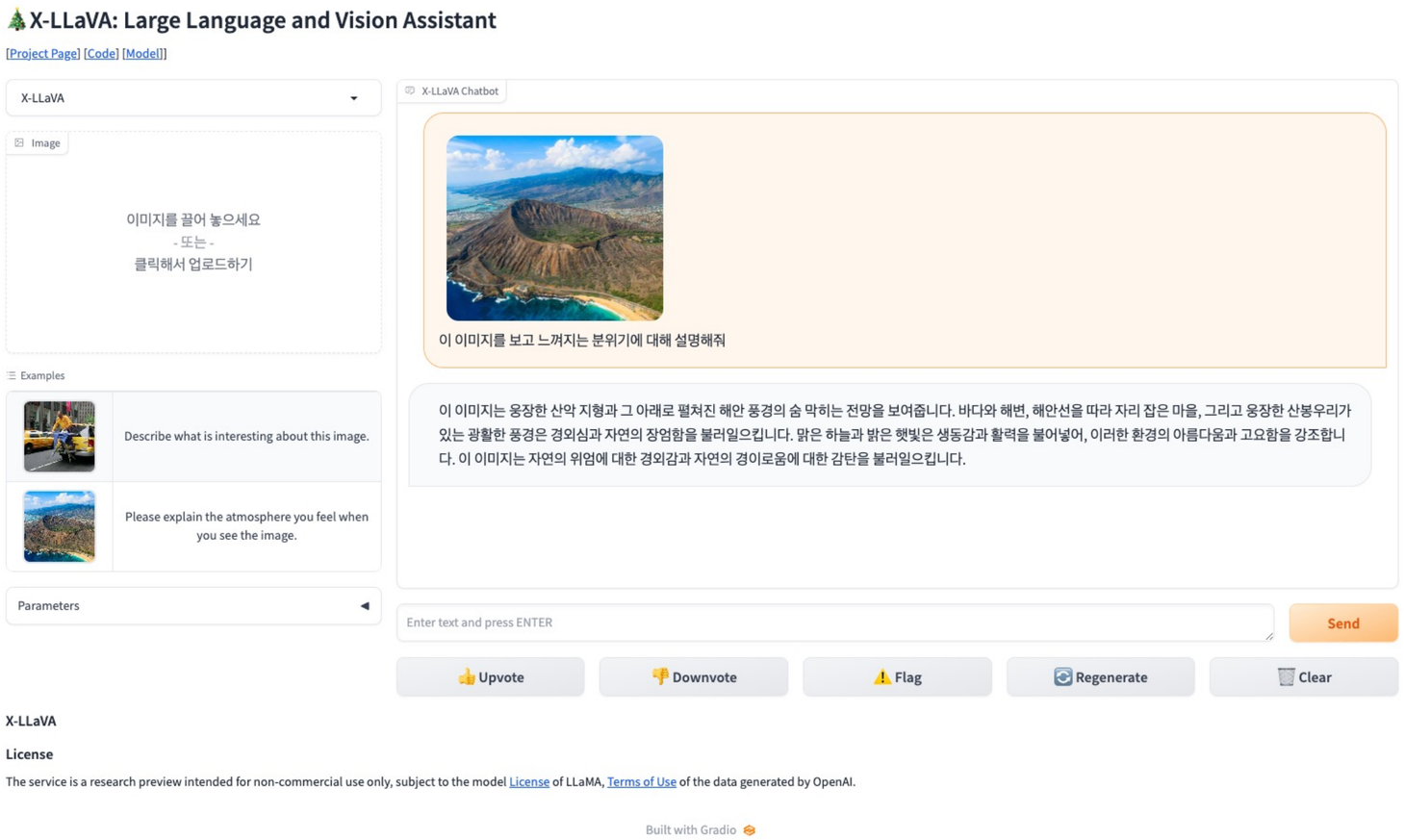
- TextBox: 텍스트 입력
- Number: 숫자 입력
- Image: 이미지 업로드 or 웹캠을 통한 입력
- Audio: 오디오 파일 업로드 or 녹음 후 입력
- Video: 비디오 파일 업로드 or 웹캠을 통한 입력
- Checkbox: boolean 값 입력
- Dropdown: 미리 정의된 옵션 중 하나 선택
- Slider: 범위 내의 숫자 값 선택
- File: 파일 업로드

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

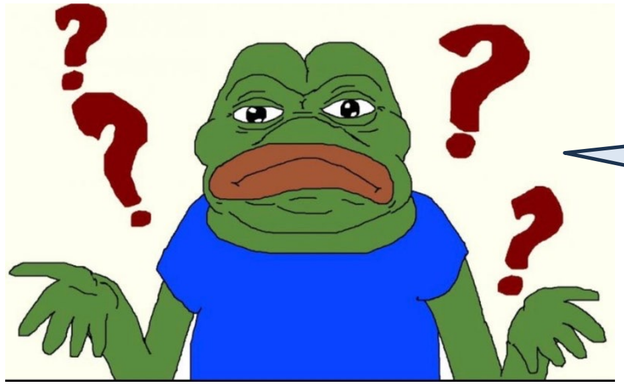
01 Blocks Structure



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 뭐임 ㅋ

배우기 힘든데 그거 왜씀 ㅋ

(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

```python
import gradio as gr


def greet(name):
    return "Hello " + name + "!"


with gr.Blocks() as demo:
    name = gr.Textbox(label="Name")
    output = gr.Textbox(label="Output Box")
    greet_btn = gr.Button("Greet")
    greet_btn.click(fn=greet, inputs=name, outputs=output, api_name="greet")

demo.launch()
```

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

01 Blocks Structure

```python
import gradio as gr


def greet(name):
    return "Hello " + name + "!"


with gr.Blocks() as demo:
    name = gr.Textbox(label="Name")
    output = gr.Textbox(label="Output Box")
    greet_btn = gr.Button("Greet")
    greet_btn.click(fn=greet, inputs=name, outputs=output, api_name="greet")

demo.launch()
```

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

02 Event Listener

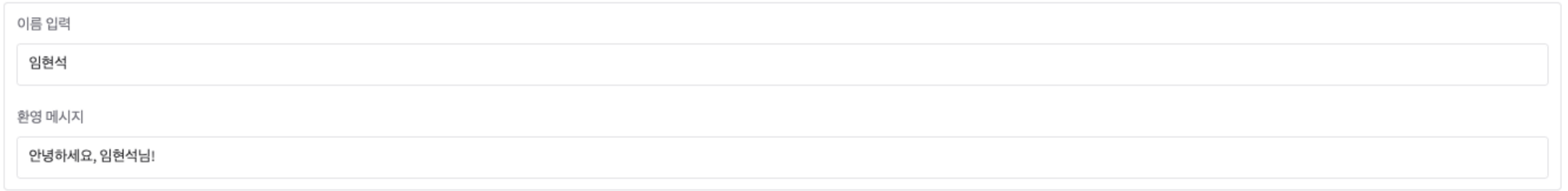
```python
import gradio as gr


def greet(name):
    return "Hello " + name + "!"


with gr.Blocks() as demo:
    name = gr.Textbox(label="Name")
    output = gr.Textbox(label="Output Box")
    greet_btn = gr.Button("Greet")
    greet_btn.click(fn=greet, inputs=name, outputs=output, api_name="greet")

demo.launch()
```

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

02 Event Listener

```python
import gradio as gr

def welcome(name):
    return f"안녕하세요, {name}님!"

with gr.Blocks() as demo:
    inp = gr.Textbox(label="이름 입력")
    out = gr.Textbox(label="환영 메시지")

    # Textbox의 값이 변경될 때마다 welcome 함수를 호출
    inp.change(fn=welcome, inputs=inp, outputs=out)

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://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

```python
with gr.Blocks() as demo:
    with gr.Row():
        btn1 = gr.Button("Button 1")
        btn2 = gr.Button("Button 2")

demo.launch()
```

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

01 Controlling Layout

```python
import gradio as gr

with gr.Blocks() as demo:
    with gr.Row(equal_height=True):
        textbox = gr.Textbox()
        btn2 = gr.Button("Button 2")

demo.launch()
```

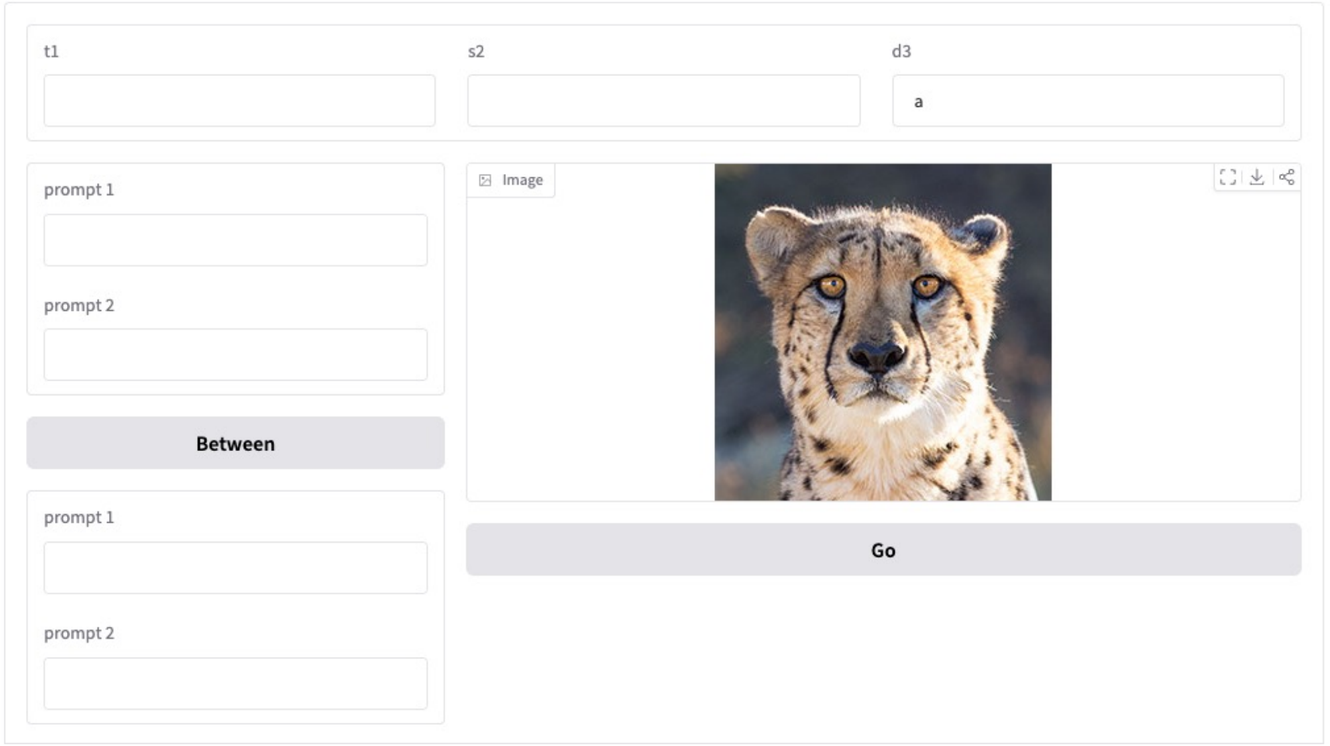
equal_height를 하지 않았을 때,

equal_height를 설정하였을 때,

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

01 Controlling Layout

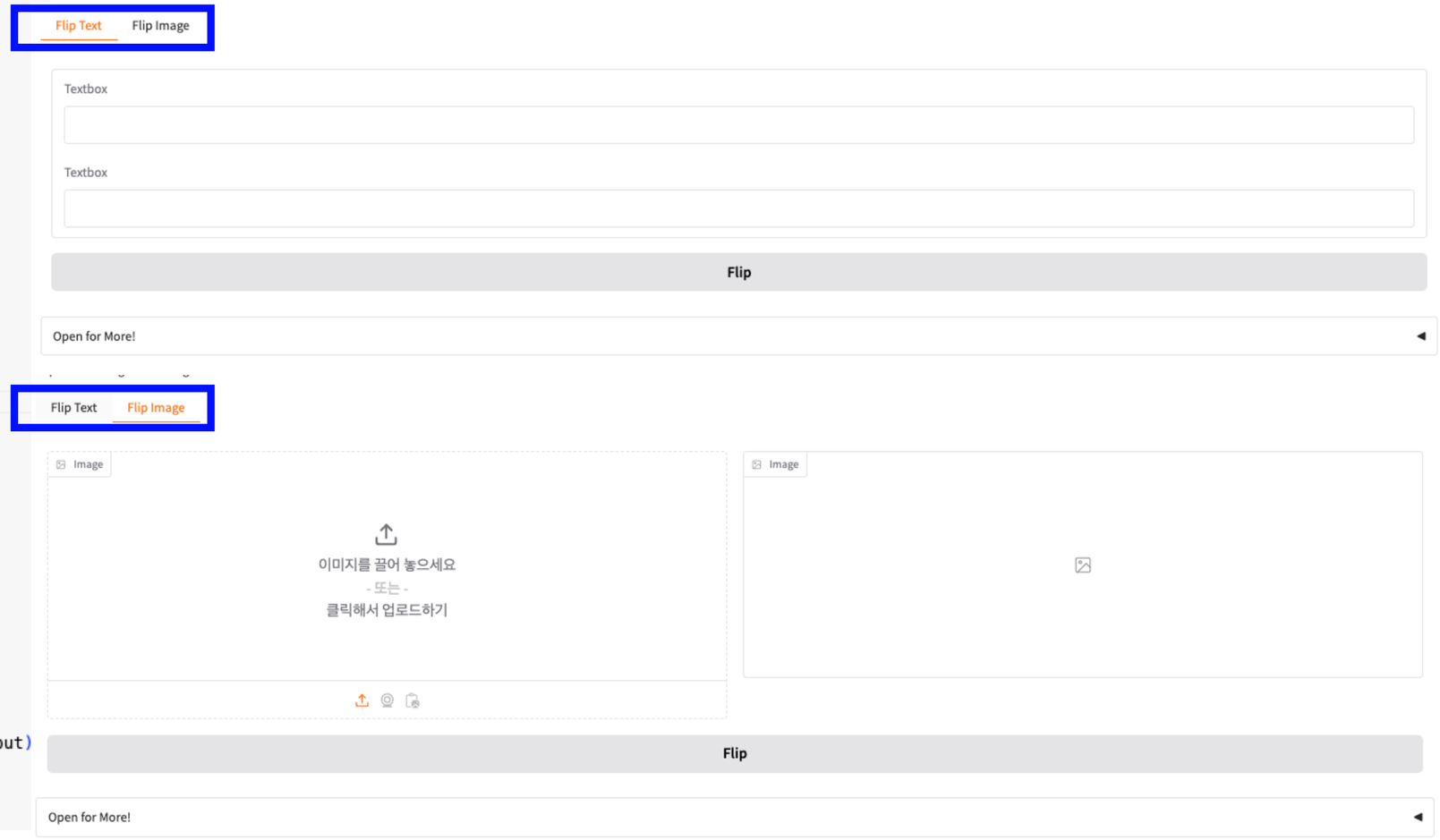
```
import gradio as gr

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

```python
import numpy as np
import gradio as gr

def flip_text(x):
    return x[::-1]

def flip_image(x):
    return np.fliplr(x)

with gr.Blocks() as demo:
    gr.Markdown("Flip text or image files using this demo.")
    with gr.Tab("Flip Text"):
        text_input = gr.Textbox()
        text_output = gr.Textbox()
        text_button = gr.Button("Flip")
    with gr.Tab("Flip Image"):
        with gr.Row():
            image_input = gr.Image()
            image_output = gr.Image()
        image_button = gr.Button("Flip")

    with gr.Accordion("Open for More!", open=False):
        gr.Markdown("Look at me...")
        temp_slider = gr.Slider(
            0, 1,
            value=0.1,
            step=0.1,
            interactive=True,
            label="Slide me",
        )

    text_button.click(flip_text, inputs=text_input, outputs=text_output)
    image_button.click(flip_image, inputs=image_input, outputs=image_output)

demo.launch()
```

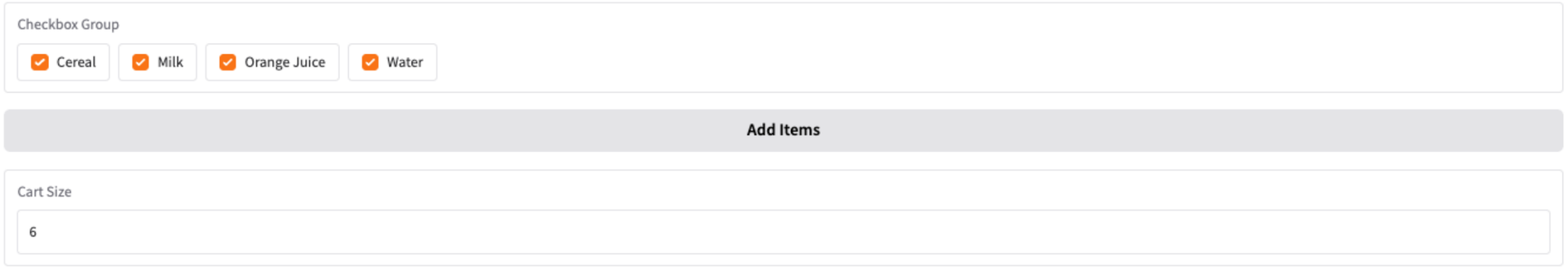


- Tab은 크롬의 탭 창과 같이 하나를 새로 생성할 수 있음.
- 각 Tab의 컨텍스트 내에 있는 Components만 표시.

02 State

```python
import gradio as gr

with gr.Blocks() as demo:
    cart = gr.State([])
    items_to_add = gr.CheckboxGroup(["Cereal", "Milk", "Orange Juice", "Water"])

    def add_items(new_items, previous_cart):
        print(new_items)
        cart = previous_cart + new_items
        return cart

    gr.Button("Add Items").click(add_items, [items_to_add, cart], cart)

    cart_size = gr.Number(label="Cart Size")
    cart.change(lambda cart: len(cart), cart, cart_size)

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)
```



- gr.State를 활용하여 사용자별로 상태를 유지해야 하는 경우 사용함

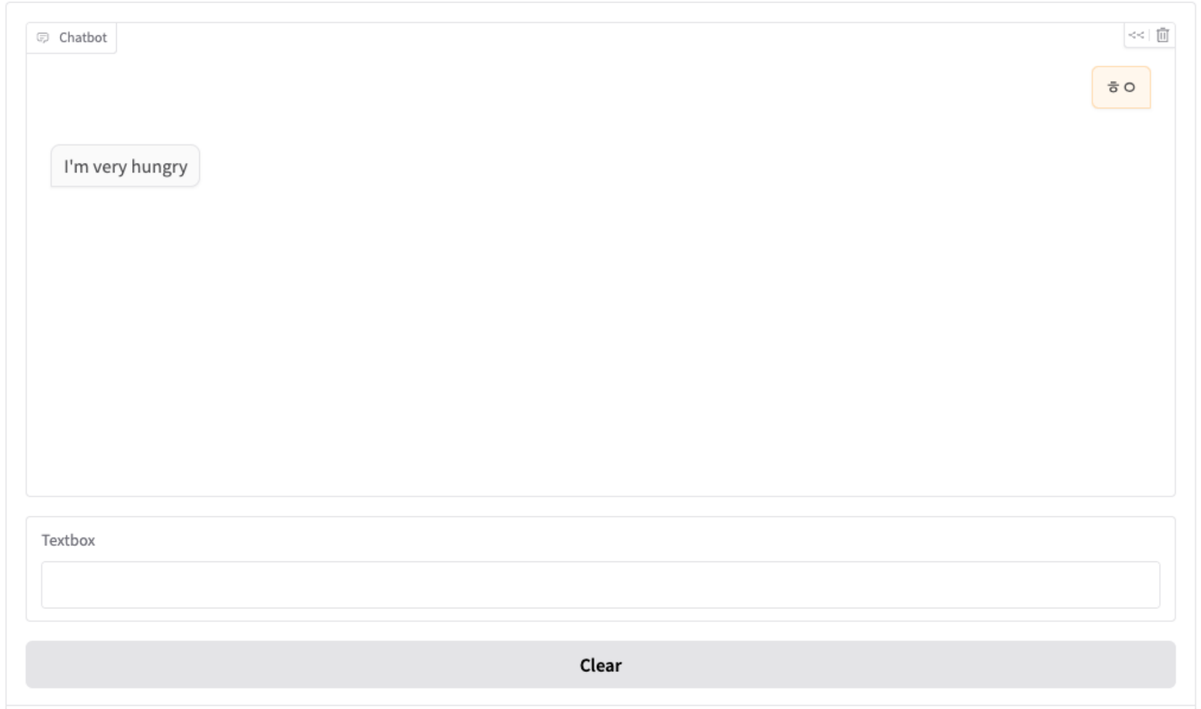
03 Visible

```python
import gradio as gr

with gr.Blocks() as demo:
    name_box = gr.Textbox(label="Name")
    age_box = gr.Number(label="Age", minimum=0, maximum=100)
    symptoms_box = gr.CheckboxGroup(["Cough", "Fever", "Runny Nose"])
    submit_btn = gr.Button("Submit")

    with gr.Column(visible=False) as output_col:
        diagnosis_box = gr.Textbox(label="Diagnosis")
        patient_summary_box = gr.Textbox(label="Patient Summary")

    def submit(name, age, symptoms):
        return {
            submit_btn: gr.Button(visible=False),
            output_col: gr.Column(visible=True),
            diagnosis_box: "covid" if "Cough" in symptoms else "flu",
            patient_summary_box: f"{name}, {age} y/o",
        }

    submit_btn.click(
        submit,
        [name_box, age_box, symptoms_box],
        [submit_btn, diagnosis_box, patient_summary_box, output_col],
    )

demo.launch()
```

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

04 ChatBot

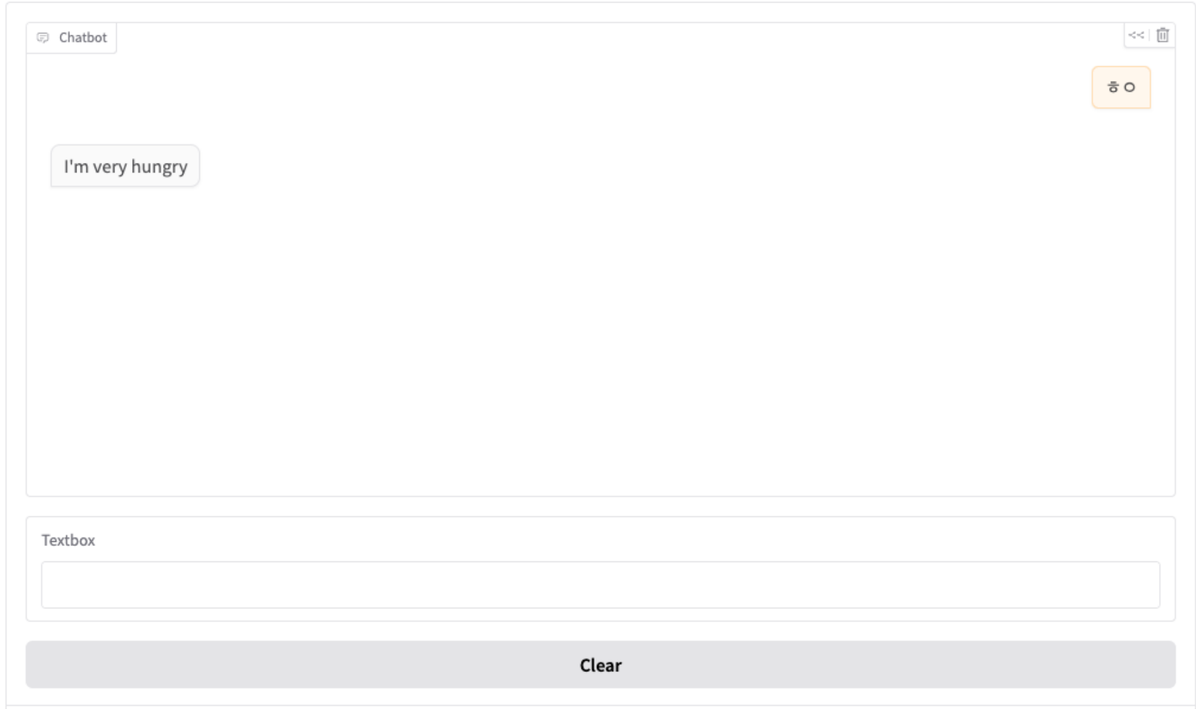
```python
import gradio as gr
import random
import time

with gr.Blocks() as demo:
    chatbot = gr.Chatbot(type="messages")
    msg = gr.Textbox()
    clear = gr.ClearButton([msg, chatbot])

    def respond(message, chat_history):
        bot_message = random.choice(["How are you?", "Today is a great day", "I'm very hungry"])
        chat_history.append({"role": "user", "content": message})
        chat_history.append({"role": "assistant", "content": bot_message})
        time.sleep(2)
        return "", chat_history

    msg.submit(respond, [msg, chatbot], [msg, chatbot])

demo.launch()
```



- Gradio에서는 챗봇을 위한 Components 또한 제공하고 있음.

04 ChatBot

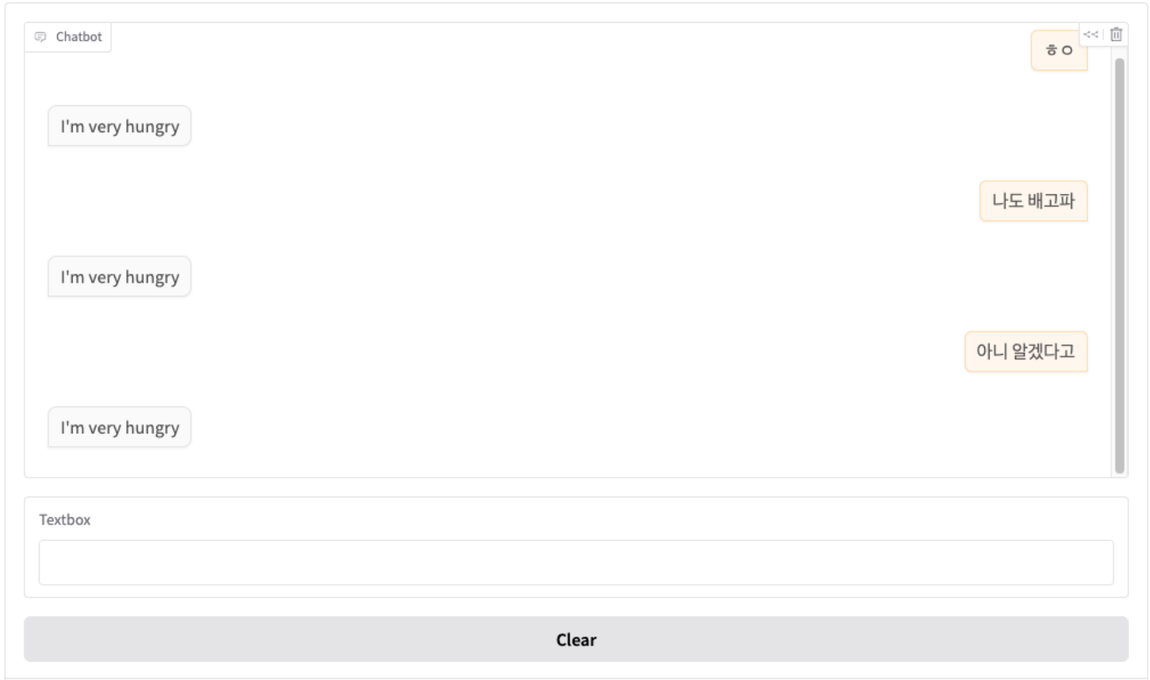
```python
import gradio as gr
import random
import time

with gr.Blocks() as demo:
    chatbot = gr.Chatbot(type="messages")
    msg = gr.Textbox()
    clear = gr.ClearButton([msg, chatbot])

    def respond(message, chat_history):
        bot_message = random.choice(["How are you?", "Today is a great day", "I'm very hungry"])
        chat_history.append({"role": "user", "content": message})
        chat_history.append({"role": "assistant", "content": bot_message})
        time.sleep(2)
        return "", chat_history

    msg.submit(respond, [msg, chatbot], [msg, chatbot])

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

```python
import gradio as gr
import random
import time

with gr.Blocks() as demo:
    chatbot = gr.Chatbot(type="messages")
    msg = gr.Textbox()
    clear = gr.ClearButton([msg, chatbot])

    def respond(message, chat_history):
        bot_message = random.choice(["How are you?", "Today is a great day", "I'm very hungry"])
        chat_history.append({"role": "user", "content": message})
        chat_history.append({"role": "assistant", "content": bot_message})
        time.sleep(2)
        return "", chat_history

    msg.submit(respond, [msg, chatbot], [msg, chatbot])

demo.launch()
```

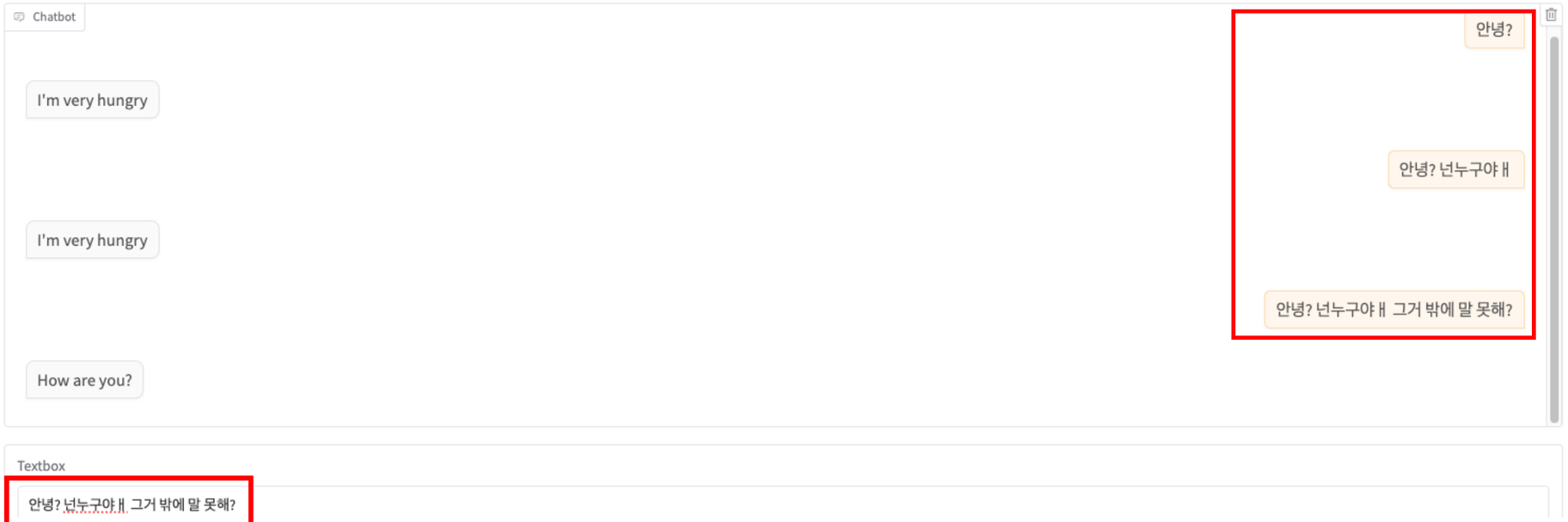


- respond 함수에서 return으로 “”을 제공하는 이유는 다음 대화를 위해 Textbox(질문이 들어가는 곳)초기화해주는 것
- 이미 대화 내용은 chat_history에 저장되어 있음.

04 ChatBot

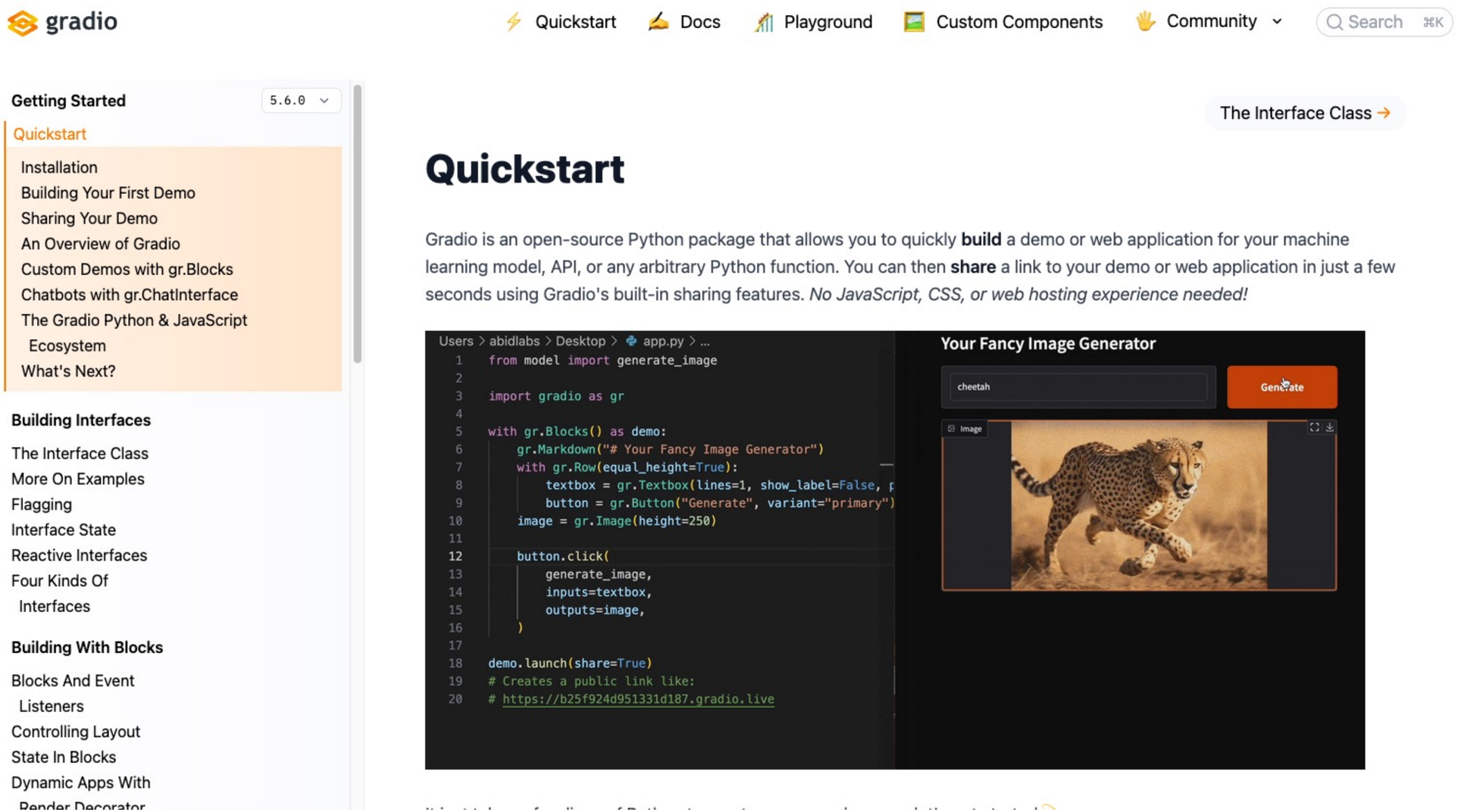
```python
import gradio as gr
import random
import time

with gr.Blocks() as demo:
    chatbot = gr.Chatbot(type="messages")
    msg = gr.Textbox()
    clear = gr.ClearButton([msg, chatbot])

    def respond(message, chat_history):
        bot_message = random.choice(["How are you?", "Today is a great day", "I'm very hungry"])
        chat_history.append({"role": "user", "content": message})
        chat_history.append({"role": "assistant", "content": bot_message})
        time.sleep(2)
        return message, chat_history

    msg.submit(respond, [msg, chatbot], [msg, chatbot])

demo.launch()
```

- “”을 return하지 않고 messages를 return하였을 때, 아래의 그림과 같이 그 전의 질문이 계속 이어지는 현상을 볼 수 있음.



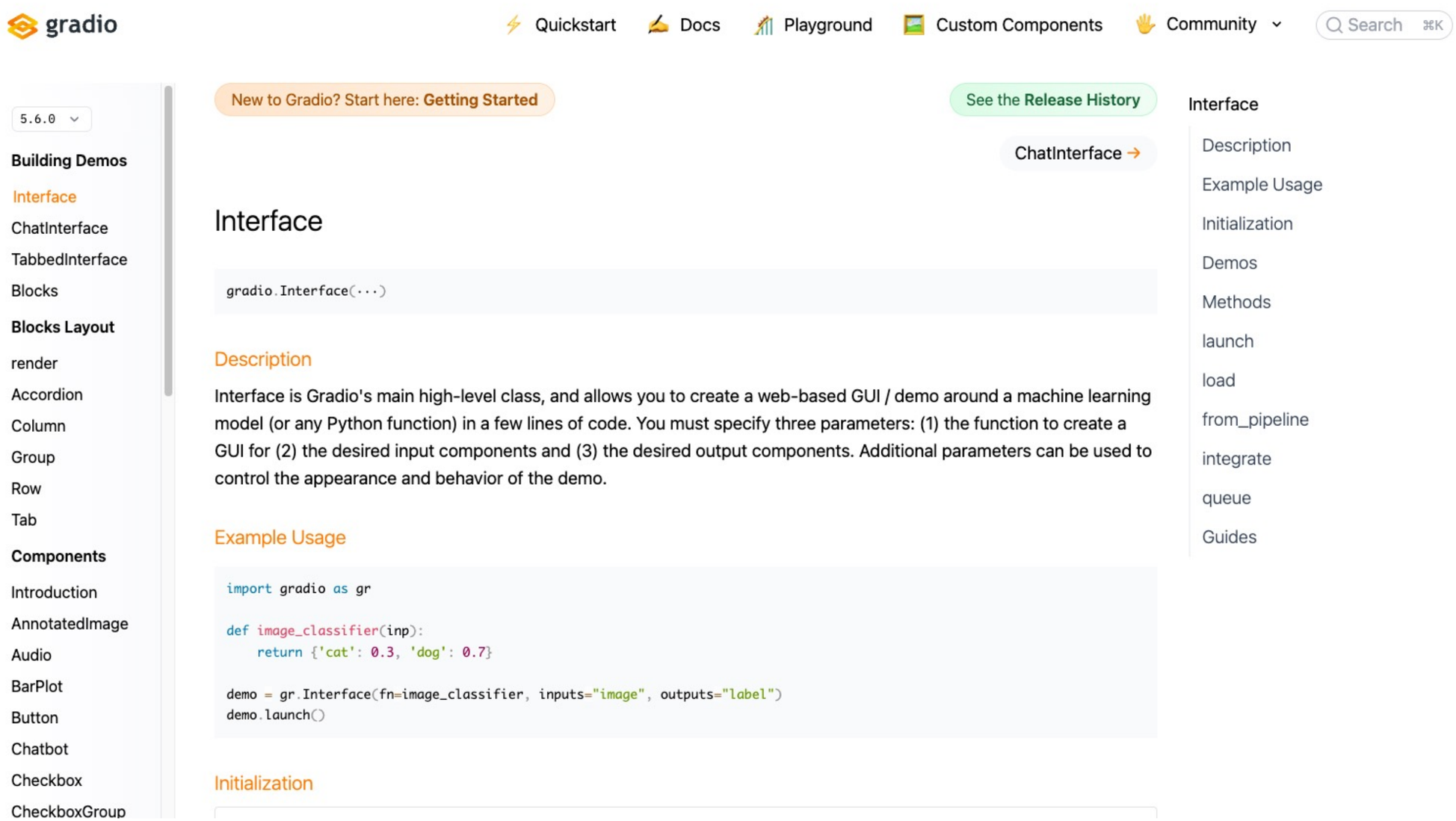
05. Gradio Docs

01 수업 자료 활용



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

02 각 클래스별 자세한 내용



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

감사합니다.