

DAY – 96

18 December 2025

Chatbot Popup – JavaScript (script.js)

1. Introduction of JavaScript File

This script.js file controls the **working and behavior** of the chatbot popup interface.

It acts as a **bridge between frontend UI and backend Flask server**.

JavaScript handles:

- Opening and closing chatbot popup
- Sending user messages
- Receiving chatbot responses
- Language switching (English / Punjabi)
- Typing animation
- Streaming-like text display
- Stop typing feature

2. Global Variable Declaration

```
let typingInterval = null;
```

```
let isStreaming = false;
```

```
let stopRequested = false;
```

```
let chatLanguage = "english";
```

Explanation:

- typingInterval
Stores the interval ID for the typing animation (“typing...”).
- isStreaming
Indicates whether the bot is currently generating a response.

- `stopRequested`
Used to stop the response generation when user clicks stop or presses Enter again.
- `chatLanguage`
Stores the selected chatbot language (default is English).

These variables help in **controlling chatbot state**.

3. Toggle Chat Popup Function

```
function toggleChat() {
```

This function is called when the chat button or close button is clicked.

```
const chatPopup = document.getElementById("chatPopup");  
chatPopup.style.display = chatPopup.style.display === "flex" ? "none" : "flex";
```

Explanation:

- If chatbot is open → close it
- If chatbot is closed → open it
- Uses flex layout for vertical alignment

```
if (chatPopup.style.display === "flex") {  
  setTimeout(() => document.getElementById("userInput").focus(), 100);  
}
```

- Automatically focuses on input field when chatbot opens.
- Improves user experience.

4. Language Change Handling Function

```
function setChatLanguage() {
```

This function is triggered when the user changes the language from the dropdown.

```
const select = document.getElementById("langSelect");  
chatLanguage = select.value;
```

- Gets selected language.
- Updates the global chatLanguage variable.

4.1 Update Chat Title

```
const title = document.getElementById("chatTitle");  
if (title) title.textContent = chatLanguage === "english" ? "Help " : "ਸਹਾਇਤਾ ";
```

- Changes chatbot title dynamically.
- Displays title in selected language.

4.2 Update Input Placeholder

```
const input = document.getElementById("userInput");  
input.placeholder =  
  chatLanguage === "english"  
    ? "Type your message here..."  
    : "ਇੱਥੇ ਆਪਣਾ ਸੁਨੇਹਾ ਲਿਖੋ...";
```

- Updates placeholder text based on selected language.
- Makes chatbot bilingual.

4.3 Update Initial Bot Message

```
const initialMsg = document.getElementById("initialBotMsg");
initialMsg.textContent =
  chatLanguage === "english"
    ? "Hey! How can I help you today?"
    : "ਹੈਲੋ! ਮੈਂ ਤੁਹਾਡੀ ਕਿਵੇਂ ਸਹਾਇਤਾ ਕਰ ਸਕਦਾ ਹਾਂ?";
```

- Updates welcome message.
- Improves localization support.

5. Send Message Function

```
async function sendMessage() {
```

This is the **main function** that:

- Sends user input to backend
- Receives chatbot response
- Displays response with animation

5.1 Stop Streaming Check

```
if (isStreaming) {
  stopRequested = true;
  return;
}
```

- If chatbot is already replying, clicking send acts as **stop button**.
- Improves user control.

5.2 Read User Input

```
const message = input.value.trim();  
if (!message) return;
```

- Removes extra spaces.
- Prevents sending empty messages.

5.3 Display User Message

```
const userMsg = document.createElement("div");  
userMsg.className = "chat-message user";  
userMsg.textContent = message;
```

- Creates user message bubble.
- Appends it to chat body.

```
chatBody.appendChild(userMsg);  
chatBody.scrollTo({ top: chatBody.scrollHeight, behavior: "smooth" });
```

- Automatically scrolls chat to bottom.

5.4 Disable Input During Bot Response

```
input.value = "";  
input.disabled = true;
```

- Prevents user input while bot is responding.

5.5 Initialize Bot Response State

```
stopRequested = false;  
isStreaming = true;  
sendBtn.textContent = "■";
```

- Shows stop icon instead of send arrow.
- Indicates bot is typing.

6. Typing Indicator Animation

```
typingInterval = setInterval(() => {
```

Explanation:

- Displays "typing..." animation.
- Adds dots every 400ms.
- Improves conversational feel.

7. Sending Request to Backend

```
const res = await fetch("/get", {
```

Explanation:

- Sends POST request to Flask backend.
- URL /get handles chatbot logic.

```
body: JSON.stringify({ msg: message, lang: chatLanguage })),
```

- Sends:
 - User message
 - Selected language

8. Handling Backend Response

```
const data = await res.json();
```

- Converts response to JSON format.

```
const fullText =
```

```
data.response || "Sorry, I don't have an answer for that. ☐";
```

- Displays default message if no response found.

9. Streaming-Like Text Display

```
await revealTextSimulatingStream(botMsg, fullText, () => {
```

Explanation:

- Displays bot reply character by character.
- Simulates real-time typing.
- Improves UX.

After Completion

```
isStreaming = false;
```

```
input.disabled = false;
```

```
sendBtn.textContent = "➤";
```

```
input.focus();
```

- Re-enables input.
- Resets send button.

10. Error Handling

```
catch (err) {
```

Explanation:

- Handles server or network errors.
- Displays friendly error message.
- Resets chatbot state.

11. Streaming Text Function

```
function revealTextSimulatingStream(element, text, onFinish) {
```

This function displays chatbot text **character by character**.

Working Steps:

- Uses Promise
- Adds text in small chunks
- Converts \n into

- Scrolls chat automatically
- Stops if user requests stop

12. Keyboard Event (Enter Key Support)

```
document.addEventListener("DOMContentLoaded", () => {
```

Explanation:

- Ensures page is fully loaded.
- Adds Enter key support.

```
if (event.key === "Enter") {
```

- Press Enter → send message
- Press Enter while streaming → stop response

13. Overall Working Flow

1. User opens chatbot
2. Selects language
3. Types message
4. Message sent to backend
5. Bot response received
6. Typing animation shown
7. Response streamed to UI

14. Conclusion

This JavaScript file plays a **critical role** in chatbot functionality.

It ensures:

- Smooth interaction
- Multilingual support
- Real-time experience
- Error handling
- User control

Together with HTML and CSS, this script creates a **complete, interactive chatbot system**.