1. DOMContentLoaded — wait until the page is ready

```
document.addEventListener('DOMContentLoaded', () => {
  console.log('script loaded');
```

- This ensures your script runs **only after** the HTML page is fully loaded.
- The console.log helps confirm that your JS file loaded correctly.

---

2. Get DOM elements

```
const messages = document.getElementById('messages');
const input = document.getElementById('msg');
const sendBtn = document.getElementById('sendBtn');
const imageInput = document.getElementById('imageInput');
const voiceBtn = document.getElementById('voiceBtn');
```

- These correspond to your HTML elements:
  - messages → chat message area.
  - msg → user text input.
  - sendBtn → “Send” button.
  - imageInput → file input for images.
  - voiceBtn → mic button for voice input.

---

3. Setup variables for voice recognition

```
let recognition = null;
let isListening = false;
```

- recognition will hold the browser’s speech recognition engine.
- isListening tracks whether the mic is currently active.

---

4. Initialize speech recognition

```
if ('webkitSpeechRecognition' in window || 'SpeechRecognition' in window) {
  const SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;
  recognition = new SpeechRecognition();
```

- Detects if browser supports speech recognition.
- Chrome uses webkitSpeechRecognition, others might use SpeechRecognition.

---

5. Configure voice recognition settings

```
recognition.continuous = false;      // stop automatically after one phrase
recognition.interimResults = false;  // don't show live partial results
recognition.lang = 'en-US';          // set recognition language
```

---

6. Handle speech recognition events

```
recognition.onresult = (event) => {
  const transcript = event.results[0][0].transcript;
  input.value = transcript;
  updateVoiceButton(false);
};
```

- When voice input is recognized → the text is placed in the input box.

Error + End handling:

```
recognition.onerror = (event) => {
  addMessage('system', `Voice input error: ${event.error}`);
};
recognition.onend = () => updateVoiceButton(false);
```

- If mic fails, you show a system message.
- When mic stops, reset button appearance.

---

7. Hide the mic button if not supported

```
} else {
  if (voiceBtn) voiceBtn.style.display = 'none';
}
```

---

8. Voice button appearance logic

```
function updateVoiceButton(listening) {
  isListening = listening;
  voiceBtn.textContent = listening ? '🎤 Listening...' : '🎤 Voice';
  voiceBtn.style.background = listening ? '#ff4444' : '';
```

}

- When mic is ON, button turns red and text changes.

---

9. Toggle voice input

```
function toggleVoiceInput() {
 if (!recognition) {
  addMessage('system', 'Voice input not supported');
  return;
 }
 if (isListening) recognition.stop();
 else recognition.start();
}
```

- Pressing the mic button alternates between “start” and “stop listening.”

---

10. addMessage() — display chat messages

```
function addMessage(who, text, imageData = null) {
 const el = document.createElement('div');
 el.className = 'message ' + who;
```

Creates a new chat message block with a role (user, bot, or system).

📸 **If there’s an image:**

```
if (imageData) {
 const img = document.createElement('img');
 img.src = imageData;
 img.className = 'chat-image';
 bubble.appendChild(img);
}
```

- It displays the selected image (as a Data URL).

💬 **If there’s text:**

```
const formattedText = text
```

.replace(/**(.*?)**/g, '$1')

.replace(/*(.*?)*/g, '$1')

.replace(/\n/g, '
');

- Supports **bold** and *italic* formatting.
- Converts line breaks to
 for multi-line messages.

📦 **Append to chat window**

messages.appendChild(el);

messages.scrollTop = messages.scrollHeight;

- Adds the message to the end.
- Scrolls chat to bottom automatically.

---

11. handleImageUpload() — read + validate images

if (!file.type.startsWith('image/')) reject(new Error('Please select an image file'));

if (file.size > 5 * 1024 * 1024) reject(new Error('Image size should be less than 5MB'));

- Ensures valid image file and size ≤ 5MB.

Then:

reader.readAsDataURL(file);

- Reads image as Base64 so you can preview it in chat immediately.

---

12. analyzeImage() — send image to server

const formData = new FormData();

formData.append('image', imageFile);

formData.append('message', textMessage);

const res = await fetch('/api/analyze-image', { method: 'POST', body: formData });

- Sends the image (and optional text) to the backend API for AI analysis.

Then:

- Reads response text.
- Detects if it's actually an HTML login page (authentication check).
- If not, parses JSON and returns it.

---

13. sendMessage() — core logic for sending

This function handles **both text messages and image uploads.**

✉ If sending an image:

addMessage('system', 'Uploading and analyzing image...');

const imageData = await handleImageUpload(imageFile);

addMessage('user', msg || 'I uploaded this image for analysis', imageData);

const analysisResult = await analyzeImage(imageFile, msg);

addMessage('bot', analysisResult.response);

- Shows “Uploading…”
- Displays user’s image + message
- Sends to server for analysis
- Displays AI-generated analysis response

💬 **If only sending text:**

addMessage('user', msg);

const res = await fetch('/api/chat', { method: 'POST', headers: {...}, body: JSON.stringify({message: msg}) });

const data = await res.json();

addMessage('bot', data.response);

- Sends message to /api/chat endpoint.
- Displays bot’s text reply.

🧹 **Cleanup**

sendBtn.disabled = false;

input.value = '';

input.focus();

- Re-enables button and resets input box.

---

14. Event listeners

sendBtn.addEventListener('click', sendMessage);

```
voiceBtn.addEventListener('click', toggleVoiceInput);

input.addEventListener('keydown', (e) => {
  if (e.key === 'Enter' && !e.shiftKey) sendMessage();
});
```

- **Send button** click → send message.
- **Voice button** click → start/stop listening.
- **Enter key** → send message (Shift+Enter adds newline).

---

15. Image input "change" event

```
imageInput.addEventListener('change', (e) => {
  const file = e.target.files[0];
  if (!file.type.startsWith('image/') || file.size > 5 * 1024 * 1024) { addMessage('system', 'Invalid image'); return; }
  sendMessage();
});
```

- When user picks an image → it validates type/size → automatically sends it to the bot.

🧠 **How the flow works in real use:**

Action	**What Happens**
You type a message and press **Enter**	Message appears instantly → sent to /api/chat → bot's response shown
You click 🎤 **Voice**	Browser listens → fills input → you press Enter to send
You select an **image**	Image shows in chat → sent to /api/analyze-image → bot returns analysis
Network error	A system error bubble appears
Unsupported mic	Mic button hidden automatically