Lab: The Multimedia Dictionary

📝 Project Overview

The objective of this project is to develop a **Multimedia Dictionary** application that assists early learners in word-to-sound recognition. The application acts as a dynamic interface: it must store a collection of words and descriptive sentences within an HTML structure, retrieve that data when a user makes a selection, and utilize the **Web Speech API** to synthesize the text into audible speech. You will be responsible for ensuring the data flows correctly from the "hidden" attributes in your HTML to the visible display boxes and, finally, to the computer's audio output.

Have 5 words/5 sentences. Have it so when then select a word, it speaks the word. It displays a sentence, with a speak button next to the sentence. Have a speak button next to the word, so they can hear the word that way too. Buttons use an onclick event:

```
<button onclick="myFunction()">Click me</button>
```

📐 1. The Interface Layout

Your goal is to build a clean, functional interface following this logic:

1. **The Word Section:** A `<select>` dropdown menu with a **"Speak Word"** button placed directly next to it.
2. **The Sentence Section:** A `<textarea>` display box (set to `readonly`) with a **"Speak Sentence"** button placed directly next to it.

🛠️ 2. The `data-*` Attribute and the `dataset` Property

In this lab, you are using HTML as a "mini-database." To do this, we use custom attributes.

What is `data-*`?

The `data-` prefix allows you to store extra information on standard HTML elements without affecting the page layout. You will use `data-sentence="..."` inside your `<option>` tags.

Why does the name change in JavaScript?

When you access these attributes in JavaScript using the `dataset` property, the browser does two things for "cleaner" code:

Strips the prefix: It removes the `data-` part because it is redundant once you are already inside the `dataset` object.

1. **Maps to a Property:** It turns the rest of the name into a standard JavaScript property.
    - **HTML:** `data-sentence` → **JS:** `element.dataset.sentence`

⚙️ 3. Implementation Logic

A. The "Drill-Down" Technique

To get the "hidden" sentence, you must look at the specific "drawer" (option) currently selected:

1. Use `selectedIndex` to find the current position in the `<select>` menu.
2. Use that number to grab the specific item from the `options` collection.
3. Access the custom `dataset.sentence` from that specific item.

B. The Gatekeeper (Validation)

Your app must be "smart" enough to know when a user hasn't made a choice yet.

- **The Logic:** The first item in your menu is an instruction ("--Select--"), located at **Index 0**.
- **The Rule:** Use an `if` statement to check the `selectedIndex`. If the index is **greater than 0**, the gate opens and allows the program to run.

🔊 4. The Multimedia Engine (Web Speech API)

Use the following function as your "engine." It is a **reusable tool** that accepts a string of text via a parameter.

JavaScript

```
function speak(textToSay) {
    const message = new SpeechSynthesisUtterance(textToSay);
    message.pitch = 1.2;
    message.rate = 1.0;
    window.speechSynthesis.speak(message);
}
```

---

5. Study Example: The Weather Informer

Use this pattern to understand how to move data from a dropdown to a display. You must adapt this logic for your dictionary.

HTML:

HTML
```
<select id="citySelect" onchange="syncWeather()">
  <option value="">-- Choose a City --</option>
  <option value="Miami" data-temp="82°F and Sunny">Miami</option>
</select>
<input type="text" id="weatherDisplay" readonly>
```

JavaScript:

JavaScript
```
function syncWeather() {
    const menu = document.getElementById("citySelect");
    const display = document.getElementById("weatherDisplay");

    if (menu.selectedIndex > 0) {
        const selectedOption = menu.options[menu.selectedIndex];
        // Extract and Display
        display.value = selectedOption.dataset.temp;
    } else {
        display.value = "";
    }
}
```

🛠️ 6. Troubleshooting Guide

Symptom	Potential Cause	How to Fix
Sentence shows "undefined"	Accessing the wrong element.	Ensure you are looking at `options[index].dataset` and NOT the `select` element itself.
Buttons don't talk	Passing the wrong data.	Ensure you are passing the `.value` (the string) instead of the whole element object.
Gatekeeper doesn't work	Wrong index check.	Remember that collections are "Zero-Indexed." The first item is always 0.

🎨 7. Visual Layout Guide

Section	Element 1	Element 2
Word Selection	Dropdown Menu	[Speak Word Button]
Sentence Display	Read-Only Textarea	[Speak Sentence Button]
