

**Introduction:**

A text-to-voice converter website represents a fascinating intersection of technology and user experience, offering a dynamic way for users to engage with content. In this type of web application, users can effortlessly transform text into lifelike spoken words through the power of technology. This innovative solution is made possible by harmoniously blending three fundamental web development components: HTML, CSS, and JavaScript.

1. \*\*HTML (index.html)\*\*: At the core of this digital marvel lies the HTML structure, carefully designed to cater to user interactions. Within the HTML, a canvas is set for users to input their desired text, creating a seamless experience. An intuitive button acts as the catalyst for this transformation, providing users with a straightforward and user-friendly means of initiating the text-to-voice conversion process. It's here that the foundations of user interaction are laid, setting the stage for the subsequent layers of functionality.

2. \*\*CSS (styles.css)\*\*: Aesthetics play a pivotal role in enhancing the user experience. CSS, the style sheet of the website, contributes significantly to the visual appeal and overall usability. With CSS, designers can fine-tune the fonts, colors, and layouts, creating an interface that is both visually pleasing and responsive. Through skillful styling, the website becomes not just a tool but an engaging and enjoyable platform for users to interact with.

3. \*\*JavaScript (script.js)\*\*: JavaScript breathes life into the static HTML and stylish CSS, providing the website with dynamic functionality. At its core, JavaScript serves as the bridge between the user and the Web Speech API, the engine responsible for the magical transformation of text into speech. This dynamic language enables the website to respond to user actions, such as button clicks, and orchestrate the entire text-to-speech conversion process. It empowers users to choose their preferred voice, set the language of communication, and effortlessly trigger the synthesis of speech.

By weaving these three technological threads together, web developers create an immersive experience where users can harness the potential of modern technology. The result is an interactive text-to-voice converter website that empowers users to customize their interactions. With a simple click, users can transform written words into auditory experiences, tailor the interface to their liking, and unlock a world of possibilities in communication and accessibility. This digital symphony of HTML, CSS, and JavaScript encapsulates the fusion of design, functionality, and user empowerment that defines the essence of web development in the modern era.

**INDEX.HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="styles.css">

    <title>Text-to-Voice Converter</title>

</head>

<body>

    <div class="container">

        <h1>Text-to-Voice Converter</h1>

        <div class="text-input">

            <label for="text">Enter text to convert to speech:</label>

            <textarea id="text" rows="4" cols="50"></textarea>

            <button id="speakButton">Speak</button>

        </div>

    </div>

    <script src="script.js"></script>

</body>

</html>

**SCRIPT.JS:**

const speakButton = document.getElementById("speakButton");

const textInput = document.getElementById("text");

// Initialize the SpeechSynthesis API

const synth = window.speechSynthesis;

speakButton.addEventListener("click", () => {

    const textToSpeak = textInput.value;

    const utterance = new SpeechSynthesisUtterance(textToSpeak);

    // Configure the voice and language

    const voices = synth.getVoices();

    utterance.voice = voices[0]; // You can choose a specific voice from the voices array

    utterance.lang = "en-US"; // Set the language

    // Speak the text

    synth.speak(utterance);

});

**STYLES.CSS:**

body {

    font-family: Arial, sans-serif;

    background-color: #f0f0f0;

    margin: 0;

    padding: 0;

}

.container {

    max-width: 400px;

    margin: 0 auto;

    padding: 20px;

    background-color: #fff;

    border-radius: 5px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

}

h1 {

    text-align: center;

    margin-bottom: 20px;

    color: #333;

}

.text-input {

    margin-bottom: 20px;

    text-align: center;

}

label {

    display: block;

    margin-bottom: 5px;

    font-weight: bold;

    color: #333;

}

textarea {

    width: 100%;

    padding: 10px;

    border: 1px solid #ccc;

    border-radius: 5px;

    font-size: 16px;

}

button {

    margin-top: 10px;

    padding: 10px 20px;

    font-size: 16px;

    background-color: #007BFF;

    color: #fff;

    border: none;

    border-radius: 5px;

    cursor: pointer;

}

button:hover {

    background-color: #0056b3;

}

**EXPLANATION**

**INDEX.HTML:**

1. `<!DOCTYPE html>`: This declaration specifies the document type and version of HTML being used (HTML5 in this case).

2. `<html lang="en">`: The opening tag for the HTML document. The `lang` attribute indicates that the document is in English.

3. `<head>`: The head section of the HTML document, which contains metadata and links to external resources.

4. `<meta charset="UTF-8">`: This meta tag specifies the character encoding used for the document (UTF-8, which supports a wide range of characters).

5. `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: This meta tag sets the viewport properties, ensuring that the website is responsive to different device widths.

6. `<link rel="stylesheet" href="styles.css">`: This line links an external CSS stylesheet named "styles.css" to apply styling to the HTML content.

7. `<title>Text-to-Voice Converter</title>`: This sets the title of the web page, which appears in the browser's title bar or tab.

8. `<body>`: The opening tag for the body of the HTML document, where the visible content is placed.

9. `<div class="container">`: A `div` element with the class "container," which is often used to structure content and apply styling.

10. `<h1>Text-to-Voice Converter</h1>`: A level-one heading that serves as the main title of the page.

11. `<div class="text-input">`: A `div` element with the class "text-input," used for structuring the input elements.

12. `<label for="text">Enter text to convert to speech:</label>`: A label for an input field with the id "text." It provides a description for the input field.

13. `<textarea id="text" rows="4" cols="50"></textarea>`: A multi-line text input field with the id "text," specifying the number of rows and columns.

14. `<button id="speakButton">Speak</button>`: A button element with the id "speakButton" used to trigger the text-to-voice conversion.

15. `</div>`: Closes the "text-input" div.

16. `</div>`: Closes the "container" div.

17. `<script src="script.js"></script>`: Links an external JavaScript file named "script.js" to provide functionality to the webpage.

18. `</body>`: Closes the body of the HTML document.

19. `</html>`: Closes the HTML document.

**STYLES.CSS:**

Certainly, let me explain the CSS code more clearly:

1. \*\*body\*\*: This CSS selector applies styles to the entire document's body. It means that any styling specified within the `body` block will affect the appearance of elements throughout the entire web page. For example, you can set the default font family, background color, and margins for all content in the body of the page.

2. \*\*container\*\*: The `container` selector defines styles for a specific HTML element with the class "container." In HTML, this class is often used to wrap a section of content, and it's a common practice to style it for layout purposes. For example, you can set the maximum width, margin, padding, background color, and border radius for elements with the "container" class.

3. \*\*h1\*\*: This selector targets level-one (`<h1>`) headings within the page. It allows you to style the main titles of your web page. For instance, you can set the text alignment, margin, and text color for these headings.

4. \*\*text-input\*\*: The `text-input` selector specifies styles for a specific section of content within your web page that has the class "text-input." This class might be used to structure and style a form or input area. You can define properties like margin, text alignment, and background color for elements with this class.

5. \*\*label\*\*: The `label` selector is used to style label elements within your HTML document. Labels are typically associated with form input fields to provide descriptions or instructions. You can style them by defining properties such as font weight, margin, and text color.

6. \*\*textarea\*\*: This selector is used to style `textarea` input fields. Textareas are often used for multi-line text input, and you can apply styles like width, padding, border, and font size to control their appearance.

7. \*\*button\*\*: The `button` selector targets button elements in your HTML. You can style buttons by setting properties like margin, padding, background color, font size, and border radius. This allows you to control the appearance of buttons used for actions on your web page.

8. \*\*button:hover\*\*: This selector is used for styling button elements when they are hovered over with the mouse cursor. It allows you to define how the button should appear when a user hovers their mouse pointer over it. For example, you can change the background color or add an animation effect to provide visual feedback to the user when interacting with the button.

These CSS selectors and styles help control the layout and appearance of different elements within your web page, ensuring a visually appealing and consistent design.

**SCRIPT.JS:**

1. `const speakButton = document.getElementById("speakButton");`: Defines a JavaScript variable that holds a reference to the HTML element with the id "speakButton."

2. `const textInput = document.getElementById("text");`: Defines a JavaScript variable that holds a reference to the HTML element with the id "text."

3. `const synth = window.speechSynthesis;`: Initializes the SpeechSynthesis API and stores it in the "synth" variable.

4. `speakButton.addEventListener("click", () => { ... });`: Adds a click event listener to the "Speak" button. When clicked, the code within the curly braces is executed.

5. `const textToSpeak = textInput.value;`: Retrieves the text entered in the textarea and stores it in the "textToSpeak" variable.

6. `const utterance = new SpeechSynthesisUtterance(textToSpeak);`: Creates a new SpeechSynthesisUtterance object with the text to be spoken.

7. `const voices = synth.getVoices();`: Retrieves the available voices from the SpeechSynthesis API and stores them in the "voices" variable.

8. `utterance.voice = voices[0];`: Sets the voice for the utterance to the first voice in the "voices" array. This is where you can choose a specific voice for speech synthesis.

9. `utterance.lang = "en-US";`: Sets the language for the utterance to "en-US" (U.S. English).

10. `synth.speak(utterance);`: Initiates the speech synthesis with the configured utterance, causing the browser to speak the entered text using the selected voice.

**Output:**





