Laboratorio práctico: Crear herramienta de análisis de texto para pruebas de velocidad utilizando manipulaciones de cadenas

Tiempo estimado necesario: 30 minutos

Lo que aprenderás

En este laboratorio, participarás en métodos de cadenas para crear una funcionalidad de prueba de escritura, mejorando tu velocidad y precisión al escribir y obteniendo información sobre la interacción de JavaScript con los elementos HTML. Al practicar frases comunes y recibir retroalimentación inmediata sobre métricas como palabras escritas y palabras por minuto (WPM), los usuarios pueden seguir su progreso e identificar áreas de mejora.

Objetivo de aprendizaje

Después de completar este laboratorio, podrás:

- **Implementar una interfaz de prueba de escritura:** Desarrollar una interfaz web interactiva utilizando HTML y JavaScript. Crear áreas de texto separadas para mostrar texto predefinido y capturar la entrada del usuario. Configurar un botón funcional para iniciar y concluir la prueba de escritura sin problemas.
- **Gestionar la ejecución de la prueba y la visualización de resultados:** Establecer funciones de JavaScript para controlar el flujo de la prueba. Inicializar el entorno de prueba llenando el texto predefinido que los usuarios deben escribir. Capturar la entrada del usuario, calcular métricas clave de escritura (palabras escritas, tiempo transcurrido) y computar la velocidad de escritura en palabras por minuto (WPM).
- **Proporcionar retroalimentación y análisis inmediatos:** Habilitar la visualización en tiempo real de los resultados esenciales de la prueba al finalizarla. Mostrar estadísticas de la prueba como el número de palabras escritas, el tiempo tomado y WPM para proporcionar a los usuarios retroalimentación inmediata sobre su rendimiento al escribir.
- **Facilitar el aprendizaje iterativo y la experiencia del usuario:** Ofrecer una plataforma amigable para practicar habilidades de escritura, fomentando la participación e interacción del usuario. Proporcionar una plataforma que permita a los usuarios seguir su progreso, aprender de la retroalimentación inmediata y mejorar su velocidad y precisión al escribir.

Requisitos previos

- Conocimiento básico de HTML y comandos de Git.
- Comprensión básica de funciones de JavaScript y métodos de manipulación de cadenas.
- Navegador web con consola (Chrome DevTools, Firefox Console, etc.).

Paso 1: Configuración del entorno

1. Primero, necesitas clonar tu repositorio principal en el **Entorno de Skills Network**. Sigue los pasos dados:

- Haz clic en la terminal en el panel de la ventana superior derecha y luego selecciona **Nueva Terminal**.

- Realiza el comando `git clone` escribiendo el comando dado en la terminal.

```
git clone <github-repository-url>
```

Nota: Coloca tu propio enlace de repositorio de GitHub en lugar de `<github-repository-url>` y debería verse como se indica a continuación:

  - El paso anterior clonará la carpeta de tu repositorio de GitHub en la carpeta del proyecto en el explorador. También verás múltiples carpetas dentro de la carpeta clonada.
  - Ahora necesitas navegar dentro de la carpeta clonada. Para esto, escribe el comando dado en la terminal:

```
cd <repository-folder-name>
```

Nota: Escribe el nombre de tu carpeta clonada en lugar de `<repository-folder-name>`. Realiza `git clone` si has cerrado sesión en el **Entorno de Skills Network** y no puedes ver ningún archivo o carpeta después de iniciar sesión.

2. Ahora, selecciona la carpeta **cloned Folder Name**, haz clic derecho sobre ella y elige **Nueva Carpeta**. Ingresa el nombre de la carpeta como **speedAnalysis**. Esto creará la carpeta para ti. Luego, selecciona la carpeta **speedAnalysis**, haz clic derecho y elige **Nuevo Archivo**. Ingresa el nombre del archivo como **speed_analysis.html** y haz clic en OK. Esto creará tu archivo HTML.
3. Ahora selecciona nuevamente la carpeta **speedAnalysis**, haz clic derecho y selecciona **Nuevo Archivo**. Ingresa el nombre del archivo como **speed_analysis.js** y haz clic en OK. Esto creará tu archivo JavaScript.
4. Crea una estructura básica de plantilla de un archivo HTML agregando el contenido proporcionado a continuación.

```
<!DOCTYPE html>
<!DOCTYPE html>
<html>
<head>
  <title>Análisis de Prueba de Velocidad</title>
</head>
<body>
  <h1>Análisis de Escritura Rápida</h1>
    <label for="inputText">Escribe el siguiente texto:</label><br>
    <textarea id="inputText"  cols="50" rows="5" readonly placeholder="Prepárate para escribir..."></textarea><br>
    <label for="userInput">Tu escritura:</label><br>
    <textarea id="userInput"  cols="50" rows="5" placeholder="Comienza a escribir cuando estés listo"></textarea><br>
    <button onclick="startTest()">Iniciar Prueba</button>
    <button onclick="endTest()">Finalizar Prueba</button>
    <div id="output"></div>
    <script src="./speed_analysis.js"></script>
</body>
</html>
```

  - Dentro del `<body>`, hay un encabezado `<h1>` que muestra “Análisis de Escritura Rápida” como el título de la página. Incluye dos elementos `<textarea>`:
    - El primer `<textarea>` con el ID inputText está marcado como de solo lectura y sirve como área de visualización para el texto que el usuario necesita escribir.
    - El segundo `<textarea>` con el ID userInput es donde los usuarios escribirán el texto mostrado.

- Dos elementos `<button>` con un evento on click configurado para las funciones **startTest() y endTest()** inician la prueba de escritura.
- Por último, hay dos elementos `<div>` con el ID output que mostrarán los resultados de la prueba.
- Para incluir un archivo JavaScript en **speed_analysis.html**, se utiliza la etiqueta de script en el archivo HTML antes de la etiqueta `</body>`.

Nota: Cuando hayas pegado el código, guarda tu archivo.

Paso 2: Definiendo variables y funciones

1. En **speed_analysis.js**, crea la variable de texto de prueba con cualquier frase utilizada para escribir. Luego inicializa dos variables llamadas startTime y endTime.

```
let testText = "El rápido zorro marrón salta sobre el perro perezoso.";
let startTime, endTime;
```

2. Crea la función **startTest** para comenzar a probar la velocidad de escritura. Para esto, incluye el código dado en el archivo **speed_analysis.js**.

```
function startTest() {
// Establecer el texto de prueba
document.getElementById("inputText").value = testText;
// Reiniciar la entrada y salida del usuario
let userInput = document.getElementById("userInput");
userInput.value = "";
userInput.readOnly = false;
userInput.focus();
document.getElementById("output").innerHTML = "";
// Iniciar el temporizador
startTime = new Date().getTime();
}
```

- Accede al elemento del área de texto en el documento HTML usando el ID **inputText**. Esto establece el valor del área de texto inputText a una variable o valor llamado testText.
- Además, la función limpia el contenido existente dentro de un elemento HTML específico con el ID output. Estos elementos sirven para mostrar los resultados de la prueba y el temporizador durante la evaluación. Reiniciar su innerHTML a una cadena vacía prepara el espacio para mostrar nuevos resultados y asegura que el temporizador comience desde cero cuando comienza la prueba.
- Además, la función **startTest**() maneja la inicialización del seguimiento del tiempo. Captura la marca de tiempo actual usando new Date().getTime() y la almacena como la variable startTime, señalando el comienzo de la prueba.

3. Crea una función endTest para finalizar la prueba y mostrar los resultados cuando el usuario termine el texto. Para esto, incluye el código dado en el archivo **speed_analysis.js** después del código de la función anterior.

```
 function endTest() {
            endTime = new Date().getTime();
            // Deshabilitar la entrada del usuario
            document.getElementById("userInput").readOnly = true;
            // Calcular el tiempo transcurrido y las palabras por minuto (WPM)
            var timeElapsed = (endTime - startTime) / 1000; // en segundos
            var userTypedText = document.getElementById("userInput").value;
            // Dividir el texto usando regex para contar las palabras correctamente
            var typedWords = userTypedText.split(/\s+/).filter(function (word) {
                return word !== "";
            }).length;
            var wpm = 0; // Valor por defecto
            if (timeElapsed !== 0 && !isNaN(typedWords)) {
                wpm = Math.round((typedWords / timeElapsed) * 60);
            }
            // Mostrar los resultados
            var outputDiv = document.getElementById("output");
            outputDiv.innerHTML = "<h2>Resultados de la Prueba de Escritura:</h2>" +
                "<p>Palabras Escribidas: " + typedWords + "</p>" +
                "<p>Tiempo Transcurrido: " + timeElapsed.toFixed(2) + " segundos</p>" +
                "<p>Palabras Por Minuto (WPM): " + wpm + "</p>";
            }
```

A continuación se desglosa el código anterior:
- `endTime = new Date().getTime();`: Esta línea captura el tiempo actual cuando termina la prueba.
- `document.getElementById("userInput").readOnly = true;`: Deshabilita la entrada en el área de texto userInput, evitando que el usuario escriba después de que la prueba haya terminado.

- Cálculo del tiempo:
  `var timeElapsed = (endTime - startTime) / 1000;`
  - endTime captura la marca de tiempo actual usando new Date().getTime() para marcar la conclusión de la prueba.
  - startTime contiene presumiblemente el tiempo de inicio cuando comenzó la prueba.
  - timeElapsed se calcula restando startTime de endTime para obtener la duración en milisegundos y luego se convierte a segundos dividiendo por 1000.
- Cálculo de palabras por minuto (WPM):
  `var userTypedText = document.getElementById("userInput").value; var typedWords = userTypedText.split(/\s+/).filter(function (word) { return word !== "";}).length;`
  - La función recupera el texto escrito por el usuario desde el área de texto userInput usando document.getElementById("userInput").value.
  - Luego calcula el número de palabras escritas dividiendo el texto de entrada usando una expresión regular para considerar palabras separadas por espacios, tabulaciones o saltos de línea. Filtrar asegura contar palabras válidas, excluyendo cadenas vacías.
- Palabras Por Minuto:
  `if (timeElapsed !== 0 && !isNaN(typedWords)) { wpm = Math.round((typedWords / timeElapsed) * 60);}`
  - Calcula las palabras por minuto. Verifica si timeElapsed no es cero y typedWords es un número válido.
  - Si es así, calcula el WPM dividiendo el número de palabras escritas por el tiempo transcurrido (en minutos) y luego multiplica por 60 para obtener palabras por minuto. Math.round() redondea el valor al número entero más cercano.
- Mostrando resultados de la prueba:
  - La función actualiza el innerHTML del elemento de salida para presentar los resultados de la prueba. Estructura el contenido con un encabezado `<h2>` que indica "Resultados de la Prueba de Escritura" y proporciona detalles como el número de palabras escritas, el tiempo transcurrido y las palabras por minuto (WPM) logradas durante la prueba.

Paso 3: Verifica la salida

1. Para ver tu página HTML, haz clic derecho en el archivo **speed_analysis.html** después de seleccionarlo, y luego selecciona "Abrir con Live Server".

2. El servidor debería iniciarse en el puerto 5500, indicado por una notificación en la parte inferior derecha.

3. Haz clic en el botón de Skills Network a la izquierda (consulta el número 1). Se abrirá la “Caja de herramientas de Skills Network”. Luego haz clic en Lanzar Aplicación (consulta el número 2). Desde allí, ingresa el puerto 5500 en el número 3 y haz clic en este botón .

4. Se abrirá tu navegador predeterminado, donde primero verás el nombre de tu carpeta clonada. Haz clic en esa carpeta, y dentro de ella, entre otras carpetas, encontrarás el nombre de la carpeta **speedAnalysis**. Haz clic en la carpeta **speedAnalysis**, y luego selecciona el archivo **speed_analysis.html**, como se muestra a continuación.

5. Se abrirá la página principal y verás la salida como se muestra a continuación.

6. Haz clic en el botón de inicio y se te mostrará el texto que necesitas escribir en el segundo cuadro de entrada.

7. Después de completar la escritura, haz clic en el botón Finalizar Prueba y se mostrará la respuesta como se muestra a continuación:

Nota: Después de pegar el código, guarda tu archivo. Puedes usar cualquier método de salida para esto. Si editas tu código, simplemente actualiza tu navegador que está corriendo a través del número de puerto 5500. De esta manera, no es necesario lanzar la aplicación una y otra vez.

Paso 4: Realizar comandos de Git

1. Realiza `git add` para agregar los últimos archivos y carpetas escribiendo el comando dado en el terminal en el entorno de git.

```
git add —-a
```

Asegúrate de que el terminal tenga la ruta como sigue:

2. Luego, realiza `git commit` en el terminal. Al realizar `git commit`, el terminal puede mostrar un mensaje para configurar tu `git config —-global` para user.name y user.email. Si es así, entonces también necesitas ejecutar el comando `git config` para `user.name` y `user.email` como se indica.

```
git config —-global user.email "you@example.com"
```

```
git config —-global user.name "Your Name"
```

Nota: Reemplaza los datos entre comillas con tus propios detalles.

Luego, realiza el comando de commit como se indica:

```
git commit -m "message"
```

3. Luego, realiza `git push` simplemente escribiendo el comando dado en el terminal.

```
git push origin
```

- Después del comando de push, el sistema te pedirá que ingreses tu nombre de usuario y contraseña. Ingresa el nombre de usuario de tu cuenta de GitHub y la contraseña que creaste en el primer laboratorio. Después de ingresar las credenciales, todas tus últimas carpetas y archivos se enviarán a tu repositorio de GitHub.

Tarea de Práctica

- También puedes calcular la longitud total del texto que has capturado en la variable `userTypedText` que el usuario ha ingresado. Para esto, necesitas utilizar:
  - la propiedad length para verificar la longitud de todo el texto usando la variable **userTypedText**.
  - Incluye también el resultado en la salida.
- La salida se verá como a continuación:

- Realiza los comandos `git add`, `git commit` y `git push` para subir tu tarea más reciente y mantener tu código actualizado en el repositorio de GitHub.

Resumen

1. **Configuración de la interfaz y establecimiento de funcionalidad:** Creaste una interfaz de prueba de escritura basada en la web utilizando HTML y JavaScript. La estructura HTML formó la base para la visualización de texto (inputText), la entrada del usuario (userInput) y un botón dinámico (Iniciar Prueba).

2. **Ejecución de la prueba y análisis de resultados:** Al hacer clic en el botón “Iniciar Prueba”, se pidió a los usuarios que escribieran un fragmento de texto predeterminado en el área de entrada designada. Al finalizar la prueba, el código calculó y mostró rápidamente métricas esenciales de escritura, como palabras escritas, tiempo transcurrido y palabras por minuto (WPM), proporcionando información inmediata sobre la competencia de escritura del usuario.

3. **Experiencia del usuario y aprendizaje iterativo:** Creaste una plataforma práctica para practicar habilidades de escritura en un entorno controlado. También proporcionaste retroalimentación inmediata sobre la velocidad y precisión de escritura, permitiendo a los usuarios evaluar su rendimiento, facilitando el aprendizaje iterativo y mejorando la competencia en escritura.

Author(s)

Richa Arora