Hands-on Lab: Create Text Analysis Tool for Speed Testing Using String Manipulations

Estimated time needed: 30 minutes

What you will learn

In this lab, you will engage in string methods to create type testing functionality, enhancing their typing speed and accuracy and gaining insights into JavaScript's interaction with HTML elements. By practicing common phrases and receiving immediate feedback on metrics like words typed and words per minute (WPM), users can track their progress and identify areas for improvement.

Learning objective

After completing this lab, you will be able to:

- **Implementing a typing test interface:** Develop an interactive web interface using HTML and JavaScript. Create separate text areas for displaying predefined text and capturing user input. Set up a functional button to initiate and conclude the typing test seamlessly.

- **Managing Ttst execution and result display:** Establish JavaScript functions to control the test flow. Initialize the test environment by populating the predefined text for users to type. Capture user input, calculate crucial typing metrics (words typed, time elapsed), and compute the typing speed of words per minute (WPM).

- **Providing immediate feedback and analysis:** Enable real-time display of essential test results upon test completion. Showcase test statistics such as the number of words typed, time taken, and WPM to provide users with immediate feedback on their typing performance.

- **Facilitating iterative learning and user experience:** Offer a user-friendly platform for practicing typing skills, encouraging user engagement and interaction. Provide a platform allowing users to track their progress, learn from immediate feedback, and improve their typing speed and accuracy.

Prerequisites

- Basic Knowledge of HTML and Git commands.

- Basic understanding of JavaScript functions and string manipulation methods.

- Web browser with a console (Chrome DevTools, Firefox Console, and so on).

Step 1: Setting up the environment

1. First, you need to clone your main repository in the **Skills Network Environment**. Follow the given steps:

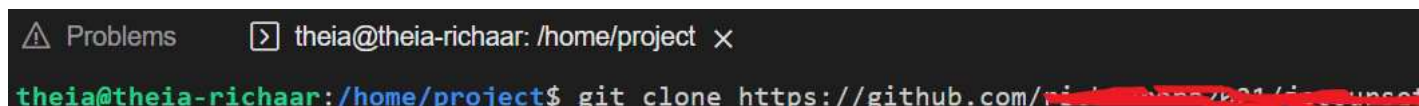- Click on the terminal in the top-right window pane and then select **New Terminal**.
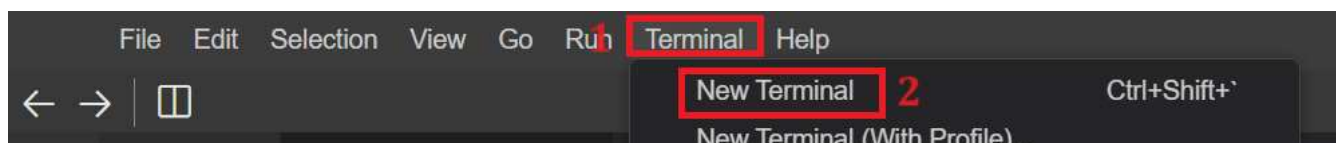
- Perform `git clone` command by writing given command in the terminal.

```
1. 1
1. git clone <github-repository-url>
```
Copied!

> **Note:** Put your own GitHub repository link instead of `<github-repository-url>` and it should look like as given below:

- Above step will clone folder for your GitHub repository under project folder in explorer. You will also see multiple folders inside cloned folder.

- Now you need to navigate inside the cloned folder. For this write given command in the terminal:

```
1. 1
1. cd <repository-folder-name>
```
Copied!

Note: Write your cloned folder name instead of `<repository-folder-name>`. Perform `git clone` if you have logged out of **Skills Network Environment** and you cannot see any files or folder after you logged in.

2. Now, select the **cloned Folder Name** folder, right-click on it, and choose **New Folder**. Enter the folder name as **speedAnalysis**. This will create the folder for you. Then, select the **speedAnalysis** folder, right-click, and choose **New File**. Enter the file name as **speed_analysis.html** and click OK. This will create your HTML file.

3. Now select **speedAnalysis** folder again, right click and select **New File**. Enter the file named **speed_analysis.js** and click OK. It will create your JavaScript file.

4. Create a basic template structure of an HTML file by adding the content provided below.

```
1.  1
2.  2
3.  3
4.  4
5.  5
6.  6
7.  7
8.  8
9.  9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
1.  <!DOCTYPE html>
2.  <!DOCTYPE html>
3.  <html>
4.  <head>
5.    <title>Speed Test Analysis</title>
6.  </head>
7.  <body>
8.    <h1>Speed Typing Analysis</h1>
9.
10.     <label for="inputText">Type the following text:</label><br>
11.     <textarea id="inputText"  cols="50" rows="5" readonly placeholder="Get ready to type..."></textarea><br>
12.
13.     <label for="userInput">Your typing:</label><br>
14.     <textarea id="userInput"  cols="50" rows="5" placeholder="Start typing when ready"></textarea><br>
15.     <button onclick="startTest()">Start Test</button>
16.         <button onclick="endTest()" id="btn">End Test</button>
17.     <div id="output"></div>
18.     <script src="./speed_analysis.js"></script>
19. </body>
20. </html>
```
Copied!

- Inside the `<body>`, there's an `<h1>` heading that displays "Speed Typing Test" as the title of the page. It includes two `<textarea>` elements:

  - The first `<textarea>` with the ID inputText is marked as read only and serves as a display area for the text the user needs to type.
  - The second `<textarea>` with the ID userInput is where users will type the displayed text.

- Two `<button>` element with an on click event set to **startTest() and endTest()** functions initiates the typing test.

- Lastly, there are two `<div>` elements with IDs output which will display test results.

- To include a JavaScript file in **speed_analysis.html**, the script tag is used in HTML file above `</body>` tag.

Note: When you have pasted the code, save your file.

Step 2: Defining variables and functions

1. In **speed_analysis.js**, create the test text variable with any sentence used for typing. Then initialize two variables named startTime and endTime.

```
1.  1
2.  2
1.  let testText = "The quick brown fox jumps over the lazy dog.";
2.  let startTime, endTime;
```
Copied!

2. Create **startTest** functions to start testing the typing speed. For this, include the given code in the **speed_analysis.js** file.

```
1.  1
```

```
2.  2
3.  3
4.  4
5.  5
6.  6
7.  7
8.  8
9.  9
10. 10
11. 11
12. 12
13. 13
1. function startTest() {
2. // Set the test text
3. document.getElementById("inputText").value = testText;
4.
5. // Reset results and timer
6. document.getElementById("output").innerHTML = "";
7. startTime = new Date().getTime();
8.
9. // Change button text and functionality
10. var button = document.getElementById("btn");
11. button.innerHTML = "End Test";
12. button.onclick = endTest;
13.          }
```

`Copied!`

- It accesses the text area element in the HTML document using the ID **inputText**. This sets the value of the inputText text area to a variable or value named testText.

- Additionally, the function clears existing content within two specific HTML elements with ID output and timer. These elements serve the purpose of displaying test results and the timer during the assessment. Resetting their innerHTML to an empty string prepares the space to showcase fresh results and ensures the timer starts from zero when the test commences.

- Moreover, the **startTest**() function handles the initialization of time tracking. It captures the current timestamp using the new Date().getTime() and stores it as the startTime variable, signaling the beginning of the test.

- The script utilizes a variable linked to an HTML element identified by the ID button. Upon the page load, the button displays the Start Test text. However, once the user initiates the test by clicking the button, it dynamically switches to display the End Test. Simultaneously, this action invokes the execution of the endTest() function, seamlessly enabling the transition between initiating and concluding the test through a single button click.

3. Create an endTest function to end the test and display results when the user finishes the text. For this, include the given code in the **speed_analysis.js** file after the above function code.

```
1.  1
2.  2
3.  3
4.  4
5.  5
6.  6
7.  7
8.  8
9.  9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31
32. 32
33. 33
1.  function endTest() {
2.          endTime = new Date().getTime();
3.
4.          // Disable user input
5.          document.getElementById("userInput").readOnly = true;
6.
7.          // Calculate time elapsed and words per minute (WPM)
8.          var timeElapsed = (endTime - startTime) / 1000; // in seconds
9.          var userTypedText = document.getElementById("userInput").value;
10.
11.          // Split the text using regex to count words correctly
12.          var typedWords = userTypedText.split(/\s+/).filter(function (word) {
13.              return word !== "";
14.          }).length;
15.
16.          var wpm = 0; // Default value
17.
```

```
18.            if (timeElapsed !== 0 && !isNaN(typedWords)) {
19.                wpm = Math.round((typedWords / timeElapsed) * 60);
20.            }
21.
22.            // Display the results
23.            var outputDiv = document.getElementById("output");
24.            outputDiv.innerHTML = "<h2>Typing Test Results:</h2>" +
25.                "<p>Words Typed: " + typedWords + "</p>" +
26.                "<p>Time Elapsed: " + timeElapsed.toFixed(2) + " seconds</p>" +
27.                "<p>Words Per Minute (WPM): " + wpm + "</p>";
28.
29.            // Reset the button
30.            var button = document.getElementById("btn");
31.            button.innerHTML = "Start Test";
32.            button.onclick = startTest;
33.        }
```

Copied!

Here is the breakdown of above code:

- `endTime = new Date().getTime();`: This line captures the current time when the test ends.
- `document.getElementById("userInput").readOnly = true;`: Disables further input in the userInput textarea, preventing the user from typing after the test is complete.

- Time calculation:
  `var timeElapsed = (endTime - startTime) / 1000;`

  - endTime captures the current timestamp using new Date().getTime() to mark the test's conclusion.
  - startTime holds the presumably set the starting time when the test began.
  - timeElapsed is calculated by subtracting startTime from endTime to get the time duration in milliseconds and then converts it to seconds by dividing by 1000.

- Calculating words per minute (WPM):
  `var userTypedText = document.getElementById("userInput").value; var typedWords = userTypedText.split(/\s+/).filter(function (word) { return word !== "";}).length;`

  - The function retrieves the user's typed text from the userInput text area using document.getElementById("userInput").value.
  - It then computes the number of words typed by splitting the input text using a regular expression to consider words separated by spaces, tabs, or newlines. Filtering ensures counting valid words, excluding empty strings.

- Word Per Minute:
  `if (timeElapsed !== 0 && !isNaN(typedWords)) { wpm = Math.round((typedWords / timeElapsed) * 60);}`

  - Calculates the words per minute. It checks if timeElapsed is not zero and typedWords is a valid number.
  - If so, it computes the WPM by dividing the number of typed words by the time elapsed (in minutes) and then multiplies by 60 to get words per minute. Math.round() rounds the value to the nearest whole number.

- Displaying test results:

  - The function updates the output element's innerHTML to present the test results. It structures the content with an `<h2>` heading indicating "Typing Test Results" and provides details such as the number of words typed, time elapsed, and words per minute (WPM) achieved during the test.

- Resetting button and functionality:

  - Resets the button text to "Start Test" and assigns the startTest() function to the click event of the button, allowing the user to start a new typing test.

Step 3: Check the output

1. To view your HTML page, right-click the **speed_analysis.html** file after selecting this file, then select "Open with Live Server".

2. The server should start on port 5500, indicated by a notification on the bottom-right side.

3. Click the Skills Network button on the left (refer to number 1). It will open the "Skills Network Toolbox". Then click Launch Application (refer to number 2). From there, you enter port 5500 at number 3 and click this button .
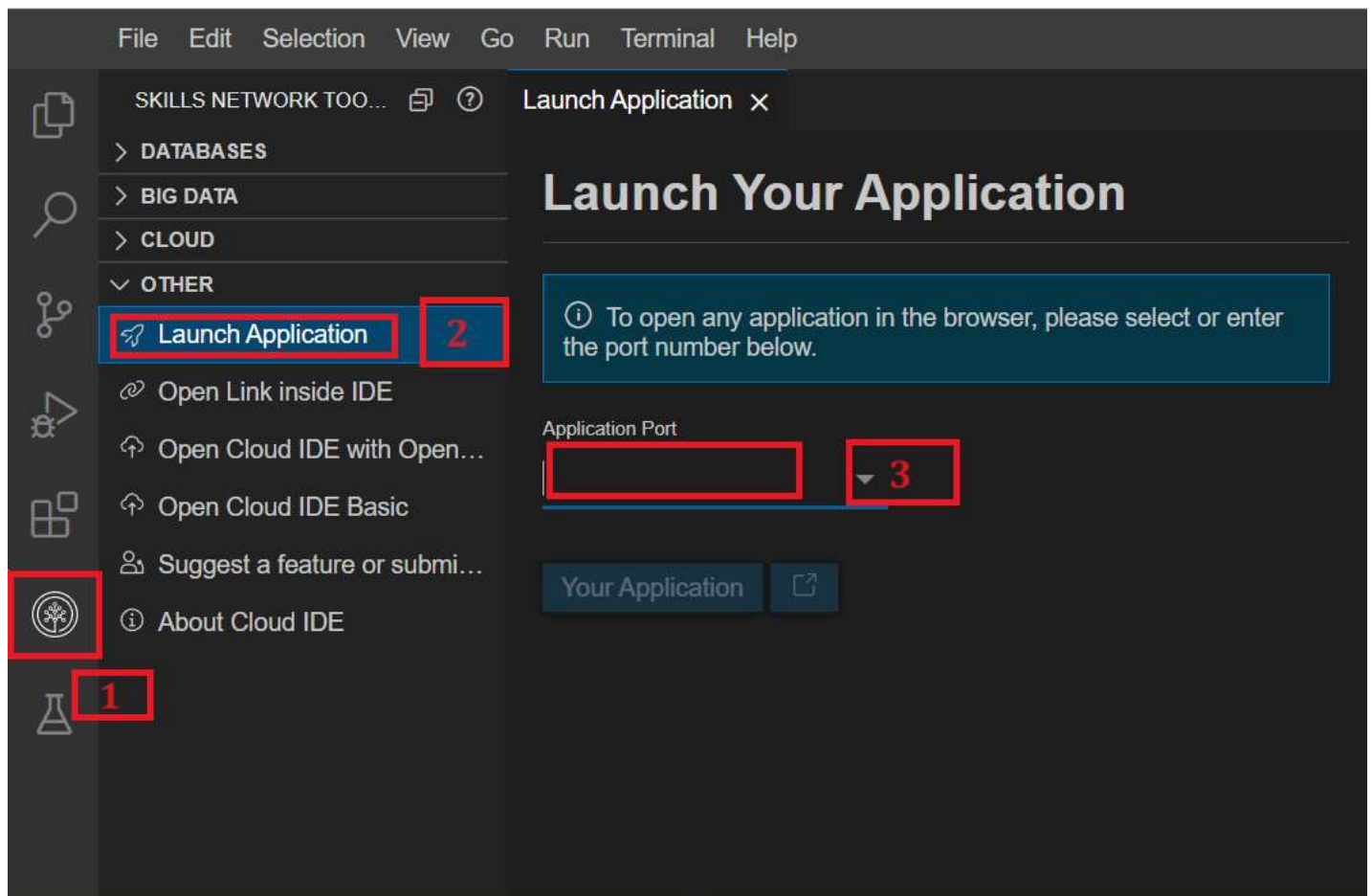
4. It will open your default browser, where you will first see the name of your cloned folder. Click on that folder, and inside it, among other folders, you will find the **speedAnalysis** folder name. Click on the **speedAnalysis** folder, and then select **speed_analysis.html** file, as shown below.
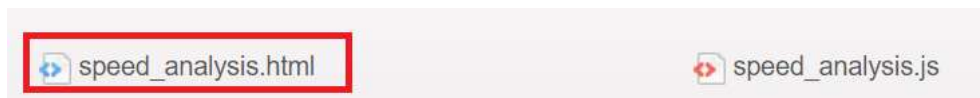
5. It will open the front page and you will see the output as shown below.

Speed Typing Analysis

Type the following text:

```
Get ready to type...
```

Your typing:

```
Start typing when ready
```

Start Test | End Test

6. Click on start button and it will show you the text that you need to type in second input box.

7. After completion of writing clik on End Test button and it will display awnswer as shown below:

Speed Typing Analysis

Type the following text:

```
The quick brown fox jumps over the lazy dog.
```

Your typing:

```
The quick brown fox jumps over the lazy dog.
```

Start Test | End Test

Typing Test Results:

Words Typed: 9

Time Elapsed: 5.11 seconds

Words Per Minute (WPM): 106

Note: After pasting the code, save your file. You can use any output method for this. If you edit your code, simply refresh your browser running through port number 5500. This way, there is no need to launch the application again and again.

Step 4: Perform Git commands

1. Perform `git add` to add latest files and folder by writing given command in terminal in git environment.

   1. 1
   1. `git add --a`
   [Copied!]

   Make sure terminal should have path as follows:

2. Then perform `git commit` in the terminal. While performing `git commit`, terminal can show message to set up your `git config --global` for user. name and user.email. If yes, then you need to perform `git config` command as well for `user.name` and `user.email` as given.

   1. 1
   1. `git config --global user.email "you@example.com"`
   [Copied!]

   1. 1
   1. `git config --global user.name "Your Name"`
   [Copied!]

   > **Note:** Replace data within qoutes with your own details.

   Then perform commit command as given:

   1. 1
   1. `git commit -m "message"`
   [Copied!]

3. Then perform `git push` just by writing given command in terminal.

   1. 1
   1. `git push origin`

Copied!

- After the push command, the system will prompt you to enter your username and password. Enter the username for your GitHub account and the password that you created in the first lab. After entering the credentials, all of your latest folders and files will be pushed to your GitHub repository.

Practice Task

- You can also calculate the total length of text which you have capturted in variable `userTypedText` that user has entered. For this you need to utilize:

  - length property to check the length of entire text using variable **userTypedText**.
  - Include in the output result as well.

- The output will look like as below:

Speed Typing Analysis

Type the following text:

```
The quick brown fox jumps over the lazy dog.
```

Your typing:

```
The quick brown fox jumps over the lazy dog.
```

| Start Test | End Test |

Typing Test Results:

Total Length:44

Words Typed: 9

Time Elapsed: 3.00 seconds

Words Per Minute (WPM): 180

- Perform `git add`, `git commit` and `git push` commands to push your latest task and keep your code updated on GitHub repository.

Summary

1. **Interface setup and functionality establishment:** You created a web-based typing test interface using HTML and JavaScript. The HTML structure formed the groundwork for text display (inputText), user input (userInput), and a dynamic button (Start Test).

2. **Test execution and result analysis:** Clicking the "Start Test" button prompted users to type a predetermined text snippet in the designated input area. Upon test completion, the code swiftly calculated and displayed essential typing metrics, such as words typed, time elapsed, and words per minute (WPM), providing immediate insights into the user's typing proficiency.

3. **User experience and iterative learning:** You created a practical platform to practice typing skills in a controlled environment. You also created immediate feedback on typing speed and accuracy, enabling users to assess their performance, facilitating iterative learning, and improving typing proficiency.