**Simple Text Formatting Tool Documentation**

**Introduction**

The Simple Text Formatting Tool is a web-based application designed to provide users with a range of text formatting and styling options. It allows users to format and style text within a text input area, making it a versatile tool for creating well-structured content. This documentation will provide an in-depth explanation of the app's features and core code concepts.

**Features**

**1. Text Formatting Options**

* **Bold**: Makes the selected text bold.
* **Italic**: Makes the selected text italicized.
* **Underline**: Adds an underline to the selected text.
* **Strike-through**: Adds a strike-through to the selected text.

**2. Font and Font Size Selection**

* **Font Name**: Allows users to choose from a list of predefined fonts.
* **Font Size**: Lets users select the font size from 1 to 7.

**3. Alignment Options**

* **Left Alignment**: Aligns the selected text to the left.
* **Center Alignment**: Centers the selected text.
* **Right Alignment**: Aligns the selected text to the right.
* **Justify Alignment**: Justifies the selected text.

**4. Spacing Options**

* **Line Spacing**: Adjusts the line spacing of the selected text.
* **Letter Spacing**: Adjusts the letter spacing of the selected text.
* **Paragraph Spacing**: Adjusts the spacing between paragraphs.

**5. Script Formatting**

* **Subscript**: Formats the selected text as subscript.
* **Superscript**: Formats the selected text as superscript.

**6. Font Selection**

* **Font List**: Users can select the font for their text from a predefined list.

**7. Font Size Selection**

* **Font Size**: Users can choose the font size from 1 to 7.

**8. Inserting Images**

* **Insert Image**: Allows users to insert images by entering the image URL.

**9. Creating Links**

* **Create Link**: Prompts users to enter a URL and creates a link from the selected text.

**Core Code Concepts**

**1. Selecting DOM Elements**

let optionsButtons = document.querySelectorAll(".options-button");

let advancedOptionsButtons = document.querySelectorAll(".adv-option-button");

let fontName = document.getElementById("fontName");

let fontSizeRef = document.getElementById("fontSize");

let writingArea = document.getElementById("text-input");

let linkButton = document.getElementById("createLink");

let alignButtons = document.querySelectorAll(".align");

let spacingButtons = document.querySelectorAll(".spacing");

let formatButtons = document.querySelectorAll(".format");

let scriptButtons = document.querySelectorAll(".script");

In this section of the code, various DOM elements are selected using **querySelectorAll** and **getElementById**. These elements are crucial for interacting with the user and applying formatting to the text input area.

**2. Initializing the Application**

const initializer = () => {

    // function calls for highlighting buttons

    // No highlighting for link, unlink, lists, undo, redo

    // since they are one time operations

    highlighter(alignButtons, true);

    highlighter(spacingButtons, true);

    highlighter(formatButtons, false);

    highlighter(scriptButtons, true);

// create options for font names

fontList.map((value) => {

    let option = document.createElement("option");

    option.value = value;

    option.innerHTML = value;

    fontName.appendChild(option);

});

// fontsize allows only till 7

for (let i = 1; i <= 7; i++) {

    let option = document.createElement("option");

    option.value = i;

    option.innerHTML = i;

    fontSizeRef.appendChild(option);}

    // default fontSizeRef

    fontSizeRef.value = "3";

};

The **initializer** function sets up the initial state of the application. It populates the font and font size dropdowns, sets default values, and prepares the application for user interaction.

**3. Adding Functionality to Buttons**

const modifyText = (command, defaultUi, value) =>

{

    // execCommand executes command on selected text

    document.execCommand(command, defaultUi, value);

};

// for basic operations which dont need value parameter

optionsButtons.forEach((button) => {

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

    modifyText(button.id, false, null);

    });

});

// options that require value parameter like color

advancedOptionsButtons.forEach((button) => {

    button.addEventListener("change", () => {

        modifyText(button.id, false, button.value);

    });

});

let insertImageButton = document.getElementById("insertImage");

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

    // Prompt the user to enter the image URL

    let imageUrl = prompt("Enter the image URL:");

    // Check if the user entered a URL

    if (imageUrl) {

        // Create an image element

        let image = document.createElement("img");

        image.src = imageUrl;

        // Insert the image at the current selection in the editor

        document.execCommand("insertImage", false, imageUrl);

    }

});

// link

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

    let userLink = prompt("Enter a url");

    // if link has http pass directly else add https

    if (/http/i.test(userLink)) {

        modifyText(linkButton.id, false, userLink);

    }

    else {

        userlink = "http://" + userLink;

        modifyText(linkButton.id, false, userLink);

    }

});

This section of the code handles the functionality of the buttons. It defines the **modifyText** function to execute commands on selected text and attaches event listeners to the buttons for different formatting options. Additionally, it allows users to insert images and create links.

**4. Click Effect on Buttons**

const highlighter = (className, needsRemoval) =>

{

    className.forEach((button) => {

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

            // needs Removal = true means only one button

            // should be highlight and other would be normal

            if (needsRemoval) {

                let alreadyActive = false;

                // if currently clicked button is already active

                if (button.classList.contains("active")) {

                    alreadyActive = true;

                }

                // remove highlight from other buttons

                highlighterRemover(className);

                if(!alreadyActive) {

                    // highlight clicked button

                    button.classList.add("active");

                }

            }

                else {

                    // if other buttons can be highlighted

                    button.classList.toggle("active");

                }

        });

    });

};

const highlighterRemover = (className) => {

    className.forEach((button) => {

        button.classList.remove("active");

    });

};

window.onload = initializer();

In this part of the code, the **highlighter** function is used to provide a visual click effect on buttons. It highlights the clicked button while removing the highlight from others, ensuring that only one button (for some options) or multiple buttons (for others) can be highlighted.

**Conclusion**

The Simple Text Formatting Tool offers a range of features for users to format and style their text easily. Understanding the core code concepts explained in this documentation will enable you to extend and customize this tool according to your specific requirements. It serves as a practical example of how to create a user-friendly text formatting application using HTML, CSS and Javascript.