Geoffrey Gee

IS 520

Theme Generator

Executive Summary

Because I am not currently employed, this project serves to cover multiple interests of mine—namely, graphic design and web development. When considering the prospect of designing a web page, it can be daunting to try to figure out what colors to use and which fonts go best together. Often times, a designer will find inspiration from other sources, but it can be very tedious to try to comb through code to find these elements. With VBA, the process can be automated to provide the designer a simple theme for whichever website provides them with inspiration. This system takes in a theme name and URL from the user, then provides the hex code and a sample of each color, as well as the fonts that were used. If the font is one that the user currently has installed, the sample text next to the font demonstrates what the font looks like. This information is all presented in an easy to read format.

Implementation Documentation

- User initiates process by clicking on the "Get Theme" button under the "Theme Generator" tab of the Ribbon
- Once clicked, the "Get Theme" button calls the macro *showForm* which brings up a user form titled "Get Theme"
- On this user form, the user enters a name for the theme and the URL for website the user wishes to generate a theme around
- If the user clicks "Cancel", the form is unloaded. If the user selects "Get Theme", the macro behind the "Get Theme" button checks for errors or inputs that will lead to errors
    - "Get Theme" Error Checks:
        - Empty Text Boxes - The user form will not pass information on if the text box for the theme name or the URL are empty. If the user attempts to generate a theme while either text box is empty, the sub procedure is exited and the user is prompted to enter a value into the empty box
        - Duplicate Theme Names - The process will not continue if the user enters a theme name that has already been used before. Because the theme name is used to generate a new worksheet titled with the theme name, a duplicate would cause an error. When the user attempts to submit the form, the theme name is checked against all existing worksheets and if the theme name has already been used, the user is prompted to enter a new theme name.
        - HTTP – Because the URL will not work without the full address, any URL that is input without http or https will cause the program to prompt the user to input the complete URL
- Once the user has input the theme name and URL into the user form and clicked "Get Theme", the user form will pass the theme name and the URL to the sub procedure *createTheme*
- *createTheme* stores the URL in a public variable, which is then passed on to *getHTML*

- *getHTML* creates a new folder “temporary” in the same path as the workbook
    - Using the *agent* class provided by Professor Allen, *getHTML* uses Internet Explorer to obtain the HTML file for the URL and save it in the “temporary” folder. The program then exits *getHTML* and returns to *createTheme*
- *createTheme* then calls the procedure *parseHTML*
- When *parseHTML* initializes, it reads the HTML file saved in the “temporary” folder and then uses Internet Explorer to obtain all the stylesheets in the HTML and save them in the “temporary” folder
- *parseHTML* then begins a loop to collect all the fonts
    - In this loop, *parseHTML* begins to read the first CSS file and begins looking for all instances of the string “font-family”
    - The procedure then finds the string of fonts listed after “font-family” and begins to get rid of extraneous things such as extra quotes, apostrophes, or phrases like “!important”
        - In case there are errors in the CSS, *parseHTML* looks for key indicators of when the string has ended and gathers only the fonts
    - Once the string of fonts has been cleaned up, it is then passed to *fontCheck*
    - *fontCheck* isolates the first font in the string of fonts and cleans it up further
        - This is done because the first font in the string is the font the designer intended to be used and all other fonts in the string are there in case it cannot be used
    - The first font is then checked against an array of collected fonts. If it is already in the array, the sub procedure is exited. If it is not in the array, the array is recast to fit an additional font and the font is added
    - *parseHTML* resumes control and the loop is repeated until all the fonts have been found and stored in an array
- *parseHTML* then begins a loop to find all the colors
    - In this loop, *parseHTML* begins to read the first CSS file and begins looking for all instances of the string “color”
    - The procedure then finds the hex color after “color”
        - In case there are errors in the CSS, *parseHTML* looks for key indicators of when the string has ended and gathers the right color
    - Any extraneous code is removed to include on the hex code
    - This code is then passed to *colorCheck*
    - *colorCheck* compares the color to all the hex colors in an array. If the hex color is already in the array, the sub procedure is exited. If it is not, the array is recast and the color is added
    - *parseHTML* resumes control and the loop is repeated until all the colors have been found and stored in an array
- These loops are repeated until each CSS stylesheet has been read and all the fonts and colors found; then *convertColors* is called
- *convertColors* turns any 3-digit hex codes into the 6-digit hex codes that one typically sees in websites. In doing so, it checks to see if the code is a 6-digit hex code and if it is not, the hex code is passed to a function called *fullHex*
- *fullHex* reads the 3-digit hex code, converts it into a 6-digit hex code, and then returns the new code to *convertColors*. This new code replaces the 3-digit code in the colors array

- After *converColors* has finished running, *parseHTML* also finishes and control is returned to *createTheme*
- *createTheme* then passes the procedure *showTheme* the theme name the user input into the system
- *showTheme* uses the theme name to create and name a worksheet for the theme to be displayed
- *showTheme* then colors the background gray and creates the area where the theme will be displayed
- Starting at the top of the theme area, *showTheme* begins to fill cells of the theme area with the colors code from the color array and the corresponding colors
    - In this process, the HEXCOL2RGB function is used. This function converts hex codes into the corresponding RGB value
- Once all the colors have been put into the theme, the fonts are displayed
    - Next to each font is the phrase "Example Text". If the corresponding font is found on the user's computer, the example text is changed to match it
- *showTheme* is finished and control goes back to *createTheme*
- In order to prevent lots of files from accumulating, *createTheme* clears the folder "temporary" and then ends the sub
- The user form is unloaded and the entire process is finished

Concepts Learned

One of the things that I learned the most about was planning. This may seem surprising, but I found that I had a tendency to get lost in the small tasks or different ideas. After I would step away from the project and then come back, I would feel frustrated by how little I had accomplished. However, I began to start my work with planning sessions wherein I would outline what I needed to accomplish that day and where I was in the project. As I began to do this, I sped up considerably and felt like I was accomplishing much more while maintaining site of the big picture.

I also learned a lot about creative problem solving. When problems would arise in my code or when I wasn't sure how to proceed, I could often tinker with the code until I found what I needed to do. By the end of the project, I was able to learn that often there are multiple answers to a problem and if there is something that prevents me from pursuing one course, I can head another.

Finally, I found it was incredibly important to keep my code clean and with comments. Early on, when something wouldn't work, I would comment it out in case I needed to return to it later. However, I never did and the build-up of old code made my work hard to navigate. Thus I began to delete code and insert comments to provide quick maneuverability. This saved me a lot of time and frustration.

Assistance

As I was attempting to figure out how to best convert hex colors to RGB values, I came across a prewritten sub-procedure titled HEXCOL2RGB. I decided that I would use this sub-procedure in my code rather than type out an identical procedure. The website where this procedure can be found is http://www.freevbcode.com/ShowCode.asp?ID=6324.