J S S Mahavidyapeetha

Sri Jayachamarajendra College of Engineering (SJCE), Mysore – 570 006

An Autonomous Institute Affiliated to

Visvesvaraya Technological University, Belgaum

"CSS Minifier in Python Language"

Mini project report submitted in partial fulfillment of curriculum prescribed for the system software laboratory for the award of the degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

by

Saif Ali.
(4JC14CS086)

Rahul Amin
(4JC14CS074)

Under the Guidance of

Mr. Guru R. and Mrs. Manimala
Assistant Professors,
Department of Computer Science & Engineering,
SJCE, Mysore

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
April 2017

J S S Mahavidyapeetha

Sri Jayachamarajendra College of Engineering (SJCE), Mysore – 570 006

An Autonomous Institute Affiliated to

Visvesvaraya Technological University, Belgaum

CERTIFICATE

This is to certify that the work entitled **"CSS Minifier in Python Language"** is a bonafied work carried out **by Saif Ali and Rahul Amin** in partial fulfillment of the award of the degree of **Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belgaum during the year 2017**. It is certified that all corrections / suggestions indicated during CIE have been incorporated in the report. The mini project report has been approved as it satisfies the academic requirements in respect of mini project work prescribed for the system software laboratory.

Lab In Charge and Guide	**Lab In Charge and Guide**
Mr. Guru R.	**Mrs. Manimala**
Assistant Professor,	Assistant Professor,
Dept. of CS & E,	Dept. of CS & E,
S.J.C.E, Mysore	S.J.C.E, Mysore

Place: Mysore

Date : 21/04/2017

DECLARATION

We hereby declare that the project work entitled “CSS Minifier in Python Language” submitted to the Sri Jayachamarajendra College of Engineering (SJCE), Mysore, is a record of an original work done by SAIF ALI and RAHUL AMIN under the guidance of Mr.Guru R and Mrs.Manimala Department of Computer Science & Engineering, Sri Jayachamarajendra College of Engineering, and this project work is submitted in the partial fulfillment of the requirements of the degree of BACHELOR OF ENGINEERING in Computer Science and Engineering of Visvesvaraya Technological University, Belgaum during the year 2017.

Saif Ali
4JC14CS086

Rahul Amin
4JC14CS074

ABSTRACT

Introduction:

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. CSS is designed primarily to enable the separation of document content from document presentation, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

CSS minifier is a tool created to minifiy the CSS files for faster access and lowering memory space usage. This basically trims whitespace and removes comments. the minification process does not affect the function of the file but rather simply optimizes it for downloading purposes. This minifier tool is created using Python language.

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our project coordinator, Mr. Guru R and Mrs. Manimala, Department of Computer Science & Engineering, Sri Jayachamarajendra College of Engineering , for their valuable inputs, able guidance, encouragement and whole-hearted cooperation . The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely fortunate to have got this all along the completion of our project work. We are also grateful to respected Dr. H.C. Vijayalakshmi HOD Department of Computer Science & Engineering,

Sri Jayachamarajendra College of Engineering for permitting us to utilize all the necessary facilities of the institution. We are also thankful to all the other faculty & staff members of our department for their kind co-operation and help.

Saif Ali
4JC14CS086

RahulAmin
4JC14CS074

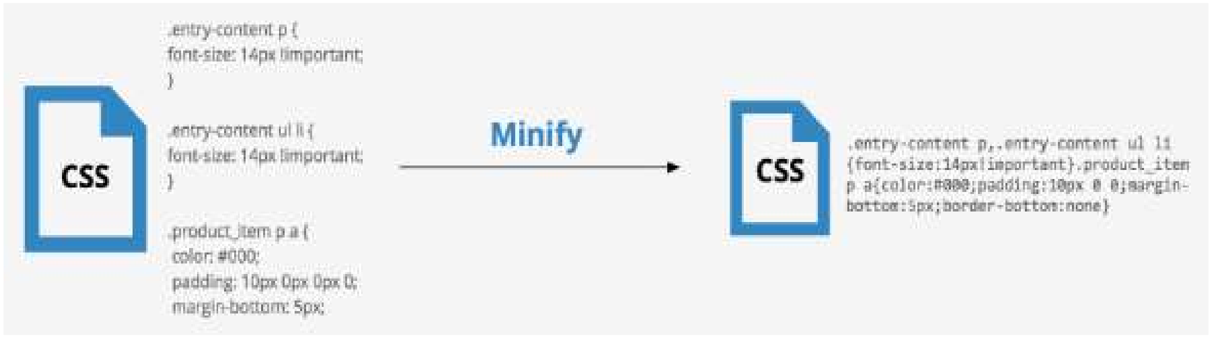
TABLE OF CONTENTS

CHAPTER 1 : INTRODUCTION

1.1 Aim/statement of the problem:

The main aim of the project is to minify the CSS file using the CSS minifier tool. To minify CSS, involves removing any unnecessary characters from within a file to help reduce its size and thus make it load faster.

Examples of what is removed during file minification includes:



```
.entry-content p {
font-size: 14px !important;
}

.entry-content ul li {
font-size: 14px !important;
}

.product_item p a {
 color: #000;
 padding: 10px 0px 0px 0;
 margin-bottom: 5px;
```

```
.entry-content p,.entry-content ul li
{font-size:14px!important}.product_item
p a{color:#000;padding:10px 0 0;margin-
bottom:5px;border-bottom:none}
```

Fig 1

1.2 Difference between minification and compression :

File minification and file compression are not the same thing. Although both aim to achieve the same goal (faster load time) they are both different in how they work. Compression is used to reduce file size by using a compression scheme such as gzip or brotli. Files are compressed before they are sent from the server to the client therefore the compression process is carried out as follow:

1. Files are compressed via a compression method
2. A request is made for the compressed version of a file
3. The compressed file is sent from the server to the client
4. The client uncompresses the file and is able to read the information

Supported compression methods can **vary by server as well as web browsers**. When a browser makes a request to the server it tells the server which compression method it supports so that the server is able to optimize the response for that browser. If the browser does not support any compression method then the server will simply respond with uncompresses data.

1.3 Objectives of the project work:

The main objectives are ,it removes the

- Whitespace characters
- Comments
- Line breaks
- Block delimiters

In the given CSS file as a input. In order to distinguish minified files from unminified files, a _min is often times included within the file name (e.g. foobar_min.css).

Most times, the minification process does not affect the function of the file but rather simply optimizes it for downloading purposes. Minifying the CSS, JS, and HTML files in your production environment is especially useful. File changes are often time not made directly in production thus avoiding the need for visually appealing code. Additionally, since Google takes speed into consideration when ranking, minification helps speed up your website making both Google and your site visitors happy.

1.7 Applications:

The decision to minify CSS, JS, and HTML files provides advantages for both the **website visitors as well as the site owner**. Although,after minification the CSS is harder to read as there are no line breaks, whitespace, etc. However, this optimized format **requires less bytes thus making it easier to download**.

By minifying your files you will benefit by reducing the amount of data transfer required, files will run more quickly in the client's browser, and compression results are enhanced. Taking advantage of minification is a great way to help further optimize your site and is easily achievable with the above mentioned online tools and plugins.

CHAPTER 2 : SYSTEM REQUIREMENTS

The different requirements for the development of the system are mentioned in this chapter.

2.1 Input Requirements:

The CSS file is given as a input to the minifier tool,The given CSS file contains some style specification and the file name is style.css.

2.2 Output Requirements:

In order to distinguish minified files from unminified files, a _min is included within the file name (e.g. foobar_min.css) and this minified CSS file contains no line breaks, whitespace, etc. It's size will be less compared to the input file.

2.3 Hardware Requirements:

This application can be run on any computer and should have sufficient memory to store and run the application.

Basic hardware requirements are as follows:

- Processor with atleast 1 GHz processor
- 4 GB RAM, or minimum RAM required to run the python interpreter.
- Monitor with decent resolution
- Keyboard
- Mouse

2.4 Software Requirements:

Operating System : Windows / Ubuntu

Software : Python2 interpreter and/or an IDE like PyCharm or IDLE

PyCharm is an Integrated Development Environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django.

PyCharm is cross-platform, with Windows, Mac OS and Linux versions.

2.5 Functional Requirements:

- The application must be able to remove all unnecessary white spaces and lines from the given CSS file.
- The application must in no way affect the normal functionality of a CSS file.

2.6 Non-Functional Requirements:

- The applications must have a few basic options for different ways to process CSS files, like leaving comments in or removing them and so on.
- Minimum time should be taken by the application to process the given CSS file and produce the minified file.
- The minified file should be recognizable with a _min.css tag added the the end of file name.
- Ex – I/P – style.css   O/P – style_min.css
- The program should display proper details of the resultant CSS file.

CHAPTER 3 : TOOLS AND TECHNOLOGY USED

Different tools and technology used for the development of the CSS Minifier application are:

PYTHON: PyCharm IDE is used for designing of the system.

Back-end design:

(a) Python Regular Expressions module - **re**
(b) Python Lists
(c) Python File I/O modules.

Front-end design:

(a) Python **Tkinter** module for basic Front end design.
(b) Python **PIL** module to use images
(c) Python **tkFileDialog** and **tkMessageBox** for graphical browsing and displaying messages.

MS Word:

This tool was used to create the report. Pycharm editor and notepad was used to edit the Readme.md file.

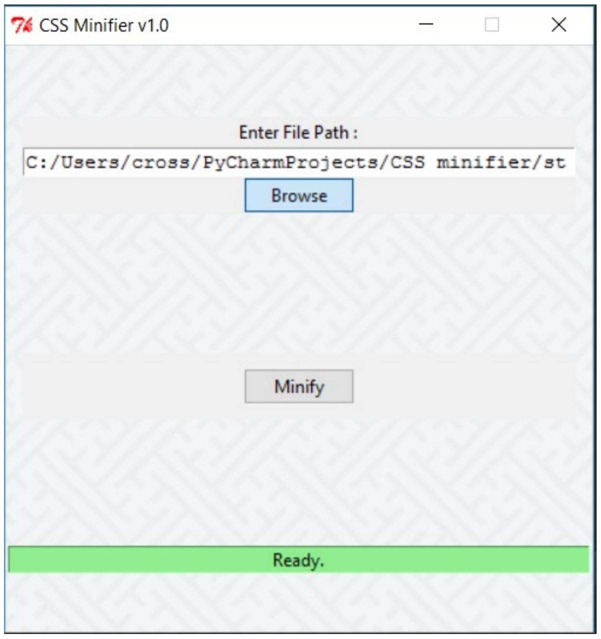
CHAPTER 4 : SYSTEM DESIGN

4.1 Back-End Design:

The system has been designed using pattern matching mechanisms of python. The **re module** of python is highly efficient when it comes to pattern matching. A specific **Core.py** file has been developed to define all the different functions performing the different changes on the given input CSS file.

The basic design of the system is represented by the hierarchy of the two python files. According to the structure of the program, the given **cssmin.py** file does the job of a manage i.e. it takes the file for input, asks for the choices of the user in processing the file, and sends the file for processing and creates the output file. This input file for processing accesses the actual processing functions from the Core.py file which then makes actual changes to the lines of the given file and returns the modified lines.

4.2 Front-End Design:



Fig(3): Front-End Design of the system

The system consists of the text field to take the path of the input CSS file from the user. The entered path can be accessed using the Browse button provided.
The minify button is then used to access Core functions and minify the given CSS file. A basic status bar is provided below to display arbitrary information about ongoing processing.

CHAPTER 5 : SYSTEM IMPLEMENTATION

Cssmin.py

This file is used as a handler to manage all functions and requirements of the user to process the given input CSS file. It accepts the file, sends it for processing and produces the output file.

```
#!/usr/bin/env python2.7
import ttk
from Tkinter import *
from tkFileDialog import askopenfilename
import tkMessageBox
from PIL import Image, ImageTk
import Core


def askToQuit():
    if tkMessageBox.askyesno("Confirm!", "Do you really want to quit such an
awesome program? "):
        sys.exit()

class MainWindow:
    def __init__(self, hello):
        try:
            hello.title('CSS Minifier v1.0')
            hello.protocol('WM_DELETE_WINDOW', askToQuit)
            self.bg_image = ImageTk.PhotoImage(Image.open("src/bg.png"))

            w = self.bg_image.width()
            h = self.bg_image.height()
            hello.geometry("%dx%d" % (w, h))
            hello.resizable(width=False, height=False)
            #hello.wm_attributes("-transparentcolor", "white")

            self.bglabel = Label(hello, image=self.bg_image)
            self.bglabel.pack(side='top', fill='both', expand='yes')
            self.bglabel.image = self.bg_image

            self.PathNameFrame = Frame(self.bglabel)
            self.PathNameFrame.pack(fill=None, expand=True, pady=10, padx=10)

            self.MFrame = Frame(self.bglabel)
            self.MFrame.pack(fill=X, expand=True, pady=10, padx=10)

            self.BottomFrame = Frame(self.bglabel)
            self.BottomFrame.pack(fill=X, expand=True, side=BOTTOM)

            self.label1 = Label(self.PathNameFrame, text="Enter File Path :",
anchor=CENTER)
            self.label1.pack(side=TOP, fill=None)

            self.entry1 = Text(self.PathNameFrame, height=1, width=60)
            self.entry1.pack(side=TOP)

            self.bbutton = ttk.Button(self.PathNameFrame, text="Browse")
            self.bbutton.pack(side=BOTTOM, fill=None, padx=5)
            self.bbutton.bind("<Button-1>", lambda event: self.getFileName(hello))

            helloButton1 = ttk.Button(self.MFrame, text="Minify")
            helloButton1.pack(side=BOTTOM, padx=10, pady=10, anchor=CENTER,
fill=None, expand=True)
            helloButton1.bind("<Button-1>", lambda event: self.minify())

            self.status = Label(self.BottomFrame, text="Ready.", bd=1,
relief=SUNKEN, anchor=CENTER, bg='light green')
            self.status.pack(side=BOTTOM, fill=X)

        except:
            return

    def getFileName(self,root):
        root.iconify()
        self.entry1.delete(0.0, END)
        fileName = askopenfilename(initialdir='/', title='Choose CSS file to
minify')
        print fileName
        self.entry1.insert(0.0, fileName)
        self.status.configure(text="Ready.", bg='light green')
        root.deiconify()


    def minify(self):
        filename = StringVar()
        filename=self.entry1.get(0.0, END)
        filename = filename.strip("\n")
        with open(filename, 'r') as source:
            lines = source.readlines()

        lines = Core.replace_tabs2(lines)
        lines = Core.remove_trailing_white_spaces(lines)
        lines = Core.remove_blank_lines(lines)
        lines = Core.remove_comments(lines)
        lines = Core.remove_blank_lines(lines)
        lines = Core.optimize_lines(lines)
        lines = Core.condense_lines(lines)

        destination = filename.replace(".css", "_min.css")
        dest = open(destination, "w")
        dest.writelines(lines)

        self.status.configure(text="File Minified!", bg='light green')



def main():
    hello = Tk()
    MainWindow(hello)
    hello.mainloop()


if __name__ == "__main__":
    main()
```

Core.py

This file contains all the different functions the given lines of the files should be processed using. The cssmin.py file accesses this file for all the processing and the modified lines are then returned back to it.

```
import re

def remove_everything_between(subs1, subs2, line):
    regex = re.compile(subs1 + r'.*' + subs2)
    return regex.sub('', line)


def remove_everything_before(subs, line):
    regex = re.compile(r'.*' + subs)
    return regex.sub('', line)


def remove_everything_past(subs, line):
    regex = re.compile(subs + r'.*')
    return regex.sub('', line)

def is_in_property_set():
    return in_property_set

def enter_property_set():
    global in_property_set
    in_property_set = True
    return

def exit_property_set():
    global in_property_set
    in_property_set = False
    return


def has_data():
    return has_data


def remove_blank_lines(lines):
    replacedLines = []
    for i,line in enumerate(lines):
        if line != '\n':
            replacedLines.append(line)
    return replacedLines

def replace_tabs(lines):
    replacedLines = []
    for line in lines:
        result = str()
        for c in line:
            if c == '\t':
                result += ' ';
            else:
                result += c
        replacedLines.append(result)
    return replacedLines

def remove_trailing_white_spaces(lines):
    stripped_lines = []
    in_line = False
    for line in lines:
        result = str()
        in_line = False
        for c in line:
            if c == ' ' and in_line == True:
                result+=c
            if c != ' ':
                in_line = True
                result+=c
        stripped_lines.append(result)
    return stripped_lines


def remove_comments(lines):
    start, end = '/*', '*/'
    escaped_start, escaped_end = '/\*', '\*/'
    in_comment = False
    newlines = []
    for line in lines:
        if not in_comment:
            start_pos = line.find(start)
            if start_pos != -1:
                in_comment = True
                end_pos = line.find(end)
                # inline multiline comment
                if start_pos < end_pos:
                    line = remove_everything_between(escaped_start, escaped_end,
line)
                    in_comment = False
                else:
                    line = remove_everything_past(escaped_start, line)
        else:
            end_pos = line.find(end)
            if end_pos != -1:
                line = remove_everything_before(escaped_end, line)
                in_comment = False
                start_pos = line.find(start)
                # start of another comment on the same line
                if start_pos != -1:
                    line = remove_everything_past(escaped_start, line)
                    in_comment = True
            else:
                line = ''
        newlines.append(line)
    return newlines


def optimize_lines(lines):
    result = []

    for line in lines:
        line = re.sub(r'[ ]*[{]', "{", line)
        line = re.sub(r'[ ]*[}]', "}", line)
        line = re.sub(r'[ ]*[:]', ":", line)
        line = re.sub(r'[:][ ]*', ":", line)
        line = re.sub(r'[ ]*[;]$]', ";", line)
        line = re.sub(r'[ ]*[\n]$]', "\n", line)

        result.append(line)

    return result
```

```python
def condense_lines(lines):
    final = []
    for line in lines:
        result = str()
        for c in line:
            if c != '\n':
                result+= c
        final.append(result)
    return final
```

CHAPTER 6
SYSTEM TESTING AND RESULT ANALYSIS

Testing and result analysis of the system can be done in the following way:

In the command line :

```
$ python cssmin.py
Enter the file to minify : style.css
The minified CSS file is style_min.css
```

I/P file – style.css

```css
.description {
	max-width: 705px;
	color: grey;
	font-size: 1.875em;

}

.screenshot {
	max-width: 460px;
	max-height: 460px;
	margin-right: 20px;
}


* {
	-webkit-box-sizing: border-box;
```

```
        -moz-box-sizing: border-box;
        -ms-box-sizing: border-box;
        box-sizing: border-box;
}

.app  {
        display: flex
}

/* Comment this shit */

.title {
        padding-left: 10px 20px 0 0;
        /* padding-top: 20px; */
        margin-bottom: 20px;
        background-color: #33bebe;
        max-width: 1165px;
        color: white;
}
```

O/P File – style_min.css

```
.description{max-width:705px;color:grey;font-size:1.875em;}.screenshot{max-width:460px;
max-height:460px;margin-right:20px;}*{-webkit-box-sizing:border-box;-moz-box-
sizing:border-box;-ms-box-sizing:border-box;box-sizing:border-
box;}.app{display:flex}.title{padding-left:10px 20px 0 0;margin-bottom:20px;background-
color:#33bebe;max-width:1165px;color:white;}
```

CHAPTER 7
CONCLUSION AND FUTURE WORK

The CSS minifier tool is successfully implemented and all the whitespace characters,line breaks,etc were removed from the input file ,making it convenient for storing and downloading for future use. Similar minifier tools can be developed for JS and HTML files. The scope of the project is only limited to CSS files ,

Google takes into account page load time into its ranking algorithm. As such, web developers the world over are trying to squeeze as much functionality into as few bytes as possible. This is where Minifiers come into the picture – they automatically remove all un-needed bytes from javascript and css files before sending them to the user. Less bytes means less load time. This is all well and good, but a new issue crops up when using Google Minify – the Expires header is not a Far Future Header.

REFERENCES

- Robert W.Sebesta: Programming the World Wide Web,4th Edition , Education,2012.
- https://www.w3schools.com/cssref/
- https://en.wikipedia.org/wiki/Cascading_Style_Sheets
- https://www.tutorialspoint.com/css/
- https://www.jetbrains.com/help/pycharm/2017.1/quick-start-guide.html
- https://confluence.jetbrains.com/display/PYH/PyCharm+Tutorials