BERGISCHE UNIVERSITÄT WUPPERTAL

Bachelor's Thesis

for obtaining the academic degree:
Bachelor of Science (B. Sc.) in Applied Science

Design and Implementation of a Web Application for Image Processing in a Browser

Author: Khalid Bellouch
Supervisor: Herr Dr. Arndt
Advisor: Herr Prof. Dr. Lang
Submission date: 23.05.2022

Erklärung

gem. § 12 Abs. 7 Prüfungsordnung 2007/2012

Hiermit erkläre ich, dass ich von mir eingereichte Abschlussarbeit (Bachelor-Thesis) selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Stellen der Abschlussarbeit, die anderen Werken dem Wortlaut oder Sinn nach entnommen wurden, in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wuppertal,

Unterschrift

Erklärung

Hiermit erkläre ich mich damit einverstanden, dass meine Abschlussarbeit (Bachelor-Thesis) wissenschaftlich interessierten Personen oder Institutionen und in Rahmen von externen Qualitätssicherungsmaßnahmen der Studienganges zur Einsichtnahme zur Verfügung gestellt werden kann.
Korrektur- oder Bewertungshinweise in meiner Arbeit dürfen nicht zitiert werden.

Wuppertal,

Unterschrift

I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

2. Acknowledgment

I wish to extend my special thanks to my parents, my brother for helping me during my residence in Germany since the first day, my whole family for supporting me and both Dr. Arndt and Prof. Dr. Lang for guiding me through the full journey for preparing this Bachelor Thesis.

3. Abstract

This bachelor's thesis describes the design and implementation of a fast and flexible web application called "PictureLab" intended for processing images by operating directly on the image's data using primarily the programming language JavaScript and making use of the Canvas API newly introduced in HTML5.

Diese Bachelorarbeit beschreibt den Entwurf und die Implementierung einer schnellen und flexiblen Webanwendung namens "PictureLab", die zur Verarbeitung von Bildern durch direkter Zugriff und Berarbeitung der Pixeldaten einer Bilddatei, wobei hauptsächlich die Programmiersprache JavaScript verwendet wird und die in HTML5 neu eingeführte Canvas-API verwendet wird.

4. Table of figures

Figure 1: Landing Page.....14
Figure 2: Editor Page.....14
Figure 3: Original Image (Source:Unsplash.com).....25
Figure 4: Brightness effect applied (Brightness value = 100).....26
Figure 5: Brightness effect applied (Brightness value = -100).....26
Figure 6: Contrast effect applied (Contrast value = 50).....29
Figure 7: Contrast effect applied (Contrast value = -50).....29
Figure 8: Saturation effect applied (Saturation value = 5).....31
Figure 9: Temperature effect applied (Temperature value = 30).....33
Figure 10: Temperature effect applied (Temperature value = -30).....33
Figure 11: How Convolution works.....34
Figure 12: Blur effect applied (Blur value = 5).....37
Figure 13: Sharpness effect applied (Sharpness value = 1).....38
Figure 14: Invert effect applied.....39
Figure 15: Grayscale effect applied.....41
Figure 16: Sepia effect applied.....43
Figure 17: Solarization effect applied.....45
Figure 18: Combined Adjustments applied.....47

5. Table of Content

2. Acknowledgment....4
3. Abstract....5
4. Table of figures....6
5. Table of Content....7
6. Introduction....9
6.1 Motivation....9
6.2 Goal of the Project....10
6.3 Building the Project....11
7. Design Phase....12
7.1 Introduction....12
7.2 Adobe XD....13
7.3 The Design....13
8. Implementation Phase....15
8.1 Introduction....15
8.2 HTML....16
8.2.1 Introduction....16
8.2.2 The Canvas API....17
8.3 CSS....18
8.4 JavaScript....19
8.4.1 Introduction....19
8.4.2 ES6/2015....20
8.5 Implementation....21
8.5.1 Introduction....21

8.5.2 Brightness....24
8.5.3 Contrast....27
8.5.4 Saturation....30
8.5.5 Temperature....32
8.5.6 Convolution....34
8.5.6.1 The Algorithm....34
8.5.6.2 Blur Effect....36
8.5.6.3 Sharpness Effect....37
8.5.7 Invert....39
8.5.8 GrayScale....40
8.5.9 Sepia....42
8.5.10 Solarize....44
9. Results....46
10. Conclusion....48
11. Glossary....49
12. Literature....51

6. Introduction

6.1 Motivation

Versatility and efficiency are and have always been some of the main points to consider when developing applications for daily life use.
In the seventies and eighties of the last century, computers were slow and had poor storage capacity, in addition to that, building cross-platform applications which would function on multiple platforms such as Windows, Linux, MacOS… was practically unfeasible. The only viable choice then was developing native or platform-specific applications, which are applications intended to work on a specific environment and optimized with that idea in mind.

As Moor's law progressed over the years, computers became faster and more capable of performing faster as well as having increased storage capacity. This has in return increased the time efficiency of modern systems, Storage was also no longer an issue in modern computers, so it allowed for developing larger sized and more complex applications which made web browsers capable of doing more than just displaying static web pages. It basically allowed them to execute code on the client's computer without connecting to the server or transferring any data.

In the recent years, with release of new versions and updates to HTML and JavaScript, the world was introduced to a new concept called web applications. Web applications of Web Apps in short are applications

that run solely on the browser independent of the base platform of the user. This new concept presents many advantages over native applications, some of which are efficiency and versatility. On top of that, web applications guarantees the privacy of the user's data since all code is executed on the client side of the website.

Web applications can be designed to perform any tasks, from calculating the BMI* of a person to artificial intelligence including handling media.

BMI: Body Mass Index

6.2 Goal of the Project

The goal of this project is to demonstrate the capabilities of the browser in displaying and preforming modifications directly on media files especially image files using JavaScript's ability to manipulate the DOM and the Canvas API which grants direct access for reading and modifying the image's image data among numerous other capabilities.

In order to illustrate these features and capabilities, a web application will be designed and implemented for users to process images.

"PictureLab" is the chosen name for the "browser-based photo editing web application" that will act as a demonstration and will offer many features to the user including loading an image file from the user's local storage and displaying and applying filters to said image and in the end saving the edited image back to local storage.

6.3 Building the Project

Building the web application will go through two phases: the designing phase and the implementation phase.

The designing process will be preformed using a design tool named "Adobe XD" and will undergo two stages:
- In the first stage we will be designing the layout and the appearance of the landing page and the editor page, including choosing a theme and a color set and a typography matching the chosen theme.
- in the second stage we will be designing the interactions between all components including connecting the landing page to the editor page, including all the buttons and animations and simulating loading and saving an image from and to local storage.

As for the implementation, we will be using HTML and CSS mainly to implement the layout that we designed in the first phase for both the landing page and the editor page, and for coding the logic behind editor itself we will use the programming language JS to:
- access the DOM and the canvas object which will enable us to read the image's pixel data.
- implement the filters' algorithms that will manipulate the image's data, which represent applying chosen effects on the image, and this is the core functionality of our web application.

7. Design Phase

7.1 Introduction

The Web design process is a documented procedure for completing a typical Web development project. Web development work is divided into high-level sections, which are then broken down into tasks and resources that can be used as a road map for each project. The end result doesn't have to be an exact match to the design as long as it concedes to the major layout.

It starts with the planning phase which is arguably the most important phase is the design process because it dictates how the project will move from here on forwards, and it's the phase where the user interaction is required the most.

After that comes the actual designing phase, where the plan made from the first phase takes shape, it goes from choosing the typography and the color palettes best suited to the goal of the website, to actually making a working mock-up using a design tool.

There are many design tools used by web developers the most famous of which are: Sketch, Figma and Adobe XD which is the one we will use to design our web application.

7.2 Adobe XD

Adobe XD (Adobe Experience Design) is a vector-based UX design tool developed by Adobe Inc. for Windows and MacOS operating systems. it allows for easy and modular creation of layouts and components that can be used to build prototypes for web pages or mobile applications.

7.3 The Design

The design chosen for the Landing page of the web application is as follows:

- Typography: Lovelo Font Family for the title and headings and Geometria Font Family for the main text.
- Colors: the color palette chosen is:

- Design System: the Material UI design system which is the design system created by Google.

Below are screenshots for the design of both the landing page and the editor page.

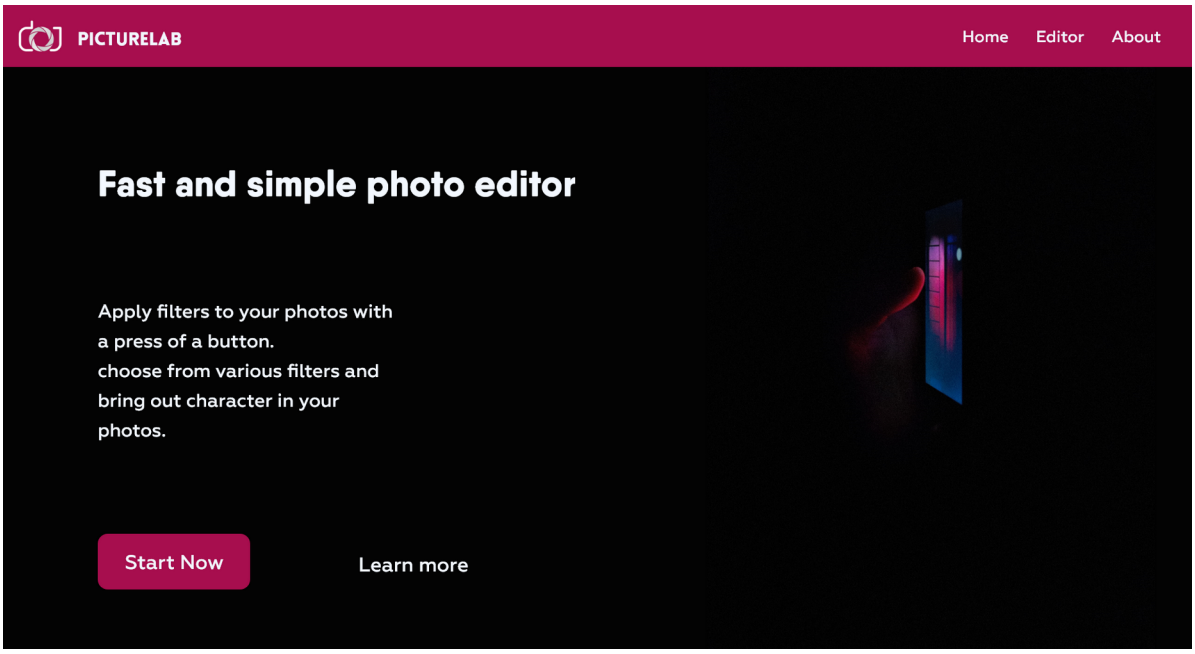

Figure 1: Landing Page

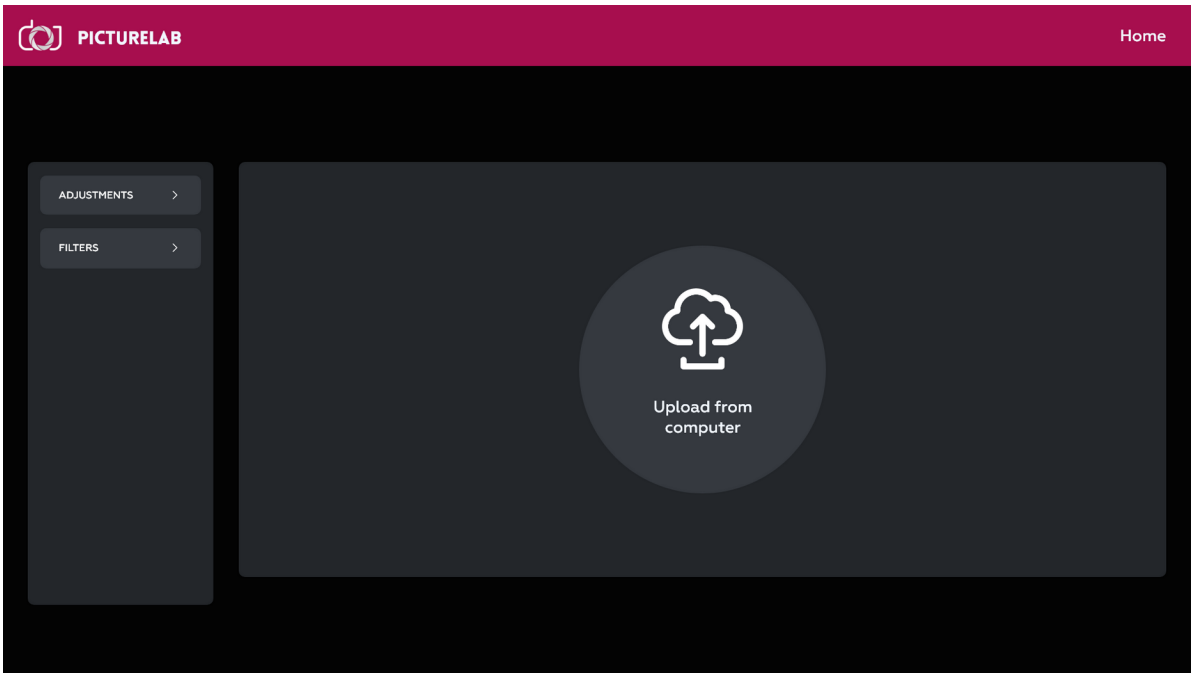

Figure 2: Editor Page

8. Implementation Phase

8.1 Introduction

Implementing the web application will go through three steps in general:

- The first step will handle creating the skeleton of the layout of both the landing page and the editor page using HTML5 as well as create a base of the editor itself by incorporating a file input element to load and save image files from and to local storage and a canvas element for all editing functionalities in the editor page which would be later handled in the third step.

- The second step will handle styling the layout created in the first step and making it look as close to the design we made as possible, for this purpose we will mainly use CSS with a little bit of JS to implement the complex layout and style the elements of both landing and editor pages, as well as implement the animations of the menus in the editor page and the mouse interactions among other things.

- The third step and considered the most important step is implementing the functionalities of the editor using JavaScript which will include: loading and reading image files from the local storage using the File API, displaying the actual image on the canvas, reading the image's pixel data and editing it by applying algorithms to it, and in the end saving the edited image back to the local storage.

8.2 HTML

8.2.1 Introduction

HTML (HyperText Markup Language) is the most essential building block of the Internet. It was first released in 1993 by Sir Tim Berners-Lee. It characterizes the semantics and construction of web content. HTML is used in conjunction with different other technologies to define the layout of a website page's appearance/show (CSS) or define functionality/logical interactions (JavaScript).

“Hypertext” means links that connect web pages together with each other, either within one website or between websites on different servers. Links are a key part of the Web. By transferring content to the Web and connecting it to pages made by others, you become a functioning part in the Web.

HTML utilizes “markup” to create elements of text, pictures, and other elements in the construction of a web page. A HTML element consists of “tags” which comprise of the component name put between the symbols: “<” and “>”. The name of a component inside a tag is case insensitive. The tags come usually in pairs (an opening tag and a closing tag) and the element’s content in between.

HTML5 is the fifth and final version released on 2014 by WHATWG as W3C Recommendation and it incorporates many new "tags" and APIs, for example: `<header>` `<footer> <section> <video>` and numerous other tags including the one we will make use of in this project: the `<canvas>` tag.

8.2.2 The Canvas API

The <canvas> element in HTML5 is a newly introduced element that uses the Canvas API to draw graphics on a web page, it can be used to achieve many functionalities which includes animation, data visualization, game graphics, video processing, and photo manipulation.

Another alternative for the Canvas API is the WebGL API which can be used in conjunction with the <canvas> element to draw hardware-accelerated 2D as well as 3D graphics. However it requires the user's computer to support this feature.

In our case will be using the canvas API to display images and have direct access to its image data in order to read and perform calculations on it.

8.3 CSS

CSS (Cascading Style Sheets) is a style sheet language used to describe the styling of structured documents (mainly HTML and XML), it was first released in November 1994 by Håkon Wium Lie who worked at CERN with Sir Tim Berners-Lee the maker of HTML.

CSS allows for controlling the appearance of basic elements such as changing the background color of a button or the font and size of a text, creating custom layouts for text to be displayed in, and even more complex features such as animating a drop down menu or an element when hovering on it with the mouse.

CSS uses simple syntax which consists of a set of rules to define the element to be styled and the properties to be modified written as a key-value pairs, an example of a CSS rule would look like this:

```
h1 {
    font-size: 24px;
    color: blue;
}
```

CSS is considered one of the three major World Wide Web technologies alongside HTML and JS and is currently maintained by World Wide Web Consortium (W3C).

8.4 JavaScript

8.4.1 Introduction

JavaScript (abbr. JS) is a lightweight, interpreted, prototype-object-oriented computer programming language with first-class-functions. It was originally implemented as a part of web browsers in 1995, however nowadays JavaScript is also used in many non-browser environments like Node.js or Acrobat Reader.

JavaScript runs mainly on the client-side of a website, which means it can be used to handle the appearance and the behavior of the web page upon interaction with the end user and it does this through the DOM API.

JavaScript is interpreted, meaning it does not compile before being executed, instead, a Parser reads our code line-by-line, understands how the code fits the syntax and what is expected from the code to do and breaks it into chunks that gets passed to a JavaScript engine which translates it into binary instructions (machine code) that the CPU executes.

The base syntax of JavaScript is similar to C++ and Java, but JavaScript is not to be mistaken for Java as they are completely different programming languages.

JavaScript conforms to the ECMAScript specifications, it has gone through many revisions in the recent years and the last and current edition of ES specification is ECMAScript 2021 released on June 2021.

8.4.2 ES6/2015

ES6 or the ECMAScript 2015 is the 6th and the major edition of the ECMAScript language specification standard, it has brought significant changes in JavaScript, many of them were much wanted from developers around the world which made JavaScript much more capable and versatile and made writing code in JS much less complicated and more understandable.

The list of changes that ES6 came with is large, here is a list of the most important ones:

- `let` and `const` Keywords
- Arrow Functions
- The spread Operator
- Default Parameters in functions
- Destructuring Assignment

- Classes
- Modules

In our project we will make use of many of these new features to simplify our code and make it easier to understand and to prevent unwanted behavior and bugs from occurring in our application.

8.5 Implementation

8.5.1 Introduction

Before we jump into the algorithms used to modify the images we need to understand how we will use JavaScript to read and operate on the image pixel data.

We will use the File API in Javascript in order to load and save the image data. The `FileReader` Object reads the data from a file in the user's computer using a `File` Object, the file in question will be selected using an HTML element `<input type="file">`.

We will use the Canvas API in JavaScript to read the Image data through `getImageData()` function which returns an `ImageData` object that includes:

- `ImageData.height`: of type `unsigned long` indicating the height of the selected canvas in pixels.

- `ImageData.width`: of type `unsigned long` indicating the width of the selected canvas in canvas.
- `ImageData.data`: of type `Uint8ClampedArray` which is an array of integer values clamped between 0 and 255 and represents the values of the pixels, each pixel is represented by 4 values following the RGBa model in which colors are represented as values in a Green, Blue, and Red channel together with an Alpha channel which represents the transparency of the pixel.

The `ImageData` object itself is immutable which means it cannot be reassigned however the values of the `ImageData.data` array can be altered, which will allow us to perform modifications and apply our algorithms on the pixels of our image. And after that we will use the function `putImageData()` to redraw the edited image data to the canvas for the user to see and to save.

The list of the algorithms used to apply as adjustments and filters to the image will include the following:

- Brightness
- Contrast
- Saturation
- Temperature
- Blur
- Sharpness
- Invert

- GreyScale
- Sepia
- Solarize

We will also be using event listeners of JavaScript to monitor and apply the modifications and filters selected by the user.

8.5.2 Brightness

Brightness is defined as the intensity of a pixel relative to another. In order to change the brightness of an image we have to increase (or decrease depending on the user's choice) the values of the pixels by a constant value.

The algorithm used to achieve this effect is simple and straightforward, first of all we read the image data, after that we increase the values of the red, green and blue channels by the amount specified by the user without altering the values of the alpha channel. In the end, we display the modified image on the canvas.

We don't need to worry about truncating the values of the pixels between 0 and 255 in any of our algorithms since `ImageData.data` Array floors the decimal numbers clamps the values automatically as mentioned above.

The Algorithm goes like this:

```
Brightness (Brightness value)
     Pixels = get the image pixel data from canvas
     newPixels = the resulting data for the edited image
     for all pixels:
          newPixels[Red] = Pixels[Red] + Brightness value
          newPixels[Green] = Pixels[Green] + Brightness value
          newPixels[Blue] = Pixels[Blue] + Brightness value
     draw edited image on canvas using the modified Pixels
```

The images below show the result of applying the algorithm:

Figure 3: Original Image (Source:Unsplash.com)

Figure 4: Brightness effect applied (Brightness value = 100)

Figure 5: Brightness effect applied (Brightness value = -100)

8.5.3 Contrast

Contrast by definition is the difference between the minimum and maximum intensity of the pixels of an image.

The algorithm used in order to achieve this effect on an image depends on the fact that adjusting contrast is done in a linear matter, that is to say, the function of adjusting the contrast of an image is a linear function:

$$f(x) = ax + b$$

first we read the values of the pixels from the canvas and then we take the contrast value chosen by the user and set it as the slop (a) and based on it we calculate the intercept value (b).

the new values of the red, green and blue channels then are calculated as such:

$$\text{new value} = (\text{old value} \times \text{ contrast value}) + \text{intercept}$$

lastly, we redraw the edited image using the new values.

Things worth mentioning are that the (a) value must be between 1 and 2 with the value of (b) equals 128 in case (a) equals 0, so that in this case the effect will be essentially greyscaling the image, and the value of (b) equals 0 when (a) equals 1 and in this case the image will stay the same as original.

The pseudo code for the algorithm is as follows:

```
// input value between -100 and 100
Contrast (input value)
     Pixels = get the image pixel data from canvas
     newPixels = the resulting data for the edited image
     Contrast value = (user value / 100) +1
     Intercept value = (1 - contrast value) * 128
     for all pixels:
          newPixels[Red] = (Pixels[Red] * Contrast value) +
                     Intercept value
          newPixels[Green] = (Pixels[Green] * Contrast value) +
                Intercept value
          newPixels[Blue] = (Pixels[Blue] * Contrast value) +
                     Intercept value
     draw edited image on canvas using the modified Pixels
```

below are the results of applying the contrast filter:

Figure 6: Contrast effect applied (Contrast value = 50)

Figure 7: Contrast effect applied (Contrast value = -50)

8.5.4 Saturation

Saturation describes the intensity of color i.e. the amount of color in a pixel proportional to its brightness. The more saturated a pixel the more colorful and bright it looks and the less saturated the more dull and washed away it looks, grayscaled and black-and-white images have no saturation.

The algorithm we will use to change the saturation will use the fact that saturation visually changes proportionally with the luminance of a pixel, so we calculate the values of the edited pixels using a modified luminance vector which in turn would depend on the original luminance vector and the saturation intensity value and it goes like this:

```
Saturation(saturation value)
     Pixels = get the image pixel data from canvas
     newPixels = the resulting data for the edited image
     declare the default rgb luminance vector luminance[Red, Green, Blue]
     calculate the modified luminance vector using the formulas
          sRed = (1 - saturationValue)*luminance[Red] + saturation value
          sGreen = (1 - saturationValue)*luminance[Green] + saturationvalue
          sBlue = (1 - saturationValue)*luminance[Blue] + saturation value
          mRed = (1 - saturationValue)*luminance[Red]
          mGreen = (1 - saturationValue)*luminance[Green]
          mBlue = (1 – saturationValue)*luminance[Blue]
     for all pixels
          Pixels[Red] = red
          Pixels[Green] = green
          Pixels[Blue] = blue
          newPixels[Red] = (sRed*red + mGreen*green + mBlue*blue)
          newPixels[Green] = (mRed*red + sGreen*green + mBlue*blue)
          newPixels[Blue] = (mRed*red + mGreen*green + sBlue*blue)
     draw edited image on canvas using the modified Pixels
```

below are the results of applying the saturation effect:

Figure 8: Saturation effect applied (Saturation value = 5)

8.5.5 Temperature

The Temperature of a color describes the warmth of the color considering that the red color is warm and blue is cool which means the closer the color to red and farther from blue the warmer it is.

With this in mind, the algorithm to control the color warmth, in other words the color temperature is simple, all we need to do is change the amount of red and blue color in a pixel to make it seam warmer or cooler.

The algorithm goes like this:

```
Temperature(Temperature value)
     Pixels = get the image pixel data from canvas
     newPixels = the resulting data for the edited image
     for all pixels:
          newPixels[Red] = Pixels[Red] + Temperature value
          newPixels[Green] = Pixels[Green]
          newPixels[Blue] = Pixels[Blue] - Temperature value
     draw edited image on canvas using the modified Pixels
```

below are the results of applying the temperature effect:

Figure 9: Temperature effect applied (Temperature value = 30)

Figure 10: Temperature effect applied (Temperature value = -30)

8.5.6 Convolution

8.5.6.1 The Algorithm

In Mathematics, Convolution is defined as an operation on two functions which produces a special composition of those functions that expresses how the shape of one affects the other, the word convolution is used to describe the resulting function as well the process of computing it.

In Image processing, convolution is considered one of the most important topics, it is defined as an operation of transforming an image by applying a kernel to each pixel depending on its value and the values of the pixels surrounding it. The kernel's size and value determine how the effect changes the image and the intensity of the change.

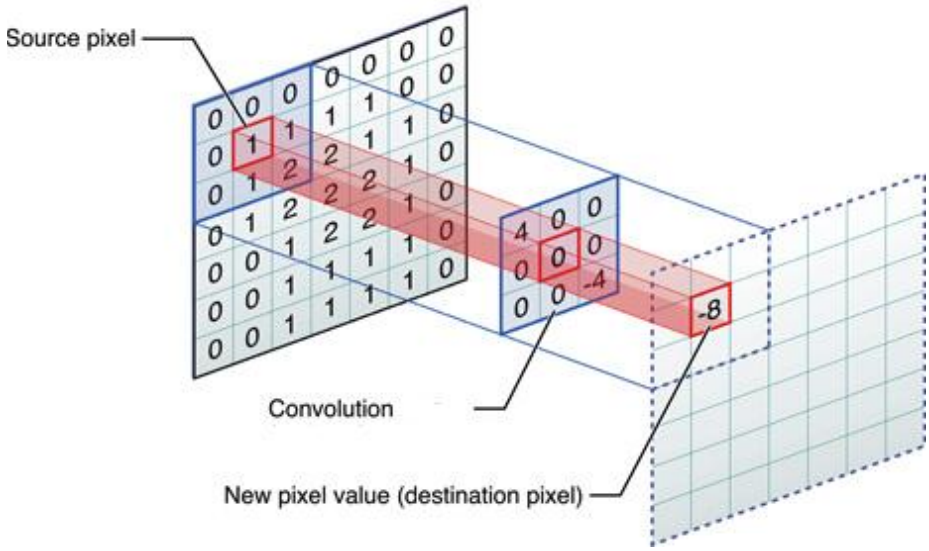

Figure 11: How Convolution works

Source: https://medium.com/@bdhuma/6-basic-things-to-know-about-convolution-daef5e1bc411

As described in Figure 9, the center of the kernel is placed above the pixel we want to operate on, making sure that the full kernel is within the boundaries of the image, then the resulting pixel value is calculated as the weighted sum of the multiplied kernel values each with the corresponding pixel value in the original image. The operation is repeated across all pixels of the image and the resulting values give us the convoluted image.

The algorithm used to apply convolution goes like this:

```
Convolution(kernel)
     Pixels = get the image pixel data from canvas
     newPixels = the resulting data for the edited image
     for all pixels channels ignoring the alpha channel:
          Calculate the resulting value using the methode described
          above multiplying each kernel value with the
          corresponding pixel value and summing the results of the
          multiplications. If the correspoding pixel is outside the
          image then take the value of the center pixel instead.
     draw edited image on canvas using the modified Pixels
```

Convolution will be used in our application in order to apply the Blur and the Sharpness effects.

In order to make the effect more intense we will be running the algorithm multiple times, which means if for example we set the effect value to 3, it would mean we run the algorithm three times.

8.5.6.2 Blur Effect

Blurring an image means essentially smoothing out the difference in colors between adjacent pixels.

For the blurring effect we will apply the convolution algorithm using the smoothing kernel.

There are two variants of the blurring effect, the first called Box Blur which is the simplest of the two and its kernel is:

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

The second variant is the Gaussian Blur which preserves more details from the source image, and its kernel is:

$1/16$	$1/8$	$1/16$
$1/8$	$1/4$	$1/8$
$1/16$	$1/8$	$1/16$

Below is the resulting image from applying the blur effect:

Figure 12: Blur effect applied (Blur value = 5)

8.5.6.3 Sharpness Effect

Sharpening an image means basically the opposite of blurring, which mean emphasizing the difference in color between adjacent pixels.

For the Sharpness effect we will apply the convolution algorithm using the sharpness kernel.

The Kernel is as follows:

−1	−1	−1
−1	8	−1
−1	−1	−1

Below is the resulting image from applying the sharpness effect:

Figure 13: Sharpness effect applied (Sharpness value = 1)

8.5.7 Invert

Inverting an image means producing the photographic negative of said image, i.e. turning the dark colors light and the light colors dark and it's done simply by subtracting the source color value from 255.

The algorithm for inverting an image goes like this:

```
Invert:
     Pixels = get the image pixel data from canvas
     newPixels = the resulting data for the edited image
     for all pixels:
           newPixels[Red] = 255 - Pixels[Red]
           newPixels[Green] = 255 - Pixels[Green]
           newPixels[Blue] = 255 - Pixels[Blue]
     draw edited image on canvas using the modified Pixels
```

below is the resulting image of applying the inverting effect:

Figure 14: Invert effect applied

8.5.8 GrayScale

Grayscaling an image refers to the process of reducing the colors of an image to different shades of gray color.

There are many algorithms that can be used to achieve the grayscaling effect, but the basic concept of them is calculating a unified value for all color channels based on the source values. This value can either be a simple or a weighted average of the source values. We will use the simple average in our algorithm.

The algorithm for achieving the grayscaling effect goes like this:

```
Invert:
     Pixels = get the image pixel data from canvas
     newPixels = the resulting data for the edited image
     for all pixels:
          average = (Pixels[Red]+Pixels[Green]+Pixels[Blue])/3
          newPixels[Red] = average
          newPixels[Green] = average
          newPixels[Blue] = average
     draw edited image on canvas using the modified Pixels
```

below is the resulting image of applying the grayscaling effect:

Figure 15: Grayscale effect applied

8.5.9 Sepia

Sepia is the process of giving a grayscaled image warmth by toning it with a Sepia color, which is a rich light tone of the brown color.

Digitally sepia toning can be achieved by calculating the values of each of the red, green and blue channels using a weighted sum, where the wights are constants determined to give the color that special brown tone of its grayscaled version.

The algorithm to achieve the sepia effect is as follows:

```
Sepia:
     Pixels = get the image pixel data from canvas
     newPixels = the resulting data for the edited image
     for all pixels:
           red = Pixel[red]
           green = Pixel[green]
           blue = Pixel[blue]
           newPixels[red] = 0.393*red + 0.769*green + 0.189*blue
           newPixels[Green] = 0.349*red + 0.686*green + 0.168*blue
           newPixels[Blue] = 0.272*red + 0.534*green + 0.131*blue
     draw edited image on canvas using the modified Pixels
```

below is the resulting image of applying the sepia effect:

Figure 16: Sepia effect applied

8.5.10 Solarize

The Term Solarization refers photography to the process of color inversion in the case of overexposure.

In image processing, the solarization process is basically the same as the inverting process, with one main difference: the inversion of color is done only for colors that are above a certain threshold. So the algorithms for inversion and solarization are basically similar.

The algorithm to achieve the solarization effect is as follows:

```
Solarize:
     Pixels = get the image pixel data from canvas
     newPixels = the resulting data for the edited image
     for all pixels:
           if Pixels[red] > threshold:
                 newPixels[Red] = 255 – Pixels[Red]
           else:
                 newPixels[Red] = Pixels[Red]
           if Pixels[Green] > threshold:
                 newPixels[Green] = 255 – Pixels[Green]
           else:
                 newPixels[Green] = Pixels[Green]
           if Pixels[Blue] > threshold:
                 newPixels[Blue] = 255 – Pixels[Blue]
           else:
                 newPixels[Blue] = Pixels[Blue]
     draw edited image on canvas using the modified Pixels
```

below is the resulting image of applying the solarization effect:

Figure 17: Solarization effect applied

9. Results

The combination of all the adjustments and effects implemented in the web application gives the user a wide range of modifications to apply to images.

Here an example of combining the effects and adjustments in practice:

Source image

adjustments done:

- Brightness with value 24.
- Contrast with value 11.
- Saturation with value 2.
- Temperature with value -12.
- Blur with value 1.
- Sharpness with value 1.

Figure 18: Combined Adjustments applied

10. Conclusion

The optimizations made to Javascript in the recent years open a large potential to developers to create applications in the web that run solely on the browser with dependency on the environment that the user runs its system on, with no need to download or install any files on the user's computer and perform more and more complicated tasks. The thing that the majority of people thought it was impossible until lately.

Combined with Canvas-API introduced in Html5 opens room to more creativity from processing image and video files to creating games on the browser.

11. Glossary

- Cross-platform application: an application that can run on various operating systems.
- Platform-specific application: an application that meet the requirements of a particular operating system and cannot run on any other environment
- Moore's law: is the observation made by Gordon Moore in 1965 that the number of transistors in a dense integrated circuit and the computing and storage power doubles about every two years.
- UX: Acr. For User Experience
- UI: Acr. For User Interface
- API: is the acronym for Application Programming Interface, which is a software or a code intermediary that allows two applications to talk to each other.
- DOM: is a cross-platform and language-independent API for HTML and XML defines the logical structure of documents and the way a document is accessed and manipulated.
- Parser: is an interpreter component that breaks data into smaller elements for easy translation into another language.
- Object: it is one of JavaScript's data types. It is used to store various keyed collections and more complex entities.

- Function: a set of statements that takes an input, performs a task or calculates a value, and returns an output.
- Variable: a symbolic name associated with a value and whose associated value may be changed.
- Type: a characterization of variable to one of five categories: Integral, Floating Point, Character, Character String, and composite types,
- RGBs: is an additive color model in which the red, green, and blue primary colors of light are added together to reproduce all other colors of the spectrum.
- Pixel: from Picture Element, a pixel is the basic unit of programmable color on a computer display or in a computer image.
- Algorithm: a set of rules and instructions to be followed in order to solve a problem.
- Luminance vector: a vector with which the gray tone values is computed from RGBs color model. The more luminant a color is the brighter is it visually. It is estimated roughly as: < 0.3 R, 0.59 G, 0.11 B $>$.

12. Literature

- Bildverarbeitung – Band II des Standardwerks Computergrafik und Bildverarbeitung By Alfred Nischwitz, Max Fischer, Peter Haberäcker, Gudrun Socher.
  https://link.springer.com/book/10.1007/978-3-658-28705-4
  (Available Uni–Intern)
- Principles of Digital Image Processing: Core Algorithms (Undergraduate Topics in Computer Science) 2009th Edition by Wilhelm Burger, Mark James Burge.
  https://link.springer.com/book/10.1007/978-1-84800-195-4
- Computer Vision: Algorithms and Applications, 2nd edition by Richard Szeliski.

  https://link.springer.com/book/10.1007/978-3-030-34372-9
- https://webflow.com/blog/the-web-design-process-in-7-simple-steps
- https://www.tutorialstonight.com/html/history-of-html.php
- https://en.wikipedia.org/wiki/HTML
- https://en.wikipedia.org/wiki/HTML5#Features_and_APIs
- https://developer.mozilla.org/en-US/docs/Web/HTML
- https://html.spec.whatwg.org/multipage/canvas.html

- https://developer.mozilla.org/en-US/docs/Web/HTML/Element/canvas
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
- https://developer.mozilla.org/en-US/docs/Web/CSS
- https://en.wikipedia.org/wiki/CSS
- https://www.w3.org/TR/css-2021/
- https://www.w3.org/Style/CSS20/history.html
- https://en.wikipedia.org/wiki/JavaScript
- https://developer.mozilla.org/en-US/docs/Web/JavaScript
- https://www.freecodecamp.org/news/execution-context-how-javascript-works-behind-the-scenes/
- https://www.boardinfinity.com/blog/top-10-features-of-es6/
- https://developer.mozilla.org/en-US/docs/Web/API/ImageData
- https://en.wikipedia.org/wiki/RGB_color_model
- https://developer.mozilla.org/en-US/docs/Web/API/FileReader
- https://docs.opencv.org/3.4/d3/dc1/tutorial_basic_linear_transform.html
- https://www.allaboutcircuits.com/technical-articles/digital-image-processing-point-operations/
- https://www.scantips.com/lumin.html

- https://en.wikipedia.org/wiki/Convolution
- https://www.analyticssteps.com/blogs/how-does-basic-convolution-work-image-processing
- https://homepages.inf.ed.ac.uk/rbf/HIPR2/invert.htm
- https://en.wikipedia.org/wiki/Grayscale
- http://leware.net/photo/blogSepia.html
- https://digicreation.blogspot.com/2008/10/solarization-of-photographs.html

The used literature and internet sources as they stand in 20.05.2022