Paint 2.0 - the Photoshop alternative

Welcome to **Paint 2.0 - the Photoshop alternative** project specification.
Please read the instruction carefully :-)

Table of contents

Intro

Photoshop is the market-leading tool for images editing and websites design, we want to compete with this bloated, over-complicated, overpriced software by creating our own alternative Paint software, we will call it “Paint 2.0”

As a developer, your task is to develop this “Paint 2.0” software based on JS and HTML 5 canvas.

Our great design team worked really hard and created a beautiful UX for this new software, they delivered it as a ZIP file, containing HTML, CSS and a base server files.
All you’ll have to do is to develop the app using JS.

Here is a link to the project seed files
https://drive.google.com/file/d/18aXQ96ezCBGv3blxKN6Byr63-CjqIhb0/view?usp=sharing

The files

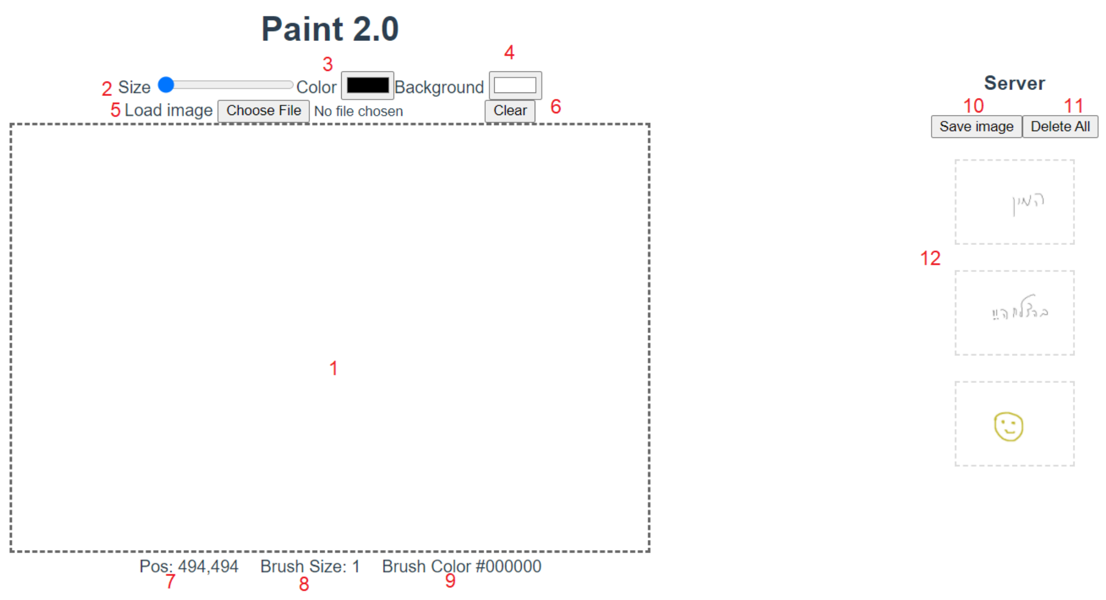
1. **client/index.html** - the main project file, you should open this file in a Chrome browser
2. **client/style.css -** all your CSS code must sit here
3. **server/server.js** - the main server file including the base code to start it
4. **server/images.json** - this file contains a JSON of images, you should control this file using the server, and this will be your database.
5. **server/packages.json** - your server packages file, feel free to add more packages

Our glorious UX

We had recruited one of the top web designers for this project, here is the latest version of his slick and modern UX design, as you can see he decided to use a modern simple design.

We have marked each of the elements in red number and we are going to refer to the element by their number in the next section.

Screenshot of the latest design



There is a CSS file attached to the project, we are not expecting you to change this perfect design, but if you need some CSS changes to support your code, feel free to change the CSS file.
Although we trust your UX instincts, our designer is very sensitive and fragile, so changing the design will hurt his feelings, please don’t waste your time improving his design, change the CSS only if you need it to support your code.

Elements behavior

1. The canvas: here the user will be able to draw, it should work like any other paint software when the user left click on his mouse and move it, a line should appear. When the user move the mouse without left-clicking no line will appear.
   The line color and size will be set by elements 2 and 3.
2. Line size: the line size will be set by this input, this should define only **future** lines, so changing it will not affect lines that already been drawn on the canvas.
3. Line color: the line color will be defined by this input, this should define only **future** lines, so changing it will not affect lines that already been drawn on the canvas.

4. Changing the background, this will **reset** the canvas to the specific color you want the background to be, you shouldn't care for an element that already been drawn or imported, and it's definitely OK if they are deleted along this process (or not).
5. Load image: this will load an image into the canvas, our research department concluded that the user doesn't need any format other than **png**, so no need to support other formats, and also no need to do any validation for the input file, we are trusting our users to use this software in the right way.
6. This will clear the canvas to a new clean state
7. This will show the mouse position relative to the canvas, as an integer number, so please remove the decimal point if you have any.
   The first number is the X (distance from the **left** border of the canvas)
   The second number is the Y (distance from the **top** border of the canvas)
8. The RGB value of the color input (element number **2**)
9. The size value of the size input (element number **3**)
10. This will store the image to our server
11. This will clear all the images in the server
12. Here you will show list of images from the server, clicking on an image will load it into the canvas.

Technical info

1. This design has been tested using the latest Chrome browser, the inputs may not look the same in other browsers, so please do your development using Chrome browser.
2. You may use **any** IDE you are comfortable with.

3. Client:
   a. You must use Angular.js or React or Vue.js framework to run the client

4. Server
   a. As your server side you must you node.js with express
   b. You have a server.js file which contains the base server
   ```
   const express = require('express');
   const app = express();
   const cors = require('cors');
   const bodyParser = require('body-parser');
   app.use(bodyParser.json({limit: '50mb'}));
   app.use(cors());
   ```

   This will eliminate the problems with CORS and storing big json files

   c. In the server side you have a file called images.json, we expect you to store the images to this file, load it from it, or delete all it content

The time machine

To show you how we want the app to behave, we developed a time machine and took a screen recording of the future developed app.

Here is a record from the future of using the app:
https://youtu.be/BcKXmFtP2FE

Please notice that although we are capable of traveling through time, we couldn't capture the color picker, file upload, and file saving dialogs, so every time the mouse is moving aimlessly on the screen, we are interacting with the dialog.
These dialogs are Chrome native dialogs, and you don't need to develop them.

When in this recording we are clicking on the `save` link, the next things happen:

1. A save file dialog opens and we are saving the file to the desktop
2. We clear the canvas
3. We click on the "Load image" button and load the file back into the canvas