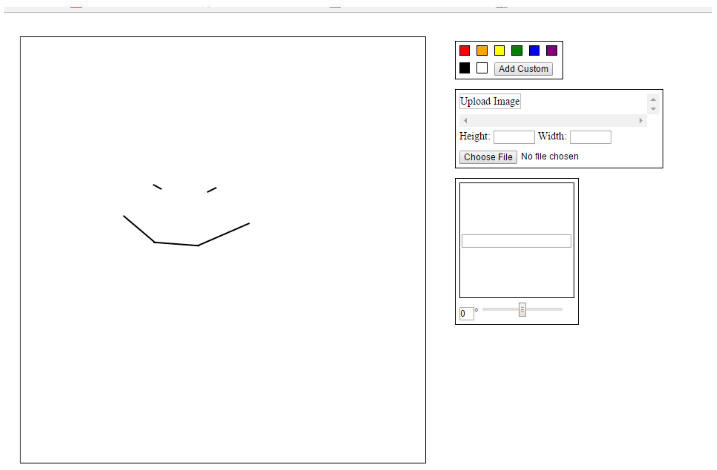
Bo Lenhardt – Net-Centric Computing – First JavaScript – 3/13/17

Introduction

We were supposed to use JavaScript to do something “fancy” to manipulate HTML or styling. My version of this eventually turned into a fairly fleshed-out basic drawing tool.

Description



The giant square is a canvas that lets you draw lines of a selected color, upload and drag an image onto it, and add text at an angle in the selected color.

Attempts to use `ondrag` and other drag-based events were buggy at best, so eventually the canvas registered `onmousedown` and `onmouseup` events to draw lines between two points.

```
(index.php)

<div id="canvas"
onmousedown="canvasDragStart(event)"
onmouseup="canvasDragEnd(event)">
</div>
```

Then, each of the tools for drawing on the canvas were added, in a plain div for formatting purposes. They were grouped, aptly, in “`colorDiv`”, “`imageAdderDiv`”, and “`textAdderDiv`”.

Drawing a line:

The canvas’s `onmousedown` simply calculates the spot that the mouse went down, and records it into `x1` and `y1` for comparison in the later called method, as well as recording that the page was in the process of dragging to reduce bugs.

The canvas’s `onmouseup` calculates what a line between the recorded spots would look like. Possibly the hardest part of designing this entire page was having it place a `div` into `canvas` that would be angled and placed to look like that line:

Bo Lenhardt – Net-Centric Computing – First JavaScript – 3/13/17

(Script.js)

```
function canvasDragEnd(event)

{

      if (!dragging)

          return;

      /*

      Here it calculates the length, angle, top-left corner, and offset from
      rotation, based on recorded location and the current location.
      Unfortunately, with this part the method was too big for one page.

      */

      canvas.innerHTML +=

              "<div style='position:absolute; " +

              "top:" + top + "px; " +

              "left:" + left + "px; " +

              "height: 0px;" +

              "width:" + length + "px; " +

              "transform: rotate(" + theta + "rad);" +

              "border: 1px solid " + color + ";" +

              "'></div>";


      dragging = false;

}
```

Note: I made an `onload` that set some default global values, including a reference to `canvas` used in many functions

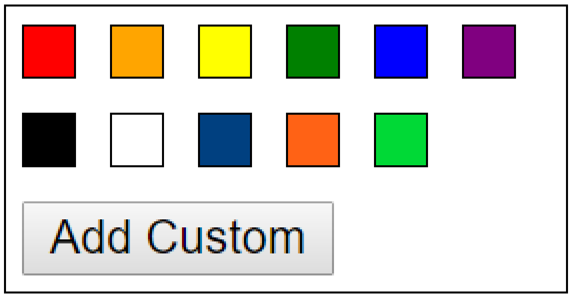
Changing the color:

The `colorDiv` creation ended up using PHP when I realized that it should come with some colors by default, although the ability to add custom colors was also one of the hardest parts of creating this page.

```
<?php
$colors = array("Red", "Orange", "Yellow", "Green", "Blue",
"Purple", "Black", "White");

$count = count($colors);

for ($i = 0; $i < $count; $i++)
{
    echo("<p id='color$i' "
    ."class='color' "
    ."onclick='colorclick(event)'"
    ."style='background-color: $colors[$i];'"
    ."></p>");
}
echo "<input id='customColorButton' type='button'
onclick='clickCustomColor()' value='Add Custom'></input>";

echo "<input id='customColorChooser' type='color'
oninput='addCustomColor(event)' ></input>";
?>
```

```
(Main.css)
.color
{
    width: 14px;
    height: 14px;
    border: 1px solid black;
}
```

(This color input is not visible to the user. The button simply simulates a click on the invisible input to open a color selection dialog to add new colors, as shown below.)



`#colorDiv` with three custom colors added

In `mainScript.js`, it has a variable `color` that represents the currently selected color, which is changed in `colorclick(event)`:

```
(Script.js)

var color;

function colorclick(event)

{

    color = event.target.style.backgroundColor;

    document.getElementById("textBox").style.color = color;

}
```

(Please continue to next page. Spacing is difficult with long functions.)

When you click on the “Add Custom” button, it simply simulates a click on a hidden color input styled out of view. The real magic happens when that input completes:

```
(mainScript.js)
function clickCustomColor()
{
    document.getElementById("customColorChooser").click();
}

function addCustomColor(event)//called by oninput event of color input
{
    //declare necessary variables
    var colorDiv = document.getElementById("colorDiv");
    var numElements = colorDiv.childElementCount;

    //create a new color to add to colordiv
    var newElement = document.createElement("p");
    newElement.id = "color" + (numElements - 2);
    newElement.className = "color";
    newElement.setAttribute("onclick", "colorclick(event)");
    newElement.style.backgroundColor = event.target.value;

    //insert into colordiv, before button and hidden color input
    colorDiv.insertBefore(newElement, colorDiv.childNodes[numElements - 1]);

    //increase height of colorDiv if necessary
    if((colorDiv.childElementCount - 2) % 6 === 3)
    {
        var previousHeight = colorDiv.getBoundingClientRect().height;
         colorDiv.style.height = (previousHeight + 25) + "px";
    }
}
```

Adding text:

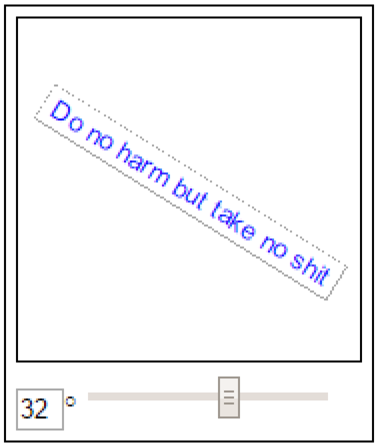
Visually the text tool comes last, but it was created after the color tool, so it will be discussed next.

Making a text box able to be dragged onto the canvas wasn't very hard. You only have to set the `textbox`'s `draggable` attribute to `true` and have its `ondragend` place a new `label` (or other simple text holder) onto the canvas. Making this text appear at an angle was a similar process to creating a line, but only the offset of its top-left corner needed to be calculated due to rotation. Having only one mouse location and preset dimensions made styling it simpler.

```
function textDragEnd(event)
{
    var box = event.target;
    var boxRect = box.getBoundingClientRect();
    var canvasRect = canvas.getBoundingClientRect();

    var theta = document.getElementById("textAngle").value * Math.PI / 180;

    var top = event.clientY - canvasRect.top - (boxRect.height / 2);
    var left = event.clientX - canvasRect.left - (boxRect.width / 2);

    canvas.innerHTML +=
            "<label style='position:absolute; "+
            " text-align:center;" +
            " top:" + top + "px;" +
            " left:" + left + "px;" +
            " height:" + boxRect.height + "px;" +
            " width:" + boxRect.width + "px;" +
            " transform: rotate(" + theta + "rad);" +
            " line-height: " + (boxRect.height) + "px;" +
            " color:" + color + ";" +
            " '>" + box.value + "</label>";
}
```

However, in order to make it appear at an angle, the user needs some way of setting its angle. Some research into `input` types gave me a slider, and I decided the user also needed a text input to set specific angles and see what angle they were actually setting it to. Having their `oninput` methods synchronize was more difficult, but eventually I got an elegant solution that both the slider and the small text box could call:

```
(mainScript.js)
function textAngleChanged(event)
{
    var num = event.target.value;
    document.getElementById("textAngle").value = num;
    document.getElementById("textAngleBox").value = num;
    document.getElementById("textBox").style.transform =
            "rotate(" + num + "deg)”;
}
```



Adding images:

This tool allows the user to upload an image and drag it over to the canvas. Most of the code for uploading user-end files was initially taken directly from the internet, but eventually I altered the code enough for my purposes that I am willing to take some credit for it.

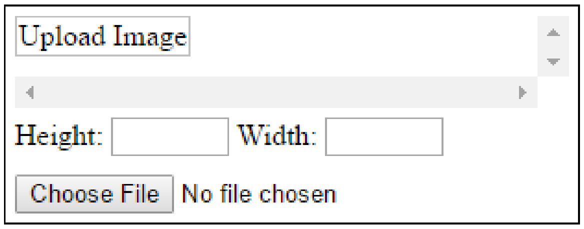
One problem that I saw coming ahead of time was that uploaded images would be very different sizes, which could make the box go out of proportion. So, the blank `img` users upload to was placed inside a scrollable div that maxed out at 300x300 pixels, with a 50x50 minimum. Users could manually resize it via text boxes - I now think I should have also given the option of proportionally scaling it, but hindsight is 20/20. More research gave me a `file` input so the user could select files, and let me alter it so that it only accepted images.

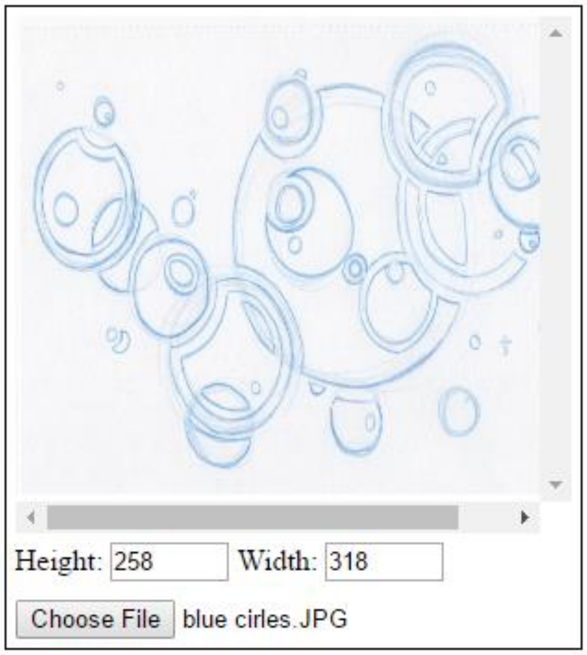
```
(index.php)
<div id="imageAdderDiv">
    <div id="imageDiv">
        <img id="image" alt="Upload Image"
                ondragStart="imageDragStart(event)"
                ondragend="imageDragEnd(event)"/>
    </div>

    <div id="imageDimensionsDiv">
        <span> Height: </span>
        <input id="imageHeight" type="number"
                oninput="imageDimensionsChanged(event)"/>

        <span> Width: </span>
        <input id="imageWidth" type="number"
                oninput="imageDimensionsChanged(event)"/>
    </div>

    <input id="imageButton" type="file"
            onchange="fileChosen(event)"
            accept="image/gif, image/jpeg, image/png"/>
</div>
```

```
(Main.css)
#imageDiv
{
    max-height: 300px;
    max-width: 300px;
    min-height: 50px;
    min-width: 50px;

    overflow: scroll;
}

#imageDimensionsDiv
{
    width: 300px;
}

#imageDimensionsDiv
input
{
    width: 20%;
}
```

The code in `fileChosen(event)` as seen above is the part that was taken from an outside source, then altered.

```
(mainScript.js)
function fileChosen(event)
{
    var image = document.getElementById("image");
    //reset sizing for new image
    image.style.height = "auto";
    image.style.width = "auto";

    //mostly copied from the internet. uploads an image
    if (event.target.files && event.target.files[0])
    {
            var reader = new FileReader();

            reader.onload = function (e)
            {
                var image = document.getElementById("image");

                image.setAttribute('src', e.target.result);

                var rect = image.getBoundingClientRect();

                document.getElementById("imageHeight").value = rect.height;
                document.getElementById("imageWidth").value = rect.width;

            };
            reader.readAsDataURL(event.target.files[0]);
    }
}
```

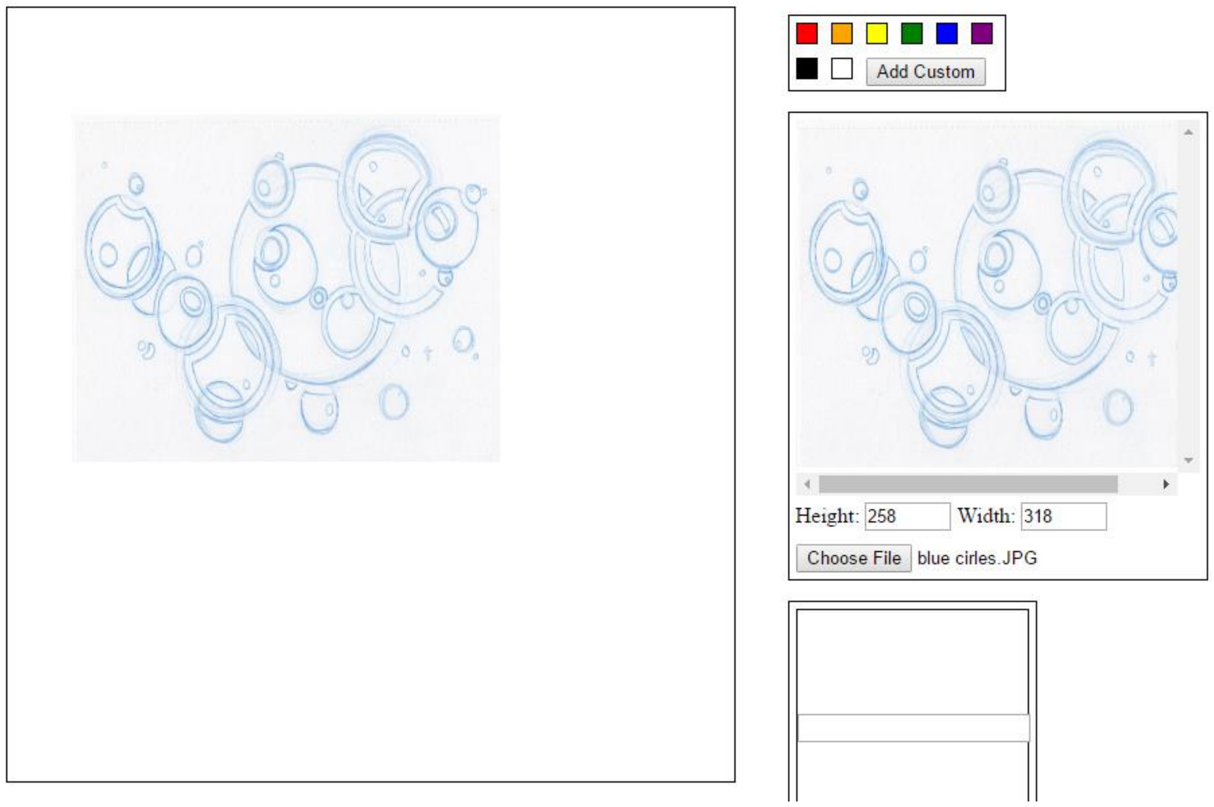




Then `#image`'s `ondragend` event allows the user to place an image on the canvas. By this point, I had learned about how to use the `document.createElement()` method:

```
(mainScript.js)
function imageDragEnd(event)
{
    var rect = canvas.getBoundingClientRect();
    var image = event.target;

    var newImage = document.createElement("img");
    newImage.src = image.src;
    newImage.style.position = "absolute";
    newImage.style.height = image.style.height;
    newImage.style.width = image.style.width;
    newImage.style.top = (event.clientY - rect.top - y1) + "px";
    newImage.style.left = (event.clientX - rect.left - x1) + "px";

    canvas.appendChild(newImage);
}
```

Bo Lenhardt – Net-Centric Computing – First JavaScript – 3/13/17



Conclusion

This was an incredibly challenging project! I ended up using JavaScript to change styling only on objects that I had already created, but a lot of code went into adding elements onto a page. However, I still gained a great understanding of JavaScript as a tool for manipulating webpages. I am sorry that I turned this in late due to technical and personal difficulties.

Code

<u>index.php</u>

```
<!DOCTYPE html>

<html>

    <head>

        <title>TODO supply a title</title>

        <link rel="stylesheet" type="text/css"
href="CSS/Main.css">

        <script src="Script/mainScript.js"></script>

    </head>

    <body onload="init()">

        <div id="canvas"

             onmousedown="canvasDragStart(event)"

             onmouseup="canvasDragEnd(event)">


        </div>

        <div id="adderDiv">

            <div id="colorDiv">

                <?php

                $colors = array("Red", "Orange", "Yellow",
"Green", "Blue", "Purple", "Black", "White");


                $count = count($colors);


                for ($i = 0; $i < $count; $i++)

                {

                    echo("<p id='color$i' "

                            . "class='color' "

                            . "onclick='colorclick(event)' "
                            . "style='background-color:
                             $colors[$i];'"
                            . "></p>");
                }
                echo "<input id='customColorButton'
type='button' onclick='clickCustomColor()' value='Add
Custom'></input>";


                echo "<input id='customColorChooser'
type='color' oninput='addCustomColor(event)'></input>";
                ?>
            </div>
            <div id="imageAdderDiv">
                <div id="imageDiv">
                    <img id="image" alt="Upload Image"
ondragStart="imageDragStart(event)"
ondragend="imageDragEnd(event)"/>
                </div>


                <div id="imageDimensionsDiv">
                    <span> Height: </span>
                    <input id="imageHeight" type="number"
oninput="imageDimensionsChanged(event)"/>
                    <span> Width: </span>
                    <input id="imageWidth" type="number"
oninput="imageDimensionsChanged(event)"/>
                </div>


                <input id="imageButton" type="file"
onchange="fileChosen(event)" accept="image/gif, image/jpeg,
image/png"/>
            </div>
            <div id="textAdderDiv">

                <div id="textBoxDiv">

                    <input id="textBox" type="text" draggable =
"true" ondragend="textDragEnd(event)"></input>

                </div>

                <div id="textAngleDiv">

                    <input id="textAngleBox" type="text"
columns="2" text="0" value="0"
oninput="textAngleChanged(event)">&deg</input>

                    <input id="textAngle" type="range" min="-
180" max="180" value="0"
oninput="textAngleChanged(event)"></input>

                </div>

            </div>

        </div>

    </body>

</html>
```

```
mainScript.js
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

var canvas;
var dragging;
var x1;
var y1;

var color;

function init()
{
    canvas = document.getElementById("canvas");
    dragging = false;
    x1 = -1;
    y1 = -1;
    color = "black";
}

//drawing lines
function canvasDragStart(event)
{
    y1 = event.clientY;
    x1 = event.clientX;
    var rect = canvas.getBoundingClientRect();
    y1 -= rect.top;
    x1 -= rect.left;
    dragging = true;
}
function canvasDragEnd(event)
{
    if (!dragging)
        return;

    var y = event.clientY;
    var x = event.clientX;

    var rect = canvas.getBoundingClientRect();
    y -= rect.top;
    x -= rect.left;

    var height = y - y1;
    var width = x - x1;

    var length = hyp(width, height);
    var theta = Math.atan(height / width);
    var top = y1 + (length / 2 * Math.sin(theta));
    var left = x1 - (length / 2 * (1 - Math.cos(theta)));

    //arctan will only give angles from left-to-right
    //need to readjust if line goes right-to-left
    if (x < x1)
    {
        top = y + (length / 2 * Math.sin(theta));
        left = x - (length / 2 * (1 - Math.cos(theta)));
    }
    canvas.innerHTML +=
            "<div style='position:absolute; " +
            "top:" + top + "px; " +
            "left:" + left + "px; " +
            "height: 0px;" +
            "width:" + length + "px; " +
            "transform: rotate(" + theta + "rad);" +
            "border: 1px solid " + color + ";" +
            "'></div>";

    dragging = false;
}

function colorclick(event)
{
    color = event.target.style.backgroundColor;
    document.getElementById("textBox").style.color = color;
}

function clickCustomColor()
{
    document.getElementById("customColorChooser").click();
}
```

```
function addCustomColor(event)
{
    //declare necessary variables
    var colorDiv = document.getElementById("colorDiv");
    var numElements = colorDiv.childElementCount;

    //create a new color paragraph to add to colordiv
    var newElement = document.createElement("p");
    newElement.id = "color" + (numElements - 2);
    newElement.className = "color";
    newElement.setAttribute("onclick", "colorclick(event)");
    newElement.style.backgroundColor = event.target.value;

    //insert into colordiv, before button and hidden color input
    colorDiv.insertBefore(newElement,
        colorDiv.childNodes[numElements - 1]);

    //increase height of colorDiv if necessary
    if((colorDiv.childElementCount - 2) % 6 === 3)
    {
        var previousHeight =
            colorDiv.getBoundingClientRect().height;
        colorDiv.style.height = (previousHeight + 25) + "px";
    }

}

//adding text
```

```
function textAngleChanged(event)
{
    var num = event.target.value;
    document.getElementById("textAngle").value = num;
    document.getElementById("textAngleBox").value = num;
    document.getElementById("textBox").style.transform =
        "rotate(" + num + "deg)";
}

function textDragEnd(event)
{
    var box = event.target;
    var boxRect = box.getBoundingClientRect();
    var canvasRect = canvas.getBoundingClientRect();

    var theta = document.getElementById("textAngle").value *
Math.PI / 180;

    var top = event.clientY - canvasRect.top –
        (boxRect.height / 2);
    var left = event.clientX - canvasRect.left –
        (boxRect.width / 2);

    canvas.innerHTML +=
            "<label style='position:absolute; "+
            " text-align: center;" +
            " top:" + top + "px;" +
            " left:" + left + "px;" +
            " height:" + boxRect.height + "px;" +
            " width:" + boxRect.width + "px;" +
            " transform: rotate(" + theta + "rad);" +
            " line-height: " + (boxRect.height) + "px;" +
            " color:" + color + ";" +
            " '>" + box.value + "</label>";
}

//adding images
function fileChosen(event)
{
    var image = document.getElementById("image");
    image.style.height = "auto";
    image.style.width = "auto";

    //mostly copied from the internet. uploads an image
    if (event.target.files && event.target.files[0])
    {
            var reader = new FileReader();

            reader.onload = function (e)
            {
                var image = document.getElementById("image");

                image.setAttribute('src', e.target.result);

                var rect = image.getBoundingClientRect();
                document.getElementById("imageHeight").value =
                    rect.height;
                document.getElementById("imageWidth").value =
                    rect.width;

            };

            reader.readAsDataURL(event.target.files[0]);
    }
}

function imageDimensionsChanged()
{
    var newHeight =
        document.getElementById("imageHeight").value;
    var newWidth = document.getElementById("imageWidth").value;

    var image = document.getElementById("image");

    image.style.height = newHeight + "px";
    image.style.width = newWidth + "px";

}

function imageDragStart(event)
{
    var rect = event.target.getBoundingClientRect();
    y1 = event.clientY - rect.top;
    x1 = event.clientX - rect.left;
}


function imageDragEnd(event)
{
    var rect = canvas.getBoundingClientRect();
    var image = event.target;


    var newImage = document.createElement("img");
    newImage.src = image.src;
    newImage.style.position = "absolute";
    newImage.style.height = image.style.height;
    newImage.style.width = image.style.width;
    newImage.style.top = (event.clientY - rect.top - y1) + "px";
    newImage.style.left = (event.clientX - rect.left - x1) +
"px";


    canvas.appendChild(newImage);
}


//helper functions
function hyp(num1, num2)
{
    return Math.sqrt(num1 * num1 + num2 * num2);
}
```

Main.css

```
/*
    Created on : Mar 3, 2017, 8:12:37 PM
    Author     : Bo
*/

#canvas
{
    border: 1px solid black;
    position: absolute;
    left: 7%;
    top: 5%;
    width: 40%;
    height: 90%;
    overflow: hidden;
}

#adderDiv
{
    position: absolute;
    top: 6%;
    left: 50%;
}

#adderDiv > *
{
    position: relative;
    clear:left;
    margin-bottom: 15px;
}

#colorDiv
{
    width: 160px;
    height: 55px;
    border: 1px solid black;
    overflow: hidden;
}

#colorDiv  > *
{
    margin: 5px;
    float: left;
}

.color
{
    width: 14px;
    height: 14px;
    border: 1px solid black;
}

/*hides color chooser out of sight*/
#customColorChooser
{
    clear: left;
```

```
}

#textAdderDiv
{
    width: fit-content;
    height: fit-content;
    border: 1px solid black;
}

#textBoxDiv
{
    margin: 5px;
    width: 171px;
    height: 171px;

    border: 1px solid black;
}

#textBox
{
    display: block;
    margin: auto;
    margin-top: 45%;
    text-align: center;
}

#textAngleDiv
{
    margin: 5px;
}

#textAngleBox
{
    width: 20px;
}

#textAngle
{
    width: 120px;
}

#imageAdderDiv
{

    border: 1px solid black;
}

#imageAdderDiv > *
{
    margin: 5px;
}

#imageDiv
{
    max-height: 300px;
    max-width: 300px;
    min-height: 50px;
    min-width: 50px;

    overflow: scroll;
}

#imageDimensionsDiv
{
    width: 300px;
}

#imageDimensionsDiv input
{
    width: 20%;
}
```