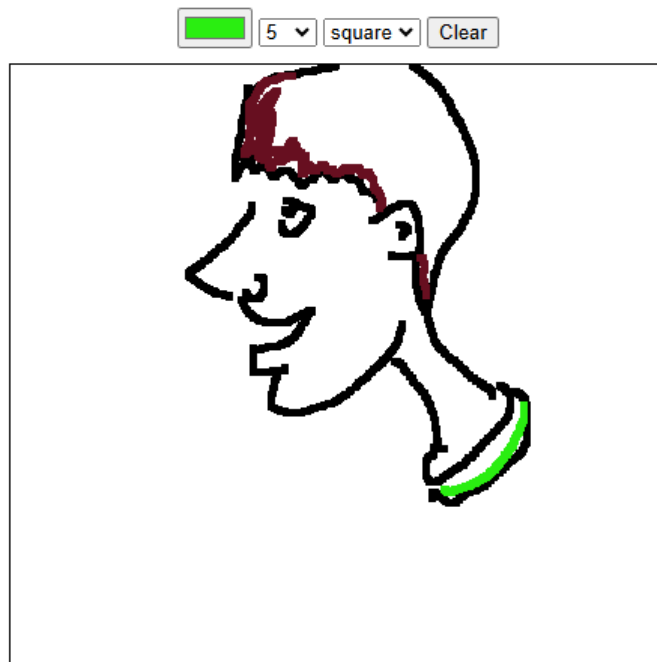


HTML Canvas

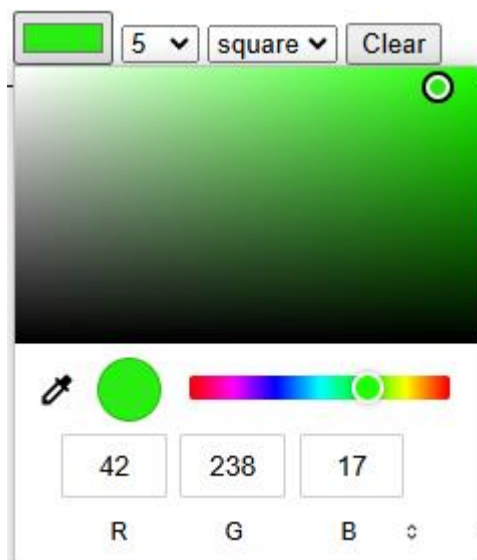
Your team task is to build the **mister-canvas** painting program:

Mister Canvas

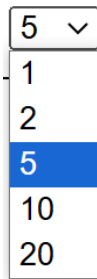


Initial Features

- User can select a brush color – default is black



- User can select a brush size (1, 2, 5, 10, 20) – default is 5

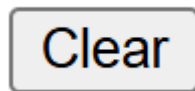


TIP: Drawing is done by filling a rect with the specified size

- User can download the image drawn on the canvas

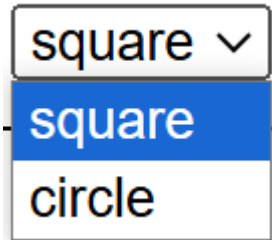
[Download](#)

- User can clear the canvas



User can select a brush shape: square or circle

TIP: Refactor so drawing is done by filling a rect or an arc with the specified size



- Canvas should be ~90% of screen width

TIP: use a `resizeCanvas` function:

```
function resizeCanvas() {  
  const elContainer = document.querySelector('.canvas-container')  
  gElCanvas.width = elContainer.clientWidth  
}
```

Data Model

```
var gElCanvas
var gCtx
var gIsMouseDown = false
var gBrush = {
  color: 'black',
  size: 5,
  shape: 'square'
}
```

Canvas

```
<div class="canvas-container">
  <canvas width="400" height="400"
    onmousedown="onDown(event)"
    onmouseup="onUp()"
    onmousemove="onDraw(event)">
  </canvas>
</div>
```

Controller functions

Here is an initial list of functions

```
> function onInit() { ...  
  }  
  
> function onDown(ev) { ...  
  }  
  
> function onUp() { ...  
  }  
  
> function onDraw(ev) { ...  
  }  
  
> function onDownloadCanvas(ellink) { ...  
  }  
  
> function onSetColor(color) { ...  
  }  
  
> function onSetSize(size) { ...  
  }  
  
> function onSetShape(shape) { ...  
  }  
  
> function onClearCanvas() { ...  
  }
```

Feature: Selecting background from user device

User can select an image as the background of the canvas, the image should cover the canvas, resize the canvas height as needed

Background photo: No file chosen

TIP: Add the needed input in the HTML and the functions:

```
function onImgInput(ev) {
  loadImageFromInput(ev, renderImg)
}

function loadImageFromInput(ev, onImageReady) {
  document.querySelector('.share-container').innerHTML = ''
  const reader = new FileReader()

  reader.onload = function (event) {
    const img = new Image()
    img.onload = () => {
      onImageReady(img)
    }
    img.src = event.target.result
  }
  reader.readAsDataURL(ev.target.files[0])
}

function renderImg(img) {
  gElCanvas.height = (img.naturalHeight / img.naturalWidth) * gElCanvas.width
  gCtx.drawImage(img, 0, 0, gElCanvas.width, gElCanvas.height)
}
```

Feature: Upload to the cloud

Allow the user to upload and share the uploaded picture to Facebook

Upload to Cloud

TIP: Add the needed form in the HTML and the function:

```
function onUploadImg(ev) {
  ev.preventDefault()
  const canvasData = gElCanvas.toDataURL('image/jpeg')

  // After a succesful upload, allow the user to share on Facebook
  function onSuccess(uploadedImgUrl) { ... }
  uploadImg(canvasData, onSuccess)
}
```

Feature: Share on Facebook

After uploading the drawn picture, user can share the uploaded picture to Facebook

Share on Facebook

TIP: the onSuccess function:

```
// After a succesful upload, allow the user to share on Facebook
function onSuccess(uploadedImgUrl) {
  const encodedUploadedImgUrl = encodeURIComponent(uploadedImgUrl)
  document.querySelector('.share-container').innerHTML = `
  <button class="btn-facebook" target="_blank"
```

```
        onclick="window.open('https://www.facebook.com/sharer/sharer.php?u=${encodedUploadedImgUrl}&t=${encodedUploadedImgUrl}')">
            Share on Facebook
        </button>`
    }
```

Feature: touch events

Add support for touch events and check from mobile

```
<div class="canvas-container">
    <canvas width="400" height="400"
        onmousedown="onDown(event)"
        ontouchstart="onDown(event)"
        onmouseup="onUp()"
        ontouchend="onUp()"
        onmousemove="onDraw(event)"
        ontouchmove="onDraw(event)"
    >
    </canvas>
</div>

// Add the function and use it in onDown and onMove
function getEvPos(ev) {}
```

Feature: Image drawing

- User can select a small image, when clicking the canvas it is drawn

Select an image:



- When image is selected, it looks like that:

Select an image:



- If a selected image is clicked it gets unselected, also when color/size/brush are changed, image gets unselected and user goes back to drawing mode

TIP: add selectedImg to the gBrush object, use it in the onDown() function

TIP: when there is a selectedImg on the gBrush object, do nothing in the onMove

TIP: add the following controller functions:

```
function onSelectImg(elImg) { ...  
}  
  
function unSelectImg() { ...  
}
```

Feature: Saving Pics

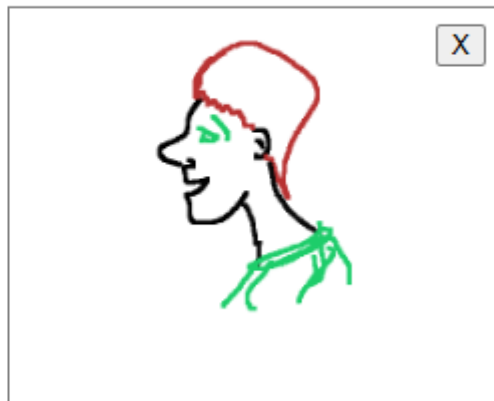
Allow the user to save his pics to localStorage

Save Picture

The pics are presented on the page.

The user can remove the pic or draw it on the canvas (on click)

Saved Pictures



TIP: add the following controller functions:

```
function renderPics() { ...  
}  
  
function onRemovePic(picId) { ...  
}  
  
function onSelectPic(picId) { ...  
}
```

TIP: pic.service should have the following functions:

TIP: It is ok to take something like a car.service and quickly refactor it

JS pic.service.js ×

ex-solution > js > services > JS pic.service.js > ...

```
1  'use strict'
2
3  const STORAGE_KEY = 'picDB'
4
5  var gPics = loadFromStorage(STORAGE_KEY) || []
6
7  > function getPics() { ...
9  }
10
11 > function removePic(picId) { ...
15   }
16
17 > function addPic(data) { ...
22   }
23
24 > function getPicById(picId) { ...
27   }
28
29 > function _createPic(data) { ...
35   }
36
37 > function _savePicsToStorage() { ...
39   }
```


Mister Canvas

 5 ▾ square ▾ Clear



Background photo: No file chosen

Select an image:



[Download](#)

Saved Pictures

