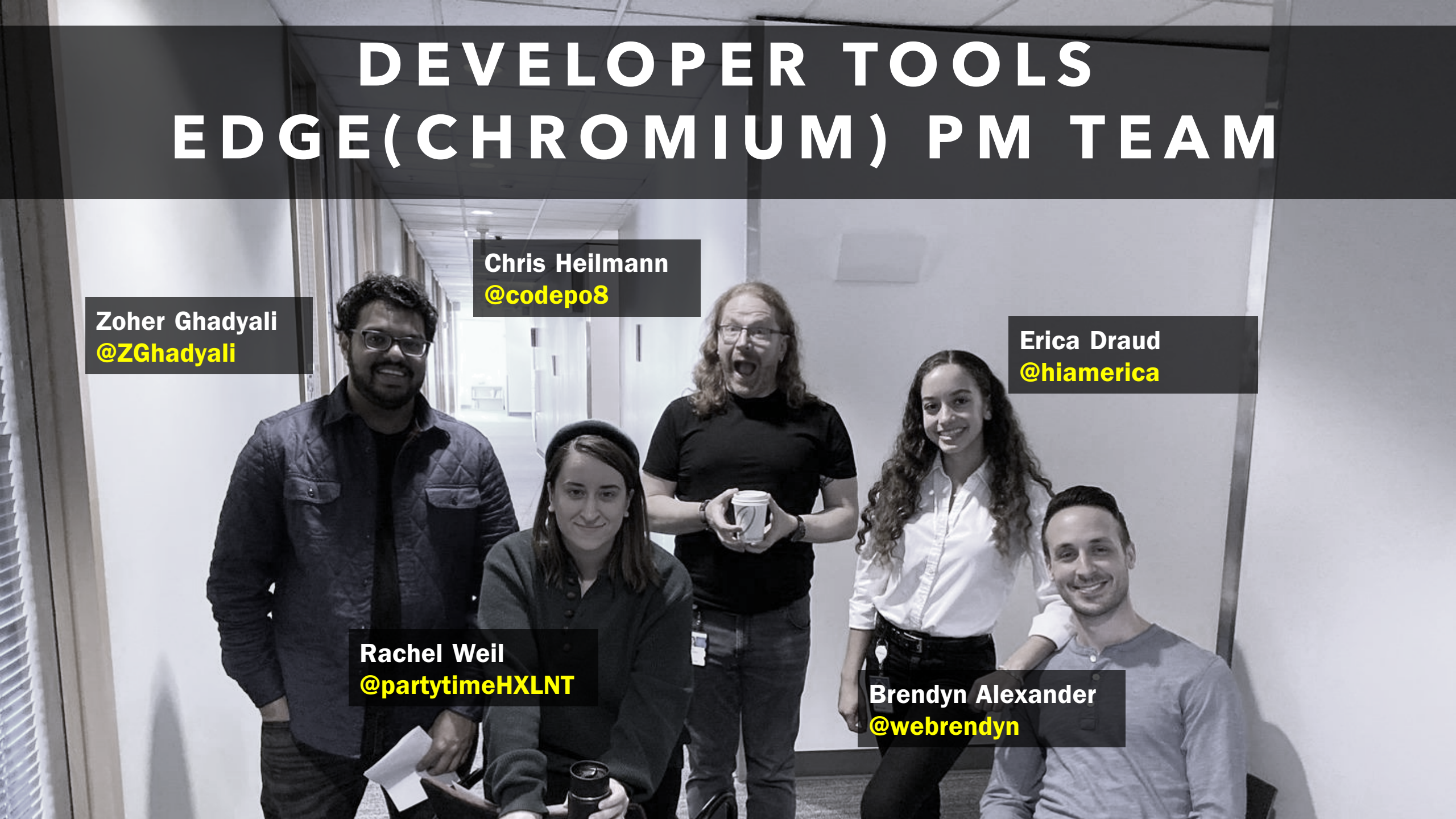
IMAGES, PIXELS, CANVAS, TIGERS AND BEARS...

DEVELOPER TOOLS
EDGE(CHROMIUM) PM TEAM



NERD, GEEK, FEARLESS CREATOR...


SOFTWARE IS ALWAYS PEEKING UNDER THE HOOD...

WHEN YOU SEE AN IMAGE, I SEE PIXELS AND DATA FORMATS...

GETTING AN IMAGE INTO THE BROWSER...

UPLOAD FORM

DRAG AND DROP

COPY AND PASTE

https://christianheilmann.com/2020/03/20/fun-with-browsers-how-to-get-an-image-into-the-current-page/

THE HTML

```
<div id="container">
  <h1>Getting an image into the browser</h1>
  <p>Drag and Drop and image, paste it, or use the upload bar below</p>
  <div>
      <input id="getfile" type="file" />
      <label for="getfile">Upload an image</label>
  </div>
  <div id="imagecontainer">
  </div>
  <output></output>
</div>
```

FILE UPLOAD JAVASCRIPT

```
/* Image from Upload */
const fileinput = document.querySelector('#getfile');
const imageFromUpload = (e) => {
  var file = e.target.files[0];
  loadImage(window.URL.createObjectURL(file), file.name);
  e.preventDefault();
}
fileinput.addEventListener('change', imageFromUpload, false);
```

DRAG AND DROP JAVASCRIPT

```
/* Image from Drag and Drop */
const imageFromDrop = (e) => {
  var file = e.dataTransfer.files[0];
  loadImage(window.URL.createObjectURL(file), file.name);
  e.preventDefault();
}
container.addEventListener('drop', imageFromDrop, false);
// Override the normal drag and drop behaviour
container.addEventListener('dragover', (ev) => {
  ev.preventDefault();
}, false);
```

COPY AND PASTE JAVASCRIPT

```
/* Image from Clipboard */
const getClipboardImage = (ev) => {
  let items = ev.clipboardData.items;
  for (var i = 0; i < items.length; i++) {
    if (items[i].type.indexOf('image') !== -1) {
      var blob = items[i].getAsFile();
      loadImage(window.URL.createObjectURL(blob));
      break;
    }
  }
}
window.addEventListener('paste', getClipboardImage, false);
```

SHOWING THE IMAGE

```
const output = document.querySelector('output');
const imagecontainer = document.querySelector('#imagecontainer');

/* Show the image once we have it */
const loadImage = (file, name) => {
  if (name) {
    output.innerText = 'Filename: ' + name;
  }
  var img = new Image();
  img.src = file;
  img.onload = function() {
    imagecontainer.appendChild(img);
  };
}
```

THIS IS BORING

UNLOCKING THE DATA IN THE IMAGE

CANVAS

Super basic in-browser painting mechanism

Built for speed, not for comfort

Powerful, once you get it

Confusing and weird till you get there

Element: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/canvas
API: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API

ETCH-A-SKETCH IN YOUR DOCUMENT

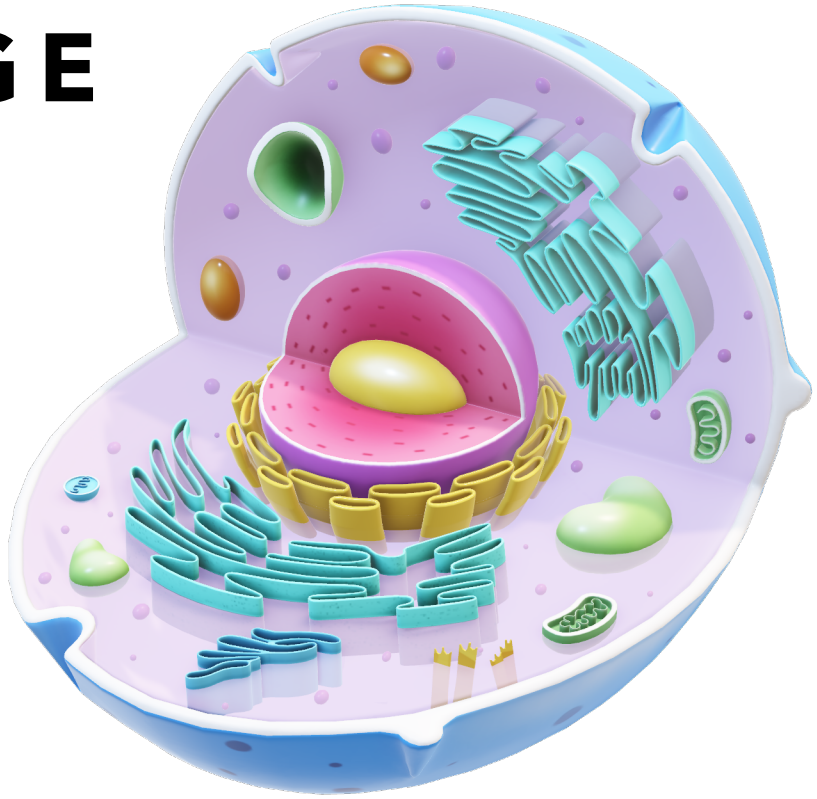
PEN UP, PEN DOWN…

```
function draw() {
  var canvas = document.getElementById('canvas');
  var ctx = canvas.getContext('2d');
  ctx.beginPath();
  ctx.arc(75, 75, 50, 0, Math.PI * 2, true); // Outer circle
  ctx.moveTo(110, 75);
  ctx.arc(75, 75, 35, 0, Math.PI, false);  // Mouth (clockwise)
  ctx.moveTo(65, 65);
  ctx.arc(60, 65, 5, 0, Math.PI * 2, true);  // Left eye
  ctx.moveTo(95, 65);
  ctx.arc(90, 65, 5, 0, Math.PI * 2, true);  // Right eye
  ctx.stroke();
}
```

INTERESTING CANVAS FEATURES

- Variable fill, stroke, line end and pattern options
- Blend modes
- Dynamic coordinate system – define coordinates once and rotate/zoom coordinate system
- Drawing stack – you can store and save the state of the canvas
- Everything results in an image

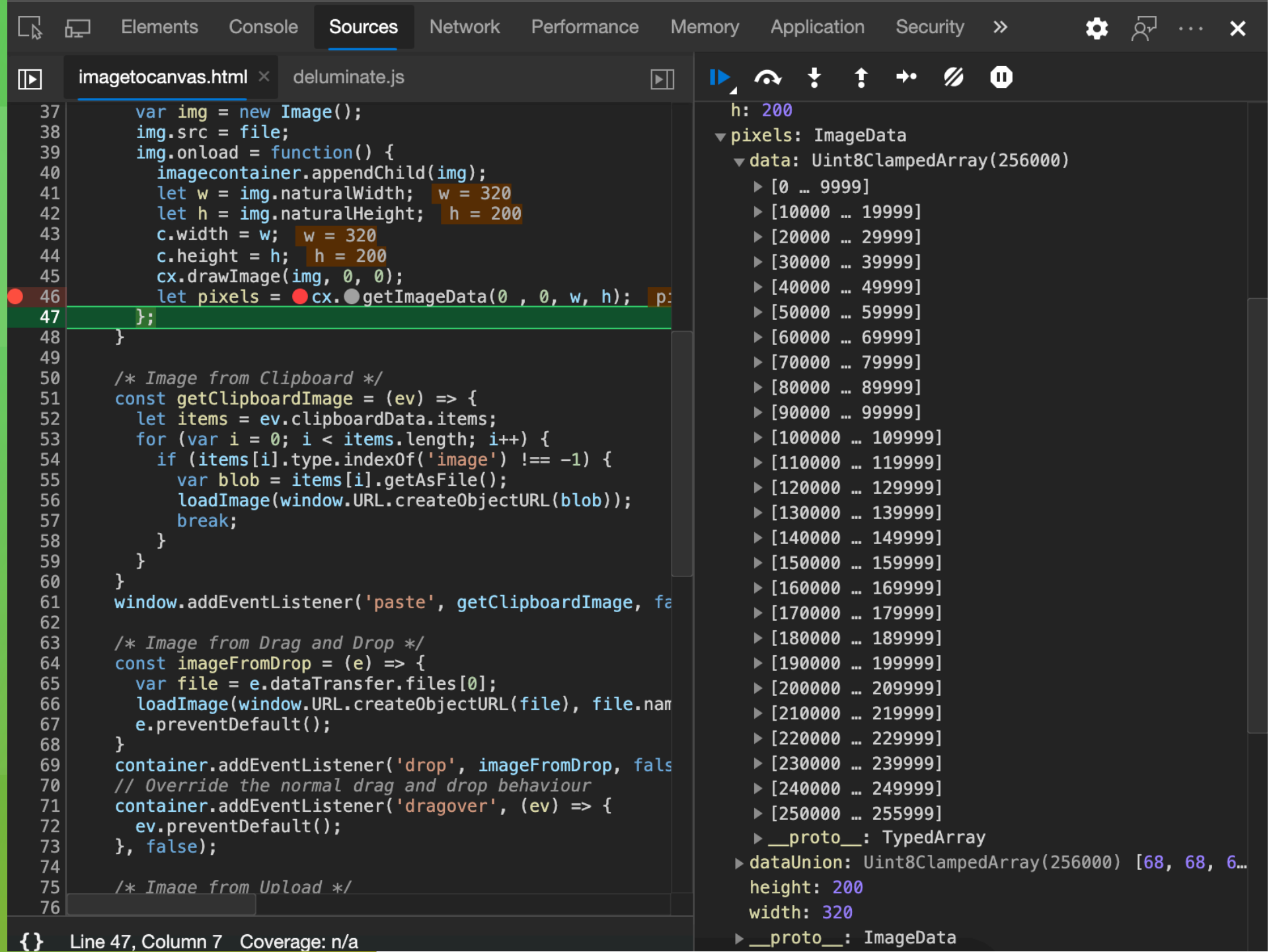
OPENING UP THE IMAGE



COPYING THE IMAGE TO THE CANVAS

```
const c = document.querySelector('canvas');
const cx = c.getContext('2d');

/* Show the image once we have it */
const loadImage = (file, name) => {
  if (name) {
    output.innerText = 'Filename: ' + name;
  }
  var img = new Image();
  img.src = file;
  img.onload = function() {
    imagecontainer.appendChild(img);
    let w = img.naturalWidth;
    let h = img.naturalHeight;
    c.width = w;
    c.height = h;
    cx.drawImage(img, 0, 0);
    let pixels = cx.getImageData(0 , 0, w, h);
  };
}
```

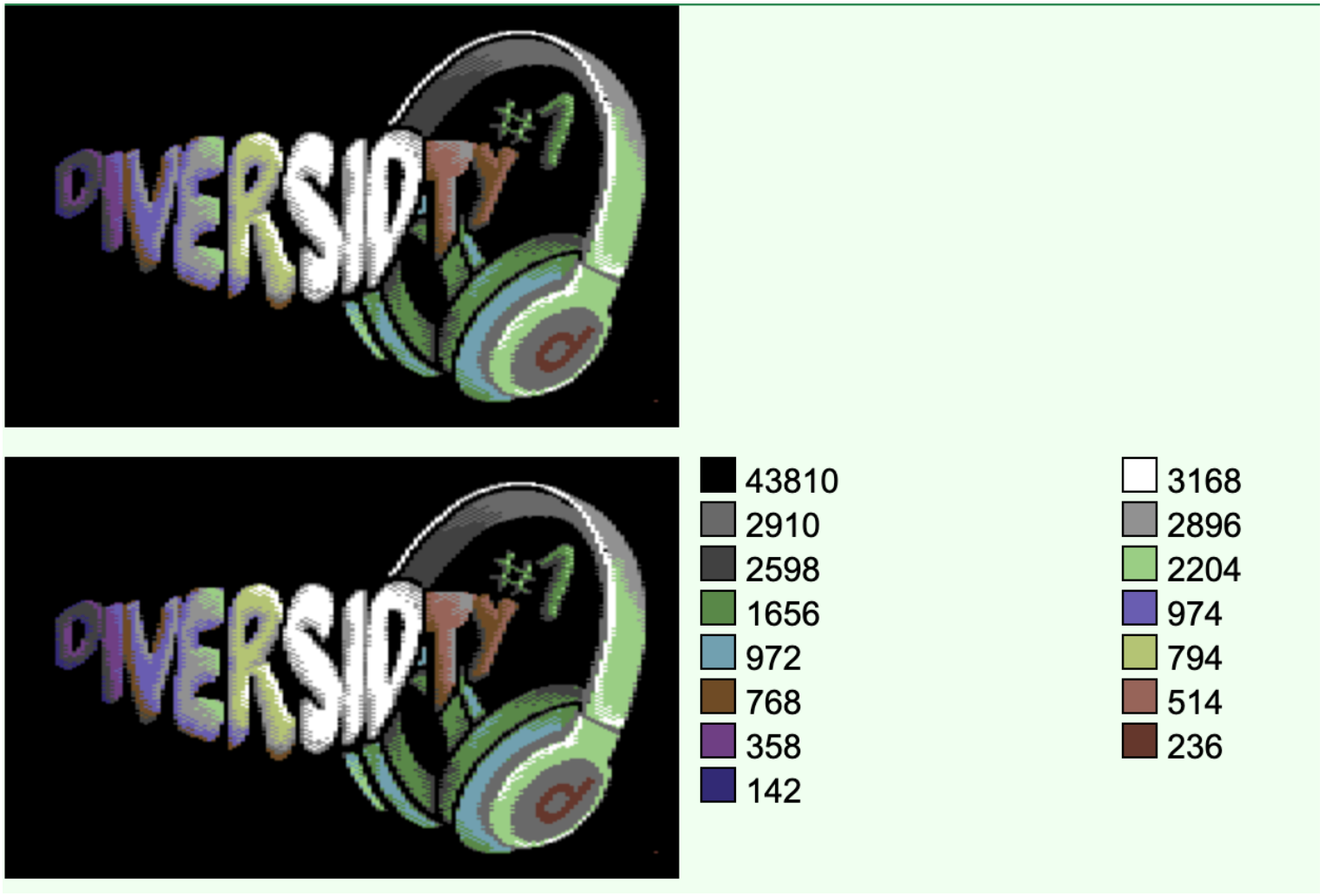

Elements
Console
Sources
Network
Performance
Memory
Application
Security
imagetocanvas.html
deluminate.js
var img = new Image();
img.src = file;
img.onload = function() {
imagecontainer.appendChild(img);
let w = img.naturalWidth; w = 320
let h = img.naturalHeight; h = 200
c.width = w; w = 320
c.height = h; h = 200
cx.drawImage(img, 0, 0);
let pixels = cx.getImageData(0 , 0, w, h);
};
}
/* Image from Clipboard */
const getClipboardImage = (ev) => {
let items = ev.clipboardData.items;
for (var i = 0; i < items.length; i++) {
if (items[i].type.indexOf('image') !== -1) {
var blob = items[i].getAsFile();
loadImage(window.URL.createObjectURL(blob));
break;
}
}
}
window.addEventListener('paste', getClipboardImage,
/* Image from Drag and Drop */
const imageFromDrop = (e) => {
var file = e.dataTransfer.files[0];
loadImage(window.URL.createObjectURL(file), file.nam
e.preventDefault();
}
container.addEventListener('drop', imageFromDrop, fals
// Override the normal drag and drop behaviour
container.addEventListener('dragover', (ev) => {
ev.preventDefault();
}, false);
/* Image from Upload */
{}
Line 47, Column 7
Coverage: n/a
h: 200
pixels: ImageData
data: Uint8ClampedArray(256000)
[0 … 9999]
[10000 … 19999]
[20000 … 29999]
[30000 … 39999]
[40000 … 49999]
[50000 … 59999]
[60000 … 69999]
[70000 … 79999]
[80000 … 89999]
[90000 … 99999]
[100000 … 109999]
[110000 … 119999]
[120000 … 129999]
[130000 … 139999]
[140000 … 149999]
[150000 … 159999]
[160000 … 169999]
[170000 … 179999]
[180000 … 189999]
[190000 … 199999]
[200000 … 209999]
[210000 … 219999]
[220000 … 229999]
[230000 … 239999]
[240000 … 249999]
[250000 … 255999]
__proto__: TypedArray
dataUnion: Uint8ClampedArray(256000) [68, 68, 6…
height: 200
width: 320
__proto__: ImageData

THE IMAGEDATA OBJECT

- **width**: Width of the canvas in pixels
- **height**: Height of the canvas in pixels
- **data**: Array of all the pixels in RGBA format

EXAMPLE:

ANALYSE AND COUNT USED COLOURS



GET IMAGE, ADD TO CANVAS, SEND PIXELS

```
<ul id="colourslist"></ul>


const c = document.querySelector('canvas');
const cx = c.getContext('2d');
const colourslist = document.querySelector('#colourslist');

/* Show the image once we have it */
const loadImage = (file, name) => {
  var img = new Image();
  img.src = file;
  img.onload = function() {
    imagecontainer.appendChild(img);
    let w = img.naturalWidth;
    let h = img.naturalHeight;
    c.width = w;
    c.height = h;
    cx.drawImage(img, 0, 0);
    analysecolours(cx.getImageData(0 , 0, w, h));
  };
}
```

LOOP OVER ALL PIXELS AND STORE THEIR VALUES IN AN OBJECT

```
const analysecolours = (pixeldata) => {
  let px = pixeldata.data;
  let colours = {};
  let all = px.length;
  for (let i = 0; i < all; i += 4) {
    let col = `${px[i]}|${px[i+1]}|${px[i+2]}|${px[i+3]}`;
    if (colours[col]) {
      colours[col]++;
    } else {
      colours[col] = 1;
    }
  }
```

SORT THE OBJECT AND PRINT OUT A LIST

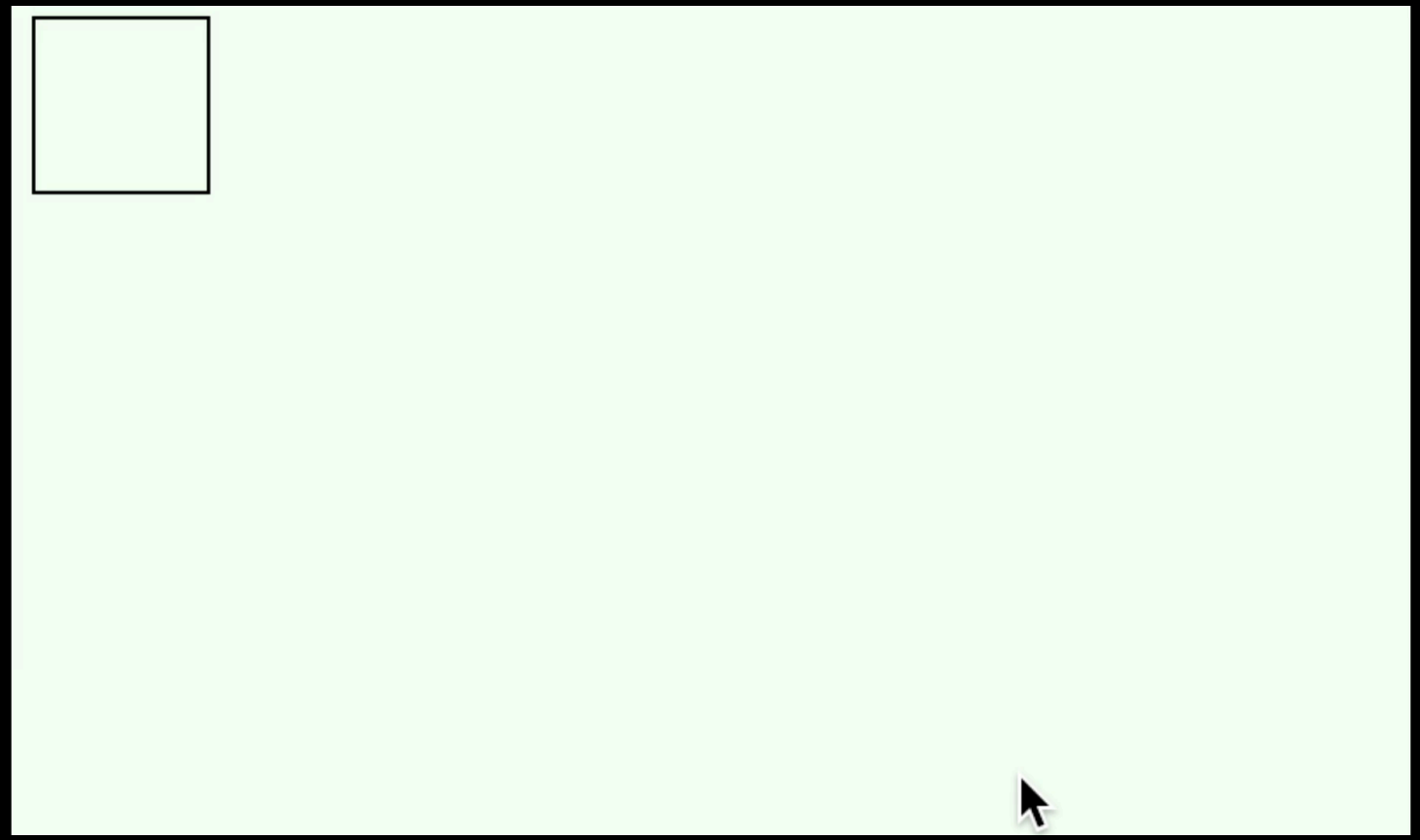
```
  sortedcolours = Object.keys(colours).sort(
    (a, b) => {return -(colours[a] - colours[b])}
  );
  var out = '';
  sortedcolours.forEach(function(key){
    var rgba = key.split('|');
    out +=
    `<li>
        <span style="background:rgba(
          ${rgba[0]},${rgba[1]},${rgba[2]},${rgba[3]}
        )"></span>
        ${colours[key]}
    </li>`;
  });
  colourslist.innerHTML = out;
}
```

USING IMAGEDATA

- You can read with getImageData and write with putImageData
- In between everything is array manipulation
- You can create pixel patterns
- You can change each pixel of an image source. This allows for writing filters, removing backgrounds, recolouring, etc…

DON'T MAKE YOUR LIFE TOO HARD

DON'T MAKE YOUR LIFE TOO HARD



THINKING TOO COMPLEX

- Get the mouse coordinates
- Loop through the array and check if y times the width of the canvas was reached
- Stop at the x index and read the colour.

OR ...

```
/* Show the image once we have it */
const loadImage = (file, name) => {
//         [...]
    addpicker(img);
  };
})
const addpicker = (img) => {
  img.addEventListener('mousemove',pick);
}
const pick = (ev) => {
  let coordinates = getposition(ev);
  let col = cx.getImageData(
    coordinates.x, coordinates.y, 1, 1
  );
  output.style.background = `rgba(${col.data.join(',')})`;
};
const getposition = (ev) => {
  let x = ev.clientX;
  let y = ev.clientY;
  let pos = ev.target.getBoundingClientRect();
  return {x: x - pos.x|1, y: y-pos.y|1};
}
```

CANVAS AND IMAGES

- drawImage() is not only there to put an image onto the canvas
- You can also crop and transform image with it
- Scaling and cropping is easier
- Rotation means you need to rotate the coordinate system around it

CROPPING NEEDS SOME GETTING USED TO...

```
context.drawImage(
  originalImage,
  Left,
  Top,
  Width,
  Height,
  newLeft,
  newTop,
  newWidth,
  newHeight
);
```

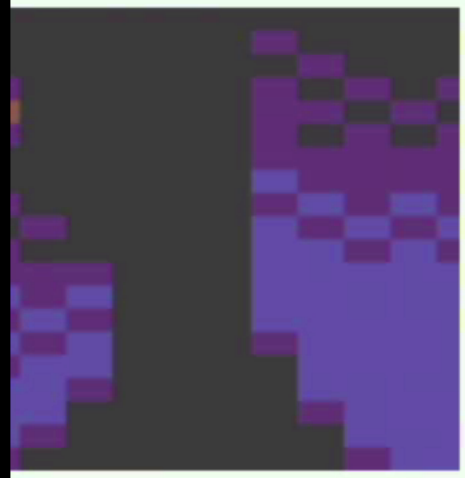
```
context.drawImage(
  originalImage,
  // Take 20 by 30 pixels from the original image
  // starting at the coordinate 10 left and 15 top
  10,
  15,
  20,
  30,
  // and copy it onto this canvas at 20 left and
  // 5 top and resize it to 200 by 300
  20,
  5,
  200,
  300
);
```

ZOOMING A WHOLE IMAGE...

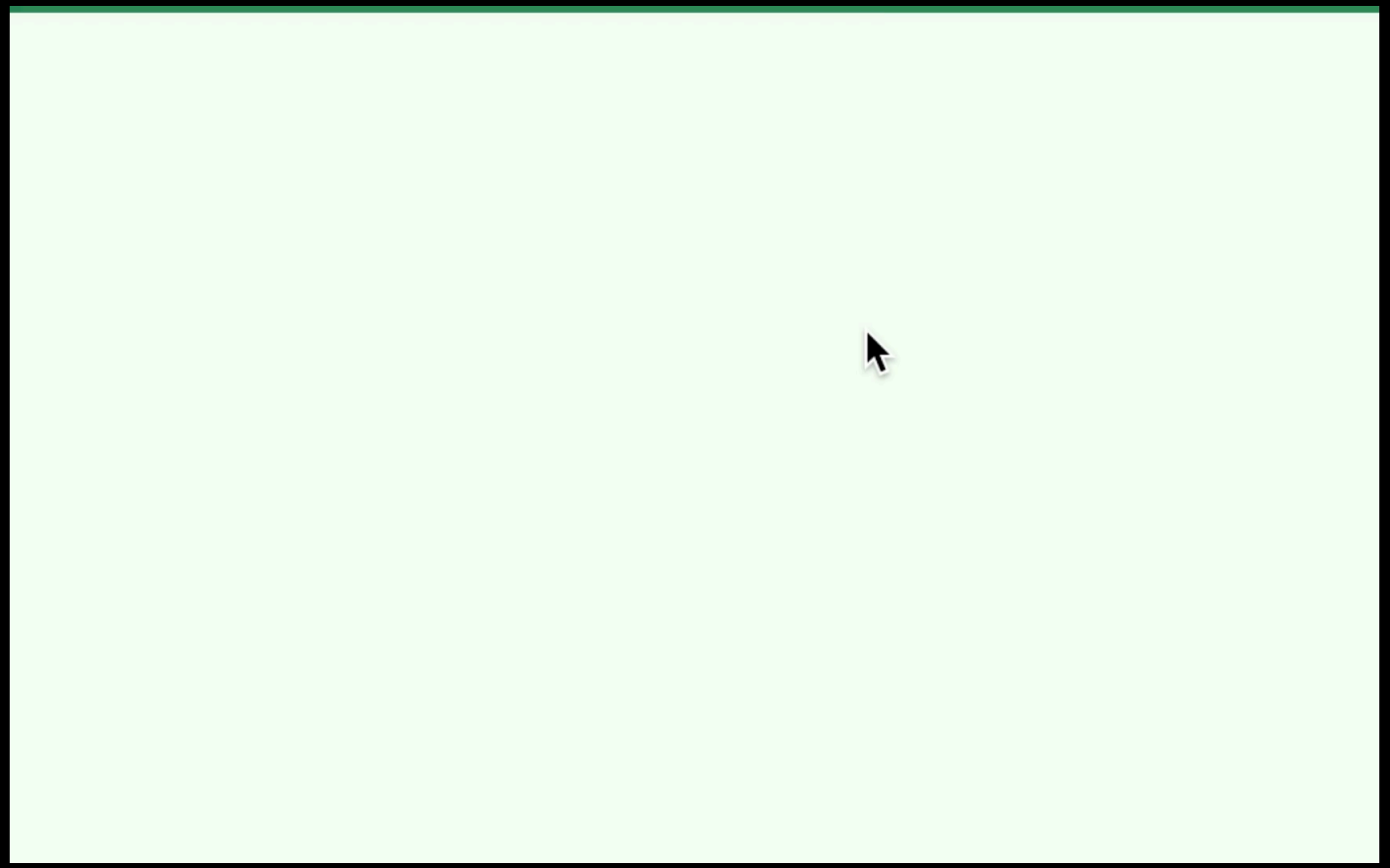
ZOOMING IS EASIER BY SCALING THE COORDINATE SYSTEM

```
/* Show the image once we have it */
const loadImage = (file, name) => {
  var img = new Image();
  img.src = file;
  img.onload = function() {
    imagecontainer.appendChild(img);
    let w = img.naturalWidth;
    let h = img.naturalHeight;
    let zoomfactor = 3;
    c.width = w * zoomfactor;
    c.height = h * zoomfactor;
    cx.imageSmoothingEnabled = 0;
    cx.scale(zoomfactor, zoomfactor);
    cx.drawImage(img, 0, 0);
  };
}
```

INTERACTIVE ZOOMING



INTERACTIVE ZOOMING



INTERACTIVE ZOOMING USING CROP AND RESIZE

```
const loadImage = (file, name) => {
  var img = new Image();
  img.src = file;
  img.onload = function() {
    imagecontainer.appendChild(img);
    img.addEventListener('mousemove',dozoom);
  };
}
const dozoom = (ev) => {
  let coordinates = getposition(ev);
  c.width = 100;
  c.height = 100;
  cx.imageSmoothingEnabled = 0;
  let img = ev.target;
  cx.drawImage(
    img, coordinates.x - 10, coordinates.y - 10,
    20, 20, 0, 0, 100, 100
  );
};
const getposition = (ev) => {
  let x = ev.clientX;
  let y = ev.clientY;
  let pos = ev.target.getBoundingClientRect();
  return {x: x - pos.x|1, y: y-pos.y|1};
}
```

VALID IMAGE RESOURCES

- Any image (must be on the same domain)
- Any video (^ that)
- Any other canvas

SAVING THE FINAL IMAGE

```
var dataURL = canvas.toDataURL();
// "
// blAAAADElEQVQImWNgoBMAAABpAAFEI8ARAAAAAElFTkSuQmCC"

var fullQuality = canvas.toDataURL('image/jpeg', 1.0);
// ...9oADAMBAAIRAxEAPwD/AD/6AP/Z"
var mediumQuality = canvas.toDataURL('image/jpeg', 0.5);
var lowQuality = canvas.toDataURL('image/jpeg', 0.1);
```

- Right-click works
- Other than that, you can do a canvas.toDataURL()
- By default, canvas creates PNG files, but you can also do WebP or JPG
- The latter takes a quality parameter

BONUS TRICK: ANCHOR AROUND CANVAS

```
<a href="#" download="zoomed.png" id="savelink">
  <canvas>
    This is a canvas
  </canvas>
</a>
```

```
const savelink = document.querySelector('#savelink');

  /* Show the image once we have it */
const loadImage = (file, name) => {
  var img = new Image();
  img.src = file;
  img.onload = function() {
    imagecontainer.appendChild(img);
    let w = img.naturalWidth;
    let h = img.naturalHeight;
    let zoomfactor = 3;
    c.width = w * zoomfactor;
    c.height = h * zoomfactor;
    cx.imageSmoothingEnabled = 0;
    cx.scale(zoomfactor, zoomfactor);
    cx.drawImage(img, 0, 0);
    savelink.href = c.toDataURL();
  };
}
```

SOME THINGS I HAVE BUILT WITH THESE TECHNIQUES

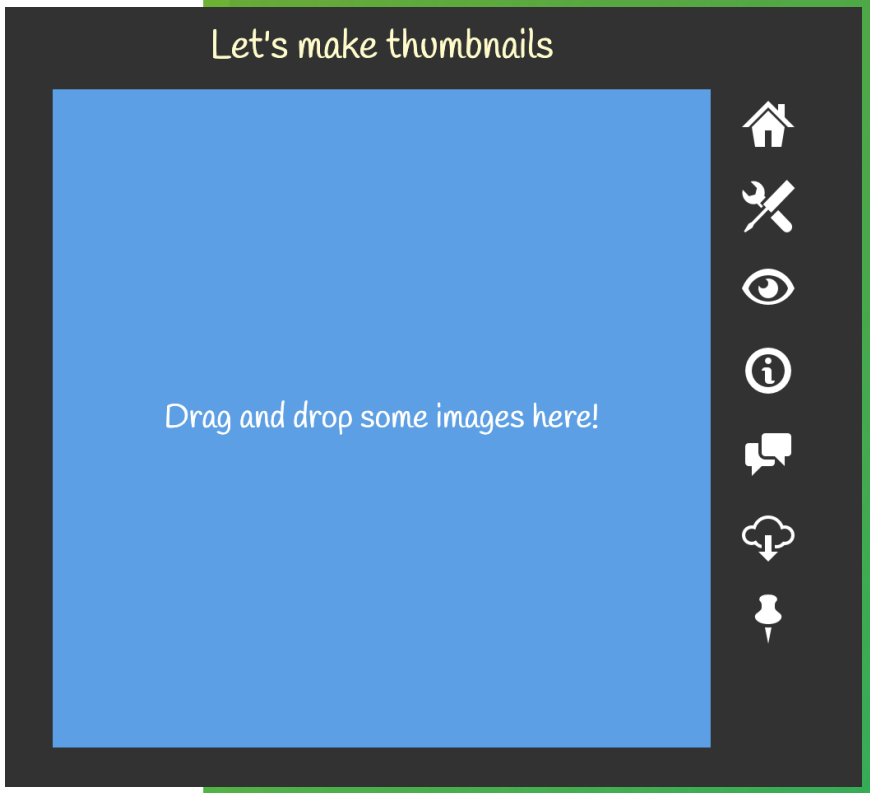
MAKETHUMBNAILS.COM

DROP A BUNCH OF IMAGES

DEFINE THE SIZE OF THE THUMBNAILS, OR CHOOSE DEFAULT

DOWNLOAD THE THUMBNAILS ONE BY ONE OR AS A ZIP

ALL ON YOUR OWN MACHINE



Remove personal data from photos before sharing them on the internet

Before you upload photos to the web, you might want to check if you don't give out too much information. Cameras, smartphones and other hardware does not only store the image information but also the time and date, what camera was used and possibly even the location on the planet in every image in EXIF data.

Using this tool you can see this data, and download an image that has all of it removed to send out.

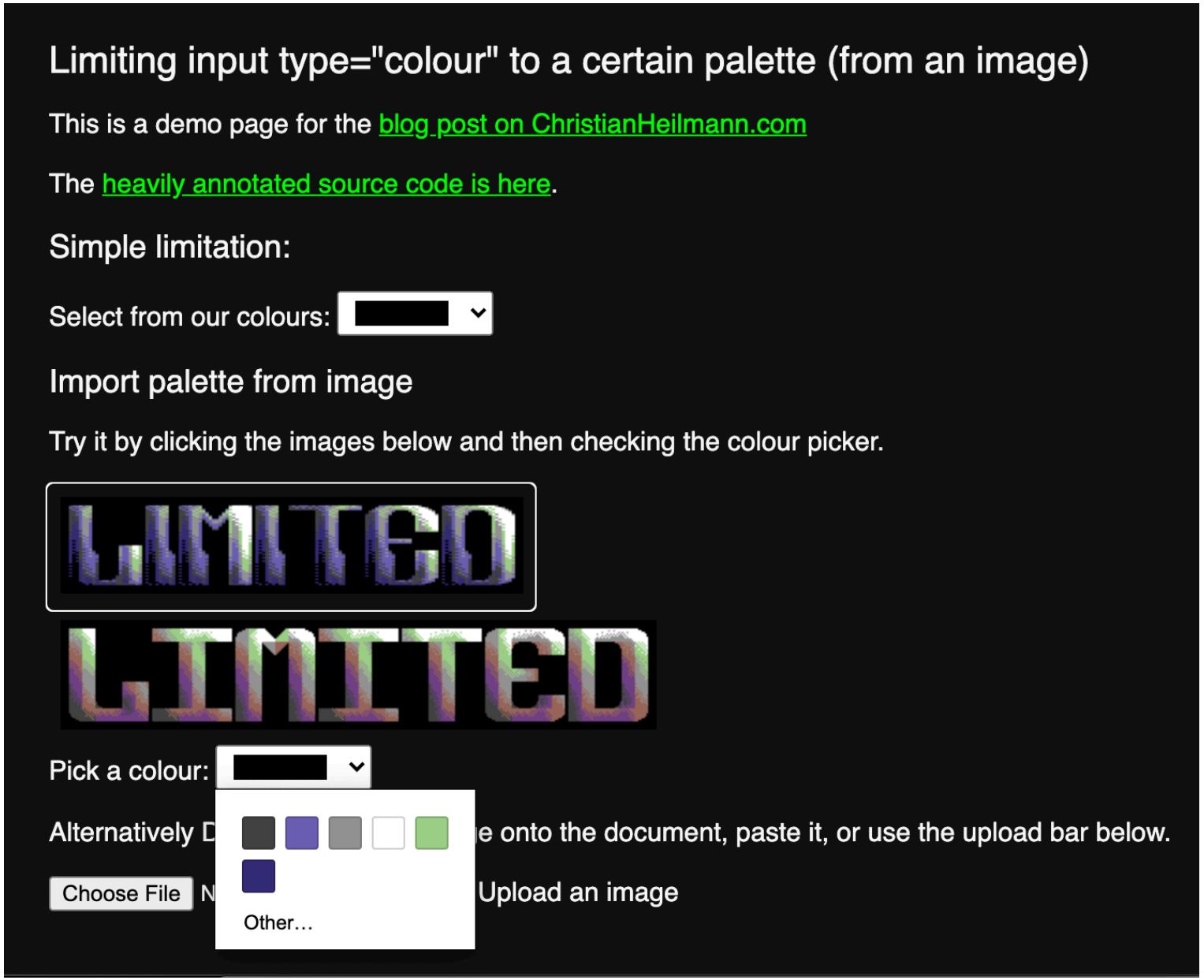
Your photo does not get uploaded anywhere, all of this happens on your device, in your browser. It even works offline.

Simply browse for your photo here and you get all the information in it. Then click the "Download clean image" link to get the image with all this information stripped from it. If there is no extra data in the image, it will tell you so.

Choose File No file chosen

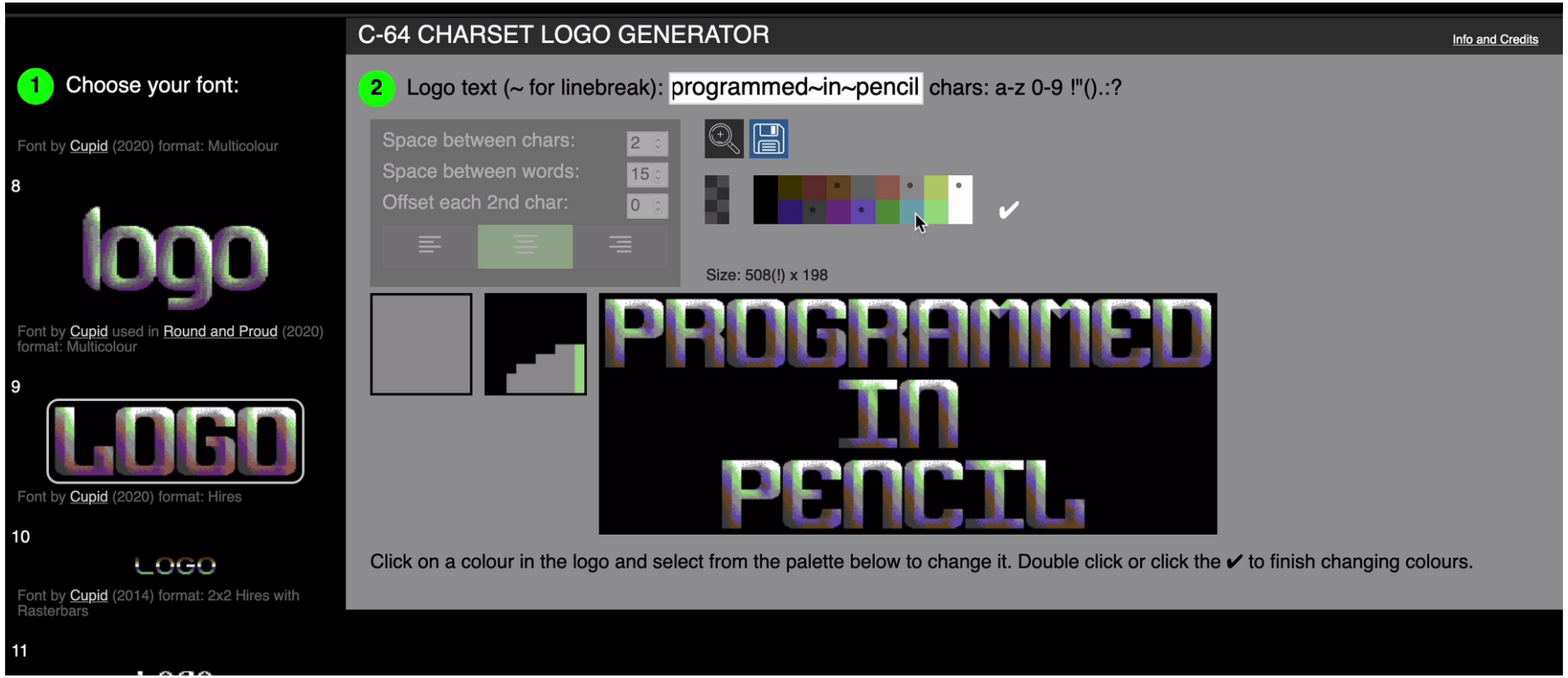
REMOVEPHOTODATA.COM

COLOUR PICKER FROM IMAGE



https://christianheilmann.com/2020/04/22/limiting-input-type-color-to-a-certain-palette-from-an-image/

LOGO-O-MATIC



https://codepo8.github.io/logo-o-matic/

LOGO-O-MATIC

https://codepo8.github.io/logo-o-matic/

BONUS TIPS:

- Canvas doesn't throw errors if you paint outside the available space, but it uses memory
- If you only need to convert with a canvas, don't even add it to the DOM
- If you do a lot of array looping and changes, use a worker to avoid the interface to slow down. Canvas, however isn't available in a worker!
- Resizing a canvas is a cheap way to clear it
- In animations, using a rgba(r,g,b,0.5) value for clearing the canvas gives a smooth ghosting effect
- A black and white map is a great way to do background collision in games – just read the next few pixels in direction of the move and if they are white – boom

THANK YOU AND HAVE FUN!

Chris Heilmann

@codepo8

@EdgeDevTools

christianheilmann.com

Click the feedback in Edge developer tools!