Fontys Easter egg

Opdrachtgever: Fonty's

Opdracht: van de opdrachtgever Fontys hebben we als groep de taak gekregen om Easter eggs te bedenken die toegevoegd kunnen worden aan de fontys website. Hierin hebben we de vrijheid gekregen om zelf het onderwerp en de implementatie te verzinnen. Voor deze opdracht hebben we 3 weken gekregen om een idee te verzinnen en een visueel prototype te creëren.

Voor deze opdracht heb ik een proof of concept gemaakt van mijn groep zijn Easter egg idee.

Proof of concepts:

Krasbanner:

Voor mijn eerste proof of concept heb ik door middel van een button press de afbeelding op de pagina verandert.

```
var canvas = document.getElementById("canvas"),
imgcanvas = canvas.getContext('2d'),

img = new Image();

    document.querySelector(".button").addEventListener("click", function(){
        imgcanvas.drawImage(img, 0, 0, canvas.width, canvas.height);
    })
    img.src = 'replace.jpg';
```

Deze button heb ik later vervangen met de .onload functie om het te automatiseren. Na het vervangen van de afbeelding moest ik een manier gaan zoeken om de nieuwe afbeeldings laag we te kunnen halen om de onderste laag zoals bij een kraslot tevoorschijn te laten komen. Hiervoor moet de code een paar dingen weten: de muis positie op het canvas, of de linker muisknop is ingedrukt, of de muis aan het bewegen is en op welke manier het canvas vervangen moet worden.

```
canvas.addEventListener("mousemove", function(e) {
    var MousePos = getMousePos(e.clientX, e.clientY);
  var leftBut = detectLeftButton(e);
  if (leftBut == 1) {
        drawDot(MousePos.x, MousePos.y);
  }
}, false);
```

Ik heb een EventListener toegvoed aan het canvas element dat bewegingen van de muis bijhoudt. Elke keer als de muis beweegt op het canvas element voort hij de functie getMousePos uit en geeft hij de horizontale en verticale waarde van de eventhandler mee.

```
function getMousePos(xRef, yRef) {
    var canvasRect = canvas.getBoundingClientRect();
    return {
      x: Math.floor((xRef-canvasRect.left)/(canvasRect.right-canvasRect.left)*canvas.width),
      y: Math.floor((yRef-canvasRect.top)/(canvasRect.bottom-canvasRect.top)*canvas.height)
    };
}
```

Om de muis positie op het canvas te achterhalen gebruiken we de functie .getBoundingClientRect() dit verteld ons de groote en de positie van het canvas element gerelateerd aan de viewport. Dit word gedaan door middel van acht waardes: left, top, right, bottom, x, y, width en height.

Ik heb gevonden hoe ik deze positie kan bereken met dank aan de antwoorden op deze stackoverflow thread. https://stackoverflow.com/questions/17130395/real-mouse-position-in-canvas nadat we de positie van de muis op he canvas hebben berekend moeten we kijken of de linker muisknop is ingedrukt.

```
function detectLeftButton(e) {
    if ('buttons' in e) {
        return e.buttons === 1;
    } else if ('which' in e) {
        return e.which === 1;
    } else {
        return e.button === 1;
    }
}
```

De if statements in deze functie kijken of de waarde ‘buttons’ en ‘which’ in de eventhandler zitten. Zo ja dan is de if statement True zo nee dan is de if statement False. Als “‘buttons’ in e” True is de return waarde ook een True or False. Dit komt omdat de return waarde ook een statement is. We kijken of e.buttons gelijk is aan 1 omdat 1 overeenkomt met de linker muisknop. De which check hier ook naar.

```
if (leftBut == 1) {
        drawDot(MousePos.x, MousePos.y);
}
```

Als de linker muis kno dus gelijk is aan 1 word de functie drawDot aangeroepen.

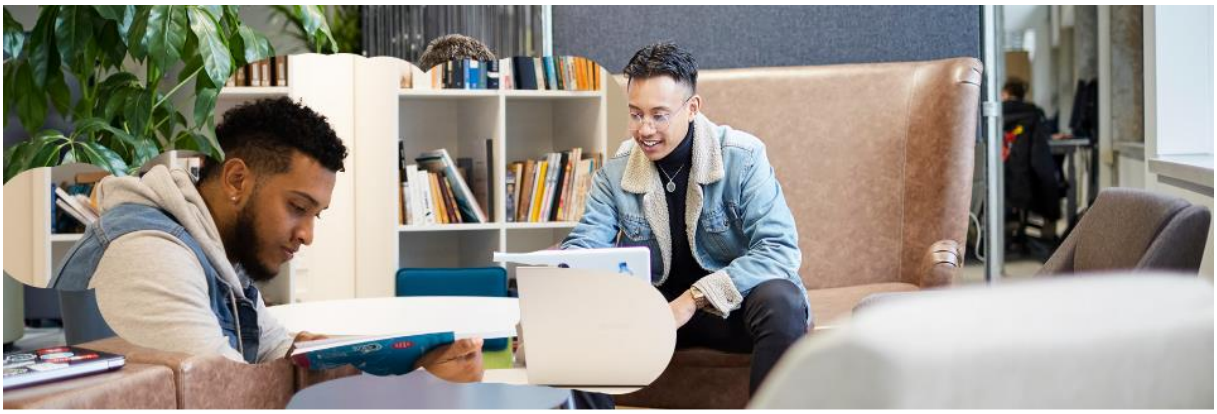
```
function drawDot(mouseX,mouseY){
    imgcanvas.beginPath();
    imgcanvas.arc(mouseX, mouseY, brushRadius, 0, 2*Math.PI);
    imgcanvas.fillStyle = '#000';
    imgcanvas.globalCompositeOperation = "destination-out";
    imgcanvas.fill();
}
```

Voor de methode img.canvas.arc heb ik gebruik gemaakt van het ik hieruit gehaald https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/arc maar ik heb er de muis x en y positie aan toegevoegd zodat de gebruiker zijn eigen pad kan tekenen. Door de globalCompositeOperation met “destination-out” word de nieuwe vorm die gemaakt word op het canvas vervangen met leegte.

destination-out

The existing content is kept where it doesn't overlap the new shape.

Door de .fill methode word de nieuwe vorm toegepast op het canvas en komt dus de afbeelding erachter tevoorschijn.



Wat heb ik geleerd:

- Werken met Canvas elementen in html en js
- Hoe je een canvas element bewerkt
- Hoe je een canvas element kan gebruiken in html