Coding Learning Log:

collected technical notes for some of the Javascript and Html processes used throughout Semester 2.

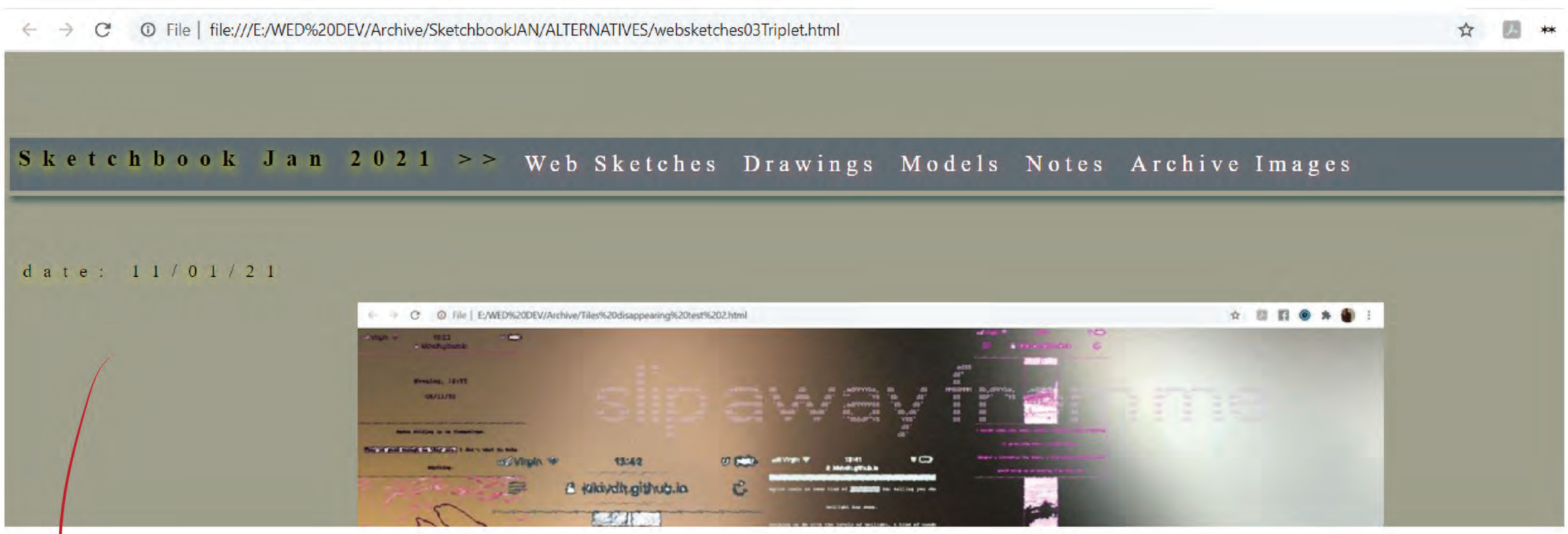
01 Jan - 10 Apr 2021

Topbar ‘Sticky’ Navigation:

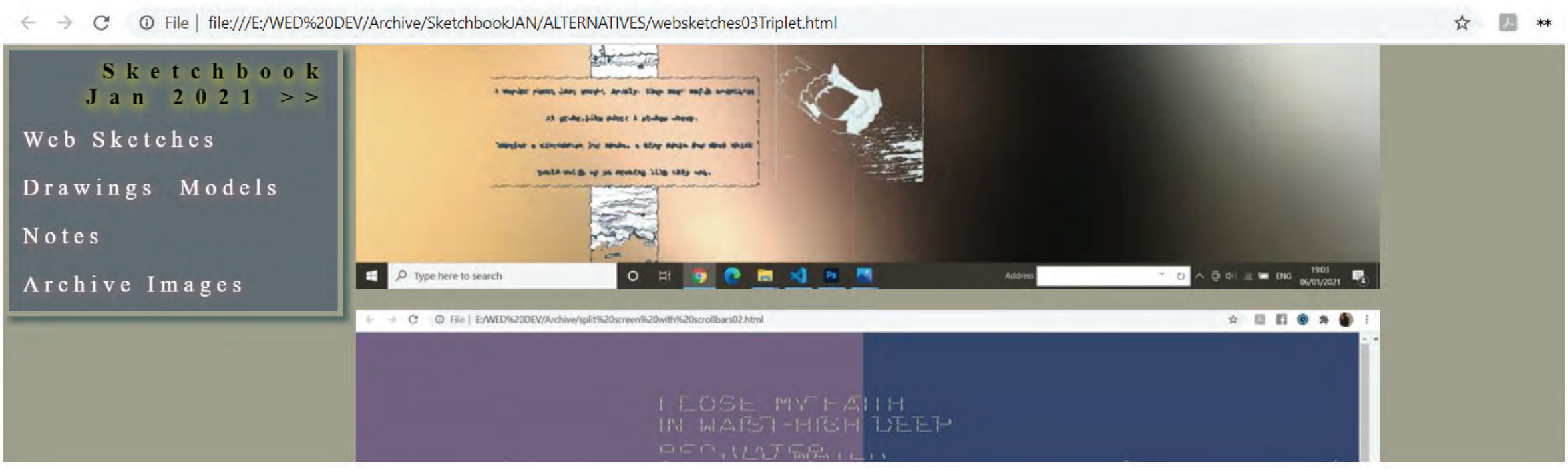
>for more complex sites with multiple layered pages, this keeps the top ‘home’ navigation bar visible in order to allow for maximum user ease (basically, much less scrolling!)

javascript

```
   <script>
window.onscroll = function() {myFunction()};

var header = document.getElementById("myHeader");
var sticky = header.offsetTop;

function myFunction() {
  if (window.pageYOffset > sticky) {
    header.classList.add("sticky");
  } else {
    header.classList.remove("sticky");
  }
}
</script>
```



>can make responsive to main stack flow: if main CSS across the header is too big, it will flex to the left. Very nice to read!



Topbar 'Sticky' Navigation:

html

```
<div class="header" id="myHeader">

        <ul>
          <li><p2><b>...</b></p2></li>

          <li><a href="" >...</a></li>
          <li><a href="">...</a></li>
          <li><a href="" >...</a></li>
          <li><a href="">...</a></li>

          </ul>
    </div>
```

>have to put in div class ID on the header and close it!

CSS

```
 .header {
  padding: 5px;
  text-align: center;
  font-family: 'Times New Roman', Times, serif;
  box-shadow: 5px 5px 5px rgb(81, 104, 96);
 font-size: 20px;
color: #383232;;
position: relative;
top: 70px; }

.sticky {
  position: fixed;
  top: 0;
  width: 19%}


 .content {
  padding: 16px;
}

ul {
  list-style-type:none;
  margin: 0;
```

>very high padding (>200px), nicely spread.

Customising Scrollbars:

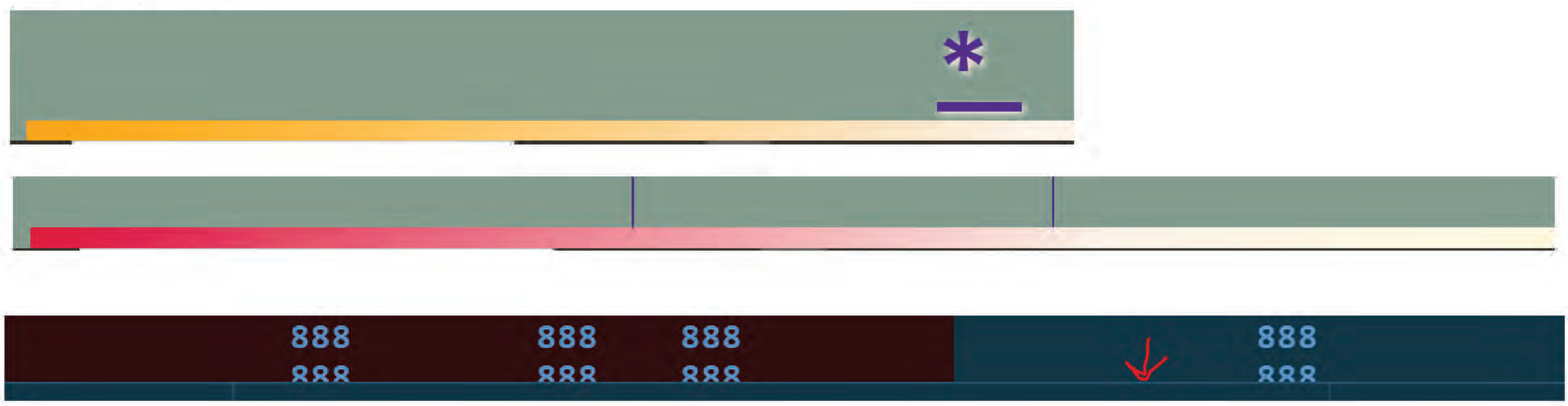
CSS

```
::-webkit-scrollbar {
  width: 20px;
}


::-webkit-scrollbar-track {
  box-shadow: inset 0 0 5px grey;
  border-radius: 10px;
}


::-webkit-scrollbar-thumb {
  background: red;
  border-radius: 10px;
}


::-webkit-scrollbar-thumb:hover {
  background: #b30000;
}
```

html

```
<div class= "scroll"></div>
```

Examples:



>Be careful not to blend the bar itself into the page: normally looks best with a >3px stroke or box-shadow to bring it out well.

Clocks: javascript countdown timer (simple)

html

```
<div class="centered">

   <h1 id="demo"></h1>

 </div>
```

javascript

```
<h1 id="demo"></h1>
    <script>

var countDownDate = new Date("Mar 27, 2021 15:37:25").getTime();

// Update the count down every 1 second
var x = setInterval(function() {

  // Get today's date and time
  var now = new Date().getTime();

  // Find the distance between now and the count down date
  var distance = countDownDate - now;

  // Time calculations for days, hours, minutes and seconds
  var days = Math.floor(distance / (1000 * 60 * 60 * 24));
  var hours = Math.floor((distance % (1000 * 60 * 60 * 24)) / (1000 *
60 * 60));
  var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 *
60));
  var seconds = Math.floor((distance % (1000 * 60)) / 1000);

  // Output the result in an element with id="demo"
  document.getElementById("demo").innerHTML = days + "d " + hours + "h
"
  + minutes + "m " + seconds + "s ";

  // If the count down is over, write some text
  if (distance < 0) {
    clearInterval(x);
    document.getElementById("demo").innerHTML = "burning all night";
  }
}, 1000);
</script>
```

Clocks: fetching current time and displaying

CSS

```
#txt{
   color:rgb(228, 169, 154);
  font-size: 60%;

  text-align: center;
  line-height: 90px;
  margin: auto;
  letter-spacing: 15px;}
```

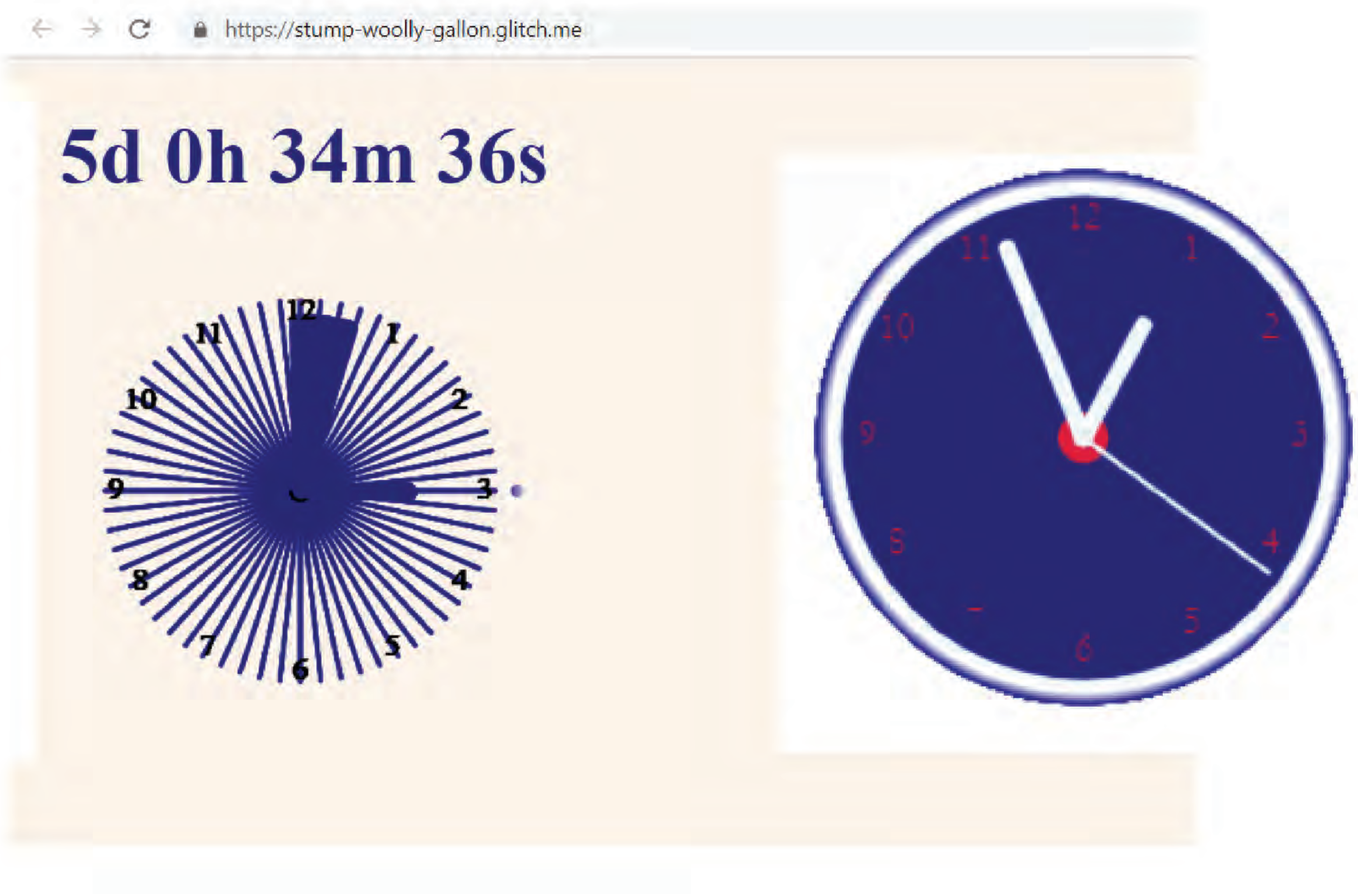
html

```
<h2><body onload="startTime()">
<div id="txt"></div></body>

    </div>
```

javascript

```
var scroll = setInterval(function(){ window.scrollBy(0,2); },
100);

      function startTime() {
  var today = new Date();
  var h = today.getHours();
  var m = today.getMinutes();
  var s = today.getSeconds();
  m = checkTime(m);
  s = checkTime(s);
  document.getElementById('txt').innerHTML =
  h + ":" + m + ":" + s;
  var t = setTimeout(startTime, 500);
}
function checkTime(i) {
  if (i < 10) {i = "0" + i};  // add zero in front of numbers <
10
  return i;
}
```

Clocks: analogue animation (more complex)



html

```
<body>

     <h1 id="demo"></h1>

<canvas id="canvas" width="300 px" height="300 px"
style="background-color:transparent">
   </canvas>
```

javascript

```
function drawHand(ctx, pos, length, width) {
    ctx.beginPath();
    ctx.lineWidth = width;
    ctx.lineCap = "round";
    ctx.moveTo(0,0);
    ctx.rotate(pos);
    ctx.lineTo(0, -length);
    ctx.stroke();
    ctx.rotate(-pos);
}

var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");
var radius = canvas.height / 2;
ctx.translate(radius, radius);
radius = radius * 0.90
setInterval(drawClock, 1000);

function drawClock() {
  drawFace(ctx, radius);
  drawNumbers(ctx, radius);
  drawTime(ctx, radius);
}
```

Clocks: analogue animation (more complex)

javascript cont'd

```
function drawFace(ctx, radius) {
  var grad;
  ctx.beginPath();
  ctx.arc(0, 0, radius, 0, 0*Math.PI);
  ctx.fillStyle = 'snow';
  ctx.fill();
  grad = ctx.createRadialGradient(0,0,radius*0.95, 0,0,radius*1.05);
  grad.addColorStop(0, 'navy');
  grad.addColorStop(1, 'linen');
  ctx.strokeStyle = grad;
  ctx.lineWidth = radius*0.05;
  ctx.stroke();
  ctx.beginPath();
  ctx.arc(0, 0, radius*0.05, 0, 1*Math.PI);
  ctx.fillStyle = 'black';
  ctx.fill();
}

function drawNumbers(ctx, radius) {
  var ang;
  var num;
  ctx.font = radius*0.15 + "px serif";
  ctx.textBaseline="middle";
  ctx.textAlign="center";
  for(num = 1; num < 13; num++){
    ang = num * Math.PI / 6;
    ctx.rotate(ang);
    ctx.translate(0, -radius*0.85);
    ctx.rotate(-ang);
    ctx.fillText(num.toString(), 0, 0);
    ctx.rotate(ang);
    ctx.translate(0, radius*0.85);
    ctx.rotate(-ang);
  }
}

function drawTime(ctx, radius){
    var now = new Date();
    var hour = now.getHours();
    var minute = now.getMinutes();
    var second = now.getSeconds();
    //hour
    hour=hour%12;
    hour=(hour*Math.PI/6)+
    (minute*Math.PI/(6*60))+
    (second*Math.PI/(360*60));
    drawHand(ctx, hour, radius*0.5, radius*0.07);
    //minute
    minute=(minute*Math.PI/30)+(second*Math.PI/(30*60));
    drawHand(ctx, minute, radius*0.8, radius*0.07);
    // second
    second=(second*Math.PI/30);
    drawHand(ctx, second, radius*0.9, radius*0.02);
```

Marquee scrolls:

CSS

```
.marquee {
            height: 45px;
            overflow: hidden;
            position: relative;
            background: linear-gradient(141deg, #829A8D 0%, #88ABA5
51%, #4682B4 75%);
            color:linen;
            border: 2px solid indigo;

        }
    .marquee:hover {
    animation-play-state: paused;
}
        .marquee p {
            position: absolute;
            width: 100%;
            height: 100%;
            margin: 0;
                font-size:24px;
            line-height:20px;
            text-align: center;
            -moz-transform: translateX(100%);
            -webkit-transform: translateX(100%);
            transform: translateX(100%);
            -moz-animation: scroll-right 2s linear infinite;
            -webkit-animation: scroll-right 2s linear infinite;
            animation: scroll-right 30s linear infinite;
        }

        @-moz-keyframes scroll-right {
            0% {
                -moz-transform: translateX(100%);
            }
            100% {
                -moz-transform: translateX(-100%);
            }
        }

        @-webkit-keyframes scroll-right {
            0% {
                -webkit-transform: translateX(100%);
            }
            100% {
                -webkit-transform: translateX(-100%);
            }
        }

        @keyframes scroll-right {
            0% {
                -moz-transform: translateX(100%);
                -webkit-transform: translateX(100%);
                transform: translateX(100%);
            }
```

Marquee scrolls (cont'd):

CSS cont'd

```
 100% {
                -moz-transform: translateX(-100%);
                -webkit-transform: translateX(-100%);
                transform: translateX(-100%);
            }
        }

#header{
  width: 100vw;
  height: 45px;
  position:fixed;
  background: linear-gradient(to right 60% 55%, #829A8D, crimson,
#829A8D);
}
```

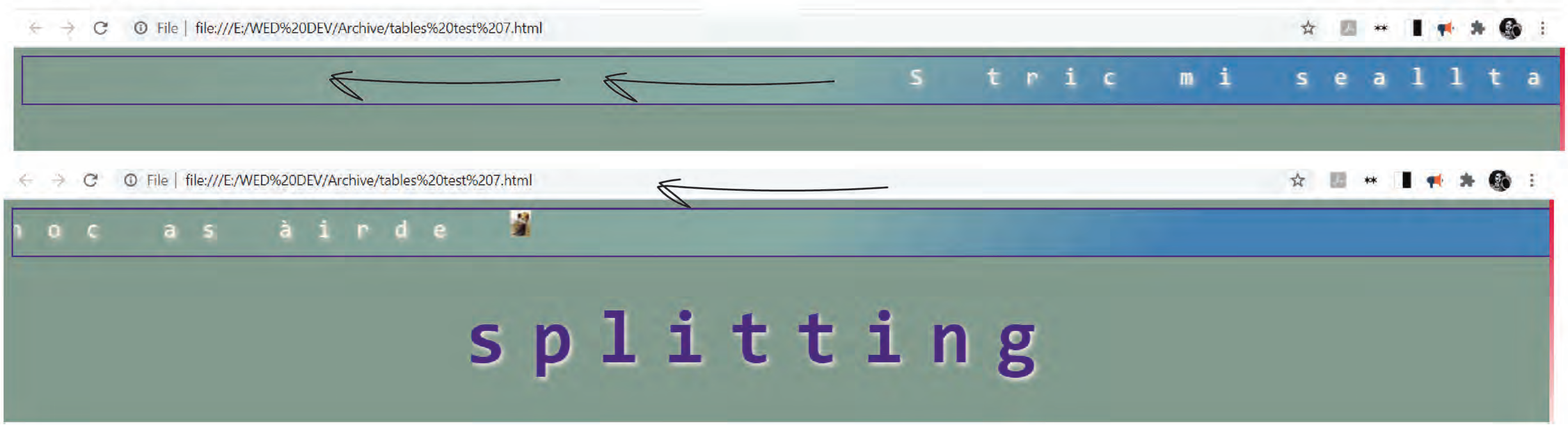
html

```
<body>

     <h1 id="demo"></h1>

<canvas id="canvas" width="300 px" height="300 px"
style="background-color:transparent">
   </canvas>
```

javascript

```
 <script>
function myFunction() {
   var element = document.div;
   element.classList.toggle("transparent");
}
</script>
```

Example:



Top parallax scroll:

>will look the same as a marquee, but the code is much more reliable and responsive: parallax will work better on mobile devices etc.

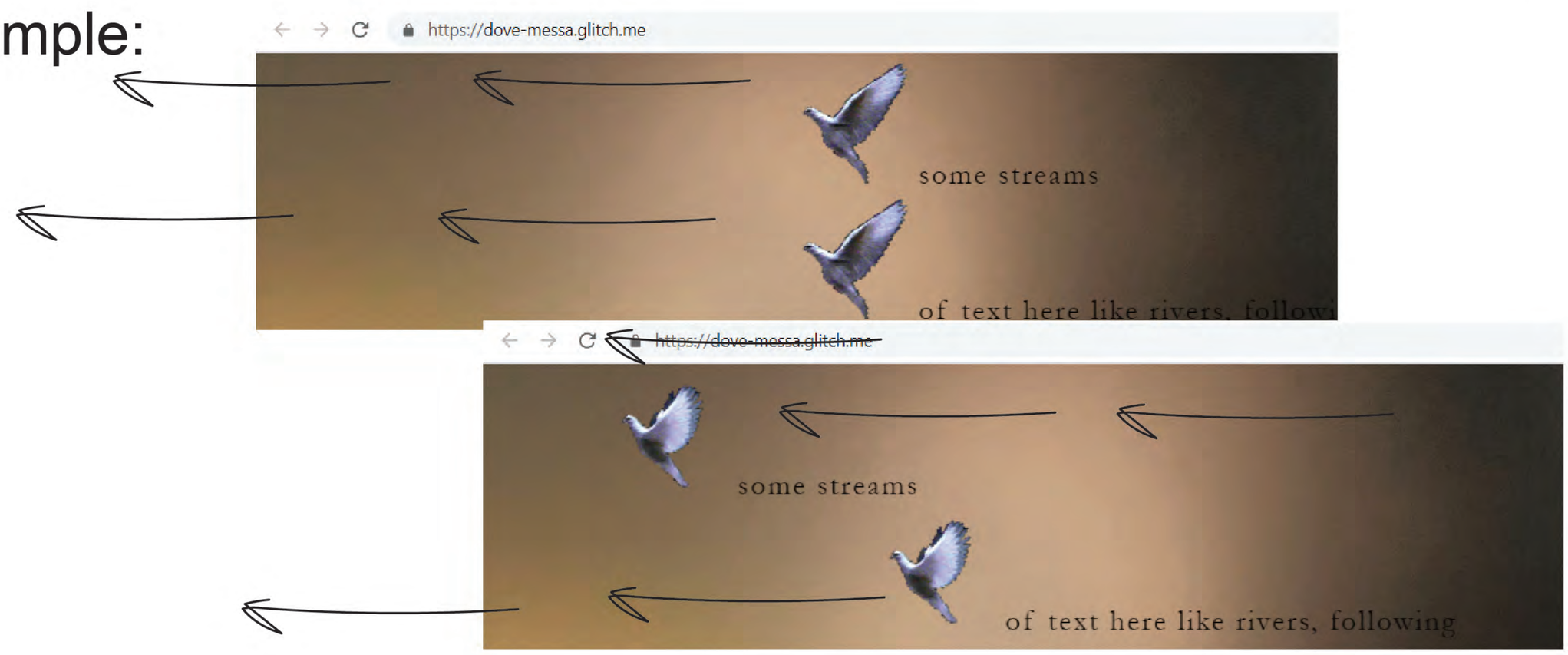
CSS

```
body, html {
  height: 100%;
}


.parallax {
  background-image: url('https://...');
  height: 500%;
  background-attachment: fixed;
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}
```

html

```
<body>

    <div class="parallax"></div>

    <div style="height:1px;background-color:#D8DFD7;
                                font-size:34px;
                            text-align:center;
                              font-family:...;
                            line-height: 50px;">

<p> scrolling div class contained here </p>


</div>
```

Example:



Flashing Button (Opacity Controls):

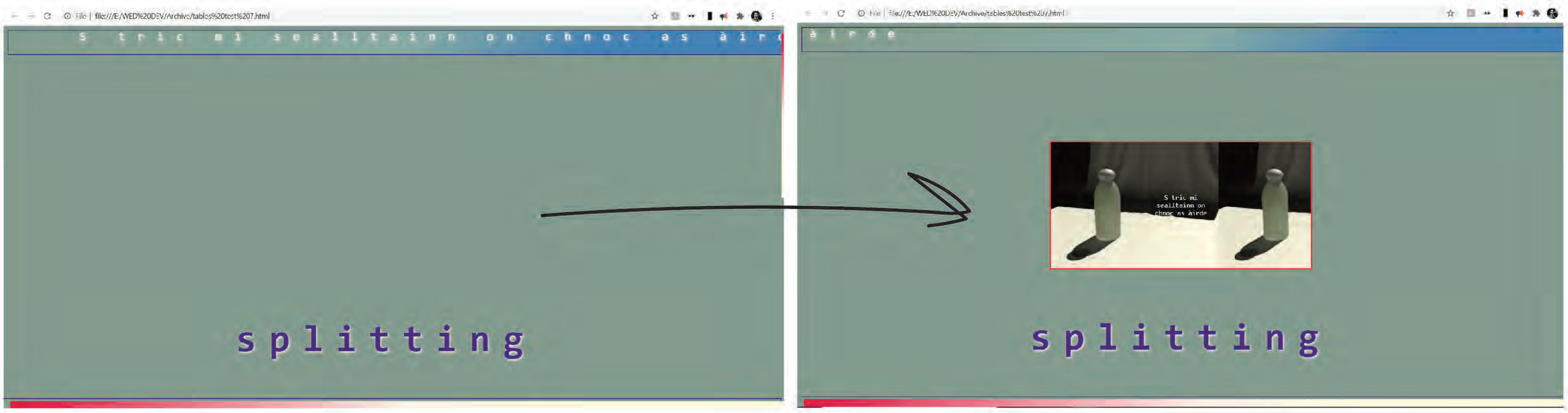
CSS

```
button {
          color: #fff;
        padding: 100px 1000px;
        font-family:Garamond, serif;
        background-im-
age:url('https://cdn.glitch.com/d515a603-a50b-461a-8383-b8afa68bf
c8c%2FIMG_0827.JPG?v=1608128767831');
        border: 3px solid red;
        transition-property: background, border;
        transition-duration: 1s, 2s;
      background-size: cover;
      opacity: 0;
    }
    button:hover {
        background-color:#D5B7AF;
        border-color: rosybrown;
      opacity: 1;
      background-size: cover;

    }
```

html

```
<div class="center">
    <button type="button">O' no one else</button>
     </div>
```

Example:



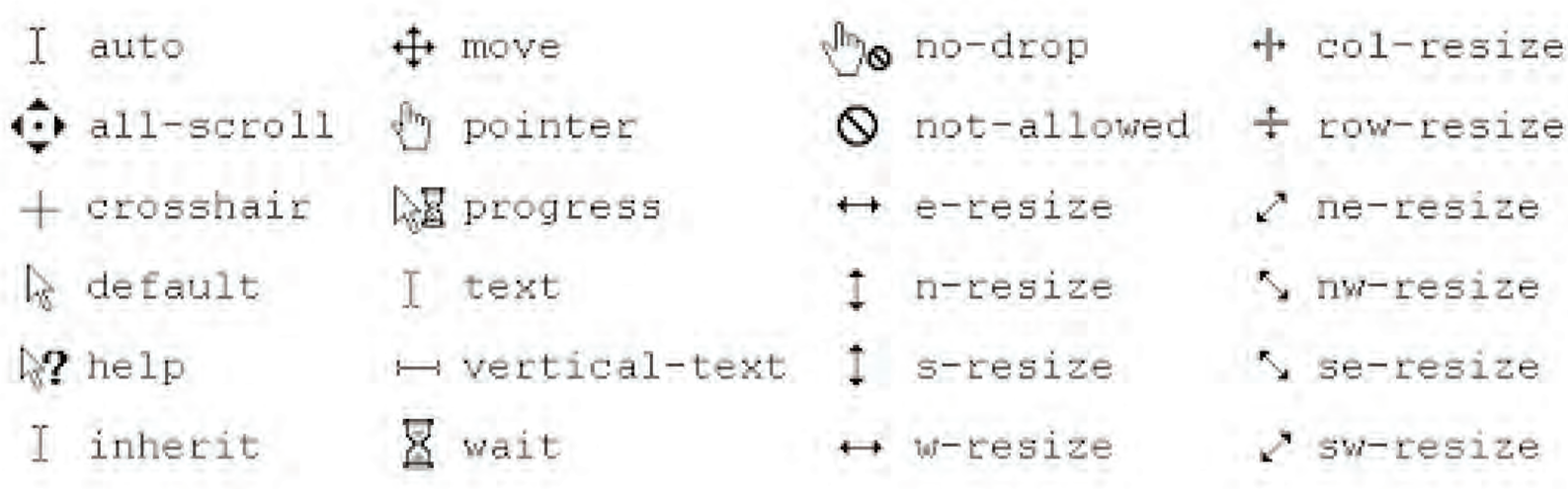
>button appears on hover- nice way of doing secret image galleries

Cursor Customisation (on specific elements):

html

```
<body>

<span style="cursor:auto">auto</span><br>

<span style="cursor:crosshair">crosshair</span><br>

<span style="cursor:default">default</span><br>

<span style="cursor:e-resize">e-resize</span><br>

<span style="cursor:help">help</span><br>

<span style="cursor:move">move</span><br>

<span style="cursor:pointer">pointer</span><br>

<span style="cursor:progress">progress</span><br>

<span style="cursor:s-resize">s-resize</span><br>

<span style="cursor:se-resize">se-resize</span><br>

<span style="cursor:sw-resize">sw-resize</span><br>

<span style="cursor:text">text</span><br>

<span style="cursor:w-resize">w-resize</span><br>


</body>
```

Example:



Flickering Tables (for a Home Navigation)

Example: ‘panel’ appears on hover.

CSS

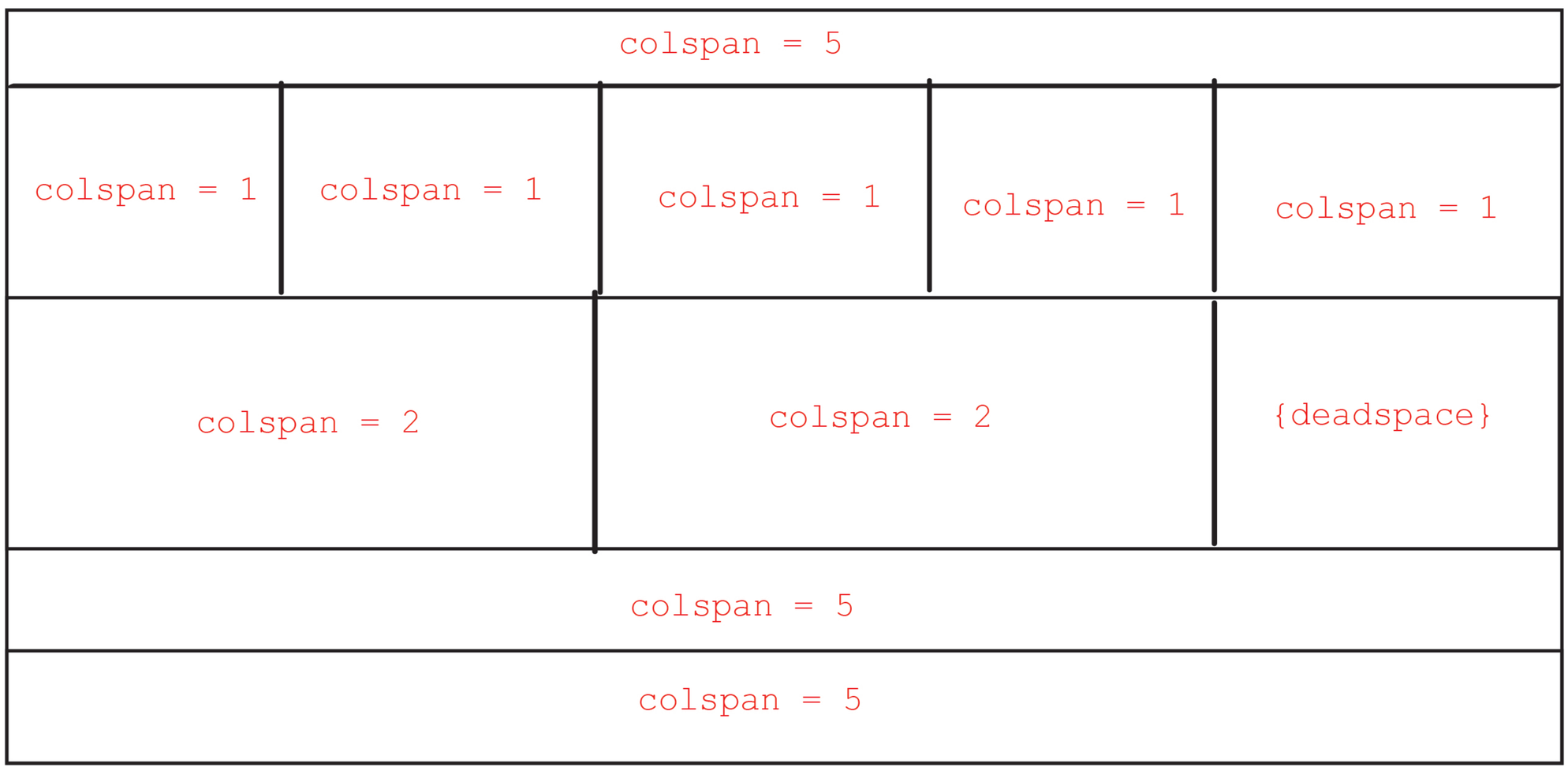
```
@keyframes visibility {
    from {opacity:100;}
    to {opacity:0;}
    }

.visibleworld {
    padding: 0px;
    opacity: 100;
    background: linen;
    }

.visibleworld:hover {
    opacity: 0;
    animation: visibility 2s ease;
    animation-fill-mode: forwards;
    }
```

```
table, th, td {
  border:2px solid indigo;
  border-collapse: collapse;
  text-align: center;
  font-size: 25px;
  font-family:monospace;
}

table {
     width: 100vw;
     height: 100vh;
}

td {
  opacity: 0;
}

td:hover {
  opacity: 1;
  color: linen;
  background-color: rosybrown;

}
```

html

```
<table>
  <tr>
    <td align="center" colspan="5">TEXT OR IMAGE</td>

  </tr>

  <tr>
  <td align="center" colspan="1">TEXT OR IMAGE</td>
  <td align="center" colspan="1">TEXT OR IMAGE</td>
  <td align="center" colspan="1">TEXT OR IMAGE</td>
  <td align="center" colspan="1">TEXT OR IMAGE</td>
  </tr>


<tr>
   <td align="center" colspan="5">TEXT OR IMAGE</td>
</tr >

  <tr>
<td align="center" colspan="5">TEXT OR IMAGE </td>
</tr >
```

Trying to understand html tables:



>some structural notes: have to layer up columns correctly, otherwise you'll get blank boxes ('deadspace').

>can be quite complex to align everything properly. Must make sure that the opacities are correct! Also, works v. nicely over a larger back-ground image (make the td. qualities all "transparent")

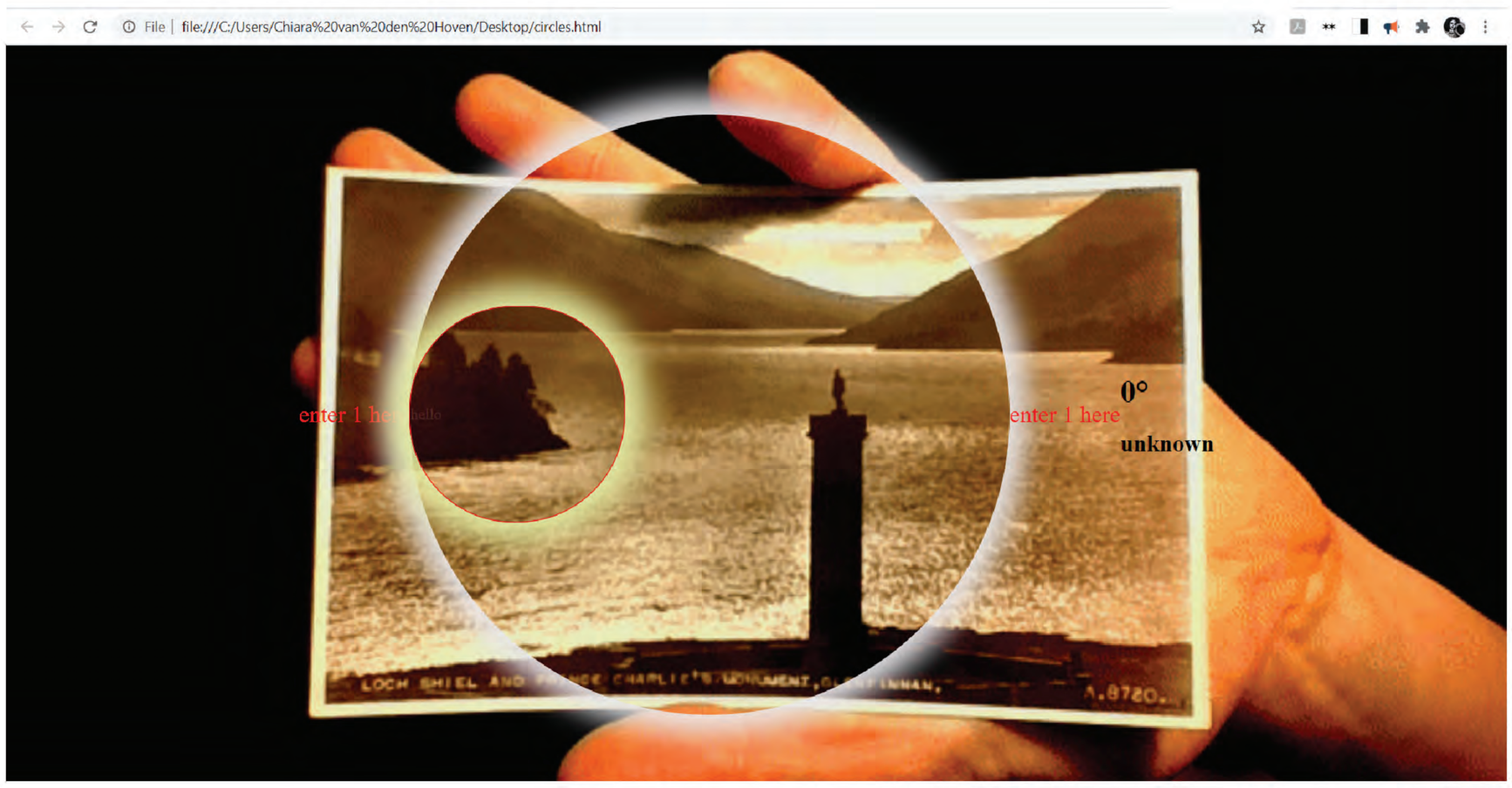
Examples:

colspan=1

colspan=5

>’colspan’ means columns/spanning (how many columns are in that section of the table. Have to work out how to make that general for an even (square) table. Do they all have to be the same? Seems to work well to add another div class (<tr>...</tr>) to add rows with separate controls...if you don’t put a <br> they seem to be part of the same block!

Responsive Nested Circles:



CSS

```
.circle {
      display: flex;
      justify-content: left;
      align-items: center;
      width: 300%;
      height: 300%;
    }

  ._1 {
    width: 40vw;
    height: 40vw;
    border-radius: 50%;
    /*border: 1px solid red;*/
    box-shadow:  0 0 10px 10px #e7e5e7;
    transition: 5s;
    background: ("...");


background-repeat: no-repeat;
background-size: cover;
    background-attachment: fixed;
  }

  ._1:hover {

background-position: center;

  }

  ._2 {
    width: 30vw;
    height: 30vw;
    border-radius: 52% 45% 55% 48%;
    box-shadow: 0px 0px 46px 25px #f1dbdb;
    transition: 10s;
    background: url ("...");

  }
```

```
._2:hover {
        width: 40vw;
        height: 40vw;
        border-radius: 55% 70% 70% 45%;
        transition: 10s;

    }
```

html - simple solution

```
<div class="circle _1">

        <div class="circle _4">


          <p>  .... </p>

          </div>


      </div>

    </div>
```

Notes on usability for these circles:

> Nesting: can put different elements inside one another (using the 40vw controls). As long as the sizing matches up, they will sit concentrically and contract together. Must also make the animation frames the same timing.

>Interacting with other JS elements: in example below I have placed the visible for a clock inside the centre circle: this sometimes glitches if the vw contraction is smaller than the elements- will flick and push to left... probably some flex-box solution to this, but I'm not sure how to prevent!

>Box-shadow: can keep transitions very smooth with box-shadow enlargements (half colour, half transparent)

```
.circle {
        display: flex;
        justify-content: left;
        align-items: center;
        width: 300%;
        height: 300%;
      }


      ._1 {
        width: 35vw;
        height: 35vw;
        border-radius: 50%;
        /*border: 1px solid red;*/
        box-shadow:  0 0 2px 2px rgb(216, 238, 238);
        transition: 5s;


      }

      ._1:hover {


        width: 20vw;
        height: 20vw;
        border-radius: 52% 45% 55% 48%;
        box-shadow: 2px 10px 46px 25px linen;
        transition: 5s;


      }
```

Clickthrough Narrative Using Javascript: COMPLEX

Initial Notes:

>Trying to create a story which is responsive to the viewer's touch in a certain area of the page. Subtitle-like text should appear at the bottom of page, like a translation...

>where/how to integrate the button? How to make the instruction clear to viewers (ie. can it be disguised?)

Will have to set up a relationship between the no. of clicks ('if' statement) and the text displayed > seems like the simplest way.
EG:

```
var clicks = 0;

 function onClick() {
   clicks += 1;

   var message = "...text 1...";
   if(clicks==1)
     { message = "... text 2...";}
else if(clicks==2)
   {message ="...text 3...";}
else if(clicks==3)
   {message =”...text 4... ";}
```

clickthruFOR CARDS.html 2 | clickthru.html | index.html

C: > Users > Chiara van den Hoven > Desktop > clickthruFOR CARDS.html > html > head >

```
<script>
var clicks = 0;

 function onClick() {
    clicks += 1;
    var message = "...";
    if(clicks==1)
      { message = ".....";}
else if(clicks==2)
    {message ="This is a drying oven";}
else if(clicks==3)
    {message ="(for wood?). ";}
else if(clicks==4)
      {message = "This is a silhouette of a dancing horse";}
else if(clicks==5)
      {message = "on a red background.";}
else if(clicks==6)
      {message = "This is a wooden embossed sign.";}
else if(clicks==7)
      {message = "Or this is a very vivid shade of red,";}
else if(clicks==8)
      {message = "which could withstand water or sunlight. ";}
else if(clicks==9)
      {message = "Or";}
else if(clicks==10)
      {message = "this is cloth,";}
else if(clicks==11)
      {message = "printed by her grandfather,";}
else if(clicks==12)
      {message = "a block printer.";}
else if(clicks==13)
      {message = "This is a painting of a sunrise";}
else if(clicks==14)
      {message = "on what appears to be a river mouth";}
```

>linking up statements: unfortunately it can only move in a linear direction, there's no way to go back to the previous statement...not sure how to solve this without making whole interlinking pages instead...I think that would be a bit too heavy! Could look into the meta tags and see if I can change something about the state of the page when it's refreshed?

Clickthrough Narrative Using Javascript: cont'd

CSS and Button controls:

```
#message {
    height: 56px;
    width: 800px;
    position: fixed;
    bottom: 5px;
    left: 50%;
    margin-left: -400px;
    overflow-y: hidden;
    text-align: center;
    font-family: 'Times New Roman', Times, serif;
    font-weight: bolder;
    font-size: 24px;
    color: rgb(12, 12, 10);
    -webkit-text-stroke: ;
    paint-order: stroke fill;
    text-shadow: 2px 2px 5px #f5f827;

}
```

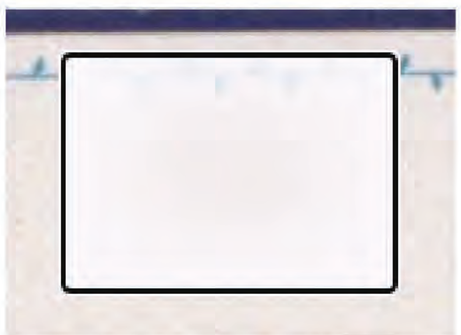
[early test]

> putting thick(ish) black stroke on the yellow text: must remember to call this class 'message', otherwise it won't show in the script!

```
#clicky {
    position: fixed;
    top: 50px;
    left: 10%;
    width: 800px;
    height: 10px;
    margin-left: -200px;
    text-align: center;
    border:none;


}

button {
    background-color:transparent;
    border: transparent;
    color: rgb(247, 234, 212);
    box-shadow:inset 2px 2px 5px rgb(245, 245, 222);
    box-shadow: 2px 2px 5px rgb(245, 245, 222);
    border-radius: 50%;
    padding: 20px;
    margin: 4px 2px;
  cursor: pointer;}

```

>Formatting the clicker: using normal button controls, but can't seem to get rid on the black outline after it has been activated... is that because of the script?

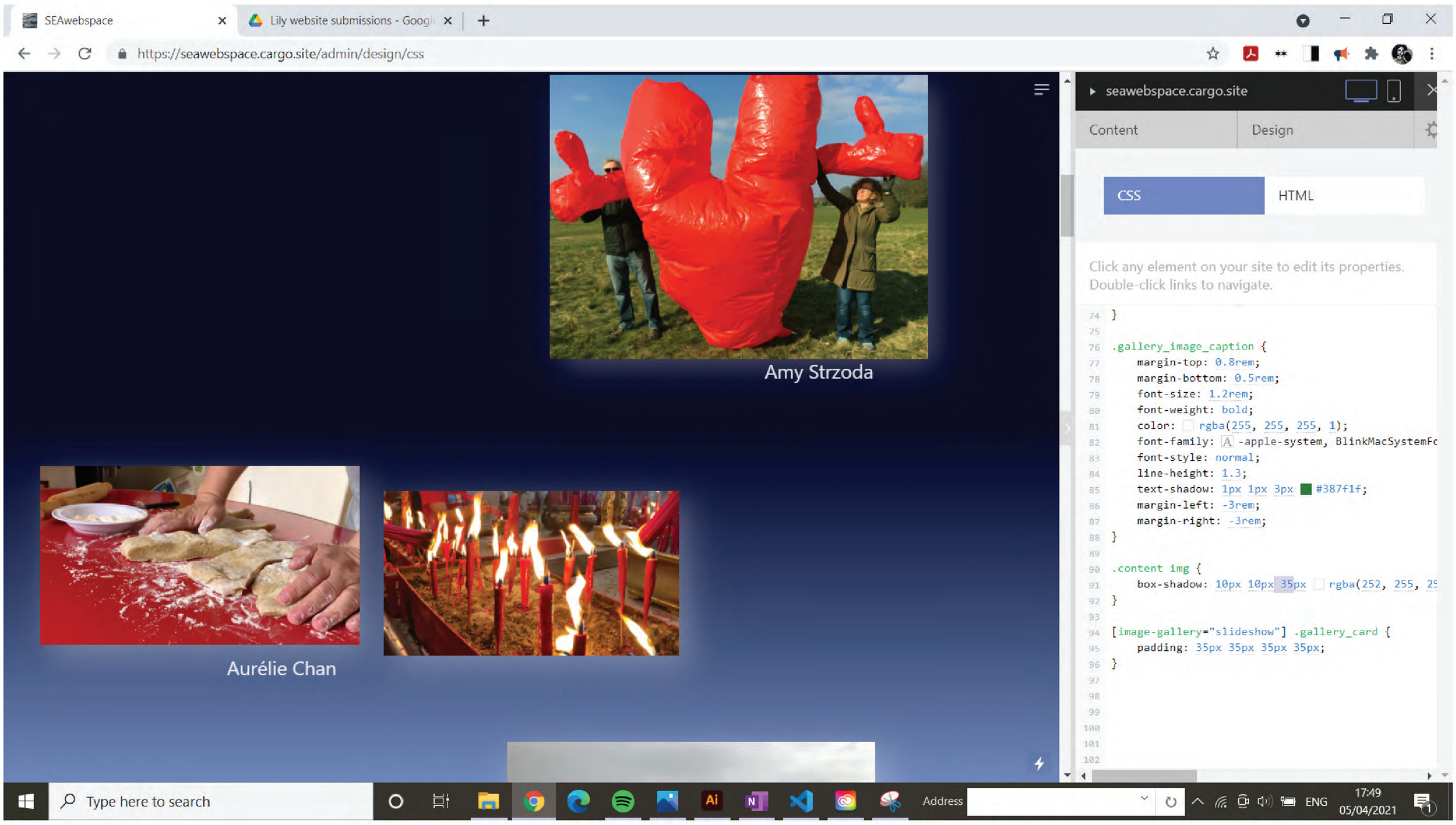
Looks like this:



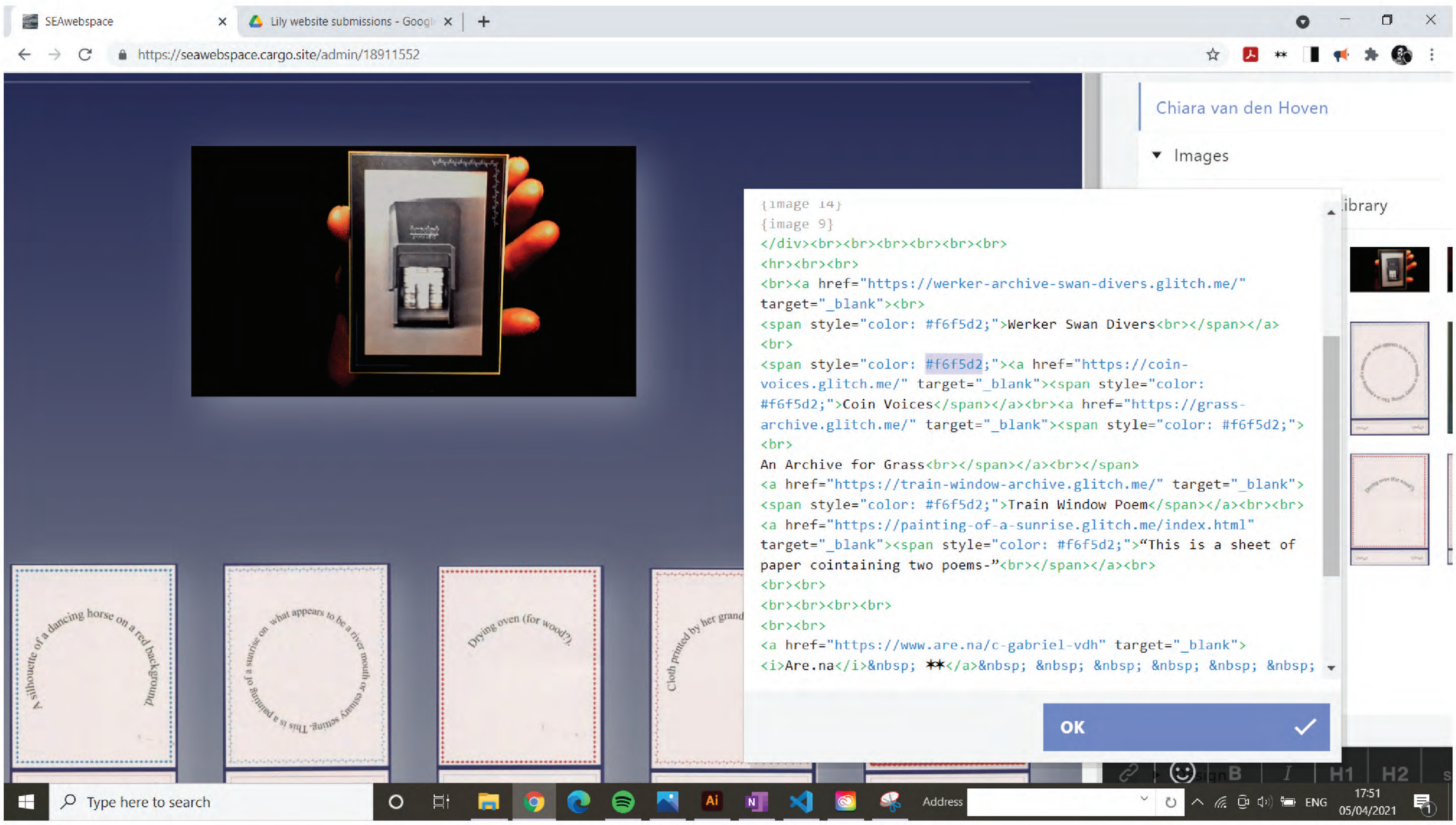
a bit boring!!

{might change to a regular 'click command...}

Programming some parts of the SEA Webspace:

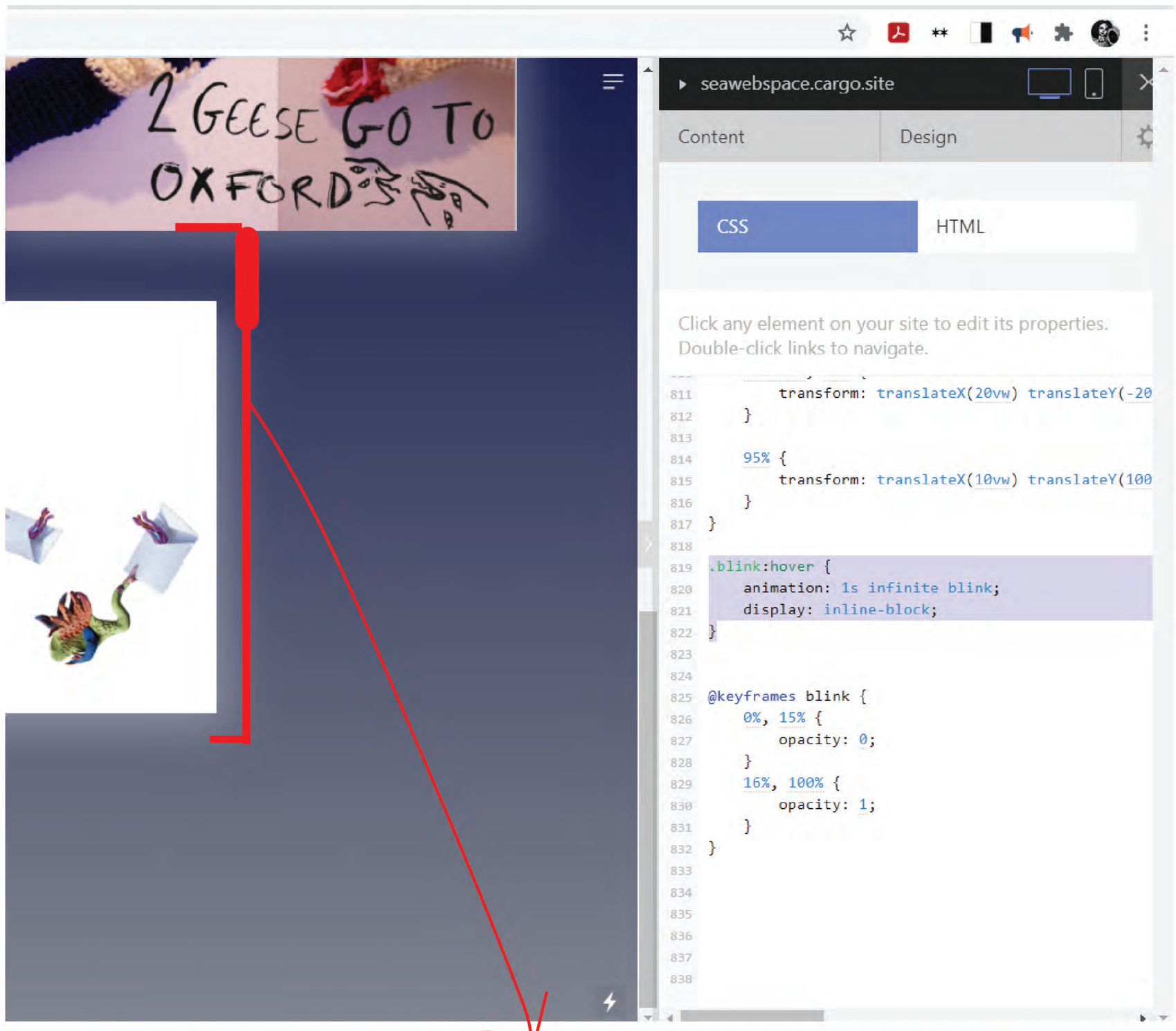


>trying to format a smooth ‘gallery’ style homepage, with works placed in a cascade. Can be a bit difficult to format for different screen sizes - need to test if this is really mobile compatable!

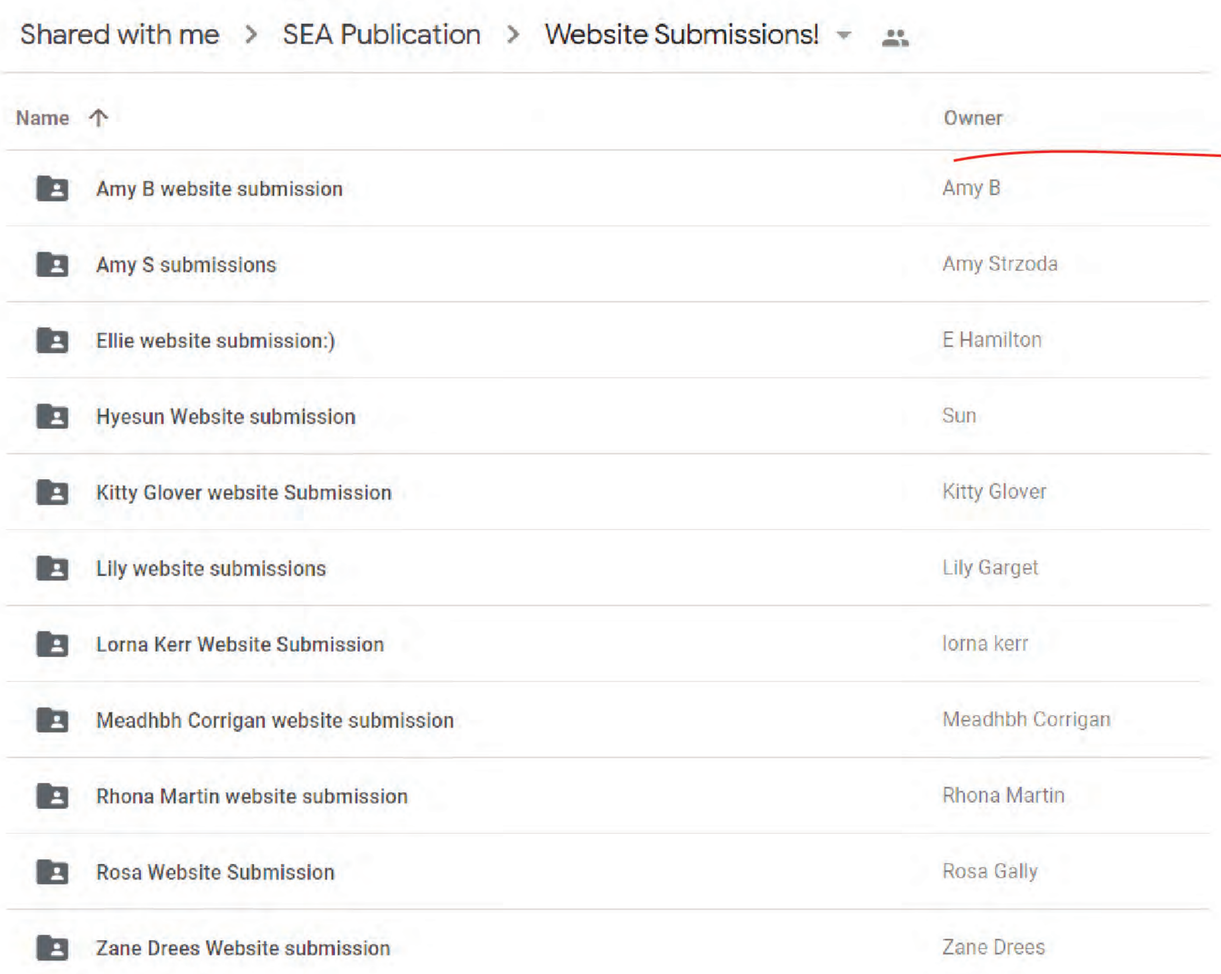


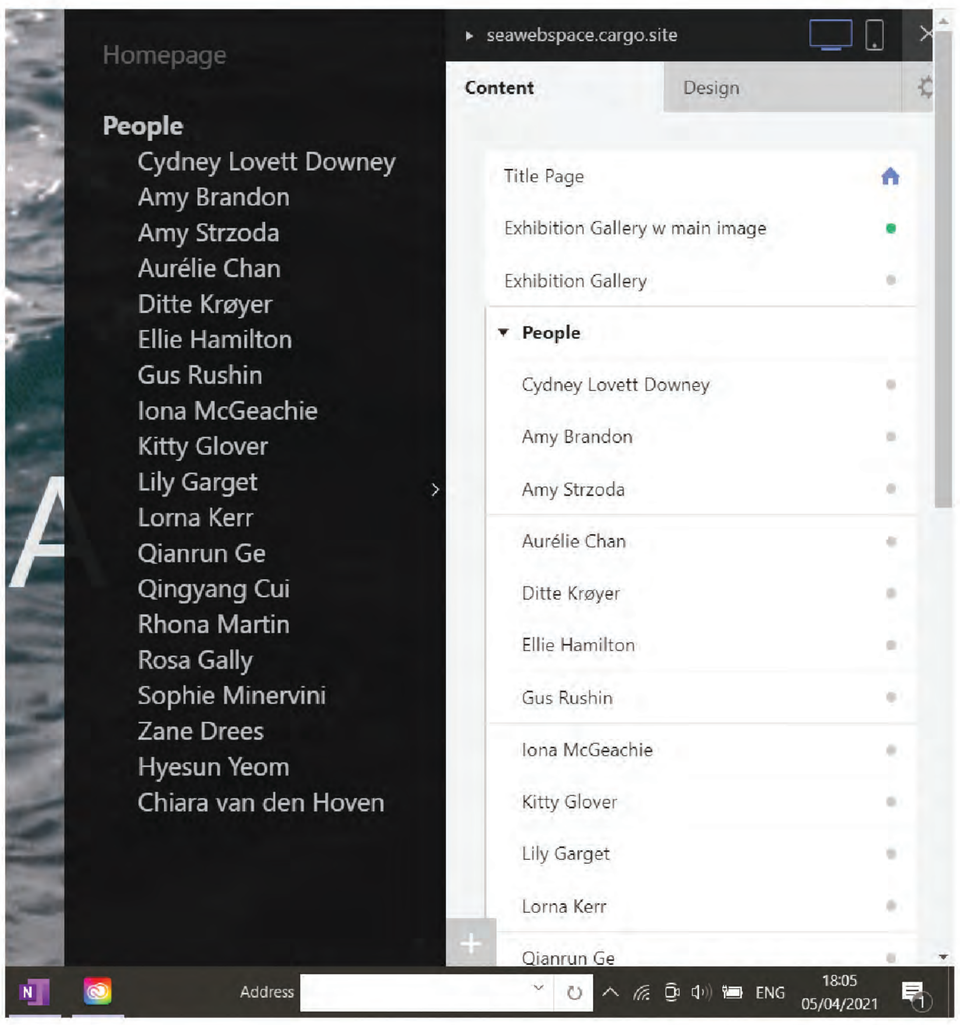
>CSS for individual pages- very smooth editing with Cargo!

Responsive image loading:



>trying to make the images load smoothly and appear spontaneously - because there are loads of high-quality images, they can appear unevenly: no way around it with the timeframe we have, but adding scrolling animations (javascript) and box-shadows (CSS) kind of disguises this!



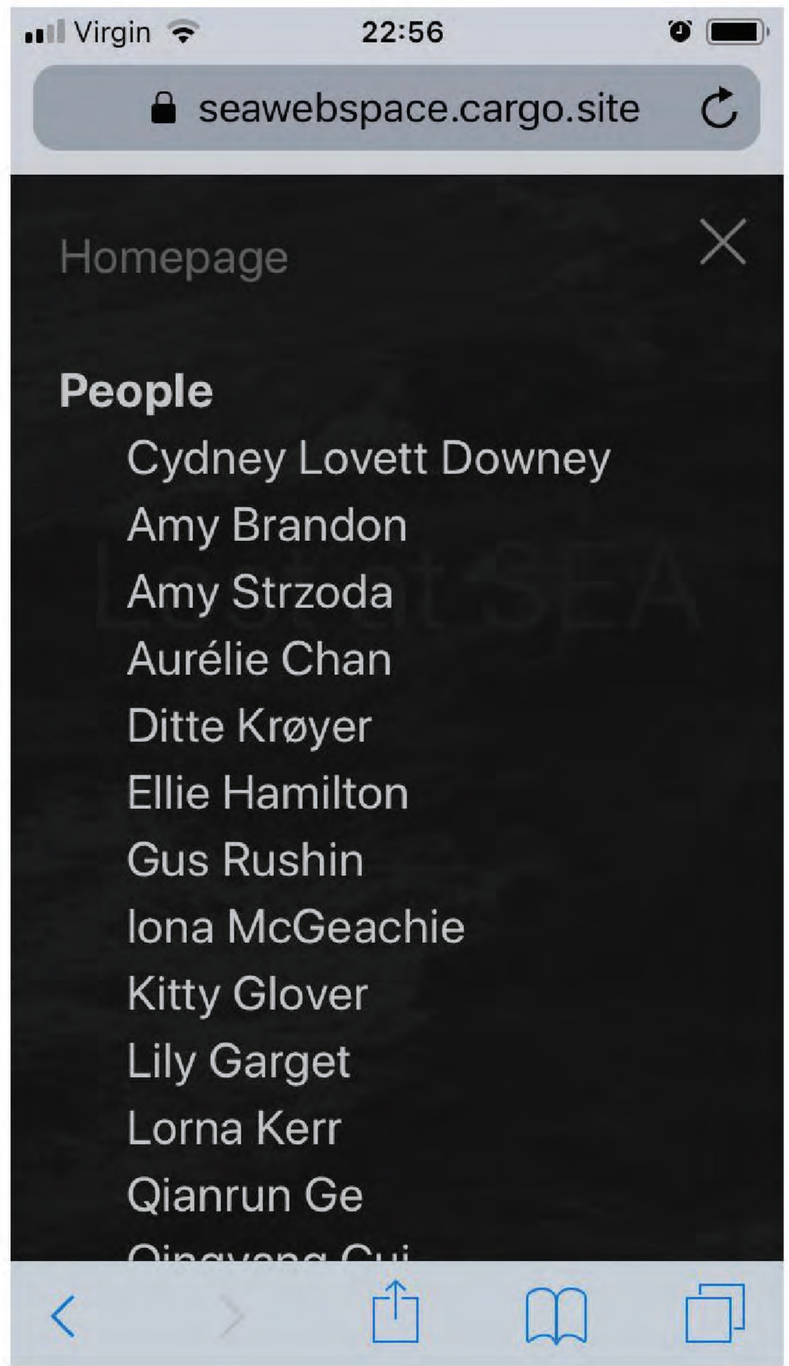


>creating a file system to upload classmate's work onto the site (should probably alphabetise this in future.

>linking and matching homepage and menu to personal profiles: need to keep a good handle on the profiles and their order, otherwise this will get messy !

Testing mobile compatability and troubleshooting:

>Home page



>Menu tweaked to be readable fullscreen (size now 200%, allows you to read names better).

>Gallery alignment needs some more work!

> all seems ok apart from the ‘exhibition gallery’ part, which is a bit scrambled in the middle section (due to text spacing inadvertedly moving images).

>only easy solution to this is to make the names an image caption instead, and the whole image into a clickable link: this would ‘stick’ the text to the image in the mobile version. This will take a while to process through though, and will likely be difficult to handle with the image galleries...