

Clock.app - README

Intro

Following the hint in moodle (the new Date() function), I have made research and learned how to make a working clock in javascript, css & html.

Learning Materials

1. W3Schools (n.d.) *JavaScript Get Date Methods*. W3Schools. Available at: https://www.w3schools.com/js/js_date_methods.asp (Accessed: 15 January 2025)
2. Horstmann, C. S. (2020) 'Constructing Dates', in *Modern JavaScript for the Impatient*. Boston: Addison-Wesley Professional, p. 105
3. (Web Dev Simplified) (7 May, 2019) 'Build A Clock With JavaScript', *YouTube video*, Available at: <https://www.youtube.com/watch?v=Ki0XXrKIHY> (Accessed: 15 January 2025)

I started the project from doing research and exploring what is available in the documentation and the book I had at the time, *Modern JavaScript for the impatient*. It also had a chapter on new Date() method in JS, and it helped me to understand the basics and core concepts that the project is build on.

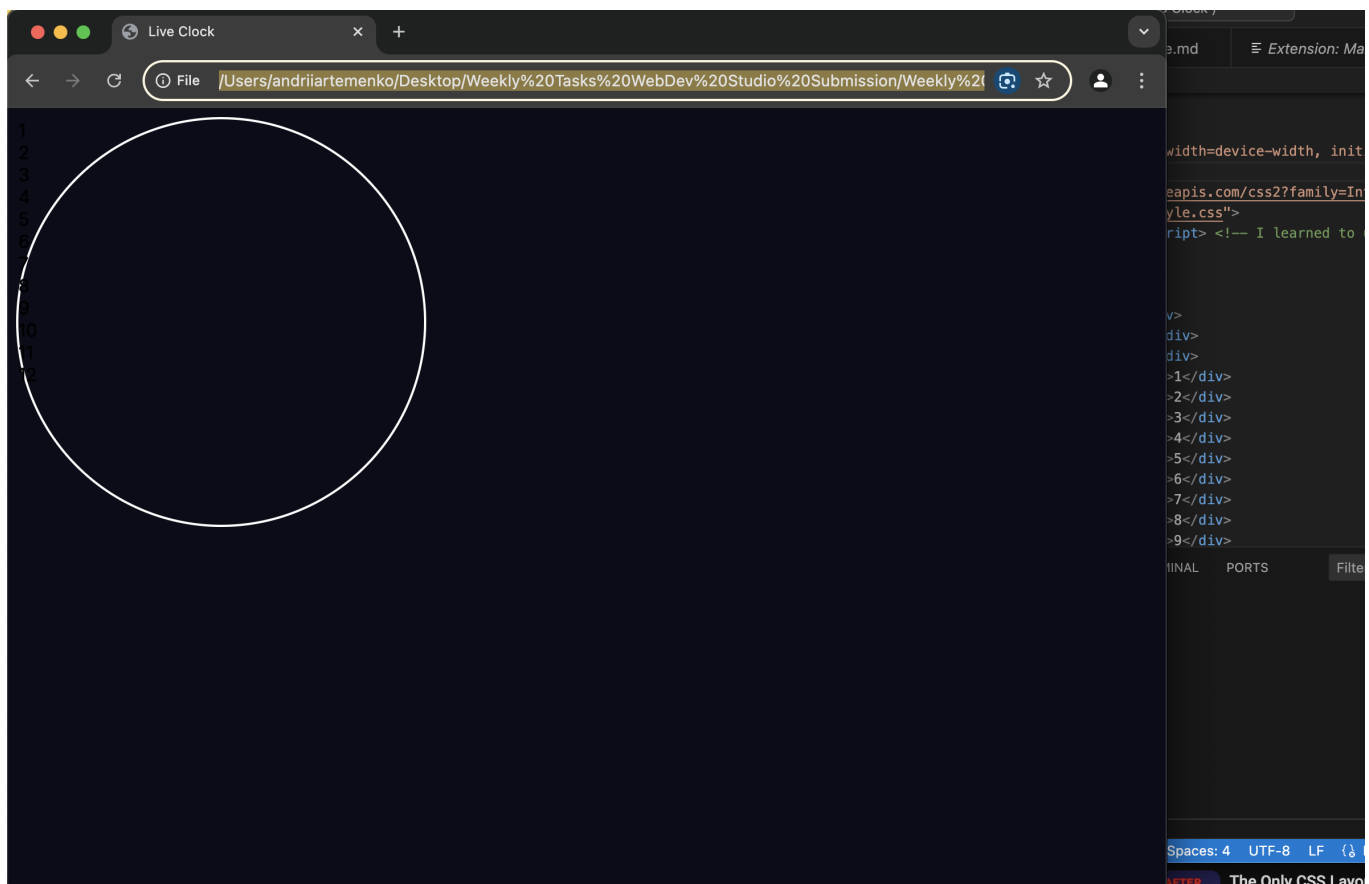
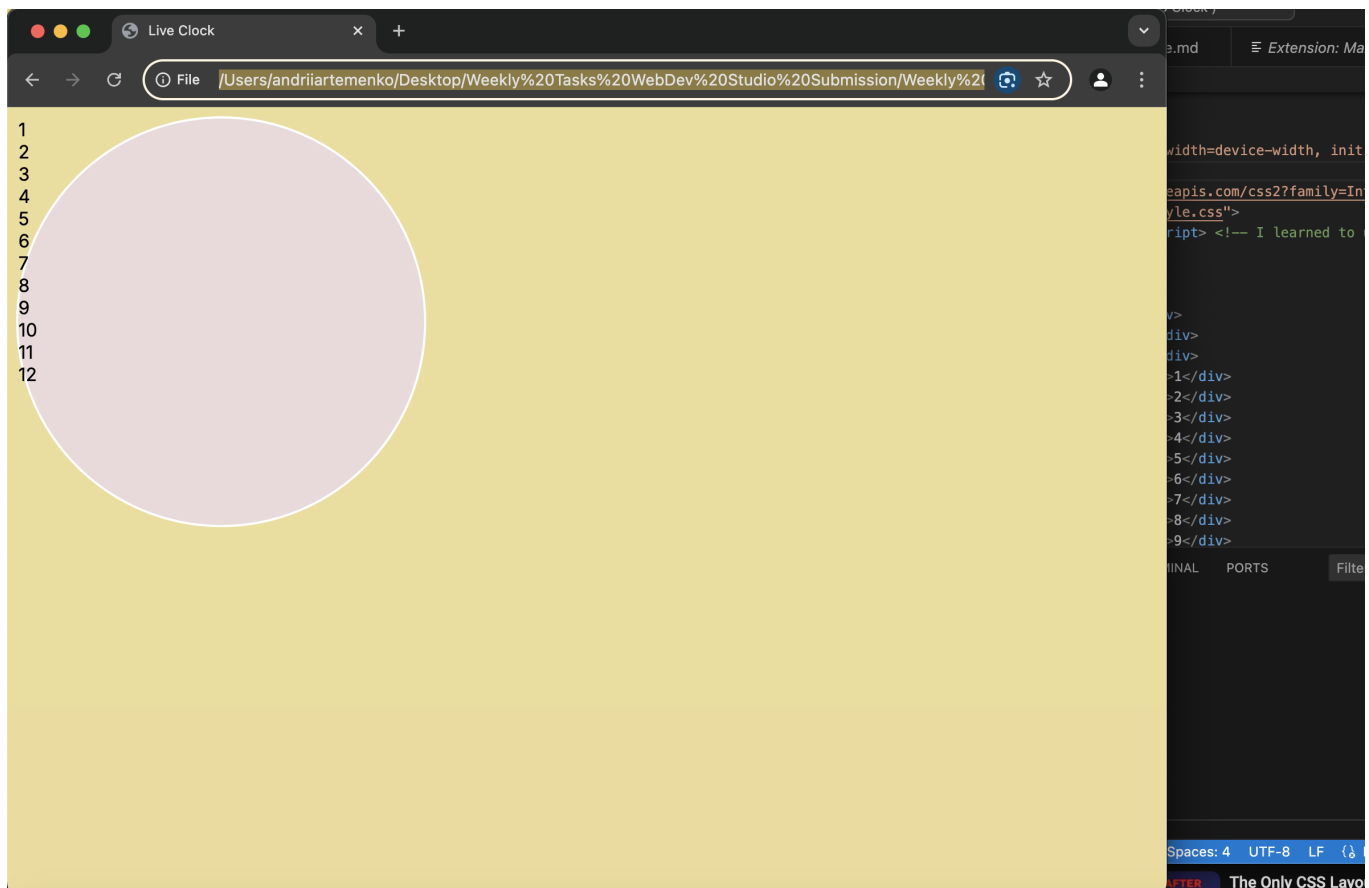
After I have read the documentation and book section, I searched for some practical implimentations on YouTube. The tutorial I have found is using new Date() method and getSeconds specifically to initialise the current date and time. The video also addresses some potential bugs and imperfections that can disrupt the animation of the clock.

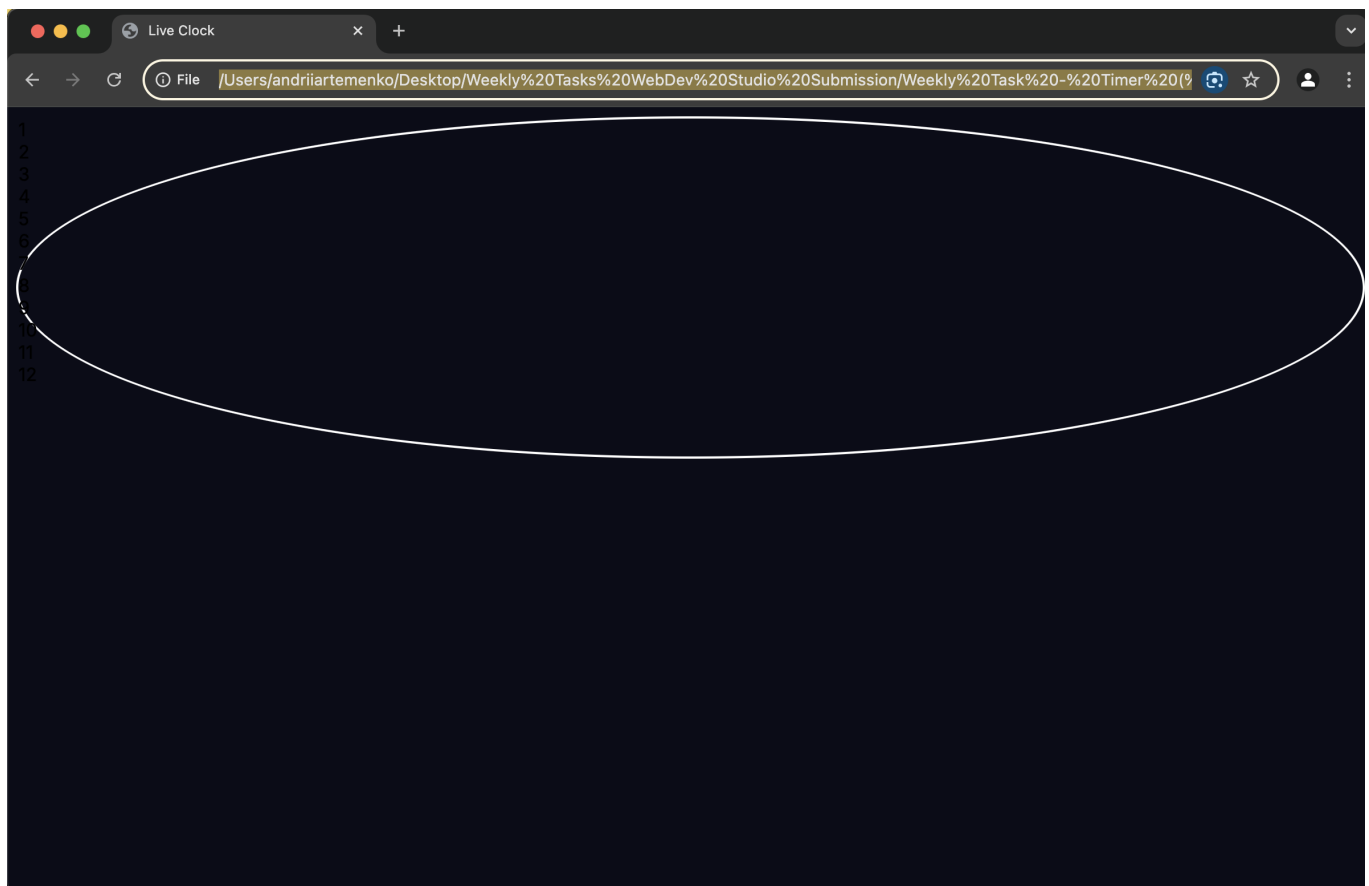
HTML & CSS Design

Here I simply build a set of divs and name the classes

```
index.html > html > body > div.clock > div.number.number12
9  </head>
10 <body>
11   <div class="clock">
12     <div class="hand hour"></div>
13     <div class="hand minute"></div>
14     <div class="hand second"></div>
15     <div class="number number1">1</div>
16     <div class="number number2">2</div>
17     <div class="number number3">3</div>
18     <div class="number number4">4</div>
19     <div class="number number5">5</div>
20     <div class="number number6">6</div>
21     <div class="number number7">7</div>
22     <div class="number number8">8</div>
23     <div class="number number9">9</div>
24     <div class="number number10">10</div>
25     <div class="number number11">11</div>
26     <div class="number number12">12</div>
27   </div>
28 </body>
29 </html>
```

Initial Styling and experementing with color, shape, form





```
app.js  ⓘ Readme.md  ≡ Extension: Markdown Preview Enhanced  # style.css  ×  □  ..

# style.css > .clock .number
2      box-sizing: border-box;
3      font-family: "Inter", sans-serif;
4  }
5
6  body{
7      background-color: #ecdd98;
8  }
9
10 .clock {
11     width:360px;
12     height:360px;
13     background-color: rgb(235, 217, 217);
14     border-radius: 50%;
15     border: 5px solid rgb(34, 20, 12);
16     position:relative;
17 }
18
19 .clock .number {
20     position:absolute;
21     width:100%;
22     height:100%;
23 }
```

Here I'll use a method which was showed in the tutorial (Web Dev Simplified, May 7, 2019) but I used a similar method before, as well. The method is simply to build out a box container and use transform rotate() to position elements along the circle in equal amounts of distance.

```
52 }
53
54 .clock .number7 {
55 |   --rotation: 210deg;
56 }
57
58 .clock .number8 {
59 |   --rotation: 240deg;
60 }
61
62 .clock .number9 {
63 |   --rotation: 270deg;
64 }
65
66 .clock .number10 {
67 |   --rotation: 300deg;
68 }
69
70 .clock .number11 {
71 |   --rotation: 330deg;
72 }
```

PROBLEMS

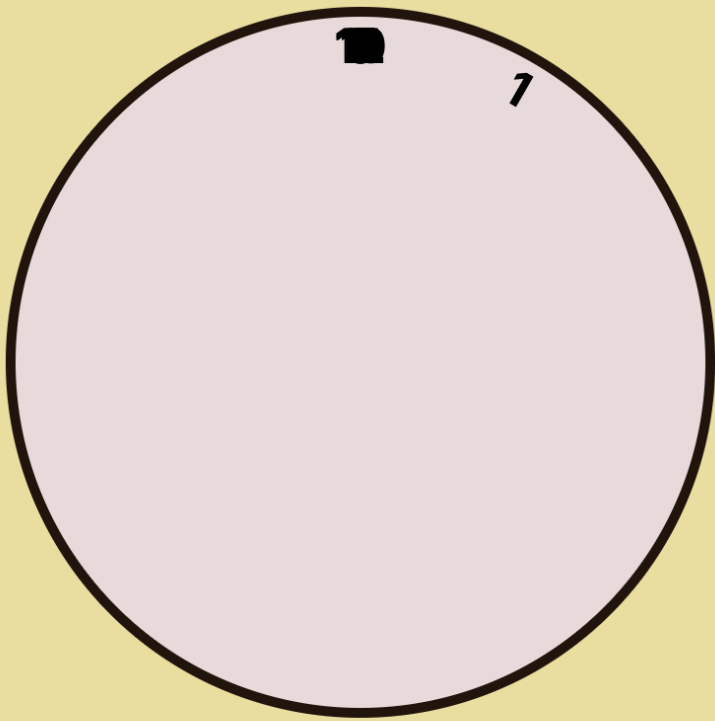
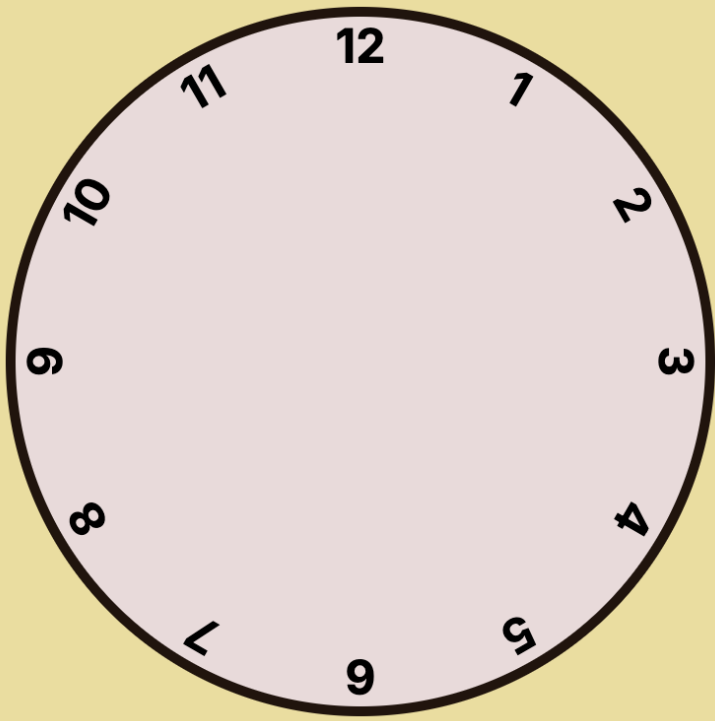
OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

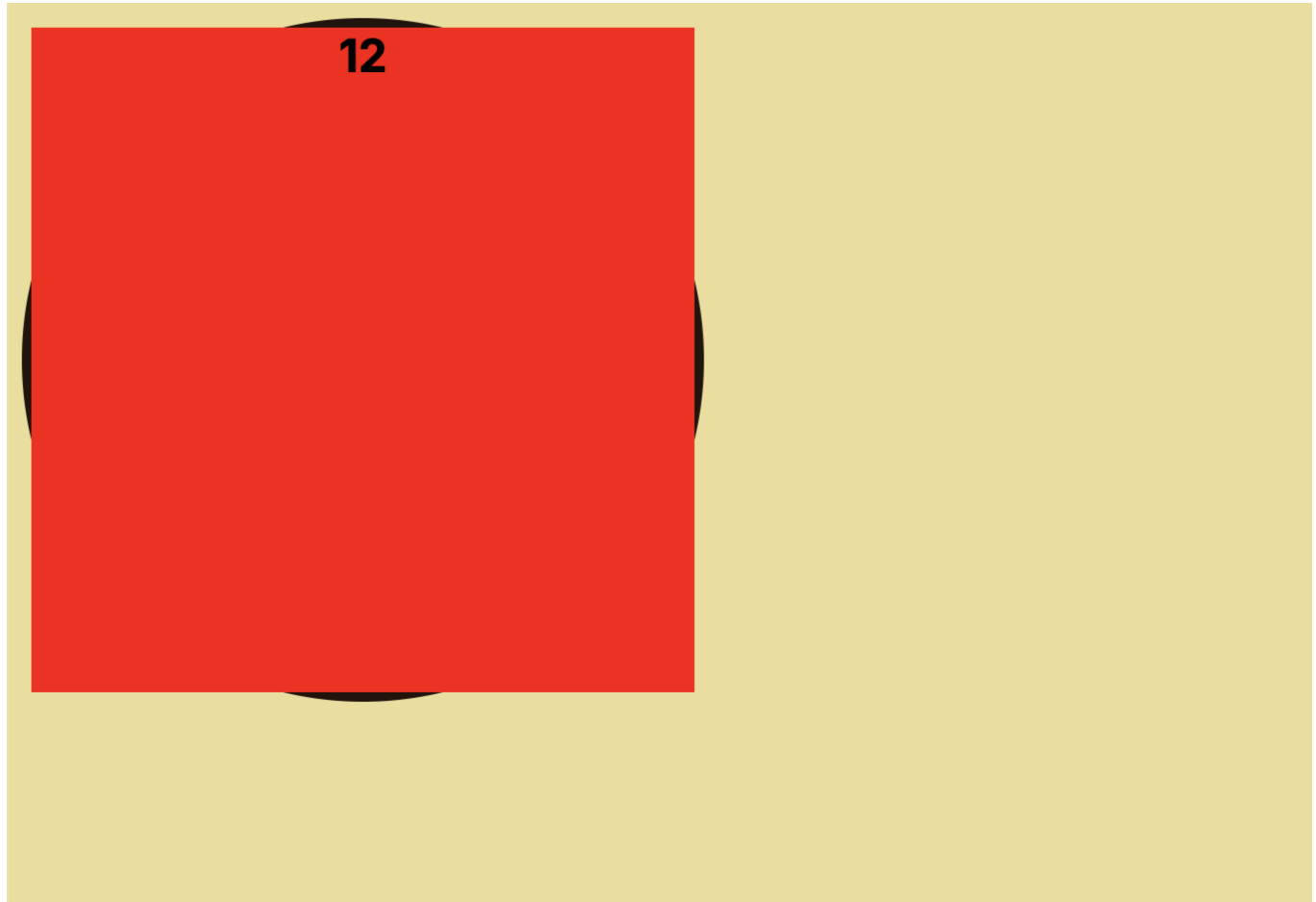
Filter (e.g. text, !exclu.



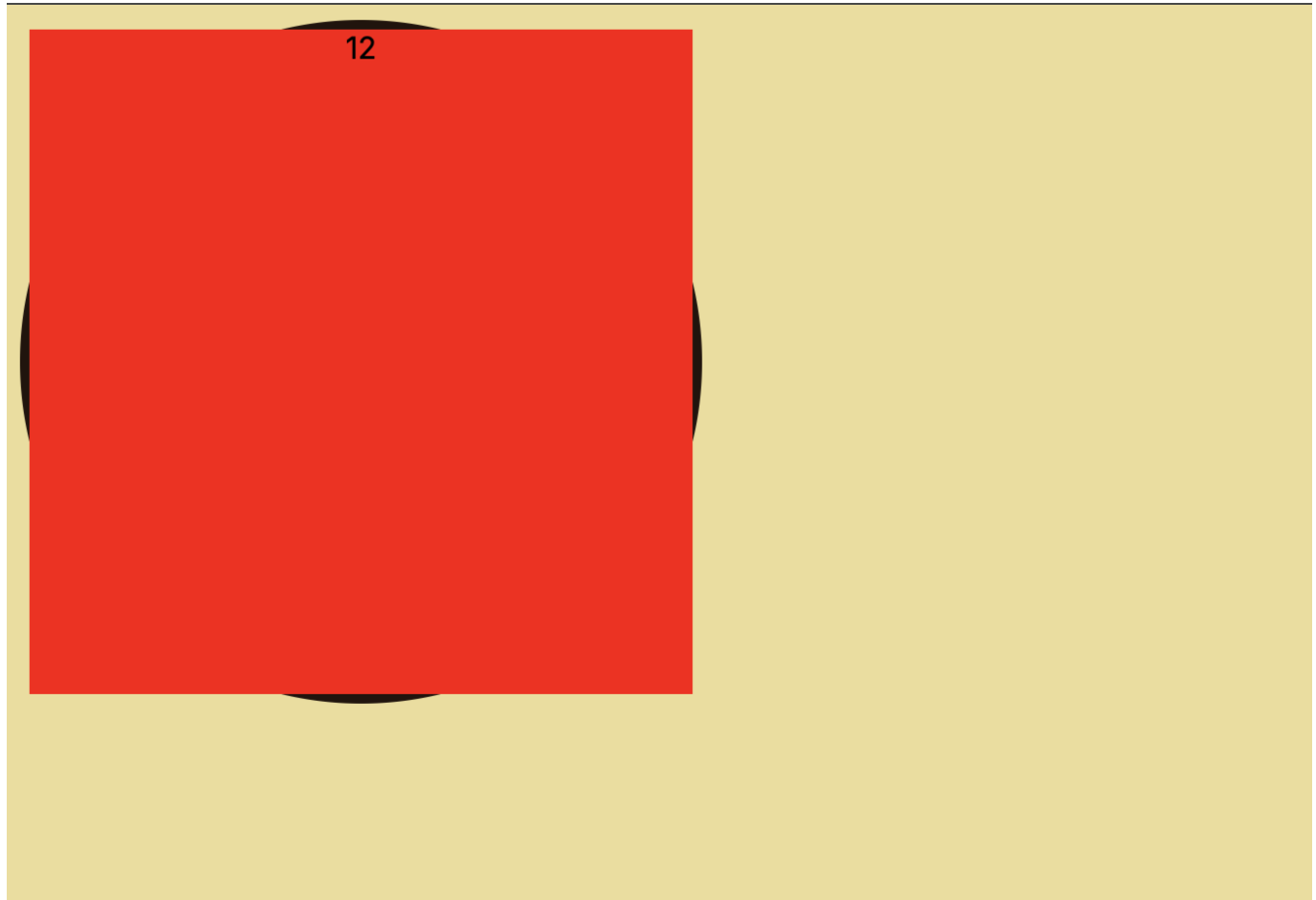
```
app.js | README.md | Extension: Markdown Preview Enhanced | # style.css x
```

```
# style.css > *
1  *, *::after, *::before {
2      box-sizing: border-box;
3      font-family: "Inter", sans-serif;
4      font-size: 24px;
5      font-weight: bold;
6  }
7
8  body{
9      background-color: #ecdd98;
10 }
11
12 .clock {
13     width:360px;
14     height:360px;
15     background-color: rgb(235, 217, 217);
16     border-radius: 50%;
17     border: 5px solid rgb(34, 20, 12);
18     position:relative;
19 }
20
21 .clock .number {
22     position:absolute;
```

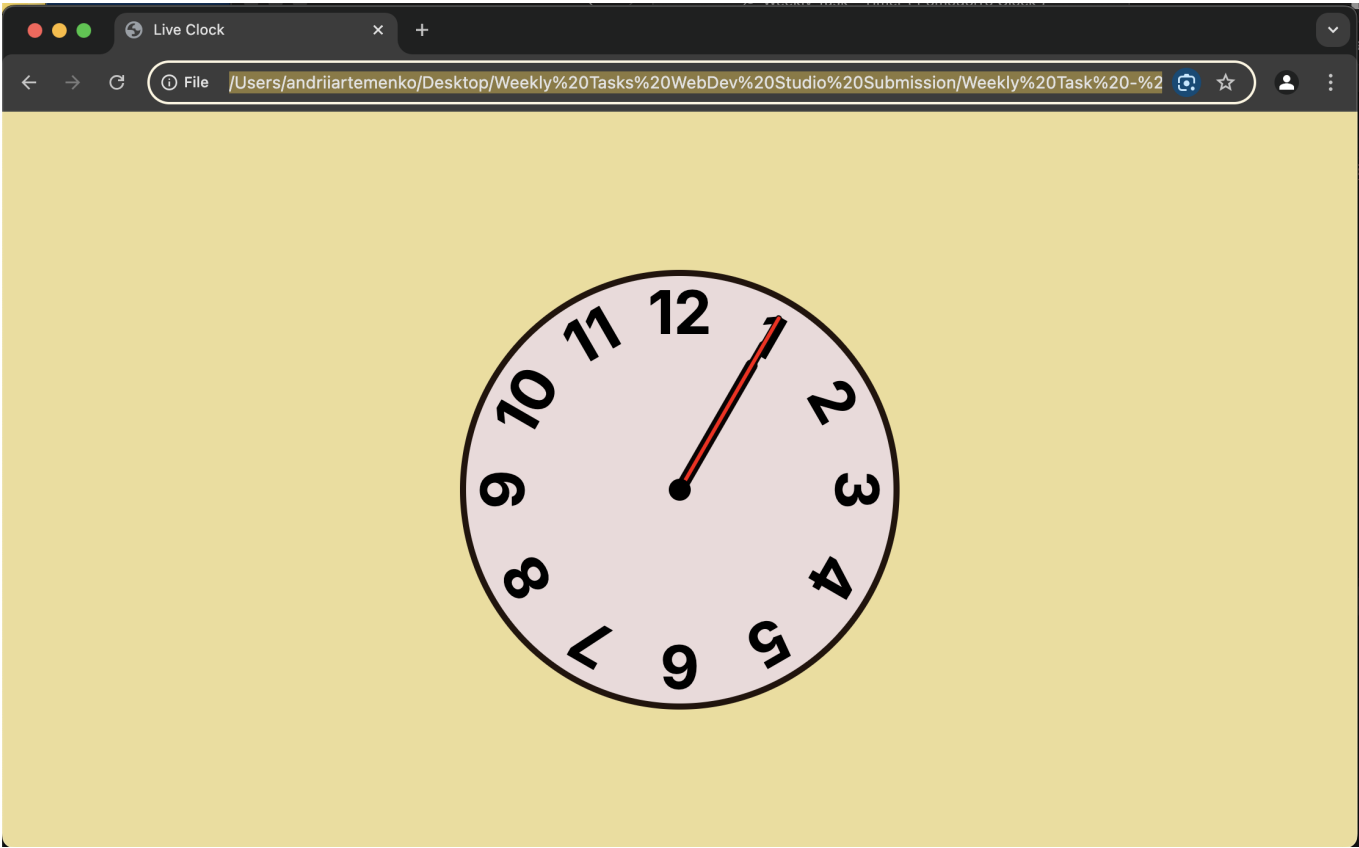
PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | Filter (e.g. text, !exclu... | ⌵

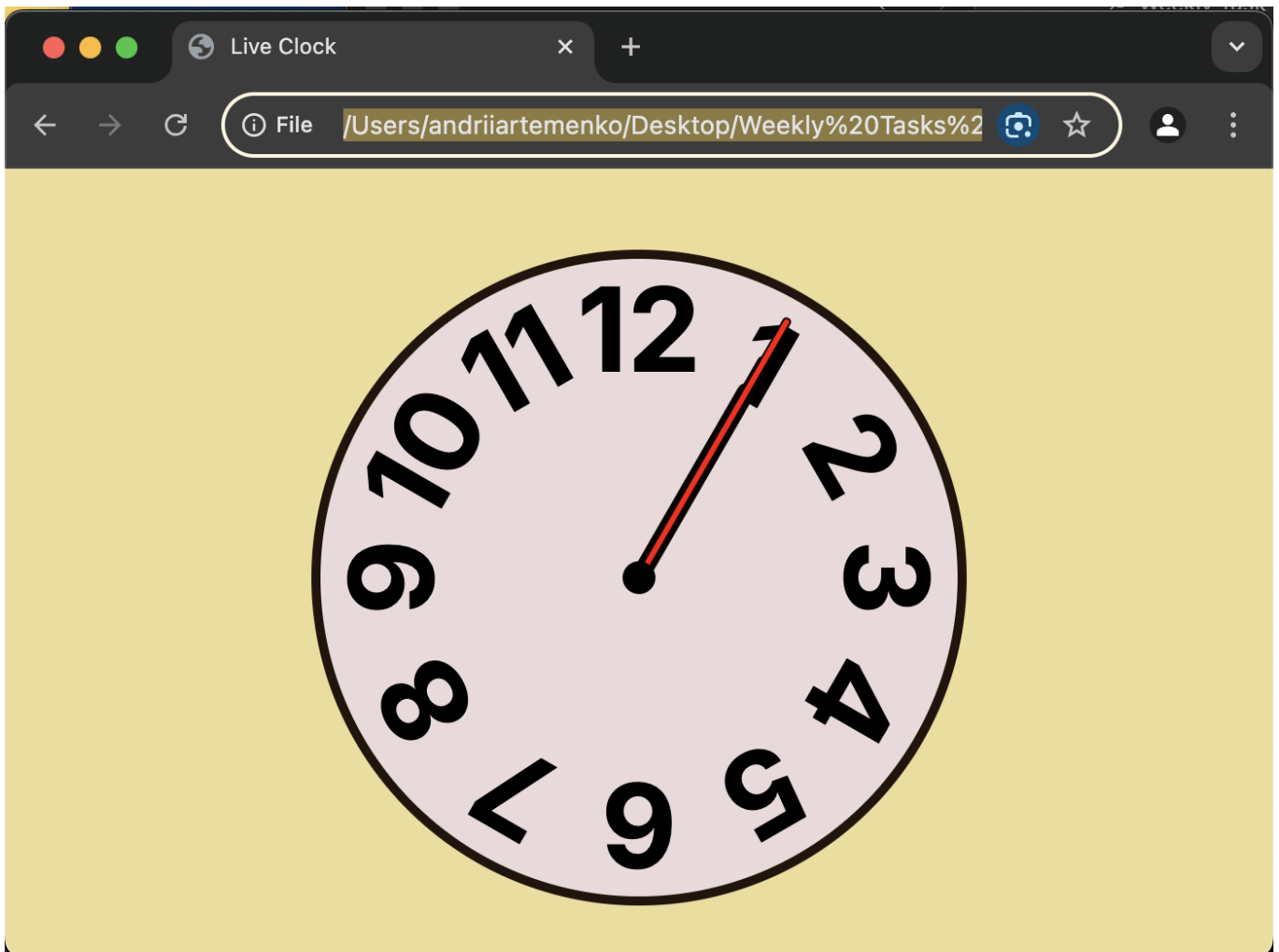


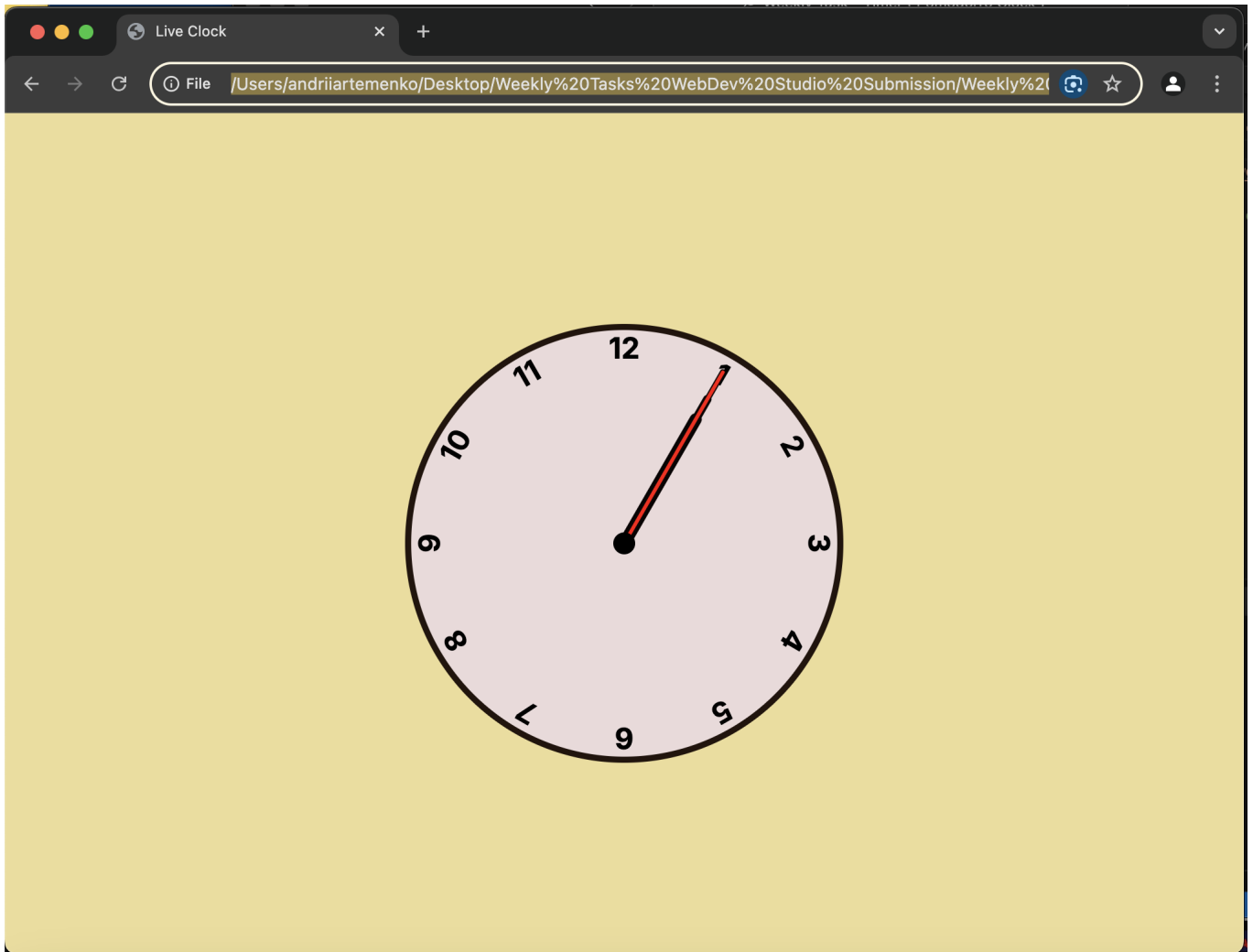
```
.. 5 app.js  ⓘ Readme.md  ≡ Extension: Markdown Preview Enhanced  # style.css ×
# style.css > .clock .number
7      background-color: #ecdd98;
8  }
9
10     .clock {
11         width:360px;
12         height:360px;
13         background-color: rgb(235, 217, 217);
14         border-radius: 50%;
15         border: 5px solid rgb(34, 20, 12);
16         position:relative;
17     }
18
19     .clock .number {
20         position:absolute;
21         width:100%;
22         height:100%;
23         background-color: red;
24         text-align: center;
25     }
```

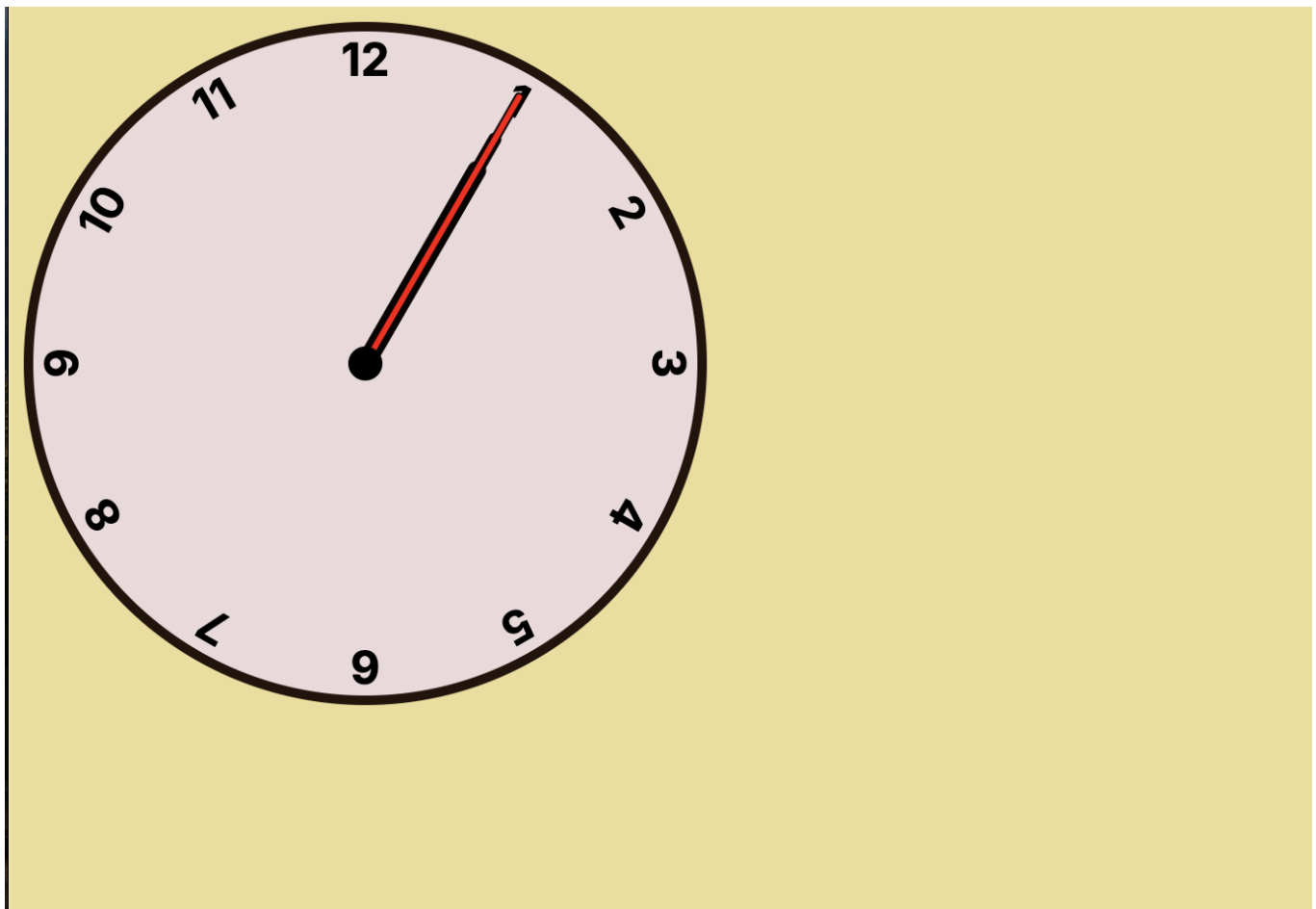


Then we" do some line styling for the hands









```
75     --rotation: 30;
76     position: absolute;
77     bottom: 50%;
78     left: 50%;
79     background-color: black;
80     border: 1px solid rgb(13, 5, 5);
81     border-top-left-radius: 10px;
82     border-top-right-radius: 10px;
83     transform: translateX(-50%) rotate(calc(var(--rotation)*1deg));
84     transform-origin: bottom;
85     z-index: 10;
86 }
87
88 .clock .hand.second {
89     width: 5px;
90     height: 47%;
91     background: red;
92 }
```

PROBLEMS

OUTPUT

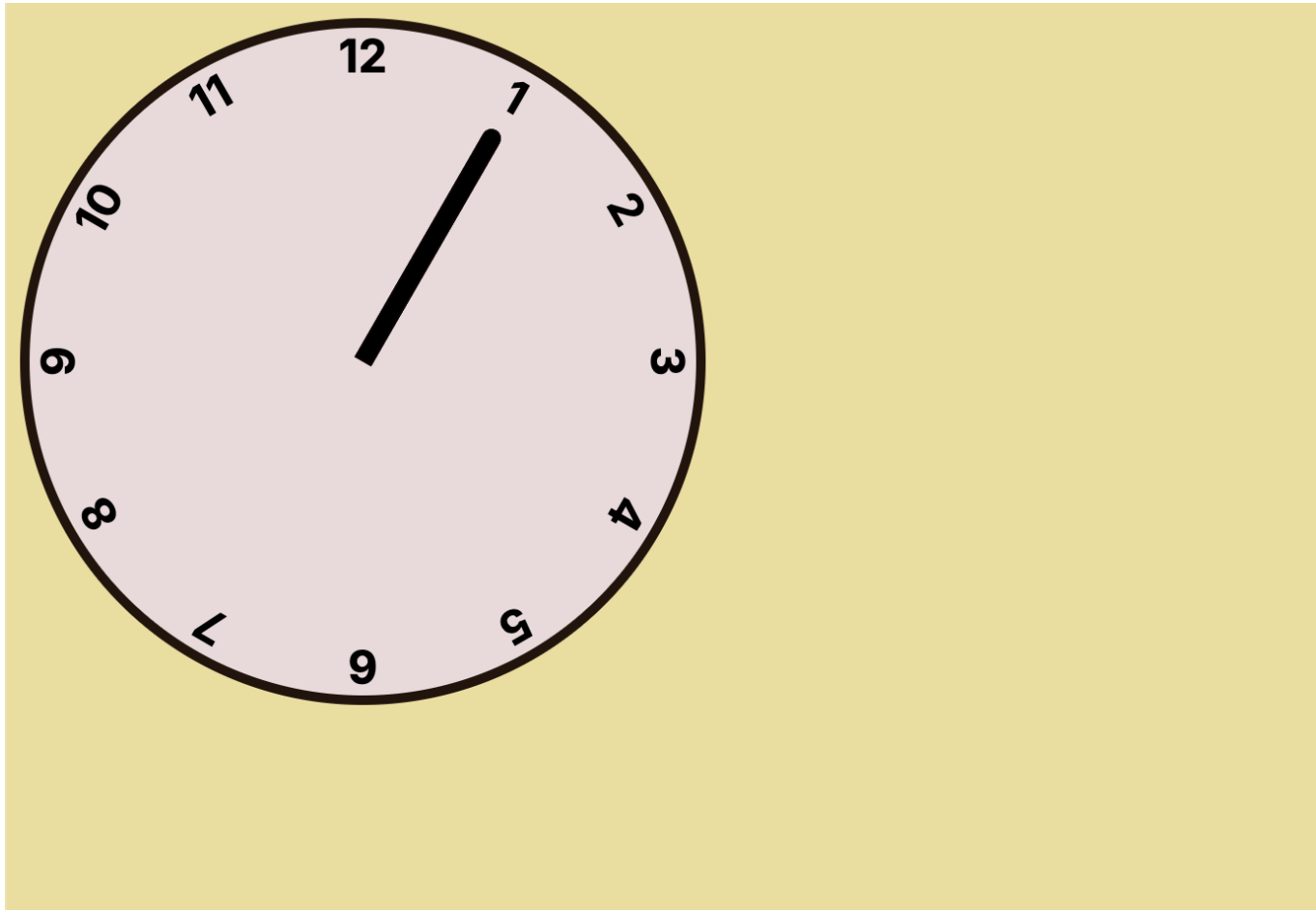
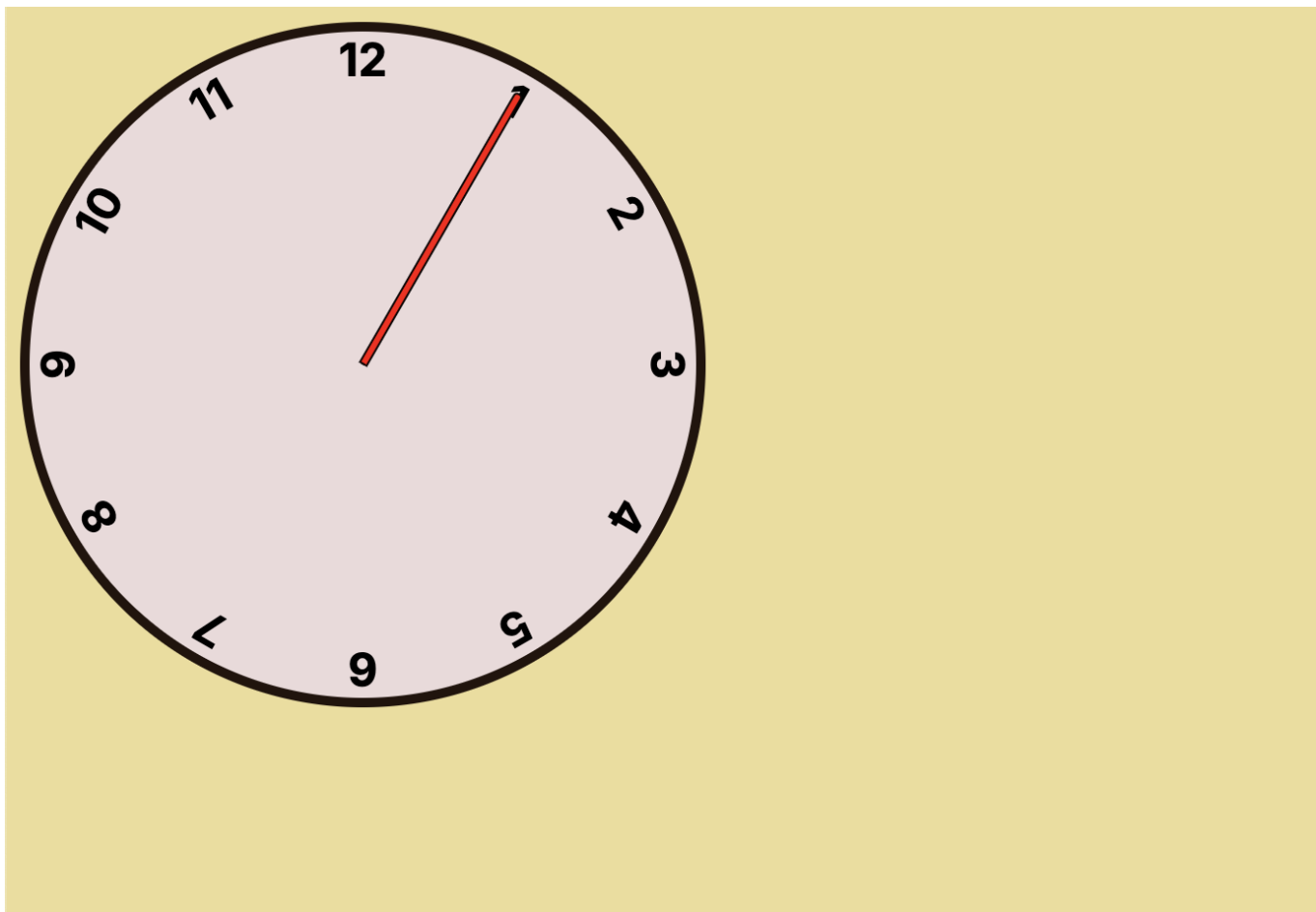
DEBUG CONSOLE

TERMINAL

PORTS

Filter (e.g. text, !exclu...

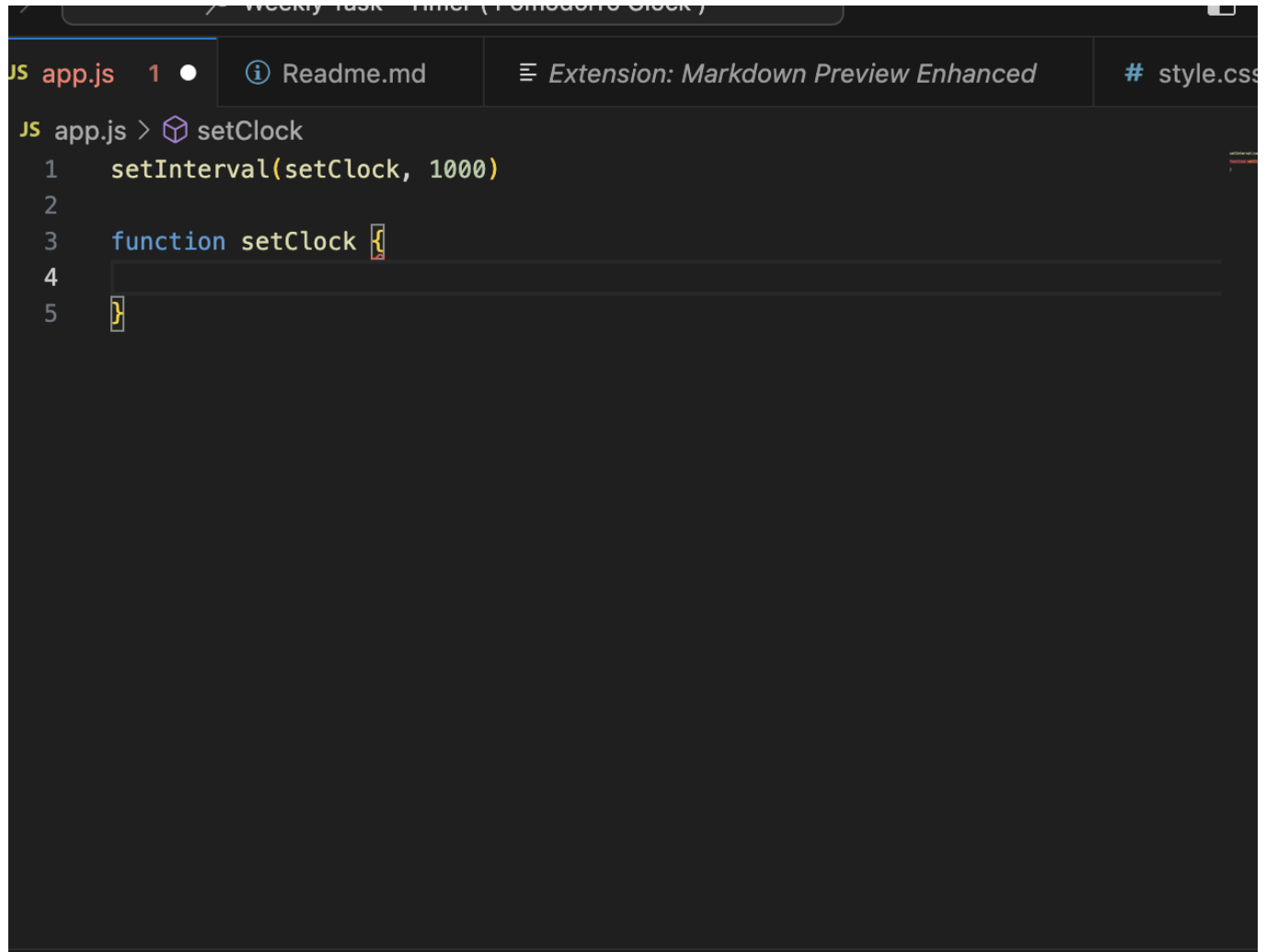




Here I also played with the sizes a little, making them bigger which would help people with impaired vision to see the clock better.

I have centered it as well.

Now I'll build javascript and use new Date() method to get accurate time

A screenshot of a code editor with a dark theme. The editor has several tabs at the top: 'app.js' (active), 'Readme.md', 'Extension: Markdown Preview Enhanced', and 'style.css'. The 'app.js' tab shows the following code:

```
JS app.js > setClock
1  setInterval(setClock, 1000)
2
3  function setClock {
4
5  }
```

Because the code needs to call the function every second to work properly, I start from setting interval in milliseconds to one second (one second = 1000 milliseconds)

Then I define the function setClock, which will use new Date() method. The method helps to get an exact current time internally in the JS.

In the example which I follow here, we will use this method to get seconds for the seconds ratio, and then multiply it by 60 to get minute and by 60 again to get hours, which is incorrect but I'll fix it in a bit.

```
Weekly Task - Timer (Pomodoro Clock)

JS app.js 1 ● ⓘ Readme.md ≡ Extension: Markdown Preview Enhanced # style.css

JS app.js > 📦 setClock > [🔗] currentDate
1  setInterval(setClock, 1000)
2
3  function setClock {
4    ⚡ const currentDate = new Date
5  }
```

[🔗] Date

var Date: DateConstructornew () => Date

[🔗] DataView

[🔗] DataTransfer

[🔗] DataTransferItem

[🔗] DataTransferItemList

[🔗] VarDate

[🔗] FormDataEvent

abc currentDate

📦 dispatchEvent

[🔗] DOMMatrixReadOnly

[🔗] DeviceOrientationEvent

[🔗] DragEvent

```
Weekly Task - Timer (Pomodoro Clock)

JS app.js 1 ● ⓘ Readme.md ≡ Extension: Markdown Preview Enhanced # style.css

JS app.js > 📦 setClock > [🔗] secondsRatio
1  setInterval(setClock, 1000)
2
3  function setClock {
4    const currentDate = new Date()
5    ⚡ const secondsRatio = currentDate.getSeconds
6  }
```

📦 getSeconds

(method) Date.getSeconds():

📦 getUTCSeconds

📦 getMilliseconds

📦 getUTCMilliseconds

```

JS app.js 1 ● ⓘ Readme.md ≡ Extension: Markdown Preview Enhanced # style.css
JS app.js > 📦 setClock > 🔗 secondsRatio
1  setInterval(setClock, 1000)
2
3  function setClock {
4      const currentDate = new Date()
5      💡 const secondsRatio = [(secondsRatio + currentDate.getSeconds())] / 60
6      const minutesRatio = currentDate.getMinutes() / 60
7      const hoursRatio = currentDate.getHours() / 60
8  }

```

So the problem arises here, that it can work with new Date() methods, but it means that there won't be a smooth transition like you see in the real clock, where the hour hand slowly moves from one number to another. To fix it, we can use seconds ratio and use it as a reference for other hands.

So the solution is to add the seconds ratio to the current date minutes and hours date, respectively, to increase the smoothness. (as the seconds hand changes every second, it will slowly affect minutes and hours hands like in the real world they would)

Also the hours are divided by 12, which fixes the hours issue.


```

app.js > ...
1  setInterval(setClock, 1000)
2
3  const hourHand = document.querySelector('data--hour--hand')
4  const minuteHand = document.querySelector('data--minute--hand')
5  const secondHand = document.querySelector('data--second--hand')
6
7  function setClock {
8      const currentDate = new Date()
9      const secondsRatio = currentDate.getSeconds() / 60
10     const minutesRatio = (secondsRatio + currentDate.getMinutes()) / 60
11     const hoursRatio = (minutesRatio + currentDate.getHours()) / 12
12 }

```

```

index.html > html > body > div.clock > div#data--second--hand.hand.second
4  <meta charset="UTF-8">
5  <meta name="viewport" content="width=device-width, initial-scale=1.0">
6  <title>Live Clock</title>
7  <link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;5
8  <link rel="stylesheet" href="style.css">
9  <script src="app.js" defer></script> <!-- I learned to use defer command
10 </head>
11 <body>
12     <div class="clock">
13         <div class="hand hour" id="data--hour--hand"></div>
14         <div class="hand minute" id="data--minute--hand"></div>
15         <div class="hand second" id="data--second--hand"></div>
16         <div class="number number1">1</div>
17         <div class="number number2">2</div>
18         <div class="number number2">2</div>



```


There was a little typo along the way that I found & fixed

```

</head>
<body>
    <div class="clock">
        <div class="hand hour" data-hour-hand></div>
        <div class="hand minute" data-minute-hand></div>
        <div class="hand second" id-data-second-hand></div>
        <div class="number number1">1</div>
        <div class="number number2">2</div>
        <div class="number number3">3</div>
        <div class="number number4">4</div>
        <div class="number number5">5</div>
        <div class="number number6">6</div>
        <div class="number number7">7</div>
        <div class="number number8">8</div>

```

```
JS app.js >  setClock
1  setInterval(setClock, 1000)
2
3  const hourHand = document.querySelector('data--hour--hand')
4  const minuteHand = document.querySelector('data--minute--hand')
5  const secondHand = document.querySelector('data--second--hand')
6
7  function setClock {
8      const currentDate = new Date()
9      const secondsRatio = currentDate.getSeconds() / 60
10     const minutesRatio = (secondsRatio + currentDate.getMinutes()) / 60
11     const hoursRatio = (minutesRatio + currentDate.getHours()) / 12
12     setRotation(secondHand, secondsRatio)
13     setRotation(minuteHand, minutesRatio)
14      setRotation(hourHand, hoursRatio)
15 }
16
17 function setRotation (element, rotationRatio) {
18     element.style.setProperty('--rotation', rotationRatio * 360)
19 }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter (e.g. text, !exclu... 

Uncaught SyntaxError SyntaxError: Unexpected token '{'
at (program) (/Users/andriartemenko/Desktop/Weekly Tasks WebDev Studio Submission/Weekly Task - Timer (Pomodorro Clock)/app.js:7:19)
No debugger available, can not send 'variables'

I also added data- attributes to seconds, minute and hour hand , to access them through the query selector, by data attribute.

Then I defined a function to set the rotation of the elements that I've defined and linked through DOM query selector.

The function setRotation takes 2 arguments - the element and the rotations ratio. As the functions runs, every second it rotates according to the set ratio that was pre-defined before.