CODE CHUNKS

ANALOG CLOCK

HTML | CSS | JAVASCRIPT

Table of content

Chapter #1

Introduction

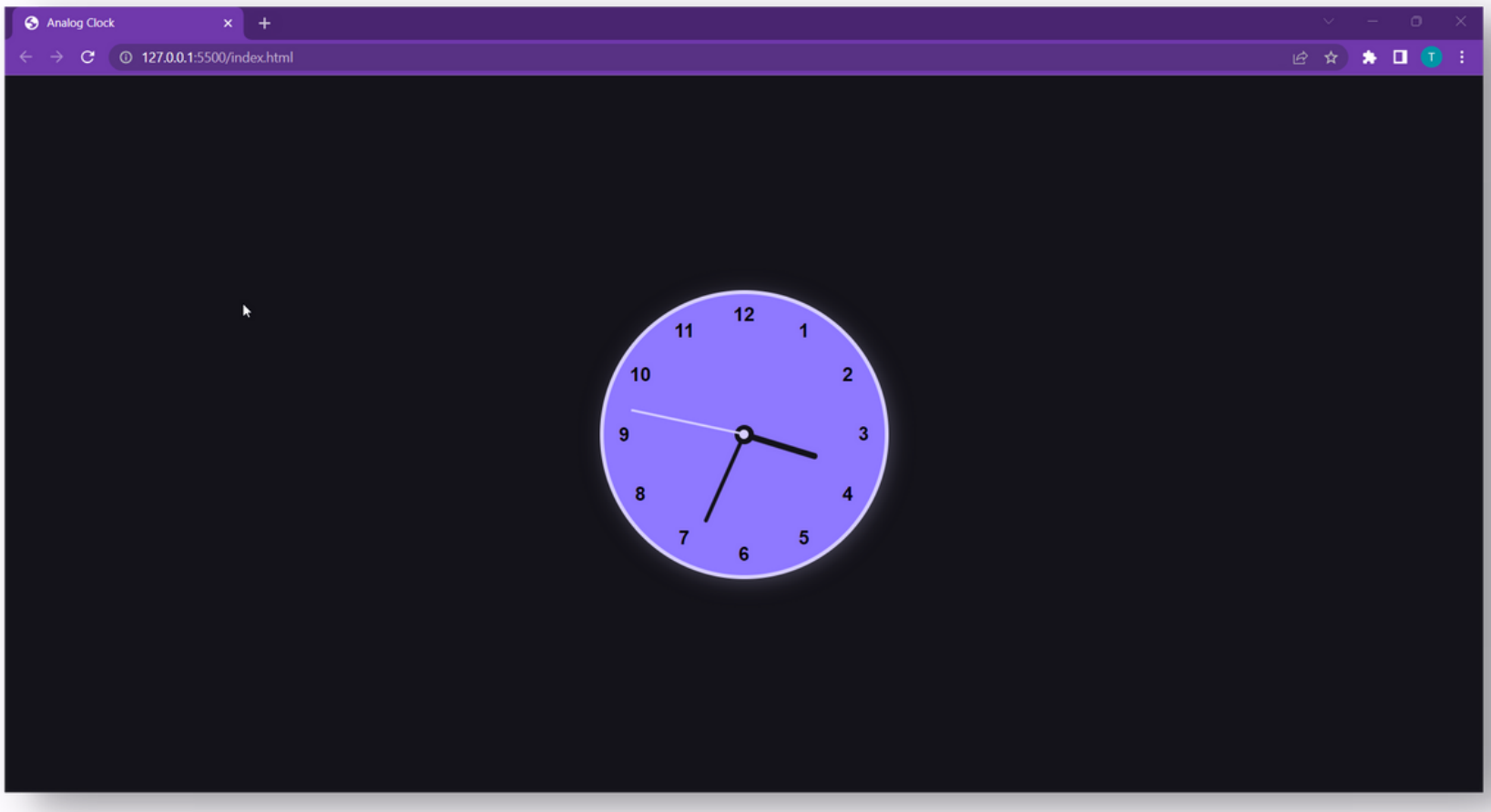
Analog Clock-html, css and javascript

In this blog/e-book, we will explore how to create an **ANALOG CLOCK** using the powerful trio of web technologies: **HTML, CSS, and JavaScript.**

We will **break down the code into smaller chunks**, explain each part in detail, and provide code snippets to guide you through the process step by step.

By the end of this guide, you will have a **clear understanding** of how to **build your own analog clock and customize it** to suit your needs.

FINAL OUTPUT:-



Chapter #2

Html Structure

Create an structure of analog clock

> The **HTML structure lays the foundation** for our analog clock. We will **examine** the **different components** and **their purposes.**

Clock Container

- The outermost '** **' element with the class '**clock**' acts as a container for the entire clock.

```
<div class="clock">
  ...
</div>
```

Clock Numbers

- These lines of code represent the **clock's numbers or hour markers.**
- Each number is enclosed within a '** **' element with the class '**numbers**'.
- The '**style**' attribute is used to set the custom CSS property '**--i**' to a specific value.
- This property will be used later in the **CSS to position the numbers correctly.**
- The '****' element inside each '** **' displays the corresponding number.

```
<div class="numbers" style="--i: 1"><span>1</span></div>
<div class="numbers" style="--i: 2"><span>2</span></div>
<div class="numbers" style="--i: 3"><span>3</span></div>
<div class="numbers" style="--i: 4"><span>4</span></div>
<div class="numbers" style="--i: 5"><span>5</span></div>
<div class="numbers" style="--i: 6"><span>6</span></div>
<div class="numbers" style="--i: 7"><span>7</span></div>
<div class="numbers" style="--i: 8"><span>8</span></div>
<div class="numbers" style="--i: 9"><span>9</span></div>
<div class="numbers" style="--i: 10"><span>10</span></div>
<div class="numbers" style="--i: 11"><span>11</span></div>
<div class="numbers" style="--i: 12"><span>12</span></div>
```

Clock Circles

- The '** **' element with the class '**circle**' represents the **circular shape present at the center of the clock.** It acts as a **container** for the **clock's hands.**

```
<div class="circle">
  ...
</div>
```

Clock Hands

- These lines of code define the clock's **hour, minute, and second hands.**
- Each hand is represented by a '** **' element with a specific '**class**' **('hour', 'minute', & 'second')** and an '**id**' attribute **('hour-hand', 'minute-hand', & 'second-hand').**
- These IDs will be **used in JavaScript to manipulate the clock hands' rotation** based on the current time.

```html
<div class="hour" id="hour-hand"></div>
<div class="minute" id="minute-hand"></div>
<div class="second" id="second-hand"></div>
```

💡 Overall, this HTML code structures the analog clock by using nested '** **' elements and appropriate '**classes**' for styling and positioning the clock's components.

Chapter #3

CSS Styling

Styling our Html Structure using CSS

In this chapter, we will explore the **CSS code** that **brings life to our analog clock.**

We will discuss the **styles applied** to **various elements, including the clock container, numbers, circle, and clock hands.**

Global Styles

- These styles apply to all elements on the page.
- The '*' selector **selects all elements,** and the '**margin: 0; padding: 0;**' rule **removes any default margin and padding.**
- The '**box-sizing: border-box;**' rule ensures that the element's **width and height include the padding and border.**

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

Body Styles

- These styles are applied to the '**<body>**' element.
- They set the **body's height and width to occupy the full viewport ('100vh' and '100vw').**
- The '**color**' property sets the text color to '**#14131a**', and the '**background-color**' property sets the background color to '**#14131a**'.
- The '**font-size**', '**font-weight**', and '**font-family**' properties define the **font styles for the text.**

- The '**display: flex;**', '**align-items: center;**', '**justify-content: center;**' properties **center the content horizontally and vertically within the body.**

```
body {
  height: 100vh;
  width: 100vw;
  font-size: 20px;
  font-weight: bolder;
  font-family: Arial, Helvetica, sans-serif;
  background-color: #14131a;
  display: flex;
  align-items: center;
  justify-content: center;
}
```

Clock Styles

- These styles are applied to the clock container, which has the class '**clock**'.
- The **width and height** properties set the **dimensions of the clock to '300px'.**
- The '**border-radius: 50%;**' creates a **circular shape by rounding the corners.**
- The '**display: flex;**', '**align-items: center;**', '**justify-content: center;**' properties **center** the **clock's contents horizontally and vertically.**
- The '**background-color**' property sets the background color to '**#8f79ff**'.
- The '**border property**' adds a **4px solid border with color #d9d1ff**, and the '**box-shadow**' property adds a **shadow effect** to the clock.
- The '**position: relative;**' property is used to **position the clock's child elements.**

```
.clock {
  width: 300px;
  height: 300px;
  border-radius: 50%;
  background-color: #8f79ff;
  border: 4px solid #d9d1ff;
  box-shadow: 3px 3px 20px #d9d1ff42;
  display: flex;
  align-items: center;
  justify-content: center;
  position: relative;
}
```

Clock Numbers Styles

- These styles are applied to the **clock's numbers or hour markers**, which are **represented by the elements with the class numbers inside the clock container.**
- The '**.clock .numbers**' selector selects these elements.
- The '**text-align: center;**' property **horizontally centers** the content of each number.
- The '**inset: 10px;**' property provides an **inset spacing of 10 pixels** from the clock's edge.
- The '**position: absolute;**' property **positions the numbers relative to the clock.**
- The '**transform: rotate(calc(var(--i) * 30deg));**' property **rotates each number by a specific angle** calculated based on the custom CSS property '**--i**'.
- The second block '**.clock .numbers span**' selects the ** element inside each number** and **applies similar rotation but in the opposite direction** to ensure the numbers are properly aligned.

```
.clock .numbers {
  position: absolute;
  text-align: center;
  inset: 10px;
  transform: rotate(calc(var(--i) * 30deg));
}

.clock .numbers span {
  display: inline-block;
  transform: rotate(calc(var(--i) * (-30deg)));
}
```

Clock Circle Styles

- These styles are applied to the **clock's circle**, which is present at the center of the circle.
- The '**.circle**' selector selects this element.
- The '**position: absolute;**' property **positions the circle relative to the clock.**
- The **width and height** properties **set the dimensions of the circle to 20px.**
- The '**border-radius: 50%;**' creates a **circular shape** by rounding the corners.
- The '**display: flex;**', '**align-items: center;**', '**justify-content: center;**' properties **center the circle's content horizontally and vertically.**
- The '**background-color**' property sets the **background color to '#14131a'.**

- The '**.circle::after**' selector **selects** the pseudo-element **::after** of the circle element.
- The '**content: " ";**' property is used to **add content to the pseudo-element.**
- The '**position: absolute;**' property **positions the pseudo-element.**
- The **width and height** properties set the **dimensions of the pseudo-element to 10px.**
- The '**border-radius: 50%;**' creates a **circular shape** for the pseudo-element.
- The **background-color** property sets the background color to **#d9d1ff.**

```
.circle {
  position: absolute;
  width: 20px;
  height: 20px;
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  background-color: #14131a;
}

.circle::after {
  content: " ";
  position: absolute;
  width: 10px;
  height: 10px;
  border-radius: 50%;
  background-color: #d9d1ff;
}
```

Clock Hands Styles

- These styles are applied to the **clock's hour, minute, and second hands**, which are represented by the elements with the classes **'hour, minute, and second'**, respectively.
- The '**.hour, .minute, .second**' selector selects these elements.
- The '**position: absolute;**' property **positions the hands relative to the clock.**
- The '**bottom: 50%;**' property **positions the hands vertically at the bottom half of the clock.**
- The '**border-radius: 3px;**' property gives a **slight rounded appearance to the hands.**
- The '**transform-origin: bottom;**' property sets **the transformation origin for the hands to the bottom.**

- The **.hour** selector sets the **width to 6px, height to 80px,** and **background color to #14131a** for the hour hand.

- The **.minute** selector sets the **width to 4px, height to 100px,** and **background color to #14131a** for the minute hand.

- The **.second** selector sets the **width to 2px, height to 120px,** and **background color to #d9d1ff** for the second hand.

```
.hour,
.minute,
.second {
  position: absolute;
  bottom: 50%;
  border-radius: 3px;
  transform-origin: bottom;
}

.hour {
  width: 6px;
  height: 80px;
  background-color: #14131a;
}

.minute {
  width: 4px;
  height: 100px;
  background-color: #14131a;
}

.second {
  width: 2px;
  height: 120px;
  background-color: #d9d1ff;
}
```

💡 These **CSS styles** are responsible for **positioning, sizing, and styling** various components of the analog clock.

Chapter #4

JavaScript Functionality

Provide logic to the clock to make it functional

The **JavaScript code** is responsible for **making our analog clock functional.**

We will **explore** how it **retrieves** the **current time**, **calculates** the **rotation angles for the clock hands**, and **updates their positions** accordingly.

Element Selection

- These lines of code **select the hour, minute, and second hands** of the clock by their respective **'id'** attributes.
- The **document.getElementById()** method retrieves the **DOM element** with the specified **'id'** value and assigns it to the corresponding variables (**'hourHand'**, **'minuteHand'**, **'secondHand'**).

```
let hourHand = document.getElementById("hour-hand");
let minuteHand = document.getElementById("minute-hand");
let secondHand = document.getElementById("second-hand");
```

Setting the Time

- The **setTime** function is defined as an **arrow function.**
- It **retrieves** the **current time** using the **Date object** and **assigns the hour, minute, and second values** to their respective **variables (hour, minute, second).**

```
let setTime = (()=> {
    let date = new Date();
    let hour = date.getHours();
    let minute = date.getMinutes();
    let second = date.getSeconds();
```

Calculating Rotation

- These lines of code **calculate the rotation angles for the hour, minute, and second hands** based on the **current time.**
- The **rotation angles** are **calculated in degrees.**
- The **hourRotation** is determined by **multiplying the hour value by 30** (since each hour represents 30 degrees on the clock) and **adding an additional rotation based on the minute value divided by 2** (since the hour hand also moves slightly based on the current minute).
- The **minuteRotation** is simply obtained by **multiplying the minute value by 6** (since each minute represents 6 degrees on the clock).
- The **secondRotation** is obtained by **multiplying the second value by 6** (since each second represents 6 degrees on the clock).

```
let hourRotation = 30*hour + minute/2;
let minuteRotation = 6*minute;
let secondRotation = 6*second;
```

Applying Transformations

- These lines of code **set the transform CSS property** of each **clock hand to rotate** them **based on the calculated rotation angles.**
- The **rotate()** function is used to **specify the rotation angle in degrees.**

```
hourHand.style.transform = `rotate(${hourRotation}deg)`;
minuteHand.style.transform = `rotate(${minuteRotation}deg)`;
secondHand.style.transform = `rotate(${secondRotation}deg)`;
```

Updating Time

- The **setInterval()** function is used to **call the setTime function repeatedly every 1000 milliseconds (1 second).**
- This ensures that the **clock's hands** are **updated every second to reflect the current time.**

```
setInterval(setTime, 1000);
```

💡 Overall, this **JavaScript code** retrieves the **current time**, **calculates the rotation angles for each clock hand,** and **updates the hands' positions on the clock every second.**

Chapter #5

Bringing It All Together

Now that we have covered the **HTML structure**, **CSS styling**, and **JavaScript functionality**, it's **time to put all the pieces together.**

We will create a **cohesive code snippet** that **integrates these components and displays a fully functional analog clock.**

Chapter #6

Fine-tuning and customization options

Congratulations! You have successfully **built your own analog clock** using **HTML, CSS, and JavaScript.**

But we're not done yet - there's **room for customization and further enhancements.**

1. Changing colors:

- **Experiment with** different **colors** to customize the clock's appearance.
- **Modify** the **background color**, **clock face color**, and **hand colors** to match your preferred style.

2. Adding additional features:

- **Extend** the **functionality** of your analog clock **by incorporating extra features.**
- **For example**, you could **add a ticking sound effect** or **a digital time display.**

3. Implementing different clock styles:

- **Explore various clock styles** to give your analog clock a unique look.
- Consider **using Roman numerals** or **different types of fonts** for the numbers, or even **creating a retro-style clock.**

4. Animating the clock:

- **Add smooth animation effects** to make your clock more visually appealing.
- Consider **animating the transition between time updates** or **adding subtle movements** to the **clock hands.**

Chapter #7

Conclusion

In this tutorial, we've **learned** how to create an **interactive analog clock using HTML, CSS, and JavaScript.**

We **started by** setting up the **HTML structure**, **applied CSS styles** for the clock's appearance, and **implemented JavaScript** code to make the clock hands update in real-time.

We also **explored fine-tuning** and **customization options** to **further enhance the clock.**

Now it's **your turn to get creative! Customize** your analog clock, **experiment** with different styles and features, and **let your imagination guide you.**

Building an analog clock is **just the beginning** of your **journey** into web development.

I hope you **enjoyed this tutorial** and **found it helpful** in learning how to create an interactive **analog clock using HTML, CSS, and JavaScript.**

Happy coding and **have fun** exploring the endless possibilities of **web development!**

YOU WILL GET THE COMPLETE SOURCE CODE ON THESE PLATFORMS

GITHUB: https://github.com/code-chunks-platform

TELEGRAM: https://t.me/code_chunks

CODPEN: https://codepen.io/code_chunks

WATCH COMPLETE VIDEO WITH EXPLATION ON OUR YOUTUBE CHANNEL

Click here to watch complete video

Also connect with us on these platforms

(Click on user-name to go to that platform)

- code_chunks
- code-chunks-platform
- Code Chunks
- Code Chunks
- Code Chunks
- Code Chunks