DAY 13

Index.html

<body> ... </body>

This is the **visible part of the web page**. It contains everything the user sees and interacts with.

<button id="darkModeToggle">?</button>

This is a **toggle button** in the top-right corner.

- It switches between **light mode and dark mode**.
- The ? **(moon)** icon changes to ☀️ when dark mode is active.

<div class="container"> ... </div>

This holds the **clock and button** together in a neat layout.

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

This is the **clock face**. It contains:

- ticks: hour markings (if you add them).
- hour, minute, second: these are the **moving clock hands**.
- center: the small **circle at the center** of the clock.

<button onclick="setTime()">Set Your Time</button>

A second button that allows the user to **set a custom time** manually (if coded in JavaScript).

Style.css

body { ... }

This styles the **entire page**:

- display: flex: Use flex layout to center everything.
- flex-direction: column: Items are stacked **top to bottom**.
- align-items: center + justify-content: center: Center the content **horizontally and vertically**.
- height: 100vh: Take the **full height** of the screen.
- margin: 0: Remove the default browser margin.
- background: Set a **dark gradient** background with three shades: dark gray to blue-gray.
- font-family: sans-serif: Use clean and modern font.
- transition: Smooth background changes (for dark mode toggle).

.container { text-align: center; }

This keeps everything inside the container **centered**.

.clock { ... }

Styles the **round clock**:

- width/height: 300px: Set clock size.
- background: white: Clock face color.
- border: Light gray border.
- border-radius: 50%: Make it **circular**.
- position: relative: Needed to position the hands and ticks **inside** it.
- box-shadow: Add a **soft shadow**.

.ticks and .tick { ... }

- .ticks: container for the lines around the clock (tick marks).
- .tick: each small black line (like for minutes).
  - Positioned at the top and centered.
  - Rotated around the clock.
- .tick.big: slightly longer lines for hour markers.

.hand, .hour, .minute, .second

These are the **clock hands**:

- .hand: base style for all hands (hour, minute, second).
  - transform-origin: bottom center: So it rotates from the center.
  - z-index: 2: So it stays above ticks.
- .hour: short and thick (black).
- .minute: longer and thinner (black).
- .second: longest and thinnest (red).

.center

The **small circle in the middle** of the clock to cover the hand's base.

.number

Used for **optional number positions** (like 12, 3, 6, 9) on the clock.

button { ... }

Styles the **buttons** like:

- "Set Your Time" or
- Dark mode toggle:

- Rounded, gray button
- White text
- Cursor changes on hover
- Hover effect: becomes darker

#darkModeToggle { ... }

Styles the **moon icon** button:

- Stays fixed at the **top-right corner**
- Transparent background
- No borders
- High z-index to stay on top

Dark Mode Styles: **body.dark-mode { ... }**

These styles apply **only when dark mode is active**:

- body.dark-mode: changes body background to dark gray and text to light.
- .clock: gets a dark gradient like the image you uploaded.
- .hand: hands turn light gray (except second hand stays red).
- .number: clock numbers become light.
- button: switches to white background with dark text.

Script.js

DARK MODE TOGGLE

```
const darkToggle = document.getElementById('darkModeToggle');
darkToggle.textContent = _; // Set moon emoji initially
```

This grabs the **dark mode button** and sets its icon when the page loads.

```
darkToggle.addEventListener('click', () => {
  document.body.classList.toggle('dark-mode');
  darkToggle.textContent = document.body.classList.contains('dark-mode') ?_:_;
});
```

- When you **click the button**, it adds/removes the "dark-mode" class to the <body>.
- It also switches the emoji:
  - = light mode (dark mode off)
  - ☀️ = dark mode is ON

CLOCK LOGIC

```
const hourHand = document.getElementById('hour');
const minuteHand = document.getElementById('minute');
const secondHand = document.getElementById('second');
```

These lines **grab the hands** of the clock.

```
function updateClock() {
  const now = new Date(); // Get current time
  const seconds = now.getSeconds();
  const minutes = now.getMinutes();
  const hours = now.getHours();

  // Calculate how much each hand should rotate
  const secondDeg = (seconds / 60) * 360;
  const minuteDeg = (minutes / 60) * 360 + (seconds / 60) * 6;
  const hourDeg = ((hours % 12) / 12) * 360 + (minutes / 60) * 30;

  // Apply the rotation to the hands
  secondHand.style.transform = `rotate(${secondDeg}deg)`;
  minuteHand.style.transform = `rotate(${minuteDeg}deg)`;
  hourHand.style.transform = `rotate(${hourDeg}deg)`;
}
```

Run this every second:

```
setInterval(updateClock, 1000);
updateClock(); // Also run once at the beginning
```

SET CUSTOM TIME BUTTON

```
function setTime() {
  let h = parseInt(prompt("Enter hour (0–23):", "3"));
  let m = parseInt(prompt("Enter minute (0–59):", "0"));
  let s = parseInt(prompt("Enter second (0–59):", "0"));
```

When you click **“Set Your Time”**, it asks you to type hour, minute, and second.

```
  if (isNaN(h) || h < 0 || h > 23 || isNaN(m) || m < 0 || m > 59 || isNaN(s) || s < 0 || s > 59) {
    alert("Invalid time input!");
    return;
  }
```

If the input is wrong, it shows an alert.

```
  const base = new Date();
  base.setHours(h, m, s, 0);
  const diff = base - new Date();
  customStart = Date.now() + diff;
}
```

- It calculates the **difference** between the current time and your custom time.
- `customStart` is used to **pretend the clock is now your custom time**.

```
function getCustomDate() {
  if (customStart)
    return new Date(customStart + Date.now() - customStart);
  return new Date();
}
```

This function returns the **real time** or your **custom-set time**.

ROMAN NUMBERS AROUND THE CLOCK

```
const romanNumerals = ['XII', 'I', 'II', 'III', 'IV', 'V', 'VI', 'VII', 'VIII', 'IX', 'X', 'XI'];
```

List of **Roman numerals** for 12 hours on the clock.

```
for (let i = 0; i < 12; i++) {
  const number = document.createElement('div');
  number.className = 'number';
```

- Create 12 `<div>`s (one for each hour)
- Add the `number` class for styling

```
  const angle = (i * 30 - 90) * (Math.PI / 180); // Start from top
  const radius = 120;
  const x = radius * Math.cos(angle);
  const y = radius * Math.sin(angle);

  number.style.left = `calc(50% + ${x}px)`;
  number.style.top = `calc(50% + ${y}px)`;
  number.textContent = romanNumerals[i];
  clock.appendChild(number);
}
```

- Use **trigonometry** to place each number in a **circle**.
- `angle`: determines where to place it (like clock hours).
- `x`, `y`: position for each number using radius and angle.
- Add the number to the clock.