DAY 5

Note: The names for my HTML, CSS, and JavaScript files (like DAYTHREE.html, DAYTHREE.css, DAYTHREE.js) are **not recommended** as they are considered **bad practice** in professional projects. However, since this is a **mini project**, I prefer using these names to **distinguish them from my other files**. For **larger or professional projects**, it's better to use standard and descriptive file names like index.html, style.css, and script.js.

DAYFIVE.HTML

❖ **<body>**

```
<div class="header">
  <h1>DAY 5</h1>
</div>
```

- The page content starts here.
- This part shows **"DAY 5"** at the top as a header.

❖ **<div class="container">**

- This wraps the main content (timer and buttons).

❖ **<div class="circle-container">**

```
  <svg class="progress-ring" width="320" height="320">
    <circle class="ring-bg" r="130" cx="160" cy="160" />
    <circle class="ring" r="130" cx="160" cy="160" />
  </svg>
  <div class="time" id="timer">25:00</div>
</div>
```

- This shows the **Pomodoro timer** inside a circular SVG.
- The circle:
  - Has a radius of `130` (so it's big).
  - Is centered (`cx=160`, `cy=160`).
  - One circle is the **background**, the other is the **progress ring**.
- Below the SVG, it displays the time **"25:00"** in the center.

❖ **<div class="button-wrapper">**

```
  <button id="start">Start</button>
  <button id="stop">Stop</button>
  <button id="reset">Reset</button>
  <button id="about" onclick="window.open('DAY 5.pdf', '_blank')">About</button>
</div>
```

- This section contains 4 buttons:
  - **Start**: Begins the countdown.
  - **Stop**: Pauses the countdown.

- **Reset**: Sets it back to 25:00.
- **About**: Opens a PDF file called **"DAY 5.pdf"** in a new tab.

DAYFIVE.CSS

1. body Styling

```
body {
  margin: 0;
  font-family: "Segoe UI", sans-serif;
  color: white;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background:
    linear-gradient(-45deg, rgba(15, 12, 41, 0.4), rgba(48, 43, 99, 0.4), rgba(36, 36, 62, 0.4), rgba(0, 0, 0, 0.4)),
    url('your-image-url');
  background-size: cover;
  background-position: center;
  background-repeat: no-repeat;
  animation: gradientBG 15s ease infinite;
}
```

- **Centers** everything vertically and horizontally.
- Adds a **dark transparent gradient** over a **background image**.
- Applies a **smooth animated background effect** using @keyframes.

2. Background Animation

```
@keyframes gradientBG {
  0% { background-position: 0% 50%; }
  50% { background-position: 100% 50%; }
  100% { background-position: 0% 50%; }
}
```

- This moves the gradient background **left to right and back** continuously.

3. Header Styling

```
.header {
  position: absolute;
  top: 20px;
  width: 100%;
  text-align: center;
  z-index: 2;
}
.header h1 {
  color: #fff;
  text-decoration: underline solid lightgray;
  margin: 0;
  padding: 10px 0;
}
```

- Shows "DAY 5" at the **top center** with an **underlined white heading**.

4. Container and Circle Layout

```
.container {
  text-align: center;
}

.circle-container {
  position: relative;
  width: 320px;
  height: 320px;
  margin: 0 auto 20px;
}
```

- Keeps all elements centered.
- Sets a **fixed size (320x320px)** for the **circle timer**.

5. SVG Progress Ring

```
.progress-ring {
  transform: rotate(-90deg);
}
.ring-bg {
  fill: none;
  stroke: rgba(41, 20, 225, 0.874); /* Blue outer ring */
  stroke-width: 8;
  stroke-dasharray: 565.48;
}
.ring {
  fill: none;
  stroke: white; /* Moving inner ring */
  stroke-width: 6;
  stroke-dasharray: 565.48;
  stroke-dashoffset: 0;
  transition: stroke-dashoffset 1s linear;
}
```

- `ring-bg`: Blue background circle.
- `ring`: White circle that **animates the progress**.
- `rotate(-90deg)`: Starts animation from **top**, not right.

6. Time Text (Inside Circle)

```
.time {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  font-size: 32px;
  font-weight: bold;
  text-shadow: 0 0 10px rgba(255, 255, 255, 0.3);
}
```

- Shows **25:00** in the center of the circle.
- Uses **white glowing effect** for better visibility.

7. Button Styling

```
.button-wrapper {
  margin-top: 20px;
}

button {
  font-size: 16px;
  padding: 10px 20px;
  margin: 5px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  color: white;
  text-transform: uppercase;
}
```

- Adds consistent **padding**, **rounded corners**, and spacing to all buttons.
- `text-transform: uppercase` = makes button text **capital letters**.

8. Individual Button Colors

```
#start {
  background-color: #07ae3c; /* Green */
}
#stop {
  background-color: #d5220e; /* Red */
}
#reset {
  background-color: orange;
}
#about {
  background-color: rgb(13, 31, 228); /* Blue */
}
```

- Easy color identification:
  ⍰ Start = green
  ⍰ Stop = red
  ⍰ Reset = orange
  ⍰ About = blue

9. Button Hover Effect

```
button:hover {
  opacity: 0.8;
}
```

- Buttons get a little **transparent when hovered** = nice visual feedback.

DAYFIVE.JS

Get HTML Elements

```
const startBtn = document.getElementById("start");
const stopBtn = document.getElementById("stop");
const resetBtn = document.getElementById("reset");
const timerEl = document.getElementById("timer");
const circle = document.querySelector(".ring");
```

This part **connects your JavaScript to your HTML buttons and timer display**:

- `startBtn` = the **Start** button
- `stopBtn` = the **Stop** button
- `resetBtn` = the **Reset** button
- `timerEl` = the text that shows **time left** (like 25:00)
- `circle` = the **white ring** in the SVG circle

Variables for Time and Interval

```
let timeLeft = 1500; // 25 mins in seconds
let totalTime = 1500;
let interval;
```

- `timeLeft` = how much time is **left** (1500 seconds = 25 minutes)
- `totalTime` = the **total full time** (used to calculate the progress)
- `interval` = to **store the timer**, so you can stop or reset it later

Show the Time on Screen

```
function updateTimerDisplay() {
  let minutes = Math.floor(timeLeft / 60);
  let seconds = timeLeft % 60;
  timerEl.innerHTML = `${minutes.toString().padStart(2, "0")}:${seconds.toString().padStart(2, "0")}`;
}
```

This function:

- Converts seconds into **minutes and seconds**
- Adds leading `0` if needed (so `05:07` not `5:7`)
- Updates the screen to show the time

Animate the Circle Progress

```
function updateCircle() {
  const radius = 130;
  const circumference = 2 * Math.PI * radius;
  const offset = circumference - (timeLeft / totalTime) * circumference;
  circle.style.strokeDasharray = `${circumference}`;
  circle.style.strokeDashoffset = offset;
}
```

This is the **cool animation** part:

- Calculates how **big the circle is** (using math formula: `2πr`)
- Then figures out how much of the circle should be visible
- As time goes down, the circle fills **in reverse** (like a progress bar)

Start the Timer

```
function startTimer() {
  if (interval) return; // avoid double starts

  interval = setInterval(() => {
    timeLeft--;
    updateTimerDisplay();
    updateCircle();

    if (timeLeft <= 0) {
      clearInterval(interval);
      alert("Time's up!");
      timeLeft = totalTime;
      updateTimerDisplay();
      updateCircle();
    }
  }, 1000);
}
```

This starts the countdown:

- Every 1 second (`1000ms`) it reduces `timeLeft` by 1
- Updates the time and circle
- When time is up, it **stops the timer** and shows an alert
- Then resets the timer to 25:00

Stop the Timer

```
function stopTimer() {
  clearInterval(interval);
  interval = null;
}
```

- This **pauses the timer** by clearing the interval

Reset Timer

```
function resetTimer() {
  stopTimer(); // first stop
  timeLeft = totalTime; // reset time
  updateTimerDisplay(); // update screen
  updateCircle(); // reset circle
}
```

- Stops the timer
- Resets everything back to 25:00 and full circle

Set Everything When Page Loads

```
updateTimerDisplay();
updateCircle();

startBtn.addEventListener("click", startTimer);
stopBtn.addEventListener("click", stopTimer);
resetBtn.addEventListener("click", resetTimer);
```

- When the page opens:
  - The time (25:00) and circle are shown
- Also, when you **click the buttons**, the right function is triggered