Để phát triển một ứng dụng Pomodoro Timer, chúng ta sẽ xây dựng từ các yếu tố cơ bản trong HTML, CSS, và JavaScript. Dự án này sẽ bao gồm các chức năng đếm ngược, chu kỳ làm việc - nghỉ ngơi, và có thể thêm âm thanh báo hiệu hoặc thay đổi giao diện để tạo trải nghiệm người dùng tốt hơn.

**1. Mô tả chức năng của Pomodoro Timer**

Pomodoro là một phương pháp quản lý thời gian, chia thời gian làm việc thành các khoảng 25 phút (Pomodoro) và nghỉ ngắn 5 phút. Sau 4 chu kỳ Pomodoro, người dùng sẽ có một khoảng nghỉ dài (15-30 phút).

Ứng dụng sẽ có các chức năng chính sau:

* **Thiết lập thời gian làm việc và nghỉ ngơi:** Người dùng có thể tùy chỉnh thời gian mỗi Pomodoro và thời gian nghỉ.
* **Đếm ngược thời gian**: Ứng dụng sẽ đếm ngược từ thời gian đã cài đặt và báo khi hết thời gian.
* **Chu kỳ lặp**: Sau mỗi Pomodoro, chuyển qua thời gian nghỉ, và sau 4 chu kỳ sẽ chuyển sang nghỉ dài.
* **Thông báo và âm thanh**: Phát âm thanh hoặc hiển thị thông báo khi hết một khoảng thời gian.

**3. Xây dựng chức năng (JavaScript)**

**JavaScript cơ bản**

1. **Khởi tạo các biến cần thiết:** Các biến để lưu thời gian Pomodoro, thời gian nghỉ ngắn và dài, chu kỳ hiện tại.
2. **Đếm ngược thời gian:** Sử dụng setInterval để đếm ngược thời gian và cập nhật giao diện mỗi giây.
3. **Chuyển đổi giữa thời gian làm việc và thời gian nghỉ:** Xác định logic để chuyển qua lại giữa thời gian làm việc và nghỉ sau mỗi chu kỳ Pomodoro.

**4. Cải tiến và nâng cấp**

* **Âm thanh báo hiệu**: Thêm âm thanh khi kết thúc mỗi phiên Pomodoro.
* **Thiết lập thời gian tuỳ chỉnh**: Cho phép người dùng nhập thời gian Pomodoro và nghỉ ngơi theo ý muốn.
* **Hiển thị thời gian biểu đồ**: Dùng thư viện Chart.js để hiển thị dữ liệu về thời gian làm việc và nghỉ ngơi.
* **Thống kê số phiên làm việc**: Tạo một mục thống kê số Pomodoro hoàn thành.

**5. Tích hợp thư viện quản lý thời gian (Moment.js/Day.js)**

Sử dụng Moment.js hoặc Day.js để quản lý thời gian một cách chính xác hơn, đặc biệt hữu ích nếu muốn thêm các tính năng phức tạp hơn về thời gian.

Dự án này sẽ giúp bạn thực hành các khái niệm về quản lý thời gian, sự kiện và logic điều khiển trong JavaScript, đồng thời tạo ra một công cụ hữu ích cho chính bạn khi làm việc theo phương pháp Pomodoro.

Nếu bạn chạy trang web Pomodoro Timer này trong một cửa sổ mới, hoặc thậm chí refresh (tải lại) trang, thì thời gian đếm ngược có khả năng **không còn chính xác** nữa. Điều này là do JavaScript sẽ chỉ đếm thời gian trong bộ nhớ khi cửa sổ hoặc tab đang mở và vẫn duy trì trạng thái.

**Các nguyên nhân chính:**

1. **Không có dữ liệu lưu trữ thời gian còn lại**: JavaScript chỉ chạy trong bộ nhớ của trình duyệt và sẽ không lưu lại trạng thái khi bạn tải lại hoặc mở trang mới. Khi đóng tab hoặc làm mới, bộ đếm sẽ quay lại thời gian ban đầu mà không tiếp tục từ điểm dừng.
2. **Bộ đếm dừng khi tab không hoạt động**: Trình duyệt sẽ giảm tài nguyên cho các tab không hoạt động hoặc bị ẩn để tiết kiệm tài nguyên hệ thống, khiến các bộ đếm setInterval và setTimeout có thể tạm dừng hoặc không chính xác.

**Cách khắc phục**

Để đảm bảo bộ đếm tiếp tục đếm ngay cả khi tải lại hoặc mở trong cửa sổ mới, bạn có thể lưu trạng thái của bộ đếm vào localStorage. Khi trang được tải lại, ứng dụng sẽ kiểm tra thời gian lưu trong localStorage và tiếp tục đếm từ thời điểm dừng trước đó.

Khi tab không hoạt động, nhiều trình duyệt sẽ làm chậm hoặc tạm dừng các hoạt động JavaScript không quan trọng như setInterval hoặc setTimeout. Điều này xảy ra vì trình duyệt muốn tiết kiệm tài nguyên khi người dùng không thực sự xem trang web. Do đó, các bộ đếm thời gian như trong Pomodoro Timer có thể không chính xác khi tab hoặc cửa sổ không hoạt động.

**Giải pháp Khắc phục**

Để đảm bảo bộ đếm vẫn chính xác bất kể tab có đang hoạt động hay không, bạn có thể sử dụng **timestamp** để tính thời gian trôi qua thay vì dựa vào setInterval. Ý tưởng là lưu thời điểm bắt đầu và mỗi lần cập nhật sẽ tính toán lại thời gian dựa trên thời gian thực tế đã trôi qua.

**Cách Thực Hiện:**

1. **Lưu thời điểm bắt đầu**: Khi bắt đầu bộ đếm, lưu thời gian hiện tại (thời điểm bắt đầu) vào localStorage.
2. **Tính toán thời gian trôi qua**: Thay vì giảm từng giây trong mỗi lần setInterval, bạn lấy thời gian hiện tại, tính toán thời gian đã trôi qua kể từ thời điểm bắt đầu, và cập nhật bộ đếm.

**Giải thích**

* **Thời gian kết thúc** (endTime) được lưu lại dựa trên thời gian hiện tại cộng với thời gian đếm ngược.
* **Hàm startTimer**: Thiết lập endTime và cứ mỗi giây, tính thời gian còn lại bằng cách lấy endTime trừ currentTime.
* **Làm mới hoặc chuyển tab**: Khi trang tải lại, script kiểm tra endTime trong localStorage. Nếu còn thời gian, bộ đếm sẽ tiếp tục từ thời điểm chính xác.

Bằng cách này, bộ đếm sẽ luôn cập nhật chính xác bất kể tab hoặc cửa sổ có đang hoạt động hay không, nhờ vào tính toán dựa trên thời gian thực tế đã trôi qua.

Để đảm bảo bộ đếm thời gian vẫn chính xác khi chuyển đổi tab hoặc làm mới trang, hãy lưu thời điểm kết thúc của mỗi chu kỳ (làm việc/nghỉ ngơi) và sử dụng timestamp để tính thời gian còn lại. Điều này giúp bộ đếm không bị ảnh hưởng bởi việc trình duyệt làm chậm hoặc tạm dừng hoạt động khi tab không hoạt động.

**Giải thích**

* **endTime**: Thời gian kết thúc của mỗi chu kỳ được tính từ thời điểm hiện tại cộng với thời gian chu kỳ và lưu vào localStorage.
* **startTimer**: Thay vì trừ đi từng giây, hàm tính thời gian còn lại bằng cách lấy thời gian kết thúc trừ thời gian hiện tại, giúp bộ đếm chính xác ngay cả khi tab không hoạt động.
* **window.load**: Khi trang tải lại, kiểm tra thời gian kết thúc còn lưu trong localStorage. Nếu còn thời gian, bộ đếm sẽ tiếp tục từ điểm dừng.