**BÁO CÁO BÀI TẬP ONLINE 2 - LẬP TRÌNH ĐỒ HỌA VỚI CANVAS**

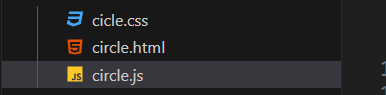
1. **Giới thiệu:**

Lập trình đồ họa là một lĩnh vực quan trọng trong phát triển giao diện người dùng. Bài tập này nhằm giúp sinh viên làm quen với việc vẽ đồ họa sử dụng thẻ <canvas> trong HTML5 bằng hai phương pháp:

**+ Dùng JavaScript thuần** (HTML + CSS + JS)

**+ Dùng React kết hợp với React Hook và Canvas**

1. **Thẻ canvas với HTML5**

 **2.1 Vẽ hình tròn với canvas**

**2.1.1 Cấu trúc file**

**circle.html: chứa phần tử <canvas> để hiển thị hình tròn**

**circle.js:** Chứa logic để vẽ và tạo hiệu ứng phóng to thu nhỏ hình tròn.

**circle.css**: thiết lập kiểu dáng cho canvas.

**2.1.2 Vẽ hình tròn với canvas trong HTML5**

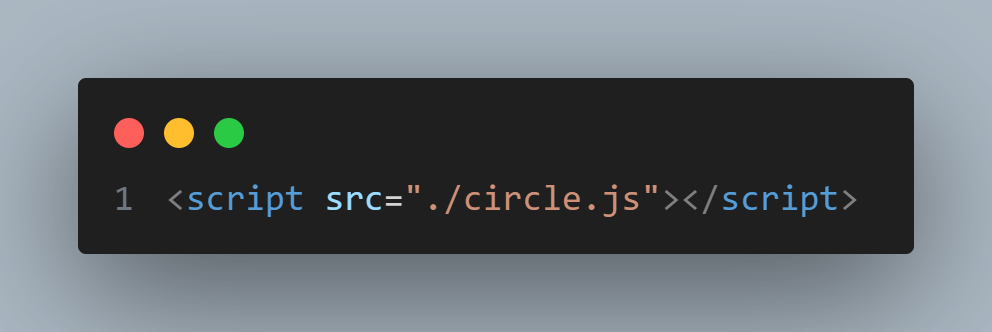
**2.1.2.1: File circle.html**



- Tạo một phần tử <canvas id="myCanvas"> để vẽ hình tròn.



- Gọi file **circle.js** để thực thi logic vẽ.



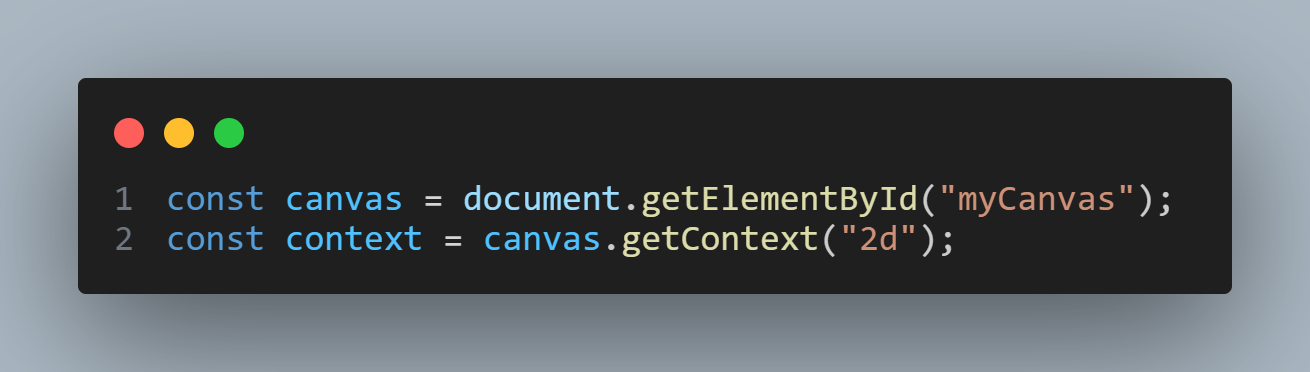
**2.1.2.2 File circle.js**



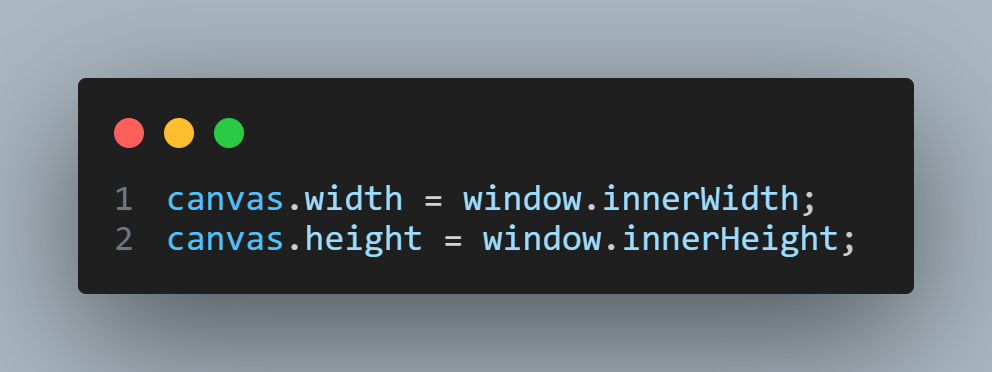
- Lấy thẻ <canvas> và tạo context vẽ 2D

+ getElementById("myCanvas"): Truy xuất phần tử <canvas>.

+ canvas.getContext("2d"): Lấy context 2D để vẽ.



- Kích thước toàn màn hình



- Khai báo biến điều khiển vòng tròn

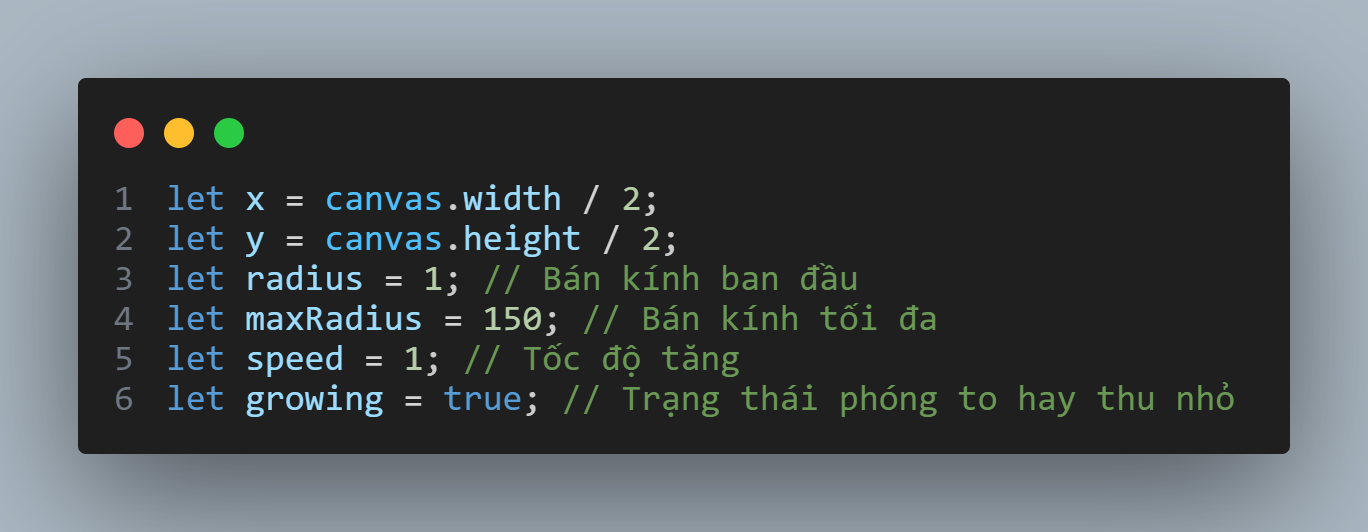
+ Xác định vị trí trung tâm (x, y).

+ radius: Kích thước ban đầu của hình tròn.

+ maxRadius: Kích thước tối đa.

+ speed: Tốc độ mở rộng.

+ growing: Trạng thái đang lớn lên hay nhỏ lại.



- Hàm thực hiện vẽ hình tròn

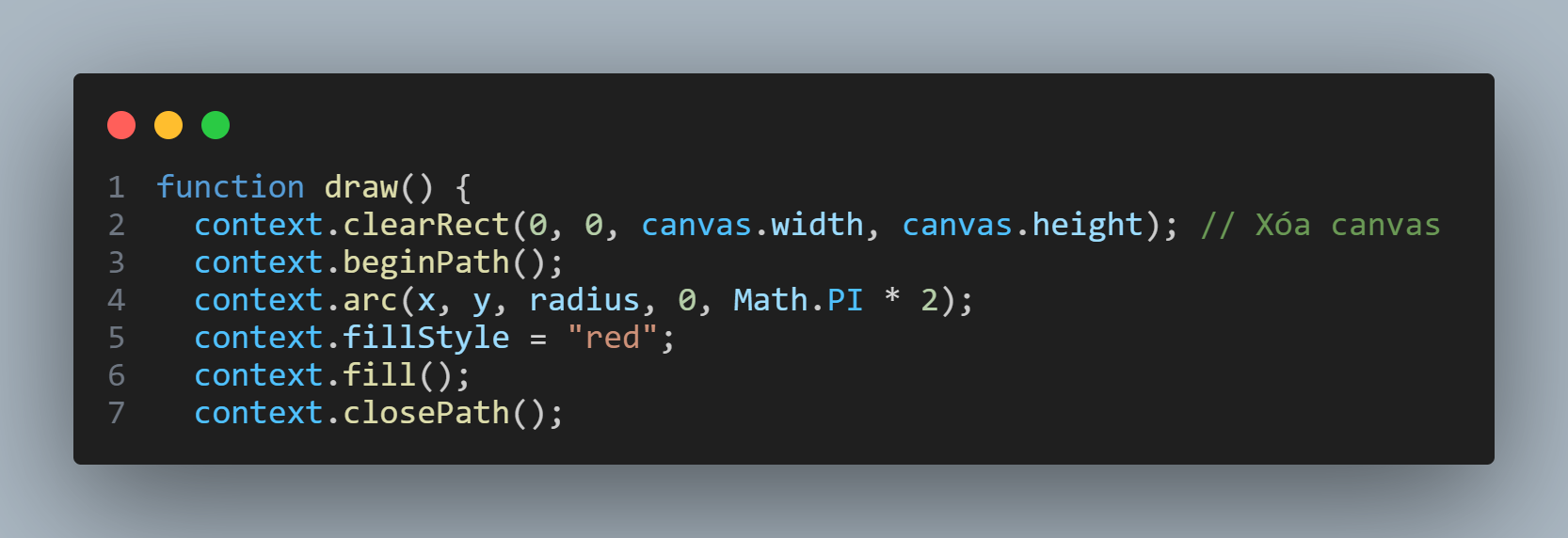
+ clearRect(0, 0, canvas.width, canvas.height): Xóa nội dung cũ để vẽ mới.

+ beginPath(): Bắt đầu vẽ hình mới.

+ arc(x, y, radius, 0, Math.PI \* 2): Vẽ hình tròn tại (x, y) với bán kính radius.

+ fillStyle = "red": Đổ màu đỏ.

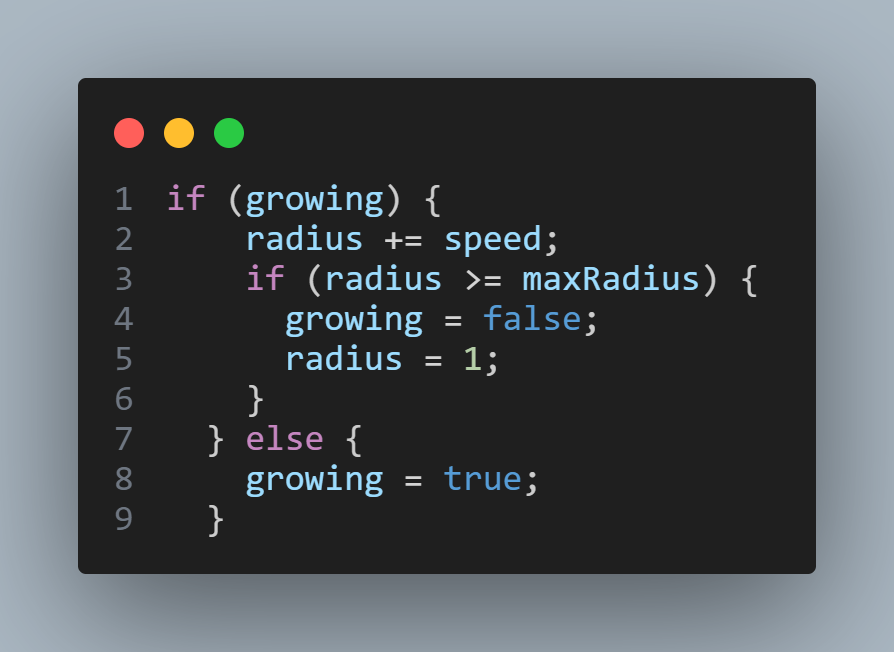
+ fill(): Tô màu hình tròn.



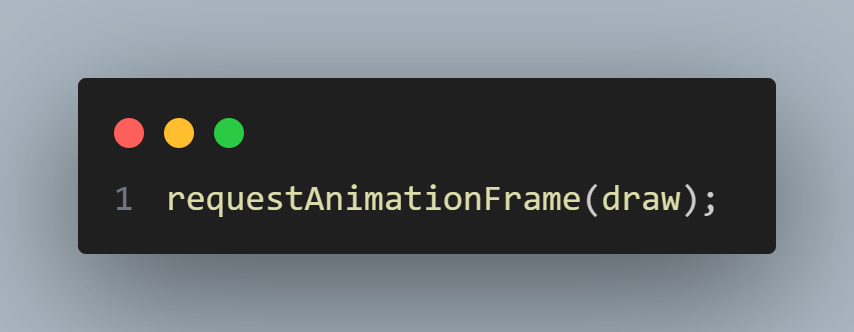
- Điều kiện cập nhật kích thước hình trong

+ Nếu growing = true, tăng bán kính.

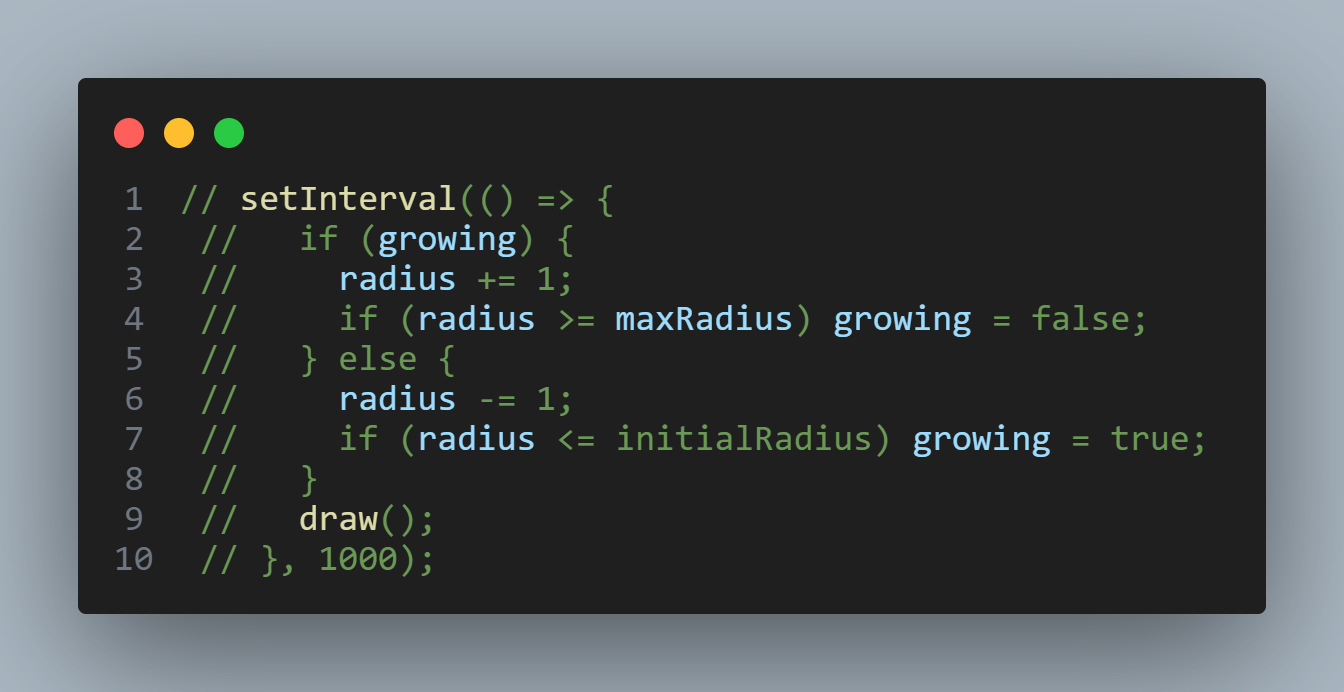
+ Nếu đạt maxRadius, đặt lại bán kính về 1 và đổi trạng thái.



- Tạo hiệu ứng hoạt hình bằng requestAnimationFrame(draw)



**C2**: Ta có thể áp dụng setInterval() để vẽ hình tròn thay vì dùng requestAnimationFrame().



**2.2 Vẽ hình bình hành với canvas**

**2.2.1 Cấu trúc file**

**index.html**: Chứa phần tử <canvas> để hiển thị hình bình hành.

**gradient.js**: Chứa logic để vẽ và tạo hiệu ứng gradient động.

**gradient.css**: (nếu có) để thiết lập kiểu dáng cho canvas.

- Tạo phần tử với canvas

+ Tạo một phần tử <canvas id="myCanvas2"> để vẽ hình bình hành.

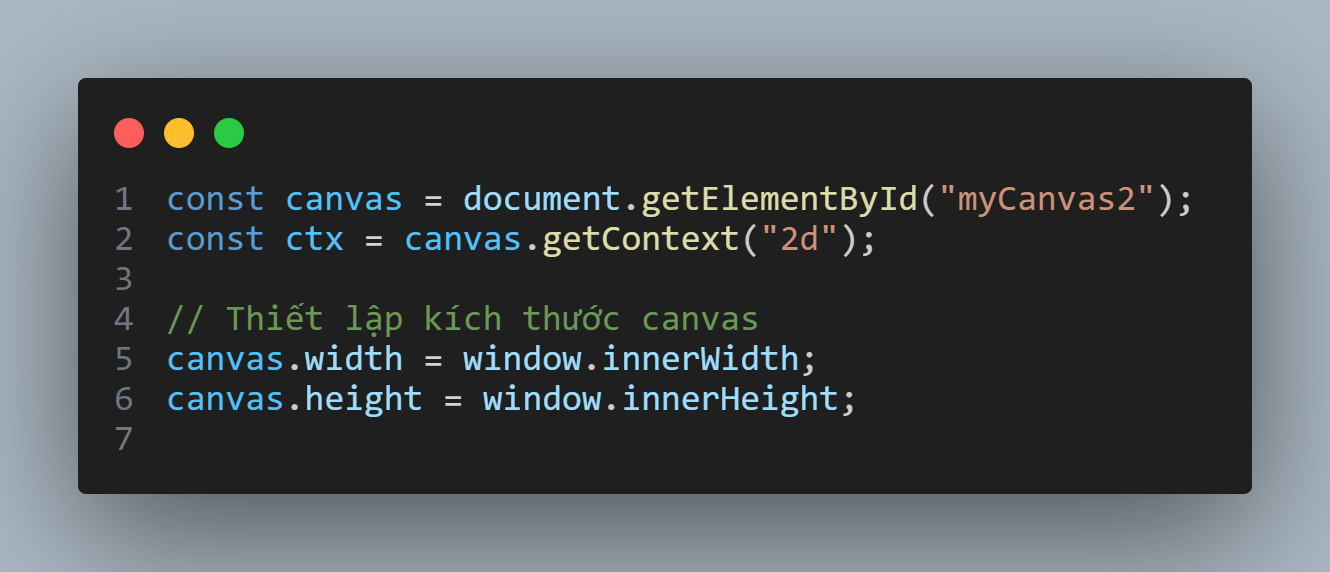
+ Gọi file **gradient.js** để thực thi logic vẽ.



- Khởi tạo canvas

+ Lấy phần tử <canvas> và tạo **context 2D** để vẽ.

+ **Thiết lập kích thước** để canvas bao phủ toàn bộ màn hình.



- Khởi tạo thông số hình bình hành

+ x, y: Tọa độ của hình bình hành.

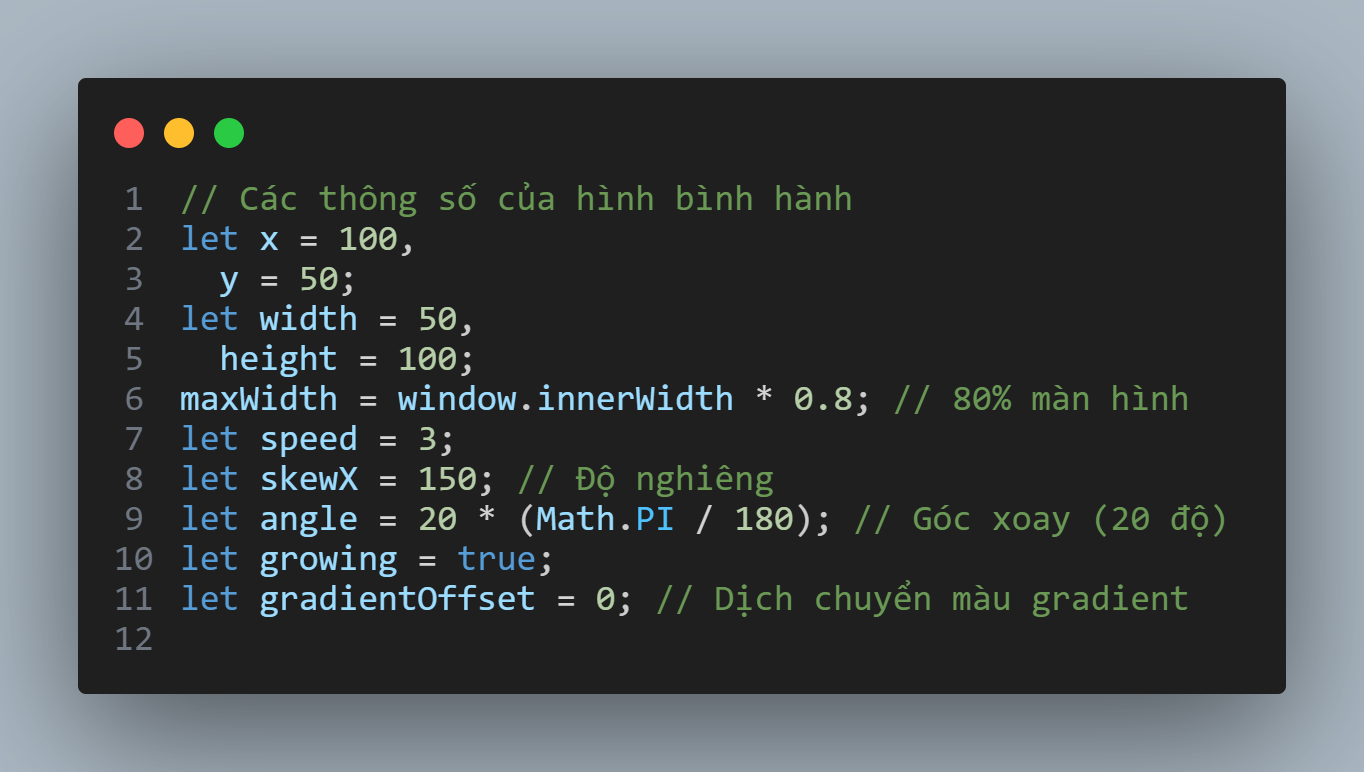
+ width, height: Kích thước ban đầu.

+ skewX: Độ nghiêng hình bình hành

+ angle: Xoay hình **20 độ**.

+ growing: Biến kiểm soát tăng kích thước.

+ gradientOffset: Điều chỉnh vị trí gradient.



- Danh sách màu gradient

+ Danh sách màu cho hiệu ứng gradient.

+ Khi hình đạt kích thước tối đa, màu sẽ đổi tuần tự.



- Hàm vẽ hình bình hành

+ translate(x, y): Di chuyển hình vẽ đến vị trí (x, y).

+ rotate(angle): Xoay hình một góc **20 độ**.

+ **Tạo gradient** với các màu trong danh sách.

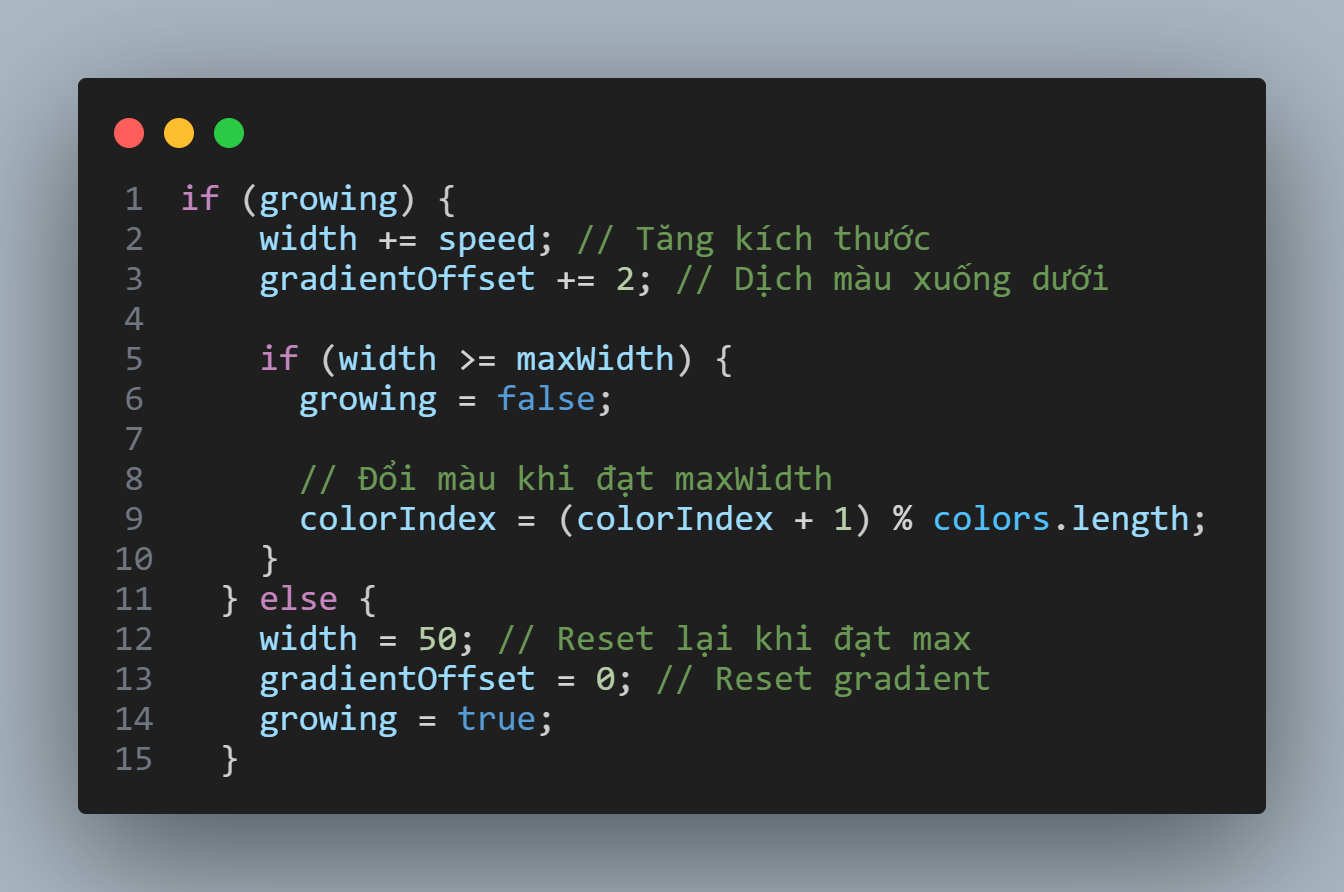
+ Vẽ hình bình hành bằng beginPath() và fill().



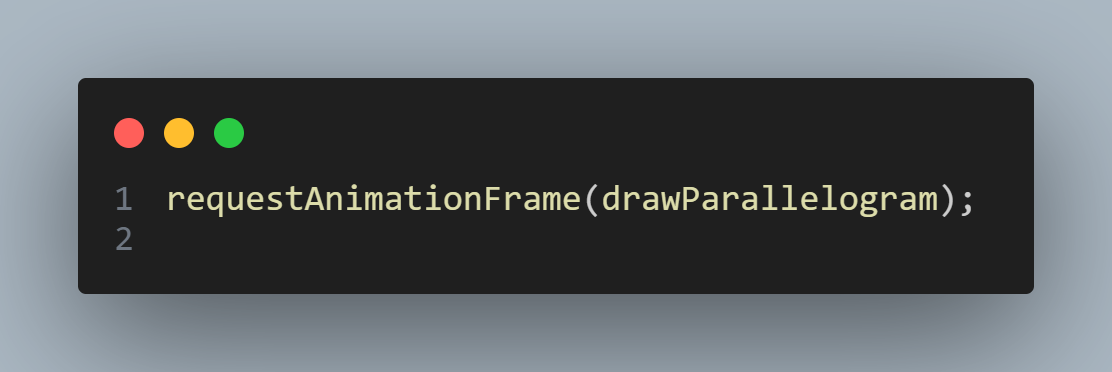
- Hiệu ứng mở rộng và đổi màu

+ Nếu growing = true, hình mở rộng dần.

+ Khi đạt maxWidth, **đổi màu** và bắt đầu lại từ đầu.



- Tạo hiệu ứng animation: **Liên tục gọi lại** hàm drawParallelogram().



1. Sử dụng **React + Hooks (useRef, useEffect)** trên <canvas>

**3.1 React + Hooks (useRef, useEffect)** để vẽ hình tròn động trên <canvas>.

- Khởi tạo và sử dụng useRef

+ useRef giúp **truy xuất phần tử** <canvas> trong React.

+ canvasRef.current sẽ chứa <canvas> .



- Cấu hình Canvas trong useEffect

+ **Lấy thẻ** <canvas> thông qua canvasRef.current.

+ **Lấy** context 2D để vẽ hình.

+ **Cập nhật kích thước canvas** bằng window.innerWidth/Height.



- Logic vẽ hình tròn

+ x, y: Tọa độ trung tâm.

+ radius: Kích thước ban đầu.

+ maxRadius: Giới hạn tối đa.

+ speed: Tốc độ mở rộng.

+ growing: Biến kiểm soát phóng to/thu nhỏ.



- Hàm draw()

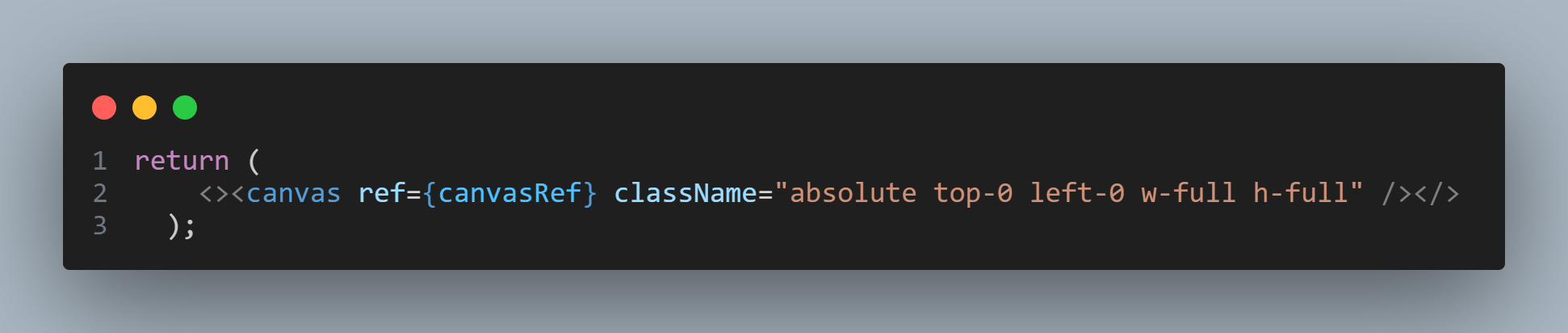
+ **Vẽ hình tròn** tại tọa độ (x, y).

+ Khi radius < maxRadius ➝ tiếp tục mở rộng.

+ Khi đạt maxRadius ➝ reset về 1 và lặp lại.



- Render Canvas: ref={canvasRef} để liên kết với useRef.



**3.2 Vẽ Hình Bình Hành Động với React và Canvas**

**-** Sử dụng useRef để lưu trạng thái

+ canvasRef: Lưu tham chiếu đến <canvas>.

+ widthRef: Lưu kích thước chiều rộng hiện tại.

+ growingRef: Biến cờ kiểm tra **đang mở rộng** hay **reset lại**.

+ colorIndexRef: Theo dõi màu sắc gradient hiện tại.

+ animationRef: Lưu ID của requestAnimationFrame để hủy khi unmount.



- Khai báo các thông số hình bình hành

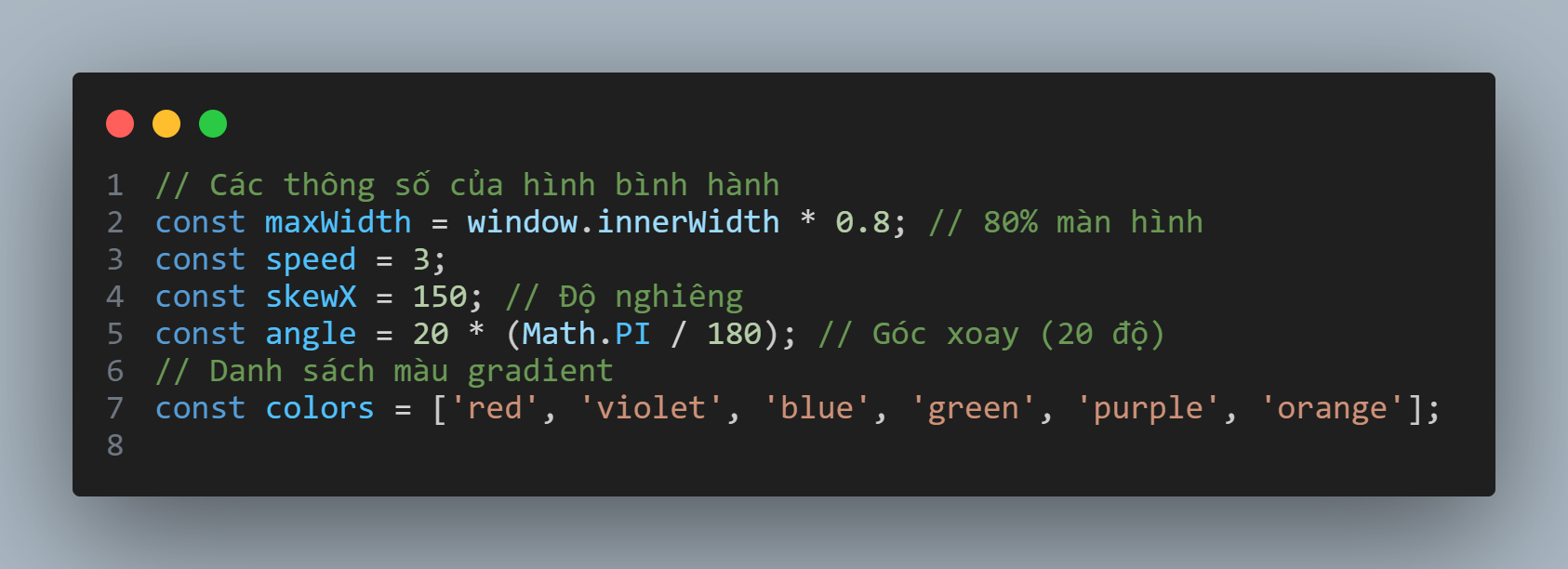
+ maxWidth: Kích thước tối đa của hình bình hành.

+ speed: Tốc độ mở rộng.

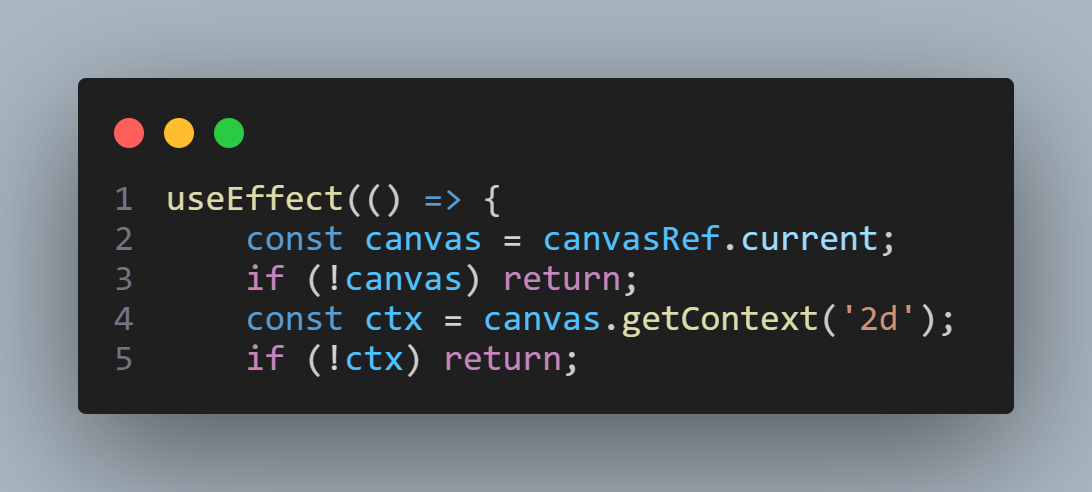
+ skewX: Độ nghiêng của hình bình hành.

+ angle: Góc xoay của hình.

+ colors: Mảng màu gradient để đổi màu mỗi lần reset.



- useEffect – Vẽ hình bình hành động: lấy **thẻ** <canvas> và **ngữ cảnh 2D (**ctx**)** để vẽ.

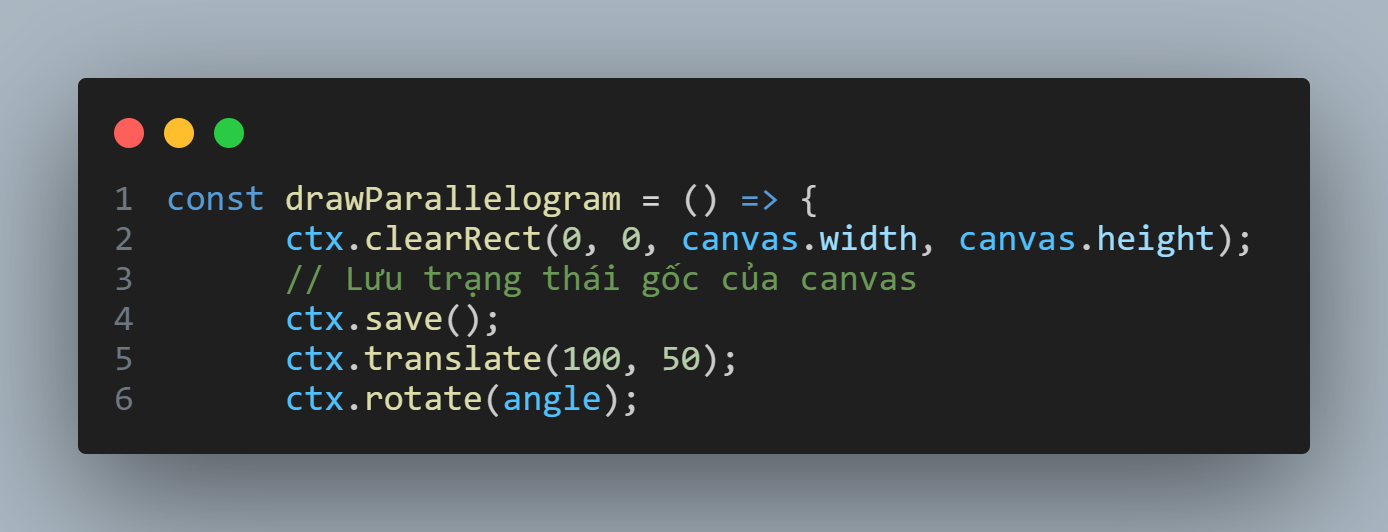


- Hàm drawParallelogram()

+ **Xóa canvas cũ (**clearRect**)** trước khi vẽ hình mới.

+ **Lưu trạng thái canvas (**save**)** để áp dụng xoay (rotate).

+ **Di chuyển đến vị trí vẽ (**translate**)** để xoay đúng tâm.

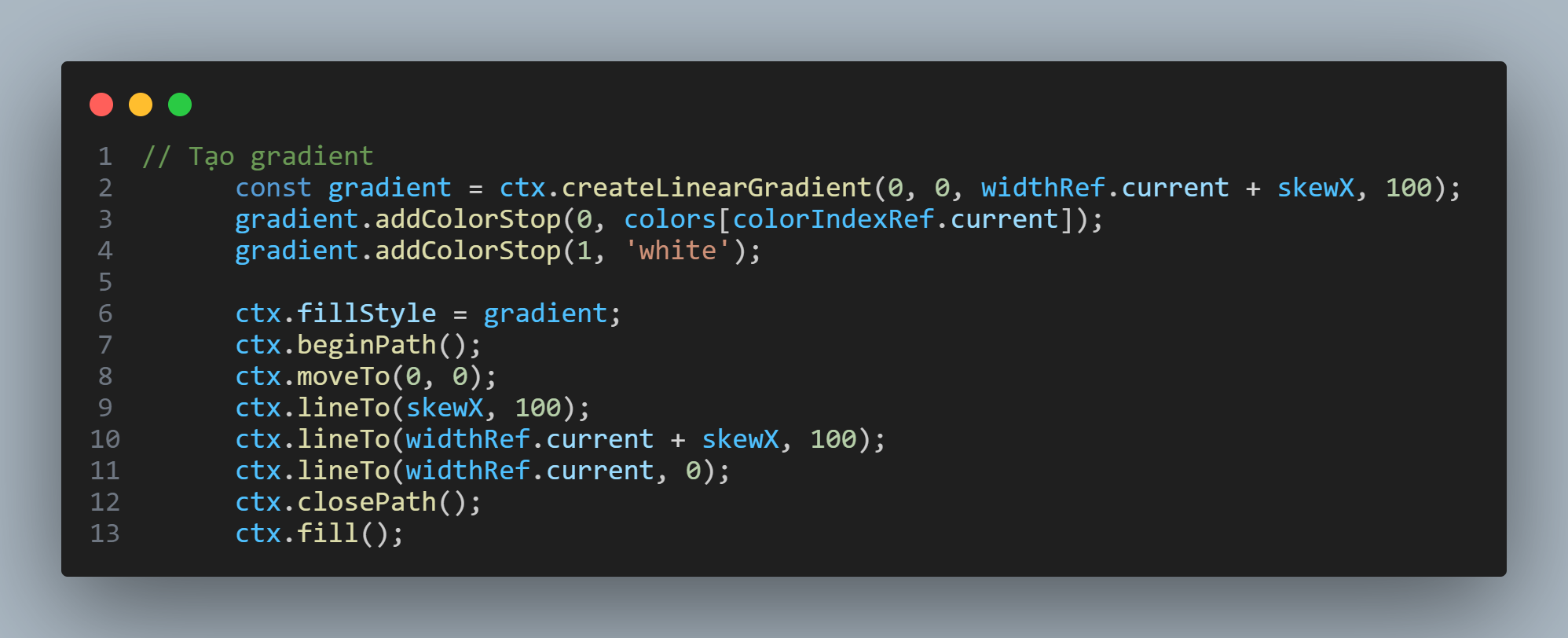


- Vẽ hình bình hành với Gradient màu

+ **Tạo màu gradient động**, thay đổi màu theo colorIndexRef.current.

+ **Vẽ hình bình hành** bằng moveTo() và lineTo().

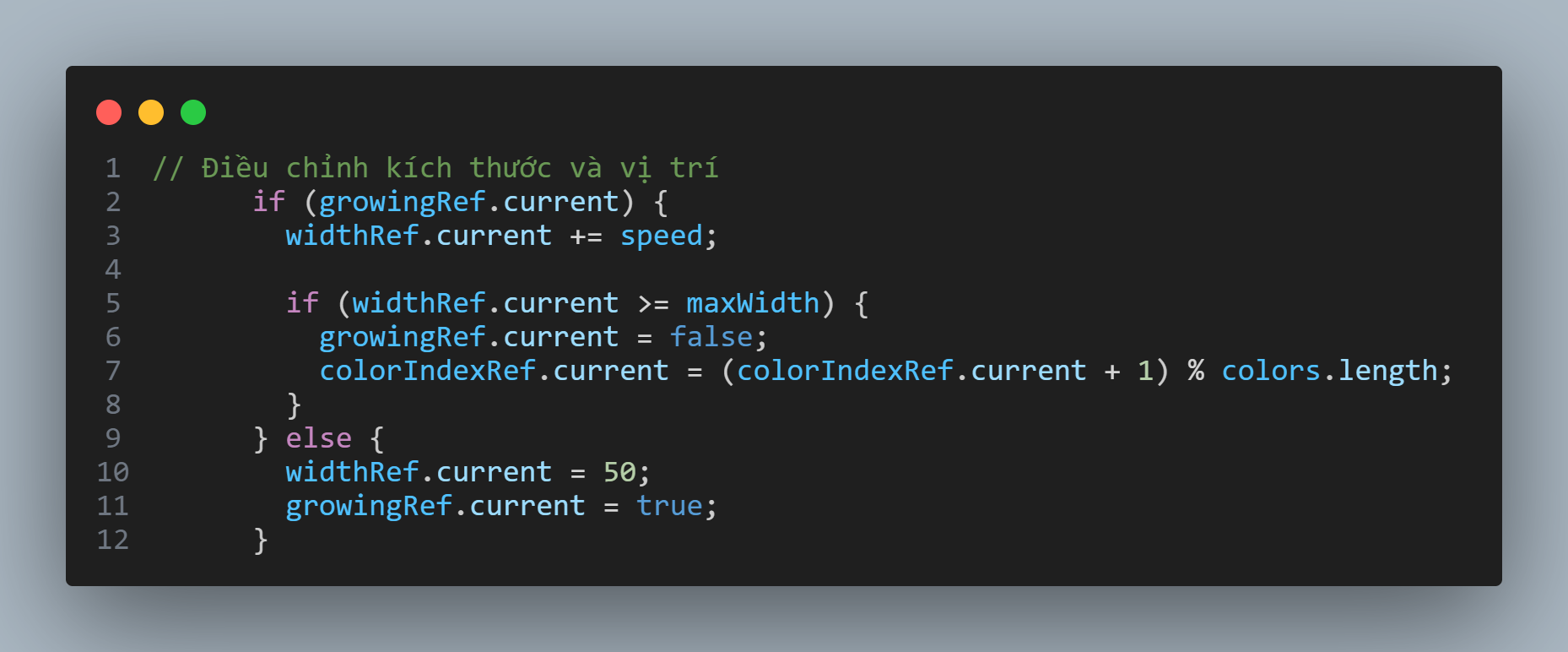
+ fillStyle = gradient để đổ màu.



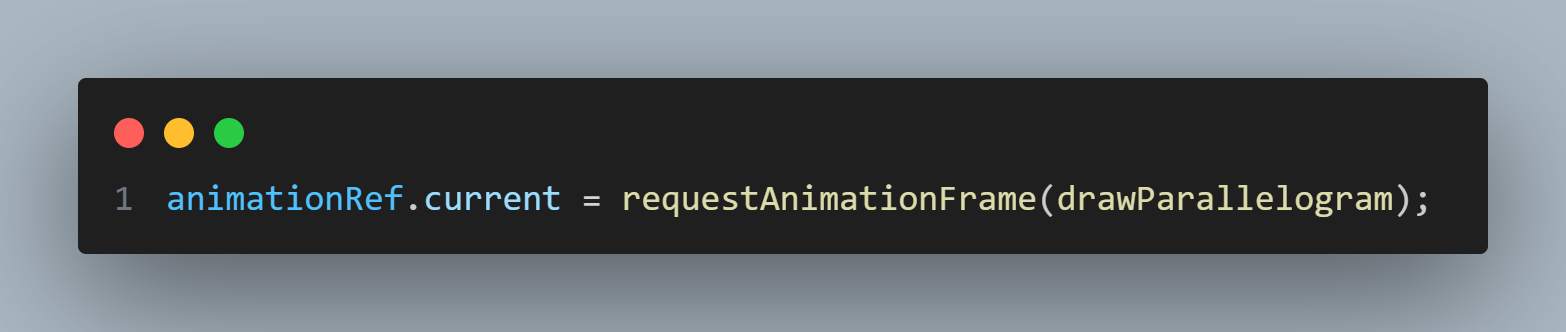
**-** Điều chỉnh kích thước và màu sắc

+ (growingRef.current = true) --> **tăng kích thước**.

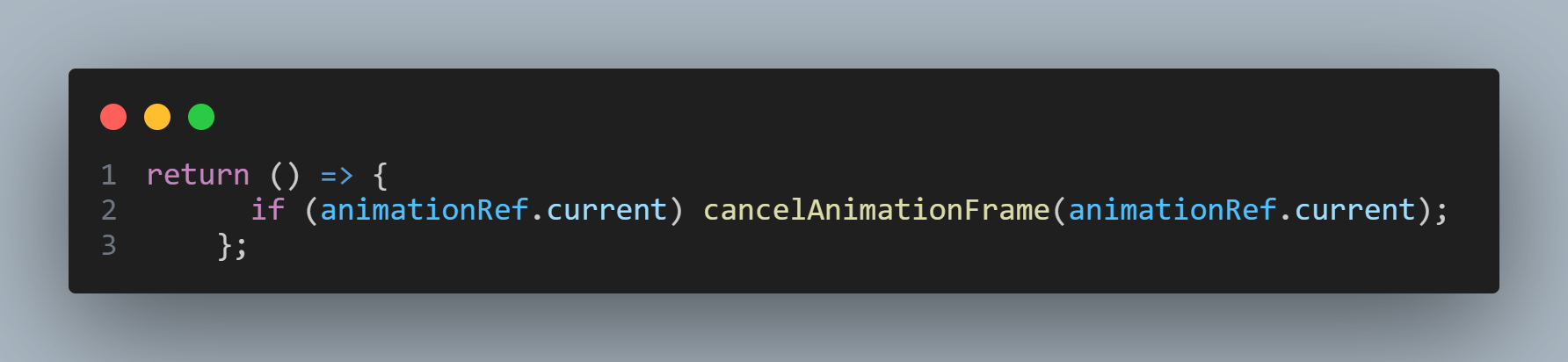
+ Nếu đạt maxWidth, **reset về 50** và **đổi màu** (colorIndexRef).



- Lặp lại hiệu ứng với requestAnimationFrame



- Cleanup khi component bị unmount



- Render Canvas

+ **Gán** ref={canvasRef} để truy cập <canvas>.

+ **Thiết lập kích thước bằng** window.innerWidth/Height.

