**CHƯƠNG TRÌNH HỌC CSS TOÀN DIỆN**Mã sinh viên: 1071020091  
Họ và tên: Nguyễn Chí Đạt

**PHIẾU HỌC TẬP CSS [14.1] - GRID: DỰNG KHUNG (CONTAINER)  
Video đã xem (Link):  
1. Kiến thức cốt lõi (Layout Hai chiều)***Nếu* ***Flexbox*** *là công cụ layout 1 chiều (hoặc ngang, hoặc dọc), thì* ***Grid*** *là công cụ layout 2  
chiều (cả ngang và dọc CÙNG LÚC). Nó cho phép bạn tạo ra các bố cục cột và hàng phức tạp  
một cách dễ dàng.  
Sau khi xem video, hãy điền vào mục đích của các thuộc tính của thẻ* ***CHA*** *(Grid Container):*

|  |  |  |
| --- | --- | --- |
| Thuộc tính (của Cha) | Mục đích sử dụng (Dùng để làm gì?) | Ví dụ giá trị phổ biến |
| **display: grid;** | (Tự điền: "Kích hoạt" môi trường layout Grid) | grid |
| **grid-template columns** | (Tự điền: Định nghĩa số lượng VÀ độ rộng của các CỘT) | 100px 100px, 1fr 2fr, repeat(3, 1fr) |
| **grid-template rows** | (Tự điền: Định nghĩa số lượng VÀ độ cao của các DÒNG) | 200px 400px, auto |
| **gap** (hoặc grid-gap) | (Tự điền: Tạo khoảng trống...) | 10px, 10px 20px |

Xuất sang Trang tính  
**Câu hỏi bắt buộc:**1. Đơn vị fr (viết tắt của "Fraction" - Phân số) có nghĩa là gì?

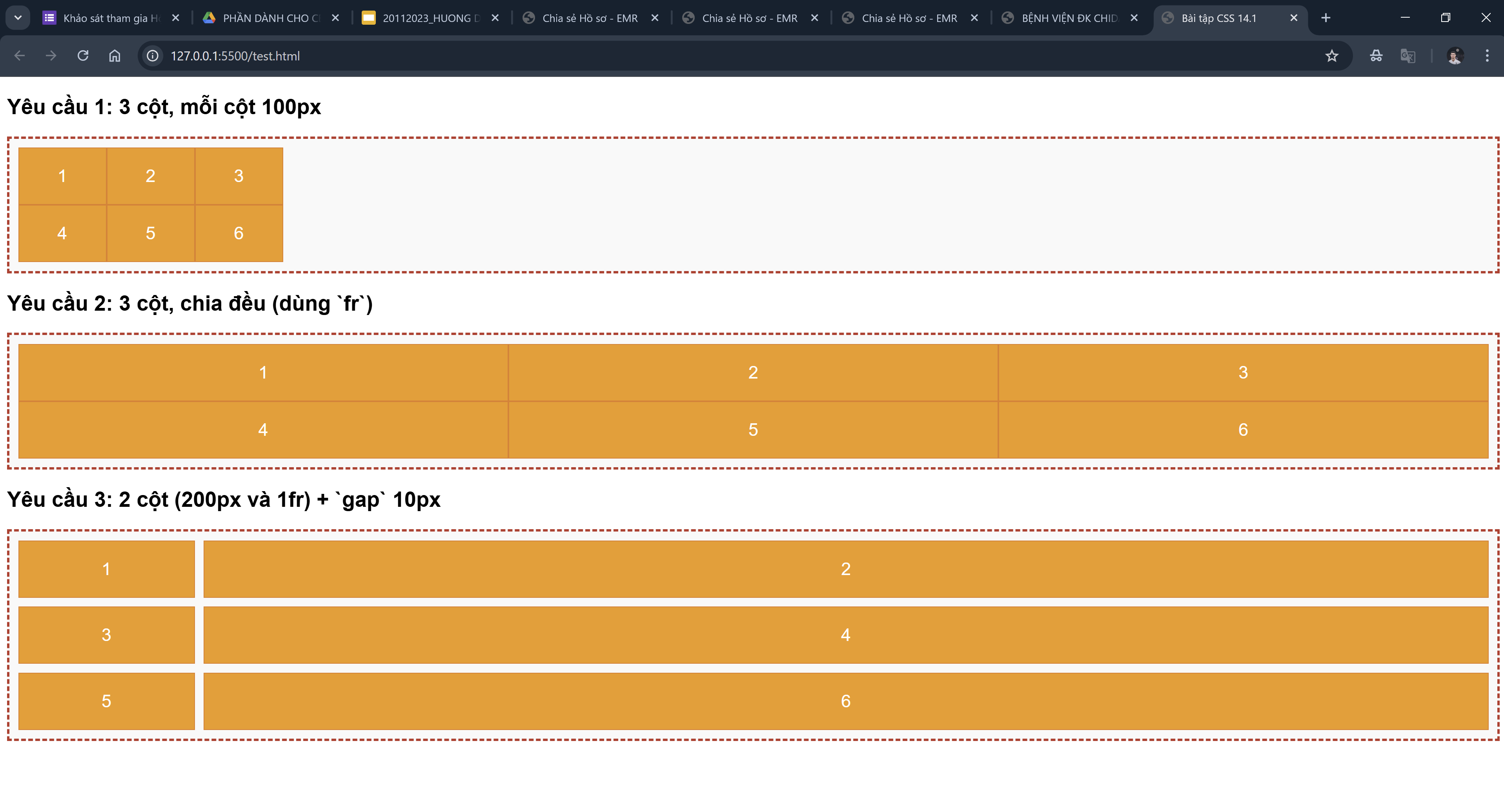
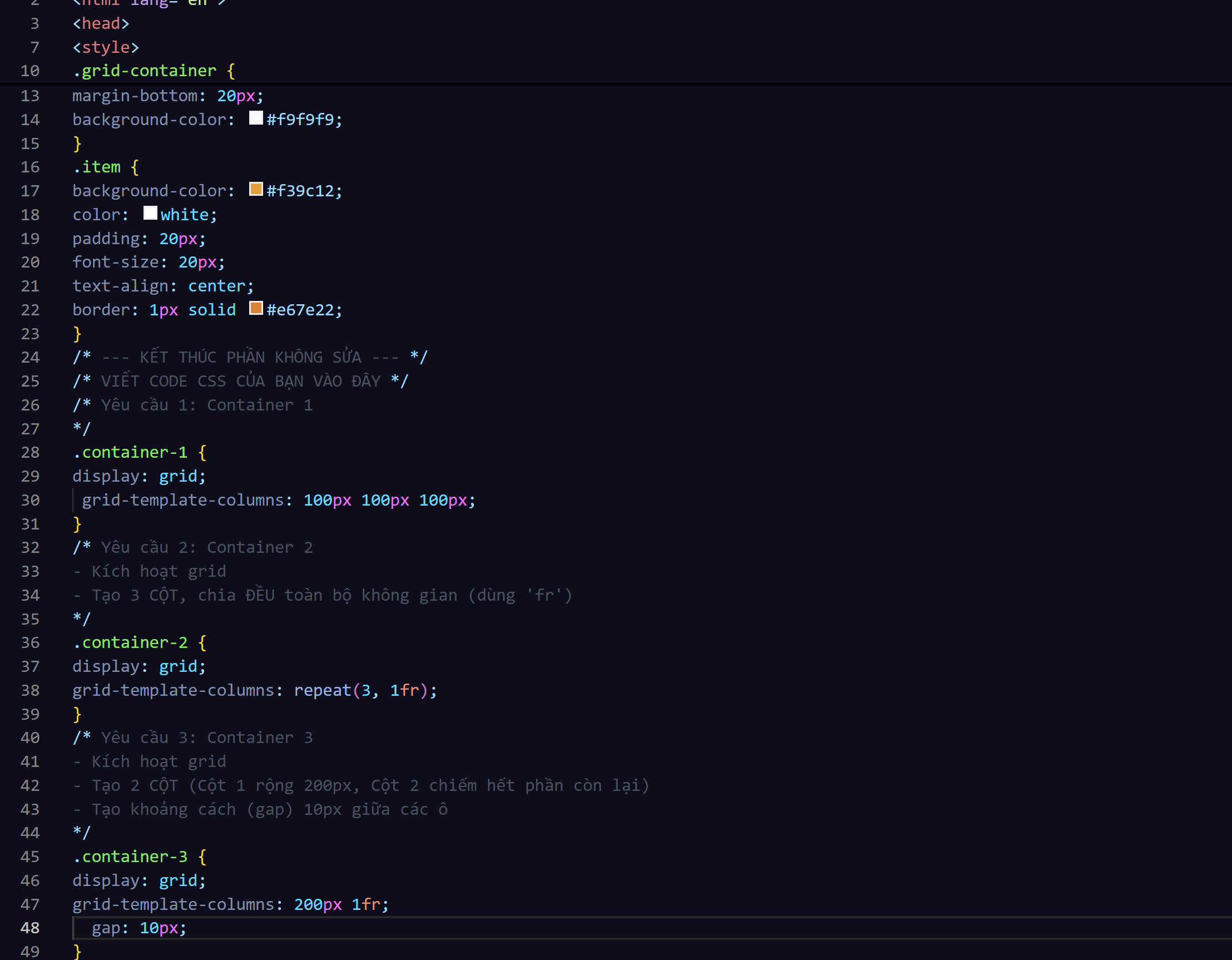
Đơn vị **fr** (viết tắt của **"fraction"**, nghĩa là **phân số**) là một **đơn vị đo lường đặc biệt trong CSS Grid Layout**, dùng để chia **phần không gian còn lại** của lưới (grid container) thành **các phần tỷ lệ**.

2. Nếu bạn viết grid-template-columns: 1fr 2fr;, bố cục sẽ được chia như thế nào? (Gợi ý:  
Tổng là 3 phần, cột 1 chiếm... phần, cột 2 chiếm... phần).

thì:

* Tổng số phần là **1fr + 2fr = 3 phần**.
* **Cột 1** chiếm **1 phần**,
* **Cột 2** chiếm **2 phần**.

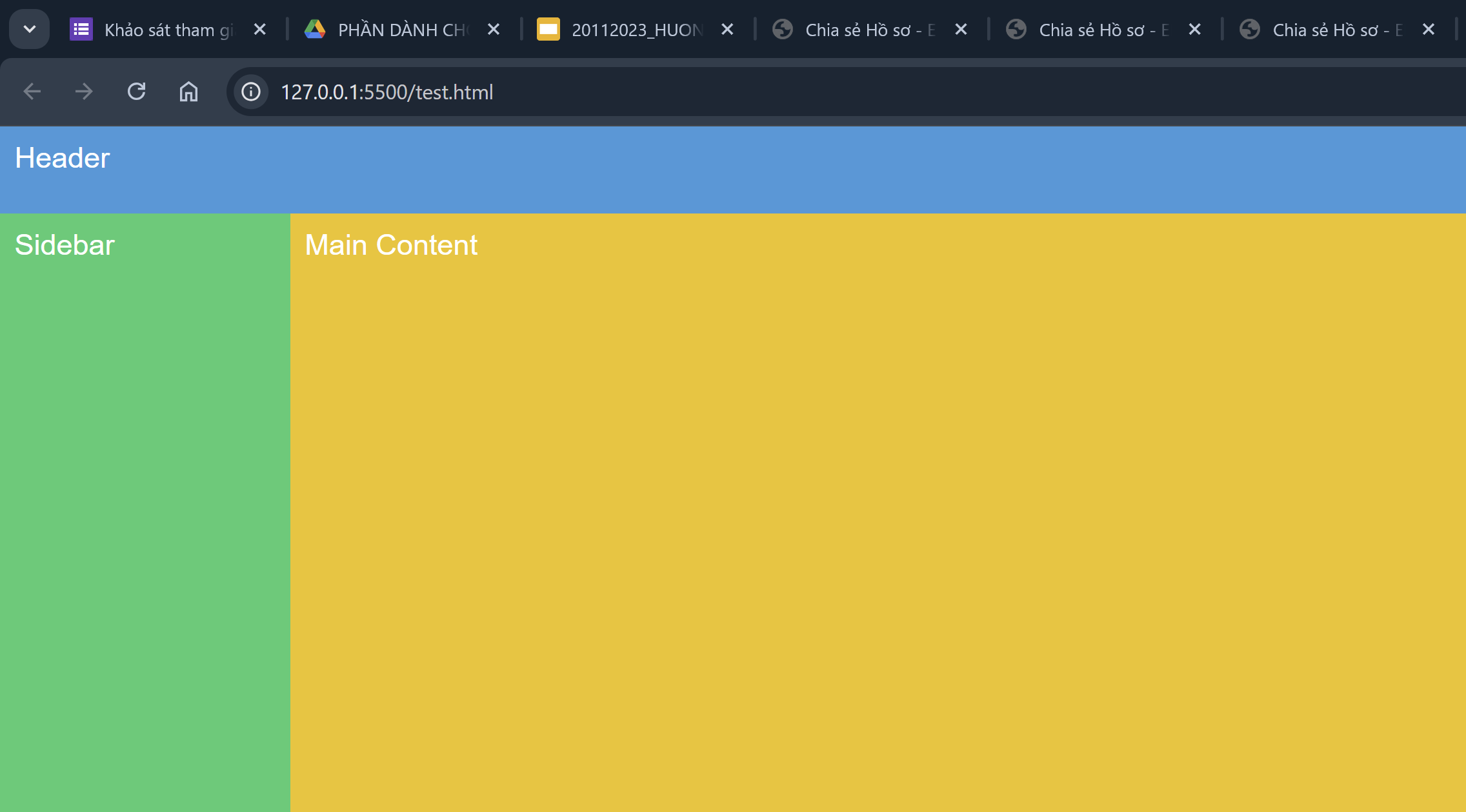
3. Hàm repeat(3, 1fr) tương đương với việc viết 1fr 1fr 1fr đúng không?

Đúng rồi  
**2. Ví dụ thực hành (Bắt buộc)***Yêu cầu: Viết CSS cho thẻ cha .grid-container để sắp xếp 6 thẻ con .item theo đúng 3 yêu cầu  
bố cục bên dưới***Mã HTML của tôi (index.html):**HTML  
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<meta name="viewport" content="width=device-width, initial-scale=1.0">  
<title>Bài tập CSS 14.1</title>  
<style>  
/\* --- KHÔNG SỬA PHẦN NÀY --- \*/  
body { font-family: Arial, sans-serif; }  
.grid-container {  
border: 3px dashed #c0392b;  
padding: 10px;  
margin-bottom: 20px;  
background-color: #f9f9f9;  
}  
.item {  
background-color: #f39c12;  
color: white;  
padding: 20px;  
font-size: 20px;  
text-align: center;  
border: 1px solid #e67e22;  
}  
/\* VIẾT CODE CSS CỦA BẠN VÀO ĐÂY \*/  
/\* Yêu cầu 1: Container 1  
- Kích hoạt grid  
- Tạo 3 CỘT, mỗi cột RỘNG 100px  
- (Hàn/\* --- KẾT THÚC PHẦN KHÔNG SỬA --- \*/  
g sẽ tự động được tạo)  
\*/  
.container-1 {  
display: grid;  
}  
/\* Yêu cầu 2: Container 2  
- Kích hoạt grid  
- Tạo 3 CỘT, chia ĐỀU toàn bộ không gian (dùng 'fr')  
\*/  
.container-2 {  
display: grid;  
}  
/\* Yêu cầu 3: Container 3  
- Kích hoạt grid  
- Tạo 2 CỘT (Cột 1 rộng 200px, Cột 2 chiếm hết phần còn lại)  
- Tạo khoảng cách (gap) 10px giữa các ô  
\*/  
.container-3 {  
display: grid;  
}  
</style>  
</head>  
<body>  
<h2>Yêu cầu 1: 3 cột, mỗi cột 100px</h2>  
<div class="grid-container container-1">  
<div class="item">1</div>  
<div class="item">2</div>  
<div class="item">3</div>  
<div class="item">4</div>  
<div class="item">5</div>  
<div class="item">6</div>  
</div>  
<h2>Yêu cầu 2: 3 cột, chia đều (dùng `fr`)</h2>  
<div class="grid-container container-2">  
<div class="item">1</div>  
<div class="item">2</div>  
<div class="item">3</div>  
<div class="item">4</div>  
<div class="item">5</div>  
<div class="item">6</div>  
</div>  
<h2>Yêu cầu 3: 2 cột (200px và 1fr) + `gap` 10px</h2>  
<div class="grid-container container-3">  
<div class="item">1</div>  
<div class="item">2</div>  
<div class="item">3</div>  
<div class="item">4</div>  
<div class="item">5</div>  
<div class="item">6</div>  
</div>  
</body>  
</html>  
**Kết quả (Chụp ảnh màn hình và dán vào đây):** *(Dán ảnh chụp màn hình trình duyệt của bạn  
vào đây. Kết quả phải là:* • *Hộp 1: 6 khối cam xếp thành lưới 3 cột 2 hàng. Chiều rộng của 3 cột CỐ ĐỊNH là 100px.*• *Hộp 2: 6 khối cam xếp thành lưới 3 cột 2 hàng. 3 cột này tự động PHÌNH RA để lấp đầy  
chiều rộng của container.*• *Hộp 3: 6 khối cam xếp thành lưới 2 cột 3 hàng. Cột 1 cố định 200px, cột 2 phình ra lấp  
đầy phần còn lại. Quan trọng: Có một KHOẢNG TRỐNG (gap) 10px giữa các khối cam. )***3. Câu hỏi/Thắc mắc của tôi  
Một điều tôi chưa hiểu rõ hoặc muốn hỏi thêm về chủ đề này:**• (Ví dụ: "Em thấy display: grid và display: flex rất giống nhau ở việc sắp xếp. Vậy khi nào  
thì em nên dùng Flexbox, khi nào thì nên dùng Grid?")\*  
**PHIẾU HỌC TẬP CSS [14.2] - GRID: SẮP XẾP ITEMS (CON)  
Video đã xem (Link): https://www.youtube.com/watch?v=yk-Z6yvDnT8&pp=ygUfLSBHUklEOiBT4bquUCBY4bq-UCBJVEVNUyAoQ09OKQ%3D%3D  
1. Kiến thức cốt lõi (Vượt ô - Spanning)***Phần mạnh nhất của Grid là cho phép 1 item (thẻ con) chiếm (span) nhiều hơn một ô (cell)  
trong lưới. Chúng ta làm điều này bằng cách chỉ định nó nên* ***bắt đầu từ đường kẻ (line) nào****và* ***kết thúc ở đường kẻ nào****.***Ghi nhớ quan trọng:**• Một lưới có **3 CỘT** sẽ có **4 ĐƯỜNG KẺ CỘT** (1, 2, 3, 4).  
• Một lưới có **2 DÒNG** sẽ có **3 ĐƯỜNG KẺ DÒNG** (1, 2, 3).  
*Sau khi xem video, hãy điền vào mục đích của các thuộc tính của thẻ* ***CON*** *(item):*

|  |  |  |
| --- | --- | --- |
| **Thuộc tính (của Con)** | **Mục đích sử dụng (Dùng để làm gì?)** | **Ví dụ cú pháp** |
| **grid-column-start** | (Tự điền: Chỉ định đường kẻ cột...) | 1 |
| **grid-column-end** | (Tự điền: Chỉ định đường kẻ cột...) | 3 |
| **grid-column** | (Tự điền: Cú pháp TẮT cho 2 thuộc tính trên) | 1 / 3 (Bắt đầu ở 1, kết thúc ở 3) |
| **grid-row-start** | (Tự điền: Chỉ định đường kẻ dòng...) | 1 |
| **grid-row-end** | (Tự điền: Chỉ định đường kẻ dòng...) | 2 |
| **grid-row** | (Tự điền: Cú pháp TẮT cho 2 thuộc tính trên) | 1 / 2 |

**Các cú pháp đặc biệt:**• grid-column: 1 / 3; có nghĩa là "chiếm 2 cột đầu tiên" (từ vạch 1 đến vạch 3).  
• grid-column: span 2; có nghĩa là "chiếm 2 cột, bắt đầu từ vị trí mặc định của nó".  
• grid-column: 1 / -1; là một mẹo hay, có nghĩa là "bắt đầu từ vạch 1 và kết thúc ở vạch  
CUỐI CÙNG" (rất hữu ích khi bạn không biết có bao nhiêu cột).  
**2. Ví dụ thực hành (Bắt buộc)***Yêu cầu: Dựng một bố cục (layout) trang web kinh điển (Header, Sidebar, Main Content,  
Footer) bằng cách sử dụng Grid. Thẻ cha đã được thiết lập sẵn, bạn chỉ cần ra lệnh cho các  
thẻ con phải nằm ở đâu.***Mã HTML của tôi (index.html):**HTML  
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<meta name="viewport" content="width=device-width, initial-scale=1.0">  
<title>Bài tập CSS 14.2</title>  
<style>  
/\* --- KHÔNG SỬA PHẦN NÀY --- \*/  
body { font-family: Arial, sans-serif; margin: 0; }  
.layout {  
display: grid;  
width: 100vw;  
height: 100vh;  
/\* Định nghĩa 2 CỘT:  
Cột 1 (sidebar) rộng 200px  
Cột 2 (main) chiếm hết phần còn lại (1fr)  
\*/  
grid-template-columns: 200px 1fr;  
/\* Định nghĩa 3 DÒNG:  
Dòng 1 (header) cao 60px  
Dòng 2 (content) chiếm hết phần còn lại (1fr)  
Dòng 3 (footer) cao 40px  
\*/  
grid-template-rows: 60px 1fr 40px;  
}  
/\* Style chung cho các ô \*/  
.layout > div {  
padding: 10px;  
color: white;  
font-size: 20px;  
}  
/\* --- KẾT THÚC PHẦN KHÔNG SỬA --- \*/  
/\* VIẾT CODE CSS CỦA BẠN VÀO ĐÂY \*/  
/\* Yêu cầu 1: Header  
- Nằm ở DÒNG 1  
- Phải CHIẾM CẢ 2 CỘT (từ vạch 1 đến vạch 3, hoặc 1 / -1)  
\*/  
.header {  
background-color: #3498db; /\* Xanh dương \*/  
}  
/\* Yêu cầu 2: Sidebar  
- Nằm ở DÒNG 2  
- Nằm ở CỘT 1  
\*/  
.sidebar {  
background-color: #2ecc71; /\* Xanh lá \*/  
}  
/\* Yêu cầu 3: Main Content  
- Nằm ở DÒNG 2  
- Nằm ở CỘT 2  
\*/  
.main-content {  
background-color: #f1c40f; /\* Vàng \*/  
}  
/\* Yêu cầu 4: Footer  
- Nằm ở DÒNG 3  
- Phải CHIẾM CẢ 2 CỘT (giống như Header)  
\*/  
.footer {  
background-color: #34495e; /\* Xám đậm \*/  
}  
</style>  
</head>  
<body>  
<div class="layout">  
<div class="header">Header</div>  
<div class="sidebar">Sidebar</div>  
<div class="main-content">Main Content</div>  
<div class="footer">Footer</div>  
</div>  
</body>  
</html>  
Kết quả (Chụp ảnh màn hình và dán vào đây):

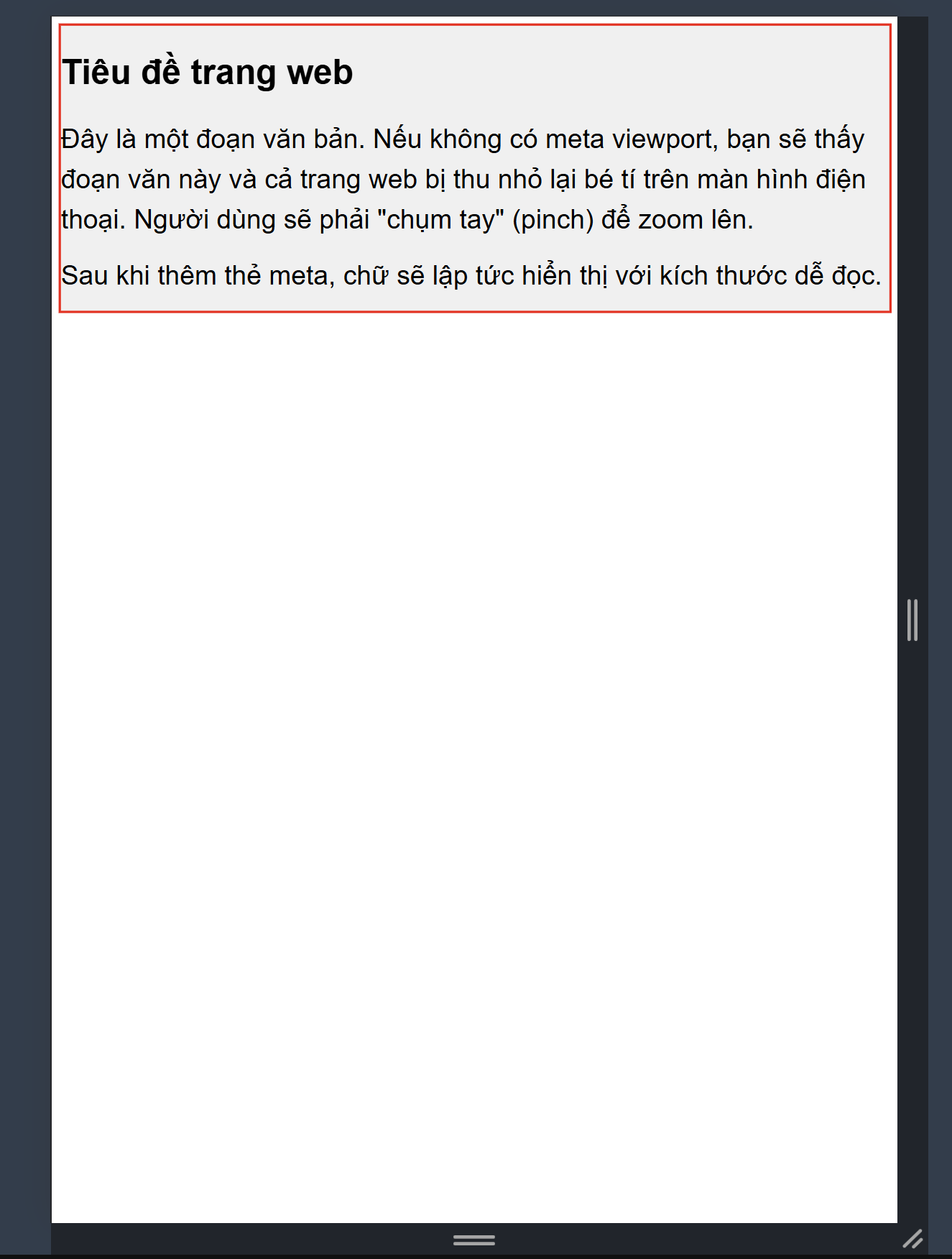
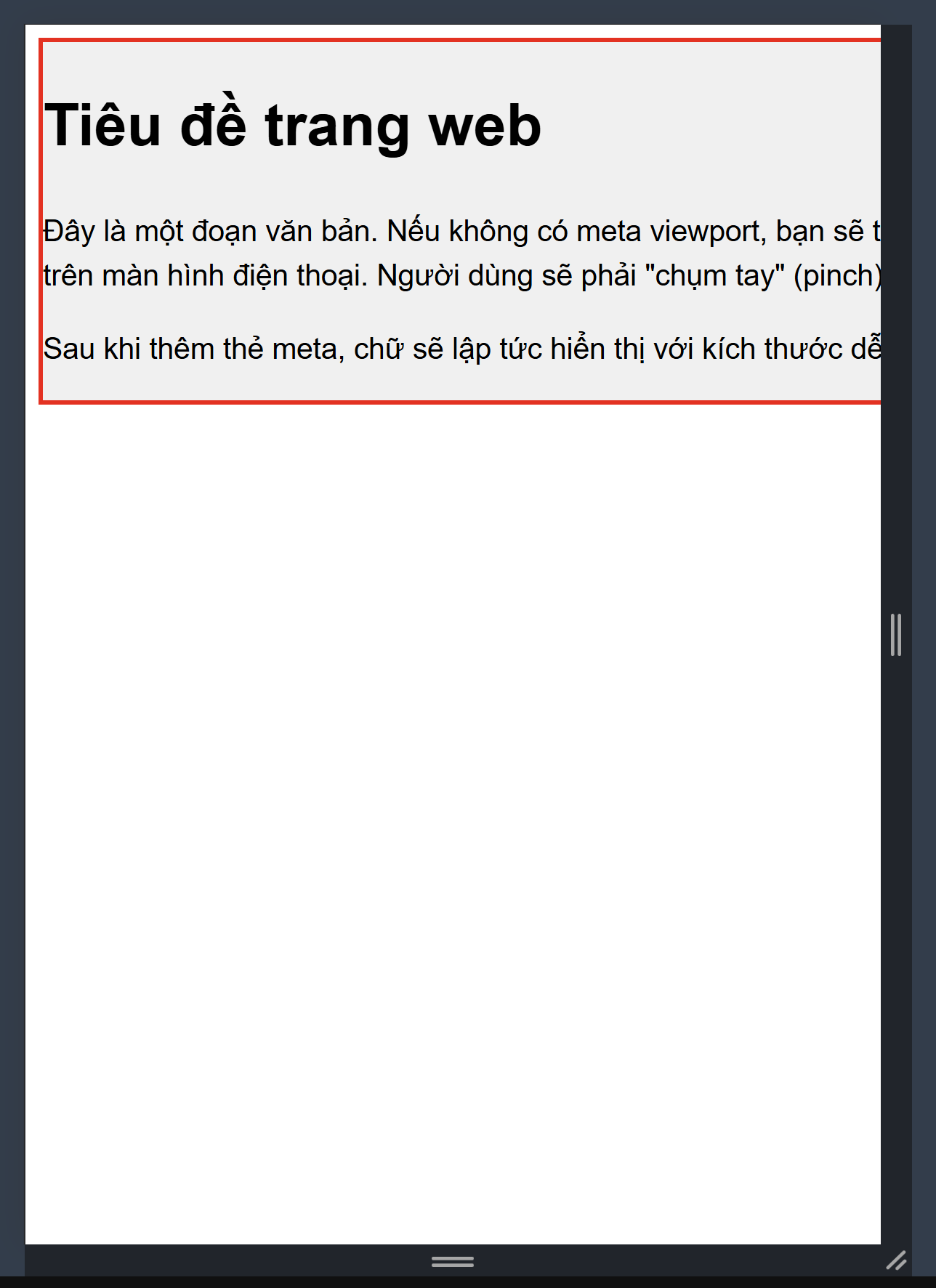


  
(Dán ảnh chụp màn hình trình duyệt của bạn vào đây. Kết quả phải là một bố cục trang web  
hoàn chỉnh:  
• *Header (xanh dương) chiếm toàn bộ chiều ngang ở trên cùng.*• *Sidebar (xanh lá) là một cột hẹp 200px ở bên trái.*• *Main Content (vàng) chiếm phần không gian chính còn lại, bên phải Sidebar.*• Footer (xám đậm) chiếm toàn bộ chiều ngang ở dưới cùng.  
)  
**3. Câu hỏi/Thắc mắc của tôi  
Một điều tôi chưa hiểu rõ hoặc muốn hỏi thêm về chủ đề này:**  
• (Ví dụ: "Em có thấy một cách viết khác là grid-template-areas. Cách đó so với cách  
dùng grid-column/row thì ưu nhược điểm là gì và khi nào nên dùng?")\*  
**PHIẾU HỌC TẬP CSS [15] - GIỚI THIỆU RESPONSIVE WEB DESIGN (RWD)  
Video đã xem (Link):  
1. Kiến thức cốt lõi (Trang web "co giãn")***Ngày nay, người dùng truy cập web bằng điện thoại, máy tính bảng, laptop, TV... Responsive  
Web Design (RWD) là kỹ thuật giúp trang web của bạn hiển thị đẹp và dễ dùng trên TẤT CẢ các  
thiết bị đó.  
Sau khi xem video, hãy tự diễn giải các khái niệm sau:***1. Responsive Web Design là gì?**• (Diễn giải bằng lời của bạn): **Responsive Web Design (RWD)** — hay **thiết kế web đáp ứng** — là **phong cách thiết kế trang web** giúp **giao diện tự động điều chỉnh** phù hợp với **kích thước và độ phân giải của thiết bị** mà người dùng đang sử dụng (máy tính, máy tính bảng, điện thoại,…).

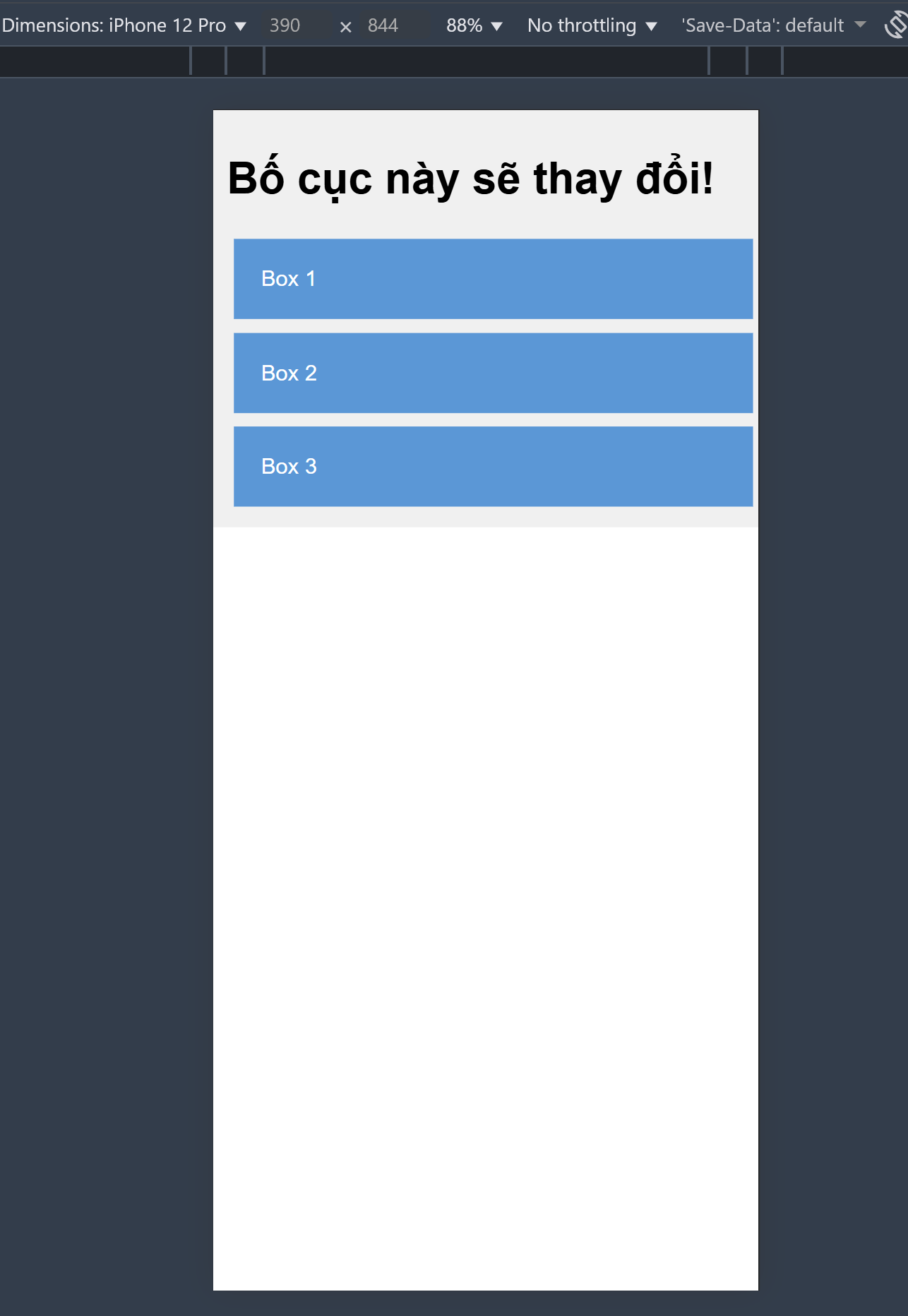
**2. Viewport là gì?**• (Diễn giải bằng lời của bạn): **Viewport** là **vùng hiển thị nội dung của trang web trên màn hình thiết bị người dùng** — nói cách khác, đó là **phần khung nhìn mà người dùng thấy được trong trình duyệt** (không tính phần nằm ngoài màn hình, phải cuộn mới thấy)  
**3. Thẻ <meta name="viewport">:**• Đây là "dòng code ma thuật" **BẮT BUỘC** phải có trong thẻ <head> của mọi trang web  
responsive.  
• Giải thích ý nghĩa của 2 thuộc tính trong thẻ: <meta name="viewport"  
content="width=device-width, initial-scale=1.0">  
o width=device-width: Yêu cầu trình duyệt: Yêu cầu trình duyệt **đặt chiều rộng viewport bằng chiều rộng thiết bị**.  
o initial-scale=1.0: Yêu cầu trình duyệt : Yêu cầu trình duyệt **hiển thị trang ở tỉ lệ thu phóng mặc định (100%)**.

**4. Triết lý "Mobile First" là gì?**• (Diễn giải bằng lời của bạn): **"Mobile First"** là triết lý **thiết kế website bắt đầu từ giao diện cho thiết bị di động trước**,  
sau đó **nâng cấp dần cho các màn hình lớn hơn** (tablet, desktop)  
**2. Ví dụ thực hành (Tác dụng "thần kỳ" của thẻ Viewport)***Yêu cầu: Quan sát sự khác biệt của 1 trang web CÓ và KHÔNG CÓ thẻ <meta  
name="viewport"> trên màn hình điện thoại.***Cách thực hiện:**1. Tạo file index.html với nội dung bên dưới (chưa có thẻ meta viewport).  
2. Mở file bằng trình duyệt (Chrome, Firefox...).  
3. Nhấn F12 (Developer Tools) và bật chế độ "Toggle device toolbar" (Biểu tượng điện  
thoại/tablet).  
4. Chọn một thiết bị (ví dụ: "iPhone 12 Pro").  
5. **Chụp ảnh màn hình 1 (Không có meta tag).**

6. Bây giờ, thêm thẻ <meta name="viewport"...> vào bên trong thẻ <head>.  
7. Tải lại trang (vẫn ở chế độ điện thoại).  
8. **Chụp ảnh màn hình 2 (Có meta tag).  
Mã HTML của tôi (index.html):**HTML  
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<title>Bài tập CSS 15</title>  
<style>  
body {  
font-size: 18px;  
line-height: 1.5;  
font-family: Arial, sans-serif;  
}  
.container {  
/\* Cố tình đặt width RỘNG để xem trên desktop \*/  
width: 960px;  
margin: 0 auto; /\* Căn giữa \*/  
border: 3px solid red;  
background-color: #f0f0f0;  
}  
</style>  
</head>  
<body>  
<div class="container">  
<h1>Tiêu đề trang web</h1>  
<p>Đây là một đoạn văn bản. Nếu không có meta viewport,  
bạn sẽ thấy đoạn văn này và cả trang web  
bị thu nhỏ lại bé tí trên màn hình điện thoại.  
Người dùng sẽ phải "chụm tay" (pinch) để zoom lên.</p>  
<p>Sau khi thêm thẻ meta, chữ sẽ lập tức  
hiển thị với kích thước dễ đọc.</p>  
</div>  
</body>  
</html>  
**KẾT QUẢ QUAN SÁT:  
Ảnh 1: KHÔNG CÓ <meta name="viewport">** *(Dán ảnh chụp màn hình giả lập điện thoại của  
bạn vào đây)*

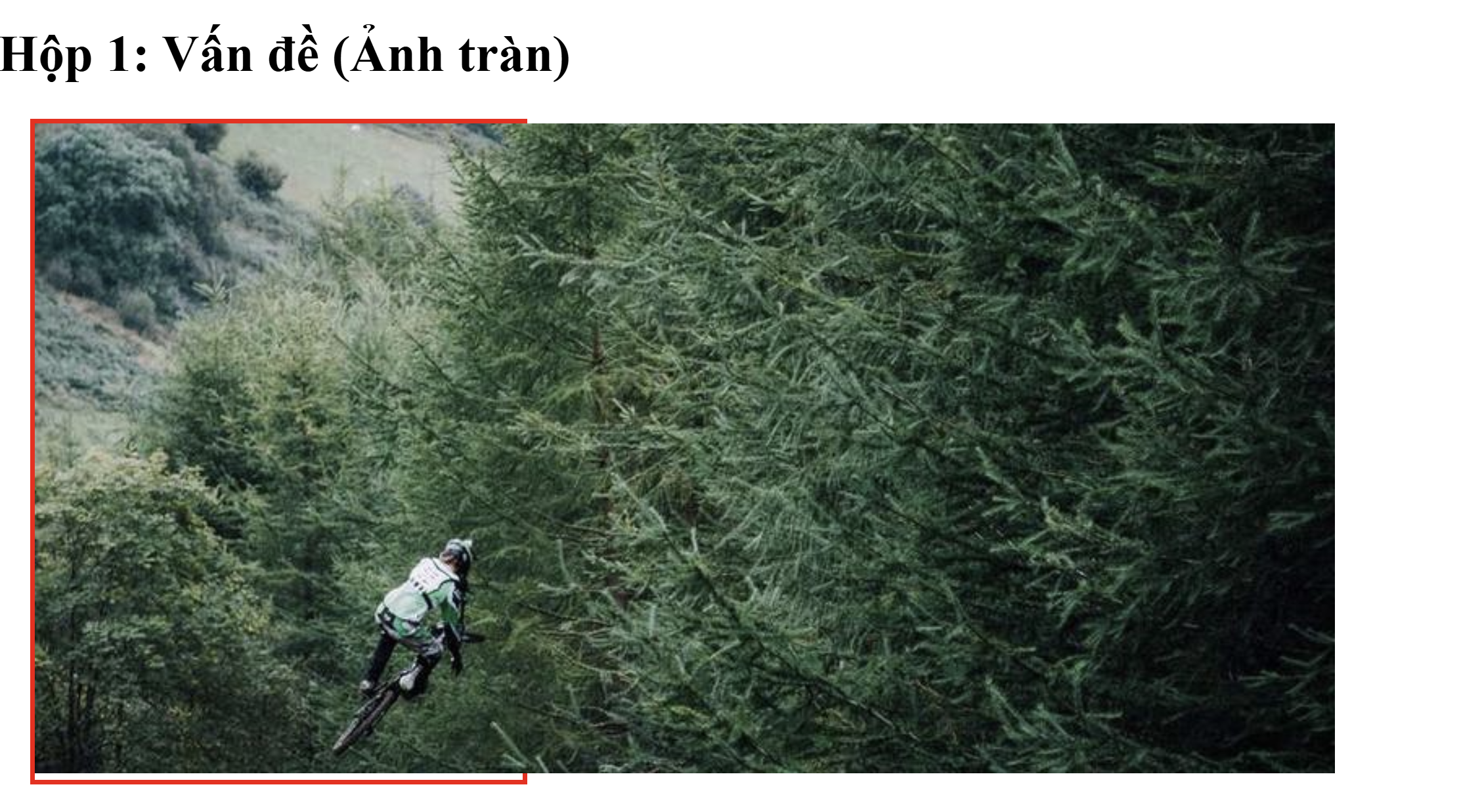
• **Nhận xét:** (Gợi ý: Trang web bị bé lại. Chữ nhỏ không thể đọc được.)  
**Ảnh 2: ĐÃ CÓ <meta name="viewport">** *(Dán ảnh chụp màn hình giả lập điện thoại của bạn  
vào đây)* 

• **Nhận xét:** (Gợi ý: Chữ đã lớn và to hơn. Tuy nhiên, cái hộp 960px màu đỏ  
đã bị tràn ra ngoài màn hình.)  
**3. Câu hỏi/Thắc mắc của tôi  
Một điều tôi chưa hiểu rõ hoặc muốn hỏi thêm về chủ đề này:**• (Ví dụ: "Sau khi thêm thẻ meta, chữ đọc được rồi, nhưng cái hộp 960px của em bị tràn  
ra ngoài màn hình. Làm thế nào để cái hộp đó cũng tự động co lại cho vừa màn hình  
điện thoại?")\*  
**PHIẾU HỌC TẬP CSS [16] - MEDIA QUERIES (Bí thuật Responsive)  
Video đã xem (Link): https://www.youtube.com/watch?v=aywAr27pkWE&pp=ygUnIE1FRElBIFFVRVJJRVMgKELDrSB0aHXhuq10IFJlc3BvbnNpdmUp  
1. Kiến thức cốt lõi (Nếu... thì...)***Media Query là cú pháp "NẾU... THÌ..." của CSS. Nó cho phép bạn áp dụng các quy tắc CSS  
chỉ khi một điều kiện nhất định (ví dụ: màn hình có chiều rộng X) được đáp ứng.  
Đây chính là chìa khóa để giải quyết vấn đề ở Bài 15.***Cú pháp cơ bản:**CSS  
/\* CSS Mặc định (cho Mobile First) \*/  
.box {  
background-color: blue;  
}  
/\* NẾU (màn hình có chiều rộng TỐI THIỂU là 768px)  
THÌ (áp dụng style bên trong)  
\*/  
@media (min-width: 768px) {  
.box {  
background-color: red;  
}  
}  
*Sau khi xem video, hãy giải thích các khái niệm sau:***1. @media:** Dùng để làm gì?

**Giúp thay đổi giao diện (CSS) cho từng loại màn hình khác nhau** — ví dụ điện thoại, tablet, laptop, desktop.  
• **2. (min-width: ...) (Triết lý Mobile First):**o (min-width: 768px) có nghĩa là: "Áp dụng style này cho các màn hình có chiều  
rộng lớn hơn hoặc bằng 768px." (lớn hơn hay nhỏ hơn?)  
• **3. (max-width: ...) (Triết lý Desktop First):**o (max-width: 600px) có nghĩa là: "Áp dụng style này cho các màn hình có chiều  
rộng nhỏ hơn hoặc bằng 600px." (lớn hơn hay nhỏ hơn?)  
• **4. Breakpoints (Điểm ngắt):**o Các giá trị như 768px (tablet), 992px (laptop) được gọi là gì? **điểm ngắt (breakpoints)”**.  
**2. Ví dụ thực hành (Giải cứu layout)***Yêu cầu: Áp dụng triết lý "Mobile First". Chúng ta sẽ style cho điện thoại trước, sau đó dùng  
min-width để "nâng cấp" layout cho màn hình lớn hơn.***Mã HTML của tôi (index.html):**  
HTML  
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<meta name="viewport" content="width=device-width, initial-scale=1.0">  
<title>Bài tập CSS 16</title>  
<style>  
/\* --- Style Chung --- \*/  
body { font-family: Arial, sans-serif; margin: 0; }  
.container {  
width: 100%; /\* Luôn vừa màn hình \*/  
max-width: 1200px; /\* Giới hạn độ rộng tối đa \*/  
margin: 0 auto;  
background-color: #f0f0f0;  
padding: 10px;  
}  
.layout {  
display: flex;  
/\* TODO Yêu cầu 1: (Mobile First)  
Mặc định, các item xếp DỌC (dùng flex-direction)  
và tự động xuống hàng (dùng flex-wrap).  
\*/  
}  
.box {  
background-color: #3498db;  
color: white;  
padding: 20px;  
margin: 5px;  
/\* TODO Yêu cầu 1 (Tiếp): (Mobile First)  
Mặc định, mỗi box chiếm 100% chiều rộng.  
(Gợi ý: dùng width: 100% hoặc flex-basis: 100%)  
\*/  
}  
/\* TODO Yêu cầu 2: (Tablet / Desktop breakpoint)  
Viết một Media Query, NẾU màn hình RỘNG TỪ 768px trở lên  
thì...  
\*/  
@media (min-width: 768px) {  
/\* ...thì đổi .layout về xếp HÀNG NGANG (row)  
\*/  
.layout {  
}  
/\* ...và đổi .box để mỗi box CHỈ chiếm 1/3  
(Gợi ý: dùng flex: 1; hoặc flex-basis: 32%;)  
\*/  
.box {  
}  
}  
</style>  
</head>  
<body>  
<div class="container">  
<h1>Bố cục này sẽ thay đổi!</h1>  
<div class="layout">  
<div class="box">Box 1</div>  
<div class="box">Box 2</div>  
<div class="box">Box 3</div>  
</div>  
</div>  
</body>  
</html>  
**KẾT QUẢ QUAN SÁT:  
Ảnh 1: Màn hình điện thoại (Dưới 768px)** *(Dán ảnh chụp màn hình giả lập điện thoại của bạn  
vào đây)* • **Nhận xét:** (Gợi ý: 3 cái hộp màu xanh xếp \_\_\_\_\_\_dọc\_\_\_\_\_ (ngang/dọc), mỗi hộp chiếm  
\_\_\_100%\_\_\_\_\_\_\_\_ chiều rộng.)  
**Ảnh 2: Màn hình Desktop (Trên 768px)** *(Dán ảnh chụp màn hình trình duyệt ở chế độ desktop  
của bạn vào đây)* • **Nhận xét:** (Gợi ý: 3 cái hộp màu xanh xếp \_\_\_\_ngang\_\_\_\_\_\_\_ (ngang/dọc), lấp đầy container.)  
**3. Câu hỏi/Thắc mắc của tôi  
Một điều tôi chưa hiểu rõ hoặc muốn hỏi thêm về chủ đề này:**• (Ví dụ: "Em thấy có 2 cách (min-width) và (max-width). Vậy khi nào thì nên dùng Mobile  
First (min-width) và khi nào nên dùng Desktop First (max-width)? Cách nào tốt hơn?")\*  
**PHIẾU HỌC TẬP CSS [17] - HÌNH ẢNH & VIDEO ĐÁP ỨNG (FLUID)  
Video đã xem (Link):  
1. Vấn đề (The Problem)***Ở các bài trước, chúng ta đã làm cho các "cái hộp" (layout) co giãn. Nhưng nếu bạn đặt một  
tấm ảnh có kích thước cố định (ví dụ width="800px") vào một cái hộp chỉ rộng 300px (trên điện  
thoại), tấm ảnh sẽ* ***bị tràn ra ngoài****, phá vỡ toàn bộ giao diện.***2. Kiến thức cốt lõi (Giải pháp "Chất lỏng")***Để giải quyết vấn đề này, chúng ta cần biến hình ảnh từ "cứng" (fixed) thành "lỏng" (fluid), tức  
là nó tự co giãn theo vật chứa nó.*Quy tắc Vàng (The Magic Rule):  
Bạn gần như SẼ LUÔN LUÔN thêm đoạn code này vào file CSS của mình cho mọi dự án:  
CSS  
img, video, iframe {  
max-width: 100%;  
height: auto;  
}  
*Sau khi xem video, hãy giải thích ý nghĩa của 2 thuộc tính này:*

|  |  |
| --- | --- |
| **Thuộc tính** | **Mục đích sử dụng (Giải thích bằng lời của bạn)** |
| **max-width: 100%;** | (Tự điền: Nghĩa là chiều rộng của ảnh CÙNG LẮM là...) |
| **height: auto;** | (Tự điền: Dùng để giữ đúng... của ảnh khi chiều rộng thay đổi) |

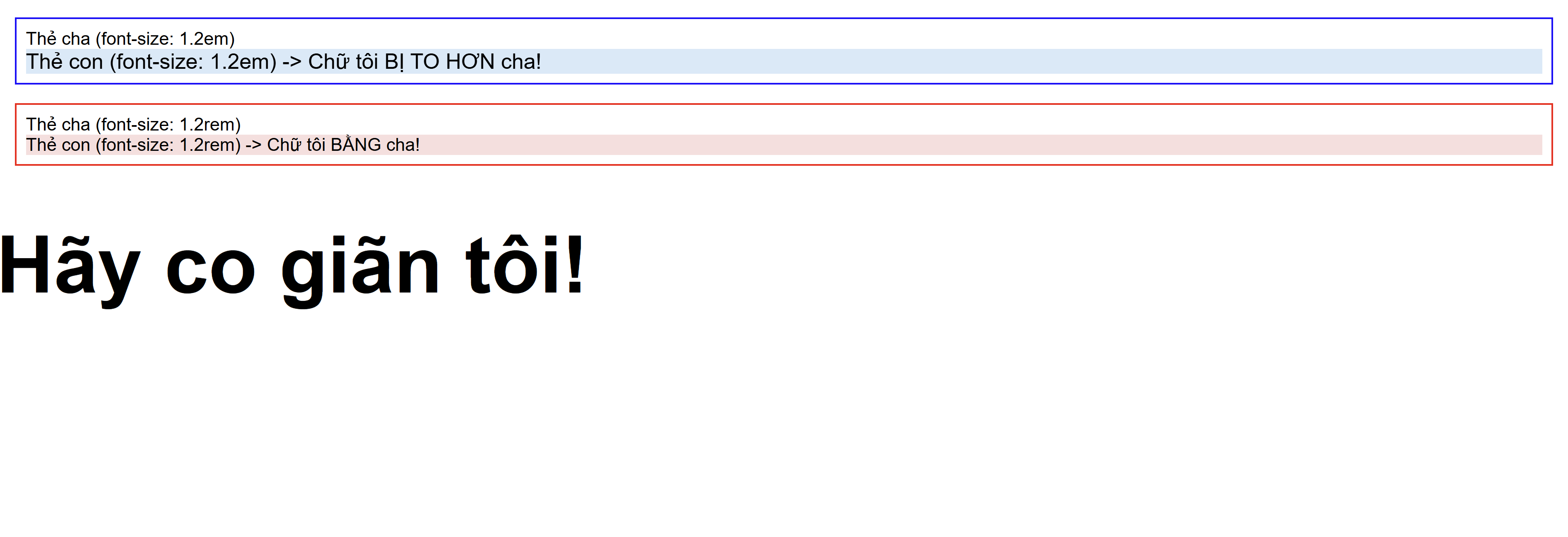
Một thuộc tính nâng cao: object-fit  
Khi bạn BẮT BUỘC cái hộp phải có kích thước cố định (ví dụ: width: 200px, height: 200px),  
nhưng ảnh của bạn lại là hình chữ nhật, ảnh sẽ bị méo. object-fit dùng để xử lý việc này.  
• object-fit: cover; (Phổ biến nhất): \_\_\_\_\_\_\_\_\_\_\_ (Giữ tỷ lệ, lấp đầy hộp, chấp nhận bị xén)  
• object-fit: contain; (Ít dùng hơn): \_\_\_\_\_\_\_\_\_\_\_ (Giữ tỷ lệ, vừa khít trong hộp, chấp nhận  
bị hở)  
**3. Ví dụ thực hành (Bắt buộc)***Yêu cầu: Tạo 2 cái hộp. Hộp 1 cho thấy vấn đề (ảnh bị tràn). Hộp 2 cho thấy giải pháp (ảnh co  
lại vừa vặn).***Mã HTML của tôi (index.html):**HTML  
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<meta name="viewport" content="width=device-width, initial-scale=1.0">  
<title>Bài tập CSS 17</title>  
<style>  
/\* Style chung cho 2 cái hộp \*/  
.container {  
width: 300px; /\* Một cái hộp RẤT NHỎ \*/  
border: 3px solid red;  
margin: 20px;  
}  
/\* KHÔNG làm gì với ảnh này \*/  
.img-problem {  
/\* Mặc định, nó sẽ giữ kích thước gốc \*/  
}  
/\* TODO: Yêu cầu  
Áp dụng "Quy tắc Vàng" cho ảnh này  
để nó tự co lại VỪA KHÍT 300px của thẻ cha.  
\*/  
.img-responsive {  
}  
</style>  
</head>  
<body>  
<h1>Hộp 1: Vấn đề (Ảnh tràn)</h1>  
<div class="container">  
<img class="img-problem"  
src="https://picsum.photos/800/400"  
alt="Ảnh bị lỗi vỡ layout">  
</div>  
<h1>Hộp 2: Giải pháp (Ảnh đáp ứng)</h1>  
<div class="container">  
<img class="img-responsive"  
src="https://picsum.photos/800/400"  
alt="Ảnh đã responsive">  
</div>  
</body>  
</html>  
**KẾT QUẢ QUAN SÁT:**Ảnh 1: Vấn đề (.img-problem)  
(Dán ảnh chụp màn hình Hộp 1 của bạn vào đây)

  
• **Nhận xét:** (Gợi ý: Tấm ảnh 800px \_\_\_bị tràn \_\_\_\_\_\_\_\_ ra khỏi cái hộp màu đỏ 300px.)  
Ảnh 2: Giải pháp (.img-responsive)  
(Dán ảnh chụp màn hình Hộp 2 của bạn vào đây)

  
• **Nhận xét:** (Gợi ý: Tấm ảnh 800px đã tự \_\_\_\_\_co lại\_\_\_\_\_\_ để nằm \_\_\_vừa khít\_\_\_\_\_\_\_\_ bên trong cái  
hộp màu đỏ 300px.)  
**3. Câu hỏi/Thắc mắc của tôi  
Một điều tôi chưa hiểu rõ hoặc muốn hỏi thêm về chủ đề này:**• (Ví dụ: "Sự khác biệt thực sự giữa width: 100%; và max-width: 100%; là gì? Em dùng  
width: 100% cho ảnh có được không và tại sao max-width lại tốt hơn?")\*  
**PHIẾU HỌC TẬP CSS [18] - CÁC ĐƠN VỊ TƯƠNG ĐỐI (REM, EM, VW, VH)  
Họ và tên:  
Lớp:  
Video đã xem (Link):  
1. Kiến thức cốt lõi (So sánh các đơn vị)***Cho đến nay, chúng ta chủ yếu dùng px (pixels). Đây là đơn vị* ***tuyệt đối*** *(cố định). Để thiết kế  
responsive và linh hoạt, chúng ta cần các đơn vị* ***tương đối****.  
Sau khi xem video, hãy hoàn thành bảng so sánh các đơn vị này:*

|  |  |  |
| --- | --- | --- |
| **Đơn vị** | **Tên đầy đủ** | **Hệ quy chiếu (Nó tính toán dựa trên cái gì?)** |
| **px** | Pixel | (Tự điền: Đơn vị cố định, 1 pixel trên màn hình) |
| **%** | Percent | (Tự điền: Kích thước của thẻ CHA trực tiếp) |
| **em** | "M" | (Tự điền: font-size của thẻ CHA trực tiếp) |
| **rem** | **Root** "M" | (Tự điền: font-size của thẻ GỐC (là thẻ <html>)) |
| **vw** | Viewport Width | (Tự điền: 1% của chiều RỘNG màn hình/viewport) |
| **vh** | Viewport Height | (Tự điền: 1% của chiều CAO màn hình/viewport) |

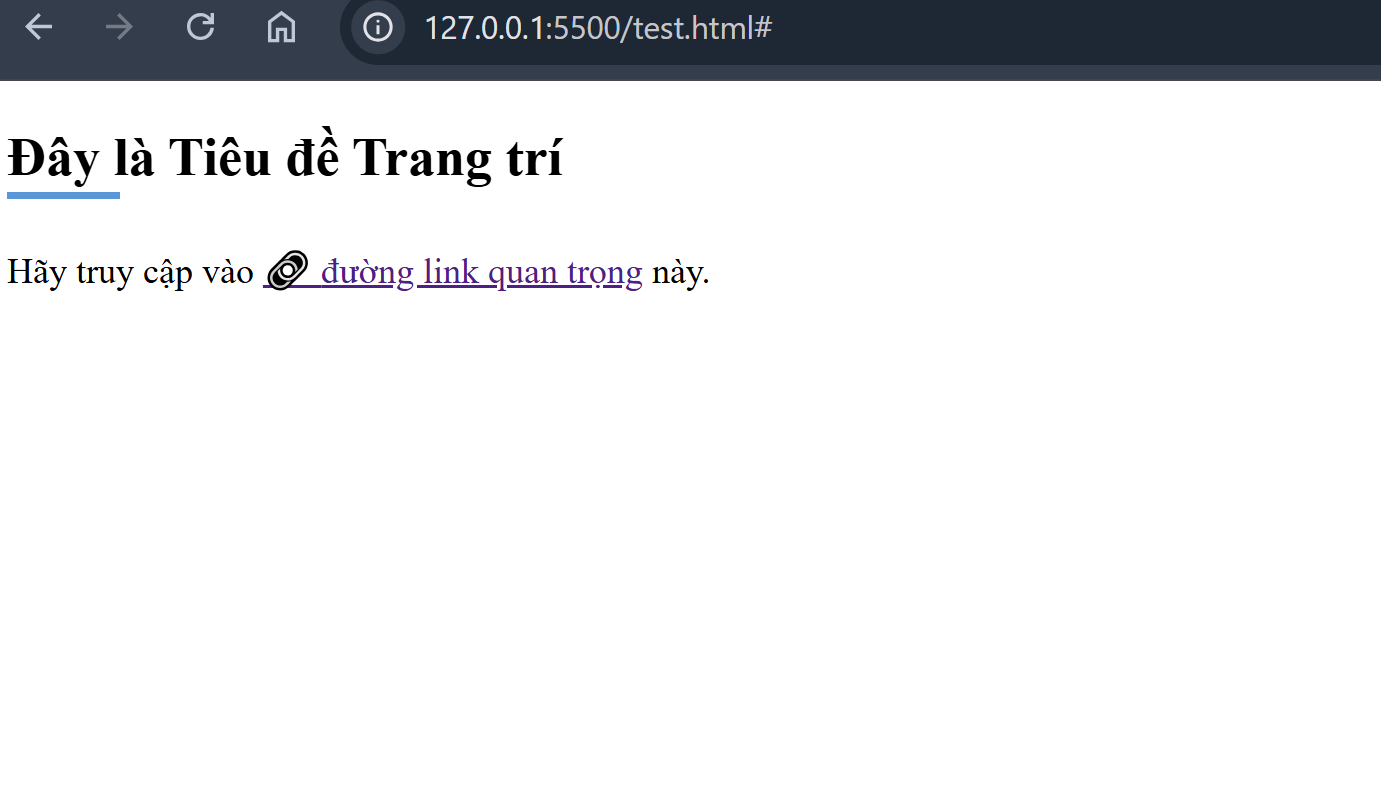
**Câu hỏi bắt buộc quan trọng:**1. Vấn đề "lồng nhau" (compounding) là gì? Nó xảy ra với em hay rem?  
2. Tại sao rem thường được coi là lựa chọn tốt nhất cho font-size trong các dự án  
responsive?  
**2. Ví dụ thực hành (Hiểm họa của em và sức mạnh của rem)***Yêu cầu: Tạo file HTML/CSS bên dưới, chạy và quan sát sự khác biệt CỰC LỚN giữa em và rem  
khi chúng lồng vào nhau.***Mã HTML của tôi (index.html):**HTML  
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<meta name="viewport" content="width=device-width, initial-scale=1.0">  
<title>Bài tập CSS 18</title>  
<style>  
/\* VIẾT CODE CSS CỦA BẠN VÀO ĐÂY \*/  
/\* Yêu cầu 1: Đặt font-size GỐC (root) là 16px \*/  
html {  
font-size: 16px;  
}  
body { font-family: Arial, sans-serif; }  
/\* Yêu cầu 2: Vấn đề của 'em'  
Mỗi thẻ con sẽ LỚN DẦN LÊN  
(vì nó = 1.2 lần font-size của cha nó)  
\*/  
.container-em {  
/\* 1.2 \* 16px = 19.2px \*/  
font-size: 1.2em;  
border: 2px solid blue;  
padding: 10px;  
margin: 20px;  
}  
.container-em .child {  
/\* 1.2 \* 19.2px = 23.04px \*/  
font-size: 1.2em;  
background-color: #d6eaf8;  
}  
/\* Yêu cầu 3: Giải pháp 'rem'  
Mọi thẻ con sẽ BẰNG NHAU  
(vì nó luôn = 1.2 lần font-size GỐC (16px))  
\*/  
.container-rem {  
/\* 1.2 \* 16px = 19.2px \*/  
font-size: 1.2rem;  
border: 2px solid red;  
padding: 10px;  
margin: 20px;  
}  
.container-rem .child {  
/\* 1.2 \* 16px = 19.2px \*/  
font-size: 1.2rem;  
background-color: #fadedE;  
}  
/\* Yêu cầu 4: Đơn vị 'vw'  
Đặt cỡ chữ bằng 5% chiều rộng màn hình.  
Hãy thử co giãn cửa sổ trình duyệt!  
\*/  
.title-vw {  
font-size: 5vw;  
}  
</style>  
</head>  
<body>  
<div class="container-em">  
Thẻ cha (font-size: 1.2em)  
<div class="child">  
Thẻ con (font-size: 1.2em) -> Chữ tôi BỊ TO HƠN cha!  
</div>  
</div>  
<div class="container-rem">  
Thẻ cha (font-size: 1.2rem)  
<div class="child">  
Thẻ con (font-size: 1.2rem) -> Chữ tôi BẰNG cha!  
</div>  
</div>  
<h1 class="title-vw">  
Hãy co giãn tôi!  
</h1>  
</body>  
</html>  
**KẾT QUẢ QUAN SÁT:**1. So sánh em và rem (Chụp ảnh màn hình 2 hộp):  
(Dán ảnh chụp màn hình Hộp 1 (xanh) và Hộp 2 (đỏ) vào đây)

  
• **Nhận xét (Bắt buộc):** Chữ "Thẻ con" ở hộp màu xanh (em) và hộp màu đỏ (rem) có kích  
thước như thế nào so với chữ "Thẻ cha" của chúng? Tại sao lại có sự khác biệt đó?  
• Ở hộp **màu xanh (em)**, chữ “Thẻ con” **to hơn** chữ “Thẻ cha” vì nó nhân theo kích thước của cha.

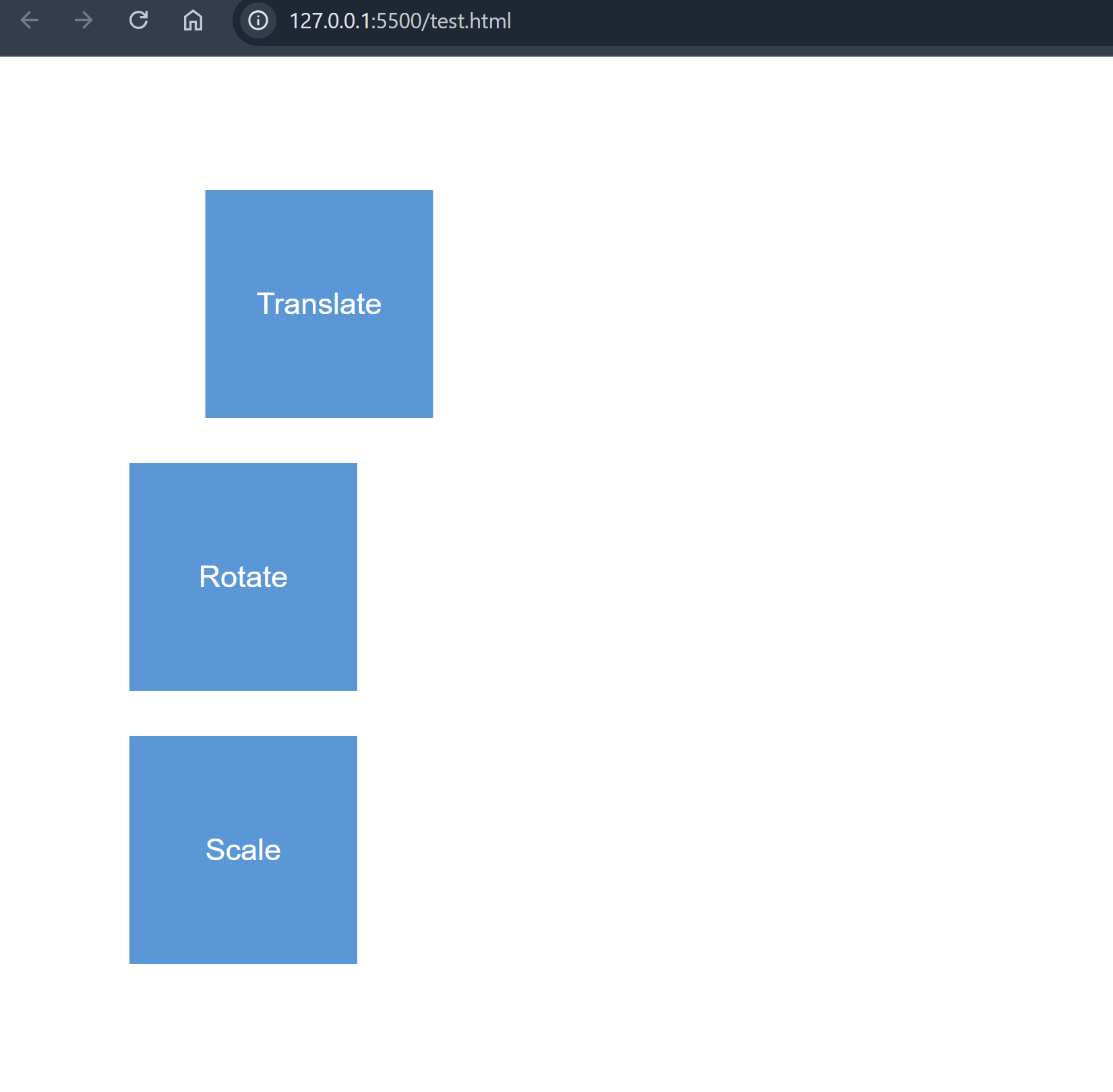
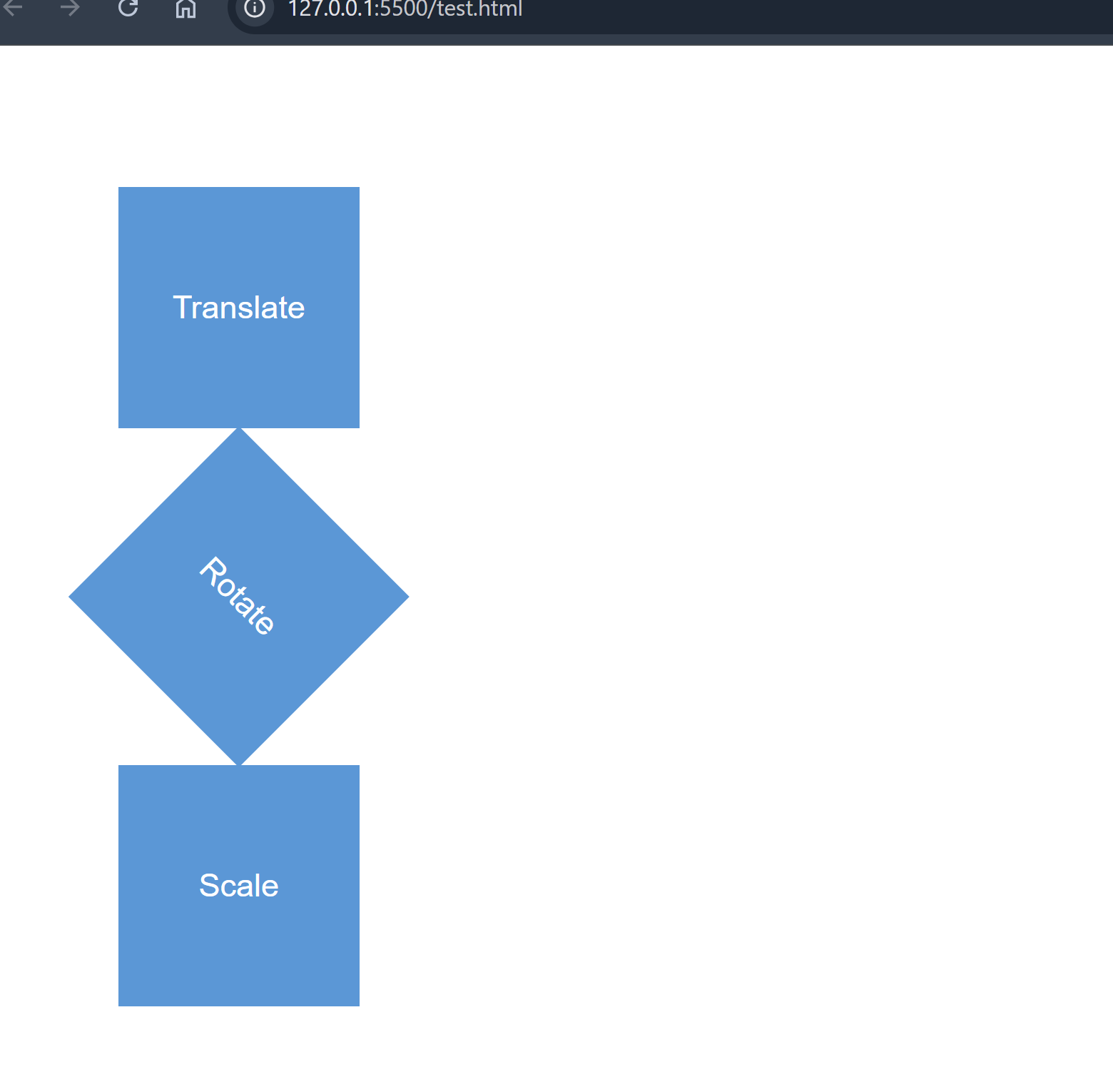
2. Thí nghiệm với vw (Video hoặc GIF):  
(Hãy quay một video ngắn hoặc mô tả lại hành động của bạn)  
• **Mô tả (Bắt buộc):** Khi bạn co/giãn cửa sổ trình duyệt, điều gì xảy ra với tiêu đề "Hãy co  
giãn tôi!"?  
•Ở hộp **màu đỏ (rem)**, chữ “Thẻ con” **bằng với** chữ “Thẻ cha” vì cả hai đều dựa theo font-size gốc (16px).  
**3. Câu hỏi/Thắc mắc của tôi  
Một điều tôi chưa hiểu rõ hoặc muốn hỏi thêm về chủ đề này:**• (Ví dụ: "Vậy tóm lại, khi nào em nên dùng rem, khi nào dùng % và khi nào dùng px? Em  
có nên bỏ hoàn toàn px không?")\*  
**PHIẾU HỌC TẬP CSS [19] - GIẢ-PHẦN TỬ (::before & ::after)  
Video đã xem (Link):** **https://www.youtube.com/watch?v=HjwRMg6eGF4&pp=ygUmR0nhuqItUEjhuqZOIFThu6wgKDo6YmVmb3JlICYgOjphZnRlcik%3D  
1. Kiến thức cốt lõi (Tạo phần tử "ảo")***Giả-phần tử (Pseudo-elements) là các "phần tử ảo" mà bạn có thể tạo và style bằng CSS.  
Chúng không tồn tại thực sự trong file HTML, nhưng hiển thị trên trang web. Hai cái phổ biến  
nhất là ::before và ::after.  
Sau khi xem video, hãy hoàn thành bảng sau:*

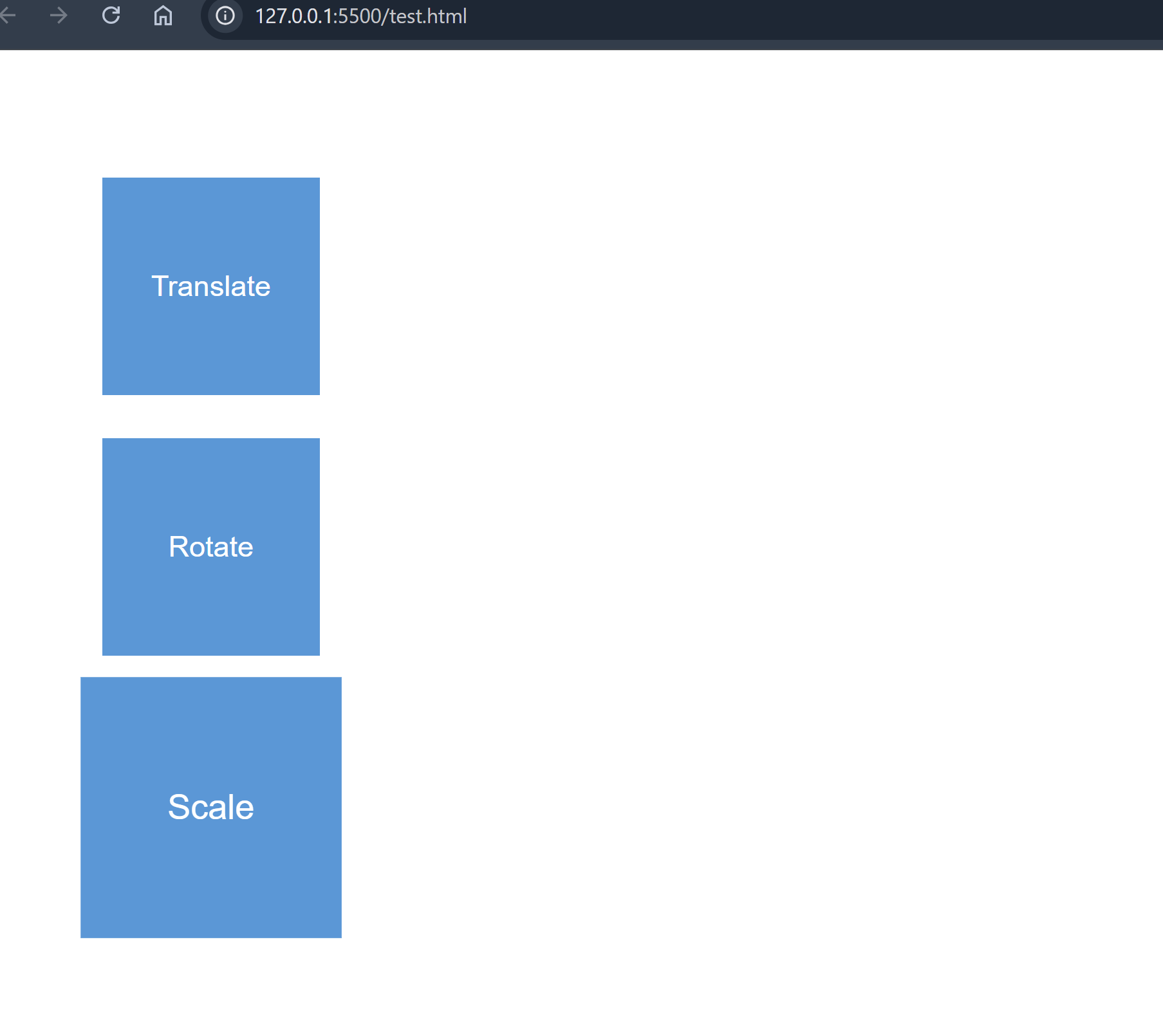
|  |  |  |
| --- | --- | --- |
| Tên Giả-phần tử | Cú pháp (Ví dụ) | Ý nghĩa (Nó chèn nội dung vào đâu?) |
| **::before** | p::before { ... } | (Tự điền: Chèn nội dung ảo VÀO BÊN TRONG thẻ p,... |
| **::after** | p::after { ... } | (Tự điền: Chèn nội dung ảo VÀO BÊN TRONG thẻ p,... |

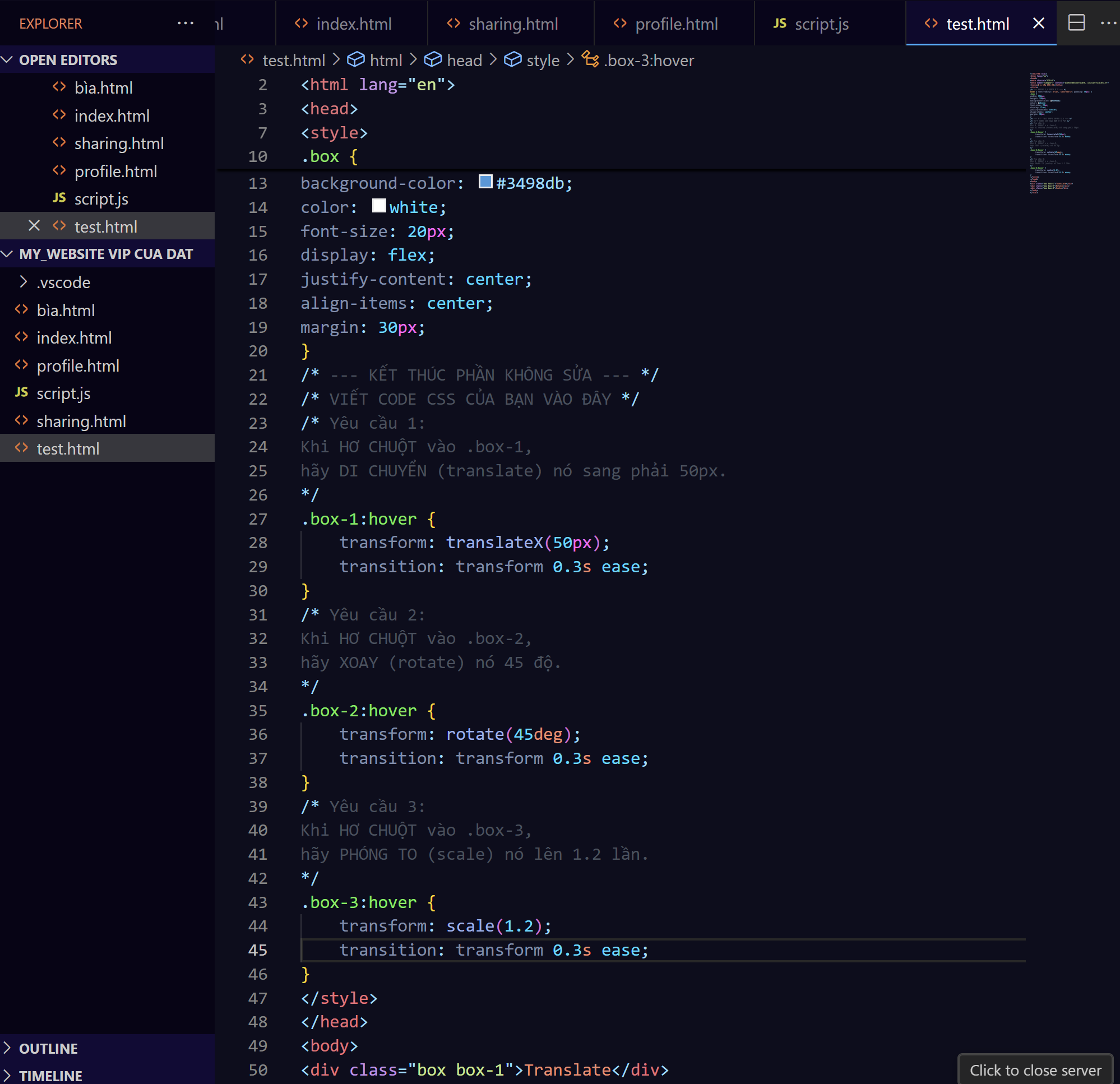
Xuất sang Trang tính  
**Câu hỏi BẮT BUỘC SỐNG CÒN:**1. Thuộc tính nào là **TUYỆT ĐỐI BẮT BUỘC** phải có (kể cả khi giá trị của nó là rỗng) để  
::before hoặc ::after có thể hiển thị?  
o **Trả lời:** Thuộc tính \_content\_\_  
2. Mặc định, các phần tử "ảo" này có display là gì? (inline hay block?)  
o **Trả lời:** \_\_inline\_\_\_  
**2. Ví dụ thực hành (Bắt buộc)***Yêu cầu: Viết CSS trong thẻ <style> để thêm các yếu tố trang trí cho tiêu đề và liên kết mà****không*** *được sửa file HTML.***Mã HTML của tôi (index.html):**HTML  
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<meta name="viewport" content="width=device-width, initial-scale=1.0">  
<title>Bài tập CSS 19</title>  
<style>  
/\* VIẾT CODE CSS CỦA BẠN VÀO ĐÂY \*/  
/\* Yêu cầu 1:  
Dùng '::before' để thêm một icon "liên kết" (🔗)  
vào TRƯỚC nội dung của thẻ .link  
(Nhớ thêm một khoảng trống sau icon)  
\*/  
.link::before {  
content: "🔗 ";  
}  
/\* Yêu cầu 2:  
Dùng '::after' để tạo một đường gạch chân trang trí  
cho tiêu đề .title  
\*/  
.title {  
/\* Cần 'relative' để định vị đường gạch (nâng cao) \*/  
position: relative;  
padding-bottom: 5px; /\* Tạo không gian cho gạch \*/  
}  
.title::after {  
/\* 1. BẮT BUỘC phải có \*/  
content: "";  
/\* 2. Tạo hình dạng cho "đường gạch" \*/  
display: block; /\* Biến nó thành block để có W/H \*/  
width: 50px;  
height: 3px;  
background-color: #3498db; /\* Màu xanh \*/  
/\* 3. Định vị nó (nâng cao) \*/  
position: absolute;  
bottom: 0;  
left: 0;  
}  
</style>  
</head>  
<body>  
<h2 class="title">Đây là Tiêu đề Trang trí</h2>  
<p>  
Hãy truy cập vào  
<a class="link" href="#">đường link quan trọng</a>  
này.  
</p>  
</body>  
</html>  
**Kết quả (Chụp ảnh màn hình và dán vào đây):** *(Dán ảnh chụp màn hình trình duyệt của bạn  
vào đây. Kết quả phải là:*

• *Tiêu đề <h2> có một đường gạch chân ngắn màu xanh bên dưới.*• *Chữ "đường link quan trọng" có một icon nằm ngay trước nó. )*  
**3. Câu hỏi/Thắc mắc của tôi  
Một điều tôi chưa hiểu rõ hoặc muốn hỏi thêm về chủ đề này:**• (Ví dụ: "Em thấy trên mạng có người viết :before (1 dấu hai chấm), có người viết ::before  
(2 dấu hai chấm). Chúng khác gì nhau và em nên dùng cái nào?")\*  
**PHIẾU HỌC TẬP CSS [20] - BIẾN ĐỔI (TRANSFORMS)  
Video đã xem (Link):  
1. Kiến thức cốt lõi (Thay đổi hình dạng)***Thuộc tính transform cho phép bạn di chuyển, xoay, phóng to/thu nhỏ một phần tử mà* ***không  
ảnh hưởng đến vị trí*** *của các phần tử khác xung quanh nó.  
Sau khi xem video, hãy điền vào mục đích của các hàm transform phổ biến:*

|  |  |  |
| --- | --- | --- |
| **Hàm transform** | **Mục đích sử dụng (Dùng để làm gì?)** | **Ví dụ cú pháp** |
| **translate()** | (Tự điền: Dùng để DI CHUYỂN...) | translate(50px, 10px) |
| **rotate()** | (Tự điền: Dùng để XOAY...) | rotate(45deg) |
| **scale()** | (Tự điền: Dùng để PHÓNG TO/THU NHỎ...) | scale(1.5) (Phóng to 1.5 lần) |
| **skew()** | (Tự điền: Dùng để LÀM MÉO/NGHIÊNG...) | skew(10deg) |

**Câu hỏi bắt buộc:**• Đâu là sự khác biệt lớn nhất giữa việc dùng transform: translate(50px); và margin-left:  
50px; để di chuyển một cái hộp? (Gợi ý: Hãy nghĩ về các phần tử xung quanh nó).  
**2. Ví dụ thực hành (Bắt buộc)***Yêu cầu: Viết CSS để khi người dùng* ***DI CHUỘT (hover)*** *vào từng hộp, nó sẽ thực hiện một  
"biến đổi" khác nhau.***Mã HTML của tôi (index.html):**HTML  
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<meta name="viewport" content="width=device-width, initial-scale=1.0">  
<title>Bài tập CSS 20</title>  
<style>  
/\* --- KHÔNG SỬA PHẦN NÀY --- \*/  
body { font-family: Arial, sans-serif; padding: 50px; }  
.box {  
width: 150px;  
height: 150px;  
background-color: #3498db;  
color: white;  
font-size: 20px;  
display: flex;  
justify-content: center;  
align-items: center;  
margin: 30px;  
}  
/\* --- KẾT THÚC PHẦN KHÔNG SỬA --- \*/  
/\* VIẾT CODE CSS CỦA BẠN VÀO ĐÂY \*/  
/\* Yêu cầu 1:  
Khi HƠ CHUỘT vào .box-1,  
hãy DI CHUYỂN (translate) nó sang phải 50px.  
\*/  
.box-1:hover {  
}  
/\* Yêu cầu 2:  
Khi HƠ CHUỘT vào .box-2,  
hãy XOAY (rotate) nó 45 độ.  
\*/  
.box-2:hover {  
}  
/\* Yêu cầu 3:  
Khi HƠ CHUỘT vào .box-3,  
hãy PHÓNG TO (scale) nó lên 1.2 lần.  
\*/  
.box-3:hover {  
}  
</style>  
</head>  
<body>  
<div class="box box-1">Translate</div>  
<div class="box box-2">Rotate</div>  
<div class="box box-3">Scale</div>  
</body>  
</html>  
Kết quả (Chụp 3 ảnh màn hình):  
Vì đây là hiệu ứng :hover, bạn hãy chụp 3 ảnh: 1 ảnh khi đang hơ chuột vào Hộp 1, 1 ảnh khi  
đang hơ chuột vào Hộp 2, và 1 ảnh khi đang hơ chuột vào Hộp 3.  
Ảnh 1: Hơ chuột vào Hộp 1  
  
Ảnh 2: Hơ chuột vào Hộp 2  
  
Ảnh 3: Hơ chuột vào Hộp 3



  
**3. Câu hỏi/Thắc mắc của tôi  
Một điều tôi chưa hiểu rõ hoặc muốn hỏi thêm về chủ đề này:**• (Ví dụ: "Ở Bài 20, cái hộp biến đổi rất nhanh. Làm thế nào để em làm cho nó XOAY hoặc  
DI CHUYỂN một cách MƯỢT MÀ (smooth) trong 0.5 giây thay vì giật một cái?")\*  
**PHIẾU HỌC TẬP CSS [21] - CHUYỂN TIẾP MƯỢT MÀ (TRANSITIONS)  
Video đã xem (Link):  
1. Kiến thức cốt lõi (Làm mượt hiệu ứng)**Ở Bài 20, khi bạn :hover vào một phần tử, sự thay đổi (ví dụ transform) xảy ra ngay lập tức,  
trông rất "giật".  
Thuộc tính transition (Chuyển tiếp) cho phép bạn báo với trình duyệt: "Này, nếu thuộc tính này  
(ví dụ background-color hay transform) thay đổi, hãy thay đổi nó một cách từ từ và mượt mà  
trong một khoảng thời gian nhất định."  
*Sau khi xem video, hãy điền vào mục đích của các thuộc tính con của transition:*

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Mục đích sử dụng (Dùng để làm gì?)** | **Ví dụ** |
| **transition-property** | (Tự điền: Chỉ định thuộc tính nào...) | transform, background-color, all |
| **transition-duration** | (Tự điền: Chỉ định thời gian...) | 0.5s (nửa giây), 300ms |
| **transition-timing function** | (Tự điền: Chỉ định "đường cong tốc độ"...) | linear, ease, ease-in-out |
| **transition-delay** | (Tự điền: Chỉ định thời gian TRÌ HOÃN...) | 1s (chờ 1 giây rồi mới chạy) |

Cú pháp viết tắt (Shorthand):  
Chúng ta gần như luôn dùng cú pháp tắt, viết tất cả trên 1 dòng. Thứ tự phổ biến là:  
transition: [property] [duration] [timing-function] [delay];  
Ví dụ: transition: all 0.3s ease;  
**2. Ví dụ thực hành (Bắt buộc)***Yêu cầu: Lấy lại BÀI TẬP CỦA BÀI 20, và thêm* ***một dòng*** *transition duy nhất để làm mượt tất cả  
các hiệu ứng.***Mã HTML của tôi (index.html):**HTML  
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<meta name="viewport" content="width=device-width, initial-scale=1.0">  
<title>Bài tập CSS 21</title>  
<style>  
/\* --- KHÔNG SỬA PHẦN NÀY (STYLE TỪ BÀI 20) --- \*/  
body { font-family: Arial, sans-serif; padding: 50px; }  
.box {  
width: 150px;  
height: 150px;  
background-color: #3498db;  
color: white;  
font-size: 20px;  
display: flex;  
justify-content: center;  
align-items: center;  
margin: 30px;  
}  
/\* Hiệu ứng HOVER (Từ Bài 20) \*/  
.box-1:hover {  
transform: translateX(50px);  
background-color: #e74c3c; /\* Thêm đổi màu \*/  
}  
.box-2:hover {  
transform: rotate(45deg);  
}  
.box-3:hover {  
transform: scale(1.2);  
}  
/\* --- KẾT THÚC PHẦN KHÔNG SỬA --- \*/  
/\* VIẾT CODE CSS CỦA BẠN VÀO ĐÂY \*/  
/\* Yêu cầu:  
Thêm thuộc tính 'transition' vào class .box  
(thêm vào trạng thái gốc, KHÔNG PHẢI :hover)  
- Áp dụng cho TẤT CẢ thuộc tính ('all')  
- Thời gian chạy là 0.4 giây (0.4s)  
- Hiệu ứng mượt là 'ease-in-out'  
\*/  
.box {  
}  
</style>  
</head>  
<body>  
<div class="box box-1">Translate</div>  
<div class="box box-2">Rotate</div>  
<div class="box box-3">Scale</div>  
</body>  
</html>  
Kết quả (Video hoặc GIF):  
Hiệu ứng tĩnh không thể hiện được bài này. Hãy quay một video ngắn (hoặc ảnh GIF) ghi lại  
cảnh bạn hơ chuột lần lượt qua 3 cái hộp.  
**Mô tả kết quả (Bắt buộc):**• Điều gì xảy ra khi bạn hơ chuột vào Hộp 1? Nó còn "giật" như Bài 20 không?  
• \_\_\_\_không\_\_\_\_\_\_\_  
• Điều gì xảy ra khi bạn BỎ CHUỘT ra khỏi hộp? (Nó có mượt mà quay về không?)  
• \_\_\_\_\_có\_\_\_\_\_\_  
**3. Câu hỏi/Thắc mắc của tôi  
Một điều tôi chưa hiểu rõ hoặc muốn hỏi thêm về chủ đề này:**• (Ví dụ: "Em đã làm mượt được khi hover. Nhưng nếu em muốn một cái hộp tự xoay  
vòng vòng liên tục mà không cần hơ chuột thì làm thế nào?")\*  
**PHIẾU HỌC TẬP CSS [22] - HOẠT ẢNH VỚI @keyframes  
Video đã xem (Link):  
1. Kiến thức cốt lõi (Tạo chuyển động phức tạp)***transition (Bài 21) chỉ là chuyển đổi 2 trạng thái (A -> B). animation cho phép bạn tạo ra một  
chuỗi chuyển động phức tạp (A -> B -> C -> D...) và lặp lại chúng mà không cần tương tác của  
người dùng.  
Tạo một animation luôn gồm 2 BƯỚC:***Bước 1: Định nghĩa "kịch bản" bằng @keyframes**• @keyframes là nơi bạn định nghĩa các "chặng" của hoạt ảnh.  
• Cú pháp:  
CSS  
@keyframes ten-hoat-anh-cua-ban {  
from { /\* Trạng thái bắt đầu \*/  
opacity: 0;  
}  
to { /\* Trạng thái kết thúc \*/  
opacity: 1;  
}  
}  
/\* Hoặc dùng % cho nhiều chặng \*/  
@keyframes ten-phuc-tap {

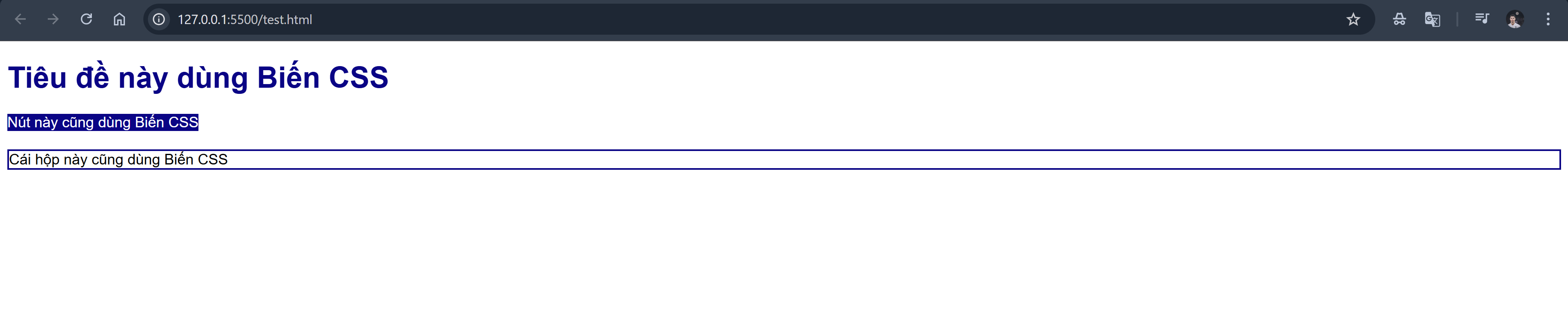
|  |  |
| --- | --- |
| 0%  50% | { left: 0px; } { left: 200px; } |

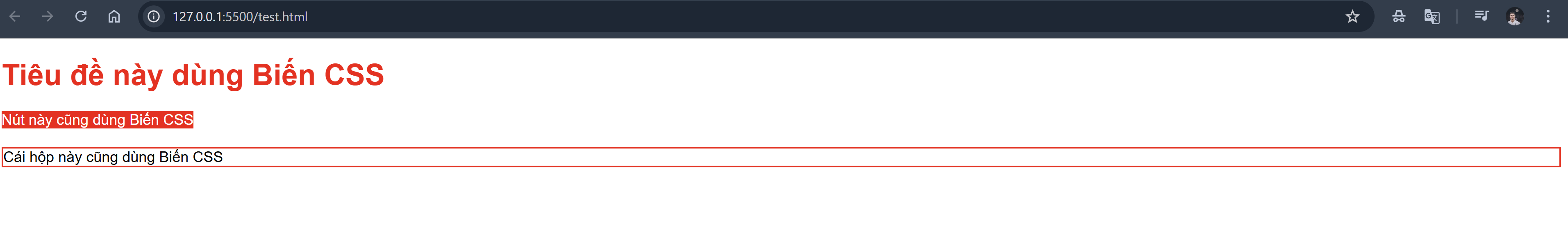
100% { left: 0px; }  
}  
**Bước 2: Áp dụng "kịch bản" đó cho phần tử**• Dùng thuộc tính animation để "gọi" @keyframes và ra lệnh cho nó chạy.

|  |  |  |
| --- | --- | --- |
| **Thuộc tính animation** | **Mục đích sử dụng (Dùng để làm gì?)** | **Ví dụ** |
| **animation-name** | (Tự điền: Tên của @keyframes...) | ten-hoat-anh-cua-ban |
| **animation-duration** | (Tự điền: Thời gian để hoàn thành 1 vòng...) | 2s |
| **animation-iteration count** | (Tự điền: Số lần lặp lại...) | 3, **infinite** (vô tận) |

|  |  |  |
| --- | --- | --- |
| **Thuộc tính animation** | **Mục đích sử dụng (Dùng để làm gì?)** | **Ví dụ** |
| **animation-direction** | (Tự điền: Hướng chạy...) | normal, **alternate** (chạy ngược lại) |
| **animation-timing function** | (Tự điền: "Đường cong tốc độ"...) | ease, **linear** (đều đặn) |

Cú pháp viết tắt (Shorthand):  
animation: [name] [duration] [timing-function] [iteration-count] [direction];  
Ví dụ: animation: spin 2s linear infinite;  
**2. Ví dụ thực hành (Bắt buộc)***Yêu cầu: Tạo 2 hoạt ảnh chạy liên tục mà không cần hơ chuột:*1. Một hộp tự xoay 360 độ (.box-spin).  
2. Một hộp tự "phập phồng" (lớn lên, nhỏ lại) (.box-pulse).  
**Mã HTML của tôi (index.html):**HTML  
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<meta name="viewport" content="width=device-width, initial-scale=1.0">  
<title>Bài tập CSS 22</title>  
<style>  
/\* --- Style chung --- \*/  
body { font-family: Arial, sans-serif; padding: 100px; }  
.box {  
width: 100px;  
height: 100px;  
background-color: #9b59b6; /\* Tím \*/  
color: white;  
display: flex;  
justify-content: center;  
align-items: center;  
margin: 50px;  
}  
/\* --- VIẾT CODE CSS CỦA BẠN VÀO ĐÂY --- \*/  
/\* Yêu cầu 1: Định nghĩa kịch bản XOAY \*/  
@keyframes spin {  
from {  
transform: rotate(0deg);  
}  
to {  
/\* TODO: Xoay đến 360 độ \*/  
}  
}  
/\* Yêu cầu 2: Định nghĩa kịch bản PHỒNG (pulse) \*/  
@keyframes pulse {  
0% {  
transform: scale(1);  
}  
50% {  
/\* TODO: Phóng to 1.2 lần \*/  
}  
100% {  
transform: scale(1);  
}  
}  
/\* Yêu cầu 3: Áp dụng kịch bản 'spin' cho .box-spin  
- Tên: spin  
- Thời gian: 2s  
- Tốc độ: linear (đều)  
- Lặp lại: infinite (vô tận)  
\*/  
.box-spin {  
}  
/\* Yêu cầu 4: Áp dụng kịch bản 'pulse' cho .box-pulse  
- Tên: pulse  
- Thời gian: 3s  
- Lặp lại: infinite (vô tận)  
\*/  
.box-pulse {  
background-color: #e74c3c; /\* Đỏ \*/  
}  
</style>  
</head>  
<body>  
<div class="box box-spin">Xoay...</div>  
<div class="box box-pulse">Phồng...</div>  
</body>  
</html>  
Kết quả (Video hoặc GIF):  
Hãy quay một video ngắn (hoặc ảnh GIF) ghi lại 2 cái hộp đang tự chuyển động.  
**Mô tả kết quả (Bắt buộc):**• Điều gì xảy ra với 2 cái hộp ngay khi tải trang (mà không cần hơ chuột)?  
• \_\_\_hộp xoay liên tục\_\_\_\_\_\_\_\_  
**3. Câu hỏi/Thắc mắc của tôi  
Một điều tôi chưa hiểu rõ hoặc muốn hỏi thêm về chủ đề này:**• (Ví dụ: "Em thấy animation có alternate (chạy qua chạy lại), transition có làm được như  
vậy không? Hay transition chỉ chạy 1 chiều?")\*  
**PHIẾU HỌC TẬP CSS [23] - BIẾN TRONG CSS (CSS VARIABLES)  
Video đã xem (Link):  
1. Kiến thức cốt lõi (Sức mạnh của "Lưu trữ")***Giả sử dự án của bạn dùng một "màu chủ đạo" (ví dụ: màu xanh) ở 50 nơi khác nhau. Khi sếp  
đổi ý muốn đổi sang "màu đỏ", bạn phải tìm và sửa 50 dòng code đó.****Biến CSS (CSS Variables)****, hay còn gọi là* ***Custom Properties****, cho phép bạn lưu giá trị đó (ví  
dụ: màu xanh) vào MỘT "biến" duy nhất. Khi cần đổi, bạn chỉ cần sửa 1 nơi duy nhất.  
Sau khi xem video, hãy điền vào 2 cú pháp quan trọng:***1. Cú pháp ĐỊNH NGHĨA Biến (Thường đặt trong :root):**• :root là bộ chọn đại diện cho thẻ <html> (thẻ gốc), giúp biến có thể được dùng ở MỌI  
NƠI.  
• Cú pháp (Tự điền):  
CSS  
:root {  
--ten-bien-cua-ban: gia-tri;  
}  
*(Ví dụ: --primary-color: #3498db;) Lưu ý: Tên biến BẮT BUỘC bắt đầu bằng \_\_\_\_\_\_\_\_\_\_\_***2. Cú pháp SỬ DỤNG Biến:**• Để gọi giá trị đã lưu, bạn dùng hàm var().  
• Cú pháp (Tự điền):  
CSS  
.box {  
background-color: var(.........);  
}  
*(Ví dụ: background-color: var(--primary-color);)***2. Ví dụ thực hành (Thay đổi giao diện trong 1 nốt nhạc)***Yêu cầu: Tạo một giao diện nhỏ dùng Biến CSS. Sau đó, thử thay đổi toàn bộ giao diện chỉ bằng  
cách sửa 2 dòng trong :root.***Mã HTML của tôi (index.html):**HTML  
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<meta name="viewport" content="width=device-width, initial-scale=1.0">  
<title>Bài tập CSS 23</title>  
<style>  
/\* VIẾT CODE CSS CỦA BẠN VÀO ĐÂY \*/  
/\* Yêu cầu 1: Định nghĩa 2 biến toàn cục  
- Biến '--main-color' có giá trị là 'darkblue'  
- Biến '--padding-size' có giá trị là '15px'  
\*/  
:root {  
}  
/\* Yêu cầu 2: Sử dụng các biến  
(KHÔNG gõ 'darkblue' hay '15px' ở dưới đây)  
\*/  
body {  
font-family: Arial, sans-serif;  
}  
h1 {  
/\* Dùng biến '--main-color' cho màu chữ \*/  
color: var(--main-color);  
}  
button {  
/\* Dùng biến '--main-color' cho màu nền \*/  
background-color: var(--main-color);  
/\* Dùng biến '--padding-size' cho padding \*/  
padding: var(--padding-size);  
color: white;  
border: none;  
font-size: 16px;  
}  
.note {  
margin-top: 20px;  
/\* Dùng biến '--padding-size' cho padding \*/  
padding: var(--padding-size);  
/\* Dùng biến '--main-color' cho viền \*/  
border: 2px solid var(--main-color);  
}  
</style>  
</head>  
<body>  
<h1>Tiêu đề này dùng Biến CSS</h1>  
<button>Nút này cũng dùng Biến CSS</button>  
<div class="note">  
Cái hộp này cũng dùng Biến CSS  
</div>  
</body>  
</html>  
**KẾT QUẢ QUAN SÁT:  
Ảnh 1: Giao diện "Màu xanh" (Lần chạy đầu)** *(Dán ảnh chụp màn hình trình duyệt của bạn  
vào đây. Kết quả là tiêu đề, nút, và viền hộp đều có màu darkblue.)*

**Thí nghiệm (Bắt buộc):**• Bây giờ, quay lại code CSS, vào phần :root và sửa 2 dòng: --main-color: darkred; --  
padding-size: 30px;  
**Ảnh 2: Giao diện "Màu đỏ" (Sau khi sửa biến)** *(Dán ảnh chụp màn hình kết quả MỚI của bạn  
vào đây.)*

• **Nhận xét (Bắt buộc):**• Sau khi sửa 2 dòng trong :root, điều gì đã xảy ra với Tiêu đề, Nút bấm và Cái hộp?  
• \_\_\_\_đổi màu chữ và tiêu đề\_\_\_\_\_\_\_  
• Kích thước padding của Nút và Hộp thay đổi như thế nào?  
• \_\_\_thay đổi lớn hơn\_\_\_\_\_\_\_\_  
**3. Câu hỏi/Thắc mắc của tôi  
Một điều tôi chưa hiểu rõ hoặc muốn hỏi thêm về chủ đề này:**• (Ví dụ: "Em có thể định nghĩa biến bên trong một class (ví dụ .container { --my-color:  
red; }) thay vì :root không? Nếu có thì nó khác gì?")\*