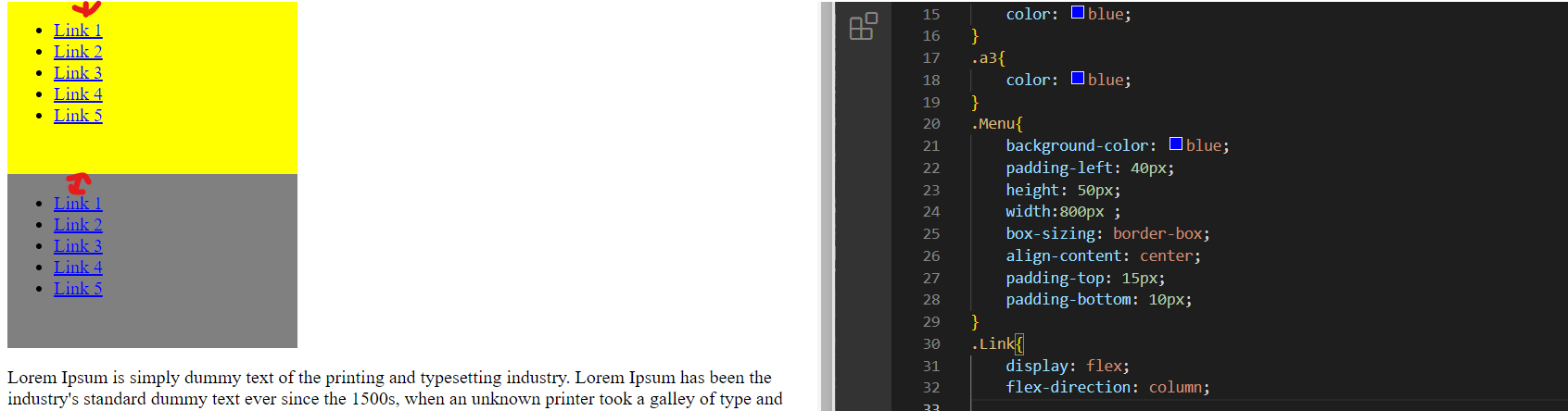
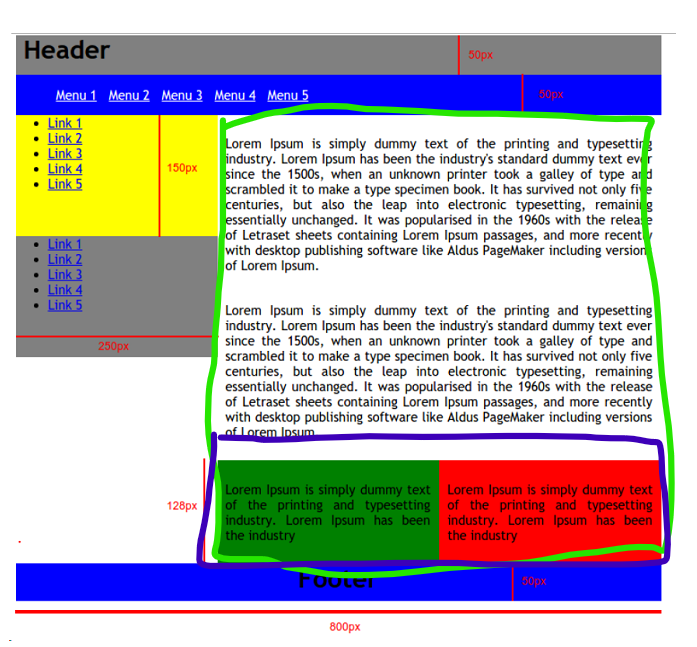


Lưu ý; Với trường hợp bên dưới đây thì nếu chúng ta muốn các đoạn text Link có màu xanh, mà chúng ta khai báo lệnh color trong class Link\_block1{} thì chương trình sẽ ko hiểu, do trong class này nội dung của nó là các thẻ ul>li\*5, do đó các dấu chấm sẽ có màu xanh.

- Muốn các đoạn text có màu xanh, thì chúng ta phải khai báo câu lệnh css này trong thẻ <a> </a>.



- Trong bài 4, Lab1 phần 2, thì khi chúng ta thiết kế css như này thì vẫn còn khoảng cách của Link1 vs lề trên, chúng ta cần phải cho nó sát lề trên như yêu cầu của thiết kế

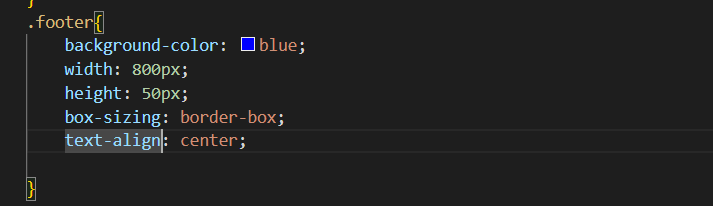


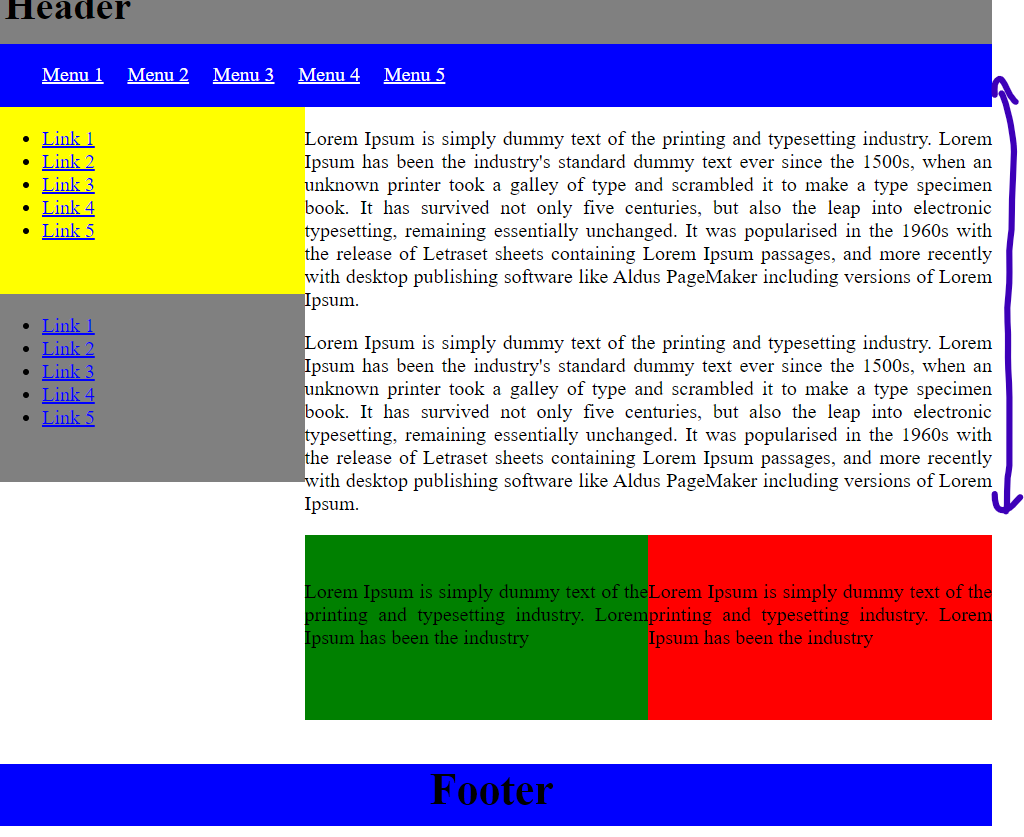
- Để chia layout cho content như trong hình thì chúng ta cần đóng gói toàn bộ phần nội dụng (hight ligh xanh lá ) thành 1 class, phần highligh tím thành 1 class, và trong class tím chia thành 2 class nhỏ để tạo background color green vs red.

- Dùng flex-box để chia layout cho các class trên

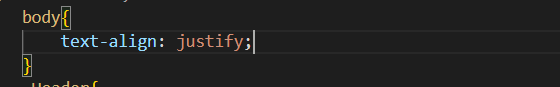


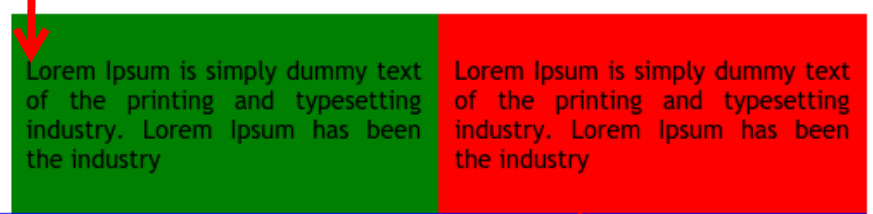
- Muốn tạo chữ Footer canh giữa như này thì chúng ta dùng text-align: center trong CSS



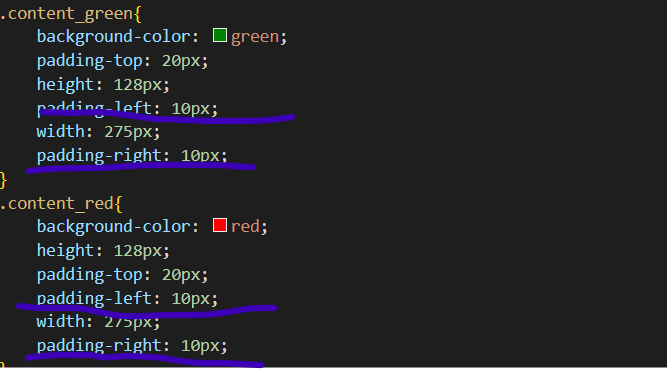


Muốn tạo canh đều cho đoạn văn bản trong thiết kế thì ngay thẻ body của html chúng ta khai báo lệnh text-align:justify trong CSS

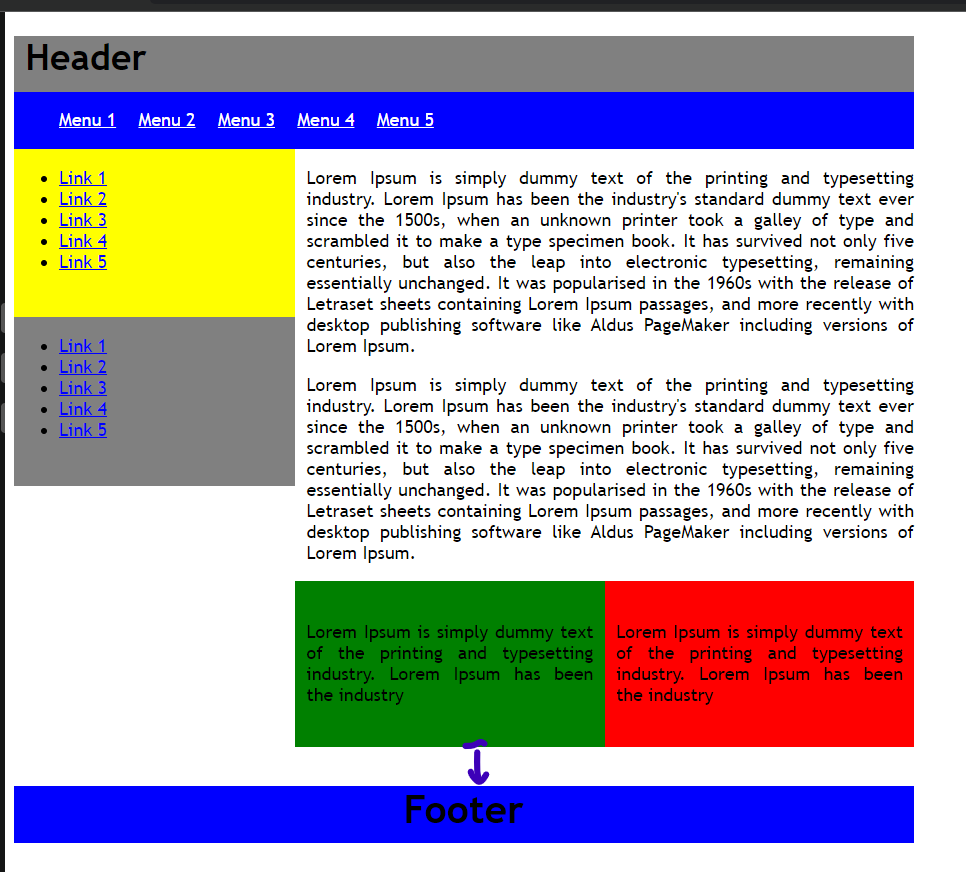




- Cần thêm padding left và padding right để cho khoảng cách lề trái và lề phải của đoạn text xanh và text đó có 1 khoảng cách như định như trong thiết kế.



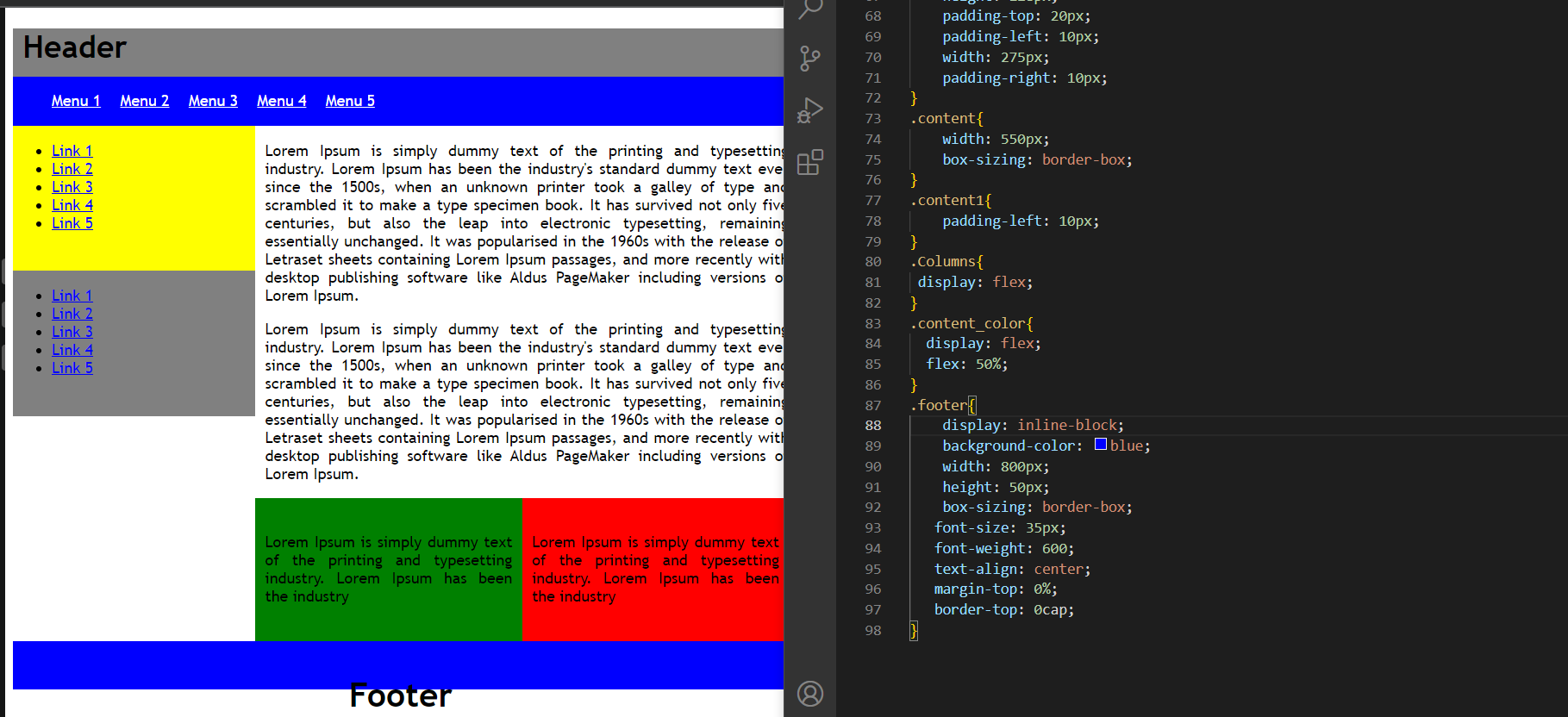
Để khoảng cách giữa footer và đoạn text bên trên khít nhau thì chúng ta xử lý như sau:



Tham khảo thêm về việc sử dụng các thuộc tính của dislay trong CSS:

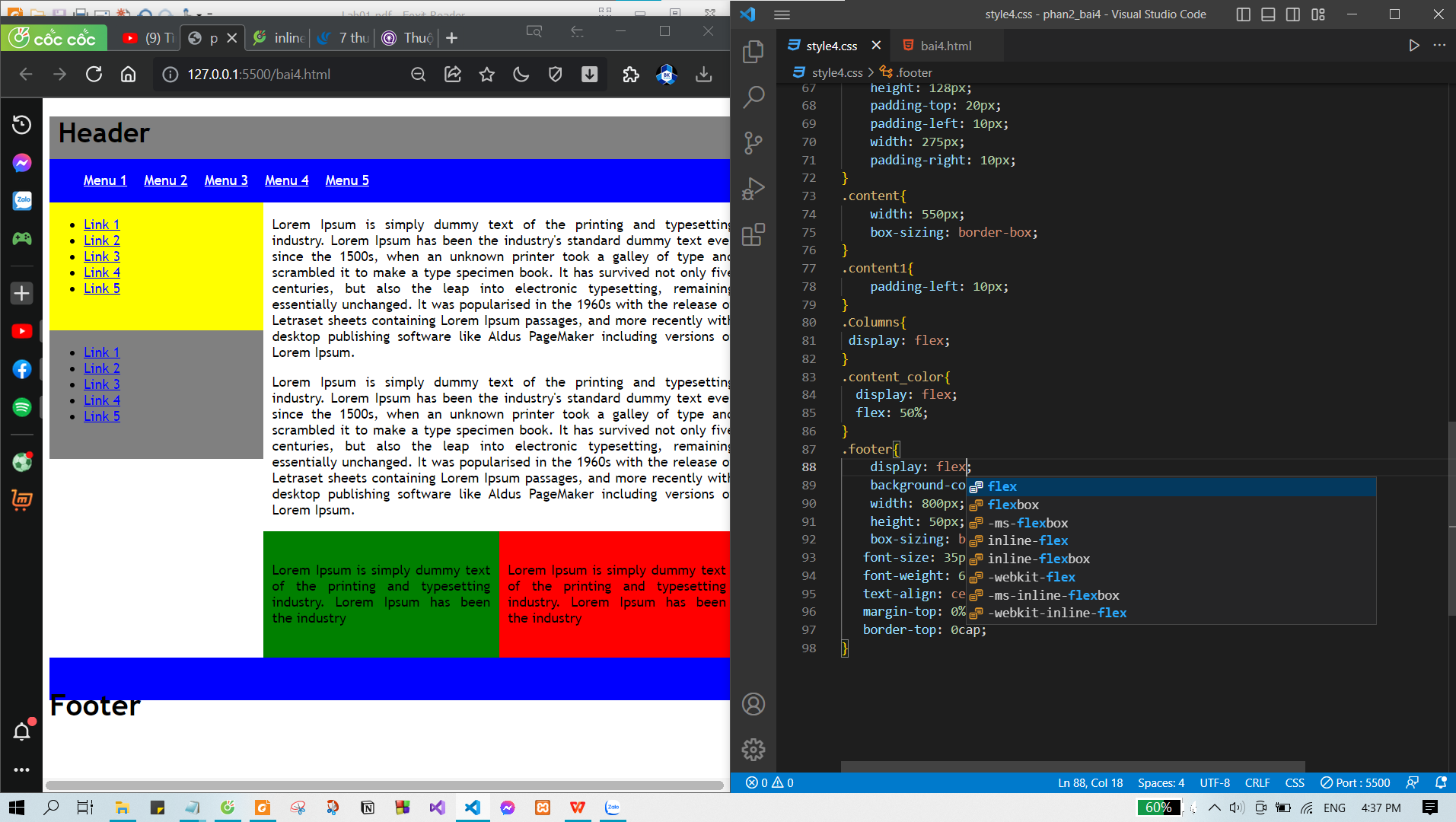
<https://tenten.vn/tin-tuc/display-css/>

<https://quantrimang.com/hoc/display-inline-block-trong-css-162940>

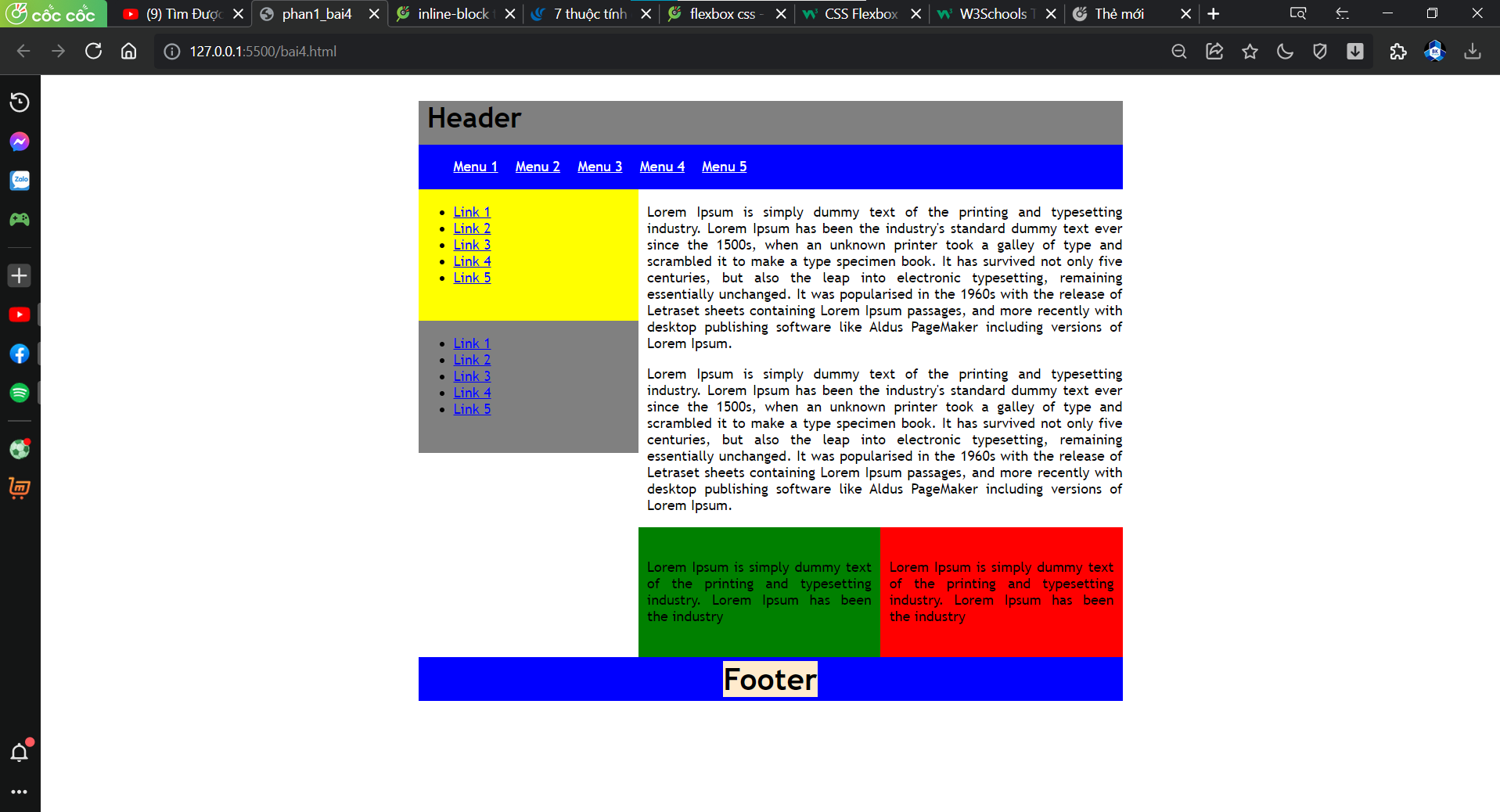


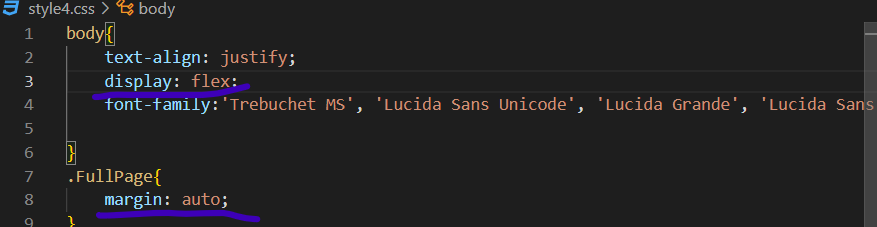
Trong trường hợp chúng ta dùng display:inline-block, thì text-align: center sẽ bị lỗi và bị nhảy như hình bên dưới đây.

Còn nếu chúng ta dùng display:inline-flex thì giao diện sẽ hiển thị như hình bên dưới đây, mặc dù chúng ta vẫn sử dụng lẹnh text-align:center để canh giữa nó ??? Khá khoai



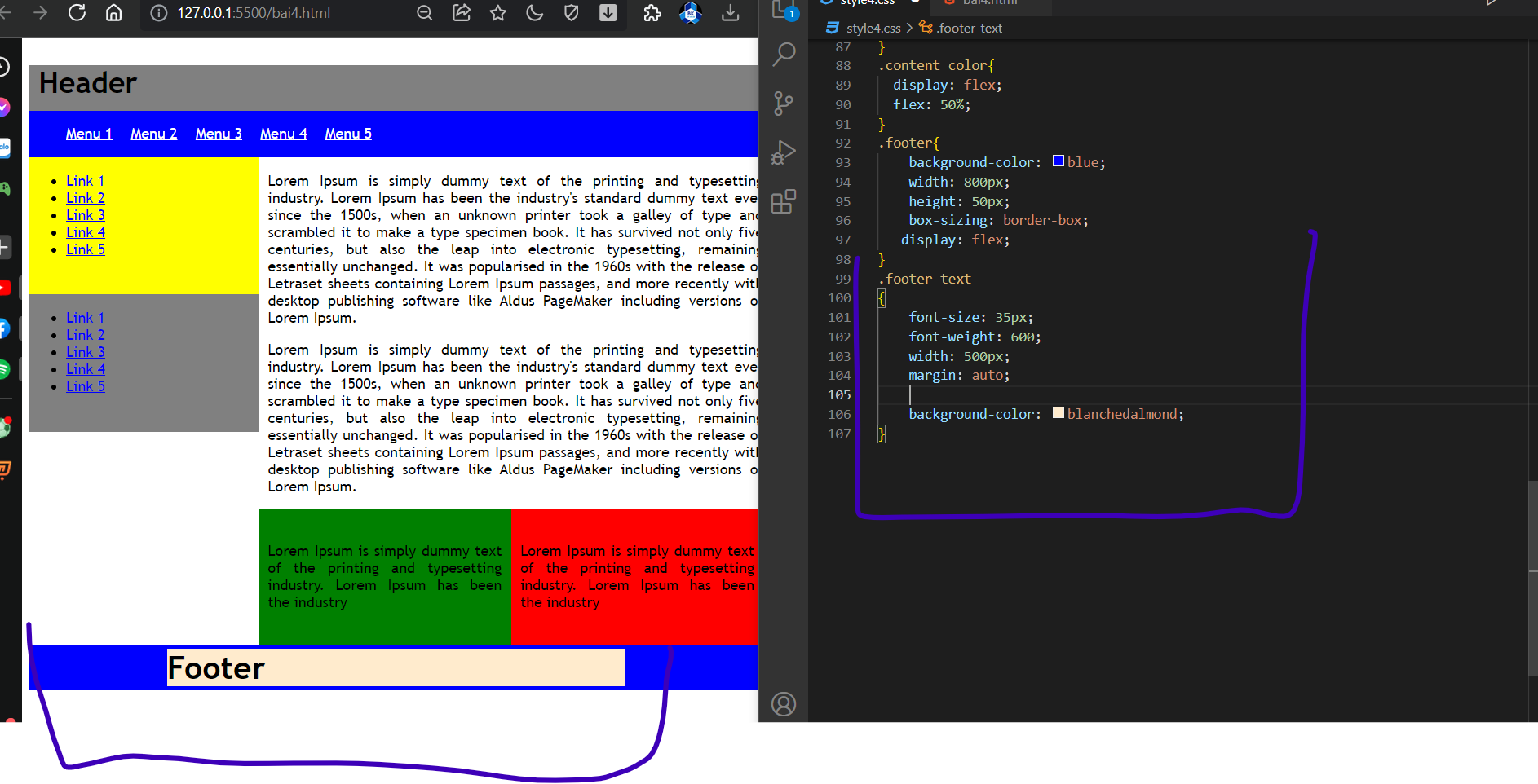
- Muốn tạo 1 giao diện canh giữa như này, thì ở thẻ <body> chúng ta khai báo display:flex, và class bao trùm lên toàn bộ content chúng ta cho margin:auto



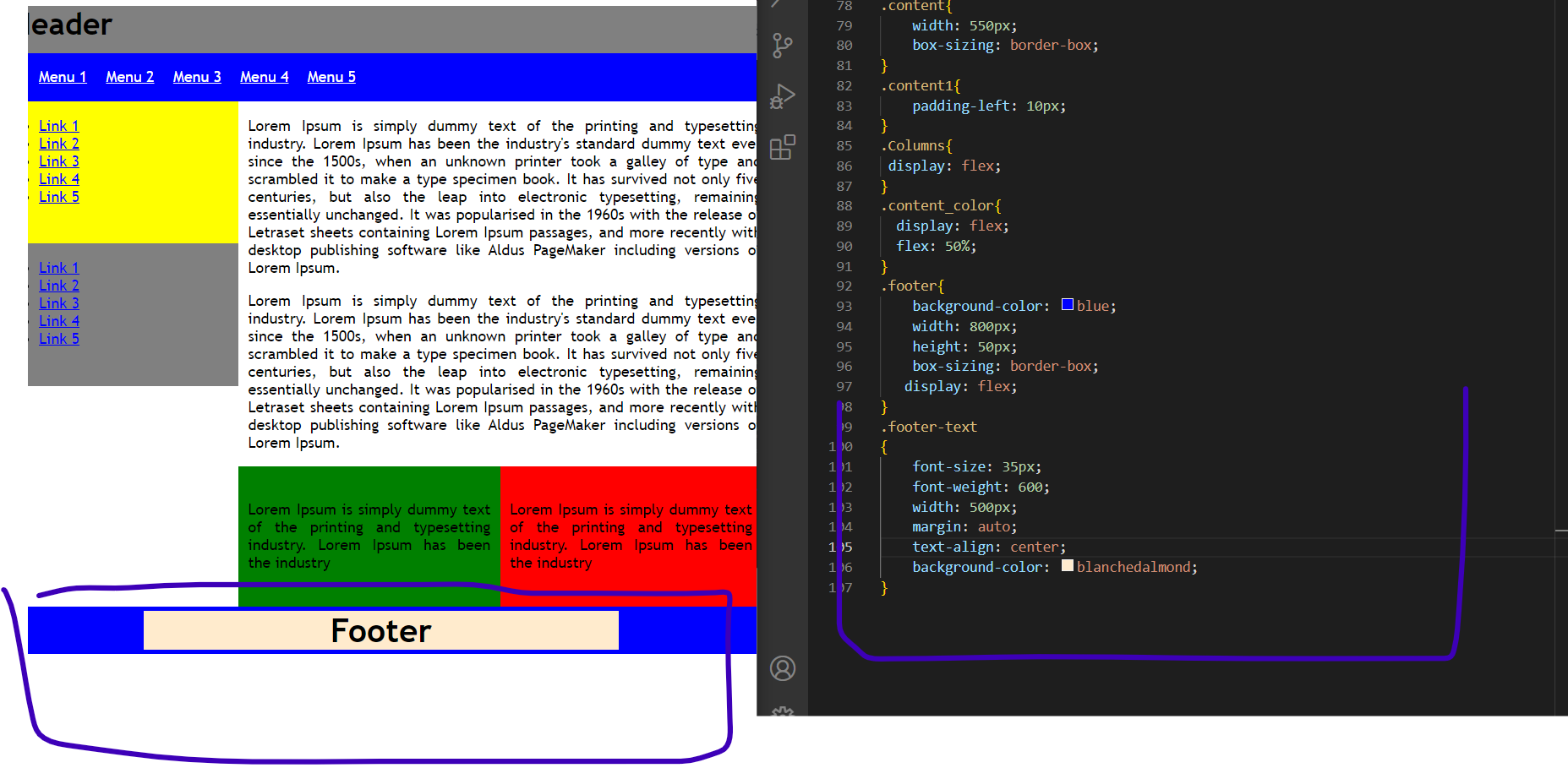


Một góc nhìn khác khi chúng ta chia layout cho giao diện như này, và sự kết hợp giữa cách chia layout này với text-align:center

Ví dụ: không có text-align:center

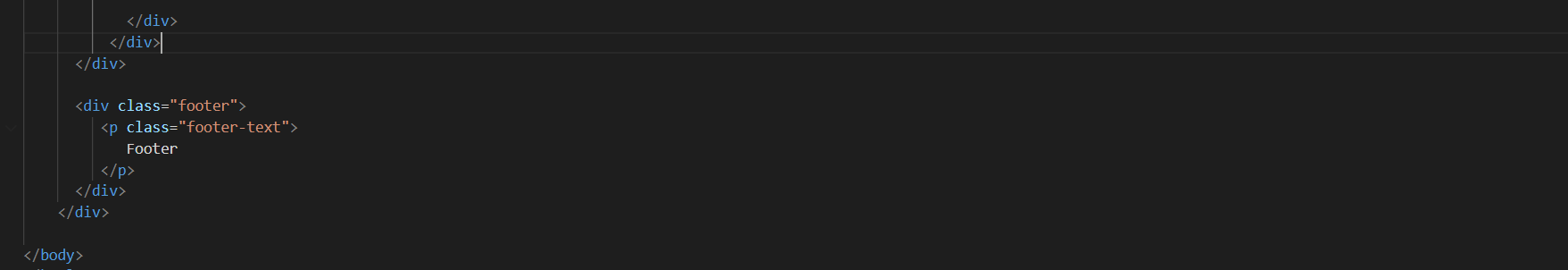


Giao diện khi có sử dụng text-align:center

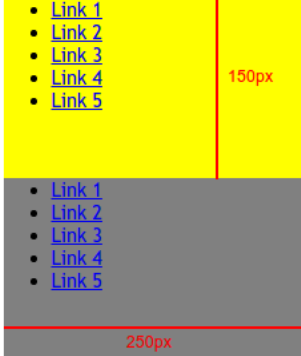


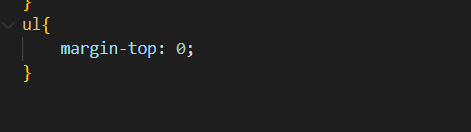
NOTE: và chúng ta sử dụng cách code này trong trường hợp chúng ta muốn canh giữa footer, nhưng cái thẻ div nó lại ko sát vế lề trên dù cho ta chỉnh cách nào cx ko dc, thì chúng ta sẽ áp dụng cách code vs ý tưởng như trên.

- Với ý tưởng này thì chúng ta sẽ chia thành 2 class như hình bên dưới cho Footer



Muốn tạo khoảng cách lề trên của Link nó sát mí như trong thiết kế, thì chúng ta cho margin-top:0

NOTE: Các thẻ link này được bọc bởi thẻ ul, do đó chúng ta sẽ chỉnh lại margin-top của thẻ ul thì mới có tác dụng.



NOTE: một tips khi xử lý CSS thì chúng ta sẽ tác động các lề lên thẻ bao bọc đoạn text đó gần nhất. Ví dụ như trong trường hợp bên trên thì trong quá trình hiện thực thì khi ta tác động lên các đội tượng khác thường ko dc, cho đến khi chúng ta tác động lên thẻ <ul>, là thẻ bao bọc đoạn text đó gần nhất

+ Với lại khi chúng ta muốn chỉnh khoảng cách khít nhau thì có lưu ý là khi chúng ta chỉnh những block cho nó khít nhau thì chúng ta hiện thực bằng cách tác động lên các thẻ display: (như đã trình bày bên trên), còn muốn tác động cho đoạn text nó khít nhau, thì chúng ta tác động vô thẻ bao bọc nó gần nhất và điều chỉnh lại thông số margin

<https://topdev.vn/blog/tim-hieu-thuoc-tinh-position-trong-css/>

NOTE: Về bản chất thì khi chúng ta khai báo canh cho các lề như margin, position thì sẽ có sự khác biệt giữa khi chúng ta tác động CSS lên thẻ <div> và những thẻ mặc định như <h1>, <body>, <html>. Do những thẻ này nếu chúng ta ko tiền xử lý xóa đi những CSS mặc định (có thể coi về điều này trong F8) thì khi chỉnh chúng ta sẽ bị khó khăn, nên lưu ý để tránh , hạn chế tác động CSS lên những thẻ html có sẵn

**<https://www.youtube.com/watch?v=QpCPndGZ_mM&t=359s>**

**(Coi thêm kiến thức về z-index: khá lú)**

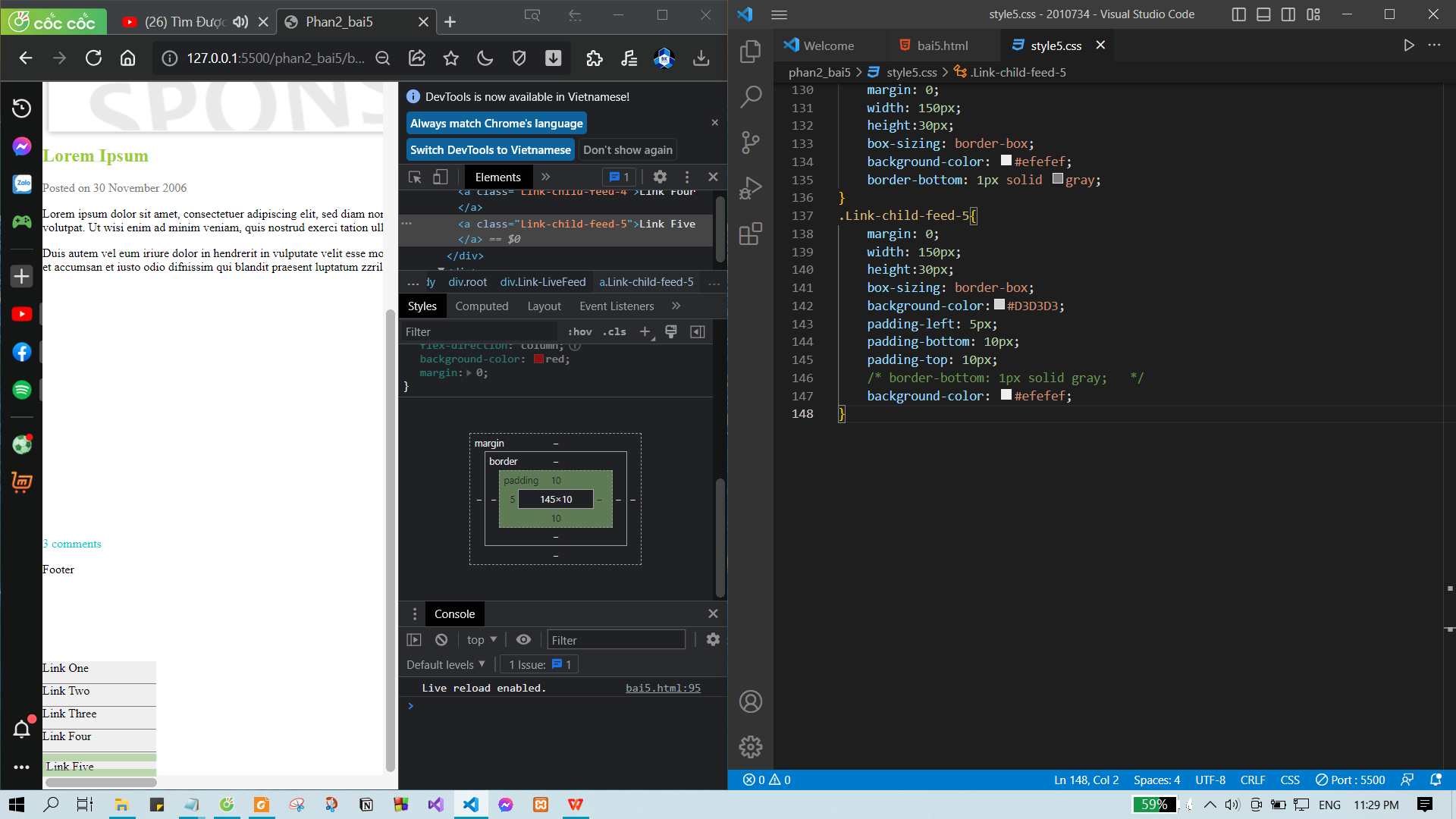
**Cách tạo đường viền như trong bài 5 - lab1 -phan2**

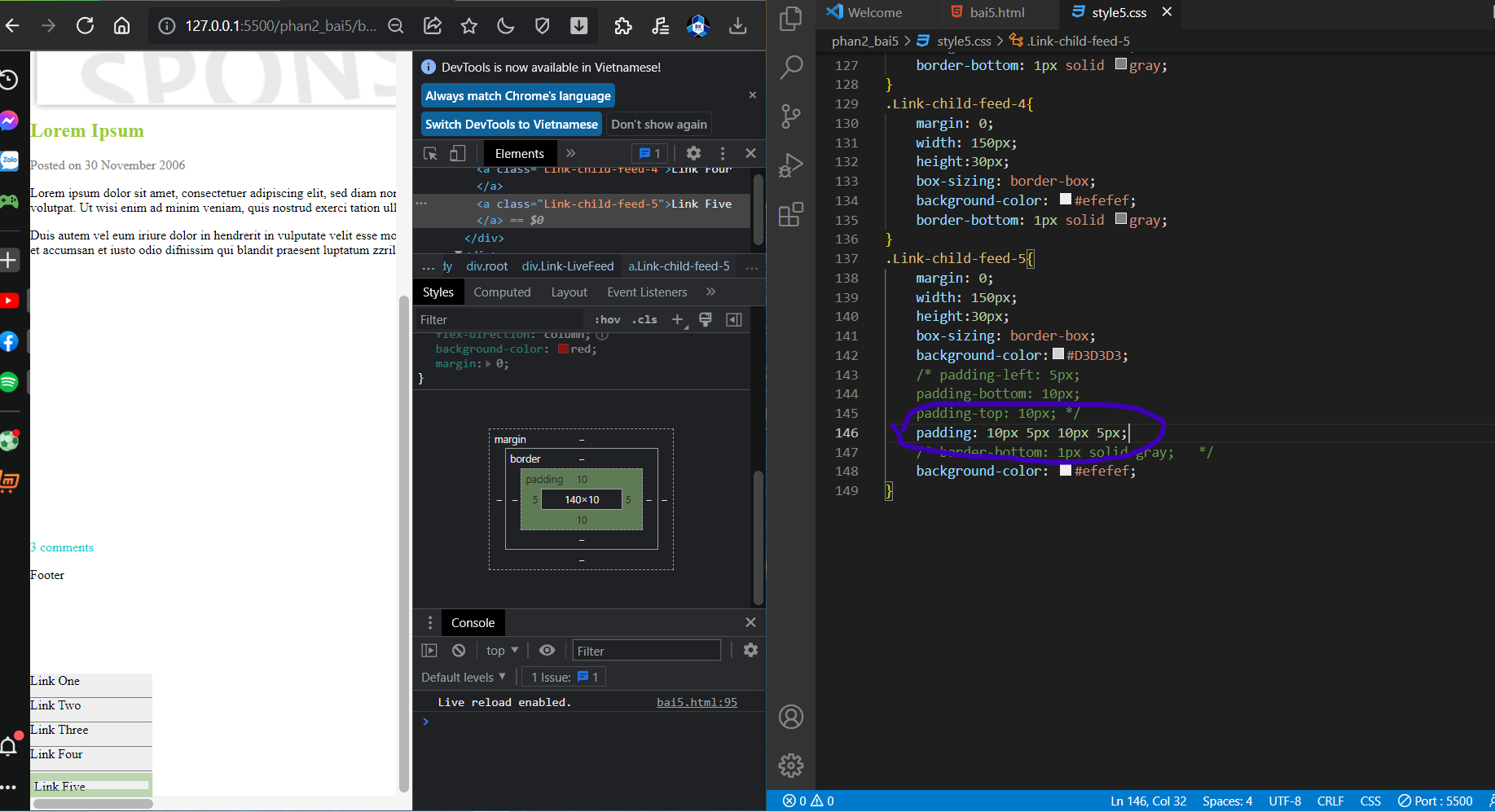
**Ngoài ra:** Có thể tham khảo thêm một số về thuộc tính bottom trong link tham khảo sau:

<https://hocwebchuan.com/reference/cssSection/pr_bdr.php>



**NOTE:**  Điều này khá là khắm lọ mà mình gặp trong quá trình code, là khi mình muốn canh cho đoạn Link trong cái box màu xám có khoảng padding-bottom bên dưới như trong đề lab1 - bài 5 - phần 2, thì nếu mình chỉ xài padding-bottom thì nó lại không nhảy theo ý mình mà cứ bị nảy lên trên một khoảng khá khó chịu, tuy nhiên khi khai báo padding-top cho nó đẩy xuống với cùng 1px thì nó lại hiển thị dc, và trong trường hợp chúng ta đều khai báo padding-top và padding-bottom thì nó lại hiển thị đúng





**NOTE:** Để chắc ăn thì có 1 tips thì chúng ta nên khai báo tất cả 4 hướng của padding cho an toàn.

<https://webcoban.vn/css/thuoc-tinh-padding-trong-css.html>

(Coi thêm minh họa về cách hiển thị của padding - có demo thử trong đường link trên)

=> Nguyên nhân của trường hợp chúng ta đang bàn từ nảy đến giờ là do nếu chúng ta chỉ khai báo mỗi padding-bottom, thì padding-top nó mặc định là 0 => nó ưu tiên đẩy đoạn text lên trên, nếu nó ưu tiên hiển thị theo bottom thì nó sẽ vi phạm padding-top:0 => chương trình sẽ không biết hiển thị như nào, do đó nó sẽ ưu tiên hiển thị giao diện là padding-top:0

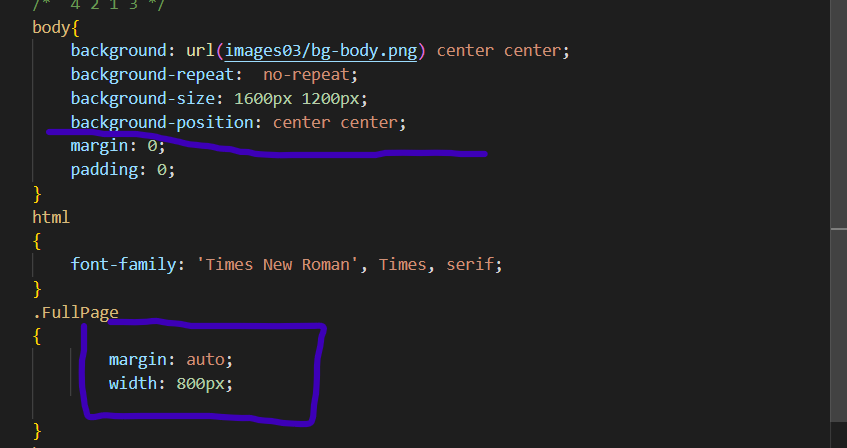
**NOTE:** Lưu ý , nếu chúng ta code theo kiểu sẽ có 1 block container, chứa các block con, thì chúng ta phải tuân theo công thức như này, nếu ko thì các block con sẽ thể hiện các kích thước chiều cao, chiều rộng ra giao diện không đúng với kích thước mà chúng ta khai báo.

=> Công thức: Height (container) = tổng chiều cao của các block con (những block mà nó chứa)

**NOTE:** Trong Lab1 - Phần 2 - Bài 5, nếu chúng ta muốn canh giữa đoạn văn vào chính giữa của 1 trang giao diện, thì chúng ta sẽ sử dụng thuộc tính margin: auto, để nó canh đều hai bên margin-left và margin- right. Tuy nhiên để thuộc tính margin này hoạt động tốt thì chúng ta ta phải khai báo độ rộng của đoạn text, hay bề rộng của vùng <div> mà chúng ta muốn canh đều, thì khi này chúng ta khai báo margin-auto, thì chương trình sẽ tự động canh đều margin-left và margin-right cho chúng ta

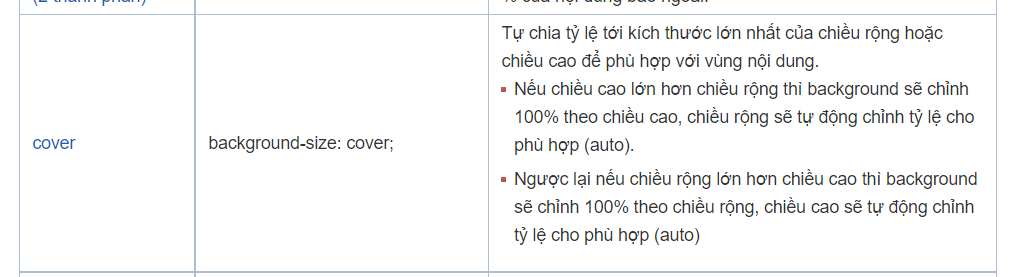
NOTE: Tham khảo thêm về background-position:

<https://webcoban.vn/css/cach-su-dung-thuoc-tinh-background-position-trong-css-de-thiet-lap-vi-tri-hinh-nen-cho-phan-tu-html.html>

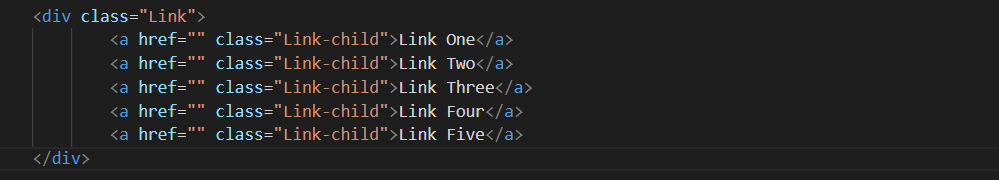


<https://hocwebchuan.com/tutorial/css3/background-size.php>

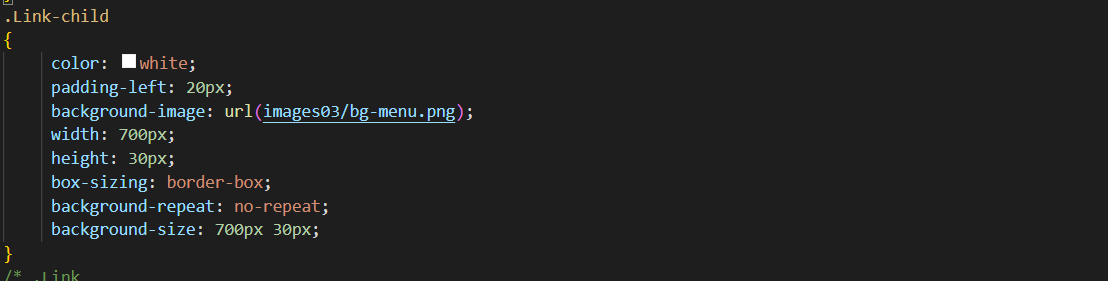
(Đọc thêm về các cách xử lý trong background-size)



Ví dụ: Chúng ta có class cha là class=”Link” bao bọc các class con là “Link-child”, thì nếu chúng ta muốn dùng 1 picture, cụ thể trong lab1 - phần 2, thì chúng ta dùng “bg-menu.png” để tạo giao diện như đề, thì vấn đề đặt ra là chúng ta có thể đặt css chèn hình/ chỉnh hình vào 2 class “Link” or “Link-child”, thì với 2 cách đặt khác nhau, giao diện sẽ trả về hai kết quả khác nhau.



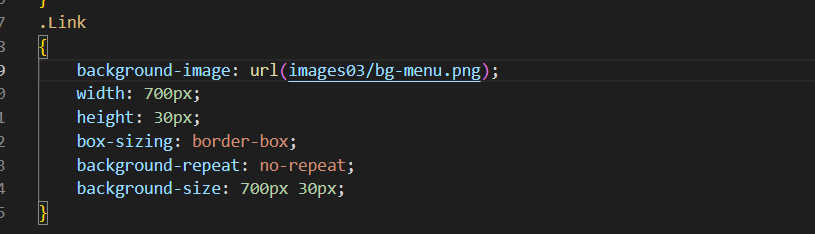
**Trường hợp 1: Đặt đoạn xử lý back-ground trong class con**



Kết quả:



**Trường hợp 2: Đặt đoạn xử lý back-ground trong class cha**



Kết quả



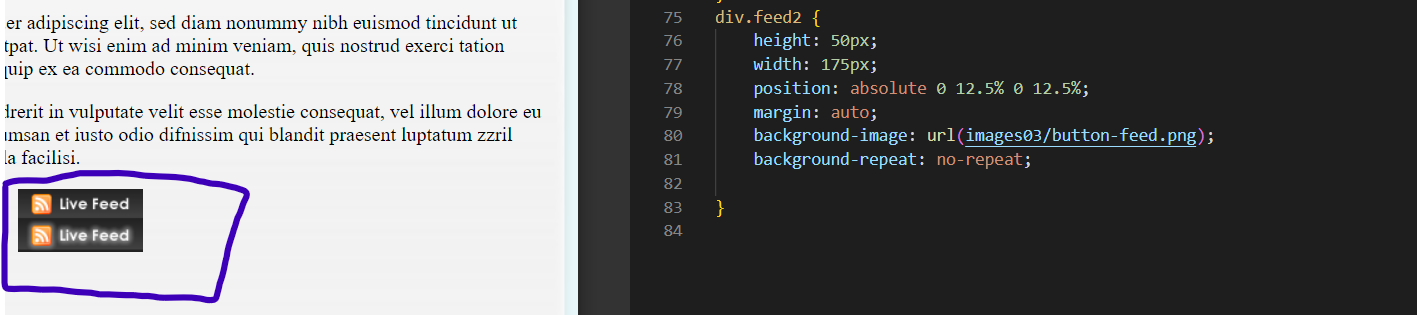
NOTE: Ví dụ trong trường hợp chúng ta có ảnh Live-feed như trong hình, thì để thu gọn nó lại chỉ còn 1 live-feed, thì cách giải quyết ban đầu là ta giữ nguyên kích thước của hình, và dùng z-index để ẩn nó đi, dùng position để dịch chuyển nó đến vị trí cần thiết, tuy nhiên với cách làm này thì sẽ không tối ưu, nguyên nhân như sau:

+ Mỗi trình duyệt sẽ có sự hiển thị khác nhau, do đó nếu chúng ta lạm dụng mò số px để nó hiển thị đúng thì qua các trình duyệt khác maybe sẽ ko hiển thị đúng như ta cần.

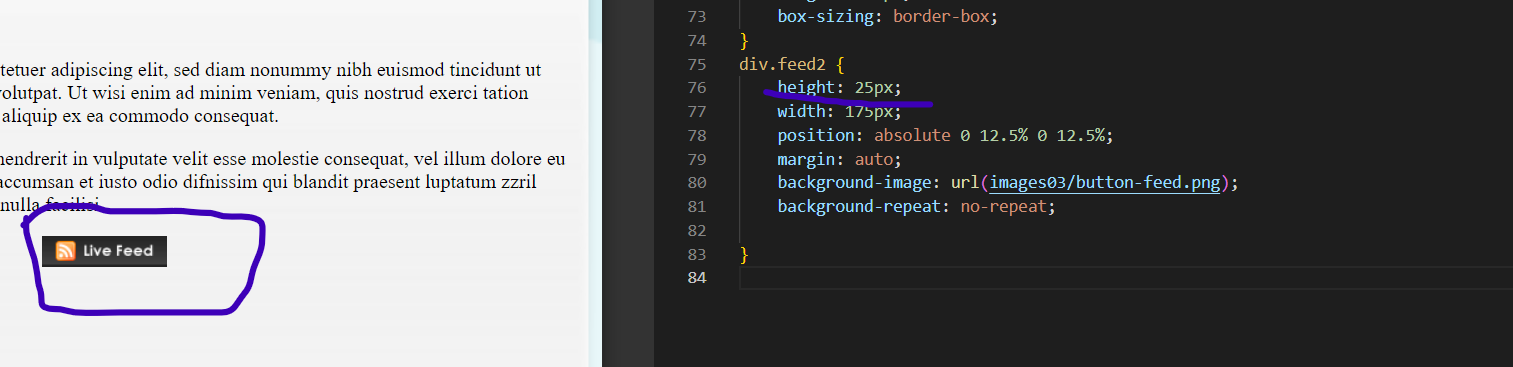
+ Thi CK là code giấy, ko có giao diện mô phỏng, do đó nếu chúng ta chỉnh tay số pixel thì khi thi CK sure kèo chúng ta ko áp dụng dc cách này

=> Cách giải quyết là chúng ta thu gọn chiều cao của nó lại cho đến khi thu dc hình như chúng ta cần.

Hình ban đầu



Hình sau khi resize độ cao

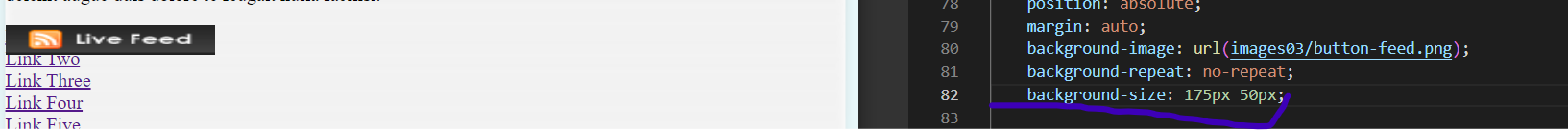


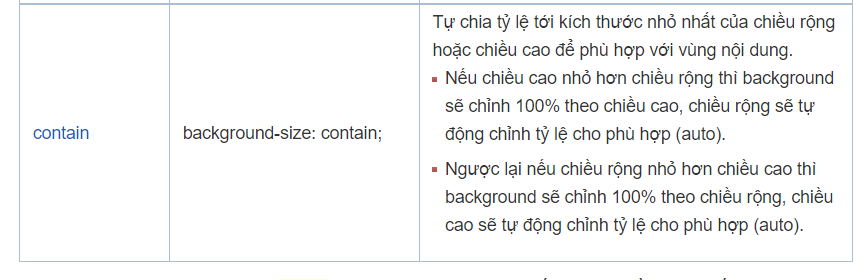
NOTE: Chúng ta phải phân biệt dc việt sử dụng background - size vs việc sử dụng width và height, nếu chúng ta sử dụng đoạn code như trên và khai báo thêm background-size, thì kết quả hiển thị ra giao diện như sau:



+ Nếu chúng ta ko sử dụng background-size, thì mang ý nghĩa là bức ảnh đó sẽ giữ nguyên kích thước nguyên bản của nó, nó sẽ được đặt trong 1 block có {width, height} như chúng ta khai báo

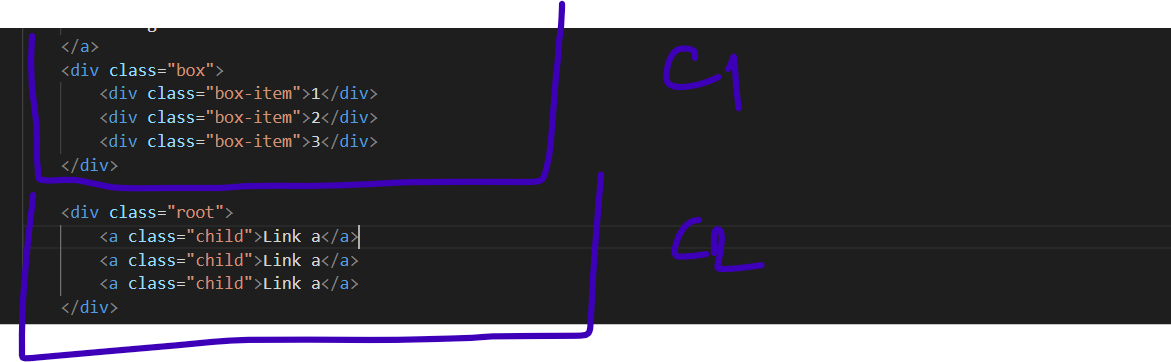
+ Tuy nhiên, nếu chúng ta chày cối, điều chỉnh độ cao của ảnh trong 1 block có độ cao giới hạn là 25px, thì chúng ta vẫn miễn cưỡng thu được ảnh như chúng ta muốn.





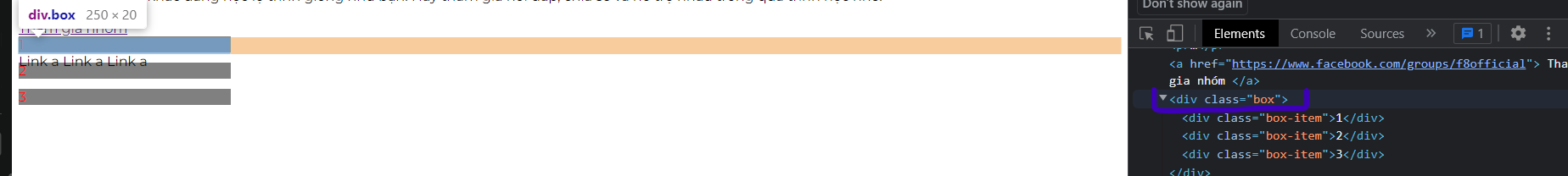
Coi thêm về chia trang theo kiểu float

<https://xuanthulab.net/thuoc-tinh-float-trong-css.html>



Chúng ta có 2 kiểu đóng gói , tuy nhiên kết quả hiển thị lại khác nhau (thẻ a lại không xuống hàng như 1,2,3. Nguyên nhân là nếu chúng ta đóng gói như kiểu 2 thì thẻ <a> mặc định là thẻ inline, nó sẽ hiển thị lên 1 dòng, còn thẻ <div> là thẻ block, nên nó sẽ xuống hàng theo từng block , như sau:



**NOTE:** Giả sử chúng ta khai báo độ cao cố định cho 1 block cha , thì liệu các block con có bị ảnh hưởng bởi chiều cao cố định của class cha hay ko. Câu trả lời là không, nên trong quá trình code giao diện, thì chúng ta hạn chế fix height cố định cho class cha, chúng ta cứ để chương trình tự hiểu height of class root = tổng height của các class con.  


Coi thêm về cách sử dụng float

<https://quantrimang.com/hoc/thuoc-tinh-float-va-clear-trong-css-162894>

NOTE:

+ Trong bài Lab 1 này, chúng ta áp dụng kiến thức về float trong bài lab1 - phần 2- bài 5, chúng ta tạo 1 thẻ <div> cha để bao trùm lên phần nội dung và phần Livefeed, sau đó dùng float để chia 2 thằng ra, 1 block thì tràn về bên trái, 1 block thì tràn về bên phải, tuy nhiên để chia được thì chúng ta phải khai báo chiều rộng mỗi block cho nó.

+ Ngoài ra chúng ta có thể sử dụng flexbox để chia ra làm 2 như những bài trên.

<https://xuanthulab.net/su-dung-flexbox-bo-cuc-trang-trong-css.html>