**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**Học phần: Các công nghê lập trình hiện đại**

**Đề tài**

**TÌM HIỂU CSS GRID**

**Nhóm 01 - Buổi chiều**

Trần Huy Hoàng 3115410042

Nguyễn Nhật Hùng 3115410046

Hà Quốc Huy 3115410049

Thái Minh Hưng 3115410054

**GVHD:** Phạm Thi Vương

*TP Hồ Chí Minh, tháng 12 năm 2019*

Mục lục

[Lời nói đầu 3](#_Toc26795588)

[Bảng phân công 4](#_Toc26795589)

[**1.** **Lịch sử Web Layout** 5](#_Toc26795610)

[*1.1.* *Presentational HTML Tags* 5](#_Toc26795611)

[*1.2.* *The Single Pixel GIF Trick* 5](#_Toc26795612)

[*1.3.* *Table Layout* 6](#_Toc26795613)

[*1.4.* *Frames* 7](#_Toc26795614)

[*1.5.* *Client-Side Imagemaps* 8](#_Toc26795615)

[*1.6.* *Cascading Style Sheets* 9](#_Toc26795616)

[*1.6.1.* *DIVS and the "Box Model"* 9](#_Toc26795617)

[*1.6.2.* *CSS Position Property* 10](#_Toc26795618)

[*1.6.3.* *Floated Layouts* 10](#_Toc26795619)

[*1.6.4.* *Mobile and Responsive Sites* 14](#_Toc26795620)

[*1.6.5.* *CSS Frameworks* 15](#_Toc26795621)

[*1.6.6.* *Flexbox* 17](#_Toc26795622)

[*1.7.* *Ưu điểm, nhược điểm và sử dụng của Float, Table, Inline Block, Flexbox* 20](#_Toc26795623)

[**2.** **CSS Grid** 24](#_Toc26795624)

[*2.1.* *Các thuật ngữ* 27](#_Toc26795625)

[*2.1.1.* *Grid lines* 27](#_Toc26795626)

[*2.1.2.* *Grid Tracks và Cells* 27](#_Toc26795627)

[*2.1.3.* *Grid Areas* 28](#_Toc26795628)

[*2.1.4.* *Gutters* 28](#_Toc26795629)

[*2.1.5.* *Grid container* 29](#_Toc26795630)

[*2.1.6.* *Explicit Grid* 29](#_Toc26795631)

[*2.1.7.* *Hỗ trợ trình duyệt* 30](#_Toc26795632)

[*2.2.* *Grid & Flexbox* 31](#_Toc26795633)

[*2.2.1.* *Một chiều và hai chiều* 31](#_Toc26795634)

[*2.2.2.* *Content-first với layout-first* 32](#_Toc26795635)

[*2.2.3.* *Khác nhau giữa CSS Grid và Flexbox* 34](#_Toc26795636)

[**3.** **Ứng dụng** 35](#_Toc26795637)

[Tài liệu tham khảo 41](#_Toc26795638)

# Lời nói đầu

Bởi vì được dựa trên nền tảng Digital, thiết kế website có thể dễ dàng tích hợp các ứng dụng công nghệ hơn hẳn so với báo chí in ấn truyền thống. Nhưng nhìn lại xem, chúng ta đã bước sang năm thứ 18 của thế kỉ mới, xu hướng tích hợp công nghệ trên còn gây cho bạn bất ngờ không ?

Điều ấn tượng ở đây là cách những người thiết kế liên tục liên tục áp dụng các công nghệ mới lên nền tảng website của mình, đồng thời luôn giữ được sự cân bằng nhất định giữa độ thân thiện với người dùng, sự rõ ràng, tính sáng tạo, sự đồng nhất với phong cách doanh nghiệp, thích ứng với mọi thiết bị,.. và tổng hợp lại trên một website.

2018 cho thấy vô vàn sự tiến bộ, lượng truy cập website bằng smartphone, tablet cuối cùng cũng đã vượt qua được máy tính. Điều đó cho thấy, mục tiêu hứa hẹn của năm 2019 là tối ưu được chức năng lướt web qua mobile bằng những cách chúng ta chưa bao giờ thấy cả, trong khi nền tảng máy tính thông thường vẫn phải tiếp tục phát triển.

Gird Layout - Một sự thay đổi lớn ở năm 2018 chính là giới thiệu xu hướng này, và đối với năm 2019, 100% xu hướng này chưa giảm được nhiệt của nó đâu. Hiệu ứng này đem tới cho người dùng sự độc nhất, khác biệt, và đôi khi là những trải nghiệm chưa bao giờ có.

Mặc dù những thương hiệu lớn có vô số những thông tin và nội dung muốn gửi tới khách hàng, thì họ vẫn đang sử dụng cấu trúc lưới truyền thống, nhưng không vì thế mà những “layouts” khác biệt này sẽ biến mất.

# Bảng phân công

|  |  |  |  |
| --- | --- | --- | --- |
| MSSV | Họtên | Côngviệc | Tự đánh giá |
| 3115410042 | Trần Huy Hoàng | Chương I, II | Hoàn thành |
| 3115410046 | Nguyễn Nhật Hùng | Chương II, III | Hoàn thành |
| 3115410049 | Hà Quốc Huy | Chương II, III | Hoàn thành |
| 3115410054 | Thái Minh Hưng | Chương I, II | Hoàn thành |

1. **Lịch sử Web Layout**
2. *Presentational HTML Tags*

Giữa những năm 1990 là thời đại mà các nhà sản xuất trình duyệt lớn (Netscape và Microsoft) đã thêm các thẻ để cho phép các nhà thiết kế kiểm soát giao diện trang của họ bằng các thẻ "mang tính trình bày" như:

in đậm <b> và <i> cho chữ nghiêng. Một loạt các thẻ cũng cho phép thay đổi kích thước phông chữ <font size = "+ 2">, mặt phông chữ <font face = "verdana"> và màu của các thành phần trang với các thẻ như <font color = "blue"> và <body bgcolor = "# e6e6fa">. Thậm chí một số mức độ căn chỉnh hình ảnh có thể được thực hiện bằng một thuộc tính cho thẻ <img>, chẳng hạn như <img src = "smiley.gif" align = "left">.

Sử dụng HTML để kiểm soát bố cục sớm trở thành một vấn đề lớn đối với các nhà phát triển web để duy trì các trang. Thay đổi sẽ phải được thực hiện trong nhiều trường hợp trên nhiều trang của một trang web.

Thêm vào đó, những người sáng lập web, như Tim Berners-Lee, đã hình dung HTML chỉ được sử dụng để xác định cấu trúc của một tài liệu, không phải để kiểm soát việc trình bày.

1. *The Single Pixel GIF Trick*

Một yêu thích thời bấy giờ là việc sử dụng các ảnh trong suốt pixel (GIF) để thêm khoảng cách giữa các thành phần trên một trang.

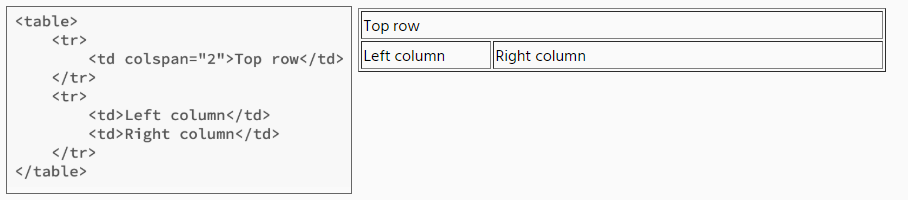
Một hình ảnh pixel đơn được tải xuống nhanh chóng và vì nó không nhìn thấy (trong suốt), nhà thiết kế có thể chỉ cần thay đổi khoảng cách ngang và / hoặc dọc bằng cách thêm các thuộc tính khoảng cách như <img src = "smiley.gif" hspace = "75"> hoặc chiều rộng và / hoặc các thuộc tính chiều cao, chẳng hạn như <img src = "single.gif" width = "150"> để tạo bất kỳ lượng không gian trống nào mong muốn.

Ví dụ : The space here... ...is created by a one pixel image set to a width of 150 pixels.

1. *Table Layout*

Việc sử dụng các thẻ trình bày và hình ảnh trong suốt pixel cung cấp các công cụ hạn chế để kiểm soát bố cục trang. Người ta đã sớm phát hiện ra rằng sử dụng các bảng dữ liệu có thể là một công cụ mạnh mẽ để xử lý bố cục.

Đến năm 1996, các bảng được thiết kế theo thông số kỹ thuật ban đầu cho HTML để giữ dữ liệu dạng bảng trong các hàng và cột, nhưng rõ ràng là các ô riêng lẻ của một bảng có thể được sử dụng để giữ các thành phần trang với nhau theo bố cục đa chiều.



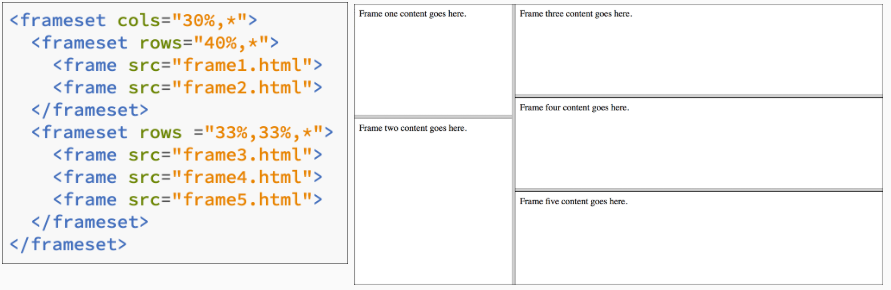
Mặc dù không được thiết kế để bố trí, các bảng đã trở thành phương pháp để bố trí nội dung trang vào những năm 2000. Giống như việc sử dụng các thẻ trình bày được nêu ở trên, việc sử dụng các bảng HTML để kiểm soát bản trình bày đã phá vỡ sự hợp lý của HTML chỉ được sử dụng để mô tả cấu trúc của một trang. Chỉnh sửa các trang là cồng kềnh và khó khăn.

1. *Frames*

Cũng trong khoảng thời gian các bảng được sử dụng để bố trí, việc sử dụng các khung HTML là phổ biến.

Khi sử dụng thẻ <frameset>, một tài liệu HTML có thể chỉ định các khu vực của trang nơi các tệp HTML khác có thể được bao gồm bằng cách sử dụng thẻ <frame src>.

Các khung thậm chí có thể được lồng vào nhau để kiểm soát tốt hơn. Ví dụ dưới đây cho thấy năm trang riêng biệt được bao gồm trong thẻ <frameset> chính.



1. *Client-Side Imagemaps*

Client-side imagemaps là hình ảnh lớn với các khu vực (hotspots) mà bạn có thể nhấp vào để liên kết đến các trang khác.

Các nhà thiết kế đồ họa yêu thích bản đồ hình ảnh vì họ có thể tạo ra các hình ảnh tinh vi trong Photoshop và biết rằng thiết kế sẽ trông giống hệt như khi xem trên trình duyệt.

hotspots được tạo bằng một số mã HTML đơn giản chỉ định tọa độ cho hình chữ nhật (lá cờ), hình tròn (quả bóng) hoặc đa giác (cái chòi).



Với imagemaps, không có mối quan tâm về cách kiểm soát bố cục trang.

Toàn bộ bố cục của một trang có thể được thực hiện với một hình ảnh lớn.

Cũng như Table và Frames, một lần nữa nó đã vi phạm mục tiêu của web để tách nội dung khỏi bản trình bày.

1. *Cascading Style Sheets*

Vào cuối năm 1990, những ông lớn trong các cuộc chiến các trình duyệt (Netscape và Microsoft) đã hợp tác với nhau và đồng ý loại bỏ các presentational tags (thẻ trình bày).

World Wide Web Consortium đã đưa ra khuyến nghị đầu tiên cho CSS (CSS1) vào năm 1996.

Một bản cập nhật, CSS 2, được xuất bản năm 1998 và nó cung cấp các cách bố trí mới như fixed ,absolute, and relative  và chỉ mục z-index.

Bản nháp CSS 3 xuất bản vào năm 1999. CSS 3 là thông số kỹ thuật hiện tại cho CSS và tiếp tục cho thấy các mô-đun mới và đa số trong đó đã cung cấp cho các công cụ mới mạnh mẽ để bố trí (layout) như Flexbox và CSS Grid.

1. *DIVS and the "Box Model"*

Phần tử <div> được tạo như một phần của CSS để chia trang thành các phần logic. Thẻ <div> có thể được sử dụng để giữ văn bản và hình ảnh trong bố cục.

Những <div> này cũng có thể thiết lập cho chiều rộng và chiều cao. Chúng có thể có đường viền và thậm chí có thể cho phép đệm (không gian xung quanh nội dung trong hộp) và lề (không gian xung quanh bên ngoài hộp).



1. *CSS Position Property*

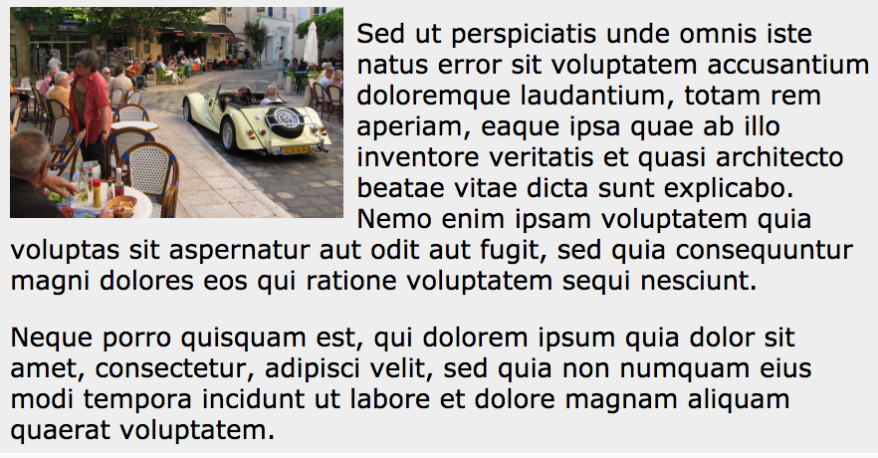
Theo mặc định, <div> s là các thành phần cấp khối và chúng xếp chồng lên nhau, trừ khi được định vị (positioned) theo một cách nào đó. Công cụ CSS đầu tiên để xử lý căn chỉnh các yếu tố này là **position** -thuộc tính vị trí (position:fixed).

Thuộc tính **position** vẫn là một phần của thông số kỹ thuật và vẫn còn rất hữu ích cho đến ngày nay. Có năm giá trị **position** khác nhau:

* *position:* **static** (tĩnh) là mặc định được định vị.
* *position:* vị trí **relative** (tương đối) so với vị trí bình thường của họ.
* *position:* **fixed** được định vị tương đối với chế độ xem (cửa sổ trình duyệt), có nghĩa là nó vẫn ở vị trí trên màn hình ngay cả khi cuộn trang.
* *position:* vị trí **absolute** (tuyệt đối) liên quan đến ancestor (gốc) vị trí gần nhất.
* *position:* vị trí **sticky** (dính) dựa trên vị trí cuộn của người dùng.

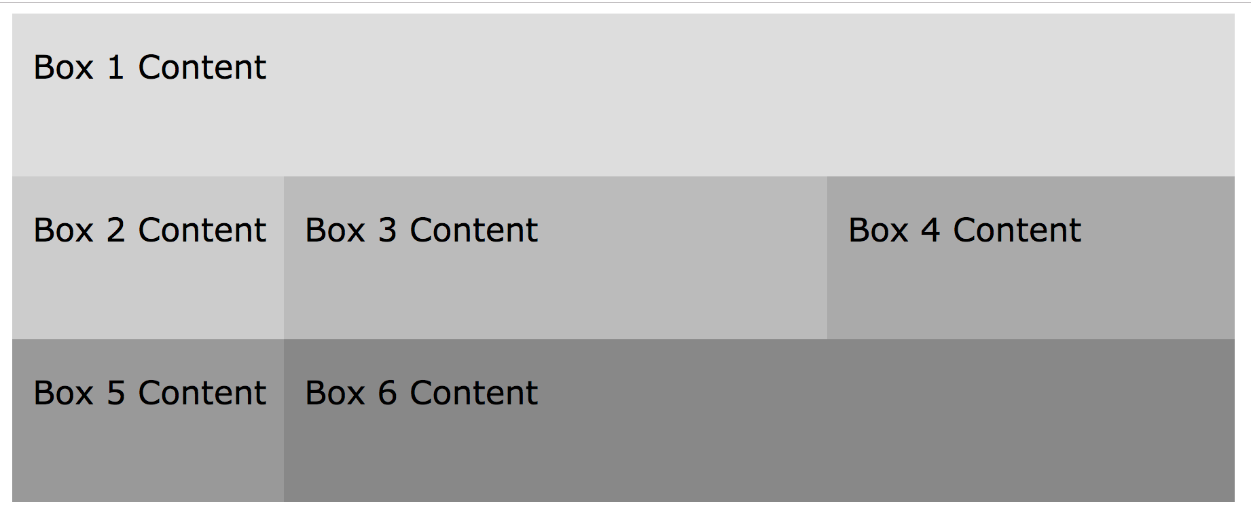
1. *Floated Layouts*

Hình ảnh dưới đây cho thấy Float được sử dụng như thế nào. Trong trường hợp này, hình ảnh được thả sang bên trái với **float: left** được áp dụng cho hình ảnh.



Khi chiều rộng của các hộp liền kề với nhau vượt quá chiều rộng của container, <div> tiếp theo chỉ cần chuyển xuống hàng tiếp theo.

***FLOATE IN ACTIONS***

**

Trong ví dụ này, container có chiều rộng 900 pixel. Các hộp lồng nhau đều được đặt thành **float: left**.

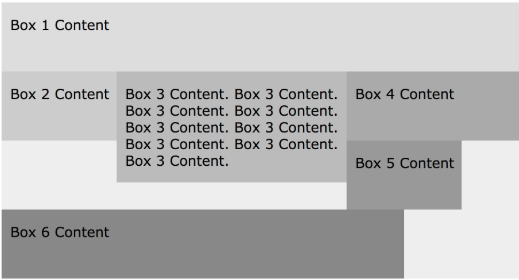
Hộp 1 rộng 900 pixel, vì vậy nó chính xác vừa với hàng đầu tiên và hộp 2 phải chuyển xuống hàng tiếp theo.

Hộp 2 rộng 200 pixel.

Hộp 3 rộng 400 pixel và Hộp 4 rộng 300 pixel.

các hộp 2, 3 và 4 có chiều rộng 900 pixel, vì vậy tất cả chúng khớp với nhau trên hàng thứ hai.

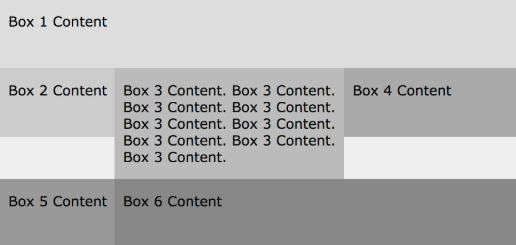
Hộp 5 không vừa, vì vậy nó di chuyển xuống hàng dưới cùng. Nó rộng 200 pixel. Cuối cùng, hộp 6 là 700 pixel, do đó, nó chiếm không gian còn lại ở hàng dưới cùng.

******

Một vấn đề phổ biến với float layout được gọi là **hanging float**.   
Nếu các phần tử float bên trái, một **hanging float** được tạo khi một phần tử trên cùng một hàng cao hơn nội dung của một hộp ở bên phải của nó.

Trong ví dụ này, hộp 5 sẽ xóa tất cả các hàng tiếp theo, nhưng thay vào đó, bị cuốn vào hộp có chiều cao lớn hơn. Trong trường hợp này, hộp 5 đang bị treo lên trên hộp 3.

***CLEARING FLOATS***



Để Sửa hanging floats yêu cầu bạn xóa float trên phần tử.

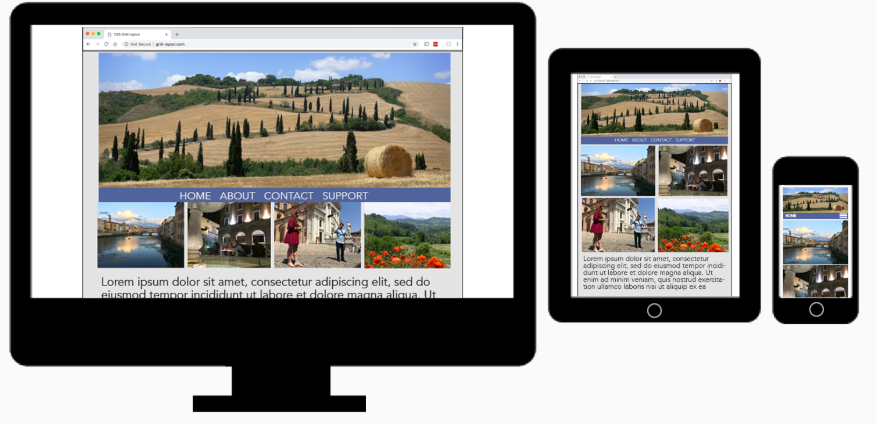
Vì vậy, chúng ta phải thêm clear:left or clear: both vào <div> cho hộp 5

Bây giờ, nó sẽ đi xuống hàng tiếp theo và bắt đầu ở phía bên trái của bố cục.

1. *Mobile and Responsive Sites*

Ngày nay, xu hướng là tạo một trang web phù hợp với kích thước hiển thị của thiết bị. Thiết kế web đáp ứng (RWD) cố gắng làm cho các trang web hiển thị tốt trên nhiều thiết bị và kích cỡ màn hình. Mục tiêu là làm cho các trang vừa dễ sử dụng vừa thỏa mãn.

Hình minh họa dưới đây cho thấy một trang web cần thay đổi bố cục như thế nào để thích ứng với các kích thước màn hình khác nhau.



Thiết kế web **Responsive** dựa trên các truy vấn phương tiện để biết khi nào cần thay đổi bố cục.

Truy vấn phương tiện là một đoạn mã CSS xác định độ phân giải của thiết bị hiển thị trang.

Sau đó, nó phục vụ các CSS hoặc bố cục khác nhau dựa trên những gì nó xác định. Trong ví dụ bên dưới, văn bản bên trong tiêu đề sẽ hiển thị ở mức 1,2em nếu độ phân giải màn hình là 768 pixel hoặc rộng hơn.

@media only screen and (min-width: 768px) {  
header {  
font-size: 1.2em;  
}  
}

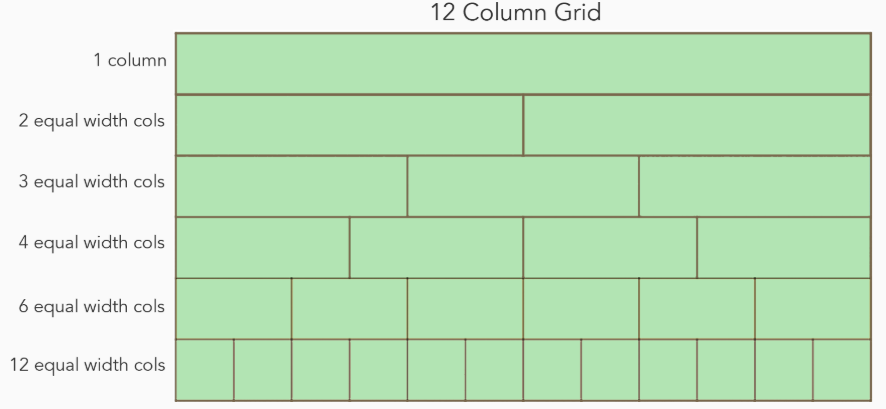
CSS Grid đặc biệt phù hợp để tạo các trang web **Responsive**.

1. *CSS Frameworks*

Khi nhu cầu về các trang web đáp ứng đã tăng lên, sự phức tạp của bố cục. Sự phức tạp này đã dẫn đến sự phát triển của một số **Frameworks** để giúp việc tạo trang web đáp ứng dễ dàng hơn. Hai loại được sử dụng rộng rãi nhất là **Foundation** và **Bootstrap**.

Hầu hết các khung sử dụng hệ thống lưới 12 cột tích hợp để tạo responsive layouts.

Với 12 cột, người ta có thể nhận được số lượng cột có chiều rộng bằng nhau lớn nhất trên một hàng. Grid (lưới) bên dưới cho thấy có thể có các cột có chiều rộng bằng 1, 2, 3, 4, 6 và 12 với lưới 12 cột.





1. *Flexbox*

**Flexbox** là một công cụ layout một chiều để bố trí các phần tử trong các hàng hoặc cột. Nó được thiết kế để giải quyết nhiều vấn đề với **position** và **float**.

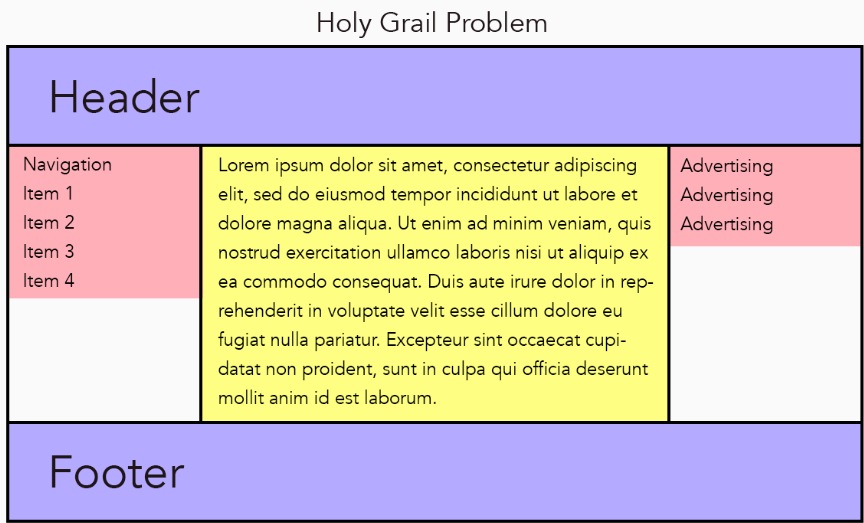
Bản thảo làm việc đầu tiên cho Flexbox đã được xuất bản vào năm 2009, nhưng nó đã trải qua một số thay đổi và bản dự thảo làm việc đã không được xuất bản cho đến năm 2013

***FINDING THE HOLY GRAIL***

Một trong những vấn đề mà Flexbox (và CSS Grid) đã giải quyết là một thứ gọi là Holy Grail Layout .

Việc tạo các cột có chiều cao bằng nhau trên cùng một hàng với số lượng nội dung khác nhau đã thách thức các nhà thiết kế web trong nhiều năm.

Trước Flexbox, tất cả các giải pháp đã biết đều có nhược điểm. Các giải pháp bao gồm divs với table-property displays, absolutely positioned divs, và JavaScript hacks.

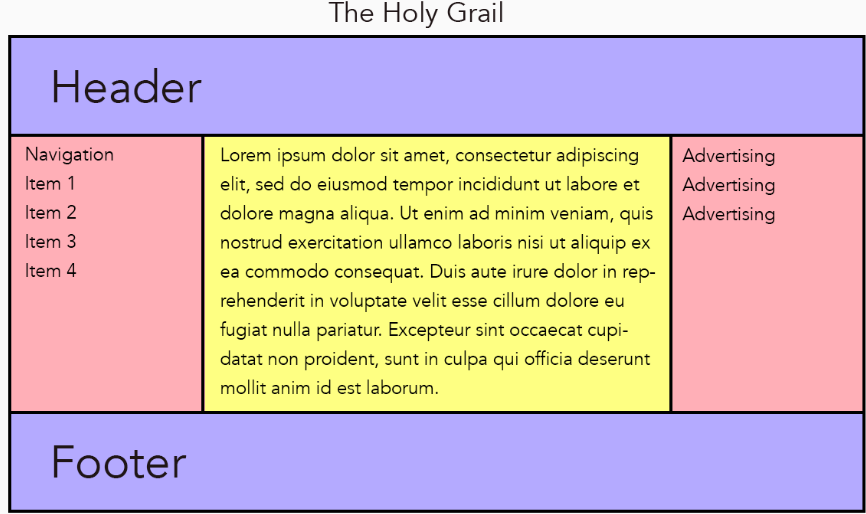


Có thể thấy rằng hàng thứ hai có ba cột.

Cột giữa là cao nhất, do đó nó lấp đầy tất cả không gian có sẵn.

Các cột thứ nhất và thứ ba có ít nội dung hơn.

Để mỗi yếu tố chiếm toàn bộ chiều cao mà không cần thêm nhiều nội dung là thách thức.



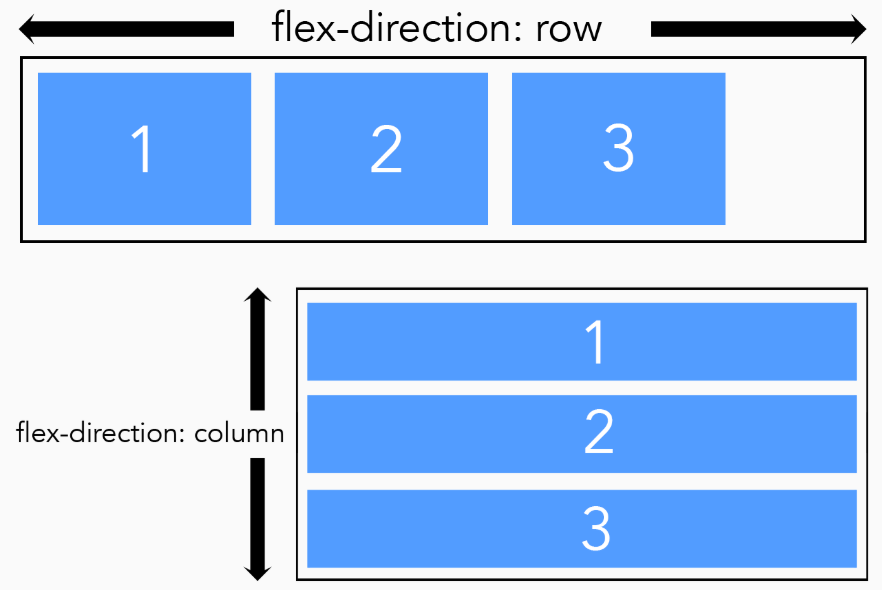
Ví dụ đây hiển thị the elusive Holy Grail - làm cho cả ba cột có chiều cao bằng nhau.

Flexbox là một cải tiến lớn ở chỗ nó cho phép căn chỉnh và căn giữa nội dung dễ dàng hơn và nó cho phép các cột có chiều cao bằng nhau, bất kể lượng nội dung là bao nhiêu.

Flexbox Layout được tạo thành từ thùng chứa flex (flex container) chứa các flex. Bạn có thể bố trí một thùng chứa theo chiều ngang (hàng) hoặc chiều dọc (cột). Bất cứ điều gì bạn chọn được gọi là trục chính (thường là hàng) và sau đó cái khác được gọi là trục chéo.

Các mục **flex** được đặt dọc theo trục chính và chúng có thể phát triển để lấp đầy không gian không sử dụng.

Bằng cách lồng các thùng chứa flex khác bên trong thùng chứa chính, bạn có thể đạt được sự kiểm soát đối với cả căn chỉnh ngang và dọc, mặc dù Flexbox về bản chất là một chiều.



Flexbox cung cấp các công cụ mạnh mẽ và hữu ích để tạo bố cục web.

1. *Ưu điểm, nhược điểm và sử dụng của Float, Table, Inline Block, Flexbox*

***FLOAT***

Ưu điểm:

* Cách phổ biến nhất để đặt mọi thứ ra; hầu hết các Grid-framwork đều theo điều này.
* Mọi người đều biết về bug float do sự phổ biến của floating và có nhiều cách để khắc phục chúng.

Nhược điểm:

* Cần được cleared. Có thể khá đau đớn nếu thay đổi độ rộng ở 2-3 điểm dừng truy vấn phương tiện, bởi vì một số float sẽ cần phải được cleared nhiều lần.
* Không định tâm dọc (vertical centering)
* Không có chiều cao bằng nhau (equal heights)
* Không độc lập trật tự nguồn.

Sử dụng cho:

* Các khối bố trí lớn không cần chiều cao bằng nhau và định tâm dọc.

***INLINE BLOCK***

Ưu điểm:

* Định tâm dọc là hữu ích nhất cho một số bố cục.
* Không cần phải được xóa. Hữu ích cho bố cục phức tạp khi các điểm dừng cần được thay đổi ở nhiều hơn một vài chiều rộng màn hình.

Nhược điểm:

* Bị từ khoảng trắng vốn có, sẽ làm phiền tính toán lưới. Không thể tạo lưới hai cột với bên trái là 30% và bên phải là 70%, với màn hình: khối nội tuyến, vì khoảng trắng sẽ phá vỡ lưới và đẩy cột thứ hai sang dòng tiếp theo.
* Không có chiều cao bằng nhau.
* Không độc lập trật tự nguồn.

Sử dụng cho:

* Khi yêu cầu định tâm dọc, nhưng không cao bằng CSS.
* Khi muốn tránh làm sạch lưới tại các điểm dừng khác nhau.
* Khi có thể giải quyết vấn đề khoảng trắng, mà không khiến phần còn lại của nhóm dev phát điên.
* Khi có thể giải quyết vấn đề về khoảng trắng, trong khi đảm bảo rằng ngôn ngữ tạo khuôn mẫu không cố gắng thêm các mục li.
* Khi đang sử dụng các mục danh sách để bố trí và không cần thẻ li đóng, khối nội tuyến thực sự tỏa sáng.

***TABLE***

Ưu điểm:

* Cung cấp trung tâm dọc và chiều cao bằng nhau.

Nhược điểm:

* Các div trình bao bọc bổ sung.
* Nếu yêu cầu khoảng cách giữa các ô của bảng, sẽ phải sử dụng thu gọn viền: tách biệt trên phần tử cha bằng màn hình: bảng (vì các ô của bảng không thể có lề), nhưng điều này sẽ thêm phần đệm vào cạnh ngoài cùng bên trái của ô đầu tiên và cạnh phải của ô cuối cùng. Điều đó có thể phá vỡ lưới.
* Nội dung có thể tràn các ô của nó và hơi khó xử lý.

Sử dụng cho:

* Những khoảnh khắc tinh tế khi cần định tâm dọc và chiều cao bằng nhau để bố trí đáp ứng chỉ với một điểm dừng để di chuyển từ bảng này sang khối khác.

***FLEXBOX***

Ưu điểm:

* Nguồn độc lập trật tự. Có thể có giá trị to lớn cho các bố cục đáp ứng và loại bỏ sự cần thiết của JS cho việc này.
* Định tâm dọc.
* Chiều cao bằng nhau.
* Các hộp Flex di chuyển dọc theo trục X và Y, dễ dàng như vậy, có thể thực sự thay đổi mọi thứ xung quanh với một vài thuộc tính.
* Hộp phát triển và co lại, có thể là cột hoặc hàng, vừa với không gian có sẵn tuy nhiên muốn khai báo điều này.
* Có nhiều cách để làm điều tương tự với flexbox.

Nhược điểm:

* Cú pháp ban đầu không trực quan.
* Không nhất quán của trình duyệt chéo.
* Khi bố cục trở nên phức tạp hơn hoặc muốn thêm một vài div, mọi thứ có thể gây nhầm lẫn.
* Rất nhiều tiền tố của nhà cung cấp, với một cú pháp khác nhau cho IE và Webkit cũ hơn.
* Không hoạt động trên IE9.

Sử dụng cho:

* Có thể bắt đầu sử dụng nó để định tâm dọc.
* Nếu không cần hỗ trợ IE9, nó hoàn hảo cho bố cục độc lập theo thứ tự nguồn, độ cao bằng nhau.
* Nên sử dụng nó cho các dự án cá nhân.
* Bố trí ứng dụng, nơi mọi thứ cần phải kéo dài và squish. Flexbox thực sự tỏa sáng ở đây.

1. **CSS Grid**

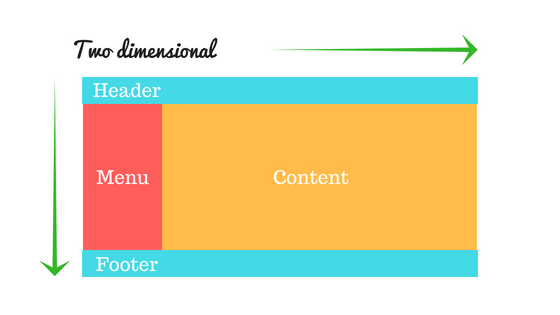
CSS Grid là một phương pháp bố trí CSS được phát triển dưới dạng bố cục hai chiều của các mục trên trang web hoặc ứng dụng, có nghĩa là nó có thể quản lý cả cột và hàng.

CSS Grid vượt trội hơn trong việc chia một trang thành nhiều phần hoặc xác định mối tương quan về kích thước, vị trí và lớp.

Layout dạng lưới (grid) là một phần gần như không thể thiếu trong bất cứ website nào. Trước đây, để phát triển grid trong CSS, chúng ta thường sử dụng một số cách như sau:

Dùng table hoặc float layout. Hai cách này giúp giải quyết ổn thỏa vấn đề, nhưng khi viết code có đôi chút không tự nhiên, dẫn đến khó bảo trì mã nguồn.

Dùng flexbox. Hạn chế của cách này là thiết kế cho lưới một chiều (chỉ có một dòng duy nhất).

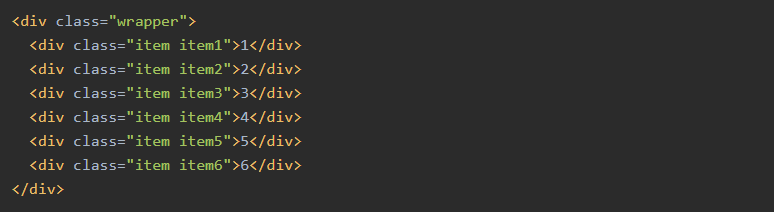


Giống tables, CSS Grid cho phép chúng ta sắp xếp các thành phần theo cột và hàng. Tuy nhiên, nhiều bố cục có thể đạt được hoặc dễ dàng hơn với CSS Grid so với bảng.

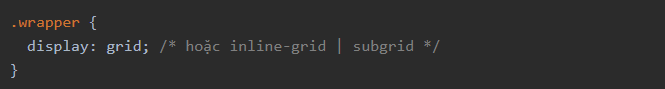
Ví dụ, một phần tử nằm trong container của CSS Grid có thể tự đặt chúng sao cho chúng thực sự chồng chéo và lớp, giống như thuộc tính position trong CSS.

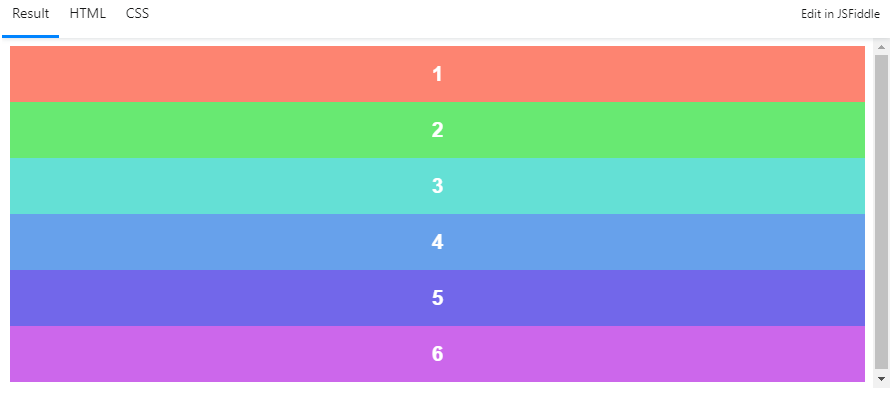
***TẠO MỘT GRID***

Một grid sẽ bao gồm hai thành phần chính: wrapper đóng vai trò grid container, và các item con là thành phần của grid.



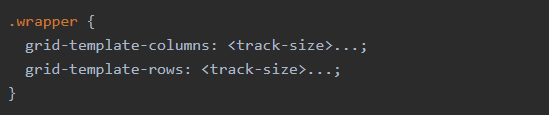
Để chuyển wrapper thành grid, chúng ta chỉ cần thay đổi thuộc tính display:





***CỘT VÀ HÀNG***

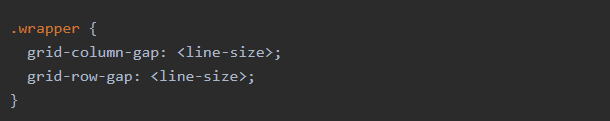
Để chia grid thành các cột và các hàng, chúng ta sẽ sử dụng 2 thuộc tính grid-template-columns và grid-template-rows.



***GRID GAPS***

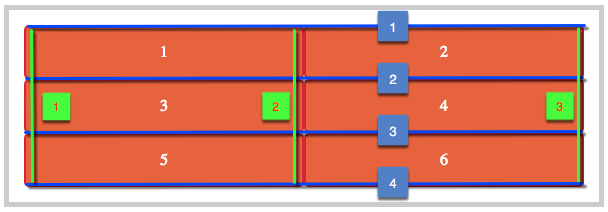
Khoảng cách giữa các cột trong grid được gọi là column-gap, còn khoảng cách giữa các hàng trong grid được gọi là row-gap.

Để thay đổi khoảng cách giữa các cột và các hàng, chúng ta sẽ sử dụng grid-column-gap và grid-row-gap.



1. *Các thuật ngữ*
2. *Grid lines*

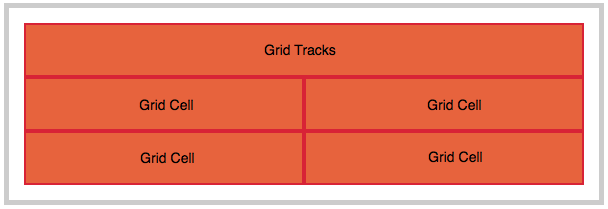
Là các đường ngang và dọc tạo thành cấu trúc grid. Chúng được sử dụng để định vị các khối trên grid.



1. *Grid Tracks và Cells*

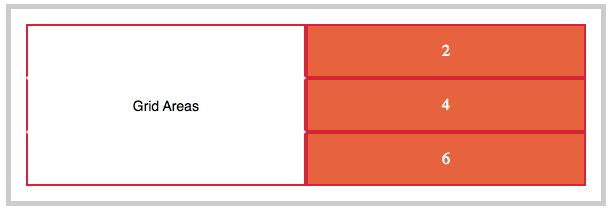
Grid Tracks là không gian giữa 2 đường liền kề nhau.

Grid Cell là khoảng không gian giữa 2 row liền kề của gird. Nó là những hàng và cột của grid. bạn có thể xem image ở dưới để hiểu rõ hơn.



1. *Grid Areas*

Các mục có thể kéo dài một ô hoặc nhiều ô theo chiều hàng hoặc theo cột để tạo thành grid areas. Vì thế grid areas không thể là một hình chữ L. bạn có thể theo dõi hình ảnh sau để hiểu hơn.

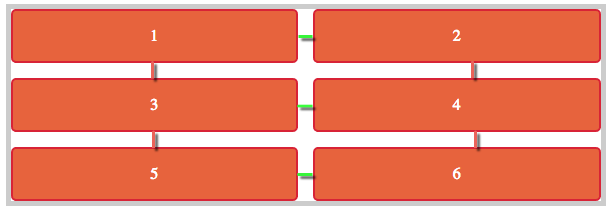


1. *Gutters*

Các khoảng cách giữa grid cells có thể được tạo ra bằng cách sử dụng grid-column-gap hoặc grid-row-gap điều thay cho việc chúng ta phải sử dụng padding hay margin:

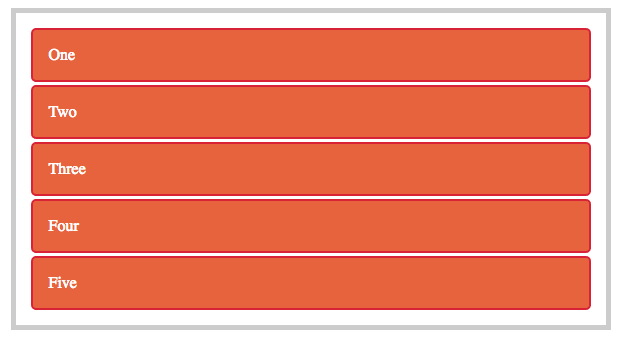
Màu xanh thể hiện grid-column-gap.

Màu đỏ thể hiện grid-row-gap



1. *Grid container*

Tạo một grid container bằng cách thiết lập thuộc tính display với giá trị grid hoặc inline-grid khi đó tất cả các item bên trong grid sẽ trở thành grid item.



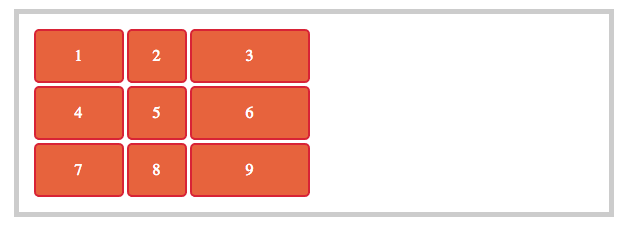
1. *Explicit Grid*

Explicit sẽ set cho 1 grid được tạo ra bởi columns and rows với thuộc tính grid-template-columns và grid-template-rows.

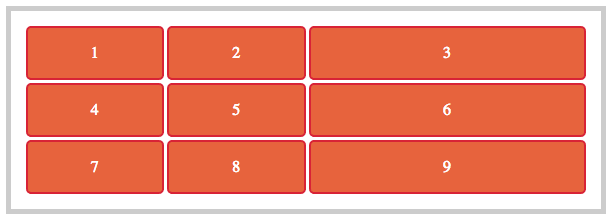


Một row trach là được tạo bởi giá trị của grid-template-columns. Giá trị này không âm, đơn vị được tính theo (px, %, em, etc.) ở ví dụ trên chúng ta thấy hàng thứ 3 nhận được giá trị height 100px các row còn lại sẽ nhận 50px theo giá trị của tham số đâu tiên.

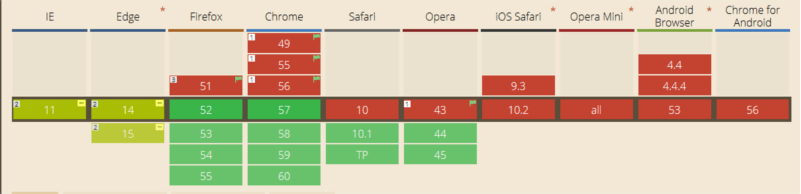
Giống như các row, các column được định nghĩa theo grid-template-columns. Ví dụ dưới chúng ta có thể thấy được chúng chỉ hiển thị 3 items trên 1 row, các items ở row dưới vì thế cũng bằng với kích thước của các row trên.



Ngoài ra chúng ta có thể dùng thuộc tính fr để hiển thị full trong một khối như grid-template-columns: 1fr 1fr 2fr; thì khi đó nó sẽ hiển thị như sau.

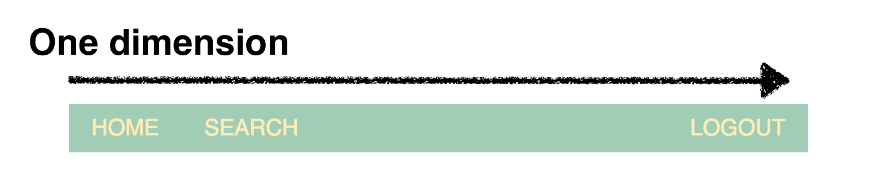


1. *Hỗ trợ trình duyệt*

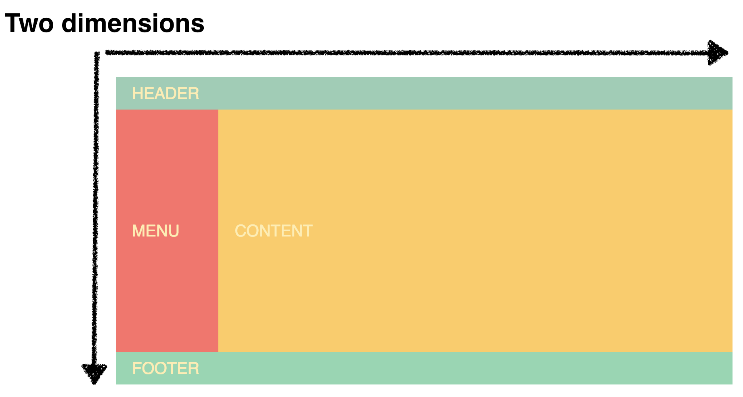
**

1. *Grid & Flexbox*
2. *Một chiều và hai chiều*

Nếu bạn đang đặt các item theo một hướng (ví dụ ba nút bên trong header), thì bạn nên sử dụng Flexbox, Nó sẽ linh hoạt hơn CSS Grid. Và cũng dễ dàng hơn để bảo trì và yêu cầu mã ít hơn.



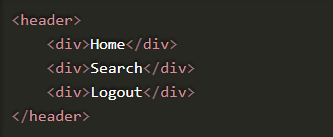
Tuy nhiên nếu bạn định tạo bố cục theo hai chiều - với cả các hàng và các cột - thì bạn nên sử dụng CSS Grid, Trong trường hợp này, CSS Grid sẽ linh hoạt hơn, làm cho code của bạn đơn giản hơn và dễ bảo trì hơn.



1. *Content-first với layout-first*

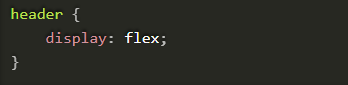
Một khác biệt quan trọng nữa giữa 2 mô-đun là Flexbox tập trung vào **nội dung** trong khi Grid tập trung vào **bố cục**.

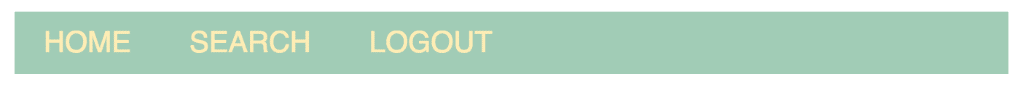
Ví dụ :





***FLEXBOX HEADER***



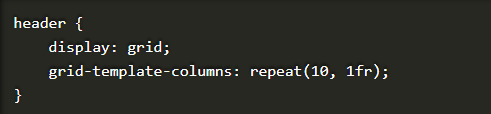


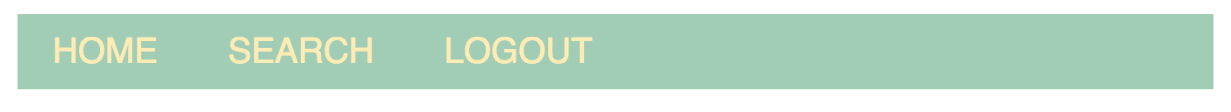


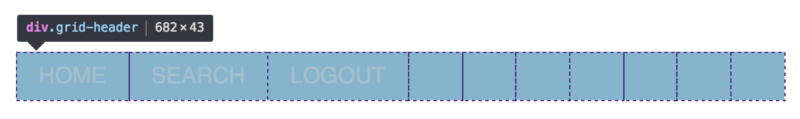


Ở đây là chúng ta để các item tự quyết định vị trí của mình. Chúng ta không phải xác định trước bất cứ điều gì khác ngoài display: flex;

***GRID HEADER***

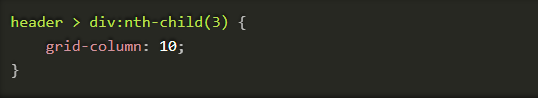


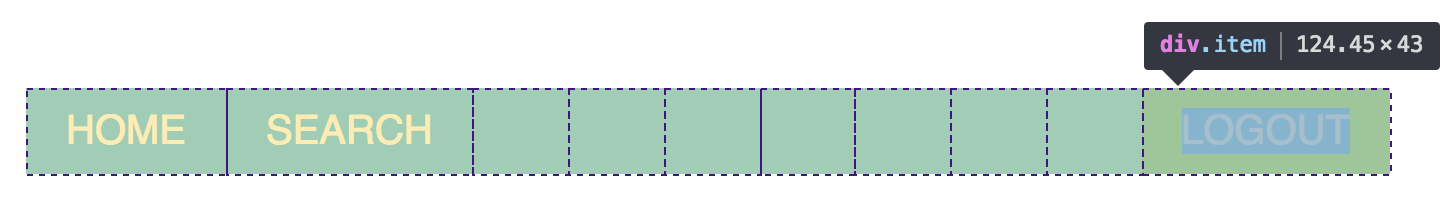




Sự khác biệt chính với cách tiếp cận này là chúng ta phải xác định các cột - bố cục - đầu tiên. Chúng ta bắt đầu với việc xác định chiều rộng của các cột, và sau đó chúng ta đặt nội dung vào các ô có sẵn trong grid.

Để thay đổi *logout* sang phía bên tay phải, chúng ta sẽ đặt nó vào cột thứ mười, như sau:





1. *Khác nhau giữa CSS Grid và Flexbox*

CSS Grid là hệ thống dựng layout 2 chiều, có nghĩa là chúng ta có thể xử lý theo cột và hàng, không giống như Flexbox là hệ thống layout 1 chiều. ( xử lý theo 1 cột hoặc 1 hàng ).

Điểm khác biệt cốt lõi giữa CSS Grid và Flexbox đó là: CSS Grid tiếp cận theo hướng nội dung còn Flexbox tiếp cận theo hướng bố cục (layout). Nếu bạn biết được rõ trước nội dung mình cần trình bày hãy dùng CSS Grid ngược lại thì chọn CSS Flexbox.

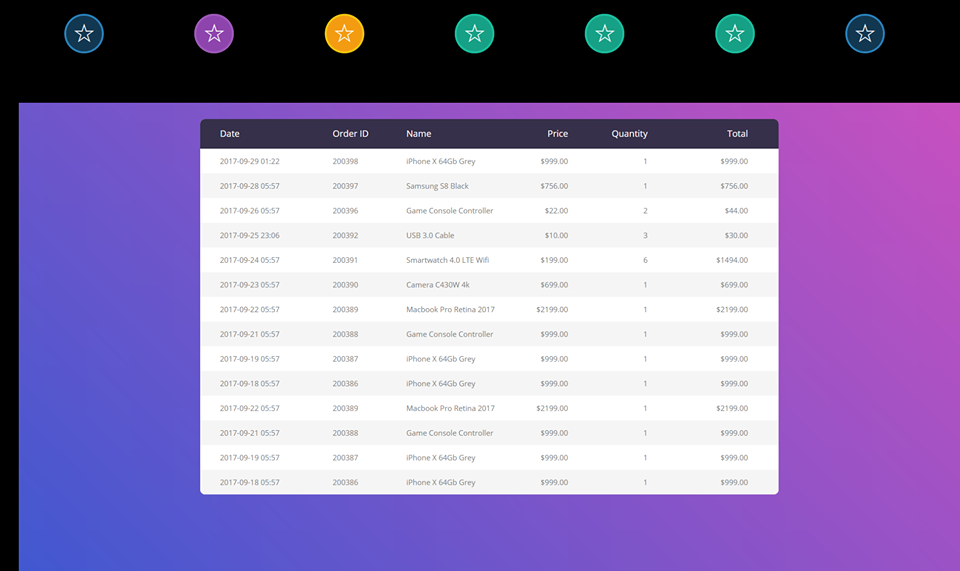
Flexbox phù hợp với các website có layout đơn giản, CSS Grid phù hợp với website có layout phức tạp hơn.

Nếu bạn chỉ cần dựng layout theo 1 hàng hoặc 1 cột thì Flexbox sẽ phù hợp hơn. Ngược lại, nếu bạn muốn xác định lưới và điều chỉnh nội dung theo 2 chiều - CSS Grid sẽ là lựa chọn của bạn.

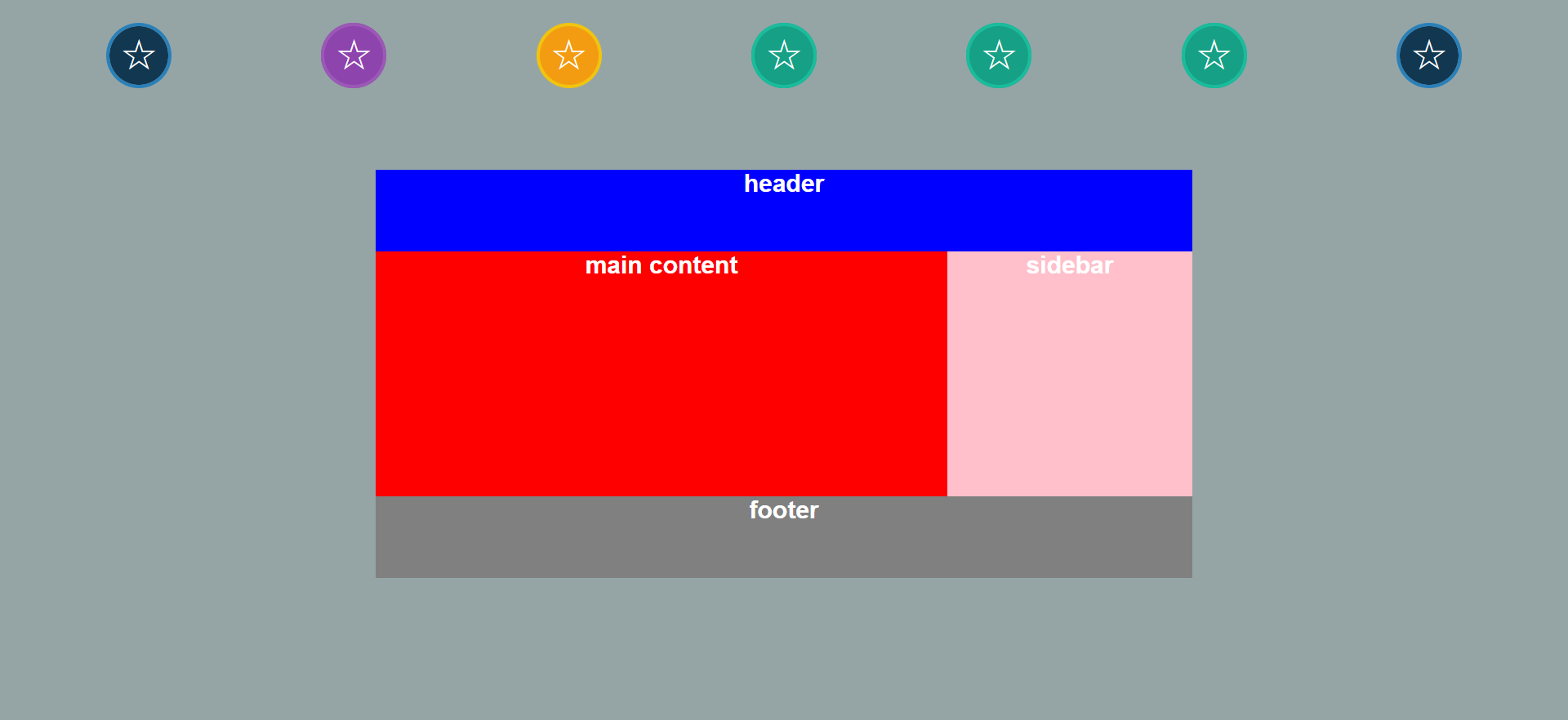
1. **Ứng dụng**

Nguồn: <https://github.com/hung7897/css-grid?fbclid=IwAR3-9h0NfYv0A28LH9NzjYtuWYhNF0yiMxQfoyc0JzJCqkiG_z5KCaDV-wQ>

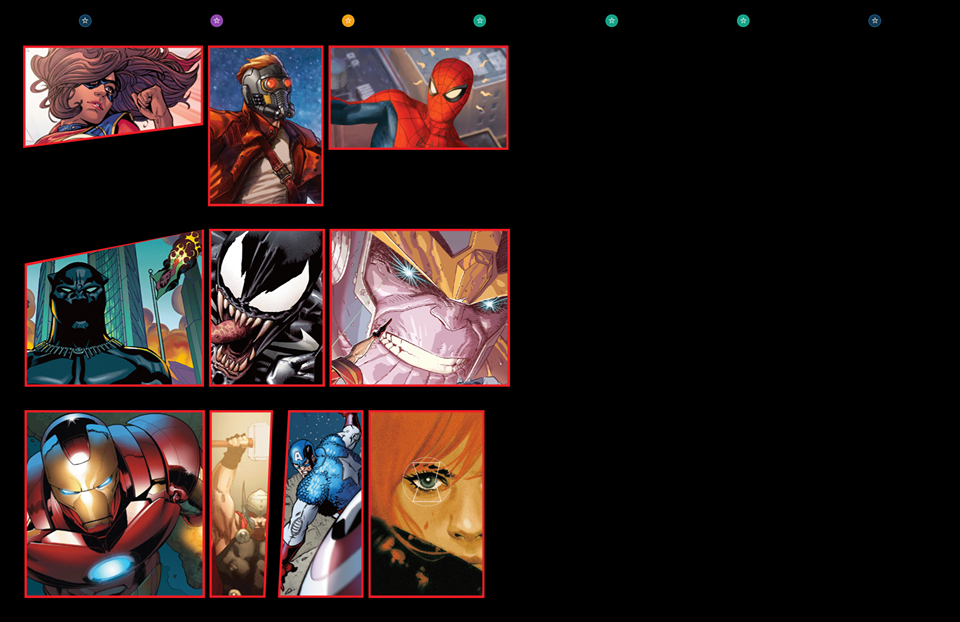
***TABLE***

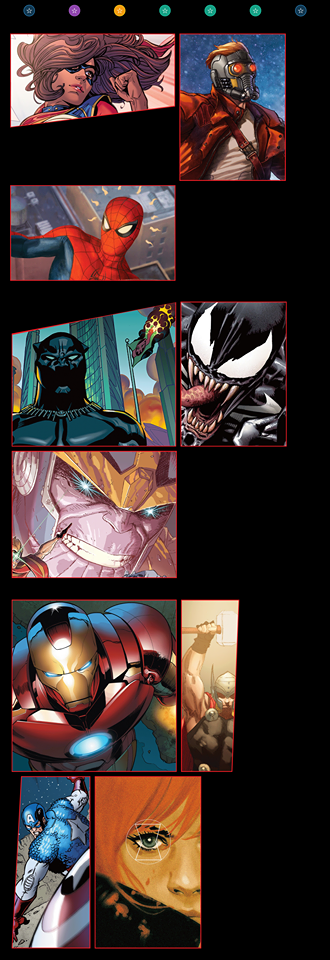
******

***FLOAT***

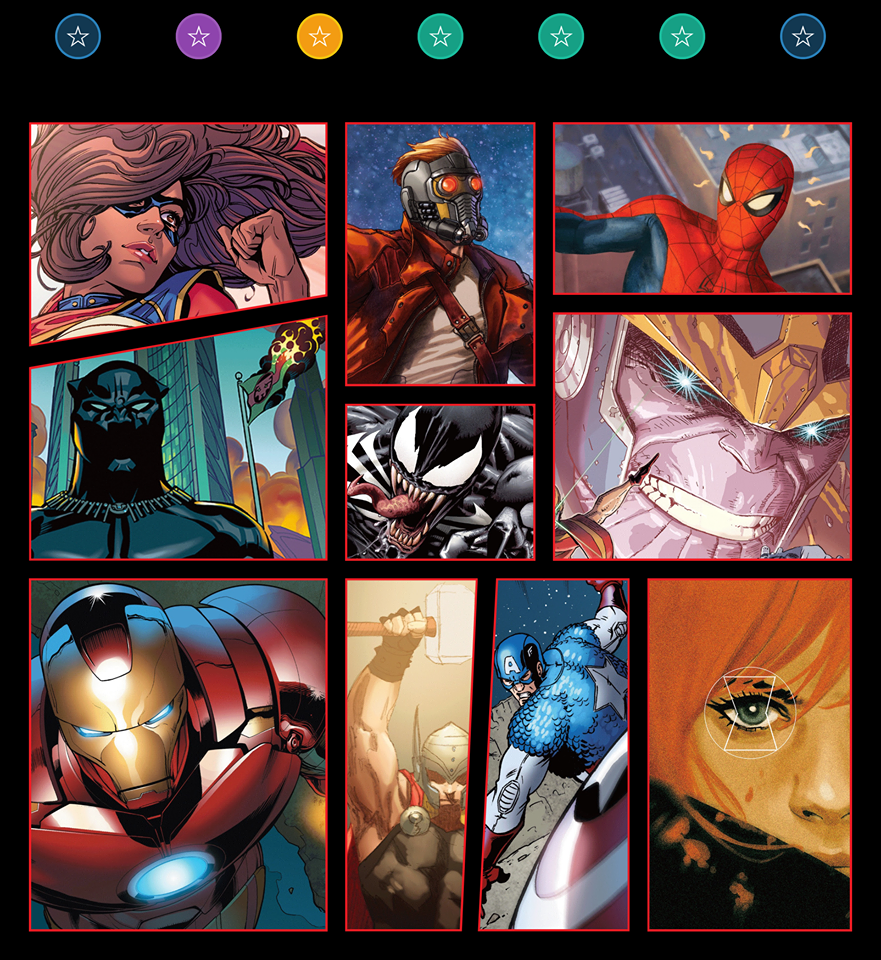
******

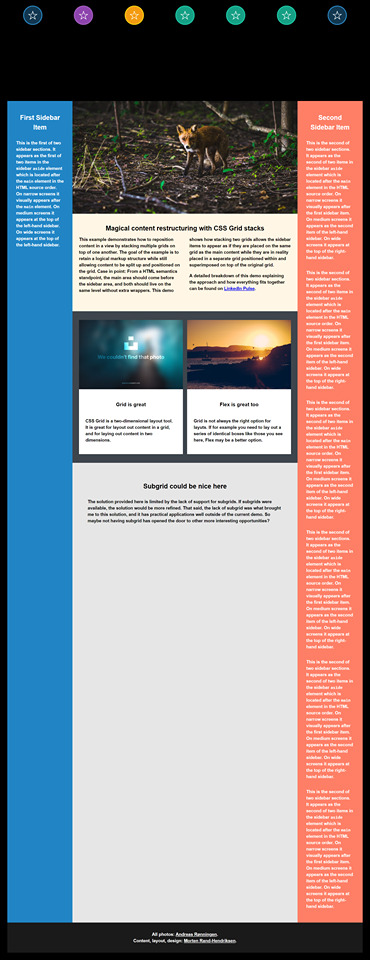
***FLEXBOX***

******

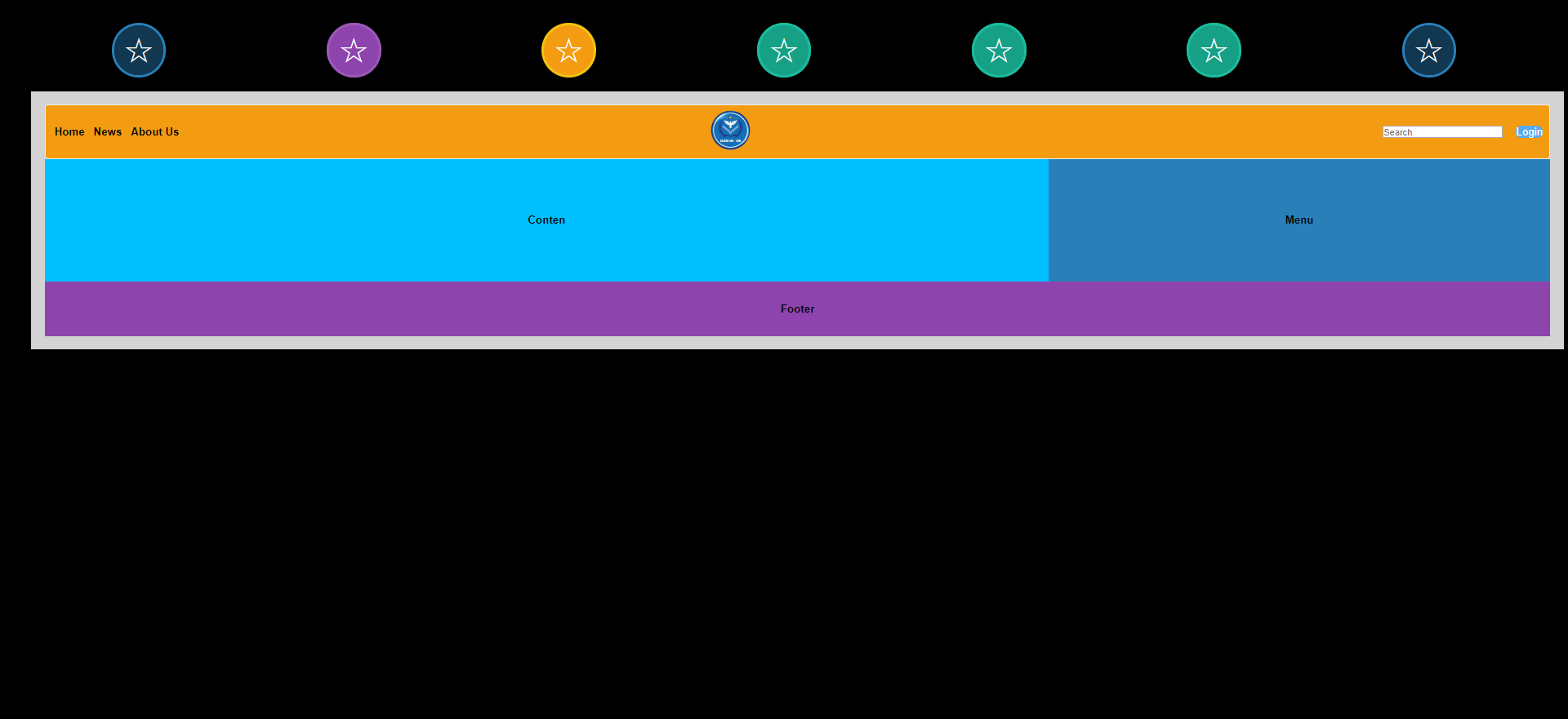
******

***CSS GRID***

******



***CSS GRID + FLEXBOX***

****

# Tài liệu tham khảo

<https://drafts.csswg.org/css-flexbox/>

<https://drafts.csswg.org/css-grid/>

<https://drafts.csswg.org/css-align/>

<http://grid-layout.com/history.html?fbclid=IwAR2crstK3qnxZUbKWLUZDD4hTqBign6KDelbmoa6KhSk9QPNXQg3LWWJmEE>

[https://www.smashingmagazine.com/2017/07/enhancing-css-layout-floats-flexbox-grid/](https://www.smashingmagazine.com/2017/07/enhancing-css-layout-floats-flexbox-grid/?fbclid=IwAR1C00rYOoVzVpDGNBs6rJ6-f5VMThPZcugOEu5kvkfwTW2nzY51xbvEagE)

[https://www.sitepoint.com/css-layouts-floats-flexbox-grid/](https://www.sitepoint.com/css-layouts-floats-flexbox-grid/?fbclid=IwAR1vjlD8sBsFRwKU9drYiPXX0ErKmhAXjdRc4SPtDEGhFySEbtV5SYCW83k)

[https://blog.karenmenezes.com/2014/apr/13/floats-inline-block-or-display-table-or-flexbox/](https://blog.karenmenezes.com/2014/apr/13/floats-inline-block-or-display-table-or-flexbox/?fbclid=IwAR2w_MXBh14SYjDi2Dd5-DAGg0GM4xjy0LTGEh_Bm53TlZPDzCpJ7mx2mgc)

[https://jurosh.com/blog/css-float-table-flex-grid](https://jurosh.com/blog/css-float-table-flex-grid/?fbclid=IwAR18yqWsaM9J0MuFRRC0nBBNNZSWO0zPPPTuAI__c0j5VA_UI_2miPzrkDM)

[https://www.youtube.com/watch?v=hYJvxsgnGMA](https://www.youtube.com/watch?v=hYJvxsgnGMA&fbclid=IwAR0hG6Qxj2h0bRwVb9R__BDtRJE2nReZbQEsAWHKCxezC71YKUpnh93E8Fs)