Bài tập Beginner – 3

Position:

Thuộc tính vị trí xác định loại phương thức định vị đước sử dụng cho một phần tử(tĩnh, tương đối, cố định, tuyệt đối)

Các position thuộc tính chỉ định loại phowng thức định vị được sử dụng cho một phàn tử

Có 5 giá trị vị trí khác nhau:

* static
* relative
* fixed
* absolute
* sticky

Các phần tử sau đó đưuọc định vị bằng các thuộc tính trên cùng, dưới cung, bên trái và bên phải. tuy nhiên, các thuộc tính này sẽ không hoạt động trừ khi thuộc Position tính đưuọc đặt đầu tiên. Họ cũng làm việc khác nhau tùy thuộc vào giá trị vị trí.

Vị trí: Static

Các phần tử HTML được định vị tĩnh theo mặc định.

Các yếu tố định vị tĩnh không bị ảnh hưởng bởi các thuộc tính trên, dưới, trái và phải.

Một yếu tố với position: static;không được định vị theo bất kỳ cách đặc biệt nào; nó luôn được định vị theo dòng chảy bình thường của trang:

vị trí: retative;

Một phần tử position: relative;được định vị so với vị trí bình thường của nó.

Đặt các thuộc tính trên cùng, bên phải, dưới cùng và bên trái của một yếu tố có vị trí tương đối sẽ khiến nó bị điều chỉnh khỏi vị trí bình thường. Các nội dung khác sẽ không được điều chỉnh để phù hợp với bất kỳ khoảng trống nào do phần tử để lại.

## vị trí: cố định;

Một phần tử có position: fixed;vị trí tương đối với chế độ xem, có nghĩa là phần tử này luôn ở cùng một vị trí ngay cả khi trang được cuộn. Các thuộc tính trên cùng, phải, dưới và trái được sử dụng để định vị phần tử.

Một yếu tố cố định không để lại một khoảng trống trong trang nơi nó thường được đặt.

Lưu ý phần tử cố định ở góc dưới bên phải của trang. Đây là CSS được sử dụng:

## vị trí: tuyệt đối;

Một phần tử có position: absolute;vị trí tương đối với tổ tiên được định vị gần nhất (thay vì vị trí tương đối với chế độ xem, như cố định).

Tuy nhiên; nếu một phần tử được định vị tuyệt đối không có tổ tiên định vị, nó sẽ sử dụng phần thân tài liệu và di chuyển cùng với cuộn trang.

**Lưu ý:** Phần tử "định vị" là phần tử có vị trí là bất cứ thứ gì ngoại trừ static.

Đây là một ví dụ đơn giản:

## vị trí: dính;

Một yếu tố với position: sticky;được định vị dựa trên vị trí cuộn của người dùng.

Một phần tử dính bật tắt giữa relativevà fixed, tùy thuộc vào vị trí cuộn. Nó được định vị tương đối cho đến khi một vị trí bù cho trước được đáp ứng trong chế độ xem - sau đó nó "đặt" đúng vị trí (như vị trí: cố định).

### Flexbox

### Flexbox là gì?

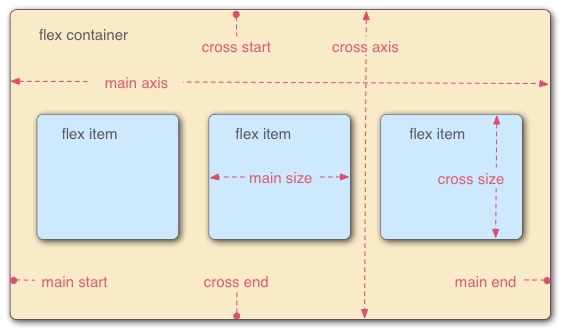
Flexbox là một kiểu dàn trang (layout mode) mà nó sẽ tự cân đối kích thước của các phần tử bên trong để hiển thị trên mọi thiết bị. Nói theo cách khác, bạn không cần thiết lập kích thước của phần tử, không cần cho nó float, chỉ cần thiết lập nó hiển thị chiều ngang hay chiều dọc, lúc đó các phần tử bên trong có thể hiển thị theo ý muốn.

Hiện nay, theo lời khuyên từ Mozilla thì chúng ta sử dụng Flexbox để thiết lập bố cục trong phạm vi nhỏ (ví dụ như những khung trong website) và khi thiết lập bố cục ở

### Thuật ngữ các thành phần trong Flexbox

Trước khi đi vào tìm hiểu sâu hơn về Flexbox, chúng ta cần nắm qua cấu trúc của Flexbox là như thế nào và một số thuật ngữ liên quan.

Dưới đây là sơ đồ cấu trúc Flexbox từ Mozilla Developer Network.

[](https://thachpham.com/wp-content/uploads/2016/06/flex_terms.jpg)

*Nguồn: Mozilla Developer Network*

Hai thành phần quan trọng nhất trong một bố cục Flexbox là gồm container và item:

* **container**: là thành phần lớn bao quanh các phần tử bên trong, bạn sẽ thiết lập kiểu hiển thị inline (sắp xếp theo chiều ngang) hoặc kiểu sắp xếp theo chiều dọc. Khi đó, các item bên trong sẽ hiển thị dựa trên thiết lập của container này.
* **item**: Các phần tử con của container được gọi là item, ở item bạn có thể thiết lập nó sẽ sử dụng bao nhiêu cột trong một container, hoặc thiết lập thứ tự hiển thị của nó.

Ngoài hai thành phần chính đó, chúng ta có thể thấy hình trên sẽ có:

* **main start**, **main end**: Khi thiết lập flexbox, điểm bắt đầu của container gọi là main start và điểm kết thúc được gọi là main end. Điều này có nghĩa, các item bên trong sẽ heienr thị từ main start đến main end (hoặc là được phép hiển thị đến main end). Và chiều vuông góc của nó là **cross start**, **cross end** cũng có ý nghĩa tương tự nhưng luôn vuông góc với main start, main end.
* **main axis**: Trục này là trục chính để điều khiển hướng mà các item sẽ hiển thị. Như bạn thấy ở trên hình main axis là trục dọc nên các item sẽ hiển thị theo chiều dọc, tuy nhiên ta có thể sử dụng thuộc tính flex-direction để thay đổi trục của main axis và lúc đó các item sẽ hiển thị theo nó. Và cross axis luôn là trục vuông góc của main axis.
* **main size**: Bạn có thể hiểu đơn giản là kích thước (chiều rộng hoặc dọc) của mỗi item dựa theo trục main axis.
* **cross size**: Là kích thước (chiều rộng hoặc dọc) của mỗi item dựa theo trục cross axis.

### Bắt đầu với Flexbox

Trước tiên mình bắt đầu với một cấu trúc đơn giản sau:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16 | <div class="container">         <div class="item item1">1</div>         <div class="item item2">2</div>         <div class="item item3">3</div>         <div class="item item4">4</div>      </div> |

Và một đoạn CSS ban đầu để thiết lập màu sắc và kích thước để dễ nhìn từng thành phần:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16 | /\*\* Global CSS \*\*/  .container {   background: red;   max-width: 960px;   max-height: 1000px;   margin: 0 auto;   padding: 5px;  }  .item {   background: blue;   margin: 5px;   color: white;   height: 50px;   text-align: center;   line-height: 50px;  } |

Bây giờ chúng ta sẽ bắt đầu làm việc với Flexbox ở đây. Trước tiên chúng ta sẽ đưa .container về hiển thị ở dạng flexbox với display: flex.

|  |  |
| --- | --- |
| 01  02  03  04 | /\*\* Flex layout \*\*/  .container {   display: flex;  } |

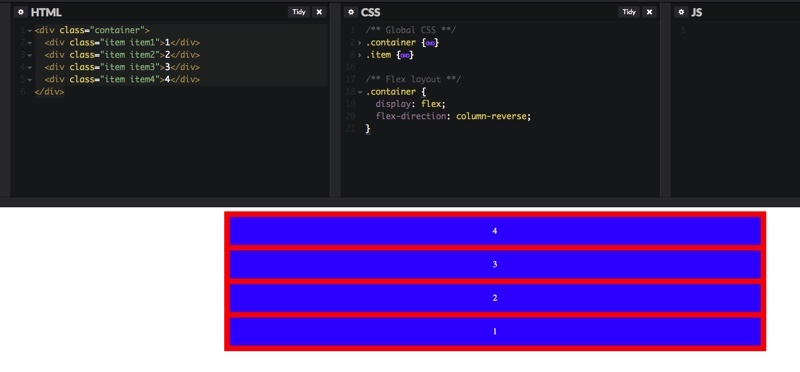
Bạn sẽ thấy các item bên trong đã tự hiển thị theo chiều dọc, tương ứng với trục main axis mặc định là chiều ngang.

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-01.jpg)

Nếu bạn muốn đổi trục thì chỉ cần thêm thuộc tính flex-direction vào container. Cụ thể:

|  |  |
| --- | --- |
| 01 | flex-direction: row | column | row-reverse | column-reverse |

* flex-direction:
  + row: Chuyển trục main axis thành chiều ngang, nghĩa là hiển thị theo hàng.
  + colum: Chuyển trục main axis thành chiều dọc, nghĩa là hiển thị theo cột.
  + row-reverse: Hiển thị theo hàng nhưng đảo ngược vị trí các item.
  + column-reverse: Hiển thị theo cột nhưng đảo ngược vị trí các item.

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-02.jpg)

Quá đơn giản đúng không?

#### flex-wrap

Bây giờ để hiểu cái này, chúng ta thử thêm chiều rộng cho mỗi item bên trong là 1000px xem chuyện gì sẽ xảy ra khi dùng flexbox.

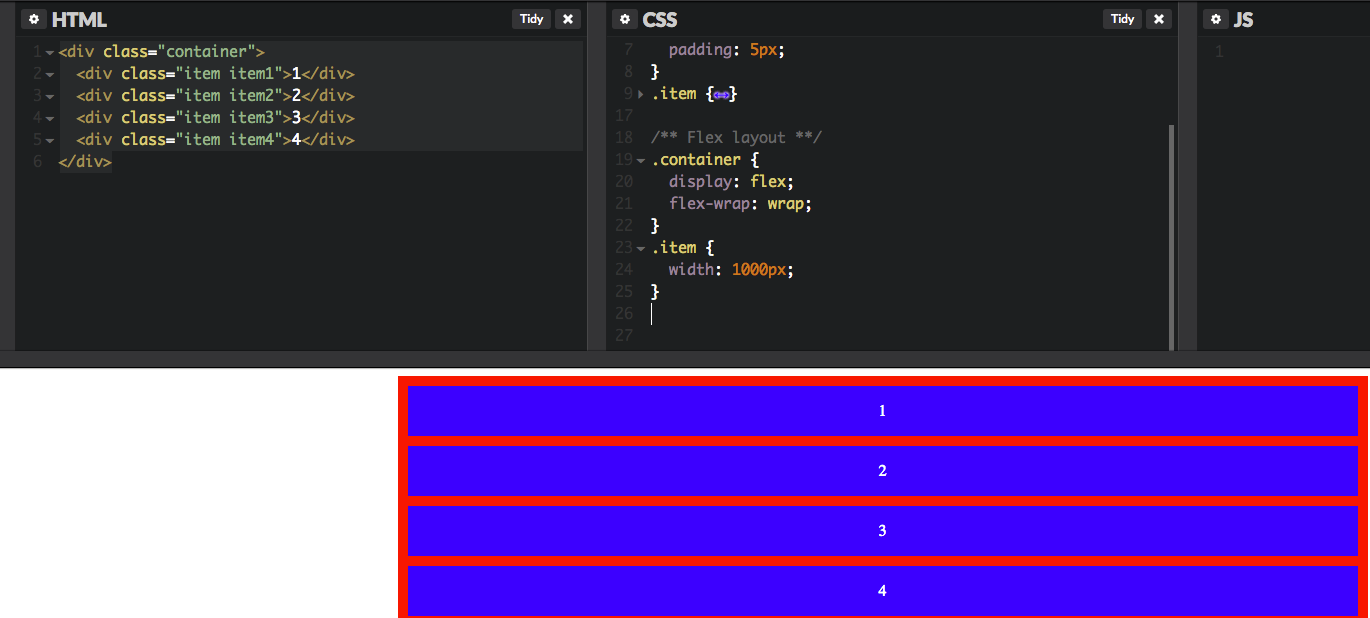
|  |  |
| --- | --- |
| 01  02  03 | .item {   width: 1000px;  } |

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-03.png)

Như bạn thấy, dù chúng ta có thêm chiều rộng cho mỗi item bên trong là 1000px nhưng nó vẫn hiển thị trên một hàng đều nhau. Lý do là mặc định, flexbox tự căn chỉnh các phần tử hiển thị đều nhau theo trục main axis của nó dựa theo chiều rộng của container. Vì vậy cho dù bạn có chỉnh chiều rộng vượt quá giới hạn của nó thì nó vẫn không bị nhảy lung tung.

Bây giờ hãy thử thêm thuộc tính flex-wrap: wrap vào container thử nhé.

|  |  |
| --- | --- |
| 01  02  03  04 | .container {   display: flex;   flex-wrap: wrap;  } |

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-04.png)

Nói nôm na là thuộc tính này cho phép container có thể bọc lại các item kể cả khi kích thước của item bị thay đổi, mặc định là nowrap. Thuộc tính này có thể áp dụng với cả chiều dọc của container và item. Các bạn xem video ở trên để rõ hơn.

#### order

Trong đoạn HTML ví dụ của mình trong bài này, mình có đặt số thứ tự cho mỗi phần tử là 1, 2, 3 và 4 với class tương ứng là .item1, .item, .item3 và .item4. Mặc định item này sẽ hiển thị theo thứ tự trong HTML, nhưng với thuộc tính order chúng ta có thể sắp xếp lại vị trí sắp xếp của các item.

Ví dụ mình có:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12 | .item1 {   order: 4;  }  .item2 {   order: 3;  }  .item3 {   order: 1;  }  .item4 {   order: 2;  } |

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-05.png)

Mặc định thứ tự sắp xếp sẽ bắt đầu từ bên trái qua phải, từ trên xuống dưới. Nếu bạn chỉnh lại trục của main axis với thuộc tính flex-direction nó sẽ thay đổi ngược lại.

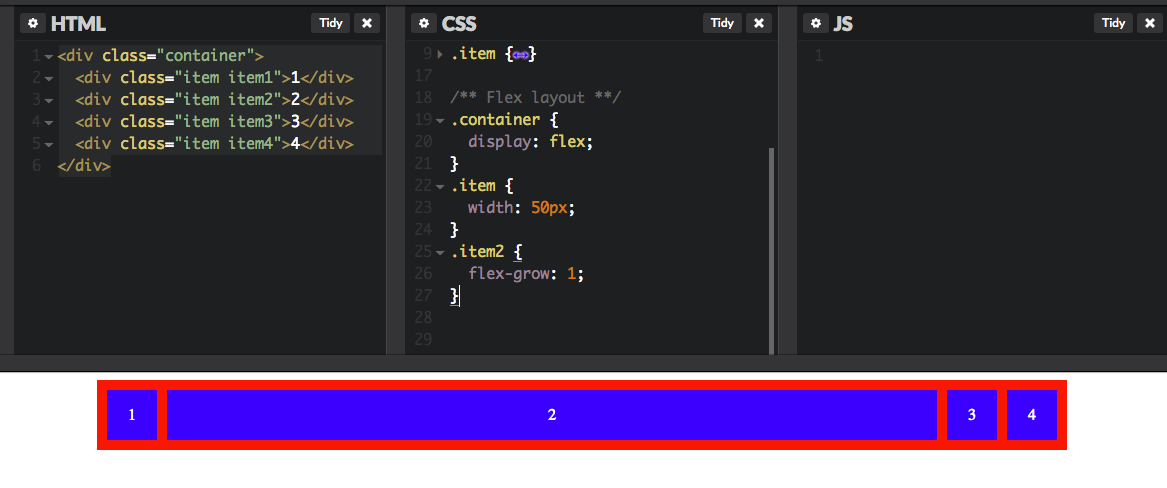
#### flex-grow

Để làm ví dụ này trước tiên mình hãy bỏ chức năng wrap đi và thiết lập chiều rộng của item là 50px.

|  |  |
| --- | --- |
| 01  02  03 | .item {   width: 50px;  } |

Bây giờ ở .item2, mình cho giá trị flex-grow là 1 thử nhé.

|  |  |
| --- | --- |
| 01  02  03 | .item2 {   flex-grow: 1;  } |

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-06.png)

Khi thiết lập nó flex-grow là 1, thì nó sẽ lấy phần trống còn lại của container đắp vào. Bây giờ hãy thử cho .item1 với flex-grow: 2 thử.

|  |  |
| --- | --- |
| 01  02  03 | .item1 {   flex-grow: 2;  } |

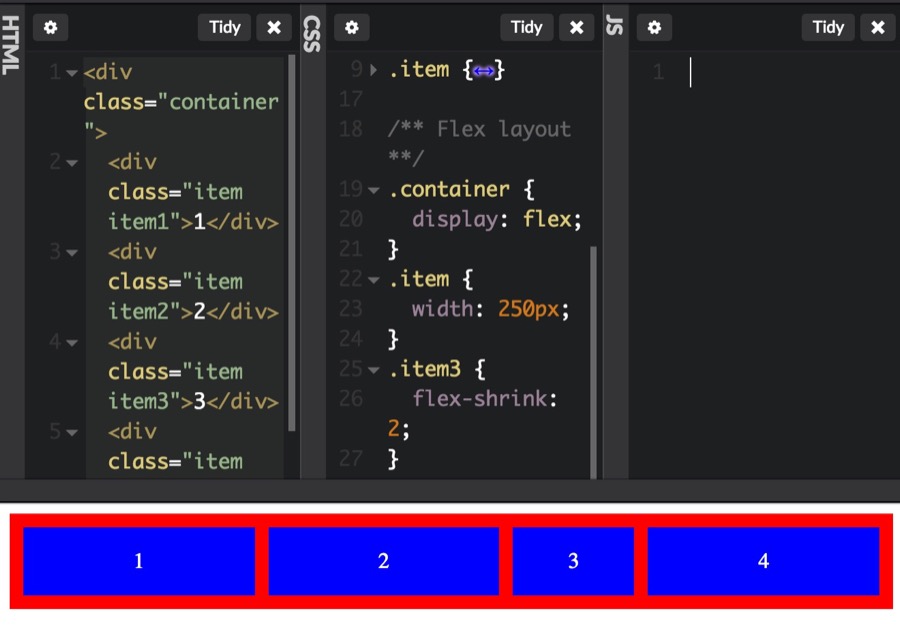
[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-07edited.jpg)

Lúc này giá trị flex-grow: 2 sẽ lấy phần dư lớn gấp đôi của flex-grow: 1.

#### flex-shrink

Bạn có thể hiểu mặc định tất cả các item đều có giá trị flex-shrink là 1. Điều này có nghĩa là khi chúng ta thu nhỏ trình duyệt lại, các phần tử đều co lại bằng nhau. Tuy nhiên giả sử mình muốn .item3 nó co lại nhiều hơn so với các item khác thì mình sẽ tăng giá trị flex-shrink của nó lên.

|  |  |
| --- | --- |
| 01  02  03 | .item3 {   flex-shrink: 2;  } |

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-08.jpg)

#### flex-basis

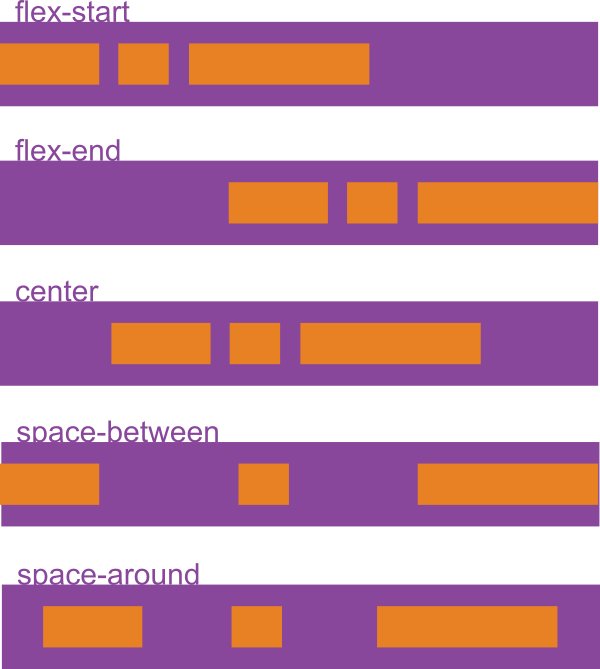
Cái này bạn có thể hiểu đơn giản nhất là gán cho item một kích thước nhất định. Bạn có thể sử dụng [giá trị tuyệt đối hoặc tương đối](https://thachpham.com/web-development/html-css/cac-don-vi-do-luong-trong-css.html) (căn cứ theo kích thước của container).

|  |  |
| --- | --- |
| 01  02  03 | .item3 {   flex-basis: 500px;  } |

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-09.jpg)

#### justify-content

Mặc định các item bên trong sẽ rải đều bắt đầu từ main start đến main end, tuy nhiên nếu container vẫn còn khoảng trống thì có thể dùng thuộc tính justify-content để điều chỉnh lại vị trí bắt đầu của nó.

Thuộc tính này có 5 giá trị và bạn có thể xem tấm ảnh bên dưới mình mượn của CSS Tricks để hiểu hơn về ý nghĩa các giá trị của justify-content.[](https://thachpham.com/wp-content/uploads/2016/06/flexbox-justify-content.jpg)

Đây là ví dụ:

|  |  |
| --- | --- |
| 01  02  03  04 | .container {   display: flex;   justify-content: flex-end;  } |

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-10.jpg)

Trong bài này mình chỉ đề cập tới một số thuộc tính hay dùng trong Flexbox thôi, bạn có thể xem thêm danh sách thuộc tính Flexbox [**tại đây**](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout).

phạm vi lớn hơn (như chia cột website) thì vẫn nên sử dụng kiểu thông thường là dàn trang theo dạng lưới (grid layout).