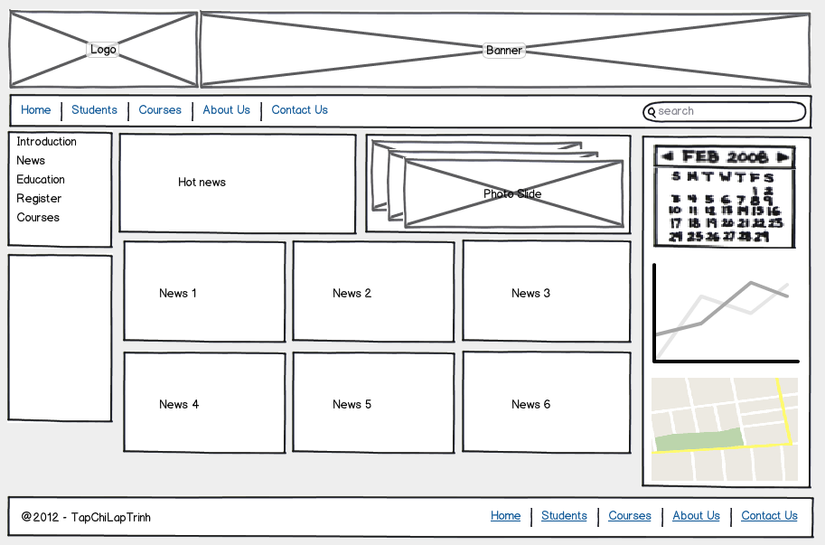
**Layout**

Layout là gì ? Tại sao chúng ta cần phải xây dựng layout trong lập trình web ?

Từ layout được sử dụng trong nhiều lĩnh vực như thiết kế sách, báo, tạp chí,… Layout ở đây có nghĩa là bố cục, cách sắp xếp, trình bày các bộ phận, chi tiết nằm trong bản thiết kế đó bao gồm việc căn chỉnh tỷ lệ, khoảng cách và lựa chọn vị trí cho các thành phần. Tương tự như vậy, trong thiết kế web, layout chính là cách dàn trang, sắp xếp các yếu tố sẽ được hiển thị trên website. Chẳng hạn bạn cần thiết kế một trang có giao diện được thiết kế với layout như sau:



## **I. Các kỹ thuật xây dựng layout:**

### **1. Xây dựng layout với table:**

Cách đơn giản nhất để tạo ra các layout là sử dụng thẻ <table> trong HTML. Những bảng này sắp xếp các dữ liệu vào các cột và hàng, vì thế bạn có thể lợi dụng hay sử dụng những hàng và cột này theo cách bạn muốn mà không cần sử dụng quá nhiều css. Bạn có thể thiết kế trang của bạn thành nhiều cột với các phần nội dung khác nhau. Bạn có thể giữ nội dung chính trong cột giữa và cột trái làm cột chứa menu, và cột phải dùng để đặt các quảng cáo. Loại layout này tương tự như cách chúng tôi bố trí trang web hiện tại. Ví dụ :

<!DOCTYPE html>

<html>

<head>

<title>Vi du Layout table</title>

</head>

<body>

<table border="0" cellpadding="0" cellspacing="0" width="100%">

<tr>

<td>

<table align="center" border="0" cellpadding="0" cellspacing="0" width="900" style="border-collapse: collapse;">

<tr>

<td bgcolor="#70bbd9">

<table border="1" cellpadding="0" cellspacing="0" width="100%" >

<tr>

<td>

<table border="1" cellpadding="0" cellspacing="0" width="100%" >

<tr>

<td width="25%" valign="top" align="center" style="padding: 30px;">

LOGO

</td>

<td width="75%" valign="top" align="center" style="padding: 30px">

BANNER

</td>

</tr>

</table>

</td>

</tr>

<tr>

<td align="center" style="padding: 10px;">

HEAD-LINK

</td>

</tr>

</table>

</td>

</tr>

<tr>

<td bgcolor="#ffffff">

<table border="1" cellpadding="0" cellspacing="0" width="100%" >

<tr>

<td width="20%" valign="top" align="left" style="padding: 150px 30px; text-align: center;">

Left

</td>

<td width="60%" valign="top" align="center" style="padding: 150px 30px">

Content

</td>

<td width="20%" valign="top" align="right" style="padding: 150px 30px; text-align: center;">

Right

</td>

</tr>

</table>

</td>

</tr>

<tr>

<td bgcolor="#ee4c50" style="padding: 30px; text-align: center;" >

Footer

</td>

</tr>

</table>

</td>

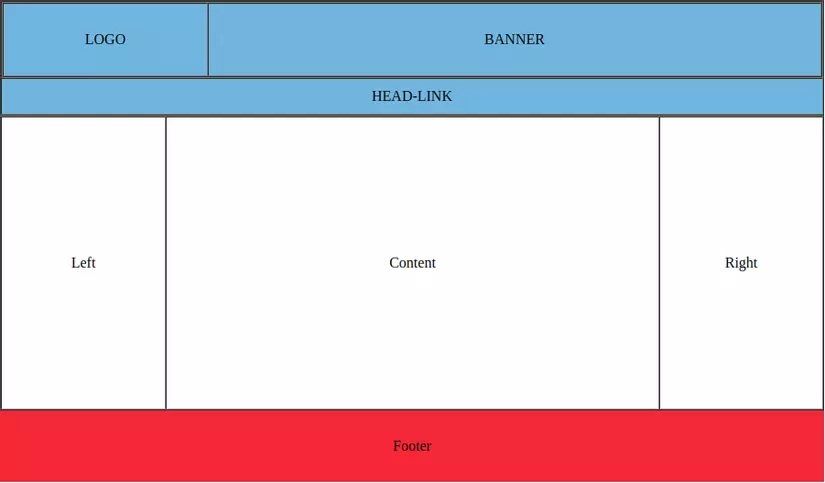
</tr>

</table>

</body>

</html>

Kết quả khi chạy đoạn code trên là:



Tuy nhiên table lại bộc lộ khá nhiều nhược điểm khi sử dụng làm layout cho một trang web có cấu trúc như trên. Các nhược điểm có thể gặp phải với table với những dạng layout này là chậm, khó tùy chỉnh và khó kết hợp với CSS Javascript để tạo lên sự linh hoạt cho trang web.

### **2. Xây dựng layout với div + css:**

### **\* Sử dụng float và clear:**

Việc chia cột trong CSS là việc bạn thiết lập những phần tử con trong một phần tử mẹ nằm trên cùng một hàng.

Ví dụ, mình muốn phần nội dung website của mình có hai cột thì mình sẽ tạo ra 3 cái

, một cái <div> mình gọi nó là container hoặc phần tử mẹ, hai cái <div> còn lại mình gọi là column (cột). Vậy bây giờ nhiệm vụ của chúng ta là làm thế nào để các phần tử con có thể được chia thành 2 cột như vậy.

Các bước chia cột:

1. Các cột phải luôn có một container, tức là phần tử mẹ bao bọc nó.
2. Thiết lập chiều rộng cho container.
3. Thiết lập chiều rộng cho hai cột, tổng chiều rộng trong hai cột phải luôn bằng hoặc ít hơn chiều rộng của container.
4. Nên sử dụng box-sizing: border-box để tính toán kích thước chính xác.
5. Sử dụng thuộc tính float với giá trị left và right để đẩy phần tử về sang trái hoặc phải.
6. Tiến hành clear float.

Ví dụ :

<header>

<div id="head">

<div class="logo">logo</div>

<div class="banner">banner</div>

</div>

<div id="head-link">head-link</div>

</header>

<section class="content">

<div id="left">left</div>

<div id="content">content</div>

<div id="right">right</div>

</section>

<footer>footer</footer>

#head {

width: 100%;

overflow: auto;

}

.logo {

width: 20%;

background-color: aquamarine;

float: left;

box-sizing: border-box;

padding: 30px;

}

.banner {

width: 80%;

background-color: aqua;

float: right;

box-sizing: border-box;

padding: 30px;

}

#head-link {

background-color: blue;

padding: 10px;

}

.content::after {

content: '';

clear: both;

display: table;

}

#left, #content, #right {

width: 15%;

float: left;

box-sizing: border-box;

padding: 150px 30px;

background-color: antiquewhite;

}

#content {

width: 70%;

background-color: aliceblue;

}

#right {

background-color: wheat;

}

footer {

padding: 30px;

background-color: brown;

}

Kết quả sau thực chạy code như sau:



Nguyên lý hoạt động của thuộc tính float:

1. Khi một phần tử được thiết lập thuộc tính float:
   * Nó sẽ được bắt đầu ở hàng phía trên , nếu hàng phía trên còn đủ chỗ trống để chứa nó.
   * Nó sẽ được bắt đầu ở hàng mới, nếu hàng phía trên không đủ chỗ trống để chứa nó.

Lưu ý: Nếu một phần tử được thiết lập thuộc tính float mà trong khi phần tử đứng trước nó không được thiết lập thuộc tính float, thì mặc định nó được bắt đầu ở hàng mới.

1. Khi hàng không đủ chỗ chứa phần tử thì phần tử phải bắt đầu ở hàng mới. Tuy nhiên, cách bắt đầu như thế nào mới là quan trọng. Khi trên một hàng có nhiều phần tử được thiết lập thuộc tính float và mỗi phần tử có chiều cao khác nhau, nếu hàng không đủ chỗ chứa phần tử thì phần tử sẽ bắt đầu bên cạnh phần tử có chiều cao thấp nhất và còn đủ khoảng trống để chứa nó.

### **\* Sử dụng Flexbox:**

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.

Thành phần quan trọng nhất của Flexbox là:

* container: là thành phần lớn bao quanh các phần tử bên trong, các item bên trong sẽ hiển thị dựa trên thiết lập của container này.
* item: là phần tử con của container, bạn có thể thiết lập nó sẽ sử dụng bao nhiêu cột trong một container, hoặc thiết lập thứ tự hiển thị của nó.

Một số thuộc tính cơ bản:

* Dùng display: flex; để tạo ra một flex container.
* Dùng justify-content để căn ngang các items.
* Dùng align-items để căn dọc các items.
* Dùng flex-direction nếu muốn các items theo hướng chiều dọc chứ không phải ngang.
* Dùng row-reverse hoặc column-reverse để đảo ngược thứ tự mặc định.
* Dùng order để tùy chỉnh thứ tự một item cụ thể.
* Dùng align-self để căn dọc một item cụ thể.
* Dùng flex để tạo ra một flexible boxes có thể stretch và shrink.

Ví dụ :

<div class="wrapper">

<header class="header-flex-2">Header</header>

<article class="main">

<p>Mô-đun Flexbox Layout (Flexible Box) (Theo khuyến nghị của W3C kể từ tháng 10 năm 2017) nhằm mục đích cung cấp một cách bố trí, sắp xếp và phân phối không gian hiệu quả hơn các item trong trong một container, ngay cả khi kích thước của chúng không xác định hoặc động ( Do đó có từ 'flex').</p>

</article>

<aside class="aside aside-1">Aside 1</aside>

<aside class="aside aside-2">Aside 2</aside>

<footer class="footer-flex-2">Footer</footer>

</div>

.wrapper {

display: flex; /\* kich hoat flex box \*/

flex-flow: row wrap;

font-weight: bold;

text-align: center;

}

.wrapper > \* {

padding: 10px;

flex: 1 100%; /\*cho tất cả phần tử bên trong có độ dài 100% và tỉ lệ chiếm không gian trống là như nhau\*/

}

.header-flex-2 {

background: tomato;

}

.footer-flex-2 {

background: lightgreen;

order: 4;

}

.main {

text-align: left;

background: deepskyblue;

height: 400px;

flex: 3 0px; /\* cho phần nội dung main ở giữa chiếm 3 phần không gian trống so với 2 phần aside bên cạnh \*/

order: 2;

}

.aside { flex: 1 0 0; } /\* 2 phần aside sẽ chỉ chiếm 1 phần không gian \*/

.aside-1 {

background: gold;

height: 400px;

order: 1;

}

.aside-2 {

background: hotpink;

height: 400px;

order: 3;

}

Kết quả khi thực hiện :



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

Để tìm hiểu rõ hơn về flex-box thì các bạn có thể xem thêm [tại đây.](https://css-tricks.com/snippets/css/a-guide-to-flexbox/)

### **\* Sử dụng Grid:**

Grid là một module tạo bố cục website trong CSS bằng cách hỗ trợ hệ thống bố cục theo dạng lưới 2 chiều, gồm hàng và cột. Trước khi có Grid, bạn có thể tạo giao diện website theo nhiều kiểu khác nhau như sử dụng thẻ div, sử dụng bảng biểu (table).

Grid ra đời nhằm đơn giản hóa việc xây dựng giao diện website và hoạt động rất tốt với Flexbox. Flexbox cũng là 1 module hỗ trợ xây dựng bố cục nhưng áp dụng với các bố cục một chiều đơn giản. Khi Grid và Flexbox kết hợp với nhau, lập trình viên có thể tạo ra nhiều bố cục website phức tạp và đa dạng hơn.

Grid cho phép bạn tạo một ma trận bố cục 2 chiều gồm các dòng và các cột. Ở mỗi dòng, cột và mỗi phần tử trong Grid bạn có thể chỉnh sửa style vì vậy Grid cũng rất thích hợp để tạo bố cục trang Web. Ngoài ra, các bảng biểu, các trò chơi có dạng bàn cờ như ca rô, cờ vua,… để rất thích hợp dùng Grid để xây dựng giao diện.

Bốn bước cơ bản để tạo layout bằng grid:

* Tạo một thành phần container, và định nghĩa nó là display: grid;.
* Sử dụng container đó để định nghĩa các grid track sử dụng các thuộc tính grid-template-columns và grid-template-rows.
* Đặt các thành phần con bên trong container.
* Thiết lập nơi mà mỗi phần tử con thuộc về trong grid bằng cách định nghĩa grid-column và grid-row của nó.

Ví dụ :

<div class="grid-container">

<div class="grid-item header">

<h3>Phần Header </h3>

</div>

<div class="grid-item sidebar-1">

Phần Sidebar 1 -<br />

Phần Sidebar 1 -<br />

Phần Sidebar 1 -<br />

Phần Sidebar 1 -<br />

</div>

<div class="grid-item content">

Phần nội dung Web <br />

Phần nội dung Web <br />

Phần nội dung Web <br />

Phần nội dung Web <br />

Phần nội dung Web <br />

Phần nội dung Web <br />

Phần nội dung Web <br />

Phần nội dung Web <br />

Phần nội dung Web <br />

</div>

<div class="grid-item sidebar-2">Phần Siderbar 2</div>

<div class="grid-item footer">Phần Footer</div>

</div>

.grid-container {

display: grid;

grid-template-areas:

'header header header header header'

'sidebar-1 content content content sidebar-2'

'footer footer footer footer footer';

text-align: center;

}

.header {

padding: 5px;

grid-area: header;

width: 100%;

height: 70px;

background-color: yellow;

box-sizing: border-box;

}

.sidebar-1 {

padding: 5px;

grid-area: sidebar-1;

background-color: pink;

}

.sidebar-2 {

padding: 5px;

grid-area: sidebar-2;

background-color: lightgreen;

}

.content {

padding: 5px;

grid-area: content;

background-color: #BB8FCE;

}

.footer {

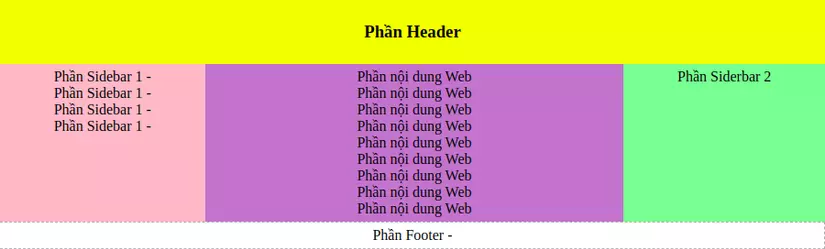
padding: 5px;

grid-area: footer;

border: 1px dashed #AAA;

}

Kết quả thực hiện được :



Để tìm hiểu kỹ hơn về grid layout thì các bạn có thể tham khảo thêm [tại đây.](https://css-tricks.com/snippets/css/complete-guide-grid/)

### **\* Sử dụng Framework (Bootstrap):**

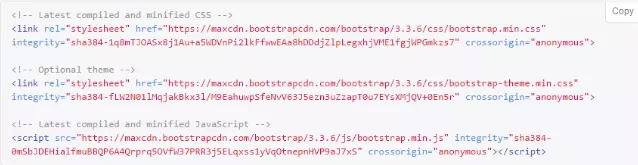
Bootstrap là một framework HTML, CSS, và JavaScript cho phép thiết kế phát triên responsive web mobile. Bootstrap là một framework cho phép thiết kế website reponsive nhanh hơn và dễ dàng hơn Bootstrap là bao gồm các HTML templates, CSS templates và Javascript tao ra những cái cơ bản có sẵn như: typography, forms, buttons, tables, navigation, modals, image carousels và nhiều thứ khác. Trong bootstrap có thêm các plugin Javascript trong nó. Giúp cho việc thiết kế reponsive của bạn dễ dàng hơn và nhanh chóng hơn.

Để sử dụng được bootstrap thì làm thế nào ?

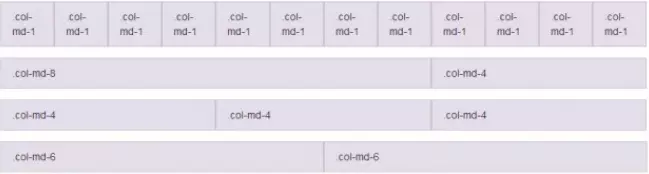
Có 2 cách để bạn có thể sử dụng Bootstrap trên web của bạn.

Download Bootstrap từ [getbootstrap.com](http://getbootstrap.com/)

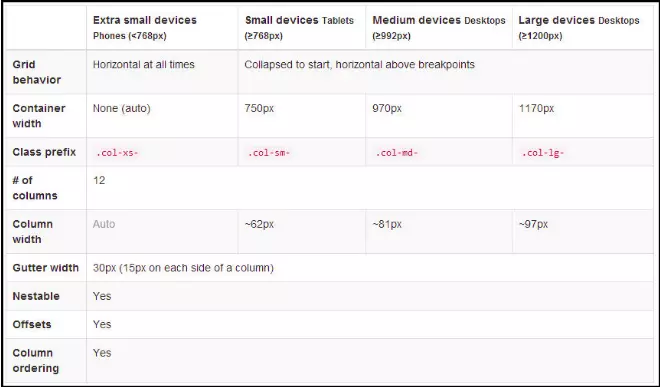
Thêm Bootstrap từ CDN



Một số lưu ý khi sử dụng Bootstrap:

* Class container: được fix sẵn độ rộng tuỳ theo độ phân giải, padding sang 2 bên 15px, thẻ này bọc hầu hết tất cả thẻ div ở sau.
* Class container-fluid: giống như thẻ container nhưng độ rộng của nó là full màn hình.
* Class row: thẻ này có margin là -15px nếu bạn muốn thẻ nào đó sát lề 2 bên của container.
* Giao diện của bootstrap là một dạng lưới được chia làm 12 cột và được đặt trong một class row.
* 

Trong đó các col- sẽ được hiểu như sau:



Ví dụ :

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap 4 Website Example</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>

<style>

.fakeimg {

height: 200px;

background: #aaa;

}

</style>

</head>

<body>

<div class="jumbotron text-center" style="margin-bottom:0">

<h1>My First Bootstrap 4 Page</h1>

<p>Resize this responsive page to see the effect!</p>

</div>

<nav class="navbar navbar-expand-sm bg-dark navbar-dark">

<a class="navbar-brand" href="#">Navbar</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#collapsibleNavbar">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="collapsibleNavbar">

<ul class="navbar-nav">

<li class="nav-item">

<a class="nav-link" href="#">Link</a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">Link</a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">Link</a>

</li>

</ul>

</div>

</nav>

<div class="container" style="margin-top:30px">

<div class="row">

<div class="col-sm-4">

<h2>About Me</h2>

<div class="fakeimg">Fake Image</div>

<p>Some text about me in culpa qui officia deserunt mollit anim..</p>

</div>

<div class="col-sm-8">

<h2>TITLE HEADING</h2>

<h5>Title description, Dec 7, 2017</h5>

<div class="fakeimg">Fake Image</div>

<p>Some text..</p>

<p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco.</p>

</div>

</div>

</div>

<div class="jumbotron text-center" style="margin-bottom:0">

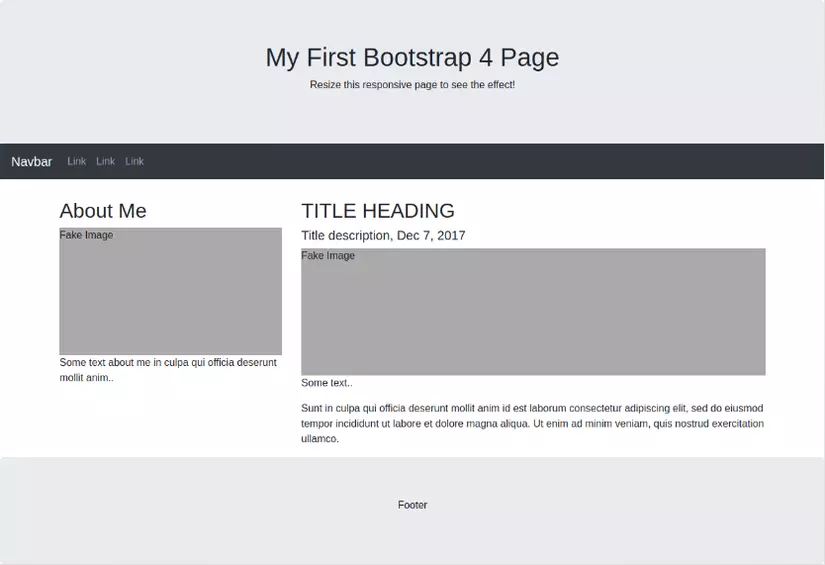
<p>Footer</p>

</div>

</body>

</html>

Kết quả thực hiện:



Chúng ta có thể thấy được việc sử dụng bootstrap tiết kiệm được rất nhiều thời gian để tạo ra 1 layout đơn giản như vậy.

Nếu bạn muốn tìm hiểu sâu hơn về bootstrap thì có thể xem thêm [tại đây.](https://www.w3schools.com/bootstrap4/)

### **3. So sánh giữa các kĩ thuật :**

### **Giữa việc dùng table và div thì cách nào tiện hơn ?**

Thẻ table/bảng : Là thẻ giúp chúng ta tạo nên một khu vực thuộc dạng bảng tính, kích thước về chiều ngang thường luôn cố định, cấu trức vững chắc nhưng khó tùy biến, khó kết hợp với các thẻ cấu trúc hoặc các thẻ table khác. Tốn nhiều dung lượng khi load trang. Vì table khi tải xong sẽ tự tính, nên quá trình các nội dung chưa tải về thì giao diện sẽ hiển thị khác (ví dụ như dùng Ajax tải nội dung chẳng hạn) còn khi nội dung tải về và được hiển thị, giao diện lại “nhảy nhảy” khác, làm cho user khó chịu.

Thẻ Div(divison)/Khu vực: Là thẻ giúp chúng ta tạo nên một khu vực dạng block, kích thước không cố định khi chưa thiết lập thông số, cấu trúc mềm dẻo, dễ dàng tùy biến, dễ dàng kết hợp với các thẻ khác. Tốn ít dung lượng khi load trang. Chỉ cần style để div có hình dạng, sau khi dữ liệu đổ về, thì nó chỉ đổ vào chỗ developer quy định chứ không nhảy lung tung như table.

Grid

Với Grid, nó dễ dàng thiết lập toàn bộ hệ thống thiết kế bố trí cho trang web của bạn. Grid còn có thể lồng layout này trong các layout khác và grid cũng cho phép bạn tạo các sắp xếp cực kỳ phức tạp với độ chính xác. Tuy nhiên grid không phải là tốt nhất để căn giữa nhanh hoặc đặt một hình ảnh trong một bài viết văn bản lớn hơn. Những kiểu bố trí này được xử lý tốt nhất thông qua các phương pháp khác.

Flex

Flex bao gồm nhiều thuộc tính phụ có thể đặc biệt hữu ích. Ví dụ, flex-grow và flex-shrink cho phép các cột thay đổi chiều rộng và tỷ lệ tùy thuộc vào chế độ xem của người dùng. justify-content cho phép các mục cách đều nhau hoặc căn giữa một cách hoàn hảo. Đây là một công cụ tuyệt vời để cho phép các thành phần UI nhỏ hơn trông đẹp nhất trên mọi thiết bị. Tuy nhiên, thật không may nếu bạn muốn bố trí các phần tử lớn hơn trên trang, có lẽ bạn sẽ muốn thêm lề và sau đó thực hiện một phép toán nhỏ để đảm bảo nó trông giống như bạn muốn. Điều đó có thể mất thời gian. Ngoài ra, mỗi phần trong thiết kế của bạn phải là một hàng có thẻ flex riêng biệt trên đó.

Float

Float & Clear là quá quen thuộc rồi. Đây là một cách tuyệt vời để đặt một hình ảnh trong một bài viết văn bản lớn hơn.

Tuy nhiên, nó không phải là một cách rất thiết thực để thiết lập toàn bộ bố cục trang web. Nếu bạn không cẩn thận, nó có thể gặp lỗi khá nhanh, và nó có một chút lộn xộn khi cố gắng dọn dẹp nó. Phương thức bố trí này không có nhiều thuộc tính tích hợp như Grid hoặc Flex và không thể tạo bố cục phức tạp.

Framework

Nếu bạn là một người không hiểu quá nhiều về CSS mà muốn tạo ra một layout đơn giản dễ dùng. Thì framework chắc chắn là cách tốt nhất. Vì bạn chỉ cần những dòng HTML và các class của framework thì bạn đã có thể tạo ra những layout cơ bản và đủ dùng. Tuy nhiên có một số điều bạn cần chú ý là mỗi hàng phải là thẻ <div> riêng. Vì vậy nó sẽ có nhiều <div> nên nó rất là loằng ngoằng và phức tạp nó lại càng phức tạp hơn khi có thêm nhiều class để responsive.



Vậy cái gì là tốt nhất ?

Định nghĩa tốt nhất của tôi sẽ là phương pháp bố trí yêu cầu số lượng dòng ít nhất trong css để đạt được bố cục phức tạp nhất. Grid chắc chắn là thắng. Tuy nhiên, tôi nghĩ rằng có một nơi dành cho tất cả các phương pháp bố trí, và chúng có thể được kết hợp với nhau khi cần thiết. Khi tạo một trang web HTML / CSS tùy chỉnh, tôi khuyên bạn nên sử dụng grid để bố trí trang web của mình, float để thêm hình ảnh đó vào bài đăng trên blog của bạn và flex để có được yếu tố UI pop-up nhỏ bé đó trông thật tuyệt vời!

### **4. Các lưu ý trong quá trình xây dựng layout website :**

Bắt đầu với nội dung: là thành phần quan trọng nhất của một webpage: tiêu đề, các đoạn văn, những bức ảnh chính, link điều hướng, video,... và tất cả những gì mà đưa người dùng đến website của bạn. Họ muốn đọc, học và trải nghiệm những gì website của bạn đề cập tới. Nội dung chính là "vua", vậy bạn nên nghĩ đến việc website sẽ hiển thị những nội dung gì trước khi nghĩ đến việc nó nên trình bày với màu sắc, font chữ nào.

Mobile first: ngày nay smartphone và máy tính bảng đã trở nên rất phổ biến, vì vậy việc trích xuất những key cần hiển thị để phù hợp cho các màn hình cỡ nhỏ mà không làm mất đi nội dung trang web là một điều hết sức cần thiết.

Xây dựng bản phác thảo: trước khi nhảy vào code HTML, bạn nên phác thảo cách đặt nội dung của mình vào các vị trí trên page trước. Vì web design là đặt các nội dung vào một box (các thẻ div hoặc các thẻ HTML khác) và đặt các box đó vào các vị trí trên page. Vậy nên hãy phát thảo các box, sắp xếp chúng nhanh trên giấy là cách tốt nhất để chúng ta thử các kiểu page layout khác nhau.

Sử dụng background image: thuộc tính background-image cung cấp một cách khác để thêm ảnh vào làm ảnh nền cho element thay cho tag <img>. Thuộc tính này không chỉ để tiết kiệm một chút bytes khi tải HTML mà còn làm đơn giản đi một số khó khăn khi tạo layout.

Chia nhỏ layout: khi layout đủ lớn, nếu không chia nhỏ công việc từ đoạn thiết kế đến coding thì bạn có thể sẽ tạo nên một mớ hỗn độn. Vậy nên việc chia nhỏ layout để tiếp tục thiết kế và coding sau khi tạo ra một bản phác thảo tổng thể là rất quan trọng.

Margin và padding: không phải lúc nào bạn cũng cần các kỹ thuật CSS phức tạp để di chuyển một element vào một vị trí cụ thể. Hãy lưu ý rằng padding và margin chỉ là các khoảng trống, vậy nên khi sử dụng các thuộc tính này bạn có thể di chuyển các element vào đúng vị trí.

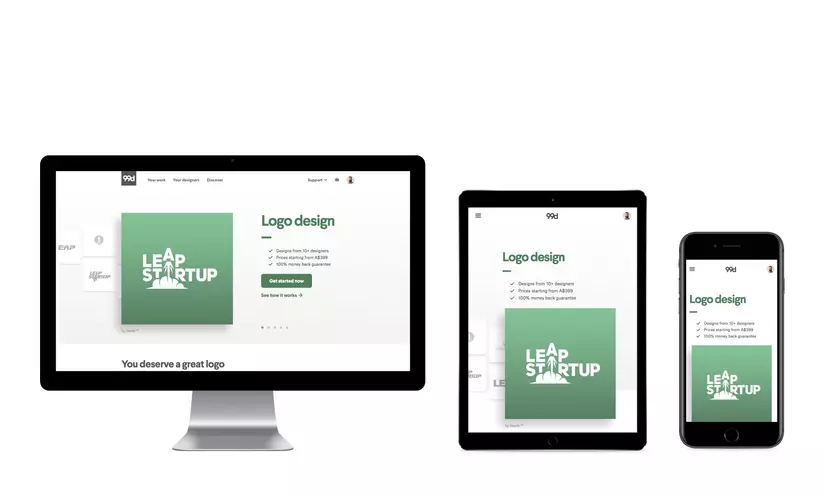
## **II. Responsive website**

Trong việc xây dựng layout cho một trang web thì việc responsive cho nó thật sự rất cần thiết. Như các bạn đã biết, với tốc độ phát triển của các thiết bị công nghệ cầm tay như: iPhone, iPad, MacBook, Laptop … Và phát triển ở mọi ứng dụng cũng như hệ điều hành rất đa dạng và phong phú như: Android, iOS, Windows phone … Và các bạn biết là chúng ta mua điện thoại hay các thiết bị đó về sử dụng để có thể nghe nhạc, xem phim, chơi game … và một tính năng quan trọng là để lướt web.

### **1. PC first**

PC first là khái niệm để chỉ tuần tự responsive giao diện từ màn hình to xuống màn hình nhỏ.

Để làm việc với mô hình này chúng ta sử dụng max-width trong media query.



Dưới đây là các media query điển hình:

/\*Ipad ngang(1024 x 768)\*/

@media screen and (max-width: 1024px){

}

/\*Ipad dọc(768 x 1024)\*/

@media screen and (max-width: 768px){

}

/\*Tablet nhỏ(480 x 640)\*/

@media screen and (max-width: 480px){

}

/\*Iphone(480 x 640)\*/

@media screen and (max-width: 320px){

}

/\*Smart phone nhỏ\*/

@media screen and (max-width: 240px){

}

Theo cách này thì khi một Selector trong css cần style đi qua từ màn hình to đến nhỏ sẽ được thay đổi theo thứ tự ưu tiên.

### **2. Mobile first**

Trái ngược với PC first thì tuần tự tiến trình responsive của chúng ta xuất phát từ màn hình nhỏ và xây dựng dần lên thiết bị có kích thước to.



Để làm việc với mô hình này chúng ta sử dụng tham số min-width trong media query:

/\*Smart phone nhỏ\*/

@media screen and (min-width: 240px){

}

/\*Iphone(480 x 640)\*/

@media screen and (min-width: 320px){

}

/\*Tablet nhỏ(480 x 640)\*/

@media screen and (min-width: 480px){

}

/\*Ipad dọc(768 x 1024)\*/

@media screen and (min-width: 768px){

}

/\*Ipad ngang(1024 x 768)\*/

@media screen and (min-width: 1024px){

}

Với mô hình này thì khi style Css cho cùng một đối tượng thì theo thứ tự được ưu tiên từ màn hình nhỏ đến to, càng về sau các các Css ở các query màn hình to hơn được ưu tiên hơn.

### **3. Các nguyên tắc cơ bản nên áp dụng khi responsive cho website**

Nguyên tắc 1: Hãy đặt mình vào vị trí của user

Họ sẽ thực hiện những thao tác gì, họ có thể xem được gì khi nhìn vào và sử dụng, thao tác trên trang web của bạn. Ở thời điểm hiện tại, UI/UX rất được coi trọng trong việc thiết kế và phát triển web, vì xu hướng người dùng càng ngày càng sử dụng các thiết bị di dộng nhiều hơn. Vậy nên, khi làm responsive cho web thì hãy tối ưu hóa nội dung hiển thị và trải nghiệm người dùng tốt nhất có thể, tránh việc hiển thị thiếu thông tin và gây khó khăn cho người dùng.

Nguyên tắc 2: Luôn luôn là Mobile First

Vì mục đích của khi làm responsive là hướng đến người dùng các thiết bị di động, mà các thiết bị này như chúng ta biết thì ngoài cấu hình yếu hơn PC/laptop, chúng còn có màn hình nhỏ hơn. Vậy nên một khi đã làm responsive cho website, hãy ưu tiên tối ưu hiệu suất và hiển thị cho các thiết bị này trước.

Nguyên tắc 3: Hiển thị nội dung kiểu dòng chảy

Nguyên tắc này có nghĩa là nội dung chỉ nên hiển thị trên 1 dòng từ trên xuống dưới, tránh tuyệt đối việc để người dùng phải vuốt ngang để có thể xem được nội dung, họ sẽ rời bỏ trang web của bạn ngay lập tức.

Nguyên tắc 4: Sử dụng các breakpoints hợp lý

Hãy liệt kê ra tất cả các breakpoints tương ứng với kích thước của các thiết bị di động phổ biến hiện nay và chọn ra những breakpoints phổ biến nhất để thực hiện việc responsive cho các devices này.

Nguyên tắc 5: Sử dụng các giá trị tương đối thay vì giá trị tuyệt đối

Nên sử dụng các giá trị tương đối cho việc set width hoặc height cho các phần tử hiển thị trên mobile, mà cụ thể ở đây là %, hạn chế việc sử dụng các giá trị tuyệt đối như px. Các giá trị tuyệt đối sẽ không thể tự resize theo chiều rộng/ngang của devices được.

Nguyên tắc 6: Hạn chế khoảng trống, giảm độ lớn font chữ và lược bỏ quảng cáo

Thường thì Khoảng cách giữa các phần tử, font chữ trên desktop khi hiển thị sẽ có khoảng cách và độ lớn khá lớn để tạo không gian thoải mái cho người dùng, nhưng nó sẽ không phù hợp trên các thiết bị di động nữa, khoảng trống và font chữ quá lớn sẽ gây khó chịu cho người dùng rất nhiều, vì thế hãy điều chỉnh sao cho phù hợp với từng kích thước màn hình. Hãy cố gắng hiển thị quảng cáo trên các thiết bị di động 1 cách tinh tế và hiệu quả nhất.