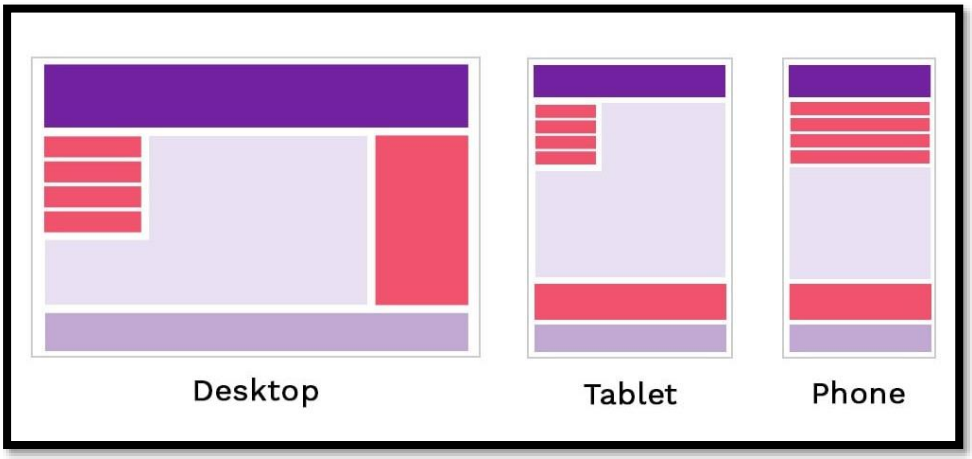
1. Mục tiêu

- Nguyên tắc cơ bản khi thiết kế Responsive.
- Xây dựng layout:
  - ✓ Sử dụng float
  - ✓ Sử dụng Flex, Grid
- Các thành phần khi xây dựng layout responsive:
  - ✓ font, img, video,…

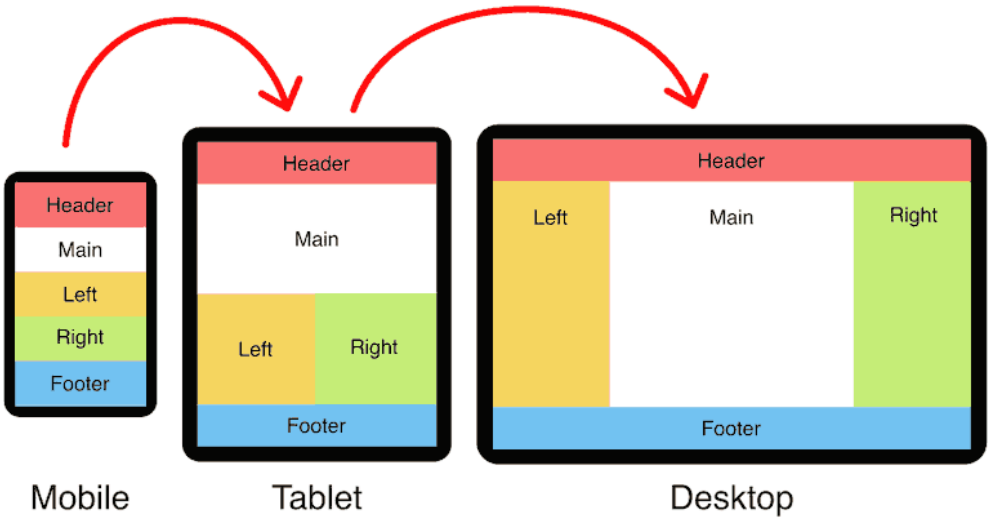
2. Tóm tắt lý thuyết

Giới thiệu Responsive Web Design

Các trang web có thể được xem bằng nhiều thiết bị khác nhau: máy tính để bàn, máy tính bảng và điện thoại. Trang web phải dược hiển thị đẹp và dễ sử dụng trên bất kể thiết bị nào. Nội dung không bị tràn ra ngoài trên thiết bị có kích thước nhỏ, mà phải thích ứng với nội dung của nó. Hình ảnh dưới đây mô tả đơn giản một ví dụ về RWD.



Khi ta sử dụng CSS và HTML để thay đổi kích thước, ẩn, co lại, phóng to hoặc di chuyển nội dung để làm cho bố cục trang web trở nên tương thích ở bất kỳ màn hình nào.



2.1. Những nguyên tắc cơ bản .

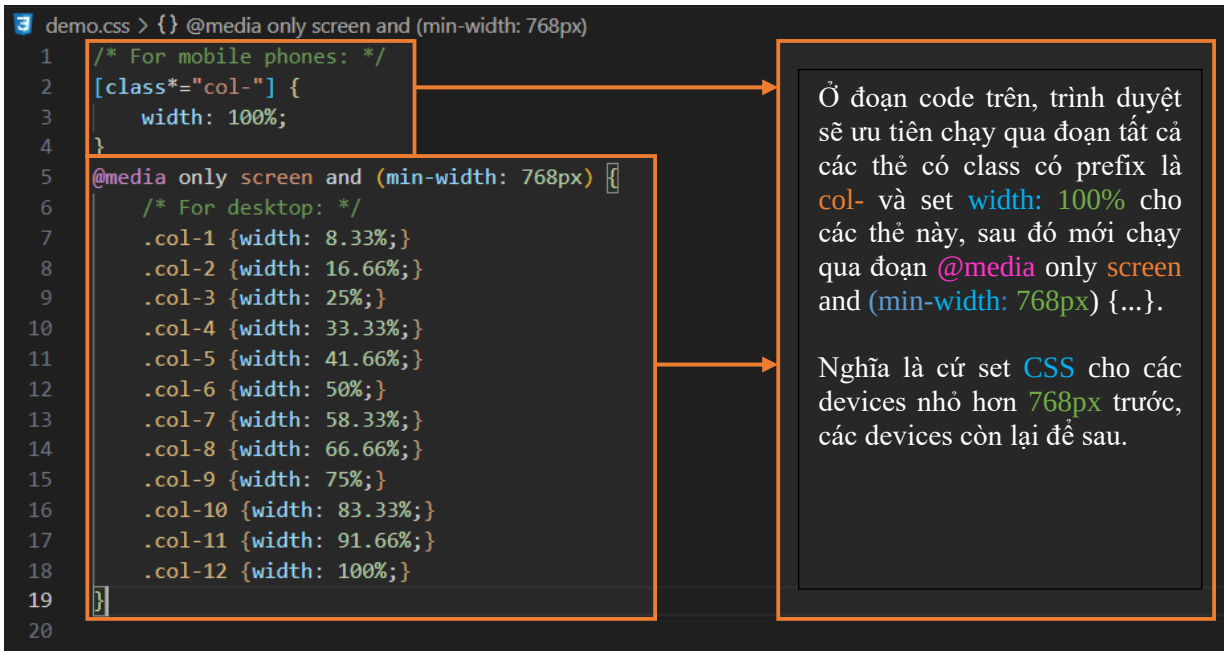
2.1.1 Hãy đặt mình vào vị trí của end-user .

Nguyên tắc đầu tiên là hãy đặt mình vào vị trí người dùng, họ sẽ thực hiện những thao tác gì, họ có thể xem được gì khi nhìn vào và thao tác trên trang web của ta. Ở thời điểm hiện tại, UI/UX rất được coi trọng việc thiết kế và phát triển web vì xu hướng người dùng ngày càng sử dụng các thiết bị di động nhiều hơn.

2.1.2 Luôn luôn là Mobile First.

Mobile First có nghĩa là thực hiện layout cho điện thoại di động trước bất kỳ thiết bị nào khác. Đây cũng là nguyên tắc thực hiện responsive của các CSS Library phổ biến như: Bootstrap, Foundation...

⇨ Vì mục đích khi làm responsive là hướng đến người dùng thiết bị di động, mà các thiết bị này thì ngoài cấu hình yếu hơn PC/laptop chúng còn có màn hình nhỏ hơn. Nên khi làm responsive cho website, phải ưu tiên tối ưu hiệu suất và hiển thị cho các thiết bị này trước.



2.1.3 Hiển thị nội dung kiểu dòng chảy.

Từ gốc của các tài liệu tiếng Anh là The flow. Nguyên tắc này có nghĩa là nội dung chỉ nên hiển thị trên 1 dòng từ trên xuống dưới, tránh việc để người dùng phải vuốt ngang để có thể xem được nội dung.

2.1.4 Sử dụng các breakpoint hợp lý.

Cần 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 responsive cho các devices này.

Tất nhiên không bỏ qua các breakpoints còn lại, mà chia ra thành nhóm - các thiết bị có kích thước giống nhau để giảm thiểu thời gian và số lượng code CSS.

Ví dụ: các thiết bị tablets có độ phân giải chiều rộng tối đa thường là 992px, thì

chỉ lấy breakpoints này làm mốc và viết CSS trong đó. Giả sử, ta muốn viết CSS chỉ dành riêng cho các thiết bị tablets, code CSS thường viết như sau:

```
@media only screen and (min-width: 992px) {
    CSS code here
}
```

Nếu ta viết min-width : 995px thì không nên . Vì trong đoạn từ 992px - 995px , nó không có tác dụng gì cho tablets cả.

2.1.5 Sử dụng các giá trị tương đối thay vì giá trị tuyệt đối.

Nên sử dụng các giá trị tương đối khi đặt giá trị width hoặc height cho các phần tử hiển thị trên mobile. Cụ thể là dùng %, hạn chế việc sử dụng các giá trị tuyệt đối như px. Vì chúng không thể tự resize theo chiều rộng/ngang của devices được. Ví dụ:

Thay vì

```
@media only screen and (min-width: 992px) {
    .image {
        width: 100%;
    }
}
```

```
@media only screen and (min-width: 995px) {
    .image {
        width: 500px;
    }
}
```

2.1.6 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.

Các thiết bị di động có kích thước khá nhỏ nên cần hiển thị đầy đủ thông tin hơn là xem các banner quảng cáo nhảy nhót khắp nơi. Đừng lạm dụng quảng cáo quá mức mà đánh mất đi số lượng lớn người dùng và nội dung hay trên trang web vì những banner quảng cáo. Hãy cố gắng hiển thị quảng cáo trên các thiết bị di động 1 cách tinh tế và hiệu quả nhất.

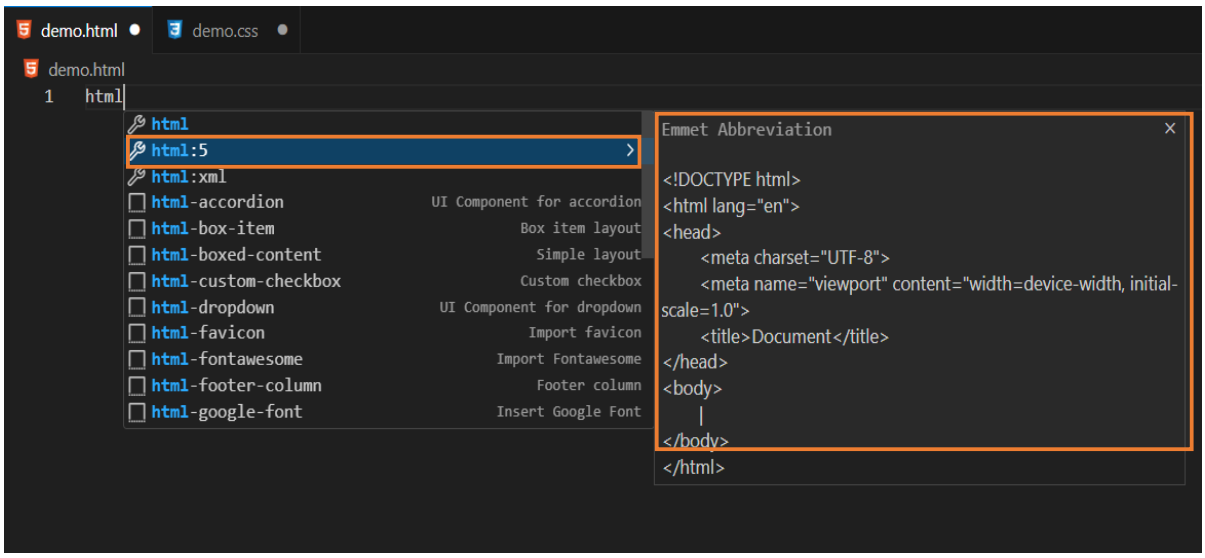
3. Cài đặt khung hình (Viewport).

Khung hình (Viewport) là khu vực hiển thị của người dùng trên trang web.

Khung nhìn thay đổi tùy theo thiết bị và sẽ nhỏ hơn trên điện thoại di động. Các trình duyệt trên các thiết bị đi động đã tự thu nhỏ toàn bộ trang web cho vừa với màn hình. Nhưng cách này chưa tối ưu. Để khắc phục việc này rất nhanh chóng.

Xây dựng khung hình(Viewport):

HTML5 đã giới thiệu một phương pháp cho phép các nhà thiết kế web kiểm soát khung nhìn thông qua thẻ <meta>.




```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Định dạng này cung cấp cho trình duyệt cách kiểm soát kích thước và tỷ lệ của trang.

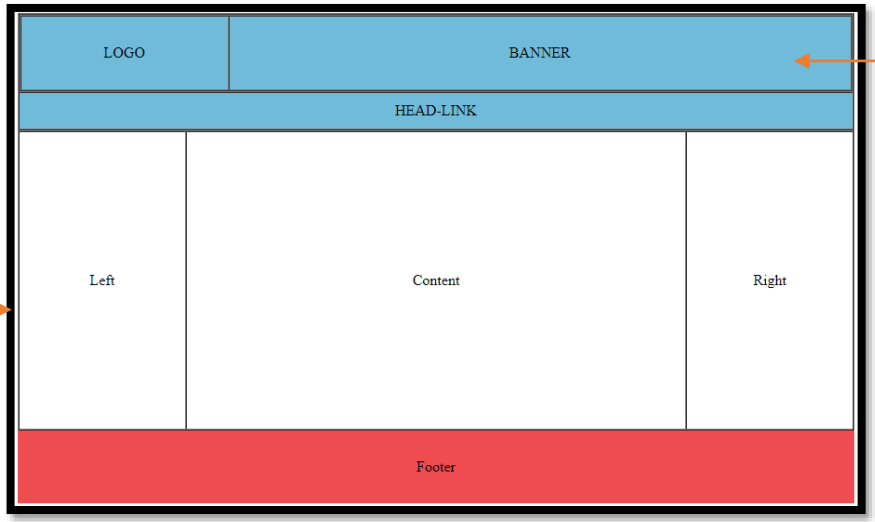
- Phần width = device-width đặt chiều rộng của trang theo chiều rộng màn hình của thiết bị (sẽ thay đổi tùy theo thiết bị).
- Phần này initial-scale = 1.0 đặt mức thu phóng ban đầu khi trang được trình duyệt tải lần đầu tiên.

4. Xây dựng Layout.

a. Xây dựng với Table.

Cách đơn giản nhất để tạo ra các layout là sử dụng thẻ <table> trong HTML. Nội dung được sắp xếp vào các cột và hàng. Vì thế ta có thể lợi dụng những hàng và cột này mà không cần sử dụng quá nhiều CSS.

Trang web có thể được thiết kế thành nhiều cột với các phần nội dung khác nhau. 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 được mô tả như hình sau:




```
<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>
```

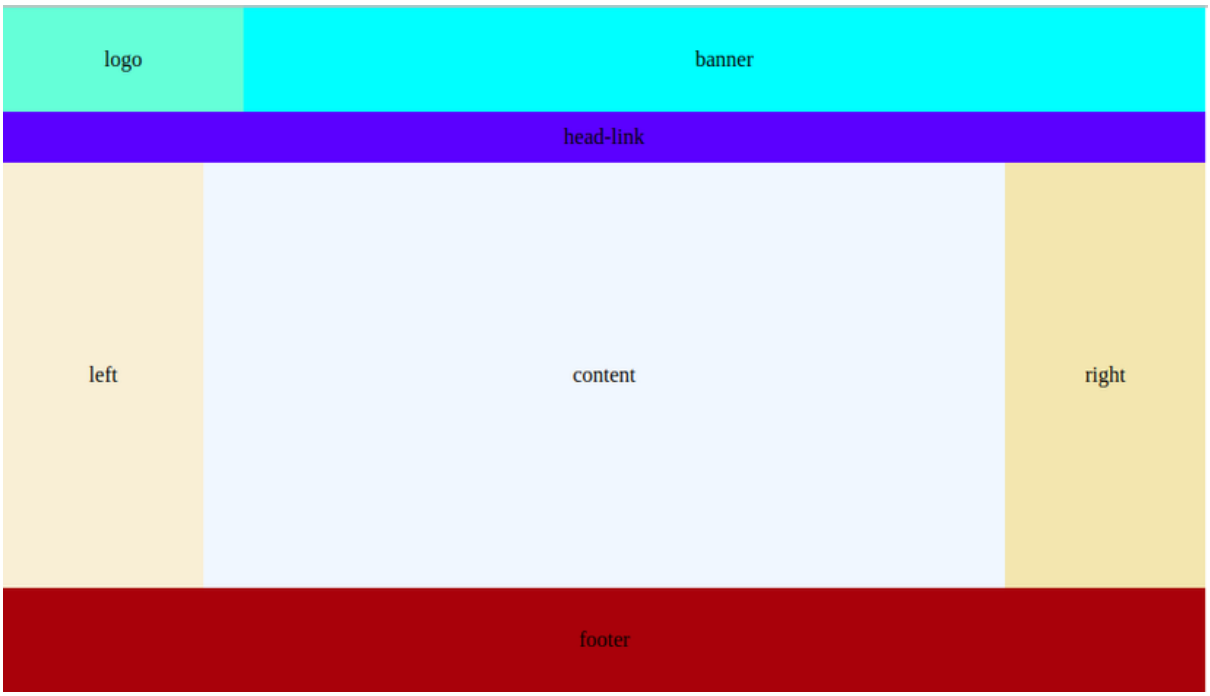
⇨ Tuy nhiên <table> lại bộc lộ khá nhiều nhược điểm khi sử dụng làm layout cho một trang web có cấu trúc như trên vì load chậm, khó tùy chỉnh và khó kết hợp với CSS - Javascript để tạo lên sự linh hoạt.

b. Sử dụng float và clear:

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

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 sang trái hoặc phải.
6. Tiến hành clear float.



Để xác định rõ phần tử cha của chúng, thuộc tính overflow: auto; được sử dụng để giữ phần tử cha vẫn nằm trong dòng.

```css
#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;
}
```

demo.html > html > head

```html
<!DOCTYPE html>
<html>

<head>
    <link rel="stylesheet" href="demo.css">
</head>

<body>
    <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>

</body>

</html>
```

```css
.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;
}
```

Trong phần này, có ba phần tử con: left, content và right. Tất cả các phần tử con đều được float về phía trái.

Để xử lý vấn đề xếp chồng nội dung, một pseudo-element được sử dụng với thuộc tính clear: both;.

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.
2. Khi hàng không đủ chỗ chứa phần tử thì phần tử phải bắt đầu ở hàng mới. 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.

c. Sử dụng Flex - box:

Flexbox là một kiểu dàn trang (layout mode) tự cân đối kích thước của các phần tử bên trong để hiển thị trên mọi thiết bị. Ta không cần thiết lập kích thước của phần tử, không cần sử dụng float, chỉ cần thiết lập nó hiển thị chiều ngang hay chiều dọc, lúc đó các phần tử bên trong có thể tự 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, ta 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.

demo.html > html > body

```html
<!DOCTYPE html>
<html>

<head>
    <link rel="stylesheet" href="demo.css">
</head>

<body>
    <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>

</body>

</html>
```

```css
.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;
}
```

```css
.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;
}
```

⇨ 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).

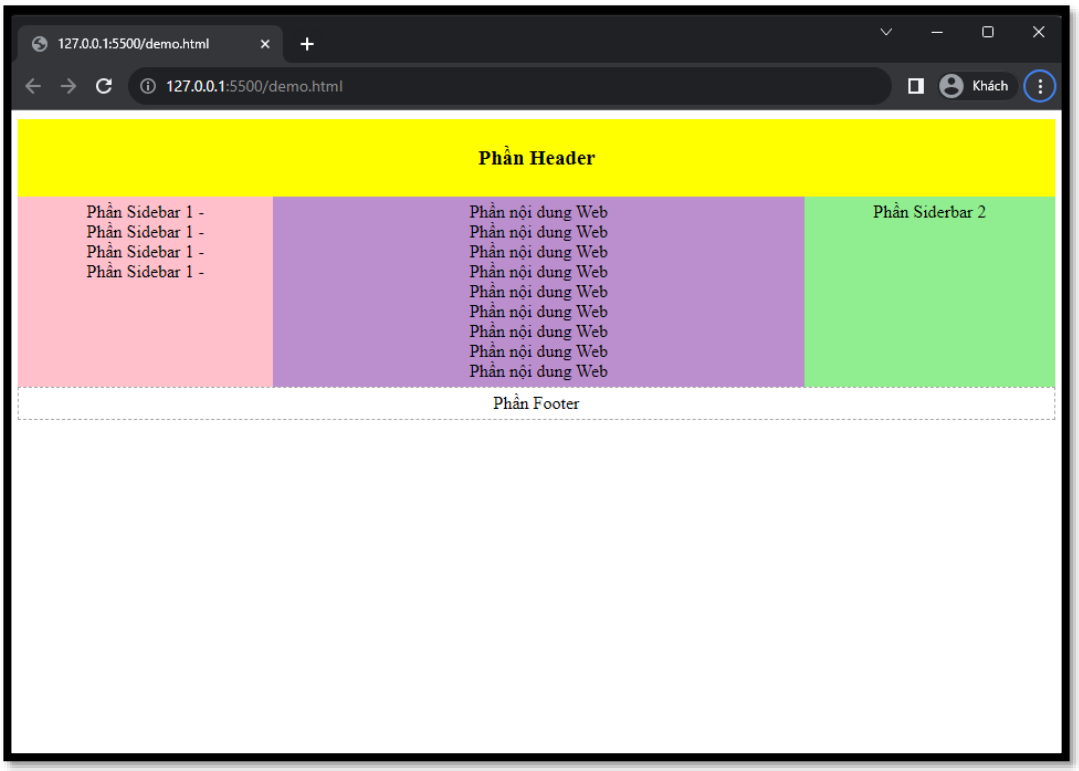
d. Sử dụng Grid:

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.

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, ta có thể tạo ra nhiều bố cục website phức tạp và đa dạng hơn.

Grid cho phép ta tạo một ma trận bố cục 2 chiều gồm các dòng và các cột. Ở mỗi dòng, cột và mỗi phần tử trong Grid có thể chỉnh sửa style. Vì vậy Grid cũng rất thích hợp để tạo bố cục trang Web.



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.

```html
<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>
```

```css
    '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;
}
```

e. Sử dụng Framework (Bootstrap):

Bootstrap là một framework HTML, CSS, và JavaScript cho phép thiết kế phát triển responsive web mobile.

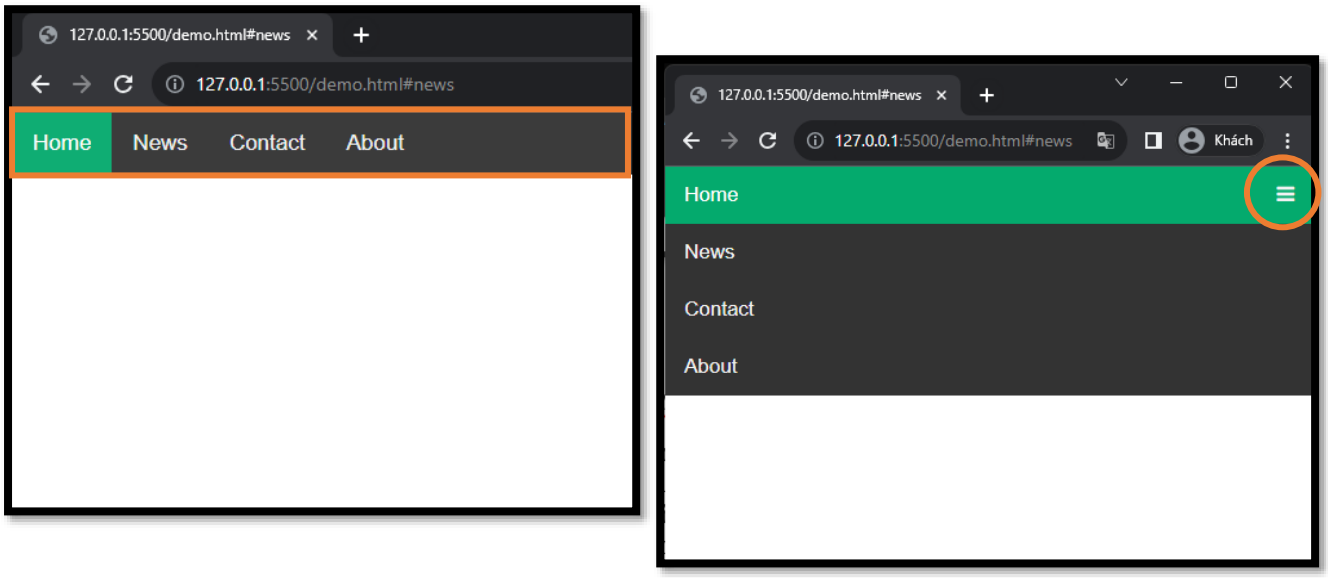
Nó cho phép thiết kế website reponsive nhanh hơn và dễ dàng hơn. Bootstrap bao gồm các HTML templates, CSS templates và Javascript tao ra những cái cơ bản có sẵn như: typography, forms, buttons, tables, navigation, modals, image carousels và nhiều thứ khác.

Trong bootstrap có thêm các plugin Javascript giúp cho việc thiết kế reponsive dễ dàng và nhanh chóng hơn.

5. Các vị trí xây dựng Responsive thông dụng.

Responsive menu:

Vị trí menu điều hướng các hoạt động của website. Thông thường với vị trí này chúng ta phải tạo responsive cho nó. Ở giao diện màn hình lớn thì menu hiển thị đầy đủ các mục chính (cấp 1) giàn ngang nhưng qua giao diện nhỏ thì chúng được ẩn đi chỉ hiển thị một nút nhỏ. Khi người dùng click vào nút đó thì hiển thị menu ra theo chiều dọc như hình sau.



Responsive Column:

Mỗi giao diện thông thường chúng ta có các vị trí sidebar left, sidebar right và content. Như vậy với ba vị trí này thì chúng ta tạm chia làm ba column. Nếu ở giao diện lớn thì chúng ta sẽ hiển thị nó ở dạng 3 column nhưng ở giao diện nhỏ thì chúng ta chỉ hiển thị nó ở dạng 1 column.

Responsive font size:

Với font size thì chúng ta hay thay đổi kích thước cho nó, với giao diện lớn thì chúng ta hiển thị kích thước lớn nhưng qua giao diện nhỏ thì đôi lúc chúng ta cần cho kích thước nhỏ lại để nội dung hiển thị trên một hàng hoặc hiển thị nhỏ lại để dễ nhìn hơn.

Responsive image:

Với hình ảnh thì nếu ta thiết lập chiều rộng và chiều cao cho nó thì khi qua giao diện nhỏ sẽ bị vỡ ngay vì kích thước của hình ảnh lớn hơn kích thước của thiết bị. Lúc này ta phải thay đổi lại kích thước sao cho hiển thị đúng với chiều rộng của thiết bị.

6. Sử dụng @media của CSS3:

Trong CSS3 có một thuộc tính ta hay gọi là hack CSS, thuộc tính này sẽ quyết định sử dụng đoạn CSS nào cho kích thước nào.

```
/* // Trình duyệt nhỏ có width là bé hơn hoặc bằng 768px */
@media only screen and (max-width: 768px){
    #sidebar{
        width: 100%
    }
}

/* // Trình duyệt nhỏ có width là lớn hơn 768px */
@media only screen and (min-width: 769px){
    #sidebar{
        width: 300px
    }
}
```

Trong ví dụ trên đoạn mã đã chia màn hình thành 2 loại kích thước khác nhau:

- Loại nhỏ: Kích thước bé hơn hoặc bằng 768px
- Loại lớn: Kích thước lớn hơn 768px

Như vậy khi ta thay đổi kích thước của trình duyệt nếu đang nằm trong khoảng nào thì CSS ở khoảng đó sẽ có tác dụng.

7. Sử dụng Javascript:

Trường hợp nếu trình duyệt không hỗ trợ CSS @media thì chúng ta có thể sử dụng Javascript để tạo CSS . Chúng ta lấy chiều rộng của trình duyệt và kiểm tra kích cỡ để load CSS tương ứng.

Tuy nhiên Javascript chỉ chạy đúng 1 lần nên khi ta thay đổi kích cỡ thì CSS không có tác dụng. Lúc này chúng ta phải sử dụng sự kiện resize của trình duyệt, để kiểm tra và thay đổi CSS.

JS demo.js > ...

```
$(window).resize(function(){

    var width = $(window).width();
    if (width <= 768){
        $('body').append('<link href="mobile.css"/>');
    }
    else{
        $('body').append('<link href="desktop.css"/>');
    }

});
```

8. Thời gian thực hành

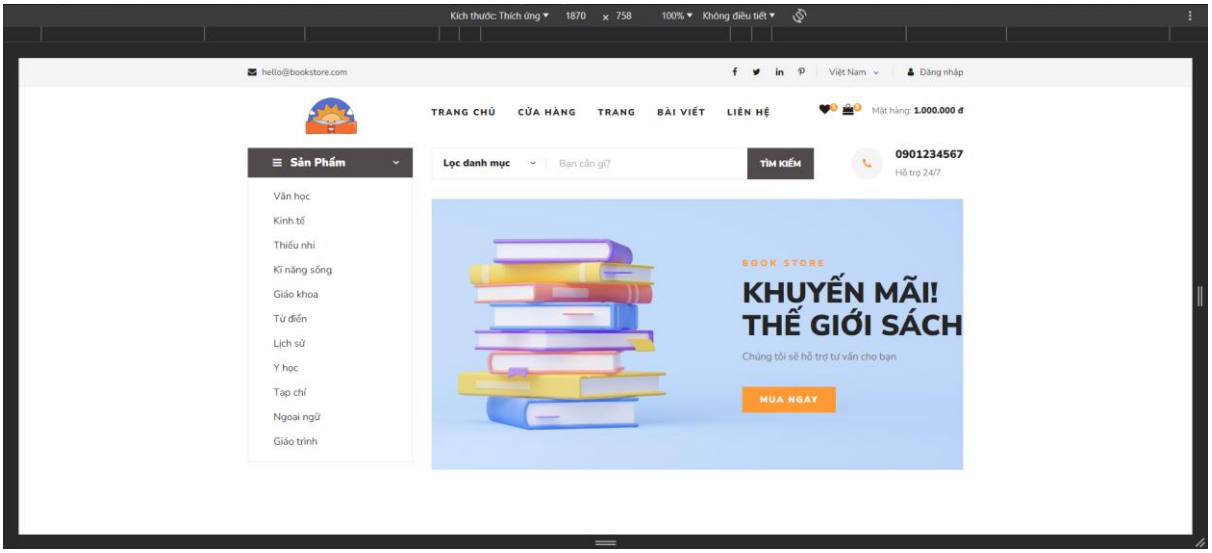
120 phút

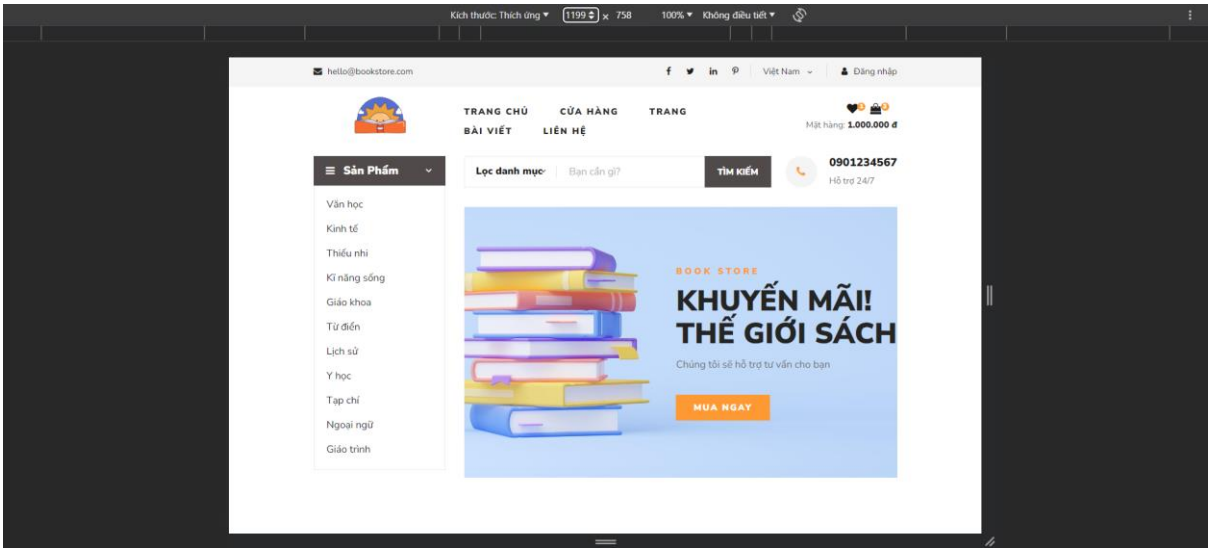
9. Đánh giá

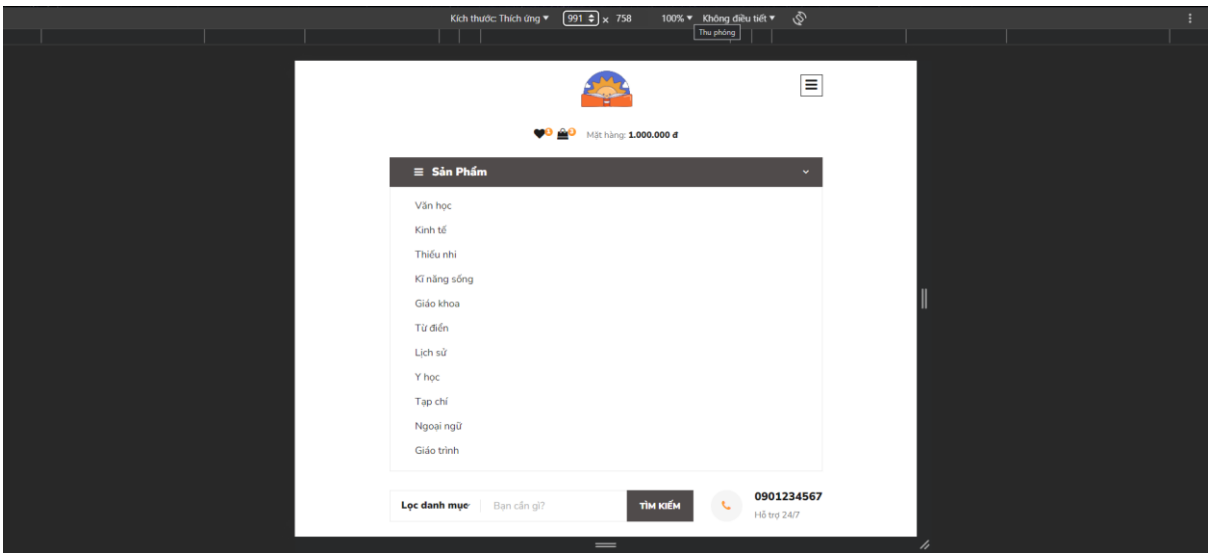
Thang điểm tối đa: 10 điểm/Lab

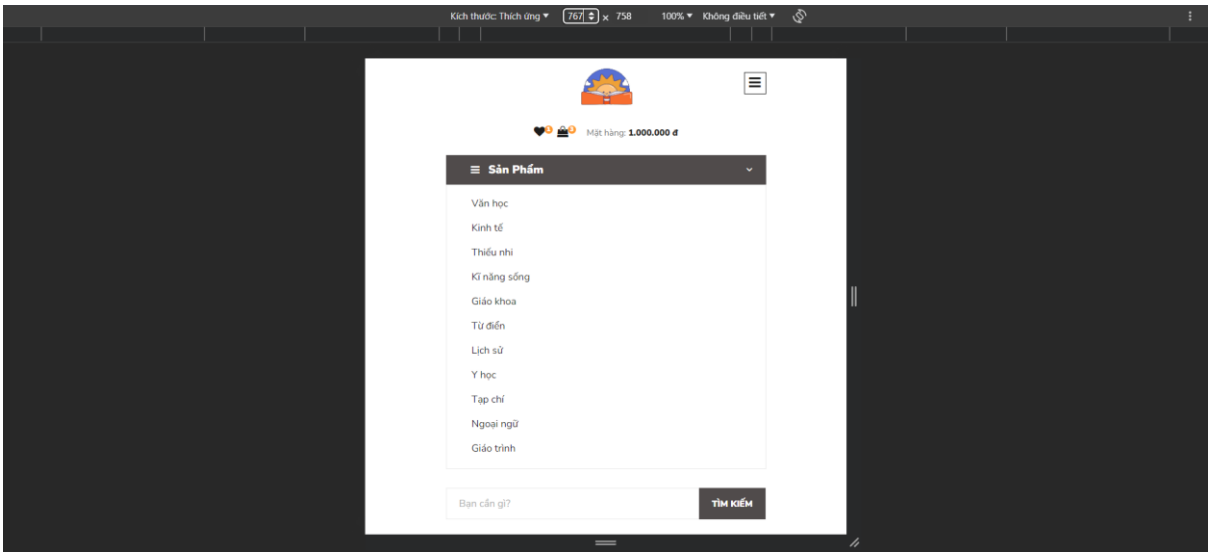
10. Ví dụ.

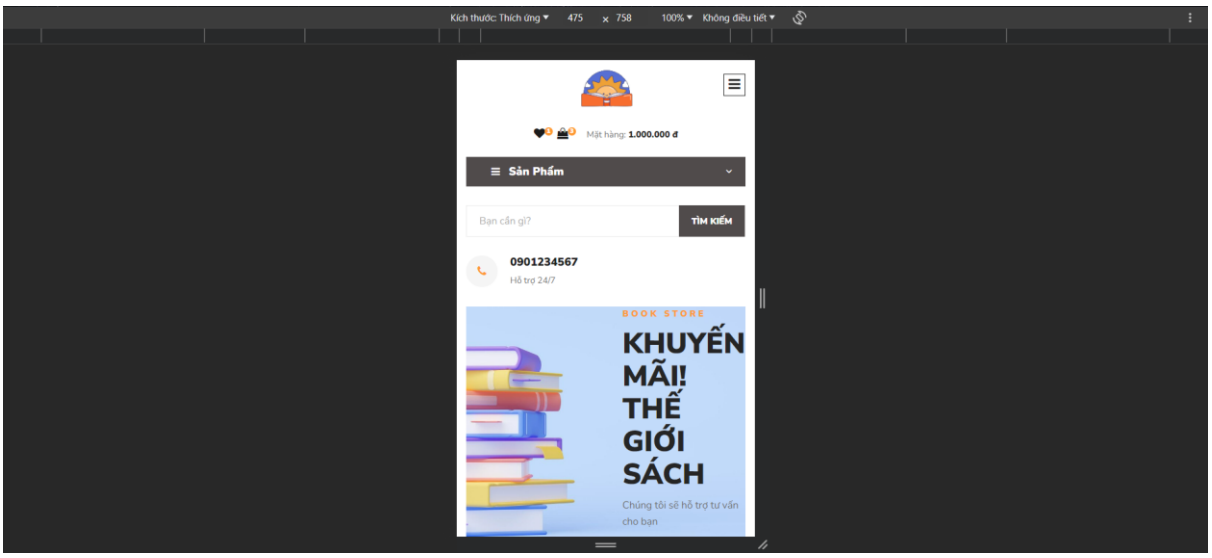
Quan sát sự thay đổi cấu trúc của trang web theo từng kích thước trong ví dụ sau:

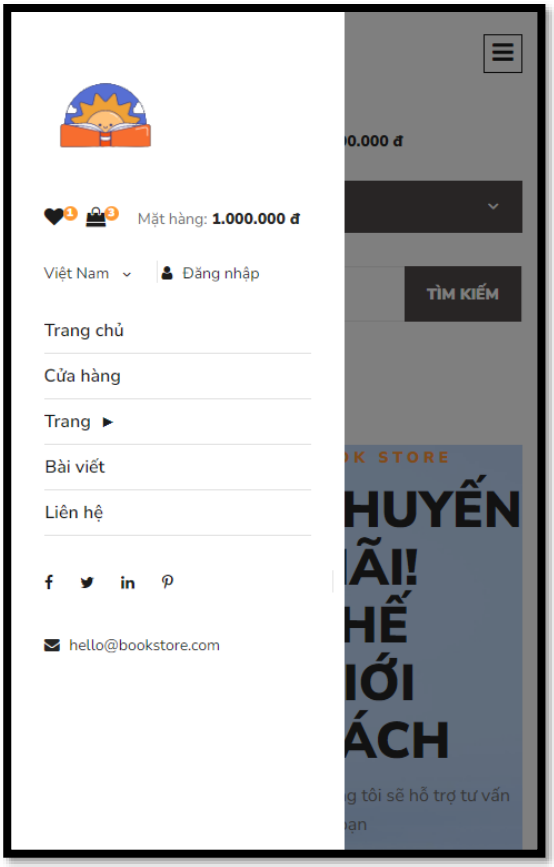





Kích thước: Thích ứng
991
x 758
100%
Không điều tiết
Thu phóng
Mặt hàng: 1.000.000 đ
Sản Phẩm
Văn học
Kinh tế
Thiếu nhi
Kĩ năng sống
Giáo khoa
Từ điển
Lịch sử
Y học
Tạp chí
Ngoại ngữ
Giáo trình
Lọc danh mục
Bạn cần gì?
TÌM KIẾM
0901234567
Hỗ trợ 24/7


Kích thước: Thích ứng
767
x 758
100%
Không điều tiết
Mặt hàng: 1.000.000 đ
Sản Phẩm
Văn học
Kinh tế
Thiếu nhi
Kĩ năng sống
Giáo khoa
Từ điển
Lịch sử
Y học
Tạp chí
Ngoại ngữ
Giáo trình
Bạn cần gì?
TÌM KIẾM


Kích thước: Thích ứng
475
x 758
100%
Không điều tiết
Mặt hàng: 1.000.000 đ
Sản Phẩm
Bạn cần gì?
TÌM KIẾM
0901234567
Hỗ trợ 24/7
BOOK STORE
KHUYẾN MÃI! THẾ GIỚI SÁCH
Chúng tôi sẽ hỗ trợ tư vấn cho bạn



- Ta có thể nhận rõ menu đã bị thu gọn từ kích thước 992px và chuyển sang dạng có button có chức năng hiển thị menu dọc.

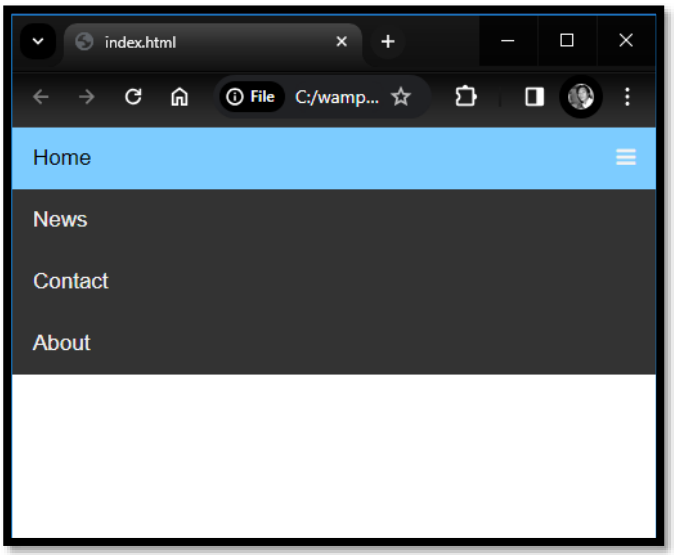
- Các chiều rộng của các thành phần cũng đều bị giảm dần từng kích thước.

- Dựa vào kích thước của ví dụ đã cho, ta có thể phân tích :

```
/*------------------------------- Responsive Media Quaries ----------------------------*/

@media only screen and (min-width: 1200px) {
    .container {
        max-width: 1170px;
    }
}

/* Medium Device = 1200px */

@media only screen and (min-width: 992px) and (max-width: 1199px) {
    .header__menu ul li {
        margin-right: 45px;
    }
    .hero__search__form {
        width: 490px;
    }
    .hero__categories__all {
        padding: 10px 25px 10px 20px;
    }
    .hero__categories ul {
        padding-left: 20px;
    }
}
```

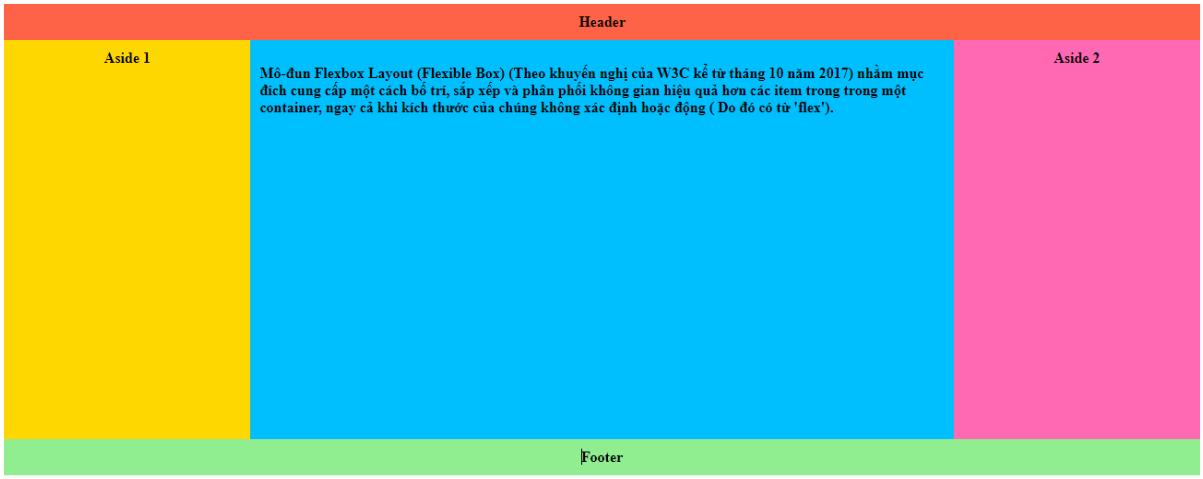
11. Yêu cầu thực hành

Bài 1: Xây dựng Menu responsive với `(max-width: 600px)`. Lưu thành *lab06_menu_responsive.html* (3đ)



Bài 2: Sinh viên sử dụng khung layout được cung cấp sẵn, chuyển đổi layout có thể responsive thành 2 cột `(min-width: 1200px)` và 1 cột `(min-width: 600px)` khi thay đổi kích thước trình duyệt. Lưu thành *lab06_bai2.html* (3đ)

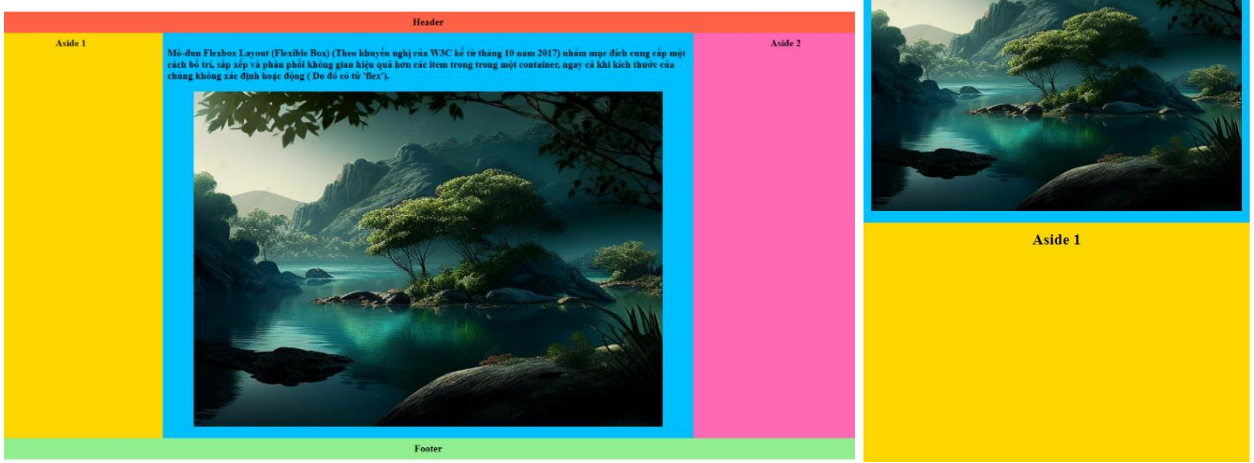
⇨ (Xem lại mục **c. Sử dụng Flex - box**:)



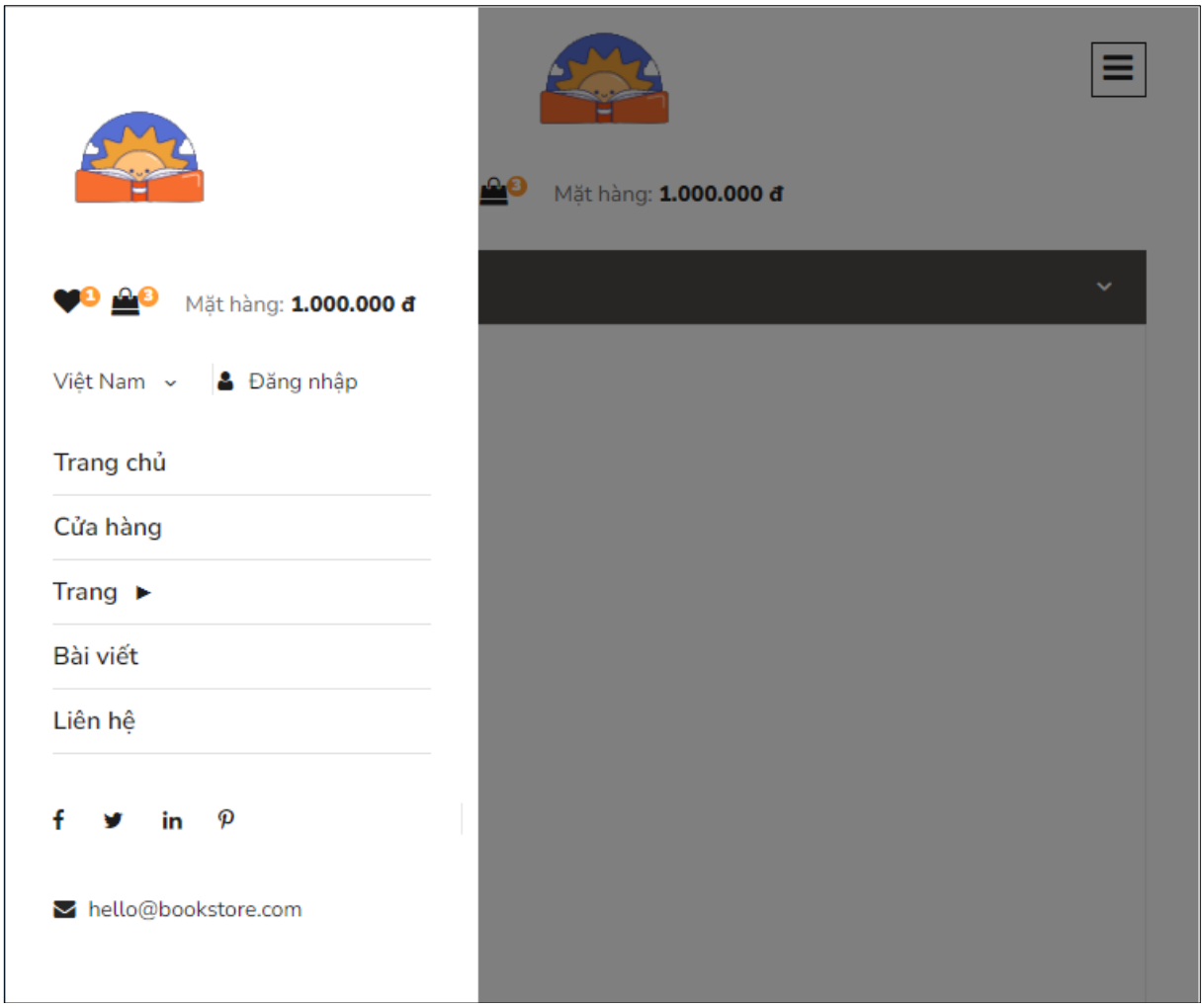
Kết quả:

Bài 3: Chèn hình *img_01012024.jpg* trong thư mục **img** vào cột 2 trong *lab06_bai2.html* trên. Thực hiện CSS thay đổi kích thước font chữ và hình ảnh khi chuyển thành 1 cột. Lưu thành *lab06_bai3.html* (2đ)

- Responsive font size: font-size của body là 16px, khi responsive 1 cột thì font là 20px.
- Responsive image: hình ảnh sẽ được thay đổi cùng tỉ lệ.



Bài 4 (2đ): Sinh viên chạy trang web trong **Vi_du**, dựa vào file ***Vi_du\sass****_responsive.scss*, hãy chuyển đổi cấu trúc và hoàn thành sử dụng kỹ thuật CSS media queries để đảm bảo trang web hiển thị chính xác theo các kích thước của ví dụ trên.




```
/* Tablet Device = 768px */

@media only screen and (min-width: 768px) and (max-width: 991px) {

}

/* Wide Mobile = 480px */

@media only screen and (max-width: 767px) {

}

/* Small Device = 320px */

@media only screen and (max-width: 479px) {

}
```