Trường ĐH Công Thương TP. HCM Khoa Công nghệ Thông tin Bộ môn Kỹ thuật phần mềm **THIẾT KẾ WEB**	**BÀI 6** **NGÔN NGỮ CSS (tt)**	

A. MỤC TIÊU:

- Thiết kế được giao diện web kết hợp HTML và CSS.
- Giới thiệu CSS Animation; Áp dụng CSS Animation
- Giới thiệu Filter, Display; Áp dụng Filter, Display
- Giới thiệu Position,; Áp dụng Position

B. NỘI DUNG THỰC HÀNH

1. Cơ sở lý thuyết

1.1. Kiến thức cần nhớ

3.3. Định dạng trang web với CSS

3.3.1. Màu sắc và nền

3.3.2. Layout – float và clear

3.3.3. Xử lý văn bản

3.3.4. Viền, Khoảng đệm, Lề

3.3.5. Box Shadow

3.3.6. Transform

3.3.5. Filter, Display

3.3.5.1. Filter

Filter là một thuộc tính cho phép áp dụng các hiệu ứng đồ họa như làm mờ, thay đổi độ sáng, độ tương phản, và nhiều hiệu ứng khác lên các phần tử HTML. Thuộc tính filter thường được sử dụng để tạo ra các hiệu ứng hình ảnh động hoặc tinh chỉnh giao diện người dùng.

Cú pháp: `filter: <filter-function> [<filter-function>]* | none;`

Các hàm filter-function phổ biến:

- blur(): Làm mờ phần tử.

Ví dụ: `filter: blur(5px);`

- brightness(): Điều chỉnh độ sáng.

Ví dụ: `filter: brightness(150%);`

- contrast(): Điều chỉnh độ tương phản.

Ví dụ: `filter: contrast(200%);`

- grayscale(): Chuyển đổi hình ảnh sang thang độ xám.

Ví dụ: `filter: grayscale(100%);`

- invert(): Đảo ngược màu sắc.

Ví dụ: `filter: invert(100%);`

- opacity(): Điều chỉnh độ trong suốt.

Ví dụ: `filter: opacity(50%);`

- saturate(): Điều chỉnh độ bão hòa màu.

Ví dụ: `filter: saturate(200%);`

- sepia(): Áp dụng hiệu ứng sepia (màu nâu đỏ).

Ví dụ: `filter: sepia(100%);`

- drop-shadow(): Tạo bóng đổ.

Ví dụ: `filter: drop-shadow(5px 5px 10px black);`

- hue-rotate(): Xoay màu sắc theo vòng tròn màu.

Ví dụ: `filter: hue-rotate(90deg);`

Ví dụ kết hợp nhiều hiệu ứng:

```
img {
     filter: grayscale(50%) blur(2px) brightness(75%);
}
```

✍ Lưu ý:

- Thuộc tính filter có thể được áp dụng cho hầu hết các phần tử HTML, bao gồm cả hình ảnh, nền, và các phần tử khác.
- Hiệu ứng filter có thể ảnh hưởng đến hiệu suất, đặc biệt là khi áp dụng cho các phần tử lớn hoặc nhiều phần tử cùng lúc.
- Thuộc tính filter được hỗ trợ rộng rãi trong các trình duyệt hiện đại, bao gồm Chrome, Firefox, Safari, Edge, và Opera. Tuy nhiên, một số hàm filter-function có thể không được hỗ trợ đầy đủ trong các trình duyệt cũ.

Ví dụ:

```
<head>
     <title>CSS Filter Example</title>
     <style>
          img {
```

```
            width: 300px;
            filter:         grayscale(50%)         blur(2px)
 brightness(75%);
        }
        img:hover {
                filter:       grayscale(0%)       blur(0px)
 brightness(100%);
        }
    </style>
</head>
<body>
    <img src="images/NuiBaDen.png" alt="Example Image">
</body>
```

3.3.5.2. Display

Thuộc tính display là một trong những thuộc tính quan trọng nhất để kiểm soát cách các phần tử HTML được hiển thị trên trang web. Thuộc tính này xác định loại hộp (box) mà một phần tử sẽ tạo ra, ảnh hưởng đến cách bố trí (layout) và cách các phần tử khác tương tác với nó.

Các giá trị phổ biến của display:

a. block: Phần tử sẽ hiển thị như một khối (block), chiếm toàn bộ chiều rộng của phần tử cha và bắt đầu trên một dòng mới.

Ví dụ: , <p>, <h1> đến <h6>, <header>, <footer>, v.v.

```
div {
    display: block;
}
```

b. inline: Phần tử sẽ hiển thị như một phần tử nội dòng (inline), chỉ chiếm không gian cần thiết và không bắt đầu trên một dòng mới.

Ví dụ: , <a>, , , v.v.

```
span {
    display: inline;
}
```

c. inline-block: Phần tử sẽ hiển thị như một phần tử nội dòng (inline), nhưng có thể đặt chiều rộng và chiều cao giống như phần tử khối (block).

Ví dụ: Nút bấm (<button>), hình ảnh (<img>), v.v.

```
button {
```

```
    display: inline-block;
    width: 100px;
    height: 50px;
}
```

d. none: Phần tử sẽ bị ẩn hoàn toàn và không chiếm không gian trên trang.

Ví dụ: Ẩn một phần tử tạm thời

```
.hidden {
        display: none;
}
```

e. flex: Phần tử sẽ trở thành một flex container, cho phép bạn dễ dàng tạo các bố cục linh hoạt (flexible layouts) với các phần tử con được sắp xếp theo hàng hoặc cột.

Ví dụ: Tạo bố cục linh hoạt

```
.container {
            display: flex;
            justify-content: space-between;
}
```

f. grid: Phần tử sẽ trở thành một grid container, cho phép bạn tạo các bố cục dạng lưới (grid layouts) với các hàng và cột được xác định rõ ràng.

Ví dụ: Tạo bố cục lưới

```
.container {
            display: grid;
            grid-template-columns: 1fr 1fr 1fr;
}
```

g. inline-flex: Tương tự như flex, nhưng phần tử sẽ hiển thị như một phần tử nội dòng (inline).

h. inline-grid: Tương tự như grid, nhưng phần tử sẽ hiển thị như một phần tử nội dòng (inline).

i. table, table-row, table-cell: Các giá trị này cho phép mô phỏng hành vi của các phần tử bảng (<table>, <tr>, <td>) mà không cần sử dụng các thẻ HTML bảng.

Ví dụ:

```
.table {
        display: table;
```

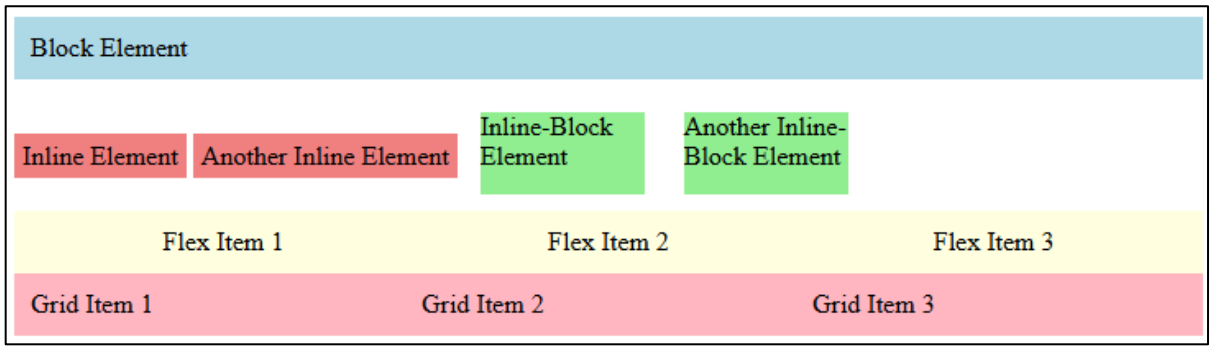
```
    }

    .table-row {
            display: table-row;
    }

    .table-cell {
            display: table-cell;
    }
```

Ví dụ tổng hợp:

```
<head>
    <title>CSS Display Example</title>
    <style>
        .block {
            display: block;
            background-color: lightblue;
            padding: 10px;
            margin: 10px 0;
        }
        .inline {
            display: inline;
            background-color: lightcoral;
            padding: 5px;
        }
        .inline-block {
            display: inline-block;
            background-color: lightgreen;
            width: 100px;
            height: 50px;
            margin: 10px;
        }
        .hidden {
            display: none;
        }
        .flex-container {
            display: flex;
            justify-content: space-around;
```

```
            background-color: lightyellow;
            padding: 10px;
        }
        .grid-container {
            display: grid;
            grid-template-columns: 1fr 1fr 1fr;
            gap: 10px;
            background-color: lightpink;
            padding: 10px;
        }
    </style>
</head>
<body>
    <div class="block">Block Element</div>
    <span class="inline">Inline Element</span>
    <span class="inline">Another Inline Element</span>
    <div class="inline-block">Inline-Block Element</div>
    <div class="inline-block">Another Inline-Block Element</div>
    <div class="hidden">This element is hidden</div>
    <div class="flex-container">
        <div>Flex Item 1</div>
        <div>Flex Item 2</div>
        <div>Flex Item 3</div>
    </div>
    <div class="grid-container">
        <div>Grid Item 1</div>
        <div>Grid Item 2</div>
        <div>Grid Item 3</div>
    </div>
</body>
```

Kết quả:



3.3.6. Position

Thuộc tính position được sử dụng để xác định cách một phần tử được định vị (positioned) trong tài liệu HTML. Thuộc tính này cho phép kiểm soát vị trí của phần tử một cách chính xác, bao gồm việc đặt phần tử ở một vị trí cụ thể, cố định trên màn hình, hoặc tương đối so với vị trí ban đầu của nó.

a. static (Mặc định)

Phần tử được định vị theo thứ tự bình thường trong luồng tài liệu (document flow).

Các thuộc tính như top, right, bottom, left, và z-index không có tác dụng với position: static.

Đây là giá trị mặc định của tất cả các phần tử.

Ví dụ:

```
div {
      position: static;
}
```

b. relative

Phần tử được định vị **tương đối so với vị trí ban đầu** của nó trong luồng tài liệu.

Có thể sử dụng các thuộc tính top, right, bottom, và left để dịch chuyển phần tử từ vị trí ban đầu.

Phần tử vẫn chiếm không gian ban đầu trong luồng tài liệu.

Ví dụ:

```
div {
      position: relative;
      top:20px; /*Dịch chuyển xuống 20px so với vị trí ban đầu */
      left:30px; /*Dịch chuyển sang phải 30px so với vị trí ban đầu */
}
```

c. absolute

Phần tử được định vị **tuyệt đối so với phần tử cha gần nhất có position khác static** (ví dụ: relative, absolute, fixed, hoặc sticky).

Nếu không có phần tử cha nào phù hợp, phần tử sẽ được định vị tương đối so với <body>.

Phần tử **không chiếm không gian** trong luồng tài liệu (nó sẽ bị loại bỏ khỏi luồng bình thường).

Có thể sử dụng top, right, bottom, và left để đặt vị trí chính xác.

Ví dụ:

```
div {
      position: absolute;
      top: 50px; /* Cách đỉnh 50px */
      left: 100px; /* Cách bên trái 100px */
}
```

d. fixed

Phần tử được định vị **tuyệt đối so với khung nhìn (viewport)**, nghĩa là nó sẽ luôn ở một vị trí cố định trên màn hình, ngay cả khi trang được cuộn.

Phần tử **không chiếm không gian** trong luồng tài liệu.

Thường được sử dụng cho các phần tử như thanh điều hướng cố định (fixed navigation bar) hoặc nút "back to top".

Ví dụ:

```
div {
       position: fixed;
       bottom: 20px; /* Cách đáy màn hình 20px */
       right: 20px; /* Cách bên phải màn hình 20px */
}
```

e. sticky

Phần tử được định vị **tương đối** cho đến khi nó đạt đến một ngưỡng cụ thể (được xác định bởi top, right, bottom, hoặc left), sau đó nó sẽ trở thành cố định (fixed).

Thường được sử dụng để tạo các tiêu đề bảng (table headers) hoặc thanh điều hướng dính (sticky navigation bar).

Ví dụ:

```
div {
```

```
        position: sticky;
        top: 0; /* Dính vào đỉnh màn hình khi cuộn đến */
}
```

Các thuộc tính đi kèm với position: Khi sử dụng position với các **giá trị khác static**, có thể sử dụng các thuộc tính sau để điều chỉnh vị trí:

- top: Khoảng cách từ cạnh trên của phần tử đến cạnh trên của phần tử cha hoặc viewport.
- right: Khoảng cách từ cạnh phải của phần tử đến cạnh phải của phần tử cha hoặc viewport.
- bottom: Khoảng cách từ cạnh dưới của phần tử đến cạnh dưới của phần tử cha hoặc viewport.
- left: Khoảng cách từ cạnh trái của phần tử đến cạnh trái của phần tử cha hoặc viewport.

Ví dụ:

```
<head>
    <title>CSS Position Example</title>
    <style>
        .box {
            width: 100px;
            height: 100px;
            background-color: lightcoral;
            margin: 10px;
            text-align: center;
            line-height: 100px;
        }
        .relative {
            position: relative;
            top: 20px;
            left: 30px;
        }
        .absolute {
            position: absolute;
            top: 50px;
            left: 150px;
        }
```

```
        .fixed {
            position: fixed;
            bottom: 20px;
            right: 20px;
        }
        .sticky {
            position: sticky;
            top: 0;
            background-color: lightgreen;
        }
        .container {
            position: relative;
            height: 300px;
            border: 2px solid black;
            margin-top: 50px;
        }
    </style>
</head>
<body>
    <div class="box">Static</div>
    <div class="box relative">Relative</div>
    <div class="container">
        <div class="box absolute">Absolute</div>
    </div>
    <div class="box fixed">Fixed</div>
    <div style="height: 1000px;">
        <div class="box sticky">Sticky</div>
    </div>
</body>
```

Giải thích:

- static: Phần tử đầu tiên được định vị theo luồng tài liệu bình thường.
- relative: Phần tử thứ hai được dịch chuyển 20px từ đỉnh và 30px từ bên trái so với vị trí ban đầu.
- absolute: Phần tử thứ ba được định vị tuyệt đối so với phần tử cha (.container).
- fixed: Phần tử thứ tư được cố định ở góc dưới bên phải của màn hình.
- sticky: Phần tử thứ năm sẽ dính vào đỉnh màn hình khi bạn cuộn trang.

✍ Lưu ý: Khi nào sử dụng position?

- relative: Khi muốn dịch chuyển phần tử từ vị trí ban đầu mà không ảnh hưởng đến các phần tử khác.
- absolute: Khi muốn định vị phần tử chính xác so với phần tử cha.
- fixed: Khi muốn phần tử luôn hiển thị ở một vị trí cố định trên màn hình.
- sticky: Khi muốn phần tử dính vào một vị trí cụ thể khi cuộn trang.

3.3.7. Animation

Trong CSS, animation là một thuộc tính mạnh mẽ cho phép tạo các hiệu ứng chuyển động (animations) cho các phần tử HTML. Bằng cách sử dụng animation, có thể kiểm soát các khung hình (keyframes) và các thuộc tính như thời lượng, độ trễ, hướng lặp lại, và tốc độ của hiệu ứng.

Các thành phần chính của CSS Animation:

a. @keyframes

Định nghĩa các khung hình (keyframes) của animation, có thể chỉ định các trạng thái (styles) tại các điểm cụ thể trong quá trình animation.

Cú pháp:

```
@keyframes tên-animation {
  từ {
    /* CSS tại điểm bắt đầu */
  }
  đến {
    /* CSS tại điểm kết thúc */
  }
}
```

Hoặc

```
@keyframes tên-animation {
  0% {
    /* CSS tại 0% */
  }
  50% {
    /* CSS tại 50% */
  }
  100% {
```

```
    /* CSS tại 100% */
  }
}
```

b. Thuộc tính animation

Thuộc tính animation là một cách viết tắt (shorthand) để thiết lập các thuộc tính con của animation. Các thuộc tính con bao gồm:

- animation-name: Tên của @keyframes.
- animation-duration: Thời lượng của animation (ví dụ: 2s).
- animation-timing-function: Hàm thời gian (timing function) để kiểm soát tốc độ của animation (ví dụ: ease, linear, ease-in-out).
- animation-delay: Độ trễ trước khi animation bắt đầu (ví dụ: 1s).
- animation-iteration-count: Số lần lặp lại animation (ví dụ: infinite để lặp vô hạn).
- animation-direction: Hướng của animation (ví dụ: normal, reverse, alternate).
- animation-fill-mode: Xác định cách áp dụng styles trước và sau khi animation chạy (ví dụ: forwards, backwards).
- animation-play-state: Cho phép tạm dừng hoặc tiếp tục animation (ví dụ: paused, running).

Cú pháp viết tắt:

```
animation: tên-animation thời-gian timing-function delay
           iteration-count direction fill-mode;
```

Ví dụ:

```
<head>
    <meta charset="UTF-8">
    <title>CSS Animation Example</title>
    <style>
        .box {
            width: 100px;
            height: 100px;
            padding-left: 10px;
            padding-top: 10px;
            background-color: lightcoral;
            animation: move 2s ease-in-out infinite;
        }
```

```
        @keyframes move {
            0% {
                transform: translateX(0);
            }
            50% {
                transform: translateX(200px);
            }
            100% {
                 transform: translateX(0);
            }
        }
    </style>
</head>
<body>
    <div class="box">Hello</div>
</body>
```

1.2. Giới thiệu bài tập mẫu

Bài 1. Tạo trang BaiTapMau1.html với giao diện và nội dung như sau:

Hướng dẫn: Sử dụng các thuộc tính

- Tạo Element cần diễn hoạt như hình trên;

```
<body>
    <div>CSS Animation</div>
</body>
```

- Định dạng CSS cho Element và thêm thuộc tính Animation

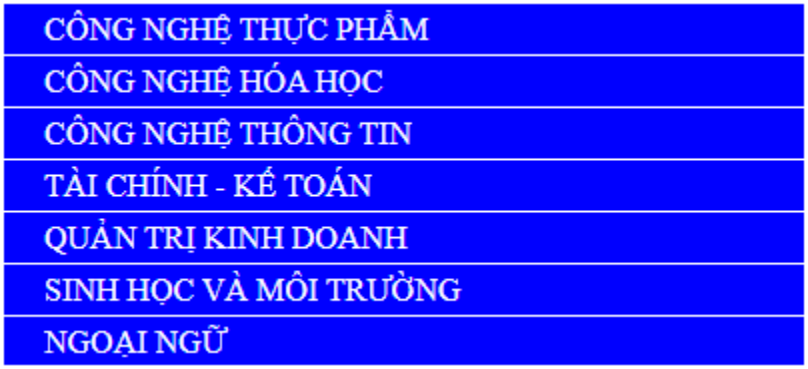
```
div{
      width:200px;
      padding:20px;
      text-align:center;
      font-weight:bold;
      font-size:25px;
      background-color:#909;
      color:#FFF;
      border-radius:10px;
      margin:100px 0px;
```

```
        position:relative;
        animation: ani 5s linear 1s 2 alternate;
    }
```

- Tạo Keyframes ani gồm 5 key cho phép thay đổi màu nền từ key 1 → key 5 tương ứng như sau: Key1: #909; Key2: #C3F; Key3: #F9F; Key4: #F6F; Key5: #F0F;

```
@keyframes ani {
        0%    {
              background-color:#909;left:0px;
        }
        25%   {
              background-color:#C3F;left:20%;
        }
        50%   {
              background-color:#F9F;left:40%;
              transform:rotate(180deg)
        }
        75%   {
              background-color:#F6F;left:60%;
        }
        100% {
              background-color:#F0F;left:85%;
        }
}
```

Bài 2. Tạo trang BaiTapMau2.html thiết kế SideBar như sau:



Và khi hover chuột qua thì chữ in đậm và màu nền chuyển sang màu #900:

Yêu cầu: Sử dụng các thuộc tính

- Dùng class selector
- Tạo Slidear có kích thước Width: 400px, H:180px, màu background #00F.
- Màu text là màu #FFF ;
- Chiều cao của mỗi mục trong menu là 25px ;
- Padding cho phần text là 0px 20px 0px 20px tương ứng Top Right Button Left;

Hướng dẫn:

Bước 1: Tạo danh sách không có thứ tự các liên kết trong khung slidebar như sau:

```
<body>
    <div class="sidebar">
        <ul>
            <li><a href="#">Công nghệ Thực phẩm</a></li>
            <li><a href="#">Công nghệ Hóa học</a></li>
            <li><a href="#">Công nghệ Thông tin</a></li>
            <li><a href="#">Tài chính - Kế toán</a></li>
            <li><a href="#">Quản trị kinh doanh</a></li>
            <li><a href="#">Sinh học và Môi trường</a></li>
            <li><a href="#">Ngoại ngữ</a></li>
        </ul>
    </div>
</body>
```

Bước 2: Tạo file styleBTM2.css

```
.sidebar {
    width: 400px;
    height: 180px;
    background-color: #00F;
}
.sidebar ul {
```

```
        list-style: none;
        padding: 0;
        margin: 0;
        color: #FFF;
        text-transform: uppercase;
    }
    .sidebar ul li {
        line-height: 25px;
        border-bottom: 1px solid #FFF;
    }
    .sidebar ul li a {
        text-decoration: none;
        color: #FFF;
        padding: 0 20px;
    }
    .sidebar ul li a:hover {
        font-weight: bold;
        background-color: #900;
    }
```

Bước 3: Kết nối styleBTM2.css vào file BaiTapMau2.html thông qua thẻ link

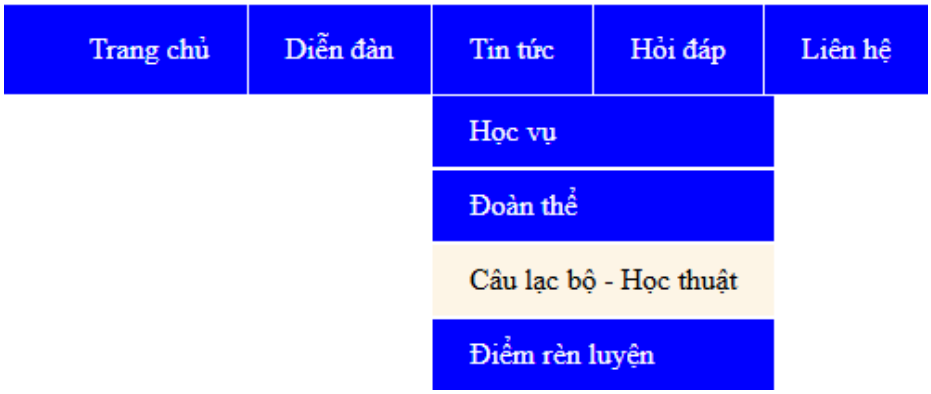
```
<link rel="stylesheet" type="text/css" href="CSS/styleBTM2.css" />
```

2. Bài tập tại lớp

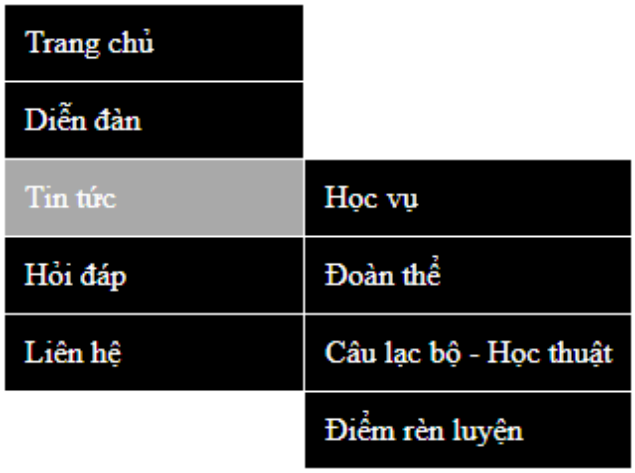
Bài 1. Thiết kế trang web Tạo menu 1 cấp ngang như sau:

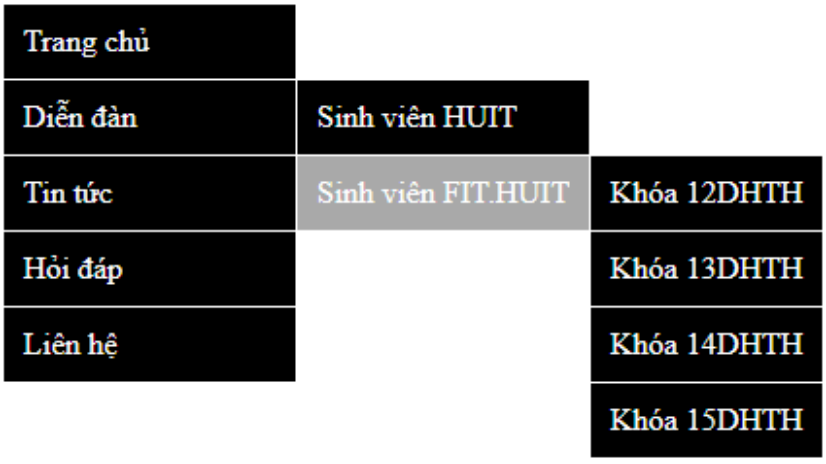
Yêu cầu:

- Dùng class Selector
- Background: blue
- Text màu : white
- Có phân cách giữa các menu
- Khi hover chuột qua màu nền chuyển sang: oldlace.



Bài 2. Tạo trang menu đa cấp đứng như sau:





Yêu cầu:

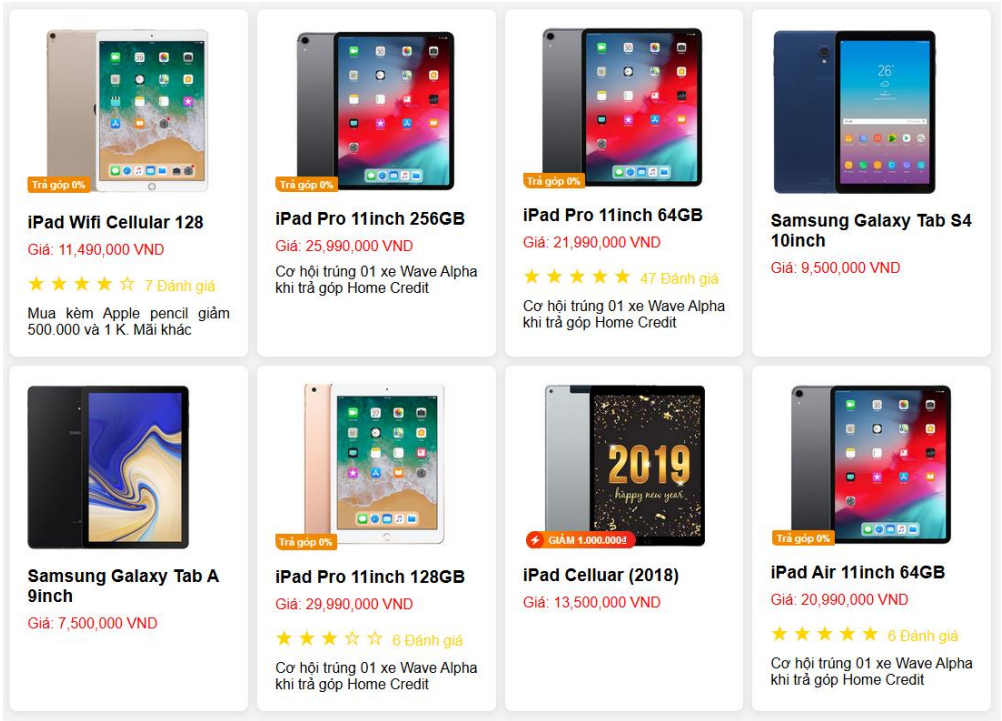
- Dùng IDSelector
- Độ rộng: 150px;
- Background: #000000 (đen)
- Text màu : #FFF (trắng)
- Border-bottom: 1px solid #FFF;
- Khi hover chuột qua màu nền chuyển sang : darkgray
- Chiều rộng menu cấp 2, 3 : max-content

- Menu cấp 3 kế thừa lại các thuộc tính của menu cấp 2
- Khi hover chuột vào mục co menu con thì menu con được hiển thị; khi rời chuột thì menu con sẽ ẩn đi.

Bài 3. Tạo trang menu đa cấp như sau:

3. Bài tập về nhà

Thiết kế trang web với giao diện và nội dung như sau:



Yêu cầu:

- Sử dụng thẻ DIV và file CSS ngoài
- Dùng HTML Selector và Class selector.