

PHIẾU HỌC TẬP CSS CƠ BẢN

Họ và tên: Nguyễn Xuân Trường

Mã sinh viên: 2451060733

Lớp: 66CNTT2

Ngày làm: 23/02/2026

PHIẾU HỌC TẬP --- CÚ PHÁP CSS & BỘ CHỌN (SELECTORS)

Tài liệu đã xem (Link):

1. Kiến thức cốt lõi (Tóm tắt bằng lời của bạn)

1.1. CSS là gì? (Viết bằng lời riêng của bạn)

CSS là viết tắt của Cascading Style Sheets

Nó có vai trò gì trong 1 trang web?

- HTML: cấu trúc & nội dung (heading, đoạn văn, form, ảnh...).
- CSS: giao diện & bố cục (style, layout, animation đơn giản...).
- JavaScript: hành vi & tương tác (click, gọi API, validate, DOM, hiệu ứng động...).

1.2. Cú pháp cơ bản của CSS

Một quy tắc CSS (CSS Rule) gồm 2 phần:

```
selector {  
  property: value;  
}
```

Giải thích:

- **Selector** (bộ chọn): chọn phần tử HTML muốn áp dụng style (ví dụ: p, .btn, #header).
- **Property** (thuộc tính): tên thuộc tính cần chỉnh (ví dụ: color, padding, display).

- **Value** (giá trị): giá trị gán cho thuộc tính (ví dụ: red, 10px, flex).

1.3. Ba cách áp dụng CSS vào HTML

Hoàn thành bảng sau:

Cách áp dụng	Vị trí code	Ưu điểm	Nhược điểm
Inline CSS	Ngay trong thẻ HTML: <code>style=""</code>	Nhanh thử ngay, ưu tiên cao	Khó bảo trì, lặp code, "bẩn code"
Internal CSS	Trong <code><style>...</style></code> ở <code><head></code>	Gọn cho 1 trang, không cần file riêng	Vẫn khó scale nhiều trang, CSS không tái sử dụng tốt
External CSS	File .css riêng, link bằng <code><link></code>	Chuẩn thực tế, tái sử dụng, dễ bảo trì	Thêm 1 request tải file (nhưng thường được cache)

2. Các loại Selector cơ bản

Hoàn thành bảng sau:

Loại Selector	Cú pháp	Ví dụ code	Ý nghĩa
Universal Selector	<code>*</code>	<code>* { margin: 0; }</code>	Chọn tất cả các phần tử trong trang HTML.
Type Selector	<code>element</code>	<code>p { color: blue; }</code>	Chọn tất cả các phần tử có class tương ứng. Có thể dùng cho nhiều phần tử khác nhau.

Class Selector	<code>.className</code>	<code>.btn { padding: 10px; }</code>	<i>Chọn tất cả các phần tử có class tương ứng.</i>
ID Selector	<code>#idName</code>	<code>#header { background: gray; }</code>	<i>Chọn một phần tử duy nhất có id đó. ID phải không được trùng trong trang</i>
Descendant Selector	<code>parent child</code>	<code>nav a { text-decoration: none; }</code>	<i>Chọn các phần tử child nằm bên trong parent. (mọi cấp con cháu)</i>
Group Selector	<code>selector1, selector2</code>	<code>h1, h2, h3 { font-family: Arial; }</code>	<i>Áp dụng cùng một style cho nhiều selector cùng lúc.</i>
Pseudo-class Selector	<code>selector:state</code>	<code>a:hover { color: red; }</code>	<i>Chọn 1 phần tử khi nó ở một trạng thái đặc biệt.</i>

3. Bài tập thực hành 1 (20 phút): Áp dụng CSS & Selector

Mục tiêu:

- Tạo file `style.css` và liên kết vào HTML
- Dùng các loại selector: type, class, id, group, descendant, pseudo-class
- Thực hành styling cơ bản cho menu và header

Cho trước file HTML (không sửa cấu trúc):

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Lab CSS Cơ Bản</title>
  <!-- TODO: Liên kết file style.css vào đây -->
</head>
<body>
  <header id="main-header">
    <h1>My Blog</h1>
  </header>

  <nav>
    <a href="#" class="nav-link">Home</a>
    <a href="#" class="nav-link">About</a>
    <a href="#" class="nav-link">Contact</a>
  </nav>

  <main>
    <section class="intro">
      <h2>Chào mừng đến với Blog</h2>
      <p>Đây là trang blog về lập trình web.</p>
    </section>
  </main>

  <footer>
    <p>© 2026 My Blog. All rights reserved.</p>
  </footer>
</body>
</html>
```

Yêu cầu CSS (viết vào file style.css):

Bước 1: Liên kết file CSS vào HTML bằng thẻ <link> trong <head>.

```
<!-- TODO: Viết code liên kết CSS vào đây -->
```

Bước 2: Viết CSS cho các yêu cầu sau:

```

/* TODO 1: Dùng universal selector để reset margin và padding về 0 cho toàn trang */

/* TODO 2: Dùng type selector cho body */
/* Yêu cầu: font-family: Arial; background-color: #f5f5f5; */

/* TODO 3: Dùng ID selector cho #main-header */
/* Yêu cầu: background-color: #333; color: white; text-align: center; padding: 20px; */

/* TODO 4: Dùng class selector cho .nav-link */
/* Yêu cầu: color: #333; text-decoration: none; padding: 10px 15px; */

/* TODO 5: Dùng group selector cho h1, h2, h3 */
/* Yêu cầu: font-family: 'Georgia', serif; */

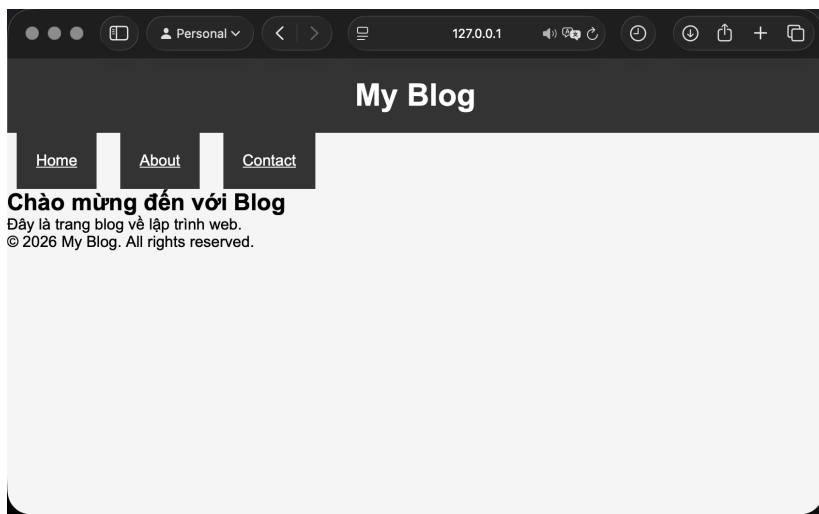
/* TODO 6: Dùng descendant selector cho nav a */
/* Yêu cầu: display: inline-block; margin: 0 10px; */

/* TODO 7: Dùng pseudo-class selector cho a:hover */
/* Yêu cầu: color: #ff6600; background-color: #eee; */

```

Kết quả mong đợi:

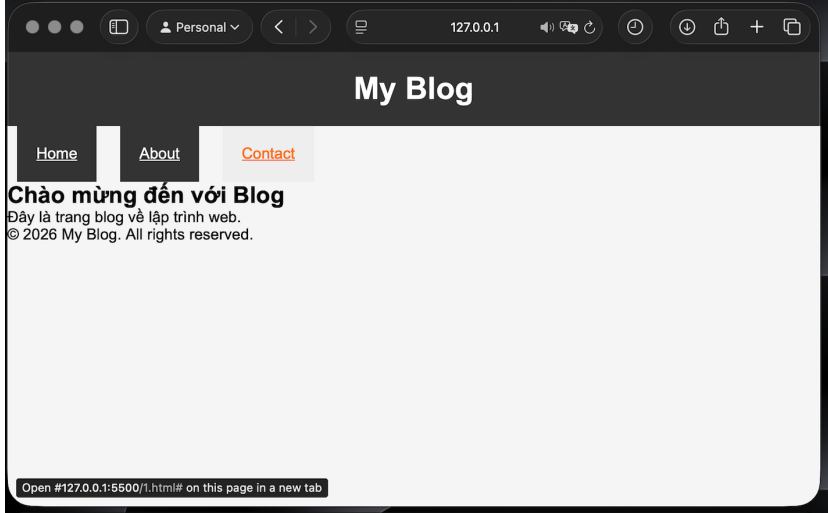
1. Menu nằm ngang, các link cách đều nhau
2. Khi rê chuột vào link, màu chữ và nền thay đổi
3. Header có nền tối, chữ trắng, căn giữa



Ảnh kết quả (Dán ảnh trình duyệt vào đây):

(Dán ảnh 1: Giao diện bình thường)

(Dán ảnh 2: Khi hover vào link)



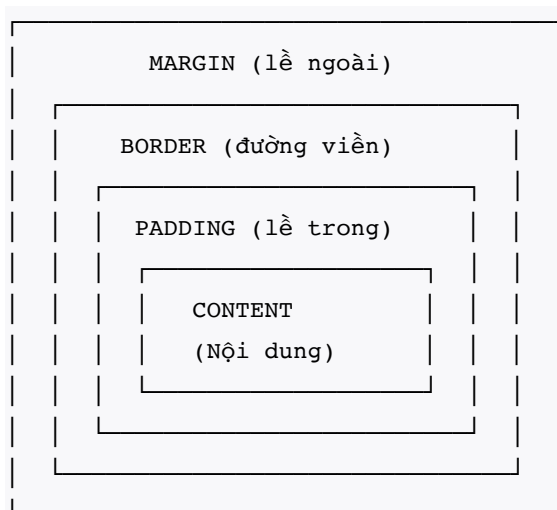
PHIẾU HỌC TẬP --- BOX MODEL & BỐ CỤC CƠ BẢN

Tài liệu đã xem (Link):

1. Kiến thức cốt lõi: CSS Box Model

1.1. Box Model gồm những thành phần gì?

Vẽ sơ đồ Box Model (từ trong ra ngoài):



Giải thích:

- **Content:** vùng chứa nội dung thật (text/ảnh).
- **Padding:** “đệm” bên trong, khoảng cách từ content ra border.
- **Border:** đường viền bao quanh padding + content.

- **Margin:** “lề ngoài”, khoảng cách giữa box này với box khác.

1.2. Hai cách tính kích thước Box

Hoàn thành bảng:

Thuộc tính	Cách tính width	Ưu/Nhược điểm
<code>box-sizing: content-box;</code>	total = width + padding + border	<i>Dễ hiểu “width là content”</i> <i>Dễ vỡ layout vì cộng thêm padding/border</i>
<code>box-sizing: border-box;</code>	total = width (đã g ồm padding + border)	<i>Dễ canh layout, ổn định, ít “vỡ”</i> <i>Khi tăng padding thì content giảm</i>

Câu hỏi: Nên dùng `box-sizing` nào cho toàn bộ trang? Tại sao?

Nên dùng `border-box` để tính kích thước ổn định, làm layout dễ hơn (đặc biệt khi làm card/grid/flex).

2. Bài tập thực hành 2 (35 phút): Box Model & Layout Card

Mục tiêu:

- Tạo 3 card bài viết với box model rõ ràng
- Dùng margin, padding, border để tách biệt các card
- Áp dụng `box-sizing: border-box`
- Bố trí card theo hàng ngang (dùng `display: inline-block` hoặc `float`)

Thêm HTML sau vào trong <main> (sau phần intro):

```
<section class="posts">
  <article class="card">
    <h3>Bài viết 1</h3>
```

```
<p>Mô tả ngắn về bài viết 1.</p>
<a href="#" class="btn">Đọc thêm</a>
</article>

<article class="card">
  <h3>Bài viết 2</h3>
  <p>Mô tả ngắn về bài viết 2.</p>
  <a href="#" class="btn">Đọc thêm</a>
</article>

<article class="card">
  <h3>Bài viết 3</h3>
  <p>Mô tả ngắn về bài viết 3.</p>
  <a href="#" class="btn">Đọc thêm</a>
</article>
</section>
```

Yêu cầu CSS:

```
/* Bước 1: Reset box-sizing cho toàn trang */
* {
  box-sizing: border-box;
}

/* Bước 2: Style cho .posts container */
.posts {
  /* TODO: Căn giữa, margin trên dưới, max-width */
}

/* Bước 3: Style cho .card */
.card {
  /* TODO:
    - width: 30% (hoặc calc(33.33% - 20px))
    - display: inline-block
    - border: 1px solid #ddd
    - padding: 20px
    - margin: 10px
    - background: white
    - border-radius: 8px
  */
}
}
```

```

/* Bước 4: Style cho .btn */
.btn {
  /* TODO:
    - display: inline-block
    - padding: 8px 16px
    - background: #007bff
    - color: white
    - text-decoration: none
    - border-radius: 4px
  */
}

.btn:hover {
  /* TODO: background: #0056b3 */
}

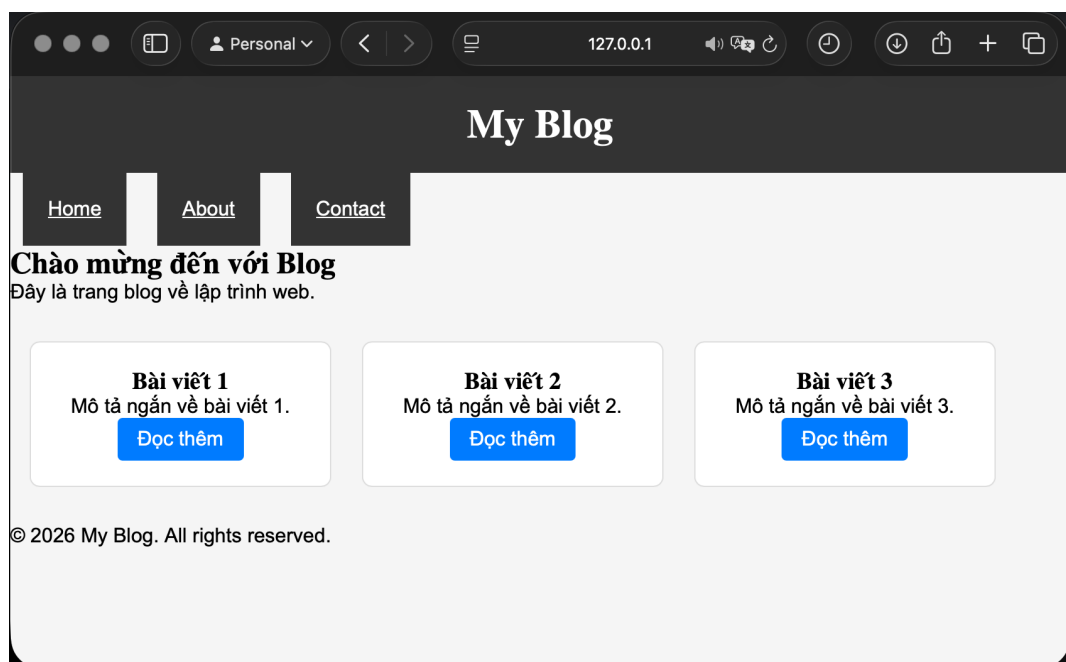
```

Kết quả mong đợi:

1. Ba card nằm ngang trên 1 hàng
2. Mỗi card có viền, padding, margin rõ ràng
3. Nút "Đọc thêm" có màu xanh, hover đổi màu đậm hơn

Ảnh kết quả (Dán ảnh vào đây):

(Dán ảnh: Giao diện 3 card)



PHIẾU HỌC TẬP --- FLEXBOX & CSS GRID

Tài liệu đã xem (Link):

1. Kiến thức cốt lõi: So sánh Flexbox và Grid

Hoàn thành bảng sau:

Đặc điểm	Flexbox	CSS Grid
Bố cục	1 chiều	2 chiều
Phù hợp với	Menu, navbar, list item, card 1 hàng...	Layout trang, gallery, dashboard nhiều hàng/cột
Thuộc tính chính	display: flex; justify-content align-items	display: grid; grid-template-columns grid-template-rows
Ví dụ sử dụng	Navbar căn giữa, card wrap	Gallery 3 cột, responsive tự đổi cột

1.1. Flexbox - Các thuộc tính quan trọng

```
.flex-container {
  display: flex;

  /* Căn chỉnh theo trục ngang (main axis) */
  justify-content: _____; /* flex-start | center | space-between | space-around
*/

  /* Căn chỉnh theo trục dọc (cross axis) */
  align-items: _____; /* flex-start | center | flex-end | stretch */

  /* Cho phép xuống hàng */
  flex-wrap: _____; /* nowrap | wrap */

  /* Khoảng cách giữa các item */
  gap: _____;
}
```

1.2. CSS Grid - Các thuộc tính quan trọng

```
.grid-container {
  display: grid;

  /* Định nghĩa số cột và kích thước */
  grid-template-columns: _____; /* repeat(3, 1fr) | 200px 1fr 2fr */

  /* Định nghĩa số hàng và kích thước */
  grid-template-rows: _____;

  /* Khoảng cách giữa các ô */
  gap: _____;
}
```

2. Bài tập thực hành 3A (20 phút): Flexbox - Menu & Card Layout

Mục tiêu:

- Dùng Flexbox cho menu ngang
- Dùng Flexbox cho layout card (thay thế inline-block)
- Responsive: card xuống 1 cột khi màn hình nhỏ

Yêu cầu CSS (cải tiến code cũ):

```
/* Cải tiến nav bằng Flexbox */
nav {
  /* TODO:
    - display: flex
    - justify-content: center
    - align-items: center
    - gap: 20px
    - background: #f8f9fa
    - padding: 15px
  */
}

/* Cải tiến .posts bằng Flexbox */
.posts {
```

```

    /* TODO:
      - display: flex
      - flex-wrap: wrap
      - gap: 20px
      - justify-content: center (hoặc space-between)
      - max-width: 1200px
      - margin: 40px auto
    */
  }

  .card {
    /* TODO:
      - flex: 0 0 calc(33.33% - 20px)
      - (hoặc flex: 1 1 300px cho responsive tự động)
    */
  }

  /* Responsive: Khi màn hình < 768px, card chiếm full width */
  @media (max-width: 768px) {
    .posts {
      /* TODO: flex-direction: column */
    }

    .card {
      /* TODO: flex: 1 1 100% */
    }
  }
}

```

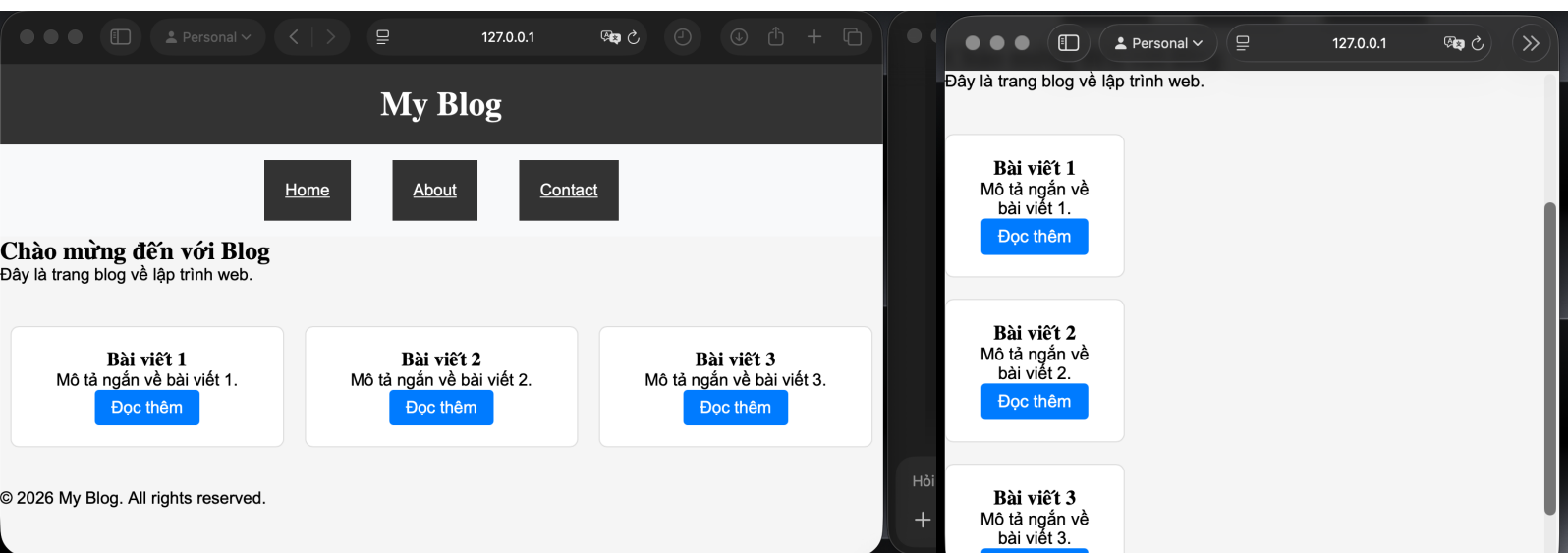
Kết quả mong đợi:

1. Menu các link nằm ngang, căn giữa, cách đều
2. Card layout linh hoạt, tự động xuống hàng khi hết chỗ
3. Khi thu nhỏ màn hình < 768px, card xếp thành 1 cột

Ảnh kết quả (Dán ảnh vào đây):

(Dán ảnh 1: Desktop view - 3 card ngang)

(Dán ảnh 2: Mobile view - card xếp dọc)



3. Bài tập thực hành 3B (20 phút): CSS Grid - Gallery Layout

Mục tiêu:

- Tạo lưới 2 hàng x 3 cột bằng CSS Grid
- Responsive: tự động giảm số cột khi màn hình nhỏ

Thêm HTML sau vào cuối `<main>`:

```
<section class="gallery">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
  <div class="item">5</div>
  <div class="item">6</div>
</section>
```

Yêu cầu CSS:

```
.gallery {
  /* TODO:
    - display: grid
    - grid-template-columns: repeat(3, 1fr) ← 3 cột bằng nhau
    - gap: 16px
    - max-width: 1200px
    - margin: 40px auto
  */
}

.item {
  /* TODO:
    - background: linear-gradient(135deg, #667eea 0%, #764ba2 100%)
    - color: white
    - display: flex
    - justify-content: center
    - align-items: center
    - height: 150px
    - font-size: 2rem
    - border-radius: 8px
  */
}

/* Responsive: Tự động giảm cột khi màn hình nhỏ */
.gallery {
  /* TODO: Dùng auto-fit/auto-fill */
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
}
```

Giải thích kỹ thuật:

repeat(auto-fit, minmax(200px, 1fr)) có nghĩa gì?

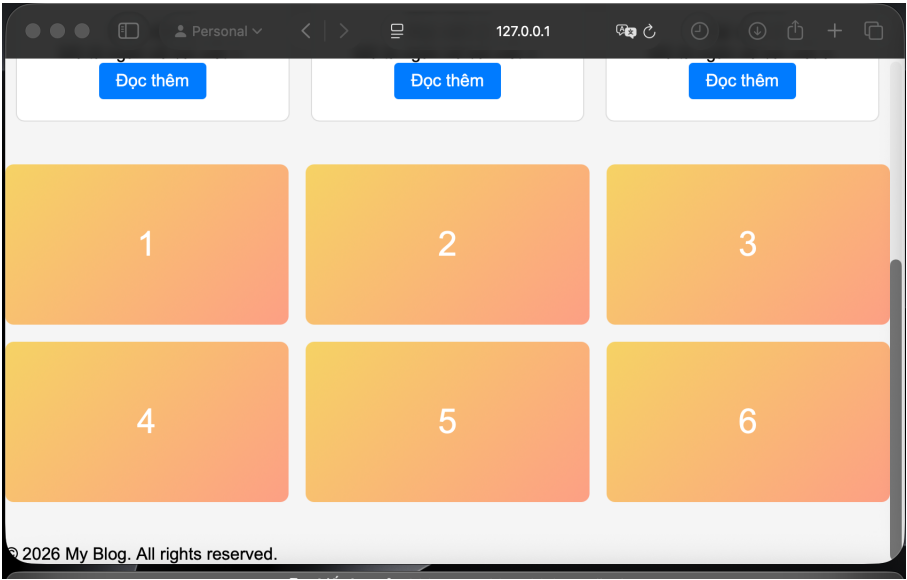
- **repeat()**: lặp lại một “mẫu cột” nhiều lần.
- **auto-fit**: tự “nhét” được bao nhiêu cột thì nhét; khi thiếu item, cột dư có thể co lại để các cột còn lại giãn ra.
- **minmax(200px, 1fr)**: mỗi cột tối thiểu 200px, tối đa chia đều phần còn lại (1fr).

Kết quả mong đợi:

- 1. Desktop: 3 cột, 2 hàng (6 ô)
- 2. Tablet: 2 cột, 3 hàng
- 3. Mobile: 1 cột, 6 hàng

Ảnh kết quả (Dán ảnh vào đây):

(Dán ảnh 1: Desktop - 3 cột)
(Dán ảnh 2: Mobile - 1 cột)



PHIẾU HỌC TẬP --- ĐỊNH VỊ CSS & RESPONSIVE DESIGN^[2]

Tài liệu đã xem (Link):

1. Kiến thức cốt lõi: CSS Position

Hoàn thành bảng:

Giá trị	Ý nghĩa	Ví dụ sử dụng
<code>static</code>	Mặc định, nằm theo luồng bình thường	<code>.box { position: static; }</code>

relative	<i>Vẫn giữ chỗ, nhưng có thể dịch bằng top/left...</i>	<i>Tooltip “neo” theo phần tử</i>
absolute	<i>Ra khỏi luồng, định vị theo ancestor gần nhất có position ! = static</i>	<i>Badge góc phải card</i>
fixed	<i>Cố định theo viewport, scroll vẫn đứng yên</i>	<i>Nút “scroll to top”</i>
sticky	<i>Bình thường cho đến khi scroll tới ngưỡng thì “dính”</i>	<i>Header dính trên cùng</i>

2. Responsive Design - Media Queries

Cú pháp cơ bản:

```
/* Desktop First */
@media (max-width: 768px) {
    /* Code CSS cho màn hình nhỏ hơn 768px */
}

/* Mobile First */
@media (min-width: 768px) {
    /* Code CSS cho màn hình lớn hơn 768px */
}
```

Breakpoints phổ biến:

- **Mobile:** < 576px
- **Tablet:** 576px - 768px
- **Desktop:** > 768px
- **Large Desktop:** > 1200px

3. Bài tập thực hành 4 (30 phút): Tạo Header Sticky + Responsive

Yêu cầu:

1. Header cố định trên cùng khi scroll (sticky)
2. Thêm nút "Scroll to Top" (fixed, góc dưới bên phải)
3. Responsive: Menu chuyển icon burger khi mobile (chỉ làm style, không cần JavaScript)

Cập nhật HTML:

```
<header id="main-header" class="sticky-header">
  <h1>My Blog</h1>
</header>

<!-- Nút Scroll to Top -->
<a href="#" class="scroll-top">↑</a>
```

Yêu cầu CSS:

```
/* Header sticky */
.sticky-header {
  position: sticky;
  top: 0;
  z-index: 100;
  /* TODO: Thêm box-shadow để tạo hiệu ứng nổi */
}

/* Nút Scroll to Top */
.scroll-top {
  position: fixed;
  bottom: 30px;
  right: 30px;
  /* TODO:
    - width: 50px, height: 50px
    - background: #007bff
    - color: white
    - border-radius: 50%
    - display: flex, justify/align center
    - text-decoration: none
    - font-size: 24px
  */
}
```

```

.scroll-top:hover {
  /* TODO: background: #0056b3, transform: scale(1.1) */
}

/* Responsive */
@media (max-width: 768px) {
  nav {
    /* TODO: flex-direction: column */
  }
}

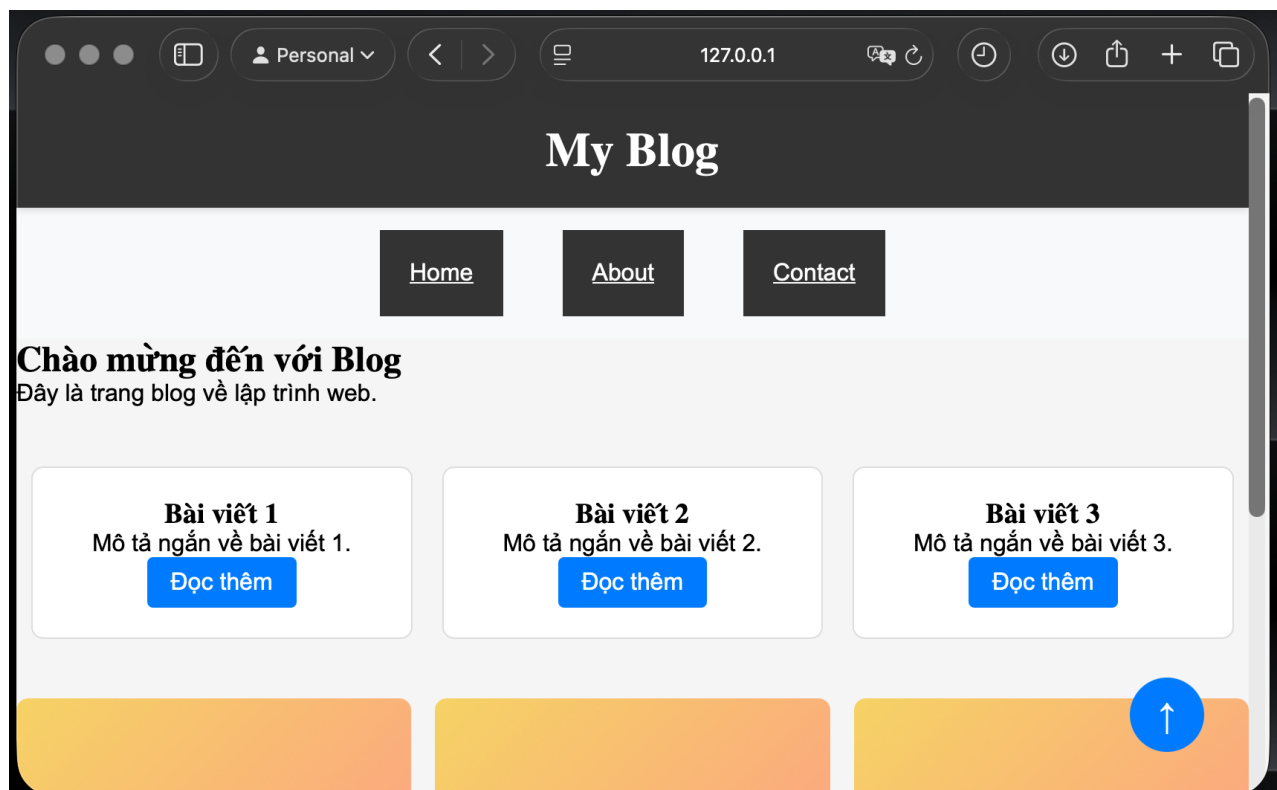
```

Kết quả mong đợi:

1. Header luôn cố định trên cùng khi scroll
2. Nút "↑" ở góc dưới bên phải, hover phóng to
3. Mobile: menu xếp dọc

Ảnh kết quả (Dán ảnh vào đây):

(Dán ảnh: Giao diện có header sticky và nút scroll)



MINI TEST - Tự Kiểm Tra

Phần 1: Selector & Cú pháp

1. Selector nào chọn tất cả thẻ `<p>` nằm trong `<div>`?

- A. `div p`
- B. `div > p`
- C. `div + p`

2. Pseudo-class `:hover` dùng để?

- A. Chọn phần tử đầu tiên
- B. Chọn phần tử khi rê chuột
- C. Chọn phần tử cuối cùng

Phần 2: Box Model

3. `box-sizing: border-box` có nghĩa?

- A. Width bao gồm content + padding + border
- B. Width chỉ tính content
- C. Width tính cả margin

4. Thứ tự từ trong ra ngoài của Box Model?

- A. Content → Padding → Border → Margin
- B. Content → Border → Padding → Margin
- C. Content → Margin → Padding → Border

Phần 3: Flexbox & Grid

5. Flexbox phù hợp với layout?

- A. 1 chiều (hàng hoặc cột)
- B. 2 chiều (hàng và cột)
- C. Không phù hợp với layout

6. `justify-content: space-between` có tác dụng?

- A. Căn giữa các item
- **B. Chia đều khoảng trống giữa các item**
- C. Chia đều khoảng trống cả 2 bên

7. Grid tốt hơn Flexbox khi nào?

- A. Menu ngang đơn giản
- **B. Layout phức tạp nhiều hàng cột**
- C. Không có sự khác biệt

Phần 4: Position & Responsive

8. position: fixed có đặc điểm?

- **A. Cố định so với viewport**
- B. Cố định so với phần tử cha
- C. Không cuộn theo trang

9. Media query @media (max-width: 768px) áp dụng cho?

- A. Màn hình > 768px
- **B. Màn hình ≤ 768px**
- C. Chỉ màn hình = 768px

Đáp án Mini Test

👉 Nhấn vào đây để xem đáp án

Phần 1:

1. **A - div p** (descendant selector)
2. **B - Chọn phần tử khi rê chuột**

Phần 2:

3. **A - Width** bao gồm content + padding + border
4. **A - Content** → Padding → Border → Margin

Phần 3:

- 5. **A** - 1 chiều (hàng hoặc cột)
- 6. **B** - Chia đều khoảng trống giữa các item
- 7. **B** - Layout phức tạp nhiều hàng cột

Phần 4:

- 8. **A** - Cố định so với viewport
- 9. **B** - Màn hình $\leq 768\text{px}$

 **TÓM TẮT CHƯƠNG (Self-Reflection)**

Ghi chú của giáo viên:

Ngày hoàn thành:

Chữ ký sinh viên: _____