Mục tiêu:

1. Hiểu về Responsive web design và Adaptive web design.
2. Biết cách sử dụng thẻ `meta` viewport để tối ưu hiển thị trên thiết bị di động.
3. Biết cách sử dụng `Media queries` để tạo giao diện đáp ứng.
4. Hiểu về Breakpoints và xu hướng thiết kế Mobile-first và Desktop-first.
5. Biết cách xử lý các vấn đề chính trong xây dựng giao diện đáp ứng.
6. Thực hành xây dựng giao diện website thích ứng đa thiết bị.

Responsive web design là gì?

Khái niệm

Responsive web design là một phương pháp thiết kế và phát triển trang web có thể hiển thị đẹp mắt và dễ đọc **trên mọi thiết bị**, từ máy tính để bàn, máy tính xách tay, máy tính bảng cho đến điện thoại di động.

Tại sao cần responsive web design?

Với sự phát triển của công nghệ, người dùng truy cập web từ nhiều thiết bị khác nhau. **Responsive web design** giúp trang web hiển thị đẹp mắt và dễ đọc trên mọi thiết bị, nâng cao hiệu suất và giúp cải thiện trải nghiệm người dùng.

Có rất nhiều vấn đề ảnh hưởng đến giao diện website và trải nghiệm người dùng khi truy cập từ các thiết bị khác nhau, sau đây là một số lý do chính cần xây dựng giao diện đáp ứng:

- **Layout**: Bố cục website và cách hiển thị khiến người đọc khó tiếp cận và tìm kiếm được thông tin cần thiết, người dùng phải cuộn ngang, thu phóng, hoặc di chuyển nhiều để xem nội dung cũng khiến trải nghiệm người dùng giảm đi.
- **Tốc độ tải trang**: Trang web không tối ưu cho thiết bị di động có thể tải chậm, gây khó chịu cho người dùng, tỷ lệ khách hàng rời bỏ trang web sẽ tăng cao.
- **Font chữ và cỡ chữ**: Font chữ và cỡ chữ không phù hợp với kích thước và thiết bị khiến giao diện trở nên khó đọc.
- **Màu sắc và độ tương phản**: Không phải màu sắc, sắc độ, độ tương phản ở cách thiết bị đều giống nhau, màu sắc và độ tương phản không phù hợp khiến giao diện trở nên khó nhìn và không thân thiện với người dùng.

> *Lưu ý: Một số website được xây dựng riêng cho từng loại thiết bị và không cần phải phát triển giao diện đáp ứng đa thiết bị, ví dụ như: các trang web phía quản trị, trang web quản lý nội dung, trang web quản lý kho hàng, v.v. đây đều là những website có tính đặc thù cao phục vụ đối tượng người dùng trên thiết bị cụ thể nên không cần xây dựng giao diện đáp ứng đa thiết bị.*

Còn rất nhiều lý do để chúng ta cân nhắc phát triển giao diện đáp ứng cho website. Tuy nhiên tuỳ vào các yêu cầu cụ thể của từng dự án mà chúng ta sẽ xác định lựa chọn giải pháp nào. Phía dưới là 2 giải pháp chính cho việc hoàn thiện ứng dụng đáp ứng đa thiết bị.

Responsive web design và Adaptive web design

Để có thể thân thiện với nhiều thiết bị và kích thước màn hình khác nhau chúng ta còn có một giải pháp khác ngoài Responsive web design đó là Adaptive web design.

Adaptive web design là phương pháp sử dụng nhiều thiết kế cố định khác nhau cho từng thiết bị cụ thể. Adaptive web design sẽ phân chia trang web thành nhiều phiên bản khác nhau, mỗi phiên bản sẽ được thiết kế riêng cho một loại thiết bị cụ thể. Từ đó giúp trang web hiển thị đẹp mắt và dễ đọc trên mọi thiết bị.

Responsive web design và **Adaptive web design** đều giúp trang web hiển thị đẹp mắt và dễ đọc trên mọi thiết bị, tuy nhiên cả hai phương pháp này có những ưu và nhược điểm riêng.

Tiêu chí	Responsive Web Design	Adaptive Web Design
Cách tiếp cận	Sử dụng một bố cục linh hoạt, điều chỉnh theo kích thước màn hình.	Sử dụng nhiều bố cục cố định, hiển thị dựa trên kích thước màn hình cụ thể.
Hệ thống lưới (Grid)	Lưới linh hoạt với các đơn vị tỷ lệ như %, em, rem.	Lưới cố định với các kích thước xác định trước.
Media Queries	Sử dụng rộng rãi, thay đổi bố cục liên tục dựa trên kích thước màn hình.	Sử dụng để chuyển đổi giữa các bố cục khác nhau tại các điểm ngắt xác định trước.
Số lượng bố cục	Một bố cục linh hoạt duy nhất cho mọi kích thước màn hình.	Nhiều bố cục (thường là 6) cho các nhóm kích thước màn hình cụ thể.
Hiệu suất	Tải tất cả các tài nguyên cần thiết, có thể ảnh hưởng đến hiệu suất trên thiết bị di động.	Tối ưu hóa tải tài nguyên theo từng thiết bị, hiệu suất tốt hơn trên các thiết bị di động.
Khả năng bảo trì	Dễ bảo trì hơn, chỉ cần duy trì một phiên bản mã nguồn.	Khó bảo trì hơn do phải quản lý nhiều bố cục và mã nguồn khác nhau.
Kiểm soát chi tiết giao diện	Ít kiểm soát chi tiết trên từng thiết bị, thiết kế linh hoạt có thể dẫn đến một số điểm không hoàn hảo.	Kiểm soát chi tiết tốt hơn, từng bố cục được tối ưu hóa cho từng thiết bị.
Thời gian phát triển	Nhanh hơn do chỉ cần thiết kế và phát triển một bố cục.	Chậm hơn do cần phát triển và tối ưu hóa nhiều bố cục khác nhau, như vậy cũng ảnh hưởng đến giá thành phát triển website.
Trải nghiệm người dùng	Trải nghiệm nhất quán trên mọi thiết bị, nhưng có thể không tối ưu hóa cho từng loại thiết bị.	Trải nghiệm người dùng được tối ưu hóa cao cho từng thiết bị cụ thể.
SEO	Thân thiện với SEO, một URL duy nhất cho mọi thiết bị.	Có thể gặp khó khăn nếu sử dụng nhiều URL khác nhau cho các thiết bị.

Tiêu chí	Responsive Web Design	Adaptive Web Design
Khả năng mở rộng	Linh hoạt hơn, dễ mở rộng để hỗ trợ các kích thước màn hình mới.	Khó mở rộng khi có kích thước màn hình mới, yêu cầu thêm bố cục mới.

> *Lưu ý: Các ứng dụng web hiện đại được đầu tư nhiều sẽ thường có phiên bản web cho người dùng và phiên bản ứng dụng cho người dùng trên thiết bị di động. Điều này giúp cải thiện trải nghiệm người dùng trên từng thiết bị cụ thể. Ví dụ: Facebook, Twitter, Instagram, Tiktok, v.v.*

Thẻ meta viewport

Viewport là gì?

Viewport hay còn được gọi là khung nhìn, là khu vực mà trình duyệt hiển thị nội dung của trang web, cũng là nơi người dùng nhìn thấy nội dung của website.

Việc website phải hiển thị trên nhiều thiết bị khác nhau gây ra khó khăn cho người phát triển web. Nhưng giờ đây chúng ta có thể khắc phục những điều này với thẻ meta viewport trong HTML.

HTML5 đã giới thiệu thẻ `<meta>` với thuộc tính `name="viewport"` để giúp chúng ta kiểm soát cách trình duyệt hiển thị nội dung trên thiết bị di động.

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

Thẻ meta này cung cấp cho trình duyệt một hướng dẫn về cách hiển thị trang web:

- `width=device-width`: Kích thước của viewport sẽ bằng với kích thước của thiết bị.
- `initial-scale=1.0`: Mức độ phóng to ban đầu của trang web.

Ví dụ về website cài đặt viewport đúng:

```
<style>
  body {
    font-family: Arial, sans-serif;
  }
  .container {
    width: 100%;
    padding: 10px;
    background-color: #f4f4f4;
  }
</style>

<div class="container">
  <h1>Welcome to My Website</h1>
  <p>
    This is a responsive website with a viewport meta tag. Lorem ipsum,
dolor
    sit amet consectetur adipisicing elit. Aliquam officiis accusamus
```

```
animi enim
    fugit nihil? Aliquid esse inventore quos est vitae dolorem ducimus
illo
    tempora adipisci unde sint odio soluta minima veritatis, perferendis
    expedita voluptatem quisquam labore. Quasi, inventore unde fugiat
suscipit
    iure corrupti doloremque porro officiis dolore ad vitae alias
provident
    eligendi ducimus labore tempora saepe dolores odio et nesciunt error
nam
    totam? Nostrum magni rerum at eveniet eius eum incidunt deleniti,
pariatur
    animi dicta commodi odit minima non quod explicabo rem alias
temporibus
    dolor harum nobis. Neque deserunt quam iure quos nobis maxime rem
quasi
    accusamus esse eligendi.
  </p>
</div>
```

Ví dụ về website không có thẻ meta cho viewport:

```
<style>
  body {
    font-family: Arial, sans-serif;
  }
  .container {
    width: 100%;
    padding: 10px;
    background-color: #f4f4f4;
  }
</style>

<div class="container">
  <h1>Welcome to My Website</h1>
  <p>
    This is a responsive website with a viewport meta tag. Lorem ipsum,
dolor
    sit amet consectetur adipisicing elit. Aliquam officiis accusamus
animi enim
    fugit nihil? Aliquid esse inventore quos est vitae dolorem ducimus
illo
    tempora adipisci unde sint odio soluta minima veritatis, perferendis
    expedita voluptatem quisquam labore. Quasi, inventore unde fugiat
suscipit
    iure corrupti doloremque porro officiis dolore ad vitae alias
provident
    eligendi ducimus labore tempora saepe dolores odio et nesciunt error
nam
    totam? Nostrum magni rerum at eveniet eius eum incidunt deleniti,
pariatur
    animi dicta commodi odit minima non quod explicabo rem alias
```

```
temporibus
    dolor harum nobis. Neque deserunt quam iure quos nobis maxime rem
quasi
    accusamus esse eligendi.
  </p>
</div>
```

Media queries

Media queries là một kỹ thuật được giới thiệu từ CSS3, nó khai báo một khối code CSS và khối code ấy chỉ được áp dụng với một số điều kiện cụ thể là đúng. Nhờ vậy chúng ta kiểm soát được cách tạo style cho trang web trong một số điều kiện cụ thể.

Cú pháp:

```
@media not|only mediatype and (media feature) and|or (media feature) {
  CSS-Code;
}
```

Trong đó:

- `not` | `only`: là những từ khoá cho biết áp dụng loại media nào.
- `mediatype`: Loại thiết bị mà bạn muốn áp dụng CSS. Sau đây là một số loại thiết bị phổ biến:
  - `all`: Tất cả các thiết bị.
  - `print`: Thiết bị in ấn.
  - `screen`: Màn hình hiển thị thông thường.
- `and` | `or`: là các từ khoá để kết hợp các điều kiện.
- `media feature`: là các kiểu điều kiện khác áp dụng kèm theo, sau đây là vài điều kiện phổ biến:
  - `width`: Chiều rộng của thiết bị.
  - `height`: Chiều cao của thiết bị.
  - `min-width`: Chiều rộng tối thiểu của thiết bị.
  - `max-width`: Chiều rộng tối đa của thiết bị.
  - `orientation`: Hướng của thiết bị (nhận giá trị `portrait` hoặc `landscape`).
  - `aspect-ratio`: Tỷ lệ khung hình của thiết bị (nhận giá trị `width/height`).
  - `resolution`: Độ phân giải của thiết bị (đơn vị là dpi hoặc dpcm).

Ví dụ:

```
<style>
  body {
    background-color: tan;
    color: black;
  }

  /* Áp dụng với độ rộng màn hình nhỏ hơn hoặc bằng 992px */
  @media screen and (max-width: 992px) {
```

```
    body {
      background-color: blue;
      color: white;
    }
  }

  /* Áp dụng với độ rộng màn hình nhỏ hơn hoặc bằng 600px */
  @media screen and (max-width: 600px) {
    body {
      background-color: olive;
      color: white;
    }
  }

  /* Áp dụng khi trang web được in */
  @media only print {
    body {
      background-color: white;
      color: black;
    }
  }
</style>

<h2>Thay đổi kích thước màn hình để thấy màu nền và màu chữ thay đổi!</h2>
<p>Bạn cũng có thể chọn thử chế độ in trang web để thấy sự thay đổi.</p>
```

Thực hành xây dựng giao diện website thích ứng đa thiết bị

Breakpoints và xu hướng thiết kế

Breakpoints là các điểm ngắt trong thiết kế giao diện đáp ứng, chúng giúp chúng ta xác định cách hiển thị giao diện trên các thiết bị khác nhau.

Có rất nhiều loại màn hình và chúng có chiều rộng khác nhau, rất khó để viết CSS cho tất cả các kích thước màn hình một cách tuyệt đối. Để đơn giản hoá việc này, người ta tạo ra các breakpoints (điểm ngắt) để phân chia các loại màn hình thành các nhóm theo kích thước chiều ngang của chúng. Sau đây là cách phân chia phổ biến:

```
@media only screen and (max-width: 600px) {
  /* Các thiết bị có kích thước hiển thị từ 600px trở xuống */
}

@media only screen and (min-width: 600px) {
  /* Các thiết bị có kích thước hiển thị từ 600px trở lên */
}

@media only screen and (min-width: 768px) {
  /* Các thiết bị có kích thước từ 768px trở lên */
}
```

```
@media only screen and (min-width: 992px) {
  /* Các thiết bị có kích thước hiển thị từ 992px trở lên */
}

@media only screen and (min-width: 1200px) {
  /* Các thiết bị có kích thước hiển thị từ 1200px trở lên */
}
```

Từ đây cũng hình thành 2 xu hướng tiếp cận khi xây dựng giao diện đáp ứng:

- **Mobile-first**: Bắt đầu xây dựng giao diện cho thiết bị di động trước, sau đó mở rộng cho các thiết bị lớn hơn.
- **Desktop-first**: Bắt đầu xây dựng giao diện cho máy tính để bàn trước, sau đó thu gọn cho các thiết bị nhỏ hơn.

Mobile-first

Mobile-first là một phương pháp thiết kế web bắt đầu bằng việc thiết kế giao diện và trải nghiệm người dùng cho các thiết bị di động trước tiên, sau đó mở rộng hoặc điều chỉnh thiết kế này cho các thiết bị có màn hình lớn hơn như máy tính bảng và máy tính để bàn.

Đặc điểm của Mobile First:

- **Ưu tiên di động**: Thiết kế và phát triển ban đầu tập trung vào các thiết bị di động với màn hình nhỏ, thường có kết nối mạng chậm hơn và tài nguyên hệ thống hạn chế.
- **Media Queries**: Trong phương pháp này, các quy tắc CSS mặc định áp dụng cho thiết bị di động (thường là các màn hình nhỏ hơn). Sau đó, media queries được sử dụng để mở rộng và điều chỉnh thiết kế cho các màn hình lớn hơn (min-width).

Ví dụ:

```
/* Thiết kế mặc định cho thiết bị di động */
.container {
  padding: 10px;
}

/* Mở rộng cho máy tính bảng và máy tính để bàn */
@media (min-width: 768px) {
  .container {
    padding: 20px;
  }
}

@media (min-width: 1024px) {
  .container {
    padding: 30px;
  }
}
```

Ưu điểm của Mobile First:

- **Tối ưu hóa hiệu suất:** Do bắt đầu với thiết kế tối giản cho thiết bị di động, trang web thường có hiệu suất tốt hơn trên các thiết bị có tài nguyên hạn chế.
- **Trải nghiệm người dùng:** Phương pháp này đảm bảo rằng trang web cung cấp trải nghiệm tốt trên các thiết bị di động, nơi mà phần lớn người dùng internet hiện nay đang truy cập.
- **Dễ dàng mở rộng:** Bắt đầu từ thiết bị di động giúp dễ dàng mở rộng thiết kế cho các thiết bị có màn hình lớn hơn mà không cần phải thu nhỏ hoặc loại bỏ các yếu tố thiết kế.

Ưu điểm của Mobile First:

- **Yêu cầu thay đổi tư duy thiết kế**: Nhà phát triển phải thay đổi tư duy từ việc thiết kế cho màn hình lớn sang tập trung vào các yếu tố thiết yếu cho màn hình nhỏ trước.
- **Có thể bỏ qua các yếu tố quan trọng cho màn hình lớn**: Khi tập trung quá nhiều vào thiết kế di động, có thể dẫn đến việc thiếu sự chú ý đến các chi tiết cần thiết cho trải nghiệm người dùng trên màn hình lớn hơn.

Desktop-first

Ngược lại với mobile-first, **desktop-first** là phương pháp thiết kế web truyền thống, trong đó việc thiết kế giao diện và trải nghiệm người dùng bắt đầu với các thiết bị có màn hình lớn như máy tính để bàn, sau đó được điều chỉnh và thu nhỏ cho các thiết bị có màn hình nhỏ hơn như máy tính bảng và điện thoại di động.

Đặc điểm của Desktop First:

- **Ưu tiên màn hình lớn**: Thiết kế và phát triển ban đầu tập trung vào các thiết bị có màn hình lớn, như máy tính để bàn, với việc tối ưu hóa trải nghiệm cho người dùng sử dụng chuột và bàn phím.
- **Media Queries**: Trong phương pháp này, các quy tắc CSS mặc định áp dụng cho các màn hình lớn. Sau đó, media queries được sử dụng để điều chỉnh thiết kế cho các màn hình nhỏ hơn (max-width).

Ví dụ:

```
/* Thiết kế mặc định cho máy tính để bàn */
.container {
  padding: 30px;
}

/* Thu nhỏ cho máy tính bảng */
@media (max-width: 1024px) {
  .container {
    padding: 20px;
  }
}

/* Thu nhỏ cho thiết bị di động */
@media (max-width: 768px) {
  .container {
    padding: 10px;
  }
}
```

Ưu điểm của Desktop First:

- **Thiết kế phức tạp:** Phương pháp này phù hợp cho các trang web phức tạp hoặc ứng dụng web, nơi trải nghiệm trên máy tính để bàn là trọng tâm.
- **Dễ thiết kế cho màn hình lớn:** Thiết kế ban đầu cho màn hình lớn cho phép nhà phát triển dễ dàng tích hợp nhiều tính năng và chi tiết hơn, sau đó có thể đơn giản hóa cho các thiết bị nhỏ hơn.

Nhược điểm của Desktop First:

- **Không tối ưu cho di động:** Phương pháp này có thể dẫn đến trải nghiệm người dùng kém trên các thiết bị di động, vì trang web có thể chứa quá nhiều chi tiết hoặc yếu tố không cần thiết.
- **Hiệu suất:** Khi thu nhỏ thiết kế từ màn hình lớn xuống nhỏ hơn, trang web có thể tải nhiều tài nguyên không cần thiết cho thiết bị di động, dẫn đến hiệu suất kém hơn.

Mỗi phương pháp đều có ưu nhược điểm riêng, từ yêu cầu thiết kế, hiệu suất, trải nghiệm người dùng cho đến khả năng mở rộng và bảo trì. Việc lựa chọn phương pháp phù hợp sẽ phụ thuộc vào yêu cầu cụ thể của dự án và đối tượng người dùng mà bạn đang phục vụ.

💻 **Bài tập nhỏ:**

Từ chính bài tập về nhà ở Buổi 9, hãy bổ sung thêm các breakpoints và media queries để tối ưu hóa giao diện cho các thiết bị di động.

LOL figures page

Những vấn đề chính trong xây dựng giao diện đáp ứng

Thiết kế layout linh hoạt

Vấn đề:

Bố cục trang web cần phải linh hoạt để có thể thay đổi và thích ứng với nhiều kích thước màn hình khác nhau. Một bố cục cố định sẽ không thể hoạt động tốt trên tất cả các thiết bị.

Giải pháp:

- **Sử dụng CSS Grid và Flexbox:** Đây là những công cụ mạnh mẽ cho phép bạn tạo ra các layout linh hoạt, dễ dàng điều chỉnh dựa trên kích thước màn hình.
- **Không sử dụng chiều rộng cố định:** Sử dụng các đơn vị tỷ lệ như phần trăm (%), em, rem thay vì pixel (px) để tạo ra các layout có thể tự động điều chỉnh theo kích thước màn hình.

Tối ưu hiển thị hình ảnh và media

Vấn đề:

Hình ảnh và phương tiện truyền thông (media) cần phải đáp ứng, tức là chúng phải tự động điều chỉnh kích thước dựa trên bố cục và kích thước màn hình. Ngoài ra, việc tải hình ảnh có thể ảnh hưởng đến hiệu suất của trang web.

Giải pháp:

- **Sử dụng loại hình ảnh linh hoạt:** Sử dụng hình ảnh có thể co giãn và thu nhỏ mà không làm mất chất lượng, ví dụ như dạng ảnh vector(SVG).
- **Sử dụng các thuộc tính như `srcset` và `<picture>`**: Cung cấp các phiên bản hình ảnh khác nhau cho các thiết bị khác nhau, giúp tối ưu hóa tải trang và hiển thị.
- **Lazy loading**: Tải hình ảnh khi người dùng cuộn trang đến vị trí của hình ảnh, giúp giảm thời gian tải trang và tăng hiệu suất (Kỹ thuật lazy loading sẽ được giới thiệu khi học đến Javascript).
- **Sử dụng thuộc tính `max-width: 100%`**: Đảm bảo rằng hình ảnh không vượt quá kích thước của phần tử chứa để ảnh không tràn ra ngoài màn hình hiển thị.

Phông chữ và kiểu chữ linh hoạt

Kích thước chữ (font-size), khoảng cách giữa các dòng (line-height), khoảng cách giữa các ký tự (letter-spacing), khoảng cách giữa các từ (word-spacing), loại phông chữ (font-family) đều là những thứ có tác động đến trải nghiệm đọc của người dùng. Lưu ý với các tiêu đề và các nội dung chính của trang web cần dễ đọc và dễ nắm bắt trên mọi thiết bị.

Quản lý điều hướng

Vấn đề:

Điều hướng trên các thiết bị di động với màn hình nhỏ có thể gặp khó khăn nếu không được thiết kế cẩn thận.

Giải pháp:

- **Sử dụng menu dạng hamburger:** Đây là một giải pháp phổ biến cho các thiết bị di động, giúp tiết kiệm không gian và giữ cho giao diện sạch sẽ (phần này sẽ được hướng dẫn chi tiết ở bài học về bootstrap).
- **Sử dụng điều hướng dạng drop-down hoặc accordion:** Điều này giúp điều hướng trở nên nhỏ gọn hơn và dễ quản lý trên màn hình nhỏ.
- **Sử dụng nút cuộn lên đầu trang:** Điều này giúp người dùng dễ dàng quay lại đầu trang mà không cần phải cuộn lên.
- **Cân nhắc vị trí các phần tử điều hướng** Ở nhiều thiết bị khác nhau, tay cầm, ngón thao tác và cách thao tác sẽ khác nhau dẫn đến cần điều chỉnh vị trí các phần tử điều hướng theo từng thiết bị. Ví dụ: nút điều hướng trên điện thoại đặt ở cuối trang để ngón cái dễ dàng tiếp cận, hoặc thay nút điều hướng bằng cử chỉ chạm, vuốt.

Hiệu suất và tối ưu hoá

Vấn đề:

Trang web đáp ứng có thể phải tải nhiều tài nguyên, gây ra vấn đề về hiệu suất, đặc biệt là trên các thiết bị di động với kết nối mạng chậm hơn.

Giải pháp:

- **Tối ưu hóa hình ảnh**: Sử dụng các định dạng hình ảnh hiện đại như `WebP` và các kỹ thuật nén để giảm kích thước tệp.
- **Minify và nén CSS/JavaScript**: Giảm thiểu kích thước tệp bằng cách minify (rút gọn) và nén để giảm thời gian tải.

- **Sử dụng kỹ thuật tải tài nguyên có chọn lọc**: Tải các tài nguyên như CSS và JavaScript một cách có chọn lọc dựa trên thiết bị hoặc viewport.

Khả năng truy cập (accessibility)

Vấn đề:

Đảm bảo rằng trang web có thể truy cập được đối với tất cả người dùng, bao gồm cả những người có khuyết tật.

Giải pháp:

- **Sử dụng các đơn vị linh hoạt cho kích thước văn bản:** Đảm bảo rằng người dùng có thể thay đổi kích thước văn bản dễ dàng mà không làm hỏng bố cục, thử nghiệm với phóng to hoặc thu nhỏ trang web để xem trang web có hiển thị đúng không.
- **Đảm bảo độ tương phản cao:** Sử dụng các màu sắc có độ tương phản cao để văn bản dễ đọc hơn, đặc biệt đối với người dùng có thị lực kém.
- **Cung cấp các tùy chọn điều hướng thay thế:** Sử dụng các thẻ HTML có nghĩa để hỗ trợ các công cụ hỗ trợ, như screen readers.

Nội dung quan trọng và nội dung ẩn (content prioritization and content hiding)

Không phải nội dung nào cũng thực sự quan trọng và cần thiết cho mọi thiết bị. Đôi khi, chúng ta cần ẩn hoặc ưu tiên hiển thị nội dung quan trọng hơn trên các thiết bị di động. Từ đó lựa chọn ẩn các nội dung không cần thiết hoặc chuyển chúng sang một trang khác khi xây dựng giao diện cho thiết bị nhỏ, từ đó nâng cao trải nghiệm người dùng.

💻 **Bài tập nhỏ:**

Xây dựng giao diện đáp ứng cho trang web giáo dục dựa trên thiết kế Figma sau (lựa chọn 3 primary breakpoints cho trang web):

Education Landing Page

Kiến thức cần nhớ

1. **Responsive Web Design** và **Adaptive Web Design** là hai phương pháp thiết kế giao diện đáp ứng, mỗi phương pháp có ưu và nhược điểm riêng. Tuy nhiên Responsive web design là phương pháp phổ biến và được ưa chuộng hơn.
2. **Viewport** là khu vực mà trình duyệt hiển thị nội dung của trang web, thẻ meta viewport giúp kiểm soát cách hiển thị trên thiết bị di động.
3. **Media queries** là công cụ giúp kiểm soát cách hiển thị giao diện trên các thiết bị khác nhau, giúp tạo ra giao diện đáp ứng.
4. **Breakpoints** là các điểm ngắt trong thiết kế giao diện đáp ứng, giúp chia các thiết bị thành các nhóm dựa trên kích thước chiều ngang vùng hiển thị của thiết bị.
5. **Mobile-first** và **Desktop-first** là hai phương pháp thiết kế giao diện đáp ứng, mỗi phương pháp có ưu và nhược điểm riêng, tuỳ vào yêu cầu thực tế của thiết kế để lựa chọn phương pháp phù hợp.
6. Khi xây dựng giao diện đáp ứng đa thiết bị cần lưu ý các vấn đề chính như thiết kế layout linh hoạt, hình ảnh, phông chữ và kiểu chữ linh hoạt, quản lý điều hướng, hiệu suất và tối ưu hóa, khả năng truy

cập và nội dung quan trọng và cân nhắc những nội dung có thể ẩn.