HTML Media & Embedding

Phần 4: Hình ảnh, Video, Audio & Embedding

Giảng viên: Dũng KT - TLU | 2024

Mục tiêu học tập

🖼️ **Responsive Images:** Hiểu và triển khai srcset, sizes, picture element.

🎨 **SVG Graphics:** Sử dụng ảnh vector Scalable Vector Graphics hiệu quả.

📹 **Video & Audio:** Nhúng đa phương tiện với các thẻ HTML5 native.

🔗 **External Embeds:** Tích hợp nội dung bên ngoài (YouTube, Maps) qua Iframe.

⚡ **Optimization:** Tối ưu hóa media cho hiệu năng tải trang và trải nghiệm.

Img element cơ bản — src & alt

Đặc điểm chính

- Là **Void Element** (phần tử rỗng), không có thẻ đóng </img>.
- Hiển thị hình ảnh raster (JPG, PNG, GIF, WebP) hoặc vector (SVG).

Thuộc tính bắt buộc

- **src** (source): Đường dẫn tới tệp hình ảnh (URL tuyệt đối hoặc tương đối).
- **alt** (alternative text): Mô tả nội dung ảnh, bắt buộc vì lý do accessibility và SEO.

```
<!-- Absolute URL -->
<img
  src="https://example.com/logo.png"
  alt="Company Logo">
<!-- Relative Path -->
<img
  src="/images/banner.jpg"
  alt="Summer Sale Banner">
```

💡 *Lưu ý: Nếu ảnh chỉ dùng để trang trí (decorative), hãy để alt="" (chuỗi rỗng) để trình đọc màn hình bỏ qua nó.*

Alt text — Tại sao quan trọng?

Accessibility
Giúp người khiếm thị hiểu nội dung hình ảnh thông qua trình đọc màn hình (Screen Readers).

SEO
Cung cấp ngữ cảnh cho Google Images, giúp cải thiện thứ hạng tìm kiếm hình ảnh.

Fallback
Hiển thị văn bản thay thế khi hình ảnh bị lỗi hoặc kết nối mạng quá chậm.

Ví dụ minh họa:

❌ **Bad:** alt="Image" hoặc alt="picture.jpg"

✅ **Good:** alt="Mèo đen đang ngủ trên ghế sofa"

Image attributes — width, height

Tại sao cần khai báo kích thước?

- **Tránh Layout Shift (CLS):** Trình duyệt sẽ dành sẵn khoảng trống cho ảnh trước khi tải xong.
- **Aspect Ratio tự động:** Các trình duyệt hiện đại tự tính toán tỷ lệ khung hình từ width/height attributes.
- **Tối ưu UX:** Người dùng không bị nhảy nội dung (nhảy dòng) khi đang đọc.

Lưu ý:

Sử dụng CSS để đảm bảo ảnh responsive (max-width: 100%; height: auto) trong khi vẫn giữ width/height HTML để giữ chỗ.

```
<img
  src="photo.jpg"
  width="800"
  height="600"
  alt="...">
```

Image attributes — lazy loading

Native Lazy Loading

- **Cơ chế:** Trì hoãn tải hình ảnh cho đến khi người dùng cuộn tới gần vị trí hiển thị (viewport).
- **Lợi ích:** Giảm lượng dữ liệu tải ban đầu (initial load size).
- **Hiệu năng:** Cải thiện chỉ số Largest Contentful Paint (LCP) và tiết kiệm băng thông mạng.

⚠️ Lưu ý quan trọng:

KHÔNG sử dụng loading="lazy" cho ảnh nằm trong màn hình đầu tiên (above-the-fold/Hero Image) vì sẽ làm chậm hiển thị.

```
<img
  src="image.jpg"
  alt="..."
  loading="lazy">
```

Scroll to Load

Trình duyệt tự động phát hiện khi ảnh vào khung nhìn

Image attributes — decoding

Attribute Values

- **sync:** Giải mã đồng bộ. Chặn render các nội dung khác cho đến khi ảnh được giải mã xong. Mặc định cho trình duyệt.
- **async:** Giải mã bất đồng bộ. Cho phép trình duyệt tiếp tục render nội dung khác, giảm giật lag (jank).
- **auto:** Để trình duyệt tự quyết định (thường là sync).

Khuyến nghị:

Sử dụng **decoding="async"** cho các ảnh không quan trọng (off-screen) hoặc ảnh lớn để tránh chặn Main Thread.

```
<img
  src="heavy-image.jpg"
  alt="..."
  decoding="async">
```

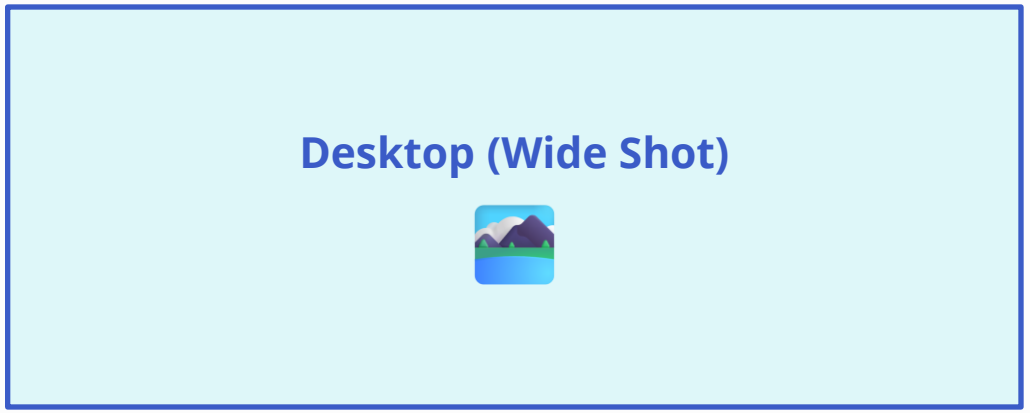
Performance Tip:
Kết hợp loading="lazy" và decoding="async" cho ảnh dưới nếp gấp (below-the-fold) để tối ưu hiệu năng tối đa.

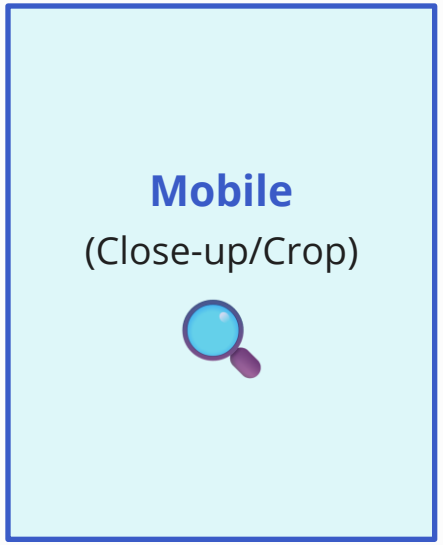
Picture element — Art direction

Art Direction là gì?

Là kỹ thuật thay đổi hoàn toàn bố cục hoặc crop của ảnh dựa trên kích thước màn hình, thay vì chỉ thay đổi độ phân giải.

```
<picture>
  <source media="(min-width: 800px)"
          srcset="desktop-wide.jpg">
  <img src="mobile-crop.jpg"
       alt="...">
</picture>
```





Responsive images — srcset & sizes

```
<img src="small.jpg"
    srcset="small.jpg 400w, medium.jpg 800w, large.jpg 1200w"
    sizes="(max-width: 600px) 100vw, 800px"
    alt="Responsive example">
```

1. srcset (Danh sách nguồn)

Cung cấp danh sách các file ảnh cùng với chiều rộng thực của chúng (width descriptor: **w**).
Ví dụ: 400w nghĩa là ảnh rộng 400 pixels.

2. sizes (Điều kiện hiển thị)

Cho trình duyệt biết ảnh sẽ chiếm bao nhiêu không gian trên màn hình (CSS width) tại các breakpoint khác nhau.
Ví dụ: Dưới 600px thì ảnh chiếm 100% width, ngược lại chiếm 800px.

💡 *Trình duyệt tự động tính toán và tải ảnh phù hợp nhất dựa trên kích thước màn hình và DPR.*

SVG — SVG như image/background

External SVG (Image Tag)

```
<img src="icon.svg" alt="...">
```

Ưu điểm:

- Đơn giản, code gọn.
- Trình duyệt cache file được.

Nhược điểm:

- KHÔNG thể đổi màu bằng CSS.
- Khó tương tác các phần tử con.

External SVG (Background)

```
background-image: url('bg.svg');
```

Ưu điểm:

- Tốt cho họa tiết trang trí, lặp lại.
- Giữ HTML sạch sẽ (semantic).

Nhược điểm:

- Cũng không thể đổi màu SVG.
- Không có alt text (không tốt cho SEO/A11y nếu là ảnh nội dung).

Picture element — Format selection

Tại sao dùng Format Selection?

- **Tối ưu kích thước:** Các định dạng mới như **WebP** hoặc **AVIF** thường nhẹ hơn JPEG/PNG từ 30-50% với cùng chất lượng.
- **Tương thích ngược (Fallback):** Trình duyệt không hỗ trợ WebP sẽ tự động bỏ qua thẻ <source> và tải thẻ <img> (JPEG/PNG).

Cơ chế hoạt động:

Trình duyệt duyệt qua các thẻ source, chọn định dạng đầu tiên mà nó hỗ trợ (type attribute), và tải ảnh đó.

```
<picture>
  <!-- Modern Format -->
  <source srcset="img.webp"
          type="image/webp">
  <!-- Fallback -->
  <img src="img.jpg"
       alt="...">
</picture>
```

WebP
~50KB

JPEG
~85KB

SVG — Inline SVG

Đặc điểm Inline SVG

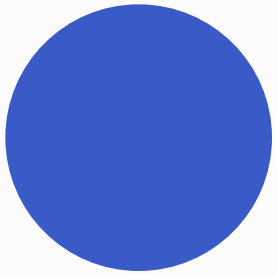
- Nhúng trực tiếp mã SVG vào tài liệu HTML.
- **Vector:** Co giãn thoải mái mà không bị vỡ hạt (pixelated).
- **Nhẹ:** Kích thước file thường rất nhỏ cho icons/logos.

Ưu điểm lớn nhất: CSS Styling

- Có thể thay đổi fill (màu nền), stroke (viền) bằng CSS.
- Hỗ trợ hiệu ứng tương tác (hover, animation).

```
<svg width="100" height="100">
  <circle cx="50" cy="50"
        r="40"
        fill="indigo"
        class="my-icon"/>
</svg>
```

Render →



Video — Ví dụ hoàn chỉnh

```
<!-- Khai báo kích thước, poster và controls -->
<video width="800" height="450"
       controls
       poster="images/poster-frame.jpg"
       preload="metadata">
  <source src="movie.mp4" type="video/mp4">
  Trình duyệt của bạn không hỗ trợ thẻ video.
</video>
```

Tại sao dùng preload="metadata"?

Chỉ tải thông tin cơ bản (thời lượng, kích thước) thay vì tải toàn bộ file. Tiết kiệm băng thông nếu người dùng không bấm play.

Tại sao dùng Poster?

Cung cấp ngữ cảnh nội dung ngay lập tức, giúp vùng video không bị trống hoặc đen xì trước khi tải xong.

Video element — Attributes cơ bản

Các thuộc tính quan trọng

- controls: Hiển thị thanh điều khiển mặc định (Play, Volume, Fullscreen).
- autoplay: Tự động phát khi tải trang (thường bị chặn nếu không có muted).
- muted: Tắt tiếng mặc định (bắt buộc nếu muốn autoplay hoạt động ổn định).
- loop: Lặp lại video vô hạn.
- poster: Ảnh hiển thị trước khi video bắt đầu phát.
- preload: Gợi ý tải trước (auto, metadata, none).

```
<video src="clip.mp4"
       controls
       muted
       poster="thumb.jpg">
</video>
```

💡 ***Lưu ý:*** *Trình duyệt hiện đại chặn auto-play có tiếng để tránh làm phiền người dùng. Luôn thêm muted nếu dùng autoplay.*

Video — Multiple sources & fallback

Đa định dạng (Multiple Formats)

- Không phải trình duyệt nào cũng hỗ trợ mọi codec.
- **MP4 (H.264):** Hỗ trợ rộng rãi nhất (Safari, IE, Chrome).
- **WebM (VP9/AV1):** Tối ưu dung lượng tốt hơn cho Chrome, Firefox.

Fallback Text

Nội dung bên trong thẻ <video> sẽ hiển thị nếu trình duyệt không hỗ trợ HTML5 Video (rất hiếm gặp hiện nay, nhưng tốt cho legacy support).

```
<video controls>
  <!-- Ưu tiên format nhẹ hơn -->
  <source src="v.webm"
          type="video/webm">
  <!-- Fallback phổ biến -->
  <source src="v.mp4"
          type="video/mp4">
  <p>Không hỗ trợ video</p>
</video>
```

Video — Responsive & Performance

Responsive Video

Đảm bảo video co giãn tốt trên mọi thiết bị bằng CSS đơn giản.

```
video {
  max-width: 100%;
  height: auto;
}
```

Performance Checklist

- **Preload Strategy:** Sử dụng preload="metadata" hoặc none. Tránh auto trừ khi video rất quan trọng.
- **Poster Image:** Luôn cung cấp poster để người dùng hiểu nội dung mà không cần tải video.
- **Data Saver:** Không autoplay trên mobile (data 3G/4G).
- **Lazy Loading:** Có thể dùng Intersection Observer để chỉ tải video khi cuộn tới (nếu có nhiều video).

Audio element — Cơ bản

Thẻ <audio>

Dùng để nhúng âm thanh vào trang web (nhạc, podcast, hiệu ứng).

Attributes tương tự Video:

- src: Đường dẫn file âm thanh.
- controls: Hiển thị trình điều khiển (Play, Pause, Volume).
- autoplay: Tự động phát (ít bị chặn hơn video nhưng vẫn cần cân nhắc UX).
- loop: Lặp lại.
- preload: metadata, auto, none.

Khác biệt: Không có thuộc tính width, height, poster.

```
<audio src="song.mp3"
    controls
    loop>
</audio>
```

Audio — Multiple sources

Định dạng âm thanh phổ biến

- **MP3:** Hỗ trợ tốt nhất trên mọi trình duyệt, chuẩn công nghiệp.
- **OGG / WAV:** Định dạng mở hoặc chất lượng cao (WAV), nhưng hỗ trợ kém hơn MP3 một chút hoặc dung lượng lớn (WAV).

Best Practice:

Luôn cung cấp MP3 làm định dạng chính hoặc fallback để đảm bảo ai cũng nghe được.

```
<audio controls>
  <!-- Source 1 -->
  <source src="audio.mp3"
          type="audio/mpeg">
  <!-- Source 2 -->
  <source src="audio.ogg"
          type="audio/ogg">
  Trình duyệt không hỗ trợ.
</audio>
```

Iframe — Embed external content

Chức năng

Tạo một ngữ cảnh duyệt web lồng nhau (nested browsing context), cho phép nhúng một trang HTML khác vào trang hiện tại.

Thuộc tính quan trọng

- src: URL của trang muốn nhúng.
- title: Mô tả nội dung iframe (quan trọng cho Accessibility).
- loading="lazy": Trì hoãn tải iframe.
- sandbox: Giới hạn quyền (scripts, forms) để bảo mật.

```
<iframe
  src="https://example.com"
  width="500" height="300"
  title="Mô tả nội dung"
  loading="lazy"
  sandbox>
</iframe>
```

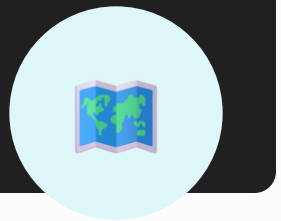
⚠️ Iframe có thể gây lỗ hổng bảo mật (clickjacking) nếu không kiểm soát nguồn nhúng.

Embedding Google Maps — Iframe

Best Practices

- **Lazy Loading:** Bản đồ Google Maps rất nặng. Luôn dùng loading="lazy" để không làm chậm tải trang ban đầu.
- **Accessibility:** Thuộc tính title giúp người khiếm thị biết đây là bản đồ khu vực nào.
- **Responsive:** Áp dụng kỹ thuật tương tự YouTube (width: 100%) để bản đồ vừa vặn với container.

```
<iframe
  src="https://google.com/maps..."
  width="600" height="450"
  style="border:0;"
  allowfullscreen=""
  loading="lazy"
  title="Bản đồ đường đi tới TLU"
></iframe>
```



Embedding YouTube — Responsive

1. Standard Embed (Fixed Size)

```
<!-- Copy từ YouTube Share -->
<iframe width="560" height="315"
  src="https://www.youtube.com/..."
  title="YouTube video player"
  allowfullscreen></iframe>
```

2. Responsive (Modern CSS)

```
/* CSS */
iframe {
  width: 100%;
  aspect-ratio: 16 / 9;
}
```

Vấn đề: Iframe mặc định có kích thước cố định (px), sẽ bị vỡ layout trên mobile.
Giải pháp: Sử dụng aspect-ratio: 16/9 để giữ tỷ lệ khung hình chuẩn trong khi width co giãn 100%.

Object & Embed (legacy) — Không khuyến nghị

⛔ **Deprecated / Legacy Technologies**

Về <object> và <embed>

- Trước đây dùng để nhúng Flash, Java Applets, ActiveX (đều đã "chết" trên web hiện đại).
- Hiện nay hiếm khi sử dụng, ngoại trừ nhúng PDF (nhưng iframe hoặc PDF.js tốt hơn).

Tại sao tránh?

- Hỗ trợ mobile kém.
- Lỗ hổng bảo mật tiềm ẩn.
- Khó tương tác bằng JS/CSS.

```
<!-- Don't do this -->
<object data="flash.swf"></object>
<embed src="music.mid">
```

Giải pháp thay thế:

Sử dụng <iframe>, <video>, <audio> chuẩn HTML5.

Best practices — Image optimization

- ✅ **Format:** Ưu tiên WebP hoặc AVIF cho web, dùng JPEG/PNG làm fallback.
- ✅ **Compression:** Luôn nén ảnh (TinyPNG, ImageOptim) để giảm dung lượng thừa.
- ✅ **Responsive:** Sử dụng `srcset` và `sizes` để tải đúng kích thước cho từng thiết bị.
- ✅ **Lazy Loading:** Dùng `loading="lazy"` cho ảnh dưới nếp gấp (off-screen).
- ✅ **Layout Stability:** Luôn khai báo `width` và `height` để tránh CLS.
- ✅ **Accessibility:** Không bao giờ quên thuộc tính `alt` mô tả nội dung.

Best practices — Video optimization

- ✅ **Format:** Cung cấp WebM (VP9) cho hiện đại và MP4 (H.264) cho tương thích.
- ✅ **Compression:** Sử dụng Handbrake hoặc FFmpeg để nén video (bitrate phù hợp).
- ✅ **Poster:** Luôn có ảnh thumbnail nhẹ để hiển thị ngay lập tức.
- ✅ **Preload:** Sử dụng `preload="metadata"` thay vì `auto` để tiết kiệm băng thông.
- ✅ **Autoplay Policy:** Nếu autoplay, bắt buộc phải có `muted` và `playsinline` (trên mobile).

Best practices — Accessibility

✅ **Images:** Luôn cung cấp alt text có ý nghĩa (hoặc rỗng nếu là ảnh trang trí).
✅ **Iframes:** Sử dụng thuộc tính title để mô tả nội dung nhúng cho trình đọc màn hình.
✅ **Video/Audio:** Cung cấp **Captions** (phụ đề) cho người khiếm thính.
✅ **Transcripts:** Cung cấp bản gỡ băng (văn bản) cho nội dung âm thanh/video dài.
✅ **Control:** Tránh auto-play âm thanh bất ngờ; người dùng phải có quyền kiểm soát phát/dừng.

Thực hành: Responsive Images

Bài tập 1: Tối ưu hiển thị ảnh

1. Tạo thẻ <img> sử dụng srcset và sizes để trình duyệt chọn ảnh 1x/2x phù hợp.
2. Sử dụng thẻ <picture> để thực hiện **Art Direction**:
   - Desktop: Ảnh phong cảnh rộng (landscape).
   - Mobile: Ảnh chân dung cắt cận cảnh (portrait/crop).
3. Mở DevTools (Network tab), thay đổi kích thước trình duyệt và reload để kiểm chứng file ảnh nào được tải.
4. Thêm width/height để đảm bảo không bị Layout Shift.

Thực hành: Video Player

Bài tập 2: Xây dựng Video Player

1. Nhúng một video vào trang web với đầy đủ controls và ảnh poster.
2. Cung cấp ít nhất 2 định dạng nguồn (ví dụ: MP4 và WebM) để đảm bảo tương thích.
3. Viết CSS để video hiển thị **Responsive** (width 100%, max-width hợp lý).
4. Thêm thuộc tính preload="metadata" và kiểm tra tab Network để xem trình duyệt tải bao nhiêu dữ liệu ban đầu.

Thực hành: Embed Content

Bài tập 3: Tích hợp nội dung bên ngoài

1. **YouTube:** Nhúng một video YouTube bất kỳ vào trang web.
   - Yêu cầu: Video phải hiển thị responsive (tỷ lệ 16:9) khi thay đổi kích thước trình duyệt.
2. **Google Maps:** Nhúng bản đồ vị trí trường Đại học Thủy Lợi (hoặc nhà bạn).
   - Yêu cầu: Sử dụng `loading="lazy"` và có `title` mô tả.
3. **Testing:** Sử dụng Device Toolbar (F12) để kiểm tra hiển thị trên iPhone/iPad.

Tóm tắt — Media & Embedding

Images

Luôn có alt, width, height. Dùng srcset cho responsive.

SVG

Inline để style bằng CSS. External để cache. Vector sắc nét.

Picture

Art Direction (crop) và Format Selection (WebP).

Video / Audio

Controls, poster, preload. Đa dạng sources (MP4/WebM).

Iframe

Nhúng nội dung ngoài. Chú ý loading="lazy" và bảo mật.

Performance

Nén ảnh/video, lazy loading, decoding async.

Q & A

📧 dungkt@tlu.edu.vn

📚 Tham khảo: MDN Web Docs - HTML Media & Embedding