TÓM TẮT 89 THẺ TAG VÀ THUỘC TÍNH CỦA THẺ HTML

13.Thẻ tag <body>

13.1.Bản chất

-Chứ TOÀN BỘ GIAO Diện

-Người dùng chỉ nhìn thấy nội dung trong body

13.1.Cú pháp #

<body> … </body>

13.2.Examble

<html>

<head>

    <meta charset=”UTF-8”>

    <title> tên tiêu đề tab </title>

</head>

    <h1> Bảng điều khiển </h1>

    <p> chữ thường </p>

    <button> bấm vào nút này </button>

</body>

</html>

13.3.Lưu ý

- Chỉ 1 <body> duy nhất và không dc đặt trong <head>

- JavaScript luôn nằm trong thẻ <script>,

nhưng <script> có thể nằm trong <head> HOẶC <body>

- Đa số là thẻ <script> thường nằm trong <body>

14.Thẻ tag <b> - chữ đậm ( trang trí)

14.1.Bản chất

- chỉ làm chữ đậm ở mặt hiện thị

14.2.Cú pháp

<b>Nội dung</b>

14.3.Examble

<p>Đây là <b>**chữ đậm**</b> </p>

14.4.Lưu ý

Không dùng để hiện thị nội dung quan trọng

14.5.Kinh nghiệm

Ít dùng - nếu quan trọng dùng <strong>

15.Thẻ tag <strong> - Chữ đâm - Quan trong (KD)

15.1.Bản chất

- Chữ đậm

- có ý nghĩa logic (quan trọng)

15.2.Cú pháp#

<strong>Nội dung</strong>

15.3.Examble

<p>

    <strong>Cảnh báo:</strong> Nhiệt độ cao

</p>

15.4.Kinh nghiệm

Nên ưu tiên <strong> hơn <b>

16.Thẻ tag <br> -xuống dòng

16.1.Bản chất

Xuống dùng

Thẻ đơn, không có đóng

16.2.Cú pháp

<br>

16.3.Examble

Dòng 1<br>

Dòng 2

16.4.Lưu ý

- K đùng dể tạo khoảng cách lớn

1. Thẻ tag <button> - nút bấm

17.1.Bản chất

Tạo nút tương tác

Dùng nhiều nhất trong giao diện điều khiển

17.2.Cú pháp#

<button>Nội dung</button>

17.3.Examble

<button onclick="batLed()">Bật LED</button>

17.4.Lưu ý

Logic xử lý nằm ở javascript

18.Thẻ tag <blockquote> - trích dẫn [ KO HC]

18.1.Bản chất

Tự thụt lề

Dùng cho **đoan trích dẫn**

18.2.Cú pháp#

<blockquote> nội dung </blockquote>

18.3.Examble

<blockquote> Đây là câu trích dẫn </blockquote>

18.4.Lưu ý

Không dùng cho văn bản thường

18.5.Kinh nghiệm

Ít dùng trong giao diện điều khiển

19.Thẻ tag <base> - Đường dẫn gốc ( ít dùng )

19.2.Bản chất

- Đặt đường dẫn gốc cho toàn trang

19.2.Cú pháp#

<base href="https://example.com/">

19.3.Examble

<a href="page.html">Trang</a>

19.4.Lưu ý

Ảnh hưởng toàn bộ link

20.<bdi> & <bdo> — ngôn ngữ đặc biệt [ KO DÙNG ]

20.1.Bản chất

Dùng cho ngôn ngữ RTL (Ả Rập, Do Thái)

21.Thẻ tag <a> -- Liên kết (LINK)

21.1.Bản chất

Dùng để chuyển trang / mở tài nguyên

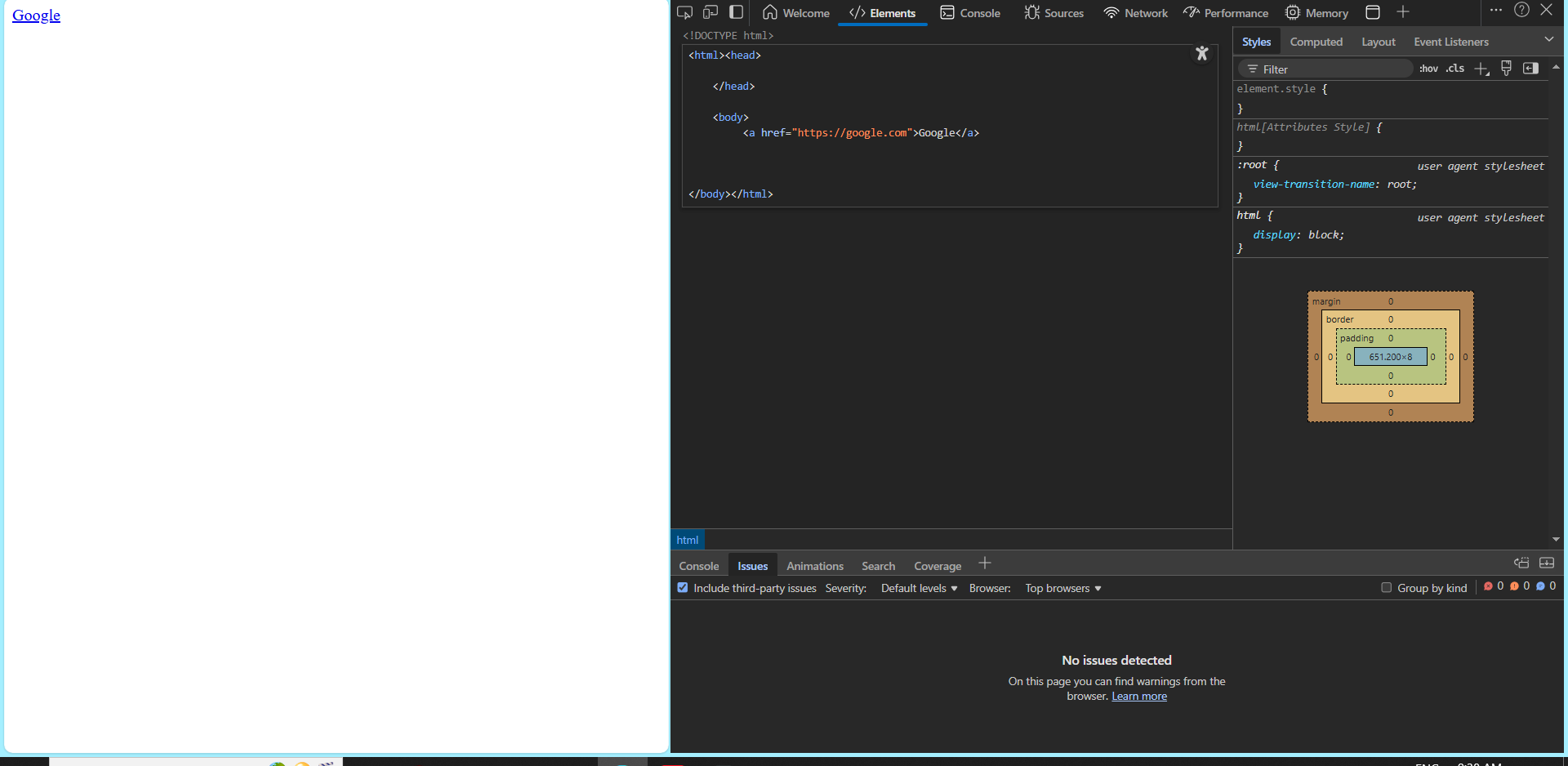
Có thể link wed,file , hoặc chạy js

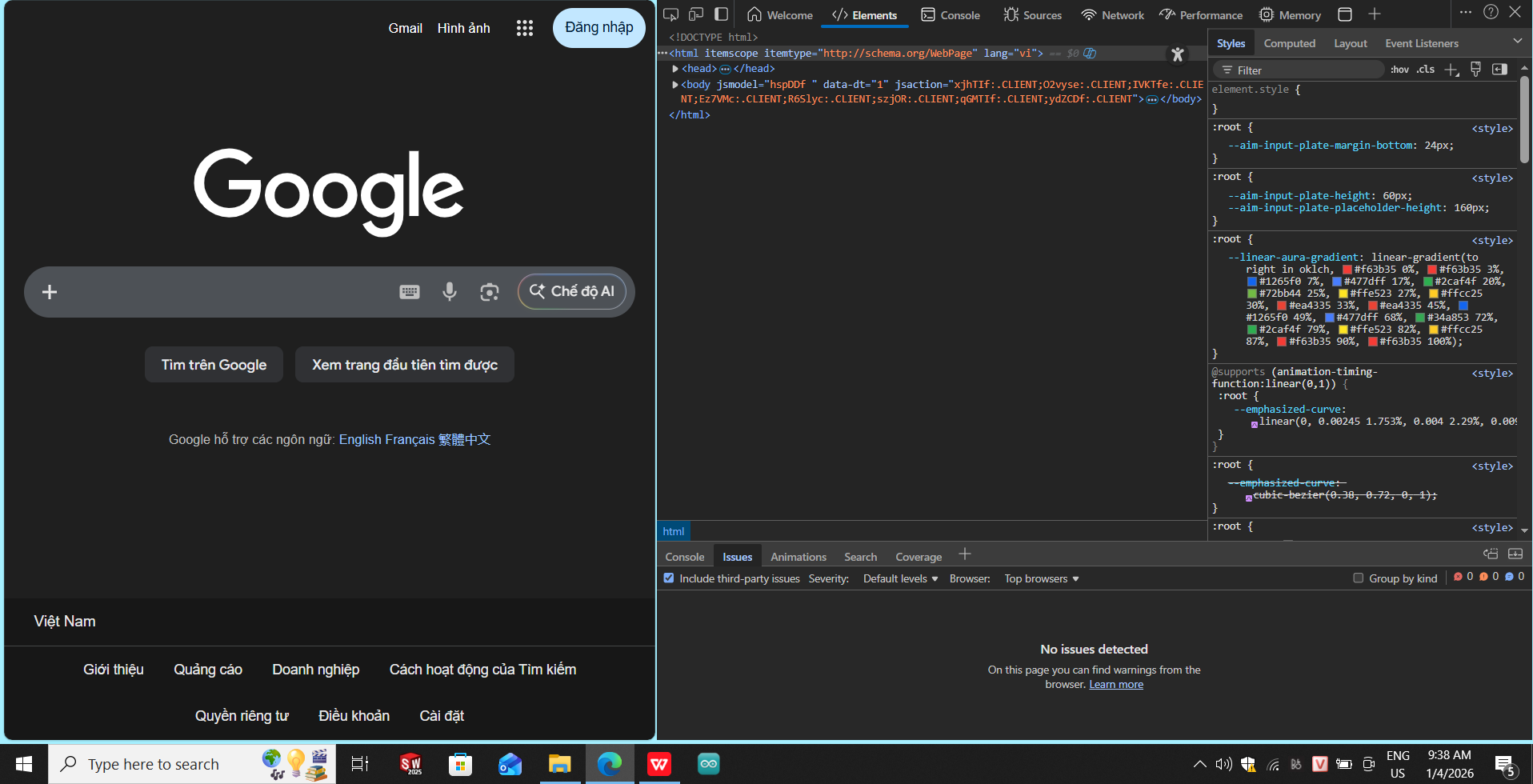
21.2.Cú pháp#

<a href="đường\_dẫn">Nội dung</a>

21.3.Examble

<a href="https://google.com">Google</a>





21.3.Chạy JS

<a href="#" onclick="batLed()">Bật LED</a>

21.4.Lưu ý

href="#" để không chuyển trang

Có thể mở tab mới: target="\_blank"

21.5.Kinh nghiệm

Dùng <button> cho điều khiển

Dùng <a> cho điều hướng

22.Thẻ tag <abbr> -- Viết tắt

22.1.Bản chất

Hiện thị chữ viết tắt

Di chuật vào sẽ thấy giải thích

22.2.Cú pháp #

<abbr title="Giải thích">Chữ</abbr>

22.3.Examble

<abbr title="HyperText Markup Language">HTML</abbr>

22.4.Lưu ý

Tốt cho SEO

KHÔNG ẢNH HƯỞNG GIAO Diện

22.5.Kinh nghiệm

Dùng trong tài liệu mô tả

23.Thẻ tag <address> --- thông tin liên hệ

23.1.Bản chất

Chứ thông tin tác giả liên hệ

23.2.Cú pháp#

<address> nội dung </address>

23.3.Examble

<address> Tác giả: Tuấn

    <br>

    Email: abc@gmail.com

</address>

23.4.Lưu ý

Thường hiện thị chữ nghiêng

23.5.Kinh nghiêm

hay đặt trong <footer>

24.Thẻ tag <area> - vùng click trong ảnh ( nâng cao)

24.1.Bản chất

Tạo vùng click trong ảnh

Đi chung với <map>

24.2.Cú pháp #

<area shape="" coords="" href="">

24.3.Examble

<img src="led.png" usemap="#m">

<map name="m">

    <area shape="rect" coords="0,0,100,100" href="#"></map>

24.4. Kinh nghiệm

Ít dùng nên bỏ qua khi mới học

Phức tạp

25.Thẻ tag <article> --- bài viết độc lập [KO HỌC]

25.1.Bản chất

Một khối nội dung độc lập

Có thể đứng riêng

25.1.Cú pháp #

<article>Nội dung</article>

25.2.Examble

<article>

    <h2>Bài viết 1</h2>

    <p>Nội dung</p>

</article>

25.3.Lưu ý

Dùng cho bài viết , port

26.Thẻ tag <aside> -- nộ dung phụ

26.1.Bản chất

Nội dung bên lề

Không phải nội dung chính

26.2.Cú pháp

<aside>Nội dung</aside>

26.3.Examble

<aside>

ghi chú

</aside>

26.4.Lưu ý

Thường nằm cạnh <main>

***Dùng cho:***

Thanh bên (sidebar)  
Menu chế độ  
Trạng thái WiFi  
Nhiệt độ, điện áp  
 Log / thông báo  
 Nút phụ (reset, config)

27.Thẻ tag <audio> --- phát âm thanh

27.1.Bản chất

Phát file âm thanh

<audio src="file.mp3" controls></audio>

27.1.Examble

<audio controls>

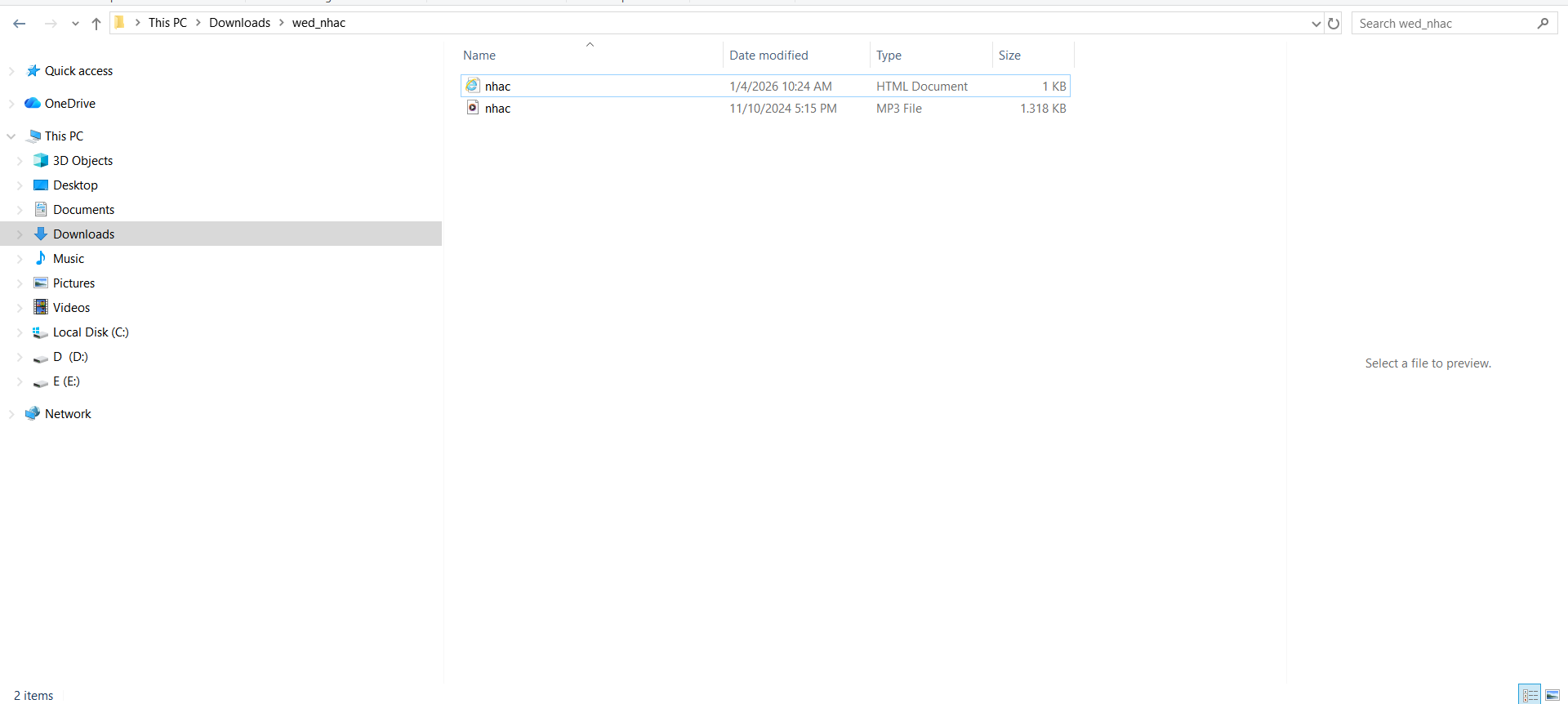
    <source src="beep.mp3" type="audio/mpeg">

</audio>

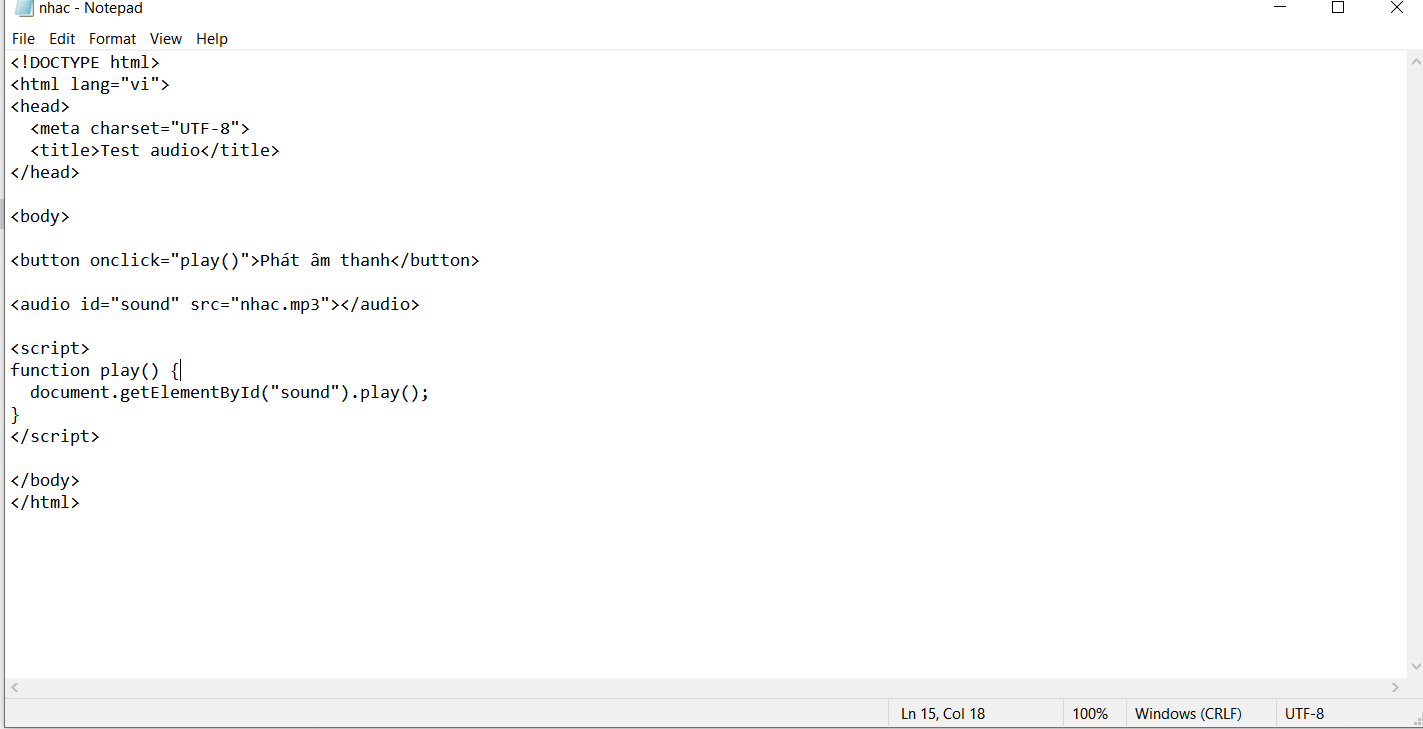
27.1.1.Setup audio ở trình duyệt-- file mb3 trong Expolrer

Tạo file thư mục ở expolrer là …

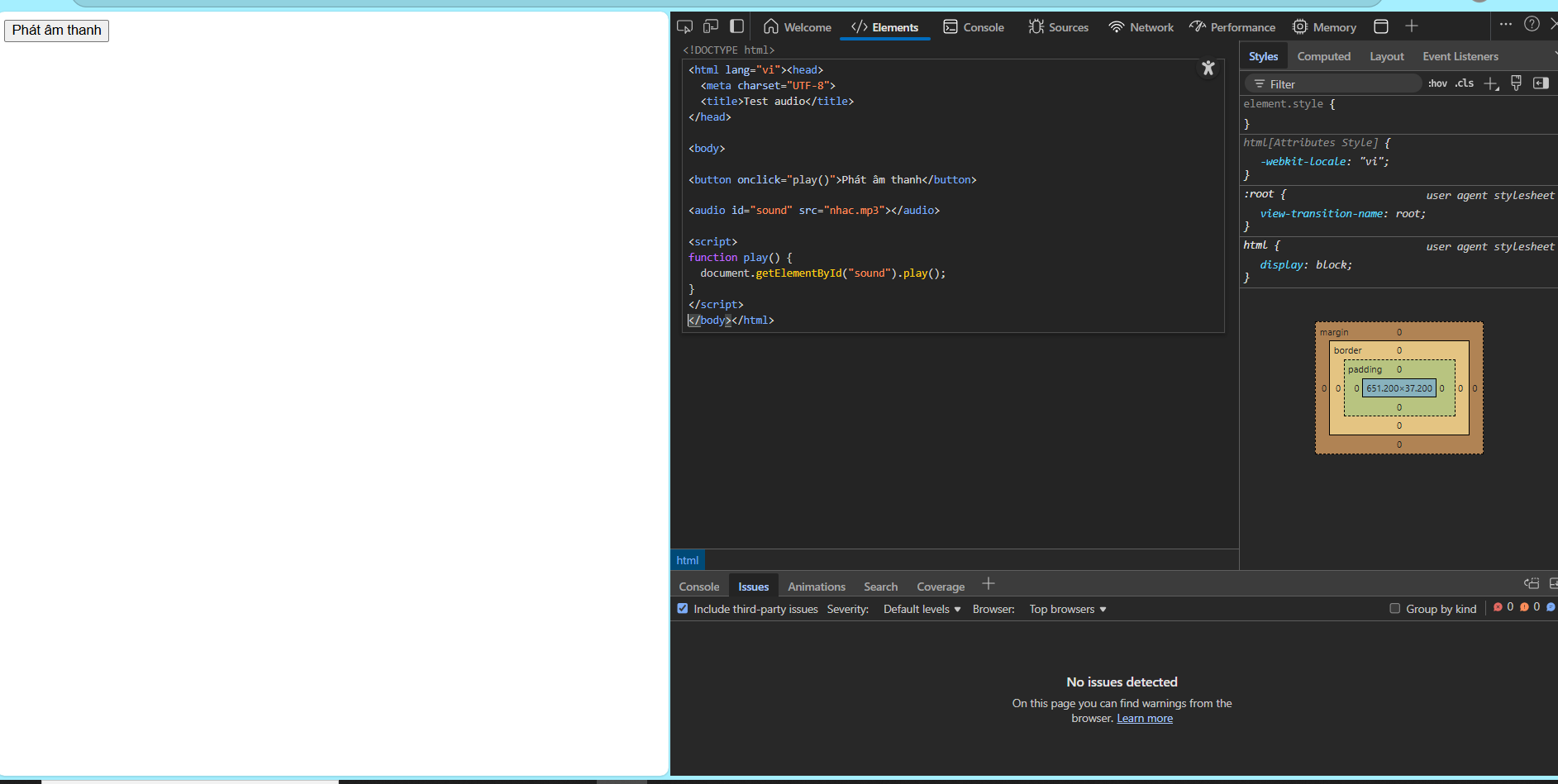
Đưa file mb3 nào đó vô trong thư mục vừa tạo



Viết code và setup file code rename đúng tên .mb3 + lưu với dạng .html



Sau khi mở file html lên trình duyệt



CODE#

<!DOCTYPE html>

<html lang="vi">

<head>

    <meta charset="UTF-8">

    <title>Test audio</title>

</head>

<body>

    <button onclick="play()">Phát âm thanh</button>

    <audio id="sound" src="nhac.mp3"></audio>

    <script>

        function play() {

            document.getElementById("sound").play();

        }

    </script>

</body>

</html>

27.1.2.Lưu ý

Không dùng autoplay

<audio autoplay></audio> <!-- dễ bị chặn -->

Phải phát bằng click

<button onclick="play()">...</button>

27.2.Lưu ý

- Trình duyện chặn audio nếu dùng file trong expolrer thì phát audio được

28.Thẻ tag <canvas> - Đồ họa

28.1.Bản chất

Vùng vẽ đồ họa bằng JavaScript (2D,game, biểu đồ)

28.2.Cú pháp#

<canvas id="cv" width="200" height="100"></canvas>

28.3.Examble

<canvas id="cv" width="200" height="100"></canvas>

<script>

    let c = document.getElementById("cv");

    let ctx = c.getContext("2d");

    ctx.fillRect(20, 20, 50, 50);

</script>

28.4.Lưu ý

Canvas không tự vẽ phải dùng js

29.Thẻ tag <caption> -- tiêu đề

29.1.Bản chất

Tiêu đề cho bảng <table>

29.2.Cú pháp #

<table>

    <caption>Danh sách LED</caption>

</table>

29.3.Examble

<table border="1">

    <caption>Trạng thái LED</caption>

    <tr>

        <th>LED</th>

        <th>State</th>

    </tr>

    <tr>

        <td>LED1</td>

        <td>ON</td>

    </tr>

</table>

29.4.Kinh nghiệm

Nên dùng cho bảng trạng thái

30.Thẻ tag <code>

30.1.Bản chất

Hiện thị mã code

30.2.Cú pháp#

<code>digitalWrite(2, HIGH);</code>

30.3.Examble

<p>

    Lệnh bật LED: <code>HIGH</code>

</p>

30.4.Lưu ý

Chỉ để hiện thị

Không chạy code

30.5.Kinh nghiệm

Hợp cho wed hưỡng dẫn wed

31.Thẻ tag <col>

31.1.Bản chất

Định dạng cột trong bảng

31.2.Cú pháp

<col style="background:red">

31.3.Examble

<table border="1">

    <colgroup>

        <col style="background:#eee">

        <col style="background:#cfc">

    </colgroup>

    <tr>

        <td>LED</td>

        <td>ON</td>

    </tr>

</table>

31.4.Lưu ý

Chỉ dùng trong bảng

32.Thẻ tag <colgroup>

32.1.Bản chất

Nhóm nhiều <col>

32.2.Cú pháp

<colgroup>...</colgroup>

32.3.Kinh nghiệm

Ít dùng , bản đơn giản không cần

33.Thẻ tag <data>

33.1.Bản chất

Gắn giá trị máy đọc được cho nội dung hiện thị

33.2.Cú pháp#

<data value="1">LED ON</data>

33.3.Examble

<p>Trạng thái: <data value="1">BẬT</data>

</p>

33.4.Lưuý

Người dùng thấy chữ

Ít dùng trong giao diện wed esp

34.Thẻ tag <datalist> --- Gợi ý

34.1.Bản chất

Danh sách gợi ý cho <input>

34.2.Cú pháp#

<datalist id="list"></datalist>

34.3.Examble

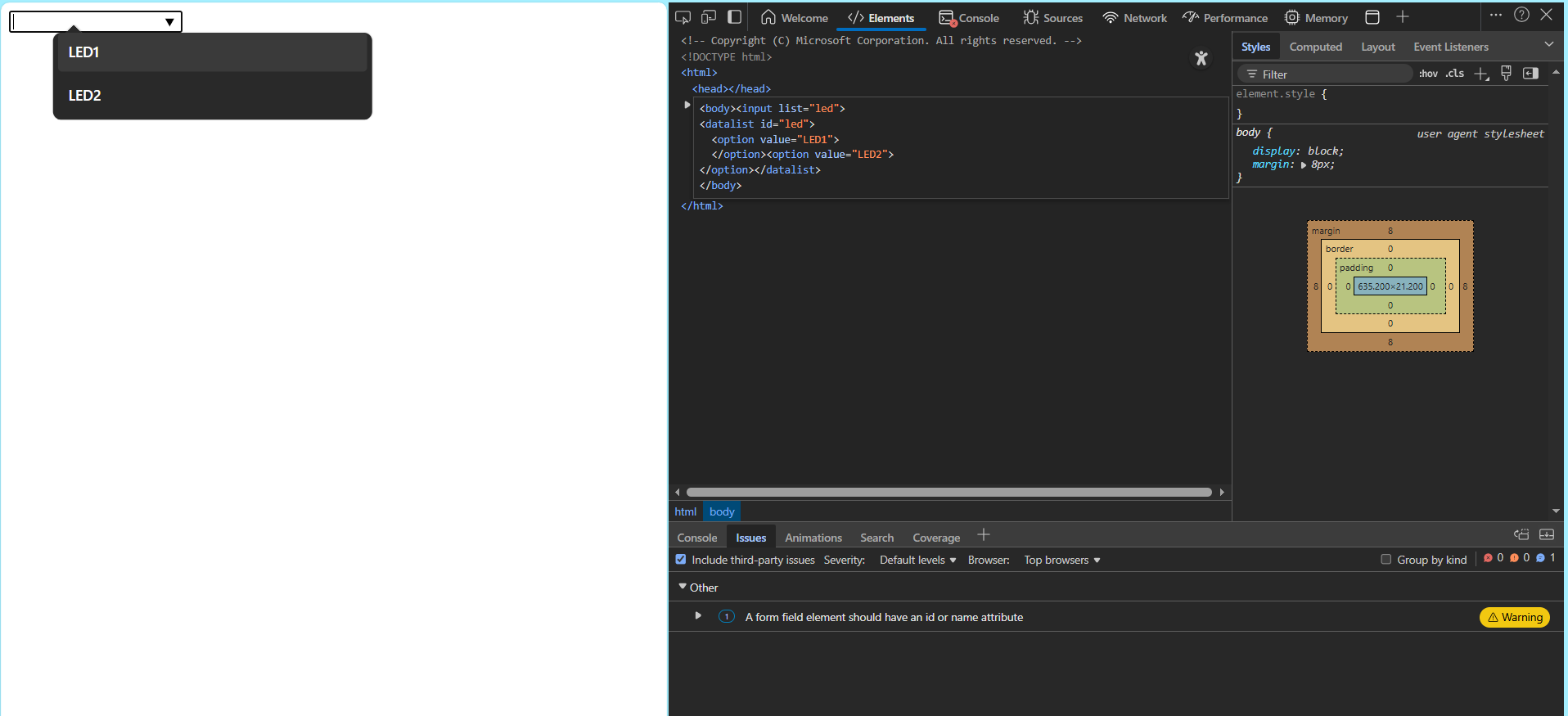
<input list="led">

<datalist id="led">

    <option value="LED1">

    <option value="LED2">

</datalist>



34.4.Lưu ý

Chỉ là gợi ý

Không phải dropdown bắt buộc

34.5.Kinh nghiệm

Rất hợp cho nhập tên thiết bị

35.Thẻ tag <dd> -- Mô tả

35.1.Bản chất

Nội dung mô tả trong danh sách

35.2.Cú pháp#

<dd> Mô tả </dd>

36.Thẻ tag <del> --- ~~gạch ngang~~

36.1.Bản chất

Nội dung bị xóa

36.2.Cú pháp#

<del>OFF</del>

36.3.Examble

<p>LED: <del>OFF</del> ON</p>

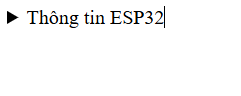
36.4.Lưu ý

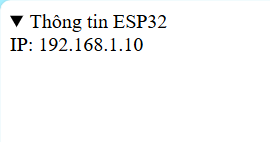
Hiện thị chữ ~~gạch ngang~~

37.Thẻ tag <details>

37.1.Bản chất

Khối mở / đóng





37.2.Cú pháp#

<details></details>

37.3.Examble

<details>

    <summary>Thông tin ESP32</summary> IP: 192.168.1.10

</details>

37.4.Lưu ý

Không cần js

37.5.Kinh nghiệm

Rất phù hợp cho giao diện ESP WED

38.Thẻ tag <dfn>

38.1.Bản chất

Định nghĩa thuật ngữ

38.2.Cú pháp

<dfn>ESP </dfn>

38.3.Examble

<p>

    <dfn>ESP32</dfn> là vi điều khiển WiFi

</p>

39.Thẻ tag <div> [ RQT ]

39.1.Bản chất

Khối chia giao diện

Mỗi <div> chiếm 1 dòng

39.2.Cú pháp #

<div></div>

39.3.Examble

<div class="box">

    <button>Bật LED</button>

</div>

39.4.Lưu ý

Không có ý nghĩa

Chỉ để layout

40.Thẻ tag <dl> [ QT]

40.1.bản chất

Danh sách định nghĩa

41.Thẻ tag <dt> [QT]

41.1.Bản chất

Tên mục trong <dl>

41.2.Examble

<dl>

  <dt>LED</dt>

  <dd>Thiết bị phát sáng</dd>

</dl>

42.Thẻ tag <em>

42.1.Bản chất

Emphasis-- nhấn mạnh ý nghĩa

Trình duyệt thường hiện thị *chữ nghiêng*

42.2.Cú pháp #

<em> Nội dung </em>

42.3.Examble

<p>LED đang <em>bật</em></p>

42.4.Lưu ý

Khác <i> :

<em> -- nhấn mạnh ngữ nghĩa

<i> --- chỉ nghiêng chữ cho đẹp

42.5.Kinh nghiệm

Dùng cảnh báo/trạng thái quan trọng

43.Thẻ tag <embed> - -nhúng file bên ngoài

43.1.Bản chất

- Nhúng file bên ngoài trực tiếp vào trang

- Dùng cho: PDF, video, audio, flash (cũ)

43.2.Cú pháp#

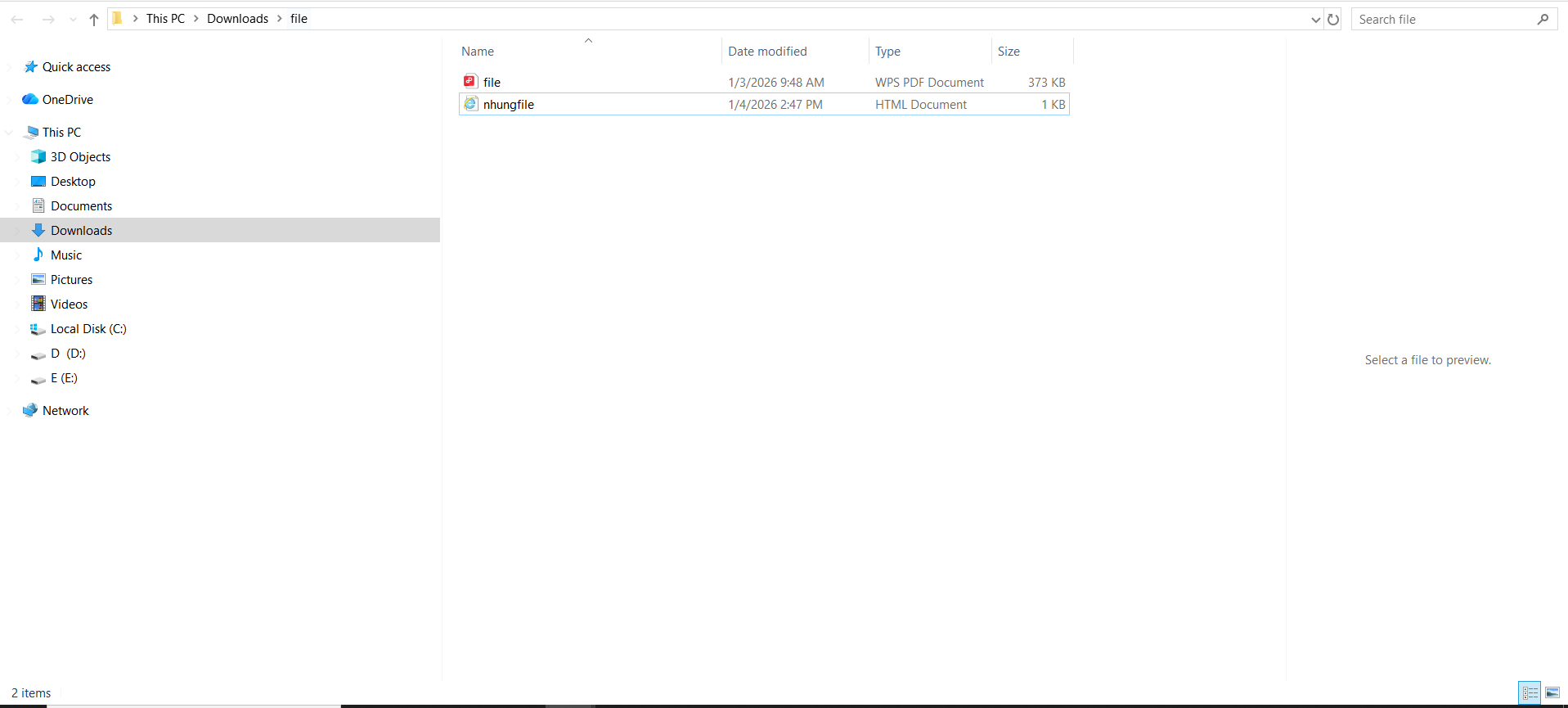
<embed src="file.pdf">

43.3.Examble

<embed src="file.pdf" width="400" height="300">

43.4.Cách nhúng file ở ngoài

tạo thư mục folder cho file pdf vào folder vừa tạo + viết code setup với đường dẫn có tên file pdf



CODE#

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>test nhúng file </title>

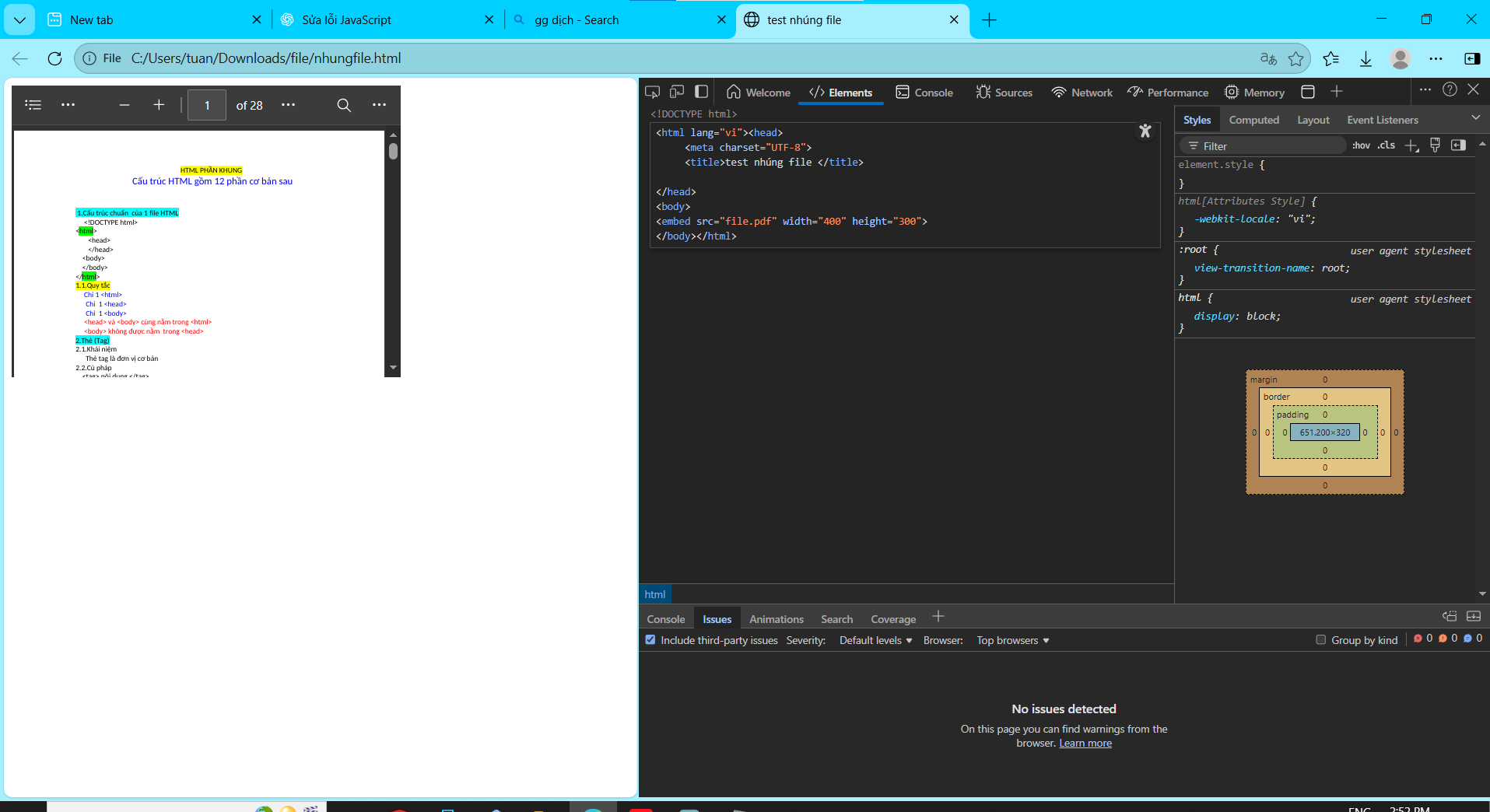
</head>

<body>

  <embed src="file.pdf" width="400" height="300">

</body>

</html>



43.5.Lưu ý

Không có thẻ đóng

Tự trình duyệt xử lý

43.6.Kinh nghiệm

Thường dùng nhúng pdf

Audio/video → nên dùng <audio>, <video> thay thế

44.Thẻ tang <form> -- Biểu mẫu (RQT)

44.1.Bản chất

Dùng để thu thập dữ liệu người dùng

Gửi dữ liệu server /ESP / JS

44.2.Cú pháp#

<form> ... </form>

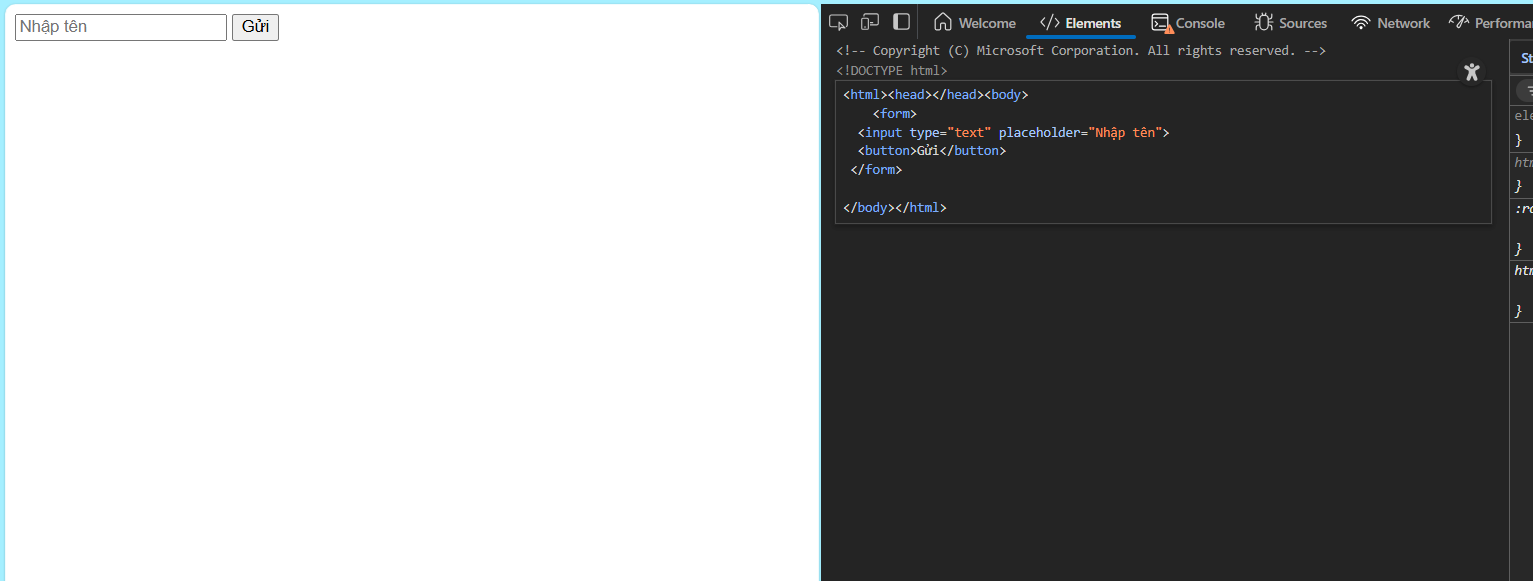
44.3.Examble

<form>

  <input type="text" placeholder="Nhập tên">

  <button>Gửi</button>

</form>



44.4.Lưu ý

Input nên nằm trong <form>

Khi nhấn button -- form có thể gửi dữ liệu

44.5.Kinh nghiệm

Giao diện điều khiển ESP32 luôn dùng form + input

Thường kết hợp JS để xử lý, HTML không làm logic

45.Thẻ tag <fieldset> -- nhóm các input

45.1.Bản chất

Gom gác input liên quan lại thành 1 khối

Có viền bao quanh mặc định

45.2.Cú pháp

<fieldset>

...

</fieldset>

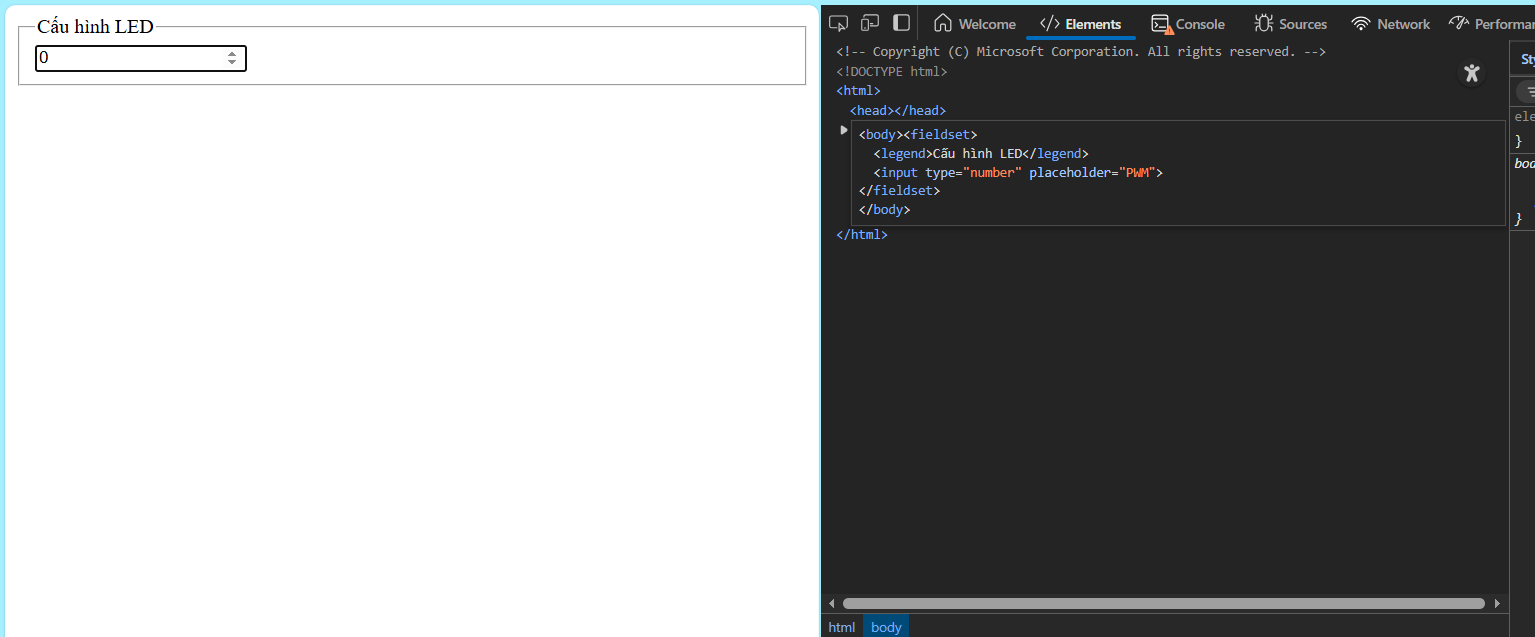
45.3.Examble

<fieldset>

  <legend>Cấu hình LED</legend>

  <input type="number" placeholder="PWM">

</fieldset>



45.4.Lưu ý

Hay dùng với <legend>

Chỉ dùng trong form

45.5.Kinh nghiệm

Dùng để nhìn giao diện rõ ràng

Rất phù hợp với wed điều khiển esp

46.Thẻ tang <legend> --- tiêu đề của fieldset [ BX]

46.1.Cú pháp #

<legend>Cấu hình</legend>

47.Thẻ tag <firuge> -- Khối hình ảnh / nội dung minh họa [RQT]

47.1.Bản chất

Chứ ảnh, sơ đồ, biểu đồ

47.2.Cú pháp

<figure> ... </figure>

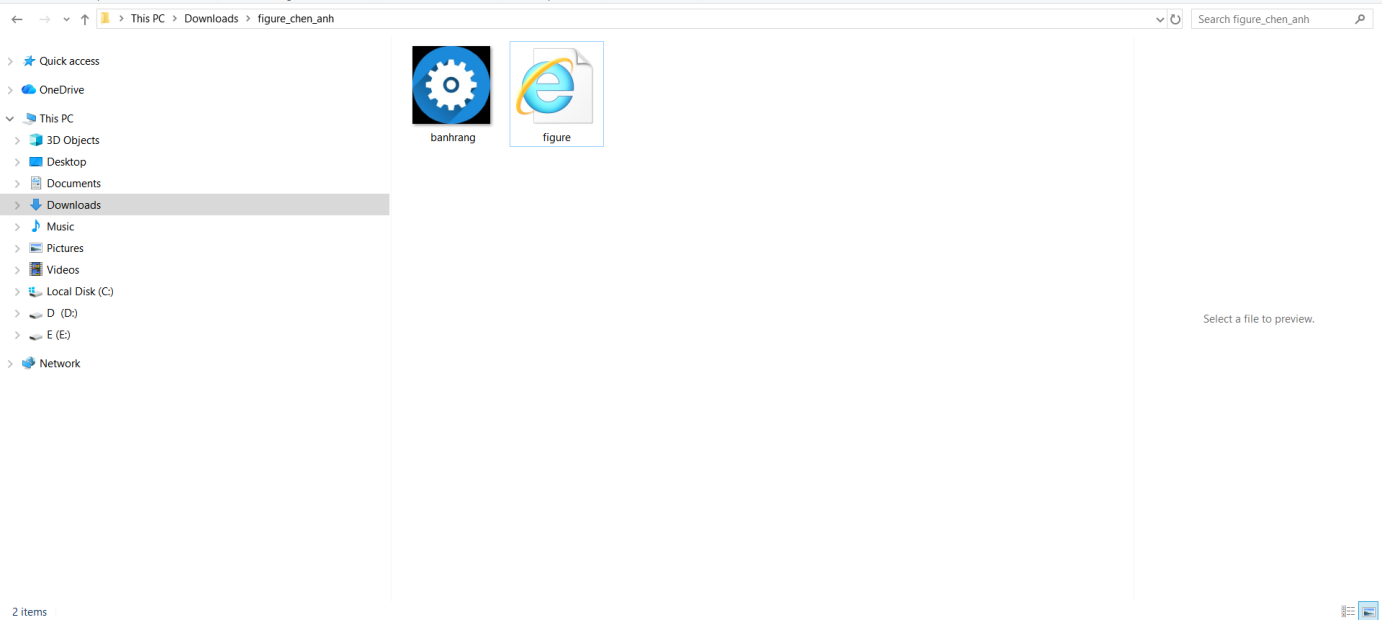
47.3.Examble

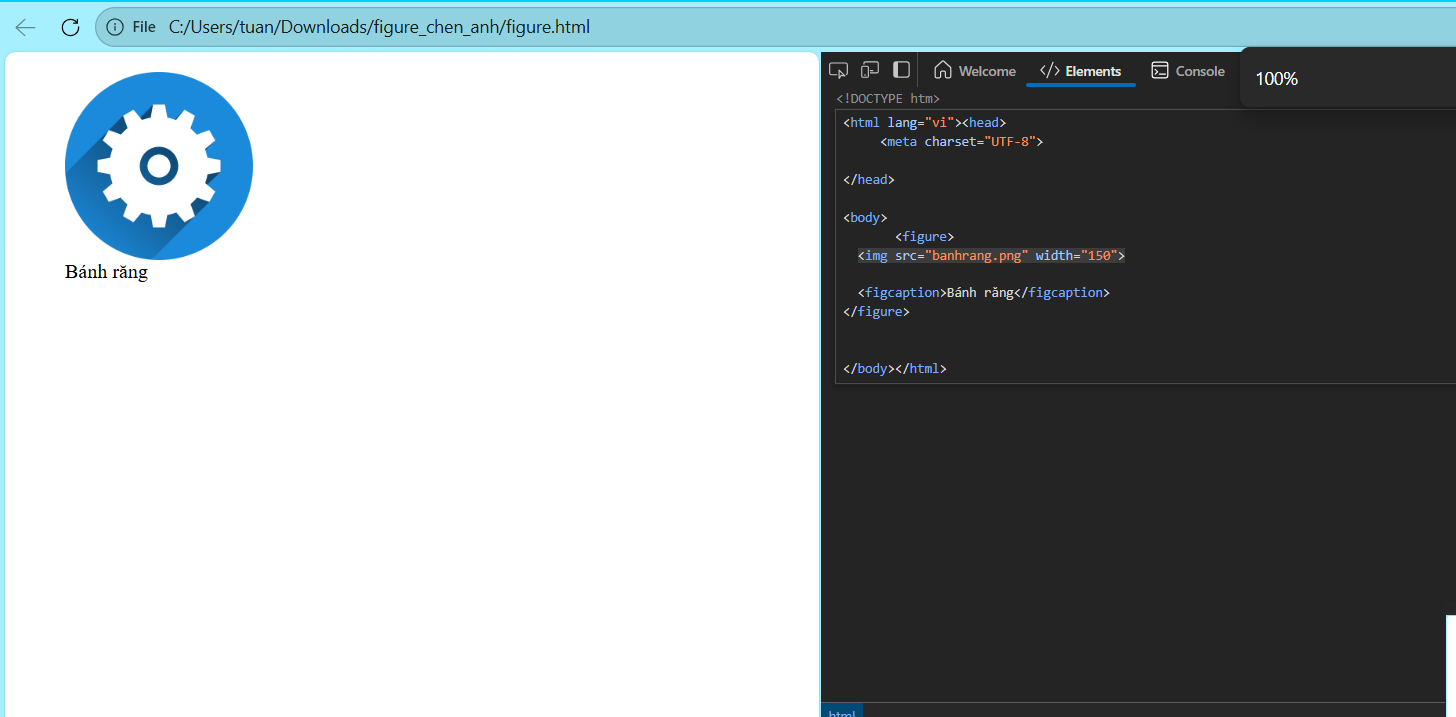
<figure>

  <img src="banhrang.png" width="150">

  <figcaption>Đèn LED</figcaption>

</figure>





47.4.Những nguồn tải down ảnh , icon chất lượng cao

https://unsplash.com

Examble:

<img src="https://images.unsplash.com/photo-xxxx" alt="Ảnh">

Pexels – Ảnh & video miễn phí

https://www.pexels.com

có ảnh portrait, landscape, tech

Pixabay – Ảnh miễn phí & icon

https://pixabay.com

Ảnh + vector + illustration

Icons8 – icon & ảnh nhỏ

https://icons8.com

Tải icon PNG/SVG

Có ảnh dùng cho GUI

Examble:

<img src="https://img.icons8.com/color/48/000000/led-on.png">

Font Awesome – icon dùng thẻ

https://fontawesome.com

Dùng CSS để chèn icon

Không cần file ảnh

Examble:

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">

<i class="fa-solid fa-lightbulb"></i>

Flaticon – icon lớn nhất

https://www.flaticon.com

Icon miễn phí

Có PNG / SVG / Font

Examble :

<img src="https://images.pexels.com/photos/xxxx.jpg" alt="Ảnh LED">

48.Thẻ tag <footer> -- chân trang / cuối trang

48.1.Bản chất

Chứa bản quyền , thông tin phụ

48.2.Cú pháp #

<footer>…</footer>

48.3.Examble

<footer>

© 2026 ESP Control

</footer>

49.Thẻ tag <h1> <h6>

h1 : tiêu đề lớn nhất

h2: tiêu đề cấp 2

h3 :tiêu cấp 3

h4 , h5,h6

50.Thẻ tag <header> --- đầu trang

50.1.Bản chất

- phần đầu nội dung hiện thị

- Thường chứa menu, tiêu đề, logo

50.2.Cú pháp

<header> nội dung </header>

50.3.Examble

<header>

  <h1>Bảng điều khiển</h1>

</header>

51.Thẻ tag <hgroup> -- nhóm tiêu đề [ít D]

51.Examble

<hgroup>

  <h1>ESP32</h1>

  <h2>Điều khiển từ xa</h2>

</hgroup>

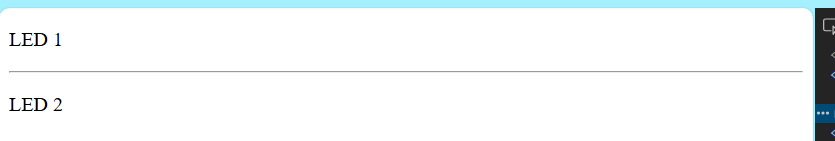
52.Thẻ tag <hr> -- GẠCH NGANG

52.1.Examble

<p>LED 1</p>

<hr>

<p>LED 2</p>



53.Thẻ tag <i> --- Chữ nghiêng

53.1.Examble

<i>Chú thích</i>

53.2.Lưu ý

Nếu nhấn mạnh ý nghĩa nên dùng <em>

54.Thẻ tag <iframe> -- nhúng trang khác

54.1.Bản chất

Nhúng wedsite / video /giao diện khác vào trang

54.2.Examble

<iframe src="https://www.youtube.com/embed/VIDEO\_ID" width="300" height="200">

</iframe>

54.3.Lưu ý

Dùngkhi:

Nhúng camera ESP

Nhúng dashboard khác

Nhúng Google Map

55.Thẻ tag <img> -- Hiện Thị ẢNH

55.1.Bản chất

Là thẻ đơn

Hiện thị ảnh

55.2.Cú pháp#

<img src="duongdan.png" alt="mô tả">

55.3.Examble

<img src="led.png" alt="Đèn LED">

Trong đó:

src : đường dẫn ảnh

alt :Mô tả khi ảnh lỗi

**Chỉnh kích thước**

<img src="led.png" width="100">

56.Thẻ tag <input> --nhập dữ liệu

56.1.Bản chất

Là thẻ đơn

Nhập dữ liệu từ người dùng

56.2.Cú pháp#

<input type="loại" name="ten" value="giá trị">

56.3.Các loại hay dùng

<input type="text">

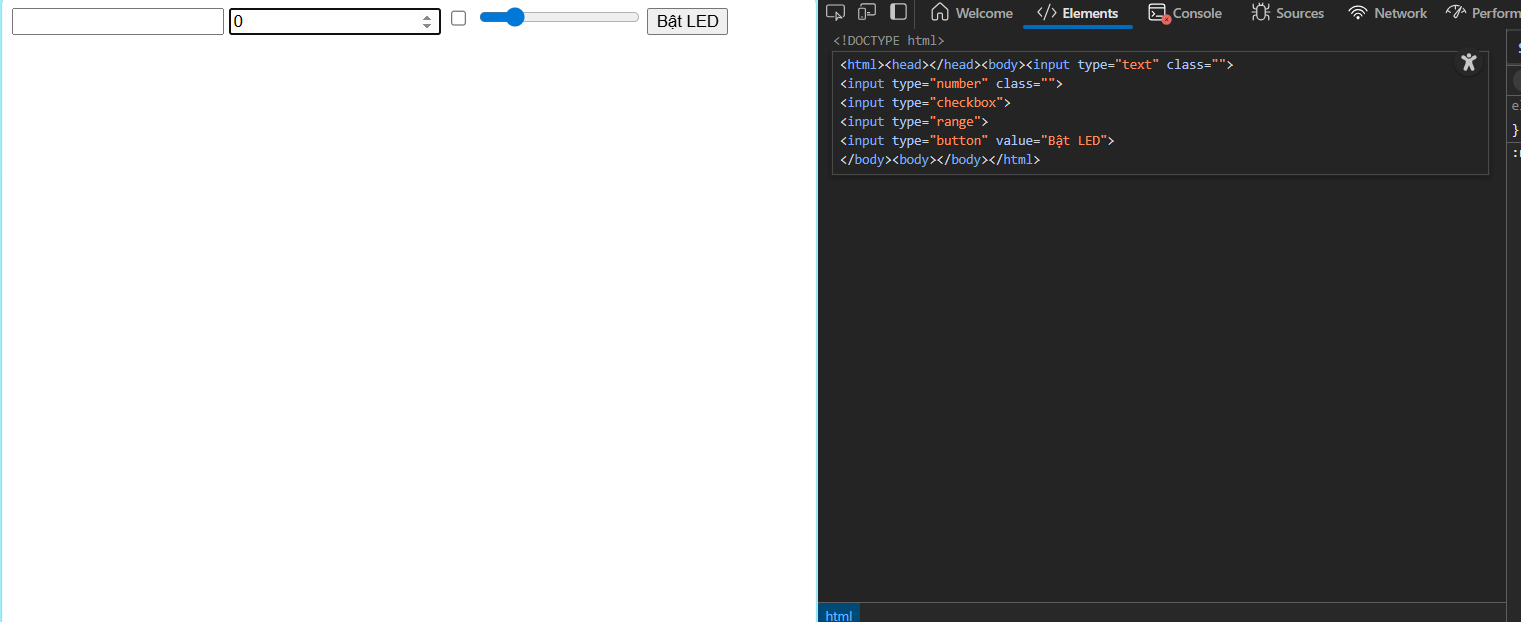
<input type="number">

<input type="checkbox">

<input type="range">

// dùng để tạo thanh trượt

<input type="button" value="Bật LED">



56.4.Code núm xoay

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Núm xoay không reset</title>

  <style>

    body {

      background: #111;

      color: #0ff;

      text-align: center;

      font-family: Arial;

      padding-top: 50px;

    }

    .knob {

      width: 150px;

      height: 150px;

      border-radius: 50%;

      background: radial-gradient(circle, #0ff, #044);

      margin: auto;

      position: relative;

      touch-action: none;

    }

    .knob::after {

      content: "";

      width: 6px;

      height: 60px;

      background: white;

      position: absolute;

      top: 10px;

      left: 50%;

      transform: translateX(-50%);

      border-radius: 4px;

    }

  </style>

</head>

<body>

  <h3>Núm xoay KHÔNG tụt min</h3>

  <div id="knob" class="knob"></div>

  <p>Giá trị: <span id="val">0</span>

  </p>

  <script>

    const knob = document.getElementById("knob");

    const val = document.getElementById("val");

    let angle = -135;     // góc hiện tại

    let lastAngle = angle;

    let dragging = false;

    function update(e) {

      if (!dragging) return;

      const rect = knob.getBoundingClientRect();

      const cx = rect.left + rect.width / 2;

      const cy = rect.top + rect.height / 2;

      const p = e.touches ? e.touches[0] : e;

      const x = p.clientX - cx;

      const y = p.clientY - cy;

      let deg = Math.atan2(y, x) \* 180 / Math.PI + 90;

      // ❗ CHẶN WRAP

      if (Math.abs(deg - lastAngle) > 180) return;

      // GIỚI HẠN

      deg = Math.max(-135, Math.min(135, deg));

      angle = deg;

      lastAngle = deg;

      const value = Math.round((angle + 135) / 270 \* 255);

      knob.style.transform = `rotate(${angle}deg)`;

      val.innerText = value;

    }

    knob.addEventListener("mousedown", () => dragging = true);

    document.addEventListener("mousemove", update);

    document.addEventListener("mouseup", () => dragging = false);

    knob.addEventListener("touchstart", () => dragging = true);

    document.addEventListener("touchmove", update, { passive: false });

    document.addEventListener("touchend", () => dragging = false);

  </script>

</body>

</html>

57.Thẻ tag <ins> -- CHỮ GẠCH CHÂN

57.1.Bản chất

Đánh dấu nội dung được thêm

57.2.Cú pháp

<ins>Nội dung</ins>

57.3.Examble

<ins>Đã cập nhật</ins>

58.Thẻ tag <kbd> --Hiện thị bàn phím

58.1.Bản chất

Dùng để hiện thi bàn phím trên màn hình khi nhập liệu

58.2.Cú pháp

<kbd>phím</kbd>

58.3.Examble

<!DOCTYPE html>

<html lang="vi">

<head>

 <meta charset="UTF-8">

  <title>Test input</title>

</head>

<body>

  <h3>Nhập dữ liệu</h3>

  <input id="txt" type="text" placeholder="Nhập gì đó...">

  <button onclick="gui()">Gửi</button>

  <p id="kq"></p>

  <script>

    function gui() {

      let giatri = document.getElementById("txt").value;

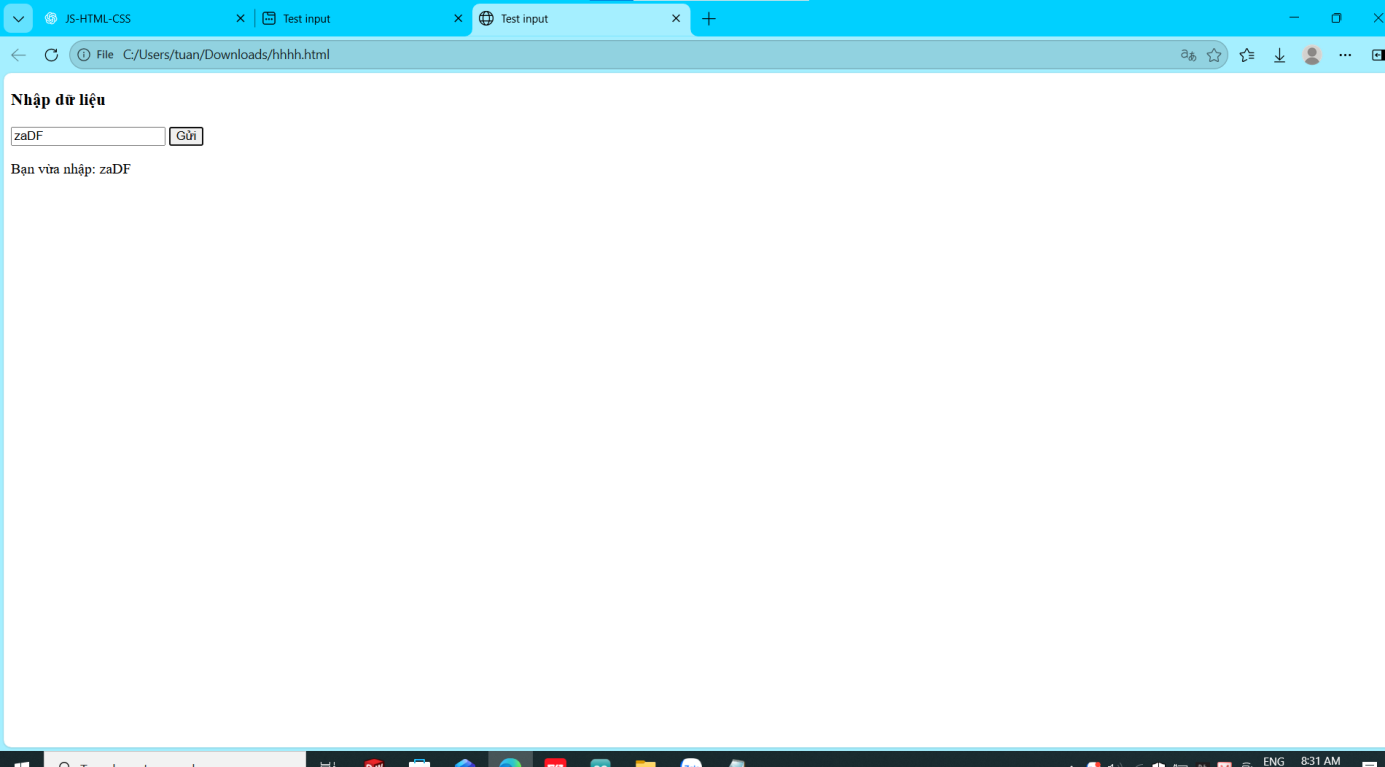
      document.getElementById("kq").innerText = "Bạn vừa nhập: " + giatri;

    }

  </script>

</body>

</html>



58.4.Examble two

<!DOCTYPE html>

<html>

<head>

  <meta charset="UTF-8">

  <title>ESP UI test</title>

</head>

<body>

  <h3>Điều khiển LED</h3>

  <button onclick="bat()">BẬT LED</button>

  <button onclick="tat()">TẮT LED</button>

  <p id="status">Trạng thái: ---</p>

  <script>

    function bat() {

      document.getElementById("status").innerText = "Trạng thái: LED BẬT";

    }

    function tat() {

      document.getElementById("status").innerText = "Trạng thái: LED TẮT";

    }

  </script>

</body>

</html>



59.Cách cài key

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Tạo key nguyên bản</title>

</head>

<body>

  <h3>Nhập mật khẩu</h3>

  <input type="password" id="pass">

  <button onclick="xuLy()">Tạo KEY</button>

  <script>

    // 1. Tạo key ngẫu nhiên

    function taoKey() {

      return Math.random().toString(36).substring(2, 18);

   }

    // 2. Băm (hash) đơn giản

    async function bam(text) {

      const data = new TextEncoder().encode(text);

      const hash = await crypto.subtle.digest("SHA-256", data);

      return [...new Uint8Array(hash)]

        .map(b => b.toString(16).padStart(2, "0"))

        .join("");

    }

    // 3. Xử lý toàn bộ

    async function xuLy() {

      let password = document.getElementById("pass").value;

      let key = taoKey();

      let ketHop = password + key;

      let hash = await bam(ketHop);

      console.log("KEY:", key);

      console.log("HASH:", hash);

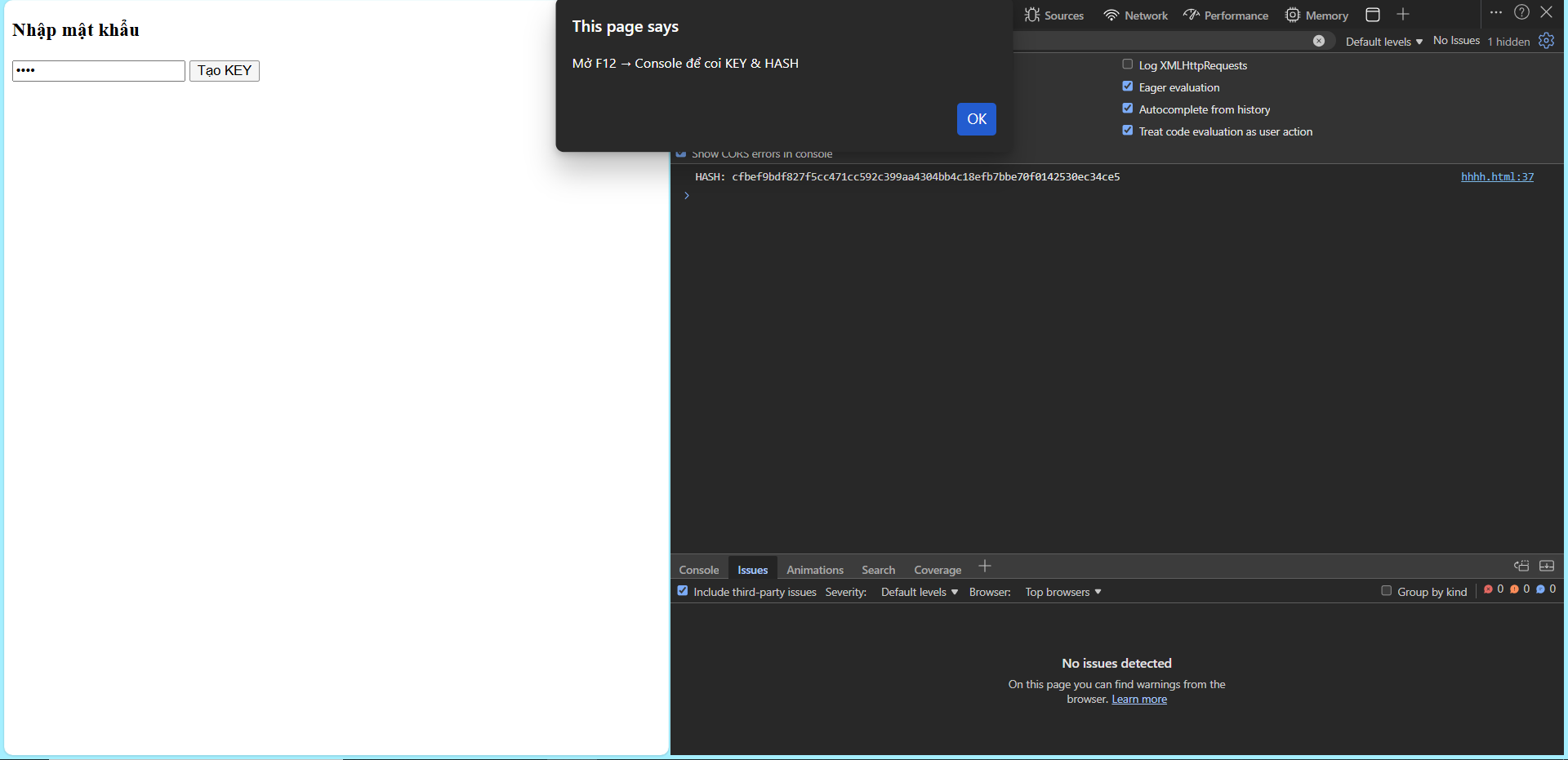
      alert("Mở F12 → Console để coi KEY & HASH");

    }

  </script>

</body>

</html>



60.Thẻ tag <label> -- Gán nhãn cho input [RQT]

60.1.Bản chất

Gắn chữ mô ta cho input

Click vào chữ → focus vào ô nhập

Tốt cho form ,Cấu hình , ESP Control

60.2.Cú pháp

<label for="user">Tên:</label>

<input id="user" type="text">

60.3.Examble <label> + <input>

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test label</title>

</head>

<body>

  <h2>Điều khiển LED</h2>

  <!-- Label + Checkbox -->

  <label for="led1">Bật LED 1</label>

  <input type="checkbox" id="led1">

  <br>

  <br>

  <!-- Label + Text -->

  <label for="ten">Tên thiết bị:</label>

  <input type="text" id="ten" placeholder="Nhập tên ESP">

  <br>

  <br>

  <!-- Label + Button -->

  <label for="btn">Gửi lệnh</label>

  <button id="btn" onclick="gui()">OK</button>

  <script>

    function gui() {

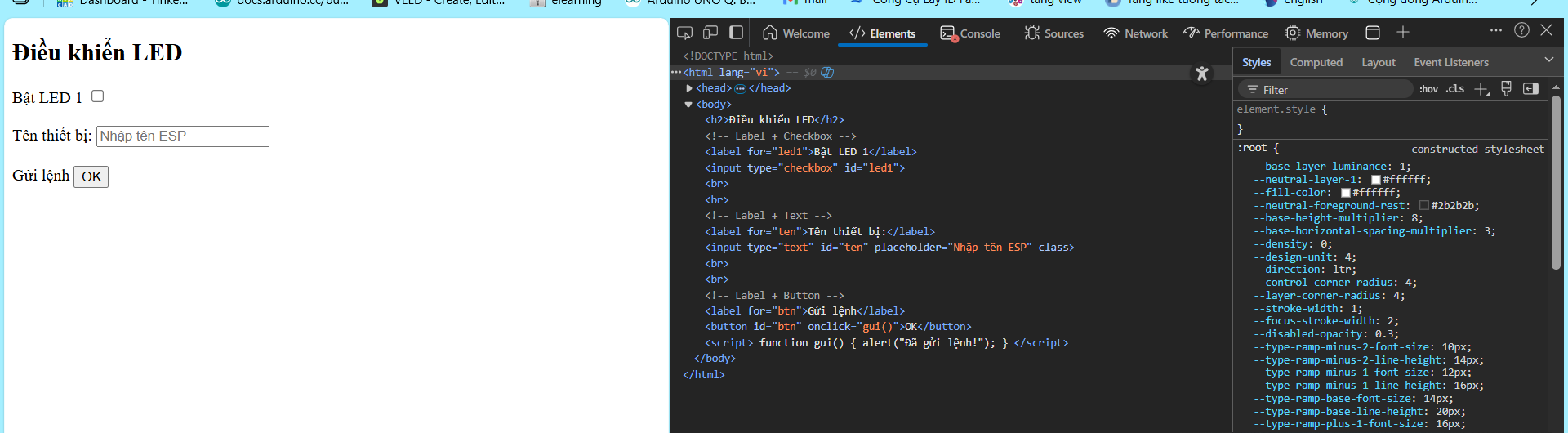
      alert("Đã gửi lệnh!");

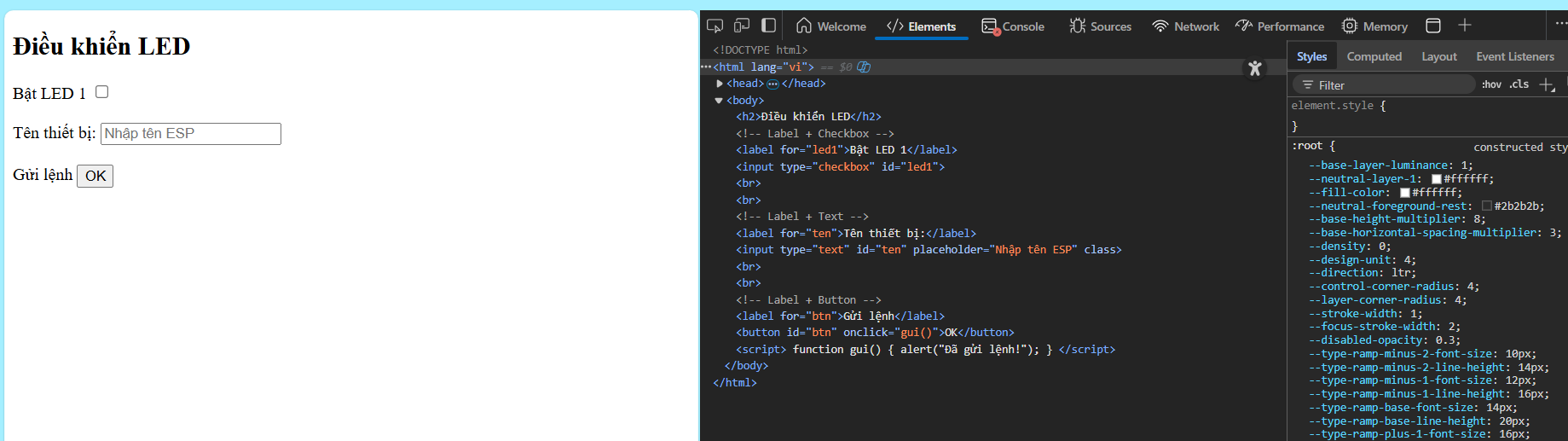
    }

  </script>

</body>

</html>





60.4.Lưu ý

for phải trùng id của input

Không dùng cho trang trí

61.Thẻ tag <legend> – Tiêu đề cho nhóm input

61.1.Bản chất

Chỉ dùng bên trong <fieldset>

Đặt tên cho một nhóm cấu hình

61.2.Cú pháp #

<fieldset>

  <legend>Cấu hình LED</legend>

</fieldset>

61.3.Examble

<fieldset>

  <legend>ESP32</legend>

  <label>LED</label>

  <input type="checkbox">

</fieldset>

61.4.Lưu ý

Không dùng 1 mình

Dùng cho form cấu hình

62.Thẻ tag <li> - Item trong danh sách [menu]

62.1.Bản chất

Là phần tử con của <ul> hoặc <ol>

Dùng làm menu, danh sách

62.2.Cú pháp #

<ul>

  <li>LED</li>

  <li>Motor</li>

</ul>

62.3.Examble (menu điều khiển )

<ul>

  <li>Bật LED</li>

  <li>Tắt LED</li>

</ul>

62.4.Lưu ý

<li> không đứng 1 mình

63.Thẻ tag <link> - Gắn file ngoài (CSS)

63.1.Bản chất

Dùng để kết nối với tài nguyên

63.2.Cú pháp #

<link rel="stylesheet" href="style.css">

63.3.Examble

<head>

  <link rel="stylesheet" href="style.css">

</head>

63.4.Lưu ý

Thẻ < link > chỉ nằm trong <head>

Và <link> không có thẻ đóng

64.Thẻ tag <map> - Bản đồ vùng click trên ảnh

64.1.Bản chất

Dùng để chia vùng trên ảnh để click

64.2.Cú pháp

<img src="board.png" usemap="#boardmap">

<map name="boardmap">

  <area shape="rect" coords="10,10,100,60" href="#">

</map>

65.Thẻ tag <mark> - TÔ SÁNG CHỮ

65.1.Bản chất

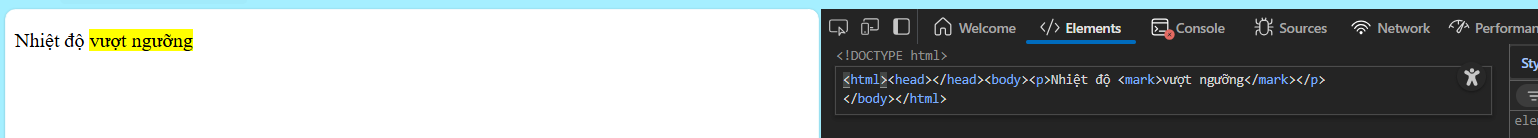
Dùng để highlight chữ quan trọng

65.2.Cú pháp

<mark>chữ cần tô</mark>

65.3.Examble

<p>Nhiệt độ <mark>vượt ngưỡng</mark></p>



66.Thẻ tag <meta> -Thông tin ẩn

66.1.Bản chất

Không hiển thị  
Dùng cho:

mã hóa

SEO

responsive

66.2.Cú pháp

<meta charset="UTF-8">

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

67.Thẻ tag <meter> -- THANH ĐO GIÁ TRỊ

67.1.Bản chất

Hiển thị giá trị trong khoảng

Không kéo được , không làm thanh trượt chỉ để làm giá trị đo

67.2.Cú pháp #

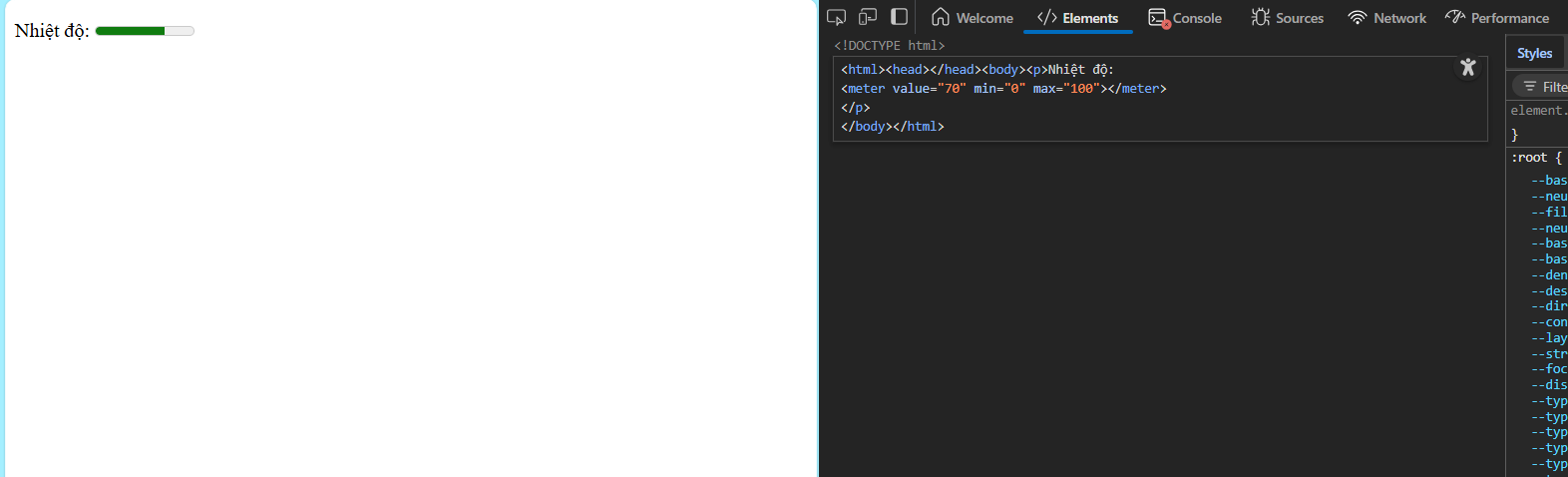
<meter value="30" min="0" max="100"></meter>

67.3.Examble

<p>Nhiệt độ: <meter value="70" min="0" max="100"></meter>

</p>

Hiện thị nhiệt



68.Thẻ tag <nav> -Điều hướng trang wed [RQT]

68.1.Bản chất

<nav> Dùng để chứa các liên kết điều hướng chính của trang wed

Không dùng cho link vặt

68.2.Cú pháp #

<nav>

  <!-- các link -->

</nav>

68.3.Examble điều hướng của bảng điều khiển

<nav>

  <a href="#">Trang chủ</a>

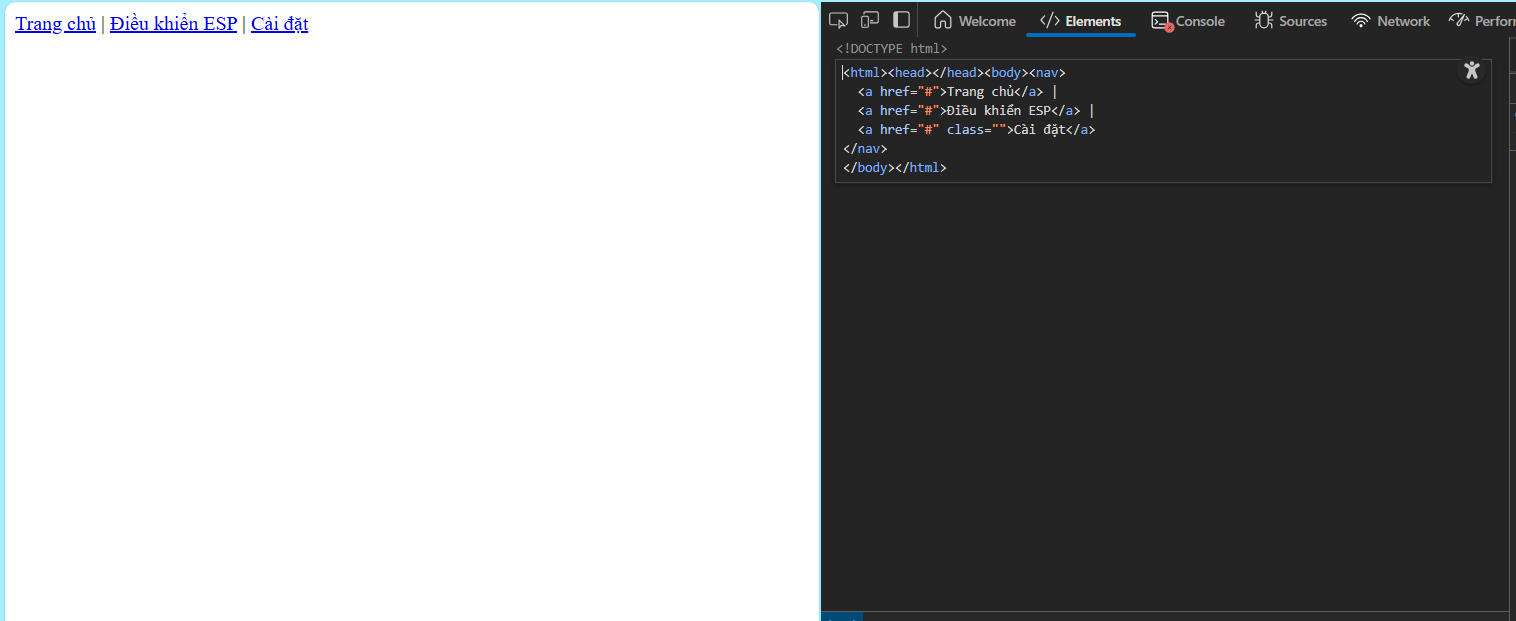
  |

  <a href="#">Điều khiển ESP</a>

  |

  <a href="#">Cài đặt</a>

</nav>



68.4.Examble điều hướng bảng điều khiển

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP32 Web Control</title>

  <style>

    body {

      font-family: Arial;

      margin: 0;

    }

    nav {

      background: #222;

      padding: 10px;

    }

    nav a {

      color: white;

      text-decoration: none;

      margin-right: 15px;

      font-weight: bold;

    }

    nav a:hover {

      color: yellow;

    }

    section {

      padding: 20px;

      display: none;

    }

    section.active {

      display: block;

    }

  </style>

</head>

<body>

  <!-- MENU ĐIỀU HƯỚNG -->

  <nav>

    <a href="#" onclick="show('home')">Trang chủ</a>

    <a href="#" onclick="show('led')">LED</a>

    <a href="#" onclick="show('sensor')">Cảm biến</a>

  </nav>

  <!-- NỘI DUNG -->

  <section id="home" class="active">

    <h2>Trang chủ</h2>

    <p>ESP32 Web Control</p>

  </section>

  <section id="led">

    <h2>Điều khiển LED</h2>

    <button onclick="alert('LED BẬT')">Bật LED</button>

  </section>

  <section id="sensor">

    <h2>Cảm biến</h2>

    <p>Nhiệt độ: 30°C</p>

  </section>

  <script>

    function show(id) {

      let sections = document.querySelectorAll("section");

      sections.forEach(s => s.classList.remove("active"));

      document.getElementById(id).classList.add("active");

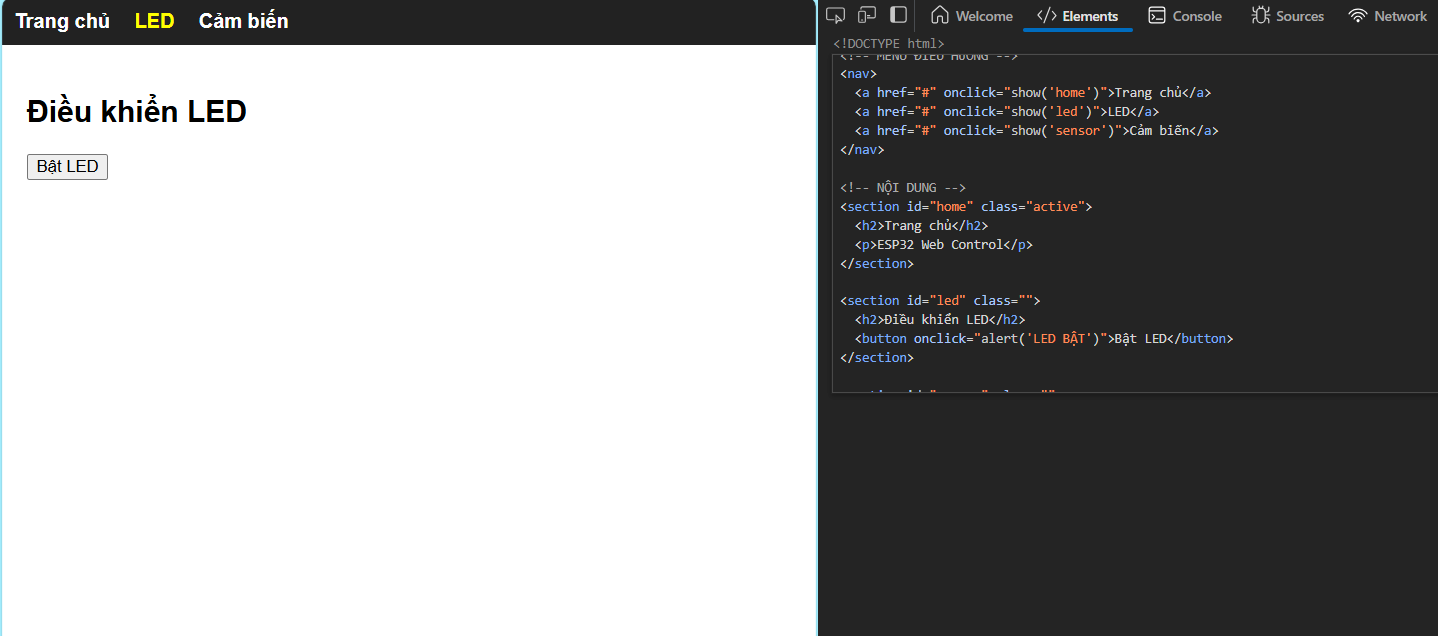
    }

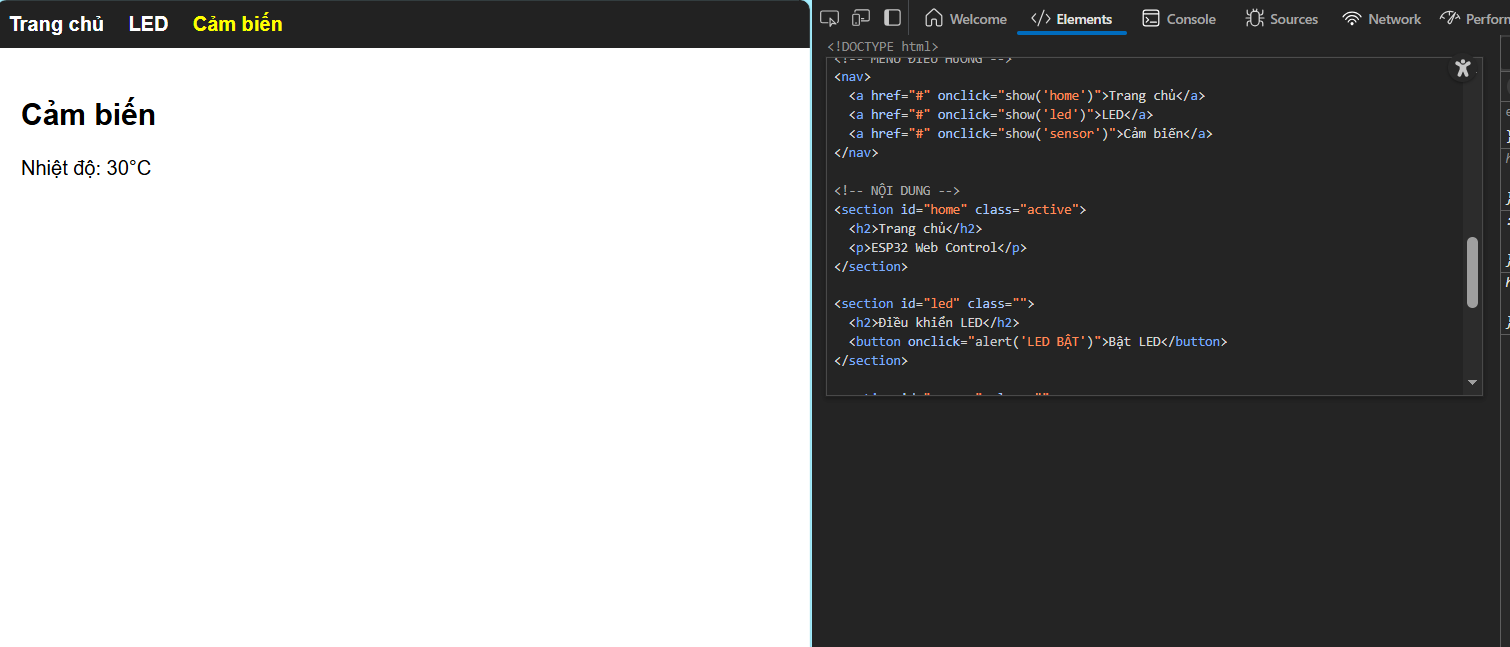
  </script>

</body>

</html>







68.5.Examble

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test HTML 1 → 9</title>

</head>

<body>

  <h2>1. &lt;details&gt; + &lt;summary&gt;</h2>

  <details>

    <summary>Bấm để mở</summary>

    <p>Nội dung bị ẩn</p>

  </details>

  <hr>

  <h2>2. &lt;progress&gt;</h2>

  <p>Đang nạp...</p>

  <progress value="60" max="100"></progress>

  <hr>

  <h2>3. &lt;meter&gt;</h2>

  <p>Nhiệt độ:</p>

  <meter value="35" min="0" max="100"></meter>

  <hr>

  <h2>4. &lt;output&gt;</h2>

  <input type="range" min="0" max="100" value="50" oninput="out.value = this.value">

  <output id="out">50</output>

  <hr>

  <h2>5. &lt;datalist&gt;</h2>

  <input list="wifi" placeholder="Chọn WiFi">

  <datalist id="wifi">

    <option value="ESP32\_AP">

    <option value="Home\_WiFi">

    <option value="Office\_WiFi">

  </datalist>

  <hr>

  <h2>6. &lt;fieldset&gt; + &lt;legend&gt;</h2>

  <fieldset>

    <legend>Điều khiển LED</legend>

    <button>BẬT</button>

    <button>TẮT</button>

  </fieldset>

  <hr>

  <h2>7. &lt;kbd&gt;</h2>

  <p>Nhấn <kbd>Ctrl</kbd> + <kbd>S</kbd> để lưu</p>

  <hr>

  <h2>8. &lt;template&gt;</h2>

  <button onclick="tao()">Tạo thẻ</button>

  <template id="tpl">

    <div style="border:1px solid black; padding:5px; margin:5px;"> ESP Device </div>

  </template>

  <script>

    function tao() {

      let t = document.getElementById("tpl");

      let clone = t.content.cloneNode(true);

      document.body.appendChild(clone);

    }

  </script>

  <hr>

  <h2>9. &lt;noscript&gt;</h2>

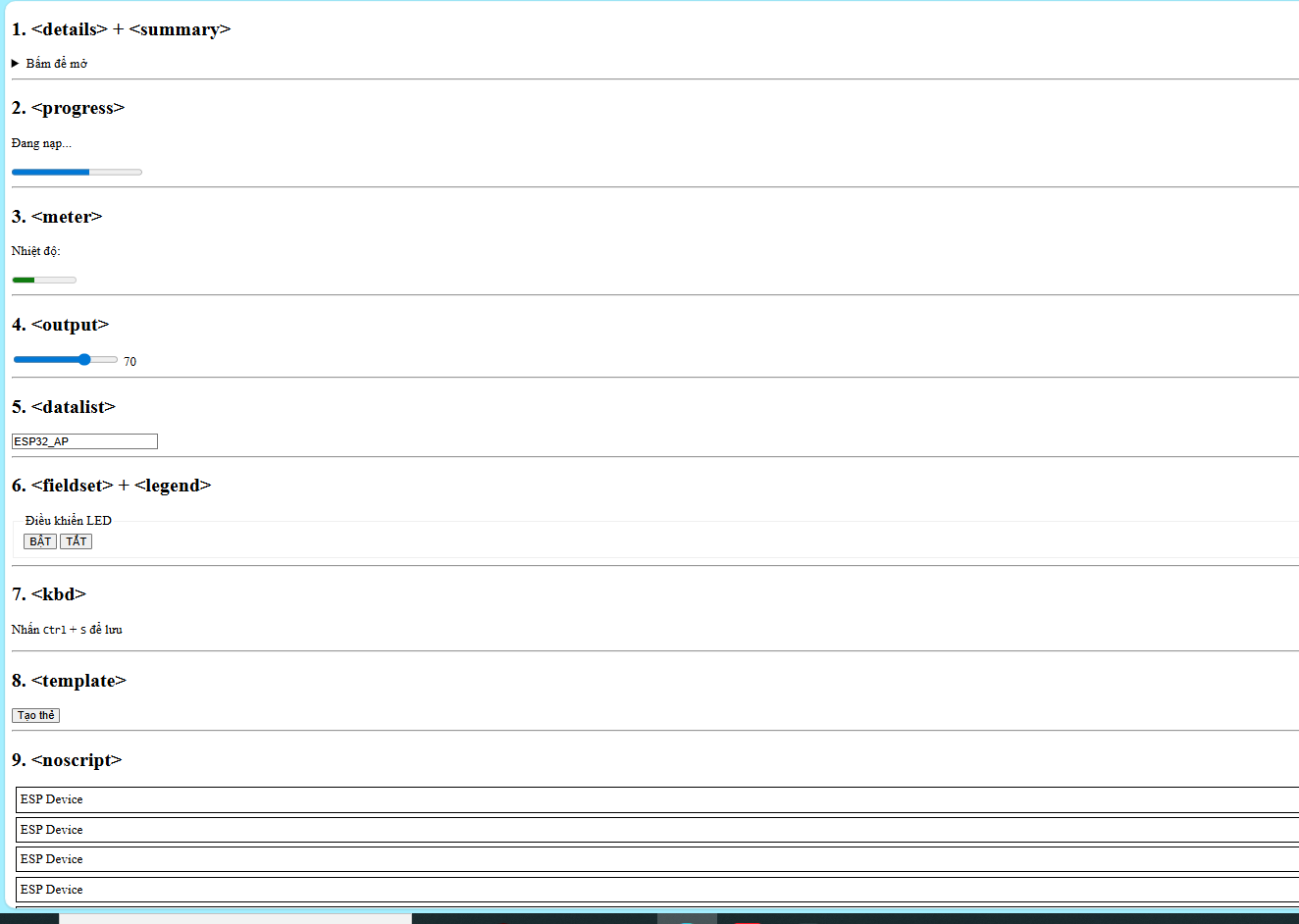
  <noscript>

    <p style="color:red">Web này cần bật JavaScript</p>

  </noscript>

</body>

</html>



69.Thẻ tag <noscript> -

69.1.Bản chất

Nội dung dự phòng khi không có javaScript

69.2.Cú pháp #

   <noscript> Nội dung khi không có JavaScript </noscript>

70.Thẻ tag <object> -- Nhúng tài nguyên bên ngoài

70.1.Bản chất

Nhúng file bên ngoài vào html

Có thể là pdf , ảnh video , html khác

70.2.Cú pháp #

<object data="file.pdf" type="application/pdf"></object>

70.3.Lưu ý

Không dùng cho logic

71.Thẻ tag <ol> - danh sách có thứ tự

71.1.Cú pháp

<ol>

  <li>LED</li>

  <li>Motor</li>

</ol>

71.2.Examble

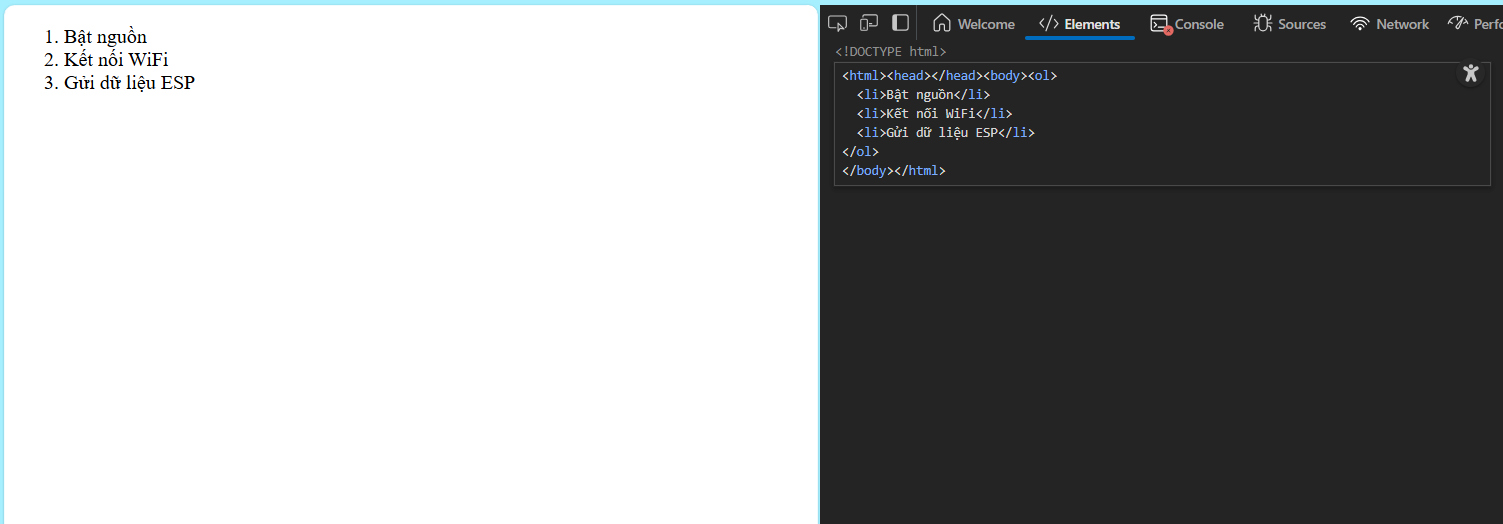
<ol>

  <li>Bật nguồn</li>

  <li>Kết nối WiFi</li>

  <li>Gửi dữ liệu ESP</li>

</ol>



72.Thẻ tag <option> -- 1 lựa chọn trong <select> [RQT]

72.1.Bản chất

Không đứng một mình

Luân nằm trong <select> hoặc <datalist>

72.2.Cú pháp#

<option value="giatri">Nội dung hiển thị</option>

72.3.Examble

<!DOCTYPE html>

<html>

<body>

  <select>

    <option>LED 1</option>

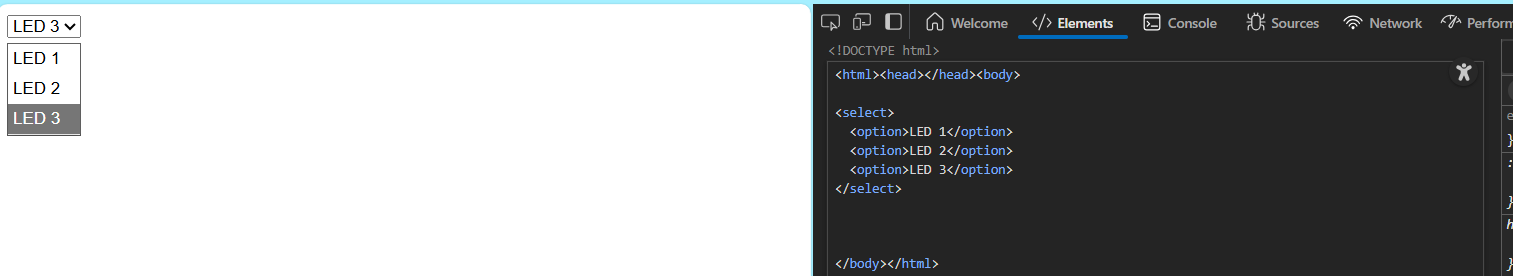
    <option>LED 2</option>

    <option>LED 3</option>

  </select>

</body>

</html>



72.4.Examble có value [QT]

<select>

  <option value="1">LED 1</option>

  <option value="2">LED 2</option>

  <option value="3">LED 3</option>

</select>

72.5.Examble đọc <option> bằng javaScript

<!DOCTYPE html>

<html>

<body>

  <select onchange="chon(this.value)">

    <option value="0">TẮT</option>

    <option value="1">BẬT</option>

  </select>

  <script>

    function chon(val) {

      alert("Giá trị chọn: " + val);

    }

  </script>

</body>

</html>

72.6.Examble option mặc đinh vs khóa

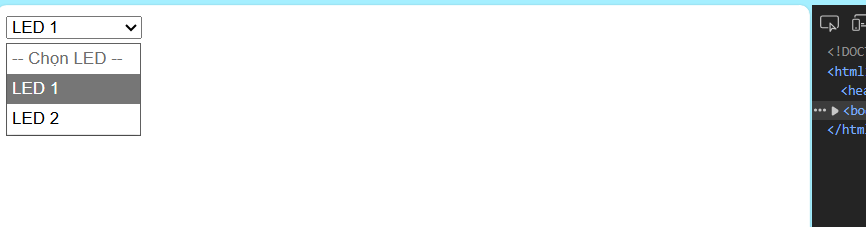
<select>

  <option value="" disabled selected>-- Chọn LED --</option>

  <option value="1">LED 1</option>

  <option value="2">LED 2</option>

</select>



72.7.Examble dùng cho ESP

<select onchange="guiLenh(this.value)">

  <option value="LED\_ON">Bật LED</option>

  <option value="LED\_OFF">Tắt LED</option>

</select>

<script>

  function guiLenh(cmd) {

    console.log("Gửi ESP:", cmd);

  }

</script>

73.Thẻ tag <optgroup> -- Nhóm các option [RQT]

73.1.Bản chất

Gom các <option> thành nhóm

73.2.Examble gom các option

<select>

  <optgroup label="LED">

    <option>LED 1</option>

    <option>LED 2</option>

  </optgroup>

  <optgroup label="Motor">

    <option>Motor A</option>

    <option>Motor B</option>

  </optgroup></select>



74.Thẻ tag <output> --Hiện thị kết quả [RQT]

74.1.Bản chất

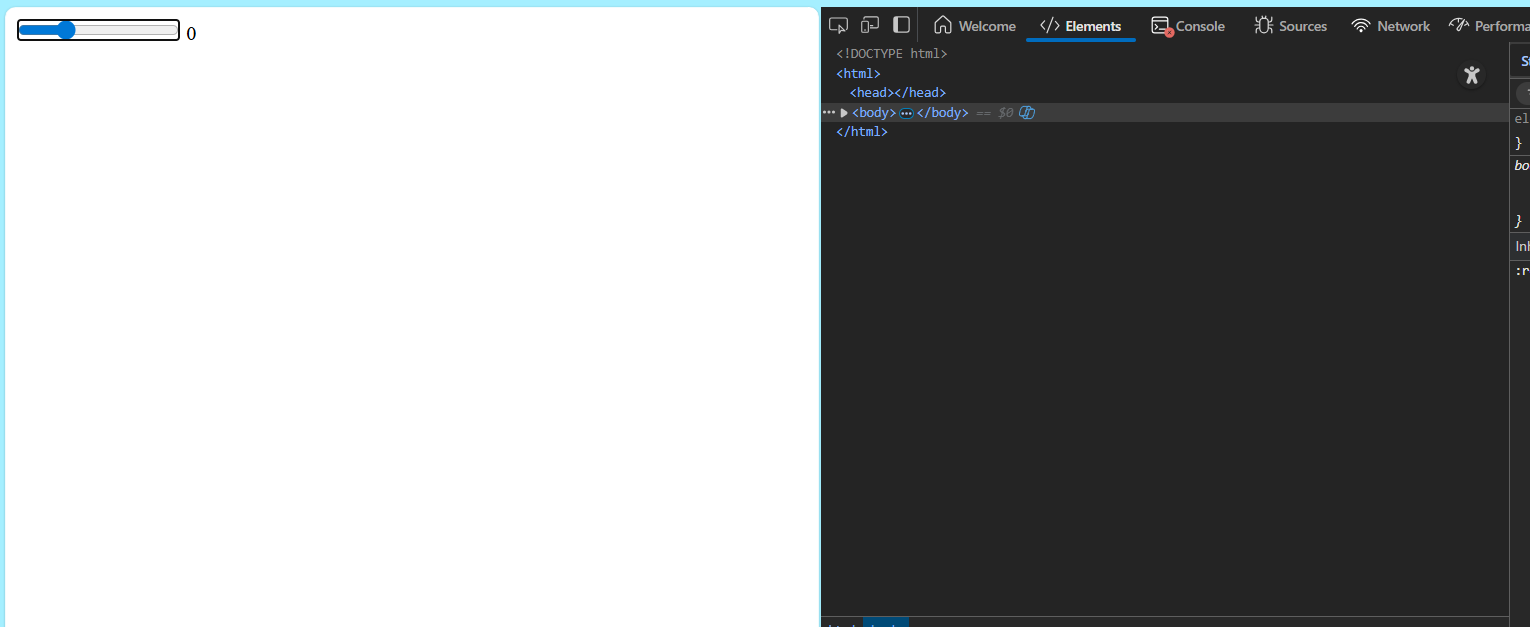
Chỉ hiện thị

Không nhập dữ liệu

74.2.Examble

<input type="range" min="0" max="255" oninput="out.value = this.value">

<output id="out">0 </output>



75.Thẻ tag <p> - đoạn văn

75.1.Bản chất

Để hiện thông tin như dưới phần tiêu đề

75.2.Cú pháp #

<p> nội dung </p>

75.3.Examble

<p>LED đang hoạt động bình thường</p>

<p>Nhiệt độ: 35°C</p>

75.4.Note ( lưu ý , ghi chú)

Ko dùng <p> để bọc div , button , input

76.Thẻ tag <pre> --- Hiện Thị Nguyên Dạng

76.1.Bản chất

Giữ nguyên : khoảng trắng , xuống dòng

Rất hợp cho code : esp ,log ESP , Dữ liệu debug

76.2.Cú pháp #

<pre>Nội dung</pre>

76.3.Examble

<pre>

ESP32 CONNECTED

IP: 192.168.1.10

PWM: 120</pre>

77.Thẻ tag <progress> --THANH TIẾN TRÌNH

77.1.Bản chất

Hiển thị tốc độ

Không phải thanh trượt

Không kéo được

77.2.Cú pháp #

<progress value="30" max="100"></progress>

77.3.Examble

<label>Đang tải dữ liệu ESP</label>

<br>

<progress value="60" max="100"></progress>



78.Thẻ tag <q> -- Trích dẫn ngắn

78.1.Examble

<!DOCTYPE html>

<html lang="vi">

<body>

  <p>

    ESP32 hoạt động ổn định vì <q>kết nối mạng và logic tách biệt rõ ràng

    </q>

  </p>

</body>

</html>

79.Thẻ tag <samp> -- Kết quả từ máy tính / chương trình

79.1.Bản chất

Hiện thị kết quả đầu ra của chương trình / thiết bị

79.2.Cú pháp #

<samp>nội dung</samp>

79.3.Examble

<p>ESP trả về: <samp>LED ON</samp>

</p>

79.4.note

chỉ để hiện thị ,ko xử lý logic

Dùng để : hiện log ,trạng thái…

80.Thẻ tag <script> ---- JavaScript [RQT]

80.1.Bản chất

Là nơi viết logic xử lý ( bấm nút , gửi lệnh , dữ liệu)

80.2.Cú pháp#

<script>

  // code JS

</script>

80.3.Examble

<script>

  function batLed() {

    alert("Bật LED");

  }

</script>

80.4.note

Html ko làm logic , js lamf logic

Gửi HTTP

81.Thẻ tag <section> ---Khu vực nội dung

81.1.Bản chất

Chia trang thành các khối có ý nghĩa

81.2.Cú pháp#

<section>...</section>

81.3.Examble

82.Thẻ tag <select> Danh sách chọn [RQT]

82.1.Bản chất

Chọn 1 giá trị trong nhiều giá trị

82.2.Cú pháp #

<select>

  <option>...</option>

</select>

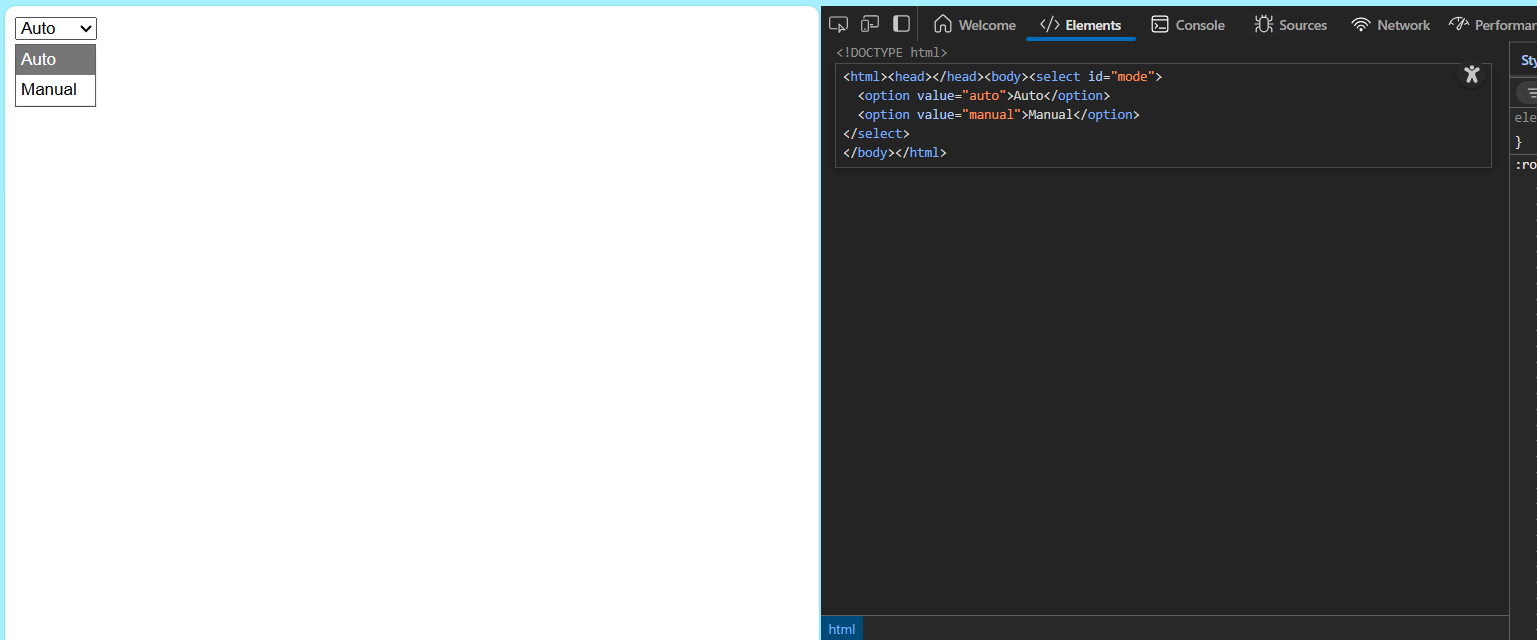
82.3.Examble

<select id="mode">

  <option value="auto">Auto</option>

  <option value="manual">Manual</option>

</select>



82.4.NONE

Dùng chung với js để lấy giá trị

Dùng cho : chọn chế độ , chọn tốc độ

83.Thẻ tag <small> -- Ghi chú nhỏ [RQT]

83.1.Cú pháp#

<small> …. </small>

83.2.Examble

<p>Nhiệt độ <small>(°C)</small>

</p>



83.3.Examble

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test small</title>

</head>

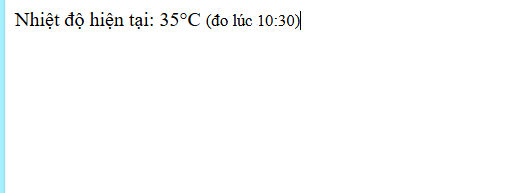
<body>

  <p> Nhiệt độ hiện tại: 35°C <small>(đo lúc 10:30)</small>

  </p>

</body>

</html>



83.4.Examble trang thái điêu khiển

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP Control VIP</title>

  <style>

    body {

      font-family: Arial;

      background: #111;

      color: #fff;

      padding: 30px;

    }

    .panel {

      width: 320px;

      background: #1e1e1e;

      border-radius: 12px;

      padding: 20px;

    }

    .device {

      margin-bottom: 20px;

    }

    .device h3 {

      margin: 0;

      font-size: 20px;

    }

    .device small {

      color: #aaa;

      display: block;

      margin-top: 3px;

    }

    .status {

      font-size: 18px;

      margin-top: 8px;

    }

    button {

      margin-top: 10px;

      padding: 8px 16px;

      border: none;

      border-radius: 6px;

      background: #00c853;

      color: black;

      font-weight: bold;

      cursor: pointer;

    }

  </style>

</head>

<body>

  <div class="panel">

    <div class="device">

      <h3>LED PHÒNG KHÁCH</h3>

      <small>GPIO 2 • Chế độ tay</small>

      <div class="status">Trạng thái: <strong>ON</strong>

      </div>

      <button>BẬT / TẮT</button>

    </div>

    <div class="device">

      <h3>QUẠT LÀM MÁT</h3>

      <small>GPIO 5 • PWM 120</small>

      <div class="status">Tốc độ: <strong>60%</strong>

      </div>

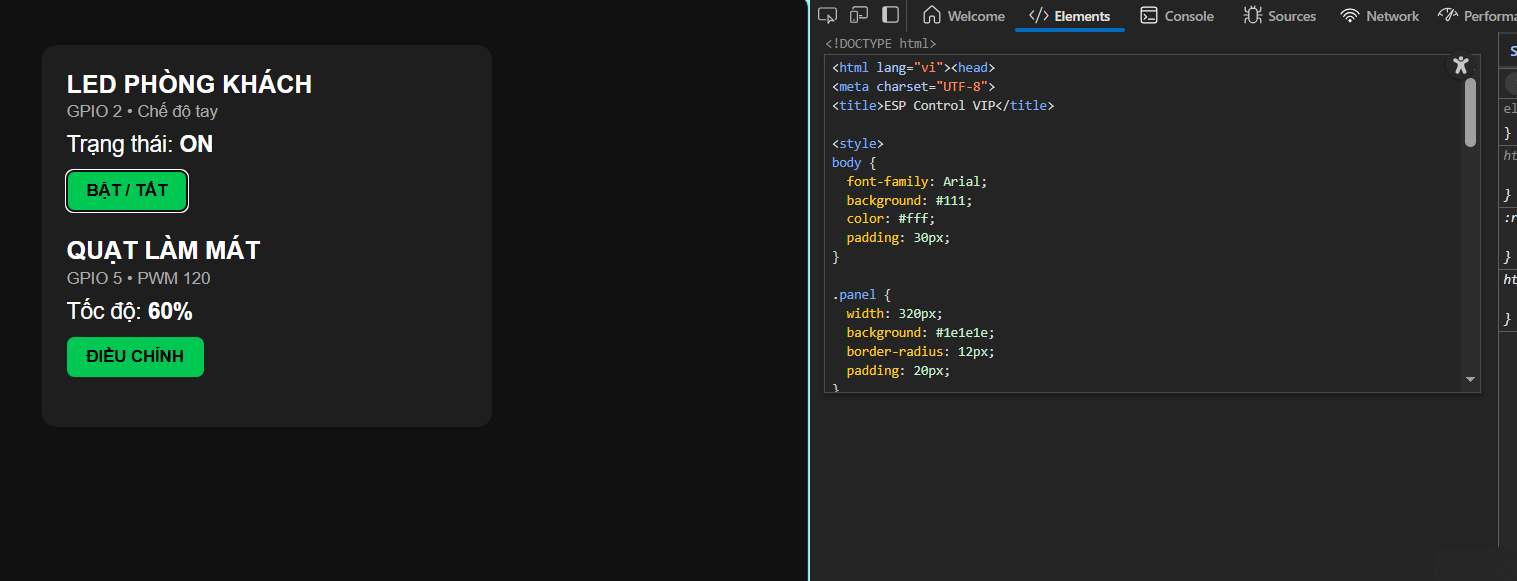
      <button>ĐIỀU CHỈNH</button>

    </div>

  </div>

</body>

</html>



83.5.Examble

83.3.Note

Không dùng cho nội dung chính

84.Thẻ tag <source> --- Nguồn media

84.1.Bản chất

Chỉ định cho file audio/video

84.2.Cú pháp #

<source src="" type="">

84.3.Examble

<audio controls>

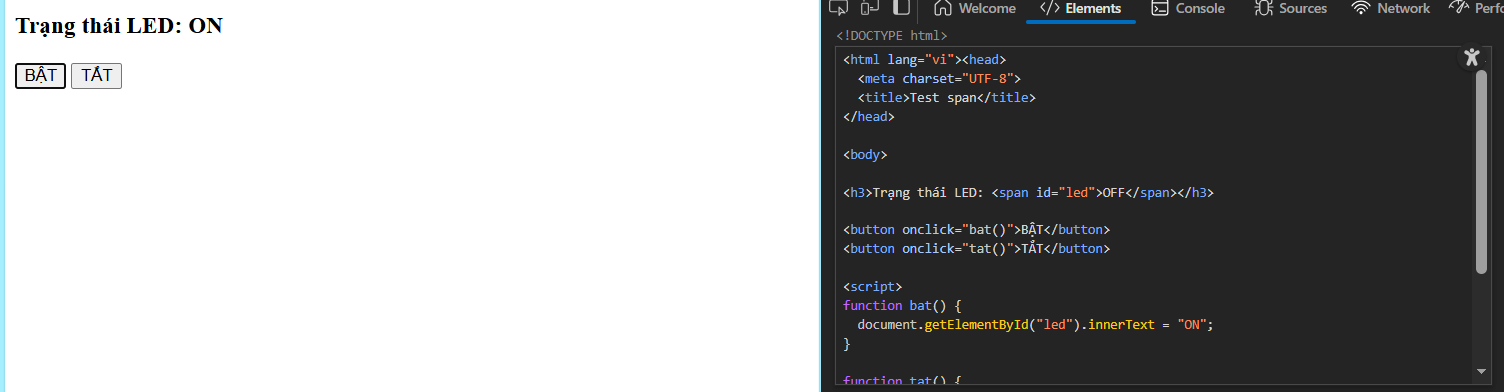
  <source src="beep.mp3" type="audio/mpeg">

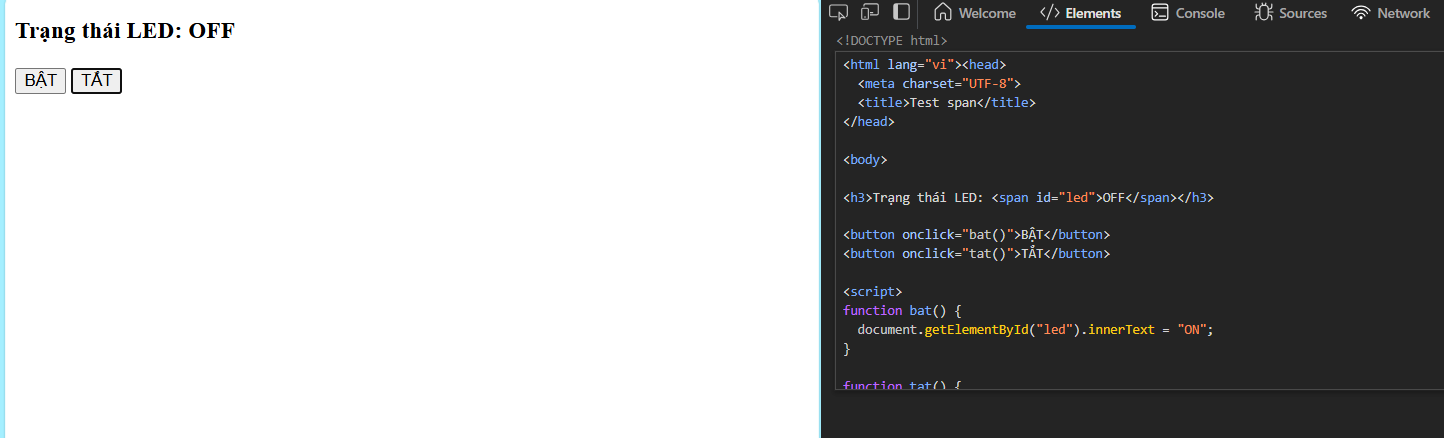
</audio>

85.Thẻ tag <span> ---Bọc chữ inline [RQT]

85.1.Bản chất

Hiện dữ liệu lên màn hình





85.2.Cú pháp #

<span>… </span>

85.3.Examble

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test span</title>

</head>

<body>

  <h3>Trạng thái LED: <span id="led">OFF</span>

  </h3>

  <button onclick="bat()">BẬT</button>

  <button onclick="tat()">TẮT</button>

  <script>

    function bat() {

      document.getElementById("led").innerText = "ON";

   }

    function tat() {

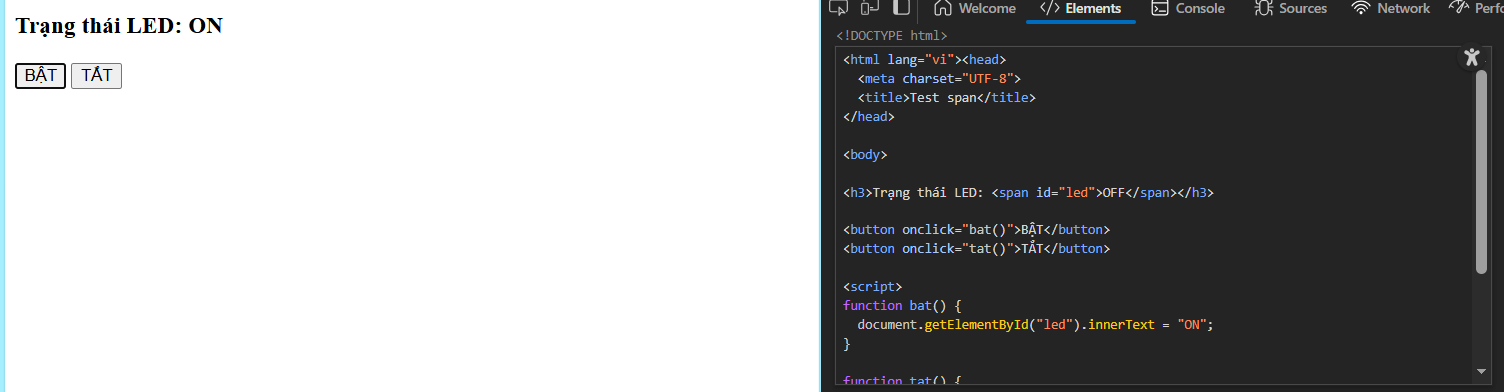
      document.getElementById("led").innerText = "OFF";

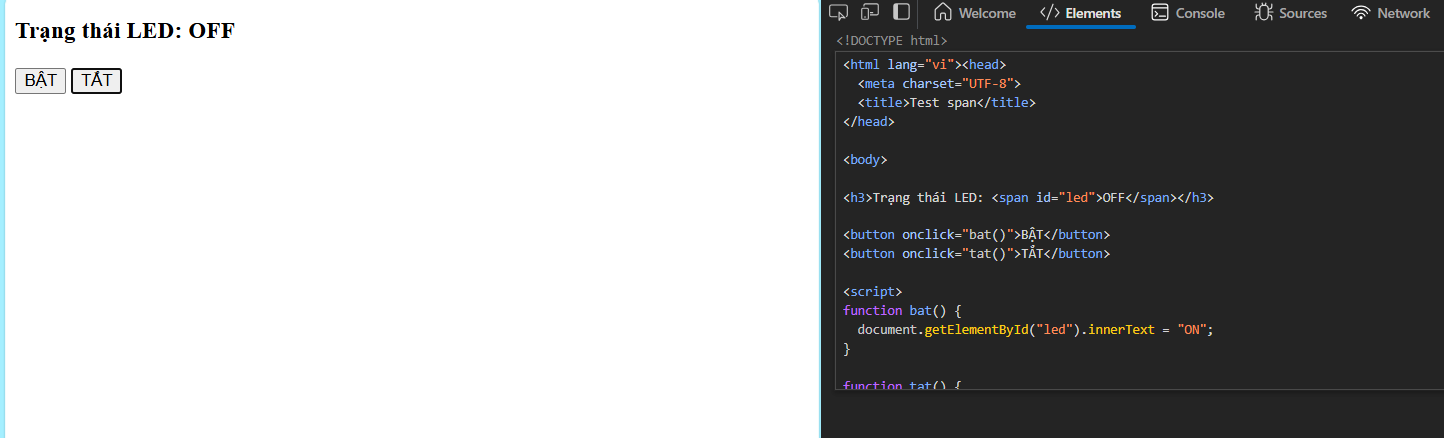
    }

  </script>

</body>

</html>





85.2.Note

Không xuống dòng

Dùng cho hiện thị trang thái real-time

86.Thẻ tag <strong> ---Nhấn mạnh nội dung

86.1.Cú pháp

<strong>… </strong>

86.2.Examble

<strong>CẢNH BÁO!</strong>

86.3.Note

Có ý nghĩa ngữ khác <b>

87.Thẻ tag <style> ---CSS trong HTML

87.1.Bản chất

Trang trí giao diện

87.2.Cú pháp #

<style>

  selector {

    thuộc tính

  }

</style>

87.3.Examble

<style>

  button {

    font-size: 20px;

  }

</style>

88.Thẻ tag <sub> & <sup>

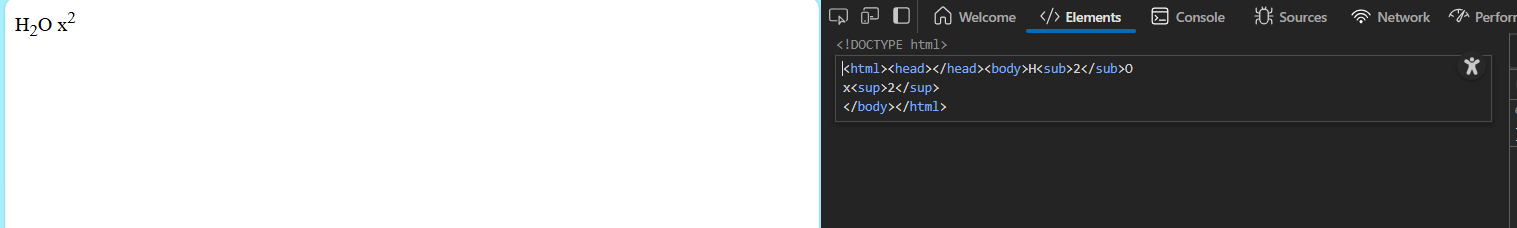
88.1.Bản chất

<sub> : Chỉ số dưới

<sup> : chỉ số trên

88.2.Examble

<sub>2</sub>O x<sup>2</sup>



89.Thẻ tag <summary> & <details> [RQT]

89.1.Bản chất

Giao diện mở / đóng

89.2.Cú pháp

<details>

  <summary>...</summary> nội dung

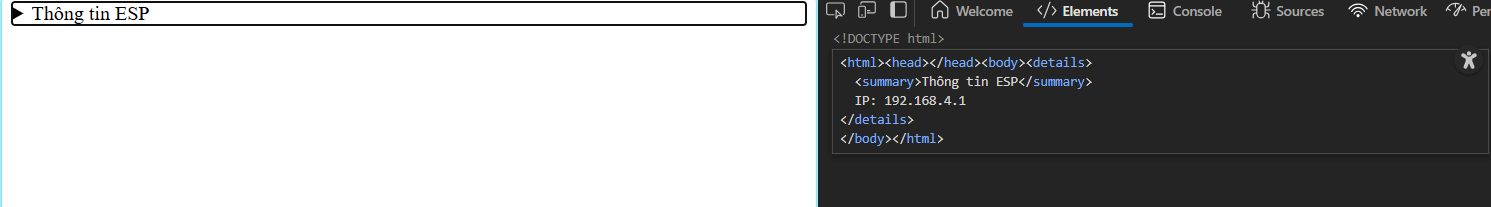
</details>

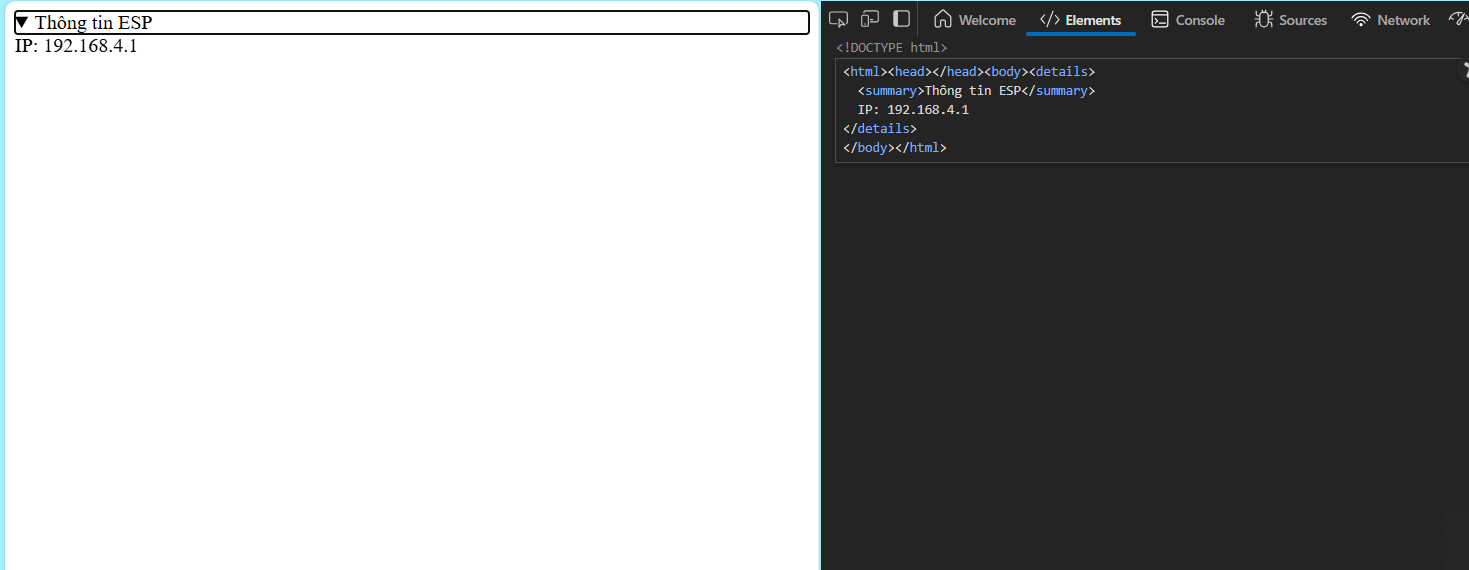
89.3.Examble

<details>

  <summary>Thông tin ESP</summary> IP: 192.168.4.1

</details>





90.Thẻ tag <table> ---Bảng dữ liệu[RHD]

90.1.Bản chất

Tạo bảng có hàng - cột

90.2.Cú pháp#

<table> … </table>

90.3.Examble

<table border="1">

  <tr>

    <th>Thiết bị</th>

    <th>Trạng thái</th>

  </tr>

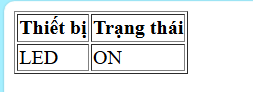
  <tr>

    <td>LED</td>

    <td>ON</td>

  </tr>

</table>



90.4.Note

Dùng dùng layout giao diện ( dùng <div> tốt hơn )

91.Thẻ tag <thead> , <tbody> , <tfoot> --- Phần bảng [RQT]

91.1.Bản chất

Chia thành đầu - chân - chân

91.2.Cú pháp #

<thead></thead>

<tbody></tbody>

<tfoot></tfoot>

91.3.Examble

<table border="1">

  <thead>

    <tr>

      <th>Tên</th>

      <th>Giá trị</th>

    </tr>

  </thead>

  <tbody>

    <tr>

      <td>Nhiệt độ</td>

      <td>30°C</td>

    </tr>

  </tbody>

  <tfoot>

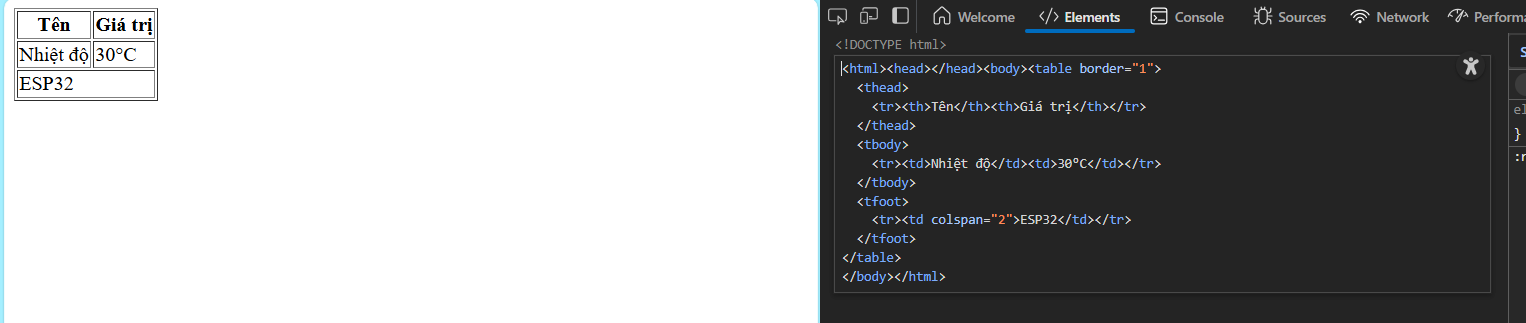
    <tr>

      <td colspan="2">ESP32</td>

    </tr>

  </tfoot>

</table>



91.4.Kinh nghiệm

JS rất hay update <tbody>

92.Thẻ tag <tr> --- Hàng [QT]

92.1.Bản chất

1 hàng trong bảng

92.2.Cú pháp#

<tr>...</tr>

93.Thẻ tag <td> -- Ô dữ liệu [QT]

93.1.Bản chất

Ô thường

93.2.Cú pháp#

<td>ON</td>

94.Thẻ tag <th> ---Ô tiêu đề [QT]

94.1.Bản chất

Ô in đâm , Căn lề

94.2.Cú p háp#

<th>Thiết bị</th>

95.Thẻ tag <template> ---KHUNG ẨN (VIP) [RQT]

95.1.Bản chất

HTML không hiện , Dùng cho js clone

95.2.Examble dùng js tạo bảng động

<template id="row">

  <tr>

    <td>LED</td>

    <td>ON</td>

  </tr>

</template>

96.Thẻ tag <textarea> -- ô nhập nhiều dòng có thể kéo góc to

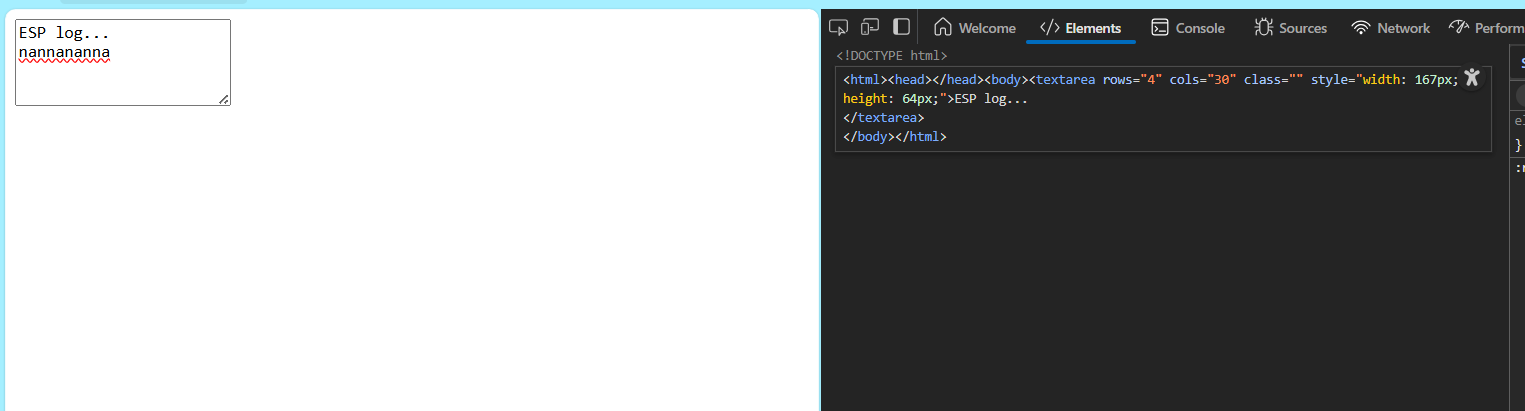
96.1.Bản chất

Nhập text dài (log,lệnh)

96.2.Examble

<textarea rows="4" cols="30">

ESP log...</textarea>



96.2.1.Exmable khu ẩn ấn hiện thêm ô thêm cột dữ liệu

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP Template Demo</title>

  <style>

    table {

      border-collapse: collapse;

      width: 300px;

    }

    th,

    td {

      border: 1px solid black;

      padding: 8px;

      text-align: center;

    }

  </style>

</head>

<body>

  <h2>Bảng trạng thái ESP</h2>

  <table>

    <thead>

      <tr>

        <th>Thiết bị</th>

        <th>Trạng thái</th>

      </tr>

    </thead>

    <tbody id="tableBody">

      <!-- JS sẽ chèn dữ liệu vào đây -->

    </tbody>

  </table>

  <!-- TEMPLATE (ẨN) -->

  <template id="rowTemplate">

    <tr>

      <td class="device"></td>

      <td class="status"></td>

    </tr>

  </template>

  <button onclick="addRow()">Thêm dữ liệu</button>

  <script>

    function addRow() {

      // lấy template

      const template = document.getElementById("rowTemplate");

      // clone template

      const clone = template.content.cloneNode(true);

      // gán dữ liệu

      clone.querySelector(".device").textContent = "LED";

      clone.querySelector(".status").textContent = "ON";

      // chèn vào bảng

      document.getElementById("tableBody").appendChild(clone);

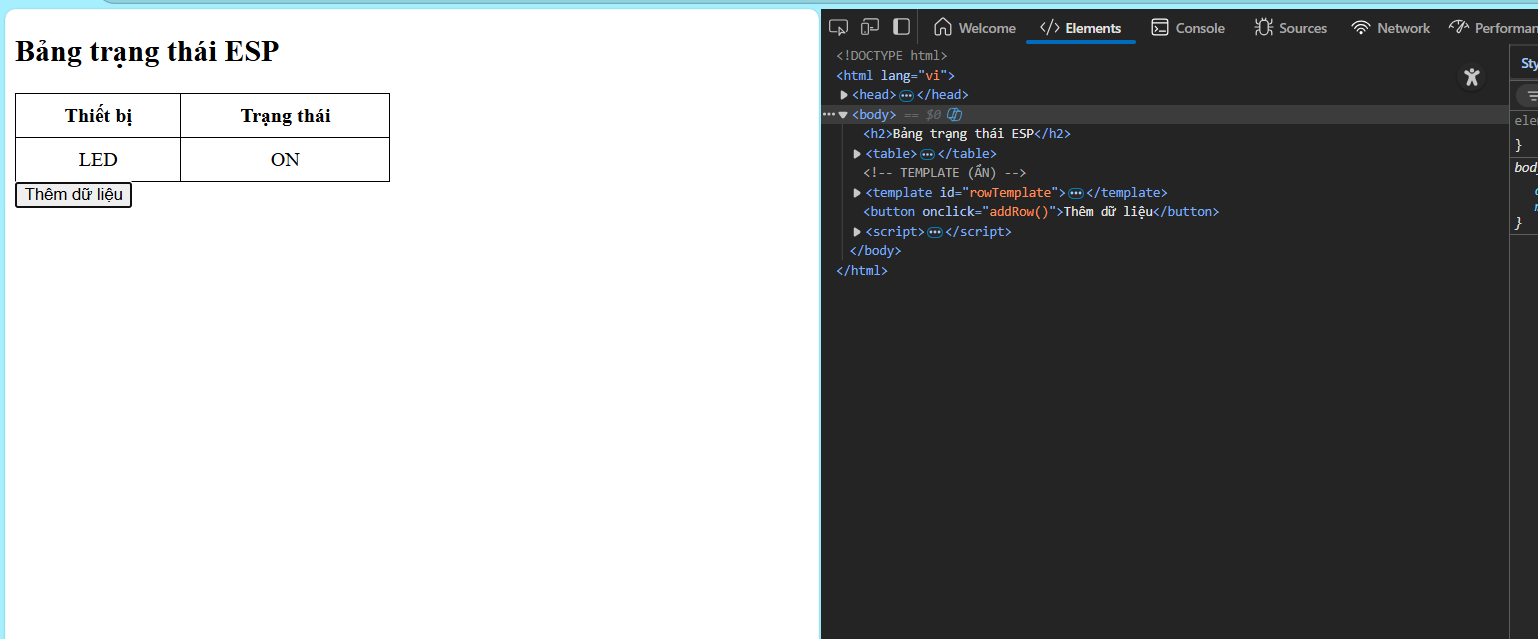
    }

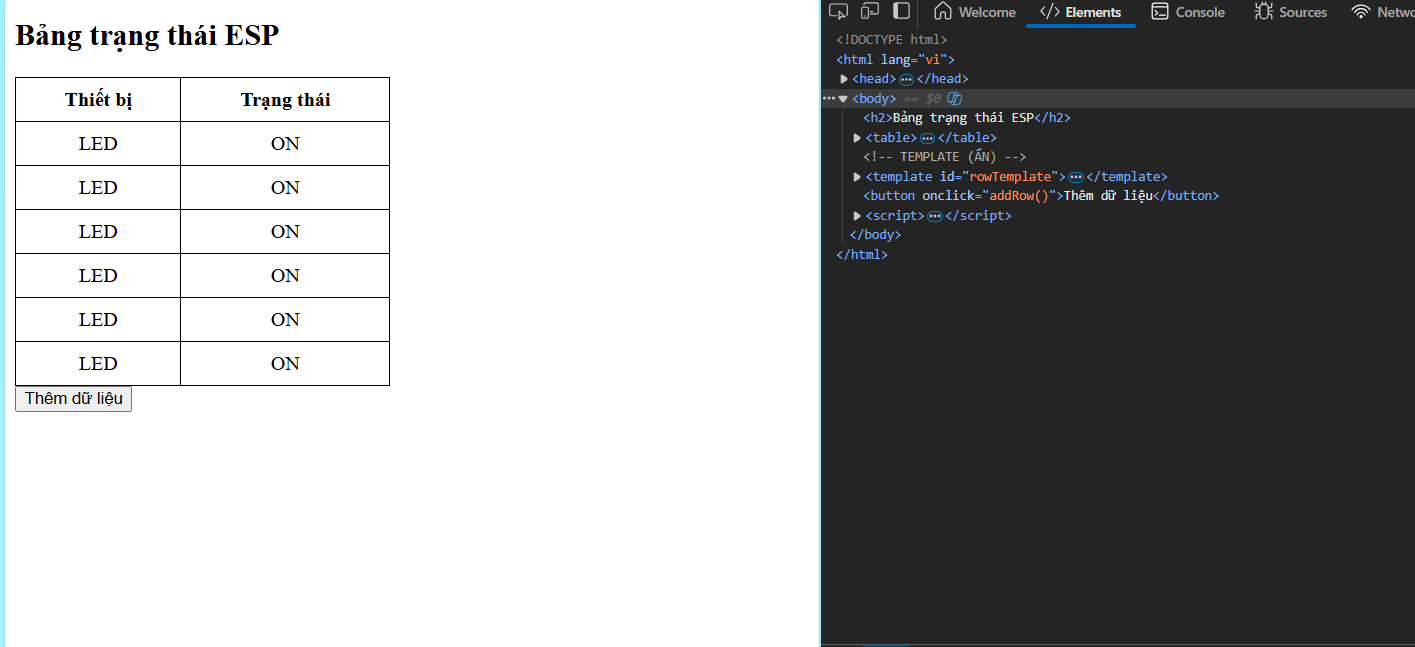
  </script>

</body>

</html>







96.3.Kinh nghiệm’

Dùng để hiện thị log ESP / MQTT

97.Thẻ tag <time> --- Thời gian

97.1.Bản chất

Hiện thị thời gian có ý nghĩa máy hiểu

97.2.Examble

<time datetime="2025-01-21">21/01/2025</time>

98.Thẻ tag <title> --- Tên TAB [RQT]

<title> tên tiêu đề trang wed </title>

98.1.Examble

<html>

<head>

  <meta charset=”UTF-8”>

  /\* khai báo phong chữ tiếng việt \*/

  <title> wed của tôi </title>

</head>

</html

99.Thẻ tag <track> --PHỤ ĐỀ VIDEO

99.1.Bản chất

Gắn subtitle cho video

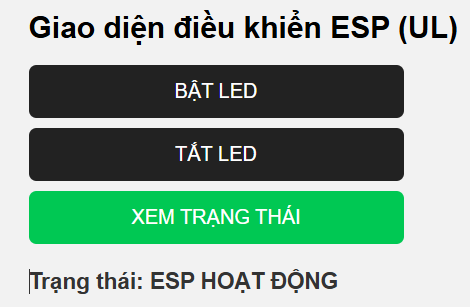
99.2.Examble

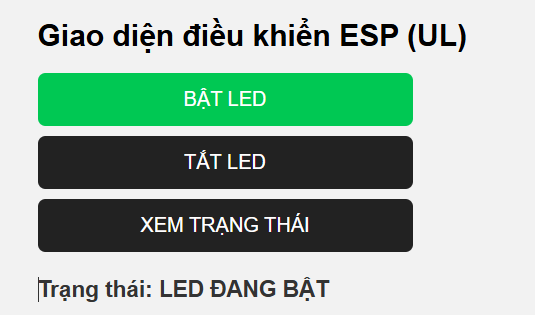
<track kind="subtitles" src="sub.vtt">

100.Thẻ tag <ul> --- Danh sách không thứ tự [QT]

100.1.Bản chất

Danh sách k có thứ tự





100.2.Cú pháp#

<ul>

  <li>...</li>

</ul>

100.3.Examble

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Demo thẻ UL - Giao diện ESP</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      background: #f2f2f2;

      padding: 30px;

    }

    h2 {

      margin-bottom: 10px;

    }

    /\* ===== UL ===== \*/

    ul {

      list-style: none;

      /\* bỏ dấu chấm \*/

      padding: 0;

      width: 300px;

    }

    li {

      background: #222;

      color: #fff;

      padding: 12px;

      margin-bottom: 8px;

      text-align: center;

      border-radius: 6px;

      cursor: pointer;

      user-select: none;

    }

    li:hover {

      background: #00c853;

    }

    /\* trạng thái \*/

    #status {

      margin-top: 20px;

      font-size: 18px;

      font-weight: bold;

      color: #333;

    }

  </style>

</head>

<body>

  <h2>Giao diện điều khiển ESP (UL)</h2>

  <!-- UL là khung -->

  <ul>

    <!-- LI là từng nút -->

    <li onclick="guiLenh('LED\_ON')">BẬT LED</li>

    <li onclick="guiLenh('LED\_OFF')">TẮT LED</li>

    <li onclick="guiLenh('CHECK')">XEM TRẠNG THÁI</li>

  </ul>

  <div id="status">Trạng thái: ---</div>

 <script>

    function guiLenh(lenh) {

      console.log("Gửi lệnh:", lenh);

      if (lenh === "LED\_ON") {

        document.getElementById("status").innerText =

          "Trạng thái: LED ĐANG BẬT";

      }

      if (lenh === "LED\_OFF") {

        document.getElementById("status").innerText =

          "Trạng thái: LED ĐANG TẮT";

      }

      if (lenh === "CHECK") {

        document.getElementById("status").innerText =

          "Trạng thái: ESP HOẠT ĐỘNG";

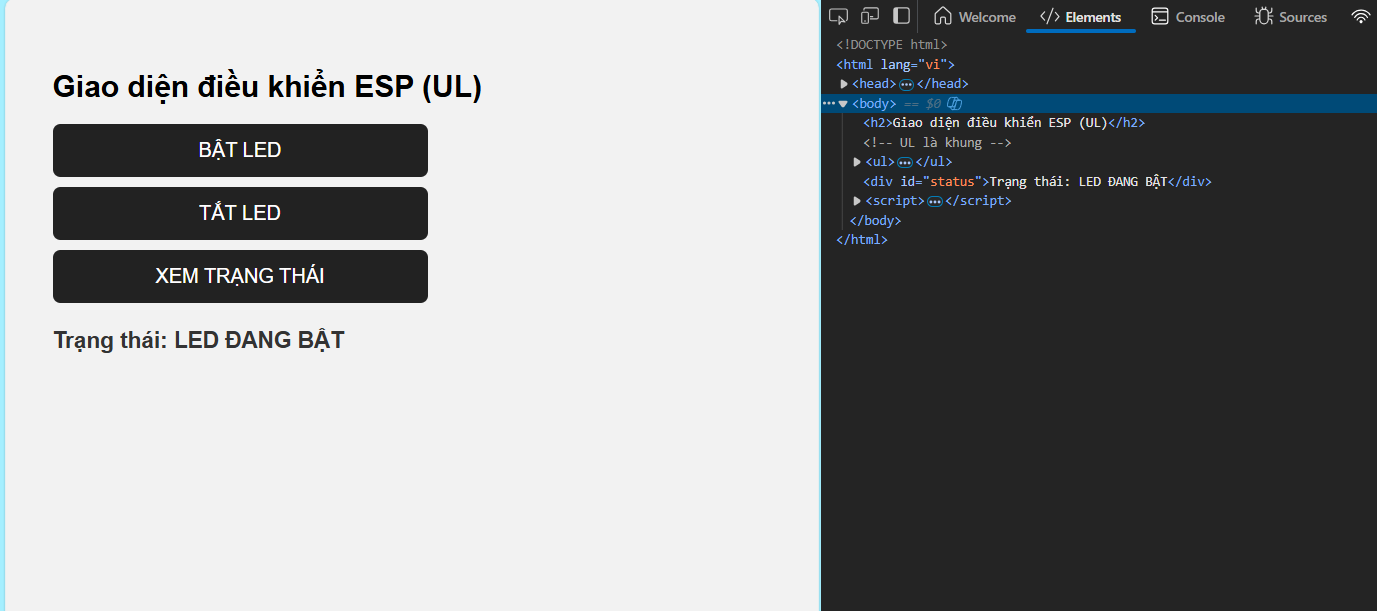
      }

    }

  </script>

</body>

</html>



101.Thẻ tag <video>

101.1.Cú pháp#

<video src="video.mp4" controls></video>

101.2.Examble

<video width="400" controls>

  <source src="demo.mp4" type="video/mp4"> Trình duyệt không hỗ trợ video

</video>

102.Thẻ tag <wbr> — Điểm NGẮT DÒNG TÙY Ý [VIP]

102.1.Bản chất

<wbr> = Word Break Opportunity

Nó chỉ đánh dấu 1 vị trí “ĐƯỢC PHÉP ngắt dòng”

Trình duyệt CHỈ ngắt nếu cần

102.2.Cú pháp

<wbr>

102.3.Examble

<p> ESP32CAM-DEVICE-ID-2026

  <wbr>0123456789ABCDE

</p>

102.4.Examble khi có WBR VÀ KO CÓ

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Demo thẻ WBR</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      padding: 30px;

      background: #f4f4f4;

    }

    .box {

      width: 260px;

      /\* cố tình làm hẹp \*/

      padding: 15px;

      background: white;

      border: 2px solid #333;

      margin-bottom: 25px;

    }

    h3 {

      margin-top: 0;

    }

    .note {

      color: #555;

      font-size: 14px;

    }

  </style>

</head>

<body>

  <h2>Demo thẻ &lt;wbr&gt; (Word Break Opportunity)</h2>

  <!-- KHÔNG có wbr -->

  <div class="box">

    <h3>❌ Không dùng &lt;wbr&gt;</h3>

    <p> ESP32CAMDEVICEID0123456789ABCDEFGHIJK </p>

  </div>

  <!-- CÓ wbr -->

  <div class="box">

    <h3>✅ Có dùng &lt;wbr&gt;</h3>

    <p> ESP32CAMDEVICEID

      <wbr>0123456789ABCDEFGHIJK

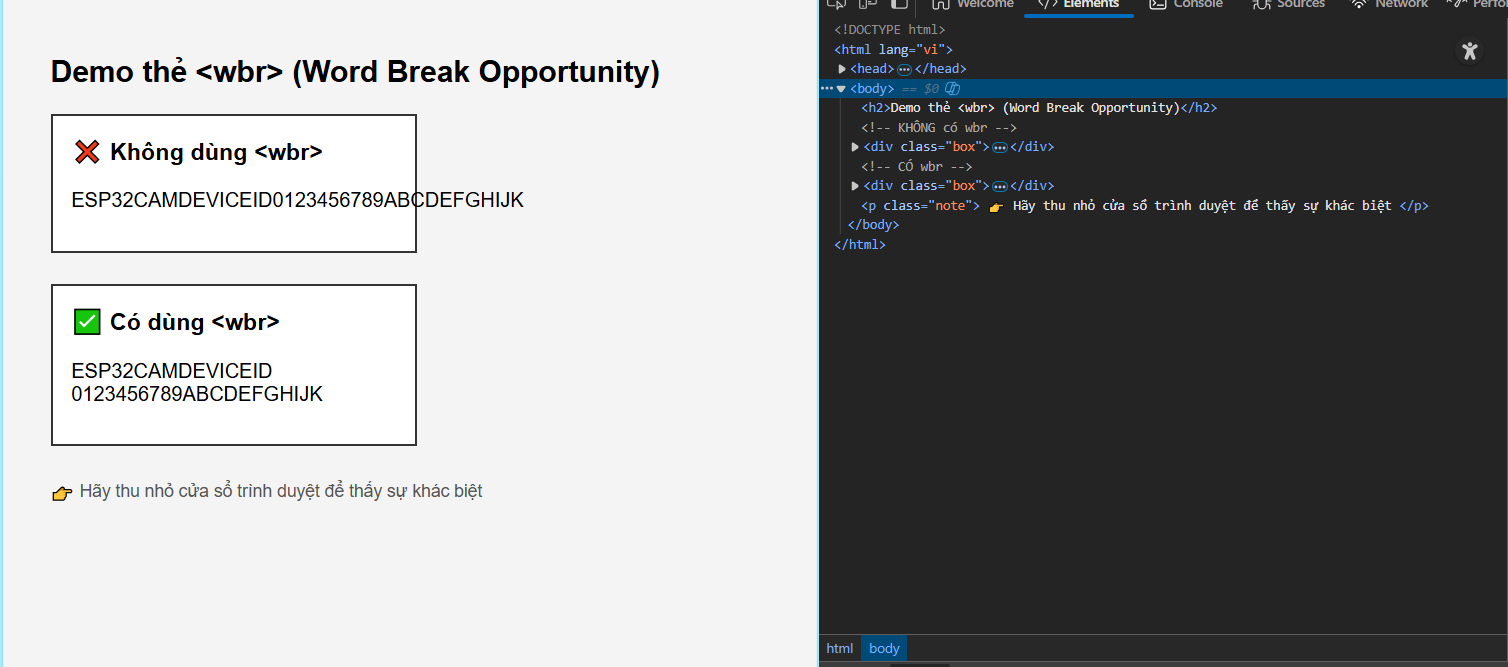
    </p>

  </div>

  <p class="note"> 👉 Hãy thu nhỏ cửa sổ trình duyệt để thấy sự khác biệt </p>

</body>

</html>



TÓM TẮT THUỘC TÍNH CỦA THẺ ĐƠN HTML

=== === === === === THUỘC TÍNH: id === === === === ===

103. Thuộc tính id

103.1.Bản chất(dùng để làm gì) -

Định danh DUY NHẤT cho một phần tử HTML -

Dùng để CSS và JavaScript truy xuất chính xác

103.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific) -

Global -

Áp dụng cho hầu hết mọi thẻ HTML

103.3.Cú pháp

<tag id="box1"> </tag>

103.4.Giá trị thường dùng -

Chuỗi(string) -

Không dấu, không khoảng trắng

103.5.Lưu ý / lỗi hay gặp -

Không được trùng id trong cùng trang -

Không bắt đầu bằng số

\*Định nghĩa phần tử:

- 1 phần tử bao gồm. ví dụ 1 nút bấm trên màn hình đó là 1 phần tử chỉ dùng "id"

- 2 phần tử là số nhiều ví dụ phần tử nút bấm và và phần tử nhập

liệu đều hiện thị trên màn hình và đó là 2 phần tử là số nhiều dung "class"

- khi dùng thuộc tính class nên thêm dấu chấm trước bộ chọn ví dụ : “.nut “

- khi dùng thuộc tính id nên thêm dấu # trước bộ chọn ví dụ : “#nut “

103.6.Examble về định danh “id”

<html>

<head>

  <style>

    #nut {

      background: turquoise;

      color: white;

      padding: 10px 20px;

      border: none;

      cursor: pointer;

    }

    /\* khi rê chuột \*/

    #nut:hover {

      background: red;

    }

    /\* khi nhấn giữ \*/

    #nut:active {

      background: violet;

    }

  </style>

<script>

    function batLed() {

      alert("ĐÃ CLICK");

    }

  </script>

</head>

<body>

  <h1> EXAMBLE VỀ ĐỊNH DANH "id" </h1>

  <button id="nut" onclick="batLed()">BẬT</button>

  <!-- nguyên khói thẻ script đều có thể nằm được ở phần body vì nó không thuộc loại thẻ hiện thị.

        -Thẻ không hiện thị như logic/ xử lý.

        \*EXAMBLE

  <script>

       function batLed() {

    alert("ĐÃ CLICK");

  }

</script>   -->

</body>

</html>

=== === === === === === END OF id=== === === === === ===

=== === === === === THUỘC TÍNH: class === === === === ===

104. Thuộc tính class

104.1. Bản chất(dùng để làm gì)

-Dùng để GOM NHIỀU thẻ HTML vào cùng 1 nhóm.

-Chủ yếu dùng cho CSS.

-Có thể dùng cho JavaScript(học sau).

104.2. Phạm vi áp dụng(Global / Event / ARIA / Tag - specific) -

  Global attribute

- Dùng được cho hầu hết các thẻ HTML(div, button, p, span, input, img, ...)

104.3. Cú pháp

  <tag class = "ten-class" > ... < /tag> -

  Có thể dùng 1 hoặc nhiều class

104.4. Giá trị thường dùng

-Chuỗi(string)

-Không dấu

-Không khoảng trắng trong tên class

-Nhiều class thì cách nhau bằng dấu cách

104.5. Lưu ý / lỗi hay gặp

-class ĐƯỢC PHÉP trùng nhau

-Một thẻ có thể có nhiều class

-Không dùng dấu.trong HTML(dấu.chỉ dùng trong CSS)

\*Định nghĩa phần tử:

--1 phần tử bao gồm. ví dụ 1 nút bấm trên màn hình đó là 1 phần tử chỉ dùng "id"

-- 2 phần tử là số nhiều ví dụ  phần tử nút bấm và và phần tử nhập

liệu đều hiện thị trên màn hình và đó là 2 phần tử là số nhiều dung "class"

104.6. Examble(ví dụ)

<html>

<head>

  <style>

    .nut {

      background: turquoise;

      color: white;

      padding: 10px 20px;

      border: none;

      cursor: pointer;

    }

    /\* khi rê chuột \*/

    .nut:hover {

      background: red;

    }

    /\* khi nhấn giữ \*/

    .nut:active {

      background: violet;

    }

  </style>

  <script>

    function batLed() {

      alert("ĐÃ CLICK");

    }

  </script></head>

<body>

  <h1>EXAMBLE VỀ ĐỊNH DANH "class"</h1>

  <button class="nut" onclick="batLed()">BẬT</button>

  <button class="nut" onclick="batLed()">TẮT</button>

 </body>

</html>

=== === === === === === END OF class === === === === === === =

=== === == = === === THUỘC TÍNH: style === === === === ===

105. Thuộc tính style

105.1. Bản chất(dùng để làm gì)

-Dùng để GÁN CSS TRỰC TIẾP cho thẻ HTML

-Viết CSS ngay trong thẻ

-Áp dụng CHỈ cho thẻ đó

105.2. Phạm vi áp dụng(Global / Event / ARIA / Tag - specific) -

Global attribute

-Dùng được cho hầu hết các thẻ HTML(div, button, p, span, input, img, ...)

105.3. Cú pháp

<tag style="thuộc-tính-css: giá-trị;"> ... < /tag>

    105.4. giá-trị thường dùng

    -Các thuộc tính CSS(color, background, padding, margin, font - size, ...)

    105.5. Lưu ý / lỗi hay gặp

    -CSS trong style PHẢI đúng cú pháp

    -Mỗi thuộc tính CSS kết thúc bằng dấu;

    -Không nên lạm dụng(code khó quản lý)

    -style ƯU TIÊN CAO hơn CSS ngoài

    105.6. Examble(ví dụ)

    <button style="background: blue; color: white;"> BẬT </button>

    <!--NGƯỜI TA HẦU NHƯ KHÔNG DÙNG style TRONG THẺ CHỦ YẾU DÙNG CSS BÊN NGOÀI(hoặc trong <style> )+class + id

      105.7.Examble thuộc tính style ngoài thẻ

      <!DOCTYPE html>

      <html lang="vi">

      <head>

        <meta charset="UTF-8">

        <title>Ví dụ CSS thuộc tính Style</title>

        <style>

          /\* STYLE CƠ BẢN CHO NÚT \*/

          .nut {

            background: #1e90ff;

            color: white;

            padding: 12px 24px;

            border: none;

            border-radius: 6px;

            cursor: pointer;

            font-size: 16px;

            transition: 0.2s;

            /\* mượt khi đổi màu \*/

          }

          /\* RÊ CHUỘT \*/

          .nut:hover {

            background: #0066cc;

          }

          /\* NHẤN GIỮ \*/

          .nut:active {

            background: #003f7f;

            transform: scale(0.7);

            /\* cảm giác “nhún” \*/

          }

        </style>

      </head>

      <body>

        <h2>Ví dụ STYLE với class</h2>

        <button class="nut" onclick="guilenh1()">LED </button>

        <button class="nut" onclick="guilenh2()">TẮT</button>

        <script>

          function guilenh1() {

            alert("Đã Bật :LED ");

          }

          function guilenh2() {

            alert("Đã Tắt :LED ");

          }

        </script>

      </body>

      </html>

=== === === === === === END OF style=== === === === === === =

=== === === === === THUỘC TÍNH: title === === === === ===

106. Thuộc tính title

106.1.Bản chất(dùng để làm gì)

-Dùng để HIỂN THỊ CHÚ THÍCH khi rê chuột lên thẻ HTML

-Giúp giải thích nhanh chức năng của phần tử

106.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific)

-Global attribute

-Dùng được cho hầu hết các thẻ HTML(div, button, p, span, input, img, a, ...)

106.3.Cú pháp

<tag title="nội dung chú thích"> ... < /tag>

    106.4.Giá trị thường dùng

    -Chuỗi(string) bao gồm cả " " và ' '

    - Có thể có dấu và khoảng trắng

    106.5.Lưu ý / lỗi hay gặp

    -Chỉ hiện khi RÊ CHUỘT(hover)

    -Trên thiết bị cảm ứng(điện thoại) thường KHÔNG hiện

    -Không style trực tiếp tooltip bằng CSS thường

    106.6.Examble về title

    <html>

    <head>

      <style>

        .nut {

          background: turquoise;

          color: white;

          padding: 10px 20px;

          border: none;

          cursor: pointer;

        }

        /\* khi rê chuột \*/

        .nut:hover {

          background: red;

        }

        /\* khi nhấn giữ \*/

        .nut:active {

          background: violet;

        }

      </style>

      <script>

        function batled1() {   <!-- phần logic này nằm ở body hay head đều được không hiện ra màn hình -->

          alert("ĐÃ BẬT :LED");

        }

        function batled2() {

          alert("ĐÃ TẮT  :LED");

        }

      </script>

    </head>

    <body>

      <h1>EXAMBLE VỀ THUỘC TÍNH "title"</h1>

      <!--

    Khi rê chuột vào nút sẽ hiện chú thích

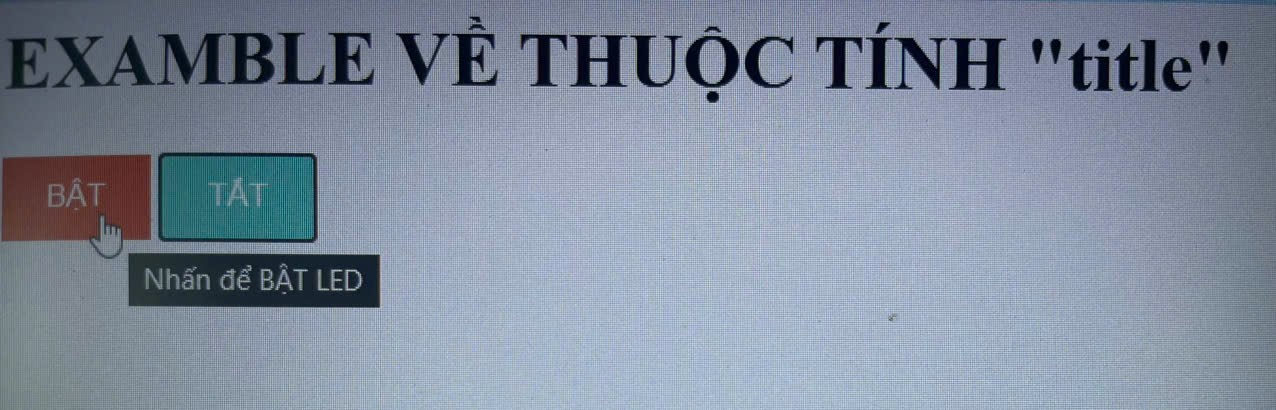
  -->

      <button class="nut" title="Nhấn để BẬT LED" onclick="batled1()"> BẬT </button>

      <button class="nut" title="Nhấn để TẮT LED" onclick="batled2()"> TẮT </button>

    </body>

</html>



=== === === == = === === END OF title=== === === === === === =

=== === === === === THUỘC TÍNH: hidden === === === === ===

107. Thuộc tính hidden

107.1.Bản chất(dùng để làm gì)

-Dùng để ẨN HOÀN TOÀN một thẻ HTML khỏi giao diện

-Thẻ vẫn tồn tại trong DOM

-Không hiển thị và KHÔNG chiếm không gian

-Thường dùng để ẨN / HIỆN bằng JavaScript

107.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific)

-Global attribute

-Dùng được cho hầu hết các thẻ HTML(div, button, p, span, input, img, section, ...)

107.3.Cú pháp

<tag hidden> ... < /tag>

    HOẶC

    <tag hidden="hidden"> ... < /tag>

        107.4.Giá trị thường dùng

        -Không cần giá trị

        -Chỉ cần CÓ thuộc tính hidden là thẻ bị ẩn

        107.5.Lưu ý / lỗi hay gặp

        -hidden KHÔNG phải là CSS

        -Không cần hidden = "true"

        hoặc hidden = "false"

        -hidden khác với display: none trong CSS

        -Khi bỏ hidden→ thẻ hiện lại

      107.6.Examble thuộc tính hidden

        <!DOCTYPE html>

        <html>

        <head>

          <style>

            .nut {

              background: turquoise;

              color: white;

              padding: 10px 20px;

              border: none;

              cursor: pointer;

              margin: 5px;

            }

            .nut:hover {

              background: red;

            }

            .nut:active {

              background: violet;

            }

          </style>

          <script>

            /\*

              Muốn lấy (hoặc gán) thuộc tính hidden của thẻ → PHẢI dùng dấu chấm trước hidden như :  nutphai.hidden

              nutphai là biến tham chiếu tới thẻ HTML

              hidden là thuộc tính (property) của thẻ

              \*/

            function anHien() {

              nutphai.hidden = !nutphai.hidden;

            }

            /\*

            - JS luôn TÍNH vế phải trước

            Trạng thái ban đầu:

            → nutphai.hidden == false  (nút đang HIỆN)

             \*khi ấy js kiểm tra vế phải nó  so sáng "trạng thái ban đầu" và "vế phải" 2 thằng này nếu trả cùng 1 kết quả sẽ là true và ngược lại\*

            JavaScript tính vế phải:

            → !nutphai.hidden = true

            →→ true (1)

            (1) gán sang (2)

            Gán sang vế trái :  nutphai.hidden (2)

            Kết quả

            → nutphai.hidden = true

            Trình duyệt thấy:

            → hidden = true

            → tự động ẨN phần tử

            \*/

            function batLed() {

              alert("CLICK NÚT BÊN PHẢI");

            }

          </script>

        </head>

        <body>

          <h1>EXAMBLE hidden + title</h1>

          <!-- NÚT TRÁI: điều khiển ẨN / HIỆN -->

          <button class="nut" title="Ẩn / Hiện nút bên phải" onclick="anHien()">ẨN / HIỆN</button>

          <!-- NÚT PHẢI: bị điều khiển -->

          <button class="nut" id="nutphai" title="Nút này sẽ bị ẩn / hiện" onclick="batLed()">LED</button>

          <!--NẾU MUẤN TRẠNG THÁI BAN ĐẦU ẨN THÌ DÙNG CODE NÀY

  <button hidden class="nut" id="nutphai" title="Nút này sẽ bị ẩn / hiện" onclick="batLed()">LED</button>

 -->

        </body>

        </html>

=== === === === === === END OF hidden=== === === === === === =

=== === === === === THUỘC TÍNH: tabindex === === === === ===

108. Thuộc tính tabindex

108.1.Bản chất(dùng để làm gì)

-Dùng để QUY ĐỊNH THỨ TỰ focus khi bấm phím TAB

-Điều khiển luồng di chuyển bàn phím trong giao diện

-Rất quan trọng với:

+Điều khiển không dùng chuột

+Giao diện công nghiệp / kiosk

+Acessibility(trợ năng)

-Cho phép một thẻ KHÔNG MẶC ĐỊNH focus vẫn có thể focus được

108.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific)

-Global attribute

-Dùng được cho hầu hết các thẻ HTML:

div, button, input, a, span, section, img, ...

108.3.Cú pháp

<tag tabindex="giá\_trị"> ... </tag>

108.4.Giá trị thường dùng

tabindex = "0"

+Theo thứ tự tự nhiên của HTML

+NÊN DÙNG NHẤT

-tabindex = "1", "2", "3", ...

+Ép thứ tự Tab theo số

+Số nhỏ focus trước

+KHÔNG khuyến khích dùng nhiều

-tabindex = "-1"

+KHÔNG focus bằng phím Tab

+Nhưng vẫn focus được bằng JavaScript

108.5.Lưu ý / lỗi hay gặp

-Không lạm dụng tabindex > 0(1, 2, 3...)

-tabindex KHÔNG phải CSS

-tabindex chỉ ảnh hưởng đến focus, KHÔNG ảnh hưởng hiển thị

-tabindex khác với disabled

-tabindex = "-1"

vẫn dùng được với element.focus()

108.6.Example thuộc tính tabindex điều khiển bằng tab và phím số

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP CONTROL – TAB + NUMBER KEY</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      background: #f5f5f5;

      padding: 30px;

    }

    h1 {

      text-align: center;

    }

    .panel {

      max-width: 400px;

      margin: auto;

      background: white;

      padding: 20px;

      border-radius: 8px;

    }

    button {

      width: 100%;

      padding: 15px;

      margin: 8px 0;

      font-size: 18px;

      cursor: pointer;

    }

    button:focus {

      outline: 3px solid dodgerblue;

    }

    .log {

      margin-top: 15px;

      padding: 10px;

      background: #eee;

    }

  </style>

</head>

<body>

  <h1>ESP CONTROL PANEL</h1>

  <div class="panel">

    <!-- TAB THEO THỨ TỰ -->

    <button id="ledOn" tabindex="1">1️⃣ BẬT LED</button>

    <button id="ledOff" tabindex="2">2️⃣ TẮT LED</button>

    <button id="motorOn" tabindex="3">3️⃣ BẬT MOTOR</button>

    <button id="motorOff" tabindex="4">4️⃣ TẮT MOTOR</button>

    <!-- RESET để cuối + KHÔNG TAB -->

    <button id="reset" tabindex="-1">❌ RESET ESP</button>

    <div class="log" id="log">Trạng thái: Chưa có lệnh</div>

  </div>

  <script>

    const log = document.getElementById("log");

    function send(cmd) {

      log.innerText = "Đã gửi lệnh: " + cmd;

      /\*

       Sau này gắn ESP thì dùng:

      fetch("http://ip-esp/" + cmd);

      hoặc WebSocket.send(cmd);

      \*/

    }

    // CLICK CHUỘT / ENTER / SPACE

    ledOn.onclick = () => send("LED\_ON");

    ledOff.onclick = () => send("LED\_OFF");

    motorOn.onclick = () => send("MOTOR\_ON");

    motorOff.onclick = () => send("MOTOR\_OFF");

    reset.onclick = () => send("RESET");

    // BẤM PHÍM SỐ

    document.addEventListener("keydown", e => {

      if (e.key === "1") ledOn.click();

      if (e.key === "2") ledOff.click();

      if (e.key === "3") motorOn.click();

      if (e.key === "4") motorOff.click();

    });

  </script>

</body>

</html>

=== === === === === === END OF tabindex === === === === === ===

=== === === === === THUỘC TÍNH: accesskey === === === === ===

109. Thuộc tính accesskey

109.1.Bản chất(dùng để làm gì)

-Dùng để GÁN PHÍM TẮT cho phần tử HTML

-Cho phép kích hoạt phần tử bằng tổ hợp phím

-Hỗ trợ điều khiển KHÔNG cần chuột

-Hữu ích cho:

+Giao diện điều khiển ESP / UNO

+Kiosk

+Accessibility(trợ năng)

109.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific)

-Global attribute

-Dùng được cho hầu hết các thẻ HTML:

button, a, input, textarea, select, div, ...

109.3.Cú pháp

<tag accesskey="phím"> ... </tag>

109.4.Giá trị thường dùng -

accesskey = "1"

accesskey = "2"

accesskey = "a"

accesskey = "b"

accesskey = "l"

Giá trị chỉ là 1 ký tự(chữ hoặc số)

109.5.Cách bấm phím(RẤT QUAN TRỌNG)

-Windows/Linux(Chrome, Edge):

ALT + accesskey

Ví dụ:

accesskey = "1"→ALT + 1

109.6.Lưu ý / lỗi hay gặp

-KHÔNG bấm mỗi phím số đơn lẻ

-Phải dùng tổ hợp phím(ALT + key)

-Trình duyệt khác nhau có thể khác tổ hợp

-Dễ xung đột với phím tắt trình duyệt

-Không nên lạm dụng quá nhiều accesskey

109.7.Example thuộc tính accesskey

<!DOCTYPE html>

<html>

<head>

  <meta charset="UTF-8">

  <title>EXAMPLE accesskey</title>

  <style>

    body {

      font-family: Arial;

      padding: 20px;

    }

    .nut {

      background: turquoise;

      color: white;

      padding: 10px 20px;

      border: none;

      cursor: pointer;

      margin: 5px;

      padding: 12px 20px;

      margin: 8px 0;

      font-size: 16px;

    }

    .nut:hover {

      background: red;

    }

    .nut:active {

      background: violet;

    }

  </style>

  <script>

    function send(cmd) {

      alert("Đã gửi lệnh: " + cmd);

      /\*

        Gắn ESP sau:

        fetch("http://ip-esp/" + cmd);

      \*/

    }

  </script>

</head>

<body>

  <h1>EXAMPLE accesskey</h1>

  <!-- ALT + 1 -->

  <button class="nut" accesskey="1" onclick="send('LED\_ON')">BẬT LED (ALT + 1) </button>

  <!-- ALT + 2 -->

  <button class="nut" accesskey="2" onclick="send('LED\_OFF')">TẮT LED (ALT + 2) </button>

  <!-- ALT + 3 -->

  <button class="nut" accesskey="3" onclick="send('MOTOR\_ON')">BẬT MOTOR (ALT + 3)</button>

  <!-- ALT + 4 -->

  <button class="nut" accesskey="4" onclick="send('MOTOR\_OFF')">TẮT MOTOR (ALT + 4)</button>

  <p> Windows: ALT + số

    <br>

    macOS: CTRL + OPTION + số

  </p>

</body>

</html>

=== === === === === === END OF accesskey === === === === === === =

= === === === === THUỘC TÍNH: contenteditable === === === === ===

110. Thuộc tính contenteditable

110.1.Bản chất(dùng để làm gì)

-Dùng để CHO PHÉP NGƯỜI DÙNG SỬA NỘI DUNG TRỰC TIẾP trên giao diện web

-Không cần input, textarea

-Thẻ vẫn là HTML bình thường nhưng gõ sửa được

-Thường dùng cho:

+Ghi chú

+Đổi tên thiết bị

+Cấu hình nhanh trên web ESP

+CMS mini / bảng điều khiển

110.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific)

-Global attribute

-Dùng được cho hầu hết các thẻ HTML:

div, p, span, h1, h2, section, td, ...

110.3.Cú pháp

<tag contenteditable> ... </tag>

HOẶC

<tag contenteditable="true"> ... </tag>

110.4.Giá trị thường dùng

-true(mặc định khi có thuộc tính)

-false(tắt khả năng chỉnh sửa)

110.5.Lưu ý / lỗi hay gặp

-contenteditable KHÔNG phải input

-Không có validate dữ liệu

-Người dùng có thể xóa hết nội dung

-Nội dung sửa KHÔNG tự lưu(phải dùng JS)

-Không dùng cho dữ liệu quan trọng nếu không kiểm soát

110.6.Examble thuộc tính contenteditable

<!DOCTYPE html>

<html>

<body>

  <p id="deviceName" contenteditable="true"> ESP32 Phòng Khách </p>

  <button onclick="docTen()">Lấy tên</button>

  <script>

    function docTen() {

      alert(deviceName.value);

    }

  </script>

</body>

</html>

/\*

Trình duyệt lưu dư liệu người dùng value,innerText…

innerText : chỉ dùng cho thẻ có chữ bên trong

value : CHỈ dùng cho thẻ NHẬP LIỆU

\*/

/\*

Muốn lấy (hoặc gán) thuộc tính value của thẻ → PHẢI dùng dấu chấm trước value như deviceName.value

deviceName là biến tham chiếu tới thẻ HTML

value là thuộc tính (property) của thẻ

\*/

=== === === === === === END OF contenteditable === === === === === === =

=== === === === === THUỘC TÍNH: enterkeyhint === === === === ===

111. Thuộc tính enterkeyhint

111.1.Bản chất(dùng để làm gì)

-Dùng để THAY ĐỔI CHỮ HIỂN THỊ của phím Enter

trên BÀN PHÍM ẢO(điện thoại)

-Giúp người dùng biết:

+Nhập xong→ DONE

+Gửi dữ liệu→ SEND

+Đi tới→ GO

-Chỉ ảnh hưởng GIAO DIỆN, KHÔNG ảnh hưởng logic ESP

111.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific)

-Global attribute

-Thường dùng cho các thẻ NHẬP LIỆU:

input

textarea

111.3.Cú pháp

<input enterkeyhint="done"><textarea enterkeyhint="send"> </textarea>

111.4.Giá trị thường dùng

-enterkeyhint = "done"→

nhập xong

-enterkeyhint = "send"→

gửi dữ liệu

-enterkeyhint = "go"→

đi tới / truy cập

-enterkeyhint = "search"→

tìm kiếm

-enterkeyhint = "next"→

chuyển ô tiếp theo

111.5.Lưu ý / lỗi hay gặp

-CHỈ CÓ TÁC DỤNG trên ĐIỆN THOẠI

-KHÔNG có tác dụng trên máy tính bàn

-KHÔNG gửi dữ liệu, KHÔNG chạy JS

-Chỉ là GỢI Ý GIAO DIỆN(UX)

-Phải kết hợp JavaScript nếu muốn xử lý Enter

111.6.Ví dụ thuộc tính enterkeyhint

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>TEST enterkeyhint</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 30px;

    }

    .mau {

      width: 100%;

      padding: 14px;

      font-size: 16px;

      border-radius: 8px;

      border: 1px solid #30363d;

      background: #161b22;

      color: blue;

      /\*#e6edf3\*/

    }

  </style>

</head>

<body>

  <h2>TEST enterkeyhint</h2>

  <p>Nhập tên thiết bị ESP:</p>

  <input class="mau" type="text" placeholder="ESP32 Phòng Khách" enterkeyhint="done"

    onkeydown="if(event.key==='Enter') alert('ĐÃ NHẬP XONG')">

  <!--

 placeholder : chữ gợi ý

 enterkeyhint:  Dùng để THAY ĐỔI CHỮ HIỂN THỊ của phím Enter

 onkeydown   : phát hiện nhấn phím

 event.key   : là tên phím vừa nhấn có thể là tất cả các phím

 điều kiện   :

 QUY TẮC có 1 lệnh không cân { }

 if (điều\_kiện) lệnh;

\*Định nghĩa phần tử:

--1 phần tử bao gồm. ví dụ 1 nút bấm trên màn hình đó là 1 phần tử chỉ dùng "id"

-- 2 phần tử là số nhiều ví dụ  phần tử nút bấm và và phần tử nhập

liệu đều hiện thị trên màn hình và đó là 2 phần tử là số nhiều dung "class"

-->

  <p> 👉 MỞ TRÊN ĐIỆN THOẠI

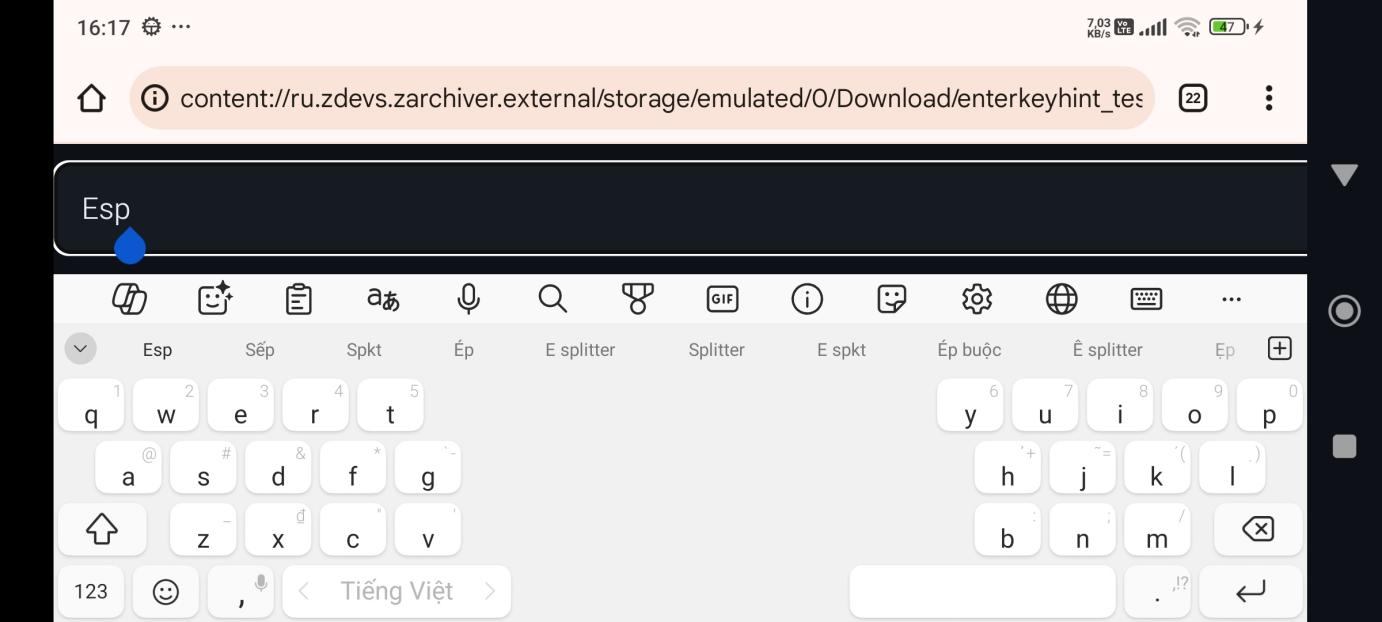
    <br>

    👉 Nhìn phím Enter trên bàn phím ảo

  </p>

</body>

</html>



=== === === = == END OF enterkeyhint === === === === ===

=== === === === === THUỘC TÍNH: placeholder và textarea === === === === ===

112. Thuộc tính placeholder và textarea

placeholder : ô nhập 1 dòng

textarea : ô nhập nhiều dòng

112.1.Bản chất(dùng để làm gì)

-Dùng để HIỂN THỊ CHỮ GỢI Ý trong ô nhập

-Giúp người dùng BIẾT NÊN NHẬP GÌ

-Chữ chỉ để nhìn, KHÔNG PHẢI dữ liệu thật

-Khi người dùng gõ→ chữ placeholder TỰ BIẾN MẤT

Thường dùng cho:

+Nhập tên thiết bị ESP

+Nhập WiFi / Password

+Nhập thông số cấu hình

+Form cài đặt nhanh trên web ESP

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

112.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific)

-Thuộc loại: Tag - specific attribute

-

CHỈ dùng cho các thẻ nhập liệu:

input

textarea

❌ Không dùng cho div, p, span, h1, ...

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

112.3.Cú pháp

<input placeholder="Nội dung gợi ý">

HOẶC

<textarea placeholder="Nội dung gợi ý"> < /textarea>

Note :

placeholder  :gợi ý 1 dòng

textarea    :gợi ý nhiều dòng

  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

112.4.Giá trị thường dùng

-Chuỗi văn bản(text)

  Ví dụ:

  "ESP32 Phòng Khách""Nhập mật khẩu WiFi""Nhập IP thiết bị"

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

112.5.Lưu ý / lỗi hay gặp

-placeholder KHÔNG PHẢI value

-Không được gửi lên server

-Không lưu dữ liệu

-Không dùng để làm dữ liệu mặc định

-Khi submit form: placeholder KHÔNG TỒN TẠI

  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -

112.6.Ví dụ minh họa(dùng cho ESP)

  <!DOCTYPE html>

<html>

<body>

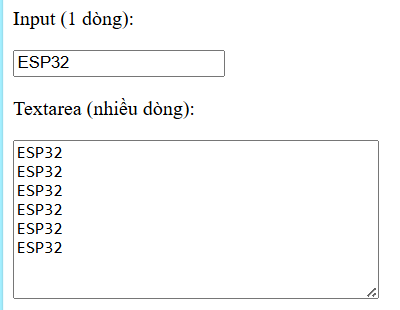
  <p>Input (1 dòng):</p>

  <input placeholder="Tên thiết bị ESP">

  <p>Textarea (nhiều dòng):</p>

  <textarea placeholder="Ghi chú cấu hình ESP"></textarea></body>

</html>



textarea : ô nhập nhiều dòng

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

112.7.So sánh nhanh placeholder vs value

placeholder:

-Chỉ để gợi ý

-Chữ mờ

-Biến mất khi gõ

value:

-Là dữ liệu thật

-Gửi được

-Không tự biến mất

112.8.Examble text 1 ô và text n ô “ placeholder và textarea”

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>TEST input & textarea</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 30px;

    }

    input,

    textarea {

      width: 100%;

      padding: 12px;

      margin-bottom: 16px;

      font-size: 16px;

      border-radius: 8px;

      border: 1px solid #30363d;

      background: #161b22;

      color: #e6edf3;

    }

    textarea {

      height: 120px;

    }

  </style>

</head>

<body>

  <h2>TEST input vs textarea</h2>

  <p> INPUT (chỉ 1 dòng):</p>

  <input placeholder="Gõ thử... Enter là xong">

  <p> TEXTAREA (nhiều dòng):</p>

  <textarea placeholder="Gõ nhiều dòng...

Enter để xuống dòng"></textarea>

</body>

</html>

=== === === === END OF placeholder === === === === ===

=== === === === === SỰ KIỆN: onkeydown === === === === ===

113.Thuộc tính sự kiện onkeydown

113.1.Bản chất(dùng để làm gì)

`onkeydown` là \*\* SỰ KIỆN(event) \*\* trong HTML

-Kích hoạt \*\* NGAY KHI người dùng NHẤN phím xuống \*\*

-Xảy ra \*\* TRƯỚC khi ký tự được nhập vào ô \*\*

Hiểu đơn giản:

\*\*VỪA ẤN PHÍM → onkeydown CHẠY NGAY \*\*

113.2.onkeydown dùng để làm gì

-Phát hiện thao tác bàn phím:

-Nhấn Enter

-Nhấn ESC

-Nhấn phím mũi tên

-Điều khiển giao diện \*\* KHÔNG cần chuột \*\*

-Rất hữu ích cho:

-Giao diện ESP

-Web điều khiển

-Form nhập liệu nhanh

-Thiết bị cảm ứng

113.3.Phạm vi áp dụng

-Thuộc nhóm: \*\* Event Attribute \*\*

-Dùng được cho nhiều thẻ HTML:

<input>

<textarea>

  <button>

  <div>

  <body>

 113.4.Cú pháp cơ bản

<tag onkeydown="lệnh"> </tag>

113.5.Examble

   <input onkeydown="alert('ĐÃ NHẤN PHÍM')">

113.6.Examble

    <input type="text"

  placeholder="Nhập tên ESP"

  onkeydown="alert('Bạn vừa nhấn phím')">

113.7.onkeydown khác gì onkeyup

onkeydown: chạy NGAY khi ẤN phím

onkeyup: chạy SAU khi THẢ phím

onkeydown phản ứng NHANH hơn

113.7.Lưu ý quan trọng

-onkeydown KHÔNG phải là thuộc tính hiển thị

-Nó chỉ dùng để bắt sự kiện

-Không tự lưu dữ liệu

-Không thay thế input, value, innerText

113.8.Examble thuộc tính sự kiện  onkeydown

   <!DOCTYPE html>

<html lang="vi">

<head>

<meta charset="UTF-8">

<title>Test onkeydown</title>

  <style>

    body{

      background:#0d1117;

      color:#e6edf3;

      font-family:Consolas, monospace;

      padding:30px;

    }

    input{

      width:100%;

      padding:14px;

      font-size:16px;

      background:#161b22;

      color:#e6edf3;

      border:1px solid #30363d;

      border-radius:8px;

    }

  </style></head>

<body>

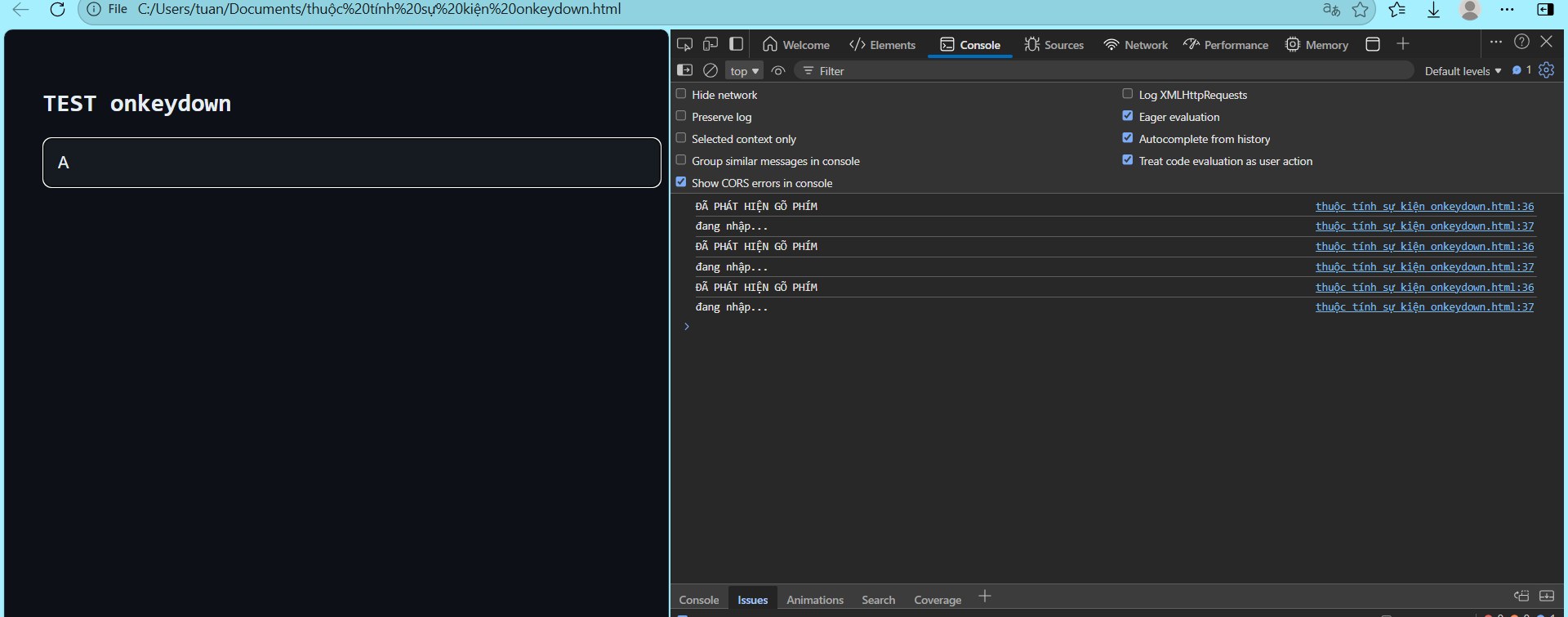
  <h2>TEST onkeydown</h2>

<input type="text" placeholder="Gõ thử bất kỳ phím nào..." onkeydown="console.log('ĐÃ PHÁT HIỆN GÕ PHÍM')

console.log('đang nhập...')">

</body>

</html>



=== === === === === THUỘC TÍNH: inert === === === === ===

114. Thuộc tính inert

114.1.Bản chất(dùng để làm gì)

-Khóa vùng nhập liệu

114.2.inert dùng để làm gì

Khoá giao diện khi:

Đang xử lý dữ liệu

Đang gửi lệnh cho ESP

Đang loading

Tắt tương tác tạm thời

Thay thế cho việc:

disable từng input

chặn từng button

114.3.Phạm vi áp dụng

Thuộc nhóm: Global Attribute

Dùng cho HẦU HẾT thẻ HTML:

div

section

form

nav

main

Áp dụng cho:

+Chính phần tử đó +

TOÀN BỘ phần tử con bên trong

114.4.Cú pháp tổng quát

Bật inert:

<tag inert>

  -Tắt inert:

  bỏ thuộc tính inert HOẶC

  inert = "false"(tuỳ trình duyệt)

  114.5.Examble

  <!DOCTYPE html>

  <html>

  <body>

    <button onclick="khoa()">KHÓA</button>

    <button onclick="mo()">MỞ</button>

    <div id="vung">

      <input placeholder="Gõ thử xem">

    </div>

    <script>

      function khoa() {

        vung.inert = true;

      }

      function mo() {

        vung.inert = false;

      }

    </script>

  </body>

  </html>

=== === === === END OF inert === === === === ===

  === === === THUỘC TÍNH: inputmode === === ===

  115. Thuộc tính inputmode

  115.1.Bản chất(dùng để làm gì)

  +HIỆN BÀN PHÍM PHÙ HỢP khi nhập

  115.2.inputmode dùng để làm gì

  ##Hiển thị bàn phím đúng loại:

  +số

  +chữ

  +email

  +URL

  115.3.Phạm vi áp dụng

  -Thuộc nhóm: Global Attribute

  -Dùng chủ yếu cho:

  +input

  +textarea

  -Có thể dùng cho thẻ khác(ít dùng)

  115.4.Cú pháp tổng quát

  <tag inputmode="giá\_trị">

    115.5.Các giá trị thường dùng

    text: bàn phím chữ(mặc định)

    numeric: bàn phím số(0– 9)

    decimal: số + dấu chấm

    tel: bàn phím số gọi điện

    email: bàn phím email(@)

    url: bàn phím URL(/ .)

    search: bàn phím tìm kiếm

    115.6.inputmode KHÁC gì type

    -inputmode:

    +Chỉ GỢI Ý bàn phím +

    Không ép dữ liệu +

    Không validation

    -type:

    +Ảnh hưởng logic form +

    Có kiểm tra dữ liệu +

    Có thể bị trình duyệt chặn

    115.7.Ứng dụng thực tế cho ESP

    -Nhập:

    +IP thiết bị→ numeric / decimal

    +Port→ numeric

    +Thời gian delay→ numeric

    +Giá trị cảm biến→ decimal

    -Giao diện ESP điều khiển bằng điện thoại

    115.8.Examble

    <!DOCTYPE html>

    <html lang="vi">

    <head>

      <meta charset="UTF-8">

      <title>Test inputmode</title>

      <style>

        body {

          font-family: monospace;

          padding: 20px;

        }

        input {

          display: block;

          margin-bottom: 12px;

          padding: 10px;

          width: 100%;

        }

      </style>

    </head>

    <body>

      <p>TEXT (mặc định):</p>

      <input inputmode="text" placeholder="Bàn phím chữ">

      <p>NUMERIC (số):</p>

      <input inputmode="numeric" placeholder="Bàn phím số">

      <p>DECIMAL (số + dấu .):</p>

      <input inputmode="decimal" placeholder="Số thập phân">

      <p>EMAIL:</p>

      <input inputmode="email" placeholder="email@example.com">

      <p>URL:</p>

      <input inputmode="url" placeholder="https://esp.local">

    </body>

    </html>

=== === === === END OF inputmode === === === === ===

=== === === === THUỘC TÍNH: data-\* === === === ===

116. Thuộc tính data-\*

-Kết nối giao diện wed với ESP

116.1.Bản chất(dùng để làm gì)

-Dùng để GẮN DỮ LIỆU TÙY CHỈNH vào thẻ HTML

-Dữ liệu này:

+KHÔNG hiển thị ra giao diện

+KHÔNG ảnh hưởng CSS

+Nhưng JavaScript ĐỌC ĐƯỢC

-Thường gọi là:

+Custom Data Attribute

116.2.data-\* dùng để làm gì(trong ESP)

-Lưu thông tin cho JavaScript xử lý

-Ánh xạ giao diện với phần cứng ESP

-Tránh viết nhiều if / biến rời rạc

-Thường dùng cho:

+GPIO(chân ESP)

+ID thiết bị

+Trạng thái ON / OFF

+Giá trị cấu hình

116.3.Phạm vi áp dụng

-Thuộc loại: Global Attribute

-DÙNG ĐƯỢC cho MỌI THẺ HTML:

div

button

input

span

textarea

v.v...

116.4.Cú pháp

Cú pháp tổng quát:

<tag data - ten="giatri">

  < /tag>

    -Quy tắc:

    +Luôn bắt đầu bằng data -

    +Tên viết thường

    +Không dấu, không khoảng trắng

    116.5.Các data-\* thường dùng trong ESP

    Data - id +

    Định danh phần tử / thiết bị

    data - value +

    Lưu giá trị cấu hình

    data - state +

    Lưu trạng thái(on / off / 0 / 1)

    -data - led +

    Gắn số LED hoặc GPIO

    116.6.Truy cập data-\* bằng JavaScript

    -JavaScript đọc data-\* thông qua dataset

    -Ví dụ:

    el.dataset.id

    el.dataset.value

    el.dataset.state

    el.dataset.led

    -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --116.7.Lưu ý quan trọng -

    data-\* KHÔNG phải:

    +Thuộc tính hiển thị

    +Thuộc tính sự kiện

    +Thuộc tính bảo mật

    data-\*chỉ dùng để:

    +LƯU DỮ LIỆU

    +PHỤC VỤ LOGIC JS

    116.8.Exambles

    < button data - led="on"> BẬT ĐÈN < /button>

        < button data - led="off"> TẮT ĐÈN < /button>

            < script>

              document.querySelectorAll("button").forEach(btn => {

                btn.onclick = () => {

                  let lenh = btn.dataset.led; // đọc data-led

                  console.log("Gửi lệnh:", lenh);

                  // ví dụ gửi request tới ESP

                  // fetch("http://esp-ip/control?led=" + lenh);

                };

              }); </script>

              === === === END OF data - \* === === === ===

              = === === THUỘC TÍNH: data-id === === === ===

              117. Thuộc tính data-id

              117.1.Bản chất(data - id là gì)

              -data-id là một thuộc tính CON của data - \*

              -Dùng để gán:

              +SỐ THỨ TỰ(STT)

              +MÃ ĐỊNH DANH

              +ID THIẾT BỊ

              cho một phần tử HTML

              data-id = "tên gọi logic"cho phần tử

              117.2.data-id dùng để làm gì(trong ESP)

              -Xác định:

              +LỆNH này thuộc về THIẾT BỊ NÀO

              +GPIO / MODULE / CHỨC NĂNG nào

              -Giảm phụ thuộc vào:

              +ID HTML

              +biến JS rời rạc

              +nhiều hàm if -else cứng

              -Rất phù hợp khi:

              +1 giao diện điều khiển NHIỀU THIẾT BỊ

              +ESP nhận lệnh theo MÃ / STT

              117.3.Cú pháp sử dụng data - id

              Cú pháp tổng quát:

              <tag data-id="giatri">

                < /tag>

                  -Quy tắc:

                  +Luôn bắt đầu bằng data -

                  +Tên thuộc tính viết thường

                  +Giá trị có thể là:

                  -số

                  -chuỗi

                  -mã định danh

                  Ví dụ giá trị:

                  data-id = "117"

                  data-id = "led1"

                  data-id = "motor\_A"

                  117.4.data-id thường dùng để lưu gì

                  SỐ GPIO

                  MÃ THIẾT BỊ

                  ID LOGIC

                  STT MODULE

                  MÃ GỬI QUA MQTT / HTTP / WS

                  117.5.Truy cập data-id bằng JavaScript

                  -JavaScript đọc data-id thông qua dataset

                  Cú pháp:##

                  el.dataset.id

                  -Lưu ý:

                  data-id → dataset.id

                  dấu "-" bị loại bỏ và viết theo camelCase

                  117.6.Vai trò của data-id trong hệ thống ESP

                  -data-id KHÔNG điều khiển phần cứng trực tiếp

                  -data-id chỉ:

                  +LƯU DỮ LIỆU

                  +PHỤC VỤ LOGIC JS

                  +GỬI MÃ LỆNH SANG ESP

                  ESP:

                  -KHÔNG biết data-id tồn tại

                  ESP chỉ biết:

                  +dữ liệu nhận được

                  +và xử lý theo dữ liệu đó

                  117.8.Lưu ý quan trọng

                  -data-id KHÔNG phải:

                  +ID HTML

                  +thuộc tính hiển thị

                  +thuộc tính bảo mật

                  +

                  thuộc tính sự kiện

                  -data - id CHỈ dùng để:

                  +GẮN DỮ LIỆU +

                  QUẢN LÝ LOGIC +

                  LIÊN KẾT GIAO DIỆN↔ PHẦN CỨNG

                  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

                  === === === === THUỘC TÍNH: data-value === === === ===

                  118. Thuộc tính data-value

                  data - value rất thường dùng cho servo

                  118.1.Bản chất(data-value là gì)

                  data-value là một thuộc tính CON của data - \*

                  -Dùng để gán:

                  +GIÁ TRỊ

                  +THÔNG SỐ

                  +CẤU HÌNH

                  cho một phần tử HTML

                  -Giá trị này:

                  +KHÔNG hiển thị ra giao diện

                  +KHÔNG ảnh hưởng CSS

                  +JavaScript ĐỌC ĐƯỢC

                  => Có thể hiểu:

                  data-value = "giá trị logic"của phần tử

                  118.2.data -value dùng để làm gì(trong ESP)

                  -Giúp:

                  +Không hard-code giá trị trong JS

                  +Thay đổi cấu hình mà KHÔNG sửa code JS

                  118.3.data-value thường biểu diễn gì

                  -GIÁ TRỊ SỐ:

                  +tốc độ

                  +góc quay

                  +mức công suất

                  -GIÁ TRỊ TRẠNG THÁI:

                  0 / 1

                  true / false

                  GIÁ TRỊ CẤU HÌNH:

                  +ngưỡng

                  +thời gian

                  +mode

                  118.4.Cú pháp sử dụng data-value

                  Cú pháp tổng quát:##

                  <tag data-value="giatri"> </tag>

                  -Quy tắc:

                  +Luôn bắt đầu bằng data -

                  +Tên viết thường, không dấu

                  +Giá trị có thể là:

                  số

                  chuỗi

                  mã cấu hình

                  Ví dụ giá trị:

                  data - value = "0"

                  data - value = "1"

                  data - value = "45"

                  data - value = "120"

                  data - value = "auto"

                  118.5.data - value thường dùng kết hợp với

                  data - id(xác định THIẾT BỊ)

                  data - led(xác định GPIO)

                  data - state(xác định TRẠNG THÁI)

                  => data-id + data-value = lệnh hoàn chỉnh

                  118.6.Truy cập data - value bằng JavaScript

                  -JavaScript đọc data - value thông qua dataset

                  Cú pháp:

                  el.dataset.value

                  -Lưu ý:

                  data-value → dataset.value

                  - Giá trị đọc được luôn là KIỂU CHUỖI(string)Nếu cần số → phải CHUYỂN KIỂU

                  118.7.Vai trò của data - value trong hệ thống ESP -

                  data - value KHÔNG điều khiển phần cứng trực tiếp

                  data - value chỉ:

                  CUNG CẤP GIÁ TRỊ

                  PHỤC VỤ LOGIC JS

                  GỬI THAM SỐ SANG ESP

                  ESP:

                  -Chỉ nhận GIÁ TRỊ

                  -Không quan tâm giá trị đó đến từ data-value

                  118.8.Lưu ý quan trọng -

                  data-value KHÔNG phải:

                  +giá trị hiển thị

                  +thuộc tính input value

                  +dữ liệu bảo mật

                  data-value CHỈ dùng để:

                  +LƯU THAM SỐ

                  +CẤU HÌNH LOGIC

                  +ĐIỀU KHIỂN THIẾT BỊ GIÁN TIẾP

                  118.9.Examble

                  <button data-id="servo1" data-value="30">Servo 30°</button>

                  <button data-id="servo1" data-value="90">Servo 90°</button>

                  <button data-id="servo1" data-value="150">Servo 150°</button>

                  === === === === THUỘC TÍNH:data-state === === === ===

                  119. Thuộc tính data-state

                  2 trạng thái 0/1

                  119.1.Bản chất(data-state là gì)

                  data-state là một thuộc tính CON của data - \*

                  -Dùng để gán:

                  TRẠNG THÁI

                  TÌNH TRẠNG HOẠT ĐỘNG cho phần tử HTML

                  -Trạng thái thường là:

                  ON / OFF

                  1 / 0

                  true / false

                  -Giá trị của data-state:

                  +KHÔNG hiển thị

                  +KHÔNG ảnh hưởng CSS

                  +JavaScript ĐỌC ĐƯỢC

                  => Có thể hiểu:

                  data-state = trạng thái logic của thiết bị

                  119.2.data-state dùng để làm gì(trong ESP) -

                  Lưu trạng thái bật / tắt trước khi gửi đi

                  -Giúp JavaScript:

                  +Biết thiết bị đang ở trạng thái nào

                  +Đổi trạng thái mà không cần nhiều biến

                  -Rất hay dùng cho:

                  +LED

                  +Relay

                  +Công tắc

                  +Chế độ hệ thống

                  119.3.data - state thường biểu diễn gì

                  TRẠNG THÁI ĐỊNH TÍNH:

                  ON / OFF

                  ENABLE / DISABLE

                  TRẠNG THÁI SỐ:

                  1 / 0

                  => data - state phù hợp với THIẾT BỊ 2 TRẠNG THÁI

                  119.4.Cú pháp sử dụng data - state

                  Cú pháp tổng quát:

                  <tag data-state="giatri"> </tag>

                  Ví dụ giá trị:

                  data - state = "on"

                  data - state = "off"

                  data - state = "1"

                  data - state = "0"

                  119.5.Truy cập data-state bằng JavaScript -

                  JavaScript đọc data-state thông qua dataset

                  Cú pháp:

                  el.dataset.state

                  -Lưu ý:

                  Giá trị đọc được luôn là CHUỖI Cần quy ước rõ ON / OFF hoặc 1 / 0

                  119.6.Vai trò của data - state trong hệ ESP

                  -data - state KHÔNG bật / tắt phần cứng trực tiếp

                  -data - state chỉ:

                  +LƯU TRẠNG THÁI

                  +PHỤC VỤ LOGIC JS

                  +GỬI TRẠNG THÁI SANG ESP

                  119.7.Lưu ý quan trọng

                  data-state KHÔNG phù hợp cho:

                  +servo

                  +giá trị liên tục

                  +điều khiển chính xác

                  => data-state dùng cho ON / OFF là TỐT NHẤT

            -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

                  === === === THUỘC TÍNH: data - led === === === ===

                  120.Thuộc tính data-led

                  120.1.Bản chất(data - led là gì)-

                  data - led là một thuộc tính CON của data - \*

                  -Dùng để gán:

                  +SỐ LED

                  +SỐ GPIO

                  +CHÂN ĐIỀU KHIỂN ESP

                  cho phần tử HTML

                 => data-led = “thiết bị vật lý nào”

                  120.2.data - led dùng để làm gì(trong ESP) -

                  Xác định:

                  +Lệnh này tác động lên LED nào

                  +GPIO nào của ESP

                  -Dùng để:

                  +Điều khiển NHIỀU LED +

                  Không hard - code GPIO trong JS

                  120.3.data-led thường lưu gì

                  SỐ GPIO:

                  2

                  4

                  5

                  MÃ LED:

                  led1

                  led2

                  => Ý nghĩa DO NGƯỜI LẬP TRÌNH ĐỊNH NGHĨA

                  120.4.Cú pháp sử dụng data-led

                  Cú pháp tổng quát:

                  <tag data - led="giatri"> </tag>

                  Ví dụ:

                  data - led = "2"

                  data - led = "4"

                  data - led = "led1"

                  120.5.Truy cập data-led bằng JavaScript

                  -JavaScript đọc data-led thông qua dataset

                  Cú pháp:

                  el.dataset.led

                  120.6.data-led thường dùng kết hợp với

                  data - state(bật / tắt)

                  data - id(định danh thiết bị)

                  data - value(giá trị điều khiển)

                  => data-led + data-state = điều khiển LED hoàn chỉnh

                120.7.Vai trò trong hệ thống ESP

                  data-led KHÔNG phải GPIO thật

                  data-led chỉ là:

                  +DỮ LIỆU MÔ TẢ

                  +GỬI SANG ESP

                  ESP:

                  -Nhận số GPIO

                  -Ánh xạ sang chân IO thật

                  120.8.Lưu ý quan trọng

                  -data-led KHÔNG phải:

                  +pinMode

                  +digitalWrite

                  +cấu hình phần cứng

                  -data - led CHỈ dùng để:

                  +ÁNH XẠ GIAO DIỆN ↔ GPIO

                  +GIẢM HARD - CODE

                  === === === === === THUỘC TÍNH: aria-activedescendant === === === === ===

                  121. Thuộc tính aria-activedescendant

                  121.1.Bản chất(dùng để làm gì)

                  -Chỉ dùng bằng phím lên xuống or enter trên màn hình LCD

                  121.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific) -

                  -ARIA attribute

                  -Áp dụng cho các phần tử cha(container)

                  -Thường dùng trong widget phức tạp(listbox, menu, autocomplete)

                  121.3.Cú pháp

                  <tag aria-activedescendant="id\_con">

                    < /tag>

                      121.4.Giá trị thường dùng

                      -Giá trị là id của phần tử con

                      -Phải TRÙNG CHÍNH XÁC với id tồn tại

                      -Không có khoảng trắng

                      121.5.Khi nào cần dùng

                      -Danh sách chọn bằng bàn phím

                      -Menu điều hướng bằng phím mũi tên

                      -Autocomplete / dropdown custom

                      121.6.Lưu ý / lỗi hay gặp

                      -Phần tử được trỏ tới PHẢI có id

                      -Không dùng nếu focus đã nằm trực tiếp trên phần tử con

                      -Dùng sai→ screen reader đọc sai mục đang chọn

                      121.7.Ví dụ

                      <ul role="listbox" aria - activedescendant="item2">

                        <li id="item1"> HTML < /li>

                        <li id="item2"> CSS < /li>

                        <li id="item3"> JavaScript < /li>

                      </ul>

                      → Mục đang active là "CSS"

                      \*Ghi nhớ nhanh:

                      -id→ định danh duy nhất

                      -aria - activedescendant → trỏ tới id đang được chọn

                      -focus ở CHA, active ở CON

                      \*Ký hiệu liên quan:

                      ARIA = Accessible Rich Internet Applications

                      active≠ focus

                      focus: đang đứng chuột / bàn phím

                      active: mục đang được chọn

                      122.8.Examble

                      <!DOCTYPE html>

                      <html lang="vi">

                      <head>

                        <meta charset="UTF-8">

                        <title>aria-activedescendant + phím ↑ ↓</title>

                        <style>

                          body {

                            font-family: Arial;

                            background: #111;

                            color: #fff;

                            padding: 30px;

                          }

                          input {

                            width: 200px;

                            padding: 6px;

                          }

                          ul {

                            margin-top: 6px;

                            padding-left: 0;

                            width: 200px;

                            border: 1px solid #444;

                          }

                          li {

                            list-style: none;

                            padding: 6px 10px;

                          }

                          li.active {

                            background: #3fb950;

                            color: #000;

                          }

                        </style>

                      </head>

                      <body>

                        <h3>aria-activedescendant (↑ ↓ + Enter)</h3>

                        <!-- PHẦN TỬ CHA GIỮ FOCUS -->

                        <input id="input" type="text" aria-activedescendant="item1" readonly />

                        <ul role="listbox">

                          <li id="item1" class="active">HTML</li>

                          <li id="item2">CSS</li>

                          <li id="item3">JavaScript</li>

                        </ul>

                        <script>

                          const input = document.getElementById("input");

                          const items = document.querySelectorAll("li");

                          let index = 0;

                          input.addEventListener("keydown", function (e) {

                            if (e.key === "ArrowDown") {

                              e.preventDefault();

                              move(1);

                            }

                            if (e.key === "ArrowUp") {

                              e.preventDefault();

                              move(-1);

                           }

                            if (e.key === "Enter") {

                              e.preventDefault();

                              input.value = items[index].textContent;

                              alert("Bạn đã chọn: " + items[index].textContent);

                            }

                          });

                          function move(step) {

                            items[index].classList.remove("active");

                            index += step;

                            if (index < 0) index = items.length - 1;

                            if (index >= items.length) index = 0;

                            items[index].classList.add("active");

                            // 🔥 CẬP NHẬT aria-activedescendant

                            input.setAttribute(

                              "aria-activedescendant",

                              items[index].id

                            );

                          }

                        </script>

                      </body>

                      </html>

=== === === === === THUỘC TÍNH: aria - busy === === === === ===

122. Thuộc tính aria-busy

122.1.Bản chất(dùng để làm gì)

-Thông báo cho trình đọc màn hình(screen reader) biết:

-Phần tử / khu vực đang ĐƯỢC XỬ LÝ(loading, đang cập nhật)

-Ngăn screen reader đọc nội dung khi nó CHƯA ỔN ĐỊNH

122.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific)

-ARIA attribute

-Áp dụng cho các thẻ có nội dung động(div, section, main, form, ul...)

122.3.Cú pháp

<tag aria - busy="true">

  < /tag>

    <tag aria - busy="false">

      < /tag>

        122.4.Giá trị thường dùng

        true: đang bận / đang xử lý / đang loadfalse: đã xong / nội dung ổn định

        122.5.Lưu ý / lỗi hay gặp

        aria - busy KHÔNG hiển thị loading cho người dùng

        -Chỉ dùng để hỗ trợ truy cập(accessibility)

        -Phải chuyển về "false"

        khi xử lý xong

        -Thường dùng CHUNG với aria-live

        -Không thay thế cho spinner / progress bar

        \*Hiểu nhanh:

        -Người dùng thường→ nhìn loading

        -Screen reader→ nghe aria - busy

        122.6.Ví dụ thực tế

        -Load dữ liệu từ server

        -ESP web UI đang đọc cảm biến

        -Đang submit form

        -Đang render lại danh sách

        Ví dụ:

        <div aria - busy="true"> Đang tải... < /div>

            <div aria - busy="false"> Dữ liệu đã sẵn sàng < /div>

                \*Ghi nhớ nhanh:

                aria-busy = "tao đang bận, khoan đọc"

                122.7.Exambles

                <!DOCTYPE html>

                <html lang="vi">

                <head>

                  <meta charset="UTF-8">

                  <title>aria-busy demo</title>

                  <style>

                    body {

                      background: #111;

                      color: #fff;

                      font-family: Arial;

                      padding: 30px;

                    }

                    button {

                      padding: 8px 16px;

                      cursor: pointer;

                    }

                    .box {

                      margin-top: 20px;

                      padding: 15px;

                      border: 1px solid #444;

                      min-height: 60px;

                    }

                    .loading {

                      color: #3fb950;

                    }

                  </style>

                </head>

                <body>

                  <h3>Demo aria-busy (cơ bản)</h3>

                  <button onclick="loadData()">Load dữ liệu</button>

                  <!-- KHU VỰC NỘI DUNG -->

                  <div id="content" class="box" aria-busy="false" aria-live="polite"> Nội dung ban đầu </div>

                  <script>

                    function loadData() {

                      const box = document.getElementById("content");

                      // 🔥 báo đang bận

                      box.setAttribute("aria-busy", "true");

                      box.innerHTML = "<span class='loading'>Đang tải dữ liệu...</span>

                      ";

                      // giả lập loading 2 giây

                      setTimeout(() => {

                        box.innerHTML = "✅ Dữ liệu đã tải xong";

                        box.setAttribute("aria-busy", "false");

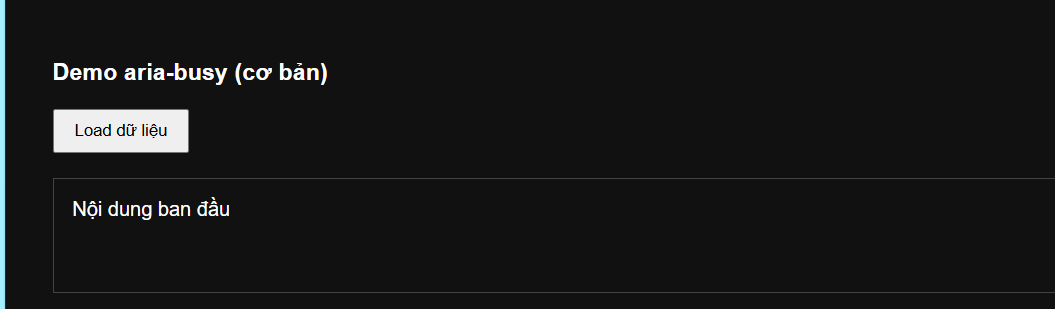
                      }, 2000);

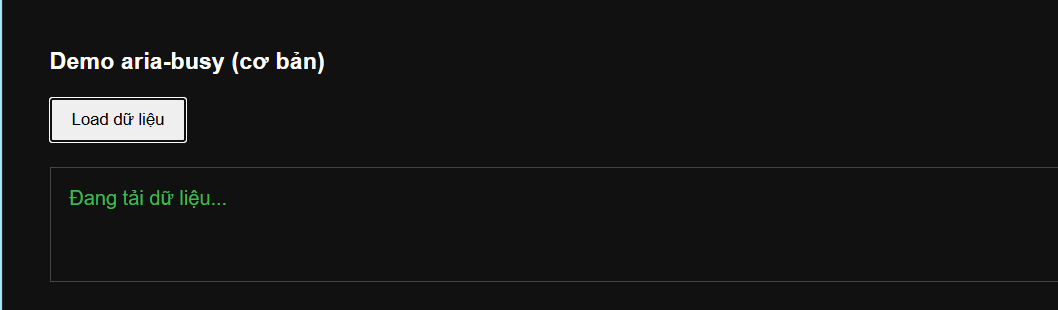
                    }

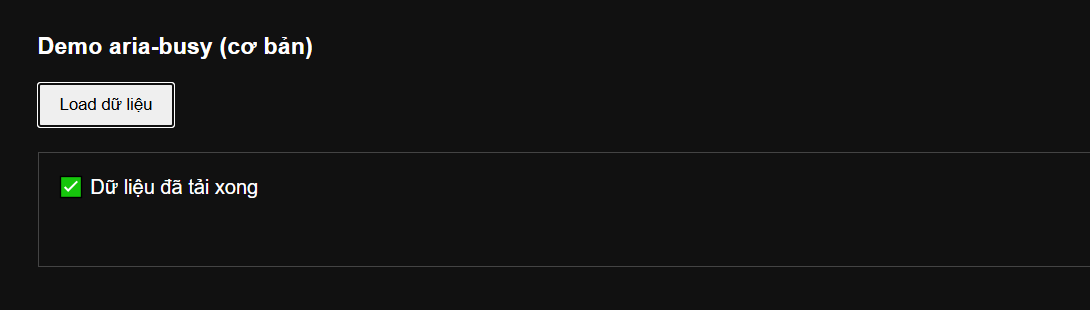
                  </script>

                </body>

                </html>







=== === === === === THUỘC TÍNH: aria - busy === === === === ===

123.Thuộc tính aria - busy

\*Dành cho:

-Web động

-Web ESP đọc cảm biến

-Trang có dữ liệu cập nhật liên tục

-Có loading dữ liệu lâu(WiFi, API, sensor nhiều)

123.1.Bản chất(dùng để làm gì)

+Báo cho trình đọc màn hình biết:

-KHU VỰC NÀY ĐANG BẬN→ CHƯA ĐỌC VỘI

-Dùng khi nội dung đang load / đang thay đổi

Hiểu đơn giản:

aria - busy = "khoan, tao đang xử lý"

123.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific) -

ARIA

-Gắn vào thẻ có nội dung thay đổi:

div, section, main, form, ul, table...

123.3.Cú pháp

<tag aria - busy="true">

  < /tag>

    <tag aria - busy="false">

      < /tag>

        123.4.Giá trị thường dùng

        true: đang xử lý / đang load / đang cập nhật false: xử lý xong / nội dung ổn định

        123.5.Lưu ý / lỗi hay gặp

        -aria - busy KHÔNG làm hiện vòng tròn loading

        -Chỉ dành cho screen reader(người khiếm thị)

        -PHẢI đổi về false khi xong việc

        -Không ảnh hưởng giao diện thường

        -Không dùng thì giao diện vẫn chạy bình thường

        123.6.Ví dụ dễ hiểu

        Ví dụ 1: Đang load dữ liệu

        <div aria - busy="true">

          Đang tải dữ liệu…</div>

        Ví dụ 2: Load xong

        <div aria - busy="false">Dữ liệu đã sẵn sàng </div>

        124.EXAMBLE

        <!DOCTYPE html>

        <html lang="vi">

        <head>

          <meta charset="UTF-8">

          <title>ARIA Busy đơn giản</title>

          <style>

            body {

              background: #0d1117;

              color: #e6edf3;

              font-family: Consolas, monospace;

              padding: 20px;

            }

            button {

              padding: 8px 16px;

              margin-right: 8px;

              border: none;

              border-radius: 6px;

              font-size: 14px;

              cursor: pointer;

              color: white;

            }

            button:first-of-type {

              background: #238636;

              /\* xanh \*/

            }

            button:last-of-type {

              background: #da3633;

              /\* đỏ \*/

            }

            #box {

              margin-top: 16px;

              padding: 12px;

              border: 1px solid #30363d;

              border-radius: 6px;

              background: #161b22;

            }

          </style>

        </head>

        <body>

          <h3>Test aria-busy (cơ bản)</h3>

        <button onclick="bat()">Bắt đầu</button>

        <button onclick="tat()">Kết thúc</button>

          <!-- thẻ tag <div> là thẻ chia khối-->

          <div id="box" aria-busy="false">Trạng thái: rảnh</div>

          <script>

            function bat() {

              box.ariaBusy = "true";

              box.innerText = "Trạng thái: đang xử lý...";

            }

            function tat() {

              box.ariaBusy = "false";

              /\*"Box" gán mã định danh cho  "  box.ariaBusy = "false";"\*/

              box.innerText = "Trạng thái: xong";   /\* innerText : đổi chữ \*/

            }

          </script>

        </body>

        </html>

=== === === === === THUỘC TÍNH: aria-checked === === === === ===[OKLA]

124. Thuộc tính aria-checked

Mô tả trạng thái logic của phần tử

<!--

✅ BẮT BUỘC phải dùng aria-checked="false" trong thuộc tính html

✅ CHỈ JS mới dùng ariaChecked

-->

124.1.Bản chất(dùng để làm gì)

-Dùng để báo trạng thái ĐÃ CHỌN hay CHƯA CHỌN cho MÁY(screen reader)

-Thường dùng cho checkbox,

switch, toggle TỰ LÀM bằng div / span

-Dành cho accessibility(không phải để trang trí)

Nói ngắn gọn:

aria - checked = nói cho MÁY biết là "đã tick hay chưa"

124.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific)

ARIA

-Không phải thẻ nào cũng dùng bừa

-Chỉ dùng khi phần tử có vai trò chọn / bật / tắt

Thường đi kèm với:

role = "checkbox"

role = "switch"

role = "menuitemcheckbox"

124.3.Cú pháp

<tag aria - checked="true"> </tag>

<tag aria - checked="false"> </tag>

<tag aria - checked="mixed"> </tag>

124.4.Giá trị thường dùng

"true" → đã chọn✅

"false" → chưa chọn⛔

"mixed" → chọn 1 phần(hiếm dùng)

Lưu ý:

-Luôn để trong dấu ngoặc kép ""

-Viết chữ thường

124.5.Lưu ý / lỗi hay gặp

❌Không dùng aria - checked cho thẻ

<input type="checkbox"> → vì input đã có checked sẵn rồi

❌Không dùng khi không có role phù hợp

❌Đừng dùng chỉ để đổi màu(ARIA không phải CSS)

✔ aria-checked chỉ để MÁY hiểu, không làm thay đổi giao diện

124.6.Ví dụ đơn giản(checkbox tự làm)

<div role="checkbox" aria - checked="false">Bật / Tắt </div>

Nghĩa là:

-Với người: chỉ là 1 khối

-Với máy: đây là 1 checkbox chưa được chọn

\*Định nghĩa phần tử(liên hệ để dễ nhớ):

1 checkbox = 1 phần tử

Trạng thái của phần tử đó:

+cho NGƯỜI→ đổi chữ / màu

+cho MÁY→ dùng aria - checked

124.7.EXAMBLE có cả thuộc tính id,class,role,onclick,

<!DOCTYPE html>

<html lang="vi">

<meta charset="UTF-8">

<title>Công tắc đơn giản</title>

<style>

  body {

    background: #0d1117;

    color: #e6edf3;

    font-family: Consolas, monospace;

    padding: 20px;

  }

  .switch {

    width: 200px;

    padding: 12px;

    border: 2px solid #30363d;

    border-radius: 6px;

    cursor: pointer;

  }

  .on {

    background: #238636;

  }

  .off {

    background: #161b22;

  }

</style>

<h3>Công tắc điều khiển</h3>

<div id="sw" class="switch off" role="switch" /\*role : Swtich ON/OFF chỉ 1 button \*/ aria-checked="false" /\* đổi giá

  trị/ghi đè\*/ onclick="doi()">🔴 ĐÈN: TẮT

</div>

<p id="log">Thiết bị: OFF</p>

<script>

  function doi() {

    if (sw.ariaChecked == "false") {

      sw.ariaChecked = "true";

      sw.className = "switch on";

      sw.innerText = "🟢 ĐÈN: BẬT";

      log.innerText = "Thiết bị: ON";

    }

    else {

      sw.ariaChecked = "false";   /\* nó đổi giá trị  từ aria-checked="true" sang aria-checked="false"  \*/

      sw.className = "switch off";

      sw.innerText = "🔴 ĐÈN: TẮT RỒI";

      log.innerText = "Thiết bị: OFF";

    }

  }

</script>

</html>

<!--

Ban đầu:

giá trị hiện tại là aria-checked="false"

→ lúc này đèn đang tắt

Khi bấm onclick="doi()"

→ trình duyệt gọi hàm function doi()

Trong hàm doi():

→ chương trình kiểm tra điều kiện trong ngoặc tròn của if

Điều kiện được so sánh:

→ nó kiểm tra xem giá trị ban đầu là  aria-checked="false" có bằng sw.ariaChecked == "false"  hay không

Kết quả so sánh:

→ thấy điều kiện thấy "false" với  "false" là đúng

Vì điều kiện đúng:

→ chương trình chạy khối lệnh trong ngoặc nhọn {} của if

Hành động trong if:

→ bật đèn

→ khi thực thi tới dòng sw.ariaChecked = "true"; thằng này có tác dụng đổi/ghi đè trạng thái

→ sw.ariaChecked = "true";  nó đổi từ giá trị aria-checked="false" sang aria-checked="true"

Người dùng bấm lần nữa onclick="doi()"

Hàm doi() lại chạy từ đầu

Lại kiểm tra điều kiện if:

→ kiểm tra aria-checked === "false"

Giá trị hiện tại lúc này:

→ aria-checked="true"

So sánh:

→ điều kiện sai

Vì điều kiện sai:

→ không chạy if

→ chuyển sang else

Hành động trong else:

→ nó thực thi đến dòng sw.ariaChecked = "false"; tác dụng nó đổi giá trị  từ aria-checked="true" sang aria-checked="false"

\*Kết quả:

→ đèn tắt

-->

=== === === === === THUỘC TÍNH: role === === === === === [OKLA]

125. Thuộc tính role

role đóng vai trò như nút swtich ON/OFF ralay

125.1.Bản chất(dùng để làm gì)

-Khai báo VAI TRÒ của một phần tử HTML

-Giúp trình duyệt, trình đọc màn hình(screen reader) hiểu phần tử này LÀ GÌ

-Thuộc hệ ARIA(Accessibility)

125.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific)

ARIA :

-Áp dụng cho HẦU HẾT các thẻ HTML

125.3.Cú pháp

<tag role="tên\_vai\_trò"> </tag>

Tên vai trò bắt buộc đặt đúng vai trò ko phải thích đặt sao đặt

125.4.Giá trị thường dùng

-button → đóng vai trò nút bấm

-switch → đóng vai trò công tắc bật / tắt

Checkbox → đóng vai trò ô chọn

dialog → hộp thoại

alert → thông báo cảnh báo

navigation → khu điều hướng

main → nội dung chính

125.5.Lưu ý / lỗi hay gặp

-role KHÔNG tự tạo hành vi(không tự click, không tự bật tắt)

-role chỉ MÔ TẢ vai trò, hành vi phải viết bằng JS

-Không dùng role khi đã có thẻ HTML đúng nghĩa(vd: có <button> rồi thì không cần role ="button")

  -role thường đi kèm aria-\* để mô tả trạng thái

3. LUẬT CỨNG của role="menuitem"

1. Phải nằm TRONG role="menu"

<div role="menu">

  <div role="menuitem">Mục 1</div>

</div>

❌ Sai:

<div role="menuitem">Mục lẻ</div>

125.6.Ví dụ về role

<div role="button"> Nhấn < /div>

    → div này KHÔNG phải button thật → nhưng được khai báo là ĐÓNG VAI TRÒ nút bấm

    <div role="switch" aria-checked="false"> </div>

    → phần tử này là công tắc → trạng thái hiện tại: tắt

    \*Ghi nhớ nhanh:

    role = "đóng vai"

    aria - \* = "trạng thái / thuộc tính phụ"

    JS = "hành động thật"

    \*So sánh nhanh:

    id → định danh DUY NHẤT

    class → nhóm nhiều phần tử

    role → nói cho máy biết phần tử NÀY LÀ GÌ

    125.6.1.Những điều cần biết về role

    ✅ role chỉ nhận GIÁ TRỊ CỐ ĐỊNH

    ARIA có danh sách role chuẩn, ví dụ:

    menu

    menuitem

    button

    switch

    dialog

    listbox

    option

    ❌ Không được:

    Thêm số

    Thêm chữ

    Tự chế tên

    125.7.EXAMBLE

    <!DOCTYPE html>

    <html>

    <body>

          <div style="

    background:red;

    width:60px;

    height:30px;

    cursor:pointer;

    text-align:center;

    line-height:30px;

    color:white;

  " onclick="this.style.background='green';">LED </div>

    </body>

    </html>

    124.8.EXAMBLE VIP

    <!DOCTYPE html>

    <html lang="vi">

    <head>

      <meta charset="UTF-8">

          <title>Switch đơn giản</title>

    </head>

    <body>

          <div role="switch" aria-checked="false" style="

    padding:20px;

    width:100px;

    text-align:center;

    color:white;

    background:red;

    cursor:pointer;

    user-select:none; " onclick="

    if (this.ariaChecked === 'false') {

      this.ariaChecked = 'true';

      this.style.background = 'green';

      this.innerText = 'ON';

    }

else {

      this.ariaChecked = 'false';

      this.style.background = 'red';

      this.innerText = 'OFF';

    }

  ">OFF </div>

    </body>

    </html>

    <!--ĐẦU TIÊN:

Giá trị ban đầu là

aria-checked="false"

Giao diện hiển thị:

-nền đỏ

-chữ OFF

Nghĩa là: công tắc đang TẮT

🔹 CLICK LẦN 1

-Người dùng bấm vào thẻ div

-Trình duyệt chạy đoạn mã trong onclick

-Chương trình kiểm tra điều kiện trong ngoặc tròn của if:

 this.ariaChecked === 'false'

Lúc này:

-giá trị hiện tại của aria-checked là "false"

-so sánh "false" === "false"

Kết quả: điều kiện ĐÚNG

🔹 THỰC THI KHỐI if

-Chương trình chạy các lệnh trong ngoặc nhọn {} của if

this.ariaChecked = 'true';

→ ghi đè / đổi trạng thái

→ từ aria-checked="false"

→ sang aria-checked="true"

this.style.background = 'green';

→ giao diện đổi sang màu xanh

→ biểu thị công tắc bật

this.innerText = 'ON';

→ đổi chữ hiển thị từ OFF → ON

🔹 CLICK LẦN 2

-Người dùng bấm lại lần nữa

-onclick lại chạy từ đầu

-Chương trình tiếp tục kiểm tra điều kiện if:

this.ariaChecked === 'false'

Lúc này:

-aria-checked="true"

-so sánh "true" === "false"

Kết quả: điều kiện SAI

🔹 THỰC THI KHỐI else

-Vì điều kiện sai → không chạy if

-Chương trình chuyển sang khối else

  this.ariaChecked = 'false';

→ đổi / gán lại trạng thái

→ từ aria-checked="true"

→ sang aria-checked="false"

this.style.background = 'red';

→ nút switch đổi sang màu đỏ

→ biểu thị tắt

this.innerText = 'OFF';

→ chữ hiển thị đổi từ ON → OFF

🔁 KẾT LUẬN (RẤT QUAN TRỌNG)

Mỗi lần onclick:

chương trình đọc trạng thái hiện tại

so sánh điều kiện

rồi ghi đè trạng thái ngược lại

👉 Vì vậy:

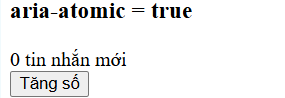
Click 1 → OFF → ON

Click 2 → ON → OFF

Quy trình này lặp đi lặp lại mỗi lần click

-->

125.Thuộc tính aria-atomic

- Tăng hoặc giảm theo thứ tự

-Chủ yếu dùng cho màn hình LCD

=== === === === === THUỘC TÍNH: aria-autocomplete === === === === ===[nghiên cứu sau]

126. Thuộc tính aria-autocomplete

Khi bấm vô ô nhập liệu gõ đại từ gì đó nó sẽ xổ ra nguyên dây dãy gợi ý

126.1.Bản chất(dùng để làm gì)

-Dùng cho ô nhập liệu(input, textbox)

-Thường dùng trong:

+Ô tìm kiếm

+Nhập lệnh

+Nhập topic MQTT

+Form có gợi ý dữ liệu

126.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific)

-Thuộc nhóm: ARIA

-Áp dụng cho:

<input>

<textarea>

  Phần tử có role = "textbox"

126.3.Cú pháp

<tag aria-autocomplete = "giá\_trị" > </tag>

Ví dụ:

<input aria-autocomplete = "list" >

126.4.Giá trị thường dùng

  none: Không có gợi ý(mặc định)

  inline: Tự điền tiếp ngay trong ô nhập

  list: Hiện danh sách gợi ý bên dưới

  both: Vừa inline + vừa list

126.5.Ý nghĩa từng giá trị(hiểu nhanh)

-none → Gõ tới đâu là tới đó, không gợi ý

-inline → Gõ "le"→ tự hiện "led\_on"

+trong input-list → Gõ "e"→

xổ danh sách:

esp / led

esp / temp

-both → Vừa tự điền + vừa xổ danh sách(xịn nhất)

126.6.Ví dụ chuẩn dễ hiểu

<input type = "text"aria-autocomplete = "list"aria - controls = "listbox">

126.7.Thuộc tính hay đi kèm(RẤT QUAN TRỌNG)

  aria-controls: trỏ tới danh sách gợi ý

  aria-expanded: cho biết danh sách đang mở hay đóng

  role = "listbox": cho khung gợi ý

  role = "option": cho từng dòng gợi ý

Ví dụ:

  <ul id = "listbox"

role = "listbox" >

  <li role = "option" > esp / led < /li> </ul>

126.8.Lưu ý / lỗi hay gặp

  aria - autocomplete CHỈ MÔ TẢ→ không tự tạo gợi ý

  Phải có JavaScript xử lý autocomplete

  Không có list mà ghi "list" → sai logic

  -Quên role = "listbox"→ screen reader không hiểu

  -aria - autocomplete≠ autocomplete(HTML)

126.9.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

    <meta charset="UTF-8">

    <title>aria-autocomplete = list</title>

    <style>

        ul {

            border: 1px solid #aaa;

            max-width: 200px;

            padding: 0;

            margin: 5px 0 0;

            list-style: none;

        }

        li {

            padding: 5px;

            cursor: pointer;

        }

        li:hover {

            background: #ddd;

        }

    </style>

</head>

<body>

    <!--

thẻ tag

<label> gán nhãn cho input

for có 1 công dụng chính: nối

<label> với

<input>

type="text"       → kiểu dữ liệu nhập

aria-autocomplete → báo có gợi ý

aria-controls     → input điều khiển ai

aria-expanded     → list mở hay đóng

role="option"     → từng dòng gợi ý

-->

    <label for="topic"> MQTT Topic:</label>

    <br>

    <input id="topic" type="text" aria-autocomplete="list" aria-controls="list" aria-expanded="false">

    <ul id="list" role="listbox" hidden>

        <li role="option">esp/led</li>

        <li role="option">esp/temp</li>

        <li role="option">esp/fan</li>

    </ul>

    <script>

        const input = document.getElementById("topic");

        const list = document.getElementById("list");

        input.addEventListener("input", () => {

            const show = input.value.length > 0;

            list.hidden = !show;

            input.setAttribute("aria-expanded", show);

        });

        list.addEventListener("click", e => {

            if (e.target.role === "option") {

                input.value = e.target.textContent;

                list.hidden = true;

                input.setAttribute("aria-expanded", "false");

            }

        });

    </script>

</body>

</html>

<!--

==================== GIẢI THÍCH SCRIPT aria-autocomplete ====================

Ban đầu:

→ Trang vừa load xong

→ input = ô nhập có id="topic"

→ list  = danh sách <ul id="list">

→ list đang hidden (ẩn)

→ aria-expanded="false" (danh sách đang đóng)

--------------------------------------------------------------------

Dòng 1:

const input = document.getElementById("topic");

→ Trình duyệt tìm phần tử có id="topic"

→ Gán phần tử input đó vào biến input

→ Từ giờ:

   input đại diện cho ô nhập MQTT Topic

--------------------------------------------------------------------

Dòng 2:

const list = document.getElementById("list");

→ Trình duyệt tìm phần tử có id="list"

→ Gán phần tử <ul> vào biến list

→ Từ giờ:

   list đại diện cho danh sách gợi ý

--------------------------------------------------------------------

Dòng 3:

input.addEventListener("input", () => { ... })

→ Lắng nghe sự kiện "input"

→ Nghĩa là:

   mỗi khi người dùng gõ / xóa / dán chữ trong ô input

   → khối lệnh bên trong sẽ được chạy

--------------------------------------------------------------------

Khi người dùng BẮT ĐẦU GÕ:

const show = input.value.length > 0;

→ input.value là nội dung người dùng đã nhập

→ length là độ dài chuỗi

So sánh:

→ nếu người dùng đã gõ ÍT NHẤT 1 ký tự

→ length > 0 → true

→ gán show = true

→ nếu ô trống

→ length = 0

→ show = false

--------------------------------------------------------------------

Dòng tiếp theo:

list.hidden = !show;

→ !show nghĩa là đảo ngược giá trị show

Trường hợp 1:

→ show = true (đã gõ chữ)

→ !show = false

→ list.hidden = false

→ danh sách HIỆN ra

Trường hợp 2:

→ show = false (ô trống)

→ !show = true

→ list.hidden = true

→ danh sách BỊ ẨN

--------------------------------------------------------------------

Dòng tiếp theo:

input.setAttribute("aria-expanded", show);

→ Cập nhật trạng thái cho trợ năng (ARIA)

Nếu show = true:

→ aria-expanded="true"

→ screen reader hiểu: danh sách đang mở

Nếu show = false:

→ aria-expanded="false"

→ screen reader hiểu: danh sách đang đóng

--------------------------------------------------------------------

--------------------------------------------------------------------

Tiếp theo là phần CLICK vào danh sách:

list.addEventListener("click", e => { ... })

→ Lắng nghe sự kiện click trên <ul>

→ Khi người dùng click vào 1 dòng <li>

→ trình duyệt gọi hàm này

→ e là sự kiện (event)

--------------------------------------------------------------------

Dòng:

if (e.target.role === "option") {

→ e.target là phần tử vừa bị click

→ role === "option" nghĩa là:

   chỉ xử lý khi click ĐÚNG vào <li role="option">

→ tránh click nhầm vào chỗ trống trong <ul>

--------------------------------------------------------------------

Nếu điều kiện ĐÚNG:

input.value = e.target.textContent;

→ Lấy nội dung chữ của <li> (vd: "esp/led")

→ Gán vào ô input

→ Người dùng thấy giá trị được điền vào input

--------------------------------------------------------------------

Dòng tiếp:

list.hidden = true;

→ Ẩn danh sách sau khi chọn xong

--------------------------------------------------------------------

Dòng cuối:

input.setAttribute("aria-expanded", "false");

→ Cập nhật trạng thái ARIA

→ Báo cho screen reader:

   danh sách đã đóng

--------------------------------------------------------------------

KẾT QUẢ CUỐI CÙNG:

→ Người dùng gõ → thấy gợi ý

→ Click chọn → input được điền

→ Danh sách tự ẩn

→ Trợ năng hiểu đầy đủ trạng thái

==================== KẾT THÚC GIẢI THÍCH ====================

-->

=== === === === === THUỘC TÍNH: aria-controls === === === === === [HIỂU]

126.Thuộc tính aria-controls

-nút bấm mở menu → phần tử điều khiển

-menu xổ → ra phần tử bị điều khiển

-aria-controls dùng để NỐI 2 phần tử đó lại với nhau

+ Bắt buộc phải có js với thực thi được hành động

126.1.Bản chất(dùng để làm gì) -

-Nói cho máy & screen reader biết: “Phần tử này điều khiển phần tử kia”

-KHÔNG tự động điều khiển

126.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific) -

-ARIA

- Dùng cho các phần tử có HÀNH VI ĐIỀU KHIỂN:

button, input, checkbox, tab, menu, toggle...

126.3.Cú pháp

<tag aria-controls="id\_phan\_tu\_bi\_dieu\_khien">

  < /tag>

    Ví dụ:

    <button aria - controls="menu"> </button>

    126.4.Giá trị thường dùng

    -Giá trị là id của phần tử khác

    -Có thể nhiều id, cách nhau bằng dấu cách

    Ví dụ:

    aria-controls = "menu panel log"

    126.5.Lưu ý / lỗi hay gặp

    -aria-controls KHÔNG thay thế JS

    -Chỉ có tác dụng KHI phần tử bị điều khiển TỒN TẠI

    -id được trỏ tới phải hợp lệ

    -Thường đi kèm với:

    aria-expanded

    aria-hidden

    aria-selected

    ❌ Sai:

    aria-controls = "abc"(nhưng không có id = "abc")

    126.6.Example minh họa aria-controls

    HTML:

    <button aria-controls="panel" aria-expanded="false">Mở panel </button>

    <div id="panel" hidden>Nội dung panel </div>

    Ý nghĩa:

    -Button điều khiển div #panel

    -Máy đọc code là hiểu ngay mối quan hệ

    -Screen reader biết:

    "Nút này mở / đóng panel"

    126.7.EXAMBLE ariaExpanded,hidden,aria-controls

<!DOCTYPE html>

<html lang="vi">

<head>

    <meta charset="UTF-8">

    <title>Test aria-controls đơn giản</title>

    <style>

        #panel {

            border: 1px solid #333;

            padding: 10px;

            margin-top: 8px;

        }

    </style>

</head>

<body>

    <button id="btn" aria-controls="panel" aria-expanded="false">Mở nội dung </button>

    <div id="panel" hidden> Đây là nội dung được điều khiển</div>

    <script>

        const btn = document.getElementById("btn");

        const panel = document.getElementById("panel");

        btn.onclick = () => {

            const dangMo = btn.ariaExpanded === "true";

            panel.hidden = dangMo;

            btn.ariaExpanded = String(!dangMo);

        };

    </script>

</body>

</html>

<!--

KHI KHÔNG DÙNG ariaExpanded

Bấm → mở / đóng

OK với người nhìn

❌ Máy không biết trạng thái

❌ Không biết đang mở hay đóng

NẾU CÓ ariaExpanded THÌ UI  CÓ NÃO

###

<button

  id="btn"

  aria-controls="panel"

  aria-expanded="false">Mở nội dung</button>

<div id="panel" hidden>Đây là nội dung được điều khiển</div>

###

BAN ĐẦU (khi trang vừa load):

- Button có:

  aria-expanded="false"

  → nghĩa là: NỘI DUNG ĐANG ĐÓNG

- Div panel có:

  hidden

  → nội dung BỊ ẨN trên màn hình

- Người dùng CHƯA bấm gì

###

btn.onclick = () => {

###

KHI NGƯỜI DÙNG BẤM NÚT:

→ Trình duyệt phát hiện sự kiện click

→ Tự động gọi hàm arrow function () => { ... }

###

const dangMo = btn.ariaExpanded === "true";

###

TRONG HÀM onclick:

- Chương trình đọc giá trị hiện tại của:

  btn.aria-expanded

- So sánh:

  btn.ariaExpanded === "true"

LÚC NÀY:

- aria-expanded đang là "false"

So sánh:

- "false" === "true" → SAI

KẾT QUẢ:

- dangMo = false

###

panel.hidden = dangMo;

###

GÁN TRẠNG THÁI ẨN / HIỆN:

- panel.hidden = false

Hiểu đơn giản:

- hidden = false → KHÔNG ẨN

→ panel HIỆN RA MÀN HÌNH

###

btn.ariaExpanded = String(!dangMo);

###

CẬP NHẬT TRẠNG THÁI ARIA:

- dangMo đang là false

- !dangMo → true

- String(true) → "true"

→ aria-expanded đổi từ:

  "false" → "true"

Ý nghĩa:

- NỘI DUNG ĐANG MỞ

- Máy đọc màn hình BIẾT trạng thái mới

===== KẾT QUẢ SAU LẦN BẤM THỨ NHẤT =====

- Nội dung: HIỆN

- aria-expanded="true"

- Người nhìn: thấy nội dung

- Máy đọc màn hình: biết đang mở

🔁 NGƯỜI DÙNG BẤM LẦN THỨ HAI

Người dùng bấm nút lần nữa

→ onclick chạy lại TỪ ĐẦU

###

const dangMo = btn.ariaExpanded === "true";

###

LÚC NÀY:

- aria-expanded = "true"

So sánh:

- "true" === "true" → ĐÚNG

→ dangMo = true

###

panel.hidden = dangMo;

###

- panel.hidden = true

→ panel BỊ ẨN

→ nội dung BIẾN MẤT khỏi màn hình

###

btn.ariaExpanded = String(!dangMo);

###

- dangMo = true

- !dangMo = false

- String(false) → "false"

→ aria-expanded đổi từ:

  "true" → "false"

Ý nghĩa:

- NỘI DUNG ĐANG ĐÓNG

===== KẾT QUẢ SAU LẦN BẤM THỨ HAI =====

- Nội dung: ẨN

- aria-expanded="false"

- UI quay về trạng thái ban đầu

### TÓM LẠI

aria-expanded KHÔNG tự làm gì

JavaScript quyết định ẩn / hiện

aria-expanded chỉ GHI NHẬN & MÔ TẢ trạng thái

###

-->

=== === === === === THUỘC TÍNH: aria-expanded === === === === ===[XEM EXAMBLE 126.7] [ KHÁ HIỂU ]

127. Thuộc tính aria-expanded

-Dùng để MÔ TẢ TRẠNG THÁI mở / đóng của một phần tử

-aria-expanded không điều khiển giao diện

-Nó chỉ NÓI cho máy đọc màn hình biết → phần tử đang mở hay đang đóng

-Thông báo cho js biết đóng hay mở

127.1.Bản chất(dùng để làm gì)

-Thuộc nhóm ARIA(Accessibility)

-Giúp trình đọc màn hình(screen reader) hiểu giao diện

-KHÔNG tự mở, KHÔNG tự đóng phần tử

Nói ngắn gọn:

aria - expanded = "nhãn trạng thái"cho máy đọc màn hình

127.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific) -

ARIA attribute

-Thường dùng cho:

+button

+a

+div(đóng vai trò button)

-Áp dụng cho phần tử có khả năng mở / đóng nội dung khác

127.3.Cú pháp

<tag aria-expanded="false"> </tag>

Ví dụ:

<button aria-expanded="false"> Menu </button>

127.4.Giá trị thường dùng

"true"→ đang MỞ(expanded)

"false"→ đang ĐÓNG(collapsed)

\*\*LƯU Ý:

-Giá trị là STRING, không phải boolean

-Luôn đặt trong dấu nháy ""

127.5.Lưu ý / lỗi hay gặp

-aria-expanded KHÔNG thay thế JavaScript

-Không tự ẩn / hiện giao diện

-Phải kết hợp với:

+hidden

+CSS

+JavaScript

-Sai phổ biến:

+Dùng true / false không có dấu nháy

+Nghĩ aria-expanded tự điều khiển UI ( GIAO DIỆN NGƯỜI DÙNG )

127.6.Ví dụ chuẩn, dễ hiểu

<button id="btn" aria-expanded="false"> Mở </button>

< div id="panel" hidden> Nội dung </div>

  JS:

  -Click →aria-expanded đổi trạng thái

  -panel ẩn / hiện theo trạng thái đó

  127.7.So sánh nhanh để dễ nhớ

  -hidden → Ẩn / hiện THẬT

  -display: none → Ẩn giao diện

  -aria-expanded → MÔ TẢ trạng thái

  -JavaScript → QUYẾT ĐỊNH logic

  128. CÁCH BẮT SỰ KIỆN click TRONG JAVASCRIPT

  CÁCH 1: onclick TRỰC TIẾP TRONG HTML(CÁCH CŨ)

  <button onclick="gui()"> Bấm </button>

  <script>

    function gui() {

      console.log("Đã click");

    }

  </script>

  Đặc điểm:

  Viết JS trong HTML Dễ hiểu cho người mới

  ❌ HTML dính JS → không sạch code

  CÁCH 2: GÁN onclick BẰNG JS + function

  <button id="btn"> Bấm < /button>

      <script>

        const btn = document.getElementById("btn");

        btn.onclick = function () {

          console.log("Đã click");

        }; </script>

      Đặc điểm:

      JS tách khỏi HTML

      Code rõ ràng, chuẩn hơn Dùng function truyền thống

      CÁCH 3: GÁN onclick BẰNG JS + ARROW FUNCTION( => )⭐

      <button id="btn"> Bấm < /button>

          <script>

            const btn = document.getElementById("btn");

            btn.onclick = () => {

              console.log("Đã click");

            }; </script>

          Đặc điểm:

          Cách viết hiện đại(ES6 + )

          Ngắn gọn, dễ đọc

          Phổ biến nhất hiện nay

          \*Lưu ý:

          btn.onclick = ...→ JavaScript

          => → hàm mũi tên(arrow function)

          () => {} tương đương function() {}

          === === === === === THUỘC TÍNH: aria-current === === === === ===[ HIỂU]

          129. Thuộc tính aria-current

          - aria-current dùng để báo phần tử nào đang là hiện tại(menu/trang đang chọn) cho trình duyệt & screen reader.

          ( như hinh là màu xanh lam)

          - Nó KHÔNG tự hiển thị hay thao tác UI; muốn thấy hoặc đổi thì phải dùng CSS / JS.

          129.1.Bản chất(dùng để làm gì)

          \*Thuộc nhóm ARIA(Accessibility)

          \*Dùng để CHỈ RA phần tử nào đang là "hiện tại"(current)

          \*Giúp trình đọc màn hình(screen reader) hiểu trạng thái giao diện

          \*Không làm thay đổi giao diện trực tiếp(chỉ hỗ trợ truy cập)

          129.2.Phạm vi áp dụng(Global / Event / ARIA / Tag-specific)

          \*ARIA

          \*Áp dụng cho nhiều thẻ HTML(a, li, button, nav, span...)

          129.3.Cú pháp

          <tag aria-current="giá\_trị"> </tag>

          Ví dụ:

          <a href="#home" aria-current="page"> Trang chủ < /a>

              129.4.Giá trị thường dùng

              \*page: trang hiện tại(menu điều hướng)

              \*step: bước hiện tại(form nhiều bước)

              \*location: vị trí hiện tại(bản đồ, danh sách)

              \*date: ngày hiện tại(lịch)

              \*time: thời điểm hiện tại

              \*true: hiện tại(chung chung)

              \*false: không phải hiện tại(thường không cần ghi)

              129.5.Lưu ý / lỗi hay gặp

              \*CHỈ 1 phần tử nên có aria-current trong cùng 1 ngữ cảnh

              \*aria-current KHÔNG thay thế class hay id

              \*Không dùng để style giao diện(CSS không bắt buộc phải dùng)

              \*Dùng khi có "nhiều lựa chọn"và cần biết cái nào đang active

              129.6.EXAMBLE

              Menu điều hướng:

              <nav>

                <a href="/home" aria-current="page"> Home </a>

                <a href="/about"> About < /a>

                    <a href="/contact"> Contact </a>

              </nav>

              -Ghi nhớ nhanh:

              \*id: định danh DUY NHẤT để code xử lý

              \*class: gom NHIỀU phần tử để style

              \*aria-current: báo cho TRÌNH ĐỌC MÀN HÌNH biết "cái nào đang hiện tại"

              -Ký hiệu liên quan hay gặp:

              aria-\*: thuộc tính hỗ trợ accessibility

              current = hiện tại/đang được chọn

              129.7.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

    <meta charset="UTF-8">

    <title>Test aria-current + CSS</title>

    <style>

        a[aria-current="page"] {

            color: red;

            font-weight: bold;

        }

    </style>

</head>

<body>

    <h3>Menu</h3>

    <a aria-current="page">WiFi</a>

    |

    <a>Config</a>

    |

    <a>Info</a>

</body>

</html>

=== === === === === THUỘC TÍNH: aria-describedby === === === === ===[HIỂU]

130. Thuộc tính aria-describedby

MÔ TẢ PHỤ##

\* Hướng dẫn nhập

(vd: ít nhất 8 ký tự, chỉ chữ và số)

\*Thông báo lỗi / cảnh báo

(vd: mật khẩu sai, email không hợp lệ)

\*Ghi chú bổ sung

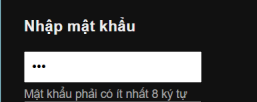
(vd: thông tin này chỉ dùng để xác minh)

\* Điều kiện / quy tắc

(vd: không chứa khoảng trắng, không dấu)

\*Thông tin ngữ cảnh

(vd: ngày áp dụng, thời gian hiệu lực)



aria-describedby: dùng khi đó “ Mật khẩu phải có ít nhất 8 ký tự”

130.1.Bản chất(dùng để làm gì)

\*Thuộc nhóm ARIA(Accessibility)

\*Dùng để LIÊN KẾT phần tử chính với phần tử MÔ TẢ thêm

\*Giúp máy đọc màn hình hiểu: "phần tử này có lời giải thích bổ sung"

\*Thường dùng cho: input, button, form, cảnh báo, hướng dẫn

130.2.Phạm vi áp dụng(Global / Event / ARIA / Tag - specific)

\*ARIA

\*Áp dụng cho NHIỀU thẻ HTML(input, button, select, textarea, a, div...)

130.3.Cú pháp

<tag aria-describedby="id\_mo\_ta"> </tag>

Ví dụ:

<input aria-describedby="help">

<div id="help"> Nhập mật khẩu ít nhất 8 ký tự < /div>

    130.4.Giá trị thường dùng

    \*Giá trị là id của phần tử MÔ TẢ

    \*Có thể chứa NHIỀU id(cách nhau bằng dấu cách)

    Ví dụ nhiều mô tả:

    aria-describedby = "help error note"

    130.5.Lưu ý / lỗi hay gặp

    \*aria-describedby KHÔNG làm hiện giao diện

    \*Phần tử mô tả(id) PHẢI tồn tại

    \*Không dùng để thay thế label

    \*Có thể mô tả nội dung ẨN(hidden, display: none)

   130.6.Ví dụ thực tế(dễ hiểu)

    Form nhập mật khẩu:

    <input type="password" aria-describedby="pw-note">

    <div id="pw-note"> Mật khẩu phải có ít nhất 8 ký tự </div>

    Giải thích:

    \*Người nhìn: đọc dòng hướng dẫn

    \*Máy đọc màn hình: đọc thêm nội dung mô tả khi focus input

    -So sánh nhanh để nhớ:

    aria - label: đặt TÊN cho phần tử

    aria - labelledby: lấy TÊN từ phần tử khác

    aria - describedby: lấy MÔ TẢ từ phần tử khác

    -Ghi nhớ nhanh:

    label = gọi tên

    describedby = giải thích thêm

    -Ký hiệu liên quan hay gặp:

    aria-\*: thuộc tính hỗ trợ accessibility

    described = được mô tả

    by = bởi

130.7.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

    <meta charset="UTF-8">

    <title>Test aria-describedby</title>

    <style>

        body {

            background: #111;

            color: #eee;

            font-family: Arial;

            padding: 20px;

        }

        input {

            padding: 8px;

            font-size: 16px;

        }

        .note {

            margin-top: 6px;

            color: #aaa;

            font-size: 14px;

        }

    </style>

</head>

<body>

    <h3>Nhập mật khẩu</h3>

    <input type="password" aria-describedby="pw-note">

    <div id="pw-note" class="note">Mật khẩu phải có ít nhất 8 ký tự</div>

    <input aria-describedby="note1 note2">

    <div id="note1">Ít nhất 8 ký tự</div>

    <div id="note2">Không chứa khoảng trắng</div>

</body>

</html>

=== === === === === THUỘC TÍNH: aria-disabled === === === === ===[ KHÁ HIỂU ]

131. Thuộc tính aria-disabled

131.1. Bản chất (dùng để làm gì)

- Thông báo cho TRÌNH ĐỌC MÀN HÌNH (screen reader) biết phần tử đang bị vô hiệu hóa

- Dùng cho TRỢ NĂNG (accessibility), KHÔNG trực tiếp khóa chức năng

- Người dùng bình thường vẫn có thể click nếu JS không chặn

=> aria-disabled = "tôi đang bị vô hiệu hóa về mặt NGỮ NGHĨA"

131.2. Phạm vi áp dụng

- ARIA attribute

- Dùng cho các phần tử tương tác:

  button, a, input, select, custom component (div role="button")

131.3. Cú pháp

<tag aria-disabled="true"> </tag>

131.4. Giá trị thường dùng

- "true"  → bị vô hiệu hóa

- "false" → hoạt động bình thường

131.5. Ví dụ cơ bản

<button aria-disabled="true">Nút bị khóa</button>

<a href="#" aria-disabled="true">Link bị khóa</a>

<div role="button" aria-disabled="true">Nút tự tạo</div>

131.6. So sánh với disabled (RẤT QUAN TRỌNG)

- disabled:

  + KHÓA THẬT

  + Không click, không focus

  + Chỉ dùng cho input, button, select, textarea

- aria-disabled:

  + CHỈ THÔNG BÁO

  + Vẫn click được nếu JS không chặn

  + Dùng cho cả thẻ không hỗ trợ disabled (a, div)

131.7. Khi nào DÙNG aria-disabled

- Khi làm nút hoặc component TỰ CODE bằng div / span

- Khi cần giữ focus nhưng vẫn báo là bị khóa

- Khi làm giao diện CHUẨN trợ năng

131.8. Khi nào KHÔNG NÊN dùng aria-disabled

- Nếu là <button>, <input> bình thường → ưu tiên dùng disabled

- Không dùng aria-disabled để thay thế logic JS

131.9. Kết hợp đúng cách (BEST PRACTICE)

<button aria-disabled="true" disabled>...</button>

Hoặc với div:

<div role="button"

     aria-disabled="true"

     tabindex="-1">...</div>

131.10. Lỗi hay gặp

- Nghĩ aria-disabled tự khóa click ❌

- Không chặn sự kiện JS ❌

- Dùng aria-disabled mà không có role phù hợp ❌

131.11. Ghi nhớ nhanh

- disabled = KHÓA CHỨC NĂNG

- aria-disabled = KHÓA Ý NGHĨA

- aria = cho MÁY ĐỌC, không phải cho trình duyệt

131.12.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

<meta charset="UTF-8">

<title>Test aria-disabled</title>

<style>

  .btn {

    padding: 10px 16px;

    margin: 6px;

    font-size: 16px;

    cursor: pointer;

  }

  /\* Chỉ làm cho giống bị khóa \*/

  .fake-disabled[aria-disabled="true"] {

    opacity: 0.5;

    cursor: not-allowed;

  }

</style>

</head>

<body>

<h3>aria-disabled + JS (chuẩn)</h3>

<!-- chỉ cần đổi aria-disabled="true" sang false là Nút hoạt động ✅ -->

<button

  id="btn3"

  class="btn fake-disabled"

  aria-disabled="true">aria-disabled + JS

</button>

<script>

  const btn = document.getElementById("btn3");

  btn.addEventListener("click", function (e) {

    if (this.getAttribute("aria-disabled") === "true") {

      e.preventDefault();

      alert("Nút đang bị khóa ❌");

      return;

    }

    alert("Nút hoạt động ✅");

  });

</script>

</body>

</html>

<!--

###

<button

  id="btn3"

  class="btn"

  aria-disabled="true">

  aria-disabled + JS

</button>

###

###

const btn = document.getElementById("btn3");

btn.addEventListener("click", function (e) {

  if (this.getAttribute("aria-disabled") === "true") {

    e.preventDefault();

    alert("Nút đang bị khóa ❌");

    return;

  }

  alert("Nút hoạt động ✅");

});

##

2.TRẠNG THÁI BAN ĐẦU (KHI TRANG VỪA LOAD)

HTML

\*Button có:

aria-disabled="true"

→ nghĩa là: NÚT ĐANG BỊ KHÓA (về mặt NGỮ NGHĨA)

CSS:

.fake-disabled

→ làm nút mờ đi, chuột hiện 🚫

→ CHỈ LÀ GIAO DIỆN, không ảnh hưởng logic

#LƯU Ý QUAN TRỌNG

→Nút VẪN CLICK ĐƯỢC

→Trình duyệt KHÔNG tự khóa

→Mọi thứ phụ thuộc vào JavaScript

3. DÒNG NÀY LÀM GÌ?

const btn = document.getElementById("btn3");

👉 JavaScript:

→document : đoc toàn bộ trang HTML

→ getElementById("btn3") Tìm phần tử có id="btn3"

→ Gán vào biến btn

→ Từ đây, btn đại diện cho nút trên màn hình

4. GẮN SỰ KIỆN CLICK

btn.addEventListener("click", function (e) {

Ý nghĩa:

addEventListener “Khi CÓ SỰ KIỆN xảy ra thì làm gì đó”

→ khi bấm btn có sự kiện xảy ra làm gì tiếp

→ Khi người dùng bấm nút

→ Trình duyệt gọi hàm function (e) { ... }

e là:

→ event object

→ Đại diện cho hành động click vừa xảy ra

5. KIỂM TRA TRẠNG THÁI aria-disabled

if (this.getAttribute("aria-disabled") === "true") {

Giải thích TỪNG Ý:

getAttribute("aria-disabled") "LẤY GIÁ TRỊ THUỘC TÍNH TRONG HTML"

this

→ chính là button đang bị click

getAttribute("aria-disabled")

→ đọc giá trị trong HTML

→ trả về CHUỖI "true" hoặc "false"

\*LÚC BAN ĐẦU:

aria-disabled = "true"

So sánh:

"true" === "true" → ĐÚNG

→ Điều kiện IF được kích hoạt

6. CHẶN HÀNH ĐỘNG CLICK

e.preventDefault();

👉 Ý nghĩa:

preventDefault();"Ngăn hành vi mặc định"

Với button:

Không cho chạy hành động tiếp theo

Với link <a>:

Không chuyển trang

📌 Đây là chỗ KHÓA THẬT

7. THÔNG BÁO CHO NGƯỜI DÙNG

alert("Nút đang bị khóa ❌");

→ Hiện thông báo:

Nút đang bị vô hiệu hóa

8. DỪNG HÀM NGAY LẬP TỨC

return;

👉 RẤT QUAN TRỌNG:

Thoát khỏi hàm

Các dòng phía dưới KHÔNG CHẠY

9. TRƯỜNG HỢP aria-disabled = "false"

Nếu HTML là:

<button aria-disabled="false">

Thì khi click:

So sánh:

"false" === "true" → SAI

→ KHÔNG vào if

→ Chạy dòng này:

alert("Nút hoạt động ✅");

-->

=== === === === === THUỘC TÍNH: aria-haspopup === === === === ===

132. Thuộc tính aria-haspopup

Đóng vai trò là menu, muấn xổ ra cần js

132.1. Bản chất (dùng để làm gì)

- Thông báo cho trình đọc màn hình biết:

-Phần tử này khi bấm vào sẽ MỞ RA một thứ khác

- Thứ được mở có thể là:

  + menu

  + dialog

  + listbox

  + grid

  + tree

- Không tự tạo chức năng

- CHỈ dùng để mô tả cho trợ năng (accessibility)

Hiểu đơn giản:

aria-haspopup = "bấm cái này sẽ bật thêm thứ khác"

--------------------------------------------------

132.2. Phạm vi áp dụng (Global / Event / ARIA / Tag-specific)

- ARIA attribute

- Áp dụng cho các thẻ có thể tương tác:

  + button

  + a

  + div (khi có role phù hợp)

--------------------------------------------------

132.3. Cú pháp

<tag aria-haspopup="true"> </tag>

hoặc dạng rõ loại popup:

<tag aria-haspopup="menu"> </tag>

--------------------------------------------------

132.4. Giá trị thường dùng

- true       → có popup (chung chung)

- menu       → mở menu

- dialog     → mở hộp thoại

- listbox    → mở danh sách chọn

- grid       → mở bảng

- tree       → mở cây dữ liệu

Ví dụ phổ biến nhất:

aria-haspopup="menu"

--------------------------------------------------

132.5. Lưu ý / lỗi hay gặp

- aria-haspopup KHÔNG tự mở popup

  → phải dùng JavaScript

- Thường dùng KÈM với:

  + aria-expanded="true | false"

- Không dùng bừa cho phần tử không tương tác

- Không thay thế được HTML/CSS/JS

--------------------------------------------------

132.6. Ví dụ dễ hiểu (thực tế)

<button aria-haspopup="menu" aria-expanded="false">

  Mở menu

</button>

Ý nghĩa:

- Nút này có menu

- Hiện tại menu đang đóng

Khi mở menu bằng JS:

aria-expanded="true"

--------------------------------------------------

132.7. So sánh nhanh để dễ nhớ

aria-haspopup  → có popup hay không

aria-expanded  → popup đang mở hay đóng

aria-controls  → popup đó là thằng nào

--------------------------------------------------

132.8. Ghi nhớ 1 câu

❗ aria-haspopup = NÓI cho máy đọc màn hình biết

❗ KHÔNG phải để làm hiệu ứng

❗ KHÔNG thay JavaScript

132.8.1.

CẦN KHI DÙNG aria-haspopup

KHI NÀO MÁY ĐỌC MÀN HÌNH CẦN BIẾT:

"Nút này bấm vào

-sẽ BẬT RA

-MỘT THỰC THỂ KHÁC

-KHÔNG PHẢI CHỈ ĐỔI TRẠNG THÁI"

\*CỤ THỂ HƠN — aria-haspopup CẦN KHI:

1. NÚT MỞ MENU NGỮ CẢNH

→ ví dụ: menu chuột phải, menu tài khoản

→ bấm nút

→ một KHỐI MENU RIÊNG bật ra

→ có nhiều lựa chọn bên trong

2. NÚT MỞ DROPDOWN CHỌN

→ danh sách lựa chọn ẩn

→ không nằm “cùng dòng” với nút

3. NÚT MỞ DIALOG / MODAL

→ cửa sổ bật lên

→ chặn hoặc tách khỏi luồng chính

4. NÚT MỞ LISTBOX / POPUP PHỨC TẠP

→ không chỉ ẩn / hiện 1 đoạn text

→ mà là MỘT THỰC THỂ CÓ VAI TRÒ RIÊNG

VẬY MÁY ĐỌC MÀN HÌNH CẦN aria-haspopup ĐỂ LÀM GÌ?

→ ĐỂ BÁO TRƯỚC CHO NGƯỜI DÙNG:

"Chuẩn bị nha,

sau khi bấm

sẽ xuất hiện

một popup / menu / dialog"

132.9.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

    <meta charset="UTF-8">

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

    <title>Test aria-haspopup</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            padding: 40px;

        }

        button {

            padding: 10px 20px;

            cursor: pointer;

        }

        #menu {

            margin-top: 10px;

            padding: 10px;

            border: 1px solid #333;

            width: 150px;

            display: none;

        }

    </style>

</head>

<body>

    <button id="btn" aria-haspopup="menu" aria-expanded="false" aria-controls="menu">

        Mở menu

    </button>

    <!--

    aria-expanded="false" : Thông  báo cho js hiện đang đóng/mơ

    aria-controls="menu" : Nói cho js biết phần từ này điều khiển phần tử kia

    aria-haspopup="menu" :menu xổ  ra

    thuộc tính role      : đóng vai  trò như nút Switch . role có 2 trạng thái

    js đọc div không biết dùng làm gì nên dùng role đóng vai như nút Switch

    role="menuitem" : "menuitem" trong ngoặc kép phải đặt đúng vai trò không phải muấn đặt sau thì đặt

    "menuitem"      :mục menu

    "menu"          :là con của menuitem

getAttribute() → LẤY GIÁ TRỊ HIỆN TẠI của thuộc tính (ví dụ aria-expanded)

setAttribute() → GÁN GIÁ TRỊ MỚI cho thuộc tính đó

"==="        : SO SÁNH NGHIÊM NGẶT

"menu.style.display = isOpen ? "none" : "block";" :TOÁN TỬ 3 NGÔI

##Cú pháp chuẩn##

điều\_kiện ? giá\_trị\_nếu\_đúng : giá\_trị\_nếu\_sai

-->

    <div id="menu" role="menu">

        <div role="menuitem">Mục 1</div>

        <div role="menuitem">Mục 2</div>

        <div role="menuitem">Mục 3</div>

        <div role="menuitem"> Mục 4 </div>

    </div>

    <script>

        const btn = document.getElementById("btn");

        const menu = document.getElementById("menu");

        btn.onclick = function () {

            const isOpen = btn.getAttribute("aria-expanded") === "true";

            // LẤY GIÁ TRỊ HIỆN TẠI của thuộc tính (ví dụ aria-expanded)

            btn.setAttribute("aria-expanded", !isOpen);

            menu.style.display = isOpen ? "none" : "block";

        };

    </script>

</body>

</html>

1.Trạng thái ban đầu (chưa bấm gì)

###

<button id="btn" aria-haspopup="menu" aria-expanded="false" aria-controls="menu">

    ###

    aria-expanded="false"

    ➜ Thông báo trạng thái hiện tại: menu đang ĐÓNG

    aria-controls="menu"

    ➜ Nút này điều khiển thằng có id="menu"

    aria-haspopup="menu"

➜ Nút này có menu xổ ra

###

    <div id="menu" role="menu" style="display:none">

    ###

        display: none

        ➜ Menu đang ẩn

        role="menu"

        ➜ Khai báo vai trò cho screen reader hiểu đây là menu

        Các role="menuitem"

        ➜ Mỗi dòng là 1 mục trong menu

        📌 JS không cần role để chạy, role là cho trợ năng (accessibility).

        2. Khi người dùng bấm nút (onclick)

        ###

        btn.onclick = function () {

        ###

        👉 Khi click → hàm chạy.

      3. Đọc trạng thái hiện tại (KHÔNG gán)

        ###

        const isOpen = btn.getAttribute("aria-expanded") === "true";

        ###

        getAttribute(...)

        ➜ LẤY GIÁ TRỊ HIỆN TẠI

        So sánh bằng === "true"

        Lần bấm đầu tiên:

        aria-expanded = "false"

        "false" === "true" → false

        👉 isOpen = false

        👉 Nghĩa là: menu mở

        4. Đảo trạng thái và ghi lại vào aria

        ###

        btn.setAttribute("aria-expanded", !isOpen);

        ###

        isOpen = false

        !isOpen = true

        👉 Gán lại:

        aria-expanded = "true"

        📌 Đây là bước:

        Cập nhật trạng thái mới cho lần click tiếp theo

        5. Hiện / ẩn menu dựa trên trạng thái CŨ

        ###

        menu.style.display = isOpen ? "none" : "block";

        ###

        Diễn giải đúng theo bạn đang hiểu:

        Nếu TRƯỚC ĐÓ đang mở (isOpen === true)

        ➜ "none" → đóng menu

        Nếu TRƯỚC ĐÓ đang đóng (isOpen === false)

        ➜ "block" → mở menu

        Lần bấm đầu tiên:

        isOpen = false

        → menu.style.display = "block"

        khi ấy trạng thái loop

132.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

    <meta charset="UTF-8">

    <title>Menu đơn giản</title>

    <style>

        #menu {

            display: none;

            border: 1px solid #000704;

            width: 100px;

            padding: 5px;

            margin-top: 5px;

        }

    </style>

</head>

<body>

    <button onclick="toggleMenu()">Mở menu</button>

    <div id="menu">

        <div>TH2</div>

        <div>TH3</div>

    </div>

    <script>

        let isOpen = true;

        function toggleMenu() {

            if (isOpen === false) {

                document.getElementById("menu").style.display = "block";

                isOpen = true;

            } else {

                document.getElementById("menu").style.display = "none";

                isOpen = false;

            }

        }

    </script>

</body>

</html>

<!--

###

<button onclick="toggleMenu()">Mở menu</button>

<div id="menu">

    <div>TH2</div>

    <div>TH3</div>

</div>

###

BAN ĐẦU (khi trang vừa load):

- CSS đã đặt:

#menu { display: none; }

→ Nghĩa là:

menu BỊ ẨN

người dùng KHÔNG thấy TH2, TH3

- JavaScript có:

let isOpen = false;

→ JS đang GHI NHỚ:

menu hiện tại đang ĐÓNG

- Nghe có vẻ vô lý nhưng display: none; cũng tương đương với việc isOpen = false;

- Người dùng:

CHƯA bấm nút nào

###

<button onclick="toggleMenu()">

    ###

    KHI NGƯỜI DÙNG BẤM NÚT LẦN ĐẦU:

    → Trình duyệt phát hiện sự kiện click

    → Gọi hàm toggleMenu()

    → Hàm chạy TỪ DÒNG ĐẦU TIÊN

    ###

    let isOpen = false;

    ###

    TRƯỚC KHI VÀO HÀM:

    - Giá trị isOpen hiện tại là:

    false

    → Nghĩa là:

    menu ĐANG ĐÓNG

    ###

    if (isOpen === false) {

    ###

    JS KIỂM TRA ĐIỀU KIỆN:

    - So sánh:

    isOpen === false

    - Giá trị thực tế:

    false === false → ĐÚNG

    → Điều kiện IF ĐÚNG

    → JS CHUI VÀO KHỐI IF

    ###

    document.getElementById("menu").style.display = "block";

    ###

    HÀNH ĐỘNG KHI IF ĐÚNG:

    - JS tìm phần tử có id="menu"

    - Ghi đè CSS:

    display: block

    → KẾT QUẢ:

    menu HIỆN RA MÀN HÌNH

    người dùng THẤY TH2, TH3

    ###

    isOpen = true;

    ###

    CẬP NHẬT TRẠNG THÁI:

    - JS ghi nhớ lại:

    menu BÂY GIỜ ĐANG MỞ

    → DÒNG NÀY LÀ "BỘ NHỚ"

    → KHÔNG CÓ DÒNG NÀY → MENU SẼ HỎNG

    NGƯỜI DÙNG BẤM NÚT LẦN THỨ HAI

    → Trình duyệt gọi toggleMenu() LẠI TỪ ĐẦU

    ###

    if (isOpen === false) {

    ###

    LÚC NÀY:

    - isOpen hiện tại là:

    true

    So sánh:

    - true === false → SAI

    → IF KHÔNG CHẠY

    → JS NHẢY SANG ELSE

    ###

    } else {

    ###

    JS BIẾT RẰNG:

    - Menu ĐANG MỞ

    - CẦN ĐÓNG MENU LẠI

    ###

    document.getElementById("menu").style.display = "none";

    ###

    HÀNH ĐỘNG TRONG ELSE:

    - Ghi đè CSS:

    "".style.display = "none";" : ghi đè, "trong ngoặc kép là mức độ ưu tiên cao nhất"

    display: none

    → KẾT QUẢ:

    menu BỊ ẨN

    TH2, TH3 BIẾN MẤT KHỎI MÀN HÌNH

    ###

    isOpen = false;

    ###

    CẬP NHẬT TRẠNG THÁI LẦN NỮA:

    - JS ghi nhớ:

    menu ĐÃ ĐÓNG

    → Chuẩn bị cho lần bấm tiếp theo

    ### TÓM LẠI TOÀN BỘ LUỒNG HOẠT ĐỘNG ###

    - CSS:

    ẨN menu ban đầu

    - HTML:

    Cung cấp nút + menu

    - JavaScript:

    isOpen = BỘ NÃO

    if → mở menu

    else → đóng menu

    - MỖI LẦN BẤM:

    JS đọc isOpen

    → quyết định HIỆN hay ẨN

    → rồi cập nhật lại isOpen

    Đây chính là:

    "ĐIỀU KHIỂN GIAO DIỆN DỰA TRÊN TRẠNG THÁI"

    ###

-->

=== === === === === THUỘC TÍNH: aria-hidden === === === === === [K QUAN TRỌNG BỎ]

133. Thuộc tính aria-hidden

→ QUAN TRỌNG: Ẩn phần tử với TRÌNH ĐỌC MÀN HÌNH (screen reader),

KHÔNG ẩn với mắt người dùng.

133.1. Dùng để làm gì

- Giữ giao diện gọn, tránh đọc dư / đọc rác

133.2. Bản chất

- Thuộc nhóm ARIA (Accessibility)

- Ảnh hưởng đến công cụ hỗ trợ (assistive technology),

KHÔNG ảnh hưởng hiển thị

133.3. Phạm vi áp dụng

- ARIA attribute

- Dùng cho HẦU HẾT thẻ HTML (div, span, button, icon, nav...)

133.4. Cú pháp (PHẦN QUAN TRỌNG – GHI KỸ)

<tag aria-hidden="true"> </tag>

<tag aria-hidden="false"> </tag>

→ "true" : Ẩn với screen reader

→ "false" : Cho phép screen reader đọc

133.5. Giá trị thường dùng

- true

- false

(chỉ dùng 2 giá trị này, KHÔNG dùng số, KHÔNG dùng chữ khác)

133.6. Khi nào NÊN dùng

- Icon trang trí (, ❌, ⭐)

- Text lặp lại không cần đọc

- Element chỉ để trang trí giao diện

133.7. Khi nào KHÔNG NÊN dùng

- KHÔNG dùng cho:

+ input

+ button

+ link

+ form cần thao tác

❌ Không bắt buộc có aria-hidden

❌ Không ảnh hưởng chạy

✅ Có thì gọn, không có cũng sống khỏe

133.8. So sánh nhanh (dễ nhầm)

- display: none

→ Ẩn luôn với MẮT + screen reader

- aria-hidden="true"

→ CHỈ ẩn với screen reader

→ Người thường VẪN THẤY

133.9. Ký hiệu / ngữ nghĩa cần nhớ

- "aria" = accessibility

- Không phải CSS

- Không thay thế hidden, display, visibility

133.10. Lỗi hay gặp

- Dùng aria-hidden="true" cho button → sai

- Nghĩ aria-hidden là ẩn giao diện → sai

- Quên xóa aria-hidden khi element cần tương tác

133.11.EXAMBLE XEM CHƠI

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP UI Test</title>

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

  <style>

    body {

      font-family: sans-serif;

      text-align: center;

      padding: 20px;

    }

    button {

      padding: 15px 25px;

      font-size: 18px;

    }

    .status {

      margin-top: 20px;

      font-size: 20px;

    }

  </style>

</head>

<body>

  <h2>Điều khiển ESP</h2>

  <button onclick="toggle()">BẬT / TẮT</button>

  <div class="status">

    <!-- ICON CHỈ ĐỂ NHÌN -->

    <span aria-hidden="true">💡</span>

    <span id="text">Đang TẮT</span>

  </div>

  <script>

    let on = false;

    function toggle() {

      on = !on;

      document.getElementById("text").innerText =

        on ? "Đang BẬT" : "Đang TẮT";

    }

  </script>

</body>

</html>

<!--

###

<button onclick="toggle()">BẬT / TẮT</button>

<div class="status">

  <span aria-hidden="true">💡</span>

  <span id="text">Đang TẮT</span>

</div>

###

BAN ĐẦU (khi trang vừa load):

- Biến on trong JavaScript có giá trị:

on = false

→ đại diện cho trạng thái ESP đang TẮT

- Trên giao diện:

Nút hiển thị chữ: "BẬT / TẮT"

Dòng trạng thái hiển thị: "Đang TẮT"

- Icon :

có aria-hidden="true"

→ CHỈ để nhìn

→ máy đọc màn hình BỎ QUA icon này

→ tránh đọc linh tinh, UI gọn hơn

- Người dùng CHƯA bấm gì

- JavaScript CHƯA chạy hàm toggle()

--------------------------------------------------

##

<button onclick="toggle()">BẬT / TẮT</button>

##

KHI NGƯỜI DÙNG BẤM NÚT:

→ Trình duyệt phát hiện sự kiện click

→ Gọi hàm toggle()

→ JavaScript bắt đầu xử lý logic bên trong

--------------------------------------------------

###

let on = false;

###

Ý NGHĨA BIẾN on:

- on KHÔNG phải ESP thật

- on là BIẾN GIẢ LẬP trạng thái

- Dùng để:

+ nhớ đang BẬT hay TẮT

+ quyết định hiển thị chữ gì trên giao diện

--------------------------------------------------

###

function toggle(){

on = !on;

}

###

DÒNG NÀY LÀ TRUNG TÂM:

- !on nghĩa là: ĐẢO TRẠNG THÁI

NẾU TRƯỚC ĐÓ:

- on = false

→ !on = true

NẾU TRƯỚC ĐÓ:

- on = true

→ !on = false

→ mỗi lần bấm nút:

trạng thái luôn đổi ngược lại

--------------------------------------------------

###

document.getElementById("text").innerText = on ? "Đang BẬT" : "Đang TẮT";

###

SAU KHI ĐÃ ĐỔI TRẠNG THÁI on:

- JavaScript tìm phần tử:

id="text"

- innerText:

là nội dung chữ HIỂN THỊ trên màn hình

- Toán tử ? : (điều kiện nhanh)

NẾU:

- on === true

→ hiển thị: "Đang BẬT"

NẾU:

- on === false

→ hiển thị: "Đang TẮT"

--------------------------------------------------

===== KẾT QUẢ SAU LẦN BẤM THỨ NHẤT =====

- on: false → true

- Chữ trạng thái: "Đang BẬT"

- Icon 💡 vẫn hiện

- Screen reader:

+ đọc "Đang BẬT"

+ KHÔNG đọc icon 💡

--------------------------------------------------

🔁 NGƯỜI DÙNG BẤM LẦN THỨ HAI

- toggle() chạy lại từ đầu

- on: true → false

- innerText đổi thành: "Đang TẮT"

--------------------------------------------------

===== TÓM LẠI =====

- JavaScript:

QUYẾT ĐỊNH LOGIC (bật / tắt)

- HTML:

HIỂN THỊ GIAO DIỆN

- aria-hidden="true":

KHÔNG làm thay đổi giao diện

CHỈ nói cho MÁY BIẾT:

"cái này trang trí thôi, đừng đọc"

- Đây là mẫu UI RẤT CHUẨN cho:

+ ESP8266

+ ESP32

+ Web điều khiển thiết bị IoT

-->

=== === === === === THUỘC TÍNH: aria-invalid === === === === ===

134. Thuộc tính aria-invalid

→ DÙNG ĐỂ BÁO LỖI CHO TRÌNH ĐỌC MÀN HÌNH (accessibility)

--------------------------------------------------

134.1. Dùng để làm gì?

- Báo cho máy đọc màn hình biết: ô này ĐANG NHẬP SAI

--------------------------------------------------

134.2. Bản chất (hiểu đơn giản)

- aria-invalid KHÔNG làm web tự báo lỗi

- Nó chỉ là “THÔNG BÁO TRẠNG THÁI” cho thiết bị hỗ trợ

--------------------------------------------------

134.3. Phạm vi áp dụng

- Thuộc nhóm ARIA

- Dùng nhiều cho:

input, textarea, select, form-related element

--------------------------------------------------

134.4. Cú pháp (PHẦN QUAN TRỌNG – GHI KỸ)

<tag aria-invalid="true"></tag>

Ví dụ thường gặp:

<input aria-invalid="true">

<input aria-invalid="false">

--------------------------------------------------

134.5. Giá trị thường dùng

- "true" → đang có lỗi

- "false" → hợp lệ

- "grammar" / "spelling" (ít dùng)

--------------------------------------------------

134.6. Kết hợp QUAN TRỌNG (rất hay dùng chung)

- aria-describedby → trỏ tới dòng mô tả lỗi

- aria-required → báo trường bắt buộc

--------------------------------------------------

134.7. Lưu ý / lỗi hay gặp

- aria-invalid KHÔNG thay thế validate thật (JS / backend)

- Chỉ dùng khi THỰC SỰ có lỗi

- Đừng gắn true suốt, máy đọc sẽ báo sai liên tục

--------------------------------------------------

134.8. Ký hiệu / ý nghĩa cần nhớ

aria = hỗ trợ truy cập

invalid = không hợp lệ

→ aria-invalid = báo phần tử KHÔNG HỢP LỆ

--------------------------------------------------

134.9. EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test aria-invalid ESP</title>

  <style>

    body {

      background: #111;

      color: #0f0;

      font-family: monospace;

      padding: 20px;

    }

    input {

      padding: 8px;

      font-size: 16px;

      background: #000;

      color: #0f0;

      border: 2px solid #0f0;

    }

    input[aria-invalid="true"] {

      border-color: red;

    }

    .error {

      color: red;

      display: none;

      margin-top: 6px;

    }

  </style>

</head>

<body>

  <h3>ESP Form Test</h3>

  <input id="name" type="text" placeholder="Nhập ít nhất 5 ký tự" aria-invalid="false" aria-describedby="err" >

  <div id="err" class="error">❌ Dữ liệu không hợp lệ</div>

  <script>

    const input = document.getElementById("name");

    const err = document.getElementById("err");

    input.addEventListener("input", () => {

      if (input.value.length < 5) {

        input.setAttribute("aria-invalid", "true");

        err.style.display = "block";

      } else {

        input.setAttribute("aria-invalid", "false");

        err.style.display = "none";

      }

    });

  </script>

</body>

</html>

###

<input id="name"

       type="text"

       placeholder="Nhập ít nhất 5 ký tự"

       aria-invalid="false"

       aria-describedby="err" />

<div id="err" class="error">❌ Dữ liệu không hợp lệ</div>

###

BAN ĐẦU – KHI TRANG VỪA LOAD (CHƯA AI ĐỤNG GÌ):

- Trình duyệt đọc HTML → tạo DOM (cây đối tượng trong RAM)

HTML ĐANG CÓ GÌ:

- id="name"

  → ĐỊNH DANH DUY NHẤT

  → JS dùng id để tìm đúng phần tử

- type="text"

  → ô nhập CHỮ

- placeholder

  → chữ gợi ý (KHÔNG phải dữ liệu thật)

- aria-invalid="false"

  → báo cho máy đọc màn hình:

    ô này ĐANG HỢP LỆ

- aria-describedby="err"

  → nếu có lỗi:

    máy đọc màn hình sẽ đọc nội dung thẻ id="err"

- div.error:

  display: none

  → lỗi BỊ ẨN

  → người dùng KHÔNG THẤY

TRẠNG THÁI LÚC NÀY:

- UI: bình thường

- Máy đọc màn hình: không báo lỗi

- Người dùng: chưa nhập gì

===

DOM LÀ GÌ?

- DOM = bản sao HTML trong bộ nhớ

- JS thao tác DOM → web thay đổi NGAY

===

###

const input = document.getElementById("name");

const err   = document.getElementById("err");

###

GIẢI THÍCH TỪNG PHẦN:

- const

  → khai báo biến KHÔNG đổi tham chiếu

- document

  → toàn bộ trang HTML

- getElementById("name")

  → tìm phần tử có id="name" trong DOM

KẾT QUẢ:

- input = ô nhập

- err   = dòng báo lỗi

=> JS ĐÃ "NẮM" ĐƯỢC HTML

===

JS KHÔNG SỬA HTML GỐC

JS SỬA DOM → TRÌNH DUYỆT TỰ VẼ LẠI

===

###

input.addEventListener("input", () => {

###

GIẢI THÍCH:

- addEventListener

  → đăng ký nghe sự kiện

- "input"

  → kích hoạt NGAY KHI:

    gõ, xoá, dán, sửa

- () => { }

  → arrow function

  → hàm chạy khi sự kiện xảy ra

=> TỪ GIỜ:

- MỖI lần người dùng thay đổi nội dung

- JS chạy lại TOÀN BỘ code bên trong

=====

NGƯỜI DÙNG BẮT ĐẦU GÕ PHÍM

=====

###

if (input.value.length < 5) {

###

GIẢI THÍCH CHI TIẾT:

- input.value

  → LẤY DỮ LIỆU người dùng đã nhập

- .length

  → ĐẾM SỐ KÝ TỰ

- <

  → toán tử SO SÁNH "nhỏ hơn"

- if (...)

  → nếu điều kiện ĐÚNG thì chạy bên trong

TRƯỜNG HỢP 1: NHẬP < 5 KÝ TỰ

- Điều kiện ĐÚNG

- JS hiểu:

  DỮ LIỆU SAI

###

input.setAttribute("aria-invalid", "true");

###

GIẢI THÍCH:

- setAttribute(tên, giá trị)

  → GẮN / ĐỔI thuộc tính HTML

- aria-invalid="true"

  → báo cho MÁY:

    ô nhập ĐANG LỖI

LƯU Ý:

- aria-invalid KHÔNG kiểm tra dữ liệu

- CHỈ ghi nhận TRẠNG THÁI

###

err.style.display = "block";

###

GIẢI THÍCH:

- err.style

  → truy cập CSS inline

- display = "block"

  → HIỆN phần tử

=> Người dùng THẤY lỗi

=> Máy đọc màn hình BIẾT lỗi

KẾT QUẢ KHI SAI:

- Viền input: ĐỎ (CSS selector)

- aria-invalid="true"

- UI + accessibility đồng bộ

###

} else {

###

TRƯỜNG HỢP 2: NHẬP ≥ 5 KÝ TỰ

- Điều kiện SAI

- JS hiểu:

  DỮ LIỆU ĐÚNG

###

input.setAttribute("aria-invalid", "false");

###

- aria-invalid chuyển về false

- Báo cho máy:

  ô nhập hợp lệ

###

err.style.display = "none";

###

- display: none

- Dòng lỗi BIẾN MẤT

KẾT QUẢ KHI ĐÚNG:

- Viền input: XANH

- aria-invalid="false"

- UI sạch

- Máy đọc màn hình không báo lỗi

=====

CSS ĐANG LÀM GÌ?

- input[aria-invalid="true"]

  → selector theo TRẠNG THÁI

- aria đổi → CSS đổi theo

=====

=====

CHỐT KIẾN THỨC CỐT LÕI:

- HTML: cấu trúc

- DOM: bản sao HTML trong bộ nhớ

- JS: quyết định logic ĐÚNG / SAI

- CSS: quyết định HIỂN THỊ

- ARIA: giúp MÁY HIỂU TRẠNG THÁI

- ESP: chỉ trả HTML, KHÔNG xử lý mấy thứ này

=====

=== === === === === HẾT === === === === ===

=== === === === === THUỘC TÍNH: aria-label === === === === === [RQR trong menu]

135. Thuộc tính aria-label

→ DÒNG NÀY QUAN TRỌNG:

aria-label = ĐẶT TÊN CHO PHẦN TỬ ĐỂ MÁY ĐỌC HIỂU

--------------------------------------------------

135.1. Bản chất (dùng để làm gì?)

- Gán TÊN MÔ TẢ cho phần tử (cho máy đọc màn hình)

--------------------------------------------------

135.2. Phạm vi áp dụng

- Thuộc nhóm ARIA

- Áp dụng cho:

  button, input, icon, link, các phần tử KHÔNG có chữ rõ ràng

--------------------------------------------------

135.3. Cú pháp (PHẦN QUAN TRỌNG – CẦN GHI NHỚ)

<tag aria-label="mô tả chức năng"></tag>

LƯU Ý:

- Nội dung trong aria-label là CHỮ

- Máy đọc màn hình sẽ ĐỌC CHỮ NÀY

--------------------------------------------------

135.4. Giá trị thường dùng

- Chuỗi (string)

- Viết rõ CHỨC NĂNG

  ví dụ: "Mở menu", "Đóng cửa sổ", "Tìm kiếm"

--------------------------------------------------

135.5. Khi nào CẦN dùng aria-label

- Nút chỉ có ICON (không có chữ)

- Input KHÔNG có label hiển thị

- Các nút điều khiển (X, ☰, 🔍…)

--------------------------------------------------

135.6. Khi nào KHÔNG cần

- Khi đã có chữ rõ ràng trong nút

- Khi đã có <label> liên kết đúng

--------------------------------------------------

135.7. Lưu ý / lỗi hay gặp

- Không nhồi chữ dài dòng

- Không mô tả mơ hồ (vd: "button")

- aria-label KHÔNG hiển thị ra màn hình

- aria-label KHÔNG thay thế text cho người nhìn

--------------------------------------------------

135.8. Ký hiệu / ngữ nghĩa cần nhớ

aria = hỗ trợ truy cập

label = nhãn / tên gọi

→ aria-label = TÊN GỌI CHO MÁY ĐỌC

--------------------------------------------------

135.9. So sánh nhanh (rất hay nhầm)

- label (HTML): cho người NHÌN

- aria-label: cho MÁY ĐỌC

- aria-describedby: mô tả CHI TIẾT hơn

--------------------------------------------------

135.10. Ghi nhớ nhanh (1 câu)

aria-label = đặt tên cho phần tử để MÁY hiểu đúng chức năng

135.11.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Menu xổ ra</title>

  <style>

    body {

      background: #111;

      color: #0f0;

      font-family: monospace;

      padding: 20px;

    }

    button {

      font-size: 24px;

      padding: 8px 14px;

      background: #000;

      color: #0f0;

      border: 2px solid #0f0;

      cursor: pointer;

    }

    #menu {

      margin-top: 10px;

      border: 1px solid #0f0;

      padding: 10px;

    }

  </style>

</head>

<body>

  <button id="btn" aria-label="Mở menu cài đặt" aria-controls="menu" aria-expanded="false">

    ≡

  </button>

  <div id="menu" hidden>

    <div>⚙ Cài đặt</div>

    <div>📶 WiFi</div>

    <div>🔄 Reset</div>

  </div>

  <script>

    const btn = document.getElementById("btn");

    const menu = document.getElementById("menu");

    btn.onclick = () => {

      const dangMo = btn.ariaExpanded === "true";

      menu.hidden = dangMo;

      btn.ariaExpanded = String(!dangMo);

    };

  </script>

</body>

</html>

<!--

###

<button id="btn"

        aria-label="Mở menu cài đặt"

        aria-controls="menu"

        aria-expanded="false">

  ≡

</button>

<div id="menu" hidden>

  <div>⚙ Cài đặt</div>

  <div>📶 WiFi</div>

  <div>🔄 Reset</div>

</div>

###

BAN ĐẦU – KHI TRANG VỪA LOAD (CHƯA AI BẤM GÌ):

TRÌNH DUYỆT LÀM GÌ:

- Đọc HTML → tạo DOM (cây phần tử trong bộ nhớ)

- Đọc CSS → ghi nhớ cách HIỂN THỊ

- Chưa chạy JS onclick

TRẠNG THÁI HTML:

- button:

  id="btn"

  → ĐỊNH DANH DUY NHẤT cho JS

  aria-label="Mở menu cài đặt"

  → ĐẶT TÊN cho nút để MÁY đọc màn hình hiểu

  → KHÔNG hiện ra UI

  aria-controls="menu"

  → nói rằng:

    nút này ĐIỀU KHIỂN thằng có id="menu"

  aria-expanded="false"

  → trạng thái ban đầu: ĐANG ĐÓNG

- icon ≡

  → chỉ là HÌNH, KHÔNG có logic

- div#menu:

  hidden

  → bị ẨN HOÀN TOÀN

  → display giống như: none

  → người dùng KHÔNG THẤY

=> KẾT LUẬN BAN ĐẦU:

- Menu: ĐÓNG

- aria-expanded: false

- UI: gọn gàng

- Máy đọc màn hình: biết menu đang đóng

===

DOM LÀ GÌ?

- DOM = bản sao HTML trong RAM

- JS thao tác DOM → giao diện đổi NGAY

===

###

const btn = document.getElementById("btn");

const menu = document.getElementById("menu");

###

GIẢI THÍCH TỪNG KÝ TỰ:

- const

  → tạo biến KHÔNG đổi tham chiếu

- document

  → toàn bộ trang web

- getElementById("btn")

  → tìm phần tử có id="btn" trong DOM

KẾT QUẢ:

- btn   → TRỎ TỚI cái button

- menu → TRỎ TỚI cái div menu

=> JS đã "CẦM ĐƯỢC" HTML

===

DẤU CHẤM (.)

- nghĩa là: ĐI VÀO BÊN TRONG đối tượng

- btn.onclick = truy cập thuộc tính onclick của btn

===

###

btn.onclick = () => {

###

Ý NGHĨA:

- onclick

  → sự kiện CLICK

- = () => { }

  → gán 1 HÀM cho sự kiện

=> KHI NGƯỜI DÙNG BẤM NÚT

→ hàm bên trong chạy

=====

NGƯỜI DÙNG BẤM NÚT MENU (≡)

=====

###

const dangMo = btn.ariaExpanded === "true";

###

GIẢI THÍCH CỰC KỸ:

- btn.ariaExpanded

  → đọc GIÁ TRỊ aria-expanded hiện tại

  → trả về CHUỖI "true" hoặc "false"

- ===

  → toán tử SO SÁNH TUYỆT ĐỐI

  → so cả GIÁ TRỊ + KIỂU DỮ LIỆU

- "true"

  → CHUỖI, không phải boolean

SO SÁNH:

- LÚC ĐẦU:

  aria-expanded = "false"

- "false" === "true"

  → SAI

=> dangMo = false

Ý NGHĨA BIẾN dangMo:

- true  → menu đang mở

- false → menu đang đóng

###

menu.hidden = dangMo;

###

GIẢI THÍCH TỪNG PHẦN:

- menu.hidden

  → thuộc tính HTML hidden

- =

  → GÁN GIÁ TRỊ

- dangMo = false

→ menu.hidden = false

Ý NGHĨA:

- hidden = false

→ KHÔNG ẨN

→ MENU HIỆN RA

===

TẠI SAO JS ĐỔI ĐƯỢC STYLE?

- hidden là THUỘC TÍNH HTML

- trình duyệt tự đổi cách hiển thị

- JS không cần viết CSS

===

###

    btn.onclick = () => {

      const dangMo = btn.ariaExpanded === "true";

      menu.hidden = dangMo;

      btn.ariaExpanded = String(!dangMo);

    };

###

GIẢI THÍCH CHI TIẾT:

✅ Dòng 4 dùng KẾT QUẢ đã lưu ở biến dangMo (từ dòng 2)

👉 rồi ĐẢO NGƯỢC nó, sau đó ép  boolean sang chuỗi và GHI VÀO btn.ariaExpanded

- !

  → toán tử PHỦ ĐỊNH

  → đảo ngược giá trị

- dangMo = false

- !false = true

- String(true)

  → chuyển boolean → chuỗi "true"

=> aria-expanded:

- từ "false"

- thành "true"

Ý NGHĨA:

- Menu ĐANG MỞ

- Máy đọc màn hình BIẾT TRẠNG THÁI MỚI

=====

KẾT QUẢ SAU LẦN BẤM ĐẦU:

- Menu: HIỆN

- aria-expanded="true"

- UI + accessibility ĐỒNG BỘ

=====

🔁 NGƯỜI DÙNG BẤM LẦN 2

- aria-expanded lúc này = "true"

SO SÁNH:

- "true" === "true" → ĐÚNG

→ dangMo = true

menu.hidden = true

→ MENU BỊ ẨN

aria-expanded = "false"

→ trạng thái quay về ĐÓNG

=====

CHỐT LẠI BẢN CHẤT (CỰC QUAN TRỌNG):

- HTML: tạo khung

- DOM: bản sao để JS thao tác

- JS: quyết định MỞ / ĐÓNG

- hidden: điều khiển HIỂN THỊ

- aria-expanded: MÔ TẢ TRẠNG THÁI

- aria-label: ĐẶT TÊN CHO MÁY

=====

=====

1 CÂU NHỚ LÂU:

ARIA KHÔNG LÀM MENU XỔ RA

JS LÀM

ARIA CHỈ GIÚP MENU "CÓ NÃO"

=====

-->

=== === === === === HẾT === === === === ===

=== === === === === THUỘC TÍNH: aria-labelledby === === === === === [ CẦN XEM LẠI CHƯA HIỂU RÕ LẮM]

136. Thuộc tính aria-labelledby

QUAN TRỌNG: Thuộc tính này GIÚP MÁY ĐỌC MÀN HÌNH BIẾT “PHẦN TỬ NÀY ĐANG ĐƯỢC GỌI TÊN LÀ GÌ”

136.1. Bản chất (dùng để làm gì)

- Dùng để GÁN TÊN / MÔ TẢ cho một phần tử

- Tên này KHÔNG hiển thị cho mắt người

- Chỉ dành cho TRÌNH ĐỌC MÀN HÌNH (accessibility)

NÓI NGẮN GỌN:

→ aria-labelledby = “mày tên là cái thằng có id kia”

---

136.2. Phạm vi áp dụng

- Thuộc nhóm: ARIA

- Áp dụng cho:

+ button

+ input

+ menu

+ dialog

+ div (khi đóng vai trò UI)

---

136.3. Cú pháp (RẤT QUAN TRỌNG – GHI NHỚ KỸ)

- Giá trị của aria-labelledby = ID của phần tử KHÁC

- ID đó PHẢI TỒN TẠI trong DOM

DẠNG TỔNG QUÁT:

aria-labelledby="tenId"

HIỂU BẢN CHẤT:

- aria-labelledby KHÔNG tự tạo chữ

- Nó CHỈ TRỎ sang 1 thằng khác có id

---

136.4. Giá trị thường dùng

- Chuỗi (string)

- Là ID (không #)

- Có thể là:

+ 1 id

+ nhiều id (cách nhau bằng dấu cách)

VÍ DỤ Ý NGHĨA:

- "title"

- "label-name"

- "menu-title info-text"

---

136.5. Cách HOẠT ĐỘNG THỰC TẾ (RẤT QUAN TRỌNG)

KHI TRÌNH ĐỌC MÀN HÌNH GẶP:

- Nó nhìn aria-labelledby

- Nó tìm phần tử có id TRÙNG

- Nó ĐỌC NỘI DUNG của phần tử đó

- XEM ĐÓ LÀ TÊN của phần tử hiện tại

👉 aria-labelledby = MƯỢN CHỮ CỦA THẰNG KHÁC

---

136.6. So sánh nhanh với aria-label (ĐỂ KHỎI NHẦM)

aria-label:

- Ghi trực tiếp chữ vào thuộc tính

- Không cần id

- Dùng khi KHÔNG có text sẵn

aria-labelledby:

- LẤY CHỮ từ phần tử khác

- BẮT BUỘC dùng id

- Dùng khi TRÊN GIAO DIỆN ĐÃ CÓ CHỮ

NHỚ 1 CÂU:

→ Có chữ sẵn → dùng aria-labelledby

→ Không có chữ → dùng aria-label

---

136.7. Lưu ý / lỗi hay gặp

- ❌ id không tồn tại → vô dụng

- ❌ id bị trùng → máy đọc sai

- ❌ dùng aria-labelledby nhưng phần tử kia bị hidden → máy không đọc

- ✅ nên dùng khi label / tiêu đề ĐÃ HIỂN THỊ

---

136.8. Ký hiệu / ngữ nghĩa cần nhớ

- aria = dành cho accessibility

- labelledby = được đặt tên bởi

→ aria-labelledby = “được gọi tên bởi ai”

---

136.9. CÂU CHỐT ĐỂ NHỚ LÂU

aria-labelledby KHÔNG PHẢI để style

aria-labelledby KHÔNG PHẢI cho người nhìn

aria-labelledby = GIAO TIẾP GIỮA UI & MÁY

136.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test aria-labelledby</title>

  <style>

    body {

      background: #111;

      color: #0f0;

      font-family: monospace;

      padding: 20px;

    }

    button {

      font-size: 22px;

      padding: 8px 14px;

      background: #000;

      color: #0f0;

      border: 2px solid #0f0;

      cursor: pointer;

    }

    #menu {

      margin-top: 10px;

      border: 1px solid #0f0;

      padding: 10px;

    }

  </style>

</head>

<body>

  <!-- THẰNG NÀY LÀ “TÊN” -->

  <div id="menuTitle">Menu cài đặt thiết bị</div>

  <!-- NÚT MƯỢN TÊN TỪ THẰNG TRÊN -->

  <button id="btn" aria-labelledby="menuTitle" aria-controls="menu" aria-expanded="false">

    ≡

  </button>

  <!-- MENU -->

  <div id="menu" hidden>

    <div>⚙ Cài đặt</div>

    <div>📶 WiFi</div>

    <div>🔄 Reset</div>

  </div>

  <script>

    const btn = document.getElementById("btn");

    const menu = document.getElementById("menu");

    btn.onclick = () => {

      const dangMo = btn.ariaExpanded === "true";

      menu.hidden = dangMo;

      btn.ariaExpanded = String(!dangMo);

    };

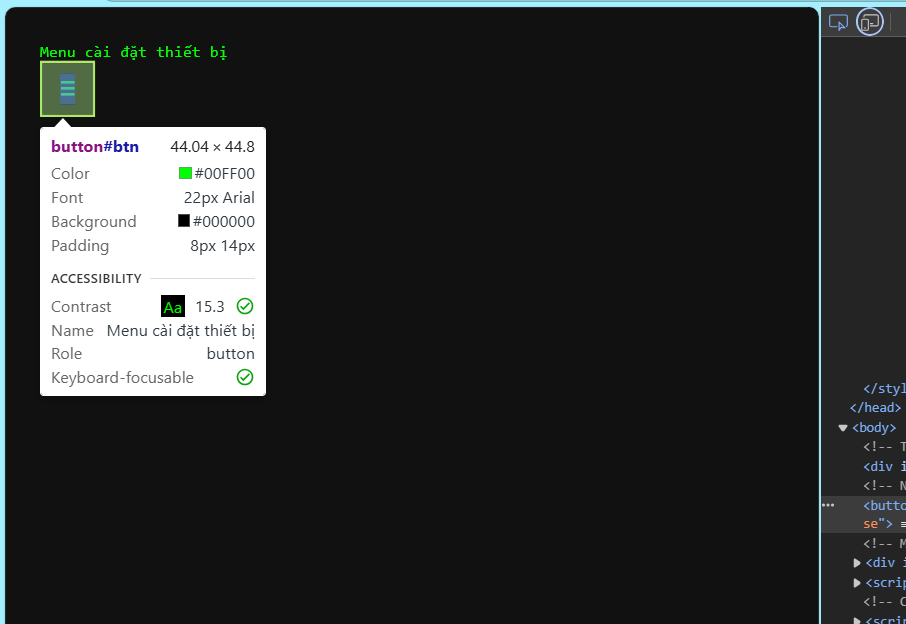
  </script>

</body>

</html>

Ý NGHĨA CỦA aria-labelledby="menuTitle" VỚI THẺ  <div id="menuTitle">Menu cài đặt thiết bị</div>

LÀ ĐÂY KHI RE CHUẬT VÀO ≡



=== HẾT 136 ===

=== === === === === THUỘC TÍNH: aria-live === === === === ===

137. Thuộc tính aria-live

→ QUAN TRỌNG: thông báo cho biết lên màn hình BIẾT nội dung đang thay đổi tự động

137.1. Bản chất (dùng để làm gì)

- Báo cho trình đọc màn hình rằng: “NỘI DUNG NÀY CÓ THAY ĐỔI”

- Dùng cho thông báo, trạng thái, kết quả cập nhật KHÔNG cần người dùng bấm

137.2. Phạm vi áp dụng

- ARIA attribute

- Dùng cho hầu hết thẻ HTML (div, span, p, section…)

137.3. Cú pháp (PHẦN RẤT QUAN TRỌNG)

aria-live="giá\_trị"

→ Gắn TRỰC TIẾP vào phần tử có nội dung thay đổi

→ Không cần JS đặc biệt, chỉ cần nội dung đổi là screen reader đọc

137.4. Giá trị thường dùng (CẦN NHỚ)

- off : Tắt thông báo (mặc định)

- polite : Đọc khi rẢNH (không ngắt lời) ✅ hay dùng nhất

- assertive : Đọc NGAY LẬP TỨC (ngắt lời) ⚠ chỉ dùng khi thật cần

137.5. Khi nào nên dùng

- Thông báo “Đã lưu thành công”

- Báo lỗi form

- Trạng thái đang tải / hoàn tất

- Kết quả tìm kiếm cập nhật

137.6. Khi KHÔNG nên dùng

- Nội dung thay đổi liên tục (đếm số, animation)

- Nội dung không quan trọng với người dùng

137.7. Lỗi hay gặp (RẤT DỄ DÍNH)

- Lạm dụng assertive → gây khó chịu

- Gắn aria-live vào phần tử KHÔNG thay đổi nội dung

- Dùng aria-live cho trang trí (sai mục đích)

137.8. Ký hiệu / thuật ngữ cần biết

- Screen reader : Trình đọc màn hình cho người khiếm thị

- Live region : Vùng nội dung động

- Accessibility : Khả năng tiếp cận (A11y)

137.9. Ghi nhớ nhanh (1 dòng)

→ aria-live = “Nội dung này đổi, nhớ đọc cho người dùng nghe”

137.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test aria-live polite</title>

</head>

<body>

  <h1>aria-live</h1>

  <button onclick="thongBao()">Lưu dữ liệu</button>

  <div id="msg" aria-live="polite" style="margin-top:10px; color:green;">

  </div>

  <script>

    function thongBao() {

      document.getElementById("msg").textContent =

        "✔ Đã lưu thành công...!";

    }

  </script>

</body>

</html>

<!--

textContent = thay chữ đang hiện trên màn hình

-->

=== === === === === HẾT === === === === ===

=== === === === === THUỘC TÍNH: aria-modal === === === === ===

138. Thuộc tính aria-modal

→ QUAN TRỌNG: Báo rằng “đây là HỘP THOẠI MODAL – người dùng PHẢI xử lý xong mới làm việc khác”



Bấm ok code mới chạy tiếp

138.1. Bản chất (dùng để làm gì)

- Khóa tương tác nền phía sau

- Giúp screen reader biết: chỉ tập trung vào hộp thoại này

138.2. Phạm vi áp dụng

- ARIA attribute

- Thường dùng cho: dialog, popup, modal

138.3. Cú pháp (PHẦN RẤT QUAN TRỌNG – CẦN CHUẨN)

aria-modal="true"

→ Chỉ dùng khi popup/modal ĐANG MỞ

→ PHẢI đi kèm role="dialog" hoặc role="alertdialog"

138.4. Giá trị thường dùng

- true : Là modal thật (chặn nền) ✅

- false : Không phải modal (ít dùng)

138.5. Khi nào NÊN dùng

- Hộp xác nhận (OK / Cancel)

- Cảnh báo quan trọng

- Form bắt buộc nhập xong mới tiếp tục

138.6. Khi nào KHÔNG nên dùng

- Tooltip

- Dropdown

- Popup nhẹ không chặn nền

138.7. Lỗi hay gặp (RẤT DỄ SAI)

- Dùng aria-modal nhưng KHÔNG có role="dialog"

- Dùng aria-modal cho popup nhỏ

- Quên quản lý focus (tab vẫn nhảy ra nền)

138.8. Ký hiệu / thuật ngữ cần biết

- Modal : Hộp thoại chặn toàn bộ nền

- Focus trap : Con trỏ chỉ được ở trong modal

- Dialog : Hộp thoại tương tác

138.9. Ghi nhớ nhanh (1 dòng)

→ aria-modal = “Đang mở hộp quan trọng, đừng cho người dùng làm gì khác”

138.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test aria-modal</title>

  <style>

    body {

      font-family: Arial;

    }

    /\* nền mờ \*/

    .overlay {

      position: fixed;

      inset: 0;

      background: rgba(0, 0, 0, 0.4);

      display: none;

    }

    /\* hộp modal \*/

    .modal {

      background: white;

      width: 300px;

      padding: 20px;

      margin: 100px auto;

      border-radius: 6px;

    }

  </style>

</head>

<body>

  <h2>Trang chính</h2>

  <button onclick="openModal()">Mở modal</button>

  <div class="overlay" id="overlay">

    <div class="modal" role="dialog" aria-modal="true" aria-labelledby="title">

      <h3 id="title">Xác nhận</h3>

      <p>Bạn có chắc muốn lưu?</p>

      <button onclick="closeModal()">OK</button>

      <button onclick="closeModal()">Hủy</button>

    </div>

  </div>

  <script>

    function openModal() {

      document.getElementById("overlay").style.display = "block";

    }

    function closeModal() {

      document.getElementById("overlay").style.display = "none";

    }

  </script>

</body>

</html>

###

GIẢI THÍCH TOÀN BỘ CƠ CHẾ aria-modal

(DIỄN GIẢI – GHI CHÚ – KHÔNG PHẢI CODE)

###

==================================================

GIAI ĐOẠN 1 – KHI TRANG VỪA LOAD (CHƯA BẤM GÌ)

==================================================

- Trình duyệt đọc HTML từ trên xuống

- CSS được áp dụng để tạo giao diện

- JavaScript được nạp nhưng CHƯA chạy hàm nào

NGƯỜI DÙNG THẤY TRÊN MÀN HÌNH:

- Dòng chữ: "Trang chính"

- Một nút bấm: "Mở modal"

KHÔNG THẤY:

- nền mờ

- hộp xác nhận

LÝ DO:

- phần tử overlay đang có:

  display = "none"

HIỂU ĐƠN GIẢN:

- overlay + modal ĐÃ CÓ SẴN TRONG TRANG

- nhưng bị CSS giấu đi

==================================================

GIAI ĐOẠN 2 – DOM LÀ GÌ TRONG TRƯỜNG HỢP NÀY

==================================================

- DOM là bản đồ toàn bộ trang web

- Mỗi thẻ HTML là 1 object trong DOM

- JavaScript KHÔNG thao tác HTML gốc

- JavaScript CHỈ thao tác object DOM

Ví dụ:

- overlay = 1 object DOM

- modal = object con nằm bên trong overlay

==================================================

GIAI ĐOẠN 3 – CSS ĐANG LÀM GÌ

==================================================

.overlay:

- position: fixed

  → dính vào màn hình

- inset: 0

  → phủ kín toàn bộ viewport

- background: rgba(0,0,0,0.4)

  → tạo nền mờ

- display: none

  → ẨN HOÀN TOÀN

.modal:

- background: white

  → hộp trắng

- width / padding

  → kích thước

- margin auto

  → căn giữa màn hình

- border-radius

  → bo góc cho đẹp

KẾT LUẬN:

- overlay là lớp CHE

- modal là hộp NỘI DUNG

- JS quyết định KHI NÀO chúng xuất hiện

==================================================

GIAI ĐOẠN 4 – NGƯỜI DÙNG BẤM "MỞ MODAL"

==================================================

- Người dùng click nút

- Trình duyệt phát hiện sự kiện click

- Thuộc tính onclick được kích hoạt

onclick="openModal()"

→ trình duyệt gọi hàm openModal()

==================================================

GIAI ĐOẠN 5 – HÀM openModal() CHẠY TỪ ĐẦU ĐẾN CUỐI

==================================================

document.getElementById("overlay")

- document:

  → đại diện cho toàn bộ trang web

- getElementById:

  → tìm phần tử có id="overlay"

- KẾT QUẢ:

  → lấy được object overlay trong DOM

.style

- style là cầu nối JS ↔ CSS

- cho phép JS thay đổi CSS trực tiếp

.display = "block"

- display là thuộc tính CSS

- "block" nghĩa là HIỂN THỊ

KẾT QUẢ THỰC TẾ:

- overlay từ ẨN → HIỆN

- nền mờ xuất hiện

- modal bên trong hiện theo

==================================================

GIAI ĐOẠN 6 – ARIA HOẠT ĐỘNG NHƯ THẾ NÀO

==================================================

role="dialog"

- báo cho máy đọc màn hình:

  → đây là hộp thoại

aria-modal="true"

- báo:

  → đang ở trong modal

  → nội dung phía sau bị vô hiệu

aria-labelledby="title"

- nói với máy:

  → tiêu đề modal nằm ở id="title"

LƯU Ý CỰC QUAN TRỌNG:

- aria KHÔNG mở modal

- aria KHÔNG đóng modal

- aria CHỈ GHI NHẬN TRẠNG THÁI

==================================================

GIAI ĐOẠN 7 – NGƯỜI DÙNG BẤM "OK" HOẶC "HỦY"

==================================================

- Cả 2 nút đều có:

  onclick="closeModal()"

→ trình duyệt gọi hàm closeModal()

==================================================

GIAI ĐOẠN 8 – HÀM closeModal() CHẠY

==================================================

document.getElementById("overlay")

- lấy lại object overlay trong DOM

.style.display = "none"

- gán lại thuộc tính CSS

KẾT QUẢ:

- overlay bị ẨN

- modal biến mất

- quay lại trang chính

==================================================

GIAI ĐOẠN 9 – GIẢI THÍCH DẤU CHẤM (.)

==================================================

Ví dụ:

overlay.style.display = "block"

Hiểu theo chuỗi:

- overlay      → object HTML

- .style       → truy cập phần style

- .display     → thuộc tính display của CSS

- = "block"    → gán giá trị mới

DẤU CHẤM = ĐI SÂU VÀO BÊN TRONG OBJECT

==================================================

GIAI ĐOẠN 10 – VÌ SAO JS ĐIỀU KHIỂN ĐƯỢC CSS

==================================================

- CSS định nghĩa hình dạng

- JS thay đổi giá trị CSS

- Trình duyệt vẽ lại giao diện

JS KHÔNG VẼ

JS CHỈ RA LỆNH

==================================================

GIAI ĐOẠN 11 – LUỒNG HOẠT ĐỘNG (1 DÂY LIỀN)

==================================================

1. HTML tạo cấu trúc

2. CSS tạo giao diện

3. JS chờ sự kiện

4. Người dùng bấm

5. Hàm chạy

6. Style bị đổi

7. UI thay đổi

8. aria mô tả trạng thái

==================================================

GIAI ĐOẠN 12 – CHỐT GỌN NHẸ

==================================================

- Modal hiển thị nhờ JavaScript

- Modal đúng chuẩn nhờ aria-modal

- aria không điều khiển

- aria chỉ giúp máy hiểu

-->

=== === === === === HẾT === === === === ===

=== === === === === THUỘC TÍNH: aria-multiselectable === === === === ===

[ khá hiểu ]

139. Thuộc tính aria-multiselectable

(DÒNG QUAN TRỌNG CẦN NHỚ NGAY):

→ Báo cho MÁY biết: danh sách này CHO PHÉP CHỌN NHIỀU hay CHỈ 1

--------------------------------------------------

139.1. Bản chất (dùng để làm gì)

--------------------------------------------------

- Cho biết một danh sách / bảng / listbox:

→ có chọn ĐƯỢC NHIỀU MỤC hay không

- DÙNG CHO:

→ máy đọc màn hình

- KHÔNG làm thay đổi giao diện

- KHÔNG tự bật / tắt gì cả

TÓM GỌN 1–2 Ý:

→ mô tả khả năng chọn NHIỀU

→ giúp UI “có não” với thiết bị hỗ trợ

--------------------------------------------------

139.2. Phạm vi áp dụng

--------------------------------------------------

- Thuộc nhóm: ARIA

- Chỉ dùng cho các role dạng danh sách

THƯỜNG GẶP NHẤT:

- role="listbox"

- role="tree"

- role="grid"

❌ KHÔNG dùng bừa cho button, div thường

--------------------------------------------------

139.3. Cú pháp (PHẦN QUAN TRỌNG)

--------------------------------------------------

aria-multiselectable="true"

aria-multiselectable="false"

LƯU Ý:

- GIÁ TRỊ LÀ CHUỖI

- viết thường

- đặt TRÊN PHẦN TỬ CHA (container)

--------------------------------------------------

139.4. Giá trị thường dùng

--------------------------------------------------

- "true"

→ cho phép chọn NHIỀU mục cùng lúc

- "false"

→ chỉ được chọn 1 mục

→ (mặc định nếu không ghi)

--------------------------------------------------

139.5. Ý NGHĨA THỰC TẾ (HIỂU NGAY)

--------------------------------------------------

- true:

→ giống checkbox list

→ chọn nhiều dòng

- false:

→ giống radio button

→ chọn 1 là hủy cái trước

--------------------------------------------------

139.6. LƯU Ý / LỖI HAY GẶP

--------------------------------------------------

- aria-multiselectable:

❌ KHÔNG tự xử lý logic chọn

❌ KHÔNG thay đổi màu / style

✅ CHỈ GHI NHẬN KHẢ NĂNG

- BẮT BUỘC đi kèm:

→ aria-selected cho từng item

- Nếu quên aria-selected:

→ máy KHÔNG biết item nào đang chọn

--------------------------------------------------

139.7. MỐI QUAN HỆ QUAN TRỌNG

--------------------------------------------------

aria-multiselectable

→ trả lời câu hỏi: CÓ ĐƯỢC CHỌN NHIỀU KHÔNG?

aria-selected

→ trả lời câu hỏi: ITEM NÀY ĐANG ĐƯỢC CHỌN HAY KHÔNG?

2 THẰNG PHẢI ĐI CẶP

--------------------------------------------------

139.8. GHI NHỚ NHANH (KHỎI HỌC THUỘC)

--------------------------------------------------

- aria-multiselectable:

→ luật chơi của danh sách

- aria-selected:

→ trạng thái từng mục

- JavaScript:

→ là thằng THI HÀNH

--------------------------------------------------

139.9. SO SÁNH NGẮN GỌN

--------------------------------------------------

- aria-expanded:

→ mở / đóng

- aria-checked:

→ bật / tắt

- aria-multiselectable:

→ 1 hay NHIỀU

--------------------------------------------------

139.10. KẾT LUẬN 1 CÂU

--------------------------------------------------

aria-multiselectable

→ không làm UI thay đổi

→ chỉ nói cho MÁY biết:

"Danh sách này chọn được MẤY CÁI CÙNG LÚC"

139.11.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test aria-multiselectable</title>

  <style>

    .listbox {

      width: 200px;

      border: 1px solid #333;

      padding: 5px;

    }

    .item {

      padding: 6px;

      cursor: pointer;

    }

    .item[aria-selected="true"] {

      background: #4caf50;

      color: white;

    }

  </style>

</head>

<body>

  <h3>Chọn nhiều mục (test aria)</h3>

  <div class="listbox" role="listbox" aria-multiselectable="true">

    <div class="item" role="option" aria-selected="false">ESP8266</div>

    <div class="item" role="option" aria-selected="false">ESP32</div>

    <div class="item" role="option" aria-selected="false">Arduino</div>

  </div>

  <script>

    const items = document.querySelectorAll('.item');

    items.forEach(item => {

      item.onclick = () => {

        const selected = item.getAttribute('aria-selected');

        item.setAttribute(

          'aria-selected',

          selected === "true" ? "false" : "true"

        );

      };

    });

  </script>

</body>

</html>

<!--

### CODE ĐƯỢC NÊU

<div class="listbox" role="listbox" aria-multiselectable="true">

  <div class="item" role="option" aria-selected="false">ESP8266</div>

  <div class="item" role="option" aria-selected="false">ESP32</div>

  <div class="item" role="option" aria-selected="false">Arduino</div>

</div>

###

BAN ĐẦU (KHI TRANG VỪA LOAD):

→ Trình duyệt đọc HTML từ trên xuống dưới

→ Gặp thẻ div có class="listbox"

LÚC NÀY:

- Đây là 1 KHỐI chứa danh sách

- role="listbox" báo cho MÁY biết:

→ đây là danh sách có thể chọn

- aria-multiselectable="true" báo cho MÁY:

→ danh sách này CHO PHÉP CHỌN NHIỀU MỤC CÙNG LÚC

BÊN TRONG listbox:

- Có 3 div con

- Mỗi div là 1 mục chọn

- role="option" nghĩa là:

→ mỗi div này là 1 LỰA CHỌN

- aria-selected="false":

→ ban đầu KHÔNG CÓ MỤC NÀO ĐƯỢC CHỌN

KẾT QUẢ BAN ĐẦU:

- Người nhìn: thấy 3 dòng chữ

- Máy đọc màn hình: hiểu đây là listbox nhiều lựa chọn

- TẤT CẢ đang ở trạng thái CHƯA CHỌN

###

TRÌNH DUYỆT TIẾP TỤC ĐỌC ĐẾN JAVASCRIPT:

const items = document.querySelectorAll('.item');

Ý NGHĨA:

- document: toàn bộ trang HTML

- querySelectorAll('.item'):

→ tìm TẤT CẢ phần tử có class="item"

- Kết quả:

→ items là 1 DANH SÁCH (NodeList) gồm 3 div:

ESP8266, ESP32, Arduino

###

items.forEach(item => { ... });

Ý NGHĨA:

- forEach: duyệt TỪNG PHẦN TỬ một

- item:

→ đại diện cho TỪNG div trong danh sách

- Vòng lặp chạy 3 lần:

1️⃣ item = ESP8266

2️⃣ item = ESP32

3️⃣ item = Arduino

###

TRONG MỖI LẦN LẶP, CHƯƠNG TRÌNH GÁN SỰ KIỆN CLICK:

item.onclick = () => { ... };

Ý NGHĨA:

- Khi NGƯỜI DÙNG bấm vào item này

- Trình duyệt sẽ GỌI hàm arrow function

LƯU Ý:

- MỖI ITEM có 1 onclick RIÊNG

- KHÔNG LIÊN QUAN item khác

→ đây là lý do CHỌN ĐƯỢC NHIỀU CÙNG LÚC

###

KHI NGƯỜI DÙNG BẤM VÀO 1 MỤC (VD: ESP32):

→ Trình duyệt phát hiện click

→ GỌI hàm onclick của item đó

CHƯƠNG TRÌNH CHẠY DÒNG ĐẦU TIÊN:

const selected = item.getAttribute('aria-selected');

Ý NGHĨA:

- getAttribute:

→ ĐỌC GIÁ TRỊ thuộc tính HTML

- Đọc aria-selected hiện tại của item

TRƯỜNG HỢP LẦN ĐẦU:

- aria-selected = "false"

→ selected = "false"

###

TIẾP THEO, CHƯƠNG TRÌNH CHẠY DÒNG NÀY:

item.setAttribute(

'aria-selected',

selected === "true" ? "false" : "true"

);

GIẢI THÍCH TỪNG PHẦN:

- setAttribute:

→ GHI / ĐỔI giá trị thuộc tính HTML

- 'aria-selected':

→ thuộc tính cần thay đổi

- Biểu thức điều kiện ? :

selected === "true" ? "false" : "true"

ĐỌC THÀNH LỜI:

→ NẾU đang là "true" → đổi thành "false"

→ NGƯỢC LẠI → đổi thành "true"

###

TRƯỜNG HỢP LẦN ĐẦU BẤM:

- selected = "false"

- So sánh:

"false" === "true" → SAI

- Chạy nhánh sau dấu :

→ gán aria-selected = "true"

KẾT QUẢ:

- aria-selected đổi từ "false" → "true"

###

NGAY LẬP TỨC SAU KHI aria-selected ĐỔI:

CSS BẮT ĐẦU HOẠT ĐỘNG:

.item[aria-selected="true"] {

background: #4caf50;

color: white;

}

Ý NGHĨA:

- CSS đọc DOM

- Thấy item nào có aria-selected="true"

→ áp style màu xanh + chữ trắng

KẾT QUẢ:

- MỤC ĐƯỢC TÔ MÀU

- Người nhìn: biết đã chọn

- Máy đọc màn hình: biết mục đang selected

###

NGƯỜI DÙNG BẤM TIẾP MỤC KHÁC (VD: Arduino):

→ LẠI LẶP LẠI TOÀN BỘ QUY TRÌNH TRÊN

→ KHÔNG ẢNH HƯỞNG item trước

VÌ:

- aria-multiselectable="true"

→ cho phép NHIỀU item cùng selected

###

NẾU NGƯỜI DÙNG BẤM LẠI 1 MỤC ĐÃ CHỌN:

- aria-selected đang là "true"

- So sánh:

"true" === "true" → ĐÚNG

- Gán ngược lại:

aria-selected = "false"

- CSS mất hiệu lực

→ item trở về trạng thái bình thường

###

TÓM TẮT LOGIC TOÀN BỘ:

- HTML:

→ định nghĩa cấu trúc + ngữ nghĩa

- ARIA:

→ mô tả trạng thái cho MÁY

- JavaScript:

→ đọc trạng thái → quyết định đổi hay không

- CSS:

→ dựa vào aria-selected để đổi giao diện

- aria-multiselectable:

→ cho phép nhiều mục được chọn độc lập

###

GIẢI THÍCH CÁC THÀNH PHẦN QUAN TRỌNG (ĐỂ CUỐI – KHÔNG CẮT LUỒNG):

- role="listbox":

→ báo đây là danh sách lựa chọn

- role="option":

→ báo mỗi item là 1 lựa chọn

- aria-multiselectable="true":

→ CHÌA KHÓA cho phép chọn nhiều

- aria-selected:

→ ghi trạng thái CHỌN / KHÔNG CHỌN

→ KHÔNG tự đổi, JS phải đổi

- getAttribute():

→ đọc giá trị HTML hiện tại

- setAttribute():

→ ghi / ghi đè giá trị HTML

- item.style.xxx:

→ đổi style TRỰC TIẾP

- item[aria-selected="true"]:

→ CSS đọc TRẠNG THÁI DOM để áp style

### KẾT LUẬN NGẮN GỌN:

ARIA KHÔNG LÀM GÌ CẢ

ARIA CHỈ GHI TRẠNG THÁI

JS QUYẾT ĐỊNH LOGIC

CSS QUYẾT ĐỊNH GIAO DIỆN

aria-multiselectable CHO PHÉP NHIỀU TRẠNG THÁI CÙNG TỒN TẠI

-->

=== === === === === THUỘC TÍNH: aria-pressed === === === === ===

(ÁP DỤNG TRÊN NÚT CÓ TRẠNG THÁI BẬT / TẮT)

1. Thuộc tính aria-pressed

 aria-pressed = nhãn TRẠNG THÁI cho nút toggle,để máy đọc màn hình hiểu nút đang bật hay tắt.

→ QUAN TRỌNG: dùng cho NÚT NHẤN CÓ TRẠNG THÁI (TOGGLE BUTTON)

140.1. BẢN CHẤT – DÙNG ĐỂ LÀM GÌ

- Ghi nhận nút đang BẬT hay TẮT

- Giúp MÁY (screen reader) biết trạng thái nút

# Nói đơn giản:

aria-pressed = NÃO CỦA NÚT

---

140.2. PHẠM VI ÁP DỤNG

- Thuộc nhóm: ARIA

- Thường dùng cho:

+ <button>

  + div đóng vai trò button (role="button")

  ---

  140.3. CÚ PHÁP (CỰC QUAN TRỌNG)

  <button aria-pressed="false">...</button>

  # Giá trị CHỈ CÓ:

  - "true" → đang BẬT

  - "false" → đang TẮT

  - "mixed" → trạng thái lưng chừng (hiếm, nâng cao)

  # BẮT BUỘC là STRING, KHÔNG phải true/false JS

  ---

  140.4. DÙNG TRONG TRƯỜNG HỢP NÀO

  - Nút BẬT / TẮT:

  + Đèn

  + Âm thanh

  + Wifi

  + Like / Favorite

  + Công tắc ảo

  # KHÔNG dùng cho:

  - Nút bấm 1 lần (Submit, Send, OK…)

  ---

  140.5. LƯU Ý QUAN TRỌNG (DỄ SAI)

  - aria-pressed KHÔNG TỰ ĐỔI

  - JavaScript PHẢI đổi giá trị

  - aria-pressed CHỈ GHI TRẠNG THÁI

  - GIAO DIỆN đổi là do CSS / JS

  # Nếu không có JS:

  - aria-pressed = vô nghĩa

  ---

  140.6. LIÊN HỆ VỚI id (PHẦN CẦN NHỚ)

  - id:

  → giúp JS TÌM ĐÚNG NÚT

  - aria-pressed:

  → giúp MÁY BIẾT NÚT ĐANG Ở TRẠNG THÁI GÌ

  id = TÊN RIÊNG

  aria-pressed = TRẠNG THÁI

  ---

  140.7. SO SÁNH NHANH (RẤT HAY NHẦM)

  aria-pressed:

  - Dành cho NÚT TOGGLE

  - 1 nút, nhiều trạng thái

  aria-checked:

  - Dành cho checkbox / switch

  - Thường là bật/tắt rõ ràng

  ---

  140.8. CÁCH HOẠT ĐỘNG (NÓI DỄ HIỂU)

  - Người dùng bấm nút

  - JS đọc aria-pressed hiện tại

  - JS quyết định đổi:

  false → true

  true → false

  - CSS nhìn aria-pressed để đổi màu

  - Máy đọc màn hình thông báo trạng thái mới

  ---

  140.9. TÓM LẠI 1 DÒNG (ĐỂ THUỘC)

  aria-pressed

  → GHI TRẠNG THÁI NÚT

  → KHÔNG LÀM GÌ CẢ

  → JS QUYẾT ĐỊNH

  → CSS HIỂN THỊ

  → MÁY THÌ HIỂU

140.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test aria-pressed</title>

  <style>

    button {

      padding: 12px 20px;

      font-size: 16px;

      cursor: pointer;

    }

    /\* Khi nút đang BẬT \*/

    button[aria-pressed="true"] {

      background: green;

      color: white;

    }

    /\* Khi nút đang TẮT \*/

    button[aria-pressed="false"] {

      background: gray;

      color: white;

    }

  </style>

</head>

<body>

  <button id="btnToggle" aria-pressed="false">

    OFF

  </button>

  <script>

    const btn = document.getElementById("btnToggle");

    btn.onclick = function () {

      const isPressed = btn.getAttribute("aria-pressed");

     if (isPressed === "false") {

        btn.setAttribute("aria-pressed", "true");

        btn.textContent = "ON";

      } else {

        btn.setAttribute("aria-pressed", "false");

        btn.textContent = "OFF";

      }

    };

  </script>

</body>

</html>

<!--

###

<button id="btnToggle" aria-pressed="false">

  OFF

</button>

###

BAN ĐẦU (khi trang vừa load):

- Trình duyệt đọc HTML từ trên xuống

- Gặp thẻ <button> → tạo 1 PHẦN TỬ trên DOM

- id="btnToggle"

  → định danh DUY NHẤT cho nút này

  → JS sẽ dùng id này để tìm đúng nút

- aria-pressed="false"

  → trạng thái ban đầu: CHƯA ĐƯỢC NHẤN

  → máy đọc màn hình hiểu:

    “toggle button, not pressed”

  → LƯU Ý:

    aria-pressed KHÔNG tự bật / tắt gì

    nó chỉ GHI NHỚ & MÔ TẢ trạng thái

- Nội dung hiển thị trên nút lúc đầu là chữ "OFF"

→ KẾT LUẬN BAN ĐẦU:

- Người nhìn: thấy nút OFF

- Máy đọc màn hình: biết nút đang TẮT

- Chưa có sự kiện nào xảy ra

###

button[aria-pressed="true"] { background: green; }

button[aria-pressed="false"] { background: gray; }

###

GIẢI THÍCH CSS:

- Đây là CSS DỰA TRÊN THUỘC TÍNH (attribute selector)

- Ý nghĩa:

  “nếu button có aria-pressed = true thì tô màu xanh”

  “nếu button có aria-pressed = false thì tô màu xám”

- CSS KHÔNG quan tâm JS

- CSS chỉ nhìn GIÁ TRỊ HIỆN TẠI của aria-pressed

- Khi JS đổi aria-pressed

  → CSS tự đổi giao diện theo

→ Đây là chỗ HTML + JS + CSS THÔNG NHAU

###

const btn = document.getElementById("btnToggle");

###

JS BẮT ĐẦU LÀM VIỆC:

- document:

  đại diện cho TOÀN BỘ trang HTML

- getElementById("btnToggle"):

  tìm đúng phần tử có id="btnToggle"

- btn:

  là BIẾN JS

  trỏ trực tiếp vào NÚT TRÊN MÀN HÌNH

→ từ giờ:

- btn = cái nút người dùng bấm

###

btn.onclick = function () { ... }

###

GÁN SỰ KIỆN CLICK:

- onclick:

  sự kiện click chuột

- function () { ... }:

  hàm xử lý khi click

- JS KHÔNG chạy ngay

- JS chỉ GHI NHỚ:

  “khi người dùng bấm nút → chạy hàm này”

===== NGƯỜI DÙNG BẤM NÚT LẦN THỨ NHẤT =====

→ Trình duyệt phát hiện click

→ Tự động gọi hàm onclick

###

const isPressed = btn.getAttribute("aria-pressed");

###

TRONG HÀM onclick – DÒNG 1:

- getAttribute("aria-pressed"):

  đọc GIÁ TRỊ HIỆN TẠI của aria-pressed

- LÚC NÀY:

  aria-pressed = "false"

- isPressed = "false"

→ LƯU Ý QUAN TRỌNG:

- ARIA luôn là CHUỖI

- "false" ≠ false (boolean)

###

if (isPressed === "false") { ... }

###

KIỂM TRA ĐIỀU KIỆN:

- === là SO SÁNH NGHIÊM NGẶT

- so cả:

  + giá trị

  + kiểu dữ liệu

- So sánh:

  "false" === "false" → ĐÚNG

→ chương trình CHẠY VÀO IF

###

btn.setAttribute("aria-pressed", "true");

###

HÀNH ĐỘNG 1 – ĐỔI TRẠNG THÁI:

- setAttribute:

  GÁN / GHI ĐÈ thuộc tính HTML

- aria-pressed đổi từ:

  "false" → "true"

→ Ý NGHĨA:

- trạng thái nút được LƯU LẠI

- máy đọc màn hình biết:

  “button, pressed”

###

btn.textContent = "ON";

###

HÀNH ĐỘNG 2 – ĐỔI CHỮ TRÊN NÚT:

- textContent:

  thay đổi NỘI DUNG VĂN BẢN bên trong thẻ

- "OFF" → "ON"

→ Người nhìn thấy nút đã bật

===== CSS TỰ ĐỘNG CAN THIỆP =====

- aria-pressed="true"

→ CSS selector khớp

→ nền nút đổi sang MÀU XANH

===== KẾT QUẢ SAU LẦN BẤM THỨ NHẤT =====

- aria-pressed="true"

- chữ: ON

- màu: xanh

- người nhìn: thấy bật

- máy đọc màn hình: biết đang bật

===== NGƯỜI DÙNG BẤM LẦN THỨ HAI =====

→ onclick chạy lại từ đầu

###

const isPressed = btn.getAttribute("aria-pressed");

###

- LÚC NÀY:

  aria-pressed = "true"

- isPressed = "true"

###

if (isPressed === "false")

###

- "true" === "false" → SAI

→ bỏ qua if

→ chạy sang else

###

btn.setAttribute("aria-pressed", "false");

###

- đổi trạng thái từ:

  "true" → "false"

→ nút được ghi nhớ là TẮT

###

btn.textContent = "OFF";

###

- chữ đổi về OFF

===== CSS LẠI TỰ CHẠY =====

- aria-pressed="false"

→ nền đổi về MÀU XÁM

===== KẾT QUẢ SAU LẦN BẤM THỨ HAI =====

- nút tắt

- aria-pressed="false"

- giao diện quay về ban đầu

===== TÓM LẠI BẢN CHẤT =====

- aria-pressed:

  dùng cho NÚT BẬT / TẮT (toggle)

  ghi nhớ trạng thái cho MÁY

- JavaScript:

  quyết định LOGIC bật / tắt

- CSS:

  dựa vào aria để đổi giao diện

- DOM:

  là cầu nối để JS điều khiển HTML

→ UI CÓ NÃO

→ người + máy đều hiểu đúng

-->





140.10.1.Exxamble

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test aria-pressed</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    button {

      padding: 14px 28px;

      font-size: 18px;

      border: none;

      cursor: pointer;

      border-radius: 6px;

    }

    /\* NÚT ĐANG TẮT \*/

    button[aria-pressed="false"] {

      background: #30363d;

      color: white;

    }

    /\* NÚT ĐANG BẬT \*/

    button[aria-pressed="true"] {

      background: #238636;

      color: white;

    }

  </style>

</head>

<body>

  <h3>Test aria-pressed (ON / OFF)</h3>

  <button id="btn" aria-pressed="false">

    OFF

  </button>

  <p id="log">Trạng thái: TẮT</p>

  <script>

    const btn = document.getElementById("btn");

    const log = document.getElementById("log");

    btn.onclick = function () {

      const trangThai = btn.getAttribute("aria-pressed");

      if (trangThai === "false") {

        btn.setAttribute("aria-pressed", "true");

        btn.textContent = "ON";

        log.textContent = "Trạng thái: BẬT";

      }

      else {

        btn.setAttribute("aria-pressed", "false");

        btn.textContent = "OFF";

        log.textContent = "Trạng thái: TẮT";

      }

    };

  </script>

</body>

</html>

=== HẾT 140 ===

=== === === === === THUỘC TÍNH: aria-readonly === === === === ===

141. Thuộc tính aria-readonly

(QUAN TRỌNG: dùng để NÓI CHO MÁY BIẾT – KHÔNG TỰ KHÓA Ô NHẬP)

141.1. Bản chất (dùng để làm gì)

- Ghi nhớ & thông báo rằng: ô này CHỈ ĐỌC, KHÔNG CHO SỬA

- Giúp máy đọc màn hình hiểu đúng trạng thái phần tử

→ NÓI NGẮN GỌN:

aria-readonly = “cho máy biết là không được sửa”

141.2. Phạm vi áp dụng

- ARIA

- Dùng cho:

input, textarea

hoặc phần tử có role giống ô nhập (textbox, gridcell…)

141.3. Cú pháp (RẤT QUAN TRỌNG)

<tag aria-readonly="true"></tag>

<tag aria-readonly="false"></tag>

→ GIÁ TRỊ BẮT BUỘC:

- "true" = chỉ đọc

- "false" = cho phép sửa

141.4. Giá trị thường dùng

- Chuỗi (string)

- CHỈ có 2 giá trị:

"true" | "false"

141.5. LƯU Ý CỰC KỲ QUAN TRỌNG (DỄ NHẦM)

- aria-readonly:

❌ KHÔNG khóa nhập liệu

❌ KHÔNG chặn gõ phím

✅ CHỈ MÔ TẢ TRẠNG THÁI CHO MÁY

- readonly (HTML):

✅ THẬT SỰ KHÓA ô nhập

❌ KHÔNG phải ARIA

→ HIỂU ĐÚNG:

readonly = KHÓA THỰC

aria-readonly = GHI NHẬN TRẠNG THÁI

141.6. Khi nào BẮT BUỘC dùng aria-readonly

- Khi bạn:

+ tự làm UI bằng div / span

+ dùng JS điều khiển nhập liệu

+ tạo ô nhập “giả” (custom input)

- Nếu chỉ dùng <input readonly>

→ KHÔNG CẦN aria-readonly (vì HTML đã đủ)

141.7. Quan hệ với CSS & JS (HIỂU BẢN CHẤT)

- JS:

đọc / ghi aria-readonly để biết trạng thái

- CSS:

có thể dựa vào aria-readonly để đổi giao diện

- ARIA:

giúp UI “có não” với người khiếm thị

141.8. TÓM TẮT 1 DÒNG (ĐỂ NHỚ)

aria-readonly:

- không khóa

- không chặn

- chỉ NÓI CHO MÁY BIẾT

- dùng khi UI KHÔNG PHẢI input thật

141.9.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>aria-readonly + lưu dữ liệu</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    input {

      width: 320px;

      padding: 10px;

      font-size: 16px;

      border-radius: 6px;

      border: 1px solid #30363d;

    }

    /\* INPUT CHỈ ĐỌC \*/

    input[aria-readonly="true"] {

      background: #161b22;

      color: #8b949e;

    }

    button {

      margin-top: 10px;

      padding: 8px 16px;

      cursor: pointer;

    }

  </style>

</head>

<body>

  <h3>Thông tin người dùng</h3>

  <input id="name" type="text" aria-readonly="true" readonly>

  <br>

  <button id="edit">✏️ Sửa</button>

  <button id="save">💾 Lưu</button>

  <p id="log"></p>

  <script>

    const input = document.getElementById("name");

    const edit = document.getElementById("edit");

    const save = document.getElementById("save");

    const log = document.getElementById("log");

    /\* ===== KHI LOAD TRANG ===== \*/

    const savedName = localStorage.getItem("username");

    if (savedName) {

      input.value = savedName;

      log.textContent = "Đã tải dữ liệu đã lưu";

    } else {

      input.value = "Chưa có dữ liệu";

    }

    /\* ===== BẤM SỬA ===== \*/

    edit.onclick = function () {

      input.removeAttribute("readonly");

      input.setAttribute("aria-readonly", "false");

      input.focus();

      log.textContent = "Đang chỉnh sửa...";

    };

    /\* ===== BẤM LƯU ===== \*/

    save.onclick = function () {

      const value = input.value;

      localStorage.setItem("username", value);

      input.setAttribute("readonly", "");

      input.setAttribute("aria-readonly", "true");

      log.textContent = "Đã lưu & khóa lại";

    };

  </script>

</body>

</html>

<!--

###

<input id="name" type="text" aria-readonly="true" readonly>

###

BAN ĐẦU (khi trình duyệt vừa load trang):

- Trình duyệt đọc HTML từ trên xuống

- Gặp thẻ <input> → tạo 1 PHẦN TỬ INPUT trên DOM

PHÂN TÍCH TỪNG THUỘC TÍNH:

- id="name"

→ định danh DUY NHẤT cho input

→ JavaScript sẽ dùng id này để TÌM ĐÚNG Ô NHẬP

- type="text"

→ input cho phép nhập CHỮ

- readonly

→ trình duyệt KHÓA ô nhập

→ người dùng KHÔNG gõ được

- aria-readonly="true"

→ GHI NHẬN TRẠNG THÁI cho MÁY

→ máy đọc màn hình hiểu:

“ô nhập này chỉ đọc, không chỉnh sửa được”

→ LƯU Ý QUAN TRỌNG:

aria-readonly KHÔNG tự khóa input

→ readonly mới là thằng khóa thật

→ aria-readonly chỉ MÔ TẢ TRẠNG THÁI

KẾT LUẬN BAN ĐẦU:

- Người nhìn: thấy ô nhập bị mờ, không gõ được

- Máy đọc màn hình: biết input đang bị khóa

- Chưa có JS nào chạy cả

###

input[aria-readonly="true"] { ... }

###

GIẢI THÍCH CSS DỰA TRÊN ARIA:

- Đây là ATTRIBUTE SELECTOR

- Ý nghĩa:

“nếu input có aria-readonly = true

thì áp dụng style này”

- CSS KHÔNG BIẾT JS

- CSS CHỈ NHÌN GIÁ TRỊ HIỆN TẠI của aria-readonly

→ Khi JS đổi aria-readonly:

CSS TỰ ĐỘNG ĐỔI GIAO DIỆN

ĐÂY LÀ CHỖ:

HTML + JS + CSS THÔNG NHAU

###

const input = document.getElementById("name");

const edit = document.getElementById("edit");

const save = document.getElementById("save");

const log = document.getElementById("log");

###

JS BẮT ĐẦU LÀM VIỆC:

- document

→ đại diện cho TOÀN BỘ trang HTML

- getElementById(...)

→ tìm phần tử có id tương ứng

- input

→ BIẾN JS trỏ thẳng tới ô input trên màn hình

- edit

→ trỏ tới nút ✏️ Sửa

- save

→ trỏ tới nút 💾 Lưu

- log

→ trỏ tới thẻ <p> hiển thị thông báo

→ TỪ GIỜ:

JS có quyền:

- đọc dữ liệu

- sửa nội dung

- đổi thuộc tính

- đổi style GIÁN TIẾP

###

const savedName = localStorage.getItem("username");

###

KHI LOAD TRANG – DÒNG ĐẦU TIÊN:

- localStorage:

bộ nhớ của TRÌNH DUYỆT

→ lưu dữ liệu kể cả khi F5 hay đóng tab

- getItem("username"):

→ lấy dữ liệu đã lưu với key = "username"

- savedName:

→ nếu CHƯA từng lưu → null

→ nếu ĐÃ lưu → chuỗi văn bản

LƯU Ý:

- localStorage CHỈ LƯU STRING

###

if (savedName) { ... } else { ... }

###

KIỂM TRA DỮ LIỆU CŨ:

- Nếu savedName TỒN TẠI:

→ input.value = savedName

→ log báo: “Đã tải dữ liệu đã lưu”

- Nếu KHÔNG có dữ liệu:

→ input.value = "Chưa có dữ liệu"

value LÀ GÌ?

- value là GIÁ TRỊ BÊN TRONG INPUT

- thay đổi value → chữ trong ô nhập đổi theo

→ KẾT THÚC GIAI ĐOẠN LOAD TRANG

###

edit.onclick = function () { ... }

###

GÁN SỰ KIỆN CLICK CHO NÚT SỬA:

- onclick:

→ sự kiện khi người dùng BẤM CHUỘT

- function () { ... }

→ hàm xử lý

→ KHÔNG chạy ngay

→ JS chỉ GHI NHỚ:

“khi click thì chạy khối này”

===== NGƯỜI DÙNG BẤM NÚT ✏️ SỬA =====

→ trình duyệt phát hiện click

→ tự động gọi hàm onclick

###

input.removeAttribute("readonly");

###

HÀNH ĐỘNG 1 – MỞ KHÓA INPUT:

- removeAttribute("readonly"):

→ GỠ BỎ THUỘC TÍNH readonly khỏi HTML

KẾT QUẢ:

- input KHÔNG còn bị khóa

- người dùng BẮT ĐẦU GÕ ĐƯỢC

LƯU Ý:

- removeAttribute tác động TRỰC TIẾP HTML

###

input.setAttribute("aria-readonly", "false");

###

HÀNH ĐỘNG 2 – CẬP NHẬT TRẠNG THÁI ARIA:

- setAttribute:

→ GÁN / GHI ĐÈ thuộc tính HTML

- aria-readonly:

"true" → "false"

Ý NGHĨA:

- máy đọc màn hình hiểu:

“ô nhập này có thể chỉnh sửa”

- CSS NHÌN THẤY aria-readonly đổi

→ giao diện đổi theo

###

input.focus();

###

HÀNH ĐỘNG 3 – ĐƯA CON TRỎ VÀO INPUT:

- focus():

→ đặt con trỏ nhấp nháy trong ô nhập

→ UX tốt hơn

###

log.textContent = "Đang chỉnh sửa...";

###

HÀNH ĐỘNG 4 – HIỂN THỊ THÔNG BÁO:

- textContent:

→ thay nội dung VĂN BẢN bên trong thẻ

→ Người dùng biết:

“mình đang ở chế độ sửa”

===== NGƯỜI DÙNG GÕ NỘI DUNG MỚI =====

→ input.value TỰ ĐỘNG thay đổi theo chữ gõ

###

save.onclick = function () { ... }

###

NGƯỜI DÙNG BẤM 💾 LƯU:

→ trình duyệt gọi hàm onclick của save

###

const value = input.value;

###

LẤY DỮ LIỆU HIỆN TẠI:

- input.value:

→ toàn bộ chữ người dùng vừa gõ

- value:

→ lưu tạm trong JS

###

localStorage.setItem("username", value);

###

HÀNH ĐỘNG LƯU DỮ LIỆU:

- setItem(key, value):

→ lưu dữ liệu vào TRÌNH DUYỆT

- "username":

→ tên định danh

- value:

→ dữ liệu người dùng nhập

→ DỮ LIỆU SỐNG DAI:

- reload vẫn còn

- tắt trình duyệt mở lại vẫn còn

###

input.setAttribute("readonly", "");

input.setAttribute("aria-readonly", "true");

###

KHÓA INPUT LẠI:

- readonly:

→ khóa thật (HTML)

- aria-readonly="true":

→ ghi nhận trạng thái cho MÁY

→ CSS đổi giao diện

→ input trở về trạng thái CHỈ ĐỌC

###

log.textContent = "Đã lưu & khóa lại";

###

THÔNG BÁO KẾT THÚC:

- người dùng biết:

→ dữ liệu đã được lưu

→ không chỉnh sửa nữa

===== TÓM LẠI BẢN CHẤT =====

- readonly:

→ khóa input cho NGƯỜI

- aria-readonly:

→ mô tả trạng thái cho MÁY

- JavaScript:

→ quyết định LUỒNG LOGIC

→ mở / khóa / lưu

- CSS:

→ nhìn aria để đổi giao diện

- DOM:

→ cầu nối để JS điều khiển HTML

→ UI CÓ NÃO

→ người hiểu

→ máy cũng hiểu

-->

=== HẾT ===

=== === === === === THUỘC TÍNH: aria-required === === === === ===

142. Thuộc tính aria-required

→ Dòng này QUAN TRỌNG:

→ aria-required dùng để NÓI CHO MÁY BIẾT:

“trường này BẮT BUỘC phải nhập dữ liệu”

--------------------------------------------------

142.1. BẢN CHẤT (dùng để làm gì)

- ĐÁNH DẤU một trường nhập liệu là BẮT BUỘC

- Giúp máy đọc màn hình hiểu:

“input này không được bỏ trống”

→ aria-required KHÔNG tự kiểm tra dữ liệu

→ nó chỉ GHI NHẬN & MÔ TẢ trạng thái

--------------------------------------------------

142.2. PHẠM VI ÁP DỤNG

- Thuộc nhóm ARIA

- Dùng cho các phần tử nhập liệu:

input

textarea

select

- KHÔNG dùng cho div, span thông thường

--------------------------------------------------

142.3. CÚ PHÁP (PHẦN QUAN TRỌNG)

<tag aria-required="true"></tag>

Giải thích cú pháp:

- aria-required="true"

→ trường này BẮT BUỘC nhập

- aria-required="false"

→ trường này KHÔNG bắt buộc

LƯU Ý CỰC QUAN TRỌNG:

- Giá trị LUÔN là CHUỖI

- "true" và "false"

- KHÔNG phải boolean true / false của JS

--------------------------------------------------

142.4. GIÁ TRỊ THƯỜNG DÙNG

- "true" → bắt buộc

- "false" → không bắt buộc (mặc định)

--------------------------------------------------

142.5. SỰ KHÁC NHAU QUAN TRỌNG

- required (HTML):

→ trình duyệt TỰ CHẶN submit

- aria-required (ARIA):

→ KHÔNG chặn

→ chỉ để MÁY + TRỢ NĂNG HIỂU

KẾT LUẬN:

- required = hành vi

- aria-required = ngữ nghĩa

--------------------------------------------------

142.6. CÁCH JS TƯƠNG TÁC (HIỂU BẢN CHẤT)

- setAttribute("aria-required", "true")

→ GHI ĐÈ giá trị trong HTML

- getAttribute("aria-required")

→ ĐỌC trạng thái hiện tại

LƯU Ý:

- ARIA KHÔNG tự đổi

- JS là thằng QUYẾT ĐỊNH LOGIC

--------------------------------------------------

142.7. LIÊN QUAN ĐẾN CSS

- CSS có thể dựa vào aria-required

- Khi JS đổi aria-required

→ CSS tự đổi giao diện

Ý nghĩa:

- JS đổi TRẠNG THÁI

- CSS đổi GIAO DIỆN

- HTML giữ NGỮ NGHĨA

→ 3 THẰNG THÔNG NHAU

--------------------------------------------------

142.8. LỖI HAY GẶP

- Nghĩ aria-required tự kiểm tra dữ liệu ❌

- Gán aria-required nhưng không validate bằng JS ❌

- So sánh "true" với true ❌

--------------------------------------------------

142.9. TÓM LẠI NHANH

- aria-required:

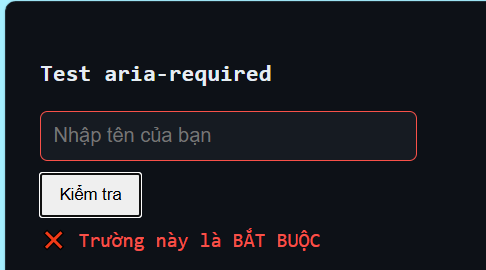
→ nói cho MÁY biết trường bắt buộc

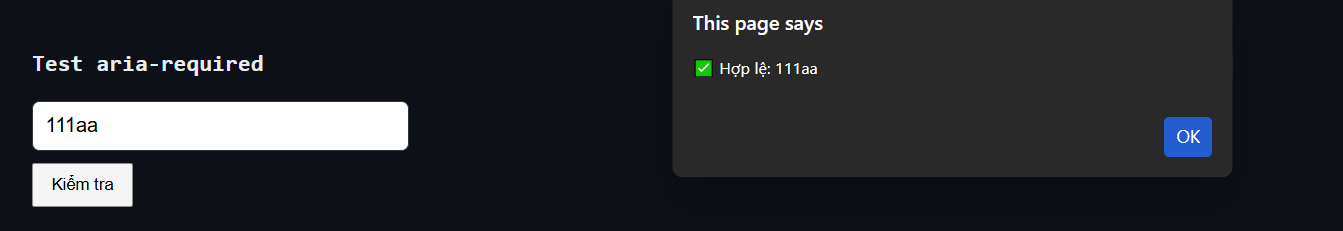
- KHÔNG tự chặn nhập

- KHÔNG thay thế required

- DÙNG CHUNG với JS để UI có não

142.10.EXAMBLE





<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test aria-required</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    input {

      width: 280px;

      padding: 10px;

      font-size: 16px;

      border-radius: 6px;

      border: 1px solid #30363d;

      outline: none;

    }

    /\* INPUT BẮT BUỘC + LỖI \*/

    input[aria-invalid="true"] {

      border-color: #f85149;

      background: #161b22;

    }

    button {

      margin-top: 10px;

      padding: 8px 14px;

      cursor: pointer;

    }

    #err {

      color: #f85149;

      display: none;

      margin-top: 6px;

    }

  </style>

</head>

<body>

  <h3>Test aria-required</h3>

  <input id="name" type="text" aria-required="true" aria-invalid="false" placeholder="Nhập tên của bạn">

  <br>

  <button id="check">Kiểm tra</button>

  <p id="err">❌ Trường này là BẮT BUỘC</p>

  <script>

    const input = document.getElementById("name");

    const btn = document.getElementById("check");

    const err = document.getElementById("err");

    btn.onclick = function () {

      const value = input.value.trim();

      if (value === "") {

        input.setAttribute("aria-invalid", "true");

        err.style.display = "block";

      } else {

        input.setAttribute("aria-invalid", "false");

        err.style.display = "none";

        alert("✅ Hợp lệ: " + value);

      }

    };

  </script>

</body>

</html>

###

<input id="name" type="text" aria-required="true" aria-invalid="false" placeholder="Nhập tên của bạn">

<button id="check">Kiểm tra</button>

<p id="err">❌ Trường này là BẮT BUỘC</p>

###

BAN ĐẦU (KHI TRANG VỪA LOAD):

- Trình duyệt đọc HTML từ trên xuống

- Gặp thẻ <input>

  → tạo 1 PHẦN TỬ NHẬP LIỆU trên DOM

- id="name"

  → định danh DUY NHẤT

  → JS sẽ dùng id này để tìm đúng ô nhập

- type="text"

  → đây là ô nhập chữ bình thường

- placeholder

  → chỉ là CHỮ GỢI Ý

  → KHÔNG phải dữ liệu thật

- aria-required="true"

  → ý nghĩa NGỮ NGHĨA:

    “ô này BẮT BUỘC phải nhập”

  → máy đọc màn hình hiểu:

    “required field”

  → LƯU Ý:

    aria-required KHÔNG tự kiểm tra

    KHÔNG chặn nhập

    chỉ DÁN NHÃN cho MÁY

- aria-invalid="false"

  → trạng thái ban đầu:

    dữ liệu CHƯA BỊ LỖI

- <p id="err">

  → là PHẦN TỬ BÁO LỖI

  → ban đầu:

    display: none

    nên KHÔNG HIỆN

→ KẾT LUẬN BAN ĐẦU:

- Người nhìn: thấy ô nhập + nút

- Máy đọc màn hình: biết ô này là BẮT BUỘC

- Chưa có sự kiện nào xảy ra

================================================

###

input[aria-invalid="true"] {

  border-color: #f85149;

  background: #161b22;

}

###

GIẢI THÍCH CSS (RẤT QUAN TRỌNG):

- Đây là CSS THEO THUỘC TÍNH (attribute selector)

- Ý nghĩa:

  “nếu input có aria-invalid = true

   thì đổi giao diện”

- CSS KHÔNG biết JS

- CSS KHÔNG tự chạy logic

- CSS chỉ NHÌN GIÁ TRỊ HIỆN TẠI trong HTML

→ KHI JS đổi aria-invalid

→ CSS TỰ ĐỘNG ĐỔI GIAO DIỆN

→ Đây là chỗ HTML + JS + CSS THÔNG NHAU

================================================

###

const input = document.getElementById("name");

const btn   = document.getElementById("check");

const err   = document.getElementById("err");

###

JS BẮT ĐẦU LÀM VIỆC:

- document

  → đại diện cho TOÀN BỘ trang HTML

- getElementById("name")

  → tìm PHẦN TỬ có id="name"

- input

  → biến JS

  → TRỎ TRỰC TIẾP vào ô nhập trên màn hình

- TƯƠNG TỰ:

  btn → nút bấm

  err → dòng báo lỗi

→ DOM:

- là CẦU NỐI

- cho phép JS:

  đọc / sửa / điều khiển HTML & CSS

================================================

###

btn.onclick = function () { ... }

###

GÁN SỰ KIỆN CLICK:

- onclick

  → sự kiện khi người dùng BẤM NÚT

- function () { ... }

  → hàm xử lý

- JS KHÔNG chạy ngay

- JS chỉ GHI NHỚ:

  “khi người dùng bấm nút → chạy hàm này”

================================================

===== NGƯỜI DÙNG BẤM NÚT "KIỂM TRA" =====

================================================

→ Trình duyệt phát hiện click

→ Tự động gọi hàm onclick

→ MỌI DÒNG TRONG HÀM CHẠY TỪ TRÊN XUỐNG

================================================

###

const value = input.value.trim();

###

DÒNG NÀY ĐANG LÀM GÌ:

- input.value

  → lấy DỮ LIỆU người dùng đã nhập

- trim()

  → cắt khoảng trắng đầu & cuối

  → tránh trường hợp nhập toàn dấu cách

- value

  → biến lưu GIÁ TRỊ THẬT

→ LƯU Ý:

- value luôn là CHUỖI (string)

- kể cả nhập số

================================================

###

if (value === "")

###

KIỂM TRA ĐIỀU KIỆN:

- ===

  → so sánh NGHIÊM NGẶT

  → so cả:

    + giá trị

    + kiểu dữ liệu

- "" là chuỗi rỗng

→ Ý NGHĨA:

- nếu người dùng KHÔNG nhập gì

→ điều kiện ĐÚNG

================================================

===== TRƯỜNG HỢP 1: KHÔNG NHẬP GÌ =====

================================================

###

input.setAttribute("aria-invalid", "true");

###

GIẢI THÍCH TỪNG PHẦN:

- input

  → phần tử HTML ô nhập

- .setAttribute()

  → GÁN / GHI ĐÈ thuộc tính HTML

- "aria-invalid"

  → thuộc tính ARIA báo LỖI

- "true"

  → đánh dấu:

    dữ liệu KHÔNG HỢP LỆ

→ HỆ QUẢ:

- HTML đổi aria-invalid="true"

- Máy đọc màn hình hiểu:

  “invalid input”

- CSS thấy aria-invalid="true"

  → đổi viền ĐỎ

================================================

###

err.style.display = "block";

###

DÒNG NÀY RẤT QUAN TRỌNG – GIẢI THÍCH KỸ:

- err

  → phần tử <p> báo lỗi

- .style

  → truy cập CSS INLINE của phần tử

- .display

  → thuộc tính CSS display

- = "block"

  → HIỆN phần tử

→ BẢN CHẤT:

- JS đang CAN THIỆP TRỰC TIẾP vào CSS

- KHÔNG cần sửa file CSS

→ HTML + CSS + JS ĐANG NỐI VỚI NHAU

================================================

===== KẾT QUẢ TRƯỜNG HỢP LỖI =====

- Ô input:

  viền đỏ

- Dòng lỗi:

  HIỆN

- aria-invalid="true"

- Người nhìn: biết lỗi

- Máy đọc màn hình: biết lỗi

================================================

===== TRƯỜNG HỢP 2: CÓ NHẬP DỮ LIỆU =====

================================================

###

input.setAttribute("aria-invalid", "false");

###

- GHI ĐÈ trạng thái lỗi

- aria-invalid từ:

  "true" → "false"

→ Ý NGHĨA:

- dữ liệu HỢP LỆ

- CSS KHÔNG còn khớp rule lỗi

→ giao diện quay về BÌNH THƯỜNG

================================================

###

err.style.display = "none";

###

- display="none"

  → ẨN dòng báo lỗi

- KHÔNG chiếm chỗ trên giao diện

================================================

###

alert("✅ Hợp lệ: " + value);

###

- alert

  → thông báo nhanh (demo)

- + value

  → nối chuỗi

→ giả lập hành vi:

- form hợp lệ

- dữ liệu sẵn sàng gửi server

================================================

===== TÓM LẠI BẢN CHẤT (ĐỌC 1 LẦN LÀ NHỚ) =====

================================================

- aria-required

  → nói cho MÁY biết ô này BẮT BUỘC

- aria-invalid

  → ghi nhớ trạng thái ĐÚNG / SAI

- JavaScript

  → quyết định LOGIC

  → kiểm tra dữ liệu

  → đổi trạng thái

- CSS

  → phản ứng theo ARIA

- DOM

  → là CẦU NỐI

  → để JS điều khiển HTML & CSS

→ UI CÓ NÃO

→ người dùng + máy đều hiểu đúng

-->

=== === === === === THUỘC TÍNH: aria-selected === === === === ===

143. Thuộc tính aria-selected

→ Dùng để BÁO cho trình đọc màn hình biết: phần tử này ĐANG ĐƯỢC CHỌN hay KHÔNG

--------------------------------------------------

143.1. Bản chất (dùng để làm gì)

- Đánh dấu trạng thái "đã chọn / chưa chọn"

- Giúp người dùng KHIẾM THỊ hiểu giao diện đang chọn cái nào

(Hiểu đơn giản: MÁY ĐỌC MÀN HÌNH DÙNG, không phải để trang trí)

--------------------------------------------------

143.2. Phạm vi áp dụng

- ARIA attribute (trợ năng)

- Thường dùng cho:

+ tab

+ menu

+ list

+ button dạng chọn

- KHÔNG tự tạo giao diện, chỉ mô tả TRẠNG THÁI

--------------------------------------------------

143.3. Cú pháp (PHẦN QUAN TRỌNG)

- Gắn TRỰC TIẾP vào phần tử HTML

- Giá trị CHỈ có 2 loại chính:

aria-selected="true" → đang được chọn

aria-selected="false" → không được chọn

⚠ Không có giá trị khác (không số, không chữ lung tung)

--------------------------------------------------

143.4. Giá trị thường dùng

- true → ĐANG chọn

- false → CHƯA chọn

(viết THƯỜNG, không viết hoa)

--------------------------------------------------

143.5. Lưu ý / lỗi hay gặp (CỰC KỲ QUAN TRỌNG)

- aria-selected KHÔNG làm đổi màu, KHÔNG làm active

- Phải KẾT HỢP với CSS hoặc JS để người dùng thường thấy được

- Không dùng bừa cho thẻ không có ý nghĩa chọn

- Nếu giao diện đổi trạng thái → aria-selected PHẢI đổi theo

--------------------------------------------------

143.6. Ngữ nghĩa cần nhớ (NOTE NÃO)

- aria-\* = dành cho TRỢ NĂNG

- Người bình thường KHÔNG thấy

- Máy đọc màn hình sẽ đọc kiểu:

"selected" hoặc "not selected"

--------------------------------------------------

143.7. So nhanh cho dễ nhớ

- id → định danh DUY NHẤT

- class → nhóm nhiều phần tử

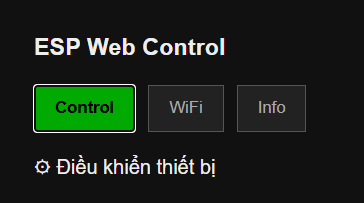
- aria-\* → nói cho MÁY HIỂU (không phải cho mắt người)

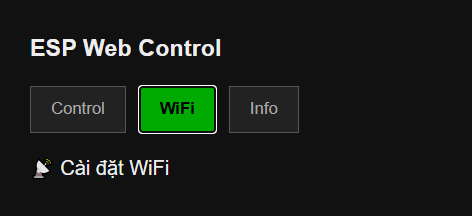
--------------------------------------------------

GHI NHỚ NHANH 1 DÒNG:

aria-selected = nói cho máy biết "cái này đang được chọn hay chưa"

143.8.EXAMBLE





<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP aria-selected test</title>

  <style>

    body {

      font-family: Arial;

      background: #111;

      color: #eee;

      padding: 20px;

    }

    .tab {

      padding: 10px 16px;

      margin-right: 6px;

      border: 1px solid #555;

      background: #222;

      color: #aaa;

      cursor: pointer;

    }

    /\* TAB đang được chọn \*/

    .tab[aria-selected="true"] {

      background: #0a0;

      color: #000;

      font-weight: bold;

    }

    .panel {

      margin-top: 16px;

      display: none;

    }

    .panel.active {

      display: block;

    }

  </style>

</head>

<body>

  <h3>ESP Web Control</h3>

  <div role="tablist">

    <button class="tab" role="tab" aria-selected="true" onclick="selectTab(0)">Control</button>

    <button class="tab" role="tab" aria-selected="false" onclick="selectTab(1)">WiFi</button>

    <button class="tab" role="tab" aria-selected="false" onclick="selectTab(2)">Info</button>

  </div>

  <div class="panel active">⚙ Điều khiển thiết bị</div>

  <div class="panel">📡 Cài đặt WiFi</div>

  <div class="panel">ℹ Thông tin ESP</div>

  <script>

    function selectTab(index) {

      const tabs = document.querySelectorAll('.tab');

      const panels = document.querySelectorAll('.panel');

      tabs.forEach((tab, i) => {

        tab.setAttribute('aria-selected', i === index ? 'true' : 'false');

        panels[i].classList.toggle('active', i === index);

      });

    }

  </script>

</body>

</html>

<!--

### GIAI ĐOẠN 1 — TRANG VỪA LOAD (TRÌNH DUYỆT ĐỌC HTML)

- Trình duyệt đọc HTML từ trên xuống, chưa có JS chạy.

- Mỗi thẻ HTML được chuyển thành MỘT node trong DOM tree.

DOM được tạo trong bộ nhớ trình duyệt:

- div[role="tablist"]

- button.tab (aria-selected="true") ← tab Control

- button.tab (aria-selected="false") ← tab WiFi

- button.tab (aria-selected="false") ← tab Info

- div.panel.active ← panel[0]

- div.panel ← panel[1]

- div.panel ← panel[2]

👉 Trạng thái DOM BAN ĐẦU:

- tab[0] có aria-selected="true"

- tab[1], tab[2] có aria-selected="false"

- panel[0] có class "active"

- panel[1], panel[2] KHÔNG có class "active"

### GIAI ĐOẠN 2 — CSS ĐỌC DOM & ÁP STYLE

CSS duyệt từng node DOM và so selector:

- .tab

→ cả 3 button đều được áp style nền xám, chữ xám

- .tab[aria-selected="true"]

→ CSS truy cập:

dấu . → truy cập thuộc tính aria-selected của node tab

→ chỉ tab[0] có giá trị "true"

→ tab[0] đổi nền xanh, chữ đen, in đậm

- .panel

→ display:none

→ cả 3 panel đều bị ẩn

- .panel.active

→ CSS kiểm tra classList của panel

→ chỉ panel[0] có "active"

→ panel[0] display:block

👉 Trình duyệt render ra màn hình:

- Tab Control đang được chọn

- Nội dung "⚙ Điều khiển thiết bị" hiển thị

### GIAI ĐOẠN 3 — NGƯỜI DÙNG CLICK (VÍ DỤ CLICK TAB WiFi)

- Người dùng click vào button WiFi

- Trình duyệt tạo sự kiện click

- HTML đã gắn sẵn:

onclick="selectTab(1)"

👉 Trình duyệt gọi JS:

selectTab(index = 1)

### GIAI ĐOẠN 4 — JS BẮT ĐẦU CHẠY (LẤY DOM)

Dòng chạy đầu tiên:

- document.querySelectorAll('.tab')

• document : đối tượng DOM gốc

• dấu . : truy cập method querySelectorAll của document

→ trả về NodeList chứa 3 button tab

Dòng tiếp theo:

- document.querySelectorAll('.panel')

→ trả về NodeList chứa 3 panel

👉 JS lúc này đang CẦM TRỰC TIẾP các node DOM đang hiển thị

### GIAI ĐOẠN 5 — forEach BẮT ĐẦU DUYỆT

- tabs.forEach((tab, i) => { ... })

- JS lặp theo thứ tự i = 0 → 1 → 2

- Mỗi vòng lặp là MỘT lần xử lý trạng thái

### GIAI ĐOẠN 6 — LẦN LẶP i = 0 (TAB CONTROL)

Dòng điều kiện:

- i === index

• === so sánh GIÁ TRỊ + KIỂU

• so sánh: 0 === 1

→ false được sinh ra NGAY TẠI ĐÂY

Toán tử ? :

- điều kiện false → chọn nhánh sau dấu :

- kết quả: 'false'

tab.setAttribute(...)

- setAttribute ghi trực tiếp vào DOM

Trước khi ghi:

- tab[0]: aria-selected="true"

JS là thằng đổi:

- tab.setAttribute('aria-selected', 'false')

Sau khi ghi:

- tab[0]: aria-selected="false"

panels[i].classList.toggle('active', false)

- dấu . : truy cập classList của panel[0]

- toggle với false → GỠ class "active"

DOM thay đổi:

- panel[0] mất class "active"

CSS phản ứng:

- panel[0] không còn display:block → bị ẩn

ARIA phản ứng:

- screen reader hiểu tab Control KHÔNG còn được chọn

### GIAI ĐOẠN 7 — LẦN LẶP i = 1 (TAB WiFi)

So sánh:

- i === index

- 1 === 1 → true

Toán tử ? :

- điều kiện true → chọn nhánh trước dấu :

- kết quả: 'true'

tab.setAttribute(...)

Trước:

- tab[1]: aria-selected="false"

JS ghi:

- aria-selected="true"

Sau:

- tab[1]: aria-selected="true"

panels[1].classList.toggle('active', true)

- toggle với true → THÊM class "active"

DOM sau khi đổi:

- panel[1] có class "active"

CSS phản ứng:

- tab WiFi đổi nền xanh

- panel WiFi display:block

ARIA:

- screen reader đọc:

"WiFi tab selected"

### GIAI ĐOẠN 8 — LẦN LẶP i = 2 (TAB Info)

So sánh:

- 2 === 1 → false

Toán tử ? :

- chọn 'false'

JS ghi:

- tab[2]: aria-selected="false"

- panel[2]: không có "active"

DOM không đổi nhiều:

- panel Info vẫn bị ẩn

CSS & ARIA không thay đổi trạng thái hiển thị

### GIAI ĐOẠN 9 — JS KẾT THÚC, DOM ỔN ĐỊNH

DOM TRƯỚC:

- Control selected

- Panel Control hiển thị

DOM SAU:

- WiFi selected

- Panel WiFi hiển thị

- Các panel khác bị ẩn

👉 JS là THẰNG GHI DOM

👉 CSS là THẰNG PHẢN ỨNG

👉 ARIA là THẰNG GIẢI THÍCH UI CHO MÁY

### TỔNG KẾT VAI TRÒ CUỐI CÙNG

- HTML: khai báo cấu trúc & trạng thái ban đầu

- DOM: mô hình sống trong trình duyệt

- JS: điều khiển logic, so sánh ===, rẽ nhánh ? :

- CSS: render giao diện theo trạng thái DOM

- ARIA: giúp screen reader hiểu tab nào đang active

👉 Trình duyệt chạy ĐÚNG THEO LUỒNG NÀY, không nhảy cóc, không suy đoán

--------------------------------------------------

===== KẾT QUẢ SAU KHI BẤM =====

- aria-selected:

  → chỉ 1 tab = true

  → 2 tab còn lại = false

- CSS:

  → tự đổi màu tab

- panel:

  → chỉ hiện nội dung đúng

- người nhìn:

  → thấy giao diện đổi

- máy đọc màn hình:

  → biết tab nào đang chọn

--------------------------------------------------

===== TÓM LẠI BẢN CHẤT =====

- aria-selected:

  → GHI NHỚ trạng thái CHỌN

  → dành cho MÁY + trợ năng

- JavaScript:

  → quyết định LOGIC

  → đổi aria + đổi class

- CSS:

  → chỉ NHÌN aria + class

  → tự đổi giao diện

- DOM:

  → là CẦU NỐI

  → để JS điều khiển HTML

→ UI CÓ NÃO

→ người + máy HIỂU ĐÚNG

→ dùng rất chuẩn cho WEB ESP

-->

=== === === === === THUỘC TÍNH: aria-valuemax === === === === ===

144. Thuộc tính aria-valuemax

→ QUAN TRỌNG: Thuộc tính này CHỈ dùng để nói cho máy đọc màn hình biết

  GIÁ TRỊ LỚN NHẤT mà một control có thể đạt tới.

  ❗ KHÔNG tự tạo giao diện, KHÔNG tự thay đổi giá trị, chỉ là “nói thông tin”.

--------------------------------------------------

144.1. BẢN CHẤT (DÙNG ĐỂ LÀM GÌ)

- Xác định MỨC TỐI ĐA của một giá trị (max value)

- Giúp screen reader hiểu: “thanh này / control này có thể tăng tới đâu”

👉 Nói ngắn gọn:

- aria-valuemax = trần trên của giá trị

- dùng cho các control có tăng / giảm giá trị

--------------------------------------------------

144.2. PHẠM VI ÁP DỤNG

- ARIA attribute

- Dùng cho các role như:

  • role="progressbar"

  • role="slider"

  • role="spinbutton"

  • role="scrollbar"

❌ KHÔNG dùng cho element thường không có giá trị số

--------------------------------------------------

144.3. CÚ PHÁP (PHẦN QUAN TRỌNG – CẦN NHỚ KỸ)

<tag

  role="progressbar"

  aria-valuemin="0"

  aria-valuenow="30"

  aria-valuemax="100">

</tag>

Giải thích cú pháp:

- aria-valuemin : giá trị nhỏ nhất

- aria-valuenow : giá trị hiện tại

- aria-valuemax : giá trị lớn nhất (đang nói tới)

👉 3 thằng này thường ĐI CÙNG NHAU

--------------------------------------------------

144.4. GIÁ TRỊ THƯỜNG DÙNG

- Số (number)

  ví dụ: 10, 100, 255

❗ LÀ SỐ, KHÔNG PHẢI CHUỖI MÔ TẢ

❌ không dùng chữ như: "full", "max", "cao nhất"

--------------------------------------------------

144.5. CÁCH TRÌNH DUYỆT + SCREEN READER HIỂU

Ví dụ:

- aria-valuemin="0"

- aria-valuenow="30"

- aria-valuemax="100"

→ Máy đọc màn hình hiểu là:

  “Giá trị hiện tại là 30 trên tổng 100”

  “30 phần trăm tiến trình”

👉 aria-valuemax giúp TÍNH TOÁN TỶ LỆ

--------------------------------------------------

144.6. LƯU Ý / LỖI HAY GẶP (RẤT QUAN TRỌNG)

- ❌ Không có aria-valuenow mà chỉ có valuemax → VÔ NGHĨA

- ❌ aria-valuenow > aria-valuemax → SAI LOGIC

- ❌ aria-valuemax < aria-valuemin → LỖI NGỮ NGHĨA

- ❌ Dùng aria-valuemax cho button thường → SAI MỤC ĐÍCH

👉 aria-valuemax KHÔNG điều khiển JS

👉 JS là thằng đổi số

👉 aria-\* chỉ là thằng MÔ TẢ

--------------------------------------------------

144.7. KÝ HIỆU / NGỮ NGHĨA CẦN NHỚ

- aria- = nói chuyện với MÁY, không phải người

- value = giá trị số

- max = mức cao nhất có thể đạt

👉 aria-valuemax = “giá trị tối đa mà control cho phép”

--------------------------------------------------

144.8. SO SÁNH NHANH (ĐỂ NHỚ)

- max trong HTML (input max):

  → có thể chặn người dùng nhập

- aria-valuemax:

  → KHÔNG chặn

  → CHỈ mô tả cho screen reader

--------------------------------------------------

144.9. TÓM GỌN 1 DÒNG (ĐỂ NHỚ NHANH)

aria-valuemax =

“Giới hạn trên của giá trị, dùng để MÔ TẢ cho máy đọc màn hình,

không tự làm thay đổi hành vi”

--------------------------------------------------

144.10. KHI NÀO NÊN DÙNG

- Khi bạn tự làm slider / progress bằng div

- Khi giao diện không phải control HTML mặc định

- Khi cần hỗ trợ người dùng khiếm thị

👉 Nếu KHÔNG cần accessibility → có thể không cần

👉 Nếu làm UI chuẩn → NÊN CÓ

=== KẾT LUẬN NHANH ===

aria-valuemax = nói cho MÁY BIẾT “tối đa là bao nhiêu”

JS mới là thằng điều khiển

CSS chỉ là thằng vẽ

144.11.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test aria-valuemax</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    .bar {

      width: 300px;

      height: 24px;

      border: 1px solid #30363d;

      background: #161b22;

      position: relative;

    }

    .fill {

      height: 100%;

      background: #238636;

      width: 30%;

    }

    button {

      margin-top: 10px;

      padding: 6px 12px;

      cursor: pointer;

    }

  </style>

</head>

<body>

  <h3>Progress test (aria-valuemax)</h3>

  <div class="bar" role="progressbar" aria-valuemin="0" aria-valuenow="30" aria-valuemax="100" id="bar">

    <div class="fill" id="fill"></div>

  </div>

  <button onclick="tang()">Tăng tiến trình</button>

  <script>

    let value = 30;

    function tang() {

      if (value < 100) {

        value += 10;

      }

      bar.setAttribute("aria-valuenow", value);

      fill.style.width = value + "%";

    }

  </script>

</body>

</html>

<!--

###

<!DOCTYPE html>

###

→ Trình duyệt bắt đầu đọc file HTML từ trên xuống.

→ Nhận biết đây là tài liệu HTML5.

→ Chưa có DOM hiển thị, chỉ mới khởi động luồng parse.

###

<html lang="vi">

###

→ Trình duyệt tạo node gốc HTML trong DOM.

→ lang="vi" được ghi vào DOM:

   - Screen reader biết nội dung dùng tiếng Việt.

→ Chưa render giao diện.

###

<head> … </head>

###

→ Trình duyệt đi vào HEAD.

→ Mục tiêu: đọc cấu hình (meta, title, style).

→ Không có phần tử hiển thị.

###

<style> … </style>

###

→ Trình duyệt đọc toàn bộ CSS.

→ Lưu rule CSS vào bộ nhớ:

   - body

   - .bar

   - .fill

   - button

→ CSS CHƯA áp ngay vì BODY chưa dựng xong.

================================================

BẮT ĐẦU DỰNG DOM BODY

================================================

###

<body>

###

→ Trình duyệt bắt đầu tạo DOM hiển thị.

→ CSS body được áp ngay khi có node body:

   - nền tối

   - chữ sáng

   - font monospace

###

<h3>Progress test (aria-valuemax)</h3>

###

→ Tạo node h3 trong DOM.

→ Text hiển thị trên màn hình.

→ Không liên quan logic JS hay ARIA.

###

<div class="bar" role="progressbar"

     aria-valuemin="0"

     aria-valuenow="30"

     aria-valuemax="100"

     id="bar">

###

→ Tạo element div trong DOM.

→ id="bar":

   - Trình duyệt tạo biến toàn cục bar → trỏ tới element này.

→ class="bar":

   - CSS .bar áp:

     width 300px

     height 24px

     border + background

→ role="progressbar":

   - Screen reader hiểu đây là thanh tiến trình.

→ aria-valuemin="0":

   - Mốc nhỏ nhất.

→ aria-valuemax="100":

   - Mốc lớn nhất.

→ aria-valuenow="30":

   - Trạng thái hiện tại = 30.

→ DOM lúc này ghi nhớ: 30 / 100.

###

<div class="fill" id="fill"></div>

###

→ Tạo div con bên trong bar.

→ id="fill":

   - Trình duyệt tạo biến toàn cục fill.

→ class="fill":

   - CSS .fill áp width = 30%.

→ Thanh xanh hiển thị chiếm 30% chiều dài bar.

→ Đây là phần hiển thị trực quan, không có ARIA.

###

</div>

###

→ Kết thúc progressbar.

→ DOM cấu trúc hoàn chỉnh.

###

<button onclick="tang()">Tăng tiến trình</button>

###

→ Tạo button trong DOM.

→ onclick="tang()":

   - Gắn handler click.

→ JS chưa chạy, chỉ chờ tương tác người dùng.

================================================

TRÌNH DUYỆT ĐỌC JAVASCRIPT

================================================

###

let value = 30;

###

→ JS tạo biến value trong bộ nhớ.

→ value = 30.

→ Logic JS và DOM đang đồng bộ:

   - value = 30

   - aria-valuenow = 30

   - fill width = 30%

###

function tang() { … }

###

→ Trình duyệt lưu hàm tang.

→ Chưa thực thi.

================================================

TRẠNG THÁI SAU KHI LOAD XONG

================================================

→ DOM:

   - bar aria-valuenow = 30

   - fill width = 30%

→ CSS:

   - giao diện hiển thị đúng

→ ARIA:

   - screen reader đọc “30 trên 100”

→ JS:

   - value = 30

================================================

NGƯỜI DÙNG CLICK BUTTON

================================================

###

<button onclick="tang()">

###

→ Người dùng click.

→ Trình duyệt phát sinh sự kiện click.

→ Gọi hàm tang().

###

function tang() bắt đầu chạy

###

→ JS call stack nhảy vào tang().

→ Dòng đầu tiên được thực thi là if.

###

if (value < 100)

###

→ So sánh:

   - value hiện tại = 30

   - 30 < 100

→ Kết quả so sánh sinh ra TRUE tại thời điểm này.

→ (Ở đây KHÔNG có === vì chỉ là phép so sánh <).

###

value += 10;

###

→ JS là thằng thay đổi trạng thái.

→ Trước khi đổi:

   - value = 30

→ Sau khi đổi:

   - value = 40

→ DOM chưa đổi.

###

bar.setAttribute("aria-valuenow", value);

###

→ dấu . :

   - truy cập method setAttribute của element bar.

→ setAttribute:

   - Trước khi ghi:

     aria-valuenow = "30"

   - Sau khi ghi:

     aria-valuenow = "40"

→ DOM ARIA được cập nhật.

###

fill.style.width = value + "%";

###

→ dấu . đầu:

   - truy cập element fill.

→ dấu . thứ hai:

   - truy cập object style của fill.

→ Gán width = "40%".

→ Trước khi đổi:

   - fill width = 30%

→ Sau khi đổi:

   - fill width = 40%

================================================

SAU KHI DOM THAY ĐỔI

================================================

###

DOM trước / sau

###

→ Trước:

   - aria-valuenow = 30

   - fill width = 30%

→ Sau:

   - aria-valuenow = 40

   - fill width = 40%

###

CSS phản ứng

###

→ Trình duyệt repaint.

→ Thanh xanh dài ra tương ứng 40%.

###

ARIA phản ứng

###

→ Screen reader cập nhật:

   - “40 trên 100”.

================================================

CLICK NHIỀU LẦN

================================================

→ Mỗi click:

   - value tăng thêm 10.

→ Khi value = 100:

   - if (value < 100) → FALSE

   - value không tăng nữa.

→ Tiến trình dừng ở mức tối đa.

================================================

TỔNG KẾT VAI TRÒ

================================================

HTML:

→ Tạo cấu trúc và khai báo ARIA.

CSS:

→ Quyết định hình dạng và màu sắc.

JS:

→ Điều khiển logic và cập nhật DOM.

DOM:

→ Lưu trạng thái giao diện hiện tại.

ARIA:

→ Giúp máy đọc màn hình hiểu đúng tiến trình.

-->

=== === === === === THUỘC TÍNH: aria-valuemin === === === === ===

145. Thuộc tính aria-valuemin

→ DÒNG NÀY QUAN TRỌNG:

→ XÁC ĐỊNH GIÁ TRỊ NHỎ NHẤT mà một thành phần dạng “giá trị thay đổi” có thể có

→ Dùng cho máy đọc màn hình, KHÔNG phải để vẽ giao diện

------------------------------------------------

145.1. BẢN CHẤT (DÙNG ĐỂ LÀM GÌ)

- Xác định MỐC DƯỚI (giá trị thấp nhất) cho progress / slider / range

- Giúp screen reader hiểu: “giá trị bắt đầu từ đâu”

→ Nói ngắn gọn:

aria-valuemin = điểm bắt đầu của thang đo

------------------------------------------------

145.2. PHẠM VI ÁP DỤNG

- ARIA attribute (thuộc nhóm Accessibility)

- Dùng cho các role có giá trị:

+ role="progressbar"

+ role="slider"

+ role="spinbutton"

+ role="scrollbar"

→ KHÔNG dùng cho thẻ thường không có giá trị thay đổi

------------------------------------------------

145.3. CÚ PHÁP (PHẦN QUAN TRỌNG – GHI NHỚ)

<tag

role="progressbar"

aria-valuemin="0"

>

→ aria-valuemin:

- Luôn là SỐ

- Viết dạng CHUỖI (trong HTML)

------------------------------------------------

145.4. GIÁ TRỊ THƯỜNG DÙNG

- "0" → phổ biến nhất (bắt đầu từ 0%)

- "1" → nếu không muốn có 0

- "-10" → vẫn hợp lệ (nếu logic cho phép)

→ Không bắt buộc phải là 0

→ Nhưng PHẢI nhỏ hơn aria-valuemax

------------------------------------------------

145.5. MỐI LIÊN HỆ QUAN TRỌNG (RẤT DỄ NHẦM)

aria-valuemin → giá trị nhỏ nhất

aria-valuenow → giá trị hiện tại

aria-valuemax → giá trị lớn nhất

→ Screen reader đọc theo logic:

“aria-valuenow trên aria-valuemax, bắt đầu từ aria-valuemin”

------------------------------------------------

145.6. LƯU Ý / LỖI HAY GẶP

- ❌ aria-valuemin > aria-valuemax → SAI NGỮ NGHĨA

- ❌ aria-valuenow < aria-valuemin → SAI TRẠNG THÁI

- ❌ Dùng aria-valuemin nhưng KHÔNG có role phù hợp → screen reader bỏ qua

→ aria-valuemin KHÔNG làm thay đổi giao diện

→ CSS KHÔNG đọc aria-valuemin

→ JS chỉ đọc nếu bạn chủ động dùng

------------------------------------------------

145.7. KÝ HIỆU / NGỮ NGHĨA CẦN NHỚ

aria- → dành cho accessibility

value → liên quan đến giá trị số

min → mốc thấp nhất

→ Ghép lại:

aria-valuemin = mốc giá trị thấp nhất cho máy đọc màn hình

------------------------------------------------

145.8. TÓM TẮT 1 DÒNG (ĐỂ NHỚ NHANH)

aria-valuemin:

→ nói cho screen reader biết: “thanh này bắt đầu từ số bao nhiêu”

145.9.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ aria-valuemin</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    .bar {

      width: 300px;

      height: 20px;

      border: 1px solid #30363d;

      background: #161b22;

    }

    .fill {

      height: 100%;

      width: 40%;

      background: #238636;

    }

  </style>

</head>

<body>

  <h3>Progress có aria-valuemin</h3>

  <div class="bar" role="progressbar" aria-valuemin="0" aria-valuenow="40" aria-valuemax="100">

    <div class="fill"></div>

  </div>

</body>

</html>

=== === === === === THUỘC TÍNH: aria-valuenow === === === === ===

146. Thuộc tính aria-valuenow

→ DÒNG NÀY QUAN TRỌNG: aria-valuenow cho biết \*\*GIÁ TRỊ HIỆN TẠI ĐANG Ở ĐÂU\*\* trên một thang đo (progress, slider…)

--------------------------------------------------

146.1. Bản chất (dùng để làm gì)

- Báo cho MÁY ĐỌC MÀN HÌNH biết \*\*trạng thái hiện tại\*\*

- Nói rõ: “bây giờ đang tới mức bao nhiêu rồi”

👉 Tóm gọn:

- aria-valuemin = điểm bắt đầu

- aria-valuemax = điểm kết thúc

- aria-valuenow = \*\*điểm đang đứng hiện tại (QUAN TRỌNG NHẤT)\*\*

--------------------------------------------------

146.2. Phạm vi áp dụng

- ARIA attribute

- Dùng cho các role có giá trị số:

  + progressbar

  + slider

  + spinbutton

  + scrollbar

⚠️ Không dùng bừa cho div thường KHÔNG có role phù hợp

--------------------------------------------------

146.3. Cú pháp (PHẦN CẦN TIN CẬY)

<tag

  role="progressbar"

  aria-valuemin="0"

  aria-valuenow="40"

  aria-valuemax="100"

>

✔ aria-valuenow:

- BẮT BUỘC là SỐ

- Nằm GIỮA valuemin và valuemax

--------------------------------------------------

146.4. Giá trị thường dùng

- Số nguyên: 0, 10, 50, 100

- Hoặc số thập phân nếu cần: 2.5, 7.8

❌ Không dùng chữ: "half", "medium"

❌ Không dùng % (phần trăm)

--------------------------------------------------

146.5. Lưu ý / lỗi hay gặp (RẤT HAY DÍNH)

- aria-valuenow > aria-valuemax ❌ (SAI NGỮ NGHĨA)

- aria-valuenow < aria-valuemin ❌

- Có aria-valuenow nhưng KHÔNG có role ❌

- Thanh chạy bằng CSS nhưng KHÔNG cập nhật aria-valuenow ❌

👉 Nhớ câu này:

CSS để MẮT THẤY – aria-valuenow để MÁY HIỂU

--------------------------------------------------

146.6. Ngữ nghĩa máy đọc màn hình hiểu

Ví dụ:

aria-valuemin="0"

aria-valuenow="40"

aria-valuemax="100"

→ Screen reader đọc:

“Progress bar, 40 trên 100”

--------------------------------------------------

146.7. Ghi nhớ nhanh (1 dòng)

aria-valuenow = \*\*GIÁ TRỊ HIỆN TẠI\*\*, là cái mà screen reader QUAN TÂM NHẤT

--------------------------------------------------

146.8. Liên hệ thực tế (dễ hình dung)

- Thanh tải file

- Thanh pin

- Thanh âm lượng

- Thanh % hoàn thành bài học

→ Mắt nhìn thấy độ dài

→ Máy đọc màn hình nhìn aria-valuenow

--------------------------------------------------

TỔNG KẾT NHANH:

- HTML: tạo cấu trúc

- CSS: vẽ hình, độ dài

- JS: cập nhật số

- ARIA (aria-valuenow): nói cho MÁY BIẾT “đang tới đâu”

- DOM: nơi tất cả các giá trị đang sống

146.9.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ aria-valuenow</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    .bar {

      width: 300px;

      height: 24px;

      border: 1px solid #30363d;

      background: #161b22;

    }

    .fill {

      height: 100%;

      width: 40%;

      background: #238636;

    }

  </style>

</head>

<body>

  <h3>Progress (aria-valuenow)</h3>

  <div

    class="bar"

    role="progressbar"

    aria-valuemin="0"

    aria-valuenow="40"

    aria-valuemax="100"

  >

    <div class="fill"></div>

  </div>

</body>

</html>

HẾT PHẦN ARIA

=== === === === === THUỘC TÍNH: onclick === === === === ===

147. Thuộc tính onclick

→ QUAN TRỌNG: onclick là cầu nối TRỰC TIẾP giữa hành động người dùng và JavaScript

147.1. Bản chất (dùng để làm gì)

- Bắt sự kiện CLICK của người dùng

- Gọi hàm JavaScript NGAY LÚC click xảy ra

147.2. Phạm vi áp dụng (Global / Event / ARIA / Tag-specific)

- Event attribute (thuộc nhóm sự kiện)

- Dùng được cho nhiều thẻ: button, div, span, img, a, v.v.

147.3. Cú pháp (PHẦN RẤT QUAN TRỌNG)

- Gọi hàm:

  <button onclick="tenHam()">...</button>

- Có tham số:

  <button onclick="tang(10)">...</button>

- Viết nhiều lệnh (KHÔNG khuyến khích):

  <button onclick="a(); b();">...</button>

⚠ onclick luôn viết DẠNG CHUỖI → JS sẽ được chạy khi click

147.4. Giá trị thường dùng

- Tên hàm JavaScript

- Có thể truyền tham số

- Không dùng dấu ; ở cuối cũng CHẠY

147.5. Lưu ý / lỗi hay gặp

- onclick="tang" ❌ (sai → không có dấu ())

- onclick="tang()" ✅

- Hàm PHẢI tồn tại trong JS

- onclick chạy SAU khi DOM đã render xong

- Không dùng onclick cho logic phức tạp (nên addEventListener)

147.6. Ký hiệu & ngữ nghĩa cần nhớ

- onclick = "..." → trình duyệt hiểu là MÃ JS

- () → gọi hàm

- () rỗng → gọi không tham số

- Click → Event → JS → DOM đổi → CSS / UI phản ứng

147.7. Ví dụ NGẮN – DỄ HIỂU

HTML:

<button id="btn" onclick="bao()">Bấm tôi</button>

JS:

function bao() {

  alert("Đã click");

}

→ Trước click:

- DOM đứng yên

- JS chưa chạy

→ Khi click:

- onclick phát sinh event click

- Hàm bao() được gọi

- JS chạy ngay

→ Sau click:

- UI thay đổi (alert hiện ra)

147.8. Ghi nhớ nhanh

- onclick = CÁI CÒ SÚNG

- click = BÓP CÒ

- JS = ĐẠN BAY

- DOM = BỊ TÁC ĐỘNG

=== KẾT LUẬN NGẮN ===

onclick dùng khi:

- Demo nhanh

- Học cơ bản

- Ví dụ đơn giản

Không dùng khi:

- App lớn

- Code cần sạch & tách biệt

147.9.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ onclick</title>

  <style>

    #box {

      width: 200px;

      height: 60px;

      background: #238636;

      color: white;

      text-align: center;

      line-height: 60px;

      margin-top: 10px;

    }

  </style>

</head>

<body>

  <!-- NÚT BẤM -->

  <button onclick="doiNoiDung()">Bấm vào đây</button>

  <!-- PHẦN TỬ BỊ TÁC ĐỘNG -->

  <div id="box">Chưa bấm</div>

  <script>

    function doiNoiDung() {

      document.getElementById("box").innerText = "Đã bấm rồi!";

    }

  </script>

</body>

</html>

=== === === === === THUỘC TÍNH: onmousedown === === === === ===

148. Thuộc tính onmousedown

⚠️ QUAN TRỌNG: onmousedown CHẠY NGAY KHI NHẤN CHUỘT XUỐNG

(không cần thả chuột)

--------------------------------------------------

148.1. Bản chất (dùng để làm gì)

- Bắt sự kiện KHI NGƯỜI DÙNG NHẤN CHUỘT XUỐNG

- Dùng cho phản ứng NHANH: giữ nút, kéo, nhấn giữ

👉 Khác onclick:

- onmousedown: nhấn là chạy

- onclick: nhấn + thả mới chạy

--------------------------------------------------

148.2. Phạm vi áp dụng

- Event attribute

- Dùng cho các thẻ tương tác:

  button, div, span, img, input, v.v.

--------------------------------------------------

148.3. Cú pháp (RẤT QUAN TRỌNG – GHI NHỚ)

<tag onmousedown="tenHam()"></tag>

- onmousedown = tên sự kiện

- "tenHam()" = hàm JS được gọi

- KHÔNG có dấu # hay .

--------------------------------------------------

148.4. Giá trị thường dùng

- Giá trị là:

  + Tên hàm JavaScript

  + Có thể truyền tham số

Ví dụ ngắn:

onmousedown="batDau()"

onmousedown="giuNut(1)"

--------------------------------------------------

148.5. Luồng chạy trong đầu trình duyệt

1️⃣ Người dùng NHẤN chuột xuống

2️⃣ Trình duyệt phát sinh sự kiện mousedown

3️⃣ Hàm trong onmousedown được gọi NGAY

4️⃣ JS chạy → DOM đổi → UI đổi

--------------------------------------------------

148.6. Ví dụ minh họa (để HIỂU NGAY)

<button onmousedown="giu()">Giữ chuột</button>

👉 Khi CHƯA thả chuột:

- Hàm giu() ĐÃ chạy rồi

--------------------------------------------------

148.7. Lưu ý / lỗi hay gặp (RẤT QUAN TRỌNG)

❌ Nhầm onmousedown với onclick

❌ Quên () sau tên hàm

❌ Dùng cho hành động cần xác nhận (nên dùng onclick)

--------------------------------------------------

148.8. Khi nào NÊN dùng onmousedown

✔ Nút bấm giữ để tăng / giảm

✔ Game (giữ để hành động)

✔ Kéo thanh (drag)

✔ Phản hồi tức thì

--------------------------------------------------

148.9. Ghi nhớ nhanh (1 dòng)

onmousedown = NHẤN XUỐNG LÀ CHẠY, CHƯA CẦN THẢ

148.10.EXAMLBE onmousedown and onmouseup

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test onmousedown</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    button {

      padding: 12px 20px;

      font-size: 16px;

      cursor: pointer;

    }

    #log {

      margin-top: 20px;

      border: 1px solid #30363d;

      padding: 10px;

      min-height: 80px;

    }

  </style>

</head>

<body>

  <h3>Test onmousedown / onmouseup / onclick</h3>

  <button onmousedown="xuong()" onmouseup="tha()" onclick="clickNut()">

    Nhấn giữ nút này

  </button>

  <div id="log">Log sự kiện:</div>

  <script>

    function xuong() {

      log.innerHTML += "<br>🟢 onmousedown: NHẤN XUỐNG";

    }

    function tha() {

      log.innerHTML += "<br>🟡 onmouseup: THẢ CHUỘT";

    }

    function clickNut() {

      log.innerHTML += "<br>🔵 onclick: NHẤN + THẢ XONG";

    }

  </script>

</body>

</html>

<!--

######################################################################

MÔ PHỎNG LUỒNG CHẠY THỰC TẾ – DEBUG TRONG ĐẦU TRÌNH DUYỆT

######################################################################

###

<!DOCTYPE html>

###

→ Trình duyệt nhận biết: đây là tài liệu HTML5

→ Bật chế độ parse HTML chuẩn, tạo cây DOM theo chuẩn hiện đại

###

<html lang="vi">

###

→ Trình duyệt tạo NODE gốc <html>

→ Gắn ngôn ngữ "vi" để:

  - Screen reader đọc tiếng Việt

  - Trình duyệt tối ưu font / phát âm

======================================================================

GIAI ĐOẠN 1: TRANG VỪA LOAD

======================================================================

###

<head> + <style>...</style>

###

→ Trình duyệt đọc CSS TRƯỚC

→ Chưa vẽ gì lên màn hình

→ CSS được lưu vào bộ nhớ chờ DOM

###

<body>

###

→ Bắt đầu tạo DOM hiển thị

###

<h3>Test onmousedown / onmouseup / onclick</h3>

###

→ DOM có thêm 1 node tiêu đề

→ CSS mặc định + font Consolas được áp

###

<button onmousedown="xuong()" onmouseup="tha()" onclick="clickNut()">

###

→ DOM tạo 1 phần tử button

→ Trạng thái ban đầu:

  - Chưa nhấn

  - Chưa có sự kiện chạy

→ Trình duyệt:

  - Ghi nhớ: khi mousedown → gọi hàm xuong

  - Ghi nhớ: khi mouseup   → gọi hàm tha

  - Ghi nhớ: khi click     → gọi hàm clickNut

###

<div id="log">Log sự kiện:</div>

###

→ DOM tạo 1 div có id="log"

→ id="log" được đưa vào bảng tra cứu DOM

→ JS có thể truy cập trực tiếp bằng tên `log`

###

<script>

###

→ Trình duyệt chuyển sang chế độ chạy JavaScript

###

function xuong() { ... }

###

→ Trình duyệt ghi nhớ hàm xuong

→ CHƯA chạy

###

function tha() { ... }

###

→ Ghi nhớ hàm tha

→ CHƯA chạy

###

function clickNut() { ... }

###

→ Ghi nhớ hàm clickNut

→ CHƯA chạy

→ KẾT THÚC LOAD TRANG

→ DOM ổn định

→ CSS đã áp

→ Không có JS nào tự chạy

======================================================================

GIAI ĐOẠN 2: NGƯỜI DÙNG TƯƠNG TÁC

======================================================================

###

Người dùng NHẤN CHUỘT XUỐNG nút

###

→ Sự kiện phát sinh: mousedown

→ Trình duyệt tra DOM:

  - Button có onmousedown → gọi xuong()

###

log.innerHTML += "... onmousedown ..."

###

→ Dấu "." :

  - log.innerHTML

  - truy cập thuộc tính innerHTML của phần tử DOM có id="log"

→ TRƯỚC KHI ĐỔI:

  - log.innerHTML = "Log sự kiện:"

→ JS là thằng đổi:

  - cộng thêm chuỗi HTML mới

→ SAU KHI ĐỔI:

  - log có thêm 1 dòng 🟢

→ CSS phản ứng:

  - div #log cao hơn

  - trình duyệt reflow nhẹ

======================================================================

###

Người dùng GIỮ CHUỘT (chưa thả)

###

→ KHÔNG có sự kiện mới

→ onclick CHƯA chạy

→ DOM KHÔNG đổi

======================================================================

###

Người dùng THẢ CHUỘT

###

→ Sự kiện phát sinh: mouseup

→ Trình duyệt gọi hàm tha()

###

log.innerHTML += "... onmouseup ..."

###

→ DOM bị thay đổi lần 2

→ TRƯỚC:

  - log có 1 dòng

→ SAU:

  - log có thêm 🟡

→ CSS:

  - cập nhật layout

======================================================================

###

Sau mouseup → trình duyệt kiểm tra điều kiện click

###

→ Điều kiện click hợp lệ:

  - nhấn xuống + thả ra trên CÙNG button

→ onclick được kích hoạt

###

log.innerHTML += "... onclick ..."

###

→ DOM đổi lần 3

→ TRƯỚC:

  - log có 2 dòng

→ SAU:

  - log có thêm 🔵

→ CSS:

  - vẽ lại vùng log

======================================================================

GIAI ĐOẠN 3: TỔNG KẾT TRẠNG THÁI

======================================================================

→ THỨ TỰ SỰ KIỆN THỰC TẾ:

  1. onmousedown  (NHẤN XUỐNG)

  2. onmouseup    (THẢ RA)

  3. onclick      (HOÀN TẤT 1 CLICK)

→ onclick KHÔNG BAO GIỜ chạy nếu:

  - nhấn rồi kéo chuột ra ngoài

  - hoặc không thả chuột

======================================================================

VAI TRÒ CUỐI CÙNG

======================================================================

HTML:

- Tạo cấu trúc

- Khai báo sự kiện

CSS:

- Quyết định nhìn thấy thế nào

- Phản ứng khi DOM đổi

JS:

- LÀ THẰNG GÂY RA THAY ĐỔI

- Ghi nội dung mới vào DOM

DOM:

- Là trạng thái hiện tại của trang

- Bị JS thay đổi

- Bị CSS đọc lại

ARIA:

- (Không dùng trong ví dụ này)

- Nếu có: screen reader sẽ đọc lại khi DOM đổi

======================================================================

KẾT LUẬN NGẮN GỌN

======================================================================

→ onmousedown = PHẢN XẠ NHANH (vừa chạm)

→ onmouseup   = KẾT THÚC HÀNH ĐỘNG

→ onclick     = CHỈ chạy khi đủ điều kiện

→ Đọc log là thấy ngay luồng suy nghĩ của trình duyệt

-->

=== === === === === HẾT === === === === ===

=== === === === === THUỘC TÍNH: onmouseup === === === === ===

149. Thuộc tính onmouseup

→ QUAN TRỌNG: Sự kiện XẢY RA KHI NGƯỜI DÙNG THẢ CHUỘT (NHẢ NÚT CHUỘT)

----------------------------------------------------------------------

149.1. BẢN CHẤT (DÙNG ĐỂ LÀM GÌ)

- Bắt thời điểm người dùng THẢ CHUỘT

- Dùng để xác nhận: hành động nhấn đã KẾT THÚC

→ Hiểu nhanh:

  onmousedown = bắt đầu nhấn

  onmouseup   = kết thúc nhấn

  onclick     = nhấn + thả HỢP LỆ

----------------------------------------------------------------------

149.2. PHẠM VI ÁP DỤNG

- Event attribute (thuộc tính sự kiện)

- Áp dụng cho:

  button, div, img, input, hầu hết phần tử HTML

----------------------------------------------------------------------

149.3. CÚ PHÁP (RẤT QUAN TRỌNG – GHI CHUẨN)

<tag onmouseup="tenHam()"></tag>

LƯU Ý:

- Giá trị là TÊN HÀM JS

- Có dấu () để gọi hàm

- KHÔNG viết code JS dài trong onmouseup

----------------------------------------------------------------------

149.4. THỜI ĐIỂM KÍCH HOẠT (CỰC KỲ QUAN TRỌNG)

- CHỈ chạy khi:

  ✔ Chuột đã được NHẤN TRƯỚC ĐÓ

  ✔ Sau đó người dùng THẢ CHUỘT

- KHÔNG quan tâm:

  - có click hay không

  - có giữ lâu hay không

----------------------------------------------------------------------

149.5. THỨ TỰ SỰ KIỆN THỰC TẾ

1. onmousedown  → nhấn chuột xuống

2. onmouseup    → thả chuột ra

3. onclick      → chỉ chạy nếu 1 & 2 hợp lệ

→ onmouseup LUÔN chạy TRƯỚC onclick

----------------------------------------------------------------------

149.6. LỖI / HIỂU SAI HAY GẶP

- Tưởng onmouseup = onclick ❌

- Viết logic click quan trọng vào onmouseup ❌

- Không có onmousedown mà mong onmouseup xử lý đầy đủ ❌

----------------------------------------------------------------------

149.7. KHI NÀO NÊN DÙNG onmouseup

- Cần biết lúc người dùng THẢ tay

- Kết thúc kéo / giữ / nhấn

- Dừng hiệu ứng đang chạy khi giữ chuột

----------------------------------------------------------------------

149.8. KÝ HIỆU / NGỮ NGHĨA CẦN NHỚ

- mouse = chuột

- up    = nhấc lên

→ onmouseup = sự kiện chuột được NHẢ RA

----------------------------------------------------------------------

149.9. GHI NHỚ 1 DÒNG (ĐỂ THUỘC)

→ onmouseup = ĐIỂM KẾT THÚC CỦA HÀNH ĐỘNG NHẤN CHUỘT

----------------------------------------------------------------------

TÓM GỌN SIÊU NHANH

- Không phải click

- Chạy khi thả chuột

- Thường đi chung với onmousedown

- onclick chỉ là hệ quả sau cùng

149.10EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test onmouseup</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    button {

      padding: 14px 22px;

      font-size: 16px;

      cursor: pointer;

    }

    #log {

      margin-top: 20px;

      border: 1px solid #30363d;

      padding: 10px;

      min-height: 80px;

    }

  </style>

</head>

<body>

  <h3>Ví dụ onmouseup</h3>

  <button onmousedown="nhan()" onmouseup="tha()">

    Nhấn và thả chuột

  </button>

  <div id="log">Log:</div>

  <script>

    function nhan() {

      log.innerHTML += "<br>🟢 onmousedown: Chuột ĐÃ NHẤN";

    }

    function tha() {

      log.innerHTML += "<br>🟡 onmouseup: Chuột ĐÃ THẢ";

    }

  </script>

</body>

</html>

=== === === === === THUỘC TÍNH: onmouseenter === === === === ===

(150) — QUAN TRỌNG: sự kiện CHỈ chạy 1 lần khi chuột VỪA ĐI VÀO phần tử

150. Thuộc tính onmouseenter

→ Ghi nhớ nhanh: KHÔNG cần click, chỉ cần đưa chuột vào là chạy

150.1. Bản chất (dùng để làm gì)

- Bắt khoảnh khắc chuột VỪA ĐI VÀO vùng của 1 phần tử

- Dùng để phát hiện “hover lần đầu” (tooltip, highlight, log…)

150.2. Phạm vi áp dụng

- Event attribute (thuộc nhóm sự kiện chuột)

- Dùng cho hầu hết thẻ HTML (div, button, img, …)

150.3. Cú pháp (RẤT QUAN TRỌNG)

<tag onmouseenter="tenHam()"></tag>

Ví dụ cú pháp đúng:

<div id="box" onmouseenter="vao()">...</div>

→ onmouseenter = gọi HÀM JS

→ KHÔNG viết logic JS trực tiếp dài dòng trong attribute

150.4. Cách hoạt động (hiểu ngay)

- Chuột từ NGOÀI → VÀO phần tử  → chạy 1 lần

- Chuột di chuyển bên TRONG phần tử → KHÔNG chạy lại

- Chuột ra ngoài rồi vào lại → chạy lại

150.5. Khác gì cái dễ nhầm (NOTE CỰC QUAN TRỌNG)

- onmouseenter ❌ KHÔNG bị ảnh hưởng bởi phần tử con

- onmouseover  ⚠️ CÓ thể chạy lại khi đi qua con bên trong

→ Muốn ổn định, ít bug hover → DÙNG onmouseenter

150.6. Giá trị thường dùng

- Gọi hàm JS: onmouseenter="show()"

- Kết hợp id để xử lý đúng 1 phần tử cụ thể

150.7. Lưu ý / lỗi hay gặp

- Không tự chạy khi load trang

- Chỉ chạy khi có hành động chuột

- Không thay thế được onclick (không phải click)

150.8. Ví dụ NGẮN – DỄ NHỚ

<div id="box" onmouseenter="vao()">

→ Khi chuột vừa vào #box

→ Trình duyệt gọi hàm vao()

150.9. Khi nào NÊN dùng

- Hiệu ứng hover

- Hiển thị gợi ý

- Log hành vi người dùng

- Tránh spam event như mouseover

=== TÓM GỌN 1 DÒNG ===

onmouseenter = “Chuột vừa chạm vào → chạy đúng 1 lần”

150.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test onmouseenter</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    #box {

      width: 200px;

      height: 120px;

      border: 2px solid #30363d;

      display: flex;

      align-items: center;

      justify-content: center;

      cursor: pointer;

    }

    #log {

      margin-top: 15px;

    }

  </style>

</head>

<body>

  <h3>Ví dụ onmouseenter</h3>

  <div id="box" onmouseenter="vao()">

    Đưa chuột vào đây

  </div>

  <div id="log">Trạng thái: chưa vào</div>

  <script>

    function vao() {

      log.innerText = "🟢 Chuột vừa ĐI VÀO box";

    }

  </script>

</body>

</html>

=== === === === === THUỘC TÍNH: onmouseleave === === === === ===

151. Thuộc tính onmouseleave

QUAN TRỌNG:

→ Sự kiện này CHỈ chạy khi chuột RỜI KHỎI phần tử

→ Không bị ảnh hưởng bởi phần tử con bên trong

151.1. Bản chất (dùng để làm gì)

- Bắt khoảnh khắc chuột RỜI RA khỏi một phần tử

- Dùng để reset trạng thái, ẩn UI, dừng hiệu ứng hover

151.2. Phạm vi áp dụng (Global / Event / ARIA / Tag-specific)

- Event (sự kiện chuột)

- Áp dụng cho hầu hết thẻ HTML hiển thị (div, button, img, a...)

151.3. Cú pháp (rất hay dùng – cần nhớ kỹ)

<tag onmouseleave="tenHam()"></tag>

→ onmouseleave gắn TRỰC TIẾP vào thẻ HTML

→ Khi chuột rời khỏi vùng thẻ → hàm JS được gọi

151.4. Thời điểm kích hoạt

- Chuột đang nằm TRONG phần tử

- Di chuyển chuột RA NGOÀI biên phần tử

→ onmouseleave chạy NGAY LẬP TỨC

151.5. Điểm KHÁC BIỆT quan trọng (dễ nhầm)

- onmouseleave ❌ KHÔNG chạy khi rê chuột qua phần tử con

- onmouseout ✅ CÓ thể chạy khi đi vào phần tử con

👉 Muốn hành vi “ra khỏi KHỐI CHA thật sự” → dùng onmouseleave

151.6. Lưu ý / lỗi hay gặp

- Không chạy nếu chuột chưa từng đi vào phần tử

- Không dùng để bắt hover (mouseenter phù hợp hơn)

- Dùng chung cặp chuẩn: onmouseenter / onmouseleave

151.7. Ký hiệu / từ khóa cần nhớ

- mouse = chuột

- leave = rời đi

→ on + mouse + leave = chuột rời khỏi

151.8. Kết hợp hay dùng trong thực tế

- onmouseenter → bật hiệu ứng

- onmouseleave → tắt / trả về trạng thái ban đầu

151.9. Ghi nhớ nhanh (1 dòng)

→ onmouseleave = “chuột vừa RỜI khỏi phần tử”

151.10.EXAMBLE onmouseenter VA onmouseleave

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test onmouseleave</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    #box {

      width: 260px;

      height: 120px;

      border: 2px solid #30363d;

      display: flex;

      align-items: center;

      justify-content: center;

      background: #161b22;

      cursor: pointer;

    }

    #log {

      margin-top: 15px;

      border: 1px solid #30363d;

      padding: 10px;

      min-height: 60px;

    }

  </style>

</head>

<body>

  <h3>Test onmouseenter / onmouseleave</h3>

  <div id="box" onmouseenter="vao()" onmouseleave="ra()">

    Rê chuột vào / ra

  </div>

  <div id="log">Log:</div>

  <script>

    function vao() {

      log.innerHTML += "<br>🟢 onmouseenter: CHUỘT VÀO";

    }

    function ra() {

      log.innerHTML += "<br>🔴 onmouseleave: CHUỘT RA";

    }

  </script>

</body>

</html>

=== === === === === THUỘC TÍNH: onwheel === === === === ===

1. Thuộc tính onwheel

- Rất quan trọng nếu dùng làm dự án như servo

⚠ Quan trọng: onwheel chỉ chạy khi NGƯỜI DÙNG LĂN BÁNH XE CHUỘT

152.1. Bản chất (để làm gì)

- Bắt hành động LĂN CHUỘT (lên / xuống / ngang)

- Dùng để điều khiển zoom, cuộn, tăng giảm giá trị, hiệu ứng theo wheel

152.2. Phạm vi áp dụng

- Event (sự kiện chuột)

- Áp dụng cho hầu hết thẻ HTML (div, body, input, img, …)

152.3. Cú pháp (RẤT QUAN TRỌNG)

<tag onwheel="tenHam()"></tag>

- onwheel = tên sự kiện

- "tenHam()" = hàm JS được gọi KHI LĂN CHUỘT

152.4. Thời điểm kích hoạt

- Chạy NGAY KHI bánh xe chuột quay

- Quay lên / quay xuống đều kích hoạt

- KHÔNG cần click chuột

152.5. Thứ tự luồng chạy (hiểu nhanh)

- Người dùng lăn chuột

→ Trình duyệt phát sinh sự kiện wheel

→ onwheel được kích hoạt

→ JavaScript chạy hàm được gắn

152.6. Những thứ quan trọng cần biết

- onwheel ≠ onscroll

  + onwheel: bắt HÀNH ĐỘNG lăn chuột

  + onscroll: bắt KẾT QUẢ cuộn xong

- onwheel có thể chạy RẤT NHIỀU LẦN khi lăn mạnh

- Thường dùng kèm:

  + zoom ảnh

  + tăng / giảm số

  + hiệu ứng mượt theo wheel

152.7. Lỗi hay gặp

- Gắn onwheel nhưng thử bằng touchpad mà tưởng không chạy

- Lăn chuột ngoài vùng phần tử → sự kiện KHÔNG kích hoạt

- Quên rằng onwheel chạy liên tục → log bị spam

152.8. Ghi nhớ nhanh (1 câu)

- onwheel = “bắt chuyển động bánh xe chuột, không cần click”

152.9. Kết hợp thường thấy

- id  → xác định đúng phần tử cần bắt wheel

- JS  → xử lý logic tăng / giảm / zoom

- CSS → phản ứng lại bằng hiệu ứng

=== KẾT LUẬN NGẮN GỌN ===

onwheel dùng khi bạn muốn UI PHẢN ỨNG THEO LĂN CHUỘT

152.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test onwheel</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    #box {

      width: 300px;

      height: 150px;

      border: 2px solid #30363d;

      display: flex;

      align-items: center;

      justify-content: center;

      margin-bottom: 15px;

      user-select: none;

    }

    #log {

      border: 1px solid #30363d;

      padding: 10px;

      min-height: 80px;

    }

  </style>

</head>

<body>

  <h3>Test sự kiện onwheel</h3>

  <div id="box" onwheel="lanChuot(event)">

    Lăn chuột trong khung này

  </div>

  <div id="log">Log:</div>

  <script>

    function lanChuot(e) {

      if (e.deltaY < 0) {

        log.innerHTML += "<br>⬆️ Lăn chuột LÊN";

      } else {

        log.innerHTML += "<br>⬇️ Lăn chuột XUỐNG";

      }

    }

  </script>

</body>

</html>

=== === === === === THUỘC TÍNH: onkeydown === === === === ===

153. Thuộc tính onkeydown

→ Sự kiện BÀN PHÍM: kích hoạt NGAY KHI PHÍM ĐƯỢC NHẤN XUỐNG

→ Dùng để bắt thao tác phím: điều khiển, nhập liệu, game, điều khiển ESP từ web

------------------------------------------------------------

153.1. Bản chất (dùng để làm gì)

- Bắt sự kiện khi người dùng NHẤN phím (chưa cần thả)

- Phù hợp cho điều khiển liên tục (giữ phím là chạy)

------------------------------------------------------------

153.2. Phạm vi áp dụng

- Event (sự kiện)

- Áp dụng cho:

  + input, textarea

  + button

  + body, document, window (bắt toàn bàn phím)

------------------------------------------------------------

153.3. Cú pháp (QUAN TRỌNG)

- Gắn trực tiếp trong HTML:

  <tag onkeydown="hamXuLy(event)">

- Hoặc bắt toàn trang:

  <body onkeydown="hamXuLy(event)">

\* LƯU Ý:

- Luôn truyền event để biết PHÍM GÌ được nhấn

------------------------------------------------------------

153.4. Giá trị / dữ liệu hay dùng

- event.key      → tên phím (a, Enter, ArrowUp…)

- event.code     → mã phím vật lý (KeyA, ArrowUp…)

- event.ctrlKey  → true nếu giữ Ctrl

- event.shiftKey → true nếu giữ Shift

------------------------------------------------------------

153.5. Thứ tự xảy ra (RẤT QUAN TRỌNG)

1. Người dùng NHẤN phím xuống

2. Trình duyệt phát sinh onkeydown

3. JavaScript được gọi NGAY

4. Nếu giữ phím → onkeydown LẶP LIÊN TỤC

→ Khác onkeyup (chỉ chạy khi THẢ phím)

------------------------------------------------------------

153.6. Lưu ý / lỗi hay gặp

- onkeydown chạy NHIỀU LẦN nếu giữ phím

- Dễ bị spam nếu không chặn logic

- Input phải ĐANG FOCUS thì mới nhận phím

- Dùng event.preventDefault() nếu muốn chặn hành vi mặc định

------------------------------------------------------------

153.7. Ý nghĩa thực tế (rất dễ nhớ)

- onkeydown = "NHẤN LÀ CHẠY LIỀN"

- Dùng cho:

  + Game (giữ phím di chuyển)

  + Điều khiển robot / ESP qua web

  + Phím tắt (Ctrl + S, Ctrl + Z…)

------------------------------------------------------------

153.8. Liên hệ ESP (ứng dụng THỰC TẾ)

- Nhấn phím W → gửi lệnh TIẾN

- Nhấn phím S → gửi lệnh LÙI

- Giữ phím → ESP chạy liên tục

- Nhả phím → dừng (kết hợp onkeyup)

------------------------------------------------------------

153.9. Ghi nhớ nhanh (1 dòng)

- onkeydown = bắt PHÍM NGAY LÚC NHẤN, giữ là chạy liên tục

153.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test onkeydown</title>

</head>

<body onkeydown="batPhim(event)">

  <h2>Nhấn bất kỳ phím nào</h2>

  <p id="kq">Chưa nhấn phím</p>

  <script>

    function batPhim(e) {

      document.getElementById("kq").innerText =

        "Bạn vừa nhấn phím: " + e.key;

    }

  </script>

</body>

</html>

153.11.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Điều khiển bằng phím</title>

</head>

<body onkeydown="dieuKhien(event)">

  <h2>Điều khiển: W A S D</h2>

  <p id="lenh">Chưa có lệnh</p>

  <script>

    function dieuKhien(e) {

      let lenh = "";

      if (e.key === "w") lenh = "TIẾN";

      if (e.key === "s") lenh = "LÙI";

      if (e.key === "a") lenh = "TRÁI";

      if (e.key === "d") lenh = "PHẢI";

      document.getElementById("lenh").innerText = lenh;

    }

  </script>

</body>

</html>

<!--

###

<!DOCTYPE html>

###

→ Trình duyệt bắt đầu tải tài liệu HTML

→ Chưa có DOM, chỉ mới đọc từng byte từ trên xuống

###

<html lang="vi">

###

→ Trình duyệt tạo NODE gốc <html>

→ Thuộc tính lang="vi" được gắn vào node này

→ DOM hiện tại: html (chưa có con)

###

<head> ... </head>

###

→ Trình duyệt tạo node <head>

→ Đọc <meta charset="UTF-8"> để biết cách giải mã chữ

→ Đọc <title> để đặt tên tab

→ KHÔNG ảnh hưởng giao diện, KHÔNG tương tác người dùng

###

<body onkeydown="dieuKhien(event)">

###

→ Trình duyệt tạo node <body>

→ Ghi nhận 1 event handler:

   onkeydown → khi có phím bấm → gọi hàm dieuKhien

→ event ở đây là OBJECT sự kiện do trình duyệt tạo

===========================================================================================

GIẢI THÍCH DẤU .

===========================================================================================

→ dieuKhien(event)

→ dấu . chưa xuất hiện ở đây

→ event là biến chứa thông tin phím bấm

###

<h2>Điều khiển: W A S D</h2>

###

→ DOM thêm node h2

→ Text hiển thị ra màn hình

→ CSS mặc định của trình duyệt áp style chữ to

###

<p id="lenh">Chưa có lệnh</p>

###

→ DOM thêm node p

→ id="lenh" là ĐỊNH DANH DUY NHẤT

→ DOM BAN ĐẦU:

   p#lenh.innerText = "Chưa có lệnh"

===========================================================================================

GIẢI THÍCH id

===========================================================================================

→ id giúp JS truy xuất CHÍNH XÁC 1 phần tử

→ Sau này document.getElementById("lenh") tìm đúng node này

###

<script>

###

→ Trình duyệt chuyển sang chế độ chạy JavaScript

→ TẠO hàm dieuKhien trong bộ nhớ

→ CHƯA chạy hàm

###

Trang load xong

###

→ DOM ổn định

→ Giao diện hiển thị

→ JS đang chờ sự kiện

========================================================================================

NGƯỜI DÙNG BẮT ĐẦU TƯƠNG TÁC

========================================================================================

###

Người dùng nhấn phím "w"

###

→ Trình duyệt phát sinh EVENT keydown

→ event object được tạo:

   event.key = "w"

###

onkeydown="dieuKhien(event)"

###

→ Trình duyệt gọi hàm dieuKhien

→ Truyền event vào tham số e

===

GIẢI THÍCH DẤU .

===

→ e.key

→ e = object sự kiện

→ .key = thuộc tính key của object đó

========================================================================================

JS BẮT ĐẦU CHẠY

========================================================================================

###

function dieuKhien(e) {

###

→ JS vào thân hàm

→ Tạo scope mới cho hàm

###

let lenh = "";

###

→ Tạo biến lenh

→ Giá trị BAN ĐẦU: chuỗi rỗng ""

###

if (e.key === "w")

###

→ So sánh:

   bên trái: e.key → "w"

   bên phải: "w"

===

GIẢI THÍCH ===

===

→ === so sánh GIÁ TRỊ + KIỂU

→ "w" === "w" → TRUE

###

lenh = "TIẾN";

###

→ Điều kiện TRUE → dòng này chạy

→ lenh đổi từ "" → "TIẾN"

###

if (e.key === "s")

###

→ "w" === "s" → FALSE

→ KHÔNG chạy

###

if (e.key === "a")

###

→ "w" === "a" → FALSE

→ KHÔNG chạy

###

if (e.key === "d")

###

→ "w" === "d" → FALSE

→ KHÔNG chạy

========================================================================================

JS GHI VÀO DOM

========================================================================================

###

document.getElementById("lenh")

###

→ document = object đại diện toàn bộ trang

→ .getElementById = phương thức tìm node

→ Trả về node <p id="lenh">

========================================================================================

GIẢI THÍCH DẤU .

========================================================================================

→ document.getElementById

→ truy cập METHOD của object document

###

.innerText = lenh

###

→ DOM TRƯỚC:

   p#lenh.innerText = "Chưa có lệnh"

→ JS là thằng ĐỔI

→ DOM SAU:

   p#lenh.innerText = "TIẾN"

========================================================================================

SAU KHI DOM ĐỔI

========================================================================================

###

CSS phản ứng

###

→ Không có CSS động

→ Chỉ re-render chữ mới

###

ARIA / Screen reader

###

→ Không có aria-\* trong code

→ Screen reader đọc lại nội dung mới của <p>

========================================================================================

TỔNG KẾT VAI TRÒ

========================================================================================

HTML → Tạo cấu trúc + gắn sự kiện

CSS → Không can thiệp logic

JS → Xử lý phím, so sánh, ghi DOM

DOM → Trạng thái trước / sau

EVENT → Cầu nối giữa người dùng và JS

-->

=== === === === === HẾT === === === === ===

=== === === === === THUỘC TÍNH: onkeyup === === === === ===

153. Thuộc tính onkeyup

⚠️ QUAN TRỌNG NGAY CHỖ NÀY:

→ onkeyup CHỈ kích hoạt KHI THẢ PHÍM

→ KHÔNG chạy lúc nhấn xuống (đó là onkeydown)

--------------------------------------------------

153.1. Bản chất (dùng để làm gì)

- Bắt sự kiện KHI NGƯỜI DÙNG THẢ PHÍM BÀN PHÍM

- Dùng để xử lý SAU KHI người dùng gõ xong 1 phím

👉 Hiểu nhanh:

- Nhấn xuống → onkeydown

- Thả ra    → onkeyup  ✅

--------------------------------------------------

153.2. Phạm vi áp dụng

- Event attribute

- Áp dụng cho:

  body, input, textarea, div (khi có focus)

--------------------------------------------------

153.3. Cú pháp (RẤT QUAN TRỌNG)

<tag onkeyup="tenHam(event)"></tag>

Ví dụ dạng chuẩn:

- onkeyup="xuLy(event)"

⚠️ event là OBJECT do trình duyệt tự tạo

→ chứa thông tin phím vừa thả

--------------------------------------------------

153.4. Giá trị thường dùng

- Gọi hàm JavaScript

- Thường dùng:

  event.key

  event.code

Ví dụ giá trị:

- "w"  "a"  "s"  "d"

- "Enter"

- "ArrowUp"

--------------------------------------------------

153.5. Luồng chạy THỰC TẾ (rất dễ nhầm)

1️⃣ Người dùng NHẤN phím

→ onkeydown (chạy trước)

2️⃣ Người dùng GIỮ phím

→ onkeydown lặp (nếu giữ lâu)

3️⃣ Người dùng THẢ phím

→ onkeyup (CHỈ CHẠY 1 LẦN) ✅

--------------------------------------------------

153.6. Ký hiệu / ngữ nghĩa cần nhớ

- event.key

  → tên phím người dùng vừa THẢ

- dấu .

  → truy cập THUỘC TÍNH của object event

- onkeyup ≠ onchange

  → onkeyup: liên quan BÀN PHÍM

  → onchange: liên quan GIÁ TRỊ thay đổi

--------------------------------------------------

153.7. Lưu ý / lỗi hay gặp

❌ Nhầm onkeyup với onkeydown

❌ Quên truyền event vào hàm

❌ Gắn onkeyup cho thẻ KHÔNG có focus

❌ Nghĩ rằng onkeyup chạy khi nhấn (SAI)

✔ Muốn bắt THẢ PHÍM → onkeyup

✔ Muốn bắt NHẤN XUỐNG → onkeydown

--------------------------------------------------

153.8. Khi nào NÊN dùng onkeyup

- Điều khiển robot / ESP bằng bàn phím

  → THẢ phím = DỪNG

- Kiểm tra dữ liệu sau khi gõ xong

- Tránh lặp sự kiện khi giữ phím

--------------------------------------------------

153.9. So nhanh với sự kiện khác

- onkeydown → phản ứng NGAY

- onkeyup   → phản ứng SAU CÙNG

- onclick   → chuột

- oninput   → thay đổi nội dung input

--------------------------------------------------

🧠 TÓM GỌN 1 CÂU:

→ onkeyup = “BẮT KHOẢNH KHẮC NGƯỜI DÙNG THẢ PHÍM”

153.10. examble

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test onkeydown / onkeyup</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    #trangthai {

      margin-top: 20px;

      padding: 15px;

      border: 1px solid #30363d;

      font-size: 20px;

    }

  </style>

</head>

<body onkeydown="nhan(event)" onkeyup="tha(event)">

  <h2>TEST ĐIỀU KHIỂN BẰNG PHÍM</h2>

  <p>Nhấn giữ: W / A / S / D</p>

  <div id="trangthai">TRẠNG THÁI: DỪNG</div>

  <script>

    function nhan(e) {

      let lenh = "DỪNG";

      if (e.key === "w") lenh = "TIẾN";

      if (e.key === "s") lenh = "LÙI";

      if (e.key === "a") lenh = "TRÁI";

      if (e.key === "d") lenh = "PHẢI";

      document.getElementById("trangthai").innerText =

        "TRẠNG THÁI: " + lenh;

    }

    function tha(e) {

      document.getElementById("trangthai").innerText =

        "TRẠNG THÁI: DỪNG";

    }

  </script>

</body>

</html>

<!--

###

Trang vừa load (HTML bắt đầu được parse)

###

Trình duyệt đọc từ <!DOCTYPE html> → biết đây là tài liệu HTML5.

Tạo cây DOM theo thứ tự:

- html

  - head

  - body

Ở giai đoạn này CHƯA có JS chạy, CHƯA có sự kiện nào phát sinh.

###

Trình duyệt tạo DOM chi tiết

###

Trong <head>:

- <meta charset="UTF-8"> → thiết lập encoding, không ảnh hưởng logic.

- <title> → chỉ ảnh hưởng tab trình duyệt.

- <style> → CSS được lưu vào bộ nhớ, CHƯA áp ngay.

Trong <body>:

- body có thuộc tính onkeydown và onkeyup

  → Trình duyệt GHI NHỚ:

    khi có phím nhấn → gọi hàm nhan(event)

    khi thả phím → gọi hàm tha(event)

- <h2>, <p> → tạo node văn bản.

- <div id="trangthai"> → tạo 1 element có id = "trangthai"

  👉 id này giúp JS truy xuất ĐÚNG phần tử này sau.

DOM ban đầu:

#trangthai.innerText = "TRẠNG THÁI: DỪNG"

###

CSS đọc DOM và áp style

###

CSS duyệt DOM:

- body → áp background, màu chữ, font.

- #trangthai → vì có dấu #

  → CSS hiểu là truy cập element có id="trangthai"

  → áp border, padding, font-size.

DOM KHÔNG đổi, chỉ có giao diện thay đổi.

###

Trình duyệt gặp <script>

###

JS được parse:

- Tạo function nhan(e)

- Tạo function tha(e)

CHƯA chạy dòng nào bên trong function

→ vì chưa có sự kiện kích hoạt.

###

Trạng thái chờ (idle)

###

Trang đứng yên.

DOM giữ nguyên:

"TRẠNG THÁI: DỪNG"

Trình duyệt lắng nghe:

- onkeydown trên body

- onkeyup trên body

###

Người dùng NHẤN GIỮ một phím (ví dụ: phím "w")

###

Hành động vật lý: ngón tay nhấn phím.

Trình duyệt phát sinh sự kiện: keydown

→ vì body có onkeydown="nhan(event)"

→ JS gọi hàm nhan(event)

###

JS bắt đầu chạy: function nhan(e)

###

Dòng chạy đầu tiên:

let lenh = "DỪNG";

→ tạo biến lenh, giá trị mặc định là "DỪNG"

###

Dòng: if (e.key === "w")

###

- e.key :

  dấu . → truy cập thuộc tính key của object event (e)

- === :

  so sánh GIÁ TRỊ và KIỂU

  so sánh: e.key (ví dụ "w") với chuỗi "w"

- Nếu đúng → true sinh ra tại đây

→ lenh = "TIẾN"

Nếu nhấn phím khác → false → bỏ qua dòng này.

###

Các if tiếp theo (s / a / d)

###

Trình duyệt lần lượt kiểm tra:

- e.key === "s" ?

- e.key === "a" ?

- e.key === "d" ?

Chỉ 1 điều kiện true, các cái còn lại false.

###

Ghi dữ liệu vào DOM

###

document.getElementById("trangthai")

- dấu . :

  truy cập method getElementById của object document

→ trả về element <div id="trangthai">

.innerText =

- dấu . :

  truy cập thuộc tính innerText của element đó

DOM TRƯỚC:

<div id="trangthai">TRẠNG THÁI: DỪNG</div>

JS ghi:

"TRẠNG THÁI: TIẾN"

DOM SAU:

<div id="trangthai">TRẠNG THÁI: TIẾN</div>

###

CSS phản ứng sau khi DOM đổi

###

CSS KHÔNG đổi rule.

Nhưng vì text thay đổi → trình duyệt re-render chữ mới.

ARIA:

- Không có aria-\* trong code

→ screen reader chỉ đọc text mới hiển thị.

###

Người dùng VẪN GIỮ phím

###

Không có keyup → không gọi hàm tha

→ trạng thái GIỮ NGUYÊN "TIẾN"

###

Người dùng THẢ phím

###

Trình duyệt phát sinh sự kiện: keyup

→ body có onkeyup="tha(event)"

→ JS gọi function tha(event)

###

JS chạy: function tha(e)

###

Không có điều kiện.

Chạy thẳng dòng:

document.getElementById("trangthai").innerText = "TRẠNG THÁI: DỪNG"

DOM TRƯỚC:

<div id="trangthai">TRẠNG THÁI: TIẾN</div>

DOM SAU:

<div id="trangthai">TRẠNG THÁI: DỪNG</div>

###

CSS / Screen reader phản ứng

###

CSS không đổi.

Trình duyệt vẽ lại text.

Screen reader đọc: "TRẠNG THÁI: DỪNG"

###

Chu kỳ hoàn tất – quay lại trạng thái chờ

###

Trang sẵn sàng nhận keydown tiếp theo.

==========================================================================================

TỔNG KẾT VAI TRÒ

========================================================================================

HTML: định nghĩa cấu trúc + nơi gắn sự kiện

CSS: chỉ lo giao diện, không đổi logic

JS: quyết định TRẠNG THÁI và ghi DOM

DOM: nơi lưu trạng thái hiện tại

ARIA: không dùng trong ví dụ này

-->

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

154. Thuộc tính oninput

→ KÍCH HOẠT NGAY KHI GIÁ TRỊ NHẬP LIỆU BỊ THAY ĐỔI (THEO THỜI GIAN THỰC)

------------------------------------------------

154.1 DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Bắt GIÁ TRỊ NGƯỜI DÙNG ĐANG NHẬP NGAY LẬP TỨC

- Dùng khi cần PHẢN ỨNG LIỀN, KHÔNG CHỜ ENTER hay rời ô

👉 Nhập tới đâu → xử lý tới đó

------------------------------------------------

154.2 BẢN CHẤT HOẠT ĐỘNG

- Là: Event (sự kiện)

- Ai kích hoạt:

  + Người dùng (gõ phím, dán, kéo thanh)

- Trình duyệt tự phát hiện GIÁ TRỊ THAY ĐỔI

- JavaScript được gọi NGAY

👉 Người dùng là thằng gây ra hành động

------------------------------------------------

154.3 PHẠM VI ÁP DỤNG

- Event-only

- Hay dùng cho:

  + input (text, number, range…)

  + textarea

- CẦN focus (con trỏ đang nằm trong ô)

- KHÔNG cần Enter

- KHÔNG cần click ra ngoài

------------------------------------------------

154.4 CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Dạng chuẩn:

  <input oninput="tenHam()">

- Ghi nhớ:

  + oninput = gọi hàm

  + Hàm chạy MỖI LẦN GIÁ TRỊ ĐỔI

- Sai hay gặp:

  + Ghi nhầm onInput (sai chữ hoa)

  + Không có dấu ngoặc ()

------------------------------------------------

154.5 GIÁ TRỊ THƯỜNG DÙNG

- Giá trị nhận được thường là:

  + Chuỗi (text)

  + Số (number)

- Ví dụ kiểu dữ liệu:

  + "123"

  + "ON"

  + 0 → 100 (thanh kéo)

👉 LẤY GIÁ TRỊ TỪ Ô NHẬP

------------------------------------------------

154.6 LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng gõ / kéo / dán

2. Giá trị ô nhập thay đổi

3. oninput được kích hoạt NGAY

4. JavaScript nhận GIÁ TRỊ MỚI

5. DOM / trạng thái / lệnh được cập nhật

6. Màn hình hoặc thiết bị phản hồi

👉 Hiểu luồng này là DEBUG ĐƯỢC

------------------------------------------------

154.7 ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Điều khiển ESP / robot / xe:

  + Thanh kéo tốc độ

  + Ô nhập góc quay

- Gửi lệnh REAL-TIME:

  + Nhập số → gửi PWM

- Hiển thị trạng thái:

  + Nhập tới đâu → hiển thị tới đó

👉 Không chờ nút bấm

------------------------------------------------

154.8 LƯU Ý / LỖI HAY GẶP

- Lỗi hay gặp:

  + Nhập 1 ký tự là chạy → gửi quá nhiều lệnh

- Dễ gây lag ESP nếu không giới hạn

- Nhớ:

  + oninput = RẤT NHẠY

👉 Dùng khi CẦN PHẢN ỨNG NGAY

------------------------------------------------

154.9 TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “oninput dùng để bắt GIÁ TRỊ NHẬP NGAY KHI NGƯỜI DÙNG ĐANG NHẬP”

154.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test oninput</title>

</head>

<body>

  <h3>Nhập gì cũng được:</h3>

  <input

    type="text"

    oninput="hienThi(this.value)"

    placeholder="Gõ thử đi">

  <p id="ketqua">Giá trị: </p>

  <script>

    function hienThi(giaTri) {

      document.getElementById("ketqua").innerText =

        "Giá trị: " + giaTri;

    }

  </script>

</body>

</html>

<!--

########################################################################################

TRÌNH DUYỆT LOAD TRANG (HTML BẮT ĐẦU ĐƯỢC ĐỌC)

########################################################################################

- Trình duyệt đọc <!DOCTYPE html> → biết đây là tài liệu HTML5

- Tạo cây DOM từ trên xuống dưới

- <html>, <head>, <meta>, <title> được đưa vào DOM

- Chưa có JS chạy, chưa có sự kiện nào xảy ra

########################################################################################

DOM BAN ĐẦU SAU KHI LOAD XONG

########################################################################################

- DOM hiện có:

  + 1 thẻ <h3> (chỉ hiển thị chữ)

  + 1 thẻ <input type="text">

    → có thuộc tính oninput

    → chưa có giá trị nhập → value = ""

  + 1 thẻ <p id="ketqua">

    → innerText ban đầu là: "Giá trị: "

- Màn hình hiển thị đúng như DOM

- Chưa có thay đổi trạng thái

########################################################################################

CSS ÁP STYLE (TRANG NÀY GẦN NHƯ KHÔNG CÓ CSS)

########################################################################################

- Không có CSS tùy chỉnh

- Trình duyệt dùng style mặc định

- Không ảnh hưởng logic, chỉ hiển thị

########################################################################################

NGƯỜI DÙNG BẮT ĐẦU GÕ PHÍM VÀO INPUT

########################################################################################

- Người dùng gõ 1 ký tự (ví dụ: "A")

- Giá trị của <input> thay đổi NGAY LẬP TỨC

- Trình duyệt phát sinh sự kiện: input event

- Vì input có oninput → sự kiện được kích hoạt

###

oninput="hienThi(this.value)"

###

- oninput gọi hàm hienThi(...)

- this:

  → đại diện cho CHÍNH thẻ <input> đang được gõ

- dấu . :

  → this.value = truy cập thuộc tính value của đối tượng input

- value lúc này = "A"

- JS chuẩn bị chạy hàm hienThi với tham số = "A"

###

function hienThi(giaTri)

###

- JS vào hàm hienThi

- giaTri = "A" (giá trị người dùng vừa gõ)

- Không có if, không có ===

- Chạy thẳng từ trên xuống

###

document.getElementById("ketqua")

###

- document:

  → đối tượng đại diện cho toàn bộ trang HTML

- dấu . :

  → gọi hàm getElementById của document

- "ketqua":

  → tìm phần tử có id="ketqua"

- DOM TRƯỚC KHI ĐỔI:

  <p id="ketqua">Giá trị: </p>

###

.innerText = "Giá trị: " + giaTri

###

- dấu . :

  → truy cập thuộc tính innerText của thẻ <p>

- Phép nối chuỗi:

  "Giá trị: " + "A" → "Giá trị: A"

- JS là THẰNG ĐỔI DOM

- DOM SAU KHI ĐỔI:

  <p id="ketqua">Giá trị: A</p>

########################################################################################

SAU KHI DOM THAY ĐỔI

########################################################################################

- Trình duyệt phát hiện DOM bị cập nhật

- Màn hình vẽ lại phần <p>

- Người dùng thấy chữ đổi NGAY

- CSS (nếu có) chỉ phản ứng hiển thị

- Không có ARIA → screen reader chỉ đọc lại nội dung mới

########################################################################################

NGƯỜI DÙNG GÕ THÊM KÝ TỰ (VD: "B")

########################################################################################

- value của input = "AB"

- oninput lại kích hoạt

- Toàn bộ luồng trên LẶP LẠI

- Mỗi lần gõ → JS chạy → DOM đổi → màn hình cập nhật

###########################################################################################

TÓM TẮT LUỒNG CHẠY CUỐI

########################################################################################

- HTML: tạo cấu trúc và gắn sự kiện oninput

- DOM: giữ trạng thái input và thẻ p

- JS: nhận value → ghi lại vào DOM

- CSS: chỉ hiển thị (không can thiệp logic)

- oninput: phản ứng REAL-TIME mỗi lần giá trị input đổi

-->

========================================

=== MẪU PHÂN TÍCH THUỘC TÍNH HTML ===

[RQT]

155. Thuộc tính onchange

→ KÍCH HOẠT khi GIÁ TRỊ của phần tử BỊ THAY ĐỔI và ĐƯỢC XÁC NHẬN

------------------------------------------------

[155.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

• Bắt thời điểm người dùng ĐỔI giá trị (chọn xong / nhập xong)

• Dùng khi KHÔNG cần realtime, chỉ cần kết quả CUỐI

------------------------------------------------

[155.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

• Chọn chế độ AUTO / MANUAL cho xe – robot – ESP

• Chọn tốc độ, mức công suất, cấu hình

• Gửi lệnh sau khi người dùng chọn xong

Ví dụ ngắn:

• Chọn AUTO → gửi lệnh AUTO

• Đổi mức tốc độ → ESP nhận giá trị mới

-----------------------------------------------

[155.3] PHẠM VI ÁP DỤNG

• Event-only (thuộc nhóm sự kiện)

• Hay dùng cho:

  ▸ input

  ▸ select

  ▸ textarea

• ⚠️ CẦN focus (phải click / chọn thì mới chạy)

------------------------------------------------

[155.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

Cú pháp chuẩn:

<tag onchange="tenHam()">

Ví dụ ĐÚNG:

<select onchange="doiCheDo()">

Ví dụ SAI (hay gặp):

<select onChange="doiCheDo()">   ← sai chuẩn viết (JS phân biệt hoa thường)

------------------------------------------------

[155.5] GIÁ TRỊ THƯỜNG DÙNG

• Không tự mang giá trị

• Giá trị lấy từ:

  ▸ this.value

• Kiểu giá trị:

  ▸ Chuỗi: "AUTO", "MANUAL"

  ▸ Số: 0–100

------------------------------------------------

[155.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1) Người dùng chọn / nhập xong

2) Giá trị phần tử THAY ĐỔI

3) onchange được kích hoạt

4) JavaScript nhận value mới

5) DOM hiển thị trạng thái mới

👉 RẤT DỄ DEBUG vì chỉ chạy 1 lần mỗi lần đổi

------------------------------------------------

[155.7] BẢN CHẤT HOẠT ĐỘNG

• Là: Event Attribute

• Ai kích hoạt:

  ▸ Người dùng (chọn / nhập)

  ▸ Trình duyệt phát hiện giá trị đổi

  ▸ JavaScript xử lý sau

👉 Hiểu nhanh: NGƯỜI ĐỔI → TRÌNH DUYỆT BÁO → JS XỬ

------------------------------------------------

[155.8] LƯU Ý / LỖI HAY GẶP

⚠️ Lỗi 1: Nghĩ rằng gõ từng chữ là chạy → SAI

⚠️ Lỗi 2: Quên focus (chưa click ra ngoài)

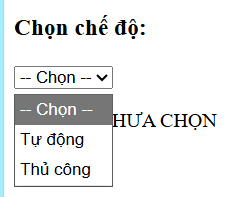
✔️ Cách nhớ nhanh:

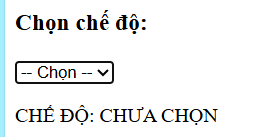
→ onchange = CHỈ chạy khi ĐỔI XONG

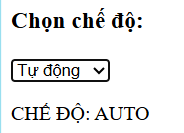
------------------------------------------------

[155.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “onchange dùng để bắt GIÁ TRỊ MỚI khi người dùng ĐỔI xong”

155.10.EXAMBLE





<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ onchange</title>

</head>

<body>

  <h3>Chọn chế độ:</h3>

  <select onchange="doiCheDo(this.value)">

    <option value="">-- Chọn --</option>

    <option value="AUTO">Tự động</option>

    <option value="MANUAL">Thủ công</option>

  </select>

  <p id="trangthai">CHẾ ĐỘ: CHƯA CHỌN</p>

  <script>

    function doiCheDo(giaTri) {

      document.getElementById("trangthai").innerText =

        "CHẾ ĐỘ: " + giaTri;

    }

  </script>

</body>

</html>

###

<!DOCTYPE html> + <html> được tải

###

Trình duyệt bắt đầu load trang → nhận ra đây là tài liệu HTML chuẩn.

Nó chuẩn bị bộ máy để parse HTML thành DOM tree.

###

<head> + <meta charset> + <title>

###

Trình duyệt đọc metadata:

- charset UTF-8 → biết cách hiển thị tiếng Việt

- title → gắn tiêu đề cho tab

CHƯA có tương tác UI, CHƯA có JS chạy.

###

<body> bắt đầu được parse

###

Trình duyệt tạo các node DOM trong bộ nhớ:

- h3

- select

- option (3 cái)

- p#trangthai

DOM ban đầu được dựng xong nhưng CHƯA có sự kiện xảy ra.

###

<select onchange="doiCheDo(this.value)">

###

Trình duyệt ghi nhớ:

- select này có gắn event onchange

- Khi GIÁ TRỊ của select THAY ĐỔI và ĐƯỢC XÁC NHẬN

→ sẽ gọi hàm doiCheDo(...)

⚠️ Lúc này CHƯA gọi hàm.

###

<p id="trangthai">CHẾ ĐỘ: CHƯA CHỌN</p>

###

DOM ban đầu:

- Text hiển thị là: "CHẾ ĐỘ: CHƯA CHỌN"

CSS mặc định trình duyệt áp style chữ thường (vì không có CSS custom).

###

<script> được load

###

Trình duyệt đọc và đăng ký hàm:

function doiCheDo(giaTri) { ... }

⚠️ Hàm CHƯA chạy, chỉ được lưu trong bộ nhớ JS.

===========================================================================================

=== GIAI ĐOẠN NGƯỜI DÙNG TƯƠNG TÁC ===

============================================================================================

###

Người dùng CLICK vào <select>

###

Select được focus.

Danh sách option xổ ra.

CHƯA có onchange vì giá trị CHƯA đổi.

###

Người dùng CHỌN option "Tự động"

###

Giá trị select đổi từ:

""  →  "AUTO"

Trình duyệt xác nhận: GIÁ TRỊ ĐÃ ĐỔI XONG

→ KÍCH HOẠT event onchange

###

onchange="doiCheDo(this.value)"

###

- this: chính là thẻ <select>

- dấu . : truy cập thuộc tính value của đối tượng select

- this.value lúc này = "AUTO"

Trình duyệt gọi:

doiCheDo("AUTO")

===========================================================================================

=== GIAI ĐOẠN JAVASCRIPT CHẠY ===

===========================================================================================

###

function doiCheDo(giaTri)

###

JS nhận tham số:

- giaTri = "AUTO"

Không có if / === trong ví dụ này.

JS chạy tuần tự từ trên xuống.

###

document.getElementById("trangthai")

###

- document: đối tượng đại diện toàn bộ DOM

- dấu . : truy cập hàm getElementById của document

- Trình duyệt tìm node có id="trangthai"

→ tìm thấy <p id="trangthai">

###

.innerText = "CHẾ ĐỘ: " + giaTri

###

- dấu . : truy cập thuộc tính innerText của thẻ p

- Trước khi đổi DOM:

  Text = "CHẾ ĐỘ: CHƯA CHỌN"

- JS ghi text mới:

  "CHẾ ĐỘ: AUTO"

===========================================================================================

=== SAU KHI DOM THAY ĐỔI ===

===========================================================================================

########################################################################################

DOM trước và sau

########################################################################################

- Trước:

  <p>CHẾ ĐỘ: CHƯA CHỌN</p>

- Sau:

  <p>CHẾ ĐỘ: AUTO</p>

########################################################################################

CSS phản ứng

########################################################################################

Không có CSS custom.

Trình duyệt repaint lại chữ mới trên màn hình.

########################################################################################

ARIA / Screen reader

########################################################################################

- Screen reader đọc lại nội dung mới của <p>

- Người dùng khiếm thị nghe được trạng thái đã đổi sang AUTO.

===========================================================================================

=== TỔNG KẾT VAI TRÒ ===

===========================================================================================

HTML: tạo cấu trúc (select, option, p)

JS: xử lý logic khi onchange xảy ra

DOM: nơi lưu trạng thái và nội dung

CSS: hiển thị kết quả (mặc định)

ARIA: giúp thiết bị hỗ trợ hiểu UI

→ onchange = chỉ chạy khi GIÁ TRỊ ĐÃ ĐỔI XONG

→ Không realtime như oninput

-->

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

156. Thuộc tính onfocus

→ KÍCH HOẠT khi PHẦN TỬ được focus (chuột click / tab phím / JS đưa vào)

------------------------------------------------

156.1 [1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- • Phát hiện thời điểm người dùng BẮT ĐẦU tương tác

- • Chuẩn bị trạng thái trước khi nhập / điều khiển

=> Hiểu nhanh: CHẠM VÀO Ô → onfocus chạy

------------------------------------------------

156.2 [2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- • Bật chế độ điều khiển xe / robot / ESP

- • Hiển thị trạng thái: "SẴN SÀNG NHẬP"

Ví dụ ngắn:

- Click ô lệnh → bật MODE ĐIỀU KHIỂN

- Focus ô → cho phép gửi lệnh

------------------------------------------------

156.3 [3] PHẠM VI ÁP DỤNG

- • Loại: Event-only

- • Hay dùng cho:

  + input

  + textarea

  + select

- • CẦN focus:

  + Click chuột

  + Tab phím

  + JS focus()

------------------------------------------------

156.4 [4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag onfocus="tenHam()">

- Ví dụ ĐÚNG:

  <input onfocus="batCheDo()">

- Ví dụ SAI:

  <div onfocus="batCheDo()">

  -> div mặc định KHÔNG focus được

------------------------------------------------

156.5 [5] GIÁ TRỊ THƯỜNG DÙNG

- • Giá trị: TÊN HÀM JavaScript

- • Có thể truyền:

  + this

  + this.value

Ví dụ hay gặp:

- onfocus="bat()"

- onfocus="xuLy(this)"

------------------------------------------------

156.6 [6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng click / tab vào phần tử

2. Phần tử nhận focus

3. Event onfocus được kích hoạt

4. JavaScript chạy

5. DOM / trạng thái giao diện đổi

=> Dùng để DEBUG luồng tương tác

------------------------------------------------

156.7 [7] BẢN CHẤT HOẠT ĐỘNG

- • Là: Event Attribute

- • Ai kích hoạt:

  + ✔ Người dùng (chuột / phím)

  + ✖ Trình duyệt tự chạy

  + ✖ JS tự phát

=> Người dùng là NGUỒN kích hoạt

------------------------------------------------

156.8 [8] LƯU Ý / LỖI HAY GẶP

- • Nhầm với oninput

  -> onfocus chỉ chạy 1 lần khi vào ô

- • Gắn cho thẻ không focus được

  -> event KHÔNG chạy

Mẹo nhớ:

- focus = CHẠM VÀO

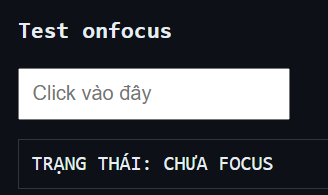
- input = ĐANG GÕ

------------------------------------------------

156.9 [9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ onfocus dùng để xử lý khi người dùng BẮT ĐẦU tương tác với phần tử

156.10.EXAMBLE





<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test onfocus</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    input {

      padding: 10px;

      font-size: 16px;

    }

    #trangthai {

      margin-top: 15px;

      border: 1px solid #30363d;

      padding: 10px;

    }

  </style>

</head>

<body>

  <h3>Test onfocus</h3>

  <input type="text" placeholder="Click vào đây" onfocus="batCheDo()">

  <div id="trangthai">TRẠNG THÁI: CHƯA FOCUS</div>

  <script>

    function batCheDo() {

      document.getElementById("trangthai").innerText =

        "TRẠNG THÁI: ĐÃ FOCUS (SẴN SÀNG NHẬP)";

    }

  </script>

</body>

</html>

<!--

###

Trang vừa load (HTML được parse)

###

- Trình duyệt đọc <!DOCTYPE html> → xác định chế độ chuẩn.

- Tạo DOM tree:

  + <body>

    - <h3> "Test onfocus"

    - <input type="text" placeholder="Click vào đây" onfocus="batCheDo()">

    - <div id="trangthai"> "TRẠNG THÁI: CHƯA FOCUS"

- LƯU Ý QUAN TRỌNG:

  + Thuộc tính onfocus được gắn SẴN vào node <input>

  + id="trangthai" tạo ra MỘT điểm định danh DUY NHẤT trong DOM.

###

CSS bắt đầu đọc DOM và áp style

###

- CSS quét DOM:

  + body → nền tối, chữ sáng, font monospace

  + input → padding, font-size

  + #trangthai → border + padding

- DOM lúc này CHƯA thay đổi nội dung:

  → div#trangthai vẫn là: "TRẠNG THÁI: CHƯA FOCUS"

###

Trạng thái DOM ban đầu (trước tương tác)

###

- input: CHƯA được focus

- div#trangthai:

  innerText = "TRẠNG THÁI: CHƯA FOCUS"

- JS:

  + function batCheDo đã được nạp vào bộ nhớ

  + CHƯA chạy dòng nào

###

Người dùng thao tác

###

- Người dùng CLICK chuột vào ô input

- Trình duyệt chuyển focus vào input này

###

Sự kiện phát sinh

###

- Event: focus

- Vì input có:

  onfocus="batCheDo()"

→ Trình duyệt GỌI hàm batCheDo()

###

JS bắt đầu chạy

###

batCheDo()

###

- Dòng chạy đầu tiên:

  document.getElementById("trangthai")

- Giải thích dấu . :

  + document: đối tượng gốc của DOM

  + .getElementById: truy cập METHOD của document

  → Tìm node có id="trangthai"

###

JS truy cập DOM

###

- DOM TRƯỚC KHI ĐỔI:

  div#trangthai.innerText =

  "TRẠNG THÁI: CHƯA FOCUS"

- Tiếp tục dấu . :

  + .innerText → thuộc tính TEXT của node div

###

JS ghi dữ liệu mới vào DOM

###

innerText = "TRẠNG THÁI: ĐÃ FOCUS (SẴN SÀNG NHẬP)"

- JS là THẰNG ĐỔI

- DOM bị cập nhật NGAY LẬP TỨC

###

DOM sau khi đổi

###

- div#trangthai.innerText =

  "TRẠNG THÁI: ĐÃ FOCUS (SẴN SÀNG NHẬP)"

- input đang ở trạng thái FOCUS

###

CSS / UI phản ứng

###

- CSS KHÔNG đổi style vì không có rule :focus

- Nhưng nội dung hiển thị thay đổi → người dùng THẤY NGAY

###

ARIA / Screen Reader (nếu có)

###

- Khi input được focus:

  + Screen reader đọc: "Edit text, focused"

- Khi text thay đổi:

  + Screen reader đọc lại nội dung mới của div

###

KẾT LUẬN LUỒNG CHẠY

###

- HTML: khai báo cấu trúc + gắn onfocus

- CSS: tạo giao diện, KHÔNG điều khiển logic

- JS: xử lý hành động khi focus

- DOM: bị thay đổi bởi JS

- onfocus: KÍCH HOẠT khi phần tử NHẬN FOCUS

- Đây là nền tảng cho:

  + Form thông minh

  + Bật chế độ nhập

  + Điều khiển UI / ESP / MODE nhập liệu

-->

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[157]. Thuộc tính onblur

→ Kích hoạt khi PHẦN TỬ MẤT FOCUS (người dùng rời khỏi nó)

------------------------------------------------

[157.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: PHÁT HIỆN thời điểm người dùng KẾT THÚC tương tác

- Ý 2: Dùng khi click ra ngoài / tab sang chỗ khác

------------------------------------------------

[157.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Kết thúc nhập liệu

  + Gửi dữ liệu sau khi nhập xong

  + Đổi trạng thái hệ thống

👉 Ví dụ ngắn:

- Rời ô nhập → gửi lệnh STOP

- Rời input → hiển thị “đã nhập xong”

------------------------------------------------

[157.3] PHẠM VI ÁP DỤNG

- Event-only

- Áp cho:

  + input, textarea, select, button

- Có cần focus không?

  + ✅ CÓ (phải focus trước thì mới blur được)

------------------------------------------------

[157.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag onblur="tenHam()">

- Ví dụ đúng:

  <input onblur="roiKhoi()">

- Ví dụ SAI:

  <div onblur="roiKhoi()"> ❌

  → div không focus được nếu không xử lý thêm

------------------------------------------------

[157.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Tên hàm JavaScript

- Ví dụ:

  + onblur="stopNhap()"

  + onblur="guiDuLieu()"

------------------------------------------------

[157.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng đang focus vào input

2. Người dùng click ra ngoài / tab sang chỗ khác

3. Trình duyệt phát sinh sự kiện blur

4. onblur được kích hoạt

5. JavaScript chạy và đổi DOM / trạng thái

------------------------------------------------

[157.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Event

- Ai kích hoạt:

  + Người dùng (rời focus)

  + Trình duyệt (phát hiện mất focus)

  + JavaScript (xử lý hậu quả)

👉 “AI làm mất focus?”

→ Người dùng

👉 “AI xử lý?”

→ JavaScript

------------------------------------------------

[157.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Dùng onblur cho thẻ không focus được

- Lỗi 2: Nghĩ rằng blur = click (❌ sai)

- Cách nhớ nhanh:

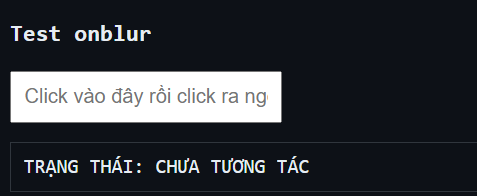
  → “BLUR = RỜI KHỎI”

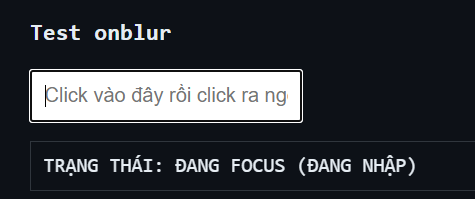
------------------------------------------------

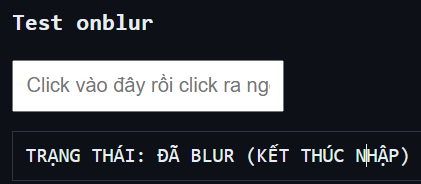
[157.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ onblur dùng để XỬ LÝ khi người dùng RỜI KHỎI phần tử 📴👆

157.10.EXAMBLE







<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test onblur</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    input {

      padding: 10px;

      font-size: 16px;

    }

    #trangthai {

      margin-top: 15px;

      border: 1px solid #30363d;

      padding: 10px;

    }

  </style>

</head>

<body>

  <h3>Test onblur</h3>

  <input type="text" placeholder="Click vào đây rồi click ra ngoài" onfocus="vaoNhap()" onblur="roiKhoi()">

  <div id="trangthai">TRẠNG THÁI: CHƯA TƯƠNG TÁC</div>

  <script>

    function vaoNhap() {

      document.getElementById("trangthai").innerText =

        "TRẠNG THÁI: ĐANG FOCUS (ĐANG NHẬP)";

    }

    function roiKhoi() {

      document.getElementById("trangthai").innerText =

        "TRẠNG THÁI: ĐÃ BLUR (KẾT THÚC NHẬP)";

    }

  </script>

</body>

</html>

<!--

###

<!DOCTYPE html> + <html> được trình duyệt đọc

###

- Trình duyệt nhận file HTML → xác định đây là tài liệu HTML5.

- Bắt đầu tạo cây DOM từ trên xuống dưới.

- Chưa có JS chạy, chưa có sự kiện nào xảy ra.

###

<head> + <style>

###

- Trình duyệt tạo node <head>, đọc <meta>, <title>.

- CSS trong <style> được parse thành CSSOM.

- CSSOM chưa áp gì cụ thể vì DOM <body> chưa tạo xong.

- Lưu rule:

  - body có background tối

  - input có padding, font-size

  - #trangthai có border, padding

###

<body> bắt đầu được parse

###

- DOM node <body> được tạo.

- Trình duyệt bắt đầu sinh các node con bên trong body.

###

<h3>Test onblur</h3>

###

- DOM thêm 1 node h3.

- CSS đọc node này → áp style font mặc định từ body.

- Chưa có tương tác, chỉ là hiển thị chữ.

###

<input ... onfocus="vaoNhap()" onblur="roiKhoi()">

###

- DOM tạo node input type="text".

- Thuộc tính onfocus:

  → đăng ký: KHI input được focus → gọi hàm vaoNhap().

- Thuộc tính onblur:

  → đăng ký: KHI input mất focus → gọi hàm roiKhoi().

- LƯU Ý:

  → onfocus / onblur CHƯA chạy, chỉ được gắn sẵn vào DOM.

- CSS áp padding, font-size cho input.

- Trạng thái ban đầu:

  → input CHƯA focus

  → chưa có event nào kích hoạt.

###

<div id="trangthai">TRẠNG THÁI: CHƯA TƯƠNG TÁC</div>

###

- DOM tạo node div với id="trangthai".

- Nội dung text ban đầu: "CHƯA TƯƠNG TÁC".

- CSS:

  → selector #trangthai match

  → áp border + padding.

- Đây là trạng thái DOM BAN ĐẦU.

###

<script> được đọc

###

- Trình duyệt parse JS.

- Tạo 2 function trong bộ nhớ:

  → vaoNhap()

  → roiKhoi()

- KHÔNG có dòng nào chạy ngay vì:

  → không có lệnh gọi hàm trực tiếp.

- JS chỉ “đứng chờ” event.

=======================================================================================

=== NGƯỜI DÙNG TƯƠNG TÁC ===

=======================================================================================

###

Người dùng CLICK vào input

###

- Trình duyệt phát hiện input nhận focus.

- Event focus xảy ra.

###

onfocus="vaoNhap()" được kích hoạt

###

- Trình duyệt gọi hàm vaoNhap().

- Chuyển quyền điều khiển sang JavaScript.

###

function vaoNhap() bắt đầu chạy

###

- Dòng chạy đầu tiên:

  document.getElementById("trangthai").innerText = ...

- Giải thích dấu . :

  → document.getElementById

    document: đối tượng DOM gốc

    .getElementById: truy cập method tìm node theo id

  → .innerText

    truy cập thuộc tính text của node div

###

DOM TRƯỚC khi đổi (vaoNhap)

###

- #trangthai:

  "TRẠNG THÁI: CHƯA TƯƠNG TÁC"

###

JS ghi dữ liệu mới vào DOM

###

- innerText bị ghi đè thành:

  "TRẠNG THÁI: ĐANG FOCUS (ĐANG NHẬP)"

###

DOM SAU khi đổi (vaoNhap)

###

- Nội dung div đã đổi.

- JS kết thúc, trả quyền lại cho trình duyệt.

###

CSS phản ứng

###

- CSS KHÔNG đổi vì:

  → chỉ đổi text, không đổi class / style.

- Màn hình hiển thị text mới.

=======================================================================================

=== NGƯỜI DÙNG CLICK RA NGOÀI ===

=======================================================================================

###

Input mất focus

###

- Trình duyệt phát hiện input bị blur.

- Event blur xảy ra.

###

onblur="roiKhoi()" được kích hoạt

###

- Trình duyệt gọi hàm roiKhoi().

###

function roiKhoi() bắt đầu chạy

###

- Dòng chạy:

  document.getElementById("trangthai").innerText = ...

###

DOM TRƯỚC khi đổi (roiKhoi)

###

- #trangthai:

  "TRẠNG THÁI: ĐANG FOCUS (ĐANG NHẬP)"

###

JS ghi dữ liệu mới

###

- innerText đổi thành:

  "TRẠNG THÁI: ĐÃ BLUR (KẾT THÚC NHẬP)"

###

DOM SAU khi đổi (roiKhoi)

###

- Text thay đổi lần thứ 2.

- JS kết thúc.

###

CSS / Màn hình phản ứng

###

- CSS vẫn giữ nguyên layout.

- Người dùng thấy trạng thái chuyển sang BLUR.

- Screen reader (nếu có):

  → đọc nội dung mới của div.

====================================================================================

=== TỔNG KẾT CUỐI ===

===================================================================================

- HTML:

  → tạo cấu trúc + gắn event onfocus / onblur.

- CSS:

  → chỉ lo hiển thị, không điều khiển logic.

- JS:

  → chỉ chạy KHI event xảy ra.

- DOM:

  → là thứ bị JS sửa trực tiếp (innerText).

- onfocus:

  → kích hoạt khi element NHẬN focus.

- onblur:

  → kích hoạt khi element MẤT focus.

👉 ĐÚNG kiểu:

“JS không tự chạy — TRÌNH DUYỆT gọi JS khi trạng thái DOM thay đổi.”

-->

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[158]. Thuộc tính onsubmit

→ Kích hoạt khi FORM được GỬI ĐI (submit dữ liệu)

------------------------------------------------

[158.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: BẮT SỰ KIỆN gửi form (nhấn nút submit / Enter)

- Ý 2: Kiểm tra, chặn, hoặc xử lý dữ liệu TRƯỚC khi gửi

------------------------------------------------

[158.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Gửi lệnh điều khiển (ESP / robot)

  + Gửi cấu hình (mode, tốc độ, trạng thái)

  + Kiểm tra dữ liệu trước khi gửi

👉 Ví dụ ngắn:

- Nhấn SUBMIT → gửi lệnh START

- Sai dữ liệu → KHÔNG gửi, báo lỗi

------------------------------------------------

[158.3] PHẠM VI ÁP DỤNG

- Event-only

- Áp cho:

  + <form> (CHỈ form mới có onsubmit)

- Có cần focus không?

  + ❌ Không cần (Enter trong input cũng submit)

------------------------------------------------

[158.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <form onsubmit="tenHam()">

- Ví dụ đúng:

  <form onsubmit="guiLenh()">

- Ví dụ SAI:

  <button onsubmit="guiLenh()"> ❌

  → button KHÔNG có onsubmit

------------------------------------------------

[158.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Tên hàm JavaScript

- Ví dụ:

  + onsubmit="guiDuLieu()"

  + onsubmit="return kiemTra()"

------------------------------------------------

[158.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng nhấn nút SUBMIT hoặc Enter

2. Trình duyệt phát sinh sự kiện submit

3. onsubmit được kích hoạt

4. JavaScript chạy:

   - Nếu return false → DỪNG gửi

   - Nếu return true / không return → TIẾP TỤC gửi

5. DOM / trang có thể reload hoặc đổi trạng thái

------------------------------------------------

[158.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Event

- Ai kích hoạt:

  + Người dùng (nhấn submit / Enter)

  + Trình duyệt (xử lý form)

  + JavaScript (quyết định có gửi hay không)

👉 onsubmit = “CỬA KIỂM SOÁT CUỐI” trước khi gửi

------------------------------------------------

[158.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Quên return false → form reload trang

- Lỗi 2: Gắn onsubmit vào thẻ không phải form

- Cách nhớ nhanh:

  → “SUBMIT = GỬI → phải kiểm soát”

------------------------------------------------

[158.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ onsubmit dùng để KIỂM SOÁT việc GỬI FORM khi người dùng SUBMIT 📤🧠

158.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test onsubmit</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    input,

    button {

      padding: 8px;

      font-size: 16px;

      margin-top: 8px;

      display: block;

    }

    #trangthai {

      margin-top: 15px;

      border: 1px solid #30363d;

      padding: 10px;

    }

  </style>

</head>

<body>

  <h3>Test onsubmit</h3>

  <form onsubmit="guiLenh(); return false;">

    <input type="text" placeholder="Nhập lệnh (VD: START)">

    <button type="submit">GỬI</button>

  </form>

  <div id="trangthai">TRẠNG THÁI: CHƯA GỬI</div>

  <script>

    function guiLenh() {

      document.getElementById("trangthai").innerText =

        "TRẠNG THÁI: ĐÃ SUBMIT FORM (ĐÃ GỬI LỆNH)";

    }

  </script>

</body>

</html>

<!--

###

<!DOCTYPE html> + <html> + <head> + <body>

###

→ Khi trang vừa load, trình duyệt bắt đầu đọc HTML từ trên xuống

→ <!DOCTYPE html> báo cho trình duyệt dùng chuẩn HTML5

→ <html lang="vi"> tạo node gốc của DOM

→ <head> và <body> được tạo thành 2 nhánh con trong DOM tree

==================================================

###

<style> ... </style>

###

→ Trình duyệt đọc CSS SAU khi DOM structure được tạo

→ CSS quét DOM:

   - body → nền tối, chữ sáng, font monospace

   - input, button → padding, font-size, display:block

   - #trangthai → viền, padding

→ LÚC NÀY: chưa có tương tác, chỉ là style tĩnh

==================================================

###

<form onsubmit="guiLenh(); return false;">

###

→ DOM tạo 1 node form

→ Thuộc tính onsubmit được gắn vào form

→ Trình duyệt ghi nhớ:

   “KHI form bị submit → chạy đoạn JS trong onsubmit”

→ return false; được gắn kèm:

   - nếu JS chạy xong mà return false

   - thì trình duyệt KHÔNG reload trang

==================================================

###

<input type="text">

###

→ DOM tạo input

→ Trạng thái ban đầu:

   - chưa focus

   - chưa có value

→ Chưa có JS nào chạy

==================================================

###

<button type="submit">

###

→ DOM tạo button

→ type="submit" báo cho trình duyệt:

   “KHI click → form sẽ submit”

==================================================

###

<div id="trangthai">TRẠNG THÁI: CHƯA GỬI</div>

###

→ DOM có 1 div với id="trangthai"

→ innerText ban đầu = "TRẠNG THÁI: CHƯA GỬI"

→ CSS đã áp viền + padding

==================================================

================== NGƯỜI DÙNG TƯƠNG TÁC ==========

==================================================

###

Người dùng click nút "GỬI" hoặc nhấn Enter

###

→ Trình duyệt phát hiện:

   - button type="submit" được kích hoạt

→ HÀNH ĐỘNG NÀY KHÔNG GỌI JS NGAY

→ Trình duyệt chuẩn bị submit FORM

==================================================

###

onsubmit="guiLenh(); return false;"

###

→ Trước khi submit thật sự:

   - trình duyệt gọi sự kiện submit

   - onsubmit được kích hoạt

→ JS bắt đầu chạy:

   guiLenh();

==================================================

================== JS BẮT ĐẦU CHẠY ==============

==================================================

###

function guiLenh() { ... }

###

→ JS engine tìm hàm guiLenh

→ Nhảy vào thân hàm

=================================================

###

document.getElementById("trangthai")

###

→ document: object gốc đại diện cho toàn bộ DOM

→ dấu . :

   - truy cập phương thức getElementById của document

→ "trangthai":

   - JS tìm node DOM có id="trangthai"

→ KẾT QUẢ:

   - trả về đúng div trạng thái

==================================================

###

.innerText = "TRẠNG THÁI: ĐÃ SUBMIT FORM (ĐÃ GỬI LỆNH)"

###

→ dấu . :

   - truy cập thuộc tính innerText của div

→ TRƯỚC KHI ĐỔI:

   innerText = "TRẠNG THÁI: CHƯA GỬI"

→ JS là thằng ghi DOM

→ SAU KHI ĐỔI:

   innerText = "TRẠNG THÁI: ĐÃ SUBMIT FORM (ĐÃ GỬI LỆNH)"

==================================================

================== DOM ĐÃ BỊ ĐỔI =================

==================================================

###

DOM thay đổi nội dung text

###

→ Trình duyệt phát hiện DOM text node thay đổi

→ CSS KHÔNG đổi (vì chỉ text đổi)

→ Màn hình update ngay lập tức

→ Người dùng thấy trạng thái mới

==================================================

###

return false;

###

→ JS trả về false cho onsubmit

→ Trình duyệt nhận kết quả false

→ HỦY hành động submit mặc định

→ Trang KHÔNG reload

→ DOM vẫn giữ nguyên trạng thái mới

==================================================

================== ARIA / ACCESSIBILITY ==========

==================================================

###

Screen reader

###

→ Khi innerText đổi:

   - screen reader đọc lại nội dung mới

→ Người dùng khiếm thị biết:

   “TRẠNG THÁI: ĐÃ SUBMIT FORM”

==================================================

================== TỔNG KẾT CUỐI =================

==================================================

HTML:

- Định nghĩa cấu trúc form, input, button, div

CSS:

- Trang trí giao diện

- KHÔNG tham gia logic

JS:

- Là thằng xử lý hành động submit

- Là thằng thay đổi DOM

DOM:

- Lưu trạng thái UI

- Bị JS ghi trực tiếp

onsubmit:

- Là điểm chặn submit

- Cho phép xử lý logic TRƯỚC khi gửi form

- return false quyết định reload hay không

-->

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[ 159 ]. Thuộc tính onload

→ Kích hoạt hành động NGAY SAU KHI trang / tài nguyên tải xong

------------------------------------------------

[159.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Chạy JS ngay khi trang đã load xong

- Dùng để khởi tạo trạng thái ban đầu (init)

------------------------------------------------

[159.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Gửi tín hiệu “HỆ THỐNG SẴN SÀNG” cho ESP

  + Load dữ liệu ban đầu

  + Set trạng thái mặc định UI

▶ Ví dụ thực tế:

- Trang mở → gửi lệnh READY

- UI hiện “ĐÃ KẾT NỐI”

------------------------------------------------

[159.3] PHẠM VI ÁP DỤNG

- Event-only

- Áp cho:

  + <body> (phổ biến nhất)

  + <img>, <iframe>, <script>

- KHÔNG cần focus

------------------------------------------------

[159.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <body onload="tenHam()">

- Ví dụ đúng:

  <body onload="khoiDong()">

- Ví dụ SAI:

  <div onload="abc()">  ❌ (div không hỗ trợ onload)

------------------------------------------------

[159.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Tên hàm JavaScript

- Ví dụ:

  + onload="init()"

  + onload="batDauHeThong()"

-----------------------------------------------

[159.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt tải HTML

2. Tạo xong DOM

3. Load xong CSS / ảnh / script

4. onload ĐƯỢC KÍCH HOẠT

5. JS chạy → ghi DOM

6. Màn hình hiển thị trạng thái sẵn sàng

------------------------------------------------

[159.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Event

- Ai kích hoạt:

  + Trình duyệt (KHÔNG phải người dùng)

- Chỉ chạy 1 lần duy nhất khi load xong

------------------------------------------------

[159.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Đặt onload sai thẻ (div, span)

- Lỗi 2: Hàm chưa khai báo → JS lỗi

- Mẹo nhớ:

  → onload = “BẬT MÁY LẦN ĐẦU”

------------------------------------------------

[159.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “onload dùng để chạy JS ngay khi trang tải xong hoàn toàn”

159.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>TEST onload</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    button {

      padding: 10px;

      font-size: 16px;

      margin-top: 10px;

    }

  </style>

</head>

<body onload="tuDongChay()">

  <h3>TEST onload</h3>

  <button onclick="bamNut()">BẤM THỬ</button>

  <p id="ketqua">CHƯA CÓ GÌ XẢY RA</p>

  <script>

    function tuDongChay() {

      document.getElementById("ketqua").innerText =

        "🔥 onload ĐÃ CHẠY (KHÔNG CẦN BẤM)";

    }

    function bamNut() {

      document.getElementById("ketqua").innerText =

        "👉 onclick CHẠY (DO BẠN BẤM)";

    }

  </script>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[160]. Thuộc tính onerror

→ DÙNG để KÍCH HOẠT JavaScript khi XẢY RA LỖI (ảnh hỏng / file lỗi / JS lỗi)

------------------------------------------------

[160.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: BẮT LỖI khi tài nguyên không tải được (ảnh, script, audio…)

- Ý 2: KÍCH HOẠT xử lý dự phòng khi có lỗi

------------------------------------------------

[160.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Báo lỗi hệ thống

  + Hiển thị trạng thái lỗi

  + Gửi cảnh báo (ESP / server)

👉 Ví dụ ngắn:

- Ảnh camera ESP lỗi → hiện chữ ❌ CAMERA MẤT KẾT NỐI

- Script lỗi → chuyển sang chế độ an toàn

------------------------------------------------

[160.3] PHẠM VI ÁP DỤNG

- Event-only

- Áp cho:

  + img, script, audio, video, iframe, body

- Không cần focus

- Tự kích hoạt khi lỗi xảy ra

------------------------------------------------

[160.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag onerror="tenHam()">

- Ví dụ đúng:

  <img src="cam.jpg" onerror="baoLoi()">

- Ví dụ SAI:

  <img onerror=baoLoi>   ❌ thiếu ()

------------------------------------------------

[160.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Tên hàm JavaScript

- Ví dụ hay gặp:

  + onerror="xuLyLoi()"

  + onerror="this.src='backup.jpg'"

------------------------------------------------

[160.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt tải trang

2. Gặp thẻ có tài nguyên (ảnh / script)

3. Tài nguyên tải THẤT BẠI

4. onerror ĐƯỢC KÍCH HOẠT

5. JavaScript chạy xử lý lỗi

6. DOM đổi (hiện thông báo / ảnh dự phòng)

7. Người dùng thấy trạng thái lỗi

------------------------------------------------

[160.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Event attribute

- Ai kích hoạt:

  + Trình duyệt (khi phát hiện lỗi)

- JavaScript là thằng xử lý hậu quả

------------------------------------------------

[160.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Quên dấu () → hàm không chạy

- Lỗi 2: Dùng cho thẻ không có tài nguyên → vô tác dụng

- MẸO NHỚ:

  → “Có tải file → có thể dùng onerror”

------------------------------------------------

[160.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ 🔥 onerror dùng để CHẠY CODE khi TÀI NGUYÊN BỊ LỖI

160.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test onerror</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    img {

      width: 200px;

      border: 1px solid #30363d;

      margin-top: 10px;

    }

    #status {

      margin-top: 15px;

      padding: 10px;

      border: 1px solid #30363d;

    }

  </style>

</head>

<body>

  <h3>TEST onerror</h3>

  <!-- CỐ TÌNH trỏ ảnh SAI để gây lỗi -->

  <img src="anh-khong-ton-tai.jpg" onerror="baoLoi()">

  <div id="status">TRẠNG THÁI: CHƯA CÓ LỖI</div>

  <script>

    function baoLoi() {

      document.getElementById("status").innerText =

        "❌ LỖI: KHÔNG TẢI ĐƯỢC ẢNH → onerror ĐÃ CHẠY";

    }

  </script>

</body>

</html>

###

<!DOCTYPE html> + <html> + <head> + <body>

###

- Trình duyệt bắt đầu LOAD trang

- Đọc HTML từ trên xuống dưới

- Tạo DOM tree trong bộ nhớ:

  + node html

  + node head (meta, title, style)

  + node body (h3, img, div, script)

- LÚC NÀY: chưa có JS nào chạy, chưa có lỗi nào xảy ra

###

<style>...</style>

###

- CSS được parser đọc NGAY KHI DOM đang được tạo

- CSS engine ghi nhớ:

  + body có nền tối, chữ sáng

  + img có width 200px, border

  + #status có viền, padding

- CSS CHƯA render ngay, chỉ lưu rule chờ DOM hoàn chỉnh

###

<h3>TEST onerror</h3>

###

- DOM node h3 được tạo

- Nội dung text "TEST onerror" được gắn vào node

- CSS kiểm tra:

  + có rule nào áp cho h3 không? → KHÔNG

- h3 sẽ dùng style mặc định + style body

###

<img src="anh-khong-ton-tai.jpg" onerror="baoLoi()">

###

- DOM node img được tạo

- Trình duyệt thấy:

  + src = "anh-khong-ton-tai.jpg"

  + onerror = "baoLoi()"

- Ngay khi DOM gắn xong img:

  → Trình duyệt BẮT ĐẦU tải tài nguyên ảnh từ src

- Đây là thời điểm quan trọng:

  + trình duyệt gửi request lấy file ảnh

###

<div id="status">TRẠNG THÁI: CHƯA CÓ LỖI</div>

###

- DOM node div được tạo

- id="status" được ghi vào DOM

- innerText ban đầu = "TRẠNG THÁI: CHƯA CÓ LỖI"

- CSS rule #status được áp:

  + có border

  + có padding

- TRẠNG THÁI DOM HIỆN TẠI:

  + status đang hiển thị: CHƯA CÓ LỖI

###

<script>function baoLoi() {...}</script>

###

- Trình duyệt gặp script

- KHÔNG chạy hàm baoLoi ngay

- Chỉ:

  + tạo function object tên baoLoi

  + lưu vào bộ nhớ JS

- DOM + CSS lúc này đã render xong lần đầu

###

(QUÁ TRÌNH LOAD ẢNH)

###

- Trình duyệt chờ phản hồi file "anh-khong-ton-tai.jpg"

- File KHÔNG tồn tại → load FAILED

- Trình duyệt xác định:

  → img bị lỗi tài nguyên

###

onerror="baoLoi()"

###

- SỰ KIỆN error phát sinh trên node img

- Thuộc tính onerror được kích hoạt

- Trình duyệt gọi JS:

  → baoLoi()

###

function baoLoi() {

###

- JS engine bắt đầu chạy hàm baoLoi

- Dòng đầu tiên được thực thi bên trong hàm

###

document.getElementById("status")

###

- document: đối tượng đại diện toàn bộ DOM

- dấu . :

  + truy cập phương thức getElementById của document

- getElementById("status"):

  + tìm node có id="status"

  + TRẢ VỀ DOM element <div id="status">

###

.innerText =

###

- dấu . :

  + truy cập thuộc tính innerText của div#status

- innerText là nội dung chữ bên trong node

###

"❌ LỖI: KHÔNG TẢI ĐƯỢC ẢNH → onerror ĐÃ CHẠY";

###

- JS ghi nội dung mới vào DOM

- TRƯỚC KHI ĐỔI:

  + div#status = "TRẠNG THÁI: CHƯA CÓ LỖI"

- JS là thằng gây thay đổi

- SAU KHI ĐỔI:

  + div#status = "❌ LỖI: KHÔNG TẢI ĐƯỢC ẢNH → onerror ĐÃ CHẠY"

###

(DOM ĐÃ THAY ĐỔI)

###

- Trình duyệt phát hiện DOM mutation

- CSS KHÔNG đổi layout (chỉ đổi text)

- Repaint diễn ra:

  + nội dung chữ mới được vẽ lên màn hình

- Người dùng thấy trạng thái lỗi ngay lập tức

###

(ARIA / Screen Reader)

###

- innerText thay đổi

- Screen reader nếu đang đọc:

  + nhận diện nội dung mới của div

  + thông báo lại trạng thái lỗi cho người dùng

###

(TỔNG KẾT VAI TRÒ)

###

- HTML:

  + định nghĩa cấu trúc (img, div, script)

- CSS:

  + quyết định giao diện hiển thị

- JS:

  + xử lý sự kiện error

  + thay đổi nội dung DOM

- onerror:

  + là EVENT attribute

  + do TRÌNH DUYỆT kích hoạt khi tài nguyên lỗi

- DOM:

  + là nơi JS ghi thay đổi

  + là thứ CSS & screen reader phản ứng theo

-->

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[161]. Thuộc tính onresize

→ Kích hoạt khi KÍCH THƯỚC cửa sổ trình duyệt (window) BỊ THAY ĐỔI

-----------------------------------------------

[161.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Phát hiện người dùng thay đổi kích thước màn hình

- Ý 2: Tự động cập nhật giao diện / trạng thái theo kích thước mới

------------------------------------------------

[161.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Điều khiển giao diện dashboard

  + Chuyển chế độ MOBILE / DESKTOP

  + Gửi trạng thái màn hình cho ESP / server

👉 Ví dụ ngắn:

- Thu nhỏ màn hình → gửi lệnh: MODE\_MOBILE

- Phóng to màn hình → gửi lệnh: MODE\_DESKTOP

------------------------------------------------

[161.3] PHẠM VI ÁP DỤNG

- Event-only

- Áp cho:

  + window (CHỦ YẾU)

  + body (ít dùng)

- Không cần focus

- Người dùng KHÔNG cần click

------------------------------------------------

[161.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <body onresize="tenHam()">

- Ví dụ đúng:

  <body onresize="xuLyResize()">

- Ví dụ SAI:

  <div onresize="xuLyResize()">   (div KHÔNG bắt resize)

------------------------------------------------

[161.5] GIÁ TRỊ THƯỜNG DÙNG

- Giá trị là:

  + TÊN HÀM JavaScript

- Dữ liệu thường đọc trong hàm:

  + window.innerWidth

  + window.innerHeight

------------------------------------------------

[161.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng kéo giãn / thu nhỏ cửa sổ

2. Trình duyệt phát hiện kích thước viewport thay đổi

3. Sự kiện resize phát sinh

4. onresize được kích hoạt

5. JavaScript chạy hàm xử lý

6. DOM / trạng thái giao diện được cập nhật

7. Màn hình hiển thị theo kích thước mới

------------------------------------------------

[161.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Event attribute

- Ai kích hoạt:

  + Người dùng (kéo cửa sổ)

  + Trình duyệt (phát hiện resize)

- JavaScript chỉ là thằng XỬ LÝ HẬU QUẢ

------------------------------------------------

[161.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Resize chạy LIÊN TỤC → gây lag

- Lỗi 2: Gắn onresize cho thẻ sai (div, span)

- Cách nhớ nhanh:

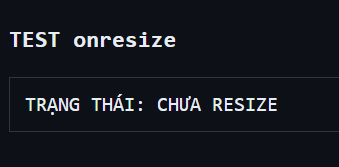
  → Resize = WINDOW, không phải element

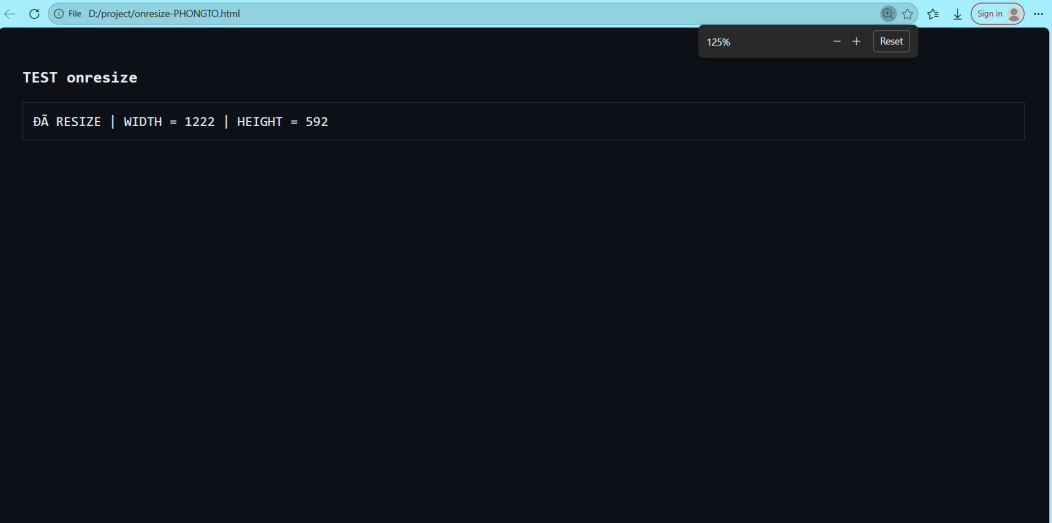
------------------------------------------------

[161.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “onresize dùng để CHẠY HÀM khi KÍCH THƯỚC CỬA SỔ bị thay đổi”

161.10.EXAMBLE





<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test onresize</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    #box {

      margin-top: 15px;

      padding: 12px;

      border: 1px solid #30363d;

    }

  </style>

</head>

<!-- onresize GẮN Ở BODY -->

<body onresize="xuLyResize()">

  <h3>TEST onresize</h3>

  <div id="box">

    TRẠNG THÁI: CHƯA RESIZE

  </div>

  <script>

    function xuLyResize() {

      document.getElementById("box").innerText =

        "ĐÃ RESIZE | WIDTH = " + window.innerWidth +

        " | HEIGHT = " + window.innerHeight;

    }

  </script>

</body>

</html>

<!--

###

<!DOCTYPE html> + toàn bộ cấu trúc HTML được trình duyệt load

###

- Trình duyệt bắt đầu đọc file HTML từ trên xuống.

- <!DOCTYPE html> báo cho trình duyệt dùng chế độ HTML5.

- <html lang="vi"> tạo node gốc của DOM.

- DOM tree được dựng lần lượt: head → body → các thẻ con.

- LÚC NÀY: chưa có JS chạy, chưa có sự kiện nào xảy ra.

###

<style> ... </style>

###

- Khi DOM node <style> được tạo, CSSOM được tạo song song.

- CSS đọc selector:

  - body → áp background, color, font cho toàn bộ trang.

  - #box → chuẩn bị style cho div có id="box".

- DOM ban đầu:

  - body tồn tại

  - div#box có innerText = "TRẠNG THÁI: CHƯA RESIZE"

- Màn hình hiển thị trạng thái CHƯA RESIZE.

###

<body onresize="xuLyResize()">

###

- Trình duyệt thấy attribute onresize.

- Ghi nhớ: "KHI viewport thay đổi kích thước → gọi hàm xuLyResize()".

- CHƯA gọi hàm, chỉ đăng ký sự kiện.

- Đây là Event attribute, do TRÌNH DUYỆT kích hoạt.

###

Trang vừa load xong – trạng thái nghỉ

###

- DOM: ổn định, không thay đổi.

- CSS: đã áp xong.

- JS: hàm xuLyResize tồn tại trong bộ nhớ.

- Không có event nào chạy.

===

=== NGƯỜI DÙNG BẮT ĐẦU TƯƠNG TÁC ===

===

###

Người dùng kéo to / thu nhỏ cửa sổ trình duyệt

###

- Viewport thay đổi width / height.

- Trình duyệt PHÁT HIỆN resize.

- Event resize được sinh ra.

###

Event resize kích hoạt onresize

###

- Vì onresize gắn trên <body>, nên trình duyệt gọi:

  xuLyResize()

- JS engine bắt đầu chạy hàm.

===

=== JS BẮT ĐẦU CHẠY ===

===

###

function xuLyResize() { ... }

###

- JS vào hàm xuLyResize.

- Không có if / === / ? : trong ví dụ này.

- Chạy thẳng từng dòng từ trên xuống.

###

document.getElementById("box")

###

- Dấu . :

  - document: đối tượng DOM gốc.

  - getElementById: truy cập method của document.

- Trình duyệt tìm node DOM có id="box".

- KẾT QUẢ: trả về DOM element <div id="box">.

###

.innerText = "ĐÃ RESIZE | WIDTH = " + window.innerWidth + " | HEIGHT = " + window.innerHeight

###

- Dấu . :

  - innerText là thuộc tính của element #box.

- JS đọc:

  - window.innerWidth → chiều ngang viewport HIỆN TẠI.

  - window.innerHeight → chiều cao viewport HIỆN TẠI.

- JS tạo ra CHUỖI mới từ các giá trị này.

- JS GHI vào DOM:

  - TRƯỚC: div#box = "TRẠNG THÁI: CHƯA RESIZE"

  - SAU: div#box = "ĐÃ RESIZE | WIDTH = ... | HEIGHT = ..."

===

=== SAU KHI DOM ĐỔI ===

===

###

DOM thay đổi

###

- Nội dung text của div#box đã khác.

- Không tạo node mới, chỉ đổi text node bên trong.

###

CSS phản ứng

###

- CSS không đổi selector → chỉ repaint text.

- Layout giữ nguyên, chỉ cập nhật chữ.

###

ARIA / Screen reader

###

- innerText thay đổi → screen reader đọc lại nội dung mới.

- Người dùng khiếm thị nghe được trạng thái mới.

===

=== LẶP LẠI ===

===

- MỖI LẦN resize tiếp theo:

  - Trình duyệt → event resize

  - onresize → xuLyResize()

  - DOM đổi → UI cập nhật

===

=== TỔNG KẾT VAI TRÒ ===

===

- HTML: khai báo cấu trúc + gắn onresize.

- CSS: hiển thị giao diện.

- JS: xử lý logic khi resize.

- DOM: nơi JS ghi thay đổi.

- Trình duyệt: là thằng PHÁT HIỆN resize và kích hoạt event.

-->

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

=======================================

[ 163 ]. Thuộc tính oncontextmenu

→ Kích hoạt khi NGƯỜI DÙNG BẤM CHUỘT PHẢI vào một phần tử HTML

------------------------------------------------

[163.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: BẮT sự kiện CHUỘT PHẢI (right-click)

- Ý 2: Dùng để CHẶN menu mặc định hoặc xử lý LỆNH RIÊNG

------------------------------------------------

[163.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Điều khiển robot / ESP bằng chuột phải

  + Gửi lệnh phụ (STOP / RESET / MENU)

  + Chặn copy / chặn menu chuột phải

👉 Ví dụ ngắn:

- Chuột trái → LỆNH CHÍNH

- Chuột phải → LỆNH PHỤ

------------------------------------------------

[163.3] PHẠM VI ÁP DỤNG

- Event-only

- Áp cho:

  + div, img, button, body

- Không cần focus (chuột tự kích hoạt)

------------------------------------------------

[163.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag oncontextmenu="tenHam()">

- Ví dụ đúng:

  <div oncontextmenu="xuLy(); return false;">

- Ví dụ SAI:

  <div oncontextmenu=tenHam>

  → Sai vì thiếu "" và ()

-----------------------------------------------

[163.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Tên hàm JavaScript

  + return false (RẤT QUAN TRỌNG)

- Ví dụ:

  + oncontextmenu="ham()"

  + oncontextmenu="ham(); return false;"

------------------------------------------------

[163.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng BẤM CHUỘT PHẢI

2. Trình duyệt phát sinh sự kiện contextmenu

3. Thuộc tính oncontextmenu được kích hoạt

4. JavaScript chạy hàm được gán

5. return false → CHẶN menu chuột phải

6. DOM bị JS thay đổi

7. Màn hình hiển thị trạng thái mới

------------------------------------------------

[163.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Event Attribute

- Ai kích hoạt:

  + Người dùng (chuột phải)

- Ai xử lý:

  + Trình duyệt gọi JavaScript

------------------------------------------------

[163.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Quên return false → menu chuột phải vẫn hiện

- Lỗi 2: Gán sai thẻ → sự kiện không chạy

- Cách nhớ nhanh:

  → Chuột phải + muốn chặn = return false

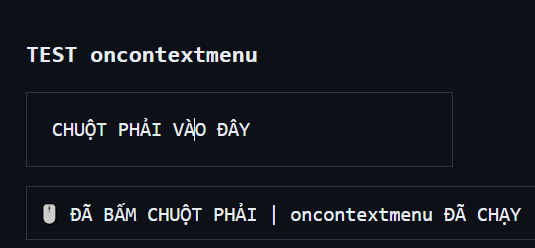
------------------------------------------------

[163.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “oncontextmenu dùng để BẮT CHUỘT PHẢI khi người dùng click phải vào phần tử”

163.10.EXAMBLE





<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test oncontextmenu</title>

  <style>

    body {

      background: #0d1117;

      color: #e6edf3;

      font-family: Consolas, monospace;

      padding: 20px;

    }

    #box {

      border: 1px solid #30363d;

      padding: 20px;

      width: 300px;

      cursor: pointer;

    }

    #status {

      margin-top: 15px;

      border: 1px solid #30363d;

      padding: 10px;

    }

  </style>

</head>

<body>

  <h3>TEST oncontextmenu</h3>

  <!-- CHUỘT PHẢI VÀO ĐÂY -->

  <div id="box" oncontextmenu="chuotPhai(); return false;">

    CHUỘT PHẢI VÀO ĐÂY

  </div>

  <div id="status">TRẠNG THÁI: CHƯA CÓ GÌ</div>

  <script>

    function chuotPhai() {

      document.getElementById("status").innerText =

        "🖱️ ĐÃ BẤM CHUỘT PHẢI | oncontextmenu ĐÃ CHẠY";

    }

  </script>

</body>

</html>

<!--

###

<!DOCTYPE html> + <html> + <head> + <body>

###

- Trình duyệt bắt đầu LOAD trang

- Đọc HTML từ trên xuống

- Tạo DOM tree:

  html

   ├─ head (meta, title, style)

   └─ body

       ├─ h3

       ├─ div#box

       ├─ div#status

       └─ script

- DOM ban đầu CHƯA có tương tác, chỉ là cấu trúc tĩnh

###

<style> ... </style>

###

- CSS đọc DOM vừa tạo

- Áp style:

  - body: nền tối, chữ sáng, font monospace

  - #box: có viền, padding, width 300px, cursor pointer

  - #status: khung trạng thái

- LÚC NÀY: giao diện hiển thị nhưng CHƯA có JS chạy

###

<div id="box" oncontextmenu="chuotPhai(); return false;">

###

- Trình duyệt gắn 1 EVENT LISTENER ngầm cho div#box

- Sự kiện: contextmenu (chuột phải)

- Khi event xảy ra:

  1) Gọi hàm chuotPhai()

  2) return false → chặn menu chuột phải mặc định

- DOM hiện tại:

  - #status.innerText = "TRẠNG THÁI: CHƯA CÓ GÌ"

###

<script> function chuotPhai() { ... } </script>

###

- Trình duyệt đọc JS

- Định nghĩa hàm chuotPhai

- CHƯA chạy hàm

- JS chỉ nằm trong bộ nhớ chờ EVENT

###

NGƯỜI DÙNG BẤM CHUỘT PHẢI VÀO div#box

###

- Hành động vật lý: click chuột phải

- Trình duyệt phát sinh event "contextmenu"

- Event target = div#box

- Vì div có oncontextmenu → JS được gọi

###

chuotPhai();

###

- JS bắt đầu chạy hàm

- Dòng chạy đầu tiên trong hàm

###

document.getElementById("status")

###

- document: đối tượng DOM gốc

- .getElementById:

  → truy cập PHƯƠNG THỨC của document

  → tìm node có id="status"

- Kết quả trả về:

  → element <div id="status">

###

.innerText =

###

- Dấu . :

  → truy cập THUỘC TÍNH innerText của element #status

- JS CHUẨN BỊ ghi dữ liệu mới vào DOM

###

"🖱️ ĐÃ BẤM CHUỘT PHẢI | oncontextmenu ĐÃ CHẠY"

###

- Chuỗi được gán vào innerText

- JS ghi đè nội dung cũ

##

DOM TRƯỚC KHI ĐỔI

###

- #status.innerText = "TRẠNG THÁI: CHƯA CÓ GÌ"

###

DOM SAU KHI ĐỔI

###

- #status.innerText = "🖱️ ĐÃ BẤM CHUỘT PHẢI | oncontextmenu ĐÃ CHẠY"

- DOM bị thay đổi bởi JS

###

CSS / MÀN HÌNH PHẢN ỨNG

###

- CSS KHÔNG đổi logic

- Trình duyệt repaint lại #status

- Người dùng THẤY nội dung mới ngay lập tức

###

return false;

###

- Chạy NGAY SAU chuotPhai()

- Báo cho trình duyệt:

  → KHÔNG hiển thị menu chuột phải mặc định

- Đây là lý do không thấy menu hệ thống

###

ARIA / SCREEN READER

###

- innerText thay đổi → screen reader đọc trạng thái mới

- Không có aria-\* nhưng vẫn đọc được vì nội dung text đổi

###

TỔNG KẾT VAI TRÒ

###

- HTML: tạo cấu trúc + gắn oncontextmenu

- CSS: hiển thị giao diện

- JS: xử lý logic khi chuột phải

- DOM: nơi bị JS thay đổi

- Trình duyệt: điều phối event + render lại UI

-->

Hết onclick

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

[RQT]

[164]. Thuộc tính href

→ Xác định ĐÍCH ĐẾN khi người dùng click vào liên kết.

-----------------------------------------------

[164.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Chỉ ra URL / vị trí sẽ được mở khi click

- Ý 2: Kích hoạt khi người dùng CLICK (chuột / Enter)

👉 Không click → không có tác dụng

------------------------------------------------

[164.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Gửi người dùng sang trang khác

  + Gửi lệnh điều hướng (control panel / dashboard)

  + Mở file, mở API, mở trang điều khiển ESP

Ví dụ ngắn:

- Click nút → mở trang điều khiển xe

- Click link → gửi lệnh ?mode=auto

------------------------------------------------

[164.3] PHẠM VI ÁP DỤNG

- Loại: Global (nhưng thực tế dùng cho link)

- Áp cho:

  + <a> (phổ biến nhất)

  + <area>

- Có cần focus không?

  + Có (phím Enter cần focus vào link)

------------------------------------------------

[164.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <a href="giá\_trị">

- Ví dụ đúng:

  <a href="/control/forward">TIẾN</a>

- Ví dụ SAI:

  <a href=>TIẾN</a>

  → thiếu giá trị → link hỏng

------------------------------------------------

[164.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Chuỗi (string)

- Giá trị hay gặp:

  + "https://example.com"

  + "/page"

  + "#id"

  + "?cmd=go"

  + "javascript:void(0)" ⚠️ hạn chế dùng

------------------------------------------------

[164.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng CLICK vào thẻ <a>

2. Trình duyệt đọc giá trị href

3. Trình duyệt gửi request tới URL

4. Trang hiện tại bị thay thế (hoặc tải nội dung mới)

5. Màn hình chuyển sang trang / trạng thái mới

Debug: nếu click mà KHÔNG đi đâu → kiểm tra href

------------------------------------------------

[164.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute (thuộc tính HTML)

- Ai kích hoạt:

  + Người dùng (click / Enter)

  + Trình duyệt xử lý trực tiếp

JS chỉ CAN THIỆP, không phải nguồn gốc

-----------------------------------------------

[164.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: href rỗng → reload trang

- Lỗi 2: sai URL → 404

- Nhớ nhanh:

  → href = “đi đâu”

  → onclick = “làm gì thêm”

------------------------------------------------

[164.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ href dùng để XÁC ĐỊNH NƠI ĐẾN khi người dùng CLICK vào liên kết

164.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Demo thuộc tính href</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      padding: 20px;

    }

    a {

      display: inline-block;

      margin: 6px 0;

      padding: 8px 12px;

      background: #1976d2;

      color: white;

      text-decoration: none;

      border-radius: 4px;

    }

    a:hover {

      background: #0d47a1;

    }

    .box {

      margin-top: 800px;

      padding: 20px;

      background: #f2f2f2;

      border: 1px solid #ccc;

    }

  </style>

</head>

<body>

  <h1>DEMO THUỘC TÍNH href</h1>

  <!-- 1. Link ra ngoài -->

  <a href="https://google.com">

    Mở Google

  </a>

  <br>

  <!-- 2. Link nội bộ -->

  <a href="/login">

    Đi tới /login

  </a>

  <br>

  <!-- 3. Gửi lệnh dạng query -->

  <a href="/control?cmd=forward">

    GỬI LỆNH: TIẾN

  </a>

  <br>

  <!-- 4. Điều hướng trong cùng trang -->

  <a href="#section2">

    Cuộn xuống PHẦN 2

  </a>

  <br>

  <!-- 5. href rỗng (lỗi hay gặp) -->

  <a href="">

    href rỗng (reload trang)

  </a>

  <br>

  <!-- 6. href + JS can thiệp -->

  <a href="/delete" onclick="event.preventDefault(); alert('JS đã chặn link');">

    href có nhưng KHÔNG đi

  </a>

  <!-- Phần đích để anchor nhảy tới -->

  <div id="section2" class="box">

    <h2>PHẦN 2</h2>

    <p>

      Khi click link có <code>href="#section2"</code>

      → trình duyệt cuộn xuống đây

      → KHÔNG reload trang

    </p>

  </div>

</body>

</html>

<!--

###

<!DOCTYPE html>

###

Trình duyệt nhận diện đây là tài liệu HTML5.

→ Chuyển sang chế độ standards mode.

→ Các quy tắc CSS, DOM, accessibility được áp dụng đầy đủ, không dùng quirks mode.

###

<html lang="vi">

###

Trình duyệt tạo node gốc của DOM: HTMLDocument.

→ Thuộc tính lang="vi" được gắn vào node <html>.

→ Screen reader ghi nhận: nội dung chính là tiếng Việt.

###

<head> … </head>

###

Trình duyệt đọc metadata trước body.

→ Chưa render nội dung hiển thị.

→ CSS và thông tin mã hóa được chuẩn bị trước khi vẽ giao diện.

###

<meta charset="UTF-8">

###

Trình duyệt biết cách giải mã ký tự.

→ Các chữ tiếng Việt hiển thị đúng, không lỗi font.

###

<style> … </style>

###

Trình duyệt parse toàn bộ CSS.

→ Tạo CSSOM (CSS Object Model).

→ Chưa áp vào DOM, chỉ lưu rule để dùng sau.

###

body { font-family: Arial; padding: 20px; }

###

CSS rule này chờ DOM body xuất hiện.

→ Khi body được tạo, CSSOM sẽ áp style:

   - font mặc định Arial

   - padding 20px xung quanh body

###

a { display, margin, padding, background, color… }

###

CSS định nghĩa hình dạng thẻ <a>.

→ Mọi node <a> trong DOM sẽ dùng rule này.

→ Chưa cần click, chỉ cần DOM tồn tại là áp.

###

a:hover { background: #0d47a1; }

###

CSS tạo pseudo-state :hover.

→ Chỉ kích hoạt khi con trỏ chuột đi vào <a>.

→ Không liên quan JS, trình duyệt tự xử lý.

###

.box { margin-top: 800px; … }

###

CSS rule dành cho class .box.

→ Chờ node có class="box" xuất hiện.

=========================================================================================

=== DOM BẮT ĐẦU ĐƯỢC TẠO ===

========================================================================================

###

<body>

###

Trình duyệt tạo node body.

→ CSS body được áp ngay:

   - padding 20px

   - font Arial

###

<h1>DEMO THUỘC TÍNH href</h1>

###

DOM node h1 được tạo.

→ CSS kế thừa font từ body.

→ Trình duyệt render chữ lên màn hình.

###

<a href="https://google.com">Mở Google</a>

###

DOM tạo node <a>.

→ Thuộc tính href được lưu trong DOM.

→ Trạng thái ban đầu:

   - href = "https://google.com"

   - Chưa có hành động nào xảy ra

→ CSS a được áp (nền xanh, chữ trắng).

→ Screen reader ghi nhận: “link, Mở Google”.

###

<a href="/login">Đi tới /login</a>

###

Tương tự link trên.

→ href trỏ tới đường dẫn nội bộ.

→ Trình duyệt chưa gửi request nào vì CHƯA click.

###

<a href="/control?cmd=forward">GỬI LỆNH: TIẾN</a>

###

DOM lưu href là chuỗi "/control?cmd=forward".

→ Ý nghĩa:

   - path = /control

   - query = cmd=forward

→ Trình duyệt chỉ ghi nhớ, chưa xử lý.

###

<a href="#section2">Cuộn xuống PHẦN 2</a>

###

href bắt đầu bằng #.

→ Trình duyệt hiểu đây là anchor nội bộ.

→ Chưa cuộn vì chưa click.

###

<a href="">href rỗng (reload trang)</a>

###

href tồn tại nhưng giá trị là chuỗi rỗng.

→ DOM lưu href = "".

→ Đây là trạng thái nguy hiểm nhưng hợp lệ.

###

<a href="/delete" onclick="event.preventDefault(); alert(...)">

###

DOM node <a> có:

→ href = "/delete"

→ onclick = inline JS handler

→ Trình duyệt ghi nhớ: nếu click, phải chạy JS này.

###

<div id="section2" class="box">

###

DOM tạo node div.

→ id="section2" được ghi vào DOM.

→ class="box" làm CSS .box được áp:

   - margin-top: 800px

→ Đây là đích cho href="#section2".

========================================================================================

=== NGƯỜI DÙNG BẮT ĐẦU TƯƠNG TÁC ===

========================================================================================

###

Người dùng đưa chuột vào link

###

Trình duyệt kích hoạt trạng thái :hover.

→ CSS a:hover đổi background.

→ Không có JS, không có DOM change.

###

Người dùng CLICK link "Mở Google"

###

Sự kiện click phát sinh trên node <a>.

→ Trình duyệt đọc DOM.a.href.

→ href = "https://google.com".

→ Không có onclick → không bị chặn.

→ Trình duyệt gửi HTTP request tới Google.

→ Trang hiện tại bị thay thế.

###

Người dùng CLICK link "/login"

###

Quy trình y hệt.

→ href nội bộ → request tới /login.

→ Reload toàn trang.

###

Người dùng CLICK link "/control?cmd=forward"

###

Trình duyệt đọc href.

→ Gửi request với query string.

→ Server / ESP / backend nhận lệnh.

→ Trình duyệt chuyển trang.

###

Người dùng CLICK link "#section2"

###

Trình duyệt đọc href.

→ Thấy ký tự # → KHÔNG gửi request.

→ Tìm DOM node có id="section2".

→ Tính toán vị trí.

→ Cuộn viewport xuống vị trí đó.

→ DOM KHÔNG đổi.

→ CSS KHÔNG đổi.

→ Screen reader thông báo vị trí mới.

###

Người dùng CLICK link href=""

###

Trình duyệt đọc href.

→ href là chuỗi rỗng.

→ Mặc định: reload lại trang hiện tại.

→ DOM bị hủy, tạo lại từ đầu.

###

Người dùng CLICK link "/delete" có onclick

###

Sự kiện click xảy ra.

→ Trình duyệt chuẩn bị hành vi mặc định: đi tới /delete.

→ JS trong onclick được gọi TRƯỚC.

###

event.preventDefault();

###

Dấu . :

→ truy cập phương thức preventDefault của object event.

→ event là object đại diện cho click.

→ preventDefault() = chặn hành vi mặc định.

→ Trước khi gọi:

   - Trình duyệt sắp điều hướng.

→ Sau khi gọi:

   - Điều hướng bị hủy.

###

alert('JS đã chặn link');

###

JS chạy tiếp.

→ Hiện popup.

→ Không có thay đổi DOM.

→ KẾT QUẢ:

- href vẫn tồn tại

- Nhưng không được thực thi vì JS chặn

========================================================================================

=== TỔNG KẾT VAI TRÒ ===

========================================================================================

HTML:

→ Khai báo cấu trúc và href.

DOM:

→ Lưu trạng thái href, id, class.

→ Là nơi trình duyệt đọc khi click.

CSS:

→ Chỉ phản ứng khi DOM tồn tại hoặc state đổi (:hover).

→ Không điều hướng.

JS:

→ Can thiệp hành vi click.

→ Có thể chặn hoặc cho phép href chạy.

ARIA / Accessibility:

→ <a href> = link thật.

→ Screen reader hiểu là “link”.

→ href quyết định ngữ nghĩa điều hướng.

-->

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[165]. Thuộc tính target

→ Quyết định NƠI MỞ nội dung liên kết sau khi người dùng kích hoạt link / form

------------------------------------------------

[165.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Xác định TAB / CỬA SỔ / KHUNG nào sẽ hiển thị kết quả

- Ý 2: Chỉ có tác dụng KHI HÀNH VI ĐIỀU HƯỚNG xảy ra (click / Enter)

Không có hành động điều hướng → target KHÔNG tác dụng

------------------------------------------------

[165.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Mở dashboard / trang giám sát ở tab mới

  + Mở tài liệu hướng dẫn mà không mất trang điều khiển

  + Giữ nguyên UI điều khiển khi gửi lệnh

Ví dụ ngắn:

- Click nút GIÁM SÁT → mở tab mới

- Click link HƯỚNG DẪN → không rời trang chính

-----------------------------------------------

[165.3] PHẠM VI ÁP DỤNG

- Loại: Attribute (không phải global)

- Áp cho:

  + <a> (phổ biến nhất)

  + <form>

- Có cần focus không?

  + Có (Enter trên link / form cần focus)

------------------------------------------------

[165.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <a href="..." target="giá\_trị">

- Ví dụ đúng:

  <a href="/monitor" target="\_blank">

- Ví dụ SAI:

  <a target="\_blank">

  → Không có href → target KHÔNG có tác dụng

------------------------------------------------

[165.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Chuỗi (string)

- Giá trị hay gặp:

  + "\_self"   → mở tại tab hiện tại (mặc định)

  + "\_blank"  → mở tab / cửa sổ mới

  + "\_parent" → mở ở frame cha

  + "\_top"    → mở ở top-level window

  + "tenFrame"→ mở vào iframe cụ thể

------------------------------------------------

[165.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng CLICK link hoặc nhấn Enter

2. Trình duyệt đọc href để biết ĐI ĐÂU

3. Trình duyệt đọc target để biết MỞ Ở ĐÂU

4. Trình duyệt quyết định:

   - Cùng tab

   - Tab mới

   - Frame chỉ định

5. Nội dung được render ở vị trí tương ứng

Nếu link không mở đúng chỗ → kiểm tra target

------------------------------------------------

[165.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute HTML

- Ai kích hoạt:

  + Người dùng (click / Enter)

  + Trình duyệt xử lý trực tiếp

- JavaScript:

  + Chỉ CAN THIỆP

  + Không phải nguồn gốc hành vi

target KHÔNG tự mở link, nó CHỈ ĐỊNH NƠI MỞ

-----------------------------------------------

[165.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Dùng target mà quên href

  → Link không đi đâu

- Lỗi 2: target="\_blank" nhưng không kiểm soát

  → Dễ mất context người dùng

- Nhớ nhanh:

  → href = đi đâu

  → target = mở ở đâu

------------------------------------------------

[165.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ target dùng để 📍CHỈ ĐỊNH NƠI MỞ nội dung khi 🔘NGƯỜI DÙNG KÍCH HOẠT LIÊN KẾT

165.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test thuộc tính target</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      padding: 20px;

    }

    a {

      display: inline-block;

      margin: 8px 0;

      padding: 8px 12px;

      background: #1976d2;

      color: #fff;

      text-decoration: none;

      border-radius: 4px;

    }

    a:hover {

      background: #0d47a1;

    }

    iframe {

      width: 100%;

      height: 200px;

      border: 2px solid #333;

      margin-top: 12px;

    }

  </style>

</head>

<body>

  <h1>DEMO thuộc tính target</h1>

  <!-- 1. Mặc định (\_self) -->

  <a href="https://example.com">

    Mở link (mặc định \_self)

  </a>

  <br>

  <!-- 2. Mở tab mới -->

  <a href="https://example.com" target="\_blank">

    Mở tab mới (\_blank)

  </a>

  <br>

  <!-- 3. Mở trong iframe -->

  <a href="https://example.com" target="frameTest">

    Mở trong iframe (target = frameTest)

  </a>

  <!-- iframe làm nơi nhận nội dung -->

  <iframe name="frameTest"></iframe>

  <br>

  <!-- 4. target có nhưng thiếu href (test lỗi) -->

  <a target="\_blank">

    target có nhưng KHÔNG có href (không chạy)

  </a>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[166]. Thuộc tính download

→ ÉP TRÌNH DUYỆT TẢI FILE VỀ thay vì mở nội dung đó

------------------------------------------------

[166.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Buộc link hoạt động theo chế độ TẢI XUỐNG

- Ý 2: Cho phép đặt TÊN FILE khi tải về

Click link → tải file, KHÔNG hiển thị nội dung

------------------------------------------------

[166.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Tải firmware / file cấu hình cho ESP

  + Tải log, dữ liệu đo, báo cáo

  + Xuất file kết quả cho người dùng

Ví dụ ngắn:

- Click “Tải firmware” → file .bin về máy

- Click “Xuất log” → tải file .txt

------------------------------------------------

[166.3] PHẠM VI ÁP DỤNG

- Loại: Attribute (không phải global)

- Áp cho:

  + <a> (chủ yếu)

- Có cần focus không?

  + Có (Enter kích hoạt tải)

------------------------------------------------

[166.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <a href="đường\_dẫn\_file" download>

- Đặt tên file:

  <a href="data.bin" download="firmware\_v1.bin">

- Ví dụ SAI:

  <a download>

  → Không có href → KHÔNG biết tải gì

------------------------------------------------

[166.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Chuỗi (string)

- Giá trị hay gặp:

  + download           → dùng tên file gốc

  + download="log.txt" → đổi tên file khi tải

------------------------------------------------

[166.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng CLICK link

2. Trình duyệt đọc href để biết FILE Ở ĐÂU

3. Trình duyệt thấy thuộc tính download

4. Chế độ điều hướng bị BỎ QUA

5. Trình duyệt chuyển sang chế độ TẢI FILE

6. File được lưu về máy người dùng

Nếu click mà mở file → kiểm tra có download chưa

------------------------------------------------

[166.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute HTML

- Ai kích hoạt:

  + Người dùng (click / Enter)

  + Trình duyệt xử lý trực tiếp

- JavaScript:

  + Không bắt buộc

  + Chỉ hỗ trợ thêm nếu cần

download = thay đổi HÀNH VI của href

------------------------------------------------

[166.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: File khác domain

  → Một số trình duyệt BỎ QUA download

- Lỗi 2: Server ép Content-Disposition

  → download có thể không hiệu lực

- Nhớ nhanh:

  → href = file

  → download = tải về

------------------------------------------------

[166.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ download dùng để ⬇️TẢI FILE khi 🔘NGƯỜI DÙNG CLICK LIÊN KẾT

166.20.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test thuộc tính download</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      padding: 20px;

    }

    a {

      display: inline-block;

      margin: 8px 0;

      padding: 8px 12px;

      background: #1976d2;

      color: white;

      text-decoration: none;

      border-radius: 4px;

    }

    a:hover {

      background: #0d47a1;

    }

    .note {

      margin-top: 16px;

      color: #555;

      font-size: 14px;

    }

  </style>

</head>

<body>

  <h1>DEMO thuộc tính download</h1>

  <!-- 1. Download cơ bản -->

  <a href="test.txt" download>

    ⬇️ Tải file test.txt

  </a>

  <br>

  <!-- 2. Download + đổi tên file -->

  <a href="test.txt" download="log\_he\_thong.txt">

    ⬇️ Tải file & đổi tên

  </a>

  <br>

  <!-- 3. Download hình ảnh -->

  <a href="https://via.placeholder.com/300" download="anh\_demo.png">

    ⬇️ Tải hình ảnh

  </a>

  <br>

  <!-- 4. So sánh: KHÔNG có download -->

  <a href="test.txt">

    ❌ Không có download (mở file)

  </a>

  <div class="note">

    📌 Ghi chú test:<br>

    - Link có <code>download</code> → trình duyệt TẢI FILE<br>

    - Link không có <code>download</code> → trình duyệt MỞ FILE<br>

    - Nếu file khác domain, trình duyệt có thể BỎ QUA download

  </div>

</body>

</html>

<!--

###

<!DOCTYPE html>

###

Trình duyệt đọc dòng này đầu tiên.

→ Nhận diện tài liệu HTML5.

→ Bật standards mode.

→ Quy tắc xử lý href + download được áp dụng ĐÚNG CHUẨN.

###

<html lang="vi">

###

DOM root node <html> được tạo.

→ Thuộc tính lang="vi" gắn vào DOM.

→ Screen reader ghi nhận: nội dung chính là tiếng Việt.

###

<head>

###

Trình duyệt chuyển sang pha chuẩn bị.

→ Chưa render UI.

→ Chỉ đọc metadata + CSS.

###

<meta charset="UTF-8">

###

Trình duyệt xác định bảng mã.

→ Emoji ⬇️ và chữ tiếng Việt hiển thị đúng.

###

<style>...</style>

###

Trình duyệt parse toàn bộ CSS.

→ Tạo CSSOM.

→ Chưa áp ngay, chờ DOM xuất hiện.

======================================================================================

=== DOM BẮT ĐẦU ĐƯỢC TẠO ===

======================================================================================

###

<body>

###

Node body được tạo trong DOM.

→ CSS body được áp:

   - font-family: Arial

   - padding: 20px

→ Màn hình bắt đầu có layout khung.

###

<h1>DEMO thuộc tính download</h1>

###

DOM tạo node h1.

→ Kế thừa font từ body.

→ Trình duyệt render tiêu đề.

###

<a href="test.txt" download>

###

DOM tạo node <a>.

→ DOM lưu:

   - href = "test.txt"

   - download = "" (boolean attribute)

→ Trạng thái BAN ĐẦU:

   - Link tồn tại

   - Chưa có tải gì

→ CSS a được áp (nền xanh).

→ Screen reader:

   - Nhận diện đây là LINK

   - Có hành vi tải xuống.

###

<a href="test.txt" download="log\_he\_thong.txt">

###

DOM tạo node <a> thứ hai.

→ DOM lưu:

   - href = "test.txt"

   - download = "log\_he\_thong.txt"

→ Khác biệt:

   - Có tên file TÙY CHỈNH.

→ Trình duyệt ghi nhớ tên file sẽ dùng khi tải.

###

<a href="https://via.placeholder.com/300" download="anh\_demo.png">

###

DOM tạo node <a> thứ ba.

→ href trỏ tới URL ngoài domain.

→ download có giá trị "anh\_demo.png".

→ Trình duyệt ghi nhớ:

   - ƯU TIÊN tải

   - Nhưng có thể bị BỎ QUA nếu server không cho.

###

<a href="test.txt">

###

DOM tạo node <a> cuối.

→ DOM lưu:

   - href = "test.txt"

   - KHÔNG có download

→ Đây là LINK ĐIỀU HƯỚNG BÌNH THƯỜNG.

###

<div class="note">

###

DOM tạo node div.

→ class="note" → CSS .note được áp:

   - chữ xám

   - font nhỏ

→ Chỉ để hiển thị thông tin, không có hành vi.

======================================================================================

=== NGƯỜI DÙNG BẮT ĐẦU TƯƠNG TÁC ===

======================================================================================

###

Người dùng CLICK link "⬇️ Tải file test.txt"

###

Sự kiện click phát sinh.

→ Trình duyệt đọc DOM.a.href.

→ Trình duyệt phát hiện:

   - Có thuộc tính download

→ HÀNH VI MẶC ĐỊNH ĐIỀU HƯỚNG BỊ THAY ĐỔI.

→ Trình duyệt KHÔNG mở file.

→ Chuyển sang chế độ DOWNLOAD.

→ File test.txt được lưu về máy.

→ DOM KHÔNG thay đổi.

→ CSS KHÔNG đổi.

###

Người dùng CLICK link "⬇️ Tải file & đổi tên"

###

Click xảy ra.

→ Trình duyệt đọc:

   - href = "test.txt"

   - download = "log\_he\_thong.txt"

→ Quyết định:

   - Tải file

   - Đổi tên thành log\_he\_thong.txt

→ File được lưu.

→ Không reload, không đổi DOM.

###

Người dùng CLICK link "⬇️ Tải hình ảnh"

###

Click xảy ra.

→ Trình duyệt đọc href (URL ngoài).

→ Thấy download.

→ Trình duyệt THỬ áp dụng download.

→ Nếu server cho phép:

   - File ảnh tải về

→ Nếu server không cho:

   - download bị bỏ qua

   - trình duyệt mở ảnh

→ Đây là điểm hay DEBUG.

###

Người dùng CLICK link "❌ Không có download"

###

Click xảy ra.

→ Trình duyệt đọc href = "test.txt".

→ KHÔNG thấy thuộc tính download.

→ Hành vi mặc định:

   - Điều hướng tới test.txt

→ Trang HTML hiện tại bị thay thế.

→ DOM bị hủy.

→ Tài liệu mới được load.

========================================================================================

=== TỔNG KẾT VAI TRÒ ===

========================================================================================

HTML:

→ Khai báo cấu trúc link và thuộc tính download.

DOM:

→ Lưu trạng thái href / download của từng <a>.

→ Là nơi trình duyệt đọc khi click.

CSS:

→ Chỉ ảnh hưởng GIAO DIỆN.

→ Không quyết định tải hay mở.

JavaScript:

→ KHÔNG cần thiết trong ví dụ này.

→ Trình duyệt xử lý download trực tiếp.

ARIA / Accessibility:

→ <a> + download vẫn là LINK.

→ Screen reader thông báo hành vi tải.

KẾT LUẬN NÃO BỘ:

- href = tải CÁI GÌ

- download = ÉP HÀNH VI TẢI

- Có download → KHÔNG điều hướng

- Không download → ĐIỀU HƯỚNG

-->

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[167]. Thuộc tính rel

→ Khai báo MỐI QUAN HỆ giữa thẻ hiện tại và tài nguyên được liên kết

-------------------------------------

[167.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Cho trình duyệt BIẾT link này LIÊN QUAN KIỂU GÌ

- Ý 2: Kiểm SOÁT HÀNH VI BẢO MẬT khi mở link (đặc biệt với tab mới)

rel KHÔNG điều hướng, nó CHÚ THÍCH cho link

------------------------------------------------

[167.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Mở dashboard bên ngoài mà KHÔNG cho truy cập ngược

  + Chặn trang ngoài đọc thông tin tab điều khiển

  + Khai báo link stylesheet / icon / preload

Ví dụ ngắn:

- Mở link GIÁM SÁT → tab mới + không bị kiểm soát

- Load CSS → trình duyệt biết đây là stylesheet

------------------------------------------------

[167.3] PHẠM VI ÁP DỤNG

- Loại: Attribute (không phải global)

- Áp cho:

  + <a> (liên kết)

  + <link> (CSS, icon, preload)

- Có cần focus không?

  + Không (rel KHÔNG phụ thuộc phím / chuột)

------------------------------------------------

[167.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <a href="..." rel="giá\_trị">

- Ví dụ đúng:

  <a href="https://example.com" target="\_blank" rel="noopener noreferrer">

- Ví dụ SAI:

  <a rel="noopener">

  → Không có href → rel KHÔNG có ý nghĩa

------------------------------------------------

[167.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Chuỗi (string)

  + Có thể NHIỀU GIÁ TRỊ, cách nhau bằng dấu cách

- Giá trị hay gặp:

  + "noopener"    → chặn truy cập window.opener

  + "noreferrer"  → không gửi header referrer

  + "nofollow"    → báo bot không theo link

  + "stylesheet"  → link CSS

  + "icon"        → favicon

------------------------------------------------

[167.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt gặp thẻ <a> hoặc <link>

2. Đọc href để biết tài nguyên ở đâu

3. Đọc rel để biết QUAN HỆ là gì

4. Trình duyệt quyết định:

   - Cách mở link

   - Mức độ bảo mật

   - Cách xử lý tài nguyên

5. Nội dung / kết nối được xử lý theo rel

Debug target="\_blank" → LUÔN kiểm tra rel

------------------------------------------------

[167.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute HTML

- Ai kích hoạt:

  + Trình duyệt đọc và xử lý

- Người dùng:

  + KHÔNG trực tiếp kích hoạt rel

- JavaScript:

  + Chỉ GIÁN TIẾP bị ảnh hưởng

rel = THÔNG TIN CHỈ DẪN cho trình duyệt

-----------------------------------------------

[167.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: target="\_blank" nhưng KHÔNG có rel

  → Rủi ro bảo mật (tab mới kiểm soát tab cũ)

- Lỗi 2: dùng sai rel cho <link>

  → CSS / icon không load

- Nhớ nhanh:

  → target = mở ở đâu

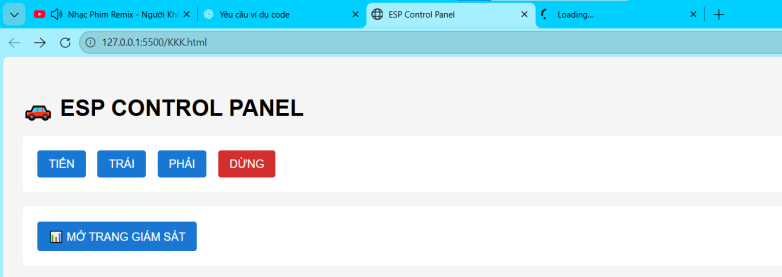
  → rel = quan hệ + an toàn

-----------------------------------------------

[167.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ rel dùng để 🔗KHAI BÁO MỐI QUAN HỆ & 🔒BẢO MẬT cho liên kết khi 🌐TRÌNH DUYỆT XỬ LÝ

167.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP Control Panel</title>

  <!-- CSS giao diện -->

  <style>

    body {

      font-family: Arial, sans-serif;

      background: #f5f5f5;

      padding: 20px;

    }

    h1 {

      margin-bottom: 10px;

    }

    .btn {

      display: inline-block;

      margin: 6px;

      padding: 10px 16px;

      background: #1976d2;

      color: #fff;

      text-decoration: none;

      border-radius: 4px;

    }

    .btn.stop {

      background: #d32f2f;

    }

    .panel {

      margin-top: 20px;

      padding: 15px;

      background: #fff;

      border-radius: 6px;

    }

  </style>

</head>

<body>

  <h1>🚗 ESP CONTROL PANEL</h1>

  <div class="panel">

    <!-- Điều khiển ESP -->

    <a class="btn" href="/control?cmd=forward">TIẾN</a>

    <a class="btn" href="/control?cmd=left">TRÁI</a>

    <a class="btn" href="/control?cmd=right">PHẢI</a>

    <a class="btn stop" href="/control?cmd=stop">DỪNG</a>

  </div>

  <div class="panel">

    <!-- Trang giám sát -->

    <a class="btn" href="http://192.168.4.1/monitor" target="\_blank" rel="noopener noreferrer">

      📊 MỞ TRANG GIÁM SÁT

    </a>

  </div>

</body>

</html>

<!--

###

<!DOCTYPE html>

###

Trình duyệt nhận diện tài liệu HTML5.

→ Bật standards mode.

→ Cho phép xử lý đầy đủ CSS, href, target, rel theo chuẩn hiện đại.

###

<html lang="vi">

###

DOM root <html> được tạo.

→ Thuộc tính lang="vi" gắn vào DOM.

→ Screen reader ghi nhận: ngôn ngữ chính là tiếng Việt.

###

<head>

###

Trình duyệt đi vào pha KHỞI TẠO.

→ Chưa vẽ giao diện.

→ Chỉ đọc metadata và CSS.

###

<meta charset="UTF-8">

###

Trình duyệt xác định bảng mã.

→ Emoji 🚗 📊 và chữ tiếng Việt hiển thị đúng.

###

<style>...</style>

###

Trình duyệt parse CSS.

→ Tạo CSSOM.

→ Chưa áp vào đâu vì DOM body chưa tồn tại.

==============================

=== DOM BẮT ĐẦU ĐƯỢC TẠO ===

==============================

###

<body>

###

Node body được tạo trong DOM.

→ CSS body được áp ngay:

   - font Arial

   - background xám nhạt

   - padding 20px

→ Khung giao diện xuất hiện.

###

<h1>🚗 ESP CONTROL PANEL</h1>

###

DOM tạo node h1.

→ Kế thừa font từ body.

→ Trình duyệt render tiêu đề.

→ Screen reader đọc: “ESP CONTROL PANEL”.

###

<div class="panel">

###

DOM tạo node div.panel.

→ class="panel" → CSS .panel được áp:

   - nền trắng

   - bo góc

   - padding

→ Đây là KHUNG điều khiển ESP.

###

<a class="btn" href="/control?cmd=forward">

###

DOM tạo node <a>.

→ DOM lưu:

   - class="btn"

   - href="/control?cmd=forward"

→ CSS .btn được áp (nút xanh).

→ Screen reader:

   - nhận diện là LINK

   - có nhãn “TIẾN”.

###

<a class="btn" href="/control?cmd=left">

###

DOM tạo link điều khiển TRÁI.

→ href chứa query cmd=left.

→ Chưa gửi lệnh gì, chỉ lưu trạng thái DOM.

###

<a class="btn" href="/control?cmd=right">

###

DOM tạo link điều khiển PHẢI.

→ Cơ chế giống 2 link trên.

###

<a class="btn stop" href="/control?cmd=stop">

###

DOM tạo link DỪNG.

→ class="btn stop":

   - CSS .btn áp trước

   - CSS .btn.stop ghi đè màu đỏ

→ Đây là trạng thái KHÁC BIỆT về giao diện.

###

<div class="panel">

###

DOM tạo panel thứ hai.

→ CSS .panel được áp.

→ Đây là khu vực GIÁM SÁT.

###

<a class="btn"

   href="http://192.168.4.1/monitor"

   target="\_blank"

   rel="noopener noreferrer">

###

DOM tạo link giám sát.

→ DOM lưu:

   - href = URL ESP monitor

   - target = "\_blank"

   - rel = "noopener noreferrer"

→ CSS .btn được áp.

→ Screen reader:

   - đọc là link

   - thông báo mở trang mới.

================================

=== NGƯỜI DÙNG BẮT ĐẦU TƯƠNG TÁC ===

================================

###

Người dùng CLICK nút "TIẾN"

###

Sự kiện click phát sinh trên <a>.

→ Trình duyệt đọc DOM.a.href.

   - dấu . :

     → truy cập thuộc tính href của object <a>.

→ href = "/control?cmd=forward".

→ KHÔNG có JS chặn.

→ Trình duyệt gửi HTTP request tới ESP.

→ ESP nhận lệnh: TIẾN.

→ Trình duyệt điều hướng (reload trang nếu ESP trả HTML).

###

Người dùng CLICK nút "DỪNG"

###

Click xảy ra.

→ Trình duyệt đọc href="/control?cmd=stop".

→ Gửi request.

→ ESP dừng động cơ.

→ DOM hiện tại bị thay thế hoặc reload.

###

Người dùng CLICK "📊 MỞ TRANG GIÁM SÁT"

###

Sự kiện click phát sinh.

→ Trình duyệt đọc href trước:

   - URL monitor.

→ Đọc target="\_blank":

   - quyết định mở TAB MỚI.

→ Đọc rel="noopener noreferrer":

   - CẮT liên kết window.opener.

   - Trang monitor KHÔNG truy cập được DOM trang điều khiển.

→ KẾT QUẢ:

- Tab mới mở trang giám sát.

- Tab điều khiển vẫn an toàn.

- Không DOM nào bị sửa ở trang gốc.

================================

=== SAU KHI DOM KHÔNG ĐỔI ===

================================

###

DOM trạng thái

###

→ Không có JS ghi DOM.

→ Không setAttribute.

→ Không classList.toggle.

→ DOM giữ nguyên.

###

CSS phản ứng

###

→ Chỉ đổi màu khi hover.

→ Không phụ thuộc logic điều khiển.

###

ARIA / Screen Reader

###

→ <a> được hiểu là LINK.

→ target="\_blank" → thông báo mở cửa sổ mới.

→ rel giúp tăng an toàn nhưng KHÔNG ảnh hưởng đọc màn hình.

================================

=== TỔNG KẾT VAI TRÒ ===

================================

HTML:

→ Khai báo cấu trúc, href, target, rel.

DOM:

→ Lưu trạng thái từng nút điều khiển.

CSS:

→ Chỉ lo giao diện (màu, layout).

JavaScript:

→ KHÔNG dùng.

→ Hành vi do TRÌNH DUYỆT xử lý.

ARIA / Accessibility:

→ Link được hiểu đúng ngữ nghĩa.

→ Trang giám sát mở riêng, an toàn.

👉 TƯ DUY CHUẨN ESP:

- href = GỬI LỆNH

- target = MỞ Ở ĐÂU

- rel = CẮT QUYỀN NGƯỢC

- KHÔNG JS vẫn điều khiển được ESP

-->

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[168]. Thuộc tính referrerpolicy

→ Quy định TRÌNH DUYỆT có gửi URL trang hiện tại (Referer) sang trang khác hay không, và gửi tới MỨC ĐỘ NÀO.

------------------------------------------------

[168.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Kiểm soát thông tin URL nguồn được gửi khi tải link / ảnh / script

- Ý 2: Dùng khi click link, load tài nguyên ngoài, gửi request HTTP

Không xử lý sự kiện, chỉ kiểm soát dữ liệu gửi kèm

-----------------------------------------------

[168.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Bảo mật URL nội bộ

  + Tránh lộ query, token, đường dẫn admin

  + Kiểm soát dữ liệu gửi sang server khác

Ví dụ ngắn:

- Click link ngoài → KHÔNG gửi URL trang gốc

- Load ảnh CDN → chỉ gửi domain, không gửi full path

------------------------------------------------

[168.3] PHẠM VI ÁP DỤNG

- Global attribute (áp cho nhiều thẻ)

- Áp cho:

  + <a>, <img>, <iframe>, <script>, <link>

- Không cần focus

- Không liên quan bàn phím / chuột trực tiếp

------------------------------------------------

[168.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag referrerpolicy="giá\_trị">

- Ví dụ đúng:

  <a href="https://example.com" referrerpolicy="no-referrer">Link</a>

- Ví dụ SAI:

  <a referrerpolicy=no-referrer>

  → Thiếu dấu ""

------------------------------------------------

[168.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị: Chuỗi cố định (string enum)

- Giá trị hay gặp:

  + no-referrer          : Không gửi Referer

  + no-referrer-when-downgrade (mặc định)

  + origin               : Chỉ gửi domain

  + origin-when-cross-origin

  + same-origin

  + strict-origin

  + strict-origin-when-cross-origin

  + unsafe-url           : Gửi FULL URL (ít dùng)

------------------------------------------------

[168.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng click link / trình duyệt load tài nguyên

2. Trình duyệt đọc referrerpolicy của thẻ

3. Trình duyệt quyết định CÓ gửi Referer hay không

4. HTTP request được gửi đi (có hoặc không header Referer)

5. Server bên kia nhận request với mức thông tin tương ứng

👉 JS KHÔNG can thiệp trực tiếp, chỉ gián tiếp qua HTML

------------------------------------------------

[168.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute (KHÔNG phải event, KHÔNG phải ARIA)

- Ai kích hoạt:

  + Trình duyệt tự động xử lý

- Người dùng chỉ là nguyên nhân gián tiếp (click / load)

👉 “Thằng quyết định cuối là TRÌNH DUYỆT”

------------------------------------------------

[168.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Nghĩ referrerpolicy là thuộc tính JS → SAI

- Lỗi 2: Dùng unsafe-url làm lộ thông tin nhạy cảm

- Cách nhớ nhanh:

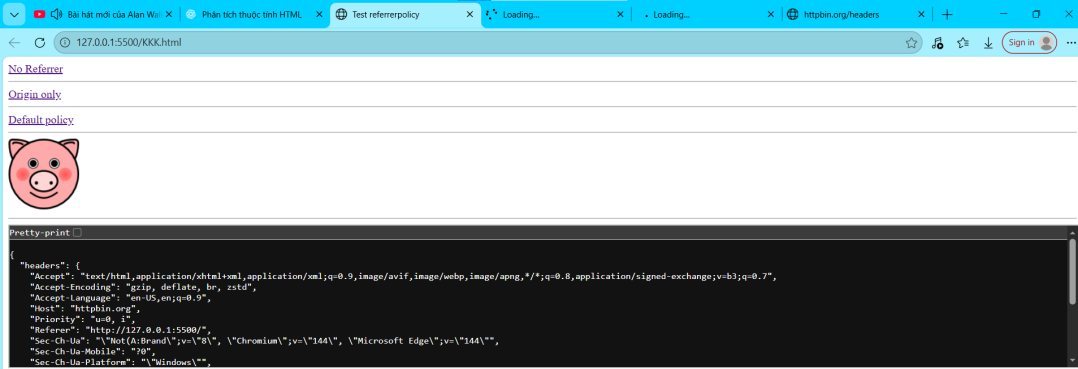
  → CÀNG strict thì CÀNG ÍT gửi thông tin

------------------------------------------------

[168.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “referrerpolicy dùng để kiểm soát URL nguồn được gửi khi trình duyệt tải link hoặc tài nguyên”

168.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test referrerpolicy</title>

</head>

<body>

  <!-- 1️⃣ KHÔNG gửi Referer -->

  <a href="https://httpbin.org/headers" referrerpolicy="no-referrer" target="\_blank">

    No Referrer

  </a>

  <hr>

  <!-- 2️⃣ Chỉ gửi ORIGIN (https://your-site.com) -->

  <a href="https://httpbin.org/headers" referrerpolicy="origin" target="\_blank">

    Origin only

  </a>

  <hr>

  <!-- 3️⃣ Mặc định (thường là strict-origin-when-cross-origin) -->

  <a href="https://httpbin.org/headers" target="\_blank">

    Default policy

  </a>

  <hr>

  <!-- 4️⃣ IMG: không gửi URL trang chứa ảnh -->

  <img src="https://httpbin.org/image/png" referrerpolicy="no-referrer" alt="test image">

  <hr>

  <!-- 5️⃣ IFRAME: chỉ gửi origin -->

  <iframe src="https://httpbin.org/headers" referrerpolicy="origin" width="100%" height="200">

  </iframe>

</body>

</html>

<!--

###

TRÌNH DUYỆT BẮT ĐẦU LOAD FILE HTML

###

- Trình duyệt đọc <!DOCTYPE html> → xác định chế độ standards

- Bắt đầu parse HTML từ trên xuống

- Chưa có JS chạy, chưa có request ra ngoài

###

<html lang="vi"> + <head> + <meta charset> + <title>

###

- Trình duyệt tạo các node DOM tương ứng:

  html → head → meta, title

- DOM lúc này CHỈ tồn tại trong bộ nhớ

- Chưa có CSS riêng → dùng CSS mặc định của trình duyệt (user-agent stylesheet)

- Chưa có referrer nào được gửi vì chưa có request mạng

###

<body> được parse

###

- Trình duyệt tạo body node

- Bắt đầu tạo các phần tử tương tác (a, img, iframe)

- Mọi thứ vẫn ở trạng thái TĨNH

==================================================

GIAI ĐOẠN 1: CÁC THẺ <a> ĐƯỢC TẠO

==================================================

###

<a href="https://httpbin.org/headers" referrerpolicy="no-referrer" target="\_blank">

###

- DOM tạo node <a>

- Thuộc tính được gắn:

  + href → URL đích

  + referrerpolicy → "no-referrer"

  + target → "\_blank"

- referrerpolicy lúc này CHỈ LÀ DỮ LIỆU, chưa kích hoạt gì

- CSS mặc định:

  + a có màu xanh

  + có underline

- ARIA: không có → screen reader đọc là “link”

###

<a href="https://httpbin.org/headers" referrerpolicy="origin" target="\_blank">

###

- Tương tự link trên nhưng:

  + referrerpolicy = "origin"

- DOM ghi nhớ giá trị này để DÙNG SAU

- Chưa có request → chưa có Referer

###

<a href="https://httpbin.org/headers" target="\_blank">

###

- DOM tạo link KHÔNG có referrerpolicy

- Trình duyệt gán policy mặc định nội bộ

  (thường là strict-origin-when-cross-origin)

==================================================

GIAI ĐOẠN 2: IMG & IFRAME ĐƯỢC TẠO

==================================================

###

<img src="https://httpbin.org/image/png" referrerpolicy="no-referrer">

###

- KHÁC với <a>:

  → img TỰ ĐỘNG tạo request NGAY KHI DOM xong

- Trình duyệt:

  1. Thấy src

  2. Thấy referrerpolicy="no-referrer"

  3. Quyết định: KHÔNG gửi header Referer

- HTTP request được gửi:

  GET /image/png

  (KHÔNG có Referer)

- DOM trước:

  + img trống

- DOM sau:

  + img có bitmap ảnh

- CSS:

  + hiển thị ảnh

- Screen reader:

  + đọc alt="test image"

###

<iframe src="https://httpbin.org/headers" referrerpolicy="origin">

###

- iframe cũng tạo request NGAY LẬP TỨC

- Trình duyệt:

  1. Đọc src

  2. Đọc referrerpolicy="origin"

  3. Tính origin của trang hiện tại

- HTTP request gửi đi:

  Referer: https://domain-hien-tai/

- DOM:

  + iframe trước: trống

  + iframe sau: có document con

- CSS:

  + iframe chiếm width/height đã khai báo

- ARIA:

  + screen reader coi iframe là vùng nội dung nhúng

==================================================

GIAI ĐOẠN 3: NGƯỜI DÙNG CLICK LINK

==================================================

###

Người dùng click vào link "No Referrer"

###

- Sự kiện click phát sinh

- Trình duyệt xử lý HÀNH VI MẶC ĐỊNH của <a>

- JS: KHÔNG có → không can thiệp

###

Trình duyệt chuẩn bị request cho <a referrerpolicy="no-referrer">

###

- Đọc thuộc tính:

  referrerpolicy = "no-referrer"

- Quyết định:

  → KHÔNG gửi Referer

- Request:

  GET /headers

  (không có Referer)

- target="\_blank":

  → mở tab mới

- DOM trang gốc: KHÔNG ĐỔI

###

Người dùng click "Origin only"

###

- referrerpolicy = "origin"

- Trình duyệt gửi:

  Referer: https://domain-hien-tai/

- Không gửi path, query

- DOM trang gốc: KHÔNG ĐỔI

###

Người dùng click "Default policy"

###

- Không có referrerpolicy trong DOM

- Trình duyệt dùng policy mặc định

- Cross-origin:

  → gửi origin, không gửi full URL

==================================================

KHÔNG CÓ GIAI ĐOẠN JS CHẠY

==================================================

- Không có:

  . (truy cập object)

  === (so sánh)

  ? : (rẽ nhánh)

  setAttribute

  classList.toggle

- MỌI QUYẾT ĐỊNH:

  → do TRÌNH DUYỆT xử lý nội bộ

==================================================

TỔNG KẾT CUỐI LUỒNG

==================================================

- HTML:

  + Khai báo referrerpolicy

- DOM:

  + Lưu giá trị referrerpolicy

- Trình duyệt:

  + Đọc DOM

  + Quyết định gửi hay không gửi Referer

- CSS:

  + Chỉ hiển thị, không can thiệp logic

- JS:

  + Không tồn tại trong luồng này

- ARIA:

  + Không bị ảnh hưởng bởi referrerpolicy

👉 Đây là ví dụ thuần “HTML ra lệnh – trình duyệt thi hành”

👉 referrerpolicy = CHÍNH SÁCH, không phải hành động

-->

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[169]. Thuộc tính name

→ Dùng để ĐỊNH DANH dữ liệu/phần tử nhằm gửi dữ liệu, gom nhóm hoặc cho JS/server nhận biết.

------------------------------------------------

[169.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Gắn TÊN cho dữ liệu để gửi đi (form submit)

- Ý 2: Dùng để gom nhóm hoặc truy xuất (radio, input, JS)

Không tự kích hoạt hành động, chỉ đặt NHÃN

------------------------------------------------

[169.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Gửi lệnh / dữ liệu lên server

  + Gom nhóm lựa chọn (radio)

  + JS đọc giá trị theo tên

Ví dụ ngắn:

- Nhập tốc độ → name="speed" → server nhận speed=50

- Chọn hướng → radio cùng name="dir" → chỉ chọn 1

------------------------------------------------

[169.3] PHẠM VI ÁP DỤNG

- Global attribute (nhiều thẻ dùng được)

- Áp cho:

  + <input>, <select>, <textarea>, <button>

  + <form>, <iframe>, <meta>

- Không cần focus (trừ khi là input để nhập)

------------------------------------------------

[169.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag name="giá\_trị">

- Ví dụ đúng:

  <input name="username">

  <input type="radio" name="mode">

- Ví dụ SAI:

  <input name>

  → Thiếu giá trị, server không nhận dữ liệu

------------------------------------------------

[169.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị: Chuỗi (string)

- Ví dụ hay gặp:

  + "username"

  + "password"

  + "speed"

  + "direction"

  + "mode"

KHÔNG dùng khoảng trắng

------------------------------------------------

[169.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng nhập/chọn dữ liệu

2. Người dùng submit form hoặc JS đọc dữ liệu

3. Trình duyệt gom dữ liệu theo cặp:

   name = value

4. Gửi request (GET/POST) hoặc JS nhận object

5. Server / JS xử lý theo TÊN đã đặt

Không có name → dữ liệu BỊ BỎ QUA

------------------------------------------------

[169.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt (khi submit)

  + JavaScript (khi đọc form)

- Người dùng chỉ tạo dữ liệu

name = “nhãn dán” cho dữ liệu

-----------------------------------------------

[169.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Quên name → server KHÔNG nhận gì

- Lỗi 2: Radio khác nhóm nhưng trùng name → chọn sai

- Cách nhớ nhanh:

  → GỬI dữ liệu = BẮT BUỘC có name

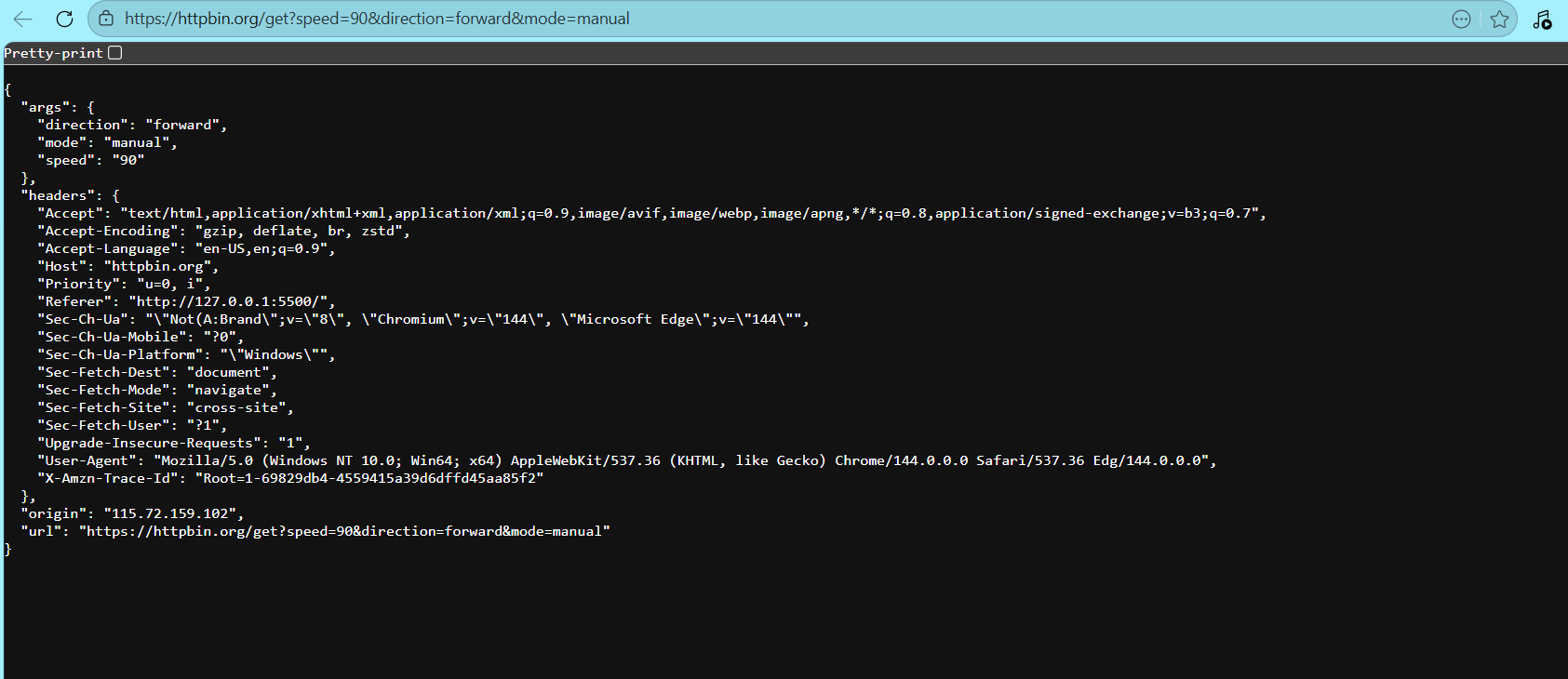
------------------------------------------------

[169.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “name dùng để đặt tên cho dữ liệu khi gửi hoặc xử lý”

169.10.EXAMBLE





<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test thuộc tính name</title>

</head>

<body>

  <!-- 1️⃣ INPUT có name → dữ liệu được gửi -->

  <form action="https://httpbin.org/get" method="GET" target="\_blank">

    <label>

      Tốc độ:

      <input type="number" name="speed" value="50">

    </label>

    <br><br>

    <!-- 2️⃣ RADIO cùng name → chỉ chọn 1 -->

    Hướng đi:

    <label>

      <input type="radio" name="direction" value="forward" checked>

      Tiến

    </label>

    <label>

      <input type="radio" name="direction" value="backward">

      Lùi

    </label>

    <br><br>

    <!-- 3️⃣ SELECT có name -->

    Chế độ:

    <select name="mode">

      <option value="auto">Tự động</option>

      <option value="manual">Thủ công</option>

    </select>

    <br><br>

    <button type="submit">Gửi dữ liệu</button>

  </form>

  <hr>

  <!-- 4️⃣ INPUT KHÔNG có name → KHÔNG được gửi -->

  <input type="text" value="Tôi sẽ bị bỏ qua">

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[170]. Thuộc tính value

→ Xác định GIÁ TRỊ DỮ LIỆU hiện tại của phần tử để hiển thị, gửi đi hoặc cho JS đọc.

------------------------------------------------

[170.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Chứa giá trị dữ liệu của input / option / button

- Ý 2: Dùng khi nhập liệu, chọn lựa, submit form

value = “thứ sẽ được gửi hoặc xử lý”

------------------------------------------------

[170.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Gửi lệnh (tốc độ, hướng, mode)

  + Gửi dữ liệu lên server

  + Xác định trạng thái hiện tại

Ví dụ ngắn:

- Nhấn nút → gửi value="start"

- Chọn radio → gửi value="forward"

------------------------------------------------

[170.3] PHẠM VI ÁP DỤNG

- Attribute (không phải event)

- Áp cho:

  + <input>, <button>, <option>, <select>

- Không cần focus để tồn tại

- Nhưng cần focus nếu người dùng nhập để value thay đổi

------------------------------------------------

[170.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag value="giá\_trị">

- Ví dụ đúng:

  <input type="text" value="abc">

  <button value="start">Start</button>

- Ví dụ SAI:

  <input value>

  → Không có giá trị → value rỗng

------------------------------------------------

[170.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Chuỗi

  + Số

- Ví dụ hay gặp:

  + "on", "off"

  + "start", "stop"

  + 0, 50, 100

Luôn là STRING khi gửi form

------------------------------------------------

[170.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt đọc value khi tạo DOM

2. Người dùng nhập/chọn → value thay đổi

3. Khi submit:

   → trình duyệt lấy value hiện tại

4. Ghép thành cặp:

   name = value

5. Gửi request hoặc JS đọc element.value

value là dữ liệu SỐNG, có thể đổi

------------------------------------------------

[170.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute + Property

- Ai kích hoạt:

  + Người dùng (nhập/chọn)

  + JavaScript (ghi/đọc)

  + Trình duyệt (submit)

HTML đặt value BAN ĐẦU, JS dùng value HIỆN TẠI

------------------------------------------------

[170.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Có name nhưng quên value → gửi rỗng

- Lỗi 2: Nhầm value hiển thị với text hiển thị

- Cách nhớ nhanh:

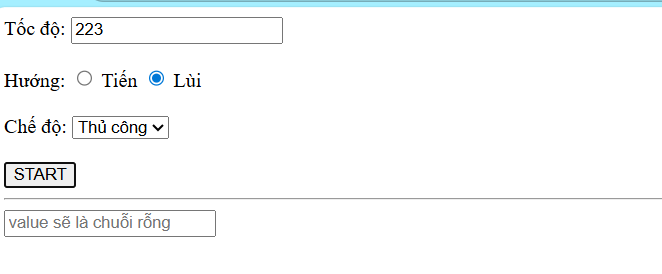
  → SERVER CHỈ NHÌN value, KHÔNG nhìn chữ ngoài

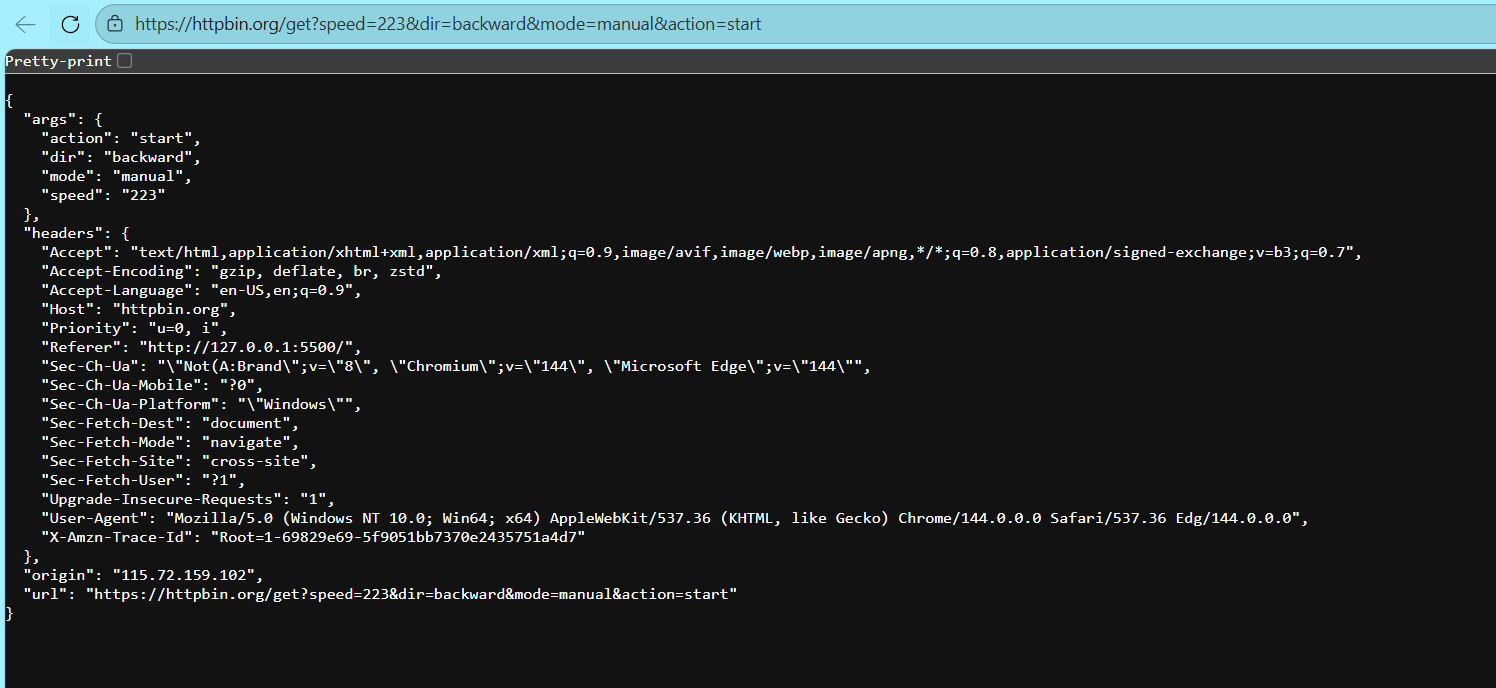
------------------------------------------------

[170.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “value dùng để chứa và gửi giá trị dữ liệu khi người dùng thao tác”

170.10.EXAMBLE





<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test thuộc tính value</title>

</head>

<body>

  <form action="https://httpbin.org/get" method="GET" target="\_blank">

    <!-- 1️⃣ INPUT: value là giá trị BAN ĐẦU -->

    Tốc độ:

    <input type="number" name="speed" value="50">

    <!-- người dùng sửa → value đổi -->

    <br><br>

    <!-- 2️⃣ RADIO: value là dữ liệu gửi đi -->

    Hướng:

    <label>

      <input type="radio" name="dir" value="forward" checked>

      Tiến

    </label>

    <label>

      <input type="radio" name="dir" value="backward">

      Lùi

    </label>

    <br><br>

    <!-- 3️⃣ SELECT + OPTION: value nằm ở OPTION -->

    Chế độ:

    <select name="mode">

      <option value="auto">Tự động</option>

      <option value="manual">Thủ công</option>

    </select>

    <br><br>

    <!-- 4️⃣ BUTTON: value là lệnh gửi -->

    <button type="submit" name="action" value="start">

      START

    </button>

  </form>

  <hr>

  <!-- 5️⃣ INPUT không có value -->

  <input type="text" placeholder="value sẽ là chuỗi rỗng">

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[171]. Thuộc tính disabled

→ Vô hiệu hóa phần tử, khiến người dùng KHÔNG thể tương tác và trình duyệt BỎ QUA nó.

------------------------------------------------

[171.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Chặn người dùng click / nhập / chọn vào phần tử

- Ý 2: Dùng khi trạng thái chưa sẵn sàng, chưa cho phép thao tác

disabled = “khóa cứng tương tác”

------------------------------------------------

[171.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

+ Khóa nút điều khiển (xe / robot / ESP)

+ Chặn gửi lệnh khi thiếu dữ liệu

+ Hiển thị trạng thái chưa sẵn sàng

Ví dụ ngắn:

- Chưa kết nối → nút START bị khóa

- Đủ điều kiện → JS mở khóa nút

------------------------------------------------

[171.3] PHẠM VI ÁP DỤNG

- Attribute (KHÔNG phải event)

- Áp cho:

+ <button>, <input>, <select>, <textarea>, <option>

- KHÔNG áp cho <div>, <span>

- Không cần focus (vì bị disabled thì KHÔNG focus được)

------------------------------------------------

[171.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

<tag disabled>

- Ví dụ đúng:

<button disabled>START</button>

<input disabled>

- Ví dụ SAI:

<button disabled="false">

→ SAI LOGIC: có disabled là BỊ KHÓA

------------------------------------------------

[171.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

+ Boolean attribute

- Giá trị hợp lệ:

+ disabled

+ (không có) → enabled

Không dùng true / false

------------------------------------------------

[171.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt parse DOM, thấy disabled

2. Gắn trạng thái "vô hiệu hóa" cho phần tử

3. Người dùng click / gõ phím

4. Trình duyệt CHẶN sự kiện ngay từ đầu

5. Không gửi dữ liệu, không phát sinh hành động

JS chỉ chạy nếu disabled bị gỡ

------------------------------------------------

[171.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute + State

- Ai kích hoạt:

+ HTML (ban đầu)

+ JavaScript (bật / tắt)

- Trình duyệt là thằng chặn tương tác

disabled = quyền lực của TRÌNH DUYỆT

------------------------------------------------

[171.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: disabled="false" nhưng vẫn bị khóa

- Lỗi 2: Quên gỡ disabled nên nút không bấm được

- Cách nhớ nhanh:

→ THẤY disabled = COI NHƯ KHÔNG TỒN TẠI

------------------------------------------------

[171.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “disabled dùng để khóa phần tử khi chưa cho phép người dùng thao tác”

171.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test disabled</title>

</head>

<body>

  <form action="https://httpbin.org/get" method="GET" target="\_blank">

    <!-- 1️⃣ INPUT bị khóa → không nhập được -->

    Tốc độ:

    <input type="number" name="speed" value="50" disabled>

    <br><br>

    <!-- 2️⃣ RADIO bị khóa → không chọn được -->

    Hướng:

    <label>

      <input type="radio" name="dir" value="forward" disabled>

      Tiến

    </label>

    <label>

      <input type="radio" name="dir" value="backward" disabled>

      Lùi

    </label>

    <br><br>

    <!-- 3️⃣ SELECT bị khóa -->

    Chế độ:

    <select name="mode" disabled>

      <option value="auto">Tự động</option>

      <option value="manual">Thủ công</option>

    </select>

    <br><br>

    <!-- 4️⃣ BUTTON bị khóa → không submit -->

    <button type="submit" disabled>

      START

    </button>

  </form>

  <hr>

  <!-- 5️⃣ SO SÁNH: input KHÔNG disabled -->

  <input type="text" placeholder="Tôi nhập được bình thường">

</body>

</html>

171.11.EXAMBLE



SAU KHI BẤM KẾT NỐI ESP



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP Control Panel</title>

  <style>

    button {

      padding: 12px 20px;

      font-size: 16px;

      margin: 6px;

    }

    button:disabled {

      background: #ccc;

      color: #666;

      cursor: not-allowed;

    }

  </style>

</head>

<body>

  <h2>ESP ROBOT CONTROL</h2>

  <!-- Trạng thái kết nối -->

  <p>

    Trạng thái ESP:

    <strong id="status">CHƯA KẾT NỐI</strong>

  </p>

  <!-- NÚT ĐIỀU KHIỂN -->

  <button disabled id="btnStart">START</button>

  <button disabled id="btnStop">STOP</button>

  <hr>

  <!-- NÚT GIẢ LẬP ESP KẾT NỐI -->

  <button id="connectBtn">Kết nối ESP</button>

  <script>

    const status = document.getElementById("status");

    const start = document.getElementById("btnStart");

    const stop = document.getElementById("btnStop");

    const connect = document.getElementById("connectBtn");

    // Giả lập ESP kết nối thành công

    connect.onclick = () => {

      status.textContent = "ĐÃ KẾT NỐI";

      // 👉 MỞ KHÓA ĐIỀU KHIỂN

      start.disabled = false;

      stop.disabled = false;

    };

  </script>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[172]. Thuộc tính autofocus

→ Ra lệnh cho TRÌNH DUYỆT tự động đặt CON TRỎ (focus) vào phần tử ngay khi trang load xong.

------------------------------------------------

[172.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Tự động focus vào input / button quan trọng nhất

- Ý 2: Dùng khi vừa load trang, KHÔNG cần click

autofocus = “vào trang là trỏ sẵn”

------------------------------------------------

[172.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Nhập lệnh nhanh (ESP / robot)

  + Nhập mật khẩu / tốc độ ngay lập tức

  + Điều khiển bằng bàn phím

@ Ví dụ ngắn:

- Mở web ESP → con trỏ nằm sẵn ở ô SPEED

- Gõ số → Enter → gửi lệnh ngay

------------------------------------------------

[172.3] PHẠM VI ÁP DỤNG

- Attribute (KHÔNG phải event)

- Áp cho:

  + <input>, <textarea>, <select>, <button>

- CẦN focus (autofocus chính là để focus)

- Mỗi trang CHỈ nên có 1 autofocus

------------------------------------------------

[172.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag autofocus>

- Ví dụ đúng:

  <input autofocus>

  <button autofocus>

- Ví dụ SAI:

  <input autofocus="false">

  → SAI LOGIC: có autofocus là SẼ focus

------------------------------------------------

[172.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Boolean attribute

- Giá trị hợp lệ:

  + autofocus

  + (không có) → không tự focus

KHÔNG dùng true / false

------------------------------------------------

[172.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt parse xong DOM

2. Trang hoàn tất load

3. Trình duyệt quét DOM tìm autofocus

4. Đặt focus vào phần tử đó

5. Con trỏ nhấp nháy / viền focus xuất hiện

JS CHƯA cần chạy

------------------------------------------------

[172.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute + Focus state

- Ai kích hoạt:

  + Trình duyệt tự động

- Người dùng KHÔNG cần thao tác

autofocus = mệnh lệnh cho TRÌNH DUYỆT

------------------------------------------------

[172.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Dùng nhiều autofocus → chỉ thằng ĐẦU TIÊN được

- Lỗi 2: autofocus trên phần tử disabled → KHÔNG hoạt động

- Cách nhớ nhanh:

  → autofocus CHỈ dành cho “ô quan trọng nhất”

------------------------------------------------

[172.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “autofocus dùng để tự động đặt con trỏ vào phần tử khi trang vừa load”

172.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test autofocus</title>

</head>

<body>

  <form>

    <!-- vào trang là con trỏ nhảy vào ô này -->

    <input type="text" placeholder="Nhập lệnh" autofocus>

    <br><br>

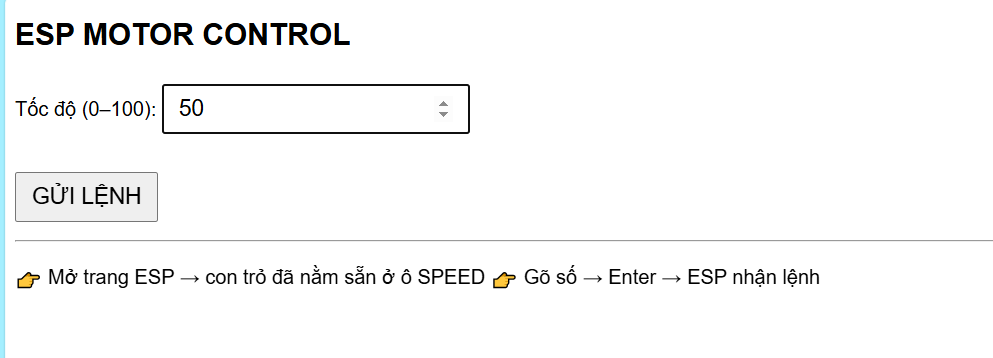
    <button>Gửi</button>

  </form>

</body>

</html>

172.11.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP Control UI</title>

  <style>

    body {

      font-family: Arial, sans-serif;

    }

    input,

    button {

      font-size: 18px;

      padding: 8px 12px;

      margin: 6px 0;

    }

  </style>

</head>

<body>

  <h2>ESP MOTOR CONTROL</h2>

  <!-- 1️⃣ autofocus: vào trang là trỏ nhảy vào đây -->

  <label>

    Tốc độ (0–100):

    <input type="number" name="speed" value="50" autofocus>

  </label>

  <br><br>

  <!-- 2️⃣ nhấn Enter là gửi lệnh -->

  <button>

    GỬI LỆNH

  </button>

  <hr>

  <p>

    👉 Mở trang ESP → con trỏ đã nằm sẵn ở ô SPEED

    👉 Gõ số → Enter → ESP nhận lệnh

  </p>

</body>

</html>

========================================

=== PHÂN TÍCH THUỘC TÍNH HTML: placeholder ===

========================================

173. Thuộc tính placeholder

→ Hiển thị chữ GỢI Ý bên trong ô nhập khi ô đang RỖNG

------------------------------------------------

[173.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Hiển thị nội dung gợi ý cho người dùng biết cần nhập gì

- Ý 2: Chỉ xuất hiện khi input chưa có giá trị và chưa được nhập

------------------------------------------------

[173.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Hướng dẫn người dùng nhập dữ liệu đúng

  + Giảm cần ghi chú ngoài giao diện

  + Tránh người dùng nhập sai kiểu dữ liệu

▶ Ví dụ ngắn:

- Ô nhập: placeholder="Nhập mật khẩu"

- Người dùng nhìn → biết cần nhập gì → nhập xong → chữ gợi ý biến mất

-----------------------------------------------

[173.3] PHẠM VI ÁP DỤNG

- Loại: Attribute (HTML)

- Áp cho:

  + <input> (text, password, email, search…)

  + <textarea>

- Có cần focus không?

  + KHÔNG cần focus

  + Chỉ cần ô đang rỗng là hiện

------------------------------------------------

[173.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag placeholder="giá trị">

- Ví dụ đúng:

  <input type="text" placeholder="Nhập tên">

- Ví dụ SAI:

  <input placeholder=Nhập tên>

  → Sai vì thiếu dấu nháy cho chuỗi

------------------------------------------------

[173.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Chuỗi (text)

- Ví dụ giá trị hay gặp:

  + "Nhập email"

  + "Tìm kiếm..."

  + "Nhập số điện thoại"

------------------------------------------------

[173.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt render input

2. Kiểm tra value của input

3. Nếu value = rỗng → hiển thị placeholder

4. Người dùng bắt đầu nhập

5. Trình duyệt ẩn placeholder

6. Khi xóa hết nội dung → placeholder hiện lại

▶ Mục này giúp DEBUG:

- Thấy placeholder không hiện → kiểm tra input có value mặc định không

------------------------------------------------

[173.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute (HTML thuần)

- Ai kích hoạt:

  + Trình duyệt tự xử lý

- JavaScript:

  + KHÔNG cần JS vẫn hoạt động

  + JS chỉ đọc/ghi value, không điều khiển placeholder trực tiếp

------------------------------------------------

[173.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Nhầm placeholder là giá trị thật

  → Placeholder KHÔNG được submit lên server

- Lỗi 2: Dùng placeholder thay cho <label>

  → Gây lỗi accessibility

- Cách nhớ nhanh:

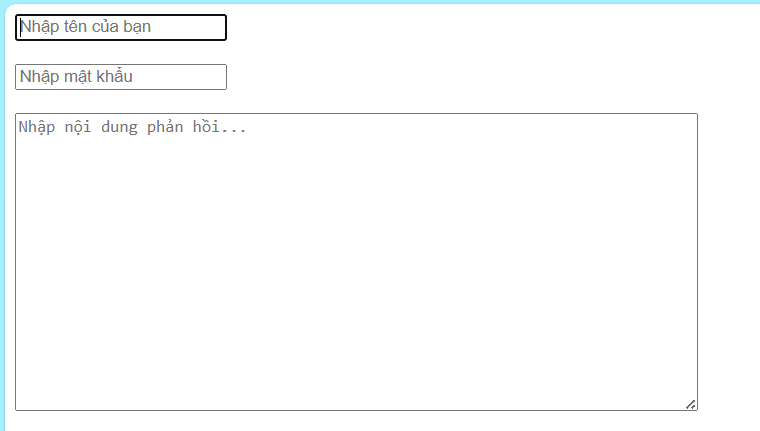
  → Placeholder chỉ là GỢI Ý, không phải DỮ LIỆU

------------------------------------------------

[173.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ placeholder dùng để HIỂN THỊ CHỮ GỢI Ý khi ô nhập đang RỖNG

173.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ placeholder</title>

</head>

<body>

  <!-- Ví dụ 1: input text -->

  <input type="text" placeholder="Nhập tên của bạn">

  <br><br>

  <!-- Ví dụ 2: input password -->

  <input type="password" placeholder="Nhập mật khẩu">

  <br><br>

  <!-- Ví dụ 3: textarea -->

  <textarea placeholder="Nhập nội dung phản hồi..." rows="4" cols="30"></textarea>

</body>

</html>

========================================

=== PHÂN TÍCH THUỘC TÍNH HTML: checked ===

========================================

174. Thuộc tính checked

  → ĐÁNH DẤU trạng thái ĐÃ CHỌN sẵn cho checkbox hoặc radio

  ------------------------------------------------

  [174.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

  - Ý 1: Tự động đặt trạng thái ĐÃ CHỌN cho input

  - Ý 2: Dùng khi cần có lựa chọn mặc định lúc trang vừa tải

  ------------------------------------------------

  [174.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

  - Dùng để:

  + Gửi lệnh ON / OFF

  + Chọn chế độ mặc định

  + Hiển thị trạng thái thiết bị

  ▶ Ví dụ ngắn:

  - Checkbox checked → chế độ AUTO bật

  - Bỏ checked → chuyển sang MANUAL

  ------------------------------------------------

  [174.3] PHẠM VI ÁP DỤNG

  - Loại: Attribute

  - Áp cho:

  + <input type="checkbox">

  + <input type="radio">

  - Có cần focus không?

  + KHÔNG cần

  + Trình duyệt tự xử lý trạng thái

  ------------------------------------------------

  [174.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

  - Cú pháp chuẩn:

  <input type="checkbox" checked>

  - Ví dụ đúng:

  <input type="radio" name="mode" checked>

  - Ví dụ SAI:

  <input type="checkbox" checked="false">

  → SAI vì chỉ cần có checked là TRUE

  ------------------------------------------------

  [174.5] GIÁ TRỊ THƯỜNG DÙNG

  - Kiểu giá trị:

  + Boolean attribute (có / không)

  - Ví dụ giá trị hay gặp:

  + checked

  + (không có thuộc tính → false)

  ------------------------------------------------

  [174.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

  1. Trình duyệt parse HTML

  2. Gặp input có checked

  3. Đặt trạng thái input = đã chọn

  4. Người dùng click → trạng thái đổi

  5. JavaScript đọc .checked = true / false

  6. Giao diện hiển thị tick hoặc chấm tròn

  ▶ Mục này giúp DEBUG:

  - Input luôn được chọn → kiểm tra có checked trong HTML không

  ------------------------------------------------

  [174.7] BẢN CHẤT HOẠT ĐỘNG

  - Là: Attribute (Boolean)

  - Ai kích hoạt:

  + Trình duyệt khi load HTML

  + Người dùng khi click

  + JavaScript khi gán .checked

  -----------------------------------------------

  [174.8] LƯU Ý / LỖI HAY GẶP

  - Lỗi 1: Dùng checked cho input text

  → Không có tác dụng

  - Lỗi 2: Nghĩ checked="false" là bỏ chọn

  → SAI hoàn toàn

  - Cách nhớ nhanh:

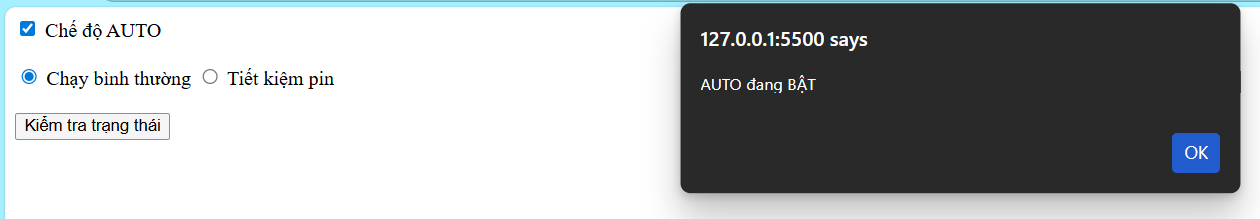
  → Có checked là BẬT, không có là TẮT

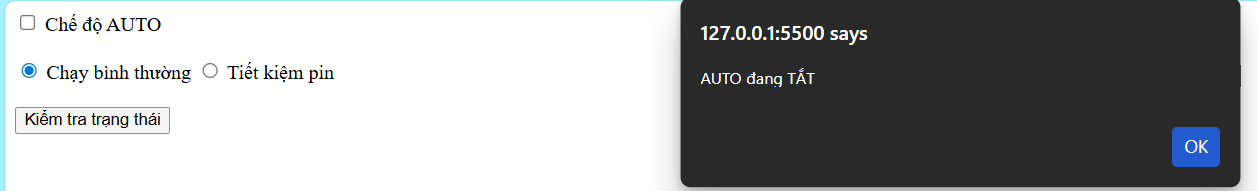
  ------------------------------------------------

  [174.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

  → checked dùng để ĐÁNH DẤU ĐÃ CHỌN khi trang vừa TẢI XONG

174.10.EXAAMBLE





<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ checked</title>

</head>

<body>

  <!-- Ví dụ 1: checkbox bật sẵn -->

  <label>

    <input type="checkbox" checked>

    Chế độ AUTO

  </label>

  <br><br>

  <!-- Ví dụ 2: radio chọn sẵn -->

  <label>

    <input type="radio" name="mode" checked>

    Chạy bình thường

  </label>

  <label>

    <input type="radio" name="mode">

    Tiết kiệm pin

  </label>

  <br><br>

  <!-- Ví dụ 3: đọc trạng thái bằng JavaScript -->

  <button onclick="kiemTra()">Kiểm tra trạng thái</button>

  <script>

    function kiemTra() {

      const cb = document.querySelector('input[type="checkbox"]');

      alert(cb.checked ? 'AUTO đang BẬT' : 'AUTO đang TẮT');

    }

  </script>

</body>

</html>

<!--

###

Trang vừa LOAD (trình duyệt bắt đầu đọc HTML)

###

- Trình duyệt đọc <!DOCTYPE html> → biết đây là HTML5

- Tạo cây DOM từ trên xuống:

  html

   ├─ head (meta, title)

   └─ body

      ├─ label

      │   └─ input type="checkbox" + checked

      ├─ label

      │   └─ input type="radio" name="mode" + checked

      ├─ label

      │   └─ input type="radio" name="mode"

      ├─ button (onclick="kiemTra()")

      └─ script (định nghĩa hàm kiemTra)

- DOM BAN ĐẦU:

  + checkbox: checked = true

  + radio "Chạy bình thường": checked = true

  + radio "Tiết kiệm pin": checked = false

- CSS mặc định của trình duyệt:

  + checkbox có dấu tick

  + radio đầu tiên có chấm tròn

  + radio thứ hai trống

###

<input type="checkbox" checked>

###

- Trình duyệt gặp thuộc tính checked

- checked là Boolean attribute:

  + CÓ mặt → true

- DOM tại node này:

  input.checked = true

- Không có JS can thiệp

- UI hiển thị: checkbox đã được tick sẵn

###

<input type="radio" name="mode" checked>

###

- Hai radio có cùng name="mode" → cùng nhóm

- Trình duyệt:

  + thấy radio này có checked → chọn nó

  + tự động đảm bảo radio còn lại trong cùng name = unchecked

- DOM sau xử lý:

  + radio 1.checked = true

  + radio 2.checked = false

###

<button onclick="kiemTra()">

###

- Trình duyệt gắn EVENT LISTENER:

  + sự kiện: click

  + handler: gọi hàm kiemTra

- Chưa có JS chạy lúc này

- DOM và UI KHÔNG đổi

###

Người dùng CLICK nút "Kiểm tra trạng thái"

###

- Người dùng click chuột

- Sự kiện click phát sinh

- Trình duyệt gọi: kiemTra()

###

function kiemTra() { ... }

###

- JS bắt đầu chạy từ dòng đầu tiên trong hàm

###

const cb = document.querySelector('input[type="checkbox"]');

###

- document: đối tượng đại diện toàn bộ DOM

- dấu . :

  + truy cập method querySelector của document

- querySelector(...) :

  + tìm PHẦN TỬ ĐẦU TIÊN khớp selector

- Selector 'input[type="checkbox"]':

  + chọn checkbox duy nhất trong DOM

- Kết quả:

  + cb trỏ tới node input checkbox

- DOM KHÔNG đổi, chỉ JS đang cầm reference

###

cb.checked

###

- dấu . :

  + truy cập thuộc tính checked của phần tử input

- Giá trị được đọc NGAY THỜI ĐIỂM NÀY

- Nếu:

  + checkbox đang tick → true

  + checkbox bị bỏ tick → false

###

cb.checked ? 'AUTO đang BẬT' : 'AUTO đang TẮT'

###

- Toán tử ? :

  + điều kiện: cb.checked

- Nếu cb.checked === true:

  → chọn chuỗi 'AUTO đang BẬT'

- Nếu cb.checked === false:

  → chọn chuỗi 'AUTO đang TẮT'

- KHÔNG có DOM change

- Chỉ tạo ra 1 giá trị chuỗi trong bộ nhớ JS

###

alert(...)

###

- Trình duyệt nhận lệnh hiển thị hộp thoại

- UI:

  + Popup hiện ra

  + Hiển thị chuỗi đã được chọn bởi ? :

- DOM:

  + KHÔNG thay đổi

- CSS:

  + KHÔNG phản ứng

- ARIA:

  + Screen reader đọc nội dung alert cho người dùng

###

Sau khi đóng alert

###

- JS kết thúc hàm kiemTra

- Không có state nào bị ghi lại

- Checkbox và radio GIỮ NGUYÊN trạng thái trước đó

================================================

TỔNG KẾT VAI TRÒ

================================================

- HTML: khai báo cấu trúc + trạng thái mặc định (checked)

- DOM: lưu trạng thái thật (input.checked true/false)

- CSS: hiển thị tick / chấm tròn dựa trên DOM

- JS: đọc trạng thái DOM, quyết định logic, KHÔNG tự đổi DOM

- ARIA / hệ thống: đọc alert cho người dùng

→ Đây là luồng chạy THỰC TẾ trong đầu trình duyệt, từng bước, không nhảy cóc

-->

========================================

=== PHÂN TÍCH THUỘC TÍNH HTML: readonly ===

========================================

175. Thuộc tính readonly

  → KHÓA việc CHỈNH SỬA nội dung ô nhập nhưng VẪN cho phép đọc giá trị

  -----------------------------------------------

  [175.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

  - Ý 1: Ngăn người dùng sửa nội dung input / textarea

  - Ý 2: Dùng khi chỉ cho XEM, KHÔNG cho SỬA

  ------------------------------------------------

  [175.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

  - Dùng để:

  + Hiển thị trạng thái thiết bị

  + Hiển thị dữ liệu cảm biến

  + Hiển thị thông tin nhưng cấm chỉnh sửa

  ▶ Ví dụ ngắn:

  - ESP gửi nhiệt độ → hiển thị readonly

  - Người dùng chỉ xem → KHÔNG sửa được

  ------------------------------------------------

  [175.3] PHẠM VI ÁP DỤNG

  - Loại: Attribute

  - Áp cho:

  + <input> (text, number, email…)

  + <textarea>

- Có cần focus không?

  + CÓ thể focus

  + Vẫn copy được nội dung

------------------------------------------------

[175.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag readonly>

- Ví dụ đúng:

  <input type="text" value="123" readonly>

- Ví dụ SAI:

  <input type="text" readonly="false">

  → SAI vì readonly là Boolean attribute

------------------------------------------------

[175.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Boolean attribute (có / không)

- Ví dụ giá trị hay gặp:

  + readonly

  + (không có → cho phép sửa)

------------------------------------------------

[175.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt render input

2. Gặp thuộc tính readonly

3. Đặt trạng thái input = KHÔNG chỉnh sửa

4. Người dùng click vào vẫn focus được

5. Gõ phím → trình duyệt CHẶN thay đổi value

6. JavaScript đọc value bình thường

▶ Mục này giúp DEBUG:

- Không sửa được → kiểm tra readonly hay disabled

------------------------------------------------

[175.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute (Boolean)

- Ai kích hoạt:

  + Trình duyệt khi load HTML

  + JavaScript khi gán .readOnly = true

------------------------------------------------

[175.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Nhầm readonly với disabled

  → readonly VẪN submit dữ liệu

- Lỗi 2: Nghĩ readonly thì không focus được

  → SAI

- Cách nhớ nhanh:

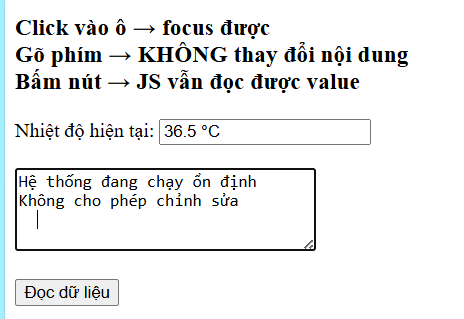
  → readonly = xem được, sửa KHÔNG

------------------------------------------------

[175.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ readonly dùng để KHÓA CHỈNH SỬA khi chỉ cần HIỂN THỊ GIÁ TRỊ

175.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ readonly</title>

</head>

<body>

  <h3>

    Click vào ô → focus được

    <br>

    Gõ phím → KHÔNG thay đổi nội dung

    <br>

    Bấm nút → JS vẫn đọc được value

  </h3>

  <!-- Ví dụ 1: input readonly -->

  <label>

    Nhiệt độ hiện tại:

    <input type="text" value="36.5 °C" readonly>

  </label>

  <br><br>

  <!-- Ví dụ 2: textarea readonly -->

  <textarea readonly rows="4" cols="30">

Hệ thống đang chạy ổn định

Không cho phép chỉnh sửa

  </textarea>

  <br><br>

  <!-- Ví dụ 3: JS đọc giá trị readonly -->

  <button onclick="kiemTra()">Đọc dữ liệu</button>

  <script>

    function kiemTra() {

      const ip = document.querySelector('input');

      alert('Giá trị đang là: ' + ip.value);

    }

  </script>

</body>

</html>

========================================

=== PHÂN TÍCH THUỘC TÍNH HTML: required ===

========================================

176. Thuộc tính required

  → ÉP BUỘC người dùng PHẢI nhập dữ liệu trước khi gửi form

  ------------------------------------------------

  [176.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

  - Ý 1: Chặn việc submit form nếu ô nhập đang RỖNG

  - Ý 2: Kích hoạt kiểm tra khi người dùng bấm nút GỬI

  ------------------------------------------------

  [176.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

  - Dùng để:

  + Ép nhập thông tin quan trọng

  + Tránh gửi lệnh thiếu dữ liệu

  + Đảm bảo hệ thống nhận đủ giá trị

  ▶ Ví dụ ngắn:

  - Chưa nhập mật khẩu → KHÔNG cho gửi

  - Nhập xong → cho gửi lệnh tới ESP

  ------------------------------------------------

  [176.3] PHẠM VI ÁP DỤNG

  - Loại: Attribute

  - Áp cho:

  + <input>

  + <textarea>

  + <select>

- Có cần focus không?

  + KHÔNG cần

  + Trình duyệt tự focus vào ô bị thiếu

------------------------------------------------

[176.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag required>

- Ví dụ đúng:

  <input type="text" required>

- Ví dụ SAI:

  <input type="text" required="false">

  → SAI vì required là Boolean attribute

------------------------------------------------

[176.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Boolean attribute (có / không)

- Ví dụ giá trị hay gặp:

  + required

  + (không có → không bắt buộc)

------------------------------------------------

[176.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng bấm nút submit

2. Trình duyệt quét các input có required

3. Phát hiện input value = rỗng

4. Trình duyệt CHẶN submit

5. Focus vào ô lỗi + hiện cảnh báo

6. Khi nhập đủ → cho submit tiếp

▶ Mục này giúp DEBUG:

- Form không gửi được → kiểm tra required

------------------------------------------------

[176.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute (HTML Validation)

- Ai kích hoạt:

  + Trình duyệt khi submit form

- JavaScript:

  + KHÔNG cần JS vẫn hoạt động

  + JS chỉ hỗ trợ kiểm tra thêm

------------------------------------------------

[176.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Nghĩ required kiểm tra nội dung đúng/sai

  → SAI, nó chỉ kiểm tra CÓ hay KHÔNG

- Lỗi 2: Dùng required ngoài form

  → Không có tác dụng submit

- Cách nhớ nhanh:

  → required = chưa nhập thì KHÔNG ĐƯỢC ĐI TIẾP

------------------------------------------------

[176.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ required dùng để BẮT BUỘC NHẬP khi người dùng GỬI FORM

176.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test required</title>

</head>

<body>

  <h3>🔐 Gửi lệnh điều khiển</h3>

  <form>

    <!-- Ô bắt buộc nhập -->

    <label>

      Mã thiết bị:

      <input type="text" required>

    </label>

    <br><br>

    <!-- Ô KHÔNG bắt buộc -->

    <label>

      Ghi chú:

      <input type="text">

    </label>

    <br><br>

    <button type="submit">GỬI LỆNH</button>

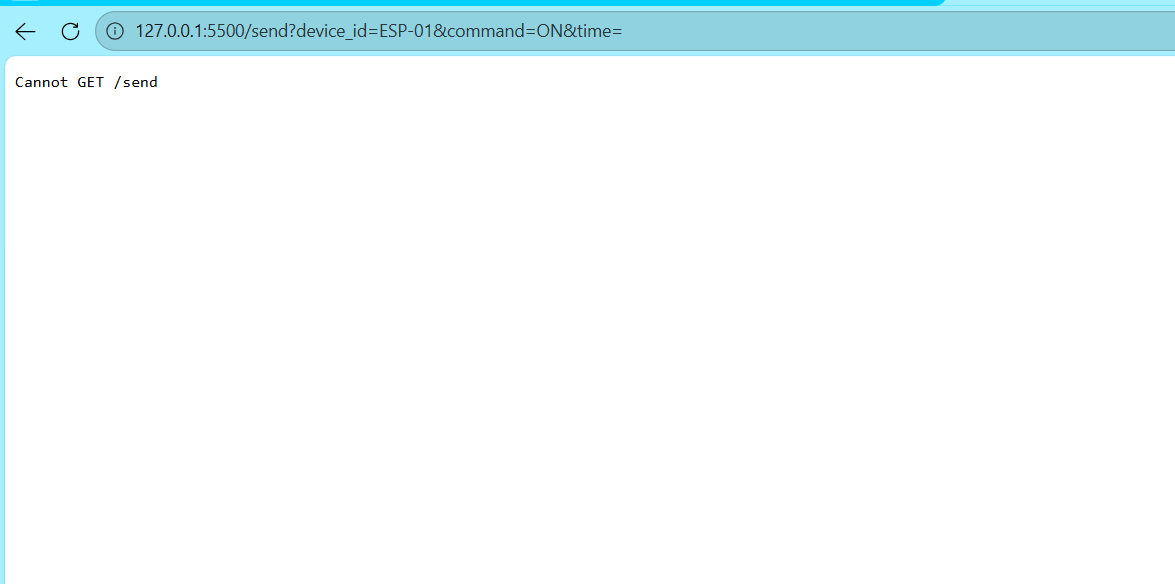
  </form>

</body>

</html>

176.11.EXAMBLE





<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP Control Panel</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      background: #111;

      color: #eee;

      padding: 20px;

    }

    .box {

      border: 1px solid #444;

      padding: 16px;

      width: 300px;

    }

    label {

      display: block;

      margin-bottom: 10px;

    }

    input,

    select,

    button {

      width: 100%;

      padding: 6px;

      margin-top: 4px;

    }

    button {

      background: #0f0;

      border: none;

      cursor: pointer;

    }

  </style>

</head>

<body>

  <h2>📡 ESP CONTROL PANEL</h2>

  <form action="/send" method="GET" class="box">

    <!-- BẮT BUỘC: ESP cần biết gửi cho thiết bị nào -->

    <label>

      ID thiết bị (ESP):

      <input type="text" name="device\_id" placeholder="ESP\_01" required>

    </label>

    <!-- BẮT BUỘC: nếu không chọn thì ESP không biết làm gì -->

    <label>

      Lệnh điều khiển:

      <select name="command" required>

        <option value="">-- Chọn lệnh --</option>

        <option value="ON">BẬT ĐỘNG CƠ</option>

        <option value="OFF">TẮT ĐỘNG CƠ</option>

      </select>

    </label>

    <!-- KHÔNG bắt buộc -->

    <label>

      Thời gian chạy (giây):

      <input type="number" name="time" placeholder="Ví dụ: 10">

    </label>

    <button type="submit">🚀 GỬI LỆNH TỚI ESP</button>

  </form>

</body>

</html>

<!--

###

Trang vừa LOAD (trình duyệt bắt đầu xử lý từ trên xuống)

###

- Trình duyệt đọc <!DOCTYPE html> → xác định chuẩn HTML5

- Bắt đầu dựng cây DOM:

html

├─ head

│ ├─ meta charset

│ ├─ title

│ └─ style

└─ body

├─ h2

└─ form.box

├─ label → input[text][required]

├─ label → select[required]

│ ├─ option value=""

│ ├─ option value="ON"

│ └─ option value="OFF"

├─ label → input[number]

└─ button[type=submit]

- DOM BAN ĐẦU:

+ input device\_id: value = "" (rỗng), required = true

+ select command: value = "" (option đầu), required = true

+ input time: value = "" (rỗng), KHÔNG required

###

<style> trong head

###

- CSS đọc cây DOM vừa tạo

- Áp style:

+ body nền đen, chữ sáng

+ form.box có viền, padding, width

+ input / select / button full width

- DOM KHÔNG đổi

- Chỉ có CÁCH HIỂN THỊ thay đổi

###

<form action="/send" method="GET" class="box">

###

- Trình duyệt đánh dấu đây là FORM

- action="/send":

+ nơi sẽ gửi request khi submit THÀNH CÔNG

- method="GET":

+ dữ liệu sẽ nằm trên URL

- class="box":

+ CSS đã áp style tương ứng

###

<input type="text" name="device\_id" required>

###

- required được trình duyệt ghi nhớ:

+ đây là ô BẮT BUỘC

- DOM trạng thái:

+ input.required = true

+ input.value = ""

- Chưa có kiểm tra xảy ra ở giai đoạn LOAD

###

<select name="command" required>

###

- required áp cho select

- option value="" đang được chọn mặc định

- DOM trạng thái:

+ select.value = ""

+ select.required = true

###

<button type="submit">

###

- Trình duyệt gắn hành vi:

+ click → kích hoạt submit form

- Chưa submit lúc này

###

Người dùng BẤM nút 🚀 GỬI LỆNH TỚI ESP

###

- Sự kiện submit chuẩn bị xảy ra

- Trình duyệt CHƯA gửi request ngay

- Bước KIỂM TRA (HTML Validation) được kích hoạt

###

Trình duyệt QUÉT các field required

###

- Kiểm tra input[name="device\_id"]:

+ value === "" → FALSE (không hợp lệ)

- Kiểm tra select[name="command"]:

+ value === "" → FALSE (không hợp lệ)

- Hai điều kiện sai → submit BỊ CHẶN

###

Hành vi CHẶN SUBMIT

###

- DOM trước:

+ form chưa gửi

- JS:

+ KHÔNG có JS chạy

- Trình duyệt:

+ tự focus vào field lỗi ĐẦU TIÊN

+ hiển thị tooltip cảnh báo

- CSS:

+ vẽ viền đỏ mặc định (UA stylesheet)

- ARIA / Screen reader:

+ thông báo ô bắt buộc chưa được điền

###

Người dùng NHẬP ID thiết bị + CHỌN lệnh

###

- input device\_id.value = "ESP\_01"

- select command.value = "ON"

- DOM thay đổi:

+ JS không can thiệp

+ người dùng là tác nhân thay đổi

###

Người dùng BẤM SUBMIT lần nữa

###

- Trình duyệt lại quét required

- input.value !== ""

- select.value !== ""

- TẤT CẢ ĐIỀU KIỆN ĐÚNG

###

Submit ĐƯỢC PHÉP xảy ra

###

- Trình duyệt tạo URL:

/send?device\_id=ESP\_01&command=ON&time=

- method GET:

+ dữ liệu gắn lên query string

- DOM trang hiện tại bị unload

- Trình duyệt điều hướng sang endpoint /send

###

ESP / Server nhận request

###

- Nhận đủ tham số:

+ device\_id

+ command

- Không có tham số bắt buộc nào bị thiếu

- ESP xử lý lệnh an toàn

================================================

TỔNG KẾT VAI TRÒ

================================================

- HTML: định nghĩa cấu trúc + required

- DOM: giữ trạng thái value thật của từng field

- CSS: phản ứng khi field lỗi / hợp lệ

- JS: KHÔNG tham gia (0 dòng)

- Trình duyệt: là thằng kiểm tra required và chặn submit

- ARIA: giúp công cụ đọc màn hình hiểu ô nào sai

→ Đây là LUỒNG CHẠY THỰC TẾ 100% trong đầu trình duyệt, từng bước, đúng kiểu debug

-->

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

177. Thuộc tính multiple

  → Cho phép phần tử NHẬN / CHỌN NHIỀU GIÁ TRỊ cùng lúc

  ------------------------------------------------

  177.2. [1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

  - Ý 1: Cho người dùng chọn NHIỀU option / file thay vì chỉ 1

  - Ý 2: Dùng khi cần gửi DANH SÁCH giá trị trong 1 lần thao tác

  ------------------------------------------------

  177.3. [2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

  - Dùng để:

  + Gửi nhiều lệnh / nhiều lựa chọn

  + Chọn nhiều thiết bị / cảm biến

  + Upload nhiều file cùng lúc

  Ví dụ ngắn:

  - Chọn 3 cảm biến → gửi ["temp","gas","light"]

  - Chọn 5 ảnh → upload 1 lần

  -----------------------------------------------

  177.4. [3] PHẠM VI ÁP DỤNG

  - Global: ❌

  - Event-only: ❌

  - ARIA-only: ❌

  - Áp cho:

  + <select>

    + <input type="file">

    - Có cần focus không?

    + Có (phải click / focus mới chọn được)

    ------------------------------------------------

    177.5. [4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

    - Cú pháp chuẩn:

    <tag multiple>

      - Ví dụ đúng:

      <select multiple>

        <input type="file" multiple>

        - Ví dụ SAI:

        <div multiple> → div KHÔNG hỗ trợ

          <input type="text" multiple> → type này KHÔNG dùng

          ------------------------------------------------

          177.6. [5] GIÁ TRỊ THƯỜNG DÙNG

          - Kiểu giá trị:

          + Boolean attribute (CÓ là bật, KHÔNG có là tắt)

          - Ví dụ hay gặp:

          + multiple

          + multiple="multiple" (ít dùng, vẫn hợp lệ)

          ------------------------------------------------

          177.7. [6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

          1. Người dùng click / giữ Ctrl / Shift để chọn nhiều

          2. Thuộc tính multiple cho phép giữ nhiều giá trị

          3. JavaScript nhận:

          - select: danh sách option được chọn

          - file: danh sách File

          4. DOM lưu danh sách giá trị (array-like)

          5. Giao diện giữ trạng thái nhiều mục đang chọn

          Dùng để debug khi chỉ nhận được 1 giá trị

          ------------------------------------------------

          177.8. [7] BẢN CHẤT HOẠT ĐỘNG

          - Là: Attribute (boolean)

          - Ai kích hoạt:

          + Trình duyệt (dựa vào việc có attribute hay không)

          + Người dùng (click / chọn)

          multiple KHÔNG tự chạy logic, chỉ MỞ KHẢ NĂNG chọn nhiều

          ------------------------------------------------

          177.9. [8] LƯU Ý / LỖI HAY GẶP

          - Lỗi 1: Dùng multiple cho thẻ KHÔNG hỗ trợ

          - Lỗi 2: Quên xử lý danh sách → chỉ lấy 1 giá trị

          - Cách nhớ nhanh:

          + multiple = LUÔN sinh RA NHIỀU giá trị

          ------------------------------------------------

          177.10. [9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

          → Thuộc tính multiple dùng để CHỌN NHIỀU GIÁ TRỊ khi người dùng thao tác chọn

177.11.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Demo thuộc tính multiple</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      padding: 20px;

    }

    select {

      width: 200px;

      height: 90px;

    }

    button {

      margin-top: 10px;

      padding: 6px 12px;

    }

    pre {

      margin-top: 12px;

      background: #111;

      color: #0f0;

      padding: 10px;

    }

  </style>

</head>

<body>

  <h3>Ví dụ 1: select multiple</h3>

  <select id="sensorSelect" multiple>

    <option value="temp">Nhiệt độ</option>

    <option value="gas">Khí gas</option>

    <option value="light">Ánh sáng</option>

    <option value="humidity">Độ ẩm</option>

  </select>

  <br>

  <button onclick="sendSensors()">Gửi lựa chọn</button>

  <pre id="output1">Chưa gửi dữ liệu</pre>

  <hr>

  <h3>Ví dụ 2: input file multiple</h3>

  <input type="file" id="fileInput" multiple>

  <br>

  <button onclick="uploadFiles()">Upload</button>

  <pre id="output2">Chưa chọn file</pre>

  <script>

    function sendSensors() {

      const select = document.getElementById("sensorSelect");

      const result = [];

      for (let option of select.options) {

        if (option.selected) {

          result.push(option.value);

        }

      }

      document.getElementById("output1").textContent =

        "Danh sách cảm biến đã chọn:\n" + JSON.stringify(result);

    }

    function uploadFiles() {

      const files = document.getElementById("fileInput").files;

      const names = [];

      for (let file of files) {

        names.push(file.name);

      }

      document.getElementById("output2").textContent =

        "File đã chọn:\n" + names.join("\n");

    }

  </script>

</body>

</html>

###

Trang vừa LOAD (trình duyệt bắt đầu đọc HTML)

###

- Trình duyệt đọc <!DOCTYPE html> → xác định HTML5

- Bắt đầu tạo DOM tree:

  html

   ├─ head (meta, title, style)

   └─ body

       ├─ h3

       ├─ select#sensorSelect [multiple]

       │   ├─ option value="temp"

       │   ├─ option value="gas"

       │   ├─ option value="light"

       │   └─ option value="humidity"

       ├─ button (onclick="sendSensors()")

       ├─ pre#output1

       ├─ hr

       ├─ h3

       ├─ input#fileInput type="file" [multiple]

       ├─ button (onclick="uploadFiles()")

       ├─ pre#output2

       └─ script (2 hàm JS)

- DOM BAN ĐẦU:

  + select.multiple = true

  + KHÔNG option nào selected

  + input.files = FileList rỗng

  + output1.textContent = "Chưa gửi dữ liệu"

  + output2.textContent = "Chưa chọn file"

###

<style> được xử lý

###

- CSS đọc DOM

- Áp style:

  + select có height lớn → hiển thị nhiều dòng

  + pre nền đen chữ xanh

- DOM KHÔNG đổi

- Chỉ thay đổi CÁCH HIỂN THỊ

###

<select id="sensorSelect" multiple>

###

- Thuộc tính multiple được trình duyệt ghi nhận

- multiple = true nghĩa là:

  + cho phép nhiều option cùng selected

- DOM:

  + select.multiple === true

- UI:

  + người dùng có thể Ctrl + Click hoặc Shift + Click

###

Người dùng CHỌN NHIỀU cảm biến

###

- Ví dụ người dùng chọn:

  + "Nhiệt độ"

  + "Độ ẩm"

- DOM trước:

  + option.selected = false cho tất cả

- Người dùng là tác nhân

- DOM sau:

  + option[value="temp"].selected = true

  + option[value="humidity"].selected = true

- CSS:

  + highlight các option được chọn

###

<button onclick="sendSensors()">

###

- Người dùng CLICK nút

- Sự kiện click phát sinh

- Trình duyệt gọi hàm sendSensors()

###

function sendSensors()

###

- JS bắt đầu chạy

###

const select = document.getElementById("sensorSelect");

###

- document: đại diện toàn bộ DOM

- dấu . :

  + truy cập method getElementById của document

- select trỏ tới node <select>

- DOM KHÔNG đổi

###

const result = [];

###

- JS tạo mảng rỗng trong bộ nhớ

- Chưa liên quan DOM

###

for (let option of select.options)

###

- select.options:

  + dấu . truy cập thuộc tính options của select

  + trả về HTMLCollection các option

- JS lặp từng option theo THỨ TỰ DOM

###

if (option.selected)

###

- dấu . :

  + truy cập thuộc tính selected của option

- Với mỗi option:

  + JS đọc giá trị true / false tại THỜI ĐIỂM NÀY

###

result.push(option.value);

###

- option.value:

  + dấu . truy cập thuộc tính value

- Nếu selected === true:

  + push value vào mảng result

- Ví dụ result sau vòng lặp:

  ["temp", "humidity"]

###

document.getElementById("output1").textContent = ...

###

- document.getElementById:

  + lấy node <pre id="output1">

- dấu .textContent:

  + ghi nội dung TEXT vào node

- DOM trước:

  + "Chưa gửi dữ liệu"

- JS là thằng đổi

- DOM sau:

  + hiển thị danh sách cảm biến đã chọn

- CSS:

  + pre tự render lại nội dung

- Screen reader:

  + đọc nội dung mới trong <pre>

###

<input type="file" id="fileInput" multiple>

###

- multiple = true:

  + cho phép chọn NHIỀU FILE

- DOM:

  + input.files là FileList (0..n file)

###

Người dùng CHỌN NHIỀU FILE

###

- Ví dụ chọn 3 file

- DOM trước:

  + input.files.length = 0

- Người dùng chọn file

- DOM sau:

  + input.files.length = 3

  + mỗi phần tử là File object

###

<button onclick="uploadFiles()">

###

- Người dùng click

- Sự kiện click phát sinh

- Trình duyệt gọi uploadFiles()

###

function uploadFiles()

###

- JS bắt đầu chạy

###

const files = document.getElementById("fileInput").files;

###

- document.getElementById:

  + lấy node input file

- dấu .files:

  + truy cập FileList hiện tại

- files là tập snapshot của file đã chọn

###

for (let file of files)

###

- JS lặp qua từng File object

- Mỗi file chứa:

  + name

  + size

  + type

###

names.push(file.name);

###

- file.name:

  + dấu . truy cập tên file

- Đẩy tên file vào mảng names

###

document.getElementById("output2").textContent = ...

###

- DOM trước:

  + "Chưa chọn file"

- JS ghi textContent mới

- DOM sau:

  + hiển thị danh sách tên file

- CSS:

  + <pre> cập nhật nội dung

- Screen reader:

  + đọc danh sách file đã chọn

================================================

TỔNG KẾT VAI TRÒ

================================================

- HTML: khai báo multiple cho select và file

- DOM: lưu trạng thái selected / files

- CSS: hiển thị trạng thái lựa chọn

- JS: đọc DOM, lọc dữ liệu, ghi kết quả

- ARIA / Screen reader: đọc nội dung cập nhật

- multiple: cho phép 1 hành động → NHIỀU GIÁ TRỊ cùng lúc

→ Đây là LUỒNG CHẠY THỰC TẾ trong đầu trình duyệt, debug từng bước, không nhảy cóc

-->

========================================

=== PHÂN TÍCH THUỘC TÍNH HTML: pattern ===

========================================

178. Thuộc tính pattern

  → ÉP GIÁ TRỊ NHẬP phải KHỚP MỘT KHUÔN MẪU (regex) trước khi cho submit

  ------------------------------------------------

  [178.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

  - Ý 1: Kiểm tra ĐỊNH DẠNG dữ liệu người dùng nhập

  - Ý 2: Chỉ kích hoạt khi submit form hoặc gọi validation

  ------------------------------------------------

  [178.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

  - Dùng để:

  + Ép đúng format lệnh gửi xuống ESP

  + Tránh dữ liệu rác / sai chuẩn

  + Kiểm tra mã, ID, password

  ▶ Ví dụ ngắn:

  - Nhập ESP\_01 → cho gửi

  - Nhập esp01 → CHẶN

  ------------------------------------------------

  [178.3] PHẠM VI ÁP DỤNG

  - Loại: Attribute (Validation)

  - Áp cho:

  + <input type="text">

  + <input type="search">

  + <input type="tel">

  - Có cần focus không?

  + KHÔNG

  + Trình duyệt tự focus khi sai

  ------------------------------------------------

  [178.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

  - Cú pháp chuẩn:

  <input pattern="biểu\_thức">

  - Ví dụ đúng:

  <input pattern="[A-Z]{3}\_[0-9]{2}">

  - Ví dụ SAI:

  <input pattern="/[A-Z]{3}/">

  → SAI vì KHÔNG dùng dấu / trong HTML pattern

  ------------------------------------------------

  [178.5] GIÁ TRỊ THƯỜNG DÙNG

  - Kiểu giá trị:

  + Chuỗi (Regex)

  - Ví dụ hay gặp:

  + [0-9]+

  + [A-Z]{2,5}

  + ESP\_[0-9]{2}

  ------------------------------------------------

  [178.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

  1. Người dùng nhập dữ liệu

  2. Người dùng bấm submit

  3. Trình duyệt đọc pattern

  4. So khớp value với regex

  5. Không khớp → CHẶN submit

  6. Khớp → cho gửi tiếp

  Mục này giúp DEBUG:

  - Submit bị chặn → nghi pattern sai

  ------------------------------------------------

  [178.7] BẢN CHẤT HOẠT ĐỘNG

  - Là: Attribute (HTML Validation)

  - Ai kích hoạt:

  + Trình duyệt khi submit

  - JavaScript:

  + KHÔNG cần JS vẫn hoạt động

  ------------------------------------------------

  [178.8] LƯU Ý / LỖI HAY GẶP

  - Lỗi 1: Quên viết regex KHỚP TOÀN BỘ chuỗi

  → pattern mặc định match TOÀN value

  - Lỗi 2: Dùng pattern cho type="number"

  → KHÔNG hoạt động

  - Cách nhớ nhanh:

  → pattern = KHUÔN ĐÚC DỮ LIỆU

  ------------------------------------------------

  [178.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

  → pattern dùng để CHẶN SUBMIT khi dữ liệu KHÔNG ĐÚNG ĐỊNH DẠNG

178.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test pattern - ESP</title>

</head>

<body>

  <h3>📡 Gửi lệnh tới ESP</h3>

  <form>

    <!-- ID ESP phải đúng định dạng: ESP\_01, ESP\_12, ESP\_99 -->

    <label>

      ID thiết bị:

      <input type="text" name="device\_id" placeholder="ESP\_01" pattern="ESP\_[0-9]{2}" required>

    </label>

    <br><br>

    <!-- Mã PIN điều khiển: đúng 4 chữ số -->

    <label>

      Mã PIN:

      <input type="text" name="pin" placeholder="1234" pattern="[0-9]{4}" required>

    </label>

    <br><br>

    <button type="submit">🚀 GỬI LỆNH</button>

  </form>

</body>

</html>

178.11.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test pattern cơ bản</title>

</head>

<body>

  <h3>📝 Đăng ký tài khoản</h3>

  <form>

    <!-- Username: chỉ chữ thường + số, dài 5–10 ký tự -->

    <label>

      Tên đăng nhập:

      <input type="text" placeholder="vd: user123" pattern="[a-z0-9]{5,10}" required>

    </label>

    <br><br>

    <!-- Mật khẩu: ít nhất 6 ký tự, có số -->

    <label>

      Mật khẩu:

      <input type="password" placeholder="ít nhất 6 ký tự" pattern="(?=.\*[0-9]).{6,}" required>

    </label>

    <br><br>

    <button type="submit">Đăng ký</button>

  </form>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[179]. Thuộc tính form

  → Liên kết một control (input / button / select) với MỘT form cụ thể dù nó KHÔNG nằm bên trong form đó

  ------------------------------------------------

  [179.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

  - Ý 1: Gán input / button “thuộc về” form nào sẽ submit

  - Ý 2: Dùng khi giao diện bị tách layout, control nằm ngoài <form>

    ------------------------------------------------

    [179.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

    - Dùng để:

    + Gửi lệnh điều khiển ESP từ nút đặt ở panel khác

    + Gom dữ liệu từ nhiều khu vực UI về chung 1 request

    + Thiết kế dashboard điều khiển (nút ở header, form ở body)

    Ví dụ ngắn:

    - Nhấn nút GỬI ở thanh trên → dữ liệu form phía dưới được submit

    - Input đặt ngoài form → vẫn gửi lệnh START tới ESP

    ------------------------------------------------

    [179.3] PHẠM VI ÁP DỤNG

    - Loại: Global attribute (nhưng chỉ CÓ TÁC DỤNG với form control)

    - Áp cho:

    + input

    + button

    + select

    + textarea

    - Không cần focus

    - KHÔNG áp cho div / span / p

    ------------------------------------------------

    [179.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

    - Cú pháp chuẩn:

    <tag form="form\_id">

      - Ví dụ ĐÚNG:

      <form id="ctrlForm"></form>

      <button type="submit" form="ctrlForm">Gửi</button>

      - Ví dụ SAI:

      <button form="ctrlForm"></div>

        → sai vì button chưa đóng đúng thẻ

        ------------------------------------------------

        [179.5] GIÁ TRỊ THƯỜNG DÙNG

        - Kiểu giá trị: Chuỗi

        - Giá trị là:

        + id của thẻ <form>

          - Ví dụ:

          + "loginForm"

          + "sendCommand"

          + "espControl"

          ------------------------------------------------

          [179.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

          1. Người dùng click button

          2. Trình duyệt đọc thuộc tính form="ctrlForm"

          3. Trình duyệt tìm <form id="ctrlForm">

            4. Thu thập TOÀN BỘ control gắn với form đó

            5. Thực hiện submit (GET / POST)

            6. Màn hình chuyển trang hoặc JS bắt sự kiện submit

            DÙ control nằm NGOÀI form, vẫn bị gom dữ liệu

            ------------------------------------------------

            [179.7] BẢN CHẤT HOẠT ĐỘNG

            - Là: Attribute

            - Ai kích hoạt:

            + Người dùng (click)

            + Trình duyệt (tự ánh xạ control → form)

            - JavaScript KHÔNG cần can thiệp

            form là “quan hệ logic”, không phải “quan hệ DOM”

            ------------------------------------------------

            [179.8] LƯU Ý / LỖI HAY GẶP

            - Lỗi 1: form="abc" nhưng không tồn tại id="abc"

            → click không làm gì

            - Lỗi 2: quên type="submit" cho button

            → button chỉ là nút thường

            - Cách nhớ nhanh:

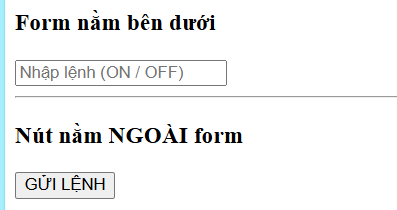
            → form luôn trỏ tới ID, KHÔNG phải name

            ------------------------------------------------

            [179.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

            → “Thuộc tính form dùng để GÁN control vào MỘT form khi click xảy ra”

179.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test thuộc tính form</title>

</head>

<body>

  <h3>Form nằm bên dưới</h3>

  <!-- FORM -->

  <form id="sendForm" action="#" method="GET">

    <input type="text" name="cmd" placeholder="Nhập lệnh (ON / OFF)">

  </form>

  <hr>

  <h3>Nút nằm NGOÀI form</h3>

  <!-- BUTTON NGOÀI FORM -->

  <button type="submit" form="sendForm">

    GỬI LỆNH

  </button>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[180]. Thuộc tính min và max

→ Giới hạn GIÁ TRỊ TỐI THIỂU / TỐI ĐA mà người dùng được phép nhập

------------------------------------------------

[180.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Chặn không cho nhập giá trị nhỏ hơn min hoặc lớn hơn max

- Ý 2: Kích hoạt khi người dùng nhập dữ liệu hoặc submit form

Không hợp lệ → trình duyệt KHÔNG cho gửi

------------------------------------------------

[180.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Giới hạn thời gian chạy động cơ ESP

  + Giới hạn tốc độ, PWM, công suất

  + Tránh gửi lệnh vượt ngưỡng an toàn

Ví dụ ngắn:

- Nhập thời gian chạy: 1 → 60 giây

- Nhập 0 hoặc 100 → BỊ CHẶN

------------------------------------------------

[180.3] PHẠM VI ÁP DỤNG

- Loại: Attribute

- Áp cho:

  + <input type="number">

  + <input type="range">

  + <input type="date">

  + <input type="time">

- Không áp cho type="text"

- Không cần focus đặc biệt

------------------------------------------------

[180.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <input type="number" min="giá\_trị" max="giá\_trị">

- Ví dụ đúng:

  <input type="number" min="1" max="60">

- Ví dụ SAI:

  <input type="text" min="1" max="60">

  → min / max KHÔNG hoạt động

------------------------------------------------

[180.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Số nguyên / số thực

  + Ngày (YYYY-MM-DD)

  + Giờ (HH:MM)

- Ví dụ hay gặp:

  + min="0" max="255"  (PWM ESP)

  + min="1" max="100"

  + min="08:00" max="18:00"

------------------------------------------------

[180.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng nhập giá trị

2. Trình duyệt đọc min và max trong DOM

3. So sánh:

   - value >= min ?

   - value <= max ?

4. Nếu SAI → chặn submit + hiện cảnh báo

5. Nếu ĐÚNG → cho phép gửi dữ liệu

JS chỉ chạy khi dữ liệu HỢP LỆ

------------------------------------------------

[180.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute (HTML thuần)

- Ai kích hoạt:

  + Người dùng nhập

  + Trình duyệt tự kiểm tra

- JavaScript KHÔNG phải là thằng kiểm tra chính

------------------------------------------------

[180.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Dùng min/max với type="text"

- Lỗi 2: Quên required → vẫn để trống được

- Cách nhớ nhanh:

  → min/max chỉ dành cho GIÁ TRỊ CÓ THỂ ĐO ĐƯỢC

------------------------------------------------

[180.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “min / max dùng để GIỚI HẠN giá trị nhập khi người dùng gửi form”

180.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test min / max</title>

</head>

<body>

  <h3>⏱ Nhập thời gian chạy động cơ</h3>

  <form>

    <!-- GIỚI HẠN: 1 đến 60 giây -->

    <label>

      Thời gian (giây):

      <input type="number" min="1" max="60" placeholder="1 - 60" required>

    </label>

    <br><br>

    <button type="submit">GỬI LỆNH</button>

  </form>

</body>

</html>

180.11.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP Motor Control</title>

  <style>

    body {

      background: #0e0e0e;

      color: #e0e0e0;

      font-family: Arial;

      padding: 20px;

    }

    .panel {

      width: 320px;

      border: 1px solid #333;

      padding: 16px;

    }

    label {

      display: block;

      margin-bottom: 12px;

    }

    input,

    button {

      width: 100%;

      padding: 6px;

      margin-top: 4px;

    }

    button {

      background: #00ff88;

      border: none;

      cursor: pointer;

      font-weight: bold;

    }

    pre {

      background: #111;

      color: #0f0;

      padding: 10px;

      margin-top: 12px;

    }

  </style>

</head>

<body>

  <h2>⚙️ ESP MOTOR CONTROL</h2>

  <form class="panel">

    <!-- min / max cho PWM -->

    <label>

      Tốc độ motor (PWM 0–255):

      <input type="number" name="speed" min="0" max="255" value="120" required>

    </label>

    <!-- min / max cho thời gian -->

    <label>

      Thời gian chạy (giây):

      <input type="number" name="time" min="1" max="60" value="10" required>

    </label>

    <button type="submit">🚀 GỬI LỆNH</button>

  </form>

  <pre>

ESP sẽ nhận:

- speed: 0 → 255 (PWM)

- time : 1 → 60 (giây)

  </pre>

</body>

</html>

========================================

=== PHÂN TÍCH THUỘC TÍNH HTML: step ===

========================================

[181]. Thuộc tính step

  → Quy định MỖI LẦN tăng/giảm giá trị là bao nhiêu khi nhập số.

  ------------------------------------------------

  [181.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

  - Ý 1: Kiểm soát BƯỚC NHẢY giá trị số (tăng / giảm theo nấc)

  - Ý 2: Kích hoạt khi người dùng bấm mũi tên, cuộn chuột, hoặc nhập số

  ------------------------------------------------

  [181.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

  - Dùng để:

  + Điều chỉnh tốc độ (xe / robot)

  + Chỉnh mức công suất ESP

  + Chọn số lượng chính xác

  Ví dụ ngắn:

  - Nhấn ▲ → tốc độ +5

  - Nhấn ▼ → tốc độ -5

  ------------------------------------------------

  [181.3] PHẠM VI ÁP DỤNG

  - Loại: Input-only

  - Áp cho:

  + <input type="number">

  + <input type="range">

  - Có cần focus không?

  + CÓ (muốn dùng phím ↑ ↓ thì input phải focus)

  ------------------------------------------------

  [181.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

  - Cú pháp chuẩn:

  <input type="number" step="giá\_trị">

  - Ví dụ đúng:

  <input type="number" step="5">

  - Ví dụ SAI:

  <input step="5">

  → Thiếu type="number" hoặc "range" nên step KHÔNG tác dụng

  ------------------------------------------------

  [181.5] GIÁ TRỊ THƯỜNG DÙNG

  - Kiểu giá trị:

  + Số (number)

  - Giá trị hay gặp:

  + 1 → tăng từng đơn vị

  + 0.1 → số thập phân

  + 5, 10 → tăng theo nấc lớn

  + any → cho phép mọi giá trị (bỏ kiểm tra bước)

  ------------------------------------------------

  [181.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

  1. Người dùng bấm ▲ ▼ hoặc cuộn chuột

  2. Trình duyệt kiểm tra thuộc tính step

  3. Giá trị mới = giá trị cũ ± step

  4. JS đọc value mới từ input

  5. Giao diện / trạng thái thiết bị thay đổi

  → Dùng để DEBUG khi số nhảy sai bước

  ------------------------------------------------

  [181.7] BẢN CHẤT HOẠT ĐỘNG

  - Là: Attribute

  - Ai kích hoạt:

  + Người dùng (chuột / phím)

  + Trình duyệt tự xử lý bước nhảy

  - JavaScript chỉ LẤY KẾT QUẢ

  ------------------------------------------------

  [181.8] LƯU Ý / LỖI HAY GẶP

  - Lỗi 1: step không chia hết cho (value - min)

  → Trình duyệt báo input không hợp lệ

  - Lỗi 2: Quên min nhưng dùng step lẻ (0.1)

  → Sai lệch số do floating

  - Cách nhớ nhanh:

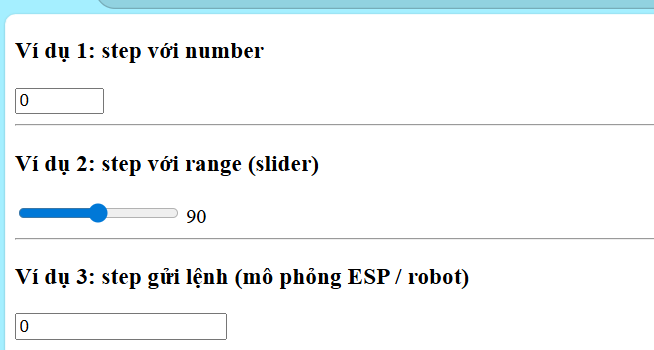
  → step luôn đi CHUNG với min

  ------------------------------------------------

  [181.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

  → step dùng để quy định MỖI LẦN tăng/giảm giá trị bao nhiêu khi người dùng thay đổi số

181.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ step</title>

</head>

<body>

  <h3>Ví dụ 1: step với number</h3>

  <!-- Mỗi lần tăng/giảm là +5 -->

  <input type="number" min="0" max="100" step="5" value="0">

  <hr>

  <h3>Ví dụ 2: step với range (slider)</h3>

  <!-- Kéo mỗi nấc là +10 -->

  <input type="range" min="0" max="180" step="10" value="90" oninput="output.value = this.value">

  <output id="output">90</output>

  <hr>

  <h3>Ví dụ 3: step gửi lệnh (mô phỏng ESP / robot)</h3>

  <input type="number" step="10" value="0" oninput="guiLenh(this.value)">

  <script>

    function guiLenh(giaTri) {

      console.log("Gửi lệnh tốc độ:", giaTri);

    }

  </script>

</body>

</html>

########################################

### GIAI ĐOẠN 1: TRANG BẮT ĐẦU LOAD

########################################

###

<!DOCTYPE html>

###

→ Trình duyệt đọc dòng này và chuyển sang CHẾ ĐỘ HTML5

→ Quyết định cách parse DOM, cách CSS và JS sẽ hoạt động

→ Chưa có DOM cụ thể, chỉ là khai báo tiêu chuẩn

###

<html lang="vi">

###

→ Trình duyệt tạo NODE GỐC <html>

→ Thuộc tính lang="vi" được gắn vào node html

→ ARIA / screen reader ghi nhận: ngôn ngữ chính là tiếng Việt

###

<head> ... </head>

###

→ Trình duyệt tạo nhánh <head> trong DOM

→ Đọc <meta charset="UTF-8"> để biết cách giải mã chữ

→ <title> được lưu, nhưng CHƯA hiển thị lên màn hình

########################################

### GIAI ĐOẠN 2: DOM BODY ĐƯỢC TẠO

########################################

###

<body>

###

→ Trình duyệt tạo node <body>

→ Từ đây trở đi, mọi node sẽ có KHẢ NĂNG HIỂN THỊ

###

<h3>Ví dụ 1: step với number</h3>

###

→ DOM thêm node <h3>

→ CSS mặc định của browser áp font-size lớn hơn chữ thường

###

<input type="number" min="0" max="100" step="5" value="0">

###

→ Trình duyệt tạo 1 input dạng NUMBER

→ DOM BAN ĐẦU:

  - value = "0"

  - min = 0

  - max = 100

  - step = 5

→ Trình duyệt nội bộ ghi nhớ:

  - Mỗi lần tăng/giảm = ±5

→ CSS mặc định render ô nhập + nút ▲ ▼

########################################

### GIAI ĐOẠN 3: TẠO SLIDER + LIÊN KẾT DOM

########################################

###

<input type="range" min="0" max="180" step="10" value="90" oninput="output.value = this.value">

###

→ DOM tạo input type="range"

→ Trạng thái BAN ĐẦU:

  - value = "90"

  - step = 10 → slider nhảy theo từng nấc 10

→ oninput được gắn:

  - Khi value thay đổi → JS nội tuyến được gọi

→ GIẢI THÍCH DẤU .

  - output.value

    → truy cập thuộc tính value của OBJECT DOM có id="output"

  - this.value

    → this = chính input range này

    → value = giá trị hiện tại của slider

###

<output id="output">90</output>

###

→ DOM tạo node <output>

→ id="output" được đăng ký trong document

→ Text node ban đầu = "90"

→ CSS mặc định hiển thị chữ inline

########################################

### GIAI ĐOẠN 4: SCRIPT ĐƯỢC ĐĂNG KÝ

########################################

###

<script> function guiLenh(giaTri) { ... } </script>

###

→ Trình duyệt đọc JS

→ Tạo function guiLenh trong global scope

→ CHƯA CHẠY, chỉ đăng ký

→ DOM KHÔNG thay đổi ở giai đoạn này

########################################

### GIAI ĐOẠN 5: NGƯỜI DÙNG TƯƠNG TÁC SLIDER

########################################

###

Người dùng kéo slider

###

→ Giá trị slider thay đổi từ 90 → 100 (ví dụ)

→ Trình duyệt kiểm tra step = 10

→ Giá trị hợp lệ → cho phép cập nhật

###

oninput="output.value = this.value"

###

→ Sự kiện input phát sinh

→ JS nội tuyến được chạy

→ TRƯỚC KHI ĐỔI:

  - output.value = "90"

→ JS là thằng đổi

→ this.value = "100"

→ output.value = "100"

→ SAU KHI ĐỔI:

  - DOM node <output> hiển thị "100"

→ CSS reflow nhẹ

→ Screen reader đọc giá trị mới

########################################

### GIAI ĐOẠN 6: NGƯỜI DÙNG NHẬP SỐ Ở INPUT CUỐI

########################################

###

<input type="number" step="10" value="0" oninput="guiLenh(this.value)">

###

→ Input number có step = 10

→ Mỗi lần ▲ ▼ hoặc nhập số:

  - Trình duyệt tự làm tròn theo step

###

oninput="guiLenh(this.value)"

###

→ Khi value đổi → gọi function guiLenh

→ this.value:

  - this = input number

  - value = giá trị hiện tại (ví dụ "30")

########################################

### GIAI ĐOẠN 7: JS CHẠY FUNCTION

########################################

###

function guiLenh(giaTri) { console.log(...) }

###

→ Dòng đầu tiên chạy:

  - nhận tham số giaTri = "30"

→ Không có if / === / ?:

  - Không so sánh

  - Không rẽ nhánh

→ Chỉ thực hiện console.log

→ DOM:

  - KHÔNG bị thay đổi

→ CSS:

  - KHÔNG phản ứng

→ Đây là mô phỏng gửi lệnh ra ESP / robot

########################################

### TỔNG KẾT VAI TRÒ

########################################

HTML:

→ Khai báo cấu trúc + step + oninput

DOM:

→ Lưu trạng thái value, min, max, step

CSS:

→ Tự động render input, slider, output

JavaScript:

→ Nhận giá trị ĐÃ BỊ GIỚI HẠN bởi step

→ Gửi lệnh / cập nhật DOM

ARIA / Screen Reader:

→ Đọc value mới mỗi khi DOM thay đổi

👉 Trình duyệt là thằng KIỂM SOÁT step

👉 JS chỉ LẤY KẾT QUẢ CUỐI CÙNG

-->

========================================

=== PHÂN TÍCH THUỘC TÍNH HTML: minlength / maxlength ===

========================================

[182]. Thuộc tính minlength & maxlength

→ Giới hạn ĐỘ DÀI CHUỖI mà người dùng được phép nhập vào input / textarea.

------------------------------------------------

[182.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: minlength bắt BUỘC nhập TỐI THIỂU bao nhiêu ký tự.

- Ý 2: maxlength chặn KHÔNG cho nhập vượt quá số ký tự cho phép (gõ tới là dừng).

------------------------------------------------

[182.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Giới hạn mật khẩu, tên đăng nhập

  + Giới hạn nội dung gửi về server / ESP

  + Tránh nhập thiếu hoặc nhập tràn dữ liệu

Ví dụ ngắn:

- Nhập mật khẩu < 6 ký tự → ❌ không gửi

- Nhập lệnh quá dài → ❌ bị chặn ngay khi gõ

------------------------------------------------

[182.3] PHẠM VI ÁP DỤNG

- Loại: Attribute

- Áp cho:

  + <input type="text | password | email | search">

  + <textarea>

- Có cần focus không?

  + Có (chỉ kiểm tra khi người dùng đang nhập)

------------------------------------------------

[182.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <input minlength="số" maxlength="số">

- Ví dụ đúng:

  <input type="password" minlength="6" maxlength="12">

- Ví dụ SAI:

  <input minlength="abc">

  → SAI vì giá trị phải là SỐ

------------------------------------------------

[182.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Số nguyên dương

- Ví dụ hay gặp:

  + minlength="3"

  + maxlength="10"

  + maxlength="255" (chuẩn backend hay dùng)

------------------------------------------------

[182.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng gõ bàn phím

2. Trình duyệt đếm số ký tự hiện tại

3. maxlength:

   - đạt giới hạn → CHẶN không cho gõ thêm

4. minlength:

   - chỉ được kiểm tra khi submit / validate

5. JS nhận value:

   - có thể hợp lệ hoặc KHÔNG hợp lệ

6. DOM giữ nguyên value, chỉ trạng thái validity thay đổi

Rất hay dùng để DEBUG form không submit được

------------------------------------------------

[182.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute (thuộc HTML constraint)

- Ai kích hoạt:

  + Người dùng gõ phím

  + Trình duyệt kiểm tra độ dài

- JavaScript:

  + Không kiểm soát giới hạn này

  + Chỉ ĐỌC kết quả (value / validity)

------------------------------------------------

[182.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Dùng minlength nhưng không có submit

  → Không thấy lỗi, tưởng là không hoạt động

- Lỗi 2: Dùng cho input type="number"

  → ❌ KHÔNG có tác dụng

- Cách nhớ nhanh:

  → min/max là cho SỐ

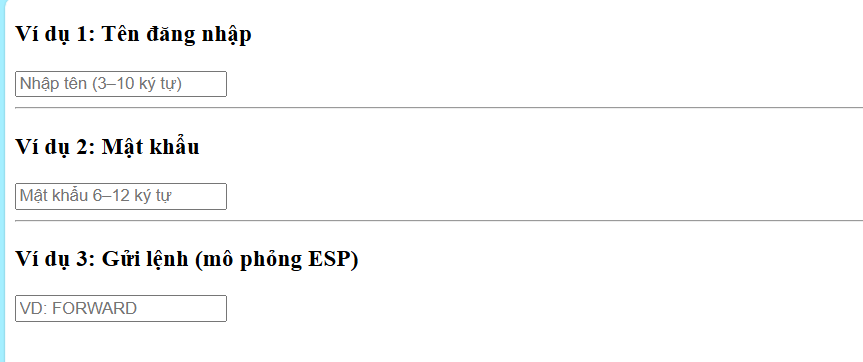
  → minlength/maxlength là cho CHUỖI

------------------------------------------------

[182.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ minlength / maxlength dùng để GIỚI HẠN ĐỘ DÀI CHUỖI khi người dùng nhập dữ liệu ✍️

182.10.EXAMBLE giới hạn ký tự



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ minlength / maxlength giới hạn ký tự</title>

</head>

<body>

  <h3>Ví dụ 1: Tên đăng nhập</h3>

  <!-- Ít nhất 3 ký tự, tối đa 10 ký tự -->

  <input type="text" minlength="3" maxlength="10" placeholder="Nhập tên (3–10 ký tự)">

  <hr>

  <h3>Ví dụ 2: Mật khẩu</h3>

  <!-- Bắt buộc nhập từ 6 đến 12 ký tự -->

  <input type="password" minlength="6" maxlength="12" placeholder="Mật khẩu 6–12 ký tự">

  <hr>

  <h3>Ví dụ 3: Gửi lệnh (mô phỏng ESP)</h3>

  <!-- Lệnh không quá 8 ký tự -->

  <input type="text" maxlength="8" oninput="guiLenh(this.value)" placeholder="VD: FORWARD">

  <script>

    function guiLenh(lenh) {

      console.log("Gửi lệnh:", lenh);

    }

  </script>

</body>

</html>

<!--

########################################

### GIAI ĐOẠN 1: TRANG BẮT ĐẦU LOAD

########################################

###

<!DOCTYPE html>

###

→ Trình duyệt nhận biết đây là tài liệu HTML5

→ Quyết định cách parse DOM, cách áp CSS mặc định, cách kiểm tra constraint (minlength / maxlength)

###

<html lang="vi">

###

→ Trình duyệt tạo NODE GỐC <html>

→ Thuộc tính lang="vi" được gắn vào DOM

→ Screen reader ghi nhớ: nội dung trang là tiếng Việt

###

<head> … </head>

###

→ DOM nhánh <head> được tạo

→ <meta charset="UTF-8">: trình duyệt biết cách đọc ký tự tiếng Việt

→ <title>: lưu tiêu đề tab, KHÔNG ảnh hưởng DOM hiển thị

########################################

### GIAI ĐOẠN 2: BODY + INPUT ĐƯỢC TẠO

########################################

###

<body>

###

→ Trình duyệt tạo node <body>

→ Mọi node phía dưới đều có khả năng hiển thị

###

<input type="text" minlength="3" maxlength="10" …>

###

→ DOM tạo input dạng TEXT

→ Trạng thái DOM BAN ĐẦU:

  - value = "" (rỗng)

  - minlength = 3

  - maxlength = 10

→ Trình duyệt nội bộ ghi:

  - Cho gõ tối đa 10 ký tự

  - Chưa kiểm tra minlength ở thời điểm này

→ CSS mặc định render ô input trống

########################################

### GIAI ĐOẠN 3: INPUT PASSWORD

########################################

###

<input type="password" minlength="6" maxlength="12" …>

###

→ DOM tạo input PASSWORD

→ Trạng thái BAN ĐẦU:

  - value = ""

  - minlength = 6

  - maxlength = 12

→ Trình duyệt ghi nhớ:

  - Gõ đến 12 ký tự thì CHẶN

  - Khi submit / validate mới kiểm minlength

→ CSS mặc định:

  - Che ký tự bằng dấu ●●●

########################################

### GIAI ĐOẠN 4: INPUT GỬI LỆNH + SỰ KIỆN

########################################

###

<input type="text" maxlength="8" oninput="guiLenh(this.value)">

###

→ DOM tạo input TEXT

→ Trạng thái BAN ĐẦU:

  - value = ""

  - maxlength = 8

→ oninput được đăng ký:

  - Mỗi lần value đổi → JS chạy NGAY

→ GIẢI THÍCH DẤU .

  - this.value

    → this = chính input đang được gõ

    → value = nội dung hiện tại của input

########################################

### GIAI ĐOẠN 5: SCRIPT ĐƯỢC ĐĂNG KÝ

########################################

###

function guiLenh(lenh) { … }

###

→ Trình duyệt đọc JS

→ Tạo function guiLenh trong bộ nhớ

→ CHƯA chạy, chỉ đăng ký

→ DOM không thay đổi

########################################

### GIAI ĐOẠN 6: NGƯỜI DÙNG BẮT ĐẦU GÕ

########################################

###

Người dùng gõ vào input Tên đăng nhập

###

→ Mỗi phím bấm:

  - Trình duyệt tăng độ dài chuỗi

  - So sánh độ dài với maxlength = 10

→ Nếu đủ 10 ký tự:

  - Phím tiếp theo bị CHẶN

→ minlength:

  - CHƯA được kiểm tra

→ DOM value thay đổi liên tục theo từng phím

########################################

### GIAI ĐOẠN 7: NGƯỜI DÙNG GÕ INPUT GỬI LỆNH

########################################

###

oninput="guiLenh(this.value)"

###

→ Mỗi ký tự được gõ:

  - Sự kiện input phát sinh

  - JS được gọi NGAY

→ TRƯỚC KHI ĐỔI:

  - value = "FOR"

→ Người dùng gõ thêm "W"

→ SAU KHI ĐỔI:

  - value = "FORW"

→ JS là thằng nhận giá trị mới

########################################

### GIAI ĐOẠN 8: JS CHẠY FUNCTION

########################################

###

function guiLenh(lenh)

###

→ Dòng đầu tiên chạy:

  - lenh = this.value (ví dụ "FORW")

→ Không có if / === / ?:

  - Không so sánh

  - Không tạo true / false

→ console.log chạy

→ DOM:

  - KHÔNG bị thay đổi

→ CSS:

  - KHÔNG phản ứng

→ Đây là mô phỏng gửi lệnh ra ESP

########################################

### GIAI ĐOẠN 9: GIỚI HẠN MAXLENGTH KÍCH HOẠT

########################################

###

maxlength="8"

###

→ Khi value đạt 8 ký tự:

  - Trình duyệt CHẶN phím tiếp theo

→ TRƯỚC:

  - value = "FORWARD"

→ SAU:

  - value vẫn = "FORWARD"

→ JS:

  - KHÔNG được gọi thêm

→ DOM:

  - Không đổi

→ Đây là chỗ rất hay DEBUG

########################################

### TỔNG KẾT VAI TRÒ CUỐI

########################################

HTML:

→ Khai báo constraint (minlength / maxlength)

DOM:

→ Lưu value + trạng thái hợp lệ

CSS:

→ Render input + che password

JavaScript:

→ Nhận value ĐÃ BỊ GIỚI HẠN

→ Không kiểm soát độ dài

ARIA / Screen Reader:

→ Đọc nội dung input theo DOM hiện tại

Trình duyệt là thằng CHẶN

JS chỉ là thằng NHẬN

minlength chỉ có tác dụng khi VALIDATE

maxlength chặn NGAY LÚC GÕ

========================================

=== PHÂN TÍCH THUỘC TÍNH HTML: autocomplete ===

========================================

[183]. Thuộc tính autocomplete

  → Điều khiển việc TRÌNH DUYỆT có tự động GỢI Ý / ĐIỀN LẠI dữ liệu người dùng đã nhập trước đó hay không.

  ------------------------------------------------

  [183.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

  - Ý 1: Bật / tắt việc trình duyệt tự nhớ và gợi ý giá trị đã nhập.

  - Ý 2: Dùng khi nhập form, đăng nhập, mật khẩu, lệnh điều khiển.

  Không xử lý click, mà kích hoạt khi GÕ PHÍM

  ------------------------------------------------

  [183.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

  - Dùng để:

  + Gợi ý tên đăng nhập, email cho người dùng

  + TẮT gợi ý khi nhập lệnh điều khiển ESP / robot

  + Tránh lộ dữ liệu nhạy cảm (mật khẩu, mã lệnh)

  Ví dụ ngắn:

  - Gõ user → trình duyệt gợi ý user cũ

  - Nhập lệnh điều khiển → ❌ KHÔNG cho gợi ý lung tung

  -----------------------------------------------

  [183.3] PHẠM VI ÁP DỤNG

  - Loại: Attribute

  - Áp cho:

  + <form>

    + <input>

    + <textarea>

- Có cần focus không?

  + Có (chỉ hoạt động khi input được focus)

------------------------------------------------

[183.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <input autocomplete="giá\_trị">

- Ví dụ đúng:

  <input type="text" autocomplete="off">

- Ví dụ SAI:

  <input autocomplete="false">

  → SAI vì autocomplete KHÔNG nhận true / false

------------------------------------------------

[183.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Chuỗi định nghĩa sẵn

- Ví dụ hay gặp:

  + "on"   → cho phép gợi ý

  + "off"  → tắt hoàn toàn

  + "username"

  + "email"

  + "current-password"

  + "new-password"

------------------------------------------------

[183.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng focus vào input

2. Trình duyệt kiểm tra autocomplete

3. Nếu = "on" hoặc có keyword:

   → lấy dữ liệu đã lưu

4. Dropdown gợi ý hiện ra

5. Người dùng chọn → value được điền vào DOM

6. JS đọc value đã được điền sẵn

Rất quan trọng khi DEBUG form tự điền sai dữ liệu

------------------------------------------------

[183.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute (hành vi trình duyệt)

- Ai kích hoạt:

  + Người dùng focus / gõ phím

  + Trình duyệt tự quyết định hiển thị gợi ý

- JavaScript:

  + KHÔNG điều khiển danh sách gợi ý

  + Chỉ nhận kết quả cuối cùng

------------------------------------------------

[183.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Dùng autocomplete="off" nhưng vẫn thấy gợi ý

  → Một số trình duyệt ƯU TIÊN trải nghiệm người dùng

- Lỗi 2: Tắt autocomplete ở <input> nhưng <form> bật

  → Trình duyệt vẫn gợi ý

- Cách nhớ nhanh:

  → Muốn CHẮC CHẮN tắt → đặt ở cả <form> và <input>

------------------------------------------------

[183.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ autocomplete dùng để KIỂM SOÁT việc trình duyệt TỰ ĐIỀN dữ liệu khi người dùng nhập ⌨️

183.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test autocomplete</title>

</head>

<body>

<h3>1️⃣ Ô có autocomplete (trình duyệt sẽ NHỚ)</h3>

<p>👉 Nhập gì đó, reload trang, click lại ô này</p>

<input

  type="text"

  name="email"

  autocomplete="email"

  placeholder="Nhập email ở đây"

>

<hr>

<h3>2️⃣ Ô KHÔNG autocomplete (trình duyệt KHÔNG NHỚ)</h3>

<p>👉 Nhập gì đó, reload trang, click lại ô này</p>

<input

  type="text"

  autocomplete="off"

  placeholder="Nhập lệnh / dữ liệu"

>

<hr>

<h3>3️⃣ Nhập lệnh điều khiển (mô phỏng)</h3>

<p>👉 Gõ FORWARD, STOP, LEFT…</p>

<input

  type="text"

  autocomplete="off"

  placeholder="VD: FORWARD"

>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[184]. Thuộc tính list

  → Liên kết thẻ <input> với <datalist> để HIỆN danh sách gợi ý khi nhập

    ------------------------------------------------

    [184.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

    - Ý 1: Kích hoạt danh sách gợi ý khi người dùng GÕ vào input

    - Ý 2: Không cần JavaScript, browser tự xử lý

    ------------------------------------------------

    [184.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

    - Dùng để:

    + Chọn lệnh điều khiển nhanh

    + Gợi ý email / mã / chế độ

    + Tránh gõ sai dữ liệu

    Ví dụ ngắn:

    - Gõ "t" → gợi ý "tiến"

    - Gõ "d" → gợi ý "dừng"

    ------------------------------------------------

    [184.3] PHẠM VI ÁP DỤNG

    - Loại: Attribute (HTML thường)

    - Áp cho:

    + <input> (text, search, email, number...)

    - Điều kiện:

    + input PHẢI được focus

    + <datalist> phải tồn tại và có id trùng

      ------------------------------------------------

      [184.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

      - Cú pháp chuẩn:

      <input list="id\_datalist">

      - Ví dụ đúng:

      <input list="lenhXe">

      <datalist id="lenhXe">

        <option value="tiến">

        <option value="lùi">

      </datalist>

      - Ví dụ SAI:

      <input list="lenhXe">

      <datalist id="lenh\_xe"> ← id KHÔNG trùng

        ------------------------------------------------

        [184.5] GIÁ TRỊ THƯỜNG DÙNG

        - Kiểu giá trị:

        + Chuỗi (string)

        - Ví dụ giá trị hay gặp:

        + "commands"

        + "emailList"

        + "modeSelect"

        ------------------------------------------------

        [184.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

        1. Người dùng click vào input

        2. Người dùng gõ ký tự

        3. Browser dò <datalist> theo list="..."

          4. Lọc option phù hợp

          5. Hiện dropdown gợi ý

          6. Người dùng chọn → value vào input

          RẤT DỄ DEBUG vì KHÔNG có JS

          ------------------------------------------------

          [184.7] BẢN CHẤT HOẠT ĐỘNG

          - Là: Attribute HTML

          - Ai kích hoạt:

          + Người dùng (bàn phím)

          + Trình duyệt (tự xử lý)

          - JavaScript:

          + Chỉ đọc value, KHÔNG điều khiển dropdown

          ------------------------------------------------

          [184.8] LƯU Ý / LỖI HAY GẶP

          - Lỗi 1: id datalist ≠ giá trị list

          - Lỗi 2: Quên đặt <option value="">

            - Lỗi 3: Dùng với <select> (SAI – list chỉ cho input)

              Mẹo nhớ:

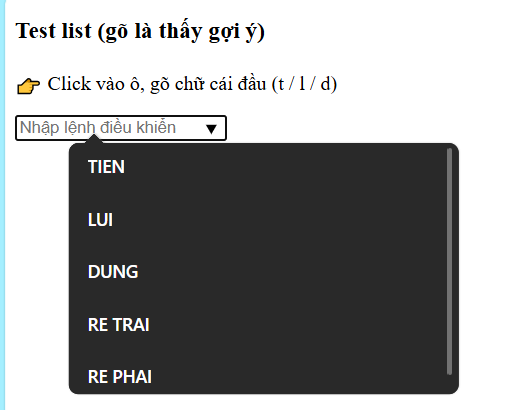
              → list = "nối input với datalist"

              ------------------------------------------------

              [184.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

              → “list dùng để HIỆN GỢI Ý khi người dùng GÕ vào input”

184.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test list + datalist</title>

</head>

<body>

  <h3>Test list (gõ là thấy gợi ý)</h3>

  <p>👉 Click vào ô, gõ chữ cái đầu (t / l / d)</p>

  <input type="text" list="lenhXe" placeholder="Nhập lệnh điều khiển">

  <datalist id="lenhXe">

    <option value="TIEN">

    <option value="LUI">

    <option value="DUNG">

    <option value="RE TRAI">

    <option value="RE PHAI">

  </datalist>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[185]. Thuộc tính accept

→ Giới hạn LOẠI FILE được PHÉP CHỌN khi người dùng mở hộp thoại upload file.

------------------------------------------------

[185.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Lọc file ngay từ lúc CHỌN FILE (chỉ hiện file hợp lệ).

- Ý 2: Dùng khi người dùng click chọn file từ máy tính.

------------------------------------------------

[185.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Chỉ cho upload ảnh / video / file cấu hình

  + Tránh người dùng chọn sai file

  + Giảm lỗi khi gửi file lên server / ESP

Ví dụ ngắn:

- Chọn ảnh → chỉ hiện .jpg .png

- Nạp firmware → chỉ hiện .bin

------------------------------------------------

[185.3] PHẠM VI ÁP DỤNG

- Loại: Attribute

- Áp cho:

  + <input type="file">

- Có cần focus không?

  + Không (kích hoạt khi click mở file picker)

------------------------------------------------

[185.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <input type="file" accept="giá\_trị">

- Ví dụ đúng:

  <input type="file" accept=".jpg,.png">

- Ví dụ SAI:

  <input type="file" accept="jpg">

  → SAI vì thiếu dấu chấm hoặc MIME type

------------------------------------------------

[185.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Chuỗi (đuôi file hoặc MIME type)

- Ví dụ giá trị hay gặp:

  + ".jpg,.png,.gif"

  + "image/\*"

  + "video/\*"

  + ".bin"

  + "application/pdf"

------------------------------------------------

[185.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng click input file

2. Trình duyệt mở hộp thoại chọn file

3. accept được đọc NGAY LÚC NÀY

4. Trình duyệt lọc danh sách file hiển thị

5. Người dùng chọn file

6. JS nhận file đã được lọc sẵn

Rất quan trọng khi DEBUG upload sai loại file

------------------------------------------------

[185.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute (hành vi trình duyệt)

- Ai kích hoạt:

  + Người dùng click

  + Trình duyệt tự lọc file

- JavaScript:

  + Không quyết định file được hiển thị

  + Chỉ kiểm tra lại file sau khi chọn

------------------------------------------------

[185.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Nghĩ accept là bảo mật

  → Sai, người dùng vẫn có thể chọn file khác bằng cách đổi filter

- Lỗi 2: Quên type="file"

  → accept KHÔNG có tác dụng

- Cách nhớ nhanh:

  → accept chỉ là LỌC HIỂN THỊ, không phải CHẶN THẬT

------------------------------------------------

[185.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “accept dùng để LỌC LOẠI FILE khi người dùng CHỌN FILE 📁”

185.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test accept</title>

</head>

<body>

  <h3>Ví dụ 1: Chỉ chọn ẢNH</h3>

  <input type="file" accept="image/\*">

  <hr>

  <h3>Ví dụ 2: Chỉ chọn file .bin (firmware ESP)</h3>

  <input type="file" accept=".bin">

  <hr>

  <h3>Ví dụ 3: Chọn nhiều loại file cụ thể</h3>

  <input type="file" accept=".jpg,.png,.pdf">

  <hr>

  <h3>Ví dụ 4: Chỉ chọn VIDEO</h3>

  <input type="file" accept="video/\*">

</body>

</html>

HẾT PHẦN

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[186]. Thuộc tính rows

  → Xác định CHIỀU CAO BAN ĐẦU (số dòng hiển thị) của ô nhập văn bản textarea

  ------------------------------------------------

  [186.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

  - Ý 1: Quy định textarea cao bao nhiêu DÒNG khi vừa render ra

  - Ý 2: Giúp người dùng NHẬP NHIỀU DÒNG dễ nhìn hơn

  ------------------------------------------------

  [186.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

  - Dùng để:

  + Nhập ghi chú

  + Nhập mô tả

  + Nhập log / message dài

  Ví dụ ngắn:

  - Nhập 1–2 dòng → dùng rows nhỏ

  - Nhập nội dung dài → tăng rows để nhìn rõ

  ------------------------------------------------

  [186.3] PHẠM VI ÁP DỤNG

  - Event-only: ❌

  - Global: ❌

  - ARIA-only: ❌

  - Áp cho:

  + <textarea> (CHỈ DUY NHẤT)

- Có cần focus không?

  + ❌ Không, chỉ ảnh hưởng giao diện ban đầu

------------------------------------------------

[186.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <textarea rows="số\_dòng"></textarea>

  - Ví dụ đúng:

  <textarea rows="4"></textarea>

  - Ví dụ SAI:

  <input rows="4">

  → ❌ rows KHÔNG dùng cho input

  ------------------------------------------------

  [186.5] GIÁ TRỊ THƯỜNG DÙNG

  - Kiểu giá trị:

  + Số nguyên dương

  - Ví dụ giá trị hay gặp:

  + 2

  + 4

  + 6

  + 10

  ------------------------------------------------

  [186.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

  1. Trình duyệt đọc thẻ <textarea>

2. Gặp thuộc tính rows

3. Tính chiều cao = số dòng \* line-height

4. Render textarea với chiều cao ban đầu

5. Người dùng nhập thêm → textarea KHÔNG tự cao lên (trừ CSS/JS)

------------------------------------------------

[186.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt (khi render HTML)

- JavaScript:

  + Có thể đọc / đổi rows sau khi load

------------------------------------------------

[186.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Dùng rows cho <input> → KHÔNG CÓ TÁC DỤNG

- Lỗi 2: Nghĩ rows giới hạn số dòng nhập → ❌ SAI

- Cách nhớ nhanh:

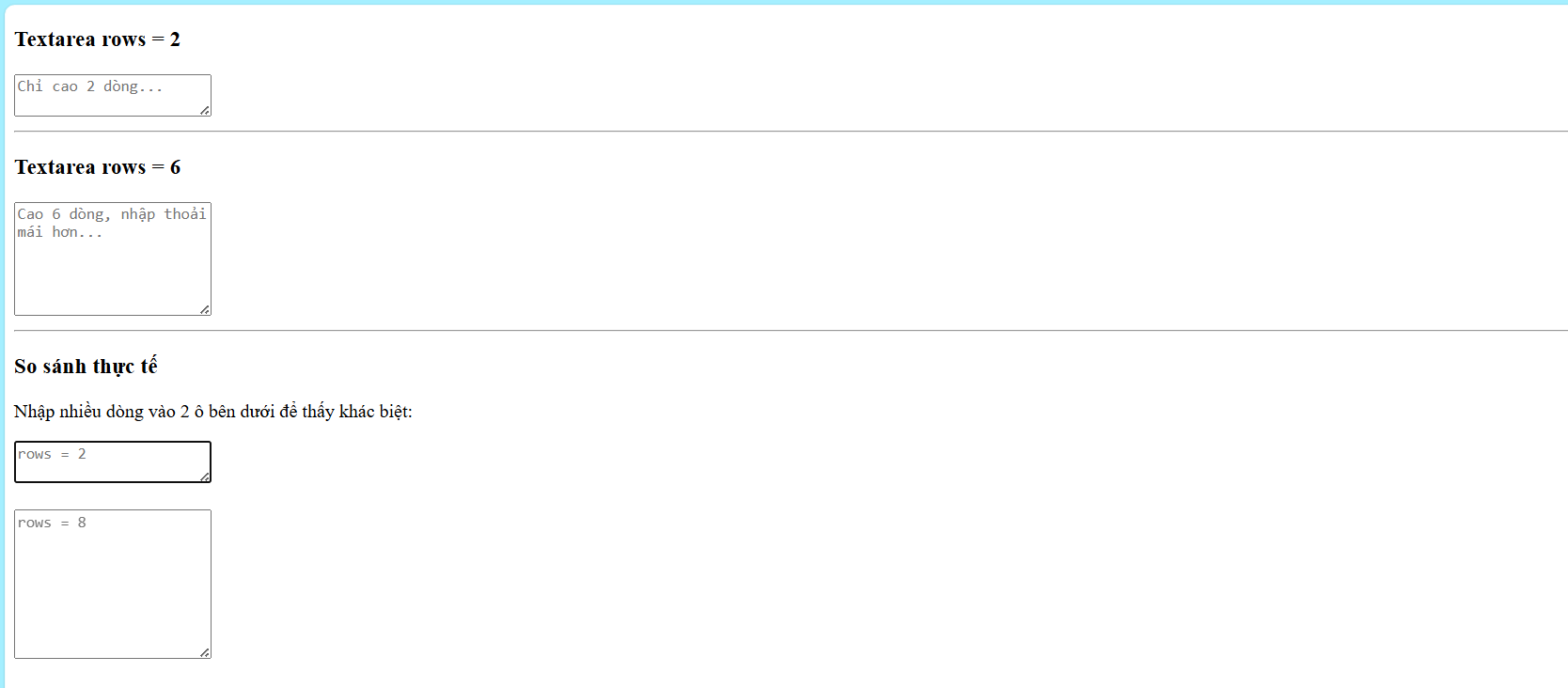
  → rows = CAO BAO NHIÊU, KHÔNG PHẢI NHẬP BAO NHIÊU

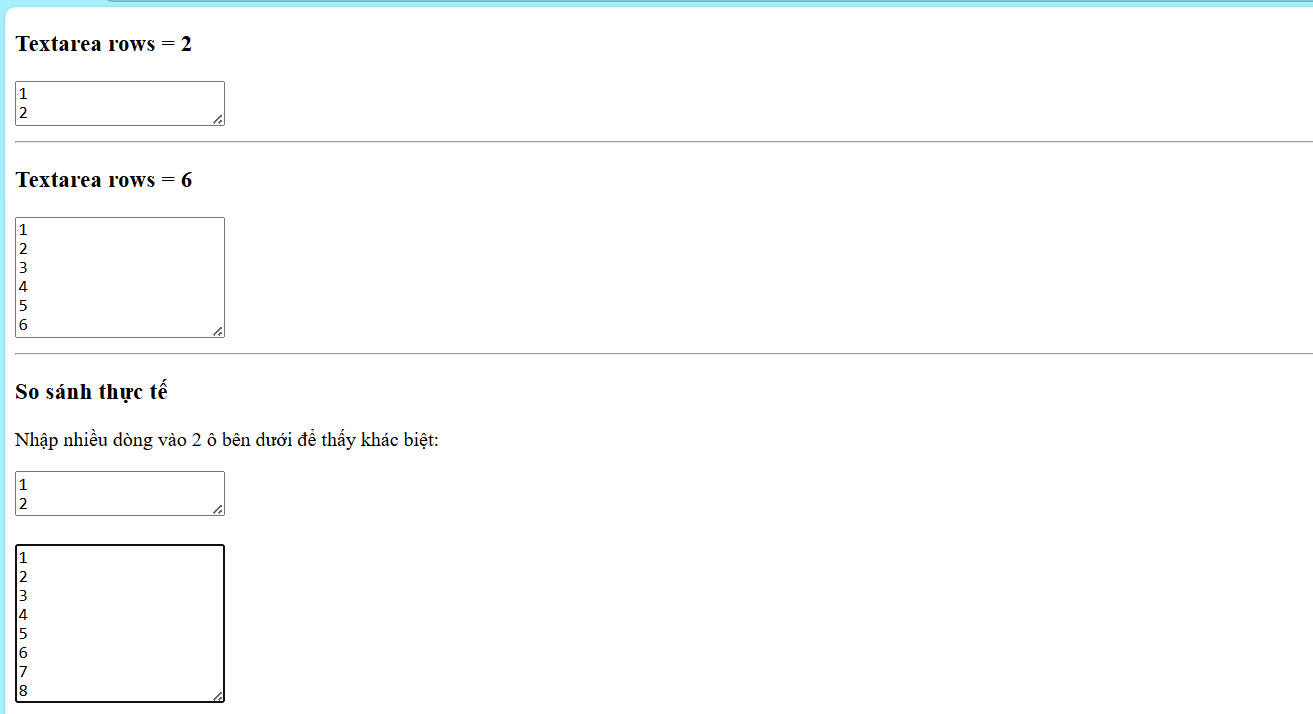
------------------------------------------------

[186.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “rows dùng để xác định textarea cao bao nhiêu dòng khi trang vừa tải”

186.10.EXAMBLE





<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ rows</title>

</head>

<body>

  <h3>Textarea rows = 2</h3>

  <textarea rows="2" placeholder="Chỉ cao 2 dòng..."></textarea>

  <hr>

  <h3>Textarea rows = 6</h3>

  <textarea rows="6" placeholder="Cao 6 dòng, nhập thoải mái hơn..."></textarea>

  <hr>

  <h3>So sánh thực tế</h3>

  <p>Nhập nhiều dòng vào 2 ô bên dưới để thấy khác biệt:</p>

  <textarea rows="2" placeholder="rows = 2"></textarea>

  <br><br>

  <textarea rows="8" placeholder="rows = 8"></textarea>

</body>

</html>

HẾT PHẦN

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[187]. Thuộc tính action

→ Quy định FORM sẽ GỬI dữ liệu đi ĐÂU khi submit

------------------------------------------------

[187.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Xác định URL / server nhận dữ liệu từ form

- Ý 2: Kích hoạt khi người dùng bấm nút submit

------------------------------------------------

[187.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Gửi lệnh điều khiển (xe / robot / ESP)

  + Gửi dữ liệu đăng nhập

  + Gửi form cấu hình thiết bị

Ví dụ ngắn:

- Bấm nút → gửi lệnh START lên server

- Bấm nút → gửi thông số tốc độ ESP

------------------------------------------------

[187.3] PHẠM VI ÁP DỤNG

- Event-only (chỉ hoạt động khi submit)

- Áp cho:

  + <form>

- Không cần focus phím

- Không áp cho input riêng lẻ

------------------------------------------------

[187.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <form action="URL">

- Ví dụ đúng:

  <form action="/send-command">

- Ví dụ SAI:

  <input action="/send-command">

  → action KHÔNG hoạt động trên input

------------------------------------------------

[187.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Chuỗi (URL)

- Ví dụ hay gặp:

  + "/login"

  + "/api/control"

  + "https://server.com/submit"

------------------------------------------------

[187.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng bấm nút submit

2. Trình duyệt đọc thuộc tính action của form

3. Gom toàn bộ dữ liệu input

4. Gửi dữ liệu tới URL trong action

5. Trình duyệt chuyển trang hoặc nhận phản hồi

------------------------------------------------

[187.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Người dùng (bấm submit)

  + Trình duyệt (tự gửi request)

- JavaScript KHÔNG bắt buộc

------------------------------------------------

[187.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Quên action → form gửi về chính trang hiện tại

- Lỗi 2: Dùng action cho input → KHÔNG chạy

- Cách nhớ nhanh:

  → action CHỈ đi với form

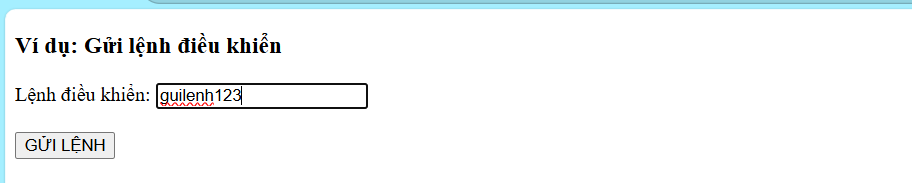
------------------------------------------------

[187.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “action dùng để chỉ ra NƠI NHẬN dữ liệu khi form được submit”

187.10.EXAMBLE







<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ action</title>

</head>

<body>

  <h3>Ví dụ: Gửi lệnh điều khiển</h3>

  <!-- action quyết định dữ liệu sẽ gửi đi đâu -->

  <form action="server-gia-lap.html">

    <label>

      Lệnh điều khiển:

      <input type="text" name="command" placeholder="VD: FORWARD">

    </label>

    <br><br>

    <button type="submit">GỬI LỆNH</button>

  </form>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[188]. Thuộc tính method

→ Quyết định CÁCH dữ liệu trong form được gửi đi khi submit

------------------------------------------------

[188.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Chọn KIỂU gửi dữ liệu (GET hoặc POST)

- Ý 2: Kích hoạt khi người dùng bấm nút submit trong <form>

------------------------------------------------

[188.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Gửi lệnh điều khiển (ESP / robot)

  + Gửi dữ liệu đăng nhập

  + Gửi trạng thái cảm biến

Ví dụ ngắn:

- Bấm nút → gửi lệnh TIẾN

- Trình duyệt gửi dữ liệu theo method đã chọn

------------------------------------------------

[188.3] PHẠM VI ÁP DỤNG

- Loại: Attribute

- Áp cho:

  + Thẻ <form>

- Không cần focus

- Không dùng trực tiếp cho input / button

------------------------------------------------

[188.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <form method="GET">

  <form method="POST">

- Ví dụ đúng:

  <form action="/control" method="POST">

- Ví dụ SAI:

  <input method="POST">

  → SAI vì method CHỈ dùng cho <form>

------------------------------------------------

[188.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị: Chuỗi

- Giá trị hay gặp:

  + "GET"

  + "POST"

------------------------------------------------

[188.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng nhập dữ liệu vào form

2. Người dùng bấm nút submit

3. Trình duyệt đọc thuộc tính method của <form>

4. Nếu method="GET":

   - Dữ liệu được gắn lên URL

5. Nếu method="POST":

   - Dữ liệu gửi ngầm trong body request

6. Trình duyệt gửi request theo method đã chọn

7. Trang / server nhận dữ liệu và phản hồi

------------------------------------------------

[188.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Người dùng (bấm submit)

  + Trình duyệt (đọc method và gửi request)

------------------------------------------------

[188.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Quên ghi method → mặc định là GET

- Lỗi 2: Dùng POST nhưng server không xử lý POST

- Cách nhớ nhanh:

  GET = gửi thấy trên URL

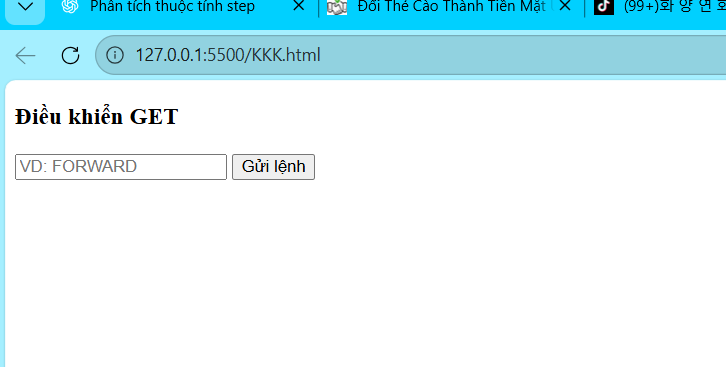
  POST = gửi ngầm, không hiện URL

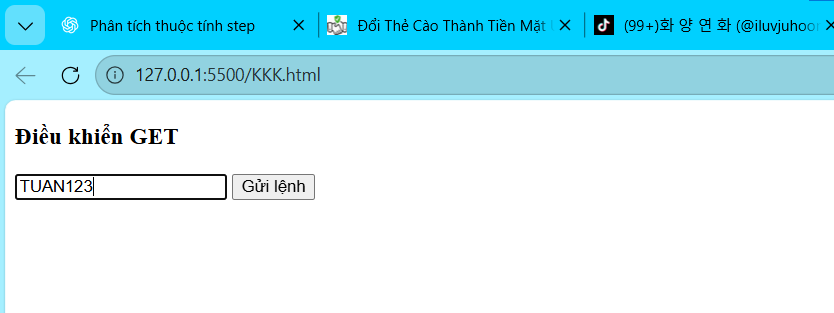
------------------------------------------------

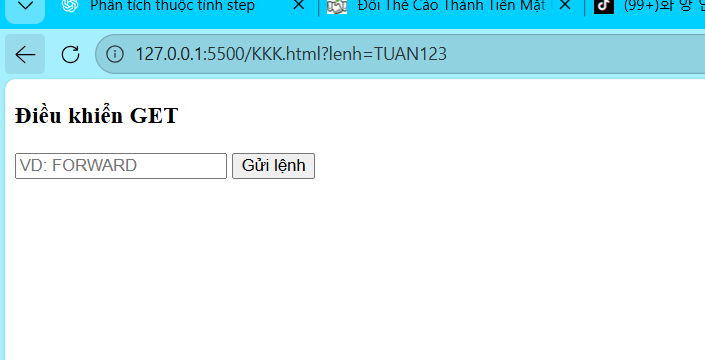
[188.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “method dùng để chọn CÁCH gửi dữ liệu khi form được submit”

188.10.EXAMBLE GET LỘ







<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ method GET</title>

</head>

<body>

  <h3>Điều khiển GET</h3>

  <!--

GET = dữ liệu lộ trên URL

-->

  <form action="" method="GET">

    <input type="text" name="lenh" placeholder="VD: FORWARD">

    <button type="submit">Gửi lệnh</button>

  </form>

</body>

</html>

188.11.EXAMBLE POST KO LỘ

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ method POST</title>

</head>

<body>

  <h3>Đăng nhập (POST)</h3>

<!--

POST → đăng nhập, dữ liệu quan trọng, không lộ URL

-->

  <form action="" method="POST">

    <input type="text" name="user" placeholder="Tên đăng nhập">

    <input type="password" name="pass" placeholder="Mật khẩu">

    <button type="submit">Đăng nhập</button>

  </form>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[189]. Thuộc tính enctype

→ Quy định CÁCH trình duyệt ĐÓNG GÓI dữ liệu form trước khi GỬI đi.

\*\*\*

Điện thoại / Máy tính

→ GỬI: HTTP REQUEST

   - Body chứa DỮ LIỆU:

     • Chuỗi: "FORWARD", "AUTO", "MyWiFi"

     • Số: 120, 80, 0

     • Trạng thái: ON / OFF, 1 / 0

   - Cách đóng gói: enctype

→ KHÔNG gửi chương trình, KHÔNG flash chip

ESP (đã chạy sẵn firmware)

→ NHẬN HTTP REQUEST

→ TÁCH dữ liệu trong request

→ CODE ESP quyết định:

   • điều khiển motor

   • đổi tham số

   • lưu cấu hình (RAM / EEPROM / Flash)

------------------------------------------------

[189.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Xác định dữ liệu form được MÃ HÓA theo kiểu nào khi submit

- Ý 2: Quyết định server / backend CÓ ĐỌC ĐƯỢC dữ liệu hay KHÔNG

-----------------------------------------------

[189.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Gửi lệnh điều khiển (robot / ESP)

  + Upload file (ảnh, log, firmware)

  + Gửi form đăng nhập / cấu hình

Ví dụ ngắn:

- Nhấn nút → gửi lệnh FORWARD

- Chọn file → gửi firmware lên ESP

------------------------------------------------

[189.3] PHẠM VI ÁP DỤNG

- Loại: Attribute (thuộc form)

- Áp cho:

  + <form>

- Không cần focus

- Chỉ có tác dụng KHI submit form

------------------------------------------------

[189.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <form action="..." method="POST" enctype="giá\_trị">

- Ví dụ ĐÚNG:

  <form method="POST" enctype="application/x-www-form-urlencoded">

- Ví dụ SAI:

  <input enctype="multipart/form-data">

  → SAI vì enctype KHÔNG áp cho input

------------------------------------------------

[189.5] GIÁ TRỊ THƯỜNG DÙNG

- application/x-www-form-urlencoded

  + Mặc định

  + Gửi text, số, lệnh đơn giản

- multipart/form-data

  + BẮT BUỘC khi upload file

  + Gửi được file + dữ liệu

- text/plain

  + Debug, test

  + Gửi dạng text thô

------------------------------------------------

[189.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng bấm nút submit

2. Trình duyệt đọc enctype của <form>

3. Trình duyệt đóng gói dữ liệu theo kiểu đã chọn

4. Dữ liệu được gửi qua HTTP (GET / POST)

5. Server nhận → giải mã → xử lý

Nếu chọn SAI enctype → server nhận RÁC / NULL

------------------------------------------------

[189.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Người dùng (submit)

  + Trình duyệt (đóng gói dữ liệu)

- JavaScript KHÔNG tự kích hoạt enctype

------------------------------------------------

[189.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Upload file nhưng KHÔNG dùng multipart/form-data

- Lỗi 2: Nghĩ rằng enctype dùng cho input (SAI)

- Cách nhớ nhanh:

  → Có FILE → multipart

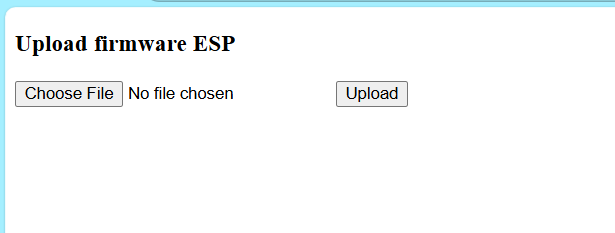
  → Không file → mặc định

------------------------------------------------

[189.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “enctype dùng để quyết định CÁCH gửi dữ liệu form khi submit xảy ra”

189.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>enctype upload file</title>

</head>

<body>

  <h3>Upload firmware ESP</h3>

  <form action="upload" method="POST" enctype="multipart/form-data">

    <input type="file" name="firmware">

    <button type="submit">Upload</button>

  </form>

</body>

</html>

<!--

📱 Điện thoại / 💻 Máy tính

→ GỬI: HTTP REQUEST

   - Body chứa DỮ LIỆU:

     • Chuỗi: "FORWARD", "AUTO", "MyWiFi"

     • Số: 120, 80, 0

     • Trạng thái: ON / OFF, 1 / 0

   - Cách đóng gói: enctype

→ KHÔNG gửi chương trình, KHÔNG flash chip

🌐 ESP (đã chạy sẵn firmware)

→ NHẬN HTTP REQUEST

→ TÁCH dữ liệu trong request

→ CODE ESP quyết định:

   • điều khiển motor

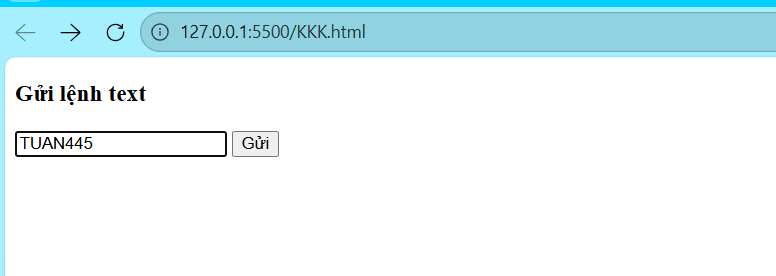
   • đổi tham số

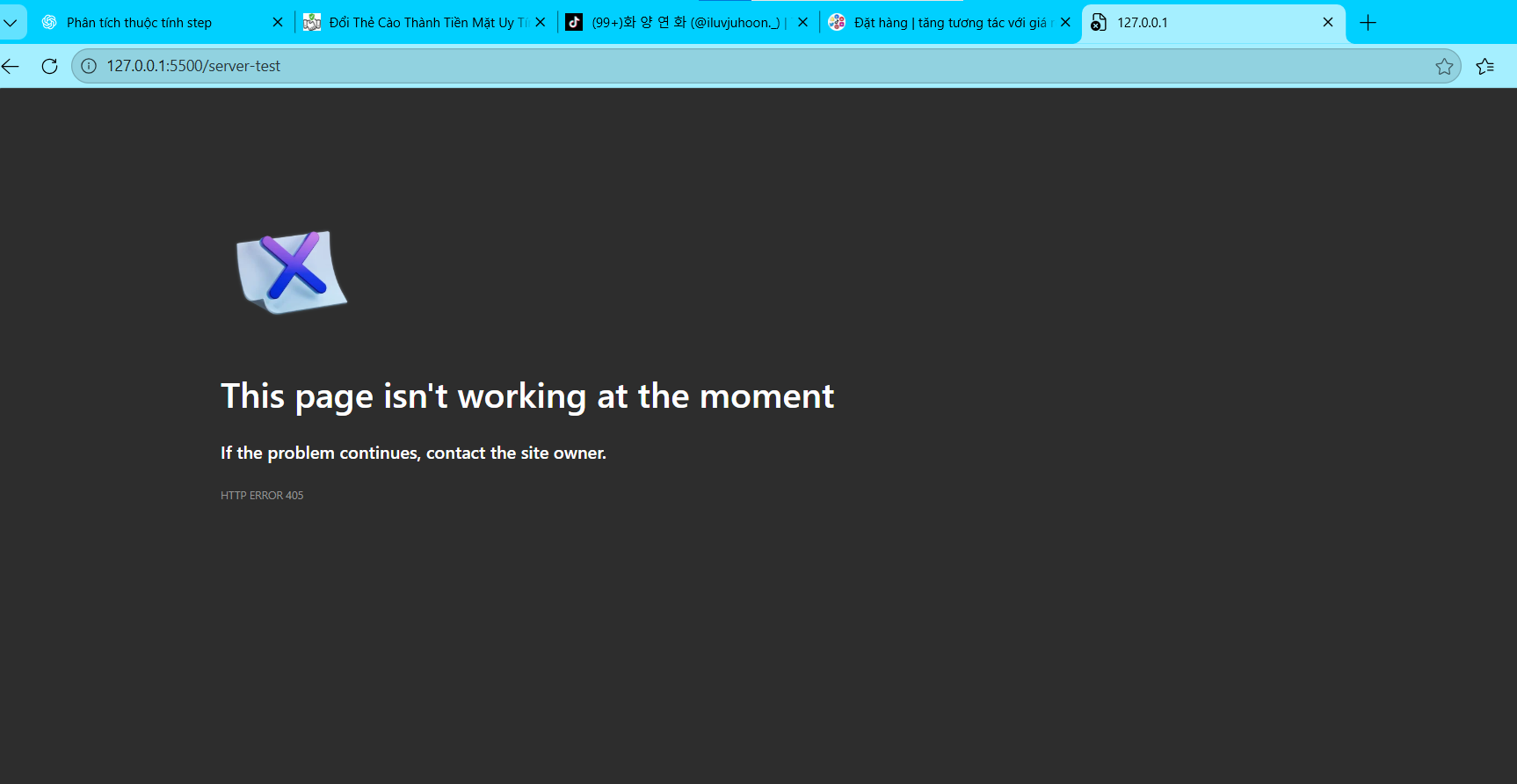
   • lưu cấu hình (RAM / EEPROM / Flash)

-->

189.11.EXAMBLE







<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>enctype mặc định</title>

</head>

<body>

<h3>Gửi lệnh text</h3>

<form action="server-test" method="POST">

  <input type="text" name="lenh" placeholder="VD: FORWARD">

  <button type="submit">Gửi</button>

</form>

</body>

</html>

========================================

=== PHÂN TÍCH THUỘC TÍNH HTML novalidate ===

========================================

[190]. Thuộc tính novalidate

→ TẮT kiểm tra hợp lệ mặc định của trình duyệt khi submit form

------------------------------------------------

[190.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: NGĂN trình duyệt chặn submit khi input chưa hợp lệ

- Ý 2: Cho phép JS / ESP tự xử lý validate theo ý mình

------------------------------------------------

[190.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Gửi cấu hình cho ESP dù thiếu / sai định dạng

  + Gửi lệnh điều khiển nhanh, không bị popup cản

  + Validate dữ liệu bằng JavaScript / firmware

Ví dụ ngắn:

- Nhấn nút → GỬI LỆNH NGAY

- Không bị trình duyệt báo: "Please fill out this field"

------------------------------------------------

[190.3] PHẠM VI ÁP DỤNG

- Loại: Form-only

- Áp cho:

  + <form>

- Không cần focus

- KHÔNG áp cho input riêng lẻ

------------------------------------------------

[190.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <form novalidate>

- Ví dụ đúng:

  <form action="/send" method="POST" novalidate>

- Ví dụ SAI:

  <input novalidate>   ← SAI (input không có quyền)

------------------------------------------------

[190.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Boolean attribute

- Cách dùng:

  + Có novalidate → BẬT tắt kiểm tra

  + Không có → trình duyệt tự kiểm tra

------------------------------------------------

[190.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng nhấn nút Submit

2. Trình duyệt thấy <form novalidate>

3. BỎ QUA toàn bộ kiểm tra:

   - required

   - minlength

   - pattern

4. HTTP request VẪN ĐƯỢC GỬI

5. JS / ESP nhận dữ liệu đầy đủ

DÙ input rỗng, sai định dạng, VẪN GỬI

------------------------------------------------

[190.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt đọc khi submit

- Ai xử lý tiếp:

  + JavaScript

  + ESP firmware

Trình duyệt KHÔNG can thiệp nữa

------------------------------------------------

[190.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Gắn novalidate vào input → KHÔNG có tác dụng

- Lỗi 2: Quên tự validate bằng JS / ESP → nhận dữ liệu rác

- Cách nhớ nhanh:

  + novalidate = "Trình duyệt đứng ngoài cuộc"

------------------------------------------------

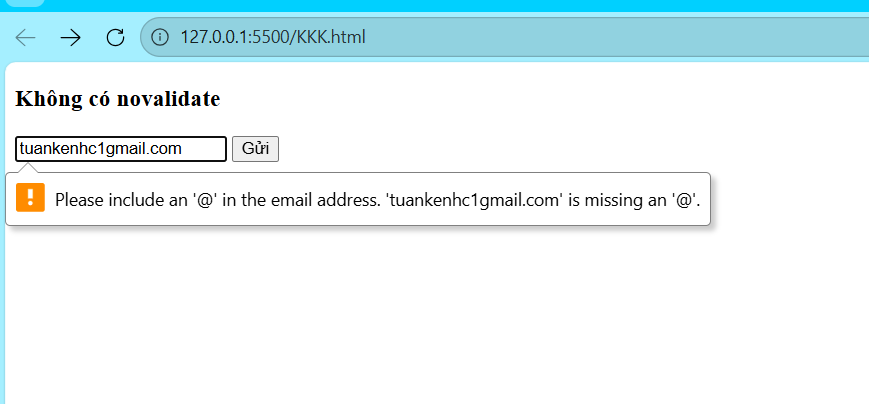
[190.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “novalidate dùng để TẮT kiểm tra form của trình duyệt khi SUBMIT xảy ra”

190.10.EXAMBLE

Để trống email → KHÔNG gửi

Nhập sai email → trình duyệt hiện cảnh báo ❌



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Không novalidate</title>

</head>

<body>

<h3>Không có novalidate</h3>

<form action="/send" method="POST">

  <input type="email" name="email" required placeholder="Nhập email">

  <button type="submit">Gửi</button>

</form>

</body>

</html>

190.11.EXAMBLE

Để trống email → VẪN GỬI

Nhập sai email → VẪN GỬI

ESP / JS nhận dữ liệu 100%

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Có novalidate</title>

</head>

<body>

  <h3>Có novalidate</h3>

  <form action="/send" method="POST" novalidate>

    <input type="email" name="email" required placeholder="Nhập email">

    <button type="submit">Gửi</button>

  </form>

</body>

</html>

<!--

Để trống email → VẪN GỬI

Nhập sai email → VẪN GỬI

ESP / JS nhận dữ liệu 100%

-->

190.12.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP Web Config (novalidate)</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      padding: 20px;

    }

    label {

      display: block;

      margin-top: 10px;

    }

    input {

      padding: 6px;

      width: 220px;

    }

    button {

      margin-top: 15px;

      padding: 8px 16px;

    }

  </style>

</head>

<body>

  <h3>Cấu hình WiFi cho ESP</h3>

  <form action="/config" method="POST" novalidate>

    <label>

      WiFi name:

      <input type="text" name="wifi" placeholder="VD: MyWiFi">

    </label>

    <label>

      Password:

      <input type="text" name="pass" placeholder="VD: 12345678">

    </label>

    <button type="submit">Lưu cấu hình</button>

  </form>

  <hr>

  <p>

    Form này:

    <br>• KHÔNG bị trình duyệt chặn

    <br>• Dù ô trống vẫn gửi

    <br>• ESP tự xử lý dữ liệu

  </p>

</body>

</html>

Hết

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[191]. Thuộc tính src

→ Chỉ ra NGUỒN TÀI NGUYÊN để trình duyệt tải và hiển thị (ảnh / video / file / nội dung nhúng)

------------------------------------------------

[191.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Xác định ĐƯỜNG DẪN file cần tải

- Ý 2: Kích hoạt tải nội dung NGAY KHI thẻ được parse

------------------------------------------------

[191.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Hiển thị hình ảnh camera/robot

  + Phát video hướng dẫn

  + Nạp trang con / tài liệu

- Ví dụ ngắn:

  - Gán src ảnh → hiển thị trạng thái

  - Đổi src video → phát nội dung mới

-----------------------------------------------

[191.3] PHẠM VI ÁP DỤNG

- Global: KHÔNG

- Event-only: KHÔNG

- ARIA-only: KHÔNG

- Áp cho thẻ:

  + <img>, <video>, <audio>, <iframe>, <script>, <source>

- Có cần focus không?

  + KHÔNG (tự tải)

------------------------------------------------

[191.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag src="duong\_dan">

- Ví dụ đúng:

  <img src="anh.png">

  <script src="app.js"></script>

- Ví dụ SAI:

  <img src=anh.png>        (thiếu dấu nháy)

  <img scr="anh.png">      (sai tên thuộc tính)

------------------------------------------------

[191.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Chuỗi (string)

- Ví dụ hay gặp:

  + "image.jpg"

  + "/assets/main.js"

  + "https://example.com/video.mp4"

  + "data:image/png;base64,..."

------------------------------------------------

[191.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt đọc HTML

2. Gặp thẻ có src

3. Gửi request tải tài nguyên

4. Nhận dữ liệu → gắn vào DOM

5. Màn hình hiển thị / script được chạy

------------------------------------------------

[191.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt (khi parse HTML)

  + JavaScript (khi gán src mới)

------------------------------------------------

[191.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Sai đường dẫn → file không tải

- Lỗi 2: Dùng src cho thẻ không hỗ trợ

- Cách nhớ nhanh:

  + src = "source" = nơi lấy dữ liệu

------------------------------------------------

[191.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “src dùng để chỉ nguồn tài nguyên để trình duyệt tải khi thẻ được đọc”

191.10.EXAMBLE

Ví dụ 1: <img src> – hiển thị hình ảnh

<img src="robot.png" alt="Robot đang chạy">

Trình duyệt đọc HTML → thấy src → tải robot.png → hiện hình ra màn hình

Ví dụ 2: <script src> – nạp file JavaScript

<script src="main.js"></script>

Trình duyệt tải main.js → chạy code JS bên trong

(RẤT hay dùng khi điều khiển ESP / robot / web app)

Ví dụ 3: <video src> – phát video

<video src="demo.mp4" controls></video>

src trỏ tới file video → trình duyệt phát nội dung

Ví dụ 4: <iframe src> – nhúng trang khác

<iframe src="https://example.com"></iframe>

Nhúng nguyên 1 trang web khác vào trang hiện tại

Ví dụ 5: JS đổi src khi đang chạy (quan trọng)

<img id="cam" src="off.png">

<script>

  document.getElementById("cam").src = "on.png";

</script>

JS đổi src → trình duyệt tải ảnh mới → màn hình đổi trạng thái

(RẤT giống: bật/tắt, tiến/dừng, ON/OFF)

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[192]. Thuộc tính alt

→ Cung cấp NỘI DUNG THAY THẾ cho hình ảnh khi ảnh không hiển thị hoặc khi dùng trình đọc màn hình

------------------------------------------------

[192.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Hiển thị CHỮ THAY ẢNH khi ảnh lỗi / không tải được

- Ý 2: Đọc MÔ TẢ ẢNH cho người dùng khiếm thị (screen reader)

------------------------------------------------

[192.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Hiển thị trạng thái khi mất hình

  + Báo ý nghĩa hình trong giao diện điều khiển

  + Giúp hệ thống đọc hiểu nội dung

- Ví dụ ngắn:

  - Camera mất kết nối → hiện chữ "CAMERA OFF"

  - Ảnh nút TIẾN → đọc là "Robot tiến"

------------------------------------------------

[192.3] PHẠM VI ÁP DỤNG

- Global: KHÔNG

- Event-only: KHÔNG

- ARIA-only: KHÔNG

- Áp cho thẻ:

  + <img> (CHỦ YẾU)

- Có cần focus không?

  + KHÔNG

------------------------------------------------

[192.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <img src="..." alt="noi\_dung\_mo\_ta">

- Ví dụ đúng:

  <img src="forward.png" alt="Robot đang tiến">

- Ví dụ SAI:

  <img src="forward.png">

  (thiếu alt → giảm accessibility)

  <img alt="Robot" src=forward.png>

  (thiếu dấu nháy giá trị)

------------------------------------------------

[192.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Chuỗi (string)

- Ví dụ hay gặp:

  + "Robot dang dung"

  + "Trang thai ON"

  + "Nut dieu khien tien"

  + ""  (ảnh trang trí, KHÔNG cần đọc)

-----------------------------------------------

[192.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt đọc thẻ <img>

2. Thử tải ảnh từ src

3. Nếu ảnh lỗi / không hiển thị

4. Hiện chữ trong alt HOẶC đọc bằng screen reader

5. Người dùng vẫn hiểu nội dung

------------------------------------------------

[192.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt (khi ảnh lỗi)

  + Trình đọc màn hình (khi quét DOM)

------------------------------------------------

[192.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Để trống alt dù ảnh mang ý nghĩa

- Lỗi 2: Nhét quá nhiều chữ vào alt

- Cách nhớ nhanh:

  + alt = nói cho người KHÔNG THẤY ẢNH biết ảnh là gì

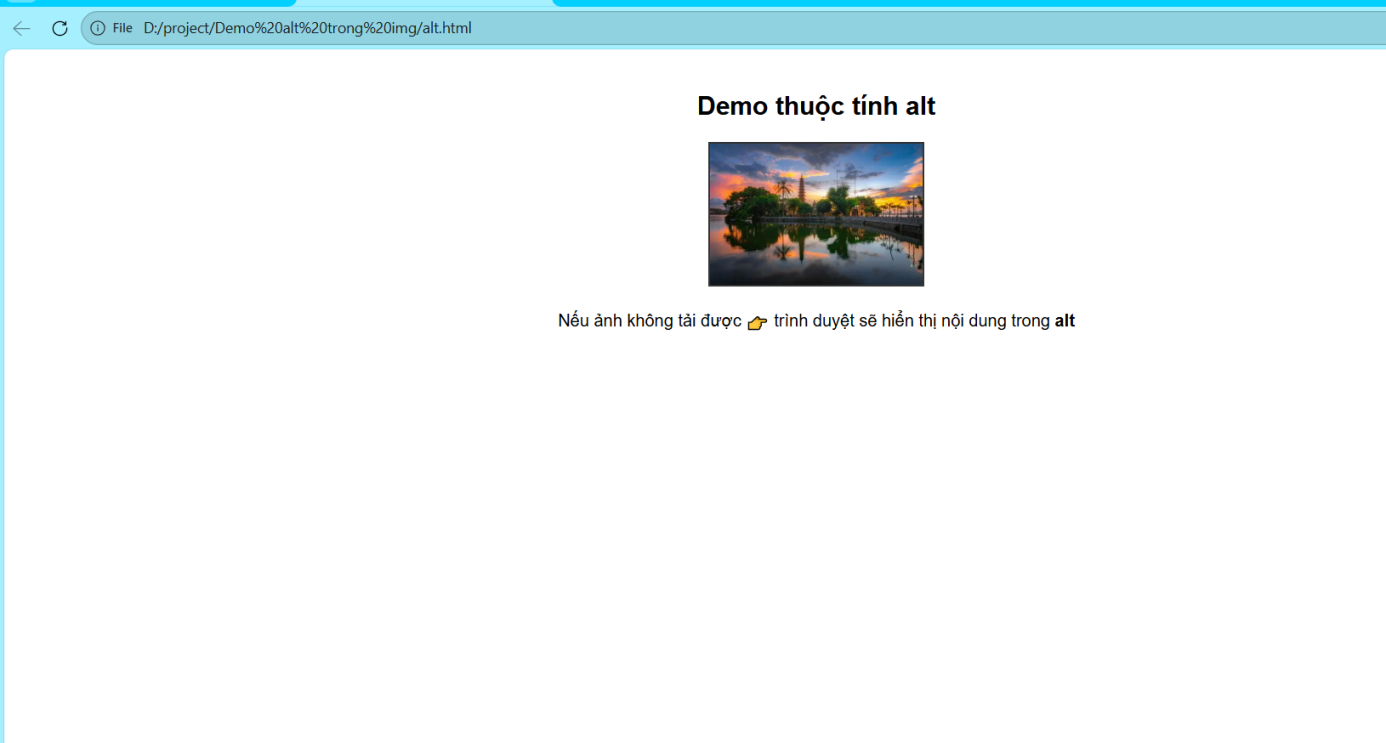
------------------------------------------------

[192.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

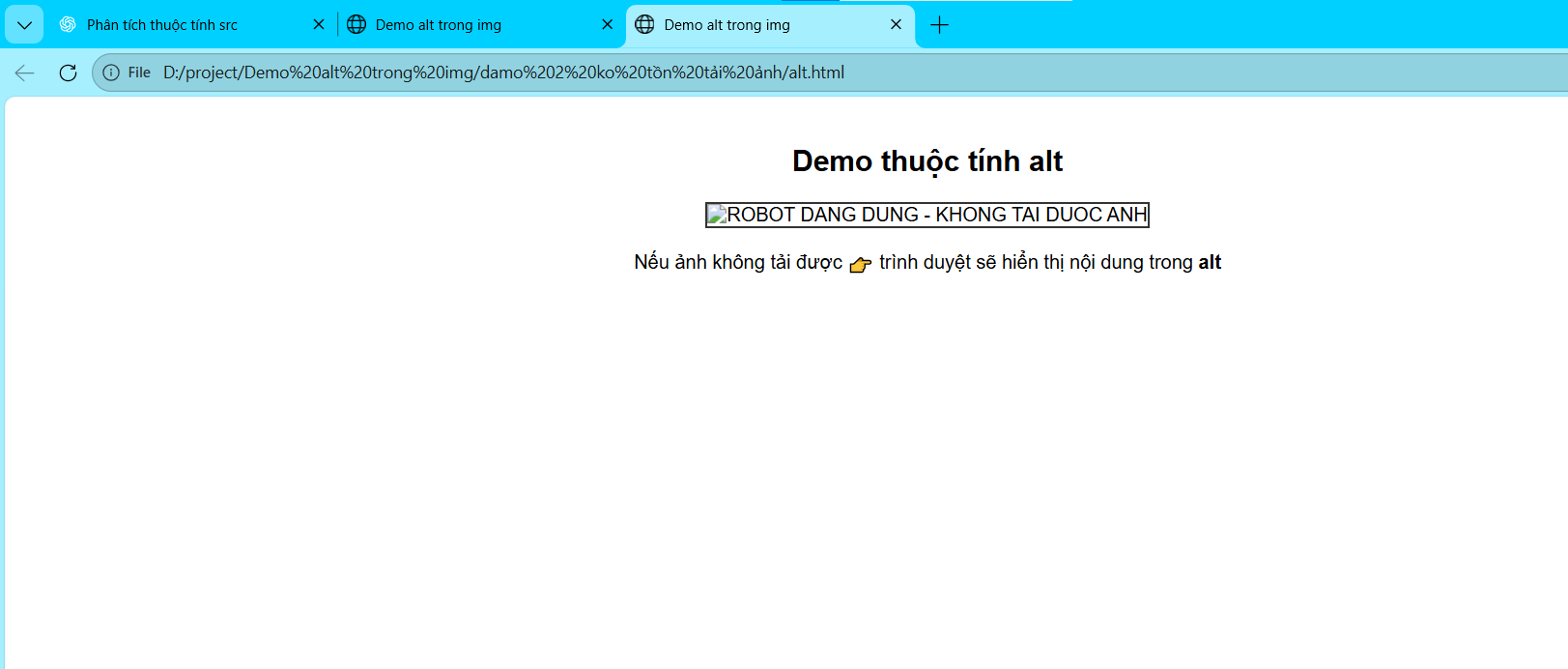
→ “alt dùng để mô tả hình ảnh khi hình không hiển thị hoặc khi được đọc”

192.10.EXAMBLE

CÓ ẢNH TỒN TẠI THÌ SẼ HIỆN NHƯ SAU



KHI KO CÓ ẢNH TỒN TẠI SẼ HIỆN : ROBOT DANG DUNG - KHONG TAI DUOC ANH



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Demo alt trong img</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      text-align: center;

      margin-top: 40px;

    }

    img {

      width: 200px;

      border: 2px solid #333;

    }

  </style>

</head>

<body>

  <h2>Demo thuộc tính alt</h2>

  <!-- CỐ TÌNH ghi sai src để thấy alt -->

  <img src="robot.webp" alt="ROBOT DANG DUNG - KHONG TAI DUOC ANH">

  <p>

    Nếu ảnh không tải được 👉 trình duyệt sẽ hiển thị nội dung trong <b>alt</b>

  </p>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[193]. Thuộc tính width

→ Quy định ĐỘ RỘNG hiển thị của phần tử (ảnh / video / bảng / iframe)

------------------------------------------------

[193.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Đặt ĐỘ RỘNG cụ thể cho phần tử

- Ý 2: Dùng khi cần khống chế kích thước hiển thị

------------------------------------------------

[193.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Hiển thị trạng thái gọn gàng

  + Căn giao diện điều khiển

  + Giữ ảnh/video không quá to

- Ví dụ ngắn:

  - width nhỏ → giao diện gọn

  - width lớn → dễ quan sát

------------------------------------------------

[193.3] PHẠM VI ÁP DỤNG

- Global: KHÔNG

- Event-only: KHÔNG

- ARIA-only: KHÔNG

- Áp cho thẻ:

  + <img>, <video>, <canvas>, <iframe>, <table>

- Có cần focus không?

  + KHÔNG

------------------------------------------------

[193.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag width="gia\_tri">

- Ví dụ đúng:

  <img src="robot.webp" width="200">

- Ví dụ SAI:

  <img width=200px>   (HTML width KHÔNG dùng px)

  <img witdh="200">   (sai tên thuộc tính)

------------------------------------------------

[193.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Số (number)

- Ví dụ hay gặp:

  + 100

  + 200

  + 640

  + 800

------------------------------------------------

[193.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt đọc HTML

2. Gặp thuộc tính width

3. Áp độ rộng cho phần tử

4. DOM giữ kích thước mới

5. Màn hình hiển thị đúng độ rộng

------------------------------------------------

[193.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt (khi parse HTML)

  + JavaScript (khi gán width)

------------------------------------------------

[193.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Nhầm width HTML với CSS width

- Lỗi 2: Ghi đơn vị px trong HTML

- Cách nhớ nhanh:

  + width HTML = SỐ

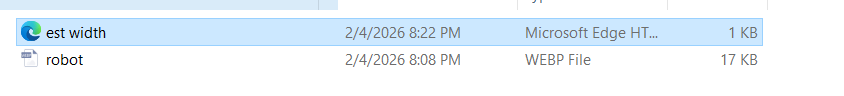
  + width CSS = có px / %

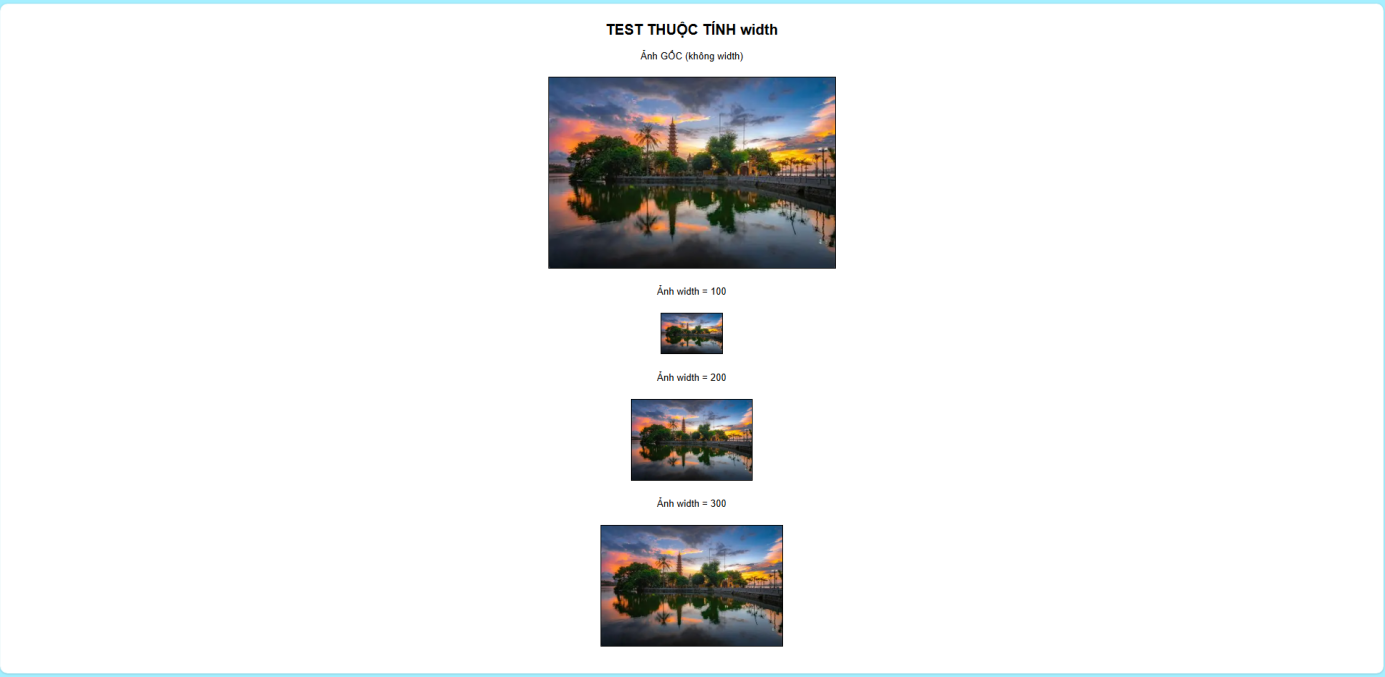
------------------------------------------------

[193.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “width dùng để đặt độ rộng phần tử khi trình duyệt hiển thị”

193.10.EXAMBLE QUY ĐỘ RỘNG CỦA ẢNH…





<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test width</title>

  <style>

    body {

      font-family: Arial;

      text-align: center;

      margin-top: 30px;

    }

    img {

      border: 2px solid black;

      margin: 10px;

    }

  </style>

</head>

<body>

  <h2>TEST THUỘC TÍNH width</h2>

  <p>Ảnh GỐC (không width)</p>

  <img src="robot.webp" alt="Robot goc">

  <p>Ảnh width = 100</p>

  <img src="robot.webp" alt="Robot nho" width="100">

  <p>Ảnh width = 200</p>

  <img src="robot.webp" alt="Robot vua" width="200">

  <p>Ảnh width = 300</p>

  <img src="robot.webp" alt="Robot lon" width="300">

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[194]. Thuộc tính height

→ Quy định ĐỘ CAO hiển thị của phần tử (ảnh / video / canvas / iframe)

------------------------------------------------

[194.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Đặt ĐỘ CAO cụ thể cho phần tử

- Ý 2: Dùng khi cần khống chế chiều cao hiển thị

------------------------------------------------

[194.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Cân đối giao diện điều khiển

  + Hiển thị camera / video đúng tỉ lệ

  + Giữ trạng thái hiển thị ổn định

- Ví dụ ngắn:

  - height nhỏ → ảnh thấp

  - height lớn → ảnh cao, dễ quan sát

------------------------------------------------

[194.3] PHẠM VI ÁP DỤNG

- Global: KHÔNG

- Event-only: KHÔNG

- ARIA-only: KHÔNG

- Áp cho thẻ:

  + <img>, <video>, <canvas>, <iframe>

- Có cần focus không?

  + KHÔNG

------------------------------------------------

[194.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag height="gia\_tri">

- Ví dụ đúng:

  <img src="robot.webp" height="150">

- Ví dụ SAI:

  <img height="150px">   (HTML height KHÔNG dùng px)

  <img heigth="150">     (sai tên thuộc tính)

------------------------------------------------

[194.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Số (number)

- Ví dụ hay gặp:

  + 100

  + 150

  + 240

  + 480

------------------------------------------------

[194.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt đọc HTML

2. Gặp thuộc tính height

3. Áp độ cao cho phần tử

4. DOM giữ kích thước mới

5. Màn hình hiển thị đúng chiều cao

------------------------------------------------

[194.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt (khi parse HTML)

  + JavaScript (khi gán height)

------------------------------------------------

[194.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Chỉ đặt height mà không đặt width → ảnh bị méo

- Lỗi 2: Nhầm height HTML với CSS height

- Cách nhớ nhanh:

  + height HTML = SỐ

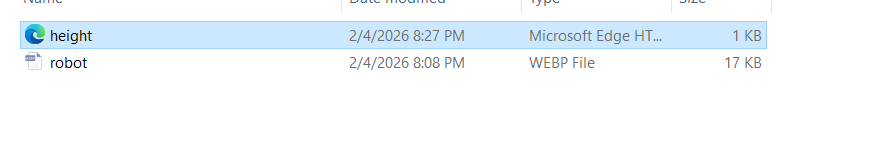
  + muốn co giãn đẹp → dùng CSS

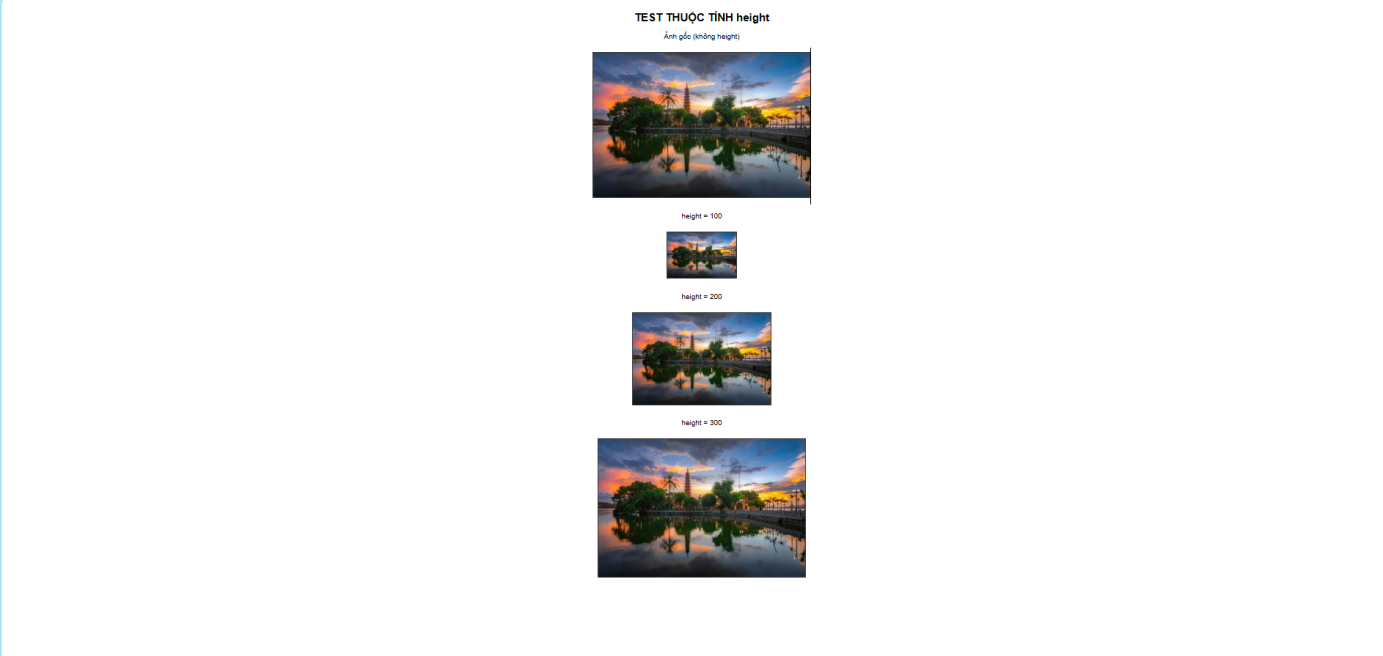
------------------------------------------------

[194.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “height dùng để đặt độ cao phần tử khi trình duyệt hiển thị”

194.10.EXAMBLE





<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test height</title>

  <style>

    body {

      font-family: Arial;

      text-align: center;

      margin-top: 30px;

    }

    img {

      border: 2px solid #333;

      margin: 10px;

    }

  </style>

</head>

<body>

  <h2>TEST THUỘC TÍNH height</h2>

  <p>Ảnh gốc (không height)</p>

  <img src="robot.webp" alt="Robot goc">

  <p>height = 100</p>

  <img src="robot.webp" alt="Robot thap" height="100">

  <p>height = 200</p>

  <img src="robot.webp" alt="Robot vua" height="200">

  <p>height = 300</p>

  <img src="robot.webp" alt="Robot cao" height="300">

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[195]. Thuộc tính loading

→ Điều khiển THỜI ĐIỂM tải tài nguyên (ảnh / iframe) để tối ưu tốc độ trang

-----------------------------------------------

[195.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Quyết định tải NGAY hay tải KHI CẦN

- Ý 2: Dùng khi trang có nhiều ảnh / nội dung dài

------------------------------------------------

[195.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Tăng tốc độ load trang

  + Giảm lag khi mở giao diện điều khiển

  + Chỉ tải camera / hình khi người dùng cuộn tới

- Ví dụ ngắn:

  - Cuộn xuống → ảnh mới tải

  - Trang mở → không bị đứng hình

------------------------------------------------

[195.3] PHẠM VI ÁP DỤNG

- Global: KHÔNG

- Event-only: KHÔNG

- ARIA-only: KHÔNG

- Áp cho thẻ:

  + <img>

  + <iframe>

- Có cần focus không?

  + KHÔNG

------------------------------------------------

[195.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag loading="gia\_tri">

- Ví dụ đúng:

  <img src="robot.webp" loading="lazy">

  <iframe src="cam.html" loading="lazy"></iframe>

- Ví dụ SAI:

  <img loading="delay">     (giá trị không tồn tại)

  <img load="lazy">         (sai tên thuộc tính)

------------------------------------------------

[195.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Chuỗi (string)

- Giá trị hợp lệ:

  + "lazy"  : tải khi cần

  + "eager": tải ngay

  + "auto" : trình duyệt tự quyết

------------------------------------------------

[195.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt đọc HTML

2. Gặp thuộc tính loading

3. Quyết định thời điểm tải tài nguyên

4. Khi tới gần vùng nhìn → bắt đầu tải

5. Nội dung hiển thị lên màn hình

------------------------------------------------

[195.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt (tự xử lý)

  + KHÔNG do người dùng trực tiếp

------------------------------------------------

[195.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Lạm dụng lazy cho ảnh trên đầu trang

- Lỗi 2: Nghĩ lazy là tải CHẬM (thực ra là tải ĐÚNG LÚC)

- Cách nhớ nhanh:

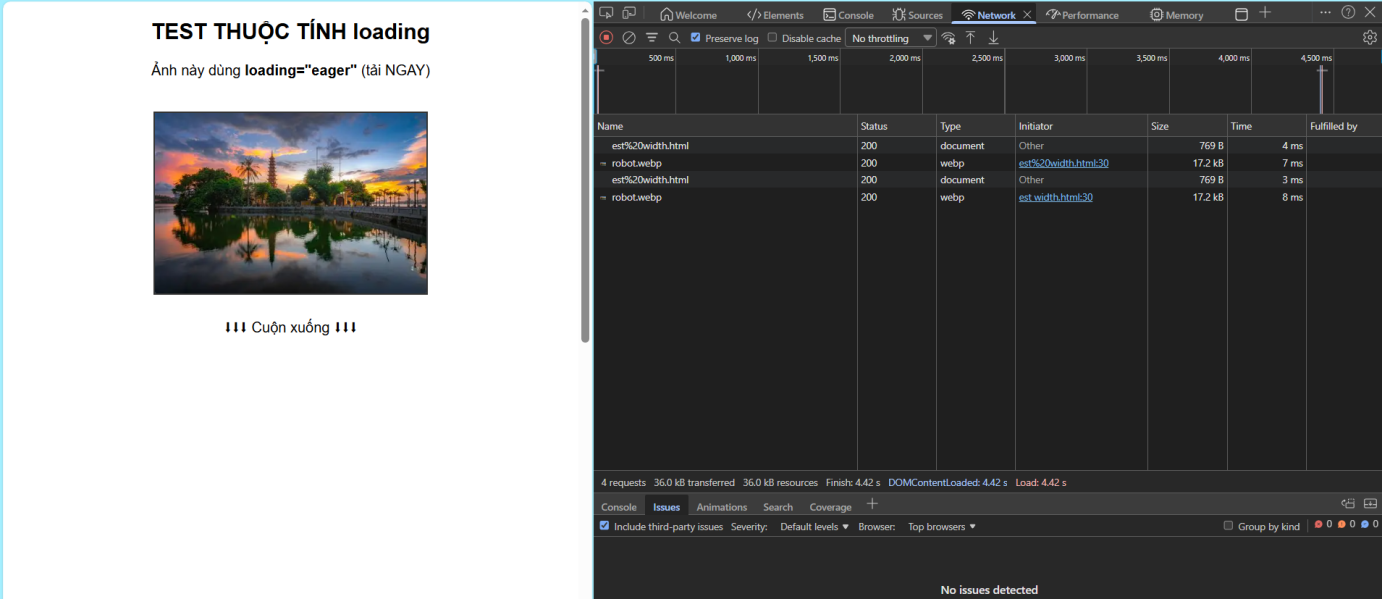
  + lazy = lười tải, đợi cần mới tải

------------------------------------------------

[195.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “loading dùng để quyết định khi nào trình duyệt tải ảnh hoặc iframe”

195.10.EXAMBLE



TRƯỜNG HỢP 1: ẢNH TRÊN (loading="eager")

Ctrl + R ngay khi trang mở

Luồng:

Trình duyệt tải HTML

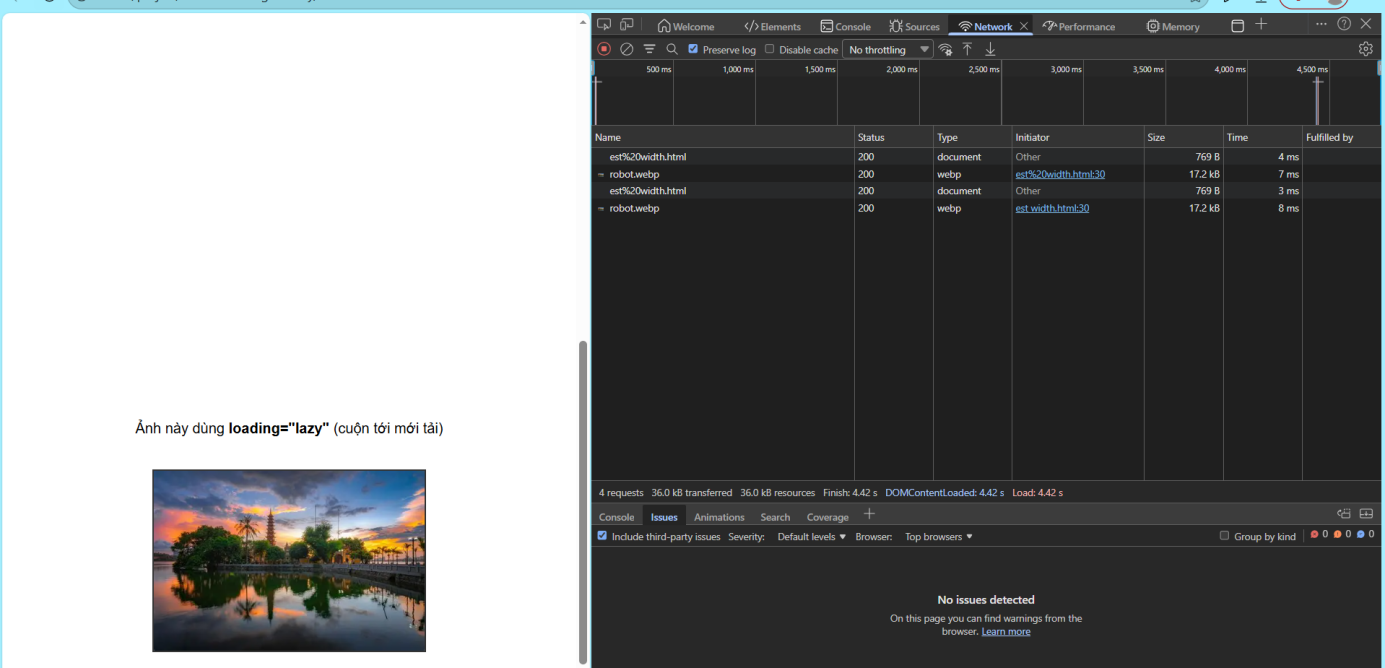
Gặp ảnh loading="eager"

TẢI NGAY

Network ghi nhận:

robot.webp: 4–7 ms

→ Ảnh luôn nằm trong danh sách request ban đầu



TRƯỜNG HỢP 2: ẢNH DƯỚI (loading="lazy")

cuộn xuống RỒI mới Ctrl + R

Luồng:

Trình duyệt tải HTML

Gặp ảnh loading="lazy"

CHƯA tải

Cuộn xuống → ảnh lọt vào viewport

LÚC NÀY MỚI GỬI REQUEST

Network ghi nhận:

robot.webp: 3–8 ms

→ Ảnh KHÔNG chiếm tài nguyên lúc load đầu

ĐIỀU QUAN TRỌNG (bạn đã hiểu đúng)

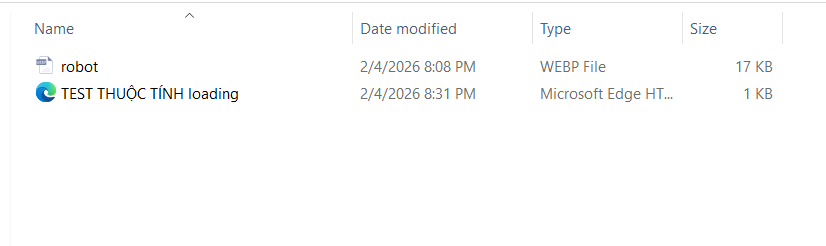
lazy KHÔNG làm ảnh tải chậm hơn

lazy làm ảnh tải ĐÚNG THỜI ĐIỂM

ms tương đương, nhưng:

eager: tải ngay

lazy: tải khi cần



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test loading lazy</title>

  <style>

    body {

      font-family: Arial;

      text-align: center;

    }

    img {

      width: 300px;

      border: 2px solid #333;

      margin: 20px 0;

    }

    .space {

      height: 800px;

    }

  </style>

</head>

<body>

  <h2>TEST THUỘC TÍNH loading</h2>

  <p>Ảnh này dùng <b>loading="eager"</b> (tải NGAY)</p>

  <img src="robot.webp" alt="Robot tai ngay" loading="eager">

  <div class="space">

    ⬇⬇⬇ Cuộn xuống ⬇⬇⬇

  </div>

  <p>Ảnh này dùng <b>loading="lazy"</b> (cuộn tới mới tải)</p>

  <img src="robot.webp" alt="Robot tai khi can" loading="lazy">

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[196]. Thuộc tính decoding

→ Điều khiển CÁCH TRÌNH DUYỆT GIẢI MÃ (decode) ảnh khi tải về.

Ảnh trên web

###

File ảnh lưu dưới dạng byte + thuật toán nén

Trình duyệt phải decode ảnh

→ mới vẽ được hình lên màn hình

Không decoding → chỉ là đống số vô nghĩa

------------------------------------------------

[196.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Quyết định ảnh được GIẢI MÃ NGAY hay GIẢI MÃ SAU

- Ý 2: Dùng khi tối ưu tốc độ hiển thị trang

------------------------------------------------

[196.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Hiển thị ảnh UI nhanh

  + Tránh lag khi load nhiều ảnh camera / dashboard

Ví dụ ngắn:

- Ảnh nút điều khiển → decode ngay

- Ảnh nền / ảnh phụ → decode sau

------------------------------------------------

[196.3] PHẠM VI ÁP DỤNG

- Loại: Attribute

- Áp cho:

  + <img>

- Không cần focus

- Không phải event

------------------------------------------------

[196.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <img decoding="giá\_trị">

- Ví dụ đúng:

  <img src="cam.jpg" decoding="async">

- Ví dụ SAI:

  <img decoding="true">

  → SAI vì decoding KHÔNG nhận true/false

------------------------------------------------

[196.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị: chuỗi

- Giá trị hợp lệ:

  + "auto"  : trình duyệt tự quyết (mặc định)

  + "sync"  : giải mã NGAY, có thể gây block

  + "async" : giải mã SAU, không block UI

------------------------------------------------

[196.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. HTML được parse

2. Trình duyệt tải file ảnh

3. decoding quyết định KHI NÀO giải mã ảnh

4. Ảnh được decode

5. Ảnh hiển thị lên màn hình

------------------------------------------------

[196.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt (render engine)

- Không liên quan trực tiếp tới JavaScript

------------------------------------------------

[196.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Tưởng decoding là lazy (SAI)

- Lỗi 2: Dùng sync cho ảnh lớn → lag trang

- Cách nhớ:

  + lazy = KHI NÀO TẢI

  + decoding = KHI NÀO GIẢI MÃ

------------------------------------------------

[196.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ decoding dùng để điều khiển THỜI ĐIỂM GIẢI MÃ ẢNH khi trình duyệt hiển thị

------------------------------------------------

[196.10] CODE VÍ DỤ THỰC TẾ

Ví dụ 1: Ảnh UI cần hiển thị mượt

<img src="button.png" decoding="sync">

Ví dụ 2: Ảnh camera / ảnh lớn

<img src="camera.jpg" loading="lazy" decoding="async">

Ví dụ 3: Mặc định trình duyệt tự xử

<img src="bg.jpg" decoding="auto">

196.11.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test decoding</title>

  <style>

    body {

      font-family: Arial;

      height: 2000px;

    }

    img {

      display: block;

      margin: 40px auto;

      width: 400px;

    }

  </style>

</head>

<body>

  <h2>Ảnh decode SYNC (giải mã NGAY)</h2>

  <img src="robot.webp" decoding="sync">

  <h2>Ảnh decode ASYNC (giải mã SAU, không block UI)</h2>

  <img src="robot.webp" decoding="async">

  <h2>Ảnh decode AUTO (trình duyệt tự quyết)</h2>

  <img src="robot.webp" decoding="auto">

</body>

</html>

========================================

=== PHÂN TÍCH THUỘC TÍNH HTML: srcset ===

========================================

[197]. Thuộc tính srcset

→ Cho trình duyệt NHIỀU nguồn ảnh để TỰ CHỌN ảnh phù hợp với màn hình & độ phân giải.

------------------------------------------------

[197.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Cung cấp NHIỀU file ảnh khác nhau cho CÙNG 1 thẻ <img>

- Ý 2: Trình duyệt tự chọn ảnh tối ưu theo kích thước màn hình / DPI

-----------------------------------------------

[197.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Hiển thị ảnh SẮC NÉT trên màn hình lớn

  + Giảm dung lượng tải trên mobile

  + Tối ưu tốc độ web

Ví dụ ngắn:

- Mobile → tải ảnh nhỏ

- Desktop → tải ảnh lớn, nét hơn

------------------------------------------------

[197.3] PHẠM VI ÁP DỤNG

- Loại: Attribute (thuộc tính HTML)

- Áp cho:

  + Thẻ <img>

- Không cần focus

- Không liên quan sự kiện chuột / phím

------------------------------------------------

[197.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <img src="anh-mac-dinh.jpg" srcset="anh1.jpg 1x, anh2.jpg 2x">

- Ví dụ đúng:

  <img src="robot-400.jpg"

       srcset="robot-400.jpg 400w, robot-800.jpg 800w"

       sizes="(max-width: 600px) 100vw, 800px">

- Ví dụ SAI:

  ❌ <img srcset="robot.jpg">

  → thiếu mô tả kích thước (w / x)

------------------------------------------------

[197.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị: Chuỗi

- Dạng giá trị:

  + width descriptor: 400w, 800w

  + pixel density: 1x, 2x

Ví dụ:

- "img-400.jpg 400w"

- "img@2x.jpg 2x"

------------------------------------------------

[197.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt phân tích HTML

2. Gặp thẻ <img> có srcset

3. Kiểm tra kích thước viewport + DPI

4. Chọn file ảnh PHÙ HỢP NHẤT

5. Tải ảnh → decode → render lên màn hình

Rất quan trọng khi debug hiệu năng ảnh

------------------------------------------------

[197.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt tự động quyết định

- JavaScript KHÔNG cần can thiệp

srcset KHÔNG phải sự kiện

------------------------------------------------

[197.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Có srcset nhưng KHÔNG có sizes → chọn ảnh sai

- Lỗi 2: File ảnh nặng nhưng đặt w nhỏ → phí băng thông

# Cách nhớ nhanh:

- src = ảnh dự phòng

- srcset = danh sách lựa chọn

------------------------------------------------

[197.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “srcset dùng để cho trình duyệt chọn ảnh phù hợp khi kích thước màn hình thay đổi”

197.10.EXAMBLE srcset theo DPI



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test srcset cơ bản</title>

</head>

<body>

  <h3>Test srcset theo DPI</h3>

  <img src="robot.webp" srcset="

    robot.webp 1x,

    robot.webp 2x

  " alt="Ảnh test srcset">

  <p>

    - Màn hình thường → tải img-400.jpg<br>

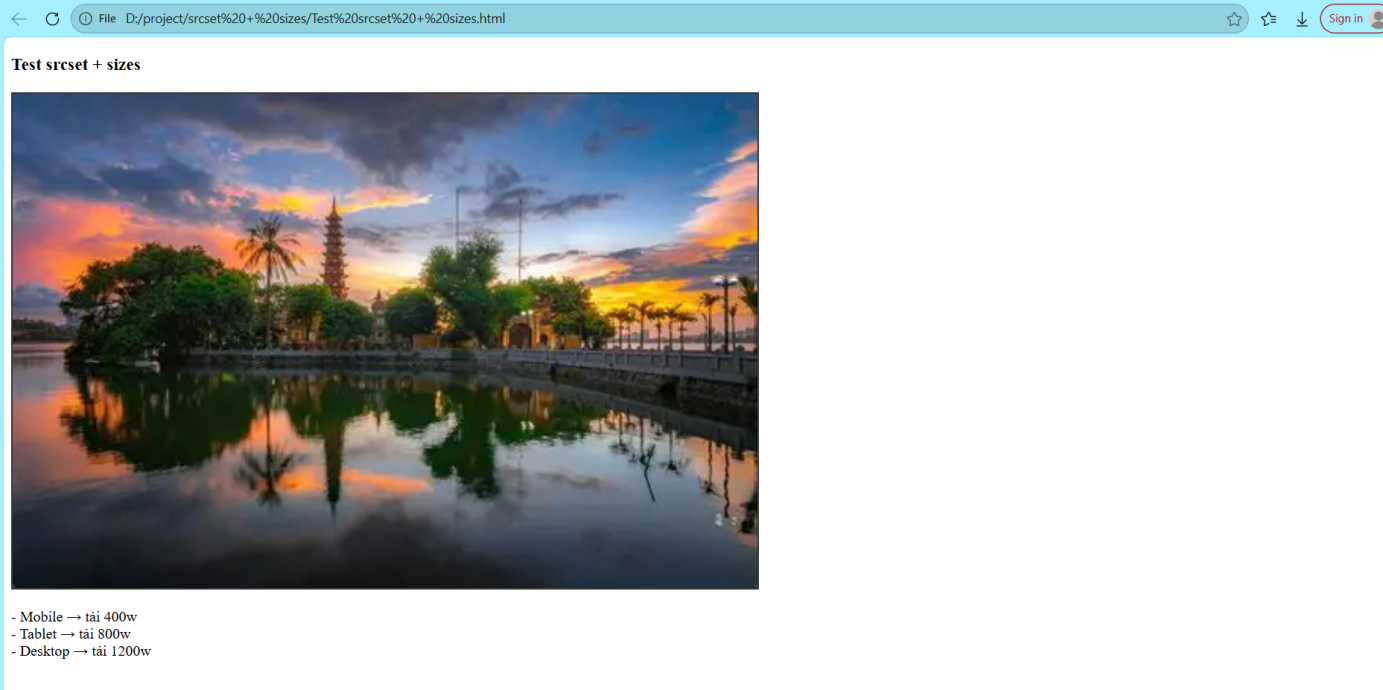
    - Màn hình Retina / 2x → tải img-800.jpg

  </p>

</body>

</html>

197.11.EXAMBLE Test srcset + sizes



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test srcset + sizes</title>

  <style>

    img {

      width: 100%;

      max-width: 800px;

      border: 2px solid #333;

    }

  </style>

</head>

<body>

  <h3>Test srcset + sizes</h3>

  <img src="robot.webp" srcset="

    robot.webp 400w,

    robot.webp 800w,

    robot.webp 1200w

  " sizes="

    (max-width: 600px) 100vw,

    (max-width: 900px) 80vw,

    800px

  " alt="Ảnh responsive">

  <p>

    - Mobile → tải 400w<br>

    - Tablet → tải 800w<br>

    - Desktop → tải 1200w

  </p>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[NC SAU]

[198]. Thuộc tính crossorigin

→ Quyết định việc trình duyệt CÓ / KHÔNG gửi thông tin xác thực khi tải tài nguyên từ domain khác.

------------------------------------------------

[198.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Cho phép trình duyệt tải tài nguyên TỪ DOMAIN KHÁC mà không bị chặn CORS

- Ý 2: Kiểm soát có gửi cookie / token khi tải ảnh, script, font hay không

------------------------------------------------

[198.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Load ảnh CDN

  + Load script / font từ server khác

  + Đo hiệu năng, canvas, WebGL không bị lỗi bảo mật

Ví dụ ngắn:

- Load ảnh từ CDN → KHÔNG lỗi CORS

- Dùng ảnh vẽ lên canvas → KHÔNG bị "tainted canvas"

------------------------------------------------

[198.3] PHẠM VI ÁP DỤNG

- Global: ❌

- Event-only: ❌

- ARIA-only: ❌

- Attribute riêng cho tài nguyên cross-domain

- Áp cho thẻ:

  + <img>

  + <script>

  + <link> (font, css)

  + <audio>, <video>

- Có cần focus không?

  → ❌ Không liên quan focus

------------------------------------------------

[198.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag crossorigin="giá\_trị">

- Ví dụ ĐÚNG:

  <img src="https://cdn.example.com/a.jpg" crossorigin="anonymous">

- Ví dụ SAI:

  <img crossorigin>

  → ❌ Thiếu giá trị → browser bỏ qua

------------------------------------------------

[198.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị: Chuỗi (string)

- Giá trị hợp lệ:

  + "anonymous"

  + "use-credentials"

- Ý nghĩa:

  + anonymous → KHÔNG gửi cookie / token

  + use-credentials → CÓ gửi cookie / token

-----------------------------------------------

[198.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. HTML gặp thẻ <img> / <script> / <link>

2. Trình duyệt thấy có crossorigin

3. Gửi request sang domain khác

4. Server trả header CORS (Access-Control-Allow-Origin)

5. Tài nguyên được load hoặc bị chặn

6. Màn hình hiển thị / JS hoạt động

Dùng mục này để DEBUG lỗi CORS

------------------------------------------------

[198.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt (khi tải tài nguyên)

- Người dùng: ❌

- JavaScript: ❌ (chỉ gián tiếp)

Hiểu nhanh: Browser là thằng quyết định

-----------------------------------------------

[198.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Server KHÔNG bật CORS → vẫn lỗi dù có crossorigin

- Lỗi 2: Dùng canvas + ảnh CDN mà quên crossorigin → canvas bị khóa

- Cách nhớ nhanh:

  → Ảnh / script khác domain = nghĩ tới crossorigin

------------------------------------------------

[198.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “crossorigin dùng để cho phép tải tài nguyên từ domain khác khi trình duyệt gửi request”

198.10.EXXMABLE

<!DOCTYPE html>

<html lang="vi">

<head>

<!--

crossorigin với <img> (hay dùng nhất)

-->

  <meta charset="UTF-8">

  <title>Test crossorigin</title>

</head>

<body>

<h3>Ảnh CÓ crossorigin</h3>

<img

  src="https://picsum.photos/300/200"

  crossorigin="anonymous"

  alt="test image">

<hr>

<h3>Ảnh KHÔNG crossorigin</h3>

<img

  src="https://picsum.photos/300/200"

  alt="test image">

</body>

</html>

198.11.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

<!--

crossorigin với <script> (CDN)

-->

  <script

    src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"

    crossorigin="anonymous">

  </script>

</head>

<body>

  <button onclick="test()">Test Axios</button>

  <script>

    function test() {

      console.log("Axios loaded:", typeof axios);

    }

  </script>

</body>

</html>

198.12.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <!--

crossorigin với <link> (CSS từ CDN)

-->

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/normalize/8.0.1/normalize.min.css"

    crossorigin="anonymous">

</head>

<body>

  <h1>Hello crossorigin</h1>

</body>

</html>

HẾT PHẦN

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[199]. Thuộc tính controls

→ Bật giao diện điều khiển (play / pause / volume / timeline) cho media (audio, video)

------------------------------------------------

[199.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Hiển thị thanh điều khiển mặc định của trình duyệt

- Ý 2: Cho người dùng tự phát / dừng / chỉnh âm lượng

------------------------------------------------

[199.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Phát nhạc nền

  + Xem video hướng dẫn

  + Kiểm tra âm thanh thiết bị

Ví dụ ngắn:

- Người dùng bấm ▶ → video chạy

- Kéo timeline → tua nhanh video

------------------------------------------------

[199.3] PHẠM VI ÁP DỤNG

- Global / Event-only / ARIA-only: ❌

- Áp cho:

  + <audio>

  + <video>

- Có cần focus không?

  + ❌ Không cần (chuột là chính)

------------------------------------------------

[199.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag controls>

- Ví dụ đúng:

  <video src="movie.mp4" controls></video>

- Ví dụ SAI:

  <video controls="true"></video>

  → ❌ controls là boolean, KHÔNG cần giá trị

------------------------------------------------

[199.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Boolean attribute

- Giá trị hợp lệ:

  + Có controls → bật

  + Không có → tắt

------------------------------------------------

[199.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trang load

2. Trình duyệt thấy thẻ audio/video có controls

3. Trình duyệt tự render thanh điều khiển

4. Người dùng thao tác (play / pause / volume)

5. Media phát hoặc dừng → âm thanh / hình ảnh thay đổi

Giúp DEBUG: Nếu KHÔNG thấy thanh điều khiển → thiếu controls

------------------------------------------------

[199.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute (boolean)

- Ai kích hoạt:

  + Trình duyệt (render UI)

  + Người dùng (click nút điều khiển)

JS KHÔNG cần can thiệp vẫn chạy

------------------------------------------------

[199.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Ghi controls="true" → không cần

- Lỗi 2: Quên controls → không thấy thanh phát

- Cách nhớ nhanh:

  → Thấy video mà không bấm được ▶ là thiếu controls

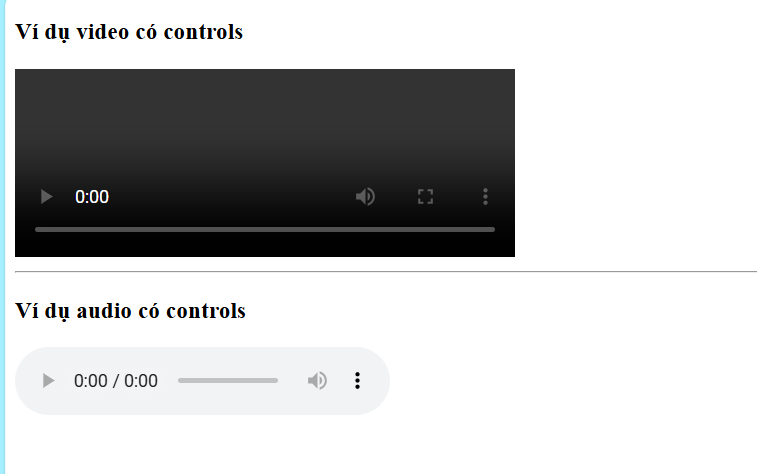
------------------------------------------------

[199.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “controls dùng để HIỆN thanh điều khiển khi media được hiển thị”

▶ ⏸ 🔊 ⏩

199.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ controls</title>

</head>

<body>

  <h3>Ví dụ video có controls</h3>

  <video src="video.mp4" width="400" controls>

    Trình duyệt không hỗ trợ video

  </video>

  <hr>

  <h3>Ví dụ audio có controls</h3>

  <audio src="music.mp3" controls>

    Trình duyệt không hỗ trợ audio

  </audio>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[ 200 ]. Thuộc tính autoplay

→ Điều khiển việc media (audio / video) TỰ ĐỘNG PHÁT khi trang vừa load.

------------------------------------------------

[200.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Tự động phát audio / video ngay khi trang tải xong

- Ý 2: Dùng cho nhạc nền, video intro, màn hình quảng cáo

------------------------------------------------

[200.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Phát nhạc nền website

  + Video quảng cáo tự chạy

  + Màn hình giám sát / camera

Ví dụ ngắn:

- Trang load → video chạy ngay

- Không cần người dùng bấm ▶

------------------------------------------------

[200.3] PHẠM VI ÁP DỤNG

- Loại: Attribute (boolean)

- Áp cho:

  + <video>

  + <audio>

- Không cần focus

- BỊ HẠN CHẾ nếu KHÔNG có muted

------------------------------------------------

[200.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag autoplay>

- Ví dụ đúng:

  <video autoplay muted></video>

- Ví dụ SAI:

  <video autoplay="true"></video>  ← sai bản chất boolean

------------------------------------------------

[200.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Boolean attribute (có là bật)

- Giá trị hay gặp:

  + autoplay

  + autoplay + muted (chuẩn nhất)

------------------------------------------------

[200.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng mở trang

2. Trình duyệt gặp thẻ video/audio có autoplay

3. Trình duyệt kiểm tra có muted không

4. Nếu hợp lệ → media tự phát

5. Âm thanh / hình ảnh xuất hiện ngay

------------------------------------------------

[200.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt (khi load trang)

- Người dùng KHÔNG cần tương tác

------------------------------------------------

[200.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: autoplay KHÔNG chạy vì thiếu muted

- Lỗi 2: trình duyệt chặn autoplay có âm thanh

- Cách nhớ nhanh:

  → Autoplay MUỐN SỐNG thì PHẢI muted

------------------------------------------------

[200.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “autoplay dùng để tự phát media khi trang vừa load xong”

200.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ autoplay</title>

</head>

<body>

  <h3>Video autoplay (CHẠY ĐƯỢC)</h3>

  <video src="video.mp4" width="400" autoplay muted loop>

    Trình duyệt không hỗ trợ video

  </video>

  <hr>

  <h3>Audio autoplay (CHẠY ĐƯỢC)</h3>

  <audio src="music.mp3" autoplay muted loop>

    Trình duyệt không hỗ trợ audio

  </audio>

</body>

</html>

========================================

=== PHÂN TÍCH THUỘC TÍNH HTML: loop ===

========================================

[201]. Thuộc tính loop

  → Bắt phần tử media (audio / video) phát lại từ đầu khi chạy tới cuối.

  ------------------------------------------------

  [201.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

  - Ý 1: Làm cho video / audio phát LẶP VÔ HẠN.

  - Ý 2: Dùng khi cần nhạc nền, video demo chạy liên tục.

  ------------------------------------------------

  [201.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

  - Dùng để:

  + Phát nhạc nền website

  + Video quảng cáo chạy vòng lặp

  + Màn hình trạng thái / kiosk

  Ví dụ ngắn:

  - Hết bài nhạc → tự phát lại

  - Hết video → quay lại từ đầu

  ------------------------------------------------

  [201.3] PHẠM VI ÁP DỤNG

  - Event-only: ❌

  - ARIA-only: ❌

  - Attribute: ✅

  Áp cho:

  - <audio>

    - <video>

      Có cần focus không?

      - ❌ Không cần focus

      ------------------------------------------------

      [201.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

      - Cú pháp chuẩn:

      <video loop></video>

      - Ví dụ đúng:

      <audio src="music.mp3" loop></audio>

      - Ví dụ SAI:

      <video loop="true"></video>

      → SAI vì loop là boolean, KHÔNG cần giá trị

      ------------------------------------------------

      [201.5] GIÁ TRỊ THƯỜNG DÙNG

      - Kiểu giá trị:

      + Boolean attribute

      - Giá trị hợp lệ:

      + Có loop → BẬT

      + Không có loop → TẮT

      ------------------------------------------------

      [201.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

      1. Người dùng mở trang

      2. Media bắt đầu phát

      3. Media chạy tới cuối

      4. Trình duyệt kiểm tra có loop không

      5. Có → quay về thời điểm 0

      6. Media phát lại từ đầu

      ------------------------------------------------

      [201.7] BẢN CHẤT HOẠT ĐỘNG

      - Là: Attribute

      - Ai kích hoạt:

      + Trình duyệt (khi media kết thúc)

      → KHÔNG phải do click

      → KHÔNG phải do JavaScript (trừ khi JS thêm loop)

      ------------------------------------------------

      [201.8] LƯU Ý / LỖI HAY GẶP

      - Lỗi 1: Dùng loop cho <img> → KHÔNG TÁC DỤNG

      - Lỗi 2: Nghĩ loop là sự kiện → SAI

      - Cách nhớ nhanh:

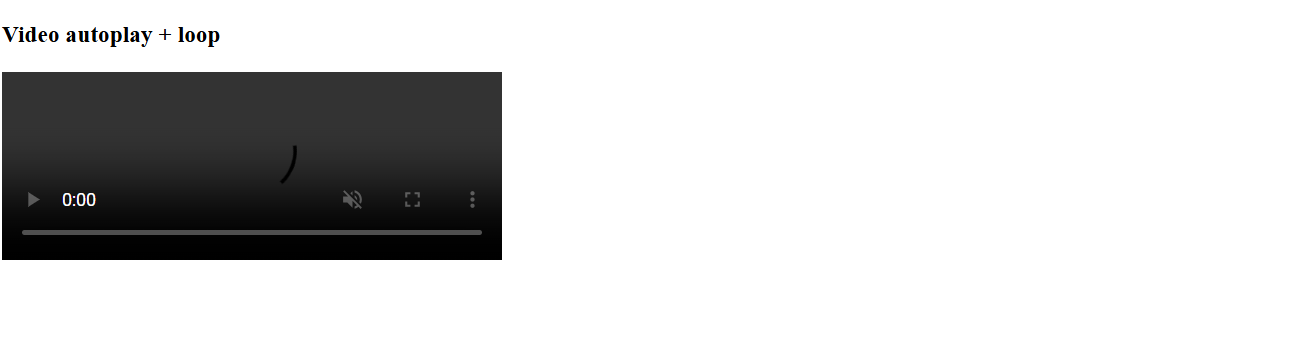
      → loop = hết là quay lại

      ------------------------------------------------

      [201.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

      → “Thuộc tính loop dùng để phát lại media khi media chạy tới cuối”

2001.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Autoplay + Loop</title>

</head>

<body>

  <h3>Video autoplay + loop</h3>

  <video src="video.mp4" width="400" autoplay loop muted controls>

    Trình duyệt không hỗ trợ video

  </video>

</body>

</html>

========================================

=== PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[202]. Thuộc tính muted

→ Tắt tiếng media (audio / video) ngay từ lúc trình duyệt bắt đầu phát.

-----------------------------------------------

[202.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: TẮT ÂM THANH của audio / video

- Ý 2: Dùng để CHO PHÉP autoplay không bị trình duyệt chặn

------------------------------------------------

[202.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Video nền website chạy tự động

  + Demo robot / xe / ESP không gây ồn

  + Preview media không cần tiếng

Ví dụ ngắn:

- Trang load → video chạy luôn nhưng KHÔNG CÓ TIẾNG

- Người dùng muốn nghe → tự bật tiếng bằng tay

------------------------------------------------

[202.3] PHẠM VI ÁP DỤNG

- Loại: Global attribute (áp trực tiếp trên thẻ)

- Áp cho:

  + <video>

  + <audio>

- Không cần focus

------------------------------------------------

[202.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <video muted></video>

  <audio muted></audio>

- Ví dụ đúng:

  <video src="demo.mp4" autoplay muted></video>

- Ví dụ SAI:

  <video muted="false"></video>  ❌ (vẫn bị coi là bật muted)

------------------------------------------------

[202.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Boolean (có / không)

- Giá trị hay gặp:

  + muted

  + muted=""

Có là TRUE, không có là FALSE

------------------------------------------------

[202.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trang được load

2. Trình duyệt thấy thẻ video/audio có muted

3. Media được phát nhưng âm thanh = 0

4. Autoplay được CHO PHÉP

5. Màn hình có hình ảnh nhưng không có tiếng

Rất dễ DEBUG autoplay bị chặn

------------------------------------------------

[202.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt (khi parse HTML)

- JavaScript có thể bật/tắt lại:

  video.muted = true / false

------------------------------------------------

[202.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Quên muted → autoplay KHÔNG chạy

- Lỗi 2: Dùng muted="false" nhưng vẫn bị tắt tiếng

- Cách nhớ nhanh:

  Autoplay = PHẢI muted

------------------------------------------------

[202.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “muted dùng để tắt tiếng media khi trang vừa load, giúp autoplay chạy an toàn 🔇”

202.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Ví dụ muted</title>

</head>

<body>

  <h3>Video autoplay + loop + muted</h3>

  <video src="demo.mp4" autoplay loop muted width="320" controls>

  </video>

</body>

</html>

202.11.EXABLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test playsinline</title>

</head>

<body>

  <h3>Video playsinline</h3>

  <video src="demo.mp4" width="320" autoplay muted loop playsinline controls>

  </video>

</body>

</html>

202.12.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test poster</title>

</head>

<body>

  <h3>Video có poster</h3>

  <video src="demo.mp4" width="360" controls poster="poster.jpg">

  </video>

</body>

</html>

========================================

=== PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[203]. Thuộc tính colspan

→ Gộp NHIỀU CỘT trong bảng (<table>) thành MỘT Ô DUY NHẤT.

------------------------------------------------

[203.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: GỘP nhiều cột lại thành 1 ô

- Ý 2: Dùng khi cần HIỂN THỊ nội dung trải ngang bảng

------------------------------------------------

[203.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Gộp tiêu đề bảng

  + Hiển thị trạng thái chung

  + Layout bảng điều khiển

Ví dụ ngắn:

- Header bảng → chiếm 3 cột

- Thông báo → gộp toàn bộ hàng

------------------------------------------------

[203.3] PHẠM VI ÁP DỤNG

- Loại: Table-only attribute

- Áp cho:

  + <td>

  + <th>

- Không cần focus

------------------------------------------------

[203.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <td colspan="số\_cột">

- Ví dụ đúng:

  <th colspan="3">Tiêu đề</th>

- Ví dụ SAI:

  <tr colspan="3"></tr>  ❌ (tr KHÔNG hỗ trợ colspan)

------------------------------------------------

[203.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Số nguyên dương

- Ví dụ giá trị hay gặp:

  + 2

  + 3

  + 4

Số cột gộp = số cột bảng cần chiếm

------------------------------------------------

[203.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt parse <table>

2. Gặp <td>/<th> có colspan

3. Trình duyệt gộp số cột tương ứng

4. Layout bảng được tính lại

5. Nội dung hiển thị trải ngang

Giúp DEBUG lỗi lệch bảng

------------------------------------------------

[203.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt (khi render bảng)

- JavaScript có thể thay đổi:

  cell.colSpan = 3

------------------------------------------------

[203.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: colspan vượt quá số cột → bảng lệch

- Lỗi 2: Gộp colspan nhưng quên giảm số <td> khác

- Cách nhớ nhanh:

  colspan = GỘP NGANG

------------------------------------------------

[203.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “colspan dùng để gộp nhiều cột trong bảng thành một ô duy nhất ⬅➡”

203.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test colspan</title>

  <style>

    table {

      border-collapse: collapse;

      width: 400px;

    }

    td,

    th {

      border: 1px solid #333;

      padding: 8px;

      text-align: center;

    }

    th {

      background: #eee;

    }

  </style>

</head>

<body>

  <h3>Ví dụ colspan</h3>

  <table>

    <tr>

      <th colspan="3">BẢNG ĐIỂM LỚP A</th>

    </tr>

    <tr>

      <th>Toán</th>

      <th>Lý</th>

      <th>Hóa</th>

    </tr>

    <tr>

      <td>8</td>

      <td>7</td>

      <td>9</td>

    </tr>

    <tr>

      <td colspan="3">👉 Đạt yêu cầu</td>

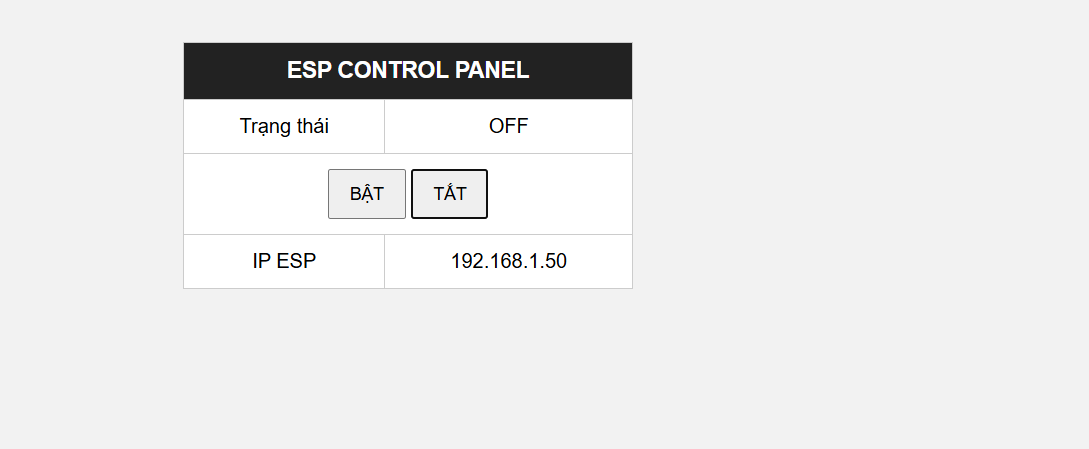
    </tr>

  </table>

</body>

</html>

203.11.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP Control Panel</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      background: #f2f2f2;

    }

    table {

      width: 360px;

      margin: 40px auto;

      border-collapse: collapse;

      background: #fff;

    }

    th,

    td {

      border: 1px solid #ccc;

      padding: 12px;

      text-align: center;

    }

    th {

      background: #222;

      color: #fff;

      font-size: 18px;

    }

    button {

      padding: 10px 16px;

      font-size: 14px;

      cursor: pointer;

    }

  </style>

</head>

<body>

  <table>

    <!-- HÀNG TIÊU ĐỀ GỘP 2 CỘT -->

    <tr>

      <th colspan="2">ESP CONTROL PANEL</th>

    </tr>

    <!-- TRẠNG THÁI -->

    <tr>

      <td>Trạng thái</td>

      <td id="status">OFF</td>

    </tr>

    <!-- NÚT ĐIỀU KHIỂN GỘP 2 CỘT -->

    <tr>

      <td colspan="2">

        <button onclick="sendCmd('ON')">BẬT</button>

        <button onclick="sendCmd('OFF')">TẮT</button>

      </td>

    </tr>

    <!-- THÔNG TIN ESP -->

    <tr>

      <td>IP ESP</td>

      <td>192.168.1.50</td>

    </tr>

  </table>

  <script>

    function sendCmd(cmd) {

      // Gửi lệnh về ESP

      fetch('/cmd?value=' + cmd);

      // Cập nhật giao diện

      document.getElementById('status').innerText = cmd;

    }

  </script>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[204]. Thuộc tính rowspan

→ Dùng để GỘP NHIỀU HÀNG (row) lại thành MỘT Ô trong bảng.

------------------------------------------------

[204.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Gộp nhiều hàng dọc thành 1 ô duy nhất.

- Ý 2: Dùng khi một dữ liệu áp dụng cho NHIỀU dòng liên tiếp.

------------------------------------------------

[204.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Gộp cột TÊN THIẾT BỊ trong bảng ESP

  + Hiển thị 1 trạng thái cho nhiều cảm biến

  + Giao diện điều khiển gọn hơn

Ví dụ ngắn:

- ESP1 → áp dụng cho 3 cảm biến

- Ô "ESP1" gộp 3 hàng cảm biến

------------------------------------------------

[204.3] PHẠM VI ÁP DỤNG

- Loại: Table-only

- Áp cho:

  + <td>

  + <th>

- Không cần focus

- Chỉ dùng trong <table>

------------------------------------------------

[204.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <td rowspan="số\_hàng">

- Ví dụ đúng:

  <td rowspan="3">ESP32</td>

- Ví dụ SAI:

  <div rowspan="2">Sai</div>

  → rowspan KHÔNG dùng cho div

------------------------------------------------

[204.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Số nguyên dương

- Ví dụ hay gặp:

  + rowspan="2"

  + rowspan="3"

  + rowspan="5"

------------------------------------------------

[204.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt đọc cấu trúc <table>

2. Gặp ô có rowspan

3. Ô đó chiếm N hàng liên tiếp

4. Các ô bên dưới bị đẩy sang cột khác

5. Bảng hiển thị gộp theo chiều DỌC

------------------------------------------------

[204.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute (thuộc tính HTML)

- Ai kích hoạt:

  + Trình duyệt (khi render bảng)

- KHÔNG phải event

- KHÔNG cần JavaScript

------------------------------------------------

[204.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: rowspan không khớp số hàng → bảng lệch cột

- Lỗi 2: Quên bỏ <td> ở các hàng bị gộp

- Cách nhớ:

  → rowspan = gộp THEO CHIỀU DỌC

------------------------------------------------

[204.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “rowspan dùng để gộp nhiều HÀNG thành một ô khi bảng có dữ liệu chung theo chiều dọc”

204.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP Web UI - rowspan</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      background: #111;

      color: #fff;

    }

    table {

      border-collapse: collapse;

      width: 100%;

      margin-top: 20px;

    }

    th,

    td {

      border: 1px solid #555;

      padding: 10px;

      text-align: center;

    }

    th {

      background: #222;

    }

  </style>

</head>

<body>

  <h2>📡 BẢNG TRẠNG THÁI ESP32</h2>

  <table>

    <tr>

      <th>Thiết bị</th>

      <th>Cảm biến</th>

      <th>Giá trị</th>

    </tr>

    <tr>

      <!-- rowspan gộp 3 hàng cho ESP32 -->

      <td rowspan="3">ESP32</td>

      <td>Nhiệt độ</td>

      <td>32 °C</td>

    </tr>

    <tr>

      <td>Độ ẩm</td>

      <td>68 %</td>

    </tr>

    <tr>

      <td>Ánh sáng</td>

      <td>450 lux</td>

    </tr>

  </table>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[205]. Thuộc tính scope

→ XÁC ĐỊNH ô <th> này ĐANG ĐẠI DIỆN cho hàng hay cột nào trong bảng

------------------------------------------------

[205.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Giúp trình duyệt & screen reader hiểu tiêu đề này ÁP cho hàng hay cột nào

- Ý 2: Dùng khi bảng có nhiều hàng / cột để KHÔNG NHẦM tiêu đề

------------------------------------------------

[205.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Hiển thị bảng trạng thái ESP

  + Phân biệt rõ cảm biến / thiết bị / giá trị

  + Hỗ trợ truy cập (accessibility)

Ví dụ ngắn (ESP):

- Cột "Nhiệt độ" → scope="col"

- Hàng "ESP32-1" → scope="row"

------------------------------------------------

[205.3] PHẠM VI ÁP DỤNG

- Loại: Table-only

- Áp cho:

  + Thẻ <th>

- Không cần focus

- Không kích hoạt bằng click / phím

------------------------------------------------

[205.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <th scope="giá\_trị">

- Ví dụ đúng:

  <th scope="col">Nhiệt độ</th>

  <th scope="row">ESP32</th>

- Ví dụ SAI:

  <td scope="row">ESP32</td>  ❌ (scope CHỈ dùng cho th)

------------------------------------------------

[205.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị: Chuỗi

- Giá trị hay gặp:

  + col      → tiêu đề cho CỘT

  + row      → tiêu đề cho HÀNG

  + colgroup → nhóm cột

  + rowgroup → nhóm hàng

------------------------------------------------

[205.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Trình duyệt load bảng HTML

2. Gặp th có scope

3. Ghi nhớ: ô này đại diện cho hàng / cột nào

4. Khi đọc bảng (screen reader):

   → ghép đúng tiêu đề + ô dữ liệu

5. Giao diện KHÔNG đổi, nhưng logic hiểu bảng CHÍNH XÁC

Mục này giúp DEBUG bảng phức tạp

------------------------------------------------

[205.7] BẢN CHẤT HOẠT ĐỘNG

- Là: Attribute

- Ai kích hoạt:

  + Trình duyệt

  + Công cụ hỗ trợ (screen reader)

- KHÔNG phải event

- KHÔNG chạy JavaScript

------------------------------------------------

[205.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Dùng scope cho <td> ❌

- Lỗi 2: Bảng phức tạp nhưng KHÔNG khai báo scope → khó hiểu

- Cách nhớ nhanh:

  → scope = "TH này ĐẠI DIỆN cho AI?"

------------------------------------------------

[205.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “scope dùng để chỉ rõ tiêu đề bảng áp cho hàng hay cột khi hiển thị dữ liệu”

------------------------------------------------

VÍ DỤ GIAO DIỆN ESP (THỰC TẾ)

------------------------------------------------

<table border="1">

  <tr>

    <th scope="col">Thiết bị</th>

    <th scope="col">Nhiệt độ</th>

    <th scope="col">Độ ẩm</th>

  </tr>

  <tr>

    <th scope="row">ESP32-1</th>

    <td>32 °C</td>

    <td>68 %</td>

  </tr>

  <tr>

    <th scope="row">ESP32-2</th>

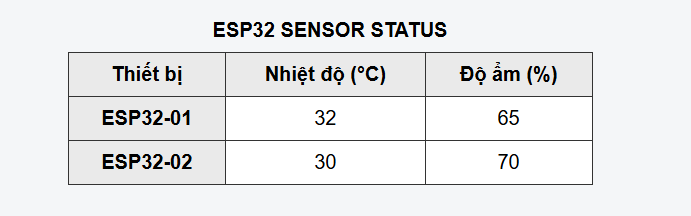
    <td>30 °C</td>

    <td>70 %</td>

  </tr>

</table>

205.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP Dashboard</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      background: #f4f6f8;

    }

    table {

      border-collapse: collapse;

      width: 420px;

      margin: 20px auto;

      background: #fff;

    }

    th,

    td {

      border: 1px solid #333;

      padding: 8px;

      text-align: center;

    }

    th {

      background: #eaeaea;

    }

    caption {

      font-weight: bold;

      margin-bottom: 8px;

    }

  </style>

</head>

<body>

  <table>

    <caption>ESP32 SENSOR STATUS</caption>

    <tr>

      <th scope="col">Thiết bị</th>

      <th scope="col">Nhiệt độ (°C)</th>

      <th scope="col">Độ ẩm (%)</th>

    </tr>

    <tr>

      <th scope="row">ESP32-01</th>

      <td>32</td>

      <td>65</td>

    </tr>

    <tr>

      <th scope="row">ESP32-02</th>

      <td>30</td>

      <td>70</td>

    </tr>

  </table>

</body>

</html>

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[206]. Thuộc tính aria-orientation

-Chủ yếu dùng phím để điều khiển góc xoay

→ Xác định HƯỚNG hiển thị / hướng hoạt động của một thành phần ARIA (ngang hay dọc).

------------------------------------------------

[206.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Báo cho trình đọc màn hình biết phần tử đang nằm NGANG hay DỌC

- Ý 2: Giúp người dùng dùng PHÍM ĐIỀU HƯỚNG đúng cách (trái-phải hoặc lên-xuống)

------------------------------------------------

[206.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Hiển thị trạng thái điều khiển

  + Điều hướng bằng phím chính xác

  + Hỗ trợ người khiếm thị

Ví dụ ngắn:

- Thanh trượt âm lượng nằm NGANG → mũi tên trái / phải

- Menu xếp DỌC → mũi tên lên / xuống

------------------------------------------------

[206.3] PHẠM VI ÁP DỤNG

- ARIA-only

- Áp cho:

  + slider

  + scrollbar

  + tablist

  + menu

- Có cần focus không?

  + CÓ (muốn dùng phím thì phần tử phải focus)

------------------------------------------------

[206.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag aria-orientation="giá\_trị">

- Ví dụ đúng:

  <div role="slider" aria-orientation="horizontal">

- Ví dụ SAI:

  aria-orientation="ngang"

  → SAI vì chỉ nhận giá trị CHUẨN, không nhận tiếng Việt

------------------------------------------------

[206.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + Chuỗi cố định

- Giá trị hợp lệ:

  + "horizontal"  (ngang)

  + "vertical"    (dọc)

------------------------------------------------

[206.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng TAB focus vào slider / menu

2. Trình duyệt đọc aria-orientation

3. Trình đọc màn hình hiểu hướng hoạt động

4. Người dùng bấm phím mũi tên phù hợp

5. Giá trị / trạng thái thay đổi đúng hướng

→ Giúp DEBUG lỗi điều hướng bằng phím

------------------------------------------------

[206.7] BẢN CHẤT HOẠT ĐỘNG

- Là: ARIA attribute

- Ai kích hoạt:

  + Trình duyệt + trình đọc màn hình

- JavaScript:

  + KHÔNG tự chạy

  + Chỉ ĐỌC thông tin để xử lý logic

------------------------------------------------

[206.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Gắn aria-orientation cho thẻ KHÔNG có role phù hợp

- Lỗi 2: Giao diện nằm ngang nhưng khai báo "vertical"

- Cách nhớ nhanh:

  → GIAO DIỆN XOAY SAO → ORIENTATION KHAI BÁO Y VẬY

------------------------------------------------

[206.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “aria-orientation dùng để khai báo HƯỚNG hoạt động (ngang / dọc) cho thành phần ARIA khi người dùng điều hướng bằng phím”

206.10.EXAMBLE

<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>ESP Slider Keyboard</title>

  <style>

    .slider {

      width: 300px;

      height: 10px;

      background: #ccc;

      margin: 40px;

      position: relative;

      outline: none;

    }

    .slider:focus {

      box-shadow: 0 0 0 2px #000;

    }

    .thumb {

      width: 20px;

      height: 20px;

      background: #000;

      position: absolute;

      top: -5px;

      left: 0;

    }

  </style>

</head>

<body>

  <div class="slider" id="slider" role="slider" tabindex="0" aria-orientation="horizontal" aria-valuemin="0"

    aria-valuemax="180" aria-valuenow="0">

    <div class="thumb" id="thumb"></div>

  </div>

  <p>Góc: <span id="angle">0</span>°</p>

  <script>

    const slider = document.getElementById('slider');

    const thumb = document.getElementById('thumb');

    const angleText = document.getElementById('angle');

    let angle = 0;

    slider.addEventListener('keydown', (e) => {

      // 🔴 CHẶN TRÌNH DUYỆT XỬ LÝ PHÍM

      if (["ArrowLeft", "ArrowRight"].includes(e.key)) {

        e.preventDefault();

      }

      if (e.key === "ArrowRight") angle += 5;

      if (e.key === "ArrowLeft") angle -= 5;

      angle = Math.max(0, Math.min(180, angle));

      slider.setAttribute("aria-valuenow", angle);

      thumb.style.left = (angle / 180) \* 280 + "px";

      angleText.textContent = angle;

      // fetch(`/servo?angle=${angle}`);

    });

  </script>

</body>

</html>

<!--

####################################################################

### GIAI ĐOẠN 1: TRANG VỪA LOAD – TRÌNH DUYỆT KHỞI TẠO THẾ GIỚI BAN ĐẦU

####################################################################

###

<!DOCTYPE html> + <html> + <head> + <body>

###

- Trình duyệt đọc HTML từ trên xuống

- Xác định đây là HTML5

- Bắt đầu tạo DOM Tree (cây đối tượng trong bộ nhớ)

###

<div class="slider" id="slider" ...>

###

- DOM tạo 1 node DIV

- Thuộc tính ban đầu:

  id = "slider"

  class = "slider"

  tabindex = 0  → phần tử CÓ THỂ focus

  role = slider → vai trò ARIA

  aria-valuenow = 0 → trạng thái logic ban đầu

###

<div class="thumb" id="thumb">

###

- DOM tạo node con

- thumb CHƯA DI CHUYỂN, left = 0 (theo CSS)

###

<p>Góc: <span id="angle">0</span></p>

###

- DOM tạo text node hiển thị số 0

- Đây là trạng thái hiển thị ban đầu cho người dùng

####################################################################

### GIAI ĐOẠN 2: CSS ĐỌC DOM – ÁP STYLE

####################################################################

###

.slider { ... }

###

- CSS tìm node có class "slider"

- Áp width, height, background

- position: relative → làm mốc cho thumb

###

.thumb { ... }

###

- CSS áp position: absolute

- top = -5px, left = 0

→ thumb nằm ở ĐẦU TRÁI slider

###

.slider:focus { box-shadow }

###

- CHƯA kích hoạt

- Chỉ chạy KHI slider được focus

####################################################################

### GIAI ĐOẠN 3: SCRIPT ĐƯỢC ĐỌC – JS CHẠY NGAY KHI LOAD

####################################################################

###

const slider = document.getElementById('slider');

###

- document: đối tượng DOM gốc

- dấu . : truy cập phương thức getElementById của document

- JS lấy node DIV#slider từ DOM

- slider biến trỏ tới node đó

###

const thumb = document.getElementById('thumb');

###

- JS lấy node DIV#thumb

- thumb trỏ trực tiếp tới phần tử con

###

const angleText = document.getElementById('angle');

###

- JS lấy node SPAN hiển thị số góc

###

let angle = 0;

###

- JS tạo biến logic angle = 0

- DOM CHƯA đổi gì

- CSS CHƯA phản ứng gì

####################################################################

### GIAI ĐOẠN 4: ĐĂNG KÝ SỰ KIỆN – CHỜ NGƯỜI DÙNG

####################################################################

###

slider.addEventListener('keydown', (e) => { ... })

###

- Dấu . : gọi method addEventListener của node slider

- Trình duyệt ghi nhớ:

  "KHI slider đang focus + có keydown → gọi hàm này"

- JS CHƯA chạy thân hàm

####################################################################

### GIAI ĐOẠN 5: NGƯỜI DÙNG TAB → SLIDER ĐƯỢC FOCUS

####################################################################

###

tabindex="0"

###

- Người dùng nhấn TAB

- Trình duyệt cho slider nhận focus

- DOM: trạng thái focus = true

###

.slider:focus

###

- CSS selector :focus khớp

- Box-shadow xuất hiện

→ người dùng BIẾT mình đang điều khiển slider

####################################################################

### GIAI ĐOẠN 6: NGƯỜI DÙNG NHẤN PHÍM ← HOẶC →

####################################################################

###

keydown event

###

- Trình duyệt phát sinh sự kiện keydown

- Đối tượng event e được tạo

- e.key chứa giá trị phím, ví dụ "ArrowRight"

####################################################################

### GIAI ĐOẠN 7: JS BẮT ĐẦU CHẠY TỪ TRÊN XUỐNG

####################################################################

###

if (["ArrowLeft", "ArrowRight"].includes(e.key))

###

- includes() được gọi trên mảng

- So sánh e.key với từng phần tử

- Nếu e.key === "ArrowLeft" HOẶC "ArrowRight"

→ kết quả TRUE

###

e.preventDefault();

###

- JS chặn hành vi mặc định

- Trước khi chặn:

  → trình duyệt sẽ cuộn trang / đổi focus

- Sau khi chặn:

  → quyền điều khiển thuộc về JS

####################################################################

### GIAI ĐOẠN 8: SO SÁNH === VÀ THAY ĐỔI BIẾN LOGIC

####################################################################

###

if (e.key === "ArrowRight") angle += 5;

###

- === so sánh:

  bên trái: e.key (chuỗi)

  bên phải: "ArrowRight"

- Nếu TRUE:

  → angle = angle + 5

- JS là thằng đổi biến

###

if (e.key === "ArrowLeft") angle -= 5;

###

- So sánh tương tự

- Nếu TRUE → angle giảm

####################################################################

### GIAI ĐOẠN 9: GIỚI HẠN GIÁ TRỊ

####################################################################

###

angle = Math.max(0, Math.min(180, angle));

###

- Math.min: nếu angle > 180 → trả về 180

- Math.max: nếu angle < 0 → trả về 0

- Kết quả cuối cùng GHI ĐÈ angle

- DOM CHƯA đổi

####################################################################

### GIAI ĐOẠN 10: JS GHI DỮ LIỆU VÀO DOM

####################################################################

###

slider.setAttribute("aria-valuenow", angle);

###

- Trước khi ghi:

  aria-valuenow = giá trị cũ

- JS gọi setAttribute

- Sau khi ghi:

  aria-valuenow = giá trị mới

- Screen reader đọc được góc mới

###

thumb.style.left = (angle / 180) \* 280 + "px";

###

- Dấu .style : truy cập inline style của DOM node

- left thay đổi

- DOM thay đổi layout

###

angleText.textContent = angle;

###

- Trước: text node = số cũ

- Sau: text node = số mới

- JS là thằng ghi

- DOM thay đổi nội dung

####################################################################

### GIAI ĐOẠN 11: CSS & ARIA PHẢN ỨNG

####################################################################

###

DOM đã đổi

###

- CSS reflow / repaint

- Thumb di chuyển

- Số hiển thị đổi

###

ARIA phản ứng

###

- Trình đọc màn hình:

  "Slider, value X"

- Người khiếm thị hiểu trạng thái mới

####################################################################

### TỔNG KẾT VAI TRÒ (CUỐI CÙNG)

####################################################################

- HTML: tạo cấu trúc & ngữ nghĩa

- CSS: hiển thị & phản ứng với DOM

- JS: điều khiển logic & thay đổi trạng thái

- DOM: nơi lưu trạng thái sống

- ARIA: mô tả trạng thái cho máy đọc màn hình

→ TẤT CẢ CHẠY THEO LUỒNG THỜI GIAN, KHÔNG CÁI NÀO TỰ NHIÊN NHẢY CÓC

-->

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[207]. Thuộc tính aria-posinset

-chủ yếu dùng phím để điều khiển

→ Cho trình duyệt & screen reader biết PHẦN TỬ NÀY đang đứng Ở VỊ TRÍ THỨ MẤY trong một TẬP HỢP.

------------------------------------------------

[207.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Xác định VỊ TRÍ THỨ TỰ của một item (số mấy)

- Ý 2: Dùng khi danh sách / menu / item được tạo ĐỘNG bằng JS

Không hiển thị cho mắt người, CHỈ để máy hiểu

------------------------------------------------

[207.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Hiển thị trạng thái danh sách

  + Điều khiển UI bằng phím

  + Hỗ trợ screen reader

Ví dụ ngắn:

- Menu có 5 mục

- Item hiện tại có aria-posinset="3"

→ Máy đọc: "mục 3 trong 5"

------------------------------------------------

[207.3] PHẠM VI ÁP DỤNG

- ARIA-only

- Áp cho:

  + role="menuitem"

  + role="option"

  + role="tab"

  + role="listitem"

- Có cần focus không?

  + CÓ (muốn đọc vị trí thì item thường được focus)

------------------------------------------------

[207.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag aria-posinset="số">

- Ví dụ đúng:

  <div role="menuitem" aria-posinset="2">

- Ví dụ SAI:

  aria-posinset="hai"

  → SAI vì CHỈ nhận SỐ

------------------------------------------------

[207.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + SỐ (integer)

- Ví dụ hay gặp:

  + 1

  + 2

  + 3

  + n

Đếm bắt đầu từ 1 (KHÔNG phải 0)

------------------------------------------------

[207.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng TAB / điều hướng tới item

2. Item được focus

3. Trình duyệt đọc aria-posinset

4. Screen reader ghép với aria-setsize

5. Thông báo: “mục X trong Y”

DÙNG để DEBUG lỗi thứ tự item

------------------------------------------------

[207.7] BẢN CHẤT HOẠT ĐỘNG

- Là: ARIA attribute

- Ai kích hoạt:

  + Trình duyệt

  + Trình đọc màn hình

- JavaScript:

  + Thường là THẰNG GÁN GIÁ TRỊ

  + Tự động cập nhật khi list thay đổi

------------------------------------------------

[207.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Không cập nhật aria-posinset khi thêm / xoá item

- Lỗi 2: posinset không khớp setsize

- Cách nhớ nhanh:

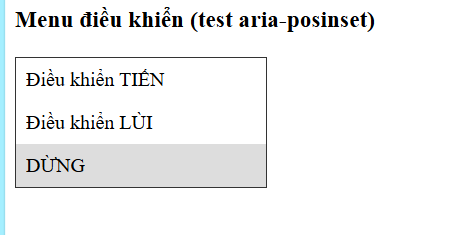
  → POS = vị trí, IN SET = trong tập hợp

------------------------------------------------

[207.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “aria-posinset dùng để nói cho máy biết ITEM này đứng THỨ MẤY trong một DANH SÁCH khi người dùng điều hướng”

207.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test aria-posinset</title>

  <style>

    .menu {

      width: 200px;

      border: 1px solid #333;

      padding: 0;

    }

    .item {

      padding: 8px;

      cursor: pointer;

      outline: none;

    }

    .item:focus {

      background: #ddd;

    }

  </style>

</head>

<body>

  <h3>Menu điều khiển (test aria-posinset)</h3>

  <div class="menu" role="menu" aria-label="Menu chính">

    <div class="item" role="menuitem" tabindex="0" aria-posinset="1" aria-setsize="3">

      Điều khiển TIẾN

    </div>

    <div class="item" role="menuitem" tabindex="-1" aria-posinset="2" aria-setsize="3">

      Điều khiển LÙI

    </div>

    <div class="item" role="menuitem" tabindex="-1" aria-posinset="3" aria-setsize="3">

      DỪNG

    </div>

  </div>

  <script>

    const items = document.querySelectorAll('.item');

    let index = 0;

    items.forEach((item, i) => {

      item.addEventListener('keydown', (e) => {

        if (e.key === 'ArrowDown') {

          e.preventDefault();

          items[index].tabIndex = -1;

          index = (index + 1) % items.length;

          items[index].tabIndex = 0;

          items[index].focus();

        }

        if (e.key === 'ArrowUp') {

          e.preventDefault();

          items[index].tabIndex = -1;

          index = (index - 1 + items.length) % items.length;

          items[index].tabIndex = 0;

          items[index].focus();

        }

      });

    });

  </script>

</body>

</html>

<!--

####################################################################

### GIAI ĐOẠN 1: TRANG VỪA LOAD – TRÌNH DUYỆT TẠO DOM BAN ĐẦU

####################################################################

###

<!DOCTYPE html> + <html> + <head> + <body>

###

- Trình duyệt đọc HTML từ trên xuống

- Xác định chuẩn HTML5

- Bắt đầu tạo DOM Tree trong bộ nhớ

###

<div class="menu" role="menu" aria-label="Menu chính">

###

- DOM tạo node DIV.menu

- role="menu" → khai báo đây là 1 MENU logic

- aria-label → đặt tên cho menu để screen reader đọc

- Chưa có focus, chỉ tồn tại cấu trúc

###

<div class="item" role="menuitem" tabindex="0" aria-posinset="1" aria-setsize="3">

###

- DOM tạo item đầu tiên

- tabindex="0" → item này CÓ THỂ focus

- aria-posinset="1" → item này đứng THỨ 1

- aria-setsize="3" → tổng số item là 3

- Đây là ITEM ĐƯỢC CHỌN BAN ĐẦU

###

<div class="item" role="menuitem" tabindex="-1" ...>

###

- Hai item còn lại được tạo

- tabindex="-1" → KHÔNG TAB tới được

- Chỉ focus khi JS chủ động gọi

####################################################################

### GIAI ĐOẠN 2: CSS ĐỌC DOM – ÁP STYLE BAN ĐẦU

####################################################################

###

.menu { ... }

###

- CSS tìm node có class "menu"

- Áp width, border

- Không có thay đổi trạng thái

###

.item { ... }

###

- CSS áp padding, cursor

- outline: none → không viền focus mặc định

###

.item:focus { background }

###

- CHƯA chạy

- Chỉ kích hoạt khi item nhận focus

####################################################################

### GIAI ĐOẠN 3: SCRIPT ĐƯỢC ĐỌC – JS KHỞI TẠO BIẾN

####################################################################

###

const items = document.querySelectorAll('.item');

###

- document: DOM gốc

- dấu . : truy cập method querySelectorAll của document

- JS lấy TOÀN BỘ node có class "item"

- items = NodeList gồm 3 phần tử

###

let index = 0;

###

- JS tạo biến index = 0

- Ý nghĩa: item đang active là item đầu tiên

- DOM CHƯA bị thay đổi

####################################################################

### GIAI ĐOẠN 4: ĐĂNG KÝ SỰ KIỆN CHO TỪNG ITEM

####################################################################

###

items.forEach((item, i) => { ... })

###

- forEach chạy 3 lần

- Mỗi lần gắn 1 keydown listener cho 1 item

- JS CHỈ ĐĂNG KÝ, CHƯA xử lý phím

###

item.addEventListener('keydown', ...)

###

- Dấu . : gọi method addEventListener của node item

- Trình duyệt ghi nhớ:

  "Nếu item này focus + có keydown → gọi hàm"

####################################################################

### GIAI ĐOẠN 5: NGƯỜI DÙNG TAB VÀO MENU

####################################################################

###

tabindex="0" (item đầu)

###

- Người dùng nhấn TAB

- Trình duyệt focus item đầu tiên

- DOM: item[0] có trạng thái focus = true

###

.item:focus

###

- CSS selector :focus khớp

- Background đổi màu

- Người dùng thấy item đang được chọn

####################################################################

### GIAI ĐOẠN 6: NGƯỜI DÙNG NHẤN PHÍM ↓

####################################################################

###

keydown event

###

- Trình duyệt tạo event keydown

- event e được truyền vào JS

- e.key = "ArrowDown"

####################################################################

### GIAI ĐOẠN 7: JS XỬ LÝ ArrowDown

####################################################################

###

if (e.key === 'ArrowDown')

###

- === so sánh:

  bên trái: e.key ("ArrowDown")

  bên phải: "ArrowDown"

- Kết quả TRUE → vào khối if

###

e.preventDefault();

###

- JS chặn hành vi mặc định

- Trước: trình duyệt có thể cuộn / đổi focus

- Sau: JS TOÀN QUYỀN điều khiển

###

items[index].tabIndex = -1;

###

- Dấu . : truy cập thuộc tính tabIndex của DOM node

- Trước: item hiện tại tabindex = 0

- JS đổi thành -1

- Item này KHÔNG còn focus được

###

index = (index + 1) % items.length;

###

- Toán học:

  index + 1 → sang item kế

  % items.length → quay vòng nếu vượt quá

- index được cập nhật sang item mới

- CHƯA tác động DOM

###

items[index].tabIndex = 0;

###

- JS gán tabindex = 0 cho item mới

- Item mới CÓ THỂ focus

###

items[index].focus();

###

- JS gọi method focus()

- DOM đổi focus từ item cũ → item mới

- CSS :focus chạy

- Screen reader đọc item mới

####################################################################

### GIAI ĐOẠN 8: ARIA PHẢN ỨNG

####################################################################

###

aria-posinset + aria-setsize

###

- Screen reader đọc:

  "Điều khiển LÙI, mục 2 trong 3"

- aria-posinset KHÔNG đổi

- Nhưng được ĐỌC DỰA TRÊN ITEM ĐANG FOCUS

####################################################################

### GIAI ĐOẠN 9: NGƯỜI DÙNG NHẤN PHÍM ↑

####################################################################

###

if (e.key === 'ArrowUp')

###

- === so sánh e.key với "ArrowUp"

- TRUE → chạy khối xử lý ArrowUp

- Logic tương tự ArrowDown nhưng index giảm

###

index = (index - 1 + items.length) % items.length;

###

- Đảm bảo không bị số âm

- Quay vòng lên item cuối nếu đang ở item đầu

####################################################################

### GIAI ĐOẠN 10: DOM & CSS CẬP NHẬT SAU MỖI PHÍM

####################################################################

###

DOM trước

###

- 1 item tabindex = 0

- 1 item focus

###

JS là thằng đổi

###

- Đổi tabindex

- Gọi focus()

###

DOM sau

###

- Focus chuyển sang item khác

- Text & ARIA không đổi

- Trạng thái điều hướng đổi

###

CSS / Screen reader phản ứng

###

- CSS đổi background

- Screen reader đọc:

  "mục X trong 3"

####################################################################

### TỔNG KẾT VAI TRÒ (CUỐI CÙNG)

####################################################################

- HTML: tạo cấu trúc menu & item

- CSS: hiển thị focus và trạng thái

- JS: điều khiển logic di chuyển

- DOM: lưu trạng thái focus sống

- ARIA: mô tả vị trí item cho máy đọc

→ Đây là LUỒNG CHẠY THỰC, từng bước, không nhảy cóc

-->

========================================

=== MẪU CHUẨN PHÂN TÍCH THUỘC TÍNH HTML ===

========================================

[208]. Thuộc tính aria-setsize

→ Khai báo cho trình duyệt & screen reader biết TỔNG SỐ PHẦN TỬ trong một TẬP HỢP logic.

------------------------------------------------

[208.1] DÙNG ĐỂ LÀM GÌ (QUAN TRỌNG NHẤT)

- Ý 1: Cho máy biết “cả nhóm này có BAO NHIÊU item”

- Ý 2: Dùng khi danh sách / menu / item được tạo ĐỘNG bằng JS

KHÔNG hiển thị ra UI, CHỈ để máy hiểu

------------------------------------------------

[208.2] ỨNG DỤNG THỰC TẾ (RẤT QUAN TRỌNG)

- Dùng để:

  + Hiển thị trạng thái danh sách

  + Điều hướng bằng phím

  + Hỗ trợ người khiếm thị

Ví dụ ngắn:

- Menu có 5 lệnh điều khiển

- Item đang focus có:

  aria-posinset="2"

  aria-setsize="5"

→ Máy đọc: “mục 2 trong 5”

------------------------------------------------

[208.3] PHẠM VI ÁP DỤNG

- ARIA-only

- Áp cho:

  + role="menuitem"

  + role="option"

  + role="tab"

  + role="listitem"

- Có cần focus không?

  + CÓ (muốn đọc vị trí thì item phải focus)

------------------------------------------------

[208.4] CÚ PHÁP (PHẦN PHẢI NHỚ KỸ)

- Cú pháp chuẩn:

  <tag aria-setsize="số">

- Ví dụ đúng:

  <div role="menuitem" aria-setsize="3">

- Ví dụ SAI:

  aria-setsize="ba"

  → SAI vì chỉ nhận GIÁ TRỊ SỐ

------------------------------------------------

[208.5] GIÁ TRỊ THƯỜNG DÙNG

- Kiểu giá trị:

  + SỐ nguyên (integer)

- Ví dụ hay gặp:

  + 1

  + 3

  + 10

Đếm tổng số item, KHÔNG trừ item ẩn

------------------------------------------------

[208.6] LUỒNG CHẠY TRONG ĐẦU TRÌNH DUYỆT

1. Người dùng TAB / điều hướng tới item

2. Item được focus

3. Trình duyệt đọc aria-setsize

4. Kết hợp với aria-posinset

5. Screen reader thông báo: “mục X trong Y”

RẤT HỮU ÍCH khi DEBUG menu động

------------------------------------------------

[208.7] BẢN CHẤT HOẠT ĐỘNG

- Là: ARIA attribute

- Ai kích hoạt:

  + Trình duyệt

  + Trình đọc màn hình

- JavaScript:

  + Là thằng THƯỜNG GÁN / CẬP NHẬT

  + Khi thêm / xoá item

------------------------------------------------

[208.8] LƯU Ý / LỖI HAY GẶP

- Lỗi 1: Không cập nhật aria-setsize khi xoá / thêm item

- Lỗi 2: setsize KHÔNG khớp với số item thật

- Cách nhớ nhanh:

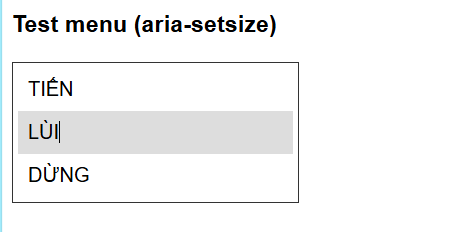
  → SETSIZE = SIZE của cả SET (cả nhóm)

------------------------------------------------

[208.9] TÓM TẮT 1 DÒNG (ĐỂ GHI NHỚ)

→ “aria-setsize dùng để khai báo TỔNG SỐ ITEM trong một NHÓM khi người dùng điều hướng bằng phím”

208.10.EXAMBLE



<!DOCTYPE html>

<html lang="vi">

<head>

  <meta charset="UTF-8">

  <title>Test aria-setsize</title>

  <style>

    body {

      font-family: Arial, sans-serif;

    }

    .menu {

      width: 220px;

      border: 1px solid #333;

      padding: 4px;

    }

    .item {

      padding: 8px;

      cursor: pointer;

      outline: none;

    }

    .item:focus {

      background: #ddd;

    }

  </style>

</head>

<body>

  <h3>Test menu (aria-setsize)</h3>

  <div class="menu" role="menu" aria-label="Menu điều khiển">

    <div class="item" role="menuitem" tabindex="0" aria-posinset="1" aria-setsize="3">

      TIẾN

    </div>

    <div class="item" role="menuitem" tabindex="-1" aria-posinset="2" aria-setsize="3">

      LÙI

    </div>

    <div class="item" role="menuitem" tabindex="-1" aria-posinset="3" aria-setsize="3">

      DỪNG

    </div>

  </div>

  <script>

    const items = document.querySelectorAll('.item');

    let index = 0;

    items.forEach((item) => {

      item.addEventListener('keydown', (e) => {

        if (e.key === 'ArrowDown') {

          e.preventDefault();

          items[index].tabIndex = -1;

          index = (index + 1) % items.length;

          items[index].tabIndex = 0;

          items[index].focus();

        }

        if (e.key === 'ArrowUp') {

          e.preventDefault();

          items[index].tabIndex = -1;

          index = (index - 1 + items.length) % items.length;

          items[index].tabIndex = 0;

          items[index].focus();

        }

      });

    });

  </script>

</body>

</html>

hết

HTML Attribute Value Types (5 loại chính) :

KIỂU GIÁ TRỊ

1. String (chuỗi văn bản)

   id="motor1"

   class="btn primary"

   title="Bật động cơ"

2. Boolean (đúng/sai – chỉ cần có mặt là true)

   required

   disabled

   checked

   readonly

   autofocus

3. Nuber (số)

   min="0"

   max="100"

   step="5"

   width="200"

   height="100"

4. Enum (chọn 1 trong danh sách cố định)

   type="text | number | email | password"

   method="get | post"

   type="submit | reset | button"

5. URL / Path (đường dẫn / liên kết)

   href="/home"

   src="robot.jpg"

   action="/api/star

#MỤC LỤC#

TÓM TẮT 89 THẺ TAG VÀ THUỘC TÍNH CỦA THẺ HTML

13.Thẻ tag <body>

14.Thẻ tag <b> - chữ đậm ( trang trí)

15.Thẻ tag <strong> - Chữ đâm - Quan trong (KD)

16.Thẻ tag <br> -xuống dòng

17.Thẻ tag <button> - nút bấm

18.Thẻ tag <blockquote> - trích dẫn [ KO HC]

19.Thẻ tag <base> - Đường dẫn gốc ( ít dùng )

20.<bdi> & <bdo> — ngôn ngữ đặc biệt [ KO DÙNG ]

21.Thẻ tag <a> -- Liên kết (LINK)

22.Thẻ tag <abbr> -- Viết tắt

23.Thẻ tag <address> --- thông tin liên hệ

24.Thẻ tag <area> - vùng click trong ảnh ( nâng cao)

25.Thẻ tag <article> --- bài viết độc lập [KO HỌC]

26.Thẻ tag <aside> -- nộ dung phụ

27.Thẻ tag <audio> --- phát âm thanh

28.Thẻ tag <canvas> - Đồ họa

29.Thẻ tag <caption> -- tiêu đề

30.Thẻ tag <code>

31.Thẻ tag <col>

32.Thẻ tag <colgroup>

33.Thẻ tag <data>

34.Thẻ tag <datalist> --- Gợi ý

35.Thẻ tag <dd> -- Mô tả

36.Thẻ tag <del> --- gạch ngang

37.Thẻ tag <details>

38.Thẻ tag <dfn>

39.Thẻ tag <div> [ RQT ]

40.Thẻ tag <dl> [ QT]

41.Thẻ tag <dt> [QT]

42.Thẻ tag <em>

43.Thẻ tag <embed> - -nhúng file bên ngoài

44.Thẻ tang <form> -- Biểu mẫu (RQT)

45.Thẻ tag <fieldset> -- nhóm các input

46.Thẻ tang <legend> --- tiêu đề của fieldset [ BX]

47.Thẻ tag <firuge> -- Khối hình ảnh / nội dung minh họa [RQT]

48.Thẻ tag <footer> -- chân trang / cuối trang

49.Thẻ tag <h1> <h6>

50.Thẻ tag <header> --- đầu trang

51.Thẻ tag <hgroup> -- nhóm tiêu đề [ít D]

52.Thẻ tag <hr> -- GẠCH NGANG

53.Thẻ tag <i> --- Chữ nghiêng

54.Thẻ tag <iframe> -- nhúng trang khác

55.Thẻ tag <img> -- Hiện Thị ẢNH

56.Thẻ tag <input> --nhập dữ liệu

57.Thẻ tag <ins> -- CHỮ GẠCH CHÂN

58.Thẻ tag <kbd> --Hiện thị bàn phím

59.Cách cài key

60.Thẻ tag <label> -- Gán nhãn cho input [RQT]

61.Thẻ tag <legend> – Tiêu đề cho nhóm input

62.Thẻ tag <li> - Item trong danh sách [menu]

63.Thẻ tag <link> - Gắn file ngoài (CSS)

64.Thẻ tag <map> - Bản đồ vùng click trên ảnh

65.Thẻ tag <mark> - TÔ SÁNG CHỮ

67.Thẻ tag <meter> -- THANH ĐO GIÁ TRỊ

68.Thẻ tag <nav> -Điều hướng trang wed [RQT]

69.Thẻ tag <noscript> -

70.Thẻ tag <object> -- Nhúng tài nguyên bên ngoài

71.Thẻ tag <ol> - danh sách có thứ tự

72.Thẻ tag <option> -- 1 lựa chọn trong <select> [RQT]

73.Thẻ tag <optgroup> -- Nhóm các option [RQT]

74.Thẻ tag <output> --Hiện thị kết quả [RQT]

75.Thẻ tag <p> - đoạn văn

76.Thẻ tag <pre> --- Hiện Thị Nguyên Dạng

77.Thẻ tag <progress> --THANH TIẾN TRÌNH

78.Thẻ tag <q> -- Trích dẫn ngắn

79.Thẻ tag <samp> -- Kết quả từ máy tính / chương trình

80.Thẻ tag <script> ---- JavaScript [RQT]

81.Thẻ tag <section> ---Khu vực nội dung

82.Thẻ tag <select> Danh sách chọn [RQT]

83.Thẻ tag <small> -- Ghi chú nhỏ [RQT]

84.Thẻ tag <source> --- Nguồn media

85.Thẻ tag <span> ---Bọc chữ inline [RQT]

86.Thẻ tag <strong> ---Nhấn mạnh nội dung

87.Thẻ tag <style> ---CSS trong HTML

88.Thẻ tag <sub> & <sup>

89.Thẻ tag <summary> & <details> [RQT]

90.Thẻ tag <table> ---Bảng dữ liệu[RHD]

91.Thẻ tag <thead> , <tbody> , <tfoot> --- Phần bảng [RQT]

92.Thẻ tag <tr> --- Hàng [QT]

93.Thẻ tag <td> -- Ô dữ liệu [QT]

94.Thẻ tag <th> ---Ô tiêu đề [QT]

95.Thẻ tag <template> ---KHUNG ẨN (VIP) [RQT]

96.Thẻ tag <textarea> -- ô nhập nhiều dòng có thể kéo góc to

97.Thẻ tag <time> --- Thời gian

98.Thẻ tag <title> --- Tên TAB [RQT]

99.Thẻ tag <track> --PHỤ ĐỀ VIDEO

100.Thẻ tag <ul> --- Danh sách không thứ tự [QT]

101.Thẻ tag <video>

102.Thẻ tag <wbr> — Điểm NGẮT DÒNG TÙY Ý [VIP]

13.Thẻ tag <body>

14.Thẻ tag <b> - chữ đậm ( trang trí)

15.Thẻ tag <strong> - Chữ đâm - Quan trong (KD)

16.Thẻ tag <br> -xuống dòng

17.Thẻ tag <button> - nút bấm

18.Thẻ tag <blockquote> - trích dẫn [ KO HC]

19.Thẻ tag <base> - Đường dẫn gốc ( ít dùng )

20.<bdi> & <bdo> — ngôn ngữ đặc biệt [ KO DÙNG ]

21.Thẻ tag <a> -- Liên kết (LINK)

22.Thẻ tag <abbr> -- Viết tắt

23.Thẻ tag <address> --- thông tin liên hệ

24.Thẻ tag <area> - vùng click trong ảnh ( nâng cao)

25.Thẻ tag <article> --- bài viết độc lập [KO HỌC]

26.Thẻ tag <aside> -- nộ dung phụ

27.Thẻ tag <audio> --- phát âm thanh

28.Thẻ tag <canvas> - Đồ họa

29.Thẻ tag <caption> -- tiêu đề

30.Thẻ tag <code>

31.Thẻ tag <col>

32.Thẻ tag <colgroup>

33.Thẻ tag <data>

34.Thẻ tag <datalist> --- Gợi ý

35.Thẻ tag <dd> -- Mô tả

36.Thẻ tag <del> --- gạch ngang

37.Thẻ tag <details>

38.Thẻ tag <dfn>

39.Thẻ tag <div> [ RQT ]

40.Thẻ tag <dl> [ QT]

41.Thẻ tag <dt> [QT]

42.Thẻ tag <em>

43.Thẻ tag <embed> - -nhúng file bên ngoài

44.Thẻ tang <form> -- Biểu mẫu (RQT)

45.Thẻ tag <fieldset> -- nhóm các input

46.Thẻ tang <legend> --- tiêu đề của fieldset [ BX]

47.Thẻ tag <firuge> -- Khối hình ảnh / nội dung minh họa [RQT]

48.Thẻ tag <footer> -- chân trang / cuối trang

49.Thẻ tag <h1> <h6>

50.Thẻ tag <header> --- đầu trang

51.Thẻ tag <hgroup> -- nhóm tiêu đề [ít D]

52.Thẻ tag <hr> -- GẠCH NGANG

53.Thẻ tag <i> --- Chữ nghiêng

54.Thẻ tag <iframe> -- nhúng trang khác

55.Thẻ tag <img> -- Hiện Thị ẢNH

56.Thẻ tag <input> --nhập dữ liệu

57.Thẻ tag <ins> -- CHỮ GẠCH CHÂN

58.Thẻ tag <kbd> --Hiện thị bàn phím

59.Cách cài key

60.Thẻ tag <label> -- Gán nhãn cho input [RQT]

61.Thẻ tag <legend> – Tiêu đề cho nhóm input

62.Thẻ tag <li> - Item trong danh sách [menu]

63.Thẻ tag <link> - Gắn file ngoài (CSS)

64.Thẻ tag <map> - Bản đồ vùng click trên ảnh

65.Thẻ tag <mark> - TÔ SÁNG CHỮ

67.Thẻ tag <meter> -- THANH ĐO GIÁ TRỊ

68.Thẻ tag <nav> -Điều hướng trang wed [RQT]

69.Thẻ tag <noscript> -

70.Thẻ tag <object> -- Nhúng tài nguyên bên ngoài

71.Thẻ tag <ol> - danh sách có thứ tự

72.Thẻ tag <option> -- 1 lựa chọn trong <select> [RQT]

73.Thẻ tag <optgroup> -- Nhóm các option [RQT]

74.Thẻ tag <output> --Hiện thị kết quả [RQT]

75.Thẻ tag <p> - đoạn văn

76.Thẻ tag <pre> --- Hiện Thị Nguyên Dạng

77.Thẻ tag <progress> --THANH TIẾN TRÌNH

78.Thẻ tag <q> -- Trích dẫn ngắn

79.Thẻ tag <samp> -- Kết quả từ máy tính / chương trình

80.Thẻ tag <script> ---- JavaScript [RQT]

81.Thẻ tag <section> ---Khu vực nội dung

82.Thẻ tag <select> Danh sách chọn [RQT]

83.Thẻ tag <small> -- Ghi chú nhỏ [RQT]

84.Thẻ tag <source> --- Nguồn media

85.Thẻ tag <span> ---Bọc chữ inline [RQT]

86.Thẻ tag <strong> ---Nhấn mạnh nội dung

87.Thẻ tag <style> ---CSS trong HTML

88.Thẻ tag <sub> & <sup>

89.Thẻ tag <summary> & <details> [RQT]

90.Thẻ tag <table> ---Bảng dữ liệu[RHD]

91.Thẻ tag <thead> , <tbody> , <tfoot> --- Phần bảng [RQT]

92.Thẻ tag <tr> --- Hàng [QT]

93.Thẻ tag <td> -- Ô dữ liệu [QT]

94.Thẻ tag <th> ---Ô tiêu đề [QT]

95.Thẻ tag <template> ---KHUNG ẨN (VIP) [RQT]

96.Thẻ tag <textarea> -- ô nhập nhiều dòng có thể kéo góc to

97.Thẻ tag <time> --- Thời gian

98.Thẻ tag <title> --- Tên TAB [RQT]

99.Thẻ tag <track> --PHỤ ĐỀ VIDEO

100.Thẻ tag <ul> --- Danh sách không thứ tự [QT]

101.Thẻ tag <video>

102.Thẻ tag <wbr> — Điểm NGẮT DÒNG TÙY Ý [VIP]

TÓM TẮT THUỘC TÍNH CỦA THẺ ĐƠN HTML

103. Thuộc tính id

104. Thuộc tính class

105. Thuộc tính style

106. Thuộc tính title

107. Thuộc tính hidden

108. Thuộc tính tabindex

109. Thuộc tính accesskey

110. Thuộc tính contenteditable

111. Thuộc tính enterkeyhint

112. Thuộc tính placeholder và textarea

113.Thuộc tính sự kiện onkeydown

114. Thuộc tính inert

115. Thuộc tính inputmode

116. Thuộc tính data-\*

117. Thuộc tính data-id

118. Thuộc tính data-value

119. Thuộc tính data-state

120.Thuộc tính data-led

121. Thuộc tính aria-activedescendant

122. Thuộc tính aria-busy

123.Thuộc tính aria - busy

124. Thuộc tính aria-checked

125. Thuộc tính role

126.Thuộc tính aria-controls

127. Thuộc tính aria-expanded

128. CÁCH BẮT SỰ KIỆN click TRONG JAVASCRIPT

129. Thuộc tính aria-current

130. Thuộc tính aria-describedby

131. Thuộc tính aria-disabled

132. Thuộc tính aria-haspopup

133. Thuộc tính aria-hidden

134. Thuộc tính aria-invalid

135. Thuộc tính aria-label

136. Thuộc tính aria-labelledby

137. Thuộc tính aria-live

138. Thuộc tính aria-modal

139. Thuộc tính aria-multiselectable

140.Thuộc tính aria-pressed

141. Thuộc tính aria-readonly

142. Thuộc tính aria-required

143. Thuộc tính aria-selected

144. Thuộc tính aria-valuemax

145. Thuộc tính aria-valuemin

146. Thuộc tính aria-valuenow

147. Thuộc tính onclick

148. Thuộc tính onmousedown

149. Thuộc tính onmouseup

150. Thuộc tính onmouseenter

151. Thuộc tính onmouseleave

152.Thuộc tính onwheel

153. Thuộc tính onkeydown

154. Thuộc tính oninput

155. Thuộc tính onchange

156. Thuộc tính onfocus

[157]. Thuộc tính onblur

[158]. Thuộc tính onsubmit

[ 159 ]. Thuộc tính onload

[160]. Thuộc tính onerror

[160]. Thuộc tính onerror

[161]. Thuộc tính onresize

[ 163 ]. Thuộc tính oncontextmenu

[164]. Thuộc tính href

[165]. Thuộc tính target

[166]. Thuộc tính download

[167]. Thuộc tính rel

[168]. Thuộc tính referrerpolicy

[169]. Thuộc tính name

[170]. Thuộc tính value

[171]. Thuộc tính disabled

[172]. Thuộc tính autofocus

173. Thuộc tính placeholder

174. Thuộc tính checked

175. Thuộc tính readonly

176. Thuộc tính required

177. Thuộc tính multiple

178. Thuộc tính pattern

[179]. Thuộc tính form

[180]. Thuộc tính min và max

[181]. Thuộc tính step

[182]. Thuộc tính minlength & maxlength

[183]. Thuộc tính autocomplete

[184]. Thuộc tính list

[185]. Thuộc tính accept

[186]. Thuộc tính rows

[187]. Thuộc tính action

[188]. Thuộc tính method

[189]. Thuộc tính enctype

[190]. Thuộc tính novalidate

[191]. Thuộc tính src

[192]. Thuộc tính alt

[193]. Thuộc tính width

[194]. Thuộc tính height

[195]. Thuộc tính loading

[196]. Thuộc tính decoding

[197]. Thuộc tính srcset

[198]. Thuộc tính crossorigin

[199]. Thuộc tính controls

[ 200 ]. Thuộc tính autoplay

[201]. Thuộc tính loop

[202]. Thuộc tính muted

[203]. Thuộc tính colspan

[204]. Thuộc tính rowspan

[205]. Thuộc tính scope

[206]. Thuộc tính aria-orientation

[207]. Thuộc tính aria-posinset

[208]. Thuộc tính aria-setsize

KIỂU GIÁ TRỊ

1. String (chuỗi văn bản)

2. Boolean (đúng/sai – chỉ cần có mặt là true)

3. Nuber (số)

4. Enum (chọn 1 trong danh sách cố định)

5. URL / Path (đường dẫn / liên kết)