**Đọc html,css,js,jquery,bootstrap**

1.html

* <html>: Khai báo bắt đầu HTML tài liệu.
* <head>: Chứa thông tin về tài liệu (siêu dữ liệu), giới hạn tiêu đề, CSS liên kết và JavaScript mã hóa.
* <body>: Chứa nội dung hiển thị trên trang web.
* <h1> - <h6>: Tiêu đề (tiêu đề) từ lớn nhất đến nhỏ nhất.
* <p>: Đoạn văn (paragraph).
* <b>hoặc<strong> : In đậm văn bản.
* <i>hoặc<em> : In nghiêng văn bản.
* Danh sách không thứ tự ( <ul>) :

<ul>

<li>Mục 1</li>

<li>Mục 2</li>

</ul>

* Danh sách có thứ tự ( <ol>) :

<ol>

<li>Mục 1</li>

<li>Mục 2</li>

</ol>

* Liên kết ( <a>) :

<a href="https://example.com">Đi đến trang web</a>

* Hình ảnh ( <img>) :

<img src="image.jpg" alt="Mô tả hình ảnh">

* Thẻ bảng (Table)

<table>

<tr>

<th>Cột 1</th>

<th>Cột 2</th>

</tr>

<tr>

<td>Dữ liệu 1</td>

<td>Dữ liệu 2</td>

</tr>

</table>

* Thẻ biểu mẫu (Form)

<form action="/submit" method="POST">

<label for="name">Tên:</label>

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

<input type="submit" value="Gửi">

</form>

* Thẻ phân chia nội dung khu vực
* <div>: Chia khối (block) lớn.
* <span>: Chia khối nhỏ, dùng để bọc nội dung bên trong dòng.

các thuộc tính của html :

Các Thuộc tính cơ bản :

* id
* class
* style
* title

2.css

các thuộc tính của css :

- Văn bản và Phông chữ :

* color
* font-size
* font-family
* font-weight
* text-align
* text-decoration
* line-height
* text-transform

- Bố cục và khoảng cách :

* margin
* padding
* border
* width
* height
* max-width
* max-height
* display
* position
* top
* right
* bottom
* left

- Nền và Màu sắc :

* background-color
* background-image
* background-size
* background-position
* background-repeat

- Box và Bóng đổ :

* box-sizing
* box-shadow
* overflow

- Flexbox và Grid :

* flex-direction
* justify-content
* align-items
* grid-template-columns
* grid-template-rows

- Khác :

* opacity
* z-index
* transition
* transform

3. JavaScript :

Các khái niệm quan trọng trong JavaScript:

* Biến (Variables): Có ba cách khai báo biến là var, let, và const.
* Toán tử (Operators):+, -, \*, /, ==, ===, !=, !==, >, <, >=, <=, &&, ||, !
* Cấu trúc điều kiện: if, else if, else, và switch.
* Vòng lặp: for, while, và do...while.
* Hàm: function, hàm mũi tên (arrow function).
* DOM : Các phương thức như getElementById(), querySelector(), createElement(), và appendChild() thường được sử dụng để thao tác với DOM

4. Jquery :

Các khái niệm quan trọng trong jquery

* ID Selector: Chọn phần tử dựa trên ID ($('#id')).
* Class Selector: Chọn phần tử dựa trên class ($('.class')).
* Tag Selector: Chọn phần tử dựa trên tên thẻ HTML ($('div')).
* Attribute Selector: Chọn phần tử dựa trên thuộc tính ($('input[type="text"]')).

Thao tác DOM:

* html(): Thay đổi hoặc lấy nội dung HTML của phần tử.
* text(): Thay đổi hoặc lấy nội dung văn bản.
* append(), prepend(), after(), before(): Thêm phần tử con vào trước/sau phần tử hiện có.
* remove(): Xóa phần tử khỏi DOM.

Duyệt DOM :

jQuery cung cấp các phương thức để di chuyển giữa các phần tử trong DOM:

* parent(), children(), siblings(), find(): Duyệt qua các phần tử liên quan.
* each(): Lặp qua tập hợp các phần tử

5. Bootstrap

Bootstrap có các thành phần chính như sau:

* Alerts: Hiển thị thông báo với nhiều kiểu màu sắc.
* Badge: Huy hiệu nhỏ để hiển thị số lượng hoặc trạng thái.
* Breadcrumb: Đường dẫn điều hướng.
* Buttons: Nút với nhiều kiểu dáng và màu sắc.
* Button Group: Nhóm các nút lại với nhau.
* Cards: Thẻ chứa nội dung, hình ảnh, tiêu đề.
* Carousel: Trình chiếu ảnh hoặc nội dung.
* Collapse: Thu gọn nội dung.
* Dropdowns: Trình đơn thả xuống.
* Forms: Các trường nhập liệu.
* Input Group: Nhóm nhập liệu có biểu tượng hoặc nút.
* List Group: Nhóm các mục danh sách.
* Modal: Hộp thoại bật lên.
* Navbar: Thanh điều hướng.
* Pagination: Phân trang.
* Progress: Thanh tiến trình.
* Tooltips: Chú thích.
* Popovers: Chú thích mở rộng.

**Liên hệ với các project đã làm: tự nhận xét xem trình độ của mình về html,css,js, ở level nào?**

Tự nhận sét về bản thân thì em cảm thấy bản thân em rất yếu kém trong khoản lập trình , tất cả những gì em học và làm chỉ toàn là bắt trước người đi trước, học tới đâu tìm hiểu tới đó vì em không thể tự học tự tìm hiểu mà không có người dẫn dắt. em biết như thế với một sinh viên năm 4 là đáng lo ngại , bản thân em cũng đã ý thức được sự yếu kém của bản thân và em cũng đang cố gắng.hiện tại em đang tìm lại định hướng cho bản thân để lấy đó làm nỗ lực học tập .

**Ngôn ngữ lập trình C# gồm các kiểu dữ liệu, các loại toán tử, class, lập trình hướng đối tượng với C#, từ khoá static, public, private.**

Các kiểu dữ liệu trong C#

* Kiểu dữ liệu cơ bản: int (số nguyên), float, double (số thực), char (ký tự), string (chuỗi), bool (giá trị đúng/sai).
* Kiểu phức tạp: array (mảng), class, struct, enum, interface.
* Kiểu giá trị và kiểu tham chiếu: Kiểu giá trị lưu trữ trực tiếp dữ liệu (int, float), còn kiểu tham chiếu lưu trữ địa chỉ của đối tượng (string, class).

Các loại toán tử trong C#

* Toán tử số học: +, -, \*, /, % để thực hiện các phép tính.
* Toán tử quan hệ: ==, !=, >, <, >=, <= để so sánh.
* Toán tử logic: &&, ||, ! để xử lý các giá trị boolean.
* Toán tử gán: =, +=, -=, \*=, /= để gán và tính toán.
* Toán tử điều kiện: ?: (toán tử ba ngôi) để thực hiện kiểm tra điều kiện.

Class trong C#

* Class là một bản mẫu (template) để tạo ra các đối tượng (object), chứa thuộc tính (property) và phương thức (method).
* Định nghĩa class: public class Person { public string Name; public int Age; }
* Tạo đối tượng từ class: Person person = new Person();

Lập trình hướng đối tượng (OOP) với C#

* Tính đóng gói (Encapsulation): Bảo vệ dữ liệu bằng cách dùng private cho các thuộc tính, và truy cập thông qua phương thức public.
* Tính kế thừa (Inheritance): Tạo class con kế thừa từ class cha dùng từ khóa :.
* Tính đa hình (Polymorphism): Cho phép phương thức có cùng tên nhưng hoạt động khác nhau trong các class khác nhau, nhờ từ khóa virtual và override.
* Tính trừu tượng (Abstraction): Ẩn chi tiết bên trong và chỉ hiển thị các chức năng cần thiết. Được thực hiện qua abstract class và interface.

Từ khóa static

* static cho phép các thành phần của class không phụ thuộc vào đối tượng, mà được truy cập trực tiếp từ class.
* Ví dụ: public static int Count; có thể gọi ClassName.Count.

Từ khóa public, private

* public: Cho phép truy cập từ bên ngoài class.
* private: Chỉ có thể truy cập bên trong class đó.

**Cách hoạt động của project loại console, class library, windows form, .net web form ?**

Console Application

Cách hoạt động:

Khi chạy, ứng dụng console khởi động trong một cửa sổ dòng lệnh (Command Prompt hoặc Terminal).

* Chạy tuần tự: Bắt đầu từ phương thức Main(), mã lệnh thực thi theo thứ tự từ trên xuống dưới.
* Giao tiếp qua dòng lệnh: Người dùng nhập dữ liệu vào console, chương trình xử lý và xuất kết quả ra console. Không có giao diện đồ họa, chỉ hiển thị văn bản.

Class Library

Cách hoạt động:

Class library không tự chạy mà được biên dịch thành file .dll.

* Được tham chiếu bởi các ứng dụng khác: Các ứng dụng console, Windows Forms, hoặc Web Forms tham chiếu tới thư viện này để sử dụng các class và phương thức chứa trong đó.
* Tăng khả năng tái sử dụng: Class library chứa mã logic và chức năng chung, có thể được sử dụng lại trong nhiều project khác nhau, giúp giảm bớt việc lặp lại mã lệnh.

Windows Forms Application

Cách hoạt động:

Khi khởi động, ứng dụng tạo và hiển thị một cửa sổ GUI. Mỗi cửa sổ là một “Form” có chứa các điều khiển (như nút, hộp văn bản).

* Dựa trên sự kiện (Event-driven): Chương trình chờ đợi các sự kiện từ người dùng như nhấn nút, nhập văn bản. Khi một sự kiện xảy ra, các hàm sự kiện tương ứng sẽ được gọi để xử lý.
* Chạy liên tục: Ứng dụng chạy trong một vòng lặp thông điệp (message loop) liên tục, chờ đợi các sự kiện và xử lý chúng, cho đến khi người dùng đóng ứng dụng.

ASP.NET Web Forms

Cách hoạt động:

Người dùng truy cập trang web qua trình duyệt, gửi yêu cầu (HTTP request) đến máy chủ.

* Trang ASPX: Khi nhận yêu cầu, trang .aspx sẽ được xử lý trên server-side (máy chủ). ASP.NET chuyển đổi trang này thành HTML và JavaScript để trình duyệt có thể hiển thị.
* Xử lý theo mô hình Postback: Khi người dùng tương tác (như nhấn nút), yêu cầu được gửi về máy chủ, trang được xử lý lại và kết quả được trả về trình duyệt. Các điều khiển server-side giúp đơn giản hóa quá trình này bằng cách tự động tạo HTML và xử lý dữ liệu phía máy chủ.

Vòng đời trang (Page Lifecycle): Trang trải qua nhiều giai đoạn như khởi tạo, tải, xử lý sự kiện, và cuối cùng là render để trả lại kết quả cho trình duyệt.