**LEARNING WEB DESIGN**

Resource: <https://learningwebdesign.com./>

Exercise materials: <https://learningwebdesign.com./5e/materials/index.html>

Articles: <https://learningwebdesign.com./articles/index.html>

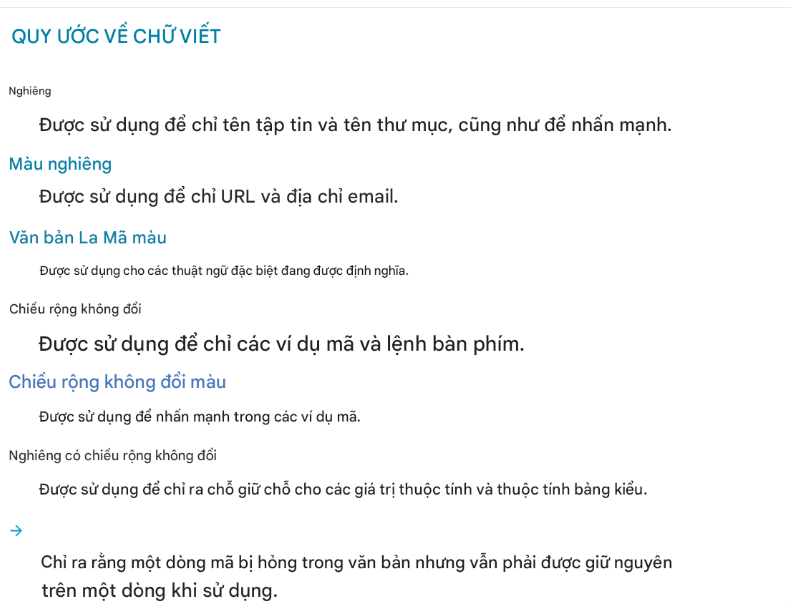
Links: <https://learningwebdesign.com./links.html>

Support: <https://learningwebdesign.com./instructors/index.html>

Github: <https://github.com/scottgruber/LWD5e>

Tech: <https://learningwebdesign.com/instructors/resources.html>’

# **QUY ƯỚC**



# **PART 01: GETTING STARTED**

## **CHAPTER01: GETTING STARTED IN WEB DESIGN**

### CÁC CÔNG CỤ KÉO THẢ =))))

WordPress (www.wordpress.com)

Squarespace (squarespace.com)

Wix (wix.com)

SiteBuilder (sitebuilder.com)

Weebly (weebly.com)

Khi bắt đầu hành trình học làm website, **bất kể động lực của bạn là gì** (học để đi làm, để kinh doanh, hay chỉ đơn giản là yêu thích), **câu hỏi đầu tiên luôn luôn là**:  
➔ **"Mình nên bắt đầu từ đâu?"**

Và đúng là, khi nhìn vào thế giới rộng lớn của lập trình web, bạn có thể cảm thấy **choáng ngợp**:

* Quá nhiều kiến thức cần học,
* Quá nhiều công nghệ khác nhau,
* Không biết nên nhảy vào phần nào trước.

💡 **Nhưng đừng lo!**  
Bạn chỉ cần **bắt đầu từ một điểm bất kỳ**, miễn là **bạn bắt đầu**. Bước đầu tiên luôn quan trọng nhất!

**Nội dung chương này sẽ giúp gì cho bạn?**

* **Chương này** giống như **tấm bản đồ định hướng** cho người mới:
  + Nó **cung cấp cái nhìn tổng quan** về **nghề nghiệp liên quan đến phát triển web**.
  + Trước khi thực sự xây dựng trang web đầu tiên, bạn sẽ được giới thiệu **những vai trò và trách nhiệm** trong quá trình tạo nên một website.
  + Qua đó, bạn có thể **xác định được vai trò** nào phù hợp với sở thích và năng lực của mình nhất.
* Ngoài ra, chương này còn:
  + **Giới thiệu các công cụ và phần mềm** (gọi là "tools of the trade" - công cụ nghề nghiệp) mà bạn sẽ cần sử dụng khi làm việc trong lĩnh vực phát triển web.

**Một số kiến thức bổ sung cực kỳ hữu ích cho bạn:**

**1. Các vai trò trong quá trình xây dựng website (sơ bộ):**

|  |  |
| --- | --- |
| Vai trò | Công việc chính |
| Web Designer | Thiết kế giao diện người dùng (UI), lựa chọn màu sắc, font chữ, bố cục, trải nghiệm người dùng (UX). |
| Front-end Developer | Viết mã HTML, CSS, JavaScript để xây dựng phần giao diện trang web mà người dùng tương tác. |
| Back-end Developer | Lập trình phần "hậu trường" của website: server, cơ sở dữ liệu, logic xử lý,... sử dụng các ngôn ngữ như PHP, Node.js, Python,... |
| Full-stack Developer | Làm cả front-end lẫn back-end. (Kiểu "chấp hết" 😆) |
| Database Administrator (DBA) | Quản lý hệ thống cơ sở dữ liệu (ví dụ như MySQL, MongoDB). |
| DevOps Engineer | Xây dựng hệ thống máy chủ, tối ưu hóa triển khai, tự động hóa quá trình vận hành web. |

🔖 **Lưu ý**:

* Ở công ty nhỏ, **một người có thể kiêm nhiều vai trò**.
* Ở công ty lớn, **mỗi vai trò có chuyên viên riêng**.

**2. Các công cụ và phần mềm phổ biến bạn sẽ dùng:**

|  |  |
| --- | --- |
| Công cụ | Dùng để làm gì |
| Text Editor (trình soạn thảo mã) | Viết mã (code). Ví dụ: Visual Studio Code, Sublime Text, Atom |
| Web Browser | Kiểm tra trang web. Các trình duyệt như Chrome, Firefox, Safari rất quan trọng. |
| Version Control System | Quản lý phiên bản code, như Git (và GitHub để lưu trữ dự án). |
| Graphics Software | Nếu thiết kế giao diện: Photoshop, Figma, Canva,... |
| Command Line Tools | Dùng để cài đặt, chạy server,... Ví dụ: Terminal (Linux/Mac), CMD/PowerShell (Windows). |
| Local Server | Dùng để chạy web trên máy cá nhân: WAMP, MAMP, LAMP, XAMPP,... |

**Một số lưu ý quan trọng thêm:**

* **Bạn không cần phải biết hết mọi thứ ngay lập tức**.  
  ➔ Chỉ cần chọn 1 vai trò, tập trung học từ những kiến thức cơ bản, sau đó mở rộng dần.
* **Ban đầu nên học Front-end trước** (HTML, CSS, JavaScript) vì:
  + Dễ thấy kết quả ngay (ví dụ: viết 1 dòng HTML → có thể xem ngay trên trình duyệt).
  + Tạo hứng thú, động lực cho quá trình học.
* **Sau này** bạn có thể mở rộng học thêm:
  + Back-end (PHP, Node.js,...)
  + Cơ sở dữ liệu (MySQL, MongoDB,...)
  + Các khái niệm nâng cao (API, bảo mật, tối ưu tốc độ,...)

**Tóm tắt dễ nhớ:**

|  |  |
| --- | --- |
| Ý chính | Ý nghĩa |
| Bạn sẽ luôn tự hỏi: "Mình bắt đầu từ đâu?" | Đó là cảm giác hoàn toàn bình thường |
| Bắt đầu bằng việc hiểu tổng quan nghề nghiệp | Biết rõ các vai trò, công việc, cơ hội |
| Làm quen với các công cụ làm việc | Text Editor, Browser, Git, Local Server,... |
| Lời khuyên vàng | Đừng cố học tất cả cùng lúc. Tập trung vào 1 mảng trước. |

### I. WHERE DO I START?

**BẮT ĐẦU TỪ ĐÂU?**

Có thể bạn đang đọc cuốn sách này vì:

* Bạn tham gia một **khóa học bài bản** về thiết kế và phát triển web.
* Bạn **tự học** để mở rộng kỹ năng cá nhân.
* Hoặc đơn giản là bạn **tò mò** muốn tìm hiểu thêm về cách hoạt động của web.

🌟 **Dù lý do nào đi nữa**, cuốn sách này là một **khởi đầu rất tuyệt vời** để bạn **hiểu cách web vận hành** và **bắt đầu học web một cách bài bản**.

**Mức độ tham gia vào lĩnh vực thiết kế web rất đa dạng**

Bạn có thể:

* **Tự xây dựng một trang web nhỏ cho bản thân**,
* Hoặc **theo đuổi sự nghiệp chuyên nghiệp** trong ngành công nghiệp thiết kế và phát triển web.

🎯 Ngoài ra, bạn có thể chọn:

* **Trở thành một "Full-Stack Developer"**, tức là biết cả **giao diện** (front-end) và **hậu trường** (back-end),
* Hoặc **chuyên sâu vào một kỹ năng cụ thể** (ví dụ chỉ chuyên làm Front-end hoặc chỉ chuyên Back-end).

👉 **Có rất nhiều hướng đi khác nhau**, bạn hoàn toàn có thể chọn lựa theo sở thích và mục tiêu cá nhân!

**Nếu bạn muốn theo đuổi nghề nghiệp chuyên nghiệp:**

Nếu bạn **muốn biến web design hoặc web development thành sự nghiệp chính thức**, thì:

* Bạn **không nhất thiết phải có bằng cấp chuyên ngành** (ví dụ: bằng cử nhân thiết kế web),
* **Nhưng nhà tuyển dụng sẽ yêu cầu bạn chứng minh năng lực thực tế**, thông qua các **dự án mẫu** (portfolio).

Các dự án mẫu có thể bao gồm:

* **Bài tập lớn/nhóm trong khóa học**,
* **Dự án cá nhân tự thực hiện**,
* **Website cho một doanh nghiệp nhỏ hay một tổ chức**.

🔖 **Điều quan trọng là**:

* Website của bạn **phải trông chuyên nghiệp**,
* **Mã nguồn HTML, CSS, JavaScript** đằng sau website **phải gọn gàng, sạch sẽ, dễ hiểu**,
* Thể hiện được khả năng của bạn trong việc **thiết kế và lập trình web**.

🛠️ (**Tớ bổ sung thêm**: Portfolio tốt chính là "bằng chứng sống" về kỹ năng của bạn, cực kỳ quan trọng trong ngành này đó!)

**Nếu bạn chỉ muốn làm một website nhỏ cho riêng mình:**

Nếu bạn **không có ý định đi chuyên sâu** mà chỉ:

* Muốn làm 1–2 website cá nhân,
* Hoặc tạo website cho sở thích cá nhân, shop nhỏ,...

👉 Thì bạn **không cần phải thành lập trình viên chuyên nghiệp**.

💡 **Giải pháp nhanh** cho bạn:

* **Sử dụng các dịch vụ website dựng sẵn**,
* **Chọn mẫu (template)** và **kéo-thả** để xây dựng website mà **không cần biết code**.

💬 (Tuy nhiên: Nếu bạn biết thêm một chút HTML/CSS, bạn sẽ **tùy chỉnh giao diện đẹp hơn, chuyên nghiệp hơn** đó nha!)

**🌟 Ghi chú thêm: "Tôi chỉ muốn có một website riêng"**

Bạn **không cần** trở thành lập trình viên chuyên nghiệp để có website!  
Hiện nay, có rất nhiều **dịch vụ xây dựng website dễ dàng**, cung cấp:

* **Kho mẫu giao diện đẹp** (templates),
* **Công cụ kéo-thả**,
* **Hệ thống hỗ trợ quản lý nội dung** (CMS - Content Management System), giúp bạn tạo ra một trang web chuyên nghiệp mà **không cần viết dòng code nào**.

Các dịch vụ phổ biến (tính đến thời điểm viết sách) bao gồm:

|  |  |
| --- | --- |
| Dịch vụ | Đặc điểm nổi bật |
| WordPress.com | Hệ thống blog và website lớn nhất thế giới. Cực kỳ linh hoạt, nhiều giao diện và plugin. |
| Squarespace | Thiết kế cực kỳ đẹp, mượt mà, phù hợp cho portfolio cá nhân, website nghệ thuật. |
| Wix | Kéo-thả đơn giản, dễ dùng, nhiều template đẹp cho mọi ngành nghề. |
| SiteBuilder | Nhanh chóng, giao diện trực quan, dễ khởi tạo website nhỏ. |
| Weebly | Thân thiện với người mới, hỗ trợ cả blog và bán hàng online. |

💡 **Lưu ý thêm**:

* Một số dịch vụ này thiên về **website cá nhân** (ví dụ blog cá nhân, portfolio).
* Một số khác hỗ trợ **thương mại điện tử** tốt hơn (ví dụ như Squarespace, Wix).
* Nếu bạn muốn bán hàng online, hãy chọn dịch vụ hỗ trợ **giỏ hàng**, **thanh toán trực tuyến**, v.v.

**🛠️ Một vài thông tin bổ sung rất hữu ích:**

**1. Hosting service là gì?**

* Là nơi **lưu trữ website** của bạn để người khác trên Internet có thể truy cập được.
* Các dịch vụ như WordPress.com, Squarespace, Wix,... đều **cung cấp luôn hosting**.

**2. CMS (Content Management System) là gì?**

* Là hệ thống giúp bạn **quản lý nội dung** website dễ dàng, mà không cần viết code phức tạp.
* Ví dụ: WordPress chính là một CMS.

**📌 Tóm lại:**

|  |  |
| --- | --- |
| Ý chính | Ý nghĩa |
| Có nhiều cách tiếp cận web | Học chuyên sâu hoặc chỉ dựng website cá nhân đơn giản |
| Nếu theo nghề | Cần portfolio tốt, không nhất thiết cần bằng đại học |
| Nếu chỉ làm website nhỏ | Dùng dịch vụ dựng web nhanh như WordPress, Wix, Squarespace |
| Luôn có lựa chọn phù hợp | Tùy mục tiêu và mong muốn của bạn |

🎯 **Kết luận dễ thương**:  
Dù bạn muốn **trở thành lập trình viên chuyên nghiệp** hay chỉ đơn giản **tạo một góc nhỏ trên Internet cho riêng mình**, thì hành trình bắt đầu từ việc **hiểu bản thân muốn gì** và **chọn cách học phù hợp nhất với mình**!

### II. IT TAKES A VILLAGE (WEBSITE CREATION ROLES)

Khi nhìn vào một trang web, tớ không chỉ thấy mỗi dòng code hay hình ảnh đẹp đẽ đâu. Đằng sau một website là **cả một quá trình** với rất nhiều quyết định và chuyên môn khác nhau kết hợp lại.  
Một website thường không tự dưng mà có. Nó **thường bắt đầu từ một kế hoạch kinh doanh** hoặc một mục tiêu cụ thể nào đó. Trước khi website chính thức "lên sóng" (launch), sẽ có **hàng loạt công việc** phải làm như:

* **Sáng tạo nội dung** (viết bài, mô tả sản phẩm, hình ảnh, video,...)
* **Tổ chức, sắp xếp thông tin** hợp lý
* **Nghiên cứu** (nghiên cứu từ khóa, thị trường, đối thủ,...)
* **Thiết kế tổng thể** (giao diện, màu sắc, trải nghiệm người dùng UX/UI)
* **Viết mã code** (HTML, CSS, JavaScript, backend,...)
* **Quản lý máy chủ (server)** để đảm bảo website hoạt động ổn định

💡 **Ghi nhớ**: Một website không chỉ là sản phẩm của lập trình viên (coder) đâu nha, mà còn là kết quả của **nhiều lĩnh vực** khác nhau cùng làm việc với nhau.

**Quy mô của các đội xây dựng website**

* **Các website lớn, nổi tiếng** (ví dụ như Amazon, Facebook, Google,...) có thể có **hàng trăm thậm chí hàng ngàn người** cùng tham gia vào quá trình phát triển và vận hành.
* **Các website nhỏ** (ví dụ như web công ty nhỏ, blog cá nhân, shop online nhỏ) thì chỉ cần **vài người** thôi là đủ.
* **Và cũng hoàn toàn có thể** xây dựng một website **một mình** nếu bạn chịu khó học đủ các kỹ năng cần thiết.

🎯 **Điều tuyệt vời của web**: Dù bạn chỉ là một cá nhân, bạn vẫn có thể tạo ra một website cực kỳ ấn tượng nếu biết cách kết hợp các kỹ năng đúng đắn!

**Các chuyên môn chính khi tạo website**

Trong phần này, tác giả sẽ giới thiệu cho bạn các **chuyên môn chính** cần có khi xây dựng một website, bao gồm:

* **Content (Nội dung)**: Viết bài, tạo hình ảnh, quay video, chỉnh sửa media.
* **Design (Thiết kế)**: Giao diện, màu sắc, kiểu chữ, trải nghiệm người dùng (UX/UI Design).
* **Coding (Lập trình)**: Viết code cho giao diện (frontend) và hệ thống hoạt động bên trong (backend).

👉 **Bạn có thể chọn chuyên sâu vào 1 mảng** nếu muốn trở thành chuyên gia, hoặc nếu tự làm website, bạn sẽ phải **kiêm luôn nhiều vai trò** (như kiểu ở nhà vừa phải nấu ăn, dọn dẹp, làm kế toán,... vậy đó).

🧠 **Ví dụ thực tế dễ hiểu hơn**:

* Khi ở nhà, bạn vừa làm **đầu bếp** nấu ăn, vừa làm **kế toán** quản lý tiền, vừa làm **thợ sửa chữa** vá cái quạt,... Bạn không nghĩ nhiều, cứ làm thôi.
* Khi tự làm web cũng vậy, bạn vừa **viết nội dung**, vừa **thiết kế giao diện**, vừa **chỉnh sửa mã nguồn**,... nhưng dần dần bạn sẽ thấy nó rất tự nhiên và "vào guồng".

**Một số kiến thức mở rộng thêm cho cậu:**

🌟 **Các vai trò phổ biến khác trong ngành web**:

* **Project Manager (Quản lý dự án)**: Điều phối công việc giữa các thành viên trong team.
* **SEO Specialist (Chuyên gia SEO)**: Tối ưu hóa website để nó dễ dàng xuất hiện trên Google.
* **Content Strategist (Chiến lược nội dung)**: Lên kế hoạch nội dung dài hạn.
* **Database Admin (Quản trị cơ sở dữ liệu)**: Xây dựng và bảo trì dữ liệu website (nhất là các web thương mại điện tử lớn).

🌟 **Lưu ý thực tế**:

* **Nếu đi làm cho công ty lớn**: Bạn chỉ cần giỏi chuyên môn một lĩnh vực.
* **Nếu làm freelance (làm tự do)** hoặc **tự kinh doanh**: Càng biết nhiều kỹ năng càng tốt nha!

**Tóm lại siêu ngắn gọn cho cậu nè:**

Xây dựng website = nhiều lĩnh vực phối hợp (kế hoạch → nội dung → thiết kế → code → vận hành server).  
Dù đội lớn hay chỉ mình bạn, bạn đều có thể tạo ra web đẹp nếu biết kết hợp các kỹ năng.  
Bạn có thể chọn chuyên sâu 1 lĩnh vực hoặc làm "đa năng" tùy mục tiêu của mình!

#### II.1. CONTENT WRANGLING (QUẢN LÝ NỘI DUNG)

**Ý chính:**

* Khi chúng ta làm web, **mục tiêu cuối cùng** là **truyền đạt nội dung**, **thông điệp** hoặc **chức năng** đến người dùng.
* Vì vậy, **nội dung** cực kỳ quan trọng, không chỉ là những bài viết dài mà còn gồm cả **các nhãn nút bấm**, **tin nhắn lỗi**, **hướng dẫn**, v.v.
* Người viết nội dung (writer) và biên tập viên (editor) là những thành viên **rất quan trọng** trong đội ngũ làm web.

**Chi tiết thêm:**

* Một trang web có thể có **nội dung động** (ví dụ: tin tức cập nhật liên tục) hoặc **nội dung tĩnh** (ví dụ: giới thiệu công ty).
* Để nội dung chất lượng và có tổ chức tốt, cần sự phối hợp giữa:
  + **Người viết nội dung** (Content Writers)
  + **Người biên tập nội dung** (Editors)
  + **Chuyên gia kiến trúc thông tin** (Information Architect)
  + **Chiến lược gia nội dung** (Content Strategist)

#### II.2. INFORMATION ARCHITECTURE (KIẾN TRÚC THÔNG TIN)

**Ý chính:**

* Người làm kiến trúc thông tin sẽ **sắp xếp nội dung** sao cho **dễ tìm**, **hợp lý**, **khoa học**.
* Công việc này không chỉ liên quan đến cách sắp xếp mà còn liên quan đến:
  + Chức năng **tìm kiếm** trên trang web
  + Sơ đồ trang web (Site Map)
  + Cách tổ chức dữ liệu, thư mục trên server

**Lưu ý mở rộng:**

* **Information Architecture (IA)** còn ảnh hưởng trực tiếp đến:
  + **Trải nghiệm người dùng (UX)**: người dùng có thấy dễ dàng tìm thứ họ cần không?
  + **Giao diện người dùng (UI)**: giao diện có trực quan và hợp lý không?
  + **Quản lý nội dung (Content Management)**: dễ thêm mới, sửa đổi nội dung không?
* Nếu cậu là người **thích sự gọn gàng, khoa học, logic**, thích **phân loại** (taxonomy), thì nghề IA sẽ cực hợp đó!

**Tham khảo thêm:**

* Cuốn sách "Information Architecture: For the Web and Beyond" - Louis Rosenfeld & Peter Morville.

#### II.3. CONTENT STRATEGY (CHIẾN LƯỢC NỘI DUNG)

**Ý chính:**

* **Chiến lược nội dung** đảm bảo **mọi chữ viết** trên trang web (từ bài dài đến cái nhãn nút) phải:
  + Phù hợp với **thương hiệu** (brand identity)
  + Đúng với **mục tiêu marketing** của tổ chức.

**Công việc của Content Strategist:**

* **Lập kế hoạch nội dung dài hạn**: dự trữ nội dung, lịch cập nhật, tái sử dụng nội dung.
* **Xây dựng giọng điệu truyền thông**: cả trên web lẫn mạng xã hội (social media).
* **Tạo mô hình dữ liệu nội dung**: để hệ thống quản lý dễ dàng.

**Lưu ý thêm:**

* Content Strategy rất quan trọng với các site lớn, vì nếu nội dung **lộn xộn** hoặc **sai tone** thì sẽ ảnh hưởng cực xấu đến hình ảnh thương hiệu.

**Tham khảo thêm:**

* Cuốn sách "Content Strategy for the Web" - Kristina Halvorson & Melissa Rich.

#### II.4. ALL MANNER OF DESIGN (TẤT TẦN TẬT VỀ THIẾT KẾ)

**Ý chính:**

* Nghe "thiết kế" (design) tưởng đơn giản, nhưng trong thực tế làm web, **thiết kế** chia thành **nhiều chuyên ngành nhỏ**.
* Người "designer" có thể đảm nhiệm một hoặc nhiều (thậm chí tất cả) các vai trò này.

**Một số vai trò thiết kế phổ biến:**

|  |  |
| --- | --- |
| Vai trò | Nội dung công việc |
| Visual Designer (Thiết kế giao diện) | Tạo ra hình ảnh, màu sắc, bố cục đẹp mắt |
| UI Designer (Thiết kế giao diện người dùng) | Thiết kế cách người dùng tương tác với web (nút bấm, biểu mẫu...) |
| UX Designer (Thiết kế trải nghiệm người dùng) | Đảm bảo người dùng thấy trang web dễ sử dụng, tiện lợi, thích thú |
| Interaction Designer (Thiết kế tương tác) | Thiết kế các hiệu ứng, cách các thành phần động trên web phản hồi với người dùng |
| Graphic Designer (Thiết kế đồ họa) | Thiết kế logo, banner, poster, icon... phục vụ cho trang web |

**Lưu ý thêm:**

* Thực tế, ở công ty nhỏ hoặc khi làm freelancer, một người có thể phải kiêm nhiệm nhiều vai trò thiết kế khác nhau cùng lúc.

**🎯 Tóm lại, cậu cần nhớ nè:**

* **Làm website** không chỉ có code!
* Cần:
  + Người quản lý nội dung (content)
  + Người sắp xếp thông tin (information architect)
  + Người xây dựng chiến lược nội dung (content strategist)
  + Người thiết kế hình ảnh và trải nghiệm (designers)
  + Và lập trình viên (developers) nữa.

👉 Nếu làm **một mình**, cậu sẽ phải "đội nhiều chiếc mũ" giống như trong nhà thì lúc thì làm bếp, lúc thì quét nhà, lúc thì sửa điện vậy! Nhưng đừng lo, dần dần sẽ thấy vui và quen tay thôiiii 💪💖

#### II.5. USER EXPERIENCE, INTERACTION, AND USER INTERFACE DESIGN:

**🎨 Trải nghiệm người dùng, Tương tác và Giao diện người dùng là gì?**

Khi nhắc đến "thiết kế", người ta thường nghĩ ngay đến **hình thức bên ngoài** – màu sắc, phông chữ, hình ảnh đẹp mắt. Tuy nhiên, **trên web**, việc quan trọng đầu tiên **không phải** là chọn màu hay font chữ, mà là **thiết kế cách mà website hoạt động**, **cách người dùng tương tác với nó**.

Trước khi trang web đẹp, nó **phải dễ dùng**, **rõ ràng mục tiêu**, và **thuận tiện cho người dùng di chuyển (navigating)**.  
Những công việc này thuộc về ba lĩnh vực chính:

* **User Experience (UX) design** – Thiết kế trải nghiệm người dùng
* **Interaction Design (IxD)** – Thiết kế tương tác
* **User Interface (UI) design** – Thiết kế giao diện người dùng

👉 **Lưu ý:** 3 lĩnh vực này thường **chồng lấn** nhau trong thực tế. Một người hoặc một nhóm nhỏ có thể cùng lúc làm cả UX, IxD và UI.

**🧠 User Experience (UX) Design – Thiết kế trải nghiệm người dùng**

UX Design không chỉ dừng lại ở việc web "nhìn đẹp", mà còn phải đảm bảo **toàn bộ trải nghiệm** của người dùng với sản phẩm là **tốt đẹp**.

**Cốt lõi của UX là:**

* Hiểu **người dùng**: Họ cần gì? Gặp khó khăn ở đâu?
* Làm cho website dễ hiểu, dễ học, dễ sử dụng.

**Donald Norman** – người đã đặt ra thuật ngữ **User Experience** – định nghĩa UX là:

“Tất cả các khía cạnh về sự tương tác của người dùng với sản phẩm: cách họ **cảm nhận**, **học cách dùng** và **sử dụng** nó.”

Trong một website/app, UX sẽ bao gồm:

* Thiết kế hình ảnh (Visual Design)
* Thiết kế giao diện (UI)
* Nội dung (Content) và cách truyền tải thông điệp
* Tốc độ tải trang, hiệu suất tổng thể

**Mục tiêu cuối cùng**:  
👉 Trải nghiệm người dùng **phải phù hợp** với thương hiệu và mục tiêu kinh doanh của tổ chức.

**🤝 Interaction Design (IxD) – Thiết kế tương tác**

Trong Interaction Design, mục tiêu là:

* Làm cho website dễ sử dụng (**easy**)
* Nhanh gọn, tiết kiệm thao tác (**efficient**)
* Mang lại sự hài lòng, thích thú (**delightful**)

Interaction Design tập trung nhiều vào:

* **Cách người dùng thao tác:** click, swipe, kéo-thả, điền form...
* **Luồng di chuyển tự nhiên:** người dùng phải thấy hành trình mượt mà, không bị lạc.

**🎛️ User Interface (UI) Design – Thiết kế giao diện người dùng**

UI Design là phần thiên về **giao diện hiển thị**:

* Bố cục sắp xếp nội dung (layout)
* Các công cụ điều hướng (buttons, links, menu...)
* Hình thức và chức năng của từng thành phần

👉 UI Design **thu hẹp hơn** so với UX: nó lo phần **"giao diện nhìn thấy và bấm vào"**.

**📄 Những sản phẩm (deliverables) mà UX, UI hoặc Interaction Designers tạo ra:**

**1. 📑 Báo cáo nghiên cứu và kiểm tra người dùng (User research and testing reports)**

* Đây là bước **rất quan trọng** trong thiết kế lấy người dùng làm trung tâm (**User-Centered Design – UCD**).
* Gồm việc **phỏng vấn**, **quan sát** người dùng để tìm ra:
  + Họ cần gì?
  + Họ gặp rào cản nào?
  + Họ mong đợi website vận hành ra sao?
* Trong suốt quá trình phát triển, các nhà thiết kế cũng thường xuyên làm **User Testing** để kiểm tra:
  + Người dùng có hiểu giao diện không?
  + Họ có biết cách tìm nội dung, hoàn thành các tác vụ không?
* Nếu kết quả không tốt ➔ Phải quay lại sửa đổi (gọi là **Iterative Design**).

**2. 🖼️ Wireframe diagrams – Sơ đồ khung trang**

* Wireframe là **bản vẽ đơn giản** chỉ gồm **các khung hình** đại diện cho:
  + Các khối nội dung
  + Thanh menu, ô tìm kiếm, nút bấm, biểu mẫu,...
* KHÔNG có màu sắc, font chữ fancy gì hết 👉 để người xem **tập trung vào cấu trúc bố cục** chứ không bị "mê mẩn cái đẹp".
* Thường kèm chú thích hướng dẫn cho nhóm phát triển hiểu cách xây dựng tính năng.

👉 Ví dụ: Ở đây sẽ có nút “Đăng ký”, nhấn vào thì popup xuất hiện.

**3. 🌐 Site diagram – Sơ đồ cấu trúc trang web**

* Đây là **sơ đồ tổng thể** cho thấy:
  + Các trang trên website liên kết với nhau như thế nào
  + Hệ thống điều hướng (navigation system) ra sao
* Dùng để:
  + Lên kế hoạch cấu trúc
  + Đảm bảo người dùng không bị lạc trên site

👉 Lưu ý: Với những website lớn, sơ đồ có thể dài hàng mét, dán kín cả bức tường luôn đó!

**4. 🎬 Storyboards và User flow charts – Bảng phân cảnh và sơ đồ dòng người dùng**

* **Storyboard**:
  + Giống như truyện tranh: mỗi "khung cảnh" mô tả 1 bước của người dùng trong hành trình sử dụng website/app.
  + Chẳng hạn: vào trang chủ → tìm sản phẩm → thêm vào giỏ → thanh toán.
* **User Flow Chart**:
  + Mang tính kỹ thuật hơn.
  + Mô tả chính xác việc: hành động nào sẽ kích hoạt quy trình kỹ thuật nào (server xử lý, hiện popup, gửi email xác nhận,...).

**📚 Một số cuốn sách kinh điển về UX, Interaction, UI mà cậu nên đọc thêm nè:**

|  |  |  |
| --- | --- | --- |
| Tựa sách | Tác giả | Ghi chú |
| The Elements of User Experience | Jesse James Garrett | Cơ bản và hệ thống về UX |
| Don't Make Me Think (Revisited) | Steve Krug | Dễ hiểu, thực tế, ai làm web cũng nên đọc |
| The Design of Everyday Things | Don Norman | Kinh điển về cách thiết kế mọi thứ dễ sử dụng |
| About Face: The Essentials of Interaction Design | Alan Cooper và nhóm | Chuyên sâu về thiết kế tương tác |
| Designing Interfaces | Jenifer Tidwell | Các mẫu thiết kế giao diện chuẩn |
| 100 Things Every Designer Needs to Know about People | Susan Weinschenk | Tâm lý học ứng dụng vào thiết kế |
| Designing User Experience | David Benyon | Giới thiệu tổng quan về UX, HCI, Interaction Design |

**📌 Ghi chú thêm cho cậu nè:**

* **Đừng nhầm lẫn** UX (tổng thể trải nghiệm) với UI (giao diện nhìn thấy).
* **Càng hiểu rõ người dùng**, web của mình càng hiệu quả.
* Luôn **kiểm thử với người dùng thật** – không đoán mò.
* **Wireframe** khác với **Mockup**: Wireframe chỉ là cấu trúc khung; còn Mockup là bản vẽ chi tiết hơn, có màu sắc, font chữ, hình ảnh...

Visual (graphic) design

Được luônnnnnn, tớ sẽ giải thích lại thật chi tiết, sâu rộng và dễ hiểu nhất cho cậu nhé, đồng thời sẽ thêm một số lưu ý và bổ sung kiến thức liên quan để cậu hiểu gốc rễ vấn đề luôn. Đây nè:

**🎨 Thiết kế Thị giác (Visual/Graphic Design)**

Vì web là một **phương tiện trực quan (visual medium)** — tức là mọi thứ hiển thị chủ yếu bằng hình ảnh, màu sắc, bố cục — nên **giao diện của trang web** đóng vai trò cực kỳ quan trọng.

✨ **Ấn tượng đầu tiên là tất cả**: Người dùng sẽ đánh giá ngay website chỉ trong vài giây đầu tiên dựa trên giao diện, dù nội dung có hay đến đâu.

**1. Vai trò của nhà thiết kế đồ họa (Graphic Designer)**

* **Tạo nên "diện mạo và cảm xúc" (Look and Feel)** của trang web.
  + **Logo**
  + **Hình ảnh**
  + **Kiểu chữ (Fonts)**
  + **Màu sắc**
  + **Bố cục (Layout)**
* Đảm bảo giao diện **đẹp, chuyên nghiệp, dễ dùng** và **phù hợp với thương hiệu** (brand) cũng như **thông điệp** (message) của tổ chức/doanh nghiệp.

💡 **Lưu ý**: Một website dù đẹp đến đâu, nếu không đồng bộ với thương hiệu (ví dụ web trường học mà màu mè như game trẻ em) thì cũng thất bại.

**2. Các phương pháp trình bày thiết kế giao diện**

Khi cần trình bày ý tưởng thiết kế cho khách hàng hoặc đối tác, các designer thường dùng:

* **Sketches** (Phác thảo tay): Vẽ sơ sơ các ý tưởng bố cục, vị trí các thành phần chính.
* **Mockups** (Bản mô phỏng):
  + Dùng phần mềm như **Photoshop**, **Figma**, **Sketch**, **Adobe XD**...
  + Tạo ra bản thiết kế trông giống thật (nhưng chưa hoạt động được) để mọi người hình dung website khi hoàn thiện.

**3. Thách thức khi thiết kế cho nhiều loại màn hình**

Ngày nay, web không chỉ xem trên máy tính mà còn trên:

* Điện thoại
* Máy tính bảng
* Tivi thông minh

Vì vậy:

* **Một thiết kế cố định (fixed layout)** cho một kích thước màn hình **là không đủ**.
* Cần **một ngôn ngữ thị giác** (visual language) linh hoạt:
  + Thống nhất trước về **màu sắc**, **kiểu chữ**, **cách dùng ảnh**, **phong cách hình ảnh**...
  + Sau đó, áp dụng ngôn ngữ đó vào các thiết kế cho từng loại màn hình khác nhau (responsive design).

**4. Công cụ hỗ trợ: Style Tiles**

* **Style Tiles** là một kỹ thuật do **Samantha Warren** đề xuất.
* Nó giúp:
  + **Tách biệt** phong cách thiết kế (màu sắc, font, phong cách nút bấm, hình ảnh...) ra khỏi bố cục cụ thể.
  + Thống nhất phong cách **ngay từ đầu**, trước khi vào sản phẩm chi tiết.
* Ví dụ:
  + Một bảng thể hiện: màu chủ đạo #0055FF, font chính Roboto, nút bấm tròn, hình ảnh phong cách tối giản, v.v...

👉 Có thể tải template và tìm hiểu thêm ở trang [styletil.es](https://styletil.es).

**5. Các nhiệm vụ khác của designer**

* **Tối ưu hóa hình ảnh**:
  + Nén hình ảnh đúng cách để web tải nhanh hơn.
  + Hiểu cách làm cho hình ảnh **phù hợp với nhiều màn hình**.
  + Mặc dù đội kỹ thuật (developers) cũng có thể làm việc này, nhưng **designer nên nắm được cơ bản**.
* **Tạo Style Guide (Hướng dẫn phong cách)**:
  + Một tài liệu ghi lại:
    - Font chữ nào dùng.
    - Màu sắc nào áp dụng.
    - Các nguyên tắc thiết kế khác.
  + Giúp cho trang web duy trì **tính nhất quán lâu dài**, cả khi có nhiều nhóm/cá nhân khác nhau cùng làm việc sau này.

👉 Tham khảo thêm nhiều tài liệu mẫu tại [styleguides.io](https://styleguides.io).

**📌 Ghi chú thêm: Element Collages của Dan Mall**

* **Element Collage** là một phương pháp tương tự Style Tiles.
* Thay vì chỉ chọn màu và font, **collage các thành phần giao diện lại** như:
  + Một mẫu nút bấm,
  + Một đoạn văn bản,
  + Một mảnh giao diện form...

👉 Xem bài viết chi tiết của Dan Mall tại: [danielmall.com/articles/rif-element-collages/](https://v3.danielmall.com/articles/rif-element-collages/)

**🧑‍💻 Designer có cần học lập trình không?**

**Câu trả lời là: Có!** (ít nhất ở mức cơ bản).

* Các designer web hiện đại **phải biết HTML và CSS**:
  + HTML (HyperText Markup Language): cấu trúc nội dung web.
  + CSS (Cascading Style Sheets): phong cách, bố cục, màu sắc.

🌟 Hiểu HTML và CSS giúp designer:

* Giao tiếp tốt hơn với lập trình viên.
* Thiết kế **khả thi** (tức là thứ mình vẽ ra lập trình viên có thể làm được).
* Tự mình tạo các **prototype hoạt động** để kiểm tra trước khi giao sản phẩm.
* **JavaScript**: Không bắt buộc, nhưng biết một ít thì càng tốt để hiểu cách các yếu tố tương tác hoạt động.

**🔥 Vì sao cần biết code ngày nay?**

* Trước đây, chỉ cần mockup bằng Photoshop xong rồi đưa cho coder.
* Nhưng bây giờ:
  + Web phải **responsive** (linh hoạt theo màn hình).
  + Thiết kế cố định (fixed-size mockup) **không còn đủ**.
* Do đó, **designer tự dựng prototype** có thể giúp:
  + Thể hiện đúng ý đồ thiết kế.
  + Trực quan hơn cho khách hàng.

**📚 Tổng kết nhanh**

|  |  |
| --- | --- |
| Mục | Nội dung |
| Thiết kế thị giác | Tạo giao diện web hấp dẫn, phù hợp thương hiệu. |
| Công cụ | Sketches, Mockups, Style Tiles, Element Collages. |
| Yêu cầu thêm | Biết tối ưu hình ảnh, tạo Style Guide. |
| Kỹ năng cần có | Biết HTML/CSS cơ bản, có thể JavaScript. |
| Xu hướng hiện đại | Tự tạo prototype responsive, linh hoạt cho nhiều màn hình. |

#### II.6. CODE SLINGING

**🛠 Code Slinging là gì?**

"Code slinging" (dịch nôm na là "quăng code" hay "cày code") là một cách nói vui chỉ công việc **viết mã (code)** và **giải quyết các vấn đề kỹ thuật** liên quan đến việc xây dựng trang web. Đây là giai đoạn cực kỳ quan trọng trong quá trình làm website.

Trong thực tế, **một phần rất lớn** của việc tạo ra một trang web bao gồm:

* Viết và sửa lỗi **các tài liệu web** như file HTML.
* Tạo và chỉnh sửa **các file CSS** để định dạng giao diện, màu sắc, bố cục...
* Xây dựng **các tập lệnh (scripts)** như JavaScript để thêm các tính năng tương tác.
* Chuẩn bị và tối ưu hóa **các hình ảnh** được sử dụng trên trang web.

👉 Tóm lại: **Code Slinging = Tạo ra và sửa chữa các thành phần cấu thành nên website.**

**🏢 Ai đảm nhiệm công việc Code Slinging?**

Tại các **công ty thiết kế web** chuyên nghiệp, công việc **xây dựng và xử lý các file website** sẽ thường do **bộ phận phát triển (development)** hoặc **bộ phận sản xuất (production)** đảm nhiệm.

Tùy theo tổ chức, nhóm này sẽ:

* Tạo ra **các file trang web** riêng lẻ.
* Hoặc tạo ra **các mẫu trang web (templates)**, để sau đó hệ thống tự động lắp ráp thành các trang động (dynamic pages).

**💻 Phân loại công việc Code Slinging**

Việc phát triển website thường được chia thành **hai mảng chính**:

|  |  |
| --- | --- |
| Mảng | Mô tả chi tiết |
| Frontend Development *(Phát triển giao diện phía người dùng)* | Xây dựng tất cả những gì người dùng có thể nhìn thấy và tương tác được trên trang web, như bố cục, màu sắc, hình ảnh, font chữ, nút bấm, menu, hiệu ứng động, v.v.  👉 Các ngôn ngữ chính: **HTML, CSS, JavaScript**. |
| Backend Development *(Phát triển hệ thống phía máy chủ)* | Xây dựng những phần mà người dùng không nhìn thấy, như quản lý cơ sở dữ liệu, xử lý logic nghiệp vụ, kiểm soát người dùng đăng nhập, xử lý đơn hàng...  👉 Các ngôn ngữ phổ biến: **PHP, Python, Java, Node.js**, cùng với **hệ quản trị cơ sở dữ liệu** như MySQL, PostgreSQL,... |

📌 **Lưu ý:**

* Ở một số công ty, **frontend** và **backend** có thể do **hai đội khác nhau** phụ trách.
* Tuy nhiên, cũng có **nhiều trường hợp** chỉ có **một người** hoặc **một nhóm nhỏ** chịu trách nhiệm **cả frontend lẫn backend**. Người làm cả hai việc đó thường được gọi là **full-stack developer**.

**📚 Bổ sung thêm kiến thức:**

* **Full-Stack Developer**: Là người có kỹ năng cả về frontend và backend. Ví dụ: Họ vừa có thể thiết kế giao diện website bằng HTML/CSS/JS, vừa có thể lập trình backend xử lý dữ liệu với PHP, Node.js...
* **CMS (Content Management System)**: Ngày nay, một số công ty dùng **các hệ thống quản lý nội dung** như WordPress, Joomla, Drupal... để giảm bớt việc viết code tay từ đầu. Nhưng dù vậy, việc hiểu sâu về code vẫn cực kỳ quan trọng để chỉnh sửa, tối ưu website theo yêu cầu riêng.

**✨ Tổng kết dễ nhớ:**

* **Code Slinging = viết và xử lý tất cả các thành phần của trang web**.
* **Có hai mảng chính:** frontend (giao diện) và backend (hệ thống phía sau).
* **Ở công ty lớn:** thường chia frontend và backend cho các nhóm chuyên biệt.
* **Ở công ty nhỏ:** một người có thể làm cả hai.

#### II.7. FRONTEND DEVELOPMENT

**🖥️ Frontend Development: Làm gì ở phía "giao diện" của website?**

**Frontend development** (phát triển giao diện người dùng) liên quan đến **mọi thứ xuất hiện trực tiếp trên trình duyệt web** – nghĩa là những gì người dùng nhìn thấy và tương tác.  
Muốn làm tốt frontend, cậu cần **nắm vững 3 công nghệ cốt lõi**:

* **HTML** (tạo nội dung)
* **CSS** (thiết kế giao diện)
* **JavaScript** (tạo tương tác).

Giờ mình sẽ đi lần lượt từng món nha!

**1. 📄 Authoring/Markup - HTML (Ngôn ngữ đánh dấu nội dung)**

* **Authoring** nghĩa là "soạn nội dung" cho website. Việc soạn này không chỉ đơn giản là gõ chữ mà còn phải **gắn thẻ (markup)** để trình duyệt hiểu nội dung đó là gì: tiêu đề, đoạn văn, danh sách, ảnh, v.v.
* **HTML (HyperText Markup Language)** là ngôn ngữ đánh dấu được dùng để **xây dựng bộ khung** của trang web.

📚 **Lưu ý**: HTML **không phải là ngôn ngữ lập trình**! Nó **chỉ là ngôn ngữ đánh dấu** - tức là "gắn nhãn" cho các phần nội dung để trình duyệt đọc và hiểu.

* Phiên bản hiện nay phổ biến là **HTML5** (thực ra là HTML 5.2 hoặc mới hơn). HTML5 có thêm rất nhiều cải tiến, ví dụ như hỗ trợ video/audio, biểu mẫu mạnh mẽ hơn, và các thẻ ngữ nghĩa mới (như <section>, <article>, v.v.).

✏️ **Cách học HTML tốt nhất**: Cứ **viết tay từng trang** thật đơn giản, làm bài tập từ những thứ nhỏ nhặt trước (cậu yên tâm, từ từ sẽ lên trình nhanh lắm đó).

**2. 🎨 Styling - CSS (Tạo phong cách cho nội dung)**

* **HTML** chỉ giúp website **có nội dung**, nhưng **CSS (Cascading Style Sheets)** mới là thứ giúp **trang web đẹp hơn, dễ nhìn hơn**.
* CSS kiểm soát **giao diện** của website, ví dụ:
  + Font chữ
  + Màu sắc
  + Khoảng cách giữa các dòng, cột
  + Bố cục tổng thể
  + Hiệu ứng nền, hình động nhẹ,…
* **"Cascading"** nghĩa là "tính kế thừa" – nhiều style áp dụng cùng lúc, có quy tắc ưu tiên.

📚 **Lưu ý**:  
Ngoài việc thiết kế cho trình duyệt, CSS còn cho phép **tùy chỉnh giao diện khi in ra giấy** hoặc **cho người khiếm thị nghe bằng máy đọc màn hình** (nhưng phần này ít khi học sâu lúc đầu).

* Nếu chỉ dùng HTML thì trang web của cậu sẽ **xấu như năm 1990** vậy đó 😆.  
  **=> Nếu cậu muốn làm web chuyên nghiệp, thành thạo CSS là bắt buộc!**

**3. ⚡ JavaScript và DOM Scripting (Tạo website có thể tương tác)**

* **JavaScript** là **ngôn ngữ lập trình chính** để giúp trang web **sống động**, **biết phản hồi** khi người dùng thao tác.

Ví dụ JavaScript có thể làm:

* Kiểm tra người dùng nhập liệu đúng chưa (ví dụ bắt buộc nhập email hợp lệ)
* Đổi giao diện theo hành động người dùng (nhấn nút đổi màu giao diện chẳng hạn)
* Tự động tải thêm bài viết khi cuộn xuống cuối trang (như Facebook)
* Lưu thông tin người dùng (ví dụ "Ghi nhớ đăng nhập")
* Tạo các tiện ích như **video player nhúng**, **form nhập liệu đẹp đẽ**,…

**DOM Scripting** là gì?

* DOM (Document Object Model) là **mô hình đối tượng tài liệu** – cách trình duyệt hiểu và quản lý trang web như một **cây các phần tử**.
* **DOM Scripting** là việc dùng JavaScript để **truy cập, thay đổi, thêm/xóa các phần tử HTML** trên trang.

👉 Ví dụ dễ hiểu:

**document.getElementById("myTitle").style.color = "red";**

(Dòng lệnh trên đổi màu chữ của thẻ có id là myTitle thành màu đỏ.)

**Frontend developers** hiện nay cũng thường cần biết thêm:

* **Framework JavaScript** như:
  + **React** (Facebook phát triển, rất mạnh mẽ)
  + **Angular** (Google phát triển)
  + **VueJS** (nhẹ, dễ học)
  + **Bootstrap** (bộ công cụ CSS + JS dựng giao diện nhanh)
* **AJAX** (Asynchronous JavaScript and XML):
  + Kỹ thuật để **tải dữ liệu mới mà không cần tải lại trang**.
  + Ví dụ: Cậu nhấn "Xem thêm bình luận" trên Facebook mà trang không bị load lại.

📚 **Lưu ý**:

* Học JavaScript cần có tư duy lập trình tốt hơn một chút (khác với HTML và CSS vốn dễ vào hơn).
* Đừng nản! Lúc đầu ai cũng cảm thấy JavaScript hơi "khó thở" chút, nhưng học theo dự án nhỏ rồi nâng dần lên là OK.

**🌐 Thông tin thêm: W3C (World Wide Web Consortium)**

* **W3C** là tổ chức chịu trách nhiệm **định ra tiêu chuẩn** cho các công nghệ web như HTML, CSS, JavaScript,…
* Người sáng lập W3C là **Tim Berners-Lee**, cũng chính là người **phát minh ra World Wide Web**!
* Trang web chính thức: [www.w3.org](https://www.w3.org)

📚 **Lưu ý**:  
Nếu cậu có thắc mắc về quy chuẩn, cách dùng chính xác về HTML, CSS, JavaScript thì **W3C là nguồn chuẩn nhất** để tra cứu nha!

**🎯 Tổng kết nhanh nè:**

|  |  |
| --- | --- |
| Công nghệ | Vai trò |
| HTML | Tạo khung/nội dung trang web |
| CSS | Thiết kế giao diện, trình bày đẹp |
| JavaScript | Tạo các tương tác, làm web động |

👉 Nếu cậu muốn trở thành **Web Developer**, **Frontend Developer**, thì **phải giỏi 3 món này** đó nhaaaa!

Backend development:

Dạ cậu ơiiiiiiiiiii, đây tớ đã giải thích lại **chi tiết, sâu rộng và dễ hiểu hơn** theo phong cách gần gũi cho cậu nhaaaaaa, tớ cũng thêm một số thông tin bổ ích và lưu ý nhỏ cho cậu luôn nè:

#### II.8. BACKEND DEVELOPMENT (PHÁT TRIỂN PHÍA MÁY CHỦ)

**Backend là gì?**

Khi mình xây dựng một trang web, có hai phần chính:

* **Frontend**: Là những gì người dùng nhìn thấy và tương tác trực tiếp trên trình duyệt (HTML, CSS, JavaScript).
* **Backend**: Là những gì xảy ra **phía sau**, **trên máy chủ**, mà người dùng **không nhìn thấy** được.  
  Nhiệm vụ của Backend là xử lý logic, quản lý dữ liệu và đảm bảo mọi thứ chạy đúng đắn.

**Backend developers** (lập trình viên Backend) sẽ tập trung làm việc với:

* **Máy chủ** (server): nơi lưu trữ website và xử lý yêu cầu từ người dùng.
* **Ứng dụng**: các chương trình chạy trên máy chủ để xử lý chức năng website (như đăng nhập, đặt hàng...).
* **Cơ sở dữ liệu** (database): nơi lưu trữ thông tin như tài khoản người dùng, sản phẩm bán, bài viết...

**Công việc cụ thể của Backend developer:**

**1. Cài đặt và cấu hình phần mềm máy chủ**

* Máy chủ cần được cài đặt phần mềm để website hoạt động. Ví dụ: cài Apache, Nginx, hoặc dịch vụ cloud như AWS.
* Backend dev phải biết cách thiết lập, tinh chỉnh để website chạy ổn định và an toàn.

*(Tớ sẽ giải thích thêm rõ về máy chủ trong chương "How the Web Works" như sách nói.)*

**2. Lập trình phía máy chủ**

Backend dev **phải biết** ít nhất **một ngôn ngữ lập trình server-side**, ví dụ:

* **PHP**: rất phổ biến với các hệ quản trị nội dung như WordPress.
* **Python**: dễ học, mạnh mẽ (với framework như Django, Flask).
* **Ruby**: đi kèm Ruby on Rails, nổi tiếng trong giới startup.
* **.NET (hoặc ASP.NET)**: của Microsoft, mạnh về các hệ thống doanh nghiệp.
* **JSP**: Java Server Pages, dùng Java để phát triển web.

Họ sử dụng các ngôn ngữ này để xây dựng:

* Xử lý **form** (ví dụ: form đăng ký tài khoản, mua hàng).
* Quản lý nội dung bằng **CMS** (Content Management System) như WordPress, Joomla.
* Xây dựng **giỏ hàng trực tuyến**, **thanh toán online**, v.v.

**3. Quản lý cơ sở dữ liệu**

Backend developers còn phải làm việc với các **hệ quản trị cơ sở dữ liệu**, để lưu trữ dữ liệu cho website:

* **MySQL**: phổ biến nhất, mã nguồn mở, dễ dùng.
* **Oracle Database**: mạnh, chuyên dùng cho các doanh nghiệp lớn.
* **SQL Server**: của Microsoft, dùng trong môi trường Windows.

Họ phải:

* Thiết kế **cấu trúc bảng dữ liệu** (database schema).
* **Thêm, sửa, xóa, tìm kiếm** dữ liệu bằng ngôn ngữ SQL.
* Đảm bảo dữ liệu **an toàn, chính xác** và **hiệu suất cao**.

🔥 **Lưu ý:** Nếu mình làm web nhỏ, đôi khi không cần tự lập trình mà có thể dùng các dịch vụ sẵn có của nhà cung cấp hosting như giỏ hàng, đăng ký email tự động,... (gọi là prepackaged solutions).

#### II.9. FULL-STACK DEVELOPER VÀ UNICORNS

**Full-Stack Developer là ai?**

* Một người **thành thạo cả Frontend và Backend**.
* Nghĩa là họ vừa biết **HTML, CSS, JavaScript** để làm giao diện, vừa biết **PHP, Python, SQL** để xử lý server và database.

👉 Vì họ làm được mọi thứ nên rất được **các công ty ưa chuộng** (đặc biệt là các công ty nhỏ).

**Unicorn là gì?**

Trong ngành web, **"Unicorn"** không chỉ là "kỳ lân" trong truyền thuyết đâu.  
Ở đây, nó chỉ những người:

* Vừa biết làm nội dung (content strategy)
* Vừa làm giao diện người dùng (UX/UI)
* Vừa lập trình frontend
* Vừa lập trình backend
* Vừa lo server, vừa quản lý dữ liệu

👉 **Nói chung, một mình cân hết mọi khâu** làm ra một trang web hoàn chỉnh!

⚡️ **Rất hiếm có người như vậy** (giống kỳ lân ngoài đời vậy đó!), nhưng nếu cậu thành công thì cực kỳ đắt giá trên thị trường lao động luôn nha!

Cụm từ **"Full-Stack Developer và Unicorns"** có thể được dịch chuẩn sang tiếng Việt là:

**"Lập trình viên Full-Stack và các công ty kỳ lân"**

**Giải thích:**

* **Full-Stack Developer**: thường được giữ nguyên trong ngành công nghệ tại Việt Nam, nhưng có thể diễn giải là "Lập trình viên toàn diện (cả front-end và back-end)" nếu cần rõ nghĩa hơn.
* **Unicorns**: trong ngữ cảnh startup/công nghệ, "unicorns" là cách nói chỉ **các công ty khởi nghiệp có định giá từ 1 tỷ USD trở lên**, thường được dịch là **"các công ty kỳ lân"**.
* **Full**: đầy đủ, toàn bộ
* **Stack**: chồng, xếp chồng

Nếu dịch hoàn toàn sát nghĩa đen, **"Full-Stack Developer"** có thể hiểu là:

**"Lập trình viên phát triển với đầy đủ các tầng (công nghệ xếp chồng lên nhau)"**

Tuy nhiên, trong ngành công nghệ, **"Full-Stack Developer"** là một thuật ngữ chuyên môn. Nó chỉ người có khả năng làm việc ở **cả phía giao diện người dùng (front-end)** và **phía máy chủ (back-end)**, cùng với cơ sở dữ liệu, API, v.v.

Vì vậy, dịch đúng ngữ cảnh và phổ biến nhất là:

**"Lập trình viên Full-Stack"** (giữ nguyên thuật ngữ, như cách người trong ngành thường dùng ở Việt Nam)  
hoặc  
**"Lập trình viên toàn diện cả front-end và back-end"** (nếu cần giải nghĩa rõ ràng hơn)

**Tóm gọn để dễ nhớ:**

|  |  |  |
| --- | --- | --- |
| Hạng mục | Frontend | Backend |
| Người dùng thấy? | Có (trực tiếp trên trình duyệt) | Không (chạy ẩn phía sau máy chủ) |
| Công nghệ chính | HTML, CSS, JavaScript | PHP, Python, Ruby, .NET, Java, SQL |
| Công việc chính | Hiển thị giao diện đẹp và tiện | Xử lý logic, dữ liệu, bảo mật |
| Database liên quan? | Không trực tiếp | Có, phải biết làm việc với databases |

**Một số lưu ý thêm cho cậu:**

* Backend yêu cầu tư duy logic cao hơn Frontend, vì phải xử lý luồng dữ liệu phức tạp.
* Cơ hội nghề nghiệp cho Backend dev **rất nhiều**, mức lương cũng thường **cao**.
* Nhưng học Backend cần có nền tảng vững về:
  + Lập trình cơ bản
  + Quản trị cơ sở dữ liệu
  + Kiến thức hệ thống mạng (sơ cấp)
* Nếu muốn sau này làm Full-stack, cậu nên học Frontend trước rồi học dần Backend nha.

#### II.10. OTHER ROLES

**🌟 Các Vai Trò Khác Ngoài "Web Design"**

Khi xây dựng và duy trì một trang web, có rất nhiều người tham gia với nhiều vai trò khác nhau. Không phải ai cũng gọi là "web designer" (nhà thiết kế web), nhưng họ đều đóng góp quan trọng trong việc đưa một trang web hoạt động hiệu quả. Dưới đây là một số vai trò phổ biến:

**🔷 Product Manager (Quản lý sản phẩm)**

* **Là ai?**  
  Là người chịu trách nhiệm vạch ra chiến lược tổng thể cho sản phẩm (trang web hoặc ứng dụng) để đạt được mục tiêu kinh doanh.
* **Công việc chính:**
  + Hiểu rõ người dùng mục tiêu (target market): khách hàng là ai, họ cần gì, thói quen sử dụng ra sao.
  + Phối hợp cùng các nhóm thiết kế và phát triển để bảo đảm sản phẩm phù hợp với thị trường.
  + Xác định thời điểm ra mắt, cách ra mắt (release plan).
  + Đưa ra quyết định liên quan đến **tính năng**, **giao diện**, và **trải nghiệm người dùng** dựa trên phân tích thị trường.
* 📝 **Ghi nhớ:** Họ không trực tiếp viết mã hay thiết kế, nhưng họ *quyết định hướng đi tổng thể* cho cả dự án web.

**🔷 Project Manager (Quản lý dự án)**

* **Là ai?**  
  Là người lập kế hoạch, điều phối công việc, đảm bảo tiến độ và ngân sách của dự án web.
* **Công việc chính:**
  + Lập **timeline** (kế hoạch thời gian) cho toàn bộ dự án.
  + Phân công nhiệm vụ cho designer, developer, tester, v.v...
  + Theo dõi tiến độ và báo cáo kết quả với cấp trên hoặc với product manager.
  + Quản lý các vấn đề phát sinh và điều chỉnh kế hoạch phù hợp.
* 📝 **Ghi nhớ:** Họ giống như "đạo diễn" của bộ phim - không đứng trên sân khấu, nhưng họ điều khiển mọi thứ từ hậu trường để đảm bảo sản phẩm *ra đời đúng hạn*.

**🔷 SEO Specialist (Chuyên gia tối ưu công cụ tìm kiếm)**

* **Là ai?**  
  Người giúp trang web của bạn **hiện lên trên Google** khi người ta tìm kiếm.
* **Công việc chính:**
  + Điều chỉnh **cấu trúc website**, **thẻ HTML**, và **từ khóa** sao cho thân thiện với các công cụ tìm kiếm như Google, Bing.
  + Viết nội dung chuẩn SEO (thẻ title, meta description, h1, ...).
  + Theo dõi và cải thiện thứ hạng tìm kiếm (ranking) thông qua Google Analytics hoặc Google Search Console.
* 🧠 **Thông tin thêm:**  
  SEO không phải "mánh khóe đen tối", mà là nghệ thuật và khoa học để *tăng khả năng hiển thị* của trang web.  
  👉 Mẹo đơn giản mà hiệu quả nhất để tăng SEO là: **Nội dung chất lượng + HTML hợp lý**.

**🔷 Multimedia Producers (Nhà sản xuất đa phương tiện)**

* **Là ai?**  
  Là người tạo ra các nội dung **đa phương tiện** như:
  + Âm thanh 🎧
  + Video 📹
  + Hoạt hình (animation) 🎞️
  + Trò chơi tương tác (interactive games) 🎮
* **Công việc chính:**
  + Sử dụng công cụ đồ họa, phần mềm dựng phim, hoặc kỹ thuật tương tác để tạo nội dung hấp dẫn.
  + Làm việc với các web developer để tích hợp nội dung này vào trang web.
  + Đảm bảo nội dung đa phương tiện không làm trang web chậm đi hoặc gây lỗi trên các thiết bị khác nhau.
* 📝 **Ghi nhớ:**  
  Với các trang web thiên về giáo dục, giải trí hoặc truyền thông, multimedia là yếu tố cực kỳ quan trọng để thu hút người dùng.

**💡 Lưu ý thêm:**

* Ở **các công ty lớn**: Mỗi người chỉ làm đúng một mảng chuyên sâu (ví dụ: "UX Lead for Error Messages" – người chuyên thiết kế trải nghiệm cho... thông báo lỗi! 😮)
* Ở **các nhóm nhỏ hoặc freelancer**: Một người có thể làm nhiều việc cùng lúc, ví dụ:
  + Thiết kế giao diện (UI Design)
  + Tương tác người dùng (Interaction Design)
  + HTML/CSS
  + Nhưng không đụng tới: JavaScript, server backend, nội dung,...

**📚 Mục tiêu của cuốn sách này là gì?**

Tác giả muốn giúp bạn có **nền tảng vững vàng về frontend** (giao diện người dùng) bao gồm:

* HTML
* CSS
* Một phần nhỏ về thiết kế giao diện và trải nghiệm

Nhờ đó, bạn có thể đảm nhận nhiều vai trò khác nhau trong nhóm phát triển web, hoặc phối hợp hiệu quả với các thành viên khác trong team.

### III. GEARING UP FOR WEB DESIGN

**💬 Các Kỹ Năng Mềm Cần Thiết Cho Một Web Designer**

Ngoài các kỹ năng kỹ thuật như HTML, CSS, thiết kế giao diện,... thì còn một **nhóm kỹ năng khác rất quan trọng** nhưng thường bị bỏ qua — đó là **kỹ năng mềm (soft skills)**. Đây là những kỹ năng liên quan đến cách bạn **giao tiếp, làm việc nhóm, thích nghi và tư duy**, chứ không liên quan đến lập trình hay thiết kế.

Dưới đây là những kỹ năng mềm thiết yếu mà một web designer (hoặc bất kỳ ai làm việc trong lĩnh vực web) *rất nên có*:

**🗣️ 1. Kỹ năng giao tiếp xuất sắc (Excellent communication skills)**

* 💬 **Vì sao quan trọng?**  
  Làm việc với web không chỉ là "ngồi gõ code", mà bạn sẽ cần **trao đổi với khách hàng, đồng nghiệp, sếp, hoặc các bên liên quan khác**.  
  Việc bạn truyền đạt ý tưởng tốt sẽ giúp công việc trôi chảy hơn, giảm hiểu lầm, tiết kiệm thời gian.
* ✅ **Cần làm gì?**
  + Nói chuyện rõ ràng, dễ hiểu (dù là nói trực tiếp, email, hay nhắn tin qua Slack/Zalo).
  + Chủ động cập nhật tiến độ, hỏi lại nếu chưa rõ yêu cầu.
  + Biết **lắng nghe** ý kiến người khác và thể hiện sự tôn trọng.
  + Đặt câu hỏi để làm rõ vấn đề khi chưa chắc chắn.
* 🧠 **Lưu ý thêm:**  
  Giao tiếp tốt không chỉ giúp bạn làm việc hiệu quả hơn mà còn **tạo được ấn tượng chuyên nghiệp** với khách hàng và đồng nghiệp.

**🔄 2. Khả năng thích nghi (Flexibility)**

* 💥 **Vì sao cần thiết?**  
  Thế giới web thay đổi liên tục. Công nghệ mới, công cụ mới, yêu cầu khách hàng mới – mọi thứ có thể "quay ngoắt 180 độ" bất cứ lúc nào.
* 🌀 **Ví dụ thực tế:**
  + Hôm nay bạn làm giao diện cho trang web A, ngày mai khách hàng đột nhiên muốn... làm lại từ đầu.
  + Dự án bạn đang làm có thể bị hủy giữa chừng.
  + Bạn có thể được yêu cầu học thêm về thiết kế UX hoặc cả lập trình front-end để hỗ trợ team.
* ✅ **Cần làm gì?**
  + Luôn sẵn sàng học cái mới.
  + Không ngại thay đổi kế hoạch.
  + Luôn giữ tinh thần cầu tiến và chủ động.
* 🌱 **Lưu ý:**  
  Người "dẻo dai" trong công việc thường sống sót tốt hơn so với người quá cố định hoặc bảo thủ.

**🧠 3. Tư duy phản biện và phán đoán tốt (Critical thinking & good judgment)**

* 🧩 **Vì sao quan trọng?**  
  Web designer không chỉ "vẽ đẹp", mà còn phải **giải quyết vấn đề**:
  + Giao diện này có hợp lý với người dùng không?
  + Làm sao để trang web load nhanh hơn?
  + Tại sao nút này người dùng không nhấn?
* 🧠 **Tư duy phản biện là gì?**
  + Là khả năng **nhìn nhận vấn đề một cách logic**, **phân tích kỹ lưỡng** trước khi đưa ra quyết định.
  + Không làm việc theo thói quen hoặc cảm tính.
* ✅ **Cần làm gì?**
  + Đặt câu hỏi: "Tại sao?", "Làm thế nào để tốt hơn?", "Có giải pháp nào khác không?"
  + Không vội vàng hành động nếu chưa hiểu rõ vấn đề.
  + Biết đánh giá tình huống và đưa ra giải pháp hợp lý.
* 🔍 **Lưu ý thêm:**  
  Đây là kỹ năng cực kỳ quan trọng nếu bạn muốn phát triển sự nghiệp lâu dài — không chỉ làm theo mà còn *đóng góp giải pháp*.

**😃 4. Thái độ tốt (A good attitude)**

* 💡 **Vì sao quan trọng?**
  + Dù bạn làm việc tại công ty hay là freelancer (làm tự do), bạn **luôn phải làm việc với người khác**: khách hàng, đồng đội, quản lý...
  + Thái độ làm việc **ảnh hưởng trực tiếp đến không khí làm việc của cả nhóm**.
* 😇 **Thái độ tốt là như thế nào?**
  + Tôn trọng mọi người, sẵn sàng giúp đỡ.
  + Không đổ lỗi, không than vãn, không mang "năng lượng tiêu cực".
  + Vui vẻ hợp tác và biết ơn khi người khác giúp đỡ bạn.
* 🎯 **Ghi nhớ:**  
  Một người giỏi mà hay cáu gắt, kiêu căng thì không ai muốn làm việc cùng. Một người có **tinh thần tích cực** luôn được yêu mến và tin tưởng.

**🧠 Tổng kết lại nè:**

|  |  |  |
| --- | --- | --- |
| Kỹ năng mềm | Vì sao quan trọng? (Ý nghĩa) | Gợi ý thực hành |
| Giao tiếp | Giúp phối hợp trơn tru với team và khách hàng | Rèn luyện nói/nghe/viết rõ ràng |
| Thích nghi | Theo kịp công nghệ và xử lý tình huống bất ngờ | Luôn sẵn sàng học cái mới |
| Tư duy phản biện | Tìm giải pháp hiệu quả, làm việc logic | Đặt câu hỏi, phân tích kỹ |
| Thái độ tốt | Tạo môi trường làm việc tích cực, được mọi người quý | Biết lắng nghe, vui vẻ, chủ động |

#### III.1. EQUIPMENT

**🛠️ Chuẩn Bị Dụng Cụ & Thiết Bị Cho Thiết Kế Web (Gearing Up for Web Design)**

Khi bắt đầu học và làm thiết kế web chuyên nghiệp, bạn không chỉ cần kỹ năng mà còn cần chuẩn bị một số thiết bị phần cứng (hardware) và phần mềm (software) hỗ trợ. Một câu hỏi phổ biến là: **"Tôi cần phải mua những gì?"** Câu trả lời có thể khác nhau tùy vào nhu cầu cá nhân, nhưng dưới đây là các thiết bị cơ bản và quan trọng mà một nhà thiết kế web nên có.

**🖥️ 1. Máy tính mạnh và hiện đại**

* Bạn có thể dùng **Mac**, **Windows**, hoặc **Linux**, tùy theo bạn quen dùng hệ điều hành nào.
* Trong môi trường làm việc chuyên nghiệp:
  + Các nhóm thiết kế (front-end) thường dùng **Mac** vì hỗ trợ tốt cho đồ họa và UI/UX.
  + Các lập trình viên backend thì thường dùng **Linux** hoặc **Windows** vì chúng linh hoạt và hỗ trợ các môi trường server tốt.
* Tin vui là:

Các tệp web thường khá nhẹ (HTML, CSS, JavaScript), nên **không cần máy siêu mạnh** nếu bạn không làm chỉnh sửa âm thanh hoặc video.

* Lưu ý thêm: Hãy đảm bảo máy của bạn vẫn được cập nhật đầy đủ phần mềm và bảo mật.

**🖥️ 2. Màn hình lớn**

* Không bắt buộc, nhưng **rất hữu ích**.
* Màn hình lớn giúp bạn:
  + Mở được nhiều cửa sổ phần mềm cùng lúc (ví dụ: code editor, trình duyệt, Photoshop).
  + Xem được nhiều nội dung trang web khi thiết kế.
* ⚠️ Quan trọng: Dù bạn dùng màn hình lớn, **vẫn phải thiết kế web sao cho hiển thị tốt trên màn hình nhỏ hơn** như điện thoại và laptop nhỏ nhé!

**💻 3. Máy tính thứ hai để kiểm thử**

* **Tại sao cần máy thứ hai?**
  + Vì trình duyệt trên các hệ điều hành khác nhau có thể hiển thị web khác nhau.
  + Ví dụ: Web có thể trông đẹp trên Mac, nhưng bị lỗi trên Windows.
* Nếu bạn làm trên Mac, thì nên kiểm tra trên máy chạy Windows và ngược lại.
* 💡 Mẹo: Nếu không có 2 máy, bạn có thể:
  + Mượn máy bạn bè.
  + Dùng phần mềm giả lập như **BrowserStack**, **CrossBrowserTesting**, hoặc dùng **máy ảo (Virtual Machine)** để cài hệ điều hành khác trong máy hiện tại.

**📱 4. Thiết bị di động để kiểm tra**

* **Web đã trở nên di động rồi!** – Rất nhiều người truy cập web bằng **điện thoại** hoặc **máy tính bảng**.
* Vì vậy, bạn **phải kiểm tra trang web của mình trên thiết bị di động thật** để đảm bảo:
  + Giao diện không bị vỡ.
  + Các nút bấm không quá nhỏ.
  + Tốc độ tải trang không quá chậm trên mạng di động.
* 📌 Lưu ý: Bạn có thể dùng điện thoại cá nhân để kiểm tra, hoặc dùng công cụ giả lập trên trình duyệt (như DevTools trong Chrome → Ctrl + Shift + M).

**📷 5. Máy quét (scanner) và/hoặc máy ảnh (camera)**

* Nếu bạn muốn:
  + Tự tạo hình ảnh cho website (ví dụ: chụp ảnh sản phẩm, làm texture, vẽ tay rồi scan).
  + Sử dụng hình ảnh thật thay vì lấy từ Internet.
* ... thì bạn sẽ cần một máy ảnh (điện thoại cũng được nếu đủ chất lượng) hoặc một máy quét (để số hóa bản vẽ tay).

**📌 Tổng kết lại những gì cần chuẩn bị:**

|  |  |
| --- | --- |
| Thiết bị / Công cụ | Vai trò chính |
| Máy tính mạnh và cập nhật | Nền tảng làm việc chính (code, thiết kế, thử nghiệm) |
| Màn hình lớn | Dễ thao tác và quan sát khi thiết kế giao diện |
| Máy thứ hai (hoặc máy ảo) | Kiểm thử giao diện trên hệ điều hành khác |
| Điện thoại / máy tính bảng | Kiểm tra hiển thị trên thiết bị di động |
| Máy ảnh / máy quét | Tạo nội dung hình ảnh cá nhân hóa cho website |

**🧠 Gợi ý thêm từ tớ:**

* Nên có một **bộ phần mềm cần thiết** như:
  + **Trình soạn thảo mã (code editor)**: VS Code, Sublime Text, Atom...
  + **Trình duyệt để kiểm thử**: Chrome, Firefox, Safari, Edge...
  + **Phần mềm đồ họa**: Adobe Photoshop, Figma, Canva, GIMP…
  + **Công cụ kiểm thử responsive**: Chrome DevTools, Responsively App...
* **Lưu ý bảo mật**: Cài phần mềm diệt virus, tránh cài app lạ từ nguồn không tin cậy.
* **Sao lưu (backup)** thường xuyên để tránh mất dữ liệu.

#### III.2. WEB PRODUCTION SOFTWARE

**🧰 PHẦN MỀM HỖ TRỢ SẢN XUẤT TRANG WEB (Web Production Software)**

Ngày nay, bạn có rất nhiều lựa chọn phần mềm để thiết kế và tạo ra các trang web. Tuy nhiên, điều này không phải lúc nào cũng vậy.

**🕰️ Hồi xưa thì sao?**

* Vào những năm đầu của Internet, nhà thiết kế web phải sử dụng những **phần mềm ban đầu vốn thiết kế cho in ấn**, như Adobe PageMaker hay QuarkXPress.
* Việc thiết kế web lúc đó khá thủ công và nhiều giới hạn.

**✅ Ngày nay thì đã khác!**

* Hiện nay đã có rất nhiều phần mềm **được phát triển chuyên biệt** cho mục đích thiết kế web:
  + Giao diện thân thiện hơn.
  + Có sẵn các tính năng hỗ trợ viết mã HTML, CSS, JavaScript.
  + Tích hợp khả năng xem trước (preview) hoặc kiểm tra responsive (phản hồi trên thiết bị khác nhau).

**🤔 Vậy có phần mềm nào tốt nhất không?**

Không có phần mềm "tốt nhất" tuyệt đối vì:

1. **Có quá nhiều phần mềm khác nhau.** Mỗi người có thể chọn phần mềm phù hợp với phong cách làm việc của họ.
2. **Mỗi người có sở thích riêng.** Có người thích viết mã bằng tay, người khác lại thích dùng công cụ kéo – thả.
3. **Công cụ mới xuất hiện liên tục.** Có thể lúc người viết cuốn sách này, một số phần mềm chưa ra đời, nhưng bây giờ chúng đã phổ biến và mạnh mẽ hơn.

**💻 LƯU Ý DÀNH CHO NGƯỜI DÙNG MAC: *Chạy Windows Trên Mac***

Nếu bạn đang dùng máy tính **Mac (Macintosh)**, và có chip **Intel** (Mac đời cũ, không áp dụng cho chip Apple Silicon như M1, M2), bạn **không cần mua máy Windows riêng** để kiểm tra trang web.

**✅ Bạn có thể chạy Windows trên Mac bằng các cách sau:**

**1. Boot Camp (miễn phí, có sẵn trong macOS Leopard trở lên)**

* Cài Windows song song với macOS.
* **Khi khởi động máy**, bạn chọn vào macOS hoặc Windows.
* Phù hợp nếu bạn muốn **chạy Windows "thật" với hiệu năng tối đa**.

**2. Dùng máy ảo (Virtual Machine – VM):**

Bạn có thể **chạy Windows như một ứng dụng trên macOS**, không cần khởi động lại. Có 3 lựa chọn phổ biến:

|  |  |  |
| --- | --- | --- |
| Tên phần mềm | Loại | Ưu điểm |
| VMware Fusion | Thương mại (có bản dùng thử) | Hiệu năng tốt, hỗ trợ nhiều hệ điều hành |
| Parallels Desktop for Mac | Thương mại (có bản dùng thử) | Giao diện đẹp, dễ dùng, tối ưu cho Mac |
| Oracle VirtualBox | MIỄN PHÍ | Nguồn mở, hỗ trợ nhiều hệ điều hành (bao gồm cả Linux) |

⚠️ Lưu ý: Dù dùng cách nào thì **bạn vẫn cần phải mua bản quyền hệ điều hành Windows** để cài vào.

**📝 CHÚ Ý QUAN TRỌNG:**

🟩 Để làm theo các bài tập trong sách (và bắt đầu học thiết kế web), bạn **không cần mua phần mềm nào cả.**

Chỉ cần:

* Một **trình soạn thảo văn bản** (text editor) cơ bản có sẵn trên máy như:
  + Notepad (Windows)
  + TextEdit (Mac – chuyển sang chế độ Plain Text)
  + Hoặc các editor miễn phí mạnh mẽ như **Visual Studio Code** (được khuyên dùng 👍)
* Một phần mềm chỉnh sửa hình ảnh **miễn phí**, như:
  + GIMP ([www.gimp.org](http://www.gimp.org))
  + Photopea (dùng trên trình duyệt: [www.photopea.com](http://www.photopea.com))
  + Canva (miễn phí, đơn giản, đẹp)

**🧠 Tớ gợi ý thêm vài phần mềm hiện đại cho cậu nè:**

|  |  |
| --- | --- |
| Mục đích | Phần mềm khuyên dùng (miễn phí hoặc có bản miễn phí) |
| Soạn mã HTML/CSS/JS | Visual Studio Code, Sublime Text, Atom |
| Thiết kế giao diện (UI/UX) | Figma, Adobe XD, Canva |
| Xem thử web trên di động | Chrome DevTools (F12 → Toggle Device Toolbar) |
| Tạo ảnh / chỉnh sửa ảnh | GIMP, Photopea, Canva |
| Làm việc nhóm | GitHub, Git, Google Drive |

**✅ Tóm tắt lại:**

* Có rất nhiều phần mềm để thiết kế web → không có một phần mềm “tốt nhất”.
* Nếu dùng Mac thì **không cần mua máy Windows**, có thể chạy Windows bằng Boot Camp hoặc phần mềm máy ảo.
* Để học ban đầu, bạn chỉ cần:
  + Trình soạn thảo văn bản.
  + Phần mềm chỉnh sửa ảnh miễn phí.
  + **Không cần tốn tiền mua phần mềm đắt đỏ**.

Coding tools:

Although you can get by with the simple text editors that come with your computer, a dedicated code editor makes the task of writing HTML, CSS, and JavaScript much easier. Code editors understand the syntax of the code you write, so they can do things for you like color coding, error detection, and automatically finishing simple tasks like closing HTML tags. Some provide page previews so you can view the results of your code as you work.

FIGURE 1-6 shows how an HTML document looks in the Sublime Text editor. Here are just a few of the better-known code editors for web production that are worth exploring:

* Sublime Text (sublimetext.com)
* Atom (free from GitHub; atom.io)
* Brackets (free from Adobe; brackets.io)
* CodeKit (codekitapp.com; Mac only)
* Adobe Dreamweaver (www.adobe.com/products/dreamweaver.html)
* Coda (panic.com/coda/)
* Microsoft Visual Studio (visualstudio.com)

User interface and layout tools:

There is a new breed of interface design tools made specifically for websites and other applications. Because they have been designed from scratch with interface design in mind, they seem to anticipate a web designer's every need. Interface design tools make it easy to design multiple layouts (such as layouts at various screen sizes) as well as export images and code for use in production. Some allow basic interactivity such as clicks and swipes, so your mockups can be shared online and used for basic interface testing.

Sketch (sketchapp.com, Mac only), shown in FIGURE 1-7, is extremely popular at the time of this writing. Other options include the following:

* Affinity Designer (affinity.serif.com/en-us/designer/)
* Adobe XD (www.adobe.com/products/xd.html)
* Figma (figma.com)
* UXPin (uxpin.com)

# **PART II. HTML FOR STRUCTURE**

## **CHAPTER 04: CREATING A SIMPLE PAGE (HTML OVERVIEW)**

**🧭 Tổng quan trước đã nè:**

**Phần I** trước đó giống như là phần mở đầu giới thiệu về thế giới thiết kế web – những điều to to và quan trọng như:

* Web là gì,
* Trình duyệt hoạt động ra sao,
* HTML/CSS là gì,
* Các công cụ cần có khi thiết kế web…

Giờ thì đã đến lúc **xắn tay áo** và **bắt tay vào làm thật một trang web đầu tiên** rồi nhaaaa!

**🧱 Mục tiêu của chương này là gì?**

Cậu sẽ cùng tớ:

1. 👀 **Làm quen với HTML**: Cách HTML hoạt động, cách mà **thẻ (tag)** và **thuộc tính (attribute)** dùng để "đánh dấu" nội dung của một trang web.
2. 🧠 **Hiểu cách trình duyệt “dịch”** các tệp HTML để hiển thị ra trang web trên màn hình.
3. 🧩 **Nắm được cấu trúc cơ bản** của một tài liệu HTML (mỗi trang web đều có cấu trúc như thế này đó!).
4. 🎨 **Nhìn thử qua CSS (style sheet)** – cách trang trí cho web (nhưng chưa học kỹ đâu, chỉ nhìn sơ cho biết thôi nhé 👀).

**💡 Ghi nhớ cực quan trọng cho cậu nè:**

❗️**Đừng lo lắng** nếu cậu chưa nhớ được hết các thẻ HTML hay các quy tắc CSS ở chương này nhé!  
Mục tiêu hiện tại là:

* Hiểu **quy trình tạo ra một trang web**,
* Làm quen với **thuật ngữ** mới,
* Và thấy được **tổng thể hình dáng** của một tài liệu HTML.

**🧱 Trong chương này, cậu sẽ học được những gì?**

|  |  |
| --- | --- |
| Phần học | Tớ sẽ giúp cậu hiểu gì |
| ✅ Giới thiệu về Elements (phần tử) và Attributes (thuộc tính) | Thẻ HTML là gì, hoạt động ra sao, cách gắn thuộc tính |
| ✅ Cách viết một trang web đơn giản | Cậu sẽ bắt đầu “code” một trang thật luôn đó 😎 |
| ✅ Các thẻ tạo cấu trúc cho trang web | Như <html>, <head>, <body>,... nó quan trọng cỡ nào? |
| ✅ Sửa lỗi khi trang web bị hư | Lỗi thường gặp và cách khắc phục khi trang không hiển thị đúng |

**🧑‍🏫 Tớ tóm tắt lại ý chính nè:**

🏗️ **Một trang web dù phức tạp đến đâu, cũng đều bắt đầu từ những nguyên tắc cơ bản giống nhau.**  
Và những nguyên tắc đó là những gì cậu sẽ học trong chương này:  
HTML là cách “đánh dấu” nội dung để máy tính hiểu, trình duyệt sẽ "dịch" ra cho người xem thấy.  
Bước đầu không cần nhớ hết thẻ hay CSS đâu, chỉ cần hiểu cách **xây dựng khung trang web** là đủ!

### I. A WEB PAGE, STEP-BY-STEP

**🧱 TỰ TAY TẠO TRANG WEB: TỪNG BƯỚC RÕ RÀNG**

Trong Chương 2 (cậu nhớ không? Là chương nói về cách web hoạt động), cậu đã **nhìn thấy một tài liệu HTML mẫu**. Nhưng bây giờ là lúc cậu **tự tay tạo ra một trang web thực sự đầu tiên** rồi đấy! 🎉

**🌟 Tạo trang web gồm 5 bước chính:**

**👉 Bước 1: Bắt đầu với nội dung**

📝 Cậu cứ tưởng tượng mình đang viết một đoạn văn bằng Word vậy. Ở bước này, **chưa dùng HTML** đâu.  
Chỉ là viết **nội dung thô** ra như:

**Chào mừng đến với trang web đầu tiên của tớ!**

**Tớ đang học cách làm web.**

🔍 Khi mở nội dung đó bằng trình duyệt, nó sẽ hiển thị y như cậu viết – **không có định dạng gì hết** (như in đậm, tiêu đề, xuống dòng, v.v).

—

**👉 Bước 2: Tạo khung sườn HTML**

🧱 Đây là bước “xây móng nhà”. Mình **bắt đầu dùng các thẻ HTML cơ bản** để trình duyệt hiểu nội dung mình muốn trình bày là gì.  
Ví dụ:

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Trang đầu tiên</title>**

**</head>**

**<body>**

**<p>Chào mừng đến với trang web đầu tiên của tớ!</p>**

**<p>Tớ đang học cách làm web.</p>**

**</body>**

**</html>**

💡 Các thẻ như <html>, <head>, <body> là **xương sống** của một tài liệu HTML.

—

**👉 Bước 3: Dùng đúng thẻ cho từng loại nội dung**

📚 Ở bước này, mình không chỉ viết nội dung nữa mà phải **dùng đúng “thẻ” cho từng loại nội dung**:

* Đoạn văn → dùng <p>
* Tiêu đề → dùng <h1>, <h2>,…
* Danh sách → dùng <ul> (unordered list), <ol> (ordered list), <li> (list item)
* Đường link → dùng <a>
* V.v.

🎯 **HTML không phải là ngôn ngữ lập trình** đâu nha. Nó chỉ là cách để nói với trình duyệt:

"Ê, đoạn này là tiêu đề đó",  
"Chỗ này là đoạn văn nhen",  
"Cái này là ảnh nha",...

—

**👉 Bước 4: Thêm hình ảnh vào trang**

🖼️ Ở bước này, mình học cách **thêm ảnh bằng HTML**.  
Sử dụng thẻ:

**<img src="hinhanh.jpg" alt="Mô tả ảnh">**

🔍 Ở đây có gì đặc biệt?

* <img> là **thẻ rỗng** (không có thẻ đóng </img>)
* src là thuộc tính (attribute) để **chỉ đường dẫn ảnh**
* alt là phần mô tả ảnh phòng khi ảnh bị lỗi không hiện.

—

**👉 Bước 5: Làm đẹp nội dung với CSS (style sheet)**

🎨 HTML giúp cậu trình bày nội dung, còn **CSS giúp “làm đẹp” nội dung đó**:

* Màu sắc, kiểu chữ, canh giữa, kích thước, v.v.

Ví dụ:

**<style>**

**p {**

**color: blue;**

**font-size: 20px;**

**}**

**</style>**

🧠 Đây là lúc cậu lần đầu tiên **nếm thử CSS** – đừng lo, chưa cần học sâu, chỉ cần biết nó làm gì là được.

**💪 Vì sao nên học HTML theo cách “cổ điển”?**

Tác giả bảo rằng:

“Tớ vẫn ủng hộ cách học HTML kiểu **cổ điển** – viết từng dòng mã bằng tay.”

Lý do là:

* ✨ **Viết tay giúp hiểu rõ bản chất** từng dòng code HTML.
* 🧰 Sau này nếu dùng phần mềm kéo-thả, cậu vẫn có thể:
  + Sửa lỗi nếu trang bị hỏng
  + Tinh chỉnh phần code
  + Không bị "ngợp" vì không hiểu nó đang tạo ra gì.

📌 **Ghi nhớ đặc biệt:**

Dù sau này dùng công cụ hiện đại, nhưng nếu hiểu rõ HTML thì cậu vẫn **kiểm soát được mọi thứ** và có thể “hô biến” trang web theo ý muốn của mình!

🎓 Thậm chí, **dân lập trình chuyên nghiệp** hiện nay vẫn **thích viết HTML bằng tay** vì:

* Có **toàn quyền điều khiển**
* Code gọn gàng, rõ ràng
* Tối ưu cho tốc độ tải và SEO nữa đó!

**🧠 Tóm tắt gọn cho cậu nè:**

|  |  |
| --- | --- |
| Bước | Ý chính cần nhớ |
| 1️ | Viết nội dung thô |
| 2️ | Tạo khung HTML |
| 3️ | Dùng đúng thẻ cho nội dung |
| 4️ | Thêm hình ảnh |
| 5️ | Làm đẹp bằng CSS sơ bộ |

### II. LAUNCH A TEXT EDITOR

**📝 CÁCH TẠO MỘT TÀI LIỆU HTML MỚI TRONG NOTEPAD (trên Windows 10)**

Đây là hướng dẫn **bước từng bước** để cậu **tự tạo một file HTML** đầu tiên bằng **Notepad – trình soạn thảo cơ bản có sẵn trong Windows**. Rất nhẹ, dễ dùng và không cần cài thêm gì hết!

**🟢 Bước 1: Mở Notepad (Trình soạn thảo văn bản)**

➡️ Làm như sau:

* Nhấn nút **Windows (hoặc Start Menu)** trên bàn phím.
* Gõ từ khóa **Notepad** (trong ô tìm kiếm).
* Khi thấy biểu tượng Notepad hiện ra, **nhấn chuột vào nó để mở**.

📄 Khi Notepad hiện lên, cậu đã sẵn sàng để bắt đầu viết HTML rồi đó!

**🟢 Bước 2: Hiện phần mở rộng tệp tin (File extension)**

🔍 **Tại sao phải làm bước này?**

* Khi lưu file HTML, cậu cần đặt đuôi .html.
* Nếu phần mở rộng đang bị **ẩn**, có thể Windows sẽ hiểu file là .txt, khiến trình duyệt không đọc được nó như một trang web.
* Bật hiện phần mở rộng giúp cậu **thấy rõ mỗi file đang có đuôi gì**, rất quan trọng khi học web đó!

🪄 **Cách làm:**

1. Mở **File Explorer** (bằng cách nhấn Windows + E hoặc mở bất kỳ thư mục nào cũng được).
2. Nhìn lên trên cùng, bấm vào **tab “View”**.
3. Bên phải, bấm vào nút **“Options”** (hoặc “Folder Options”).

**🟢 Bước 3: Bỏ chọn dòng “Hide extensions for known file types”**

🛠️ Trong cửa sổ Folder Options:

* Bấm sang **tab “View”** một lần nữa.
* Cuộn xuống tìm mục **“Hide extensions for known file types”** (Tạm dịch: “Ẩn phần mở rộng của những loại tệp đã biết”).
* **Bỏ dấu tick** ở mục đó (tức là **không ẩn nữa** nha).

**🟢 Bước 4: Lưu lại cài đặt**

✔️ Sau khi bỏ tick:

* Bấm **OK** để lưu lại thiết lập.
* Bây giờ, mỗi khi cậu xem file trong máy tính, cậu sẽ thấy rõ tên file kiểu như:
  + index.html (file HTML)
  + style.css (file CSS)
  + script.js (file JavaScript)

💡 **Lưu ý:**  
Nếu không bật hiển thị đuôi file, đôi khi khi cậu gõ index.html, Windows sẽ lưu thành index.html.txt mà cậu không biết → trình duyệt sẽ không hiểu đó là trang web.

💡 **Mẹo nhỏ:**  
Khi lưu file HTML đầu tiên, nhớ:

* Chọn **Save As**
* Đặt tên là: index.html
* Phần **Save as type**: chọn **All Files** (Tất cả các loại)
* Encoding: **UTF-8** là tốt nhất.

### III. STEP 1: START WITH CONTENT

#### III.1. NAMING CONVENTIONS

**📛 QUY TẮC ĐẶT TÊN FILE TRONG LẬP TRÌNH WEB – DỄ HIỂU VÀ CỰC KỲ QUAN TRỌNG**

Khi cậu bắt đầu xây dựng một website, việc **đặt tên file đúng cách** là điều cực kỳ quan trọng vì:

* Trình duyệt cần biết file nào là HTML, CSS, ảnh, v.v.
* Dễ quản lý và liên kết giữa các file.
* Tránh lỗi khi upload lên web (đặc biệt là trên máy chủ – server).

**✅ 1. Sử dụng đúng phần mở rộng file (File extension)**

📌 **Phần mở rộng (đuôi file)** giúp hệ thống và trình duyệt hiểu được **loại nội dung** trong file.

|  |  |
| --- | --- |
| File loại gì? | Đuôi bắt buộc phải là gì? |
| Trang web | .html hoặc .htm |
| Hình ảnh | .jpg, .jpeg, .png, .gif, .svg |

👉 Ví dụ:

* index.html ✅
* logo.png ✅
* style.css ✅
* **Không được** đặt kiểu như: index.docx ❌ hay image.jpeg.txt ❌

**✅ 2. Không được dùng khoảng trắng (space) trong tên file**

⛔ Không nên đặt tên kiểu như:  
my resume.html ❌  
Trình duyệt hoặc link URL sẽ đọc thành: my%20resume.html, dễ gây lỗi.

✅ Thay vào đó, hãy dùng:

* **Gạch dưới**: my\_resume.html
* **Gạch ngang**: my-resume.html ← phổ biến nhất

👉 Gợi ý: Tớ khuyên dùng **gạch ngang (-)** vì nó:

* Chuẩn SEO hơn
* Được Google ưu tiên hơn khi đọc link

**✅ 3. Tránh các ký tự đặc biệt nguy hiểm**

⛔ Không được dùng các ký tự như:

**?, %, #, /, \, :, ;, &, \*, @, ~, `, =, ", ', etc.**

✅ Chỉ nên dùng:

**a-z, A-Z, 0-9, dấu gạch ngang (-), gạch dưới (\_), và dấu chấm (.)**

💡 Vì sao?

* Ký tự đặc biệt có thể bị hiểu nhầm là **câu lệnh điều khiển URL** (như dấu ? là cho tham số).
* Khi upload file lên **web server**, mấy ký tự đó có thể khiến link bị lỗi hoặc bị chặn.

**✅ 4. Phân biệt chữ hoa và chữ thường (case-sensitive)**

⚠️ Một số **máy chủ (server)** phân biệt chữ hoa – chữ thường trong tên file.

Ví dụ:

* About.html khác hoàn toàn với about.html

→ Vì thế, tốt nhất là:  
✅ **Chỉ dùng chữ thường (lowercase) trong tất cả tên file**

🧠 Tớ ví dụ:

* home.html, about-us.html, contact.css là chuẩn chỉnh!

**✅ 5. Đặt tên ngắn gọn và dễ nhớ**

⛔ Không nên đặt tên file quá dài như:  
this\_is\_a\_super\_long\_file\_name\_for\_my\_first\_website\_version\_final\_final\_v2.html 😵

✅ Hãy rút gọn như:

* about.html
* my-bio.html
* contact-us.html

💡 Cậu muốn rõ nội dung thì có thể dùng tên **nhiều từ nhưng ngắn và ngăn cách bằng dấu gạch ngang**:  
long-document-title.html là được rồi!

**✅ 6. Tự đặt quy tắc riêng (Self-imposed conventions)**

📚 Nếu cậu làm một **dự án lớn** hoặc sau này làm web chuyên nghiệp, hãy:

* Đặt ra một **quy chuẩn nhất quán cho toàn bộ tên file**
* Ví dụ:
  + Tất cả đều dùng chữ thường
  + Dùng - để ngăn cách từ
  + Tên file luôn liên quan đến nội dung

✨ Điều này giúp:

* Quản lý hàng trăm file dễ dàng hơn
* Khi liên kết (link) giữa các file không bị nhầm

**💡 MẸO CỦA TỚ**

|  |  |
| --- | --- |
| Mẹo | Mô tả |
| ✅ Dùng chữ thường | Để tránh lỗi phân biệt chữ hoa – chữ thường |
| ✅ Dùng dấu - | Chuẩn SEO, dễ đọc |
| ✅ Tránh dấu cách | Tránh URL lỗi hoặc khó đọc |
| ✅ Không ký tự đặc biệt | Tránh bị lỗi khi chạy web hoặc upload |
| ✅ Đặt tên ngắn, rõ nội dung | Dễ nhớ, dễ quản lý |
| ✅ Có quy chuẩn thống nhất | Càng về sau, càng dễ làm việc nhóm |

#### III.2. WHAT BROWSERS IGNORE

**🧹 Những Thứ Trình Duyệt Sẽ *BỎ QUA* Khi Hiển Thị Trang Web**

Mặc dù file HTML của mình có thể được viết rất dài dòng, có dấu cách, xuống dòng hay tab… nhưng **khi hiển thị ra trình duyệt (Chrome, Firefox, Edge, Safari, v.v.) thì nhiều thứ sẽ bị “ngó lơ”** 🫣.

**1️. Nhiều khoảng trắng liên tiếp (Multiple-character spaces)**

📌 Khi trong HTML có nhiều dấu cách (space) liền nhau, **trình duyệt chỉ hiển thị 1 dấu cách duy nhất**.

🔍 Ví dụ trong file HTML:

**<p>long, long ago</p>**

🖥️ Trình duyệt sẽ hiển thị ra:

**long, long ago**

✅ Dù có 5 hay 10 dấu cách, kết quả vẫn **chỉ là 1 khoảng trắng** thôi nha!

📌 Nếu cậu **thật sự muốn hiện nhiều khoảng trắng**, hãy dùng:

* **&nbsp; (non-breaking space) – mỗi cái là 1 khoảng trắng không bị gộp**

💡 Ví dụ:

**<p>long,&nbsp;&nbsp;&nbsp;long&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;ago</p>**

→ Sẽ hiện đúng nhiều khoảng trắng như mong muốn!

**2️. Xuống dòng (Line breaks / Carriage returns)**

📌 Khi trong file HTML có xuống dòng (enter), **trình duyệt sẽ coi như dấu cách trắng thôi**.

🔍 Ví dụ:

**<p>This is a line**

**that is broken into**

**multiple lines.</p>**

🖥️ Kết quả hiển thị trên trình duyệt:

**This is a line that is broken into multiple lines.**

→ Các dòng **sẽ nối lại thành 1 dòng duy nhất**!

✅ Nếu cậu muốn **xuống dòng thực sự** khi hiển thị, dùng thẻ:

* **<br> ← dòng mới**
* **<p> ← đoạn văn mới**
* **<div> hoặc thẻ khối khác**

**3️. Ký tự tab (Tab characters)**

📌 Khi trong mã HTML có **phím Tab**, trình duyệt sẽ hiểu là **một khoảng trắng bình thường**.

🔍 Ví dụ:

**<p>This is text</p> <!-- tab giữa các từ -->**

🖥️ Hiển thị ra sẽ chỉ là:

**This is text**

→ Không có khoảng cách lớn, không thụt lề gì hết ❌

✅ Tab chỉ có tác dụng làm **mã HTML dễ đọc hơn cho lập trình viên**, không ảnh hưởng giao diện website.

**4️. Thẻ sai cú pháp hoặc không được hỗ trợ (Unrecognized markup)**

📌 Nếu cậu viết **thẻ HTML sai chính tả** hoặc dùng **thẻ không tồn tại**, **trình duyệt sẽ bỏ qua** hoặc hiển thị nội dung bên trong như văn bản thường.

🔍 Ví dụ:

**<bogustag>Oops!</bogustag>**

🖥️ Trình duyệt sẽ **bỏ qua cái thẻ <bogustag>**, và chỉ hiện:

**Oops!**

💡 Lưu ý:

* Có thể gây lỗi giao diện, hoặc gây **kết quả hiển thị không như ý**.
* Mỗi trình duyệt có thể **xử lý thẻ sai khác nhau** → nên **luôn kiểm tra kỹ thẻ HTML** nha cậu!

✅ Hãy dùng **công cụ kiểm tra cú pháp (HTML validator)** như:

* <https://validator.w3.org/> ✅

**5️. Nội dung trong chú thích (HTML comments)**

📌 Mọi nội dung nằm trong **thẻ chú thích** <!-- --> đều **không hiển thị trên trang web**.

🔍 Ví dụ:

**<!-- Đây là phần chú thích -->**

**<p>Xin chào!</p>**

🖥️ Trình duyệt **chỉ hiển thị**:

**Xin chào!**

→ Dòng "Đây là phần chú thích" bị **ẩn hoàn toàn**.

💡 Dùng **chú thích** để:

* Ghi chú giải thích mã
* Ghi nhớ vị trí sửa code
* Tạm thời ẩn một đoạn HTML khi đang thử nghiệm

**✅ Tóm tắt dễ nhớ – Cậu nhớ bảng này là xong luôn!**

|  |  |
| --- | --- |
| Trường hợp | Trình duyệt xử lý ra sao? |
| Nhiều dấu cách liền nhau | Hiển thị thành **1 khoảng trắng duy nhất** |
| Xuống dòng trong code | Hiển thị như **1 khoảng trắng** |
| Dùng phím Tab trong HTML | Hiển thị như **1 khoảng trắng** |
| Thẻ HTML sai/lạ | **Bỏ qua**, hoặc hiển thị nội dung bên trong như text thường |
| Chú thích HTML (<!-- -->) | **Không hiển thị** ra ngoài trình duyệt |

**🌟 Mở rộng thêm – Nếu muốn “giữ nguyên khoảng trắng và xuống dòng” thì sao?**

👉 Cậu có thể dùng **CSS**:

**white-space: pre;**

📌 Áp dụng cho thẻ HTML nào mà cậu muốn hiển thị đúng mọi dấu cách, tab, xuống dòng, v.v.

🔍 Ví dụ:

**<pre>**

**This is spaced.**

**Line 1**

**Line 2**

**</pre>**

Hoặc:

**<p style="white-space: pre;"> Hello world!**

**Line 2 here.</p**>

→ Trình duyệt **sẽ giữ nguyên định dạng** y như trong code!