**ÔN TẬP LÝ THUYẾT**

***1. Nguyên tắc hoạt động của website:***

1.1 Khái niệm và thuật ngữ:

a. The Internet:

• Internet là mạng lưới kết nối các máy tính với nhau.

• Không thuộc sở hữu của bất kì ai.

• Hoạt động dựa trên hệ thống các tiêu chuẩn và quy định.

• Mục đích: chia sẻ thông tin

• Email

• File transfer (FTP)

• ...

• Protocol: Tập hợp các quy tắc được thiết lập nhằm xác

định cách để định dạng, truyền và nhận dữ liệu qua hệ

thống mạng.

b. World Wide Web:

• World Wide Web (WWW) hay thường gọi là WEB

• Chỉ là một trong những cách thức thông tin được chia sẻ

thông qua Internet.

• Cho phép các tài liệu được liên kết với nhau thông qua các

liên kết siêu văn bản (hypertext link) từ đó hình thành nên

“mạng lưới” (web) thông tin khổng lồ.

• Web sử dụng protocol được gọi là HTTP (HyperText

Transfer Protocol).

• ➔ WEB là 1 phần của Internet.

c. The Server:

• “Serve up” tài liệu khi có yêu cầu → Server

• Hiểu chính xác, “server” là phần mềm giúp cho máy tính

giao tiếp với nhau chứ không chỉ là bản thân máy tính.

• Nhiệm vụ của server software:

• Waiting for request

• Retrieve the information

• Send the information back

• → Thực tế hay dùng server để chỉ máy chủ cung cấp thông

tin

• Trong hệ thống WEB, máy tính phải được chạy phần mềm

web server riêng có thể xử lý các yêu cầu với giao thức

HyperText Transfer → HTTP servers

• 2 HTTP server phổ biến nhất:

• Apache (open source software)

• Microsoft Internet Information Services (IIS)

d. IP Address:

• IP: Internet Protocol

• Mọi máy tính và thiết bị kết nối Internet được gắn 1 địa chỉ

IP duy nhất với định dạng 1 số 32 bit, chia làm 4 số 8 bit

xxx.xxx.xxx.xxx (IPv4)

• Khó nhớ, Domain Name System (DNS) được phát triển

cho phép tham chiếu IP Address bởi domain name.

• VD: “google.com”: 172.217.24.68

• Tip: dùng lệnh ping trong cmd để xem địa chỉ IP của domain

name.

• → DNS server

• Có thể dùng nhiều domain name cho cùng 1 địa chỉ IP.

e. Web Hosting:

• Mỗi tên miền được gắn với một IP nhất định, nơi lưu trữ

các nội dung và tài nguyên của website, đó chính là web

hosting.

• Để đảm bảo cho truy xuất của người dùng luôn tìm thấy

đích, tên miền cần được gắn với một IP cố định.

• Tuy nhiên không phải cá nhân hay đơn vị nào cũng có thể

có riêng một IP cố định.

• → Dịch vụ lưu trữ website – Web Hosting ra đời.

f. The Client:

• Phần mềm gửi yêu cầu cho Server được gọi chung là

Client.

• Các phần mềm Client phổ biến để duyệt WEB thường gọi

là Browser (trình duyệt web)

• Desktop browsers

• Mobile browsers

• Ngoài ra còn 1 số các công nghệ khác như Screen Reader

• Internet Explorer, Chrome, Firefox, Safari, Opera, ...

• Cùng 1 trang web có thể hiện thị khác nhau trên các trình

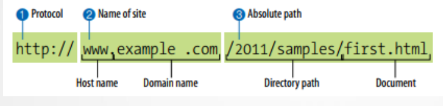
duyệt khác nhau.

g. Web Page Address:

• Tất cả trang thông tin và các tài nguyên trên Web đều có

địa chỉ riêng của nó gọi là URL (Uniform Resource

Locator)



• “http://”: xác định protocol được sử dụng

• “www.example.com”: xác định website bằng domain name

• “/2012/samples/first.html”: đường dẫn đầy đủ đến tài liệu HTML

chỉ định (first.html)

h. Web Page Address:

• Một số trang web không có tên file

• https://www.google.com/maps

• https://www.apple.com/

• https://www.samsung.com/vn/

• ...

• Khi server nhận yêu cầu mở directory name, nó tìm trong

directory đó file mặc định và mở file đó.

• index.html, default.html

• index.php, index.asp

• ...

i. HTML document:

• Các nội dung hình ảnh, âm thanh,... của website được sinh

ra bằng tài liệu đơn giản, text-only

• HyperText Markup Language (HTML): defines dozens of

text elements that make up documents

• Headings, paragraphs, emphasized text, links,

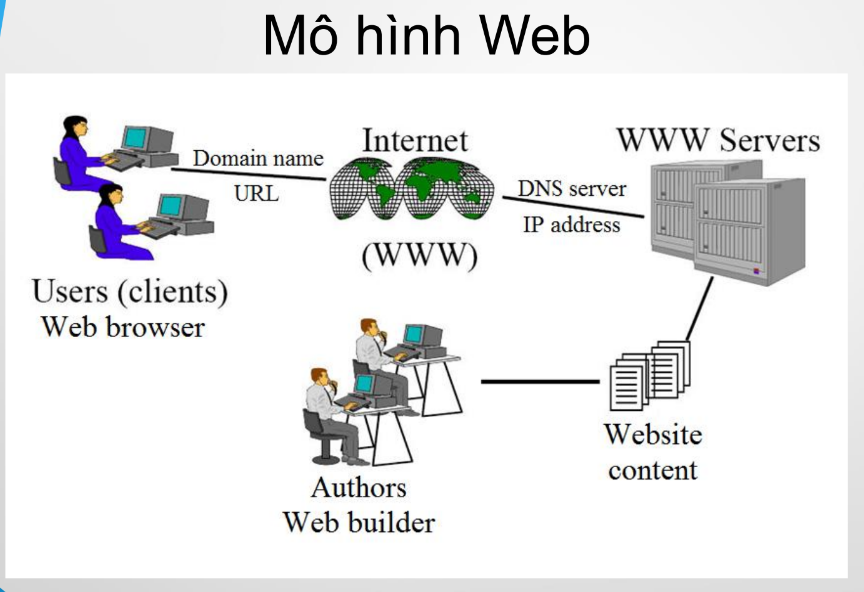
• Elements that add information about the document (such as its

title),

• Media such as images and videos,

• ...

1.2. Nguyên tắc hoạt động website:

******

* Từ máy client gõ địa chỉ URL của trang web vào thanh địa chỉ của web browser, khi đó web browser sẽ dựa vào tên domain để tìm kiếm IP của web server (bằng cách dò trên DNS Server)
* Khi có IP, web browser sẽ gửi yêu cầu đến web server để lấy nội dung trang web
* Web server sẽ xử lý yêu cầu của client: lấy nội dung của website mà client yêu cầu để gửi client.
* Khi nhận được nội dung trang web thì web browser sẽ hiển thị lên cho người dùng.
* Trình duyệt web sẽ tách địa chỉ website ra làm ba phần:
* Giao thức (http)
* Máy chủ tên miền (www.webname.com)
* Tên tệp (file.htm)
* Trình duyệt web liên hệ với máy chủ tên miền (DNS server) để phân giải tên miền “www.webname.com” thành địa chỉ IP của web server
* Trình duyệt sẽ gửi tiếp 1 kết nối tới máy chủ địa chỉ IP tương ứng cổng 80.
* Dựa trên giao thức HTTP, trình duyệt gửi yêu cầu đến máy chủ, yêu cầu tệp: http://webname.com/file.htm
* Máy chủ sẽ gửi đoạn text dạng html từ Authors Web builder đến trình duyệt của web client. Trình duyệt đọc các thẻ html, định dạng trang web và kết xuất ra màn hình.

1.3 Web hosting service:

* Khái niệm về dịch vụ lưu trữ web: Lưu trữ web hay web hosting là dịch vụ cung cấp dung lượng lưu trữ cho trang web hoặc ứng dụng trên máy chủ qua đường truyền internet. Ngay khi website của bạn sẵn sàng trên internet, nó có thể được truy cập bởi các máy tính khác có kết nối mạng.
* Mua tên miền:

• Tên miền = Tên website

• Tên ngắn

• Gợi nhớ

• Dễ đọc, không gây nhầm lẫn

• Thể hiện được tên công ty hoặc nhãn hiệu chính của công ty

• Vd: IBM.com, Lenovo.com

* Một số NCC tên miền phổ biến ở VN:

• https://www.matbao.vn/

• https://inet.vn/

• https://tenten.vn/

• <https://nhanhoa.com/>

* Thuê hosting:

• Hệ điều hành của máy chủ

• Dung lượng

• Băng thông

• Ngôn ngữ hỗ trợ

• Hệ quản trị CSDL hỗ trợ

• Email POP3

• Secure Sockets Layer (SSL)

• ...

* Một số free hosting:

• https://www.infinityfree.net/

• https://byet.host/

• https://www.wix.com/

• https://www.awardspace.com/

* Dung lượng lưu trữ (disk space): là khoảng không gian mà người dùng được lưu trữ dữ liệu trên ổ cứng của server và sức chứa của gói Hosting. Do đó, dữ liệu càng nhiều tương ứng với dung lượng Hosting càng lớn. Dung lượng lưu trữ nhiều hay ít là tùy thuộc vào gói Hosting mà bạn lựa chọn.
* Băng thông (bandwidth): là thuật ngữ dùng để chỉ lượng dữ liệu được truyền trong một giây.Khái niệm này cũng đại diện cho tốc độ truyền dữ liệu của một đường truyền.
* Tên miền con (sub-domain)là phần mở rộng và là phần bổ sung xuất hiện trước của tên miền chính. Subdomain là một phần tách ra từ [domain](https://vietnix.vn/domain-la-gi/) và hoạt động như một [website](https://vietnix.vn/website-la-gi/) bình thường. Đặc biệt, subdomain giúp bạn tạo ra nhiều website trên các lĩnh vực khác nhau thuộc tên miền chính.
* Email (có domain riêng): là dịch vụ lưu trữ email gắn với tên miền bạn có, chẳng hạn khi iNET có tên miền "iNET.vn", email theo tên miền của iNET sẽ là "...@iNET.vn". Người dùng khi check email luôn cảm nhận email theo tên miền nghiêm túc hơn các dạng email miễn phí và đó cũng là lý do tại sao phần lớn doanh nhân, doanh nghiệp sử dụng email theo tên miền nhằm nâng cao uy tín, khẳng định thương hiệu
* Ngôn ngữ lập trình là ngôn ngữ để viết các chương trình của máy tính. Theo ước tính hiện nay có khoảng hơn 200 loại ngôn ngữ được sử dụng cho công việc lập trình. Chúng ta có thể dễ dàng phân chia các ngôn ngữ bằng nhiều cách khác nhau. Nhưng nhìn chung sẽ có 2 loại chính là:
  + 1 Ngôn ngữ giao diện người dùng (front-end).Ngôn ngữ này được dùng để thiết kế và xây dựng giao diện cho trang web hoặc ứng dụng web. Nó giúp người dùng có thể xem và tương tác trực tiếp tại đó. Cụ thể, HTML, CSS, JavaScript,… là những ngôn ngữ giao diện người dùng phổ biến hiện nay.
  + 2 Ngôn ngữ máy chủ (ngôn ngữ lập trình server).Là những ngôn ngữ lập trình được dùng để viết các chương trình, tiện ích chạy trên các server như Linux, Windows,… Python, Java, C/C++, PHP, Ruby… là những ngôn ngữ máy chủ phổ biến hiện nay.
  + Nhìn chung, ngôn ngữ lập trình chiếm số lượng nhiều và có đặc điểm khá tương đồng nhau, tuy nhiên cú pháp sử dụng của chúng là riêng biệt. Vậy nên, các lập trình viên cần phải học các quy tắc, cú pháp và cấu trúc ngôn ngữ, sau đó tiến hành viết mã nguồn trong một trình soạn thảo hoặc IDE rồi biên dịch code thành ngôn ngữ máy thì máy tính mới có thể hiểu được.

***2. Giao diện website:***

2.1 Các khái niệm về website:

* Phân loại website
  + Phân loại website theo cấu trúc website: Website tĩnh (static website), Website động (dynamic website).
  + Phân loại website theo quyền sở hữu: Website doanh nghiệp, Website cá nhân, Website chính phủ.
  + Phân loại website theo chức năng: Website bán hàng, Website tin tức, Mạng xã hội, Diễn đàn / Forum, Website giải trí, Website rao vặt
  + Phân loại website theo các lĩnh vực hoạt động: Các website được phân theo lĩnh vực thường là: web du lịch, xây dựng, giáo dục, thời trang, mỹ phẩm, tin tức,…
* Những thành phần của website:
  + Trong đó có 4 thành phần chủ đạo nhất chính là: Domain, hosting, mã nguồn và cuối cùng là nội dung website, chi tiết về 4 thành phần này cụ thể như sau:
  + Domain (tên miền): Là địa chỉ mà người truy cập sử dụng để tìm ra website doanh nghiệp trên mạng lưới internet. Website muốn hoạt động được bắt buộc phải có domain.
  + Hosting: Đây là nơi lưu trữ toàn bộ liệu của trang web bao gồm thông tin, email, dữ liệu…Nếu không có hosting website sẽ không thể xuất hiện trên internet và cũng không thể tiếp cận được với người dùng.
  + Source code (mã nguồn): Là tập hợp của rất nhiều dòng lệnh khác nhau giúp tạo ra những tác vụ mà người dùng có thể thực hiện ngay trên website.
  + Nội dung website: Là tất cả thông tin dữ liệu về các sản phẩm, dịch vụ mà công ty bạn cung cấp có thể được trình bày bằng văn bản, hình ảnh, video…để giới thiệu đến người truy cập.
* Sitemap
  + Sitemap là một file liệt kê tất cả nội dung trang web ở định dạng XML. Qua đó các công cụ tìm kiếm như Google có thể dễ dàng khám phá và index nội dung của bạn.
  + Các loại sitemap: XML (Dành cho bot của công cụ tìm kiếm), HTML (Được hiển thị giúp người dùng truy cập vào giao diện website nhanh chóng, dễ dàng).
  + Về dạng:
    - Sitemap Index: Tập hợp các Sitemap được đính kèm và được dùng để đặt trong file robots.txt
    - Sitemap-category.xml: Tập hợp cấu trúc của các danh mục trên website.
    - Sitemap-products.xml: Sitemap dành cho các link chi tiết về các sản phẩm trên trang.
    - Sitemap-articles.xml: Giúp Google tìm thấy nội dung trên các trang web được chấp thuận cho Google Tin tức..
    - Sitemap-tags.xml: Sitemap dành cho các thẻ trên website.
    - Sitemap-video.xml: Sitemap chuyên sử dụng cho video trên các page, website. Được dùng đặc biệt để giúp Google hiểu nội dung video trên website.
    - Sitemap-image.xml: Giúp Google tìm thấy tất cả các hình ảnh được lưu trữ trên website của bạn.
* Trang chủ:
  + Toàn bộ website đều được thiết lập quanh trang chủ.
  + Trang chủ xác định rõ chủ đề website
  + Trang chủ xác định phong cách thiết kế của website
  + Lưu ý khi sử dụng đồ hoạ
* Hệ thống liên kết:
  + Liên kết: là thuật ngữ rất phổ biến trong tối ưu website, nó có chức năng liên kết từ trang này sang trang khác, cho phép bạn truy cập từ đối tượng hiện tại tới một địa chỉ web nào đó bằng cách nhấp chuột.
  + Có những loại liên kết:
    - Liên kết ngoại (External Link): là một dạng liên kết mà đường link của nó đi từ một tên miền sang một tên miền khác, liên kết với trang web uy tín sẽ làm gia tăng sự uy tín của website bạn.
    - Liên kết nội ( Internal Link ) là loại liên kết dẫn dắt người dùng từ trang này sang trang khác cùng một tên miền đang được sử dụng, việc này có tác dụng gia tăng tương tác giữa người dùng và website.
    - Backlink: Cũng giống như là External Link là liên kết từ tên miền này sang tên miền khác, nhưng hình thức hoạt động có điều ngược lại đó là trang web của bạn được tìm đến từ trang web khác, điều này có nghĩa là liên kết chỉ đến web của bạn được đặt trên một trang/bài viết của đối tượng khác.
* Phân biệt giao diện cho người dùng cuối và giao diện cho người quản trị web:
  + Phần admin chỉ dành cho người quản trị website – những người có tài khoản và quyền để vào. Còn phần public là thông tin công cộng ai cũng xem được.

2.2 Giao diện website

* Quan điểm khi thiết kế giao diện website
  + Đảm bảo tính thống nhất trong thiết kế
  + Sắp xếp bố cục hợp lý
  + Tránh lạm dụng các hiệu ứng hình ảnh
  + Tránh sử dụng quá nhiều chi tiết trên một thiết kế
  + Cung cấp các phản hồi hành động
  + Màu sắc giao diện đồng bộ, thể hiện giá trị thương hiệu
  + Thiết kế giao diện tương thích thiết bị di động
* Nguyên tắc CRAP
  + CRAP bao gồm:
    - Contrast (sự tương phản)
    - Repetition (sự lặp lại)
    - Alignment(sự sắp hàng)
    - Proximity (khoảng cách gần)
* 8 Lỗi Thường Gặp Khi Thiết Kế Website
  + Kích cỡ website quá lớn.
  + Màu sắc và độ tương phản.
  + Không biết mua hàng thế nào.
  + Điều hướng kém, bố cục rối rắm.
  + Không phân loại nội dung.
  + Không tương tác với người truy cập.
  + Website có thiết kế quá mức cần thiết.
  + Nhồi nhét quá nhiều thứ vào 1 website.
* Phân biệt lập trình winform và lập trình webform.

|  |  |
| --- | --- |
| Winform | Webform |
| Sử dụng trên các ứng dụng  Windowns của máy. | Sử dung trực tiếp trên web thông qua Internet. |
| Tốc độ xử lí nhanh.  Thao tác trên nhiều giao diện.  Đảm bảo an toàn, bảo mật thông tin.  Chạy trên các phiên bản  Windowns khác nhau. | Tốc độ xử lí phụ thuộc do mạng.  Phụ thuộc nhiều vào máy chủ. |
| Phải cài đặt phần mềm trên máy.  Chỉ phù hợp sử dụng trên máy bàn.  Đồ họa thiếu tính trực quan  giao diện kém thân thiện với  người dùng. | Bảo mật tuyệt đối, bảo trì linh hoạt.  Tốc độ xử lí dữ liệu nhanh.  Có thể đăng nhập mọi lúc, mọi nơi.  Quản lý chuỗi hệ thống tiện lợi.  Thao tác đơn giản và nhanh chóng.  Hoạt động tốt ngay cả khi không có  kết nối internet. |

* Ví dụ minh họa:

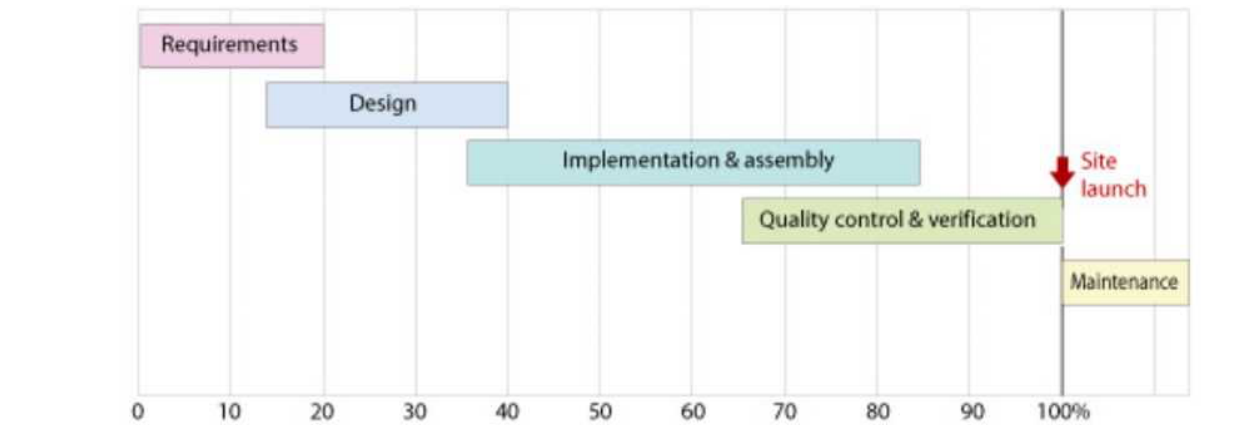
1. Khi bạn đăng nhập vào tài khoản gmail,facebook, nhập thông tin mua hàng online, đăng ký form hay bất kì thao tác nào trên internet… Tất cả chúng đều thông qua webform.
2. Webform là mô hình lập trình rất hiện đại, đơn giản, rõ ràng, linh động, hoạt động độc lập… là một trong những công nghệ độc đáo của Microsoft.
3. Winform là công nghệ của Microsoft, cho phép lập trình các ứng dụng Windows trên PC.

4. Quá trình xây dựng website:

4.1 Tổng quan quá trình xây dựng website:

* Bước 1: Tiếp nhận yêu cầu, thông tin từ khách hàng
* Bước 2: Lập kế hoạch
* Bước 3: Lựa chọn tên miền website và hosting
* Bước 4: Thiết kế giao diện
* Bước 5: Xây dựng tính năng website
* Bước 6: Kiểm tra và chỉnh sửa
* Bước 7: Đào tạo
* Bước 8: Bảo trì

4.2 Chi tiết mỗi giai đoạn trong quá trình này



Requirements: Đây là giai đoạn xác định các yêu cầu chức năng và phi chức năng mà hệ thống phần mềm cần có. Kết quả của giai đoạn này là bản tài liệu đặc tả yêu cầu. Tài liệu này sẽ là nền tảng cho những giai đoạn tiếp theo cho đến cuối dự án.

Design: Là giai đoạn định ra làm thế nào để hệ thống phần mềm đáp ứng đúng yêu cầu của khách hàng. Giai đoạn này thực hiện phân tích, thiết kế hệ thống phần mềm.

Implementaation & assembly: Là giai đoạn thực hiện sản phẩm dựa trên đặc tả yêu cầu và tài liệu thiết kế module.Triển khai hệ thống ra môi trường của khách hàng.

Quanlity control & verification: Tester sẽ nhận sản phẩm từ dev và thực hiện kiểm thử cho nhóm các thành phần và kiểm thử hệ thống. Khâu kiểm thử cuối cùng sẽ là Kiểm thử chấp nhận, giai đoạn này còn có sự tham gia của khách hàng.

Maintenace: Đây là giai đoạn cài đặt, cấu hình và đào tạo cho khách hàng. Giai đoạn này sửa chữa những lỗi của sản phẩm (nếu có) và phát triển những thay đổi mới được khách hàng yêu cầu.

4.3 Vai trò của mỗi nhóm người trong quá trình xây dựng website:

Vai trò của người thiết kế: Đây là việc tạo ra bộ mặt website hoàn chỉnh và trình bày các ý tưởng nội dung lên trên website ấy. Thông qua Internet, người dùng sẽ truy cập được trang này với các thiết bị điện tử như điện thoại, máy tính bảng, máy tính, laptop, …. Hiểu đơn giản, thiết kế web là thiết kế một trang web. Người thiết kế web sẽ chịu trách nhiệm diễn đạt tốt nhất ý tưởng nội dung của website. Từ bố cục, màu sắc, hình ảnh, … đến cách bố trí link sao cho thật hài hòa và hợp mắt người dùng. Người làm công việc thiết kế web được gọi là chuyên viên thiết kế web (Web Designer). Bộ mặt website được gọi là giao diện (Template) website. Giao diện này có thể ở dạng động hoặc tĩnh.

Vai trò của lập trình viên:

- Phối hợp với các bộ phận khác để đưa ra ý tưởng cho các mẫu thiết kế phần mềm, ứng dụng mới.

- Xây dựng phần mềm, ứng dụng mới bằng các ngôn ngữ lập trình thích hợp.

- Phát triển và xây dựng các tính năng mới cho ứng dụng.

- Nâng cấp phần mềm và các hệ thống để đảm bảo tính bảo mật và hiệu quả hơn.

- Phối hợp với các Content/Technical Writers để viết các tài liệu hỗ trợ người dùng.

- Kiểm tra và bảo trì các chương trình, ứng dụng định kỳ, tiến hành sửa lỗi khi có vấn đề xảy ra.

Vai trò của khách hàng: Khách hàng là bên đã ra lệnh cho kết quả dự án. Người đó có thể tích cực tham gia dự án hoặc giữ khoảng cách xa hơn. Đôi khi khách hàng cũng là người dùng của kết quả dự án, nhưng đó không phải lúc nào cũng như vậy. Giả sử một trường đại học muốn một ứng dụng web cho nhân viên và sinh viên của mình, thì trường đại học là khách hàng và nhân viên và sinh viên có vai trò người dùng.

1. Khái niệm client-script và sử dụng NNLT Javascript

1.1 Kiến thức về NNLT thông dịch client-script Javascript

* Javascript chính là một ngôn ngữ lập trình web rất phổ biến ngày nay. Javascript được tích hợp đồng thời nhúng vào HTML để hỗ trợ cho website trở nên sống động hơn. Chúng cũng đóng vai trò tương tự như một phần của website, cho phép Client-side Script từ người dùng tương tự máy chủ (Nodejs) để tạo ra những website động.
* Lịch sử phát triển của Javascript
  + Brendan Eich chính là người đã phát triển Javascript tại Netscape với tiền thân là Mocha. Sau đó, Mocha được đổi thành LiveScript và cuối cùng mới đổi thành JavaScript.
  + Năm 1998, JavaScript với phiên bản mới nhất là ECMAScript 2 phát hành và đến năm 1999 thì ECMAScript 3 được ra mắt.
  + Năm 2016, ứng dụng JavaScript đã đạt kỷ lục lên tới 92% website sử dụng, đồng thời cũng được đánh giá là một công cụ cực kỳ quan trọng đối với lập trình viên.
* Javascript dùng để làm gì?
  + Thay đổi nội dung HTML
  + Thay đổi giá trị thuộc tính HTML
  + Thay đổi kiểu HTMLẨn các phần tử HTML
  + Hiển thị các phần tử HTML
* Đặt thẻ Script ở đâu
  + Trong JS, chúng ta có 3 cách để đặt thẻ Script phổ biến như:
  + Internal: Viết ở trong file HTML hiện tại;
  + External: Viết ra một file JS khác và tiếp tục import vào;
  + Inline: Viết trực tiếp ở trong các thẻ HTML;

Một số câu hỏi thường gặp liên quan đến JavaScript

* Thẻ script là gì?
  + Thẻ Script là một công cụ được sử dụng để khai báo cho trình duyệt web biết được nội dung ở bên trong file chính là một đoạn mà JS.
* Type = text/javascript là gì?
  + Khi mà bạn khai báo thẻ Script thì sẽ có xuất hiện thuộc tính gọi là type=”text/javascript. Đây chính là một tham số được sử dụng với mục đích chính là khai báo media type cho các dữ liệu nằm ở bên trong của thẻ script. Hệ thống sẽ tự mặc định là text/javascript nếu như bạn không khai báo.
* Javascript có làm được backend không?
  + Trên lý thuyết thì JS chính là một dạng ngôn ngữ Client nên bạn không thể thao tác được ở phía server. Tuy nhiên, khi mà server của bạn được cài NodeJS thì ở phía server có thể sử dụng được ngôn ngữ lập trình JS để code. Bởi lẽ, NodeJS chính là sử dụng Javascript.
* Có nên học Javascript không?
  + Nếu bạn có mong muốn đi theo lĩnh vực lập trình web, lập trình app mobile thì bạn nên học thêm các kiến thức về Javascript. Tất nhiên, bạn không chỉ học mỗi Javascript thôi mà cần phải bổ sung thêm các kiến thức về FW JS khác để hỗ trợ cho công việc của mình.

1.2 Kiến thức về sử dụng Javascript

* Tìm hiểu hàm console.log
  + Hiều đơn giản hàm console.log là hàm dùng để debug
* Kiến thức về biến cần nắm

Biến dùng để lưu giá trị dữ liệu - để sử dụng đi sử dụng lại ở nhiều nơi - và khi thay đổi cta chỉ cần thay đổi một chỗ - dễ thay đổi, sửa chữa.

***Declare variable: khai báo biến***

// const and let

const number = 100;

console.log(number)

let otherNumber = 200

otherNumber = "Welcome to my JavaScript course"

console.log(otherNumber)

// Hoisting

const anotherNumber = true;

let othervalue = undefined;

console.log(anotherNumber);

// => const và let không bị hoisting

// var sẽ bị hoisting

console.log(a);

var a = 2;

* Khi nào nên sử dụng var, let hoặc const?
  + Luôn khai báo biến
  + Luôn sử dụng const nếu giá trị không nên thay đổi
  + Luôn sử dụng const nếu kiểu không thay đổi (Mảng và Đối tượng)
  + Chỉ sử dụng let nếu bạn không thể sử dụng const
  + Chỉ sử dụng var nếu bạn PHẢI hỗ trợ các trình duyệt cũ.
* Mã định danh JavaScript
* Tất cả các biến JavaScript phải được xác định bằng các tên duy nhất.
* Những tên duy nhất này được gọi là định danh (identifiers).
* Mã định danh có thể là tên ngắn (như x và y) hoặc tên mô tả nhiều hơn (tuổi, tổng, tổng số lượng).
* Các quy tắc chung để xây dựng tên cho các biến (số nhận dạng duy nhất) là:
* Tên có thể chứa các chữ cái, chữ số, dấu gạch dưới và ký hiệu đô la.
* Tên phải bắt đầu bằng một chữ cái.
* Tên cũng có thể bắt đầu bằng $ và \_ (nhưng chúng tôi sẽ không sử dụng nó trong hướng dẫn này).
* Tên phân biệt chữ hoa chữ thường (y và Y là các biến khác nhau).
* Không thể sử dụng các từ dành riêng (như từ khóa JavaScript) làm tên.
* Sử dụng String cơ bản

// String

"Hello world";

"My name is LV";

`I am frontend developer`;

// double quotes, single quote, backticks (template literal)

const name = "LV";

const job = "frontend deveoper";

// console.log(name);

// console.log(typeof name); // xem kiểu dữ liệu của nó là gì

const newStr = "My name is LV and I am frontend developer";

const newStr2 = "My name is " + name + " and I am" + job;

// console.log(newStr2);

* Các phương thức trong String

// ${variable}

const newStr3 = `My name is ${name} and I am ${job}`;

console.log(newStr3);

Các phương thức trong String

// ${variable}

// const newStr3 = `My name is ${name} and I am ${job}`;

// console.log(newStr3);

// .length

// console.log(newStr3.length); // -> length = 40

// index là vị trí của từng kí tự trong chuỗi (bao gồm cả khoảng trắng)

const myStr = " Frontend Developer ";

console.log(myStr.split(" "));

console.log(myStr.split(""));

console.log(myStr.split("/"));

console.log(myStr.toLowerCase());

console.log(myStr.toUpperCase());

// Kiểm tra đầu cuối

console.log(myStr.startsWith("Frontend")); // trả về giá trị true, false

console.log(myStr.endsWith("Developer")); // trả về giá trị true, false

// Kiểm tra sự tồn tại của chuỗi

console.log(myStr.includes("end")); // true

console.log(myStr.includes("abc")); // false

// Kiểm tra kí tự nằm ở vị trí bao nhiêu

console.log(myStr.indexOf("D"));

console.log(myStr.lastIndexOf("e"));

// Thay thế chuỗi

console.log(myStr.replace("Developer", "Designer"));

// Muốn lặp lại chuỗi nhiều lần

console.log(myStr.repeat(5));

// Cắt và tạo ra chuỗi mới

// slice(start, end);

console.log(myStr.slice(0, 8)); // Frontend

console.log(myStr.slice(0)); // Frontend Developer

console.log(myStr.slice(-1)); // r

// Bỏ khoảng trống thừa ở 2 đầu

console.log(myStr.trim());

console.log(myStr.trimStart()); // remove space left

console.log(myStr.trimEnd()); // remove space right

const myStr2 = "Frontend";

// Lấy kí tự theo vị trí Index

console.log(myStr2.charAt(2));

// Lấy ra một phần của chuỗi

// substr(index, length) -> index: vị trí, length: số lượng kí tự muốn lấy

console.log(myStr2.substr(1, 5)); // ronte

// substring -> Lấy ra các kí tự của chuỗi

// substring(start index, end index)

console.log(myStr2.substring(1, 5)); // ront

* Sử dụng nhiều phương thức của String cùng lúc

const myStr3 = " Frontend Developer ";

// Loại bỏ khoảng trống 2 bên, đưa tất cả về IN HOA, thay chữ `Developer` thành chữ `Developer and Designer`, sau đó repeat 2 lần

console.log(

myStr3

.trim()

.replace("Developer", "Developer and Designer")

.toUpperCase()

.repeat(2)

);

console.log(

`Result of myStr3: ${myStr3

.trim()

.replace("Developer", "Developer and Designer")

.toUpperCase()

.repeat(2)}`

);

* Tìm hiểu kiểu dữ liệu number

// Số nguyên 1 2 3 999

// Số thập phân: 3,4 5,8 -> 3.4 5.8

console.log(5 + 7);

console.log(typeof 10);

const number1 = "5";

const number2 = "4.8";

console.log(parseInt(number1)); // 5

console.log(parseFloat(number2)); // 4.8

const number3 = -10;

// Math.abs(value)

console.log(Math.abs(number3)); // 10

// Math.floor(value) -> Làm tròn xuống ví dụ: 4.3 -> 4

console.log(Math.floor(4.3));

// Math.ceil(value) -> Làm tròn lên ví dụ: 4.3 -> 4

console.log(Math.ceil(4.3));

// Math.round(value) -> Làm tròn gần nhất

console.log(Math.round(4.5)); // 4.5 -> 5 dưới 4.5 -> 4

// toFixed(number) ví dụ 0.3333 -> 0.33

// toFixed(2) // sẽ biến số thành string

console.log(parseFloat(1 / 3).toFixed(2));

console.log(Math.random()); // random 0 -> 1

console.log(Math.ceil(Math.random() \* 10)); // random 1 -> 10

// Math.max(1, 3, -5, 100); -> Trả về con số lớn nhất

console.log(`Max: ${Math.max(1, 2, 3)}`);

// Math.min(1, 3, -5, 100); -> Trả về con số nhỏ nhất

console.log(`Min: ${Math.min(1, 2, 3)}`);

// isNaN(value) and Number.isNaN(value)

// NaN -> Not a Number

console.log(isNaN("this is a string")); // true

console.log(isNaN("12345")); // isNaN(12345) -> false

console.log(Number.isNaN("this is a string")); // false

console.log(Number.isNaN(NaN)); // false

* Tìm hiểu các kiểu dữ liệu khác

// undefined -> khai báo nhưng chưa gán dữ liệu

let a;

console.log(a);

// null -> nothing

let d = null;

console.log(d);

// Boolean -> true or false

// falsy values vs truthy values

// falsy values: "", 0, null, undefined, false

// trusy values: "abc", 1, true, 100, 1000...

* Tìm hiểu Number

"4.5"; // 4.5 parseFloat

"4"; // 4 parseInt

// Number(value)

console.log(Number("4.5"));

console.log(Number("4"));

console.log(Number("abc")); // NaN

console.log(Number(undefined)); // NaN

console.log(Number(null)); // 0

console.log(Number(false)); // 0

console.log(Number(true)); // 1

console.log(Number("")); // 0

console.log(Number(NaN)); // NaN

* Tìm hiểu hàm String

// String(value)

console.log(String(4.5)); // "4.5"

console.log(String(null)); // "null"

console.log(String(undefined)); // "undefined"

console.log(String(NaN)); // "NaN"

console.log(String(false)); // "null"

console.log(String(true)); // "null"

* Tìm hiều hàm Boolean

// Boolean(value) -> true or false

console.log(`Boolean of zero: ${Boolean(0)}`); // false

console.log(`Boolean of "": ${Boolean("")}`); // false

console.log(`Boolean of false: ${Boolean(false)}`); // false

console.log(`Boolean of null: ${Boolean(null)}`); // false

console.log(`Boolean of undefined: ${Boolean(undefined)}`); // false

console.log(`Boolean of NaN: ${Boolean(NaN)}`); // false

console.log(`Boolean of text Vu: ${Boolean("Vu")}`); // true

* Type coercion là gì

// type coercion

// + - \* /

// -> weird

// -> Dấu + nó đặc biệt hơn các dấu khác

console.log(1 + 2);

console.log(10 + 10);

console.log(10 + "10"); // 1010

console.log("10" + 10); // 1010

// 10 -> "10" + "10" ->"1010"

// console.log(String(10) + "10")

console.log("10" - 10); // 0 Number("10") - 10

console.log(null + ""); // "null"

console.log(null + undefined); // NaN

console.log("" - 1); // Number("") -> 0 - 1 = -1

console.log(false - true); // -1

console.log(null + 10); // 10

**Các ví dụ:**

// true + false -> Number(true) + Number(false) -> 1 + 0 -> 1

// 12 / "6" -> 12 / Number("6") -> 12 / 6 -> 2

// "number" + 15 + 3 -> ("number" + 15) + 3 -> "number15" + 3 -> "number15" + "3" -> "number153"

// 15 + 3 + "number" -> 18 + "number" -> "18" + "number" -> "18number"

// "foo" + + "bar" -> "foo" + (+ "bar") -> "foo" + NaN -> "foo" + "NaN" -> "fooNaN"

// 'true' == true -> Number("true") == Number(true) -> NaN == 1 -> false

// false == 'false' -> Number(false) == Number('false') -> 0 == NaN -> false

// null == '' -> false

// !!"false" == !!"true" -> Boolean("false") == Boolean("true") -> true == true -> true

// +"123s" -> NaN

* Functions

// Khai báo function (hàm)

/\*

Cú pháp (syntax): function functionName(parameters: tham số) {

// Your code here

}

\*/

/\* Đề: tính tổng 2 số a và b \*/

function sum(a = 0, b = 0) {

return a + b;

// ko có return thì kq của hàm trả về là underfined

}

// function sum(parameter = defaultValue)

// invoke function

console.log(sum(5, 10));

// invoke function sum(arguments: đối số)

// Lưu function vào một biến rồi gọi sau

function add(a = 0, b = 0) {

return a + b;

}

const sum2 = add; // gán function add cho biến sum2 nhưng ch được gọi

// Tham số là function

// Tính trung bình của a và b ->(a + b)/2

function average(a, b, fn) {

const total = fn(a, b);

return total / 2;

}

let result = average(200, 300, sum2);

console.log(`result ${result}`);

// Function declaration

// Anonymous function (function expresstion)

//-> Không bị hoisting

const logName = function () {

console.log("your name");

};

logName();

// IIFE -> Immediately Invoked Function Execution

(function () {

console.log("This is IIFE function");

})();

// Scope

// Global scope, function scope

let myName = "LV"; // global scope

function logYourName() {

let myName2 = "LuonVuiTuoi"; // function scope

console.log(myName2);

}

console.log(myName);

if (2 > 1) {

// Block scope

let message = "Hello LV";

const message2 = "Hello LV";

// global scope

// hoisted

var message3 = "Hello LV";

}

// alert(message3);

// Lexical scope: là định nghĩa phạm vi của cái biến đó, bởi vị trí của nó đc khai báo trong source code của cta

let aNewName = "Evondev"; // global scope

function sayHello() {

let message5 = "Hi"; // block scope

console.log(`${message5} ${aNewName}`);

}

sayHello();

// Closure

// parent function

function sayHello2() {

let message = "Hi";

// inner function

function sayHi() {

console.log(message);

}

return sayHi;

}

let hello = sayHello2(); // otherMessage is no longer accessible

hello();

function sayHello3(message) {

return function hiYourName(name) {

console.log(`${message} ${name}`);

};

}

let hello2 = sayHello3("welcome to javascript");

hello2("closure");

// arrow function: anonymous function

// ECMAScript

// Normal function: anonymous function

const square = function (x) {

return x \* x;

};

square(5); // 25

// Arrow function

const square1 = () => {

return 1000;

};

// rút gọn

const square2 = () => 1000;

const square3 = (x) => x \* x;

/\* Khi xử lí nhiều dòng

const square = () => {

handle code

return; } \*/

* Khái niệm cơ bản về XSS: XSS (Cross site scripting ) là một lỗ hổng ứng dụng web trong đó một người dùng cuối có thể tấn công bằng cách chèn vào các website động (ASP, PHP, CGI, JSP ...) những thẻ HTML hay những đoạn mã script nguy hiểm có thể gây nguy hại cho những người sử dụng khác. Lỗ hổng XSS đã tồn tại từ lâu nhưng kịch bản hiện nay vẫn có thể thực hiện với những kiểu tấn công mới trong tương lai.

***2. Khái niệm & sử dụng mô hình Javascript + HTML DOM, xử lý sự kiện***

2.1 Kiến thức về sự kiện và xử lý sự kiện

**DOM toàn tập:**

* // 1. DOM là gì?

// Document Object Model

Selecting-node

// DOM đơn giản là những HTML tag cta đã gặp

// DOM attribute VD: src, href

// DOM node VD: body tag, html tag, a tag

* // 2. Selecting nodes: lựa chọn node, truy vấn node trong HTML

// 2.1 document.querySelector("selector") -> trả về 1 node nếu tồn tại selector đó, ngược lại sẽ trả về null

// selectors: .header, p, body, #heading

const singleNode = document.querySelector("h1");

const singleNode2 = document.querySelector(".container");

const singleNode3 = document.querySelector("#spinner");

// console.log(singleNode)

// console.log(singleNode2)

console.log(singleNode3);

// 2.2 document.querySelectorAll("selector") -> Trả về một nodeList chứa danh sách các node - nếu ko có thì trả về empty

// nó có thể loop và sử dụng forEach

// nó giống array là có thể loop nhưng ko sử dụng được các phương thức như: push, pop, shift

const multiNodes = document.querySelectorAll(".item");

console.log(multiNodes);

// 2.3 document.getElementsByClassName("className") -> trả về một HTMLCollection chứa danh sách các node, nếu không có thì trả về empty

const classNodes = document.getElementsByClassName("item");

console.log(classNodes);

// 2.4 document.getElementsByTagName("tagName") -> trả về một HTMLCollection chứa danh sách các node, nếu không có thì trả về empty

const tagNodes = document.getElementsByTagName("li");

console.log(tagNodes);

// 2.5 document.getElementsById("id") -> trả về 1 node

const idNode = document.getElementById("spinner")

console.log(idNode)

* Property-method

// 1. selector.getAttribute("attribute") -> lấy ra giá trị của attribute selector

// selector: 1 cái

// attribute -> thuộc tính href, src, id, class, style,

const link = document.querySelector(".link");

console.log(link.getAttribute("href"));

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

// li.getAttribute("class")

li.forEach((item) => {

console.log(item.getAttribute("class"));

});

console.log(li);

// 2. selector.setAttribute("attribute", value) -> set giá trị cho attribute nào đó của selector

link.setAttribute("target", "\_blank");

const listLinks = document.querySelectorAll("a.link");

listLinks.forEach((item) => item.setAttribute("target", "\_blank"));

// 3. selector.removeAttribute("attribute") -> xóa attribute của selector

const p = document.getElementById("spinner");

p.removeAttribute("style");

// 4. selector.hasAttribute("attribute") -> kiểm tra selector có attribute nào đó hay ko - nếu có thì trả về true, ngược lại trả false

console.log(p.hasAttribute("class")); // false

console.log(p.hasAttribute("id")); // true

if (p.hasAttribute(id)) {

// do something here

}

* Text

// 1. textContent -> Lấy ra nội dung(only text) bao gồm text trong html tag(nếu có) của selector

const spinner = document.querySelector("#spinner");

// Thay đổi text content

// spinner.textContent = "Hello new content";

// 2. innerText

console.log(spinner.textContent);

console.log(spinner.innerText);

// spinner.textContent = `<div class="demo">hello html</div>`;

// 3. innerHTML -> lấy ra nội dung của selector bao gồm cả HTML

console.log(spinner.innerHTML);

spinner.innerHTML = `<div class="demo">hello html</div>`;

* Traversing

// 1. parentNode, parentElement, removeChild

// parentNode, parentElement

const span = document.querySelector("span");

console.log(span.parentNode.parentNode);

console.log(span.parentElement.parentElement);

// selector.parentNode hoặc selector.parentElement.removeChild(delete selector)

// span.parentNode.removeChild(span);

// span.remove();

// 2. nextElementSibling: tìm tới phần tử kế tiếp vs previousElementSibling: tìm tới phần tử đứng sau nó

const nextSpan = span.nextElementSibling.nextElementSibling;

console.log(nextSpan)

const previousLink = span.previousElementSibling;

console.log(previousLink)

// 3. childNodes: trả về hết các node bên trong bao gồm textNodes - children: trả về các node không bao gồm textNodes -> trả về một mảng

console.log(span.childNodes) // lấy luôn cả khoảng trắng dưới dạng text

console.log(span.children[1])

// 4. firstChild vs firstElementChild

console.log(span.firstChild) // text

console.log(span.firstElementChild) // strong

// 5. lastChild vs lastElementChild

console.log(span.lastChild) // text

console.log(span.lastElementChild) // strong

// 6. nextSibling vs previousSibling

console.log(span.nextSibling)

console.log(span.previousSibling)

* class

// 1. selector.classList.add("class")

const container = document.querySelector(".container");

// container.classList.add("is-active");

// 2. selector.classList.remove("class")

// container.classList.remove("container");

// 3. selector.classList.contains("class")

// console.log(container.classList.contains("container")); // false

// console.log(container.classList.contains("is-active")); // true

// 4. selcctor.classList.toggle("class")

// Diễn tả hoạt động của toggle bằngc code

// if (container.classList.contains("is-active")) {

// container.classList.remove("is-active");

// } else {

// container.classList.add("is-active");

// }

// container.classList.toggle("is-active");

// 5. selector.classname -> trả về chuỗi các class của selector

// const title = document.querySelector(".title");

// console.log(title.className);

// title.className = "title" // cách để thay thế or xóa nhiều class

// menu toggle class

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

menu.classList.add("is-show");

* Insert

// 1. element.insertAdjacentText(position, text)

const h3 = document.querySelector("h3");

// h3.insertAdjacentText("position", "text");

// position: beforebegin, afterbegin, beforeend, afterend

h3.insertAdjacentText("beforebegin", "begin");

h3.insertAdjacentText("afterbegin", "afterbegin");

h3.insertAdjacentText("beforeend", "beforeend");

h3.insertAdjacentText("afterend", "afterend");

// 2. element.insertAdjacentElement(position, node)

const strong = document.createElement("strong");

strong.classList.add("bold");

h3.insertAdjacentElement("beforeend", strong);

// 3. element.insertAdjacentHTML()

/\*

<ul class="menu">

<li>1</li>

<li>2</li>

<li>3</li>

</ul>

\*/

const template = `

<ul class="menu2">

<li>1</li>

<li>2</li>

<li>3</li>

</ul>`;

document.body.insertAdjacentHTML("beforeend", template) // parse

<!-- beforebegin -->

<!-- <h3> -->

<!-- afterbegin -->

<!-- <a href="#" class="demo">demo</a>

<span class="span">

<strong>evondev</strong>

<strong class="strong">evondev</strong>

</span>

<span class="span2">tuan</span>

<span class="span3">tuan2</span> -->

<!-- beforeend -->

<!-- </h3> -->

<!-- afterend -->

* Crud

// thêm xóa sửa node trong javaScript

// 1. Tạo ra Element trong javaScript: document.createElement("tag")

const div = document.createElement("div");

// 2. selector.appendChild

// document.body -> thẻ body

// document.querySelector("body")

const body = document.body;

body.appendChild(div);

div.classList.add("container");

div.className = "container wrapper";

div.textContent =

" Lorem ipsum dolor, sit amet consectetur adipisicing elit. Laboroptio at illo neque quossoluta natusiure amet autem ipsum nobis dolores odio eius iusto sunt cum fugiatquis.";

div.innerHTML = `<div class="content"> <h3></h3> </div> `;

div.setAttribute("data-name", "evondev");

// 3. document.createTextNode

const text = document.createTextNode("Hello my name is LV");

const h1 = document.createElement("h1");

body.appendChild(h1);

h1.appendChild(text);

// 4. element.cloneNode(boolean) -> nếu truyền giá trị true vào thì nó sẽ lấy hết nội dung mà thẻ muốn copy - truyền false thì ngược lại chỉ lấy được cái tag

const h1Clone = h1.cloneNode(true);

body.appendChild(h1Clone);

// 5. element.hasChildNodes() -> Kiểm tra có phần tử con hay không - Có thì trả về -> true, Không thì trả về false

console.log(h1.hasChildNodes()); // true

console.log(document.querySelector("h3").hasChildNodes()); // false

* Other

// document.title -> thẻ title cửa web

console.log(document.title); // evondev

document.title = "Welcome to Javascript course";

// document.head

console.log(document.head);

// <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />

const link = document.createElement("link");

link.setAttribute("rel", "preconnect");

link.setAttribute("href", "https://fonts.gstatic.com");

// document.head.appendChild(link)

const head = document.getElementsByTagName("head");

head[0].appendChild(link);

console.log(head);

// insertBefore

// parentNode.insertBefore(newNode, exitingNode)

const ul = document.querySelector("ul");

// document.body.insertBefore(ul, document.querySelector("h3"));

// insertAdjacentElement

// document.querySelector("h3").insertAdjacentElement("beforebegin", ul);

// Sự khác nhau:

// adj thì chỉ chuyển vào được 4 vị trí cố định: beforebegin, beforeend, afterbegin, afterend

// insertbefore thì ngay chỗ exitingNode thì mình có thể chọn những element phía trước or phía sau nó luôn.

document.body.insertBefore(ul, document.querySelector("h3").nextElementSibling);

// replaceChild

// node.replaceChild(newnode, oldnode);

const span = document.createElement("span");

span.textContent = "abc";

document.body.replaceChild(span, document.querySelector(".boxed"));

// Convert HTML collection, NodeList to Array -> để sử dụng được những method của array

const li = document.getElementsByTagName("li");

console.log(li);

// li.forEach((item)=>item) // sẽ bị lỗi Vì html collection ko dùng đc for each

// Array.from(HTML collection or NodeList)

// [...HTMLCollection], [... NodeList]

[...li].filter((item) => item);

console.log(li);

// 4 tag duy nhất và cách truy xuất

// html -> document.documentElement

// body -> document.body

// head -> document.head

// title -> document.title

* Event:
* Sự khác nhau giữa onclick và addEventListener
  + onclick: chấp nhận 1 event handle
  + addEventListener: chấp nhận nhiều event handle cùng lúc - có bubbling và capturing - có once =(default: false) if equal true → chỉ chạy một lần duy nhất

// 1. Sự kiện là gì? (event) Sự kiện là một hành động tác động lên các đối tượng HTML, qua đó ta có thể bắt được sự kiện và yêu cầu javascript thực thi một chương trình nào đó.

// 2. Thêm 1 sự kiện:

// selector.addEventListener("eventName", handler, [options])

// click, keydown, keypress, keyup, mouseover, ...

// handler: function

// capture, bubbling

***Event click***

// 1. Sự kiện là gì? (event)

// 2. Thêm 1 sự kiện:

// selector.addEventListener("eventName", handler, [options])

// click, keydown, keypress, keyup, mouseover, ...

// handler: function

// capture, bubbling

// const button = document.querySelector(".button");

// const span = document.querySelector(".button span");

// function handleClick() {

// console.log("click button");

// }

// button.addEventListener("click", function () {

// console.log("click me");

// });

// Lỗi sai khi dùng function vào eventListener

// button.addEventListener("click", handleClick());

// Đúng

// button.addEventListener("click", handleClick, { capture: true });

// event: e

// span.addEventListener(

// "click",

// function () {

// console.log("click span 2");

// },

// { capture: true }

// );

// document.body.addEventListener(

// "click",

// function () {

// console.log("click body");

// },

// { capture: true }

// );

// span.addEventListener(

// "click",

// function (e) {

// // e.stopPropagation(); // để ngăn chặn quá trình nổi bọt và chỉ lấy sự kiện mình muốn

// e.stopImmediatePropagation(); // trường hợp mà cái element đó được sử dụng nhiều sự kiện

// console.log("click span");

// },

// { capture: true }

// );

// bubbling: nổi bọt

// Sự kiện click chạy từ trong ra ngoài

// capture: truyền vào giá trị boolean (default: false) truyền vào true - thì nó sẽ ngược lại vs bubbling

// target vs curretTarget

// button.addEventListener("click", function (event) {

// event.target: nó sẽ chọn chính xác element mà mình click tới

// console.log(event.target);

// event.curreentTarget: nó sẽ chọn phần tử mà mình click

// console.log(event.currentTarget);

// });

// event.preventDefault();

const link = document.querySelector(".link");

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

e.preventDefault(); // bỏ đi mặc định của thẻ a -> thẻ a sẽ ko load ra trang mới đc nữa

// console.log("click me");

// selector.style.property = value;

// event.target.style.property = value;

// console.log(e.target);

// console.log(e.target.style);

// e.target.style.color = "red";

// background-color -> backgroundColor

// position -> position

// Lấy custom attribute data-name -> event.target.dataset.name

// const name = e.target.dataset.name;

// Nếu mà tên dài thì dùng camelCase

const name = e.target.dataset.nameAbc;

console.log(name);

});

***VÍ DỤ EVENT CLICK***

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

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

<title>Evondev</title>

<link rel="stylesheet" href="css/reset.css" />

<link rel="stylesheet" href="scss/css/hi.css" />

<link

rel="stylesheet"

href="./fontawesome-free-6.3.0-web/fontawesome-free-6.3.0-web/css/all.min.css"

/>

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />

</head>

<body>

<!-- data-yourName="value" -->

<!-- <a href="http://evondev.com" class="link" data-name="LV">evondev</a> -->

<!-- <a href="http://evondev.com" class="link" data-name-abc="LV">evondev</a> -->

<!-- <button class="button"><span>Button</span></button> -->

<!-- EX1: Change color -->

<!-- <button class="change">Change color</button> -->

<!-- EX2: Counter -->

<!-- <div class="counter">

<button class="counter-decrease">

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

</button>

<h1 class="counter-number">0</h1>

<button class="counter-increase">

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

</button>

</div> -->

<!-- EX8: Dropdown -->

<!-- <div class="dropdown">

<div class="dropdown\_\_select">

<span class="dropdown\_\_selected">Call to action</span>

<i class="fa fa-caret-down dropdown\_\_caret"></i>

</div>

<ul class="dropdown\_\_list">

<li class="dropdown\_\_item">

<span class="dropdown\_\_text">New project</span>

<i class="fa fa-plus-circle dropdown\_\_icon"></i>

</li>

<li class="dropdown\_\_item">

<span class="dropdown\_\_text">View profile</span>

<i class="fa fa-user dropdown\_\_icon"></i>

</li>

<li class="dropdown\_\_item">

<span class="dropdown\_\_text">Settings</span>

<i class="fa fa-cog dropdown\_\_icon"></i>

</li>

<li class="dropdown\_\_item">

<span class="dropdown\_\_text">Logout</span>

<i class="fa fa-circle dropdown\_\_icon"></i>

</li>

</ul>

</div> -->

<script src="./dom/events/ex.js"></script>

</body>

</html>

// EX1: Change color

// const button = document.querySelector(".change");

// button.addEventListener("click", handleChangeColor);

// const colors = [

// "#ffa400",

// "#00aefd",

// "#07a787",

// "#ffb86c",

// "#2979ff",

// "#e74c3c",

// ];

// function handleChangeColor() {

// // Lấy random màu từ mảng colors

// const color = colors[Math.floor(Math.random() \* colors.length)];

// document.body.style.backgroundColor = color;

// }

// EX2: counter

// const buttonMinus = document.querySelector(".counter-decrease");

// const buttonPlus = document.querySelector(".counter-increase");

// const counterNumber = document.querySelector(".counter-number");

// let counterValue = parseInt(counterNumber.textContent);

// buttonPlus.addEventListener("click", function () {

// if (counterValue >= 10) return 10;

// counterValue++;

// counterNumber.textContent = counterValue;

// });

// buttonMinus.addEventListener("click", function () {

// if (counterValue <= 0) return 0;

// counterValue--;

// counterNumber.textContent = counterValue;

// });

/\* EX8: Modal \*/

// const dropdownSelect = document.querySelector(".dropdown\_\_select");

// const dropdownItems = document.querySelectorAll(".dropdown\_\_item");

//constdropdownSelected = document.querySelector(".dropdown\_\_selected");

// const dropdownList = document.querySelector(".dropdown\_\_list");

// const dropdown = document.querySelector(".dropdown");

// const dropdownCaret = document.querySelector(".dropdown\_\_caret");

// dropdownSelect.addEventListener("click", function (e) {

// dropdownList.classList.toggle("show");

// dropdownCaret.classList.toggle("fa-caret-up");

// dropdownCaret.classList.toggle("fa-caret-down");

// });

// dropdownItems.forEach((item) =>

// item.addEventListener("click", function (e) {

// const text = e.target.querySelector(".dropdown\_\_text").textContent;

// dropdownSelected.textContent = text;

// dropdownList.classList.remove("show");

// dropdownCaret.classList.toggle("fa-caret-up");

// dropdownCaret.classList.toggle("fa-caret-down");

// })

// );

// document.addEventListener("click", function (e) {

// if (!dropdown.contains(e.target)) {

// dropdownList.classList.remove("show");

// dropdownCaret.classList.toggle("fa-caret-up");

// dropdownCaret.classList.toggle("fa-caret-down");

// }

// });

***LOAD EVENT***

// document.addEventListener("DOMContentLoaded");

// Điều kiện ở này là khi HTML load xong

document.addEventListener("DOMContentLoaded", function () {

console.log("DOM loaded");

});

// window.addEventListener("load")

// Điều kiện là khi website load xog

window.addEventListener("load", function () {

console.log("your web loader");

});

***MOUSE EVENT***

const button = document.querySelector(".button");

// mousemove, mouseover, mouseenter, mouseleave

// mousemove: khi mình di chuyển trên element thì sự kiện sẽ xảy ra

/\* button.addEventListener("mousemove", function () {

console.log("mousemove");

}); \*/

// mouseover: nó sẽ chạy khi rê chuột vào phần tử và con của phần tử đó

/\* button.addEventListener("mouseover", function () {

console.log("mouseover");

}); \*/

// mouseenter: nó sẽ chạy khi rê chuột vào phần tử

/\* button.addEventListener("mouseenter", function () {

console.log("mouseenter");

}); \*/

// mouseleave: nó sẽ chạy khi rê chuột vào phần tử rồi rê ra ngoài

// button.addEventListener("mouseleave", function () {

// console.log("mouseleave");

// });

// pageX, pageY, clientX, clientY

document.addEventListener("mousemove", function (e) {

console.log(`pageY:${e.pageY}`);

console.log(`pageY:${e.clientY}`);

});

// clientY: nó sẽ lấy tòa độ theo viewport

// pageY: nó sẽ lấy tòa độ của document, khi mà có scroll thì nó sẽ thay đổi

// X: chiều ngang

// Y: chiều dọc

***VÍ DỤ MOUSE EVENT:***

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

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

<title>Evondev</title>

<link rel="stylesheet" href="css/reset.css" />

<link rel="stylesheet" href="scss/css/hi.css" />

<link

rel="stylesheet"

href="./fontawesome-free-6.3.0-web/fontawesome-free-6.3.0-web/css/all.min.css"

/>

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />

</head>

<body>

<!-- <button class="button">

<span>button</span>

</button> -->

<!-- <a href="http://evondev.com" class="link" data-name="LV">LuonVuiTuoi</a> -->

<!-- EX1: Tooltip -->

<!-- <div class="tooltip">

<div class="tooltip-text">abc</div>

<span class="text">Hello word</span>

</div> -->

<!-- <span class="text" data-tooltip="Welcome to my tooltip">Hello word</span> -->

<!-- EX2: Link hover effect -->

<!-- <ul class="menu">

<li class="menu-item">

<a href="#" class="menu-link">Home</a>

</li>

<li class="menu-item">

<a href="#" class="menu-link">About</a>

</li>

<li class="menu-item">

<a href="#" class="menu-link">Work</a>

</li>

<li class="menu-item">

<a href="#" class="menu-link">Contact</a>

</li>

</ul> -->

<!-- EX3: Image zoom -->

<div class="image-wrapper">

<img

src="./images/Screenshot 2023-03-01 105245.png"

alt=""

class="image"

/>

<div class="image-cover"></div>

</div>

<script src="./dom/events/ex\_mouse.js"></script>

</body>

</html>

/\* EX1: Tooltip \*/

// window.addEventListener("load", function () {

// const text = this.document.querySelector(".text");

// text.addEventListener("mouseenter", function (e) {

// const title = e.target.dataset.tooltip;

// // console.log(title);

// const tooltipDiv = document.createElement("div");

// tooltipDiv.className = "tooltip-text";

// tooltipDiv.textContent = title;

// document.body.appendChild(tooltipDiv);

// const cords = e.target.getBoundingClientRect();

// // console.log(cords);

// const { top, left, width} = cords;

// const tooltipHight = tooltipDiv.offsetHeight;

// const triangleHeight = 20;

// tooltipDiv.style.left = `${left - width / 2}px`;

// tooltipDiv.style.top = `${top - tooltipHight - triangleHeight}px`;

// });

// text.addEventListener("mouseleave", function () {

// const tooltip = document.querySelector(".tooltip-text");

// if (!tooltip) return;

// tooltip.parentNode.removeChild(tooltip);

// });

// });

/\* EX2: Link hover effect \*/

// window.addEventListener("load", function () {

// const link = [...document.querySelectorAll(".menu-link")];

// link.forEach((item) =>

// item.addEventListener("mouseenter", handleHoverLink)

// );

// const line = this.document.createElement("div");

// line.className = "line-effect";

// this.document.body.appendChild(line);

// function handleHoverLink(e) {

// const { width, height, left, top } = e.target.getBoundingClientRect();

// // console.log({ width, height, left, top });

// const offsetBottom = 5;

// line.style.width = `${width}px`;

// line.style.left = `${left}px`;

// line.style.top = `${top + height + offsetBottom}px`;

// }

// const menu = this.document.querySelector(".menu");

// menu.addEventListener("mouseleave", function (e) {

// line.style.width = 0;

// });

// });

/\* EX3: Zoom image \*/

window.addEventListener("load", function () {

const imageCover = this.document.querySelector(".image-cover");

imageCover.addEventListener("mousemove", handleHoverImage);

const image = this.document.querySelector(".image");

const imageWrapper = this.document.querySelector(".image-wrapper");

let imageWrapperWidth = imageWrapper.offsetWidth;

let imageWrapperHeight = imageWrapper.offsetHeight;

function handleHoverImage(e) {

const pX = e.pageX;

const pY = e.pageY;

image.style = "width: auto; height:auto; max-height:unset";

let imageHeight = image.offsetHeight;

let imageWidth = image.offsetWidth;

let spaceX = (imageWidth / 2 - imageWrapperWidth) \* 2;

let spaceY = (imageHeight / 2 - imageWrapperHeight) \* 2;

imageWidth += spaceX;

imageHeight += spaceY;

let ratioX = imageWidth / imageWrapperWidth / 2;

let ratioY = imageHeight / imageWrapperHeight / 2;

let x = (pX - imageWrapper.offsetLeft) \* ratioX;

let y = (pY - imageWrapper.offsetTop) \* ratioY;

image.style = `left: ${-x}px; top: ${-y}px; width: auto; height: auto; max-height: unset; transform: none;`;

imageCover.addEventListener("mouseleave", handleLeaveImage);

function handleLeaveImage() {

image.style = "";

}

}

});

***KEYDOWN EVENT***

// 1. keydown: Hoạt động khi nhấn bàn phím xuống

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

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

// console.log(e.key);

// console.log(e.keyCode);

// console.log(e.which);

// ascii MDN

// e.key: key nhập vào

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

// console.log("You hit Enter!");

// } else {

// console.log(e.key);

// }

// console.log(e.key);

// if (e.key === "Home") {

// console.log("working");

// }

});

***KEYUP EVENT***

// 2. keyup: Sự kiện này sẽ xảy ra khi nhấn phím rồi thả ra

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

// console.log(e.key);

// e.preventDefault();

});

***KEYPRESS EVENT***

// 3. keypress: sự kiện này xảy ra khi các bạn nhấn phím

// keypress sẽ ignore các phím như delete, mũi tên, page up, page down, home, end, ctrl, alt, shift, esc

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

// console.log(e.key);

// if (e.key === "Home") {

// console.log("working");

// }

});

// Thứ tự ưu tiên keydown -> keypress -> keyup

***CHANGE EVENT***

// 4. change: xảy ra khi gõ xong, nhấn Enter hoặc nhấn chuột ra ngoài 1 lần

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

// console.log("working");

});

***FOCUS EVENT***

// 5. focus:

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

// console.log("input is focusing");

});

***BLUR EVENT***

// 6. blur

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

// console.log("input is blur");

});

***SUBMIT EVENT***

// 7. submit form

const form = document.querySelector(".form");

form.addEventListener("submit", function (e) {

e.preventDefault();

const username = this.elements["username"].value;

const password = this.elements["password"].value;

if (!username) {

alert("please enter your username");

return;

}

if (!password) {

alert("please enter your password");

return;

}

if (password.length <= 3) {

alert("too short");

return;

}

// Truy xuất giá trị

// this, e.target

// console.log(this);

// console.log(e.target);

// this.elements

// console.log(this.elements);

// console.log(this.elements["username"]);

// console.log(this.elements["username"].value);

// const username = this.elements["username"].value;

// const gender = this.elements["gender"].value;

// const hobby = this.querySelectorAll(`input[name="hobby"]`);

// console.log(hobby);

// let hobbyValues = [];

// [...hobby].forEach((item) => hobbyValues.push(item.value));

// console.log(hobbyValues);

// stuck -> bị bí

});

// Validation

// email: abc@gmail.com abc

***VÍ DỤ - LÝ THUYẾT REGULAR***

// Regular expression

<!-- Regex -->

<!-- EX1: email validation -->

<!-- <div class="input-wrapper">

<input type="text" class="input" id="" placeholder="Your email" />

<div class="input-icons">

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

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

</div>

</div> -->

<!-- EX2: password validation -->

<div class="input-wrapper">

<input

type="text"

class="input"

id=""

placeholder="Your password"

/>

<div class="input-icons">

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

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

</div>

<div class="check-list">

<div class="check-item check-length">

<i class="fa fa-circle"></i> At least 8 characters

</div>

<div class="check-item check-upper">

<i class="fa fa-circle"></i> Contains at least 1 uppercase

</div>

<div class="check-item check-number">

<i class="fa fa-circle"></i> Contains at least 1 number

</div>

<div class="check-item check-special">

<i class="fa fa-circle"></i> Contains at least 1 special

character

</div>

</div>

</div>

<script src="./dom/Regular/regex.js"></script>

// regular expression: biểu thức chính quy

// Chức năng: đùng để thay thế một đoạn chữ nào đó - tìm kiếm - khai thác thông tin.

// 1. 2 cách khai báo với regex

const re1 = /hello/;

const rs2 = new RegExp("hello");

// hello word

// regax.test(value) -> true or false

// console.log(re1.test("hello word"));

// 2. Anchor ^ $

// ^ string bắt đầu với từ nào đó

// $ string kết thúc với từ nào đó

/hi/.test("hi"); // true

/hi$/.test("welcome to hi"); // true

/hi$/.test("welcome to hello"); // false

/^hi/.test("hi welcome to hi"); // true

/^hi/.test("welcome to hi"); // false

// 3. ranges []

// [a-z]: các kí tự từ a->z in thường

// [A-Z]: các kí tự từ a->z in HOA

// [0-9]: các số từ 0->9

// [a-z0-9A-Z]: các số từ 0->9 hoặc a->z hoặc A->Z

/[a-z]/.test("a"); //true

/[a-z]/.test("A"); //false

/[A-Z]/.test("A"); //true

/[A-Z]/.test("b"); //false

/[0-9]/.test("1000"); //TRUE

/[0-9]/.test("AA1000abc"); //TRUE

/[0-9]/.test("Aabc"); //false

/[a-z0-9A-Z]/.test("Aabc12CAa"); //true

/[^A-Z]/.test("A"); //false -> Phủ định

// 4. Meta Characters

// \d: khớp vs tất cả các số nó <=> [0-9]

/\d/.test("1234"); // true

// \D: ngược lại vs \d <=> [^0-9]

/\D/.test("123"); // false

// \w: khớp vs các kí tự, số và dấu gạch dưới <=> [a-zA-Z0-9\_]

/\w/.test("\_"); // true

// \W: ngược lại \w <=> (^a-zA-Z0-9\_)

/\W/.test("\_"); // false

// \s: khớp vs các kí tự khoảng trắng: spaces, tab, newline

/\s/.test(" "); //true

// \S: các kí tự không phải khoảng trắng

/\S/.test(" "); // false

// \n: khớp vs newline

// \t: khớp vs tab characters

// .: khớp vs tất cả mọi thứ ngoại trừ newline(\n)

// [^]: Khớp vs tất cả mọi thứ bao gồm newline

// 5. Quantifiers: {n}, {n,m} + ? \*

// string.match(regax) "abc".match(/\w/) -> []

// {n}: số lượng nhất định

// {n, m}: số lượng trong khoảng từ n -> m

// +: lấy một hoặc nhiều kí tự thỏa điều kiện

const str1 = "Welcome to 12344444111";

// console.log(str1.match(/\d\d\d\d/)[0]);

// console.log(str1.match(/\d{4}/)[0]);

// console.log(str1.match(/\d{4,6}/)[0]);

// console.log(str1.match(/\d+/)[0]);

const str2 = "color or colour?";

// ?: có thể có hoặc không có kí tự nào đó

// console.log(str2.match(/colou?r/g)); // ["color", "colour"]

// flag

// g: global

// i: case insensitive

// m: multipe lines

// \*: không có hoặc là nhiều

const str3 = "1234";

// console.log(str3.match(/\d?/g));

// 6. groups()

/(\d{3})(\w+)/.test("123"); // false

/(\d{3})?(\w+)/.test("123"); // true

// 7. Escaping \ / [ ] ( ) { } ? + \* | . ^ $

/\?/.test("?");

/\\*/.test("\*");

// 8. Boundaries \b \B

// \b

"My name is LV".match(/\bLV/g); // ['LV']

// \B

"My name isLV".match(/\BLV/g); // ['LV']

// 9. replace

const str4 = "hello welcome to hello to my hello";

// console.log(str4.replace("hello", "ev")); // ev welcome to hello to my hello

// string.replace(regex, value);

str4.replace(/hello/g, "ev"); // ev welcome to ev to my ev

"welcome 12345".match(/\d+/g);

// Email validattion

// valid email: abc@gmail.com

// const emailInput = document.querySelector(".input");

// emailInput.addEventListener("input", function (e) {

// const value = e.target.value;

// const regaxEmail =

// /^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)\*)|(\".+\"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

// if (regaxEmail.test(value.trim())) {

// console.log("valid email");

// e.target.classList.add("valid");

// e.target.classList.remove("invalid");

// } else {

// console.log("invalid email");

// e.target.classList.add("invalid");

// e.target.classList.remove("valid");

// }

// if (!value) {

// e.target.classList.remove("invalid");

// }

// });

// password validate

window.addEventListener("load", function () {

const passwordInput = this.document.querySelector(".input");

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

const value = e.target.value;

const checkLengthClass =

e.target.parentNode.querySelector(".check-length");

const checkUpperClass =

e.target.parentNode.querySelector(".check-upper");

const checkNumberClass =

e.target.parentNode.querySelector(".check-number");

const checkSpecialClass =

e.target.parentNode.querySelector(".check-special");

const checkItem = e.target.parentNode.querySelectorAll(".check-item");

if (!value) {

[...checkItem].forEach((item) => {

item.classList.remove("unactive");

item.classList.remove("active");

});

return;

}

if (value.length < 8) {

checkLengthClass.classList.add("unactive");

checkLengthClass.classList.remove("active");

} else {

checkLengthClass.classList.add("active");

checkLengthClass.classList.remove("unactive");

}

passwordInputClass(checkUpperClass, value, /[A-Z]/);

passwordInputClass(checkNumberClass, value, /[0-9]/);

passwordInputClass(checkSpecialClass,value,/[!@#$%^&\*()\_+{}[\]?<>-]/);

});

});

function passwordInputClass(selector, value, regex) {

if (!regex.test(value)) {

selector.classList.add("unactive");

selector.classList.remove("active");

} else {

selector.classList.remove("unactive");

selector.classList.add("active");

}

}

***3. Khái niệm server-script và sử dụng NNLT PHP***

PHP (Personal Home Page) là viết tắt hồi quy của “Hypertext Preprocessor”. PHP là một chuỗi ngôn ngữ kịch bản được chạy ở phía server nhằm sinh ra mã HTML trên client. Ngôn ngữ PHP thường được sử dụng để lập trình web vì nó có thể dễ dàng kết nối với các website có sử dụng HTML. Trải qua nhiều cập nhật, PHP đã được tối ưu hóa và sở hữu những ưu điểm như: tốc độ nhanh, cách viết mã rõ ràng, dễ học cho những lập trình viên mới vào nghề. Nhờ tính ứng dụng của mình, PHP đã trở thành một ngôn ngữ lập trình được ưa chuộng.

* Ngôn ngữ lập trình PHP làm được những gì?

Ngôn ngữ lập trình PHP có khả năng tạo dữ liệu, nội dung website và hay nhận dự liệu cookie. Thêm vào đó, khi sử dụng ngôn ngữ này bạn có thể thực hiện được nhiều thao tác, công năng khác nhau.

Một số ứng dụng của ngôn ngữ PHP trong ngành IT:

Thiết lập các chương trình cho server máy chủ

Tạo dòng tập lệnh

Xây dựng các ứng dụng làm việc

Hỗ trợ cho mọi loại cơ sở dữ liệu khác nhau

* Tính năng từ PHP bạn nên biết

PHP được sử dụng rộng rãi bởi sự linh hoạt, dễ sử dụng, dễ chạy cùng nhiều tính năng tiện lợi như:

Dễ dàng xuất ra nhiều loại văn bản khác nhau như XML,…

Hỗ trợ cho các cơ sở dữ liệu

Kết nối với các dịch vụ, giao thức như HTTP, LDAP, NNTP,…

Dễ dàng sử dụng chỉ cần một ổ cắm mạng

Có thể trao đổi văn bản, dữ liệu với nhiều ngôn ngữ khác

Xử lý nhiều mẫu văn bản, biểu thức, phần mở rộng, công cụ, cú pháp khác nhau

Truy xuất và mở rộng dữ liệu XML sang SimpleXML, XMLWriter,…

* Đặc trưng của PHP

Ngôn ngữ PHP đơn giản, dễ sử dụng và là mã nguồn mở, miễn phí.

PHP là ngôn ngữ lập trình quen thuộc với các nhà phát triển và giúp cho quá trình xây dựng web trở nên nhanh hơn.

Phù hợp cho lập trình viên mới.

PHP nhẹ, có nhiều Framework và nâng cao tính bảo mật nếu sử dụng Framework.

PHP giúp cho quá trình tải ứng dụng nhanh hơn và có khả năng tích hợp với nhiều ngôn ngữ lập trình khác.

PHP có thể kết nối cơ sở dữ liệu hiệu quả và giúp quá trình tải ứng dụng nhanh hơn.

* Phân biệt cơ chế hoạt động client-script và server-script

Ngôn ngữ Script có thể chạy frontend hoặc backend. Trong khi các Script chạy frontend (phía client) được xử lý bởi trình duyệt web, script chạy backend (phía server) được chạy trên các servers trước khi chúng được gửi tới trình duyệt:

Đối với Script chạy frontend (phía client): khi trình duyệt yêu cầu trang chứa các scripts client-side, server sẽ gửi source codes có thể thực thi cho trình duyệt. Do đó, người dùng có thể dễ dàng nhìn thấy scripts phía client.

Đối với Script chạy backend (phía server): Servers web sẽ phân tích code bằng cách chuyển chúng thành định dạng HTML đơn giản. Ngôn ngữ Script này cho phép các nhà quản trị web ẩn source code.

* Kiến thức về sử dụng PHP

Cú pháp PHP cơ bản

<?php

// PHP code goes here

?>

**Trong PHP, các keywords (vd if, else, while, echo, vv.), classes, functions, và các functions do người dùng định nghĩa không phân biệt chữ hoa chữ thường.**

Trong ví dụ dưới đây, cả ba câu lệnh echo đều như nhau và hợp lệ:

<!DOCTYPE html>

<html>

<body>

<?php

ECHO "Hello World!<br>";

echo "Hello World!<br>";

EcHo "Hello World!<br>";

?>

</body>

</html>

**Note: Tuy nhiên; tất cả các tên biến có phân biệt hoa thường!**

Tạo biến trong PHP

<?php

$txt = "Hello world!";

$x = 5;

$y = 10.5;

?>

**Quy tắc cho các biến PHP:**

Một biến bắt đầu bằng dấu $, sau đó là tên biến

Tên biến phải bắt đầu bằng một chữ cái hoặc ký tự gạch dưới

Một tên biến không thể bắt đầu bằng một số

Tên biến chỉ có thể chứa các ký tự chữ, số và dấu gạch dưới (A-z, 0-9, và \_ )

Tên biến có phân biệt chữ hoa chữ thường ($age và $AGE là hai biến khác nhau)

**Keyword global được sử dụng để truy cập biến global từ bên trong một hàm.**

<?php

$x = 5;

$y = 10;

function myTest() {

global $x, $y;

$y = $x + $y;

}

myTest();

echo $y; // outputs 15

?>

**Thông thường, khi một hàm được thực thi xong, tất cả các biến của nó sẽ bị xóa. Tuy nhiên, đôi khi chúng ta muốn biến local KHÔNG bị xóa. Chúng ta cần nó cho công việc khác.**

Để làm điều đó, sử dụng keyword static khi bạn lần đầu tiên khai báo biến:

<?php

function myTest() {

static $x = 0;

echo $x;

$x++;

}

myTest();

myTest();

myTest();

?>

**PHP echo và print**

echo và print tương tự nhau. Chúng đều dùng để xuất dữ liệu ra màn hình.

Có một sự khác biệt nhỏ: echo không có giá trị trả về trong khi print có giá trị trả về là 1 nên có thể được sử dụng trong biểu thức. echo có thể nhận nhiều tham số (nhưng thường hiếm dùng) trong khi print chỉ nhận một tham số. echo nhanh hơn một chút so với print.

**PHP có các kiểu dữ liệu sau:**

String

Integer

Float (hoặc Double)

Boolean

Array

Object

NULL

Resource: Kiểu dữ liệu đặc biệt resource không phải là kiểu dữ liệu trong thực tế. Đó là việc lưu trữ một tham chiếu đến các hàm và tài nguyên bên ngoài PHP.

Hàm var\_dump() trả về kiểu dữ liệu (data type) và giá trị (value)

<?php

$x = "Hello world!";

$y = 'Hello world!';

$x = 5985;

var\_dump($x);

$x = 10.365;

$x = true;

$y = false;

$cars = array("Volvo","BMW","Toyota");

class Car {

function Car() {

$this->model = "VW";

}

}

// create an object

$herbie = new Car();

// show object properties

echo $herbie->model;

$x = "Hello world!";

$x = null;

var\_dump($x);

**// Tip: Nếu một biến được tạo mà không có giá trị, nó sẽ tự động được gán một giá trị NULL.**

?>

**Các hàm xử lý chuỗi trong PHP**

<?php

// strlen() - Trả về độ dài của chuỗi

echo strlen("Hello world!"); // outputs 12

// str\_word\_count() - Đếm số từ trong chuỗi

echo str\_word\_count("Hello world!"); // outputs 2

// strrev() - Đảo ngược một chuỗi

echo strrev("Hello world!"); // outputs !dlrow olleH

// strpos() - Tìm kiếm một văn bản trong một chuỗi

//Hàm strpos() tìm kiếm một văn bản cụ thể trong một chuỗi. Nếu tìm thấy, Hàm trả về vị trí ký tự khớp đầu tiên. Nếu không tìm thấy, nó sẽ // trả về FALSE.

// Tìm vị trí từ "world" trong chuỗi "Hello world!":

echo strpos("Hello world!", "world"); // outputs 6

// str\_replace() - Thay thế văn bản trong một chuỗi

echo str\_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!

?>

**PHP Numbers**

PHP Integers

PHP có các hàm sau để kiểm tra xem loại biến có phải là số nguyên không:

is\_int()

is\_integer()

is\_long()

<?php

$x = 5985;

var\_dump(is\_int($x));

$x = 59.85;

var\_dump(is\_int($x));

?>

**PHP Floats**

PHP có các hàm sau để kiểm tra xem loại biến có phải là float không:

is\_float()

is\_double()

<?php

$x = 10.365;

var\_dump(is\_float($x));

?>

**Giá trị số lớn hơn PHP\_FLOAT\_MAX được coi là vô hạn.**

PHP có các hàm sau để kiểm tra xem một giá trị số là hữu hạn hay vô hạn:

is\_finite()

is\_infinite()

**PHP NaN**

NaN là viết tắt của Not a Number - Không phải số.

PHP có các chức năng sau để kiểm tra xem giá trị không phải là số:

is\_nan()

**PHP Numerical Strings**

Hàm PHP is\_numeric() có thể được sử dụng để tìm xem một biến có phải là số không. Hàm trả về true nếu biến là một số hoặc một chuỗi số, false nếu không.

<?php

$x = 5985;

var\_dump(is\_numeric($x));

$x = “5985”;

var\_dump(is\_numeric($x));

$x = “59.85” + 100;

var\_dump(is\_numeric($x));

$x = “Hello”;

var\_dump(is\_numeric($x));

?>

**Note: Từ PHP 7.0: Hàm is\_numeric() sẽ trả về FALSE cho các chuỗi số ở dạng thập lục phân - hexa (vd. 0xf4c3b00c), vì chúng không còn được coi là chuỗi số.**

**PHP Cast Chuỗi và Float thành Integer**

(int), (integer), hoặc hàm intval() thường được sử dụng để chuyển đổi một giá trị thành một số nguyên.

<?php

// Cast float to int

$x = 23465.768;

$int\_cast = (int)$x;

echo $int\_cast;

echo "<br>";

// Cast string to int

$x = "23465.768";

$int\_cast = (int)$x;

echo $int\_cast;

?>

**PHP Functions**

function functionName() {

code được thực thi;

}

**Note: Tên hàm phải bắt đầu bằng một chữ cái hoặc dấu gạch dưới. Tên hàm KHÔNG phân biệt chữ hoa chữ thường.**

**Các đối số của hàm**

Thông tin có thể được truyền đến các hàm thông qua các đối số. Một đối số giống như một biến.

<?php

function familyName($fname, $year) {

echo "$fname Refsnes. Born in $year <br>";

}

familyName("Hege", "1975");

familyName("Stale", "1978");

familyName("Kai Jim", "1983");

?>

**PHP được coi là Loosely Typed Language (Hiểu nôm na là không cần khai báo kiểu biến trong PHP)**

Trong ví dụ sau, chúng ta gửi cả số và chuỗi vào hàm mà không sử dụng strict .Sử dụng strict để báo lỗi khi không đúng kiểu dữ liệu:

<?php declare(strict\_types=1); // strict requirement

function addNumbers(int $a, int $b) {

return $a + $b;

}

echo addNumbers(5, "5 days");

**// strict được kích hoạt và "5 days" không phải số, vì vậy sẽ xuất hiện cảnh báo lỗi**

?>

**Khai báo kiểu trả về**

Để khai báo một kiểu cho hàm có trả về, thêm một dấu hai chấm ( : ) và kiểu dữ liệu bạn muốn trả về ngay trước dấu ngoặc nhọn ( { ) khi khai báo hàm:

<?php declare(strict\_types=1); // strict requirement

function addNumbers(float $a, float $b) : float {

return $a + $b;

}

echo addNumbers(1.2, 5.2);

?>

**PHP $GLOBALS**

$GLOBALS là một biến super global trong PHP được sử dụng để truy cập các biến toàn cục từ bất kỳ đâu trong tập lệnh PHP (cả từ bên trong hàm).

**PHP $\_SERVER**

$\_SERVER là biến super global của PHP chứa thông tin về các tiêu đề, đường dẫn và vị trí tập lệnh.

**PHP $\_REQUEST**

PHP $\_REQUEST là biến super global của PHP được sử dụng để thu thập data sau khi gửi form HTML.

**PHP $\_POST**

PHP $\_POST là biến super global được sử dụng để thu thập form data sau khi gửi form HTML với method="post". $\_POST cũng được sử dụng rộng rãi để truyền biến.

**PHP $\_GET**

PHP $\_GET là biến super global được sử dụng để thu thập form data sau khi gửi form HTML với method="get".

$\_GET cũng có thể thu thập data được gửi trên URL.

**PHP session và cooky.**

**1. Session trong PHP**

<?php session\_start();

// Nếu click vào nút Lưu Session

if (isset($\_POST['save-session']))

{

// Lưu Session

$\_SESSION['name'] = $\_POST['username'];

}

?>

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

</head>

<body>

<h1>

<?php

// Hiển thị thông tin lưu trong Session

// phải kiểm tra có tồn tại không trước khi hiển thị nó ra

if (isset($\_SESSION['name']))

{

echo 'Tên Đăng Nhập Là: ' . $\_SESSION['name'];

}

?>

</h1>

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

<input type="text" name="username" value=""/> <br/>

<input type="submit" name="save-session" value="Lưu Session"/>

</form>

</body>

</html>

// Xóa session name

unset($\_SESSION['name']);

// Xóa hết session

session\_destroy();

**2. Cookie trong PHP**

Lưu trữ Cookie

Để lưu trữ Cookie ta dùng cú pháp sau và phải đặt trước thẻ html: setcookie($name, $value, $expire, $path, $domain).

Trong đó:

$name: là tên của Cookie

$value: giá trị của Cookie

$expire: thời gian sống của Cookie

$path : đường dẫn lưu trữ Cookie

$domain: tên của domain

<!DOCTYPE html>

<?php

setcookie('username', 'thehalfheart', time() + 3600);

?>

<html>

<head>

<title></title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

</head>

<body>

</body>

</html>

Lấy giá trị Cookie

if (isset($\_COOKIE['username']))

{

echo $\_COOKIE['username'];

}

Xóa Cookie

Để xóa Cookie bạn chỉ việc thiết lập thời gian sống của nó sang quá trị âm nhiều hơn hoặc bằng giá trị sống lúc bạn thiết lập.

Ví dụ: xóa Cookie đã thiết lập ở trên

setcookie("username", "", time()-3600);

Lập trình hướng đối tượng trong PHP

Phân biệt cách dùng từ khoá static::method() với self::method()

=> Kết luận:

Self: Truy xuất đến class khai báo nó.

Static: Truy xuất đến đối tượng hiện tại.

***4. Kiến thức cơ bản lập trình với CSDL: PHP & mySQL***

Những khác biệt chính giữa MySQL và SQL Server:

Môi trường, Syntax,

Câu lệnh **MySQL**

SELECT age

FROM person

ORDER BY age ASC

LIMIT 1 OFFSET 2

Câu lệnh **Microsoft SQL Server**

SELECT TOP 3 WITH TIES \*

FROM person

ORDER BY age ASC

, SQL Server còn hơn là một RDBMS, Storage engines, Hủy Query, Bảo mật, Chi phí, Công đồng hỗ trợ, IDEs

**Truy xuất CSDL bằng PHP**

define('HOST', 'localhost');

define('DATABASE', 'webbanhang');

define('USERNAME', 'root');

define('PASSWORD', '');

define('PRIVATE\_KEY', 'sjdgfsdj(\*&\*&6234jhsdgfjhsdsdfk&\*^UUUdd');

<?php

require\_once('config.php');

// insert, update, delete, select

// SQL: insert, update, delete

function execute($sql)

{

//open connection

$conn = mysqli\_connect(HOST, USERNAME, PASSWORD, DATABASE);

mysqli\_set\_charset($conn, 'utf8');

//query

mysqli\_query($conn, $sql);

//close connection

mysqli\_close($conn);

}

// SQL: select -> lay du lieu dau ra (select danh sach ban ghi, lay 1 ban ghi)

function executeResult($sql, $isSingle = false)

{

$data = null;

//open connection

$conn = mysqli\_connect(HOST, USERNAME, PASSWORD, DATABASE);

mysqli\_set\_charset($conn, 'utf8');

//query

$resultset = mysqli\_query($conn, $sql);

if ($isSingle) {

$data = mysqli\_fetch\_array($resultset, 1);

} else {

$data = [];

while (($row = mysqli\_fetch\_array($resultset, 1)) != null) {

$data[] = $row;

}

}

//close connection

mysqli\_close($conn);

return $data;

}

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "data";

// tạo kết nối

$conn = new mysqli($servername, $username, $password, $dbname);

// kiểm kết nối

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$sql = "SELECT id, firstname, lastname, email, reg\_date FROM users";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

// Load dữ liệu lên website

while($row = $result->fetch\_assoc()) {

echo "id: " . $row["id"]. " - Tên: " . $row["firstname"]. " "

. $row["lastname"]. " - Email: ". $row["email"]. " - Ngày đăng ký: ". $row["reg\_date"]."<br>";

}

} else {

echo "0 results";

}

$conn->close();

?>

**Kết hợp HTML Form và CSDL bằng PHP**

<form id="contact-form" method="post" class="form">

<input1type="1hidden"1name="\_token" value="cGijOmfAx2dlipWbpcszpLQwtDxOlPMd9xG4DUwn">

<?php

require\_once('layouts/header.php');

$user = getUserToken();

if (!empty($\_POST)) {

if ($user == null) {

echo "<script> window.location.href='admin/authen/login.php';</script>";

}

$first\_name = getPost('first\_name');

$last\_name = getPost('last\_name');

$email = getPost('email');

$phone\_number = getPost('phone');

$subject\_name = getPost('subject\_name');

$note = getPost('note');

$created\_at = $updated\_at = date('Y-m-d H:i:s');

$sql = "insert into FeedBack(firstname, lastname, email, phone\_number, subject\_name, note, status, created\_at, updated\_at) values('$first\_name', '$last\_name', '$email', '$phone\_number', '$subject\_name', '$note', 0, '$created\_at', '$updated\_at')";

execute($sql);

}

?>

<section class="hero">

<div class="container">

<ol class="breadcrumb justify-content-right">

<li class="breadcrumb-item"><a href="index.php">Trang Chủ</a></li>

<li class="breadcrumb-item active">Liên Hệ</li>

</ol>

</div>

</section>

<section style="background: #fafafa;" class="py-6">

<div class="container">

<div class="row">

<div class="col-md-4 text-center text-md-left">

<svg class="svg-icon svg-icon-light text-primary w-3rem h-3rem mb-3">

<use xlink:href="#navigation-map-1"> </use>

</svg>

<h4 class="ff-base">Địa Chỉ</h4>

<p class="text-muted">153/8 Đ. Nguyễn Thông, Phường 9, Quận 3, Thành phố Hồ Chí Minh, Việt Nam

</p>

</div>

<div class="col-md-4 text-center text-md-left">

<svg class="svg-icon svg-icon-light text-primary w-3rem h-3rem mb-3">

<use xlink:href="#audio-call-1"> </use>

</svg>

<h4 class="ff-base">Call</h4>

<p class="text-muted">Hãy gọi cho chúng tôi để nhận được phục vụ tốt nhất.</p>

<p class="text-muted"><strong>

<a href="tel:0932-525-463" class="phone\_number">0932.525.463</a>

</strong></p>

</div>

<div class="col-md-4 text-center text-md-left">

<svg class="svg-icon svg-icon-light text-primary w-3rem h-3rem mb-3">

<use xlink:href="#mail-1"> </use>

</svg>

<h4 class="ff-base">Email</h4>

<p class="text-muted">Gửi Email cho chúng tôi - để nhận chi tiết về dịch vu.</p>

<ul class="list-unstyled text-muted">

<li>

<a href="mailto:nguyenlongvux1@gmail.com" class="email">nguyenlongvux1@gmail.com</a>

</li>

</ul>

</div>

</div>

</div>

</section>

<section class="py-6">

<div class="container">

<header class="mb-5">

<h2 class="text-uppercase h5">Gửi Phản Hồi</h2>

</header>

<div class="row">

<div class="col-md-7 mb-5 mb-md-0">

<form id="contact-form" method="post" class="form">

<input type="hidden" name="\_token" value="cGijOmfAx2dlipWbpcszpLQwtDxOlPMd9xG4DUwn">

<div class="controls">

<div class="row">

<div class="col-sm-6">

<div class="form-group">

<label for="firstname" class="form-label">Tên \*</label>

<input type="text" name="first\_name" id="firstname" placeholder="Nhập tên" required="required" class="form-control">

</div>

</div>

<div class="col-sm-6">

<div class="form-group">

<label for="lastname" class="form-label">Họ \*</label>

<input type="text" name="last\_name" id="lastname" placeholder="Nhập họ" required="required" class="form-control">

</div>

</div>

</div>

<div class="form-group">

<label for="email" class="form-label">Địa Chỉ Email \*</label>

<input type="email" name="email" id="email" placeholder="Nhập địa chỉ email" required="required" class="form-control">

</div>

<div class="form-group">

<label for="phone\_number" class="form-label">Số Điện Thoại \*</label>

<input type="telno" name="phone" id="phone\_number" placeholder="Nhập số điện thoại" required="required" class="form-control">

</div>

<div class="form-group">

<label for="subject" class="form-label">Tiêu Đề \*</label>

<input type="text" name="subject\_name" id="subject" placeholder="Nhập tiêu đề" required="required" class="form-control">

</div>

<div class="form-group">

<label for="message" class="form-label">Nội Dung Tin Nhắn \*</label>

<textarea rows="4" name="note" id="message" placeholder="Nhập nội dung" required="required" class="form-control"></textarea>

</div>

<a href="checkout.php"><button class="btn btn-outline-dark">Gửi</button></a>

</div>

</form>

</div>

</div>

</div>

</section>

<?php

require\_once('layouts/footer.php');

?>

**Khái niệm cơ bản về SQL injection.**

SQL Injection là một kỹ thuật cho phép những kẻ tấn công lợi dụng lỗ hổng của việc kiểm tra dữ liệu đầu vào trong các ứng dụng web và các thông báo lỗi của hệ quản trị cơ sở dữ liệu trả về để inject (tiêm vào) và thi hành các câu lệnh SQL bất hợp pháp

SQL injection có thể cho phép những kẻ tấn công thực hiện các thao tác trên cơ sở dữ liệu của ứng dụng, thậm chí là server mà ứng dụng đó đang chạy.

**Tính nguy hiểm của tấn công SQL Injection**

Tùy vào mức độ tinh vi, SQL Injection có thể cho phép kẻ tấn công:

Vượt qua các khâu xác thực người dùng

Chèn, xóa hoặc sửa đổi dữ liệu

Đánh cắp các thông tin trong CSDL

Chiếm quyền điều khiển hệ thống

**Phân loại các kiểu tấn công SQL Injection**

SQL Injection có thể chia nhỏ thành các dạng sau

In-band SQLi

Error-based SQLi

Union-based SQLi

Inferential SQLi (Blind SQLi)

Blind-boolean-based SQLi

Time-based-blind SQLi

Out-of-band SQLi

**In-band SQLi**

Đây là dạng tấn công phổ biến nhất và cũng dễ để khai thác lỗ hổng SQL Injection nhất

Xảy ra khi hacker có thể tổ chức tấn công và thu thập kết quả trực tiếp trên cùng một kênh liên lạc

In-Band SQLi chia làm 2 loại chính:

Error-based SQLi

Union-based SQLi

**Error-based SQLi**

Là một kỹ thuật tấn công SQL Injection dựa vào thông báo lỗi được trả về từ Database Server có chứa thông tin về cấu trúc của cơ sở dữ liệu.

Trong một vài trường hợp, chỉ một mình Error-based là đủ cho hacker có thể liệt kê được các thuộc tính của cơ sở dữ liệu

**Union-based SQLi**

Là một kỹ thuật tấn công SQL Injection dựa vào sức mạnh của toán tử UNION trong ngôn ngữ SQL cho phép tổng hợp kết quả của 2 hay nhiều câu truy vấn SELECTION trong cùng 1 kết quả và được trả về như một phần của HTTP response

**Inferential SQLi (Blind SQLi)**

Không giống như In-band SQLi, Inferential SQL Injection tốn nhiều thời gian hơn cho việc tấn công do không có bất kì dữ liệu nào được thực sự trả về thông qua web application và hacker thì không thể theo dõi kết quả trực tiếp như kiểu tấn công In-band

Thay vào đó, kẻ tấn công sẽ cố gắng xây dựng lại cấu trúc cơ sở dữ liệu bằng việc gửi đi các payloads, dựa vào kết quả phản hồi của web application và kết quả hành vi của database server.

Có 2 dạng tấn công chính

Blind-boolean-based

Blind-time-based SQLi

**Blind-boolean-based**

Là kĩ thuật tấn công SQL Injection dựa vào việc gửi các truy vấn tới cơ sở dữ liệu bắt buộc ứng dụng trả về các kết quả khác nhau phụ thuộc vào câu truy vấn là True hay False.

Tuỳ thuộc kết quả trả về của câu truy vấn mà HTTP reponse có thể thay đổi, hoặc giữ nguyên

Kiểu tấn công này thường chậm (đặc biệt với cơ sở dữ liệu có kích thước lớn) do người tấn công cần phải liệt kê từng dữ liệu, hoặc mò từng kí tự

**Time-based Blind SQLi**

Time-base Blind SQLi là kĩ thuật tấn công dựa vào việc gửi những câu truy vấn tới cơ sở dữ liệu và buộc cơ sở dữ liệu phải chờ một khoảng thời gian (thường tính bằng giây) trước khi phản hồi.

Thời gian phản hồi (ngay lập tức hay trễ theo khoảng thời gian được set) cho phép kẻ tấn công suy đoán kết quả truy vấn là TRUE hay FALSE

Kiểu tấn công này cũng tốn nhiều thời gian tương tự như Boolean-based SQLi

**Out-of-band SQLi**

Out-of-band SQLi không phải dạng tấn công phổ biến, chủ yếu bởi vì nó phụ thuộc vào các tính năng được bật trên Database Server được sở dụng bởi Web Application.

Kiểu tấn công này xảy ra khi hacker không thể trực tiếp tấn công và thu thập kết quả trực tiếp trên cùng một kênh (In-band SQLi), và đặc biệt là việc phản hồi từ server là không ổn định