1. CƠ SỞ LÝ THUYẾT
   1. Giới thiệu dịch vụ WWW
      1. Web page là gì?

Một website có rất nhiều trang webb, mỗi trang web đó gọi là webpage. Một webpage có thể là một trang tin, một post bất kỳ.

* + 1. Website là gì?

Website còn gọi là một trang web, trang mạng, là một tập hợp các trang web bao gồm văn bản, hình ảnh, video,…, thường nằm trong một tên miền (domain name) hoặc tên miền phụ (subdomain). Trang web được lưu trữ (Web hosting) trên máy chủ web (server web) có thể truy cập thông tin qua Internet.

* + 1. Web Brower là gì?

Một web Brower là một phần mềm ứng dụng để truy xuất, trình diễn và chuyển các nguồn thông tin trên mạng hệ thống mạng toàn cầu. Một nguồn thông tin được nhận dang bởi một Unifrom Resouree Identifier (URI) và có thể là một trang web, phim, video , hình ảnh hoặc các mẫu tin khác.

* + 1. World Wide Web là gì?

World Wide Web, hay Web, thường gọi tắt là WWW, là không gian thông tin toàn cầu nơi con người có thể truy cập để đọc, viết thông tin qua các thiết bị kết nối mạng [Internet](https://vtc.vn/trao-luu-hoc-chung-qua-internet-giup-nhieu-nguoi-thi-do-tai-han-quoc-d458682.html).

Đây là một trong những ứng dụng Internet đầu tiên và không phải là Internet.

Nhà khoa học người Anh [Tim Berners-Lee](https://vi.wikipedia.org/wiki/Tim_Berners-Lee) được cho là đã phát minh ra World Wide Web khi làm việc cho [CERN](https://vi.wikipedia.org/wiki/CERN) vào tháng 3 năm 1989 bằng cách gửi Quản lý thông tin: Đề xuất và viết [trình duyệt web](https://vi.wikipedia.org/wiki/Tr%C3%ACnh_duy%E1%BB%87t_web) đầu tiên vào năm 1990. Trình duyệt được phát hành bên ngoài CERN năm 1991, lần đầu tiên cho các tổ chức nghiên cứu khác bắt đầu vào tháng 1 năm 1991 và công chúng trên Internet vào tháng 8 năm 1991. World Wide Web là trung tâm cho sự phát triển của [Thời đại Thông tin](https://vi.wikipedia.org/wiki/Th%E1%BB%9Di_%C4%91%E1%BA%A1i_Th%C3%B4ng_tin) và là công cụ chính mà hàng tỷ người sử dụng để tương tác trên Interne

* 1. Giới thiệu về HTML
     1. HTML là gì?
* HTML: HyperText Markup Language – Ngôn ngữ đánh dấu siêu văn bản. Do Tim Berne r Lee phát minh và được W3C (World Wide Web Consortium) đưa thành chuẩn năm 1994.
* HTML sử dụng các thẻ (tags) để định dạng dữ liệu và không phân biệt chữ hoa, chữ thường. Các trình duyệt thường không báo lỗi cú pháp HTML. Nếu viết sai cú pháp chỉ dẫn đến kết quả hiển thị không đúng với dự định.
  + 1. Lịch sử phát triển:
* 1989: HTML ra đời bởi Tim Berners-Lee dựa trên cơ sở của ngôn ngữ SGML (Standard Generalized Markup Language)
* 1990: HTML 1.0 được đưa vào sử dụng,
* 1994: HTML 2.0,
* 1996: HTML 3.2,
* 1999: HTML 4.01 (chuẩn phổ biến)
* 2000: XHTML 1.0 (chuẩn phổ biến)
* 2001: XHTML 1.1,
* 2002: XHTML 2.0,
* 2008: HTML 5 ra đời với nhiều tính năng nổi bật.
  + 1. Cấu trúc của một file HTML:

File HTML bao giờ cũng bắt đầu bằng thẻ <html> và kết thúc bằng thẻ </html>. Cặp thẻ này báo cho trình duyệt Web biết rằng nó đang đọc một file có chứa các mã HTML, còn thẻ </html> có tác dụng như kết thúc file HTML.

Bên trong cặp thẻ <html> ... </html> là các cặp thẻ <head> ... </head> và <body> ... </body> là phần thân, tại đây bạn có thể nhập vào các đoạn văn bản cùng các thẻ khác quy định về định dạng của trang.

Ngoài ra để ghi chú thích, tiện cho việc xem tag HTML hoặc cập nhật một trang Web, ta cần đặt chú thích vào giữa <!-- và -->.

*Ví dụ 1:*

<!DOCTYPE html>

<html>

<head>

<!-- Thông tin trang website -->

</head>

<body>

<!-- Nội dung website -->

</body>

</html>

* 1. Giới thiệu về CSS
     1. CSS là gì?
* CSS (Cascading Style Sheet) là mẫu quy định cách thức thể hiện các thẻ HTML. CSS được đưa vào HTML 4.0 để giải quyết một số vấn đề về cách thức hiển thị và định dạng dữ liệu. Giúp tiết kiệm được rất nhiều thời gian và công sức cho việc thiết kế web. Ta có thể định nghĩa nhiều style vào một thẻ HTML (Cascading).
  + 1. Phân loại Style: có 4 loại style
* Inline Style (Được quy định trong 1 thẻ HTML cụ thể),
* Internal Style (Được quy định trong thẻ <head> của trang HTML),
* External Style Được quy định trong file .CSS ngoài.
* Browser Default (Thiết lập mặc định của trình duyệt).
  + 1. Tác dụng của CSS

Hạn chế tối thiểu việc làm rối mã HTML của trang Web bằng các thẻ quy định kiểu dáng (chữ đậm, chữ in nghiêng, chữ có gạch chân, chữ màu), khiến mã nguồn của trang Web được gọn gàng hơn, tách nội dung của trang Web và định dạng hiển thị, dễ dàng cho việc cập nhật nội dung.

Tạo ra các kiểu dáng có thể áp dụng cho nhiều trang Web, giúp tránh phải lặp lại việc định dạng cho các trang Web giống nhau.

* + 1. Sử dụng CSS

Có 3 cách để sử dụng CSS.

"Inline CSS": Áp dụng trực tiếp trên một đối tượng nhất định bằng thuộc tính style:

<**span** style="font-weight:bold; text-decoration:underline; color:#FF0000;">Đoạn text cần in đậm, gạch chân, màu đỏ</**span**>

* "Internal CSS": Đặt CSS ở đầu trang Web để áp dụng kiểu dáng cho toàn bộ trang ấy, khi đó chỉ cần đặt đoạn CSS vào trong cặp thẻ <style> rồi đặt vào trong phần header của Web (giữa <head> và </head>):

<**style** type="text/css">

**body** {**font-family**:verdana; **color**:#0000FF;} */\* Kiểu chữ trong trang Web là "Verdana", màu chữ thông thường là màu xanh dương \*/*

</**style**>

* "External CSS": Đặt các thuộc tính CSS vào một tệp tin riêng biệt (\*.css), khi đó có thể tham chiếu đến từ nhiều trang Web khác nhau:
* Ví dụ về nội dung tệp *style.css*:

**body** {**font-family**:verdana; **color**:#0000FF;}

* Tham chiếu tới tệp tin CSS trên từ trang Web bằng đoạn mã (mã có thể nằm ngoài thẻ <head>):

<**link** rel="stylesheet" type="text/css" href="style.css"/>

* 1. Giới thiệu về Javascript
     1. Javascript là gì ?
  + Javascript là ngôn ngữ kịch bản dùng để tạo các client-side scripts và server-side scripts, làm cho việc tạo các trang Web động và tương tác dễ dàng hơn. Các ứng dụng client chạy trên một trình duyệt như Netscape Navigator hoặc Internet Explorer.
  + Java và JavaScript là hai ngôn ngữ hoàn toàn khác nhau: Java là ngôn ngữ lập trình hướng đối tượng được phát triển bởi hãng Sun Microsysttôis; JavaScript là ngôn ngữ kịch bản WEB được phát triển bởi Netscape.
    1. Khả năng của Javascript
  + JavaScript có thể tăng cường tính động và tính tương tác của các trang web. Cung cấp sự tương tác người dùng; Thay đổi nội dung động; Xác nhận tính hợp lệ của dữ liệu.
    1. Các thành phần cú pháp chính

***Khoảng trắng***

[Dấu cách](https://vi.wikipedia.org/w/index.php?title=D%E1%BA%A5u_c%C3%A1ch&action=edit&redlink=1), [tab](https://vi.wikipedia.org/wiki/Tab) và [ký tự dòng mới](https://vi.wikipedia.org/w/index.php?title=K%C3%BD_t%E1%BB%B1_d%C3%B2ng_m%E1%BB%9Bi&action=edit&redlink=1) sử dụng bên ngoài một [chuỗi ký tự](https://vi.wikipedia.org/w/index.php?title=Chu%E1%BB%97i_k%C3%BD_t%E1%BB%B1&action=edit&redlink=1) được gọi là [khoảng trắng](https://vi.wikipedia.org/w/index.php?title=Kho%E1%BA%A3ng_tr%E1%BA%AFng&action=edit&redlink=1). Khác với C, khoảng trắng trong JavaScript có thể ảnh hưởng trực tiếp tới ý nghĩa của câu lệnh. Sử dụng phương pháp "tự động thêm [dấu chấm phẩy](https://vi.wikipedia.org/wiki/D%E1%BA%A5u_ch%E1%BA%A5m_ph%E1%BA%A9y)", bất cứ một dòng JavaScript nào thích hợp sẽ được coi là một câu lệnh hợp lệ (giống như có dấu chấm phẩy trước ký tự dòng mới).

Tuy trong phần lớn trường hợp, dấu chấm phẩy trước khi kết thúc một dòng JavaScript là không cần thiết để đoạn [mã nguồn](https://vi.wikipedia.org/wiki/M%C3%A3_ngu%E1%BB%93n) hoạt động chính xác, [lập trình viên](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_vi%C3%AAn) nên sử dụng dấu chấm phẩy sau mỗi câu lệnh để đoạn mã nguồn dễ nhìn hơn. Ngoài ra, do đặc thù của JavaScript - chuyên dùng trên trang web, kích cỡ của đoạn mã nguồn là quan trọng, có một số [phần mềm](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m) có thể làm giảm kích cỡ của đoạn mã nguồn JavaScript bằng cách bỏ đi những khoảng trắng không cần thiết, để những phần mềm này hoạt động chính xác, lập trình viên cần thêm dấu chấm phẩy vào cuối mỗi câu lệnh.

***Chú thích***

Cú pháp [**chú thích**](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh#COMMENT) của JavaScript giống với [**C++**](https://vi.wikipedia.org/wiki/C%2B%2B). Lập trình viên có thể chú thích trên nhiều dòng bằng cách bao bọc chú thích với /\* và \*/ hoặc sử dụng // để chú thích từ vị trí // đến hết dòng.

*// Đây là chú thích trên một dòng.*

*/\**

*Đây là chú thích trên nhiều dòng.*

*Đây là chú thích 1*

*Đây là chú thích 2*

*\*/*

***Biến***

Trước khi sử dụng [**biến**](https://vi.wikipedia.org/wiki/Bi%E1%BA%BFn_s%E1%BB%91) trong JavaScript, lập trình viên không nhất thiết phải khai báo biến. Có 3 cách để định nghĩa biến trong JavaScript:

*// ECMAScript 5 trở về trước*

**var** tên\_biến

*// Từ ECMAScript 6*

**let** ten\_bien

**const** ten\_bien

***Toán tử***

Một [**toán tử**](https://vi.wikipedia.org/wiki/To%C3%A1n_t%E1%BB%AD) xác định phép toán sẽ được thực hiện trên các giá trị của các [**biến**](https://vi.wikipedia.org/wiki/Bi%E1%BA%BFn_s%E1%BB%91), và các biểu thức. Javascript cung cấp nhiều loại toán tử khác nhau để thực hiện việc tính toán, và đánh giá từ đơn giản đến phức tạp.

Các toán tử của Javascript được phân thành sáu thể loại dựa trên loại hành động của chúng thực hiện với các toán hạng. Bao gồm toán tử số học, toán tử quan hệ, toán tử luận lý, toán tử thao tác bit, toán tử gán, toán tử đặc biệt.

***Toán tử số học***

Các toán tử số học là các toán tử nhị phân, khi chúng thực hiện các phép tính cơ bản trên hai toán hạng. Toán tử xuất hiện ở giữa hai toán hạng, cho phép bạn thực hiện các phép tính với giá trị số và chuỗi. Các toán tử bao gồm: + (cộng), - (trừ), \* (nhân), / (chia), % (chia lấy dư).

Ví dụ:

**var** result;

result = 2 + 4; *// result = 6*

result = 2 - 4; *// result = -2*

result = 2 \* 4; *// result = 8*

result = 2 / 4; *// result = 0.5*

result = 2 % 4; *// result = 2*

***Toán tử tăng, và giảm***

Các toán tử tăng và giảm là các toán tử đơn hạng, vì chúng chỉ thực hiện được trên một toán hạng duy nhất. Toán tử tăng làm tăng giá trị lên 1, trong khi toán tử giảm làm giảm giá trị xuống 1, các toán tử có thể được đặt trước, hoặc sau toán hạng. Các toán tử bao gồm: ++ (tăng), -- (giảm).

Ví dụ:

**var** x = 2;

**var** y;

y = x++; *// x = 3, y = 2*

y = ++x; *// x = 3, y = 3*

y = x--; *// x = 1, y = 2*

y = --x; *// x = 1, y = 1*

***Toán tử quan hệ***

Toán tử quan hệ là các toán tử dùng để so sánh giữa hai toán hạng. Sau khi thực hiện một so sánh, chúng trả lại một giá trị true (đúng) hay false (sai). Các toán tử bao gồm: ==(bằng nhau), != (khác nhau), === (bằng nhau và cùng loại), !== (khác nhau và khác loại), > (lớn hơn), < (nhỏ hơn), >= (lớn hơn hoặc bằng), <= (nhỏ hơn hoặc bằng).

Ví dụ:

**var** result;

result = 3 == "3"; *// result = true*

result = 3 != 3; *// result = false*

result = 3 === "3"; *// result = false*

result = 3 !== "3"; *// result = true*

result = 3 > 4; *// result = false*

result = 3 < 4; *// result = true*

result = 3 >= 3; *// result = true*

result = 3 <= 4; *// result = true*

Toán tử luận lý

Các toán tử luận lý là các toán tử nhị phân thực hiện các phép toán logic trên hai toán hạng. Chúng thuộc loại toán tử quan hệ, vì chúng trả về một giá trị boolean. Các toán tử bao gồm: && (và), || (hoặc), ! (phủ định).

Ví dụ:

**var** x = 2, y = 5;

**var** result;

result = (x == 3) && (y == 5); *// result = false*

result = (x == 3) || (y == 5); *// result = true*

result = !(x == 3); *// result = true*

***Toán tử đặc biệt***

Toán tử điều kiện còn được biết đến với tên gọi [**toán tử tam phân**](https://vi.wikipedia.org/w/index.php?title=To%C3%A1n_t%E1%BB%AD_tam_ph%C3%A2n&action=edit&redlink=1). Cú pháp của toán tử này như sau:

điều\_kiện ? biểu\_thức\_đúng: biểu\_thức\_sai;

Toán tử này sẽ trả lại giá trị là kết quả của biểu\_thức\_đúng nếu điều\_kiện có giá trị bool bằng [**true**](https://vi.wikipedia.org/w/index.php?title=True&action=edit&redlink=1), ngược lại nó sẽ trả lại giá trị bằng biểu\_thức\_sai.

Câu lệnh điều khiển[[**sửa**](https://vi.wikipedia.org/w/index.php?title=JavaScript&veaction=edit&section=13) | [**sửa mã nguồn**](https://vi.wikipedia.org/w/index.php?title=JavaScript&action=edit&section=13)]

Câu lệnh if... else[[**sửa**](https://vi.wikipedia.org/w/index.php?title=JavaScript&veaction=edit&section=14) | [**sửa mã nguồn**](https://vi.wikipedia.org/w/index.php?title=JavaScript&action=edit&section=14)]

Cú pháp if... else dùng trong trường hợp muốn rẽ nhánh theo điều kiện. Cú pháp này tương đương với nếu x thì làm y, còn nếu không thì làm z. Các câu lệnh if... else có thể lồng trong nhau.

Cú pháp:

**if** (biểu\_thức\_1) {

khối lệnh được thực hiện nếu biểu thức 1 đúng;

}

**else** **if** (biểu\_thức\_2) {

khối lệnh được thực hiện nếu biểu thức 2 đúng;

}

**else** {

khối lệnh được thực hiện nếu cả hai biểu thức trên đều đúng;

}

Ví dụ:

**var** x = prompt("Nhập vào giá trị của x:");

x = parseFloat(x);

**if** (!isNaN(x)) {

**if** (x > 0) {

alert("x > 0");

}

**else** **if** (x == 0) {

alert("x = 0");

}

**else** {

alert("x < 0");

}

}

**else** {

alert("giá trị bạn nhập không phải là một số");

}

Đoạn mã nguồn trên mở một hộp thoại yêu cầu nhập vào một giá trị số, sau đó hiển thị thông báo số đó lớn hơn 0, bằng 0 hay nhỏ hơn 0.

***Câu lệnh switch... case***

Cú pháp *switch*cũng là cú pháp điều kiện như if... else hay toán tử tam phân. Tuy nhiên, cú pháp switch thường được dùng khi chỉ cần so sánh bằng với số lượng kết quả cần kiểm tra lớn. Cách sử dụng cú pháp *switch*:

**switch** (biểu\_thức\_điều\_kiện) {

**case** kết\_quả\_1:

khối lệnh cần thực hiện nếu biểu\_thức\_điều\_kiện bằng kết\_quả\_1;

**break**;

**case** kết\_quả\_2:

khối lệnh cần thực hiện nếu biểu\_thức\_điều\_kiện bằng kết\_quả\_2;

**break**;

**default**:

khối lệnh cần thực hiện nếu biểu\_thức\_điều\_kiện cho ra một kết quả khác;

**break**;

}

Sau mỗi khối lệnh trong một mục kiểm tra kết quả (trừ mục default), lập trình viên cần phải thêm vào break.

***Vòng lặp***

***Vòng lặp while***

[**Vòng lặp**](https://vi.wikipedia.org/w/index.php?title=V%C3%B2ng_l%E1%BA%B7p&action=edit&redlink=1) while có mục đích lặp đi lặp lại một khối lệnh nhất định cho đến khi biểu thức điều kiện trả về [**false**](https://vi.wikipedia.org/w/index.php?title=False&action=edit&redlink=1). Khi dùng vòng lặp while phải chú ý tạo lối thoát cho vòng lặp (làm cho biểu thức điều kiện có giá trị false), nếu không đoạn mã nguồn sẽ rơi vào [**vòng lặp vô hạn**](https://vi.wikipedia.org/w/index.php?title=V%C3%B2ng_l%E1%BA%B7p_v%C3%B4_h%E1%BA%A1n&action=edit&redlink=1), là một [**lỗi lập trình**](https://vi.wikipedia.org/w/index.php?title=L%E1%BB%97i_l%E1%BA%ADp_tr%C3%ACnh&action=edit&redlink=1). Vòng lặp while thường được dùng khi lập trình viên không biết chính xác cần lặp bao nhiêu lần. Cú pháp của vòng lặp while như sau:

**while** (biểu\_thức\_điều\_kiện) {

khối lệnh cần thực hiện nếu biểu\_thức\_điều\_kiện trả về **true**;

}

***Vòng lặp do... while***

Về cơ bản, vòng lặp do... while gần như giống hệt như vòng lặp while. Tuy nhiên, trong trường hợp biểu thức điều kiện trả về false ngay từ đầu, khối lệnh trong vòng lặp whilesẽ không bao giờ được thực hiện, trong khi đó, vòng lặp do... while luôn đảm bảo khối lệnh trong vòng lặp được thực hiện ít nhất một lần. Ví dụ:

**do** {

alert("do... while"); *// Bạn sẽ nhận được thông báo "do... while" một lần duy nhất*

} **while** (0 > 1);

Cú pháp của vòng lặp *do... while* như sau:

**do** {

khối lệnh;

}

**while** (biểu\_thức\_điều\_kiện);

***Vòng lặp for***

Vòng lặp *for* thường được sử dụng khi cần lặp một khối lệnh mà lập trình viên biết trước sẽ cần lặp bao nhiêu lần. Cú pháp của vòng lặp *for* như sau:

**for** (biểu\_thức\_khởi\_tạo; biểu\_thức\_điều\_kiện; biểu\_thức\_thay\_đổi\_giá\_trị) {

Khối lệnh cần lặp;

}

Khi bắt đầu vòng lặp *for*, lập trình viên cần khởi tạo một biến nhất định bằng biểu\_thức\_khởi\_tạo để dùng trong biểu\_thức\_điều\_kiện, nếu biểu\_thức\_điều\_kiện trả về true, khối lệnh cần lặp sẽ được thực hiện, sau khi thực hiện xong khối lệnh cần lặp, biểu\_thức\_thay\_đổi\_giá\_trị sẽ được thực hiện, tiếp theo, biểu\_thức\_điều\_kiện sẽ lại được kiểm tra, cứ như vậy cho đến khi biểu\_thức\_điều\_kiện trả về false, khi đó vòng lặp sẽ kết thúc.

***Hàm***

Hàm là một khối các câu lệnh với một danh sách một hoặc nhiều [**đối số**](https://vi.wikipedia.org/wiki/Tham_s%E1%BB%91_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)) (có thể không có đối số) và thường có tên (mặc dù trong JavaScript hàm không nhất thiết phải có tên). Hàm có thể trả lại một giá trị. Cú pháp của hàm như sau:

*//ECMAScript 5 trở về trước*

**function** tên\_hàm(đối\_số\_1, đối\_số\_2) {

*//câu lệnh*

}

*//ECMAScript 6 trở đi*

**const** tên\_hàm = (đối\_số\_1, đối\_số\_2)=>{

*//câu lệnh*

}

*//Thực thi*

tên\_hàm(1, 2); *// Gọi hàm tên\_hàm với hai đối số 1 và 2 ứng với đối\_số\_1 và đối\_số\_2*

tên\_hàm(1); *// Gọi hàm tên\_hàm với đối\_số\_1 có giá trị 1, đối\_số\_2 có giá trị undefined*

Trong JavaScript, khi gọi hàm không nhất thiết phải gọi hàm với cùng số đối số như khi định nghĩa hàm, nếu số đối số ít hơn khi định nghĩa hàm, những đối số không được chuyển cho hàm sẽ mang giá trị undefined.

Các kiểu cơ bản sẽ được chuyển vào hàm [**theo giá trị**](https://vi.wikipedia.org/w/index.php?title=Theo_gi%C3%A1_tr%E1%BB%8B&action=edit&redlink=1), đối tượng sẽ được chuyển vào hàm [**theo tham chiếu**](https://vi.wikipedia.org/w/index.php?title=Theo_tham_chi%E1%BA%BFu&action=edit&redlink=1).

Hàm là [**đối tượng hạng nhất**](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BB%91i_t%C6%B0%E1%BB%A3ng_h%E1%BA%A1ng_nh%E1%BA%A5t&action=edit&redlink=1) trong JavaScript. Tất cả các hàm là đối tượng của nguyên mẫu Function. Hàm có thể được tạo và dùng trong phép toán gán như bất kỳ một đối tượng nào khác, và cũng có thể được dùng làm đối số cho các hàm khác. Do đó, JavaScript hỗ trợ [**hàm cấp độ cao**](https://vi.wikipedia.org/w/index.php?title=H%C3%A0m_c%E1%BA%A5p_%C4%91%E1%BB%99_cao&action=edit&redlink=1). Ví dụ:

Array.prototype.fold =

**function** (value, functor) {

**var** result = value;

**for** (**var** i = 0; i < **this**.length; i++) {

result = functor(result, **this**[i]);

}

**return** result;

}

**var** sum = [1,2,3,4,5,6,7,8,9,10].fold(0, **function** (a, b) { **return** a + b; });

Đoạn mã nguồn trên sẽ trả lại kết quả là 55.