**BÀI TẬP VỀ NHÀ WEB (B1)**

***Câu 1:***

\*) Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS), nó là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. Git cung cấp cho mỗi lập trình viên kho lưu trữ (repository) riêng chứa toàn bộ lịch sử thay đổi.

\*) - **Git** là một hệ thống kiểm soát sửa đổi, một công cụ để quản lý lịch sử mã nguồn của bạn.

**- GitHub** là một dịch vụ lưu trữ cho kho Git.

=> Vì vậy, chúng không giống nhau: **Git** là **công cụ** , **GitHub** là **dịch vụ cho các dự án sử dụng Git** .

## \*) Các lệnh git cơ bản:

**1) git config**

Tác dụng : Để set user name và email của bạn trong main configuration file.  
Cách xài : Để kiểm tra tên và kiểu email trong cấu hình dùng ***git config -- global user.name*** và ***git config -- global user.email***. Để set email hoặc tên mới ***git config -- global user.name = “Hải Nguyễn”*** và ***git config -- global user.email = “hainguyen@gmail.com”***

**2) git init**

Tác dụng : Khởi tạo 1 git repository 1 project mới hoặc đã có.

Cách xài: ***git init*** trong thư mục gốc của dự án.

**3) git clone**

Tác dụng: Copy 1 git repository từ remote source.

Cách xài: ***git clone <:clone git url:>***

**4) git status**

Tác dụng: Để check trạng thái của những file bạn đã thay đổi trong thư mục làm việc. VD: Tất cả các thay đổi cuối cùng từ lần commit cuối cùng.

Cách xài: ***git status*** trong thư mục làm việc.

**5) git add**

Tác dụng: Thêm thay đổi đến stage/index trong thư mục làm việc.

Cách xài: ***git add***

**6) git commit**

Tác dụng: commit nghĩa là một action để Git lưu lại một snapshot của các sự thay đổi trong thư mục làm việc. Và các tập tin, thư mục được thay đổi đã phải nằm trong Staging Area. Mỗi lần commit nó sẽ được lưu lại lịch sử chỉnh sửa của code kèm theo tên và địa chỉ email của người commit. Ngoài ra trong Git bạn cũng có thể khôi phục lại tập tin trong lịch sử commit của nó để chia cho một branch khác, vì vậy bạn sẽ dễ dàng khôi phục lại các thay đổi trước đó.

Cách dùng: ***git commit -m ”Đây là message, bạn dùng để note những thay đổi để sau này dễ dò lại”***

**7) git push/git pull**

Tác dụng: Push hoặc Pull các thay đổi đến remote. Nếu bạn đã added và committed các thay đổi và bạn muốn đẩy nó lên hoặc remote của bạn đã update và bạn apply tất cả thay đổi đó trên code của mình.

Cách dùng: ***git pull <:remote:> <:branch:>*** and ***git push <:remote:> <:branch:>***

**8) git branch**

Tác dụng: liệt kê tất cả các branch (nhánh).

Cách dùng: ***git branch*** hoặc ***git branch -a***

**9) git checkout**

Tác dụng: Chuyển sang branch khác

Cách dùng: ***git checkout <: branch:>*** hoặc ***\*\* \_ git checkout -b <: branch:>*** nếu bạn muốn tạo và chuyển sang một chi nhánh mới.

**10) git stash**

Tác dụng: Lưu thay đổi mà bạn không muốn commit ngay lập tức.

Cách dùng: ***git stash*** trong thư mục làm việc của bạn.

**11) git merge**

Tác dụng: Merge 2 branch lại với nahu.

Cách dùng: Chuyển tới branch bạn muốn merge rồi  dùng ***git merge <:branch\_ban\_muon\_merge:>***

**12) git reset**

Tác dụng: Bạn đã đưa một tập tin nào đó vào Staging Area nhưng bây giờ bạn muốn loại bỏ nó ra khỏi đây để không phải bị commit theo.

Cách dùng: ***git reset HEAD tên\_file***

**13) git remote**

Tác dụng: Để check remote/source bạn có hoặc add thêm remote

Cách dùng: ***git remote*** để kiểm tra và liệt kê. Và ***git remote add <: remote\_url:>*** để thêm.

**14) git add**

Tác dụng: Để đưa một tập tin vào Staging Area

Cách dùng: ***git add tên\_file*** hoặc muốn thêm hết file của thư mục thì ***git add all***

***Câu 2:***

\*) Phần front-end của một trang web là phần tương tác với người dùng. Tất cả mọi thứ bạn nhìn thấy khi điều hướng trên Internet, từ các font chữ, màu sắc cho tới các menu xổ xuống và các thanh trượt, là một sự kết hợp của HTML, CSS, và JavaScript được điều khiển bởi trình duyệt máy tính của bạn.

\*) IDE viết tắt của cụm từ Integrated Development Environment là phần mềm cung cấp cho các lập trình viên một môi trường tích hợp bao gồm nhiều công cụ khác nhau như chương trình viết mã lệnh hay code editor, chương trình sửa lỗi hay debugger, chương trình mô phỏng ứng dụng khi chạy thực tế hay simulator.... Nói cách khác thì IDE là một phần mềm bao gồm những gói phần mềm khác giúp phát triển ứng dụng phần mềm.

\*) **Framework**là các đoạn code đã được viết sẵn, cấu thành nên một bộ khung và các thư viện lập trình được đóng gói. Chúng cung cấp các tính năng có sẵn như mô hình, API và các yếu tố khác để tối giản cho việc phát triển các ứng dụng web phong phú, năng động. Các **framework** giống như là chúng ta có khung nhà được làm sẵn nền móng cơ bản, bạn chỉ cần vào xây dựng và nội thất theo ý mình.

### *\*)****JQuery***

Là thư viện mã nguồn mở viết bằng ngôn ngữ javascript, giúp đơn giản cách viết javascript và tăng tốc độ xử lý các xự kiện trên trang web. Phát hành vào tháng 1 năm 2006 tại BarCamp NYC bởi John Resig. Được sử dụng bởi hơn 52% trong 10.000 truy cập nhiều nhất các trang web. Jquery không những có thể thay đổi giao diện(CSS) mà còn có thể thay đổi nội dung trang bên trong trang web. Vậy jquery có những ưu và nhược điểm gì?

\*) ***Bootstrap là một framework bao gồm các HTML, CSS và JavaScript template dùng để phát triển website chuẩn responsive***. **Bootstrap** cho phép quá trình thiết kế website diễn ra nhanh chóng và dễ dàng hơn dựa trên những thành tố cơ bản sẵn có như typography, forms, buttons, tables, grids, navigation, image carousels… Cùng **Mắt Bão** tìm hiểu tính năng và lợi ích mang lại cho lập trình viên của ***Bootstrap là gì*** nhé!

***Bootstrap*** là một bộ sưu tập miễn phí của các ***mã nguồn mở*** và công cụ dùng để tạo ra một mẫu webiste hoàn chỉnh. Với các thuộc tính về giao diện được quy định sẵn như kích thước, màu sắc, độ cao, độ rộng…, các **designer** có thể sáng tạo nhiều sản phẩm mới mẻ nhưng vẫn tiết kiệm thời gian khi làm việc với **framework** này trong quá trình ***thiết kế giao diện website***.

***Câu 3:***

## Một số tính năng của Visual Studio

* **Hỗ trợ nhiều ngôn ngữ lập trình**

Visual Studio Code hỗ trợ nhiều ngôn ngữ lập trình như C/C++, C#, F#, Visual Basic, HTML, CSS, JavaScript, … Vì vậy, nó dễ dàng phát hiện và đưa ra thông báo nếu chương chương trình có lỗi.

* **Hỗ trợ đa nền tảng**

Các trình viết code thông thường chỉ được sử dụng hoặc cho Windows hoặc Linux hoặc Mac Systems. Nhưng Visual Studio Code có thể hoạt động tốt trên cả ba nền tảng trên.

* **Cung cấp kho tiện ích mở rộng**

Trong trường hợp lập trình viên muốn sử dụng một ngôn ngữ lập trình không nằm trong số các ngôn ngữ Visual Studio hỗ trợ, họ có thể tải xuống tiện ích mở rộng. Điều này vẫn sẽ không làm giảm hiệu năng của phần mềm, bởi vì phần mở rộng này hoạt động như một chương trình độc lập.

* **Kho lưu trữ an toàn**

Đi kèm với sự phát triển của lập trình là nhu cầu về lưu trữ an toàn. Với Visual Studio Code, người dùng có thể hoàn toàn yên tâm vì nó dễ dàng kết nối với Git hoặc bất kỳ kho lưu trữ hiện có nào.

* **Hỗ trợ web**

Visual Studio Code hỗ trợ nhiều ứng dụng web. Ngoài ra, nó cũng có một trình soạn thảo và thiết kế website.

* **Lưu trữ dữ liệu dạng phân cấp**

Phần lớn tệp lưu trữ đoạn mã đều được đặt trong các thư mục tương tự nhau. Ngoài ra, Visual Studio Code còn cung cấp các thư mục cho một số tệp đặc biệt quan trọng.

* **Hỗ trợ viết Code**

Một số đoạn code có thể thay đổi chút ít để thuận tiện cho người dùng. Visual Studio Code sẽ đề xuất cho lập trình viên các tùy chọn thay thế nếu có.

* **Hỗ trợ thiết bị đầu cuối**

Visual Studio Code có tích hợp thiết bị đầu cuối, giúp người dùng khỏi phải chuyển đổi giữa hai màn hình hoặc trở về thư mục gốc khi thực hiện các thao tác.

* **Màn hình đa nhiệm**

Người dùng Visual Studio Code có thể mở cùng lúc nhiều tệp tin và thư mục – mặc dù chúng không hề liên quan với nhau.

* **Intellisense**

Hầu hết các trình viết mã đều có tính năng nhắc mã Intellisense, nhưng ít chương trình nào chuyên nghiệp bằng Visual Studio Code. Nó có thể phát hiện nếu bất kỳ đoạn mã nào không đầy đủ. Thậm chí, khi lập trình viên quên không khai báo biến, Intellisense sẽ tự động giúp họ bổ sung các cú pháp còn thiếu.

* **Hỗ trợ Git**

Visual Studio Code hỗ trợ kéo hoặc sao chép mã trực tiếp từ GitHub. Mã này sau đó có thể được thay đổi và lưu lại trên phần mềm.

* **Bình luận**

Việc để lại nhận xét giúp người dùng dễ dàng nhớ công việc cần hoàn thành.

***Câu 4:***

## \*) Thẻ <meta /> là gì?

Thẻ HTML <html> hay thẻ <meta> trong HTML được sử dụng để cung cấp metadata về tài liệu HTML. Metadata sẽ không được hiển thị trên trình duyệt, nhưng nó sẽ được bộ máy tìm kiếm phân tích để lấy thông tin cơ bản về trang web của bạn. Nói cách khác là nó dùng để cung cấp thêm "thông tin về trang web" cho trình duyệt và các công cụ tìm kiếm, những thông tin này không hiển thị lên màn hình cho người dùng nhìn thấy, tuy nhiên trình duyệt và các công cụ tìm kiếm có thể đọc và hiểu được, thường được đặt dưới thẻ <head> của một trang web..

Trước đây một số nhà phát triển website còn cho rằng thẻ meta có thể được đặt trong body của một trang, vậy trường hợp nào để trong <head> và trường hợp nào để trong <body>. Đối với các thẻ meta định nghĩa trang web thì phải để trong <head>. Ví dụ như: meta description. Nhưng trong một số trường hợp bạn cần đánh dấu văn bản trong trang web. Thì bạn có thể sử dụng các thẻ meta trong <body>.

Vì vậy: Các thẻ meta được sử dụng để định nghĩa cho cả trang web thì đặt ở trong <head>. Một số thẻ meta được dùng để đánh dấu văn bản cho một phần nội dung trang web thì có thể đặt trong <body>.

## \*) DOCTYPE Là Gì

HTML có nhiều phiên bản khác nhau như HTML5, HTML 4, HTML 3, XHTML... Khai báo <DOCTYPE> được các trình duyệt sử dụng để xác định loại văn bản HTML.

* Khai báo <DOCTYPE> phải được đặt trước tất cả các thẻ khác.
* Khai báo <DOCTYPE> không phải là thẻ.

Hiện tại HTML5 là phiên bản mới nhất và đang được sử dụng phổ biến trên hầu hết các website.

### HTML5 DOCTYPE

Với HTML phiên bản 5 (hay HTML5, lưu ý số 5 viết liền với HTML) khai báo <DOCTYPE> sẽ như sau:

**<!DOCTYPE html>**

\*) **HTML** (viết tắt của từ **Hypertext Markup Language**, hay là "Ngôn ngữ Đánh dấu Siêu văn bản") là một [ngôn ngữ đánh dấu](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_%C4%91%C3%A1nh_d%E1%BA%A5u" \o "Ngôn ngữ đánh dấu) được thiết kế ra để tạo nên các [trang web](https://vi.wikipedia.org/wiki/Website" \o "Website) trên [World Wide Web](https://vi.wikipedia.org/wiki/World_Wide_Web). Cùng với [CSS](https://vi.wikipedia.org/wiki/CSS) và [JavaScript](https://vi.wikipedia.org/wiki/JavaScript), HTML là một trong những ngôn ngữ quan trọng trong lĩnh vực thiết kế website. HTML được định nghĩa như là một ứng dụng đơn giản của [SGML](https://vi.wikipedia.org/wiki/SGML) và được sử dụng trong các tổ chức cần đến các yêu cầu xuất bản phức tạp. HTML đã trở thành một chuẩn mực của [Internet](https://vi.wikipedia.org/wiki/Internet) do tổ chức [World Wide Web Consortium](https://vi.wikipedia.org/wiki/W3C) (W3C) duy trì. Phiên bản chính thức mới nhất của HTML là HTML 4.01 (1999). Sau đó, các nhà phát triển đã thay thế nó bằng [XHTML](https://vi.wikipedia.org/wiki/XHTML). Hiện nay, phiên bản mới nhất của ngôn ngữ này là [HTML5](https://vi.wikipedia.org/wiki/HTML5).

Lưu ý: HTML không phải là ngôn ngữ lập trình.

\*) Trong [tin học](https://vi.wikipedia.org/wiki/Tin_h%E1%BB%8Dc), các **tập tin định kiểu theo tầng** – dịch từ [tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh" \o "Tiếng Anh) là **Cascading Style Sheets**(**CSS**) – được dùng để miêu tả cách trình bày các tài liệu viết bằng ngôn ngữ [HTML](https://vi.wikipedia.org/wiki/HTML) và [XHTML](https://vi.wikipedia.org/wiki/XHTML).[[1]](https://vi.wikipedia.org/wiki/CSS#cite_note-1) Ngoài ra ngôn ngữ định kiểu theo tầng cũng có thể dùng cho [XML](https://vi.wikipedia.org/wiki/XML), [SVG](https://vi.wikipedia.org/wiki/SVG), [XUL](https://vi.wikipedia.org/w/index.php?title=XUL&action=edit&redlink=1). Các đặc điểm kỹ thuật của CSS được duy trì bởi [World Wide Web Consortium](https://vi.wikipedia.org/wiki/W3C) ([W3C](http://www.w3c.org/)). Thay vì đặt các thẻ quy định kiểu dáng cho văn bản [HTML](https://vi.wikipedia.org/wiki/HTML) (hoặc [XHTML](https://vi.wikipedia.org/wiki/XHTML)) ngay trong [nội dung](https://vi.wikipedia.org/w/index.php?title=N%E1%BB%99i_dung&action=edit&redlink=1" \o "Nội dung (trang chưa được viết)) của nó, bạn nên sử dụng CSS.

\*)**JavaScript** là một ngôn ngữ lập trình của HTML và WEB. Nó là nhẹ và được sử dụng phổ biến nhất như là một phần của các trang web, mà sự thi hành của chúng cho phép Client-Side script tương tác với người sử dụng và tạo các trang web động. Nó là một ngôn ngữ chương trình thông dịch với các khả năng hướng đối tượng.

***CÁCH KHAI BÁO CSS:***

1. file css (.css) / file js (.js)

2. <style> </style> (html)

<script> </script>

***VD:***

1.p{

pr name: pr value;

}

2. .clasA{

}

3. #id{

}

4. [attri = ‘attr value”]

\*

~

+

>

-

**Kiểm tra các property mà các bạn đã học!**

**Độ ưu tiên của các khai báo css**

12/4/2021

**I. BOOTSTRAP**

- là thư viện của css

- sinh ra để k phải code css nhiều (1 dạng framework)

- Cung cấp các class đã Css sẵn

- Cung cấp các attribute để tạo nên responsive

- tuân theo layout (mặc định chia làm 12 cột(trên mọi thiết bị) qua class "col")

- Chia theo flex và %

- các tiền tố:

xs(mobile) sm(ipad) md(ipad pro) lg(laptop) xl(projector)

.col-xs-12

? col-offset (bù cột)

- Cách viết bootstrap:

+ container -> row -> col (Nếu dàn trải thì dùng container-fluid)

+ Khi khai báo file bootstrap:

. khai báo file bootstrap trong file css

**II. RESPONSIVE** (Chia màn hình tỉ lệ 1:1 để ảnh/nội dung tương thích vs thiết bị đang dùng)

- cách vết: @media (min/max-width:kích cỡ màn hình)

vd: @media only screen and (min-width:700px) {}

700px là con số đc quy định riêng

- Cách responsive hoạt động: giống cách css hoạt động, hoạt động trên màn hình nào thì lấy css của màn hình đó đặt xuống

dưới cùng để nó hiện lên (độ ưu tiên, ưu tiên thằng phía sau)

- Cái nào thay đổi theo màn hình thì mới để vào @media

- Nếu 2 màn hình có cùng cách chia css thì chia cho màn hình nhỏ nhất.

- Set chiều cao full màn hình (100vh) (k có scroll)

dùng thuộc tính height: height calc(100vh)

BTVN: Responsive Design Web (buổi trước)

***\*) Animation***

1. @keyframe

- from - to; 0% - 50% - 100%...

vd: @keyframe jump{

from{

back-ground: red;

transform: scale(1);

}

to{

back-ground: red;

transform: scale(1);

}

}

2. - transition

- transforms

***Buổi 8: (28/4/2021)***

**JAVASCRIPTS**

1. innerText: chỉ in ra Text
2. innerHTML: in ra 1 đoạn HTML
3. Nháy đơn và kép đều hợp lệ (Trong JS)

Nếu dung cả 2 dấu thì nháy đơn trong nháy đôi

1. Biến toàn cục và cục bộ (var & scope)

***Buổi 7:***

***JavaScript***

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**1. Cú pháp:**

.<script> </script>

.js (lưu)

- Luôn khai báo ở dưới cùng.

? Tìm hiểu

+) Kiểu dữ liệu (Str): string, number, null, underfine, boolean, symbol, objects...

+) Khai báo biến (Declare):

var, let, const

- var vs let: + giống: đều khai báo biến

+ khác: phạm vi

- var,let vs const: biến vs số

. const được sử dụng để khai báo 1 hằng số, và giá trị của nó không thay đổi trong suốt chương trình.

. let khai báo biến chỉ có thể truy cập được trong block bao quanh nó được xác định bằng cặp {}.

. var khai báo biến có thể truy cập ở phạm vi hàm số hoặc bên ngoài hàm số, toàn cục.

+) Vòng lặp (Loop)

1. Vòng lặp for

.Cú pháp

let i = 0 : khởi tạo biến cho vòng lặp

i < 10 : điều kiện để vòng lặp thực hiện

i++ : tăng giá trị biến chạy lên 1 mỗi khi thực hiện xong hành động

Bạn có thể bỏ trống giá trị ban đầu trong cú pháp của vòng lặp for nếu trước đó đã gán giá trị của biến chạy

Bạn cũng có thể bỏ trống giá trị thứ hai trong cú pháp của vòng lặp for. Lúc này, nếu giá trị thứ hai trả về giá trị true thì vòng lặp tiếp tục thực thi, nếu là false thì vòng lặp sẽ dừng lại.

Nếu bỏ trống giá trị thứ hai thì bắt buộc trong vòng lặp phải có lệnh break, nếu không thì vòng lặp sẽ chạy mãi không dừng lại.

Giá trị thứ ba trong cú pháp vòng lặp for cũng không thực sự cần thiết nếu như bạn thay đổi giá trị của biến chạy trong vòng lặp.

Tuy nhiên, việc bỏ trống các giá trị trong cú pháp của vòng lặp for là không nên. Dù ngắn gọn nhưng nếu dùng không đúng lúc sẽ khiến code của chúng ta khó đọc lại.

Đoạn code trên là ví dụ của vòng lặp for trong Javascript, rất dễ sử dụng phải không nào. Nhưng cú pháp này khá dài, liệu có cách nào rút gọn hơn không ?

Câu trả lời là có. Như ở đầu bài mình có đề cập, chúng ta sẽ sử dụng for...of và for...in

2. Vòng lặp for...of

Vòng lặp này được ra mắt ở phiên bản ES6. Tương tự như for, vòng lặp này được sử dụng để duyệt từng phần tử của đối tượng duyệt. Số lượng lặp bằng với số phần tử của đối tượng.

let student: khai báo biến chạy

students: một Array, String, Map, WeakMap, Set (không có Object)

Và kết quả cũng giống như chúng ta sử dụng vòng lặp for nhưng có thể thấy cú pháp của for...of tường minh, dễ hiểu hơn là for.

Tuy nhiên for...of không duyệt theo kiểu giảm dần giống như trong for được.

\*for...of

Vòng lắp được ra mắt trong phiên bản ES6. Hàm này có thể sử dụng để duyệt phần lớn các đối tượng từ Array, String, Map, WeakMap, Set ,...

3. Vòng lặp for...in

Vòng lặp này thường được sử dụng với mục đích là lặp trong một object chứ không phải trong array hay string giống như hai vòng lặp trên. Số lượng lặp tương ứng với số thuộc tính của object mà ta duyệt.

let key: khai báo biến chạy

obj: đối tượng duyệt (thường là object)

for...in cũng có thể dùng với array, lúc này các thuộc tính của mỗi phần từ trong array chính là index.

\* For...in

Vòng lặp này có đôi chút khác biệt với các hàm phía trên, For ... in mục đích chủ yếu được dùng để loop trong một object chứ không phải array . Số lượng vòng lặp sẽ tương ứng với số lượng thuộc tính của object

Mỗi array cũng là một object đặc biệt, do đó ta vẫn có thể sử đụng for...in cho array, tuy nhiên key sẽ tương ứng với giá trị index của từng phần tử

4. For loop

Đầu tiên hẳn sẽ là loop phổ biến nhất và có phần lớn trong các ngôn ngữ lập trình : for

Với vòng lặp for ta sẽ khởi tạo biến đếm, kiểm tra điều kiện và tăng hoặc giảm biến được thực hiện trên cùng một dòng, do đó khá dễ dàng cho những người mới tiếp cận để debug và cũng giảm khả năng sinh ra lỗi

5. Do ... While

\*\*do-while\*\* về cơ bản khá giống với \*\*while\*\*, chúng chỉ khác nhau duy nhất.

Đối với \*\*Do While\*\* dù điều kiện lặp như thế nào thì đoạn code vẫn được chạy ít nhất 1 lần còn nếu điều kiện thỏa mãn thì sẽ tương tự như \*\*While\*\* : tạo ra thêm vòng lặp

6. forEach()

Hàm này thì có vẻ đã không quá còn basic như những hàm phía trên nữa.

forEach sẽ lặp lại từng phần tử trong mảng theo thứ tự index và thực thi function được truyền vào. Lưu ý rằng forEach sẽ không thực thi function đầu vào cho các phần tử không có giá trị

+) Mảng (Array)

Mảng JavaScript

Mảng Array là một đối tượng định nghĩa sẵn trong JS, nó là đối tượng chứa danh sách các dữ liệu. Có hai cách tạo ra đối tượng mảng, sử dụng toán tử new và khởi tạo bằng ký hiệu []

Cách 1, tạo mảng với new Array(), ví dụ:

var courses = new Array("HTML", "CSS", "JS");

Cách 2, tạo mảng với khai báo dữ liệu trong []

var courses = ["HTML", "CSS", "JS"];

Mỗi phần từ có một chỉ số trong mảng, phần tử thứ nhất chỉ số là 0. Để truy cập tới phần tử mảng dùng ký hiệu tên-mảng[chỉ-số-phần-tử]

var cacmonhoc = new Array("HTML", "CSS", "JS");

var monhoc = courses[0]; // HTML

cacmonhoc[1] = "C++"; //đổi phần tử thứ 2 thành C++

Khi cố gắng truy cập phần tử mảng, mà phần tử đó không tồn tại thì nó trả về giá trị bằng undefined

var courses = new Array("HTML", "CSS", "JS");

document.write(courses[10]);

//Xuất ra "undefined" vì phần tử thứ 10 không có

Tạo mảng bằng cách chỉ định số lượng phần tử

Ở cách tạo thứ 2 này, các phần tử không được gán giá trị ngay, mảng sẽ tạo ra số lượng phần tử yêu cầu sau đó giá trị từng phần tử được gán sau.

var courses = new Array(3);

courses[0] = "HTML";

courses[1] = "CSS";

courses[2] = "JS";

Một số thuộc tính và phương thức

Một số thuộc tính và phương thức định nghĩa sẵn cho các đối tượng mảng

Số lượng phần tử trong mảng

Thuộc tính length cho biết số lượng phần tử trong mảng

var courses = ["HTML", "CSS", "JS"];

document.write(courses.length);

//Outputs 3

Kết hợp 2 mảng

Để kết hợp 2 mảng thành một mảng, dùng phương thức concat

var c1 = ["HTML", "CSS"];

var c2 = ["JS", "C++"];

var courses = c1.concat(c2);

Ví dụ trên có được mảng courses bằng cách nối c2 vào c1. Vậy courses có bốn phần tử: HTML, CSS, JS, C++. Nhớ là concat không tác động gì trên mảng gốc, mà nó chỉ sử dụng các mảng gốc để tạo ra mảng kết hợp mới. Vậy mảng c1, c2 vẫn không có sự thay đổi nào

Duyệt qua các phần tử mảng bằng forEach

courses.forEach(function(item, index, array) {

console.log(item, index);

});

//HTML 0

//CSS 1

//JS 2

Thêm một phần tử vào cuối bằng push

courses.push("GO");

Thêm một phần tử vào đầu bằng unshift

courses.unshift("C++");

Xóa phần tử cuối của mảng bằng pop

courses.pop();

Xóa phần tử đầu của mảng bằng shift

courses.shift();

Xóa các phần tử theo chỉ số splice

//Xóa 2 phần từ bắt đầu phần tử ở vị trí index

courses.splice(index, 2);

Nhân bản một mảng với slice

var courses2 = courses.slice();

Mảng kết hợp

Nhiều ngôn ngữ lập trình có hỗ trợ đánh chỉ số không chỉ là bằng số mà có thể bằng text, đây là các mảng kết hợp. Rất tiếc là JavaScript không hỗ trợ loại mảng này.

Tuy nhiên, bạn nhớ lại về đối tượng, truy cập thuộc tính đối tượng hoạt động khá giống với mảng kết hợp.

var person = []; //mảng rỗng

person["name"] = "John";

person["age"] = 46;

document.write(person["age"]);

//Outputs "46"

Bạn thấy person hoạt động giống như một đối tượng. Các chỉ số name, age không phải là chỉ số mảng mà là thuộc tính của đối tượng person. Do đó nếu bạn lấy số phần tử mảng person.length sẽ trả về 0

Cách đơn giản để duyệt qua các phần tử mảng

var index;

var a = ["a", "b", "c"];

for (index = 0; index < a.length; ++index) {

console.log(a[index]);

}

+) Function

1. Function là gì?

Function (hàm, chức năng), gọi chung là subprogram (chương trình con) có thể được gọi ở bên ngoài hoặc bên trong chính nó.

Nó bao gồm tập hợp các câu lệnh gọi là function body. Các giá trị có thể truyền đến một hàm, và một hàm có thể trả về giá trị. Bây giờ, với các ứng dụng hiện đại, các function có thể là một chương trình hoàn chỉnh, chứ không phải là k

2. Function không có tham số và không trả về bất cứ giá trị gì.

function sayHello () {

console.log("Hello !");

}

sayHello();

Function ở trên không có một tham số nào, và không trả về một giá tri.

3. Arrow function

const sayHello = () => {

console.log("Hello !");

}

sayHello();

Một arrow function có cú pháp ngắn hơn cú pháp function bình thường, nó có thể không có đối số, super hoặc new.target của nó. Những function này phù hợp nhất cho các non-method function và chúng không thể sử dụng như các constructor. Có điều gì khi tôi nói, function ở trên không trả về giá trị gì?

Nếu tôi cố gắng để lưu trữ kết quả của function được gọi ở trên vào một biến nó sẽ nhận giá trị "undefined".

+) async/await

+) XMLHTTPRequest/Axios

+) Prototype

+) Object

+) Json

2 Selector:

+) ByID

Phương thức getElementById trả về phần tử có thuộc tính ID là giá trị được chỉ định.

Phương thức này là một trong những phương thức phổ biến nhất trong HTML DOM và được sử dụng gần như mọi lúc bạn muốn thao tác (hoặc lấy thông tin từ) một phần tử trên tài liệu của bạn.

Trả về null nếu không có phần tử nào có ID được chỉ định tồn tại.

ID phải là duy nhất trong một trang. Tuy nhiên, nếu có nhiều hơn một phần tử với ID được chỉ định tồn tại (trùng ID), phương thức getElementById() trả về phần tử đầu tiên trong mã nguồn.

Cú pháp

var element = document.getElementById(elementID)

Tham số:

elementID là dạng chuỗi, giá trị của thuộc tính ID của phần tử bạn muốn nhận về

Giá trị trả lại:

Một đối tượng Element được mô tả trong đối tượng DOM khớp với elementID, hoặc null nếu không tìm thấy phần tử nào phù hợp.

Ví dụ

Lấy phần tử có id=“demo” và thay đổi màu của nó:

var x = document.getElementById("demo"); // Get the element with id="demo"

x.style.color = "red"; // Change the color of the element

HTML:

<html>

<head>

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

<title>getElementById example - daipho.com</title>

</head>

<body>

<p id="para">Some text here</p>

<button onclick="changeColor('blue');">blue</button>

<button onclick="changeColor('red');">red</button>

</body>

</html>

JavaScript:

function changeColor(newColor) {

var elem = document.getElementById('para');

elem.style.color = newColor;

}

+) ByName

+) ByTag

+) ByClassName

+) Focus

+) querrySelector

Phương thức querySelector() trả về phần tử đầu tiên là phần tử con của phần tử mà nó được gọi ra khớp với nhóm bộ chọn được chỉ định.

Lưu ý: Phương thức querySelector() chỉ trả về phần tử đầu tiên khớp với các bộ chọn được chỉ định. Để trả về tất cả các kết quả phù hợp, sử dụng phương thức querySelectorAll() để thay thế.

Nếu bộ chọn khớp với ID trong tài liệu được sử dụng nhiều lần (Lưu ý rằng “id” phải là duy nhất trong một trang và không nên được sử dụng nhiều lần), nó sẽ trả về phần tử khớp đầu tiên.

- Cú pháp

element = document.querySelector(CSS selectors)

hoặc

element = baseElement.querySelector(CSS selectors);

Giá trị tham số

CSS selectors:

String.

Chỉ định một hoặc nhiều bộ chọn CSS để khớp với phần tử. Chúng được sử dụng để chọn các phần tử HTML dựa trên id, lớp, loại, thuộc tính, giá trị của thuộc tính, v.v.

Đối với nhiều bộ chọn, hãy tách riêng từng bộ chọn bằng dấu phẩy. Phần tử trả về phụ thuộc vào phần tử nào được tìm thấy đầu tiên trong tài liệu.

Giá trị trả về:

Một đối tượng NodeList, biểu diễn phần tử đầu tiên khớp với (các) bộ chọn CSS được chỉ định. Nếu không tìm thấy kết quả phù hợp, null sẽ được trả về. Ném ngoại lệ SYNTAX\_ERR nếu bộ chọn được chỉ định không hợp lệ.

+) querrySelectorAll

Phương thức querySelectorAll() là phương thức JavaScript truy lục tất cả các phần tử trong tài liệu khớp với một bộ chọn CSS được chỉ định, giá trị trả về một đối tượng NodeList tĩnh (không phải NodeList thực). Đối tượng NodeList đại diện cho một mảng các nút phần tử, các nút có thể được truy cập bằng số chỉ mục – chỉ số bắt đầu bằng 0. Nếu như phương thức querySelector() chỉ trả về phần tử đầu tiên thì querySelectorAll() trả về tất cả các phần tử khớp với bộ chọn.

Cú pháp

Truy lục tất cả các phần tử trong trang tài liệu HTML:

var x = document.querySelectorAll(selectors);

Truy lục tất cả các phần tử con:

var elementList = parentNode.querySelectorAll(selectors);

Tham số:

selectors là DOMString chứa một hoặc nhiều bộ chọn CSS để so khớp. Chuỗi này phải là chuỗi chọn CSS hợp lệ; nếu không, phương thức sẽ ném ra một ngoại lệ SyntaxError (Xem bài viết Bảng đối chiếu các CSS Selector nếu cần). Nhiều bộ chọn có thể được xác định bằng cách tách chúng bằng dấu phẩy.

Giá trị trả về:

Một NodeList tĩnh (không thực) chứa một đối tượng Element cho mỗi phần tử khớp với ít nhất một trong các bộ chọn được chỉ định hoặc một NodeList rỗng trong trường hợp không có kết quả phù hợp.

Lưu ý: Nếu các bộ chọn được chỉ định có kèm một phần tử giả CSS (CSS pseudo-element), danh sách trả về luôn rỗng.

Ngoại lệ:

SyntaxError – Chuỗi bộ chọn CSS được chỉ định không hợp lệ.

+) Events Js (add event Listener)

Bảng các sự kiện thông dụng trong javascript.

STT Event Name Description

1 onclick Xảy ra khi click vào thẻ HTML

2 ondbclick Xảy ra khi double click vào thẻ HTML

3 onchange Xảy ra khi giá trị (value) của thẻ HTML đổi. Thường dùng trong các đối thẻ form input

4 onmouseover Xảy ra khi con trỏ chuột bắt đầu đi vào thẻ HTML

5 onmouseout Xảy ra khi con trỏ chuột bắt đầu rời khỏi thẻ HTML

6 onmouseenter Tương tự như onmouseover

7 onmouseleave Tương tự như onmouseout

8 onmousemove Xảy ra khi con chuột di chuyển bên trong thẻ HTML

9 onkeydown Xảy ra khi gõ một phím bất kì vào ô input

10 onload Xảy ra khi thẻ HTML bắt đầu chạy, nó giống như hàm khởi tạo trong lập trình hướng đối tượng vậy đó.

11 onkeyup Xảy ra khi bạn gõ phím nhưng lúc bạn nhã phím ra sẽ được kích hoạt

12 onkeypress Xảy ra khi bạn nhấn môt phím vào ô input

14 onblur Xảy ra khi con trỏ chuột rời khỏi ô input

15 oncopy Xảy ra khi bạn copy nội dung của thẻ

16 oncut Xảy ra khi bạn cắt nội dung của thẻ

17 onpaste Xảy ra khi bạn dán nội dung vào thẻ

Một số phương thức phổ biến khi sử dụng DOM

Node.getElementsByTagName('tagName')

trả về một danh sách các node con, cháu của một node với tag là 'tagName'. Ở đây, tagName có thể là body, h1, p, img, ...

Node.getElementsByClassName('className')

Tương tự như phương thức trên, phương phức này trả về một danh sách các node con, cháu của một node với thuộc tính class = 'className'.

Node.getElementById('id')

trả về node có thuộc tính id = 'id'.

Node.removeChild(child)

bỏ đi node child của một node và trả về node bị bỏ đi.

Node.appendChild(child)

thêm node child vào cuối danh sách childNodes của một node.

Node.insertBefore(newNode, referenceNode)

chèn thêm node newNode vào phía trước node referenceNode.

Node.replaceChild(newChild, oldChild)

thay thế node oldChild bằng node newChild.

Document.createTextNode(data)

Tạo ra một node kiểu text với giá trị text là data.

Document.createElement(tagName)

Tạo ra một node mới có kiểu là tagName (h1, h2, p, img, div,...)

Document.querySelector(selectors)

Trả về node đầu tiên thoả mãn selectors. Trong đó, selectors là một string chứa một hay nhiều CSS-selector phân cách nhau bằng dấu phẩy.

Document.querySelectorAll(selectors)

Trả về danh sách các node thoả mãn selectors, theo thứ tự sử dụng thuật toán DFS. Perfect practice makes perfect - Complete JavaScript - completejavascript.com