1. **Browsing the web**

* Khi gõ 1 tên miền sẽ gửi tên miền đến ISP và ISP sẽ gửi đến DNS Server, DNS Server sẽ gửi lại địa chỉ IP của tên miền cho browser. Browser khi nhận được IP sẽ gửi request đến địa chỉ IP đó, Khi đó Server có địa chỉ đó sẽ trả về các file bằng cách gửi response.
* Nếu biết địa chỉ IP trước ta có thể gõ trực tiếp lên thanh URL.
* Trong Cmd có thể dung lệnh tracert url để xem địa chỉ ip của đích và các địa chỉ ip của các server đi qua.
* Tốc độ có thể phụ thuộc vào vị trí Server, số trạm mà gói tin sẽ qua và kích thước tập tin trả về.
* [www.vox.com/a/internet-maps](http://www.vox.com/a/internet-maps) : www là một ngôn ngữ chung mà computer có thể nói hay dung

1. **HTML**

* Trong thẻ form sử dụng method attribute, sẽ có 2 cách là GET và POST. Trong đó, GET sẽ là mặc định, lúc này thông tin sẽ hiện trong thanh url. Với POST, thì thông tin được đưa vào phần body của gói tin nên sẽ ko hiện.
* Form còn có attribute là action hay dung với php.
* Khi submit sẽ có dạng như sau :
* ?firstname=Mickey&lastname=Nguyễn+Thanh&email=thanhdan398%40gmail.com&password=ádasdasd&birthday=2020-06-02&gender=on&cat=on
* Trong đó, dấu “?” có nghĩa như nó sẽ nhận 1 chuỗi các data.
* NOTE : cần phải có value và name cho tất cả các thẻ input Khi đó sẽ gửi đi thông tin là name và value. Nếu không sẽ gửi giá trị là on như ví dụ trên.

1. **CSS**

* Khi gặp property mới ta có thể vào trang css-trick và tìm hiểu về nó.
* Truy cập trang : <https://paletton.com/> để chọn màu.
* Việc áp dụng luật của CSS sẽ phụ thuộc vào : Specificity, Importance, Source Order

\*\*\* Các nguyên tắc sẽ như sau :

* **Elements có thể kế thừa styles từ các elements khác(độ ưu tiên thấp nhất)** : Có thể xem đây như là mối quan hệ parent-child. Do có độ ưu tiên thấp nhất nên child chỉ cần có 1 rule riêng sẽ override parent bất chấp có !important.
* **SVG inline attribute (độ ưu tiên thấp thứ hai)** : Điều này có nghĩa là 1 CSS rules trong stylesheet sẽ override nó. Việc sử dụng các attribute svg inline sẽ giúp cho việc sao chép các svg và attribute dễ dàng và ta có thể style lại bằng css.
* **Stylesheets :** Sẽ được chia thanh 2 dạng là external(sử dụng link) và embedded(dùng style). Cả 2 đều có độ ưu tiên như nhau do đó sẽ xét theo order hay vị trí.
* **Specificity hay selectors (Có độ ưu tiên cao hơn luật order) :** Cách mà chúng ta chọn elements cũng ảnh hưởng đến độ ưu tiên. **Tags < classes < ids.**
* **Ordering :**  độ ưu tiên này sẽ có tác dụng tăng dần từ trái-sang-phải hay từ trên-xuống
* **Inline Styles(độ ưu tiên cao thứ 2 sau !important) :** sử dựng style trong attribute của element, do đó chỉ có !important mới có thể override nó. Nó sẽ override tất cả các giá trị trên.
* **The !important key :** được dùng để override ordering, **specificity, inline rules,**



* Tính toán specificity : theo cách tính sau

**Style Attribute, ID, ( Class, pseudo-class, attribute selector), element**

* Từ trái qua phải các tham số sẽ là 0,0,0,0 khi có 1 style thì sẽ tăng lên 1. Ví dụ :

**+ ul#nav li.active a => 0,1,1,3**

**+ #footer \*:not(nav) li => 0,1,0,2**

* **Note : Universal selector(\*), combinators ( +,>,~, ‘ ’, || ), negative pseudo-class (:not()), :is(), :where() sẽ không ảnh hưởng đến specificity Nhưng selectors đươc gán bên trong :not(selector) thì lại có**
* **Nên sử dụng !important 1 cách thận trọng.**
* **Việc có cùng specificity nhưng việc chọn selector gần hơn sẽ không ảnh hưởng đến specifycity mà sẽ dựa vào order.**

1. **BOOTSTRAP**

* CDN hay còn gọi là Content Delivery Network.
* CDN là gì : <https://www.cloudflare.com/learning/cdn/what-is-a-cdn/>
* Có thể tập dùng bootstrap, sử dụng
* Meta tag là gì : <https://www.w3schools.com/tags/tag_meta.asp>
* Viewport meta tag là gì : <https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag>
* Utf-8 là gì : <https://www.quora.com/What-is-UTF8>
* CSS Tricks Perfect Background : <https://css-tricks.com/perfect-full-page-background-image/>
* Dịch vụ email : mailchimp Tìm hiểu them về phần này.
* Đẩy web lên online : ta sẽ sử dụng github.
* Sử dụng nguồn animate.css để tạo các animation có sẵn Có thể tải về tham khảo cách làm.
* Tra cứu các CSS/HTML templates để tập thực hành.
* Sử dụng trang caniuse để xem hỗ trợ css trên các trình duyệt.
* Đọc lại queries media : <https://css-tricks.com/snippets/css/media-queries-for-standard-devices/>
* Trang tham khảo các hình, templates free : <https://interfacer.xyz/>

1. **Javascript and JS Advance**

\*\*\*\*\*\*\*\*\*\*\*\*\* FUNCTION \*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Xem lại var, let, const
* Các phiên bản ECMAScript mới có thể được hỗ trợ hay thực hiện thông qua **Babel** complier của JS https://babeljs.io
* Currying in function :
  + Const multi = (a,b) => a \* b;
  + Const multiCur = (a) => (b) => b \* a;
  + Const muiltiByA = multiCur(A)
  + multiCurA(B) 🡺 sẽ trả về a \* b hay multiCur(A)(B)
* Compose in Function : f và g là 2 function

Const compose = (f,g) => (a) => f(g(a));

* Khi viết hàm JS, Ta cần quan tâm việc tránh side effects, Khi đó ta sẽ chỉ dung rerurn. Khái niệm này được gợi là Deterministic. Tức là dù input là gì thì nó sẽ luôn trả về 1 value ????????????

\*\*\*\*\*\*\*\*\*\*\*\*\* ARRAY \*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Xem lại map, filter, reduce method đặc biệt là reduce

\*\*\*\*\*\*\*\*\*\*\*\*\* OBJECT \*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Xem lại Reference, Context hay This, instantiation Prototype và class
* Xem lại cách clone Array và Object.
* ES7 (2016) : Có thêm Method .includes(substr or subarr or property) và x\*\*2
* ES8 (2017) :

+ str.padStart(value number) và str.padEnd(value number) dung cho string : Sẽ tạo thêm số value khoảng trống hay dấu cách vào trước hay sau string. Trả về string mới ko ảnh hưởng string cũ

+ chỉnh sửa parameters trong function cho phép có dấu , trong đó. Vd func(a,b,c,d,) => cho phép dư dấu phẩy ở cuối

+ Object.values(obj) và Object.entries(obj) và Object.keys(obj) : xem lại.

+ Async và Await

* ES10 (2019) :

+ arr.flat(value) dùng cho array : value là số lớp làm phẳng(flat). Tức là nếu [1,2,3,[4,5]].flat() => [1,2,3,4,5] nhưng nếu [1,2,[3,4,[5]]].flat() => [1,2,3,4,[5]] Để lấy hoàn toàn ta set số layer là 2 : [1,2,[3,4,[5]]].flat(2) => [1,2,3,4,5] : Có thể dùng kết hợp với entries : [1,2,3,,,,,5].flat() => [1,2,3,5]

+ arr.flatMap() :

+ str.trimStart() và str.trimEnd().

+ Object.fromEntries(arr) áp dụng cho array chứa cặp [key, value] : trả về object Ngược lại với object.entries(obj)

+ try{} catch(error){}

* Object là enumerable chỉ có thể dùng for\_in ko dùng đc với for\_of
* Array và String là Iteratable có thể dùng cả 2 for ở trên.

\*\*\*\*\*\*\*\*\*\*\*\*\* DEBUGGING \*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Keyword là : debugger; : khi browser gặp từ debugger nó sẽ dừng lại. nó sẽ mở 1 window ngay tại keyword.

\*\*\*\*\*\*\*\*\*\*\*\*\* HOW JS WORKS \*\*\*\*\*\*\*\*\*\*\*\*\*\*

* 1 JS Engine sẽ có 2 phần chính là Memory Heap và call stack

+ Memory heap là nơi cấp phát bộ nhớ.

+ Call Stack : là nơi code được đọc và thực thi (executed) cho ta biết mình đang ở đâu trong chương trình

* Memory leak là các biến được khai báo nhưng ko dùng đến, dẫn đến memory heap có thể bị full. Do đó var global ko nên .
* Call stack : sẽ đọc line code đầu tiên và put vào call stacks và sau đó executes đó sau đó sẽ pop line code đó ra. Vd phức tạp hơn như sau :

Const one = () => {

Const two = () {

Console.log(“hi there”);

}

Two();

}

One();

* Lúc này trong call stack sẽ là (từ trái qua xem như từ dưới lên ) :

one() | two() | consolelog()

* JS là ngôn ngữ 1 luồng : tức là chỉ chỉ làm 1 chuyện tại 1 lúc, call stack là dạng first in last out. Nhiều call stack thì đc xem là multi-threads.
* JS là non-blocking : tức là bất đồng bộ (asynchronous)
* JS Run-Time Environment : là 1 phần của browser , chứa các phần mở rộng. nằm ở phía top của engine. Gồm có Web APIs (DOM, AJAX, Timeout), Callback Queue, Event Loop. Trong đó setTimeOut là 1 phần của Web APIs. Từ đó, ta có thể thực hiện 1 chương trình asynchronous.
* Lệnh trong Bash :

+ start directory/file : mở directory hoặc file.

1. **GIT-GITHUB**

* **Git clone url(https) : sẽ clone project về máy**
* **Git status**
* **Git add filename or . để add tất cả file có thay đổi.**
* **Git commit -m “message”**
* **Git push**
* **Git pull : được dùng để update bản mới nhất từ github**
* **Git branch : hiện ra các branch**
* **Git branch branchname : tạo thêm branch có tên là branchname**
* **Git checkout branchname**
* **Git checkout -b branchname : tạo và chuyển đến branch mới.**
* **Cách merge :**

**+ tại branch cần merge : dùng lệnh git merge master or tên branch cần merge.**

**+**

1. **NPM**

* Sử dụng trang npm để tìm kiếm các packages và các packages này đươc chứa trong npm Registry. Mọi ng ai cũng có thể submit packages hoặc modules và nó chứa 2 thứ là js file và package.json file
* Package.json là metafile mô tả package đó.
* Thưởng sẽ có 5 loại packages :

+ package được dnngf cho browser như jquery..

+ nodejs (node packages manager)

* React thực chất là 1 package của facebook
* Nodejs cũng dùng engine v8.
* Khi dùng **lệnh npm init** để khởi tạo package.json file. Lúc này sẽ tạo các thông tin như sau :

+ package name : có thể bỏ qua.

+….

+ Có thể enter cho qua

* Để cài npm mới nhất ta dùng lệnh sau :

**Npm install npm@latest -g.**

* **Npm install -g live-server : -g** sẽ cài global cho toàn máy ko chỉ riêng folder.
* ở live-server thì địa chỉ mặc định là 127.0.0.1:8080 hay có thể thay bằng localhost:8080.
* Thư viện npm lodash là thư viện js (viết js gọn hơn) : cài bằng npm local
* Npm install lodash
* Khi có file package.json : ta sẽ dùng lệnh npm install để cài đặt lại các package đã lưu rong package.json.
* Npm install -g browserify :: giúp tạo file bundle
* Để cài lại các packages có trong npm ta sẽ dùng lệnh : npm install
* Npm run tên\_script\_trong\_file.json : sẽ chạy file script được định nghĩa trong file.json
* Vd :

“scripts” : {

“build”: “browserify scripts.js > bundle.js && live-server”

}

* Npm run build
* Để kiểm tra các version của 1 packages ta làm như sau :

<https://semver.npmjs.com/>

* Nên kiểm tra và update hay cài các version thích hợp.

# **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*REACT JS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

* Cài đặt react app

Npm install -g create-react-app

* Sau đó tạo app

create-react-app name\_app (name luôn là lowercase)

# **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*HTTP-HTTPS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

* HTTP là protocol dùng để trao đổi dữ liệu với server.
* Browser gửi request để yêu cầu data , thông qua HTTP server hiểu được yêu cầu của browser và gửi lại response.
* Trong HTTP có các keywwords sau :

+ GET : nghĩa là 1 get hay lấ, nhận file

+ POST : nghĩa là ta muốn post hay add some data vào database

+ PUT : gửi 1 số data và sẽ update data có sắn trong database

+ DELETE : Xóa 1 phần nào đó của database

* Server sẽ gửi về 2 thông tin là :các http message hay status code : tham khảo tại trang : <https://www.w3schools.com/tags/ref_httpmessages.asp>
* Và các loại data như HTML, CSS….