# Agile

**La mot mindset, no co nhung cai quy luat de tao nen mot quy trinh trong lam viec**

**Javascript**

# Scrum vs Waterfall

**Waterfall:**

Phat trien mot cach tuan tu, de qua giai doan tiep theo thi can phan hoan thanh giai doan truoc do thanh cong. Khi ma khach hang muon thay doi gi thi kho de co the quay lai duoc

**Scrum**

Tap trung den khang hang hon, chia thanh cac sprint, sprint keo dai tu 2tuan den 1 thang, la sau khi phat trien chuc nang roi nhan lai feedback cua khach hang, roi lai phat trien tiep cho phu hop voi yeu cau cua khach hang, nen scrum the hien su linh hoat va dap ung duoc su thay doi lien tuc tu khach hang

# Server side rendering vs Client side rendering

Ssr la tra ve html document cho client hien thi

Cls la tra ve mot html voi div rong va file Js. Sau do no chay file Js roi append HTML vo div

—----------------------------

**Ssr:**

**-Tot cho ceo**

**-request dau tien nhanh hon**

**Csr**

**-Tuong tac vs nguoi dung tot hon, chuyen trang khong can phai load lai**

# Data type

5 type String , number, boolean, null, undefined

Type of null is OBJECT

Type of undefined is undefined

2 non -primitive : Object vs Array

# Context

This: Tu khoa this la muon noi den doi tuong ma no thuoc ve

Dac tinh:

1. Trong mot phuong thuc, this tham chieu toi doi tuong truy cap phuong thuc
2. Dung ngoai phuong thuc, this tham chieu toi doi tuong global (window)

Luu y

This trong ham tao la dai dien cho doi tuong se duoc tao

This trong 1 ham la undefined khi o strict mode

bind(), call(), apply() de tham chieu this den doi tuong khac

arrowfunction khong co this, this trong arrow se la cai this gan do

var person = {

hhee() {

console.log(this)

},

haha: () => {

console.log(this);

}

}

person.hhee()

person.haha()

Hhee chi toi Person, haha chi toi window

unction hoho() {

this.name = "huy"

this.hehe = function () {

console.log(this)

}

}

var ecec = new hoho();

This trong hehe chi toi doi tuong ecec, con this.name (this trong ham tao) se chi toi doi tuong se duoc tao

function hoho() {

this.name = "huy"

this.hehe = function () {

function test() {

console.log(this)

}

test()

}

}

var ecec = new hoho();

This trong test se chi toi doi tuong window

function hoho() {

this.name = "huy"

this.hehe = function () {

test = () => {

console.log(this)

}

test()

}

}

Dung arrow function thi this trong test se la this gan do , tuc la this trong hehe , this luc nay se la doi tuong se duoc tao

# Rest operator

function logger(...params) {

console.log(params)

}

logger(1, 2, 3, 4)

function logger([a, b, ...rest]) {

console.log(rest)

}

logger([1, 2, 3, 4])

function logger({name,age,...rest}) {

console.log(rest)

}

logger({name:"Javascript", age: 15, description:"content"})

# Spread

var arr1 = [1,2,3]

var arr2 = [4,5]

var arr3 = [...arr1, ...arr2]

console.log(arr3)

var arr1 = [1, 2, 3]

var arr2 = [4, 5]

var arr3 = [...arr1, ...arr2]

console.log(arr3)

var object1 = { name: 'javascript' }

var object2 = { price: 1000 }

var object3 = {

...object1,

...object2

}

console.log(object3)

var obj1 = {

name: "javascript",

price: 100,

description: "content"

}

var obj2 = {

...obj1,

description: "content khac"

}

console.log(obj2)

var arr1 = [1, 2, 3]

function logger(...rest) {

for (let i = 0; i < rest.length; i++) {

console.log(rest[i])

}

}

logger(...arr1)

# Destructuring

var obj1 = {

name: "javascript",

price: 100,

description: "content",

children: {

name: "reactjs"

}

}

var { name: parentName, children: { name } } = obj1

console.log(parentName)

console.log(name)

var obj1 = {

name: "javascript",

price: 100,

}

var { name, description = "default value" } = obj1

console.log(description)

Khi khong co lay ra default value

var obj1 = {

name: "javascript",

price: 100,

description: "content"

}

var { name, description = "default value" } = obj1

console.log(description)

# 

# SCOPE

**Global:** Toan cau, cho nao cung truy cap duoc

**Local:** ham, function, dung var

**Block:** dung let,const , pham vi {}

-Khi goi moi ham luon co pham vi moi duoc tao ra

-Cac ham co the truy cap cac bien khi duoc khai bao trong pham vi cua no va ben ngoai cua no

===

Bien toan cau bi xoa khoi bo nho khi f5 hoac tat tab , hoc ket thuc chuong trinh

Bien trong block scope, khi thoat khoi block scope {}, thi se tu bi xoa khoi bo nho

Tuong tu nhu block scope, sau khi thuc hien xong function, thi cung se tu bi xoa khoi bo nho

**Exercise :** <https://kaledoux.medium.com/javascript-exercises-scope-hoisting-and-closures-423893501195>

# Primitive and Reference Types JS

**Primitive**

Let a = 1 (tao vung nho voi gia tri bang 1, gan cho a)

a = 2 (sua vung nho cua a voi gia tri bang 2)

Let b = a (tao vung nho voi gia tri bang 2(copy tu a), gan cho b)

**Reference**

Let a = {name : “BMW”} (tao vung nho voi gia tri {name : BMW}, gan **dia chi** cua vung nho do cho a )

Let b = a (gan **dia chi** cua a vao b)

===============

Let a = {name : “BMW”}

a. name = “Mercedes” (khong tao vung nho moi) // tong cong 1 vung nho duoc tao ra

Let a = {name : “BMW”}

a = {name : “Mercedes”} (tao vung nho moi) // tong cong 2 vung nho duoc tao ra

Let a = {name : “BMW”, profile : {price : 200}} // van la 2 vung nho duoc tao ra

// tao ra vung nho co gia tri {price: 200}, gan **dia chi (#001)** cua vung nho do cho profile

// tao vung nho voi gia tri {name : “BMW”, profile : #001}}, gan **dia chi** cua vung nho vua duoc tao cho a

**Pass by value (truyen kieu tham tri)**

Function f(a,b){

a = 0;

b = 0

console.log(a,b) //0 0

}

a = 1

b =1

f(a,b)

console.log(a,b) // 1 1

Hàm được gọi bằng cách truyền trực tiếp giá trị của biến làm đối số. Thay đổi đối số bên trong hàm không ảnh hưởng đến biến được truyền từ bên ngoài hàm.

**Pass by reference (truyen kieu tham chieu)**

Function f(obj){

obj.name = “Mercedes”

console.log(obj) {name : “Mercedes”}

}

Let obj = {name : “BMW”}

f(obj)

console.log(obj) // {name : “Mercedes”}

# **React HOOK**

**TINH NANG TIEU BIEU CUA REACT**

**JSX,**

**Component,**

**Virtual Dom,**

**Npm cung cap library**

**customHook**

La hook do minh tu dinh nghia ra

La 1 function hoi dac biet

Co the su dung hook khac nhu useState, useEffect

Dat ten theo dang: use+Something

## useRef(),

Hoat dong: cung nhu useState, nhan vao initial ref , luon luon tra ve mot object, co key la current ben trong object, value cua thang current la thang minh truyen vao luc khoi tao, hoac la khi minh gan lai. De gan lai ref thi se la thang do.current = thang minh muon gan lai.

Co the dung de lay cai element trong dom, bang cach ref={ten cua bien minh gan voi useRef}

Tac dung: khi render lai no van giu duoc cai ma minh luu, no co the dung luu cai gia tri cua cai state truoc (tu nho lai la dung useEffect de su ly gan ref)

## React.memo, high order component

**MEMO: (HOC)** dùng để bao bọc bên ngoài của 1 component.

Hoat dong: so sanh props cua lan render truoc va lan render hien tai, so sanh bang toan tu ===, neu chi can 1 props thay doi thi component se duoc render lai, neu khong thi khong render lai component.

Tac dung: tang performance, khong render lai nhung component con khi khong can thiet

Dành cho câu hỏi làm sao để khi component cha render lại còn component con thì không

**UseCallBack(),** dung vs React.memo

Hoat dong : useCallback(), doi so dau tien nhan vao callback, la ham cua minh, thu hai la mang dependency, neu depency la [], thi nhung lan render tiep theo , no se luon tra lai tham chieu cua callback cua lan dau tien khi ma no duoc mount , neu co dependency thi khi ma dependency thay doi, no se tao ra callback moi, va no se tao ra mot cai tham chieu moi.

Tac dung: dung vs react memo, tranh viec khien props thay doi, lam component phai render lai,

-> dung de tang performance

**UseMemo()**, dung de khong render lai mot cai logic khong can thiet

Hoat dong: doi so dau tien nhan callback, trong cai callback, tra ve cai ket qua ma minh muon memo, doi so thu hai la mang dependency, neu de dependency la [], thi no se thuc hien cai tinh toan mot lan thoi, nhung lan render lai no se ko tinh toan, neu co dependency, thi khi dependency thay doi, thi no se tinh toan lai.

Tac dung: tang performance, khong phai render lai logic khi khong can thiet

REF: <https://www.youtube.com/watch?v=TGaxZyZzB7E&list=PL_-VfJajZj0UXjlKfBwFX73usByw3Ph9Q&index=39>

## useReducer()

Hoat dong: 2 doi so, doi so dau la reducer, doi so 2 la initialState. Thang reducer thi se la mot function nhan vao state hien tai, va action, ben trong se la switch case, va tuy vao action ma minh thay doi thang state. UseReducer tra ve array, voi gia tri thu nhat la thang state, gia tri thu 2 la dispatch, (dispatch dung de kich hoat action).

Tac dung: dung khi state cua component phuc tap, co nhieu action lam thay doi cai state do, hoac khi ma state do value la nested object hoac array thi nen dung useReducer de cho code de dang va de hieu hon

## useContext()

**React context**

Giup pass down data giua cac component ma khong can dung den props , nghia la no giup chia se data mot cach de dang hon

**Tac dung**

Tranh props drilling, truyen props qua nested component, component khong can thiet dung den props do

Vi du component A => component B => component C, A phai truyen qua B roi B truyen C, trong khi B khong su dung props

**Su dung**

1. Create context
2. Lay context do, wrap context provider xung quanh thang consume component
3. Put value cho provider
4. Lay value cua provider bang cach dung useContext (Hook), context.consumer

# React context vs Redux

-debug

-middleware

-addons

-cross platform

-better performance

# Module export ES6

**EXPORT**

1 file chi duoc export default 1 lan

Import khong can destructuring

Export { name1, name2, name3} // export list

Export const name1 =2 , name2 = 3, name3 = 4

Import {name1, name2, name3} // dung destructuring de lay

Import \* as something

Somthing.name1, something.name2, something.name3 // khong dung destructuring

//export tai file khac

Export {default} from …. // export default cua file khac

Export {default as something} //chuyen export default thanh export thuong

Export \* from // export thuong cua file khac, khong bao gom export default

Export \* as something from // export thuong cua file khac, khong bao gom export default

// import {something} de lay

Export {default, something} // export default, va export thuong cua file khac

[**https://reactjs.org/blog/2015/12/18/react-components-elements-and-instances.html**](https://reactjs.org/blog/2015/12/18/react-components-elements-and-instances.html)

# 

# High Order Function

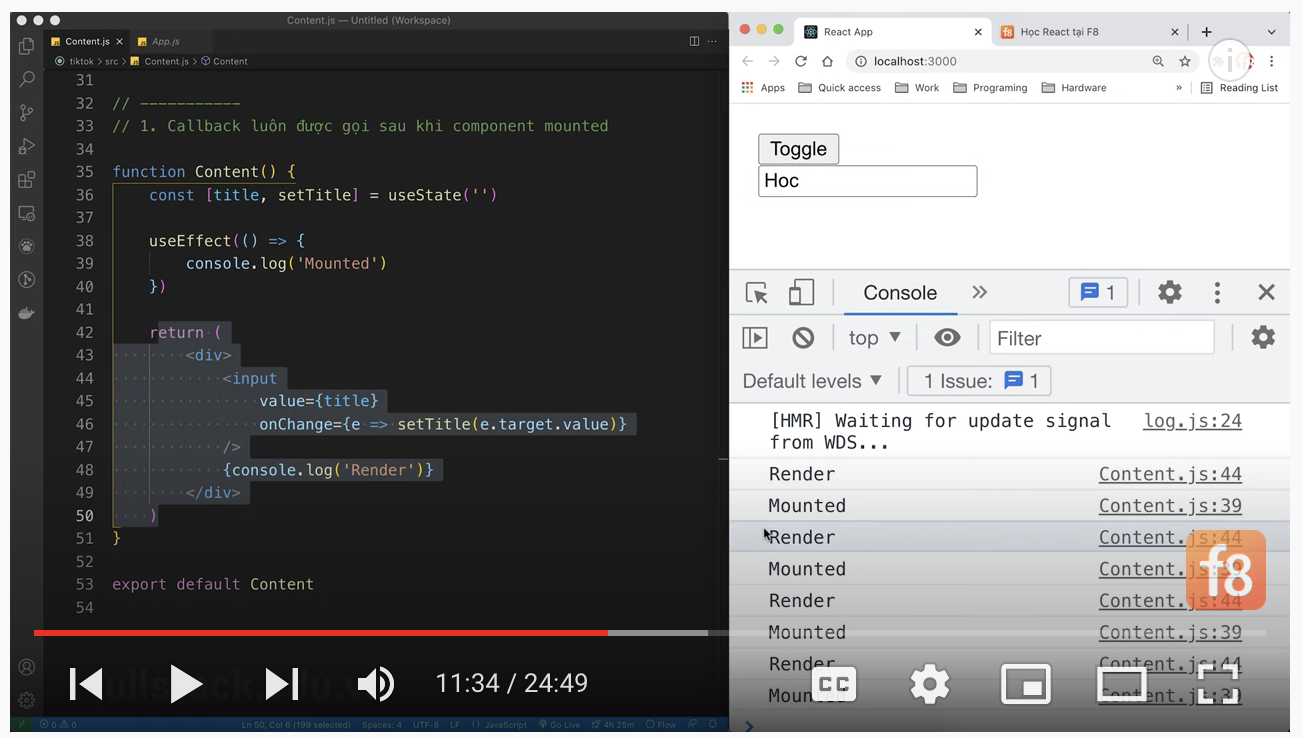
# 

# High Order Component

# useEffect()

**useEffect(callback)**

được gọi mỗi lần khi component được re-render lại

chú ý chỉ được gọi khi element đã được thêm vào DOM ( nghĩa là nó render phần return của component đó xong xuôi hết rồi mới gọi tới useEffect)

**useEffect(callback,[])**

chỉ gọi 1 lần, khi component được mounted.

**useEffect(callback,[deps])**

chỉ gọi thi deps thay đổi

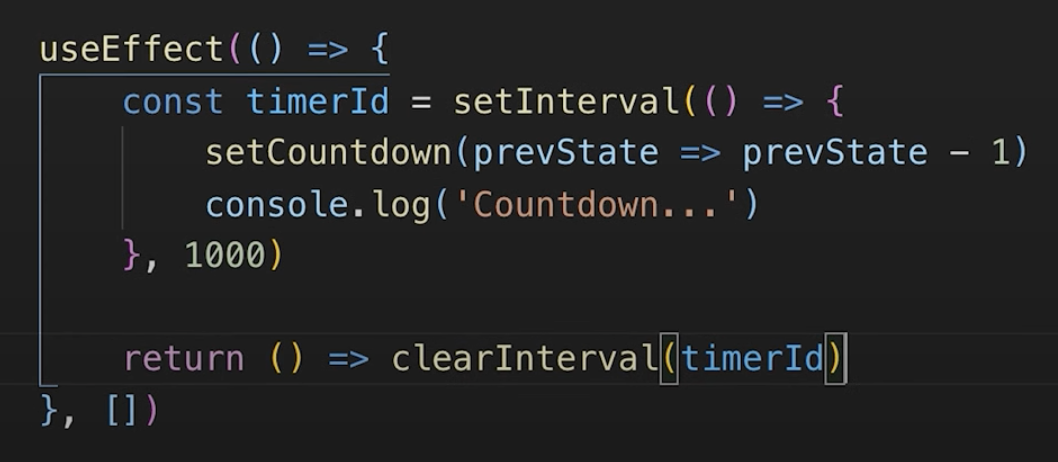
**\*Chung tất cả 3 cái trên**

+ callback luôn được gọi sau khi component mounted

+ Clean up function được gọi trước khi component unmounted

Trong callback của useEffect có thể return ra 1 hàm return () => {} (hàm đó dùng để clean up) thường dùng để clear eventListeners, clear Interval, clear Timeout

* Clean up function luôn được gọi trước khi callback đc gọi trư lần đầu tiên nó đc mounted



**Refs**

Tham chieu toi element trong dom

Manage focus, text selection, media playback

Trigger animation

**Forward Ref**

forwardRef là một method cho phép các components cha truyền các refs (tham chiếu) xuống các component con của chúng. forwardRef đưa cho component con một tham chiếu đến một phần tử DOM được tạo bởi cha của nó và điều này cho phép chúng có thể đọc và sửa đổi phần tử đó ở bất cứ đâu mà nó đang được sử dụng

**UseImperativeHandle()**

**Debounce va throttle**

# LIFECYCLE in CLASS COMPONENTS

**Lifecycle trong reactjs co 3 loai**

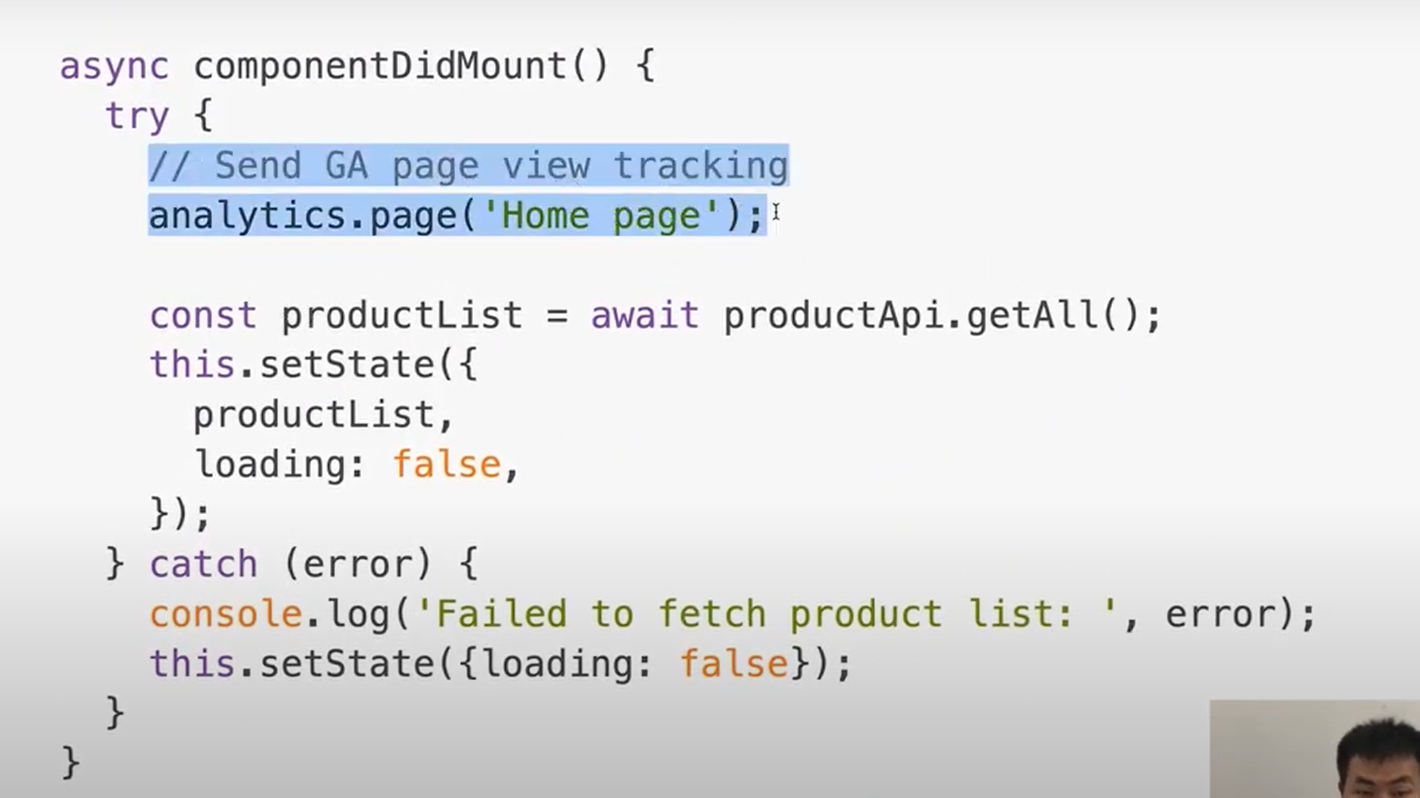
* **Mounting (thuc hien 1 lan)**
* **Update (co the thuc hien nhieu lan)**
* **Unmounting ( thuc hien 1 lan)**

**ComponentDidMount()**

Gọi 1 lần trước khi render

Khởi tạo dữ liệu cho component: gọi API, biến đổi dữ liệu, cập nhật state,

Gửi tracking page view ….

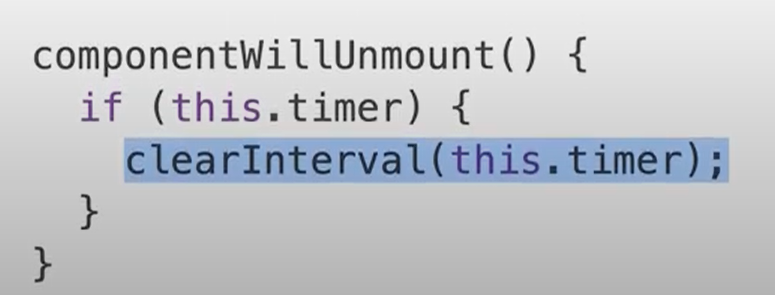
****

**ComponentWillUnmount()**

Được gọi trước khi component bị hủy

Thường dùng để clear timeout hoặc interval

Reset dữ liệu trên redux nếu cần thiết



**ComponentDidUpdate()**

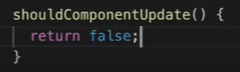
Được gọi khi có sự thay đổi về state và props của component đó

**shouldComponentUpdate()**

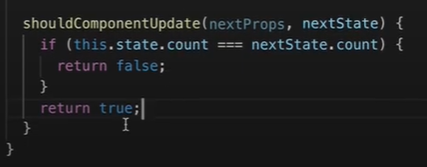
**Q:** Khi component cha thay đổi, làm sao để component con không thay đổi?

A: Dùng shouldComponetUpdate

Mặc định thằng này trả về true, còn nếu trả về false thì thằng này ko render lại



Nếu state của component không thay đổi thì nó sẽ không render lại , cho dù cha có render



REF: <https://www.youtube.com/watch?v=0oQDQD2yOPs>

**React Interview**

# What is JSX?

JSX stands for JavaScript XML.

JSX allows us to write HTML in React.

JSX makes it easier to write and add HTML in React.

JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.

# Why class is className in React?

JSX compile to javascript nen class co the bi hieu class trong javascript

# Describe data flow in React?

One-way-binding, cac component co moi quan he parent-child, truyen props tu parent xuong child

Drilling props, thi dung context, redux, global state

# How would you delay an api call until it is mounted?

Component did mount, useEffect []

# Ternaries va && , nen dung cai nao

Nen dung ternareis, k hieu tu bat clip xem lai

REF:<https://www.youtube.com/watch?v=8bhp89AIsnc>

# Difference between == va ===

1. ES5, ES6
2. What is IIFE ? Immediately Invoked Function Expression, the hien tinh dong goi, vi bien ben trong la private
3. What is this keyword? What does it do

# What is difference between absolute and relative, static?

Relative:

Căn theo vị trị của thẻ đó

Absolute

Căn theo vị trí của thẻ cha gần nhất có thuộc tính position

Static

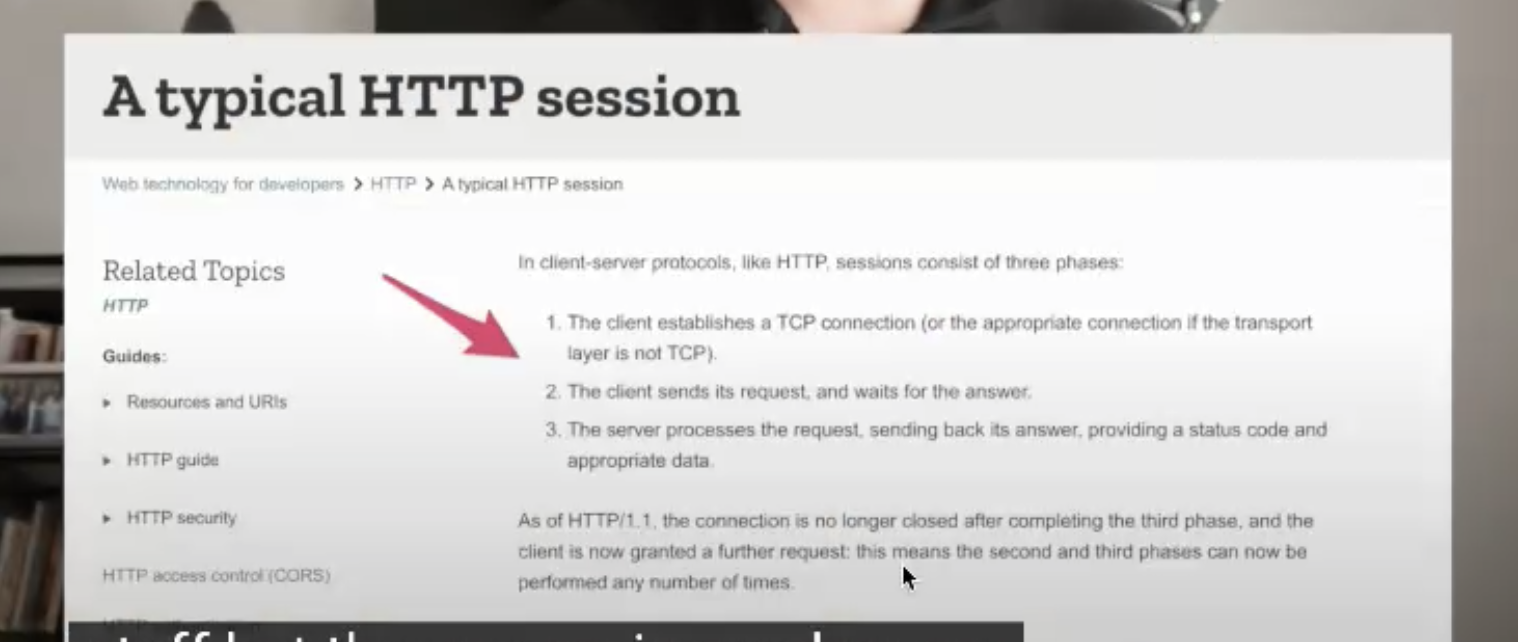
Là default của mấy elements đó rồi

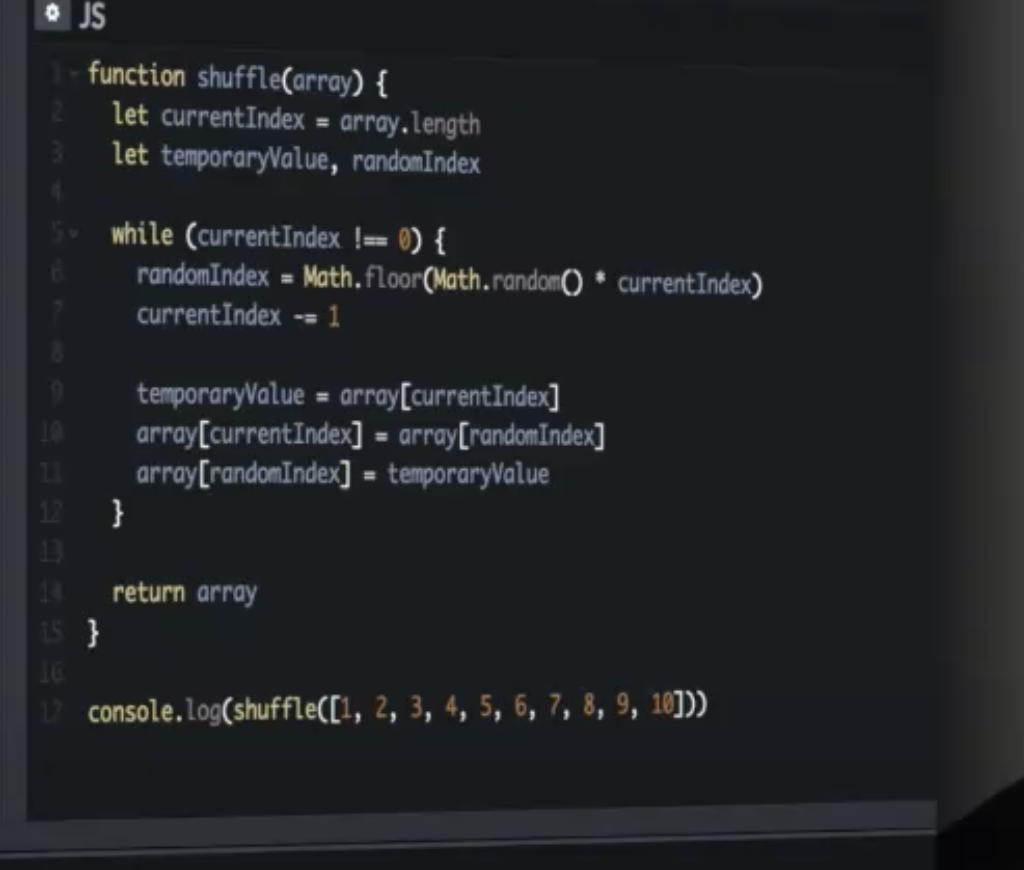
# What are three parts of an HTTP request ?

Trc khi sent pre-flight,

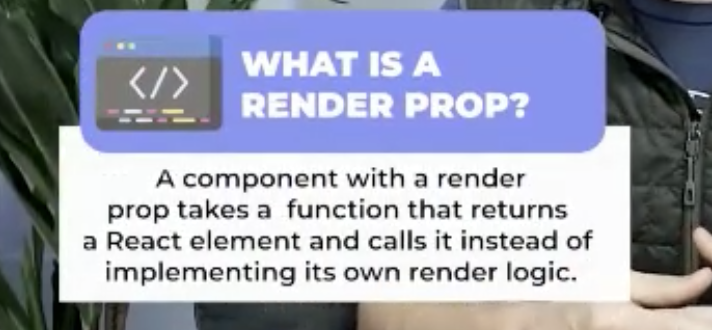
Trong khi sent pending,

Sau khi sent (resolve, reject)

****

12. Shuffle an array of integer

13.What is Higher Order Component

14. What is a render props

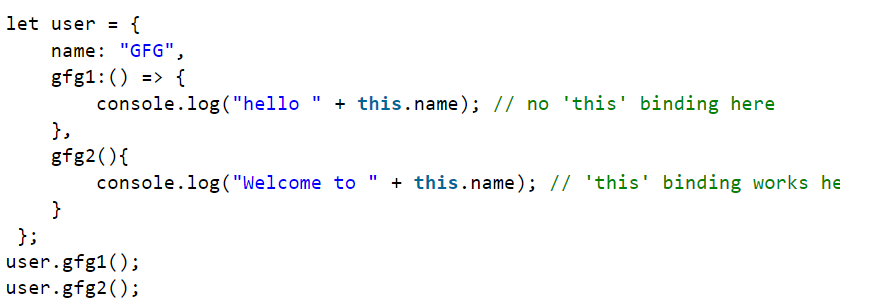
The term [“render prop”](https://cdb.reacttraining.com/use-a-render-prop-50de598f11ce) refers to a technique for sharing code between React components using a prop whose value is a function.

[React Render Props Made Simple!](https://www.youtube.com/watch?v=3IdCQ7QAs38&t=587s)

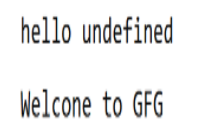
15. How to write custom Hook

**16 .Arrow function vs normal function**

* Arrow func do not have their own this



Result:



* Arguments objects are not available in arrow functions, but are available in regular functions.

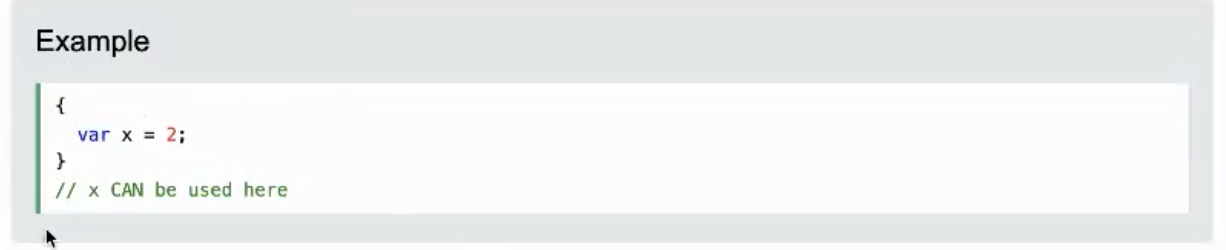
**1.mấy cách khai báo css:**

3 cách style trên html, style.css, inline

**2. khai báo var vs let**

**3.Scope trong javascript**

**ES5 Global scope , local scope, ES6 có thêm Block Scope (dung let const) vì biến tạo let, const tạo trong block scope k thể access từ bên ngoài**

****

**4.SessionStorage vs LocalStorage ( so sanh khac nhau)**

**LocalStorage:**

**+ tắt tab hoặc browser thì vẫn lưu. Không bao giờ hết hạn**

**+ mo lai van co   
SessionStorage:**

**+ tồn tại liên tục miễn khi tab và browser vẫn mở. Load hoặc restore trang web thì vẫn tồn tại**

**+ Scope = per window ( nghia la ngta mo 1 window moi thi van ko co)**

**5. Tại sao nên dùng express?**

**Vì express là FRAMEWORK của NodeJS nên nó CUNG CẤP cho một số function sẵn, một số module sẵn, code sẵn.**

**Lợi ích:**

* **Giúp việc lập trình thoải mái hơn.**
* **Giúp tạo ra kiến trúc MVC tốt hơn.**

**6. Loại function in Javascript.**

**Có 3 loại: Normal function, Arrow function and Anonymous function.**

# 7. Call, Bind and Apply differences

**Define:**

* **Call invokes the function and allows you to pass in arguments one by one. call(first one Context, argument, argument)**
* **Apply invokes the function and allows you to pass in arguments as an array. apply(context, [argument, argument])**
* **Bind invokes a new function, allowing you to pass in a this array and any number of arguments.**

****

**Hai dòng code mang ý nghĩa tương đương nhau.**

**Nếu “.apply()” không có arguments thì sẽ dùng như call.**

**Ref:** [Javascript Interview Questions ( Call, Bind and Apply ) - Polyfills, Output Based, Explicit Binding](https://www.youtube.com/watch?v=VkmUOktYDAU&ab_channel=RoadsideCoder)

# 8. Tại sao khi xài mongo, phải “npm mongoose” vào thêm.

**Mongoose là 1 library viết sẵn, giúp đỡ mình làm nhanh hơn, chính xác hơn.**

# 9. Restful API là gì? Có bao nhiêu method? Kể tên vài HTTP status.

**Restful API là những API sử dụng tiêu chuẩn REST.**

**Tiêu chuẩn REST là quy định cách sử dụng các HTTP method (delete, get, post, put…)**

**Các loại method:**

* **Get**
* **Post**
* **Put**
* **delete**

**Các loại HTTP status:**

* **Đầu 100 là đầu information response.**
* **Đầu 200 là đầu thành công.**
* **Đầu 300 là đầu redirect.**
* **Đầu 400 là client error - bad syntax or cannot be fulfilled.**
* **Đầu 500 là server error - server failed to fulfill.**

**10/ Environment variable là gì? Chức năng?**

Là biến dùng cho những môi trường nhau. Vd như API key, link to the backend, AWT secret key. Lưu trong .env

Dùng là “PROCESS.env.tên-biến”

Thường dùng gitignore để ignore những biến đó.

Package: dotenv

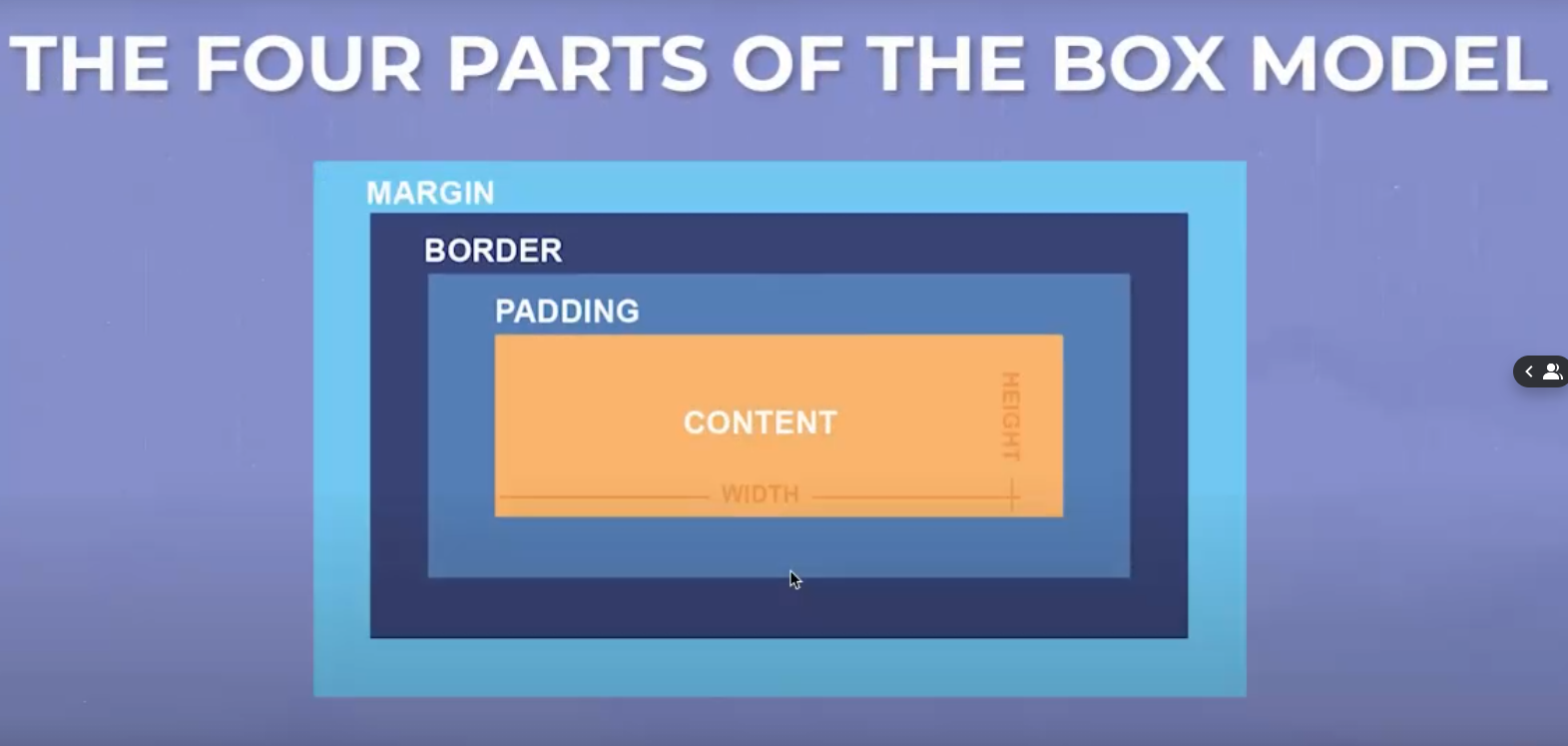
**11/ Làm sao để bảo mật user - password**

Trong node server thì dùng bcrypt.

* Để hash password thì dùng bcrypt.hash()
* Để check thì dùng bcrypt.compare()

**12/ What is the css box model?**

**The CSS Box Model is used to create a definition for the way the HTML elements are organized on the screen.**

****

**13/ What is the viewport?**

**CSS Viewport is defined as the visible area on a window screen which refers to the displays of the mobile devices.**

**14/ Viewport height ?**

**This unit is based on the height of the viewport. A value of 1vh is equal to 1% of the viewport height.**

**15/ Position Property**

**The position property specifies the type of positioning method used for an element (static, relative, absolute, fixed, or sticky(sticky không được hỗ trợ nhiều bởi các trình duyệt web).**

**Relative Postioning:** An element with position: relative, is positioned relative to its normal position.

Không phụ thuộc vào ai hết, phụ thuộc vị trí nó đang đứng ( lấy chính vị trị mình làm gốc toạ độ)

Có thể đè lên các thằng khác

Thay đổi vị trí của chính elements so với origin flow

Top, right, bottom, left chỉ hoạt động khi có position

**Absolute Position**: (An element with position absolute: is relative to the nearest positioned ancestor (instead of positioned relative to the viewport, liked fixed)

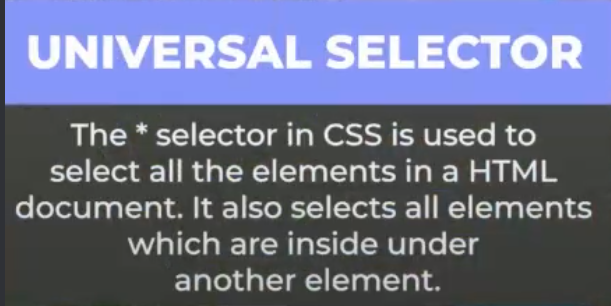
* phụ thuộc vào thẻ cha gần nhất có thuộc tính position
* lấy thẻ cha đó làm gốc toạ độ ( ko phân biệt thẻ cha đó là relative hay absolute)
* thẻ con không có width, height, thì top,R,L,B = 0 thì bao phủ hết lớp cha.
* Completely remove from DOM

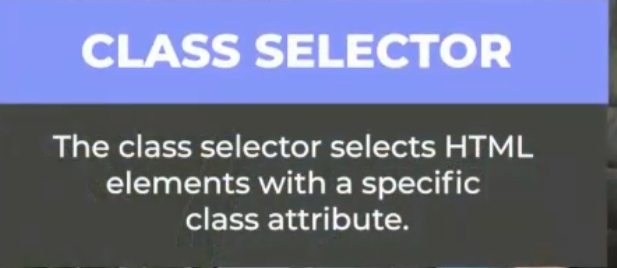
**Fixed Position** (An element with position: fixed is positioned relative to the viewport which means it always stays in the same even if the page is scrolled. The top, right, bottom, and the left properties are used to position the element)

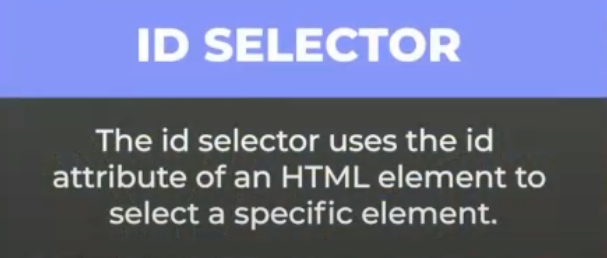
Sticky: (không được hỗ trợ bởi nhiều trang web): bám dính nghĩa là cái thẻ có thể đang nằm ở giữa trang nhưng khi kéo lên trên đầu, thì nó dính ở trên cùng (như header luôn)

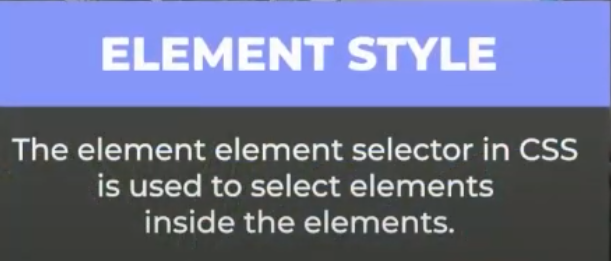
Static: là mặc định của các HTML elements (xuất hiện như các thứ tự trong file code HTML)

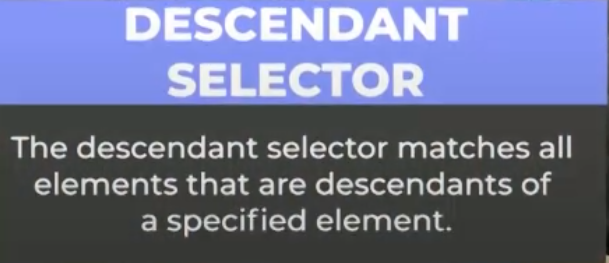
**16/ What are the different types of selectors?**

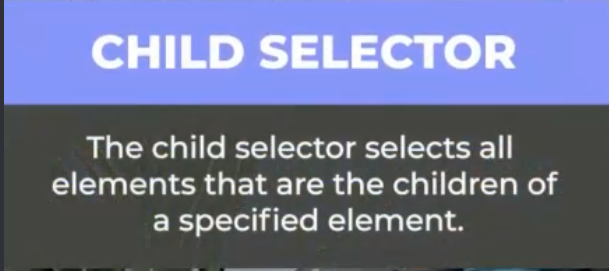
****

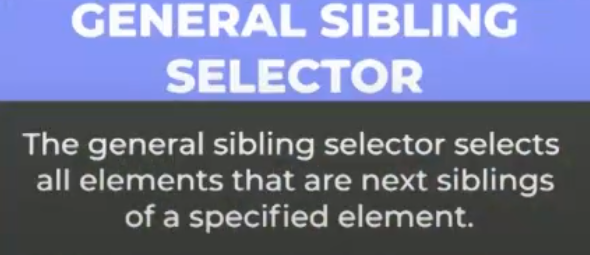
****

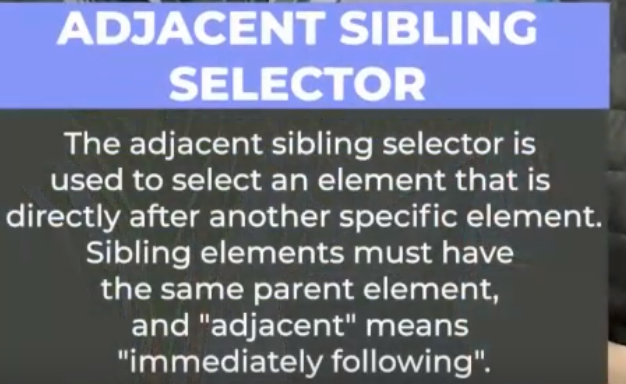
****

****

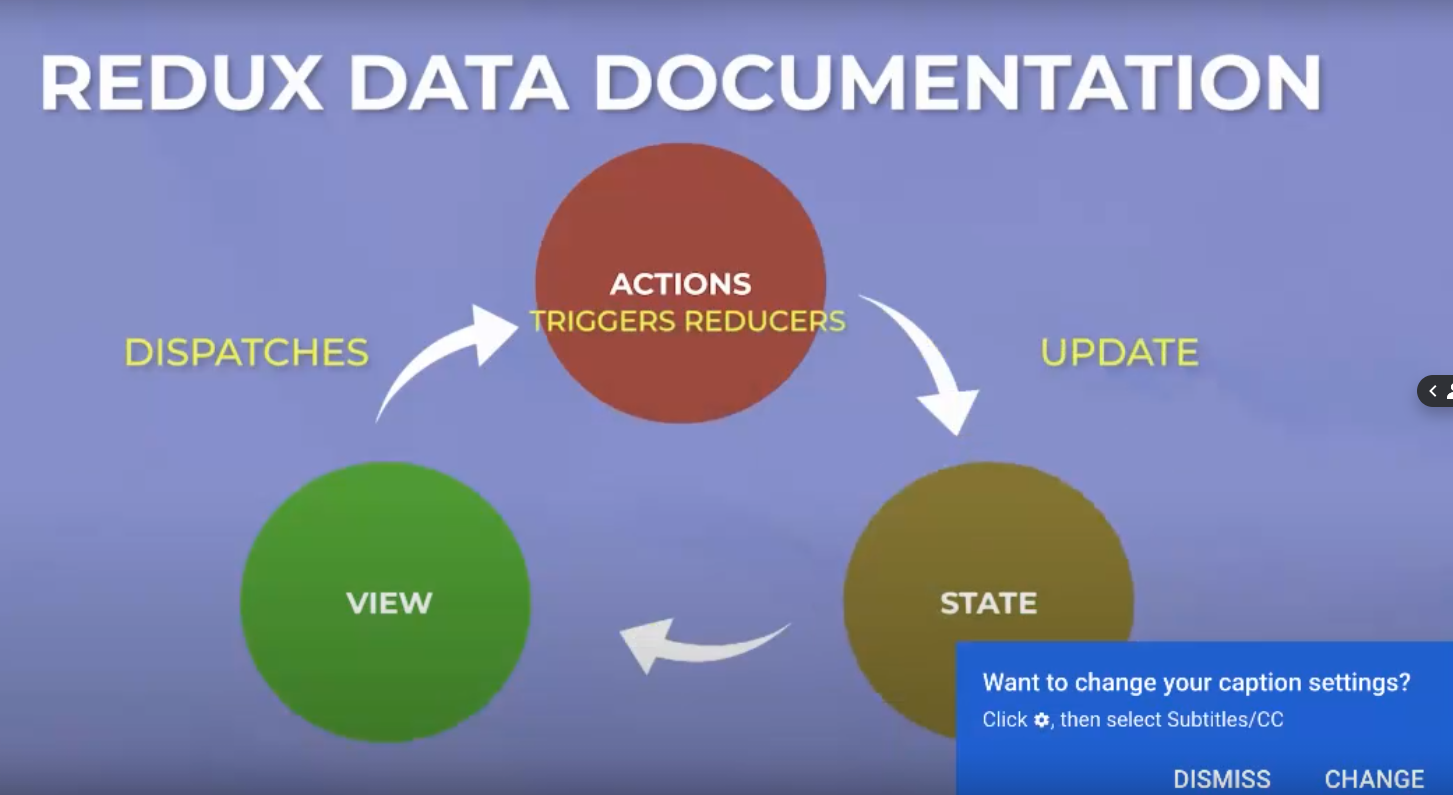
****

****

****

****

**17/ Redux data Documentation**

****

**18/ What are some times when you might not need Redux (What are some alternative Redux)?**

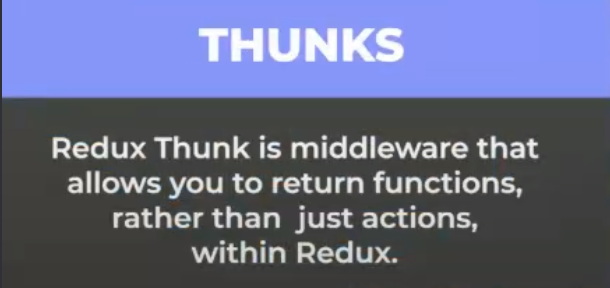
When you not need Redux:

If you dont have any global state that needs to be shared across your entire application

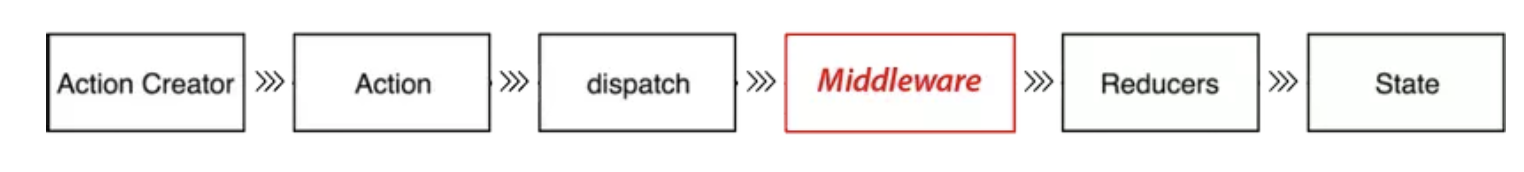
Alternative for Redux:

* React’s Context API
* Redux Toolkit

**19/ What is Thunks?**

****

Quay lại với Thunk, Redux Thunk cho phép trả về các Action là một function thay vì chỉ là một PJO, nó đóng vai trò là một Middleware được đặt trước thời điểm reducer nhận request để nhận biết các action có trả về một PJO hay không, nếu đó là một PJO, Thunk sẽ chuyển action đó đến Reducer như thường lệ, nếu action trả về là một function, Redux Thunk sẽ "chặn" action đó lại và đợi cho đến khi một lệnh asynchronous nào đó trong function hoàn tất và trả về kết quả (như giá trị response ở trên). Đến đây chúng ta đã nhận được một PJO và Redux Thunk sẽ cho action này đến Reducer như bình thường. Do vậy chúng ta có sơ đồ như sau:



**17) side effects**

**thuật ngữ chung của nghành phần mềm, khi nói về nói về một tác động bên ngoài làm cho data của phần mềm thay đổi (call API, set state, update DOM)**

**18) pure component?**

**19) Cách để empty của array?**

**Cách 1:**

| A = []; |
| --- |

Cách 2:

| A.length =0 |
| --- |

**Cách 3:**

| a.splice(0, a.length) |
| --- |

**Cách 4:**

| while(a.length>0){ a.pop() } |
| --- |

**20) --save --dev (-D)**

is used to save the package for development purposes.

Lưu trong dev-dependencies trong package.json

- - save

Thì lưu vô dependencies nhưng từ npm 5 trờ lên thì không cần làm vậy nữa r

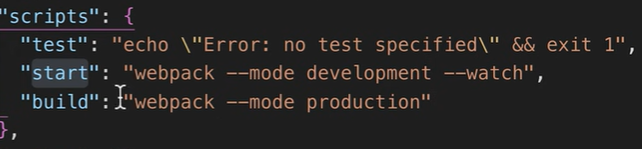
21) Babel

Babel là một công cụ chuyển đổi mã lệnh JavaScript hay JavaScript transpiler, được dùng với mục đích chuyển đổi mã lệnh JavaScript được viết dựa trên tiêu chuẩn ECMAScript phiên bản mới về phiên bản cũ hơn trước đó.

do JavaScript chủ yếu được chạy trên môi trường trình duyệt và mỗi trình duyệt khác nhau như Chrome, Firefox, Internet Explore, Safari... có những quy định riêng để viết JavaScript. Điều này dẫn đến không chỉ có duy nhât một quy định (ngôn ngữ) JavaScript và nếu như bạn code tuân theo "ngôn ngữ" JavaScript của một trình duyệt thì khi chạy trên trình duyệt khác hoàn toàn có khả năng code của bạn sẽ không chạy như ý muốn.

* Babel-core: Chuyển đổi ES6 về ES5
* Babel-loader: Cho phép chuyển các file JS sử dụng Babel & webpack
* Babel-preset-env: giúp sử dụng JS mới nhất trên nhiều môi trường khác nhau, chuyển es6, es7,về es5
* Babel-preset-react: hỗ trợ chuyển đổi jsx về js

# 22) Package.json



Npm start thì nó sẽ tự động gọi cái thằng nằm trong package.json

Ngoại trừ npm start, tất cả các thằng còn lại đều phải npm **run** build, npm run test

# 23) NPX create-react-app

Npx sẽ kiểm tra trong máy có thư viện đó chưa,

Nếu chưa, nó tự lên npmjs để install

# 24) package-lock.json == yarn.lock

Được tạo ra để quản lý hết tất cả thư viện (dependencies) kể cả thư viện trong thư viện

Package-lock sẽ được dùng để cài đặt cái modules do nó chỉ định rõ phiên bản

Vì dụ 5 modules cài cùng dùng chung 1 thư viện khác dependences con, thì package-lock,json sẽ coi và cài cái đó 1 lần thôi thay vì 5 lần → increase performance

# **25) Yarn vs npm**

NPM: install tuần tự từng dòng 1

Yarn: cài song song được, khi cài lại node modules còn nhanh hơn yarn còn lưu node modules vô 1 chỗ nào đó trong máy (cache), trước khi cài lại kt xem nó có chưa, có rồi thì lấy vô thôi

REF: <https://www.sitepoint.com/yarn-vs-npm/>

# **26) Toan tu ??**

a ?? []

a phải là null hoặc undefined thì mới lấy cái sau

KHÁC với ||

a || []

a là falsy ( 0, null, NAN, ‘’) thì mới lấy cái sau

# 27) JSON là gì?

* là 1 định dạng dữ liệu (không phải kiểu dữ liệu) “chuỗi” string
* var json = ‘1’, var json = ‘true', var json = ‘[“JS", “REACT"]’; var json = ‘{“name" : “Tuong"}’
* là viết tắt của Javascript Object Notation
* từ mảng muốn lưu vô localStorage thì JSON.stringify() ( JS -> JSON)
* từ localStorage về mảng lại thì JSON.parse() (JSON -> JS)

# 28) Cách đổi từ string sang number trong JS?

var price = ‘100’

* thêm dấu + đằng trước price
* dùng Number(price)
* Dung parseInt(price)

# 29) span vs div

Span: inline element

Div: block element

# 30) preventDefault

**preventDefault ()** dùng để hủy bỏ một phương thức. Nó ngăn không cho sự kiện thực hiện hành vi mặc định. Ví dụ: bạn có thể ngăn việc gửi biểu mẫu khi nhấp vào nút gửi như ví dụ dưới đây:

# 31) Prototype

Prototype là cơ chế mà các object trong javascript kế thừa các tính năng từ một object khác. Tất cả các object trong javascript đều có một prototype, và các object này kế thừa các thuộc tính (properties) cũng như phương thức (methods) từ prototype của mình.

Bạn cần chú ý rằng bản thân prototype là một object trong JS, được gọi là prototype object (đối tượng prototype). Chúng ta cần biết điều này để tránh nhầm lẫn với thuộc tính prototype của function.

32) High order function

33) High order component

34) Currying

<https://itguru.vn/blog/50-cau-hoi-phong-van-va-nguon-cau-hoi-danh-cho-reactjs-developer/>

<https://itguru.vn/blog/cau-hoi-phong-van-javascript-cung-huong-dan-tra-loi/>

<https://itguru.vn/blog/50-cau-hoi-phong-van-ve-giai-thuat-va-cau-truc-du-lieu-danh-cho-lap-trinh-vien/>

# 35) Đô ưu tiên trong CSS

+Inline > internal > external

+Nếu cùng trong internal thì

#id > . class > tag (h1)

+Xếp theo thang điểm thì

Inline 1000

Id 100

Class 10

Tag 1

→Vì thế nên #heading\_id.heading\_class = 110 > #heading = 100

+Nếu điểm bằng nhau thì thằng ghi sau thắng

# 36) Đơn vị trong CSS

px, %, rem, em

* % phụ thuộc vào thẻ **chứa nó**
* Rem phụ thuộc vào html font-size ( nếu không có thì 1rem bằng 16px)
* Em phụ thuộc vào thằng gần nhất có font-size
* Vh,vw phụ thuộc vào màn hình trình duyệt

# 37) Box-sizing: border-box

Padding và border sẽ làm element đó bự ra không theo width vs height ban đầu

Nên dùng box-sizing: border-box thì nó sẽ điều chỉnh theo width vs height theo padding và border để giữ nguyên được kích thước ban đầu

# 38) ES5 vs ES6

Arrow function

Template String

Let+ const

Enhanced Object Literal

Classes

Destructuring

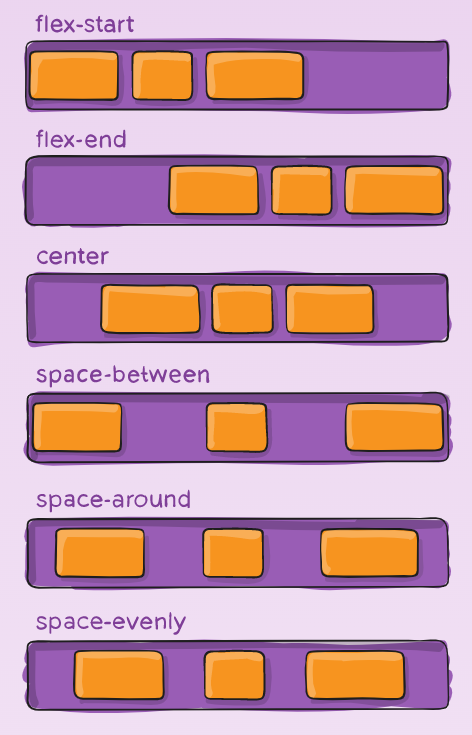
Promises

Async Await

# 39) Justify-content:

Align item theo main axis của box, display: flex, flex-direction: row

# 



**Algorithm**

# 40) Merge Sort

Là thuật toán chia để trị, chia nhỏ các phần tử ra rồi sắp xếp chúng lại khi ta merge

Cho 1 mảng, rồi cứ chia đôi, chia đến khi mỗi mảng là 1 phần từ

Bắt đầu merge lại, so sánh merge

Cách merge thì hai mảng đã sắp xếp, cứ lấy hai thằng đầu mỗi mảng, thằng nào bé hơn cho vô mảng trước, rồi move tiếp, rồi so sánh tiếp

Độ phức tạp: O(nlogn) ( cả ba trường hợp best worst average)

# 41) Heap sort

Từ 1 arr , ta phải build 1 cái heap từ cái arr đó, sau đó ta cần biến cái heap đó thành MAX HEAP ( root node index 0 là lớn nhất hoặc bằng các children node)

Swap thằng đầu và cuối node sau đó remove last node ra khỏi heap

Tiếp tục làm như vậy tới node cuối cùng

Only 1 one element trong heap thì end

Độ phức tạp: O(nlogn) ( cả ba trường hợp best worst average)

# 42) Quick sort

Đệ quy và chia để trị

Tìm 1 element gọi là Pivot, cái mà chia arr thành 2 phần,

Mà phần bên trái sẽ bao gồm những phần từ nhỏ hơn pivot,

Phần bên phải là những phần tử lớn hơn pivot

Mình sẽ đệ quy theo 3 bước sau

1. Mang pivot đúng vị trí của nó, phần trái bé, phần phải lớn
2. Quick sort phần bên trai
3. Quick sort phần bên phải

Độ phức tạp của thuật toán O(nlogn)

Best: O(nlogn)

Wrost: O(n^2)

# 43) Insertion sort

Này chỉ là 2 vòng lặp lồng nhau

B1:

B2: chia 2 mảng 1 bên là sorted , bên phải là unsorted

B3: chọn 1 số thường là số thứ 2 ( index 1)

So sánh với từng thằng với sorted arr , nếu nó mình chọn bé hơn thì dịch mấy thằng kia lên 1 vị trí, rồi thế nó vào vị trí đúng

B4: chọn market là thằng tiếp

Trường hợp: tốt nhất là khi mảng đã được sorted rồi, có thếm 1 thằng mới vô

O(n)

Worst case: O(n^2)

Average: O(n^2)

# 44) Bubble sort:

Đẩy thằng lớn nhất về cuối cùng

Dùng 2 vòng lặp

Swap trong vòng lặp thứ 2

void bubbleSort(int arr[], int n)

{

int i, j;

for (i = 0; i < n-1; i++)

// Last i elements are already in place

for (j = 0; j < n-i-1; j++)

if (arr[j] > arr[j+1])

swap(&arr[j], &arr[j+1]);

}

# 45) Selection sort

Đưa thằng bé nhất lên đầu, tìm bé nhất rồi swap nó lên đầu

Dùng 2 vòng lặp

Swap trong vòng thứ nhất

void selectionSort(int arr[], int n)

{

int i, j, min\_idx;

// One by one move boundary of unsorted subarray

for (i = 0; i < n-1; i++)

{

// Find the minimum element in unsorted array

min\_idx = i;

for (j = i+1; j < n; j++)

if (arr[j] < arr[min\_idx])

min\_idx = j;

// Swap the found minimum element with the first element

swap(&arr[min\_idx], &arr[i]);

}

}

# 46) What is React?

Controversial questions, in React community, many ppl think it is library, or framework.

React is a library for js to create UI interfaces

# 47) What is jsx?

Jsx stand for js XML

JSX allows us to write HTML in React.

JSX makes it easier to write and add HTML in React.

# 48) DOM

DOM là Document Object model

Dom represent cho html document mà mình thấy trên browser,

Provide method, property for us to interact with such getElementById, addEventListener, styles

# 49) Virtual DOM

# 50) What is React?

Controversial questions, in React community, many ppl think it is library, or framework.

React is a library for js to create UI interfaces

# 51) What is jsx?

Jsx stand for js XML

JSX allows us to write HTML in React.

JSX makes it easier to write and add HTML in React.

# 52) DOM

DOM là Document Object model

Dom represent cho html document mà mình thấy trên browser,

Provide method, property for us to interact with such getElementById, addEventListener, styles

# 53) Virtual DOM

**Adv:**

Virtual DOM là 1 dom ảo giống vs DOM thiệt nhưng nó là 1 light DOM

Khi có sự thay đổi gì trong virtual DOM, thì nó sẽ so sánh vs DOM thiệt rồi, update lên DOM.

Nó sẽ update UI mà không ảnh hưởng tới mấy phần ko thay đổi khác, => improve performance.

**DisAdv:**

Với những app nhỏ và simple thì virtual performance overhead. Tại vì app nhỏ, có thay đổi gì, thì đều update liền lên DOM.

# 54 ) Uncontrolled vs Controlled component (inputs)

Controlled:

Là component có state, setState lại trong component đó, tất cả đều thông qua state

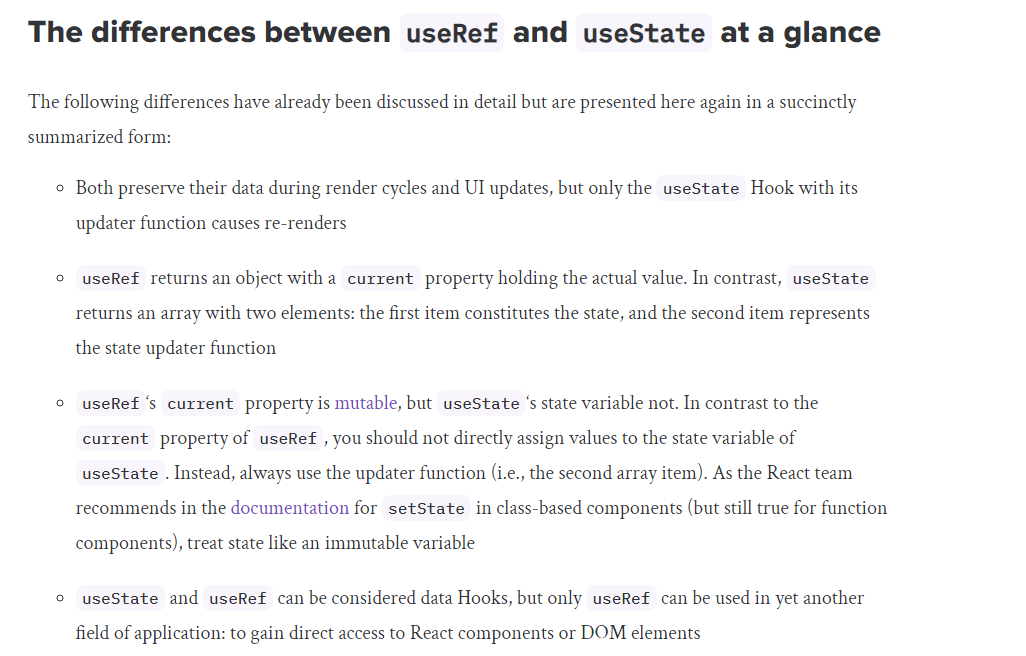
Uncontrolled:

Là get state của input direct từ DOM thông qua ref



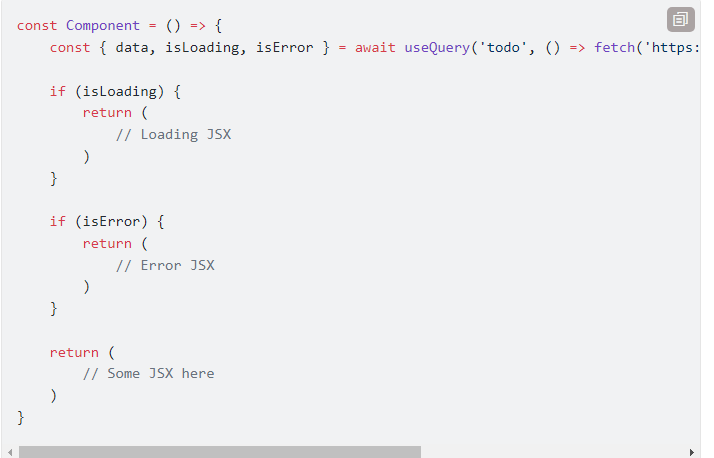
In React, an <input type="file" /> is always an uncontrolled component because its value can only be set by a user, and not programmatically.

# 55) useState vs UseRef



# 56) React-query

Là 1 thư viện để query data một cách nhanh chóng, không cần useEffect vs useSate như trước:

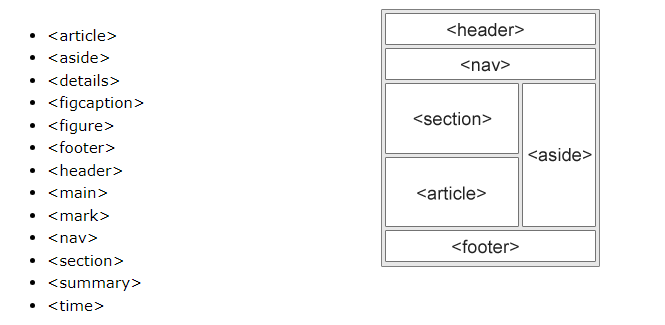


# 57) Semantic HTML

Semantic elements = elements with a meaning.

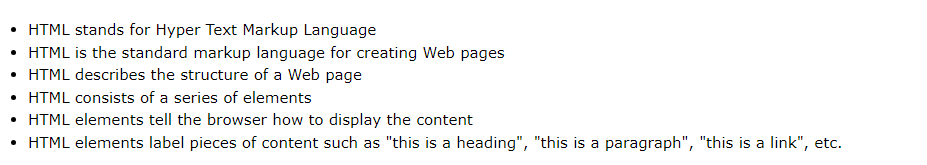
A semantic element clearly describes its meaning to both the browser and the developer.

Examples of semantic elements: <form>, <table>, and <article> - Clearly defines its content.

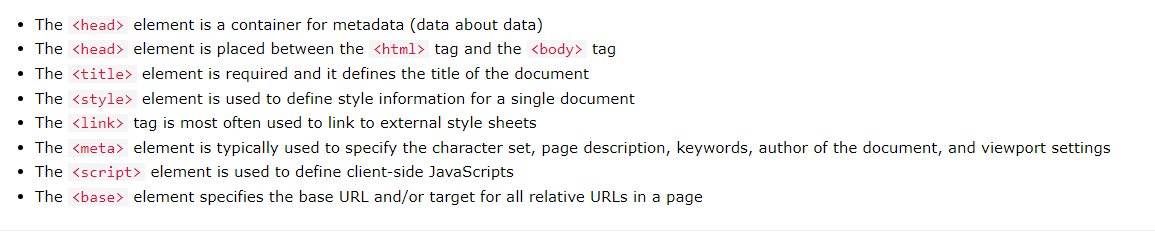


# 58) HTML

Hyper Text Mark Language



# 59) <Head>



# 60) <Body>

