**Chú ý quan trọng. Nếu là thuộc tính. Thì gắn =**

**Còn nếu là phương thức thì dùng ()**

alert(‘’) hiển thị 1 cái thông báo

Ảnh có chứa văn bản

Mô tả được tạo tự động

var: khai báo biến

confirm(‘’) gửi yêu cầu xác nhận

Ảnh có chứa văn bản

Mô tả được tạo tự động

prompt(‘’) hiển thị hộp thoại

Ảnh có chứa văn bản

Mô tả được tạo tự động

setTimeout() cho 1 đoạn code được chạy sau 1 khoảng thời gian, tính bằng mili giây

clearTimeout(timeout) dừng thực thi câu lệnh timeout khi mà click vào

setInterval() thực thi liên tục câu lệnh sau 1 khoảng thời gian

tiền tố: ++a

hậu tố: a++

* Sự khác nhau giữa == và ===

== so sánh về giá trị

=== vừa so sánh về giá trị, vừa so sánh về kiểu dữ liệu

Symbol là kiểu dữ liệu nguyên thủy của JS, cùng với string, number, boolean, null và undefined

* **String**

Tim vị trí chuỗi Ảnh có chứa văn bản

Mô tả được tạo tự động

* Cắt chuỗi. slice(x,y)
* Ghi đè chuỗi. replace(‘chuỗi cũ’,’chuỗi mới’)
* Ghi đề tất cả các chuỗi giống nhau replace(/chuỗi cũ/g, ‘chuỗi mới’)
* Chuyển đổi chuỗi sang chữ hoa. toUpperCase()
* Chuyển đổi sang chữ thường. toLowerCase()
* Loại bỏ kí tự khoảng trắng 2 đầu. trim()
* Tách chuỗi thành mảng. split(‘chuỗi’)
* Hiển thị vị trí kí tự. charAt(vị trí)
* **Nối mảng** sử dụng **concat** ví dụ: arr1.concat(arr2).
* **Chuyển string sang int float. Var x = parseFloat(‘chuỗi’)**
* **Số**
* (12.12).toFixed(x). lấy chính xác số thực sau đấu phẩy x số
* (12).toString(). Chuyển sang string
* Kiểm tra đây có phải là 1 số hữu hạn hay không. Number.isFinite()
* **Mảng**
* Nối các thành phần của mảng với 1 chuỗi. Array.join(‘’)
* Xóa phần tử cuối mảng. Array.pop()
* Xóa phần tử đầu mảng. Array.shift()
* Chèn phần tử vào đầu mảng. Array.unShift(‘Tên phần tử’)
* Xóa bất kì 1 vị trí trong mảng. Array.splice(vị trí xác định, số lượng phần tử xóa)
* Cách kiểm tra phần tử có thuộc 1 mảng hoặc 1 chuỗi hay ko, sử dụng includes
* Ảnh có chứa văn bản

  Mô tả được tạo tự động
* Cắt phần tử. slice(vị trí thứ nhất, vị trí thứ 2)
* **Nối mảng**Ảnh có chứa văn bản

  Mô tả được tạo tự động
* **Object**
* Xóa key trong object. Sử dụng delete.

delete Obj.key

* Có 2 cách để lấy giá trị của key trong object
* var obj = {
* name: 'Do',
* age: 18
* }
* console.log(obj['name']);
* console.log(boj.name);

* Thêm key vào 1 obj.
* obj['name2'] = 'hihi';
* Constructor

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Prototype

Ảnh có chứa văn bản

Mô tả được tạo tự động

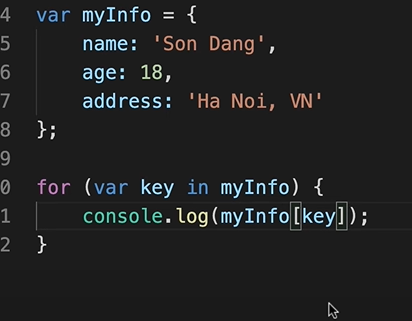
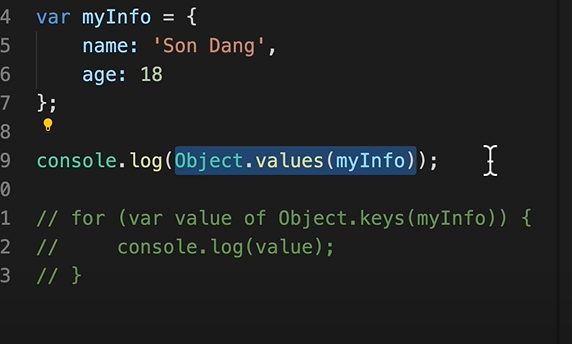
Prototype là 1 nguyên mẫu, dùng để thêm vào 1 obj

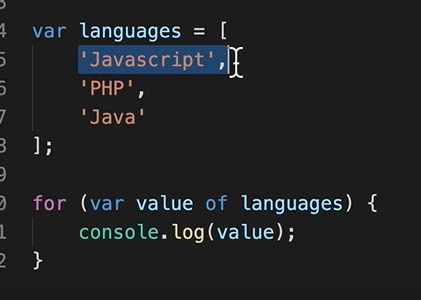
* Để trả về là 1 chuỗi string, sử dụng các keyword trong đó thì mình dùng
* return `${this.firstName} ${this.lastName}`;

**Math**

* random ngẫu nhiên chữ số. Math.floor(Math.random() \* số)

**Vòng lặp for in**

* **=> lấy dữ liệu từ 1 obj**
* ****
* ****
* **Lấy dữ liệu từ 1 mảng** dùng for of

****

**Vong lap**

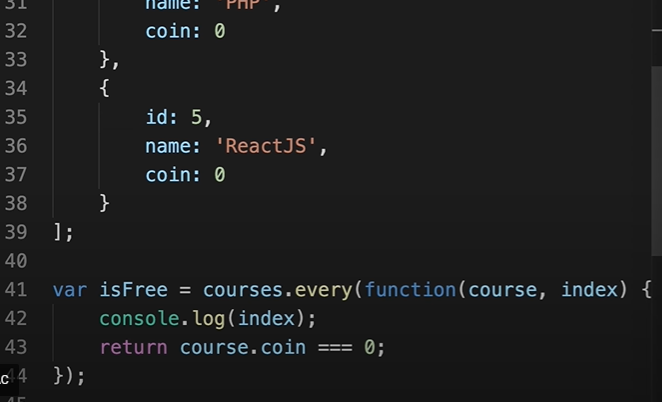
* Foreach duyệt qua từng phần tử của mảng

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

* **Filter:** trả về những phần tử trong mảng thỏa mãn thuộc tínhẢnh có chứa văn bản

  Mô tả được tạo tự động
* **Find:** trả về phần tử đầu tiên tìm thấy trong mảng
* **Every:** kiểm tra tất cả phần tử trong mảng nếu thỏa mãn

**Course.coin ===0 thì trả về true**

**Ở đây, course là thành phần trong mảng (obj), còn index là vị trí của course trong mảng**

* **Some:** chỉ cần 1 phần tử thỏa mãn thì nó trả về kết quả đúng

**Cách thứ 2, ko dùng hàm lồng vào nhau**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

* **Map:** trả về cái gì thì nó thay thế vào mảng giá trị như thế.
* **Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình

  Mô tả được tạo tự động**

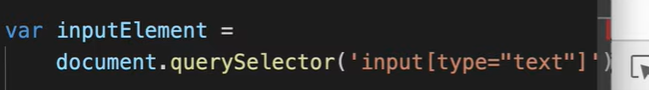
**HTML DOM**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

****

**<input type = “text”/>**

Lấy tất cả các phần tử sử dụng querySelectorAll(‘’)

setAttribute

Ảnh có chứa văn bản

Mô tả được tạo tự động

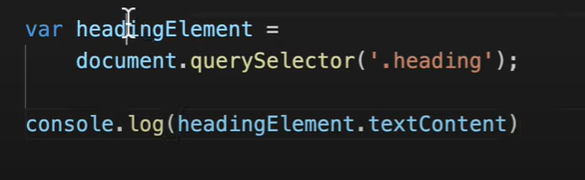
Lấy giá trị của attribute sử dụng getAtttribute

**textContent và innerText**

* **innerText:** chỉ lấy được nội dung nằm trong các elements

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

* **textContent:** lấy hết tất cả textNode ở trong thẻ được sử dụng

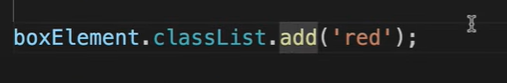
**Thêm thuộc tính cho CSS**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

**classList property**

* **add:** Thêm thuộc tính class

****

* **remove:** Xóa thuộc tính
* **contains:** kiểm tra thuộc tính này có thuộc class hay không

****

* **toggle:** có thì bỏ, không có thì thêm (thường dùng trong việc css chữ nhấp nháy, đóng mở menu)

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

* **Thêm click**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

**Lấy giá trị của thẻ input**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

**e.target là lấy cái elements của nó**

**PreventDefault:** Loại bỏ hành vi mặc định của trình duyệt trên thẻ HTML

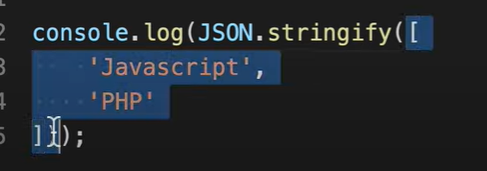
**StopPropagation:** Dùng tránh sự kiện nổi bọt. bên dưới là thẻ button lồng trong div

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Json: Là** 1 định dạng dữ liệu (chuỗi)

* Chuyển từ JSON qua dạng chuỗi, array, obj thì dùng JSON.parse(Tên biến)
* Ngược lại. dùng stringify. Ví dụ chuyển từ JS dạng mảng sang JSON



**Promise** là 1 khái niệm sinh ra để xử lí bất đồng bộ

Trước khi có promise, chúng ta thường sử dụng callback. Dùng callback sẽ xảy ra 1 vấn đề đó là callback hell, rất là khó nhìn, code rỗi. promise được sinh ra từ 1 phiên bản JS mới hơn, phiên bản ES6, giúp xử lí tình trạng callback hell, giúp chúng ta viết code dễ đọc, dễ hiểu.

Để tạo ra promise thì sử dụng từ khóa new.

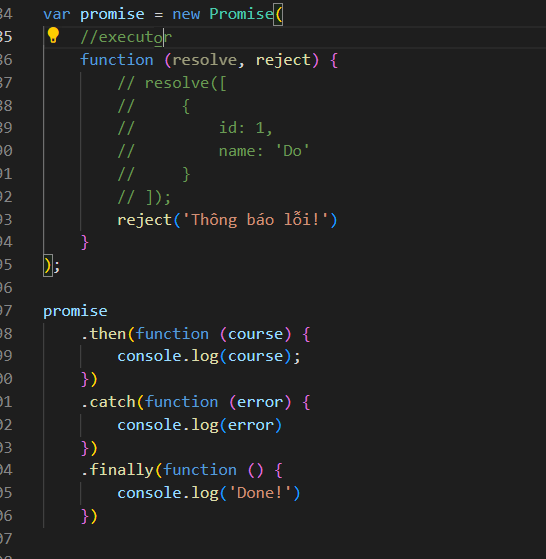
Trong đấy mình thêm 1 hàm executor với 2 tham số resolve và reject.

Resolve: thao tác xử lí thành công

Reject: thao tác xử lí thất bại

-sync: đồng bộ

-async: bất đồng bộ



**Chain:**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

**Promise.all**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

Ở đây, result ở dạng mảng

Result nhận từ resolve

Cả 2 promise 1 và 2 phải trả về resolve thì mới thực hiện thao tác then được

Chỉ mất 5s để thực hiện thao tác, không phải 7s

Nếu 1 trong 2 promise mà trả về reject thì nó thực hiện thao tác catch

**Json-server**

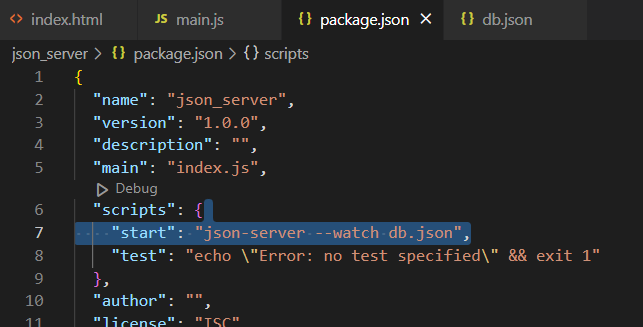
Tạo 1 thư mục có tên là json\_server

Tiếp theo mở terminal, sử dụng câu lệnh npm init

Tiếp theo: npm i json-server

Tạo file db.json ở tron thư mục node-module

Mở file pakeage.json. trong phần scripts tạo thêm



**Sử dụng postman**

Có các phương thức sau

Get: lấy dữ liệu ra

Post: thêm (body)

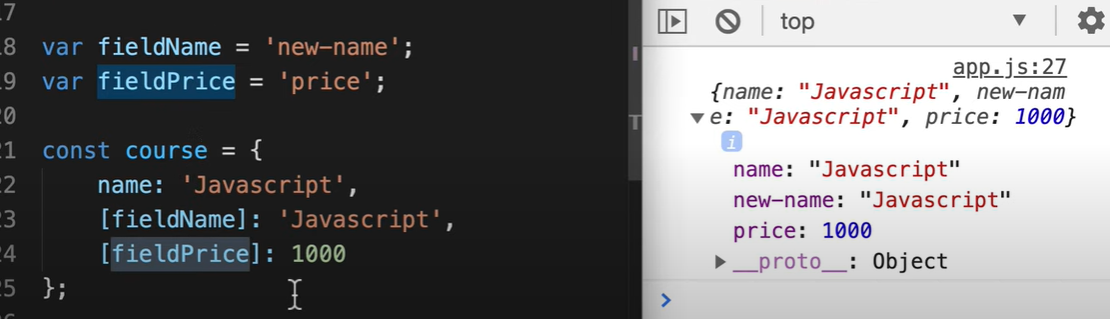
Put: chỉnh sửa

**Fetch**

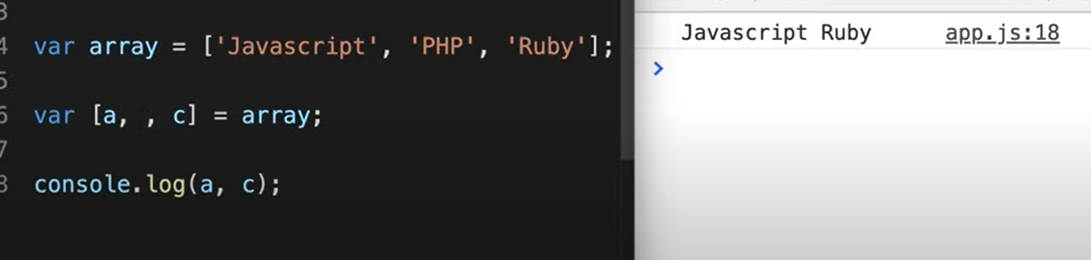
Làm nhiệm vụ gọi lên 1 API để lấy ra những nội dung lưu trữ ở BE và FE sẽ nhận những nội dung đó render ra, hiển thị ra trình duyệtẢnh có chứa văn bản

Mô tả được tạo tự động

**Enhanced object literals**Lấy value của 1 biến làm key trong object. Thêm []

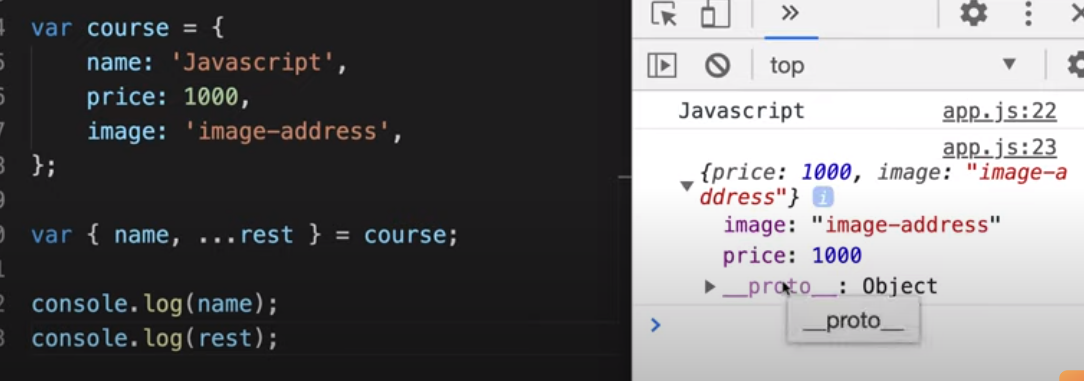


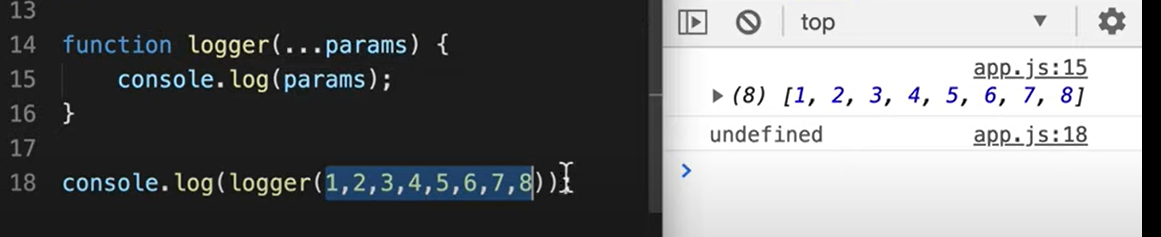
**Destructuring, rest**

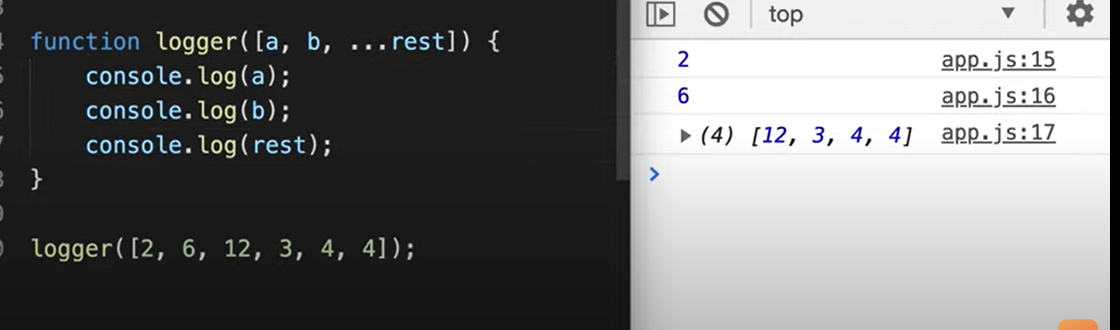
**Des:** Phân rã

Rest: chứa các phần tử còn lại ở trong mảng hay object

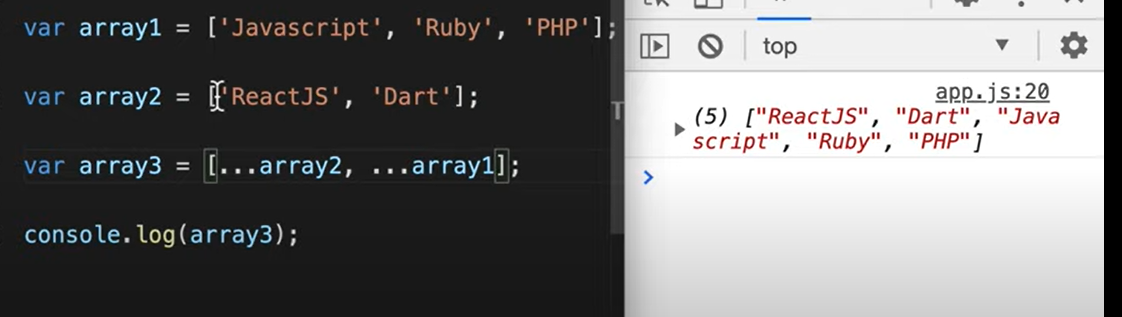
Rest được sử dụng cùng với des khi mình định nghĩa ra 1 cái tham số

****



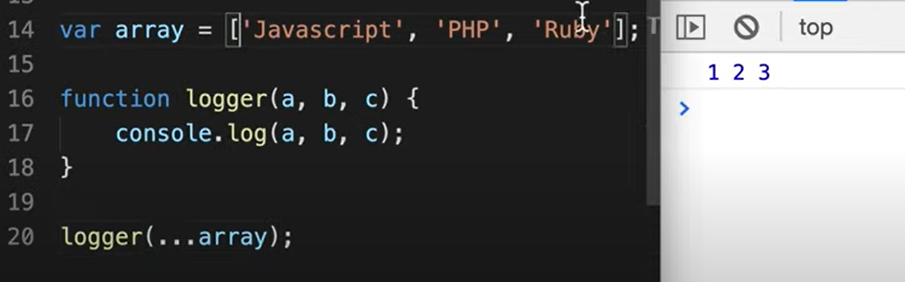


**Spread**



Có thể dùng trong trường hợp gộp 2 cái obj lại với nhau

* Toán tử spread nó cũng giống với rest nhưng cách sử dụng nó khác nhau
* Dùng spread nó sẽ loại bỏ cặp ngoặc đi



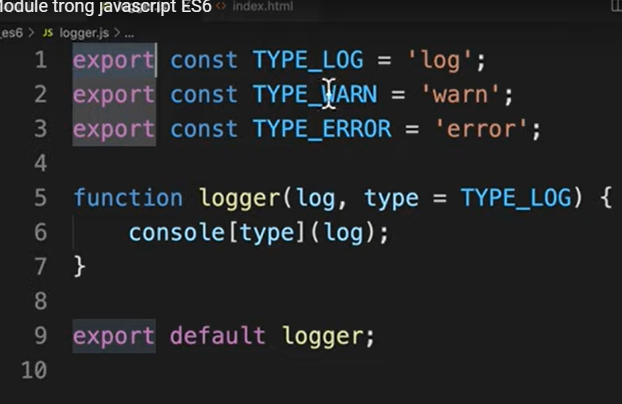
**Tagged temple literals**

Sử dụng 1 func và mình có thể gọi func đó qua tagged

**Modules:**

* Là tư tưởng khi chúng ta bóc tách ra xử lí 1 cái nghiệp vụ cụ thể ra 1 nơi riêng, thường thì chúng ta tách nó ra 1 file hoặc l 1 tập hợp
* Import/ export





* ở đây import logger là từ export default

còn nếu import {} là từ export const

* export \* as. Lấy hết tất cả export nhỏ nhỏ ra ngoài dưới dạng obj
* Trong cùng 1 package. Các lớp con có thể export ra ngoài cho nhau theo 2 cách

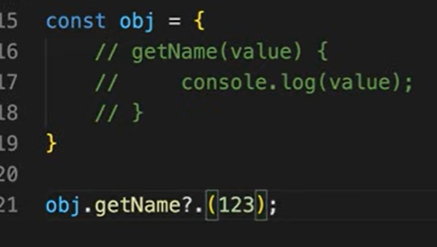
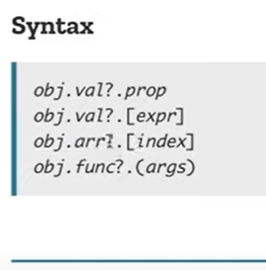
****

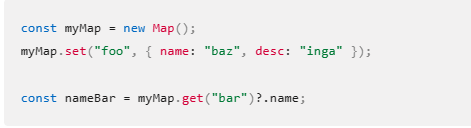
**Optional chaning**

Sử dụng ?.

Kiểm tra bên trong có tồn tại hay không







**Thực hiện thao tác cuộn màn hìnhẢnh có chứa văn bản

Mô tả được tạo tự động**

**defineProperty**

**audio.play (.pause) (.currentTime-🡪 .duration)**

**audio.ontimeupdate = func**

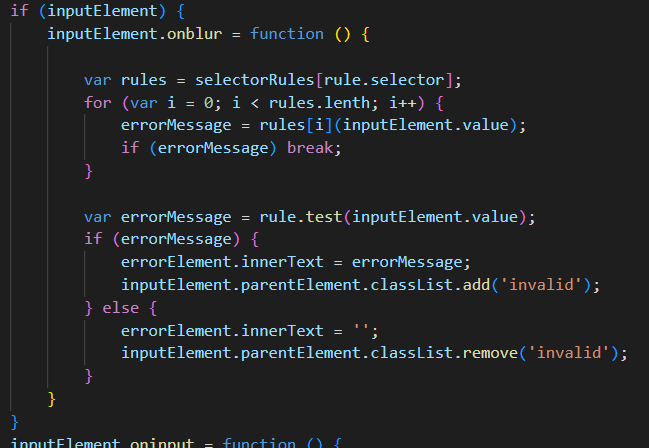
**Ảnh có chứa văn bản

Mô tả được tạo tự động**

**removeChild**

**appendChild**

**e.target.closest(‘tên lớp’) : xét tại lớp mình đang chỉ vào**

****

**onblur:** Khi click chuột ra ngoài

Ảnh có chứa văn bản

Mô tả được tạo tự động

Trim(): đếm các kí tự