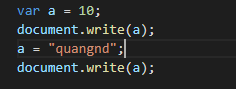
# Biến – Toán tử

## Biến



Kết quả:



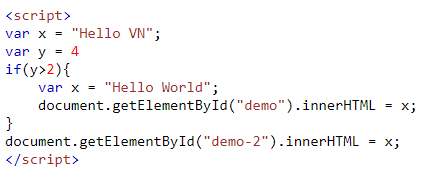
Biến trong phụ thuộc vào giá trị mà biến đó nhận tại thời điểm xử lý lệnh. Một biến có thể nhận nhiều kiểu dữ liệu khác nhau.

Biến toàn cục là biến bên ngoài hàm

Biến cục bộ là biến bên trong một hàm

### Var

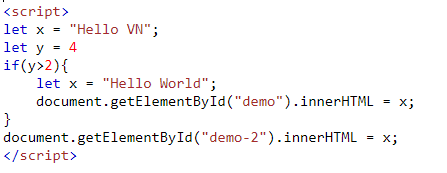
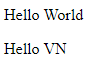
* Khi điều kiện xảy ra var sẽ thay đổi hoàn toàn biến được khai báo ban đầu.



* Với biến var ta sẽ không thể biết biến bị thay đổi khi nào nếu như ta có hàng triệu dòng code

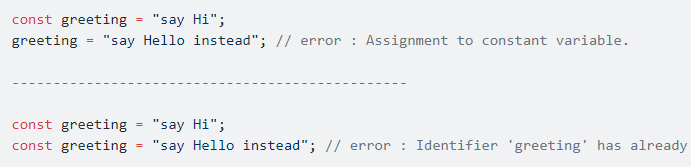
### Let

* Khi điều kiện xảy ra let sẽ chỉ cập nhật lại biến đó trong điều kiện thỏa mãn và khi thoát khỏi điều kiện nó không thể thay đổi hoàn toàn biến đó với giá trị ban đầu

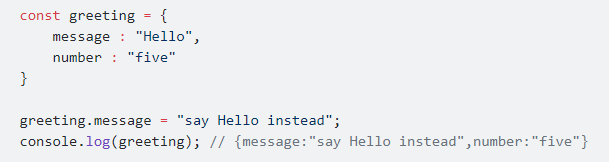
 

### Const

* Với kiểu biến là kiểu nguyên thủy thì ta không thể thay thế hay cập nhật giá trị mới



* Với kiểu biến là kiểu dữ liệu đối tượng thì ta chỉ có thể cập nhật giá trị cho thuộc tính của biến đó



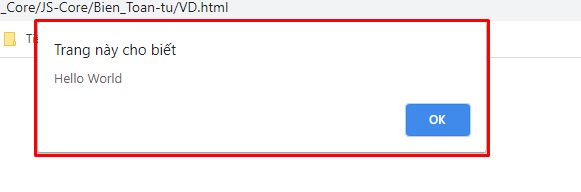
## Toán tử

Giống với cách khai báo toán tử ở các ngôn ngữ lập trình khác

# Các hàm cơ bản thường dùng

## Alert

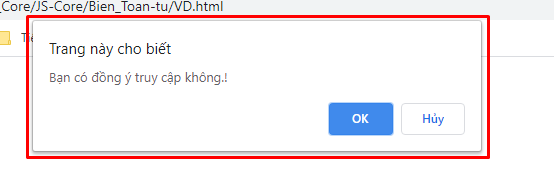




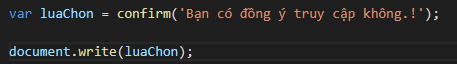
* Ứng dụng: Thông báo lỗi cho người dùng, thông báo cho người dùng chú ý hơn

## Confirm





* Ứng dụng: Thông báo cho người dùng lựa chọn
* Để lấy được thông tin lựa chọn của người dùng thông qua hàm confirm ta sẽ làm như sau:



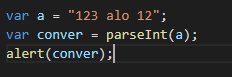
* Nếu chọn đồng ý thì trên tab website sẽ hiện thị là true ngược lại sẽ hiện false:

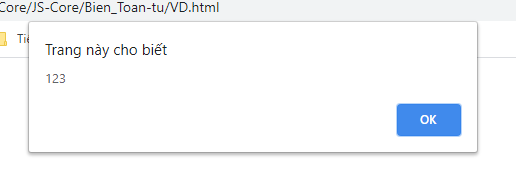


* Như vậy ta có thể dùng biến này để có thể xét các trường hợp của nó.

## ParseInt

* Cắt số nguyên đầu tiên của chuỗi



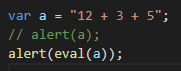


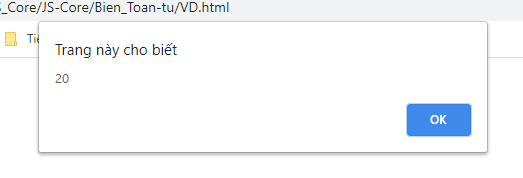
## ParseFloat

* Cũng giống như parseInt nhưng là dùng cho số thực

## Eval

* Biến chuỗi thành biểu thức toán học
* Eval có thể biến chuỗi thành các mã js

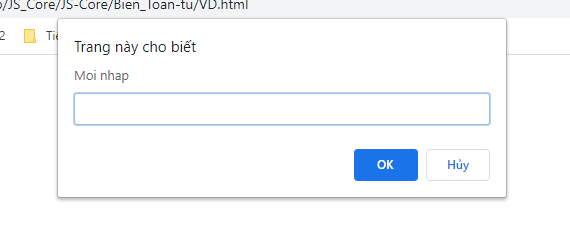




## Prompt

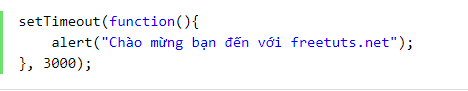
* Xuất hiện form cho phép người dùng nhập dữ liệu





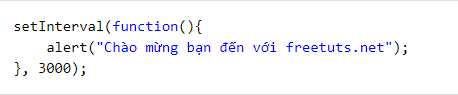
## Set timeout

* Cho thực hiện câu lệnh sau một khoảng thời gian được tùy chỉnh



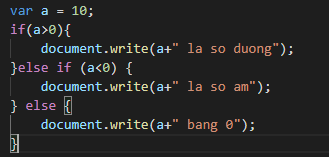
## Set interval

* Thực hiện câu lệnh liên tục và cách nhau một khoảng thời gian tùy chỉnh



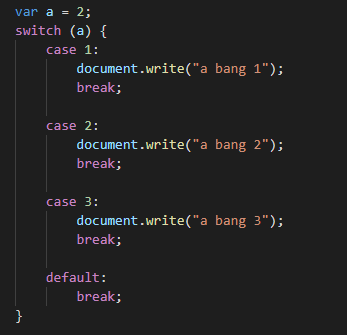
# Biểu thức điều kiện

## If else





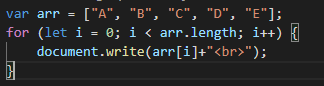
## Switch case





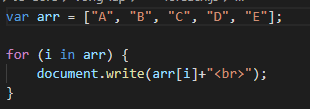
# Vòng lặp

## For

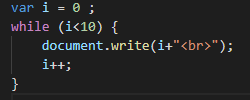




## Foreach

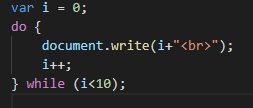
  


## While





## Do while





# Kiểu dữ liệu

## Kiểu chuỗi

Để ép 1 giá trị nào đó về kiểu chuỗi ta sử dụng: string.toString()

### Các hàm xử lý chuỗi

1. Length – Lấy độ dài của chuỗi
2. IndexOf – Tìm kiếm chuỗi

* Nếu tìm thấy thì nó sẽ trả về vị trí xuất hiện đầu tiên của chuỗi con còn không thì nó sẽ trả về -1



* Ngoài ra lastIndexOf() sẽ tìm chuỗi ở vị trí cuối cùng. Ngoài ra hàm search() cũng có thể tìm kiếm chuỗi con trong một chuỗi
* split(): Hàm cắt chuỗi

### Cắt chuỗi

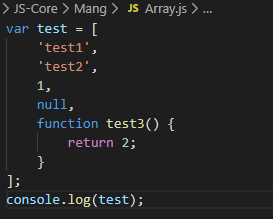
* slice(start, end)
* substring(start, end)
* substr(start, length)

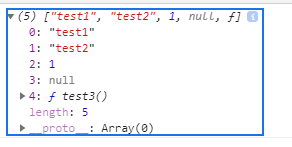
### Lặp chuỗi

# Mảng

* Mảng là một biến có nhiều giá trị.
* Giá trị trong một mảng có thể là tập hợp gồm nhiều kiểu dữ liệu
* Chỉ mục trong mảng sẽ được đánh tự động

## Tạo mảng





## Truy xuất mảng

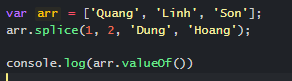




## Các hàm xử lý với mảng

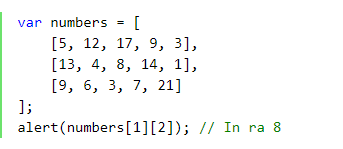
* array.valueOf() – Nối các phần tử lại với nhau và một chuỗi cách nhau bởi dấu phẩy
* array.push() – Thêm phần tử vào cuối mảng
* array.pop() – Xóa phần tử cuối mảng
* array.shift() – Xóa phần tử đầu tiên của mảng
* array.unshift() – Thêm phần tử vào vị trí đầu tiên
* array.splice() – Hàm thay thế phần tử trong mảng (Nên sử dụng để xóa phần tử ra khỏi mảng vì nó xóa cả vị trí của phần tử đó)

Cú pháp: splice(position\_add, num\_element\_remove, value1, value2, ...).



* array.sort() – Sắp xếp theo thứ tự abc
* array.reverse() – Đảo ngược sắp xếp
* array.concat() – Nối hai mảng với nhau
* array.slice() – Lấy một phần tử con trong mảng

## Mảng 2 chiều

* Khai báo: 

# Đối tượng

Khai báo: var Comment = {};

VD:

var keyEmail = 'email';

// Object

var myInfo = {

    name: 'Quang ND',

    age: '27',

    address: 'Ha Noi',

    [keyEmail]: 'quangnd512@gmail.com',

    // Khai báo function trong object

    getName: function() {

        return this.name;

    }

}

// Gọi đối tượng function

myInfo.getName() // object myInfo

// Đổi 1 giá trị trong object

myInfo.name = 'Quang ND 123'; //giá trị name trong object = Quang ND 123

// Sửa key

myInfo['age-name'] = '27';

// Lấy value

myInfo.name;

myInfo['name'];

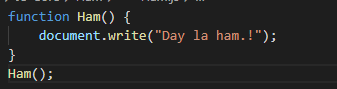
// xóa Key

delete myInfo.age

# Hàm

* Hàm là một khối mã và chỉ làm việc cụ thể

## Tạo hàm

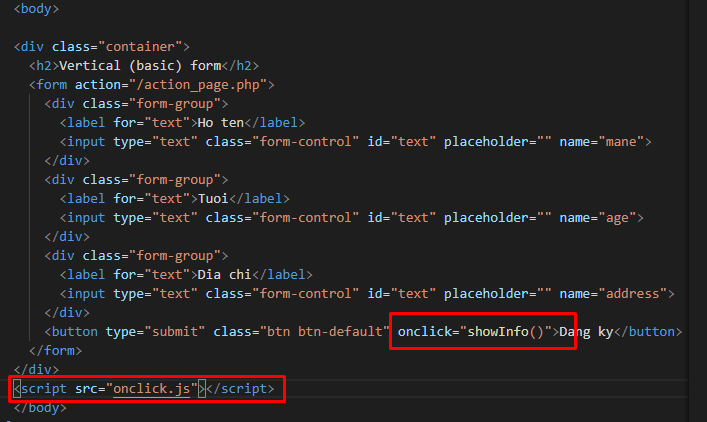


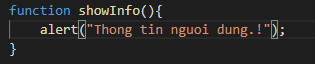


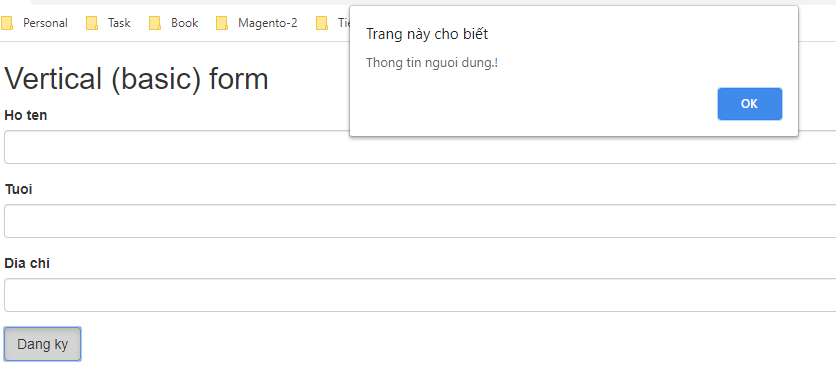
# Sự kiện

## Onclick

* Bắt sự kiện click chuột



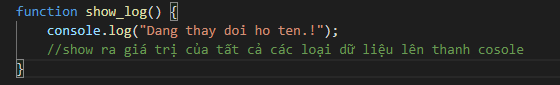


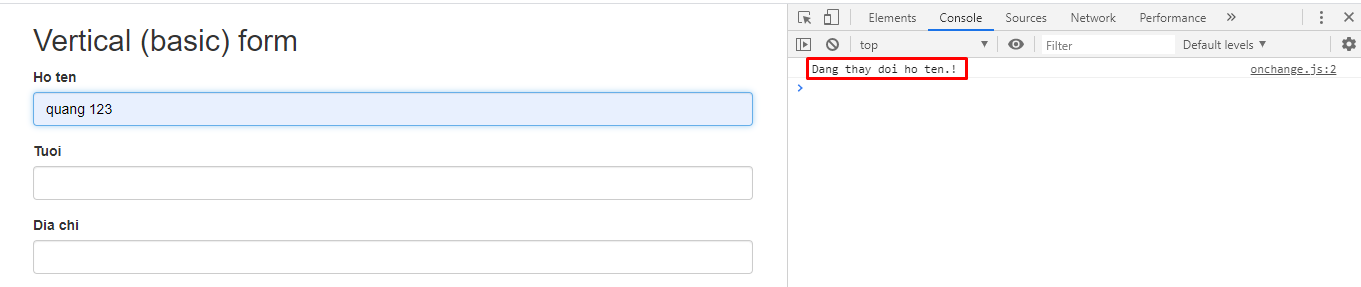


## Onchange

* Bắt sự kiện thay đổi giá trị textfield



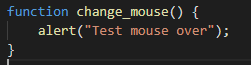
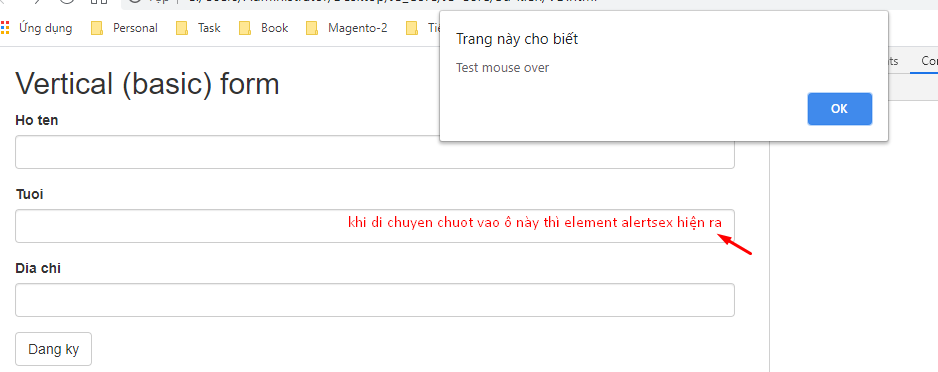




## Onmouseover

* Bắt sự kiện khi di chuyển chuột đến



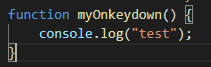
## Onmouseout

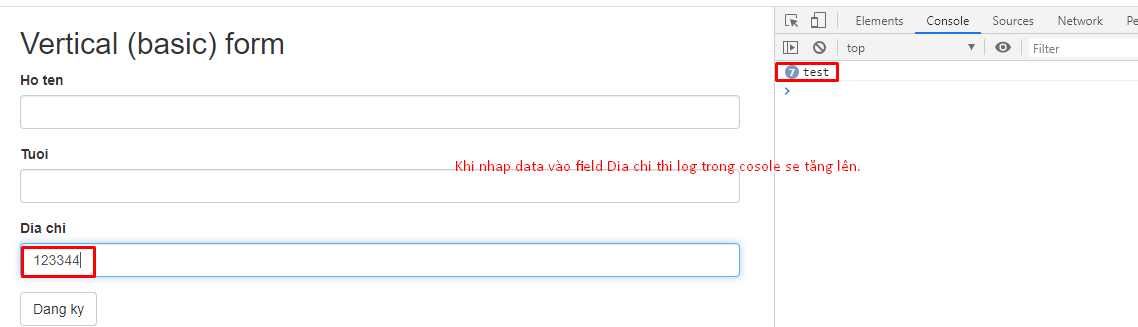
* Bắt sự kiện khi di chuyển chuột đi. Khai báo giống như Onmouseout

## Onkeydown

* Đếm số lượt click, thêm text vào field, …

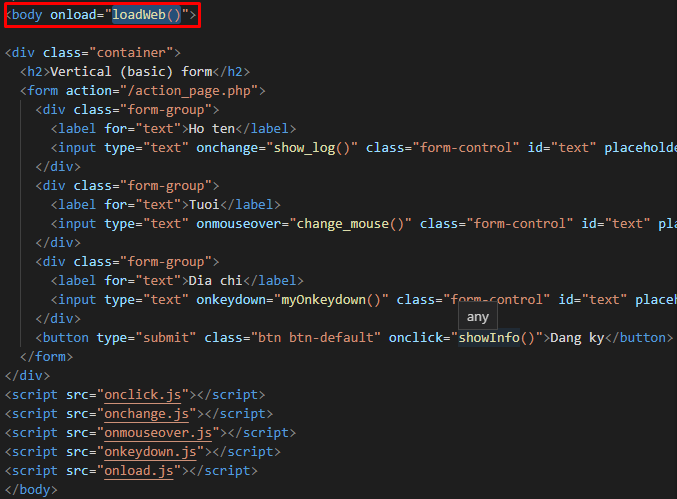


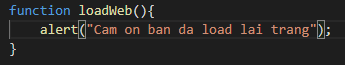


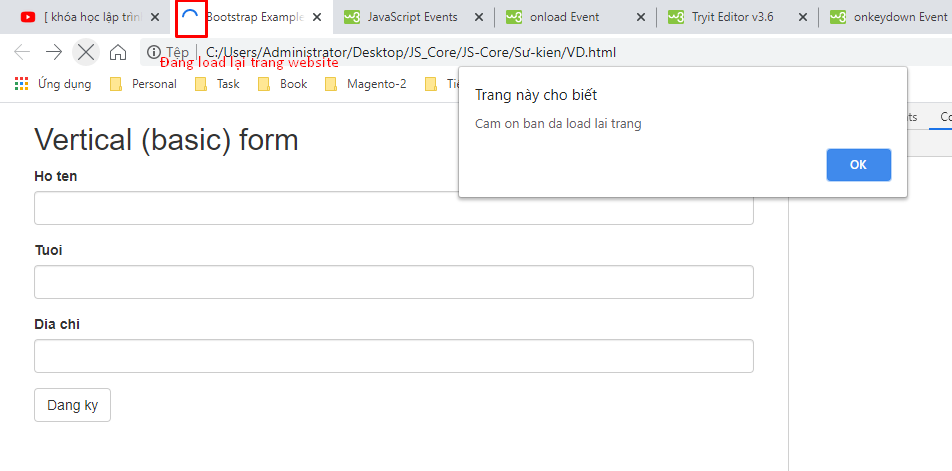


## Onload

* Sự kiện được thực hiện khi ta load lại trang website







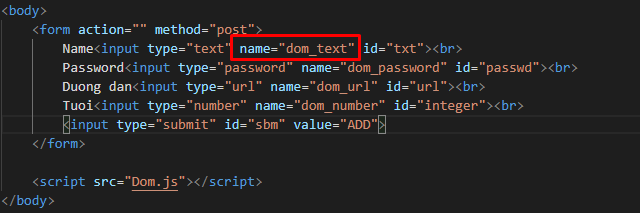
# DOM

* Nếu muốn truy xuất bất cứ yếu tố nào trong trang HTML thì ta dùng tới DOM trong Javascript

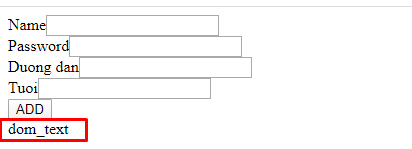
## Các thẻ tìm kiếm phần tử HTML

### Documment.getElementById(“id”)

* Tìm kiếm 1 element HTML theo id. Nó sẽ trả về giá trị là tên của thuộc tính đó

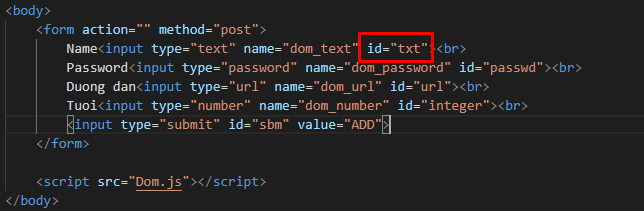




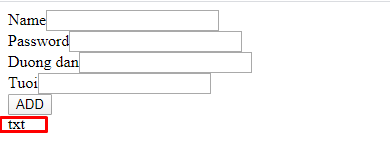


### Documment.getElementByTagName(“tag\_name”)

* Tìm kiếm các element được khai báo theo tên. Nó sẽ tìm kiếm tất cả các thẻ có cùng tên

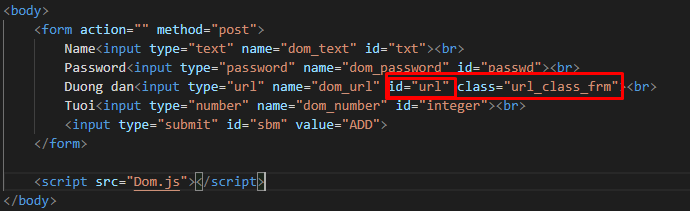




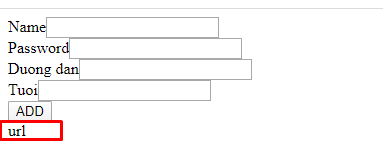


### Documment.getElementByClassName(“class\_name”)

* Tìm kiếm các element được khai báo theo tên class. Nó sẽ tìm kiếm tất cả các thẻ có cùng tên class



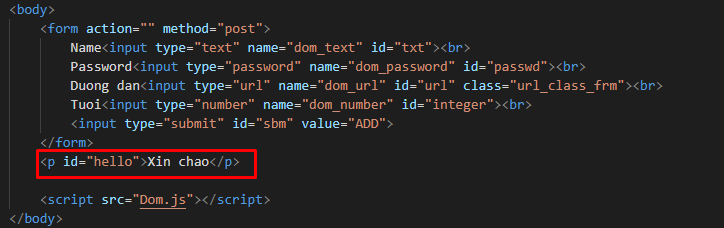




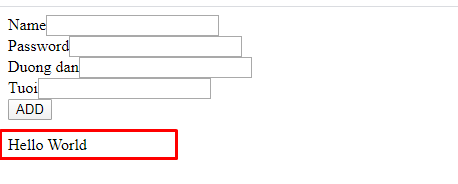
## Các thẻ thay đổi giá trị HTML

### innerHTML

* Thẻ này dùng để thay đổi giá trị của thẻ HTML

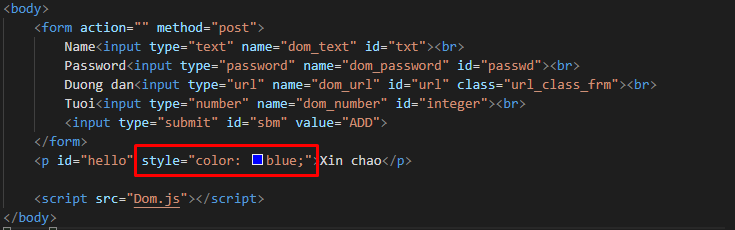




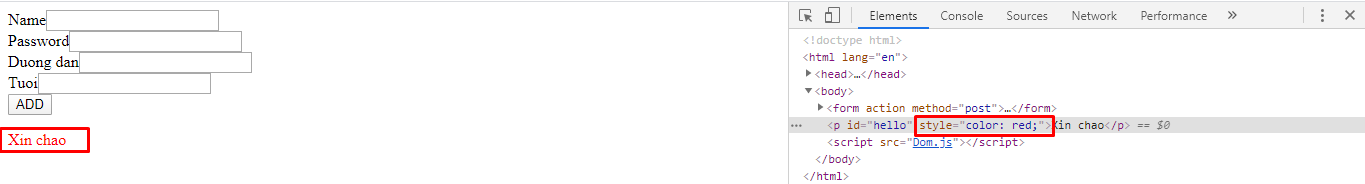


### Attribute

* Thẻ này dùng để thay đổi giá trị của thuộc tính

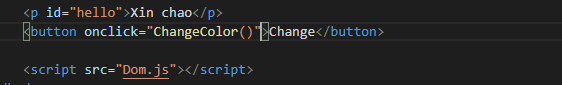


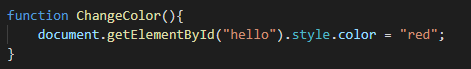




### Style.property

* Thẻ này dùng để thay đổi style của thẻ html





Trước khi nhấn buttom

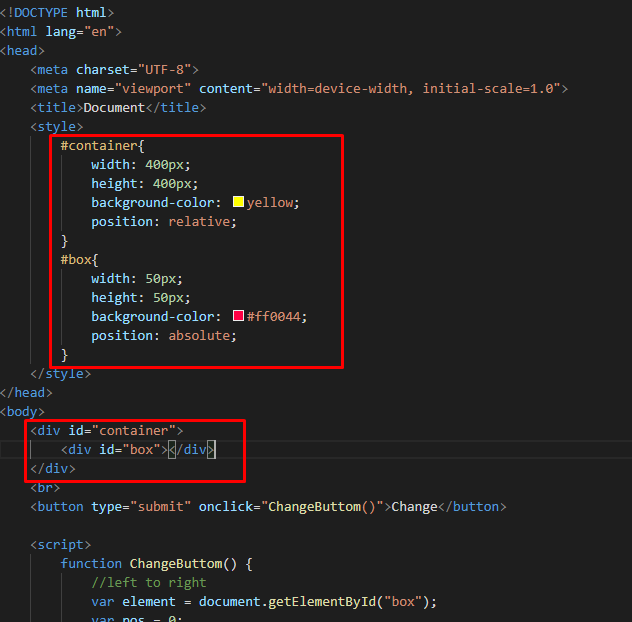


Sau khi nhấn buttom



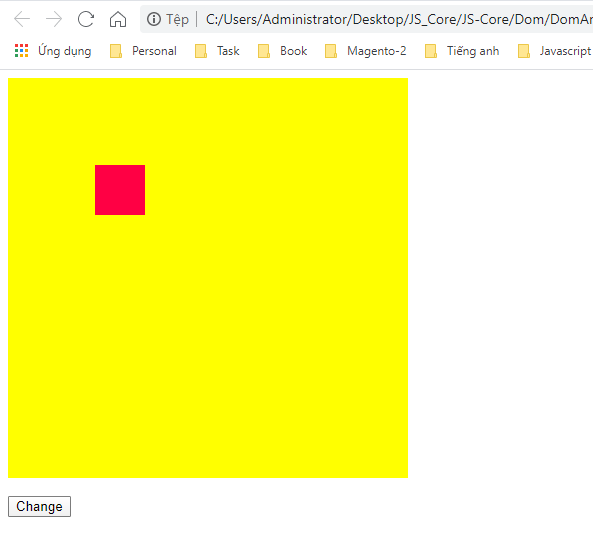
## Một vài ví dụ khác

### DOM Animations

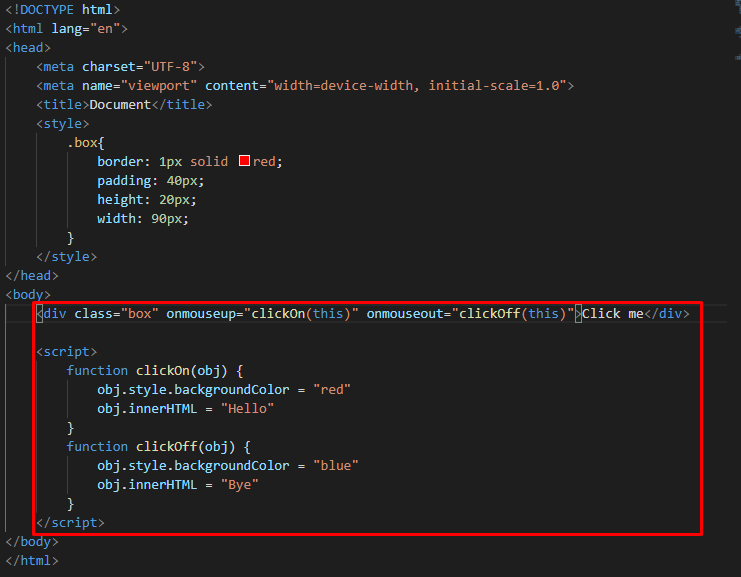




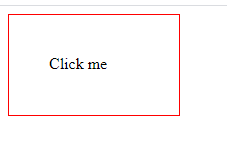
* Sau khi nhấn vào change thì box màu đỏ sẽ di chuyển từ góc trên bên trái xuống góc dưới bên phải



## Từ khóa this



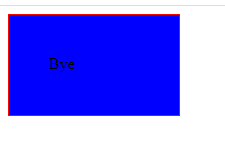
* Khi mới vào lần đầu tiên



* Khi click vào



* Khi chi chuột ra ngoài sau khi click



# Browser BOM

* BOM (Browser Object Model) là các đối tượng liên quan đến trình duyệt và mỗi một trình duyệt sẽ có những đối tượng khác nhau. Nên giữa các BOM với nhau sẽ không có tiêu chuẩn nào cả.

## JS Window

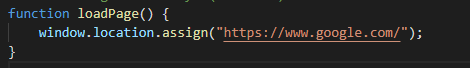
### Screen

* Screen.width, screen.height : Trả về giá trị chiều rộng, chiều cao màn hình của máy khách
* Screen.availWidth, Screen.availHeight : Trả về giá trị chiều rộng, chiều cao màn hình của máy khách khi đã trừ đi thanh taskbar

### Window.location

* Window.location.href : Hiển thị địa chỉ url
* Window.location.hostname : Hiển thị tên trang website
* Window.location.pathname : Hiển thị đường dẫn file của trang html
* Window.location.protocol : Hiển thị tên giao thức của website (http or https)
* Window.location.assign(“url”) : Nhúng đường dẫn load page





|  |  |
| --- | --- |
|  | Khi bấm vào buttom google nó sẽ chuyển sang trang google.com |
|  | |

### Window.history

### Window.navigator

### Window.alert

* Cách hoạt động giống với hàm alert()
* Ngoài ra còn có các hàm: window.confim(), window.prompt()

### Timing Event

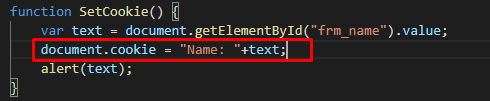
* Cho phép chúng ta thiết lập thời gian cho website
* Window.setTimeout(giaTri, time) : Xác định thời gian để giá trị được thực thi rồi dừng lại. Có hai biến được truyền vào hàm:
  + Giá trị: Giá trị được truyền vào để thực thi
  + Thời gian: Thời gian từ lúc chạy browse đến lúc thực thi câu lệnh
* Window.clearTimeout(giaTri) : Thực hiện dừng lệnh setTimeout\
* Window.setInterval(giaTri, time): Giống như setTimeout là cũng xác định thời gian để giá trị được thực thi nhưng sẽ lặp lại hành động sau một khoảng thời gian đã thiết lập
* Window.clearInterval (giaTri) : Tương tự như hàm clearTimeout nhưng được áp dụng để dừng hàm setInterval

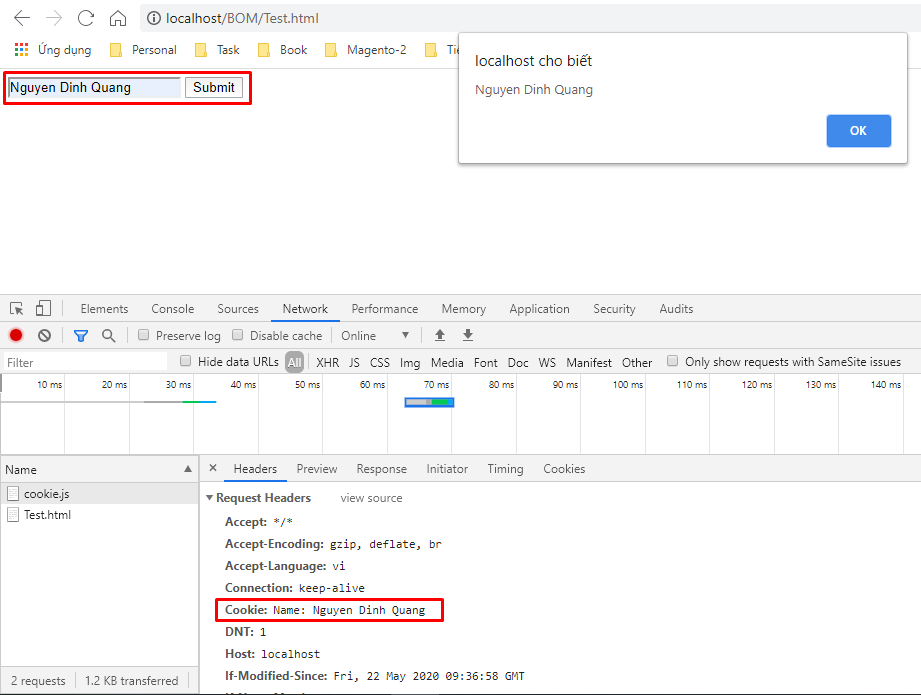
## Cookies

### Định nghĩa

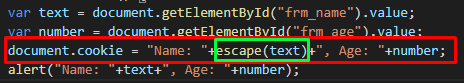
* Cookie cho phép lưu thông tin người dùng trong trang web
* Khi người dùng truy cập trang web thông tin của họ sẽ được lưu lại và khi họ quay trở lại vào lần sau thì cookie sẽ nhớ thông tin của họ
* Điều này giúp trang web được xử lý nhanh hơn và người dùng sẽ không phải khai báo thông tin của mình nhiều lần

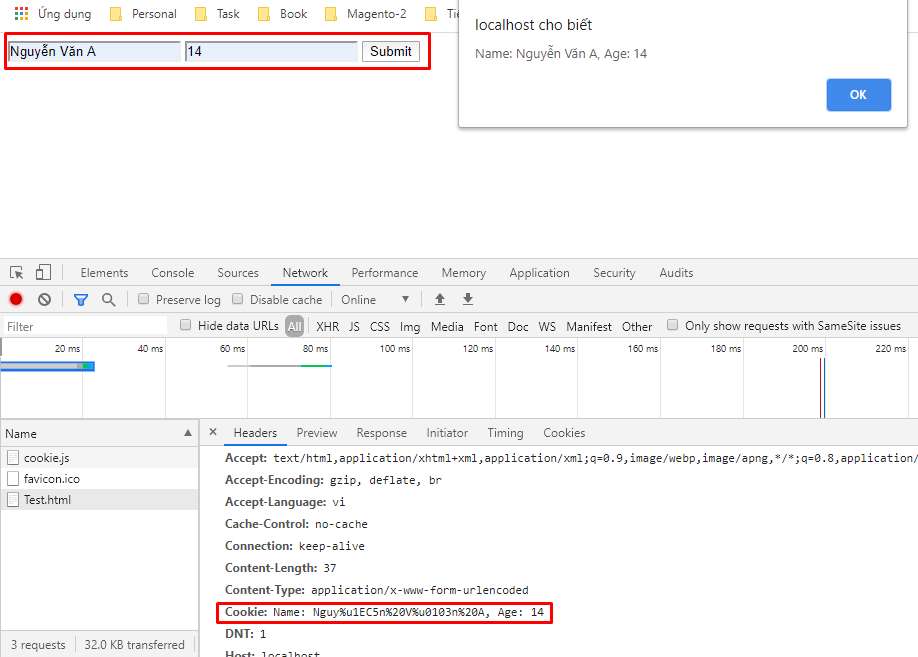
### Ghi cookie



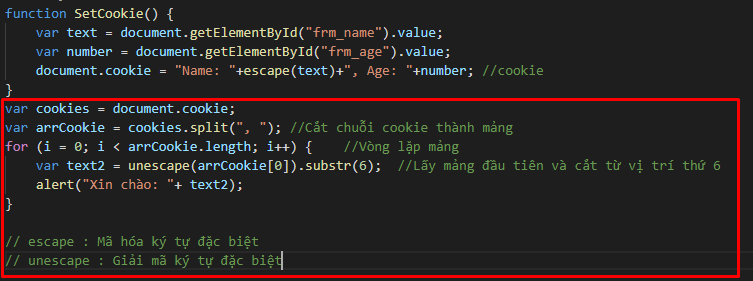


* Để lưu cookie những dữ liệu kiểu chữ tiếng việt hoặc khoảng trắng ta cần mã hóa chúng

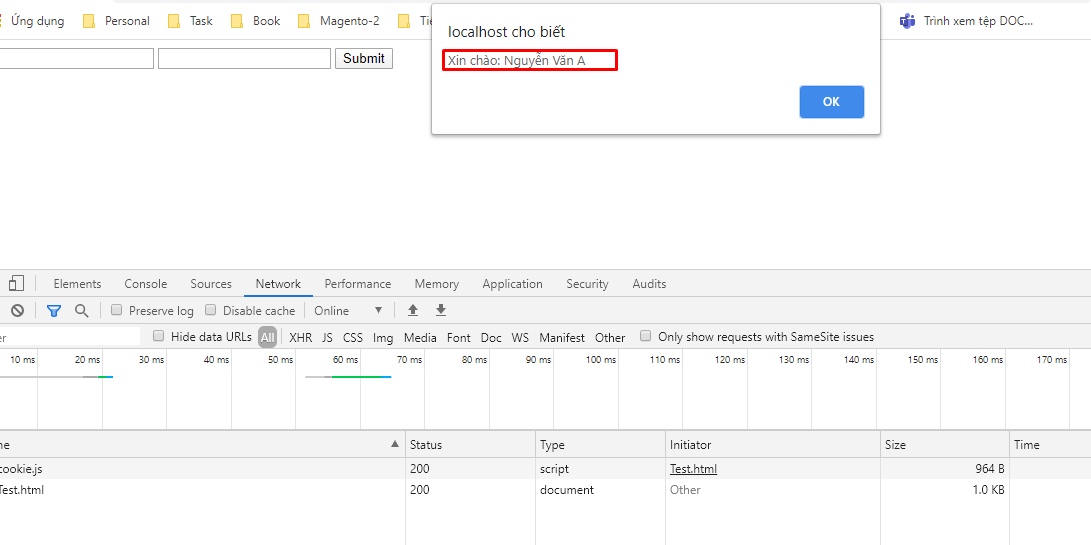




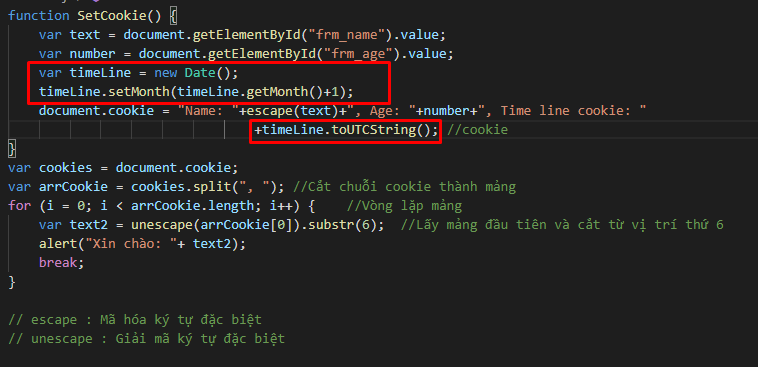
### Đọc cookie

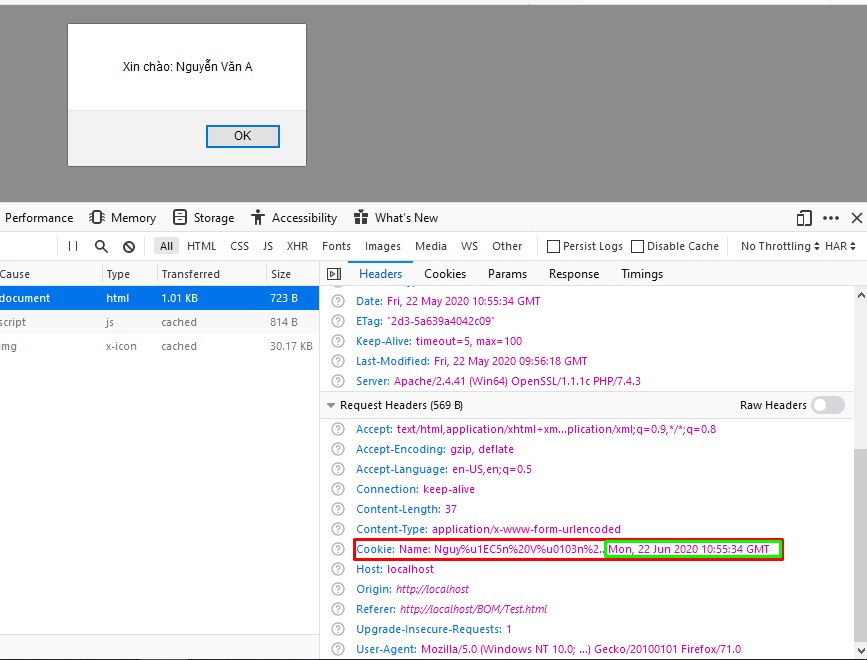


* Khi load lại trang web thì ta có:



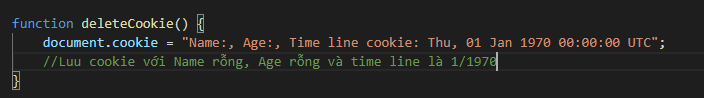
### Thiết lập thời gian sống



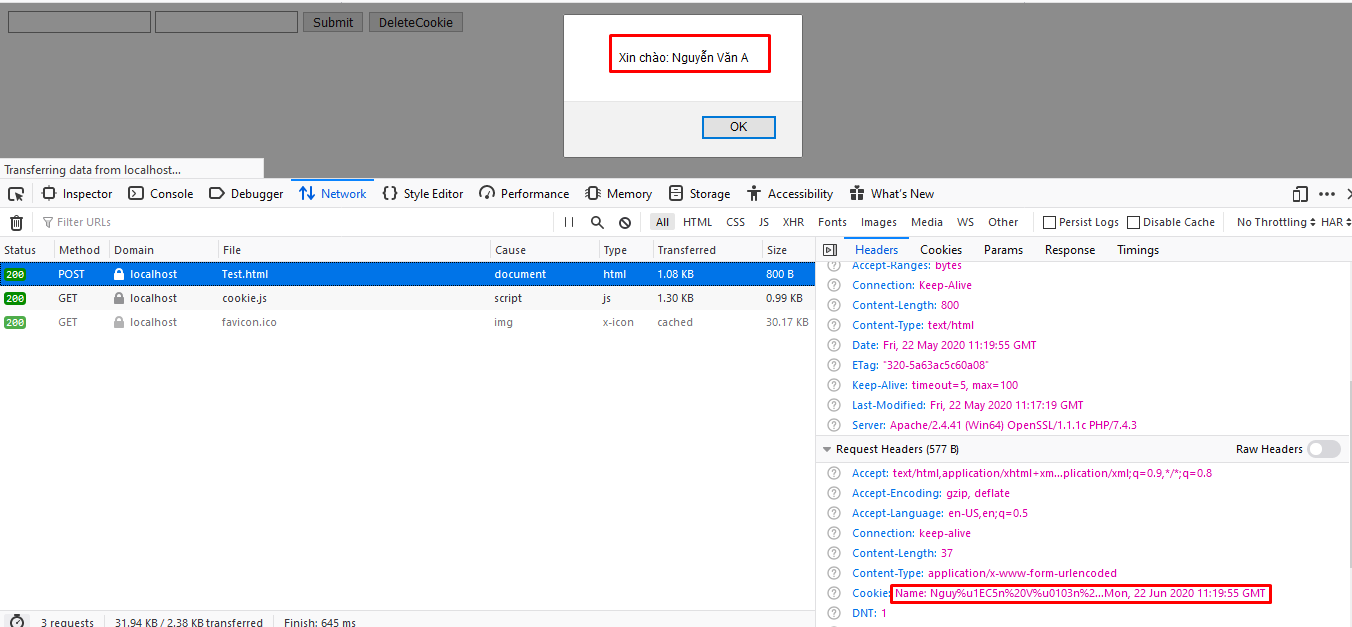


* Khi ta thiết lập lại ngày cho máy tính thì cookie sẽ ko còn tồn tại.

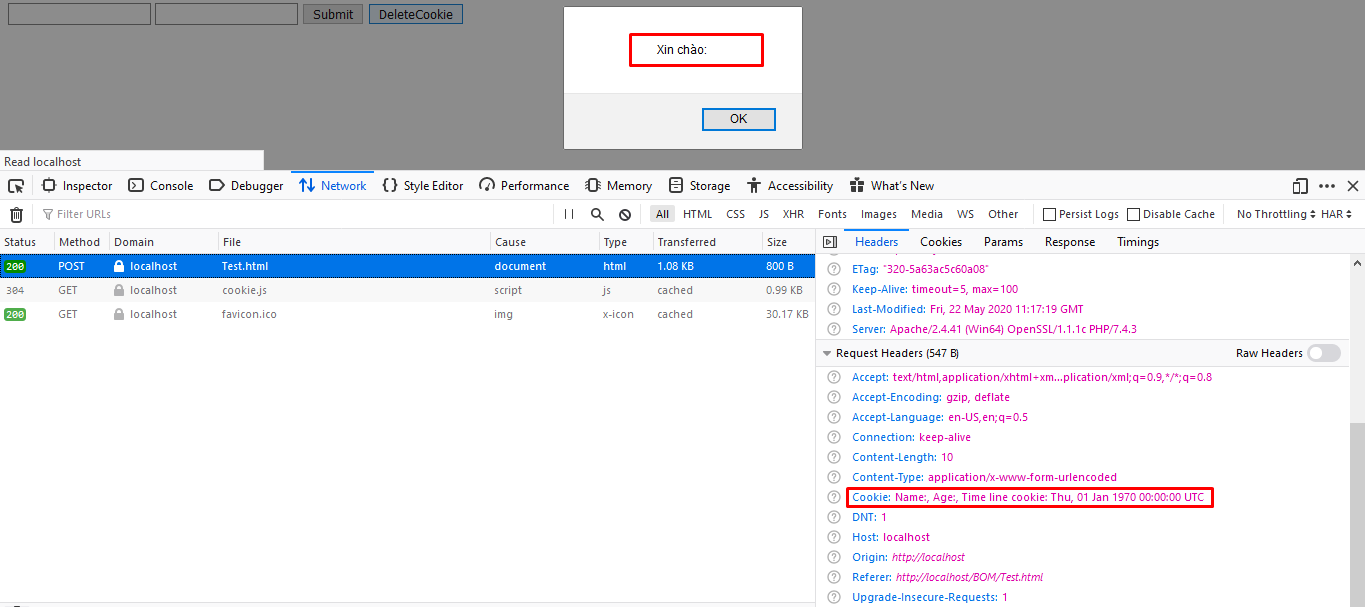
### Xóa cookie



* Khi chưa xóa:



* Khi đã xóa



# Các khái niệm khác

## Strict mode

Ở chế độ này thì các đoạn code của chúng ta phải được viết rất nghiêm ngặt và cẩn thận.

- Để sử dụng chế độ này ta sử dụng từ khóa: "use strict"

- "use strict" toàn cục là được đặt ở đầu file

- "use strict" cục bộ là được đặt ở đầu mỗi function

### Các trường hợp đặt “use strict”

1. Sử dụng biến chưa được định nghĩa: Bạn không thể sử dụng một biến mà chưa được định nghĩa trước đó.
2. Không chấp nhận delete biến: Bạn không thể delete các hàm, biến nếu chạy ở chế độ strict mode
3. Định thuộc tính nghĩa hai lần: Nếu trong một Object bạn định nghĩa tên key bị trùng thì sẽ bị lỗi.
4. Khai báo tham số bị trùng: Nếu bạn khai báo các tham số bị trùng tên thì sẽ bị lỗi
5. Lỗi literals và escape với number: Bạn không được sử dụng literals và escape với kiểu number.
6. Khai báo tên biến trùng với key: Bạn không thể khai báo tên của biến trùng với key của Javascript.
7. Từ khóa with bị loại bỏ: Bạn không thể sử dụng từ khóa with trong chế độ strict mode.
8. Không được khai báo biến trong lệnh eval: Eval là một hàm dùng để biên dịch một chỗi chứa mã javascript. Nếu trong chuỗi đó có chứa đoạn code khai báo biến thì sẽ bị lỗi.
9. Từ khóa this không còn là đối tượng windows nữa: Trước đây, khi bạn sử dụng this trong một hàm không phải là constructor, hoặc sử dụng this bên ngoài chương trình chính, thì con trỏ this lúc này chính là đối tượng window. Tuy nhiên, ở chế độ strict mode thì this sẽ mang giá trị undefined.

Chế độ strict mode luôn được bật trong module

## Biểu thức RegEx

* RegEx (regular expression) là tập hợp những quy tắc, dựa vào những quy tắc này ta sẽ viết ra những biểu thức so khớp giữa các chuỗi với nhau.
* Cú pháp: **/pattern/modifiers**
  + Trong đó:

pattern là chuỗi Regular Expression

modifiers là thông số cấu hình cho chuỗi pattern, và nó có các giá trị sau:

i : so khớp không quan tâm đến chữ hoa chữ thường

g : so khớp toàn bộ chuỗi cần tìm

m : so khớp luôn cả các dữ liệu xuống dòng (multiline)

### Hàm test()

* Để kiểm thử một biểu thức regular expression thì chúng ta sẽ sử dụng hàm test

Ex: 

* Kiểm tra xem trong chuỗi ‘Toi la Quang ND’ có từ Quang không
* Chuỗi Regular Expression luôn có cú pháp sau: **/^pattern$/** với ^ là bắt đầu chuỗi, $ là kết thúc chuỗi.

Ex: 

* Trong ví dụ này thì chuỗi phải bắt đầu bằng từ Quang thì biến mới có giá trị là true

### Bảng các biểu thức RegEx trong js căn bản

| **Biểu thức** | **Ý nghĩa** |
| --- | --- |
| [abc] | Tìm các chữ cái a,b hoặc c |
| [^abc] | Tìm các ký tự không phải a, b và c |
| [0-9] hoăc [a-z] | Tìm các ký tự từ 0-9 hoặc từ a-z |
| [^0-9] | Tìm các ký tự không phải chữ số từ 0-9 |
| (x|y) | ìm ký tự x hoặc y |
| \d | Tìm các chữ số |
| \s | Tìm khoảng trắng |
| n+ | Tìm 1 hoặc nhiều chữ n liên tiếp nhau |
| n\* | Tìm 0 hoặc nhiều chữ n liên tiếp nhau |
| n? | Tìm 0 hoặc 1 chữ n |

Ex: 

* Trong ví dụ này ta cần tìm trong chuỗi có ít nhất một chữ o

### Bảng quy tắc các giá trị đơn

| **Expression** | **Description** |
| --- | --- |
| . | Tìm ký tự bất kì |
| \w | Tìm ký tự chữ cái |
| \W | Tìm các ký tự không phải là chữ cái |
| \d | Tìm ký tự là chữ số |
| \D | Tìm ký tự không phải là chữ số |
| \s | Tìm ký tự là khoảng trắng |
| \S | Tìm ký tự không phải khoảng trắng |
| \b | Tìm so khớp bắt đầu hoặc kết thúc chuỗi |
| \B | Tìm so khớp không phải bắt đầu hoặc kết thúc chuỗi |
| \0 | Tìm ký tự NULL |
| \n | Tìm ký tự xuống hàng |
| \t | Tìm ký tự tab |

### Bảng quy tắc số lượng các ký tự

| **Expression** | **Description** |
| --- | --- |
| + | Kiểm tra ký tự xuất hiện một hoặc nhiều lần |
| \* | Kiểm tra ký tự xuất hiện không hoặc nhiều lần |
| ? | Kiểm tra ký tự xuất hiện không hoặc một lần |
| {X} | Kiểm tra ký tự xuất hiện đúng X lần |
| {X,Y} | Kiểm tra ký tự xuất hiện tối thiểu X lần và tối đa Y lần |
| {X,} | Kiểm tra ký tự xuất hiện ít nhất X lần |
| ^ | Kiểm tra ký tự bắt đầu chuỗi |
| $ | Kiểm tra ký tự kết thúc chuỗi |

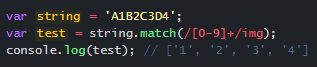
Ex: Kiểm tra chuỗi là các chữ số và dài 8 ký tự



### Phương thức match()

* match() sẽ trả về kết quả dựa vào cấu trúc của chuỗi pattern.

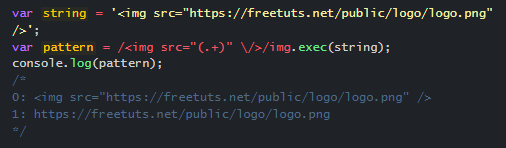
Ex: Lấy tất cả các số trong chuỗi



### Phương thức exec()

* lấy kết quả từng group chuỗi RegEx và hàm này được áp dụng nhiều trong thực tế

Ex: Viết biểu thức lấy đường dẫn của thẻ img

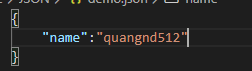


# JSON

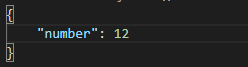
* Javascript Object Notation là một cú pháp để lưu trữ và trao đổi dữ liệu. JSON là văn bản được viết bằng ký hiệu đối tượng JavaScript.

## Cú pháp

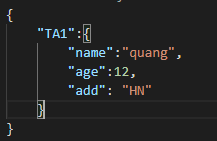
* Chuỗi: Phải được đặt trong dấu ngoặc kép



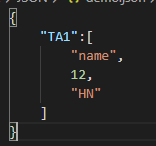
* Số: Không cần phải có dấu ngoặc



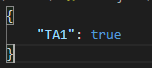
* Một đối tượng:



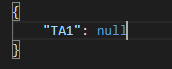
* Một mảng



* Boolean



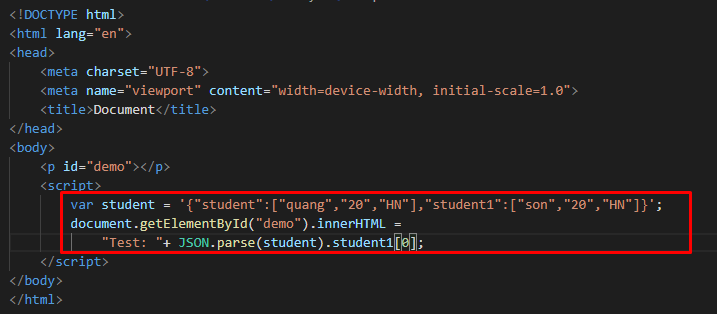
* Null



## Các hàm xử lý JSON

### JSON.parse()

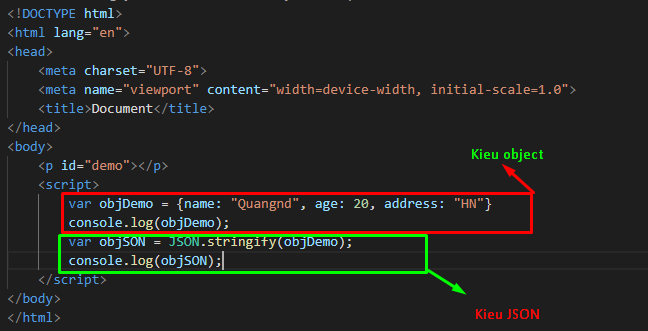
* Dùng để chuyển đổi văn bản thành đối tượng trong Javascript

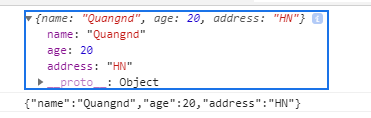




### JSON.Stringify()

* Hàm chuyển đổi object javascript thành kiểu JSON



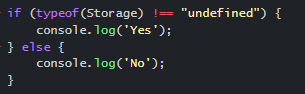


# localStorage và sessionStorage

* Local Storage là một module của javascript, hoạt động trên các trình duyệt web như Chrome / Firefox. Nó dùng để lưu trữ dữ liệu tạm thời và riêng biệt cho từng domain, nên đây được xem là giải pháp để thay thế cho cookie.
* Điểm khác biệt lớn nhất giữa cookie và local storage là local storage cho phép bạn lưu trữ thông tin tương đối lớn lên đến 5MB, ngoài ra local storage không gửi thông tin lên server như cookie. Cả local storage và cookie đều không làm ảnh hưởng đến hiệu xuất của trang web.
* Chúng ta có 2 loại local storage trong javascript, đó là:
  + **localStorage**: Lưu trữ dữ liệu vô thời hạn, dữ liệu sẽ được lưu trữ cho tới khi người dùng clear history.
  + **sessionStorage**: Lưu trữ dữ liệu cho mội phiên làm việc, có nghĩa dữ liệu sẽ bị mất khi bạn tắt browser.

## LocalStorage

* localStorage lưu trữ dữ liệu vô thời hạn, có nghĩa dữ liệu sẽ không bị mất cho tới khi bạn sử dụng tính năng clear history của trình duyệt, hoặc bạn sử dụng chính localStorage API để xóa.
* Đoạn code dưới đây sẽ giúp ta kiểm tra trình duyệt có hỗ trợ localStorage hay là không.



* Có 5 phương thức của localStorage sau:
  + **setItem()**: Thêm dữ liệu vào localStorage
  + **getItem()**: Lấy dũ liệu từ localStorage
  + **removeItem()**: Xóa dữ liệu ra khỏi localStorage
  + **clear()**: Xóa toàn bộ dữ liệu ra khỏi localStorage
  + **key()**: Lấy tên key của dữ liệu đang lưu trữ trong localStorage

### setItem()

* Dùng để thêm dữ liệu vào localStore, dữ liệu sẽ được lưu trữ ở dạng **key => value**
* Cú pháp: localStorage.setItem(key, value);

EX: 

### getItem()

* Dùng để lấy dữ liệu, tham số truyền vào chính là tên key muốn lấy. Nếu không tồn tại thì trả về undefined.

EX: 

### removeItem()

* Dùng để xóa dữ liệu, tham số truyền vào là key muốn xóa.

EX: 

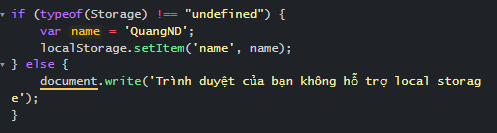
### clear()

* Xóa toàn bộ dữ liệu của localStorage.

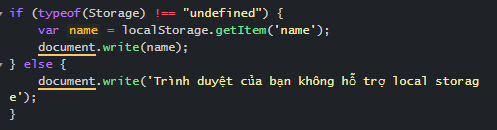
Ex: 

* VD: Lưu trữ tên QuangND và lấy tên đó ra

a.html dùng để lưu dữ liệu



b.html dùng để lấy ra dữ liệu vừa lưu



## SessionStorage

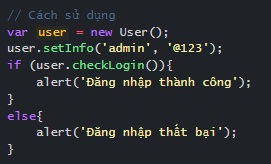
* Công dụng của sessionStorage cũng tương tự như localStorage, chỉ có một điểm khác đó là dữ liệu của sessionStorage sẽ mất khi bạn đóng trình duyệt, còn localStorage thì không.
* Ngoài ra, sessionStorage sẽ lưu trữ khác nhau cho mỗi tab. Nghĩa là khi bạn lưu dữ liệu ở tab này, xong bạn mở tab khác lên thì sẽ không thể sử dụng được dữ liệu đó.

# Javascript nâng cao

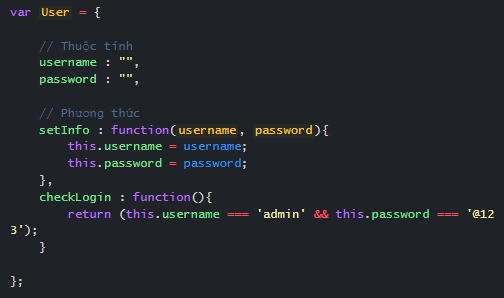
## Tạo class trong javascript

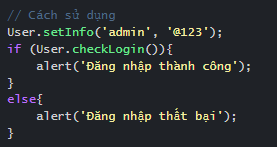
### Tạo bằng Function



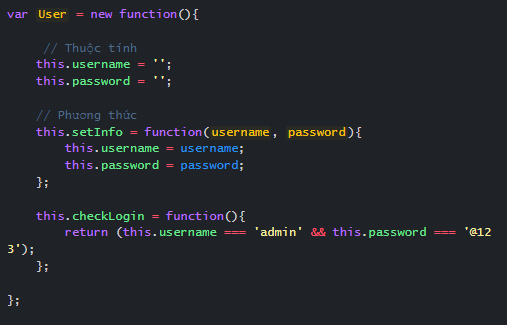


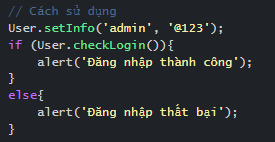
### Tạo bằng object





### Tạo bằng Singleton Object

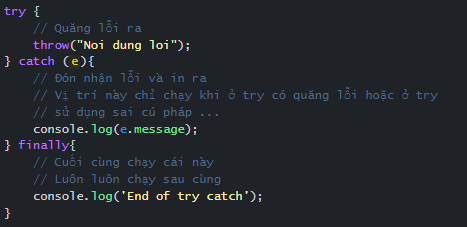




* Ta nên tạo class bằng cách sử dụng function vì ta có thể khai báo một lần và tạo nhiều đối tượng cho nó bằng từ khóa new

## Xử lý try catch

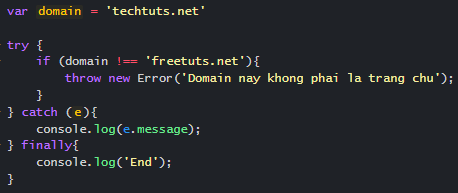
* Cú pháp:



### Throw new Error()

* Tham số e trong catch chính là một error object mặc định của javascript. Nhưng thực tế thì có một số trường hợp không phải là một lỗi, mà nó là một tính năng.

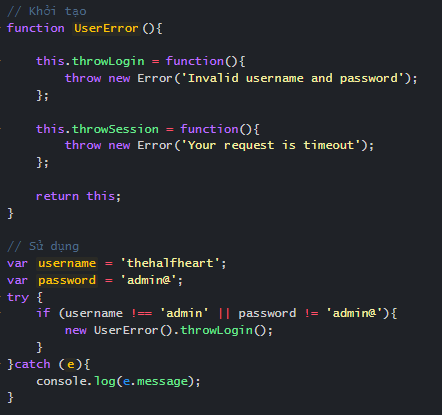
**VD:** Sử dụng throw new Error('message') để xuất thông báo lỗi.



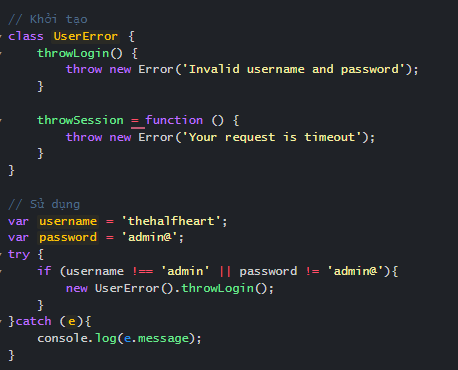
* e.message = 'Domain nay khong phai la trang chu'

### Tạo một object error sử dụng trong try catch

1. Sử dụng function



1. Sử dụng class



## Từ khóa this trong JS

* <https://freetuts.net/hieu-ro-hon-ve-tu-khoa-this-trong-javascript-757.html>

## Closure javascript

* <https://freetuts.net/closure-trong-javascript-758.html>

## Callback