# Ký hiệu cơ bản:

<\*>: chưa tìm hiểu

<\*!>: chưa tìm hiểu + quan trọng;

<…>: đang viết giở

# Client

## Vuejs + javascrip

### axios …

Lấy dữ liệu từ Api

## Angular + typescript

### httpClient …

### TypeScript

#### For

For..in: lấy chỉ số của từng phần tử

var someArray = [9, 2, 5];

for (var item in someArray) {

console.log(item); // 0,1,2

}

For..of: lấy giá trị của từng phần tử

var someArray = [9, 2, 5];

for (var item of someArray) {

console.log(item); // 9,2,5

}

#### Class

class Point {

x: number; // default public

y: number;

constructor(x: number, y: number) {

this.x = x;

this.y = y;

}

add(point: Point) {

return new Point(this.x + point.x, this.y + point.y);

}

}

var p1 = new Point(0, 10);

var p2 = new Point(10, 20);

var p3 = p1.add(p2); // {x:10,y:30}

#### abstract class

abstract class FooCommand {

abstract execute(): string;

}

class BarErrorCommand extends FooCommand {} //lỗi vl

class BarCommand extends FooCommand {

execute() {

return `Command Bar executed`;

}

}

const barCommand = new BarCommand();

barCommand.execute(); // Command Bar executed

#### contructor

class Foo {

constructor(public x:number) {

}

}

Hàm tạo và dùng luôn

function Person(age) {

this.age = age;

this.growOld = () => {

this.age++;

}

}

#### So sánh class và interface

* Khác nhau khi compile ra javascript:

Interface khi compile sẽ bị loại bỏ. nó chỉ có mục đích nhắc hint, báo lỗi khi viết code bằng editor, sau đó nó sẽ bị loại bỏ khỏi mã đích.

Class thì không bị loại bỏ khi compile

Chỉ sử dụng Class khi ta có logic nghiệp vụ thực sự cần được implement để thực thi. Ngược lại, nếu chỉ dùng nó để tạo 1 ràng buộc kiểu cho params hay variable, ta nên dùng Interface để giảm kích thước chương trình sau khi biên dịch

#### Tham số còn lại

Các tham số còn lại (được biểu thị bằng đối số cuối cùng) cho phép bạn nhanh chóng chấp nhận nhiều đối số trong hàm của mình và nhận chúng dưới dạng một mảng. Điều này được chứng minh trong ví dụ dưới đây....argumentName

* Đơn gianrL truyền thừa thì nó sẽ nhận hét vào param cuối và nó là 1 mảng( nếu không khai báo cụ thể kiểu dữ liệu)

function iTakeItAll ( first , second , ... allOthers ) {

bàn điều khiển . nhật ký ( allOthers ) ;

}

iTakeItAll ( 'foo' , 'bar' ) ; // []

iTakeItAll ( 'foo' , 'bar' , 'bas' , 'qux' ) ; // ['bas', 'qux']

#### Cấu trúc mảng

Một câu hỏi lập trình phổ biến: "Làm thế nào để hoán đổi hai biến mà không sử dụng biến thứ ba?". Giải pháp TypeScript:

var x = 1 , y = 2 ;

[ x , y ] = [ y , x ] ;

bàn điều khiển . log ( x , y ) ; // 2,1

#### Gán mảng

Toán tử spread cho phép bạn dễ dàng đặt *phiên bản mở rộng* của một mảng vào một mảng khác. Điều này được chứng minh trong ví dụ dưới đây:

var list = [ 1 , 2 ] ;

list = [ ... list , 3 , 4 ] ;

bàn điều khiển . log ( danh sách ) ; // [1,2,3,4]

Bạn có thể đặt mảng đã mở rộng ở bất kỳ vị trí nào và nhận được hiệu quả như mong đợi:

var list = [ 1 , 2 ] ;

list = [ 0 , ... list , 4 ] ;

bàn điều khiển . log ( danh sách ) ; // [0,1,2,4]

#### Gán object

Vd1:

const point2D = {x: 1, y: 2};

/\*\* Create a new object by using all the point2D props along with z \*/

const point3D = {...point2D, z: 3};

Vd2:

const point2D = {x: 1, y: 2};

const anotherPoint3D = {x: 5, z: 4, ...point2D};

console.log(anotherPoint3D); // {x: 1, y: 2, z: 4}

const yetAnotherPoint3D = {...point2D, x: 5, z: 4}

console.log(yetAnotherPoint3D); // {x: 5, y: 2, z: 4}

Vd3:

const foo = {a: 1, b: 2, c: 0};

const bar = {c: 1, d: 2};

/\*\* Merge foo and bar \*/

const fooBar = {...foo, ...bar};

// fooBar is now {a: 1, b: 2, c: 1, d: 2}

Thư viện có sẵn chủa angular

### Angular

#### Class vs style

(V-bind:””)

Set điều kiện = có ngặc{}

Không điều kiện = [] hoặc khôn cần dấu

<p [className]="'red'">students works is real!</p>

<p [ngClass]="{'red': true}">{{name}}</p>

-với nhiều class

< div [ngClass] = " ['example-class', 'other-class'] " > </ div >

<p [ngClass]="{'red': true, 'bg-cl': false}">{{name}}</p>

-với biến điều kiện

<p [ngClass]="{'red': true, 'bg-cl': isBackground}">{{name}}</p>

Or

<p [ngClass]="{'red': true}" [class.bg-cl] ="isBackground">{{name}}</p>

-style

<p [style.color]="'#cccccc'">attribute</p>

#### Attribute

<img src='{{avatar}}' alt="">

Or

<p [style.color]="'#cccccc'">attribute</p>

<img

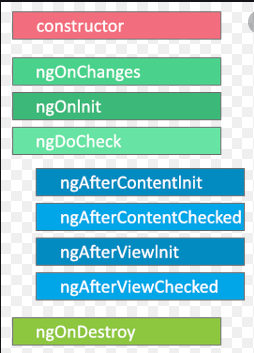
[src]='avatar' alt="">

* avatar là 1 biến

##### hidden

<input [hidden]="isBackground"/>

#### LifeCycle



Vd về vòng đơig angular

Angular is running in development mode. Call enableProdMode() to enable production mode.

Method A ngOnInit. value:

Method A ngAfterContentInit. value:

Method A ngAfterContentChecked. value:

Method Child ngOnChanges. value:

Method Child ngOnInit.

Method Child ngAfterContentInit. value:

Method Child ngAfterContentChecked. value:

Method Child ngAfterViewInit. value:

Method Child ngAfterViewChecked. value:

Method A ngAfterViewInit. value:

Method A ngAfterViewChecked. value:

Method A ngAfterContentChecked. value:

Method Child ngAfterContentChecked. value:

Method Child ngAfterViewChecked. value:

Method A ngAfterViewChecked. value:

#### Event

(v-on: or @)

<button (click)="isBackground = !isBackground">ẩn/tắt</button>

#### Directive

Attribute directives

Tạo mới 1 directive

ng generate directive highlight

Ta có cấu trúc bên trong:

content\_copyimport { [Directive](https://angular.io/api/core/Directive), [ElementRef](https://angular.io/api/core/ElementRef) } from '@angular/core';

@[Directive](https://angular.io/api/core/Directive)({

selector: '[appHighlight]' // tên chỉ thị để add vào html tag mong muốn

})

export class HighlightDirective {

constructor(el: [ElementRef](https://angular.io/api/core/ElementRef)) { el.nativeElement.style.backgroundColor = 'yellow'; }

}

Và đây là khi sử dụng:

<p appHighlight>Highlight me!</p>

#### Pipe

Giống filter

Nó cung cấp các hàm như sau

##### Chung

* sử dụng nhiều pipe 1 lúc

<p>Today is {{today | date | lowercase}}</p>

##### Tạo pipe riêng

Tạo 1 file ts riêng nhé

Temp-converter.pipe.ts

import { Pipe } from '@angular/core';

@Pipe({

name: 'tempConverter'

})

export class ChangeStrPipe implements PipeTransform {

transform(value: string): string {

let newStr: string = 'htm' + value;

return newStr;

}

}

Thêm vào trong ngModul

**import** { TempConverterPipe } **from** './pipes/temp-converter.pipe';

@NgModule({

declarations: [

*// các Component, Directive, Pipe khác*

TempConverterPipe

],

imports: [...],

providers: [...],

*//...*

})

**export** **class** AppModule { }

##### [AsyncPipe](https://angular.io/docs/ts/latest/api/common/index/AsyncPipe-pipe.html)

<span>Wait for it... {{ greeting | [async](https://angular.io/api/common/AsyncPipe) }}</span>

Đợi khi nào có greeting thì mới hiển thị với greeting là 1 lời hứa

this.greeting = new Promise<string>((resolve, reject) => { this.resolve = resolve; });

! Dùng tương tự với biến kiểu observable hoặc subject

##### [CurrencyPipe](https://angular.io/docs/ts/latest/api/common/index/CurrencyPipe-pipe.html)

##### [DatePipe](https://angular.io/docs/ts/latest/api/common/index/DatePipe-pipe.html)

Pre-defined format options

| **Option** | **Equivalent to** | **Examples (given in en-US locale)** |
| --- | --- | --- |
| 'short' | 'M/d/yy, h:mm a' | 6/15/15, 9:03 AM |
| 'medium' | 'MMM d, y, h:mm:ss a' | Jun 15, 2015, 9:03:01 AM |
| 'long' | 'MMMM d, y, h:mm:ss a z' | June 15, 2015 at 9:03:01 AM GMT+1 |
| 'full' | 'EEEE, MMMM d, y, h:mm:ss a zzzz' | Monday, June 15, 2015 at 9:03:01 AM GMT+01:00 |
| 'shortDate' | 'M/d/yy' | 6/15/15 |
| 'mediumDate' | 'MMM d, y' | Jun 15, 2015 |
| 'longDate' | 'MMMM d, y' | June 15, 2015 |
| 'fullDate' | 'EEEE, MMMM d, y' | Monday, June 15, 2015 |
| 'shortTime' | 'h:mm a' | 9:03 AM |
| 'mediumTime' | 'h:mm:ss a' | 9:03:01 AM |
| 'longTime' | 'h:mm:ss a z' | 9:03:01 AM GMT+1 |
| 'fullTime' | 'h:mm:ss a zzzz' | 9:03:01 AM GMT+01:00 |

##### [DecimalPipe](https://angular.io/docs/ts/latest/api/common/index/DecimalPipe-pipe.html)

<p>B: {{b | [currency](https://angular.io/api/common/CurrencyPipe):'CAD':'symbol':'4.2-2'}}</p>

4: trước dấy ,

2-2: sau dấu , từ 2 đến 2 ký tựu

<p>B: {{b | [currency](https://angular.io/api/common/CurrencyPipe):'CAD':'symbol':'4.1-5'}}</p>

4: trước dấy ,

1-5: sau dấu , từ 1 đến 5 ký tựu

##### [JsonPipe](https://angular.io/docs/ts/latest/api/common/index/JsonPipe-pipe.html)

##### [LowerCasePipe](https://angular.io/docs/ts/latest/api/common/index/LowerCasePipe-pipe.html)

##### [PercentPipe](https://angular.io/docs/ts/latest/api/common/index/PercentPipe-pipe.html)

##### [SlicePipe](https://angular.io/docs/ts/latest/api/common/index/SlicePipe-pipe.html)

##### [TitleCasePipe](https://angular.io/docs/ts/latest/api/common/index/TitleCasePipe-pipe.html)

##### [UpperCasePipe](https://angular.io/docs/ts/latest/api/common/index/UpperCasePipe-pipe.html)

#### Selectors

##### \*ng-if

###### Kiểu 1:

<span \*ngIf="massageChild == undefined">{{massageChild | uppercase}}</span>

Thỏa mãn thì show

###### Kiểu 2:

<p>skill: <span \*ngIf="messageSkill == '';else showMessageSkill">không có dữ liệu</span></p>

<ng-template #showMessageSkill>messageSkill</ng-template>

Else thì show #showMessageSkill đúng thì hiện không có dữ liệu

* # #nullMessageSkill Đặt nó ở đâu thì đặt

###### Kiểu 3:

là kiểu 2 dạng viết tắt

<p>skill: <span \*ngIf="messageSkill == '';then nonMessageSkill else showMessageSkill"></span></p>

<ng-template #nonMessageSkill>không có dữ liệu</ng-template>

<ng-template #showMessageSkill>messageSkill</ng-template>

<ng-template> Được truyền vào bên trong <span>

Lúc này không thể thêm bất cứ thứ gì khác vào bên trong <span> được

#### Lắng nghe sự kiện resize

#### Sử dụng @HostListener

@HostListener("window:resize", [])

  private onResize() {

    setTimeout(function(){

      $(".size-screen").css("background-color","red");

    }, 300);

    $(".size-screen").css("background-color","green");

    this.screen.sizeX = $(window).width();

  }

#### ResizeObserver

Resize đối với các loại tag

Install cái này đã: npm i @juggle/resize-observer

Vd:

  ngAfterViewInit() {

    const me= this;

    const resizeObserver = new ResizeObserver((entries) =>{

      for (let entry of entries) {

        me.screen.sizeX = entry.contentRect.width;

        me.screen.sizeY = entry.contentRect.height;

      }

    });

    const els = document.querySelectorAll("body");

    resizeObserver.observe(els[0]);

  }

Nhớ phải dùng querySelectorAll để truyền els[0]

#### @HostListenter

Mẫu:

@[HostListener](https://angular.io/api/core/HostListener)(DomEvent, ['$event.target']) functionName(btn) { …}

Vd:  
@[HostListener](https://angular.io/api/core/HostListener)('window:keydown', ['$event'])

handleKeyDown(event: KeyboardEvent) { …}

- DomEvent: thẻ dom:sự kiện sử lý

- functionName: hàm thực thi function . (thường là hàm rỗng và được định nghĩa lại trong ngOnInit())

This.handleKeyDown = function(event: KeyboardEvent){…};

\*Có thể chỉ định lắng nghe phím từ bàn phím

vd: chỉ lắng nghe sự kiện Ctrl + S

@[HostListener](https://angular.io/api/core/HostListener)('window:keydown.control.s', ['$event'])

handleKeyDown(event: KeyboardEvent) { …}

or: @HostListener('keydown.shift.tab', ['$event'])

#### View Child

Vd 1: dùng như 1 componet để gọi hàm hoặc lấy dữ liệu

@ViewChild('primaryColorSample')

sample: ColorSampleComponent;

vd2:

#### ngZone

tổng quan: đối với vue hoặc react khi thay đổi bất cứ thứ gì dữ liệu sẽ được render vào dom ảo trước khi render lên dom thật. Sẽ có 1 cơ chế so sánh giữa dom ảo và dom thật để render, khi này sự ưu việt được thể hiện so với angular vì bản chất angular lắng nghe và render liên tục dù là 1 thay đổi nhỏ.

Lợi dụng điều này ta có thể chạy thay đổi 1 số dữ liệu là không

Khi Sử dụng **jquery** với thường là không gây ảnh hưởng đến dữ liệu bên trong các component. Lợi dụng điều này ta có thể đặt các câu lệnh **jquery** bên trong runOutsideAngular() .Khi đó cho dù có thay đổi dữ liệu angular cũng sẽ không tự động cập nhật lên html cho đến khi kết thúc khối lệnh trong runOutsideAngular

* Khi ta dùng j**query** hoặc **setTimeout** ,**setInterval CHU**

zone.run

example:

import { Component, HostListener, NgZone } from '@angular/core';

@Component({

  selector: 'my-app',

  template: `

  In Angular zone: {{timestampNgZone}}<br>

  Outside Angular zone: {{timestampOutsideNgZone}}<br><br>

  <br />

  Change detection only runs for the user click, but not for the async callback. So, the second binding is out-of-date because it has the last value when re-evaluated after the next user click.

  `

})

export class AppComponent  {

  timestampNgZone: string | Date =  'never';

  timestampOutsideNgZone: string | Date =  'never';

  constructor(private ngZone: NgZone) {

  }

  @HostListener('pointerdown')

  onPointerdown() {

    const timestamp = new Date();

    this.timestampNgZone = timestamp

    this.ngZone.runOutsideAngular(() => {

      setTimeout(() => this.timestampOutsideNgZone = timestamp);

    }

    )

  }

}

=>

In Angular zone: Tue May 11 2021 00:03:33 GMT+0700 (Indochina Time)  
Outside Angular zone: Tue May 11 2021 00:03:29 GMT+0700 (Indochina Time)

Khi click: cả 2 biến cùng được cập nhật. nhưng timestampOutsideNgZone được chạy trong runOutsideAngular nên không được cập nhật trên html. Mặc dù trong file .ts giá trị của nó tương đương timestampNgZone .Giá trị hiển thị runOutsideAngular chính là giá trị của lần click trước. Ngay khi click. zone đã so sánh value có sự khác nhau đối với runOutsideAngular Nó cập nhật lên html giá trị đang có. Nhưng giá trị mới của lần ấn này lại bị sẽ được hiển thị ở lần ấn kế tiếp

Lưu ý: khi thực hiện click là đã thông báo có sự thay đổi. vì thời gian thông báo tương đương với thời gian thay đổi trong runOutsideAngular Nên phải dùng setTimeOut( hoặc setInterval) đẻ **zone** không nhận ra sự thay đổi này

1 example khác: https://stackblitz.com/edit/ngzone-demo?file=src%2Fapp%2Fng-zone.component.ts

// Loop outside of the Angular zone

  // so the UI DOES NOT refresh after each setTimeout cycle

  processOutsideOfAngularZone() {

    this.label = 'outside';

    this.progress = 0;

    this.\_ngZone.runOutsideAngular(() => {

      this.\_increaseProgress(() => {

        // reenter the Angular zone and display done

        this.\_ngZone.run(() => { console.log('Outside Done!'); });

      });

    });

  }

  \_increaseProgress(doneCallback: () => void) {

    this.progress += 1;

    console.log(`Current progress: ${this.progress}%`);

    if (this.progress < 100) {

      window.setTimeout(() => this.\_increaseProgress(doneCallback), 10);

    } else {

      doneCallback();

    }

  }

increaseProgress được gọi lần đầu tiên tại thời điển tương đương 1 sự kiện gọi đến processOutsideOfAngularZone Nên htmlsẽ hiện 1%. Từ đệ quy hàm đó trở đi trong setTimeOut thì html sẽ không hiện process nữa và html sẽ hiện 100% sau khi chạy ra khỏi runOutsideAngular

##### RUN()

Bên thứ 3 đang cập nhật dữ liệu của bạn. có thể đặt nó trong funtction để phát hiện thay đổi

myFunction(){

this.zone.run(this.someFunctionThatIsRunByAThirdPartyCode);

}

#### dedetechange

cos theer doc nhung cai rat hay : <https://www.facebook.com/groups/AngularVietnam/permalink/958719831293535/>

Có thể áp đụng dtf. Detectchanges thay cho

 this.\_ngZone.run(() => {});

để cập nhật value cho khung nhìn sau khi chay xong [runOutsideAngular](#_ngZone)

##### ChangeDetectionStrategy.OnPush

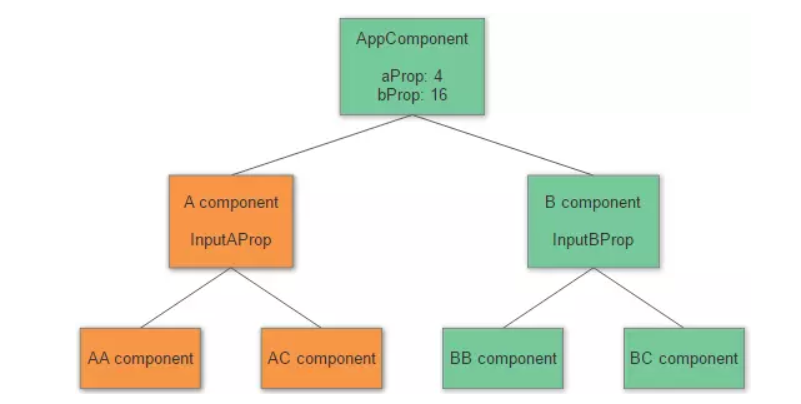
Nếu giá trị input không bị thay đổi (áp dụng tính chất mutable)

Thì component chính nó và tất cả con , cháu của nó sẽ không được render lại dom mặc dù giá trị có thay đổi.

Lưu ý với dữ liệu được input dạng mutable khi bị thay đổi sẽ không được cập nhật lại trên dom ở component bị set OnPush vì nó đang so sánh mutable như là con trỏ nên không nhận ra sự thay đổi sâu

Lưu ý 2: Nếu các thuộc tính trong component được set value từ subcirbe từ 1 subject or observable (không phải được set value từ sử lý sự kiện (click,..)) thì nó sẽ không được render lại trên dom

##### Detach



* Nếu A component set detach thì toàn bộ nó và thành phần con của nó sẽ không được cập nhật lên dom.

Còn cha nó và những nhánh khác vẫn hiển thị bình thường nếu có value dạng mutable hoặc có emit truyền lên

* Khi set detach, value được input vào những component con cháu sẽ không nhận được giá trị mới cho đế khi rettach hoặc detectChange (trừ mutable)

##### Reattach

Hủy trạng thái detach

##### detectChange

Cập nhật 1 lần giá trị hiện tại của component và các compont con của nó

##### markForCheck

Khi ở chế độ onpush chúng ta thay đổi trạng thái bên trong và nó không phải là một phần của xử lý sự kiện hoặc kích hoạt có thể quan sát được (như set value từ subcribe từ 1 observable or subject) nó sẽ không render lại dom . Vì vậy phải thêm .markForCheck vào thì mới render lại dom

* Có vd trong project angular tutorial (vẫn có thể dùng detectChange)

  ngOnInit(): void {

    this.data.subscribe(data => {this.oldParamSinge = data;}) // không render lại dom

    this.data.subscribe(data => {

      this.oldParamSinge = data;

      this.cdf.markForCheck(); // render lại được

    })

  }

##### So sánh detectChange and ngZone

ngZone phát hiện thay đổi trên toàn bộ ứng dụng.( using view bindings or @HostBinding() or  event listeners cũng sẽ render lạo toàn bộ ứng dụng

detectChange. Chạy trên các thành phần cụ thể (chạy cục bộ trong component) và theo thứ tự từ cha xuống con

#### Ng-

##### Ng-template

Được dùng với if, switch, for để làm các khung mẫu hiển thị

###### \*ngTemplateOutlet

Ứng dụng như ng-content nhưng linh loạt hơn. Có thể set 2 chỗ và lấy ra bất kì chỗ nào(có thể khác component

**Vd:1 đặt 1 slot như sau**

crm-grid.componnet.html – con

…

<ng-container \*ngTemplateOutlet=”temOut”></ ng-container>

…

<!— set thêm 1 chỗ sử dụng đến nó 🡪

<ng-container \*ngTemplateOutlet=”temOut”></ ng-container>

….

crm-grid.componnet..ts

…

@Input()

temOut : TemplateRef<any>;

…

Truyền đối tượng tự ngoài vào

crm-related-account.component.html - cha

…

<crm-grid #grid … [temOut] = “temOut” …>

…

<ng-template #temOut>…</ ng-template>

…

</ crm-grid>

Vd2: Đơn giản hơn trong phạm vi 1 componnet

<div>

<ng-container \*ngTemplateOutlet="companyLogoTemplate"></ng-container>

<h1>Company History</h1>

<div>{{companyHistory}}</div>

</div>

<form (ngSubmit)="onSubmit()">

<ng-container \*ngTemplateOutlet="companyLogoTemplate"></ng-container>

<h1>User info</h1>

<label>Name:</label><input type="text" [(ngModel)]="userName" />

<label>Account ID:</label><input type="text" [(ngModel)]="accountId" />

<button>Submit</button>

</form>

<div class="footer">

<ng-container \*ngTemplateOutlet="companyLogoTemplate"></ng-container>

</div>

10+

<ng-template #companyLogoTemplate>

<div class="companyLogo">

<img [src]="logoSourceUrl">

<label>The ACME company, {{employeeCount}} people working for you!</label>

</div>

</ng-template>

##### Ng-container

##### Ng-content

###### select

tương tự như slot trong vuejs

link tham khảo: <https://scotch.io/tutorials/angular-2-transclusion-using-ng-content>

Ta dùng ng-content để xác dịnh vị trí đặt slot:

Component-boder-layout.componnet.html

…

<ng-content select=”.slot-header”

…

Khi sử dụng component –form và thêm slot

Component-edit. componnet.html

…

<component-boder-layout >

<div class=” slot-header” >nội dung hiển thị</div>

</componnet.html>

…

#### Dynamic Component

##### entryComponents

#### observable

Chuyển đổi 1 stream thành 1 observable : of(1, 2, 3);

##### Pipe

##### startWith

vd:

//emit (1,2,3)

const source = of(1, 2, 3);

//start with 0

const example = source.pipe(startWith(0));

//output: 0,1,2,3

const subscribe = example.subscribe(val => console.log(val));

##### delay

delay(<milisecond>);

pipe(delay(1000));

##### mergeMap

* Thường kết hợp với forkJoin() – dùng để gộp nhiều observable
* Sử dụng khi muốn gọi 1 obervable sau khi có 1 kết quả của observable khác

Cách thường vẫn làm

Source.subscrise((res) => {

sourceSubData(res).subscrise( (resSub) => {

// todo

}

);

});

Sử dụng với mergeMap

Source.pipe(mergeMap( (res) =>

forkJoin(

of(res),

sourceSubData(res)

)

)).subscrise((resps) => {

Console.log(resps[0]);

Console.log(resps[1]);

});

#### Subject …

#### ViewChild

@ViewChid

Cú pháp:

@ViewChild(<tên đánh dấu>) <tên sử dụng> : ElementRef;

* Dùng với thẻ không phải là component
* Thường thao tác với nativeElement: <>.nativeElement;

@ViewChild(<tên đánh dấu>) <tên sử dụng> : <tên component>;

* Dùng khi muốn thao tác vơi component đó

Static trong viewChild

@ViewChild(<tên đánh dấu> {static: <true/false>, read: <tên component>}) <tên sử dụng> : <tên component>;

Static = true; cho phép load nội dung luôn trước khi ngOnInit được tạo – có thể sử dụng được ở ngOnInit

Static – fales: sử dụng được chỉ khi ở ngAfterViewInit, ngAfterContentInit

! Lưu ý

* Khôn1g nên đặt viewChild trong ngIf, ng-switch, ng-template, ng-container để tránh bị lỗi underfine

#### ViewChildren

@ ViewChildren

#### Unsubcrise

* Cho phép bạn hủy đăng ký tất cả các subcrise trong 1 component khi nó bị distroy

Luôn tạo ra 1 thuộc tính subject<any>

private \_unDestroy : Subject<any> = new Subject<any>();

Tiếp theo: hãy takeUnitl tất cả các subscript cho nó

 this.personService.data.pipe(takeUntil(this.\_unDestroy), startWith({name: '123123123',  age: 0 })).subscribe(data => {

        me.person.name = data.name;

        me.person.age = data.age;

    });

Cuối cùng: thêm 1 hàm để hủy thuộc tính đại diện này đi trong ngOnDestroy()

 baseFunction.unDestroy(this.\_unDestroy);

export class baseFunction {

    public static unDestroy(\_unsubscribe: Subject<any>){

        if(\_unsubscribe){

            \_unsubscribe.next();

            \_unsubscribe.complete();

            \_unsubscribe.unsubscribe();

        }

    }

}

* Cần học
* Eval() trong typescript
* elementRef
* BÓc tác cách property base control
* -Hostbinding for component (add class) get set conditon
* Mat-tab-label – tìm trong code cũ phân hệ mạng xã hội
* activateRoute: -https://stackblitz.com/edit/angular-resolver-simple-example?file=src%2Fapp%2Fapp-routing.module.ts
  + param ---
  + resolve: (từ khóa tìm kiếm: data.viewdata)
  + .data: thì gộp chung data với data of resolve
  + .queryParams.: lấy param từ this.route.navigate([“url”],{ queryParams : {<value>}})
* Dymamic componet : componentFactoryResolve
* Backend: allowAnonymous, các loại addsingleton, authentication JWT

## CSS, SCSS

### Selector

|  |  |  |
| --- | --- | --- |
| **Bộ chọn** | **Ví dụ** | **Mô tả** |
| [\*](https://hocwebchuan.com/reference/cssSection/se_all.php) | \* | Chọn tất cả các thành phần html. |
| [tag](https://hocwebchuan.com/reference/cssSection/se_tag.php) | p | Chọn tất cả các thành phần cùng loại. |
| [tag01 tag02](https://hocwebchuan.com/reference/cssSection/se_tag_tag.php) | div p | Chọn tất cả thành phần <tag02> bên trong thành phần <tag01>. |
| [tag01,tag02](https://hocwebchuan.com/reference/cssSection/se_tag_more.php) | div,p | Chọn tất cả thành phần <tag01> và <tag02>. |
| [tag01+tag02](https://hocwebchuan.com/reference/cssSection/se_tag_pluss.php) | div+p | Chọn tất cả thành phần <tag02> được đặt kế và sau thành phần <tag01>. |
| [tag01>tag02](https://hocwebchuan.com/reference/cssSection/se_tag_gt.php) | div>p | Chọn tất cả thành phần <tag02> có thành phần cha là <tag01>. |
| [.class](https://hocwebchuan.com/reference/cssSection/se_class.php) | .className | Chọn tất cả các thành phần có cùng tên class. |
| [#id](https://hocwebchuan.com/reference/cssSection/se_id.php) | #idName | Chọn tất cả các thành phần có cùng tên id. |
| [:active](https://hocwebchuan.com/reference/cssSection/se_active.php) | a:active | Chọn các liên kết được kích hoạt. |
| [:after](https://hocwebchuan.com/reference/cssSection/se_after.php) | p:after | Thêm nội dung ngay phía sau thành phần. |
| [:before](https://hocwebchuan.com/reference/cssSection/se_before.php) | p:before | Thêm nội dung ngay phía trước thành phần. |
| [:first-child](https://hocwebchuan.com/reference/cssSection/se_firstchild.php) | p:first-child | Chọn thành phần đầu tiên của thành phần cha chứa nó. |
| [:first-letter](https://hocwebchuan.com/reference/cssSection/se_firstletter.php) | p:first-letter | Chọn ký tự đầu tiên của một thành phần. |
| [:first-line](https://hocwebchuan.com/reference/cssSection/se_firstline.php) | p:first-line | Chọn dòng đầu tiên của một thành phần. |
| [:focus](https://hocwebchuan.com/reference/cssSection/se_focus.php) | input:focus | Thành phần sẽ focus khi được chọn. |
| [:hover](https://hocwebchuan.com/reference/cssSection/se_hover.php) | a:hover | Chọn các liên kết được hover (di chuyển chuột lên thành phần). |
| [:lang(mã ngôn ngữ)](https://hocwebchuan.com/reference/cssSection/se_lang.php) | p:lang(vi) | Chọn thành phần với thuộc tính ngôn ngữ có giá trị bắt đầu với "mã ngôn ngữ". |
| [:link](https://hocwebchuan.com/reference/cssSection/se_link.php) | a:link | Định dạng cho tất cả liên kết khi chưa được click. |
| [:visited](https://hocwebchuan.com/reference/cssSection/se_visited.php) | a:visited | Định dạng cho các thành phần đã được click. |
| [[Thuộc tính]](https://hocwebchuan.com/reference/cssSection/se_attr.php) | [target] | Chọn tất cả các thành phần có cùng thuộc tính (attribute). |
| [[Thuộc tính|=ngôn ngữ]](https://hocwebchuan.com/reference/cssSection/se_attr_value_lang.php) | [lang|=vi] | Được sử dụng để chọn tất cả các thành phần với thuộc tính có giá trị = "ngôn ngữ". |
| [[Thuộc tính~=giá trị]](https://hocwebchuan.com/reference/cssSection/se_attr_value_contains.php) | [title~=myWeb] | Chọn tất cả các thành phần với thuộc tính chứa một giá trị. |
| [[Thuộc tính=giá trị]](https://hocwebchuan.com/reference/cssSection/se_attr_value.php) | [target=\_blank] | Chọn tất cả các thành phần với thuộc tính bằng giá trị. |

**selector[attribute\*=value]**

vd: div[class\* = “temp”] {…}

chọn tất cả các thẻ div có class=”temp”;

vd: [class\* = “temp”] {…}

chọn tất cả các thẻ có class=”temp”;

### Float

xác định thành phần dồn về 1 phía

* left - xác định thành phần được nổi về phía bên trái.
* right - xác định thành phần được nổi về phía bên phải.
* none - xác định thành phần không được nổi về phía nào(**mặc định**).
* inherit - xác định thành phần thừa hưởng thuộc tính từ thành phần cha.

muốn xóa thuộc tính float từ thẻ nào ta chỉ cần gọi thuộc tính [clear](#_clear) trong thẻ đó

### Clear

Thuộc tính clear trong CSS có tác dụng loại bỏ các thuộc tính float (left - right) trên một thành phần. Để giải quyết vấn đề trong mục chú ý của [thuộc tính float](#_float) phía trên.

* left - xóa bỏ float: left.
* right - xóa bỏ float: right.
* both - xóa bỏ cả float:left và float: right.
* none - không xóa bỏ gì cả (**mặc định**).
* inherit - thừa hưởng giá trị từ thành phần cha.

### Position

Không gian của thẻ con(có parding) không bao gồm pading cuả thẻ cha

#### Absolute

các thẻ con trong thẻ cha có thuộc tính này sẽ không phân biệt với nhau khi di chuyển sẽ làm phá vỡ cấu trúc

các thẻ con này phải có left(right), top(botton) để xác định vị trí tương đối với thẻ cha

* muốn dùng absolute thì thẻ cha phải set position:.. . Nếu không nó sẽ lấy cha gần nhất được set position. Nếu không có cha nào thì nó sẽ lấy body

#### Relative

vd:

<!DOCTYPE html>

<html>

<head>

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

<style>

.relative{

height: 300px;

width: 300px;

margin: 100px auto;

background: blue;

position: relative;

text-align: center;

}

.absolute{

position: relative ;

height: 40px;

width: 40px;

}

.red{

top: 0px;

left: 100px;

background: red;

}

.yellow{

top: 0px;

left: 110px;

background: yellow;

}

.white{

background: white;

margin:0 auto;

}

</style>

</head>

<body>

<div class="relative">

<div class="absolute red"></div>

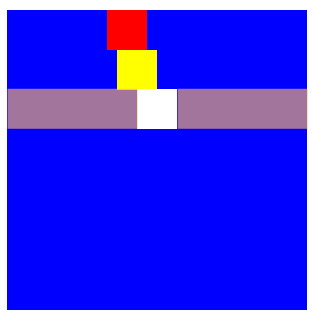
<div class="absolute yellow"></div>

<div class="absolute white"></div>

</div>

</body>

</html>



trong vd trên white là thẻ con thứ 3 nên top: 0 tưng ứng với cạnh dưới của thẻ vàng:

cụ thể cạnh trên của white sẽ cách cạnh trên của cha nó 1 đoạn: 40\*2=80px

ở đây relative đã phân vùng xác đinh cho từng thằng con. nên khi chúng di chuyển lên nhau( top có thể âm) chúng sẽ không ảnh hưởng tới nhau và không bị phá vỡ cấu trúc

#### Sự khác nhau giữ position relative và absolute

|  |  |
| --- | --- |
| absolute | relative |
|  |  |
| Ngoài magin còn có 1 vùng positon. position(left, right, top, bottom). được tính từ cạnh của cha  margin:  không có, nếu không thiết lập cả left-right( hoặc top-bottom  thiết lập left-right( hoặc top-bottom) có ảnh hưởng tới margin   * nên thiết lập heigh, weith cho phần tử   **Lưu ý:**  cài đặt position(left, right, top, bottom) càng to thì margin(left, right, top, bottom) càng nhỏ  nếu muốn căn vị trí, căn giữa,..thì phải cài đặt lại position với left-right, top-bottom   1. Nếu không biết chính xác left và right căn giữa thì cứ để **left :0; right:0 margin: auto;** | position(left, right, top, bottom)=0  bất kể di chuyển đêì sẽ có thuộc tính âm  - di chuyển sang phải ; right<0  - di chuyển lên trên: top<0  margin:  luôn chiếm hết phần dư 2 bên và thiết lập position(left, right, top, bottom) không ảnh hưởng tới margin  vd: |

#### Sticky …

### Animate …

url: <https://hocwebchuan.com/reference/cssSection/css3/pr_animation.php>

vd:

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 100px;

height: 100px;

background: red;

position: relative;

animation: myonly 2s infinite;

}

@keyframes myonly {

from {left: 0px;}

to {left: 100px; background-color: blue;}

}

</style>

</head>

<body>

<h1>The @keyframes Rule</h1>

<p><strong>Note:</strong> The @keyframes rule is not supported in Internet Explorer 9 and earlier versions.</p>

<div></div>

</body>

</html>

Vd2: key frames

.component-fade-enter-active {

  animation: bounce-in 0.5s;

}

.component-fade-leave-active {

  animation: bounce-out 2s;

}

@keyframes bounce-in {

  0% {

    transform: translatey(-25px);

    opacity: 0;

  }

  100% {

    transform: translatex(0px);

    opacity: 1;

  }

}

@keyframes bounce-out {

  100% {

    opacity: 0;

    transform: translatex(0px);

  }

}

Vd 3: keyframes

&::after {

position: absolute;

top: 0;

right: 0;

bottom: 0;

left: 0;

transform: translateX(-100%);

background-image: linear-gradient(

90deg,

rgba(#fff, 0) 0,

rgba(#fff, 0.2) 20%,

rgba(#fff, 0.5) 60%,

rgba(#fff, 0)

);

animation: shimmer 2s infinite;

content: '';

}

@keyframes shimmer {

100% {

transform: translateX(100%);

}

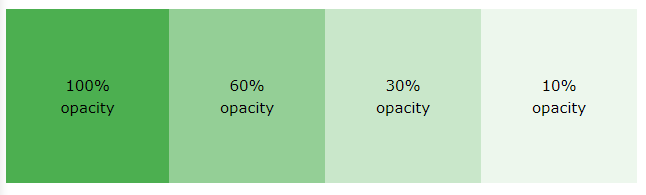
}

}

#### Transform

### RGBA

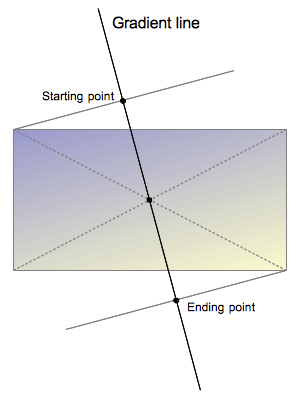
Nếu bạn không muốn áp dụng độ mờ đục cho các phần tử con, như trong ví dụ ở trên, sử dụng các giá trị màu RGBA. Ví dụ sau thiết lập độ mờ cho màu nền chứ không phải văn bản:



Ngoài RGB, CSS3 giới thiệu giá trị màu RGB với kênh alpha (RGBA) – xác định độ mờ của màu. Giá trị màu RGBA được chỉ định bằng: rgba (đỏ, xanh, xanh, alpha). Thông số alpha là một số giữa 0,0 (hoàn toàn trong suốt) và 1,0 (đục hoàn toàn).

1. div {
2. background: **rgba**(76, 175, 80, 0.3) /\* Green background with 30% opacity \*/
3. }

### Gradient



background-image: linear-gradient(

90deg, // góc của grandient line // mặc định “|” à 0deg;

rgba(#fff, 0) 0,

rgba(#fff, 0.2) 20%,

rgba(#fff, 0.5) 60%,

rgba(#fff, 0)

);// từ trái sang có 4 loại màu, loại 2 tọa độ tâm là 20% dọc theo đường gradient line màu 3 ở tọa độ 60 % dọc theo đường grandient line

&::after {

position: absolute;

top: 0;

right: 0;

bottom: 0;

left: 0;

transform: translateX(-100%);

background-image: linear-gradient(

90deg,

rgba(#fff, 0) 0,

rgba(#fff, 0.2) 20%,

rgba(#fff, 0.5) 60%,

rgba(#fff, 0)

);

animation: shimmer 2s infinite;

content: '';

}

@keyframes shimmer {

100% {

transform: translateX(100%);

}

}

}

animation: shimmer 2s infinite;

content: '';

}

@keyframes shimmer {

100% {

transform: translateX(100%);

}

}

}

### Box-sizing: border-box

Cố định kích thức border khi pading thay đổi

* Height(width) = content+ padding + border
* Nếu dom con được có position:absolute(true) top(left) tính từ padding, không tính border. Nếu không set: Box-sizing: border-box thì tính từ bên trong border

ví dụ:

|  |  |
| --- | --- |
| 01  02  03 | box-sizing: border-box;  -moz-box-sizing: border-box;  -webkit-box-sizing: border-box; |

Trong đó, nếu viết không có tiền tố là dành cho trình duyệt IE8, Opera 7, Firefox và Google chrome bản mới. -webkit là dành cho Google Chrome bản cũ và -moz là dành cho Firefox bản cũ.

### Display

#### Display: inline



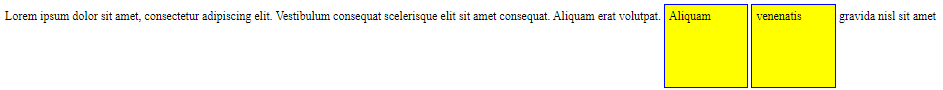
Với kiểu này thì các item sẽ nằm trên cùng một dòng. Nếu các items vượt quá độ dài của dòng thì item sẽ xuống dòng mới

Các item có kiểu display này **không thể set width và height**.

Các inline item sẽ chỉ có thể điều chỉnh margin và padding **left and right** (top và bottom thì không thể).

Là mặc định của các thẻ b, p, span

#### Display: inline-block

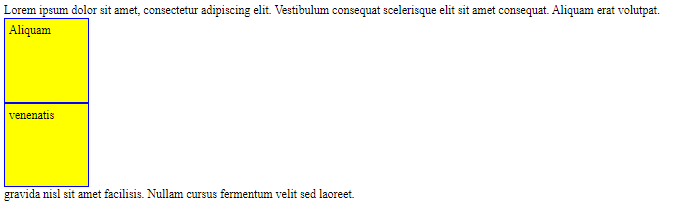


Kiểu display: inline-block sẽ được sắp xếp giống với kiểu display: inline, nghĩa là các items sẽ được xếp cùng nhau trên một dòng . Tuy nhiên các items sẽ có thuộc tính của display: block như là có set width, height, margin, padding đủ 4 hướng.

#### Display: block

Khác với kiểu display: inline thì các item có kiểu display: block luôn được xuống dòng và chiếm toàn bộ width nếu width không được set

* nó tự động xuống dòng
* Đứng sau nó cũng tự động xuống dòng
* Là mặc định của các thẻ div,..



### Flex

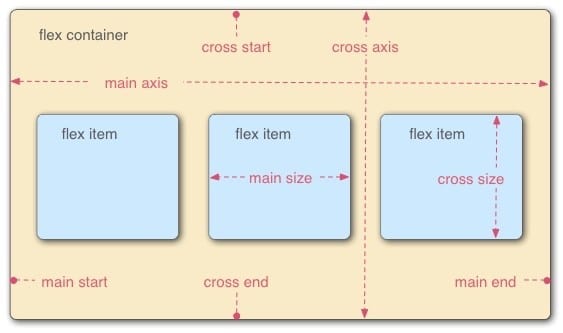
*lexbox* là một kiểu dàn trang (layout mode) mà nó sẽ tự cân đối kích thước của các phần tử bên trong để hiển thị trên mọi thiết bị. Nói theo cách khác, bạn không cần thiết lập kích thước của phần tử, không cần cho nó float, chỉ cần thiết lập nó hiển thị chiều ngang hay chiều dọc, lúc đó các phần tử bên trong có thể hiển thị theo ý muốn.

Hiện nay, theo lời khuyên từ Mozilla thì chúng ta sử dụng Flexbox để thiết lập bố cục trong phạm vi nhỏ (ví dụ như những khung trong website) và khi thiết lập bố cục ở phạm vi lớn hơn (như chia cột website) thì vẫn nên sử dụng kiểu thông thường là dàn trang theo dạng lưới (grid layout).

#### Thuật ngữ các thành phần trong Flexbox

Trước khi đi vào tìm hiểu sâu hơn về Flexbox, chúng ta cần nắm qua cấu trúc của Flexbox là như thế nào và một số thuật ngữ liên quan.

Dưới đây là sơ đồ cấu trúc Flexbox từ Mozilla Developer Network.

[](https://thachpham.com/wp-content/uploads/2016/06/flex_terms.jpg)

*Nguồn: Mozilla Developer Network*

Hai thành phần quan trọng nhất trong một bố cục Flexbox là gồm container và item:

* container: là thành phần lớn bao quanh các phần tử bên trong, bạn sẽ thiết lập kiểu hiển thị inline (sắp xếp theo chiều ngang) hoặc kiểu sắp xếp theo chiều dọc. Khi đó, các item bên trong sẽ hiển thị dựa trên thiết lập của container này.
* item: Các phần tử con của container được gọi là item, ở item bạn có thể thiết lập nó sẽ sử dụng bao nhiêu cột trong một container, hoặc thiết lập thứ tự hiển thị của nó.

Ngoài hai thành phần chính đó, chúng ta có thể thấy hình trên sẽ có:

* main start, main end: Khi thiết lập flexbox, điểm bắt đầu của container gọi là main start và điểm kết thúc được gọi là main end. Điều này có nghĩa, các item bên trong sẽ heienr thị từ main start đến main end (hoặc là được phép hiển thị đến main end). Và chiều vuông góc của nó là **cross start**, **cross end** cũng có ý nghĩa tương tự nhưng luôn vuông góc với main start, main end.
* main axis: Trục này là trục chính để điều khiển hướng mà các item sẽ hiển thị. Như bạn thấy ở trên hình main axis là trục dọc nên các item sẽ hiển thị theo chiều dọc, tuy nhiên ta có thể sử dụng thuộc tính flex-direction để thay đổi trục của main axis và lúc đó các item sẽ hiển thị theo nó. Và cross axis luôn là trục vuông góc của main axis.
* main size: Bạn có thể hiểu đơn giản là kích thước (chiều rộng hoặc dọc) của mỗi item dựa theo trục main axis.
* cross size: Là kích thước (chiều rộng hoặc dọc) của mỗi item dựa theo trục cross axis.

#### Bắt đầu với Flexbox

Trước tiên mình bắt đầu với một cấu trúc đơn giản sau:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16 | <div class="container">         <div class="item item1">1</div>         <div class="item item2">2</div>         <div class="item item3">3</div>         <div class="item item4">4</div>      </div> |

Và một đoạn CSS ban đầu để thiết lập màu sắc và kích thước để dễ nhìn từng thành phần:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16 | /\*\* Global CSS \*\*/  .container {   background: red;   max-width: 960px;   max-height: 1000px;   margin: 0 auto;   padding: 5px;  }  .item {   background: blue;   margin: 5px;   color: white;   height: 50px;   text-align: center;   line-height: 50px;  } |

Bây giờ chúng ta sẽ bắt đầu làm việc với Flexbox ở đây. Trước tiên chúng ta sẽ đưa .container về hiển thị ở dạng flexbox với display: flex.

|  |  |
| --- | --- |
| 01  02  03  04 | /\*\* Flex layout \*\*/  .container {   display: flex;  } |

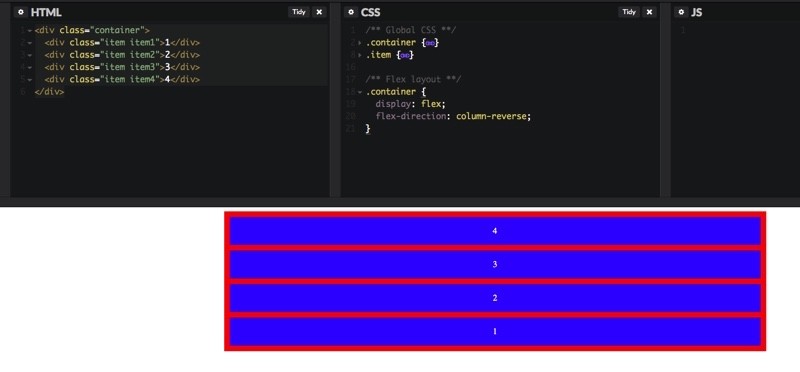
Bạn sẽ thấy các item bên trong đã tự hiển thị theo chiều dọc, tương ứng với trục main axis mặc định là chiều ngang.

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-01.jpg)

Nếu bạn muốn đổi trục thì chỉ cần thêm thuộc tính flex-direction vào container. Cụ thể:

|  |  |
| --- | --- |
| 01 | flex-direction: row | column | row-reverse | column-reverse |

* flex-direction:
  + row: Chuyển trục main axis thành chiều ngang, nghĩa là hiển thị theo hàng.
  + colum: Chuyển trục main axis thành chiều dọc, nghĩa là hiển thị theo cột.
  + row-reverse: Hiển thị theo hàng nhưng đảo ngược vị trí các item.
  + column-reverse: Hiển thị theo cột nhưng đảo ngược vị trí các item.

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-02.jpg)

Quá đơn giản đúng không?

**flex-wrap**

Bây giờ để hiểu cái này, chúng ta thử thêm chiều rộng cho mỗi item bên trong là 1000px xem chuyện gì sẽ xảy ra khi dùng flexbox.

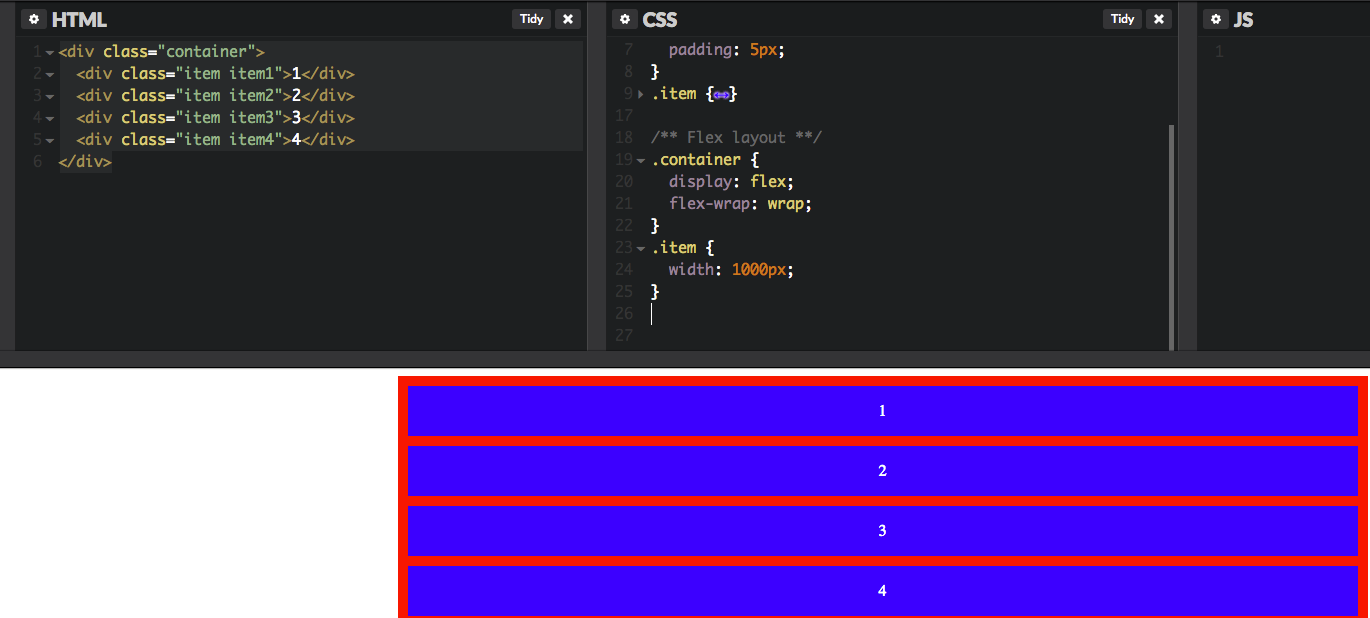
|  |  |
| --- | --- |
| 01  02  03 | .item {   width: 1000px;  } |

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-03.png)

Như bạn thấy, dù chúng ta có thêm chiều rộng cho mỗi item bên trong là 1000px nhưng nó vẫn hiển thị trên một hàng đều nhau. Lý do là mặc định, flexbox tự căn chỉnh các phần tử hiển thị đều nhau theo trục main axis của nó dựa theo chiều rộng của container. Vì vậy cho dù bạn có chỉnh chiều rộng vượt quá giới hạn của nó thì nó vẫn không bị nhảy lung tung.

Bây giờ hãy thử thêm thuộc tính flex-wrap: wrap vào container thử nhé.

|  |  |
| --- | --- |
| 01  02  03  04 | .container {   display: flex;   flex-wrap: wrap;  } |

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-04.png)

Nói nôm na là thuộc tính này cho phép container có thể bọc lại các item kể cả khi kích thước của item bị thay đổi, mặc định là nowrap. Thuộc tính này có thể áp dụng với cả chiều dọc của container và item. Các bạn xem video ở trên để rõ hơn.

**order**

Trong đoạn HTML ví dụ của mình trong bài này, mình có đặt số thứ tự cho mỗi phần tử là 1, 2, 3 và 4 với class tương ứng là .item1, .item, .item3 và .item4. Mặc định item này sẽ hiển thị theo thứ tự trong HTML, nhưng với thuộc tính order chúng ta có thể sắp xếp lại vị trí sắp xếp của các item.

Ví dụ mình có:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12 | .item1 {   order: 4;  }  .item2 {   order: 3;  }  .item3 {   order: 1;  }  .item4 {   order: 2;  } |

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-05.png)

Mặc định thứ tự sắp xếp sẽ bắt đầu từ bên trái qua phải, từ trên xuống dưới. Nếu bạn chỉnh lại trục của main axis với thuộc tính flex-direction nó sẽ thay đổi ngược lại.

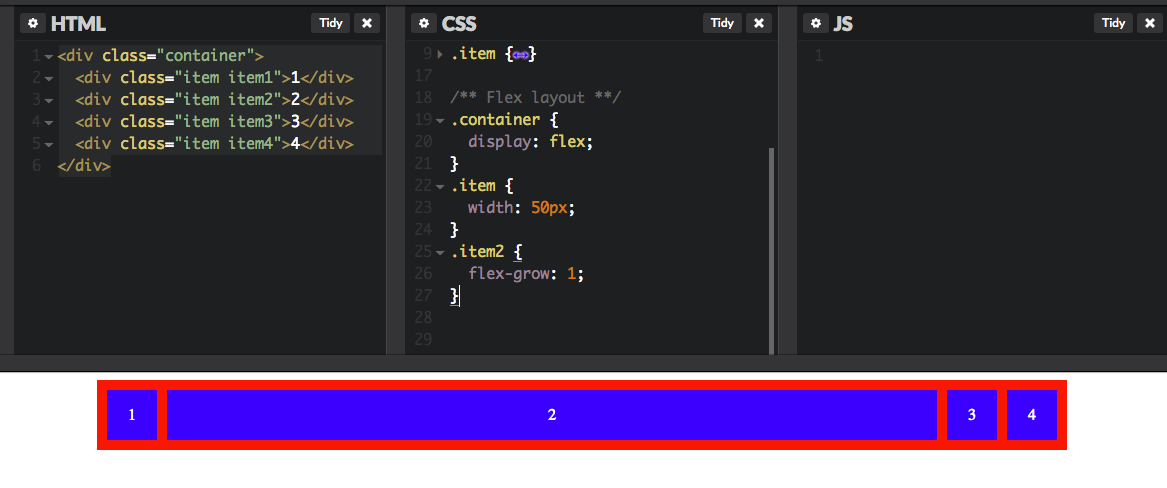
**flex-grow**

Để làm ví dụ này trước tiên mình hãy bỏ chức năng wrap đi và thiết lập chiều rộng của item là 50px.

|  |  |
| --- | --- |
| 01  02  03 | .item {   width: 50px;  } |

Bây giờ ở .item2, mình cho giá trị flex-grow là 1 thử nhé.

|  |  |
| --- | --- |
| 01  02  03 | .item2 {   flex-grow: 1;  } |

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-06.png)

Khi thiết lập nó flex-grow là 1, thì nó sẽ lấy phần trống còn lại của container đắp vào. Bây giờ hãy thử cho .item1 với flex-grow: 2 thử.

|  |  |
| --- | --- |
| 01  02  03 | .item1 {   flex-grow: 2;  } |

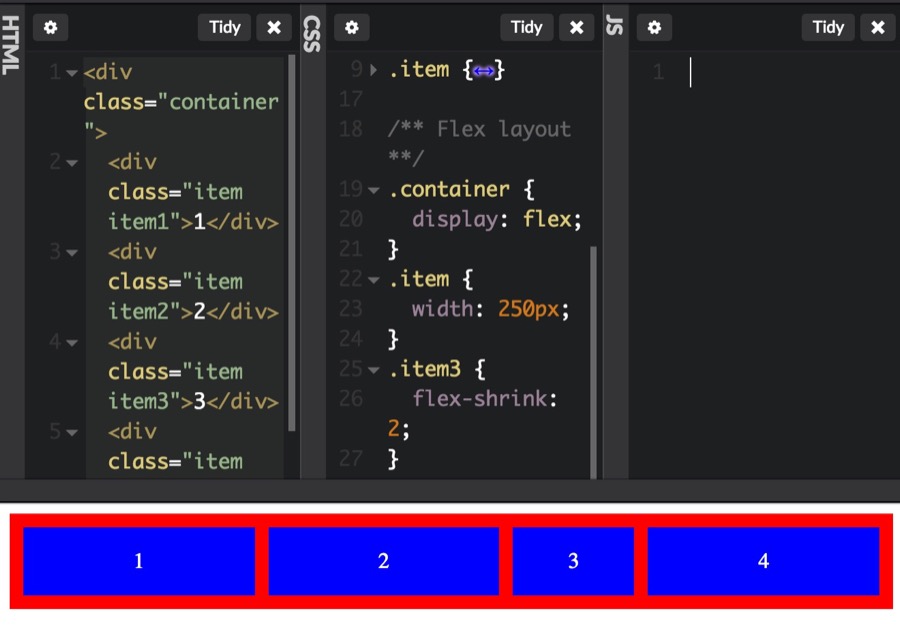
[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-07edited.jpg)

Lúc này giá trị flex-grow: 2 sẽ lấy phần dư lớn gấp đôi của flex-grow: 1.

**flex-shrink**

Bạn có thể hiểu mặc định tất cả các item đều có giá trị flex-shrink là 1. Điều này có nghĩa là khi chúng ta thu nhỏ trình duyệt lại, các phần tử đều co lại bằng nhau. Tuy nhiên giả sử mình muốn .item3 nó co lại nhiều hơn so với các item khác thì mình sẽ tăng giá trị flex-shrink của nó lên.

|  |  |
| --- | --- |
| 01  02  03 | .item3 {   flex-shrink: 2;  } |

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-08.jpg)

**flex-basis**

Cái này bạn có thể hiểu đơn giản nhất là gán cho item một kích thước nhất định. Bạn có thể sử dụng [giá trị tuyệt đối hoặc tương đối](https://thachpham.com/web-development/html-css/cac-don-vi-do-luong-trong-css.html) (căn cứ theo kích thước của container).

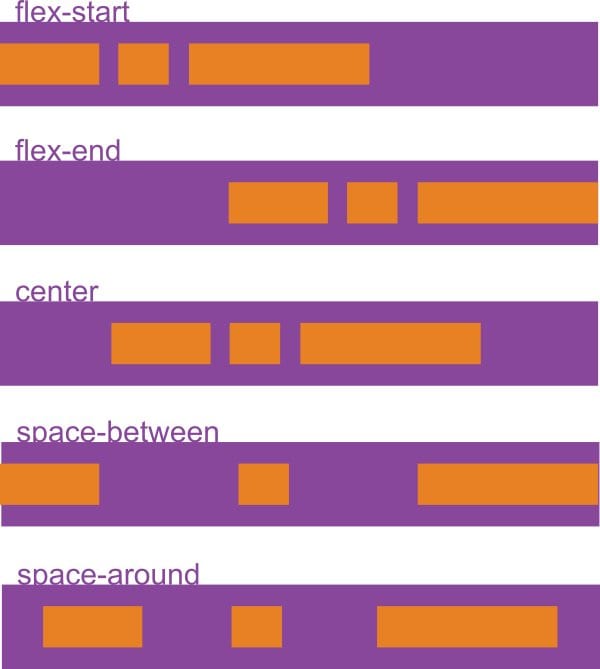
|  |  |
| --- | --- |
| 01  02  03 | .item3 {   flex-basis: 500px;  } |

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-09.jpg)

**justify-content**

Mặc định các item bên trong sẽ rải đều bắt đầu từ main start đến main end, tuy nhiên nếu container vẫn còn khoảng trống thì có thể dùng thuộc tính justify-content để điều chỉnh lại vị trí bắt đầu của nó.

Thuộc tính này có 5 giá trị và bạn có thể xem tấm ảnh bên dưới mình mượn của CSS Tricks để hiểu hơn về ý nghĩa các giá trị của justify-content.

[](https://thachpham.com/wp-content/uploads/2016/06/flexbox-justify-content.jpg)

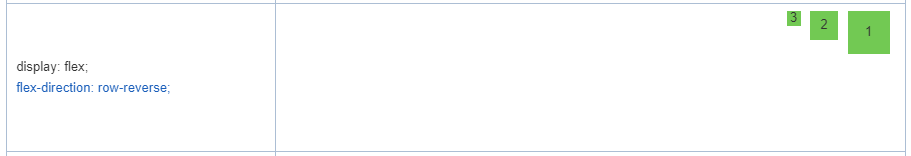
Đây là ví dụ:

|  |  |
| --- | --- |
| 01  02  03  04 | .container {   display: flex;   justify-content: flex-end;  } |

[](https://thachpham.com/wp-content/uploads/2016/06/css-flexbox-10.jpg)

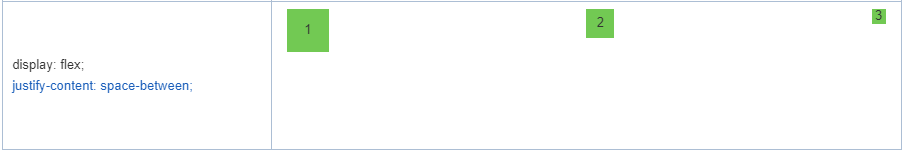
**Kết hợp giá trị**

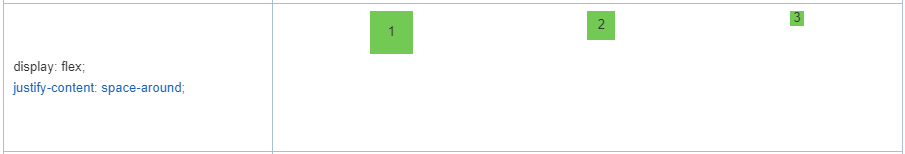








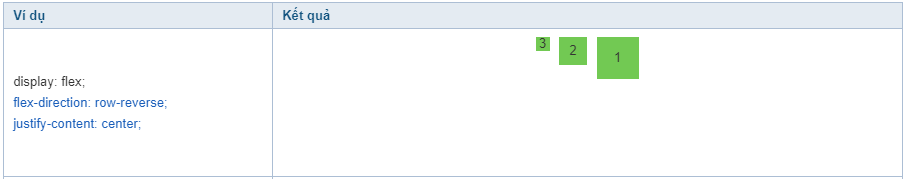




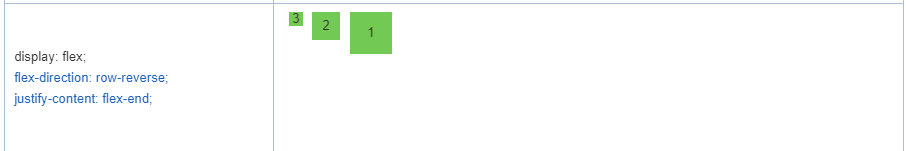


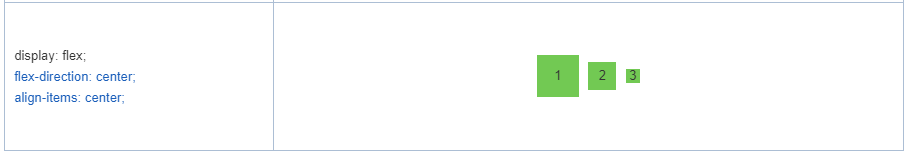


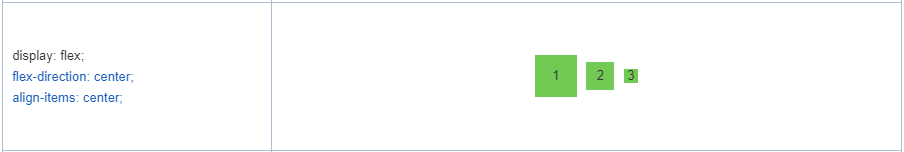


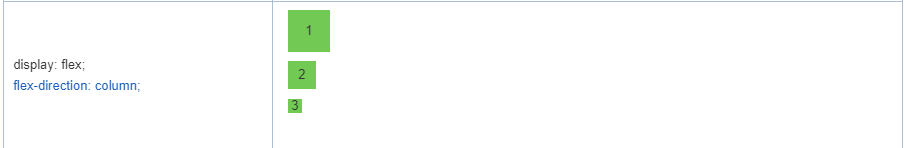


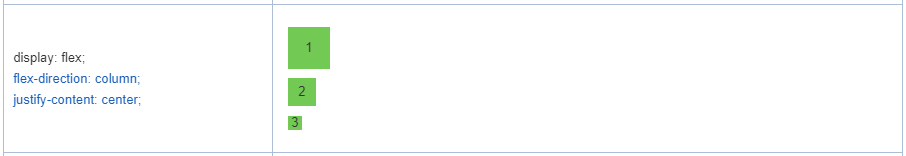




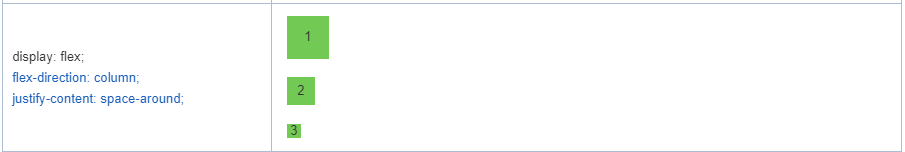


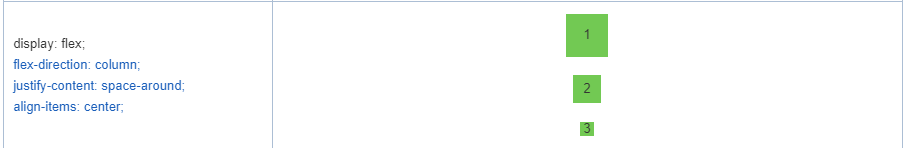


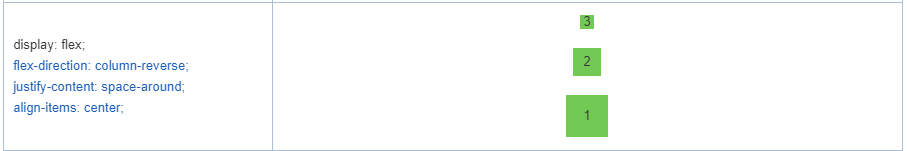




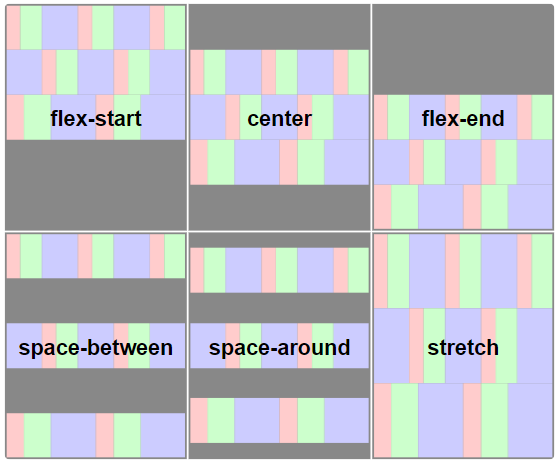








#### Align-content

****

#### **So sánh align-item và align-content**

|  |  |
| --- | --- |
| **align-content : center** | **align-item: center** |
|  |  |
| **Có tác dụng cho nhiều dòng(cột)**  **Phạm vi là giống justify-content** | **Căn giữa ở đây là căn giữa cho từng dòng** |

### Grid

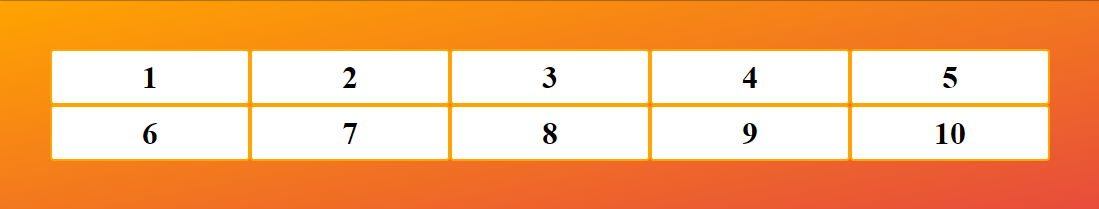
Tài liệu tham khảo : https://www.youtube.com/watch?v=t6CBKf8K\_Ac&t=1357s

#### **grid-template-columns**

có thể dùng px, %, em rem, auto hay gì tùy, muốn điền bao cột thì điền bấy nhiêu giá trị

**container {**

**grid-template-columns: 20% 20% 20% 20% 20%;**

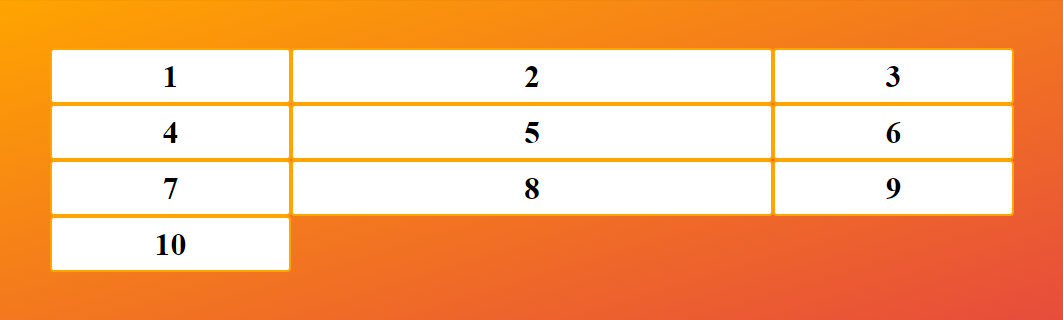
[](https://evondev.com/wp-content/uploads/2018/08/grid-template-columns1.png)

#### fraction-unit

grid-template-columns: 1fr 2fr 1fr chẳng hạn thì layout nó sẽ tạo ra 3 cột với cột giữa là 2fr sẽ gấp đôi 2 cột còn lại

**container {**

**grid-template-columns: 1fr 2fr 1fr;**

**}**[](https://evondev.com/wp-content/uploads/2018/08/grid-fr.png)

#### grid-auto-flow

Như ở trường hợp trên khi có một hoặc nhiều phần tử bị dư do không đủ cột hay hàng thì nó sẽ rớt xuống tạo nên hàng mới. Thì mình muốn set cho các phần tử bị rớt ra tạo nên Implicit Tracks có chiều cao nhất định nào đó. Chúng ta sẽ dùng thuộc tính grid-auto-flow: row kết hợp với thuộc tính grid-auto-rows để set cho các phần tử đó

grid-auto-flow: row;

grid-auto-rows: 20px; // những item rớt ra bên ngoài sẽ có height là 20px  
 grid-auto-columns:

Tương tự grid-auto-rows:

1. Nếu không set grid-template-columns thì sẽ tự động set tất cả các cột theo giá trị. Hoặc set những item dư ra

#### **grid-gap**

Thuộc tính này giúp chúng ta tạo khoảng cách giữa các phần tử với nhau theo cột và hàng. Thuộc tính này bao gồm thuộc tính grid-row-gap và grid-column-gap được viết như sau grid-gap: grid-row-gap grid-column-gap.

#### **repeat**

Hàm này giúp chúng ta tạo nhiều cột hoặc hàng có cùng giá trị một cách nhanh chóng. Ở bài trước khi chúng ta muốn tạo 5 cột bằng nhau thì chúng ta sẽ dùng grid-template-columns: 20% 20% 20% 20% 20% đúng không nào.

Thì với hàm repeat() mà CSS Grid cung cấp thì chúng ta có thể viết nó như thế này grid-template-columns: repeat(5, 20%) trong đó 5 là số cột muốn tạo còn 20% là đơn vị chúng ta sẽ dùng.

Nếu bạn muốn tạo 6 cột có cột đầu tiên và cột cuối cùng độ rộng là 10% còn 4 cột ở giữa mỗi cột 20% thì đơn giản chúng ta viết thế này grid-template-columns: 10% repeat(4,20%) 10%

#### grid-column

Thuộc tính này giúp chúng ta thiết lập vị trí và kích thước của phần tử theo cột. Nó là viết tắt của 2 thuộc tính grid-column-start và grid-column-end. Cú pháp của nó như sau grid-column: grid-column-start / grid-column-end.

Trong đó giá trị của grid-column-start và grid-column-end là các tracks line

vd:

grid {

display: grid;

height: 200px;

grid-template-columns: repeat(6, 1fr);

grid-template-rows: 200px;

}

#item1 {

background-color: #07a787;

}

#item2 {

background-color: #ffa400;

grid-column: 2 / 4;

}

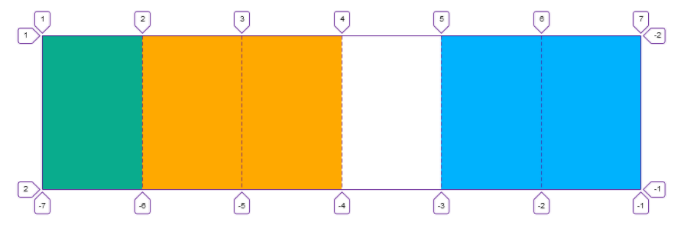
#item3 {

background-color: #00aefd;

grid-column: span 2 / 7;

}

-ttrước tiên set cho grid là 6 cột tương ứng 7 tracks line : grid-template-columns: repeat(6, 1fr);

Còn phần tử item3 có một giá trị là lạ đó là span. Và với grid-column: span 2 / 7 nên nó sẽ chiếm 2 cột tính từ track line số 7 chạy từ phải qua trái. Giá trị span sẽ tương ứng số cột mà các bạn muốn. Ví dụ span 2 nó sẽ chiếm 2 cột, span 1 là một cột.

Ở đây có 2 trường hợp. Trường hợp 1 nếu giá trị span nằm ở đầu như thế này(grid-colum-start) grid-column: span 2 / 7 thì sẽ tính ngược từ phải qua trái bắt đầu từ line số 7 và span 2(chiếm 2 cột).

Trường hợp 2 nếu grid-column: 5 / span 2  giá trị span nằm ở sau(grid-column-end) thì cũng tạo ra kết quả y Codepen ở trên. Nhưng phần tử sẽ tính từ line số 5 và chiếm 2 cột. Kết quả đều như nhau.

- tương tự vói Rows

#### grid areas

- 1 cách khác để căn bố cục thay vì dùng gird-column, grid-row

Vd:

trong grid:

.grid-2{

    display: grid;

    grid-template-columns: repeat(4,1fr) ; // vẫn cứ phải chia hàng chi cột

    grid-template-rows: repeat(3,40px) ;

    grid-template-areas: // set như map vậy

    "m h h h"

    "m c c c"

    "m f f f";

}

trong item ta chỉ việc gán với cú pháp grid-area:

.grid-2 div:nth-child(1){

  background-color: slateblue;

  grid-area: h;

}

.grid-2 div:nth-child(2){

  background-color: olivedrab;

   grid-area: m;

}

.grid-2 div:nth-child(3){

  background-color: olivedrab;

  grid-area: c;

}

.grid-2 div:nth-child(4){

  background-color: thistle;

   grid-area: f;

}



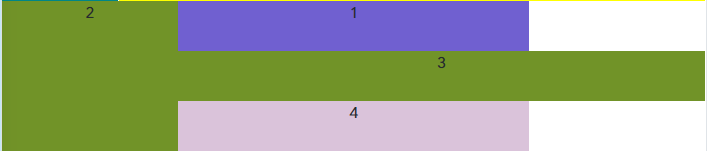
- thay dấu “.” nếu muốn để trống

 grid-template-areas:

    "m h h ."

    "m c c c"

    "m f f .";



#### grid-auto-flow

Nếu bạn có một vài grid items bạn không xác định vị trí cụ thể trên grid. Một thuật toán sắp chỗ sẽ tự động làm việc đó. Thuộc tính này sẽ xác định cách thuật toán hoạt động.

Giá trị:

**row** – (mặc định) thuật toán sẽ thêm vào theo từng hàng, thêm hàng mới nếu cần  
**column** – thuật toán sẽ thêm vào theo từng cột, thêm hàng cột nếu cần  
**dense** – thuật toán sẽ lấp vào chỗ trống với nếu vừa với các grid items đứng sau

#### justify-content

Căn phần tử theo chiều ngang khi tổng số phần tử nhỏ hơn kích thước grids

#### align-content

Căn phần tử theo chiều dọc khi tổng số phần tử nhỏ hơn kích thước grid

#### align-items

Căn chỉnh các items theo trục dọc nội tuyến (mặc định từ trên xuống dưới). Giá trị này áp dụng cho tất cả items nằm trong container.

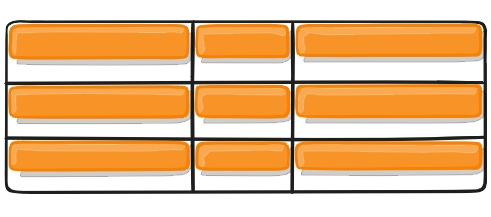
Giá trị:

start – căn chỉnh items dồn về phía cạnh bắt đầu (mặc định là cạnh trên) của cell  
end – căn chỉnh items dồn về phía cạnh kết thúc (mặc định là cạnh dưới) của cell  
center – căn chỉnh items dồn về vào giữa cell  
stretch – (mặc định) lấp đầy toàn bộ chiều rộng của cell

Ví dụ

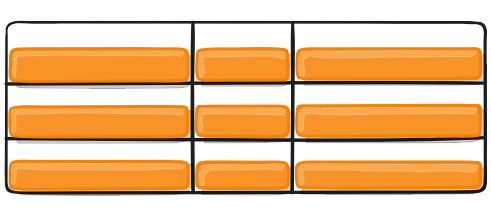
css

|  |  |
| --- | --- |
| 1  2  3 | .container {    align-items: start;  } |



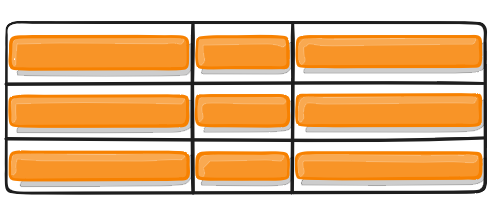
css

|  |  |
| --- | --- |
| 1  2  3 | .container {    align-items: end;  } |



css

|  |  |
| --- | --- |
| 1  2  3 | .container {    align-items: center;  } |



#### justify-items

Căn chỉnh các items theo trục ngang nội tuyến (mặc định từ trái sang phải). Giá trị này áp dụng cho tất cả items nằm trong container.

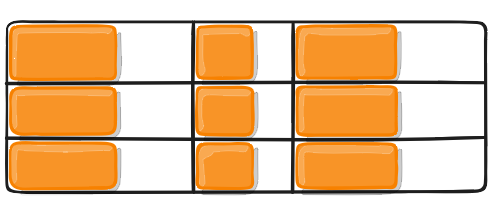
Giá trị:

**start** – căn chỉnh items dồn về phía cạnh bắt đầu (mặc định bên trái) của cell  
**end** – căn chỉnh items dồn về phía cạnh kết thúc (mặc định bên phải) của cell  
**center** – căn chỉnh items dồn về vào giữa cell  
**stretch** – (mặc định) lấp đầy toàn bộ chiều rộng của cell

Ví dụ:

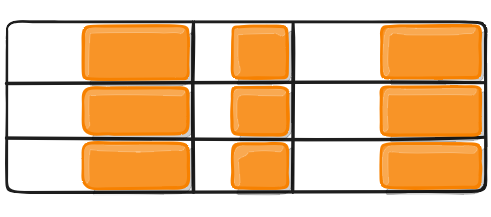
css

|  |  |
| --- | --- |
| 1  2  3 | .container {    justify-items: start;  } |



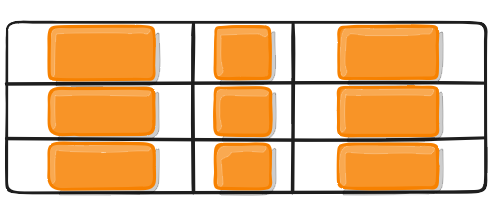
css

|  |  |
| --- | --- |
| 1  2  3 | .container {    justify-items: end;  } |



css

|  |  |
| --- | --- |
| 1  2  3 | .container {    justify-items: center;  } |



#### justify-self

template

.item {

justify-self: start | end | center | stretch;

}

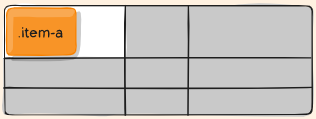
* Tương tự như justify- content nhưng là áp dụng trong 1 item

Vd:

.item-a {

justify-self: start;

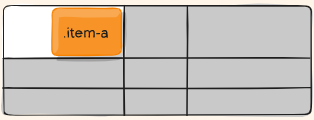
}



.item-a {

justify-self: end;

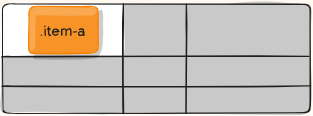
}



.item-a {

justify-self: center;

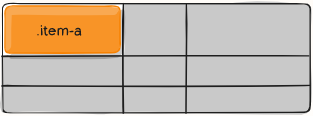
}



.item-a {

justify-self: stretch;

}



#### align-self

template

.item {

align-self: start | end | center | stretch;

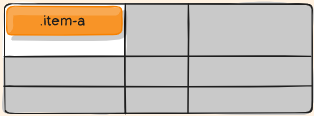
}

Tương tự như align-content nhưng mà áp dụng cho 1 item

.item-a {

align-self: start;

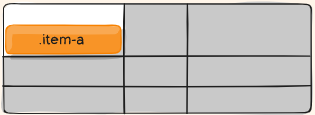
}



.item-a {

align-self: end;

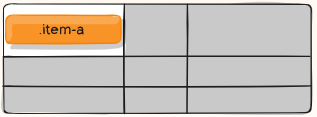
}



.item-a {

align-self: center;

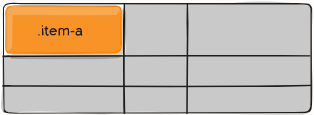
}



.item-a {

align-self: stretch;

}



#### auto-fit và autofill

.grid-container--fill {

grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));

}

.grid-container--fit {

grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));

}

.grid-element {

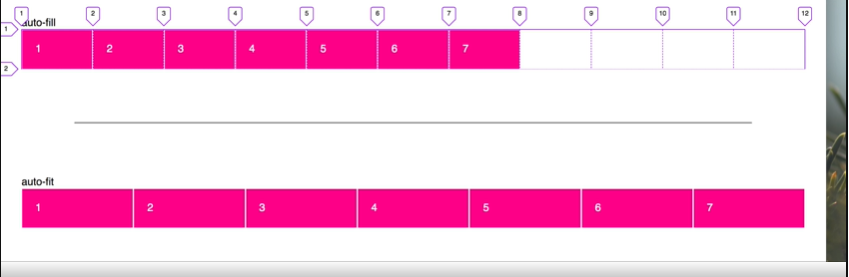
background-color: deepPink;

padding: 20px;

color: #fff;

border: 1px solid #fff;

}





Fit là đầy luôn

### background-position

background-position: 5pxcter

; background-position: 5px center

background-position-x:: 5px x;

* 1. background-position-y:: center cen

height-width

### .SCSS

#### :host

{

// nội dung css được viết trong này sẽ có hiệu lực ở chính component của nó, hay phạm vi ảnh hưởng chỉ trong component, component con sẽ không có

}

## Ajax …

## Html

### Thẻ <tr> -<table>

#### Định nghĩa và sử dụng

* Tag <tr> được dùng để tạo một hàng trong bảng HTML.
* Một tag <tr> được sử dụng bên trong [<table>](https://hocwebchuan.com/reference/tag/tag_table.php), và thường kèm với [<th>](https://hocwebchuan.com/reference/tag/tag_th.php), tất cả được chứa bên trong [<td>](https://hocwebchuan.com/reference/tag/tag_td.php), trong đó:
  + <th> xác định phần tử tiêu đề của <table>
  + <td> xác định phần tử nội dung của <table>
* Tag <tr> có thể chứa một hoặc nhiều <th> và <td>

**Sự khác nhau giữa HTML và XHTML**

|  |  |  |
| --- | --- | --- |
| **HTML4.01** | **XHTML1.0** | **XHTML1.1** |
| Không có sự khác biệt | | |

#### Cấu trúc

<tr></tr>

Ví dụ

**Html viết:**

<table>  
<tr>  
<th>A</th> // thẻ th để làm tiêu đề  
<th>B</th>  
</tr>  
<tr>  
<td>1</td>  
<td>5</td>  
</tr>  
<tr>  
<td>9</td>  
<td>10</td>  
</tr>  
</table>

**Hiển thị trình duyệt:**

|  |  |
| --- | --- |
| **A** | **B** |
| 1 | 5 |
| 9 | 10 |

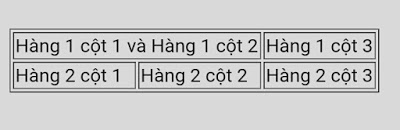
#### Thuộc tính colspan và rowspan cho thẻ Table | Học HTML

[](https://4.bp.blogspot.com/-3qtc4AvqKU4/WMuO-mbKmuI/AAAAAAAAAEw/13_Juk-WKpEpN8YHIPjoHV_xm43qJY9cwCPcB/s1600/hoc_html_2.jpg)

Ở bài này chúng tôi sẽ giới thiệu đến các bạn hai thuộc tính colspan và rowspan cho thẻ tạo bảng table.  
  
Colspan: **Thuộc tính colspan** dùng để nối hai hoặc nhiều ô với nhau tính từ ô đặt thẻ này đi tới phía trước.  
Ví dụ: áp dụng thuộc tính colspan vào bảng table

<table border= "1">  
    <tr>  
        <td colspan="2">Hàng 1 cột 1 và Hàng 1 cột 2</td>  
        <td>Hàng 1 cột 3</td>  
    </tr>  
    <tr>  
        <td>Hàng 2 cột 1</td>  
        <td>Hàng 2 cột 2</td>  
        <td>Hàng 2 cột 3</td>  
    </tr>  
</table>

Kết quả khi chạy ví dụ trên

[](https://2.bp.blogspot.com/-caXCzrb4WPI/WNsL6H2qPlI/AAAAAAAAAJ0/IS1tY6we7EIICHql2qlplhPEyFudC1a0ACLcB/s1600/IMG_20170329_081913.JPG)

Rowspan: **Thuộc tính rowspan** dùng để nối hai hoặc nhiêu ô với nhau tính từ ô đặt thẻ đi xuống phía dưới.  
  
Ví dụ: áp dụng thuộc tính rowspan và bảng table

<table border= "1">  
    <tr>  
        <td rowspan="2">Hàng 1 cột 1</td>  
        <td>Hàng 1 cột 3</td>  
        <td>Hàng 1 cột 3</td>  
    </tr>  
    <tr>  
        <td>Hàng 2 cột 2</td>  
        <td>Hàng 2 cột 3</td>  
    </tr>  
</table>

Kết quả khi chạy ví dụ trên

[](https://3.bp.blogspot.com/-MhGE9dNedeA/WNsL6n3KhjI/AAAAAAAAAJ4/_ELwLtYrW-kX0NzUCHuIpOm1VNvteRDMgCLcB/s1600/IMG_20170329_081934.JPG)

Giá trị của 2 thuộc tính ở trên là số cột hoặc hàng cần gọp.  
  
✴ Lưu ý: hai thuộc tính trên bạn phải đằt nó nằm bên trong cặp thẻ td như ở 2 ví dụ trên.  
  
Và bạn cũng có thể kết hộp 2 thuộc tính trên và cùng một bảng table như ở ví dụ sau.

<table border="1" cellspacing="0" cellpadding="5">  
  
    <tr>  
        <td rowspan="2">Hàng 1 cột 1</td>  
        <td colspan= "2">Hàng 1 cột 3</td>  
  
  
    </tr>  
    <tr>  
        <td>Hàng 2 cột 2</td>  
        <td>Hàng 2 cột 3</td>  
    </tr>  
</table>

Kết quả khi chạy ví dụ trên

[](https://4.bp.blogspot.com/-w1niqakhrGc/WNsL8Lbf0QI/AAAAAAAAAJ8/pyWH5t1ve8o5pwh5tuDRuCWYzD69-4WWACLcB/s1600/IMG_20170329_081948.JPG)

#### Kỹ thuật vẽ bảng

table tr:last-child td {

1.  border-bottom-width: 1px; // vì từng ô bị trồng lên nhau nên xóa border-bottom nhưng cell ở tr cuối thì đữ lại

}

table td {

1. 
2.  border: 1px solid #dee2e6;
3. ~~ border-bottom-width: 0;~~
4.  border-left-width: 0;

}

**! Tương tự với: border-left-width**

### Thẻ <FORM>

url tham khảo: <https://timoday.edu.vn/bai-25-the-form-trong-html/>

#### action

**Thuộc tính action** xác định hành động được thực hiện khi form được gửi đi khi người sử dụng kích nút submit. Cách phổ biến nhất khi gửi dữ liệu của form tới server là sử dụng một nút submit. Thông thường form được gửi tới một trang web chạy trên máy chủ web. Trong ví dụ trên, một script chạy ở phía server được xác định để xử lý  form đã gửi:

<form **action=**"action\_page.php">

Nếu thuộc tính action bỏ qua, action sẽ được thiết lập tới trang hiện tại.

#### method

Thuộc tính **method**xác định kiểu phương thức HTTP (**GET**hoặc **POST**) được sử dụng gửi dữ liệu trên form:

<form action="action\_page.php" **method=**"get">

hoặc:

<form action="action\_page.php" **method=**"post">

##### get – sử dụng khi nào?

Bạn có thể sử dụng GET (mặc định phương thức này). Sử dụng GET nếu việc gửi form là thụ động (giống như bạn thực hiện truy vấn trên các máy tìm kiếm) và dữ liệu không cần mã hoá, không chứa các thông tin nhạy cảm như mật khẩu, v.v. Khi bạn sử dụng GET, dữ liệu form sẽ bị nhìn thấy trên thanh địa chỉ của trang như ví dụ dưới:

action\_page.php?txtHoDem=Trần+Hùng&txtTen=Dũng

|  |  |
| --- | --- |
|  | GET là phù hợp nhất với số lượng dữ liệu ít. Giới hạn về kích thước dữ liệu GET được thiết lập trong trình duyệt của bạn. |

##### post – sử dụng khi nào?

Bạn nên sử dụng POST: Trong trường hợp nếu form cập nhật dữ liệu hoặc dữ liệu trên form gửi đi bao gồm các thông tin nhạy cảm như mật khẩu, mã thẻ ngân hàng,v.v. POST cung cấp cơ chế bảo mật hơn bởi vì dữ liệu được gửi đi không được hiển thị trên thanh địa chỉ của trang.

#### name

Để lấy được dữ liệu đúng khi gửi đi, mỗi trường phải có một thuộc tính name. Ví dụ này sẽ chỉ gửi trường “Họ đệm”, theo bạn tại sao lại không gửi trường “Tên”?:

Chỉ định một tên được sử dụng để nhận diện các thành phần trên form (với DOM sử dụng: document.forms.name).

Ví dụ

<!DOCTYPE html>

<html>

<body><form action="action\_page.php">

Họ đệm:<br>

<input type="text" name="txtHoDem" value="Trần Hùng"><br>

Tên:<br>

<input type="text" value="Dũng"><br>

<input type="submit" value="Submit">

</form>

</body>

</html>

#### <fieldset> **<legend>**

Phần từ <fieldset> nhóm dữ liệu liên quan trong một form. Phần tử <legend> xác định một phụ đề cho phần tử <fieldset>.

vd: [file:///C:/Users/BlueMen/Desktop/Lap\_trinh\_web/Unit1.html](../../Users/BlueMen/Desktop/Lap_trinh_web/Unit1.html)

<form action="action\_page.php">

        <fieldset>

            <legend>Thông tin cá nhân:</legend>

            Họ đệm:<br>

            <input type="radio" name="firstname" value="Dũng"><br> Tên:

            <br>

            <input type="radio" name="lastname" value="Trần Hùng"><br>

            <input type="submit" value="Submit">

        </fieldset>

        <fieldset>

            <input type="radio" name="firstname1" value="Dũng1"><br> Tên:

            <br>

            <input type="radio" name="lastname1" value="Trần Hùng1"><br>

            <input type="submit" value="Submit">

        </fieldset>

    </form>

#### autocomplete\*

#### enctype\*

#### novalidate\*

#### targetse\*

### thẻ <output>

### Thẻ <INPUT>

url tham khảo: <https://timoday.edu.vn/bai28-cac-thuoc-tinh-cua-the-input/#Thuoc_tinh_formaction>

#### type

##### text

**<input type=”text”>** sử dụng cho nhập dữ liệu đầu vào cho đầu vào dạng văn bản (một dòng):

##### password

**<**input type=”password”> định nghĩa một trường password:

Các kí tự trong trường password được ẩn đi (nó được mã hóa hiển thị thành hình sao hoặc các chấm tròn).

##### Submit <important!>

[<input>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input)các yếu tố của loại **submit**được hiển thị dưới dạng các nút. Khi [click](https://developer.mozilla.org/en-US/docs/Web/Events/click)sự kiện xảy ra (thường là do người dùng nhấp vào nút), [tác nhân người dùng sẽ](https://developer.mozilla.org/en-US/docs/Glossary/user_agent) cố gắng gửi biểu mẫu đến máy chủ.

<input type="submit" value="Send Request">

Nếu bạn không chỉ định value, nút sẽ có nhãn mặc định, được chọn bởi tác nhân người dùng. Nhãn này có thể là một cái gì đó dọc theo dòng "Gửi" hoặc "Gửi truy vấn." Dưới đây là ví dụ về nút gửi với nhãn mặc định trong trình duyệt của bạn:

<input type="submit">

**<input type=”submit”>** định nghĩa một nút để gửi dữ liệu từ form người sử dụng nhập tới nơi xử lý dữ liệu của form này (**form-handler**)

Form-handler thường mà một trang chạy ở phía server, được viết bằng các ngôn ngữ ở phía máy chủ như PHP, ASP.NET, JSP … cho xử lý dữ liệu đầu vào.

Form-handler được chỉ định bởi các thuộc tính  “action” trong form đó:

Ví dụ

<html>

<body>

<form action="action\_page.php">

Họ đệm:<br>

<input type="text" name="txtHoDem" value="Trần Hùng"><br>

Tên:<br>

<input type="text" name="txtTen" value="Dũng"><br>

<input type="submit" value="Submit">

</form>

</body>

</html>

Nếu bạn kích nút “Submit” dữ liệu trên form sẽ được gửi tới trang “action\_page.php”. Phía máy chủ sẽ thực hiện xử lý dữ liệu gửi và đưa ra đầu ra. Về xử lý phía server bạn có thể sử dụng [ngôn ngữ lập trình PHP](http://timoday.edu.vn/category/khoa-hoc/php/) Đoạn code HTML ở trên sẽ được hiển thị trên trình duyệt

##### radio

**<input type=”radio”>** định nghĩa một nút  **radio.**

Các nút Radio cho phép người sử dụng chọn duy nhất một lựa chọn trong danh sách các lựa chọn:

<input type="radio" name="gender" value="male" checked> Male<br>

Các radio trong cùng 1 form sẽ được lựa cọn duy nhất 1 radio!

##### checkbox

**<input type=”checkbox”>** định nghĩa một nút **checkbox**.

Checkboxes cho phép người sử dụng chọn không hoặc nhiều lựa chọn trong các lựa chọn đưa ra .

##### button

**<input type=”button”>** định nghĩa một **button**:

<input type="button" onclick="alert('Website Tìm ở đây chào bạn!')" value="Kíchh Tôi!"></body>

! nên chọn botton thay vì input type=”botton”

##### Nunber

Thẻ**<input type=”number”>** được sử dụng cho trường dữ liệu đầu vào có chứa các giá trị là số.

Bạn có thể cài đặt các giới hạn trong kiểu đầu vào này.

<input type="number" name="quantity" min="1" max="5">

##### date (hỗ trợ bảng chọn)

Thẻ **<input type=”date”>** được sử dụng khi trường dữ liệu đầu vào chứa các giá trị là ngày.

<input type="date" name="bday" max="1979-12-31">

##### color (hỗ trợ bảng chọn)

Thẻ **<input type=”color”>** được sử dụng khi trường dữ liệu đầu vào chứa các giá trị là màu sắc.

##### Range

Thẻ **<input type=”range”>** được sử dụng khi trường dữ liệu đầu vào chứa dữ liệu là giá trị trong phạm vi nhất định.

<input type="range" name="points" min="0" max="10">

##### month

Thẻ **<input type=”month”>** cho phép người sử dụng có thể lựa chọn một tháng và năm.

##### week

Thẻ **<input type=”week”>** cho phép người sử dụng lựa chọn tuần và năm.

##### time

Thẻ **<input type=”time”>**cho phép người dùng lựa chọn một thời gian (không có múi giờ).

##### datetime-local (hỗ trợ bảng chọn)

Thẻ **<input type=”datetime-local”>** cho phép người sử dụng lựa chọn ngày và thời gian (không có múi giờ).

##### email

Thẻ **<input type=”email”>** được sử dụng cho các trường dữ liệu đầu vào có chứa một địa chỉ e-mail.

Phụ thuộc vào sự hỗ trợ của các trình duyệt, địa chỉ e-mail có thể được tự động xác nhận khi gửi đến.

Một vài smartphones có thể nhận ra các loại email, và bổ sung thêm “.com” vào bàn phím để phù hợp với dữ liệu đầu vào của email.

##### search

Thẻ **<input type=”search”>**được sử dụng cho phép người dùng tìm kiếm các trường (một trường search hoạt động thông thường như một trường text).

##### Tel

Thẻ **<input type=”tel”>** được sử dụng cho các trường đầu vào có dữ liệu chứa một số điện thoại.

Phần tử  tel hiện nay chỉ được hỗ trợ trong trình duyệt Safari 8.

##### url

Thẻ **<input type=”url”>** được sử dụng cho các trường đầu vào mà chứa địa chỉ URL.

Phụ thuộc vào sự hỗ trợ của các trình duyệt, trường địa chỉ url có thẻ tự động xác nhận khi gửi đến.

Một vài smartphones nhận ra kiểu đầu vào url, và bổ sung thêm “.com” vào bàn phím để phù hợp với địa chỉ đầu vào url.

##### reset

reset lại các phần tử trong thẻ form có đi kèm event reset. [vd xem dưới đây](#_onreset)

##### giới hạn đầu vào của các thẻ input

chủ yếu với các thẻ number , range,..

|  |  |
| --- | --- |
| **Thuộc tính** | **Mô tả** |
| disabled | Xác định trường dữ liệu đầu vào bị vô hiệu hóa |
| max | Xác định giá trị lớn nhất cho trường dữ liệu đầu vào |
| maxlength | Xác định số lượng kí tự tối đa cho trường dữ liệu đầu vào |
| min | Xác định giá trị nhỏ nhất cho trường dữ liệu đầu vào |
| pattern | Xác định biểu thức chính quy để kiểm tra giá trị cho trường dữ liệu đầu vào |
| readonly | Xác định trường dữ liệu đầu vào là chỉ đọc (không thể thay đổi) |
| required | Xác định các yêu cầu của trường dữ liệu đầu vào(trường không thể để trống) |
| size | Xác định độ rộng (theo các kí tự) của trường dữ liệu đầu vào |
| step | Xác định các khoảng số hợp lí cho các trường dữ liệu đầu vào |
| value | Xác định giá trị mặc định cho trường dữ liệu đầu vào |

#### readonly

Thuộc tính **readonly** chỉ đinh phần tử input chỉ có thể đọc (không thể thay đổi dữ liệu trên phần tử đó):

#### disabled

Thuộc tính **disabled** chỉ định trường input sẽ bị ẩn. Phần tử **disabled**sẽ không được sử dụng và không kích được trên nó. Các phần tử **disabled** sẽ không được gửi đi (submit).

#### size

Thuộc tính **size** chỉ định kích thước của trường input (số ký tự):

(độ rộng của khung input, size càng nhỏ thì khung càng ngắn)

#### maxlength

Thuộc tính **maxlength** chỉ định độ dài tối đa cho phép của trường input:

Vd: cho nhập tối đa 10 ký tự

<input type="text" name="firstname" maxlength="10">

#### value autocomplate

Thuộc tính autocomplete quy định một form hoặc một thẻ input trong form có được sử dụng chức năng tự động điền, gợi ý thông tin hay không.

Khi thuộc tính autocomplete có giá trị “on”, trình duyệt sẽ tự động gợi ý những giá trị dựa trên thông tin người dùng nhập vào.

**Mẹo:** Bạn có thể thiết lập giá autocomplete là “on” cho thẻ form và “off” cho một vài thẻ input hoặc ngược lại.

Thuộc tính autocomplete có thể sử dụng trong thẻ <form> và thẻ <input> với các kiểu: text, search, url, tel, email, password, datepickers, range và color.

url demo: <https://webvn.com/example/vi-du-thuoc-tinh-autocomplete/>

#### placeholder

Thuộc tính placeholder được sử dụng để đưa ra những gợi ý cho người dùng trước khi nhập liệu vào thẻ.

Thuộc tính placeholder được sử dung trong thẻ input với các type có giá trị: text, search, url, tel, email và password.

Thuộc tính placeholder trong thẻ input không được hỗ trợ trong Internet Explorer 9 và các phiên bản trước.

url demo: <https://webvn.com/example/vi-du-thuoc-tinh-placeholder/>

#### for

vd: <label for="username">Click me</label>

<input type="text" id="username">

Liên kết thẻ input vào trong label cho phép vùng nhấn tăng lên,

* For giống tên thuộc tính id của thẻ input

#### form

Thuộc tính form form để chỉ định một hay nhiều form mà chứa  phần tử <input> này. **Mách nhỏ:** Để ám chỉ nhiều hơn một form, sử dụng khoảng trắng để phân tách danh sách các id của form.

Thẻ sẽ được thêm vào dưới cùng của form

Ví dụ

Trường Last name được đặt bên ngoài form HTML (nhưng vẫn là một phần của form):

Thẻ input ám chỉ là 1 thành phần của form 1, nếu muốn thêm vào form có id là form2, form3 thì ta viết:

Last name: <input type="text" name="lname" form="form1 form2 form3">

<!DOCTYPE html>

<html>

<body>

<form action="action\_page.php" id="form1">

First name: <input type="text" name="fname"><br>

<input type="submit" value="Submit">

</form>

<p>Trường "Last name" bên dưới là ở bên ngoài phần tử form nhưng nó vẫn là một phần của form.</p>

Last name: <input type="text" name="lname" form="form1">

</body>

</html>

#### formaction

Thuộc tính formaction chỉ định tới đường dẫn của một file mà sẽ xử lý dữ liệu của các control input khi form được submit. Thuộc tính formaction viết chồng với thuộc tính action của phần tử <form>. Thuộc tính formaction được sử dụng cho các thẻ input kiểu type=”submit” và type=”image”.

**Ví dụ**

Một form HTML với hai nút submit, thực hiện hai hành động khác nhau:

<!DOCTYPE html>

<html>

<body>

<form action="action\_page.php">

First name: <input type="text" name="fname"><br>

Last name: <input type="text" name="lname"><br>

<input type="submit" value="Submit"><br>

<input type="submit" formaction="demo\_admin.php" value="Submit as admin">

</form>

<p><strong>Chú ý:</strong> Thuộc tính formaction của thẻ input không hỗ trợ trong Internet Explorer 9 và các phiên bản cũ hơn của IE.</p>

</body>

</html>

#### formenctype

Thuộc tính formenctype xác định cách dữ liệu form nên được mã hoá khi gửi dữ liệu tới server (chỉ cho form với method=”post”). Thuộc tính formenctype viết chồng với thuộc tính enctype của thẻ <form>. Thuộc tính formenctype được sử dụng với các thẻ input kiểu type=”submit” và  type=”image”.

**Ví dụ**

Gửi dữ liệu của form với mã hoá mặc định (cho nút submit thứ nhất) và mã hoá kiểu “multipart/form-data” (cho nút submit thứ hai):

<!DOCTYPE html>

<html>

<body>

<form action="demo\_post.php" method="post">

First name: <input type="text" name="fname"><br>

<input type="submit" value="Submit">

<input type="submit" formenctype="multipart/form-data" value="Submit as Multipart/form-data">

</form>

<p><strong>Chú ý:</strong> Thuộc tính formenctype của thẻ input không hỗ trợ trong Internet Explorer 9 và các phiên bản cũ hơn của IE.</p>

</body>

</html>

#### formmethod

Thuộc tính formmethod định nghĩa phương thức HTTP cho gửi dữ liệu của form tới một URL thực hiện nó (đường dẫn file chứa code server). Thuộc tính formmethod viết chồng thuộc tính method của phần tử <form>. Thuộc tính formmethod có thể được sử dụng với <input> kiểu type=”submit” and type=”image”.

**Ví dụ**

Nút submit thứ hai viết chồng method HTTP của form:

<!DOCTYPE html>

<html>

<body>

<form action="action\_page.php" method="get">

First name: <input type="text" name="fname"><br>

Last name: <input type="text" name="lname"><br>

<input type="submit" value="Submit">

<input type="submit" formmethod="post" formaction="demo\_post.php" value="Submit using POST">

</form>

<p><strong>Chú ý:</strong> Thuộc tính formmethod của thẻ input không hỗ trợ trong Internet Explorer 9 và các phiên bản cũ hơn của IE.</p>

</body>

</html>

#### formnovalidate

Thuộc tính novalidate là một thuộc tính kiểu boolean. Khi xuất hiện trong phần tử <input> thì phần tử đó sẽ không bị kiểm tra tính hợp lệ khi gửi dữ liệu. Thuộc tính formnovalidate viết chồng thuộc tính novalidate của phần tử <form>. Thuộc tính formnovalidate có thể được sử dụng với thẻ <input> kiểu type=”submit”.

Ví dụ  
Một form với hai nút submit (với việc có và không có kiểm tra dữ liệu hợp lệ):

<!DOCTYPE html>

<html>

<body>

<form action="action\_page.php">

E-mail: <input type="email" name="email"><br>

<input type="submit" value="Submit"><br>

<input type="submit" formnovalidate value="Submit without validation">

</form>

<p><strong>Chú ý:</strong> Thuộc tính formnovalidate của thẻ input không hỗ trợ trong Internet Explorer 9 và các phiên bản cũ hơn của IE hoặc Safari.</p>

</body>

</html>

#### name

tên biến, chính là key trong key= value khi gửi dũ liệu

#### formtarget

Thuộc tính *formtarget* chỉ định một tên hoặc một từ khoá mà cho biết nơi để hiển thị kết quả khi nhận được sau khi submit form. Thuộc tính *formtarget* viết đề thuộc tính *target* của phần tử <form>. Thuộc tính formtarget có thể được sử dụng với thẻ <input> kiểu type=”submit” và type=”image”.

Ví dụ

<!DOCTYPE html>

<html>

<body>

<form action="action\_page.php">

First name: <input type="text" name="fname"><br>

Last name: <input type="text" name="lname"><br>

<input type="submit" value="Submit as normal">

<input type="submit" formtarget="\_blank" value="Submit to a new window/tab">

</form>

<p><strong>Chú ý:</strong> Thuộc tính formtarget không hỗ trợ trong Internet Explorer 9 và các phiên bản cũ hơn của IE.</p>

</body>

</html>

#### height và width

Thuộc tính height và width chỉ định chiều rộng và cao của phần tử <input>. Thuộc tính height và width chỉ được sử dụng với <input type=”image”>.

|  |  |
| --- | --- |
| [Mách nhỏ](https://i1.wp.com/timoday.edu.vn/wp-content/uploads/2015/09/lamp.jpg) | Luôn luôn chỉ định kích thước của ảnh. Nếu trình duyệt không biết kích thước, trang nhấp nháy khi loạt các ảnh. |

Ví dụ

<!DOCTYPE html>

<html>

<body>

<form action="action\_page.php">

First name: <input type="text" name="fname"><br>

Last name: <input type="text" name="lname"><br>

<input type="image" src="img\_submit.gif" alt="Submit" width="48" height="48">

</form>

</body>

</html>

#### list (~combobox)

Thuộc tính list tham chiếu đến phần tử <datalist> mà chứa các giá trị trị tuỳ chọn định nghĩa trước cho một phần tử <input>.

Giống như 1 combobox

**Ví dụ**

Phần tử <input> với các giá trị định nghĩa trước trong một <datalist>:

<!DOCTYPE html>

<html>

<body>

<form action="action\_page.php" method="get">

<input list="browsers" name="browser">

<datalist id="browsers">

<option value="Internet Explorer">

<option value="Firefox">

<option value="Chrome">

<option value="Opera">

<option value="Safari">

</datalist>

<input type="submit">

</form>

<p><b>Chú ý:</b> Thẻ datalist không hỗ trợ trong Internet Explorer 9 và các phiên bản cũ hơn của IE hoặc Safari.</p>

</body>

</html>

#### <option> của <datalist>

Khác 1 chút so với [<option>](#_<option>) của [<select>](#_Thẻ_<SELECT>)

không sử dụng nội dung bên trong thẻ <option>

không cần thẻ đóng </option>

##### value

option sẽ hiển thị nội dung chính từ value. thẻ option không có value sẽ hiển thị nội dung bên trong thẻ <option>nôi dung</option>

##### label

nội dung của thẻ label sẽ được hiển thị bên dưới nội dung của thẻ value khi chúng ta show phần datalist, giống như phần mô tả vậy

##### title và selected

không có ý nghĩa trong này

##### disabled

thay vì làm mờ và không thể thao tác thì option đó sẽ mất muôn trong datalist

#### multiple

Thuộc tính multiple là một thuộc tính kiểu boolean. Khi xuất hiện, phần tử sẽ cho người sử dụng nhập hơn một giá trị trong phần tử<input>. Thuộc tính multiple làm việc với các kiểu input sau: email và file.

Ví dụ

Một trường upload file chấp nhận nhiều giá trị, có thể lấy 1 lúc nhiều file, cò không khai báo multiple thì chỉ chọn được 1 file:

<!DOCTYPE html>

<html>

<body>

<form action="action\_page.php">

Select images: <input type="file" name="img" multiple>

<input type="submit">

</form>

<p>Thử chọn nhiều hơn một file khi browser các file.</p>

</body>

</html>

#### min và max

Thuộc tính min và max xác định giá trị nhỏ nhất và lớn nhất cho một phần tử <input>. Các thuộc tính min và max làm việc với các kiểu thẻ <input> sau: number, range, date, datetime, datetime-local, month, time and week.

Ví dụ

Các phần tử <input> với các giá trị min và max, xem lại [bài 27](http://timoday.edu.vn/bai-27-cac-kieu-input-trong-html/):

<!DOCTYPE html>

<html>

<body>

<form action="action\_page.php">

Enter a date before 1980-01-01:

<input type="date" name="bday" max="1979-12-31"><br>

Enter a date after 2000-01-01:

<input type="date" name="bday" min="2000-01-02"><br>

Quantity (between 1 and 5):

<input type="number" name="quantity" min="1" max="5"><br>

<input type="submit">

</form>

<p><strong>Chú ý:</strong> Thuộc tính max và min của thẻ input không hỗ trợ trong Internet Explorer 9 và các phiên bản cũ hơn của IE hoặc Firefox.</p>

<p><strong>Chú ý:</strong> Thuộc tính max và min sẽ không làm việc với ngày tháng và thời gian trong Internet Explorer 10.</p>

</body>

</html>

#### pattern

huộc tính pattern chỉ định một biểu thức chính quy (regular expression) mà giá trị của phần tử <input> sẽ được kiểm tra lại. Thuộc tính pattern làm việc với các loại thẻ <input> sau: text, search, url, tel, email, và password.

**Mách nhỏ:** Sử dụng thuộc tính title để trợ giúp thông tin pattern cho người sử dụng.

**Ví dụ**

Một trường input có thể chỉ chứa 3 ký tự (không phải kiểu số hoặc các ký tự đặc biệt):

<!DOCTYPE html>

<html>

<body>

<form action="action\_page.php">

Country code: <input type="text" name="country\_code" pattern="[A-Za-z]{3}" title="Three letter country code">

<input type="submit">

</form>

<p><strong>Chú ý:</strong> Thuộc tính pattern của thẻ input không hỗ trợ trong Internet Explorer 9 và các phiên bản cũ hơn của IE hoặc Safari.</p>

</body>

</html>

#### required

Thuộc tính required là một thuộc tính kiểu boolean. Khi xuất hiện nó xác định rằng trường input không được để trống trước khi submit form. Thuộc tính required làm việc với thẻ <input> kiểu: text, search, url, tel, email, password, date pickers, number, checkbox, radio, và file.

#### step

Thuộc tính step xác định bước nhảy của một giá trị số cho phần tử <input>. Ví dụ: nếu step=”3″, các số có thể là -3, 0, 3, 6, v.v.  
**Mách nhỏ:** Thuộc tính step có thể được sử dụng cùng với các thuộc tính max và min để tạo khoảng các giá trị hợp lệ. Thuộc tính step làm việc với các thẻ <input> kiểu: number, range, date, datetime, datetime-local, month, time và week.

! thường dùng khi nhập giá trị tiền tệ với step 1000. Vì 1000 VNĐ là đơn vị nhỏ nhất

**Ví dụ**

Trường input với chỉ định bước nhảy của một giá trị số:

<input type="number" name="points" step="3">

#### data-lpignore…

#### data-testid…

#### role…

#### aria-autocomplete…

#### aria-expanded…

#### aria-control…

### Thẻ <SELECT>

Tag <select> được dùng để tạo một danh sách chọn lựa (danh sách thả xuống).

Tag [<option>](https://hocwebchuan.com/reference/tag/tag_option.php) bên trong thành phần select, định nghĩa giá trị tùy chọn trong danh sách.

Tag <select> và <option> được thường được đặt trong [<form>](https://hocwebchuan.com/reference/tag/tag_form.php) cho người dùng lựa chọn danh sách.

Cách sử dụng: <select thuoctinh="giatri"></select>

#### disabled

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Giá trị** | **Ví dụ** | **Mô tả** |
| [disabled](_blank) | disabled | disabled="disabled" | Xác định danh sách thả xuống không hiển thị. |

#### multiple

thay vì hiển thị 1 ô , nó sẽ hiển thị tất cả các tùy chọn

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Giá trị** | **Ví dụ** | **Mô tả** |
| [multiple](_blank) | multiple | multiple="multiple" | Hiển thị nhiều tùy chọn. |

**Khi lấy giá trị :**  var foo = $('#multiple').val(); => foo là 1 list

#### name

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Giá trị** | **Ví dụ** | **Mô tả** |
| [name](_blank) | name | name="selName" | Xác định tên cho danh sách lựa chọn. |

#### size

thay vì hiển thị 1 ô nó sẽ hiển thị theo số size

theo ví dụ dưới thì sẽ nhìn thấy 1 lúc 3 tùy chịn

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Giá trị** | **Ví dụ** | **Mô tả** |
| [size](_blank) | Số | size="3" | Xác định số tùy chọn có thể thấy trong danh sách lựa chọn. |

#### <option>

##### disabled

Giống như Thuộc tính **disabled** của trường input. Nó sẽ bị ẩn. Phần tử **disabled**sẽ không được sử dụng và không kích được trên nó. Các phần tử **disabled** sẽ không được gửi đi (submit).

##### label

nếu không có text trong thẻ option thì nội dung sẽ được hiển thị của label

vd: nếu bỏ chữ “Danh sách 01” thì option 1 sẽ hiển thị là label1

<select>  
<option label="label01">Danh sách 01</option>  
<option label="label02">Danh sách 02</option>  
<option label="label03">Danh sách 03</option>  
<option label="label04">Danh sách 04</option>  
</select>

##### selected

giống với các thuộc tính như checked, multiple, thì selected phải được viết như ở vd dưới, nó chỉ định option mặc định của 1 select

<select>  
<option>Danh sách 01</option>  
<option selected ">Danh sách 02</option>  
<option>Danh sách 03</option>  
<option>Danh sách 04</option>  
</select>

### Thẻ <OUTPUT>

Phần tử <output> thể hiện cho kết quả của phép tính (như được thực hiện bởi script).

Ví dụ:  
Thực hiện một phép tính và hiển thị kết quả thông qua phần tử <output>:

<!DOCTYPE html>

<html>

<body>

<form action="action\_page.php" oninput="x.value=parseInt(a.value)+parseInt(b.value)">

0<input type="range" id="a" name="a" value="50">100 + <input type="number" id="b" name="b" value="50"> =

<output name="x" for="a b"></output>

<br><br>

<input type="submit">

</form>

</body>

</html>

#### for

Chỉ định mối quan hệ giữa kết quả của phép tính và các phần tử sử dụng trong phép tính

( mình chưa hiểu? vì khi mình xóa đi cũng không có ảnh hưởng gì

#### form

Chỉ định một hoặc nhiều form mà phần tử output thuộc về, giống [form](#_form) của input

#### name

Chỉ định tên cho phần tử đầu ra

#### Thiết lập css mặc định

output {  
display: inline;  
}

### Thẻ <IFRAME>

**Cú pháp Iframe**  
Một iframe được định nghĩa bằng thẻ iframe như sau:

**<iframe src=”**[**URL**](https://www.dammio.com/glossary/url)**”></iframe>**

Thuộc tính **src** đặc tả đường dẫn website của trang bên trong iframe.

#### **width, height**

dùng để định nghĩa chiều rộng và chiều cao của 1 Iframe. Các giá trị width, height có thể là giá trị pixel (hay dạng phần trăm như 80%).

### Thẻ <NAV>

Thẻ <nav> định nghĩa các liên kết điều hướng.

Lưu ý là KHÔNG phải tất cả các liên kết của tài liệu phải nằm bên trong phần tử <nav>. Phần tử <nav> chỉ dành cho khối liên kết chính có mục đích điều hướng.

Các trình duyệt, như màn hình đọc cho người khuyết tật, có thể sử dụng các phần tử này để xác định có nên đọc/xuất một nội dung nào đó hay không.

* Tag <nav> định nghĩa link danh mục (navigation).
* Tag <nav> thường sử dụng cho global menu, local link, topic path, pager link,...

<nav>

<a href="/html/">HTML</a> |

<a href="/css/">CSS</a> |

<a href="/js/">JavaScript</a> |

<a href="/jquery/">jQuery</a>

</nav>

Thiết lập CSS mặc định

Hầu hết các trình duyệt sẽ hiển thị thẻ <nav> với các giá trị mặc định sau:

*nav {*  
*display: block;*  
*}*

### Thẻ<SVG>

#### SVG là gì?

SVG (Scalable Vector Graphics), là một định dạng hình ảnh (tương tự như JPG, PNG,... mà chúng ta vẫn thường dùng) sử dụng cấu trúc XML để hiển thị hình ảnh dưới dạng vector. Vì là hình ảnh dạng vector nên chúng ta có thể hiển thị, co giãn (scale) thoải mái mà không làm giảm chất lượng hình ảnh. Một ưu điểm của SVG là tất cả mọi element và attribute của các element đó đều có thể animateVí dụ một file SVG để vẽ hình tròn: :

<svg width="100" height="100">

<circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />

</svg>

#### Dùng trong HTML thế nào?

##### Dùng trực tiếp:

Bạn có thể chèn trực tiếp nội dung file SVG vào trang HTML, cho vào 1 cái thẻ DIV chẳng hạn.

<div class="circle">

<svg width="100" height="100">

<circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />

</svg>

</div>

##### Dùng thông qua thẻ IMG

Hoặc bạn có thể dùng qua thẻ IMG như một hình ảnh bình thường.

<img src="circle.svg" />

#### Dùng trong CSS thế nào?

<div class="icon-facebook"></div>

#### Sử dụng Background Image

.icon-facebook {

background-image: url("facebook.svg");

...

}

#### Những thành phần SVG cơ bản

Bạn có thể đối mặt với rất nhiều sự phức tạp với SVG, nhưng nó không cần thiết cho những icon mà chúng ta sẽ tạo ra. Danh sách dưới đây khái quát những khối mà chúng ta sẽ cần.

* <svg> Bao bọc và định nghĩa toàn bộ hình. <svg> chính là đồ hoạ vector mà <html> là một trang web.
* <line> Tạo những đường thẳng đơn.
* <polyline> Tạo những đường đa đoạn.
* <rect> Tạo hình chữ nhật và hình vuông.
* <ellipse> Tạo hình tròn và hình oval.
* <polygon> Tạo hình đa giác, với ba hoặc nhiều cạnh.
* <path> Tạo bất kỳ hình nào mà bạn thích bằng cách định nghĩa những điểm và đường thẳng giữa chúng.
* <defs> Định nghĩa những tài nguyên có thể sử dụng lại. Không có gì nằm trong phần <defs> được hiển thị. <defs> là vector mà <head> là một trang web.
* <g> Gom nhiều hình dạng thành một nhóm. Đặt các nhóm trong phần <defs> để cho phép chúng được sử dụng lại.
* <symbol> Liên kết một nhóm, những với một số tính năng phụ. Thông thường được đặt trong phàn <defs>.
* <use> Lấy những tài nguyên được định nghĩa bên trong phần <defs> và làm cho chúng hiển thị trong SVG.

#### Ma thuật

<https://webdesign.tutsplus.com/vi/tutorials/how-to-hand-code-svg--cms-30368>

### Thẻ <SECTION>

Thẻ <section> định nghĩa khu vực trong một tài liệu, chẳng hạn như các chương, phần đầu, phần chân hoặc bất cứ khu vực nào khác của tài liệu.

Nó cũng chả có gì ngoài việc phân ra các vùng chức năng, nhưng nên có section với các class riêng để đổ dữ liệu vào

### <article>

article là phần tử để bao bọc nội dung độc lập, nó có thể là một bài post của diễn đàn, một bài viết của trang, một bài báo, một bình luận ... hoặc bất kỳ một nội dung độc lập nào.

Thẻ <article> trong HTML5 được dùng thay thế cho thẻ <div> của HTML4.

<article>

<h1>Tiêu đề bài viết</h1>

<p>Các nội dung bài viết</p>

</article>

### <aside>

Các <aside>thẻ định nghĩa một số nội dung ngoài nội dung nó được đặt trong.

Nội dung sang một bên nên liên quan gián tiếp đến nội dung xung quanh.

**Mẹo:** Các <aside>nội dung thường được đặt như một thanh bên trong một tài liệu.

### EVENT trong HTML

url tham khảo: <https://webvn.com/thu-vien/thu-vien-html/thu-vien-su-kien/>

### Các bắt sự kiện

1 <Tên-phần-tử Tên-sự-kiện="đoạn mã sẽ được thực thi khi sự kiện xảy ra">

2 viết riêng trong file script không cần viết bên trong thẻ ở file html

cách này cho phép bạn phân chia rõ hơn giữa html và javascript

window.[onload](#_onload) = function() {

     var banner\_obj = this.document.getElementById("banner");

     banner\_obj.addEventListener('click', clickhere);

}

function clickhere() {

     alert('ok banner');

}

Thực tế bạn có thể sử dụng cách thêm sự như loại 1 nhưng sẽ không chuyên nghiệp và hay bằng cách sử dụng hàm addEventListener(). Từ hàm này bạn có thể liên tưởng qua Events trong jQuery có cách xử lý y chang như vậy nên sau này nếu đi phỏng vấn có hỏi câu này thì bạn trả lời bản chất xử lý sự kiện trong jQuery cũng sử dụng hàm này nhé.

#### oninput

Thuộc tính oninput được kích hoạt khi người dùng nhập hoặc thay đổi thông tin trong thẻ. thay đổi ngay và luôn

Thuộc tính oninput kích hoạt khi giá trị của thẻ <input> và <textarea> thay đổi.

**Mẹo**: Sự kiện này tương tự như sự kiện [onchange](https://webvn.com/su-kien-onchange/). Điểm khác biệt là sự kiện oninput diễn ra ngay khi giá trị của thẻ thay đổi, trong khi thuộc tính onchange diễn ra khi bạn chuyển trỏ chuột sang một thành phần khác. Một sự khác biệt nữa đó là sự kiện onchange có thể sử dụng trong thẻ <keygen> và thẻ <select>.

Cú pháp

|  |
| --- |
| <element oninput = "script"> |

script có thể viết luôn như ví dụ của [thẻ <output>](#_Thẻ_<OUTPUT_>) hoặc có thể viết hàm trong file js:

VD:

<p>Nhập thông tin vào ô bên dưới để kích hoạt thuộc tính oninput.</p>

    <input type="text" id="myInput" oninput="myFunction()">

    <p id="demo"></p>

Nếu để oninput ở thẻ <form> thì tất cả các [thẻ <input>](#_Thẻ_<INPUT>) và <textarea> trong form đó đều có oninput: nghĩa là. bất kỳ thay đổi nào về giá trị của bất kỳ thẻ nào thì đều gọi thực thi “script”. Tương tự với các [<option>](#_<option>) trong [<select>](#_Thẻ_<SELECT>)

Nếu để oniput ở [thẻ <input>](#_Thẻ_<INPUT>) và <textarea> thì chỉ khi thay đổi giá trị thẻ đó thì mới thực thi “script”

#### onbeforeunload

Sự kiện onbeforeunload được kích hoạt khi người dùng đóng cửa sổ hoặc chuyển tới trang khác.

Sự kiện này cho phép bạn hiển thị một hộp thoại thông báo tới người dùng muốn ở lại hay chuyển tới trang khác.

Thông báo mặc định sẽ xuất hiện trong hộp thoại xác nhận, ở trên mỗi trình duyệt sẽ xuất hiện một thông báo khác nhau. Tuy nhiên, thông báo chuẩn sẽ là “Are you sure you want to leave this page?”. Bạn không thể bỏ thông báo này.

Tuy nhiên, mình chưa tìm ta cách để đưa thông báo riêng theo câu chữ của mình vào

thường đặt sự kiện này ngay thẻ body

**Lưu ý:** Trong Firefox, chỉ xuất hiện thông báo mặc định (không xuất hiện thông báo tùy chỉnh (nếu có)).

#### onload

Thuộc tính onload được kích hoạt khi đối tượng đã được tải xong.

onload thường được sử dụng nhất trong thẻ <body> để chạy một script khi trình duyệt của bạn đã tải đầy đủ tất cả thành phần của website (bao gồm ảnh, script, CSS, v.v…). Tuy nhiên, nó cũng có thể sử dụng trong những thẻ khác (xem phần “Chi tiết” bên dưới).

Thuộc tính onload có thể được sử dụng để kiểm tra phiên bảnvà kiểu trình duyệt của người dùng, sau đó tải các thành phần của website để tương thích với trình duyệt dựa trên thông tin thu được.

vd:

<!DOCTYPE html>

<html>

<body onload="checkCookies()">

<p id="demo"></p>

<script>

function checkCookies() {

var text = "";

if (navigator.cookieEnabled == true) {

text = "Cookies đã được bật.";

} else {

text = "Cookies chưa được bật.";

}

document.getElementById("demo").innerHTML = text;

}

</script>

</body>

</html>

#### onblur

Thuộc tính onblur được kích hoạt ngay khi người dùng chuyển con trỏ nháy ra ngoài thẻ.

Onblur thường được sử dụng nhiều nhất trong việc kiểm tra tính hợp lệ của imail.

Định dạng lại forn chữ, hoa, thường hay tự động thêm vào đầu(sđt: +84) thêm vào cuối(.com, .es)

vd:

Nhập tên của bạn: <input type="text" name="fname" id="fname" onblur="myFunction()">

    <p>Khi bạn di chuyển con trỏ nháy ra khỏi ô nhập tên, một hàm sẽ được kích hoạt để chuyển toàn bộ ký tự đã nhập thành viết hoa.</p>

    <script>

        function myFunction() {

            var x = document.getElementById("fname");

            x.value = x.value.toUpperCase();

        }

    </script>

#### onchange

Thuộc tính onchange được kích hoạt ngay khi giá trị của thẻ thay đổi.

**Mẹo:** Sự kiện này tương tự như sự kiện [oninput](#_oninput). Điểm khác biệt là sự kiện oninput diễn ra ngay khi giá trị của thẻ thay đổi, trong khi thuộc tính onchange diễn ra khi bạn chuyển trỏ chuột sang một thành phần khác. Một sự khác biệt nữa đó là sự kiện onchange có thể sử dụng trong thẻ <keygen> và thẻ <select>. Xem ví dụ bên dưới.

<p>Chọn tên xe từ danh sách dưới đây.</p>

<select id="mySelect" onchange="myFunction()">

<option value="Audi">Audi

<option value="BMW">BMW

<option value="Mercedes">Mercedes

<option value="Volvo">Volvo

</select>

<p>Khi bạn chuyển sang một xe mới, một hàm sẽ được kích hoạt để hiển thị giá trị ra ngoài.</p>

<p id="demo"></p>

<script>

function myFunction() {

var x = document.getElementById("mySelect").value;

document.getElementById("demo").innerHTML = "Bạn đã chọn xe: " + x;

}

</script>

! Cũng giống như form . Nêý đặt event ở select thì tất cả các option sẽ có tác dụng

### onfocus

trái ngược với [onblur](#_onblur)

Thuộc tính onfocus được kích hoạt ngay chi người dùng chuyển con trỏ vào thẻ.

Onfocus được sử dụng hầu hết trong thẻ <input>, <select> và <a>.

### onkeydown và onkeyup

onkeydown: Thuộc tính onkeydown kích hoạt khi người dùng nhấn một phím trên bàn phím.

onkeyup: Thuộc tính onkeyup được kích hoạt khi người dùng thả một phím (trên bàn phím).

VD:

<!DOCTYPE html>

<html>

<body>

<p>Khi bạn nhấn và giữ một phím trong thẻ bên dưới, màu nền sẽ chuyển sang màu đỏ. Thả phím ra, màu nền sẽ chuyển sang màu xanh lá cây.</p>

<input type="text" id="demo" onkeydown="keydownFunction()" onkeyup="keyupFunction()">

<script>

function keydownFunction() {

document.getElementById("demo").style.backgroundColor = "red";

}

function keyupFunction() {

document.getElementById("demo").style.backgroundColor = "green";

}

</script>

</body>

</html>

### onkeypress

Thuộc tính onkeypress được kích hoạt khi người dùng nhấn một phím (trên bàn phím) vào thẻ.

**Lưu ý:** Sự kiện onkeypress không kích hoạt với một số phím (v.d. ALT, CTRL, SHIFT, ESC) trong mọi trình duyệt. Để xác định người dùng đã nhấn một phím, bạn sử dụng sự kiện [onkeydown](#_onkeydown_và_onkeyup) vì sự kiện này làm việc với tất cả các phím.

### onreset

Thuộc tính onreset được kích hoạt khi thông tin trong form được xóa về mặc định (reset).

onreset sẽ được đặt trong thẻ form và trong form sẽ có thẻ [<input type=”reset”>](#_reset), khi người dùng ấn nút này. mọi dữ liệu trong form sẽ bị reset lại

! Bạn thường nên tránh bao gồm các nút đặt lại trong biểu mẫu của bạn. Chúng hiếm khi hữu ích và thay vào đó có nhiều khả năng làm người dùng thất vọng khi nhấp vào chúng do nhầm lẫn (thường là trong khi cố gắng nhấp vào [nút gửi](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/submit) ).

vd

<form>

<div>

<label for="example">Type in some sample text</label>

<input id="example" type="text">

</div>

<div>

<input type="reset" value="Reset the form"

accesskey="r">

</div>

</form>

### onresize

Thuộc tính onresize được kích hoạt khi cửa sổ trình duyệt thay đổi kích thước.

thẻ thường dùng: thẻ body

! event này ít dùng

### onsubmit

Thuộc tính onsubmit được kích hoạt khi thông tin từ form được gửi đi.

thuộc tính này nên để ở thẻ form

### querySelector

* Lấy ra phần tử đầu tiên của bộ chọn. Tương tự như [.first()](#_.first()) trong jquery

Vd: document.querySelector(".example");

-lấy ra phần thử đầu tiên có class là: example

## JavarScripts

### list

#### Tạo mới

vd: var cartList = [];

#### Thêm phần tử vào mảng

vd:

cartList.push({

Amount: $(item).val(),

Products: {

id\_product: $(item).data('id')

}

});

### object

#### Tạo mới

var <tên đối tượng> = {

<tên thuộc tính>: <giá trị của thuộc tính>,

<tên thuộc tính>: <giá trị của thuộc tính>,

....

}

vd:

var bill = {

Item: cartList,

Id\_customer: $('#txtCustomer').data('id'),

Subtotal : $('#SubTotal').text(),

Total: $('#Total').text(),

DiscountCode: $('#txtdiscountCode').val()

}

hoặc;

### for()

*for (var i = 0; i < 5; i++) {*

*callbacks.push( function() { return i; } );*

*}*

*for (let i = 0; i < 5; i++) {*

*callbacks.push( function() { return i; } );*

*}*

### foreach()

foreach(item =>{…});

foreach((item, index)=>{…});

foreach(function(item, index){…});

### function

Function <tên hàm> (tham số){..}

Nó không quan tâm tham số(thừa thiếu, sai kiểu dữ liệu không quan trọng). chỉ cần đúng tên hàm là nó sẽ gọi. Nếu nhiều hàm trùng tên, nó sẽ gọi hàm cuối cùng tìm thấy

(function(\*){}) (\*); khai báo xong gọi luôn;

(function(name){ consoler.Log(“hello” +name)(“hieu”); =>hellohieu

#### Prototype

Mọi function đều có prototype

* + Cho phép thêm thuộc tính hoặc phương thức vào đối tượng

Vd: function Person(first, last, age, eyecolor) {  
  this.firstName = first;  
  this.lastName = last;  
  this.age = age;  
  this.eyeColor = eyecolor;  
}  
  
Person.prototype.nationality = "English"; // thêm trường nationlity

Person.prototype.name = function() {// thêm hàm name()  
  return this.firstName + " " + this.lastName;  
};

Tổng quát hơn ta có \:

String.prototype.translate = function(){var content = this; …};

* + Sử dụng:var english = “tiếng việt”.stranstale(); // “tiếng việt” = this;

### Table

Khi render dữ liệu. vd có 1000 row thì nó sẽ raw 1000 row rồi mới hiển thị

Ta có thể dùng giải pháp render bằng thẻ dip raw đến đâu hiển thị đến đấy

### Biến

#### var

Là từ khóa định nghĩa 1 biến có pham vi truy cập trong 1 function. **Biến var** có tác dụng trong function mà nó được định nghĩa.

#### let

Là từ khóa định nghĩa 1 biến có phạm vi truy cập trong 1 block – khối code. **Biến let** có phạm vi trong dấu 1 cặp dấu {} bao quanh nó

Let có tác dụng hơn var trong trường hợp:

*for (let i = 0; i < 5; i++) {*

*callbacks.push( function() { return i; } );*

*}*

Khi đó i chỉ được hiểu trong khối lệnh của for

#### const

Là từ khóa định nghĩa 1 biến sẽ là hằng số. **Biến const** lưu trữ giá trị không thể thay đổi được trong suốt vòng đời của biến.

### Argument

Vd:  
Lấy giờ địa phương theo định dạng ISO

[https://www.techrepublic.com/article/convert-the-local-time-to-another-time-zone-with-this-javascript/#:~:text=getTimezoneOffset()%20\*%2060000%3B,the%20location%20is%20behind%20UTC.](https://www.techrepublic.com/article/convert-the-local-time-to-another-time-zone-with-this-javascript/#:~:text=getTimezoneOffset() * 60000%3B,the location is behind UTC.)

Công thức:

var tzoffset = (new Date()).getTimezoneOffset() \* 60000; //offset in milliseconds

var localISOTime = (new Date(Date.now() - tzoffset)).toISOString().slice(0, -1);

### Splice

\*không áp dụng cho chuỗi

Splice(Vị trí bắt đầu, số lượng bị xóa, item thay thế, item thay tthế 2,..)

* Splice(Vị trí bắt đầu) : xóa từ vị trí đó đến hết
* Splice(Vị trí bắt đầu, , số lượng bị xóa) : xóa và không thay thế
* Splice(Vị trí bắt đầu, số lượng bị xóa, item thay thế, item thay tthế 2,..): các item sẽ đc chèn vào vị trí xóa

Hàm sẽ xóa trực tiếp vào phần tử và trả về những phần tử bị xóa

Vd:

var subject = ["php", "css", "html", "js"];

subject.splice(1);

* php

subject.splice(1, 2);

* php,js

subject.splice(1, 1, "python", "c#", "ios");

* php,python,c#,ios,html,js

### slice

slice(start)

slice(start, end)

* trả về giá trị củavị trí bắt đầu đến vị trí kết thúc của chuỗi -1
* nếu không có end thì lấy đến hết

vd:

const animals = ['ant', 'bison', 'camel', 'duck', 'elephant'];

console.log(animals.slice(2));

// expected output: Array ["camel", "duck", "elephant"]

console.log(animals.slice(2, 4));

// expected output: Array ["camel", "duck"]

### toFixed ()

Định dạng số chữ số sau dấy phảy

let numObj = 12345.6789

numObj.toFixed() // Returns '12346': note rounding, no fractional part

numObj.toFixed(1) // Returns '12345.7': note rounding

numObj.toFixed(6) // Returns '12345.678900': note added zeros

(1.23e+20).toFixed(2) // Returns '123000000000000000000.00'

(1.23e-10).toFixed(2) // Returns '0.00'

2.34.toFixed(1) // Returns '2.3'

2.35.toFixed(1) // Returns '2.4'. Note it rounds up

2.55.toFixed(1) // Returns '2.5'. Note it rounds down - see warning above

-2.34.toFixed(1) // Returns -2.3 (due to operator precedence, negative number literals don't return a string...)

(-2.34).toFixed(1) // Returns '-2.3'

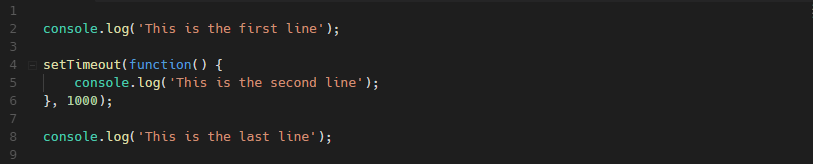
x = findMax(1, 123, 500, 115, 44, 88);  
  
function findMax() {  
  var i;  
  var max = -Infinity;  
  for (i = 0; i < arguments.length; i++) {  
    if (arguments[i] > max) {  
      max = arguments[i];  
    }  
  }  
  return max;  
}

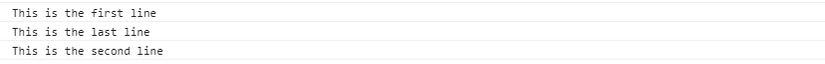
Nếu một hàm được gọi với **quá nhiều đối số** (nhiều hơn được khai báo), các đối số này có thể được truy cập bằng cách sử dụng **đối tượng đối số** .

### Bất đồng bộ

### số khái niệm

1 ví dụ về setTimeout quen thuộc

Kết quả chạy đoạn code:



#### CALL STACK

- là một dạng cấu trúc dữ liệu ghi lại vị trí các lệnh đang được thực hiện trong chương trình. Khi lệnh bắt đầu được thực hiện sẽ được đưa vào đỉnh của stack và sau khi thực hiện xong sẽ được lấy ra khỏi ngăn xếp.

#### **WEB APIs**

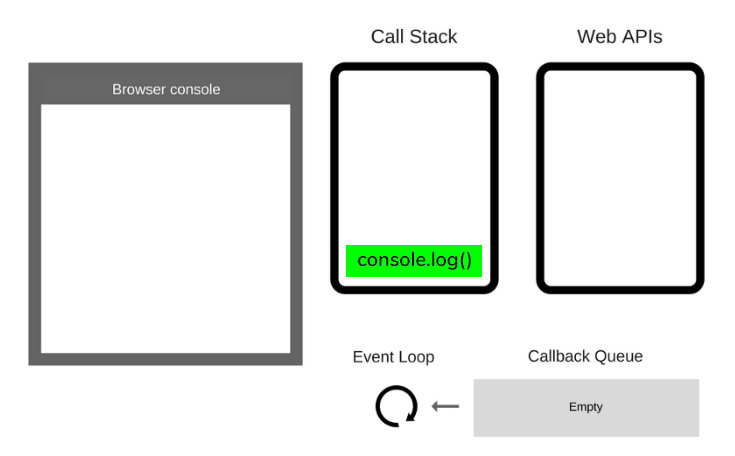
- vể bản chất đây chính là các thread mà ta không thể truy cập trực tiếp mà chỉ có thể gọi được đến nó. Các thread này do trình duyệt cung cấp.

#### **CALLBACK QUEUE**

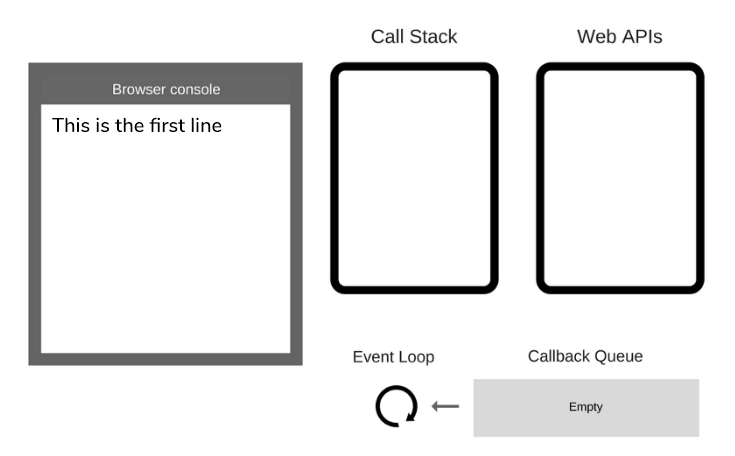
- là một dạng cấu trúc dữ liệu với nguyên tắc First-In-First-Out (vào trước ra trước).

#### EVENT LOOP

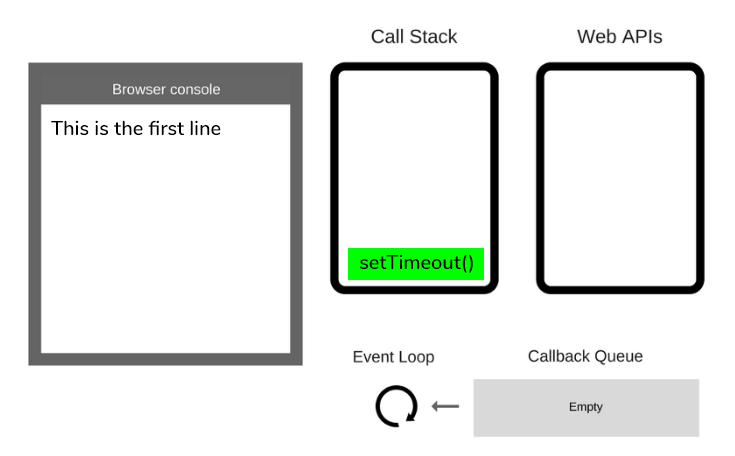
- có nhiệm vụ giám sát tình trạng của CALL STACK và CALLBACK QUEUE. Để hiểu được quá trình thực hiện của cơ chế bất đồng bộ ta sẽ đưa ví dụ thứ hai vào và thực hiện trong mô hình trên.



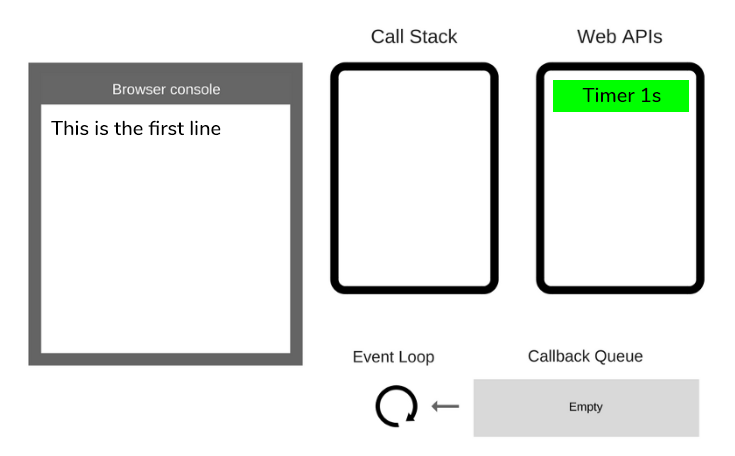
* Lệnh này lập tức trả về dòng chữ This is the first line đồng nghĩa với việc nó đã chạy xong và được đẩy ra khỏi **CALL STACK**.

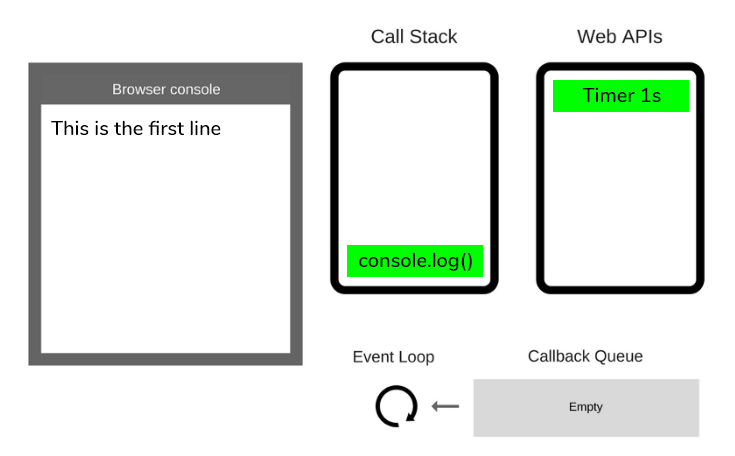
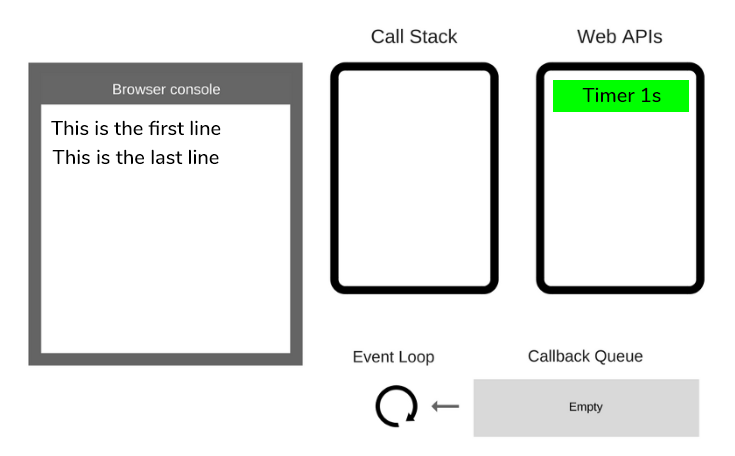


* Tiếp đến hàm setTimeout(function() { console.log('This is the second line'); }, 1000); được đưa vào trong **CALL STACK** để thực hiện.

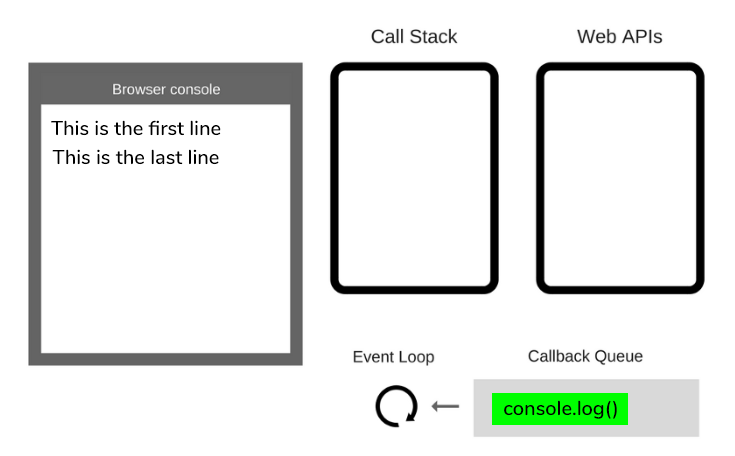


* Tuy nhiên hàm nay không trả về kết qua ngay mà phải đợi 1 giây. Hàm setTimeout() ở đây chính là một API mà **WEB APIs** cung cấp. Lập tức đoạn code này được chuyển vào trong **WEB APIs** và trình duyệt sẽ tạo ra một bộ hẹn giờ tương ứng với thời gian trên là 1 giây trước khi trả về kết quả

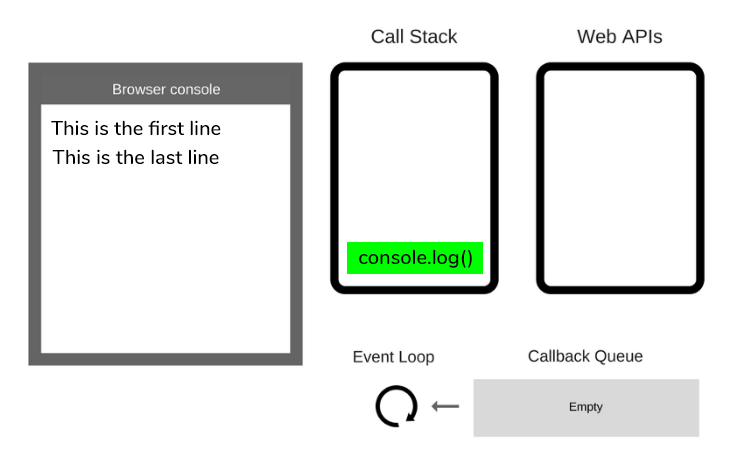


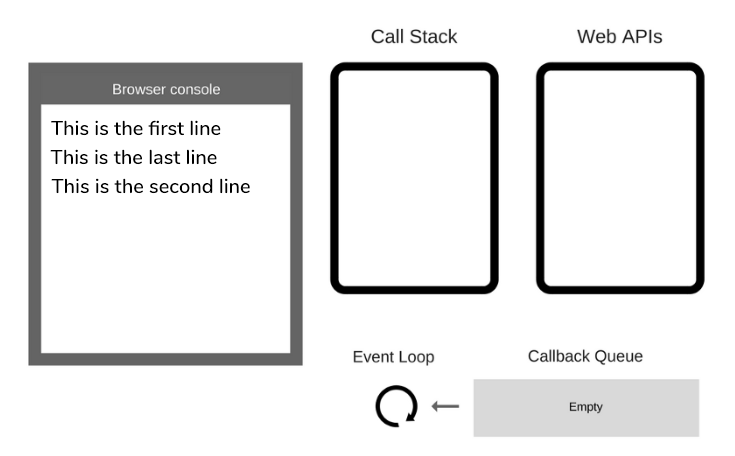
* Khi đoạn code thứ 2 được chuyển sang **WEB APIs** thì lập tức đoạn code cuối cùng console.log('This is the last line ') đã được đưa vào **CALL STACK** để thực
* hiện và trả về kết quả là dòng chữ This is the last line. Sau đó đoạn code này cũng được đẩy ra khỏi **CALL STACK**

* Sau khi bộ giờ trong **WEB APIs** chạy xong thì kết quả trả về lúc này không được in ngay ra màn hình mà nó được đẩy vào **CALLBACK QUEUE**



* **EVENT LOOP** với chức năng liên tục giám sát xem **CALL STACK** đã trống chưa và **CALLBACK QUEUE** có gì không. Lúc này **CALLBACK QUEUE** đang chư kết quả mà **WEB APIs** trả về nên và **CALL STACK** lúc này cũng đã trống do toàn bộ code trong chương trình đã được thực hiện nên nó sẽ đẩy kết quả trong **CALLBACK QUEUE** vào lại **CALL STACK** và đoạn code console.log(''This is the second line) được thực hiện và trả kết quả ra màn hình.



Đó là toàn bộ quá trình diễn ra trong cơ chế chạy bất đồng bộ của javascript, tuy nhiên ta cần chú ý một số điểm sau:

* **EVENT LOOP** chỉ tiến hành đẩy hàm trả về từ **CALLBACK QUEUE** sang **CALL STACK** khi **CALL STACK** đã thực hiện xong hết các đoạn code khác của chương trình
* Thứ tự các kết quả được trả về từ **WEB APIs** sang **CALLBACK QUEUE** sẽ không theo thứ tự đưa vào mà sẽ tùy thuộc hàm nào chạy xong trước sẽ được đẩy vào **CALLBACK QUEUE** trước đồng nghĩa với việc sẽ được chuyển qua **CALL STACK** trước.

#### Call Back

Call Back là một hàm được truyền vào một hàm khác với tư cách như một tham số của hàm đó. Ví dụ như

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | function nauGa(callback)       nauNuocSoi();      vatLongGa();      callback();  }  function luocGa(){      //  }    function nuongGa(){  //  }  nauGa(luocGa);  nauGa(nuongGa); |

Ở đoạn mã trên, chúng ta thấy rằng, hàm luocGa và nuongGa được dùng như tham số trong hàm nauGa Với Javascript, một ngôn ngữ hướng sự kiện, call back được sử dụng rất nhiều khi xử lý các sự kiện, ví dụ như

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | $('#button').click(function(){  alert("hê nô");  }) |

Chúng ta có thể áp dụng call back để đồng bộ hóa các đoạn mã không đồng bộ. Ví dụ như ở đoạn mã trên. Nấu nước sôi cần một khoảng thời gian chờ nước sôi, chúng ta không phải làm gì. Ta có thể biểu diễn thời gian chờ này bằng hàm setTimeout() trong javascript.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | function soCheGa(){     nauNuocSoi();     vatLongGa();  }  function nauNuocSoi(){     setTimeout(function(){       console.log("nau nuoc soi")     }, 1000)  }  function vatLongGa(){  console.log("vat long ga");  }  soCheGa(); |

Và nếu như theo đúng cách chạy của Javascript thì hành động vặt lông gà sẽ được thực hiện trước hành động nấu nước sôi. (à, quên mất, nấu nước sôi là để nhúng gà vào vặt lông chứ không phải để luộc gà đâu nhé ) .Mà nếu chúng ta vặt lông gà luôn mà không cần nhúng nước nóng thì tội cho bác gà quá. Vì thế để cho gà có thể ra đi thanh thản chúng ta cần đồng bộ hóa lại quy trình bằng callback như sau

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | function soCheGa(callback){        nauNuocSoi(vatLongGa);      }       function nauNuocSoi(callback){        setTimeout(function(){          console.log("nau nuoc soi");          callback();        }, 1000)      } |

Tuy nhiên, Callback cũng có nhược điểm. Đó là khi chúng ta muốn nhiều hành động bất đồng bộ thực hiện theo đúng thứ tự liên tiếp nhau, chúng ta phải gọi nhiều hàm callback lồng vào nhau nhiều lần, gây ra đoạn code rất khó kiểm soát và không tối ưu. Đây gọi là tình trạng Callback Hell. Ví dụ như muốn in các số từ 1 đến 10, mà mỗi hành động in đều là một hàm bất đồng bộ

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | function printNumber(number, callback){    setTimeout(      () =&gt; {        console.log(number)        callback()      },      Math.floor(Math.random() \* 100) + 1    )  }    function printAll(){    printNumber(1, function(){      printNumber(2, function(){        printNumber(3, function(){          printNumber(4, function(){            printNumber(5, function(){              printNumber(6, function(){                printNumber(7, function(){                  printNumber(8, function(){                    printNumber(9, function(){                      printNumber(10, function(){                        })                    })                  })                })              })            })          })        })      })    })  } |

#### Promise

Promise là một đối tượng bao hàm một hàm chứa các đoạn code không đồng bộ. Hàm này chứa 2 tham số là hai hàm callback để giải quyết sau khi mã đồng bộ thực hiện thành công hay thất bại. Promise cung cấp cho ta hai phương thức xử lý sau khi đoạn mã bất đồng bộ thực hiện thành công hoặc thất bại. Hàm then() dùng để xử lý sau khi mã bất đồng bộ được thực hiện thành công và hàm catch() dùng để xử lý sau khi mã bất đồng bộ thực hiện thất bại

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | function printNumber(number){      return new Promise((resolve, reject) =&gt; {        setTimeout(          () =&gt; {          if(number&lt;0){           reject();           }else{               console.log(number);               resolve();           }          },        1000        )      })  }  printNumber(1)  .then(()=&gt;printNumber(2))  .reject(()=&gt;console.log("number &lt; 0")) |

Phương thức then có thể thực thi một hàm, một Promise hay một đối tượng. Nếu chúng ta dùng then để trả về một Promise thì ta có thể tận dụng để xử lý tình trạng Callback Hell

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | printNumber(1)  .then(()=&gt;printNumber(2))  .then(()=&gt;printNumber(3))  .then(()=&gt;printNumber(4))  .then(()=&gt;printNumber(5))  .then(()=&gt;printNumber(6))  .then(()=&gt;printNumber(7))  .then(()=&gt;printNumber(8))  .reject(()=&gt;console.log("number &lt; 0")) |

Tuy nhiên, dù Promise đã giải quyết được vấn đề Callback Hell, nhưng chúng ta có thể thấy, đoạn mã vẫn chưa thực sự rõ ràng và dễ hiểu. Trong phương thức chúng ta vẫn phải truyền vào một hàm, mà hàm đó trả về một hàm khác có giá trị trả về là 1 Promise. Chúng ta tạm gọi đây là tình trạng Promise Hell.

#### Asyn/Await

Async / Await là một tính năng ngôn ngữ là một phần của tiêu chuẩn ES8. Từ khóa Async để khai báo rằng hàm này sẽ xử lý các hàm bất đồng bộ, nó sẽ chờ kết quả của các hàm bất đồng bộ được trả về sau đó mới thực hiện tiếp. Hàm bất đồng bộ đó phải trả về một Promise và được khai báo với từ khóa Await

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | function printNumber(number){    return new Promise((resolve, reject) =&gt; {      setTimeout(        () =&gt; {         console.log(number)         resolve()        },       Math.floor(Math.random() \* 100) + 1      )    })  }    async function printAll(){    await printNumber(1);    await printNumber(2);    await printNumber(4);    await printNumber(5);    await printNumber(6);  }  printAll(); |

Đến đây thì chúng ta đã thấy Asyn/Await đã giải quyết triệt để được tình trạng Callback Hell cũng như Promise Hell.

-Với 1 hàm bất kỳ vd: async function say(){ return "hi"; }

Khi async đối tượng trả về luôn được chuyển thành promise (kể cả không return



Nếu muốn lấy kết quả của hàm async ta cần thêm await

#### Promise.all

## jQuery

### Selection

<#id hoặc .class > <#id hoặc .class > …

.<class>[<tên thuộc tính =”<nội dung thuộc tính>”]

#<id>[<tên thuộc tính =”<nội dung thuộc tính>”]

### Đọc và đổi nội dung html

đổi nội dung HTML để tạo 1 hoặc nhiều thẻ mới tạo ra giao diện mới bê trong selector( selector vẫn còn

đổi nội dung text chỉ để thay đổi text bên trong selector

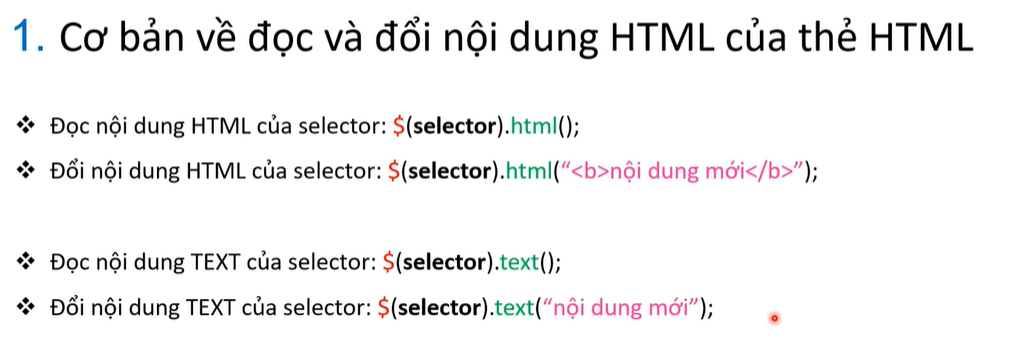
VD Nội dung mới

let index = 5;

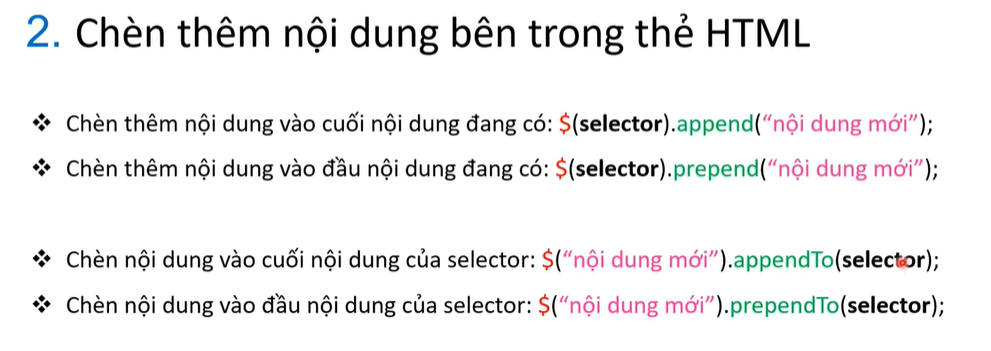
                for (; index <= 20; index += 1) {

                    $(".father").after("<Div>số:  +" + index + "</div>");

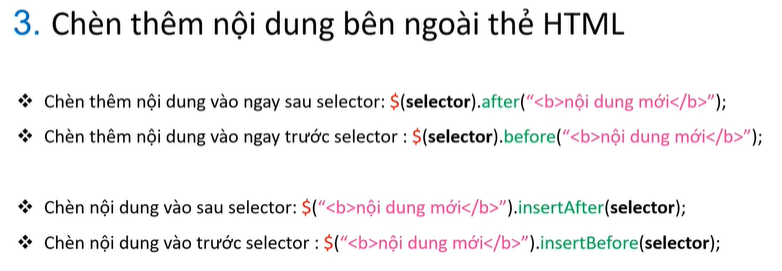
                }

nội dung mới sẽ là 1 string nhưng nếu muốn chuyền 1 tham số biến đổi như index ta có thể là như trên

### Chèn nội dung bên trong thẻ HTML



### Chèn nội dung bên ngoài thẻ HTML



### Xóa bỏ hoặc thay thế, làm rỗng nội dung và thẻ HTML

* làm rỗng nội dung selector: $(selector).empty();

toàn bộ nội dung bao gồm text và thẻ bên trong selector sẽ bị xóa, nhưng selector sẽ không bị xóa

* xóa bỏ selector khỏi tài liệu: $(selector).remove(); Hoặc $(selector).detach();

toàn bộ nội dung bên trong và thẻ sẽ bọ xóa

* Loại bỏ các phần tử cha của selector:$(selector).unwrap();

chỉ 1 có phần tử cha trực tiếpnó bị xóa. thẻ ông, thẻ cụ thì vẫn còn nguyên

* thay thế selector bằng nội dung mới: $(selector).replaceWith(“<b>nội dung mới</b>”);

hoặc: $(“<b>nội dung mới</b>”).replaceAll(selector);

nội dung mới ở đây phải là 1 hoặc nhiều thẻ mới thế vào vị trí selector bị xóa

### Get, Set thuộc tính thẻ

$(selector).attr(“thuộc tính”);

$(selector).attr(“thuộc tính”, “giá trị mới”);

$(selector).attr({ *attribute*:value, *attribute*:value, *attribute*:value, .... });

Vd: $("img").attr({ width:"500", height:"100" });

$(selector).prop(“thuộc tính”);

$(selector). prop(“thuộc tính”, “giá trị mới”);

**Hàm prop() trong jquery** cũng dùng để lấy các thuộc tính của thẻ, tuy nhiên cách trả về kết quả lại khác một xíu. Đối với những thuộc tính có dạng true/false thì thay vì trả về giá trị của thuộc tính như hàm attr() thì nó sẽ trả về true hoặc false.

vd

theImage.src = $("#pr\_draggable img").attr("src");

### Thay đổi CSS

#### Thay thế class

Cách đơn giản nhất để chèn CSS class vào phần tử HTML đó là sử dụng method addClass() của jQuery. Có hai cách sử dụng method .addClass() của jQuery:

##### .addClass(className)

Với cách này chúng ta sẽ sử dụng đối số truyền vào là một chuỗi chứa một hoặc nhiều CSS class sẽ được thêm vào phần tử:

$('#my-el').addClass("new-class-1 new-class-2");

Kết thúc câu lệnh trên phần tử #my-el sẽ được gắn thêm hai CSS class là new-class-1 và new-class-2.

##### .addClass(function)

Với cách này chúng ta sẽ sử dụng đối số truyền vào là một hàm thay vì một chuỗi. Ví dụ như sau:

$("ul li").addClass(**function** (index) {

**return** "item-" + index;

});

Với cách sử dụng method .addClass() như trên thì jQuery sẽ lặp qua từng phần tử li có trong danh sách và index chính là thứ tự ứng với phần tử này trong danh sách. Kết quả trả về quả hàm sẽ xác định CSS class được thêm vào phần tử.

Kết thúc câu lệnh trên mỗi phần tử li sẽ được thêm lần lượt các CSS class bắt đầu từ phần tử đầu tiên (nếu có) sẽ là item-0, phần tử thứ hai (nếu có) là item-1, phần tử thứ 3 (nếu có) item-2...

##### .removeClass(*classname*);

 Phương thức **removeClass()** dùng để xóa *"tên lớp"* khỏi giá trị thuộc tính class của phần tử.

Vd: $("p").removeClass("background");

#### Thay thế thuộc tính CSS

##### [**.css('**](https://hocwebchuan.com/reference/jquery/jquery_css.php#anchor01)*thuộc tính***','***giá trị***')**

vd: $('p.test').css('color','red');

##### [**.css({'**](https://hocwebchuan.com/reference/jquery/jquery_css.php#anchor02)*thuộc tính 01***':'***giá trị 01***','***thuộc tính 02***':'***giá trị 02***',...})**

vd: $('p.test').css({'color':'blue','border':'1px dashed red'});

##### [**.css('**](https://hocwebchuan.com/reference/jquery/jquery_css.php#anchor03)*thuộc tính***',function(index,***giá trị***){})**

vd: $('p.test').css('width', function(index,value){

return parseFloat( value ) \* 2;

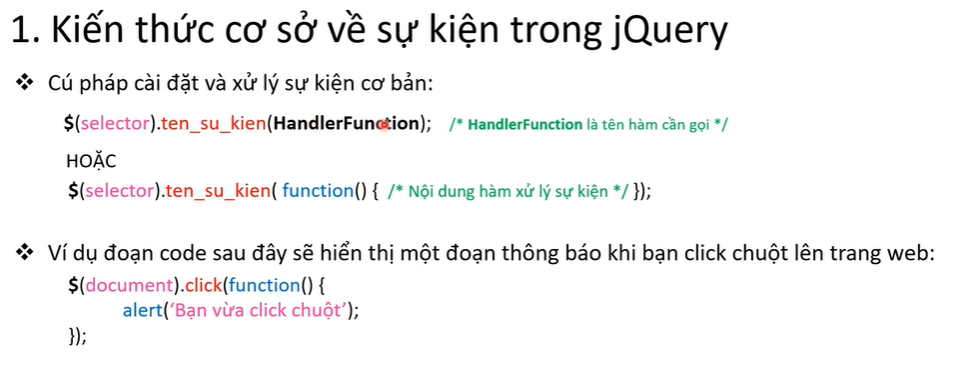
});

#### Lấy giá trị style của thành phần:

##### [**.css('**](https://hocwebchuan.com/reference/jquery/jquery_css.php#anchor04)*thuộc tính***'**)

vd: var ulHeight = $('ul').css('height');

### Điều khiển sự kiện cơ bản



Hoặc [**.on('**](https://hocwebchuan.com/reference/jquery/jquery_on.php#anchor01)***Sự kiện*', function(){...})**

**// sự kiện: tên sự kiện trong jquenry(không có on)**

#### .trigger (“event name”)

* Kích hoạt sự kiện

Vd: khi nhấn botton sẽ kích hoạt sự kiện chọn file

<script>

$(document).ready(function(){

$("input").select(function(){

$("input").after(" Text marked!");

});

$("button").click(function(){

$("input").trigger("click");

});

});

</script>

</head>

<body>

<input type="file" value="Hello World"><br><br>

<button>Trigger the select event for the input field</button>

</body>

#### .error(HandlerFunction);

sự kiện: đã có lỗi xảy ra với selector

vd: $(window).resive(

#### .resize(HandlerFunction);

sự kiện: selector đã bị thay đổi kích thước

vd: $(window).resive(ResiveHappend);

cửa sổ trình duyệt bị thay đổi thì sẽ gọi đến hàm ResiveHappend()

#### .scroll(HandlerFunction);

sự kiện:đá có thao tác cuộn trên selector

áp dụng với những selector có thanh cuộn

#### .ready(HandlerFunction);

sự kiện : selector đá tải xong nội dung và sẵn sàng

Thường dùng với toàn bộ trang HTML: nếu không muốn để link scrip xuống dưới cuối cùng trong HTML

vd: $(document).ready(function(){ //code});

#### .load(HandlerFunction);

sự kiện: đã tải xong nội dung

#### .unload(HandlerFunction);

sự kiện trang web bị tắt hoặc chuyển hướng đi web khác

#### .holdReady(true/false);

đây là 1 hàm jquery giúp chúng ta có thể tạm giữ cho sự kiện ready khong xảy ra(khi cần thiết). Giá trị TRUE sẹ thực hiện việc chờ(tạm giữ) và giá trị (false) sẽ xóa bỏ sự tạm giữ

#### .click(HandlerFunction);

sự kiện: click chuột

#### .contextmenu(HandlerFunction);

sự kiện: click chuột phải

#### .bdclick(HandlerFunction);

sự kiện Double click

#### .hover(HandlerFunction);

sự kiện rê chuột lên đối tương

#### .mousedown(HandlerFunction);

sự kiện: click nhưng chưa nhấc chuột lên

#### .mouseup(HandlerFunction);

sự kiện: click đã nhấc chuột lên

#### .mouseover(HandlerFunction);

sự kiện: rê chuột lên đối tượng

#### .mouseout(HandlerFunction);

sự kiện: chột rời khỏi đối tượng đối tượng

#### .mouseenter(HandlerFunction);

sự kiện: rê chuột vào đối tượng

#### .mouseleave(HandlerFunction);

sự kiện: chột rời khỏi đối tượng đối tượng

#### .mousemove(HandlerFunction);

sự kiện: chuột di chuyển trên đối tượng

#### .toggetle(HandlerFunction1, HandlerFunction2, …);

Trường hợp đặc biệt: toggle giúp bạn điều khiển sự kiện click chuột theo số lần và lặp lại tuần tự

* VD: có 2 handler: click 1 lần gọi handler1, clịck 2 lần gọi hanler 2, click 3 lần gọi handler3, click lần 4 trở về handler1

#### .keypress(HandlerFunction);

sự kiện: gõ phím lên seletor

#### .key(HandlerFunction);

sự kiện: thao tác đè phím xuống

vd tính năng: ấn giữ phím a để xem thông tin

#### .keyup(HandlerFunction);

sự kiện: thao tác nhả phím sau khi đè xuống

#### .focus(HandlerFunction);

sự kiện: đang target vào selector( sẵn sàng) tương tự [onfocus](#_onfocus)

cụ thể: khi bạn click vào 1 thẻ input có type = [“text”](#_text) con trỏ chuyển thành dấu nháy văn bản. vừa xảy ra sự kiện click vừa xảy ra sự kiện focus

#### .blur(HandlerFunction);

sự kiện: hủy taget selector(sau khi forcus) tương tự [onblur](#_onblur)

Cụ thể: khi bạn nhập xong dữ liệu bạn click ra 1 ngoài sự kiện blur sẽ được gọi>

>> có thể dùng để kiểm tra điều kiện của dữ liệu, thông báo hợp lệ

#### .change(HandlerFunction);

sự kiện: thay đổi giá trị của selector( chỉ sau khi blur) tương tự [onchange](#_onchange)

#### .select(HandlerFunction);

sự kiện: tô đen một đoạn văn bản trong selector

#### .submit(HandlerFunction);

sự kiện: bắt đầu việc submit form

#### .blind- đính kèm sự kiện

được sử dụng để đính kèm xử lý thêm sự kiện cho thành phần. tương tự [.on](#_Điều_khiển_sự)

các sự kiện trong đây có thể vừa dùng “JavascriptEventName”(có on+<tên sự kiện>) hoặc sự kiện trong jquenry(không on)

#### [**.bind(**](https://hocwebchuan.com/reference/jquery/jquery_bind.php#anchor01)JavascriptEventName**,function(){})**

#### [**.bind('**](https://hocwebchuan.com/reference/jquery/jquery_bind.php#anchor03)**JavascriptEventName(1) JavascriptEventName(2)',function(){})**

vd: $(function(){

$('div.test').bind('onclick ondblclick',function(){

$('span').text('Đã click');

});

});

#### [**.bind({**](https://hocwebchuan.com/reference/jquery/jquery_bind.php#anchor04)**Sựkiện1:function(){}, Sựkiện2:function(){}})**

vd: $(function(){

$('div.test').bind({

click : function(){

$('span').text('Click');

},dblclick : function(){

$('span').text('Double click');

}

});

});

#### .offset()

Lấy tọa độ hiện tại của thành phần đầu tiên, hoặc thiết lập các tọa độ của từng thành phần trong tập hợp các thành phần phù hợp liên quan đến văn bản.

* tọa độ được tính từ góc màn hình, không phụ thuộc vào position

có 2 thuộc tính của offset

.offset().top;

.offset().left;

set tọa độ:

#### .offset({ top: 10, left: 20 });

set tọa độ: với top =10; left =20;

tọa độ này tính từ góc màn hình chứ không giống {left; right} của position

#### .offset(function(index,tọa độ){...})

set tọa độ theo ý muốn

* **function phải có return**

vd: mỗi lần click thì tăng left và top thêm 10 px

$(this).offset(function(index, coordinates){

if (index == 0) {//Chỉ tác động lên thành phần đầu tiên

coordinates.top += 10;

coordinates.left += 10;

}

return $(this).offset(coordinates);

});

#### .val()

-lấy dữ liệu val của thẻ. luôn trả về dạng string.

#### .data-...

Không nên đặt tên là in hoa vì chungd sẽ trở về in thường vd:

<button type="button" class="btn btn-default edit" data-IdGVDT="${item.IdGVDT}”>

thực chất thuộc tính data của thẻ trên là : data-idgvdt

### delegate()

Phương thức delegate() không được dùng nữa trong phiên bản 3.0. Sử dụng phương thức [on ()](https://www.w3schools.com/jquery/event_on.asp) để thay thế.

Vd:

Khi một phần tử <p> bên trong một phần tử <div> được nhấp, hãy thay đổi màu nền của tất cả các phần tử <p>:

$("div").delegate("p", "click", function(){  
    $("p").css("background-color", "pink");  
});

Mặc dù vậy nó đặc biệt hữu ích khi bắt sự kiện từ 1 thẻ được render từ lệnh [.append()](#_.append()) hoặc những lệnh thêm thẻ tương tự. vì trong trường hợp này những cách gọi khác không thể sử dụng được

→ Như cụ thể: thẻ p đc tạo ra như sau: $("div").append(<p>Hieu</P>)

### .prev()

.prev(): xác định thành phần cùng cấp ngay trước thành phần sử dụng .prev().

VD:

!DOCTYPE HTML>

<html>

<head>

<meta charset="utf-8">

<title>Tiêu đề</title>

<script src="https://code.jquery.com/jquery-latest.js"></script>

<script>

$(function(){

$('li.second').prev().css('background-color','yellow');

});

</script>

</head>

<body>

<ul>

<li>li thứ nhất</li>

<li class="second">li thứ hai</li>

<li>li thứ ba</li>

<li>li thứ tư</li>

<li>li thứ năm</li>

</ul>

</body>

</html>

Hiển thị trình duyệt:

* li thứ nhất
* li thứ hai
* li thứ ba
* li thứ tư
* li thứ năm

### .next()

xác định thành phần cùng cấp ngay kế tiếp thành phần sử dụng . ngược với prev

.shoping\_\_cart\_\_tota

### .before()

Chèn nội dung, di chuyển thành phần vào ngay trước mỗi thành phần đã có.

vd: $('div').before('<p>nội dung thêm vào</p>');

### .after()

#### [**.after(**](https://hocwebchuan.com/reference/jquery/jquery_after.php#anchor01)**nội dung)**

Chèn nội dung, di chuyển thành phần vào ngay sau mỗi thành phần đã có.

$('div').after('<p>nội dung thêm vào</p>');

### .find()

.find(): Giúp tìm thành phần trong thành phần cha.

#### [**.find(**](https://hocwebchuan.com/reference/jquery/jquery_find.php#anchor01)**bộ chọn)**

$('li').find('.test');

* Đã được thêm vào từ phiên bản 1.6

#### [**.find(**](https://hocwebchuan.com/reference/jquery/jquery_find.php#anchor02)**jquery object)**

$('li').find($('div.test')); tìm thành phần con là thẻ div có class=“test”

### .append()

thêm nội dung vào vị trí cuối cùng bên trong phần nội dung của phần tử được chọn.

<body>

<button type="button">Click here</button>

<ul>

<li>Tài liệu học HTML</li>

<li>Tài liệu học CSS</li>

~~<li>Tài liệu học CSS</li>~~ // nội dung sẽ được thêm

</ul>

<script>

$(document).ready(function(){

$("button").click(function(){

$("ul").append("<li>Tài liệu học JavaScript</li>");

});

});

</script>

</body>

### .prepend()

thêm nội dung vào vị trí đầu tiên bên trong phần nội dung của phần tử được chọn. tương tự [.append()](#_.append())

### .parent()

Lấy ra selecter của cha nó

### .hasClass()

hasClass(‘tên class’): trả về kiểu bool. Kiểm tra xem selector có class đó không

### [.first()](https://api.jquery.com/first/#first)

Chọn ra phần tử đầu tiên

! var countItem = $('.countItem').text(); trường hợp nếu không có first(). Có 2 thẻ mỗi thẻ có text là 5 => countItem= 55; 3 thẻ thì sẽ là 555

### [.last()](https://api.jquery.com/first/#first)

Chọn ra phần tử cuối cùng

### .map()

* Cơ bản giống each() dùng để lặp qua các phần tử trong DOM
* Tốt hơn về hiệu suất và code ngắn hơn.

VD:

$('p').append($('input').map(function(){

return this.id;

}).get().join(", "));

<body>

<p>Value: </p>

<input type="text" id="text01" />

<input type="text" id="text02" />

<input type="text" id="text03" />

<input type="text" id="text04" />

</body>

Hiển thị:

Value: text01, text02, text03, text04

Map thường dùng kết hợp với get() với join(“, “)-các phần tử cách nhau bởi dấu ‘,”

### .preventDefault()

* Ngăn nút submit gửi biểu mẫu
* Ngăn một liên kết theo dõi URL

### .stopPropagation ()

dừng việc tạo sự kiện cho các phần tử mẹ, ngăn không cho bất kỳ trình xử lý sự kiện mẹ nào được thực thi.

Vd:

<script>

$(document).ready(function(){

$("span").click(function(event){

event.stopPropagation();

alert("The span element was clicked.");

});

$("p").click(function(event){

alert("The p element was clicked.");

});

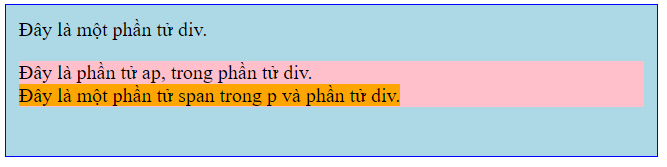
$("div").click(function(){

alert("The div element was clicked.");

});

});

</script>



**Lưu ý:** Nhấp vào từng phần tử ở trên. Khi nhấp vào phần tử **div** , nó sẽ cảnh báo rằng phần tử div đã được nhấp. Khi nhấp vào phần tử **p** , nó sẽ trả về cả phần tử p và phần tử div, vì phần tử p nằm bên trong phần tử div. Nhưng khi nhấp vào phần tử **span** , nó sẽ chỉ trả về chính nó chứ không phải p và phần tử div (mặc dù nó nằm bên trong các phần tử này). Event.stopPropagation () ngăn sự kiện nhấp chuột chuyển sang các phần tử mẹ.

### Hàm trong hàm

**1 Vd với preview image**

1 $('.addPhoto').click(function () {

2 var name = $('.modal img').attr('name');

3 var selectedImg = '.content img[name="' + name + '"]';

4 var parent = $(selectedImg).parent().parent();

5 var newFile = parent.find('input:file');

6 newFile.trigger('click');

7 newFile.change(function () {

8 if (this.files.length > 0) {

9 parent.find('input:checkbox').attr('checked', 'checked');

10 var reader = new FileReader();

11 reader.addEventListener('load', function () {

12 $(selectedImg).attr('src', this.result);

13 });

14 reader.readAsDataURL(this.files[0])

15 }

16 });

17 });

khác biệt nằm ở dòng 9; khi thao tác trực tiếp với thẻ input file *($('.addPhoto').click không được gọi* ) dòng 9 lúc này sẽ không được gọi vì nó không hiểu parent là ai

//

Luôn đúng: newFile.change có thể bắt sự kiện sau khi thao tác trực tiếp với thẻ input file hoặc trong khi trong $('.addPhoto').click(function () đang chạy

khác biệt khi dùng hàm trong hàm là sự kiện con sẽ hoạt động sau khi chạy hết chương trình(ngoại trừ các hàm sự kiện đồng cấp phía sau sẽ chạy sau)

như vd trên: với dòng 6: input file chỉ update khi hoàn thành *($('.addPhoto').click* Nếu không có hàm ở dòng 7. Thì …files.length luôn bằng 0 vì input file chưa update.

//

- việc dùng hàm trong hàm cho phép sử dụng lại các biến của hàm cha với điền kiện hàm con được gọi trong khi duyệt hàm cha. Hàm con sẽ được đánh dấu để duyệt sau khi chạy hết hàm cha.( nhưng vẫn giữ lại biến) . như vd trên: dòng 14 sẽ được duyệt trước hàm reader.addEventListener()

Lưu ý. Hàm con thường đc gọi với những sự kiện bị động như, load, chage, ..

### Animation

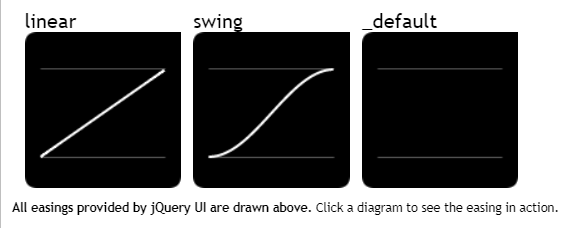
#### show – hiệu ứng hiển thị các tham số bên trong mỗi hàm

mỗi tham số đều không bắt buộc

- độ bền là thời gian thực hiện animation

- function sẽ được gọi khi animation hoàn tất

- easing có thể sử dụng swing hoặc linear( <https://api.jqueryui.com/easings/>)



* "swing" - chậm ở đoạn đầu & cuối, nhanh ở đoạn giữa.(có gia tốc)
* "linear" - tốc độ không đổi.

[**.hide(**](https://hocwebchuan.com/reference/jquery/jquery_hide.php#anchor04)*độ bền***,'*easing*',function(){...})**

### hire – Hiệu ứng mất

[**.hide(**](https://hocwebchuan.com/reference/jquery/jquery_hide.php#anchor04)*độ bền***,'*easing*',function(){...})**

### toggle- hiệu ứng hiển thị hoặc biến mất

[**.toggle(**](https://hocwebchuan.com/reference/jquery/jquery_hide.php#anchor04)*độ bền***,'*easing*',function(){...})**

*Lưu ý:*

* Một phần tử bị ẩn bởi phương thức toggle() thì cũng tương tự như việc phần tử đó được thiết lập thuộc tính display với giá trị none trong CSS.
* Phương thức toggle() chỉ có thể hiển thị những phần tử bị ẩn bởi thuộc tính display:none chứ không có tác dụng với phần tử bị ẩn bởi thuộc tính visibility:hidden trong CSS.

### slideDown- hiển thị với hiệu ứng trượt một phần tử đang ẩn

[**.slideDown(**](https://hocwebchuan.com/reference/jquery/jquery_hide.php#anchor04)*độ bền***,'*easing*',function(){...})**

- *Lưu ý:* Phương thức **slideDown()** chỉ có tác dụng với phần tử bị ẩn bởi thuộc tính display:none chứ không có tác dụng với phần tử bị ẩn bởi thuộc tính visibility:hidden trong CSS.

### slideUp- ẩn với hiệu ứng trượt

[**.slideUp(**](https://hocwebchuan.com/reference/jquery/jquery_hide.php#anchor04)*độ bền***,'*easing*',function(){...})**

- *Lưu ý:* Một phần tử bị ẩn bởi phương thức **slideUp()** thì cũng tương tự như việc phần tử đó được thiết lập thuộc tính display với giá trị none trong CSS.

### slideToggle- ẩn/hiện với hiệu ứng trượt

[**.slideToggle(**](https://hocwebchuan.com/reference/jquery/jquery_hide.php#anchor04)*độ bền***,'*easing*',function(){...})**

- Phương thức **slideToggle()** dùng để thực hiện luân phiên giữa việc hiện & ẩn một phần tử.

- *Lưu ý:*

* Một phần tử bị ẩn bởi phương thức slideToggle() thì cũng tương tự như việc phần tử đó được thiết lập thuộc tính display với giá trị none trong CSS.
* Phương thức slideToggle() chỉ có thể hiển thị những phần tử bị ẩn bởi thuộc tính display:none chứ không có tác dụng với phần tử bị ẩn bởi thuộc tính visibility:hidden trong CSS.

### fadeln – tăng dần opacity đến 1

[**.fadeIn(**](https://hocwebchuan.com/reference/jquery/jquery_hide.php#anchor04)*độ bền***,'*easing*',function(){...})**

*Lưu ý:* Phương thức **fadeIn()** chỉ có tác dụng với phần tử bị ẩn bởi thuộc tính display:none chứ không có tác dụng với phần tử bị ẩn bởi thuộc tính visibility:hidden trong CSS.

### fadeOut- giảm dần opacity đến 0

[**.fadeOut(**](https://hocwebchuan.com/reference/jquery/jquery_hide.php#anchor04)*độ bền***,'*easing*',function(){...})**

khi opacity về 0, selector sẽ có thuộc tính display: node, các thẻ dưới sẽ bị đẩy lên thế chỗ selector đó

- *Lưu ý:* Một phần tử bị ẩn bởi phương thức **fadeOut()** thì cũng tương tự như việc phần tử đó được thiết lập thuộc tính display với giá trị none trong CSS.

### fadeToggle -  luân phiên giữa việc hiện & ẩn một phần tử.

[**.fadeToggle(**](https://hocwebchuan.com/reference/jquery/jquery_hide.php#anchor04)*độ bền***,'*easing*',function(){...})**

khi opacity về 0, selector sẽ có thuộc tính display: node, các thẻ dưới sẽ bị đẩy lên thế chỗ selector đó

- Lưu ý:

Một phần tử bị ẩn bởi phương thức fadeToggle() thì cũng tương tự như việc phần tử đó được thiết lập thuộc tính display với giá trị none trong CSS.

Phương thức fadeToggle() chỉ có thể hiển thị những phần tử bị ẩn bởi thuộc tính display:none chứ không có tác dụng với phần tử bị ẩn bởi thuộc tính visibility:hidden trong CSS.

[**.fadeToggle(**](https://hocwebchuan.com/reference/jquery/jquery_hide.php#anchor04)*độ bền***,x,'*easing*',function(){...})**

### fadeTo – opacity dần đến x thuộc[0;1]

[**.fadeToggle(**](https://hocwebchuan.com/reference/jquery/jquery_hide.php#anchor04)*độ bền***,x,'*easing*',function(){...})**

có thể đậm lên hoặc nhạt đi

- *Lưu ý:* Phương thức **fadeTo()** chỉ làm cho phần tử bị phai màu chứ không khiến phần tử bị ẩn đi.

### animation

#### animate()

$(selector).animate({style}, độ bền,'easing',function(){...});

ref: <http://webcoban.vn/jquery/phuong-thuc-animate-trong-jquery-cach-tao-hieu-ung-chuyen-dong-cho-phan-tu-trong-jquery.html>

{stype} : <! bắt buộc>

- Chỉ định cặp thuộc tính và giá trị được thay đổi.

*(giá trị phải được đặt bên trong cặp dấu nháy đơn hoặc nháy kép)*

- *Ví dụ:* Nếu bạn muốn thay đổi giá trị thuộc tính width của phần tử thành 500px thì bạn phải viết là *width:"500px"*

- *Lưu ý:* Nếu bạn muốn chỉ định nhiều cặp thuộc tính và giá trị được thay đổi thì bạn phải đặt một dấu phẩy ở giữa mỗi hai cặp.

- *Lưu ý:* Tên thuộc tính sử dụng trong phương thức animate() tương đối khác so với tên thuộc tính trong CSS. Tên thuộc tính trong phương thức animate() phải được viết dưới dạng camel-cased *(tức là ký tự đầu tiên của từ thứ hai trở đi phải viết hoa)*

- *Lưu ý:* Phương thức animate() không hỗ trợ những thuộc tính mà giá trị của nó xác định bởi tên gọi, điển hình như: background-color, color, font-family, ....

VD:

$("button").click(function(){

$("p").animate(

{width:"300px",height:"300px"},

3500,

"linear",

function(){

alert("KẾT THÚC HIỆU ỨNG CHUYỂN ĐỘNG");

}

);

});

VD: có thể thêm vòng lặp để chạy liên tục, Lưu ý: nhưng để vô hạn sẽ gặp lỗi

$("button").click(function(){

for(let index =1; index<=5; index++){

$(".child").animate({left:"400px"},2000);

$(".child").animate({top:"200px"},2000);

$(".child").animate({left:"0px"},2000);

$(".child").animate({top:"0px"},2000);

}

});

VD: và đâu là 1 vòng lặp vô hạn

$(document).ready(function(){

colors = ['#FFB30C', '#58EC00', '#0087EC', '#EEEEEE', '#FF5A00' ]

var i = 0;

animate\_loop = function() {

$('body').animate({backgroundColor:colors[(i++)%colors.length]

}, 500, function(){

animate\_loop();

});

}

animate\_loop();

});

#### danh sách các thuộc tính được hỗ trợ trong phương thức animate():

* [backgroundPositionX](http://webcoban.vn/jquery/demo?file=2737)
* [backgroundPositionY](http://webcoban.vn/jquery/demo?file=2738)
* [borderWidth](http://webcoban.vn/jquery/demo?file=2739)
* [borderBottomWidth](http://webcoban.vn/jquery/demo?file=2740)
* [borderLeftWidth](http://webcoban.vn/jquery/demo?file=2741)
* [borderRightWidth](http://webcoban.vn/jquery/demo?file=2742)
* [borderTopWidth](http://webcoban.vn/jquery/demo?file=2743)
* [borderSpacing](http://webcoban.vn/jquery/demo?file=2744)
* [margin](http://webcoban.vn/jquery/demo?file=2745)
* [marginBottom](http://webcoban.vn/jquery/demo?file=2746)
* [marginLeft](http://webcoban.vn/jquery/demo?file=2747)
* [marginRight](http://webcoban.vn/jquery/demo?file=2748)
* [marginTop](http://webcoban.vn/jquery/demo?file=2749)
* [outlineWidth](http://webcoban.vn/jquery/demo?file=2750)
* [padding](http://webcoban.vn/jquery/demo?file=2751)
* [paddingBottom](http://webcoban.vn/jquery/demo?file=2752)
* [paddingLeft](http://webcoban.vn/jquery/demo?file=2753)
* [paddingRight](http://webcoban.vn/jquery/demo?file=2754)
* [paddingTop](http://webcoban.vn/jquery/demo?file=2755)
* [height](http://webcoban.vn/jquery/demo?file=2756)
* [width](http://webcoban.vn/jquery/demo?file=2757)
* [maxHeight](http://webcoban.vn/jquery/demo?file=2758)
* [maxWidth](http://webcoban.vn/jquery/demo?file=2759)
* [minHeight](http://webcoban.vn/jquery/demo?file=2760)
* [minWidth](http://webcoban.vn/jquery/demo?file=2761)
* [fontSize](http://webcoban.vn/jquery/demo?file=2762)
* [bottom](http://webcoban.vn/jquery/demo?file=2763)
* [left](http://webcoban.vn/jquery/demo?file=2764)
* [right](http://webcoban.vn/jquery/demo?file=2765)
* [top](http://webcoban.vn/jquery/demo?file=2766)
* [letterSpacing](http://webcoban.vn/jquery/demo?file=2767)
* [wordSpacing](http://webcoban.vn/jquery/demo?file=2768)
* [lineHeight](http://webcoban.vn/jquery/demo?file=2769)
* [textIndent](http://webcoban.vn/jquery/demo?file=2770)

#### stop()

dừng animation

.stop(*stopAll*,*goToEnd*);

|  |  |  |  |
| --- | --- | --- | --- |
| topAll |  | Không bắt buộc | - Tham số stopAll xác định việc nếu phần tử được thiết lập nhiều hiệu ứng chuyển động thì phương thức stop() sẽ chỉ dừng hiệu ứng chuyển động hiện tại hay là dừng hiệu ứng chuyển động hiện tại lẫn tất cả những hiệu ứng chuyển động còn lại.  - Tham số stopAll có thể được xác định bởi một trong hai giá trị:   * *true:* dừng hiệu ứng chuyển động hiện tại lẫn tất cả những hiệu ứng chuyển động còn lại. * *false:* chỉ dừng hiệu ứng chuyển động hiện tại.   - *Lưu ý:* Mặc định thì phương thức stop() chỉ dừng hiệu ứng chuyển động hiện tại, các hiệu ứng chuyển động còn lại vẫn được tiếp tục. |
| goToEnd |  | Không bắt buộc | - Tham số goToEnd xác định việc khi hiệu ứng chuyển động hiện tại bị dừng lại thì giá trị thuộc tính của hiệu ứng chuyển động đó chỉ bằng ở điểm dừng lại hay được chuyển đến trạng thái cuối cùng.  - Tham số goToEnd có thể được xác định bởi một trong hai giá trị:   * *true:* giá trị thuộc tính được chuyển đến trạng thái cuối cùng. * *false:* giá trị thuộc tính chỉ bằng ở điểm dừng lại.   - *Lưu ý:* Mặc định, khi hiệu ứng chuyển động hiện tại bị dừng lại thì giá trị thuộc tính của hiệu ứng chuyển động đó chỉ bằng điểm dừng lại. |

vd:

<script>

$(document).ready(function(){

$("#start").click(function(){

$(".child").animate({left:"400px"},3000);

$(".child").animate({top:"200px"},3000);

$(".child").animate({left:"0px"},3000);

$(".child").animate({top:"0px"},3000);

});

$("#stop").click(function(){

$(".child").stop(false,true);

});

});

</script>

## jQueryUI

url: <https://api.jqueryui.com/>

### option

$(selector).”ten widget/ interaction” ("option", "thuộc tính của option")

**vd**

$(selector).progressbar("option", "max")

$(selector).progressbar("option", "max", 1024) – set giá trị mới

### Methods

$(selector).”ten widget/ interaction” ("thuộc tính của methods")

**vd:**

$("#my-prb").progressbar("value")

### Events

tạo function trong lúc khởi tạo

### Tiện ích (widget)

#### dialog

giống như 1 messageBox trong C#

tương tự như 1 alert

cơ chế display: none <->block

#### datepicker

#### progressbar

Mình phải gắn vào 1 thẻ div

với max là giá trị lớn nhất

value tương ứng với độ chạy của thanh, 0<value<max

### Tương tác (interaction)

#### draggable

Cho phép các selector được di chuyển bằng chuột.

#### dropable

#### kết hợp với draggable

* gắn vào 1 thẻ div
* với accept là 1 selector . Thường là selector của 1 draggble cho phép chấp nhận thả lên nó
* áp dụng với việc thêm vào rỏ trong bán hàng

#### resizable

#### selectable

#### sortable

* gắn vào 1 thẻ <ul>, các <li> ở trong sẽ tự động được gắn

Cho phép sắp xếp lại các li trong ul

* xắp xếp lại ảnh,

##### option-connectWith: “selector”

selector: selector của 1 ul khác

cho phép liên kết với các ul khác nhau, có thể chuyển đổi <li> từ <ul> gốc sang <ul> trong connectWith

lưu ý: <ul> trong connectWith cũng phải là sortable

<li> sẽ thay đổi css theo những <li> bên <ul> connectWith

#### Hiệu ứng (effects)

#### easing

#### color animation

## Cơ chế chèn icon

html: <botton> <span>thẻ span để icon</span> </botton>

background-image: url(..);

position: absolute;

overflow: hidden; background-repeat: no-repeat;

Nếu image là 1 ảnh to gồm nhiều icon trong đó ó icon ta cần lấy ta sẽ dùng cơ chế hiển thị 1 phần của image

background-position: -…px -…px;

background-position

kích thước icon là : height: …; width: …;

botton

icon

icon

icon

icon

icon

icon

## This.

### disabled\*

### form\*

### formAction\*

### formMethod\*

### formNoValidate\*

### formTarget\*

### name

get, set name của nó

### type

get, set type của nó

### value

get, set type của nó dùng với thẻ input

### willValidate\*

### validity\*

### validationMessage\*

### labels\*

### title

get, set title của nó

### lang\*

### translate\*

### dir\*

### hidden\*

### accessKey\*

### draggable\*

### spellcheck\*

### autocapitalize\*

### contentEditable\*

### isContentEditable: \*

### inputMode\*

### parentNode:

* 1 biến kiểu offsetParent cũng là 1 this
* Nó cho biết mọi thông tin về thẻ cha của nó
* cha nó là thẻ bao ngoài của nó

vd:  
<div id="parent\_bt" name="parent-offer" title="parent" style="position:relative;">

        <button type="button" style="position: relative;">Click here</button>

    </div>

với

 $("button").click(function() {

        var obj = $(".display\_background .display\_img").offset();

        $("#s1").text(this.offsetTop + "px");

        var str = $(this).offsetParent().attr("id");

        $("#s2").text(str);

    });

Có thể .attr(“title”); .attr(“name”); .attr(“value”, 5);

hoặc .offsetTop;

hoặc . offsetParent(); tiếp để lấy thông tin ông nội nó

### offsetParent

offsetParent là phần tử cha gần nhất có positive: relative or absolute hoặc phần nội dung của trang. Còn [parentNode](#_parentNode:): là cha trực tiếp

### offsetTop

get,set tọa độ top của nó theo position, # với [offset();](#_.offer())

### offsetLeft

get,set độ left của nó theo position, # với [offset();](#_.offer())

### offsetWidth

get,set width của nó

### offsetHeight

get,set height của nó

### style\*

### innerText

get,set nội dung bên trong thẻ

*giống js dom content*

### outerText

get, set nội dung bao gồm cả thẻ

*giống js dom content*

# Service

## Kỹ thuật

## Worker

## DLL

## Packages

# Database

## SQL server

## My SQL

## Elastic Search

## MongoDB

# Visual studio

## Attach to process – ctrl +alt +p

Immediate window – ctrl + alt +i

## Show all brearkpoint – ctrl + alt + b

## To specify breackpoint – chột trái -> ctrl+F10

F11- nhẩy vào trong

Shift + F11 - nhẩy ra ngoài

F10 - nhẩy qua

Ctrl + ‘-‘ – quay ngược lại vị trí trước đó

# Visual studio code

# Nodejs service