**I. Biến, comments, built-in**

**1. Sử dụng JavaScript với HTML**

B1: Tạo folder thư mục rồi insert và VS code

B2: Tạo 2 new file main.js và index.html

B3: Trong file index.html thực hiện tạo 1 html đơn giản:

- Gõ “html5” => Enter

- Thêm thẻ script với src trỏ đến file main.js

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script src="main.js">

    </script>

</body>

</html>

B4: Trong file main.js tạo 1 ví dụ đơn giản:

alert('Hello')

B5: Chạy web sử dụng “Go Live”. Ấn tổ hợp Ctrl + Shift +I , chọn Console để xem giao diện run file js.

**2. Khái niệm biến và cách sử dụng**

**Biến là gì?**

Trong quá trình xây dựng website hoặc các ứng dụng với Javascript chúng ta sẽ cần phải làm việc với các dạng thông tin dữ liệu khác nhau. Ví dụ:

1. Phần mềm kế toán - Chúng ta sẽ làm việc với những con số
2. Website bán hàng - Làm việc với dữ liệu thông tin sản phẩm, đơn hàng và giỏ hàng
3. Ứng dụng Chat - Dữ liệu là những đoạn chat, tin nhắn, thông tin người chat

Biến được sử dụng để lưu trữ các thông tin trên trong quá trình ứng dụng Javascript hoạt động.

**Khai báo biến**

Để khai báo biến ta sẽ bắt đầu bằng từ khóa var (var là viết tắt của từ variable - nghĩa là biến). Khai báo biến có cú pháp như sau:

var [dấu cách] [tên biến];

Theo cú pháp trên, mình sẽ định nghĩa một biến có tên là fullName với dự định để lưu tên đầy đủ của mình vào đó.

var fullName;

Tiếp theo, ta có thể lưu thông tin vào biến fullName này:

var fullName; // khai báo biến

fullName = 'CudHnyuH'; // gán giá trị

Chú ý có dấu nháy đơn '' bao ngoài chữ CudHnyuH. Đó là cách để thể hiện dữ liệu dạng chuỗi (văn bản) trong Javascript.

Khi đoạn mã trên được chạy (thực thi) Javascript sẽ tạo biến với tên fullName và gán giá trị 'CudHnyuH' cho biến này. Một vùng nhớ trong RAM của máy tính sẽ được sử dụng để phục vụ việc lưu trữ những giá trị của biến khi chương trình được thực thi.

Chuỗi 'CudHnyuH' đã được lưu vào vùng nhớ tương ứng với biến fullName. Ta có thể truy cập tới chuỗi này qua tên biến:

var fullName;

fullName = 'CudHnyuH';

alert(fullName); // hiển thị giá trị của biến

Để đơn giản và ngắn gọn, ta có thể kết hợp việc khai báo biến và gán giá trị cho biến thành một dòng:

var fullName = 'CudHnyuH'; // khai báo và gán giá trị

alert(fullName);

Ta cũng có thể khai báo nhiều biến trong cùng một dòng cách nhau bởi dấu , như sau:

var fullName = 'CudHnyuH', age = 23, workAt = 'Hust';

Trông có vẻ ngắn gọn, tuy nhiên mình khuyên các bạn không nên dùng cách này. Khi cần khai báo nhiều biến hơn thì cách này trở nên rất khó đọc.  
  
Ta nên khai báo biến trên mỗi dòng khác nhau để dễ đọc hơn (nên dùng cách này):

var fullName = 'CudHnyuH';

var age = 23;

var workAt = 'Hust';

Một số cách khai báo biến trên nhiều dòng khác như sau:

var fullName = 'CudHnyuH',

age = 23,

workAt = 'Hust';

Thậm chí có cả phong cách sau:

var fullName = 'CudHnyuH'

, age = 23

, workAt = 'Hust';

Về mặt kỹ thuật thì tất cả các cách đều tương tự nhau. Vì vậy dùng cách nào là tùy theo sở thích của bạn.

Khi gán giá trị dạng số cho biến chúng ta không sử dụng dấu nháy đơn '' bao bọc bên ngoài. Như ví dụ trên thì age = 18 ta sẽ viết luôn là số 18 và không sử dụng dấu nháy.

**3. Cú pháp comments là gì?**

**/\*\* => Gợi ý comment trong VS**

## Giới thiệu

Hầu hết các ngôn ngữ lập trình đều có cú pháp dành cho việc comment code. Trình biên dịch sẽ bỏ qua những dòng và khối comment trong code của bạn. Vì vậy comment mang mục đích để lập trình viên có thể chú thích code mà không ảnh hưởng tới việc thực thi của ngôn ngữ lập trình.

## Cú pháp

#### Comment trên một dòng

Chúng ta sẽ sử dụng 2 dấu gạch chéo // trước comment. Cú pháp như sau:

// [Nội dung comment]

Áp dụng:

// Hàm alert sẽ bật lên hộp thoại

// trên trình duyệt

alert('Học lập trình tại F8');

Hoặc sử dụng ở phía sau dòng code:

var userName = 'sondn'; // Gán 'sondn' cho userName

#### Comment nhiều dòng - một khối

Cú pháp:

/\* [Nội dung comment] \*/

// hoặc

/\*

[Nội dung comment]

\*/

Áp dụng:

/\*\*

Ở bài trước chúng ta đã học về biến

Và hôm nay chúng ta học về comment

\*/

var fullName;

var age;

Bất cứ nội dung gì nằm giữa ký tự /\* và \*/ sẽ là comment.

## Sử dụng khi nào?

Chúng ta sử dụng comment với 2 mục đích chính:

#### Mục đích chú thích code

Chúng ta sẽ chú thích trên các đoạn code để hướng dẫn cách sử dụng, mô tả cách hoạt động hoặc giải thích tại sao ta lại làm như vậy. Việc làm này thường áp dụng đối với những đoạn code phức tạp, với những code đơn giản thì không cần sử dụng comment. Hãy áp dụng nguyên tắc đặt tên biến đã học để code của bạn viết ra sẽ tự giải thích luôn ý nghĩa của chính nó.

Đây là ví dụ cho việc comment để mô tả cách sử dụng:

/\*

Example:

var checked = true;

<Checkbox

checked={checked}

onChange={event => doSomething(event.target.checked)}

>

Label for Box

</Checkbox>

\*/

function Checkbox() {

...

}

Việc lạm dụng comment, sử dụng không đúng lúc, không đúng chỗ sẽ gây dư thừa không cần thiết, làm mã của bạn trở nên khó nhìn và gây phản tác dụng. Vì vậy hãy cân nhắc sự cần thiết mỗi khi có ý định sử dụng comment trong mã của bạn.

Ví dụ dưới đây mô tả việc sử dụng comment chưa hiệu quả:

// Khai báo biến

var email;

var password;

// Lấy giá trị người dùng nhập vào email input

emaill = event.target.value;

// Lấy giá trị người dùng nhập vào password input

password = event.target.value;

Comment như trên trở nên dư thừa không cần thiết vì đã là lập trình viên Javascript thì ai cũng sẽ hiểu đoạn code như vậy để làm gì.

Ngoại lệ khi bạn là người mới học, việc sử dụng comment như trên là cần thiết để giúp bạn ghi nhớ chức năng, nhiệm vụ của từng đoạn code. Khi bạn đã hiểu và ghi nhớ được những kiến thức đó - Hãy ngừng comment như vậy!

#### Mục đích Vô hiệu hóa đoạn code

Bản chất là trình biên dịch code sẽ bỏ qua những comment (không thực thi chúng). Vì vậy khi một đoạn code được comment lại thì đoạn code đó sẽ không chạy. Đôi khi bạn sẽ muốn tạm bỏ đi một đoạn code nào đó trong ứng dụng của bạn, đó chính là lúc bạn có thể sử dụng comment.

Ví dụ:

var email = 'email@domain.com';

var address = 'Hà Nội, Việt Nam';

alert(email); // alert này sẽ chạy

// alert(address); // alert này không chạy

* **Windows: Ctrl + /**
* **MacOS: Cmd + /**

**4. Thuật ngữ Built-in là gì?**

*a. Alert*

Alert(‘’);

VD: alert(‘Hello!’);

*b. Console*

console.log(‘’): in ra thông báo trên tag console

VD:

Var fullName = ‘CudHnyuH’;

Console.log(fullName);

*c. Confirm*

confirm(‘Xac nhan du tuoi!’);

*d. Prompt*

prompt(‘Xac nhan du tuoi!’);

*e. Set timeout*

setTimeout(funtion () {

alert(‘Thong bao’)

}, 1000);//Thực thi 1 đoạn code sau 1 thời gian ấn định, trong ví dụ này là 1000ms

*f. Set interval*

setInterval(funtion () {

console.log(‘Hello’);

},1000);//Thực thi đoạn code liên tục 1000ms/1 lần.

**5. Làm quen với toán tử**

a. Toán tử số học(Arithmetic)

var a = 1 + 2;

console.log(a);

*b. Toán tử gán(Assignment)*

var fullName = ‘CudHnyuH’;

*c. Toán tử so sánh(Comparison)*

var a=1;

var b=2;

if(a==b){

alert(‘Dung’);

};

*d. Toán tử logic(Logical)*

var a=1;

var b=2;

if(a>0 && b>0){

alert(‘a&b lon hon 0’);

};

**6. Toán tử số học**

*a. Cộng, trừ, nhân, lũy thừa, chia, chia lấy dư*

var a=1;

var b=2;

var c=a+b; //a-b; a\*b; a/b; a%b

console.log(c);

*b. Tăng/giảm 1 giá trị số*

var a=1;

a++; //a--

console.log(a);

**7. Toán tử gán**

=

+=

-=

\*=

/=

\*\*=

**8. Toán tử ++ và –**

Đây là 2 toán tử nghe qua thì rất dễ hiểu, nhưng để hiểu nguyên lý về cách hoạt động của nó chúng ta sẽ phải mất thêm một chút thời gian đó. Để hoàn thành bài học về 2 toán tử này, chúng ta sẽ cùng trải qua một số bài học sau nhé.

Ok, bắt đầu thôi!

## Toán tử ++

Toán tử ++ giúp tăng giá trị của một biến mang giá trị số lên 1. Có 2 cách để sử dụng toán tử ++ là:

1. Dùng làm hậu tố: variable++ (toán tử nằm sau biến)
2. Dùng làm tiền tố: ++variable (toán tử nằm trước biến)

### *#1 Sử dụng ++ làm hậu tố*

Ở đây, chúng ta sẽ xét ví dụ sử dụng toán tử ++ làm hậu tố trước (vì trong thực tế, chúng ta thường dùng kiểu hậu tố nhiều hơn):

var number = 1;

number++; // dùng làm hậu tố, ++ ở phía sau biến

console.log(number); // 2

number++;

console.log(number); // 3

Sau mỗi khi sử dụng toán tử ++, giá trị của biến number được tăng lên 1. Có vẻ khá dễ dàng để hiểu cách hoạt động của nó phải không?

Tuy nhiên, hãy xem xét thêm ví dụ sau:

var number = 1;

console.log(number++); // 1(Trả về giá trị trước khi tăng)

console.log(number); // 2

console.log(number++); // 2

console.log(number); // 3

*👉 Toán tử ++ khi dùng là hậu tố sẽ****tăng giá trị của biến lên 1****và****trả về giá trị trước khi tăng****.*

### #2 Sử dụng ++ làm tiền tố

Ở ví dụ này, chúng ta sử dụng ++ làm tiền tố. Tuy nhiên, kết quả trông sẽ không khác gì khi dùng ++ làm hậu tố:

var number = 1;

++number; // dùng làm tiền tố, ++ ở phía trước biến

console.log(number); // 2

++number;

console.log(number); // 3

Nhưng khi xem xét kỹ hơn, các bạn sẽ nhìn ra điểm khác:

var number = 1;

console.log(++number); // 2(Trả về giá trị sau khi tăng)

console.log(number); // 2

console.log(++number); // 3

console.log(number); // 3

*👉 Toán tử ++ khi dùng là tiền tố sẽ****tăng giá trị của biến lên 1****và****trả về giá trị sau khi tăng***.

## Toán tử - -

Cách hoạt động tương tự như toán tử ++, điểm khác biệt là thay vì cộng thêm 1, thì toán tử -- sẽ trừ đi 1.

## Tổng kết

* x++ tăng giá trị biến lên 1 và trả về giá trị **trước** khi tăng
* ++x tăng giá trị biến lên 1 và trả về giá trị **sau** khi tăng
* x-- giảm giá trị biến xuống 1 và trả về giá trị **trước** khi giảm
* --x giảm giá trị biến xuống 1 và trả về giá trị **sau** khi giảm
* var a = 1;
* var b = ++a \* a--;
* console.log(b); // Output: ?

## Giải thích

Toán tử ++ hoặc -- khi được dùng làm **tiền tố** (++a) sẽ thực hiện phép toán + hoặc - 1 đơn vị trước rồi mới trả về giá trị.

Còn khi được sử dụng làm **hậu tố** (a--) thì sẽ trả về giá trị trước rồi mới thực hiện + hoặc -.

var a = 1;

var b = ++a \* a--;

// var b = 2 \* 2

++a thực hiện phép + trước rồi mới trả về giá trị => ++a = 2 và giá trị a được lưu =2.

a-- trả về giá trị trước(a-- = 2) rồi mới thực hiện phép trừ và giá trị a =1

**9. Toán tử nối chuỗi**

Var firstName = ‘Cud’;

Var lastName = ‘HnyuH’;

Console.log(firstName+lastName);

**10. Kiểu dữ liệu Boolean**

Var age =16;

Var canBuyAlcohol = age >= 18;

**11. Câu lệnh điều kiện If/else**

If ( ) {

} else {

};

**12. Toán tử logical( &&; ||; !)**

Var a=1;

Var b=2;

Var c=3;

If (a>0 && b>0) {

Console.log(‘Dieu kien dung!’);

};

**13. Kiểu dữ liệu**

- Number type

Var a=1;

- String type

Var fullName = ‘CudHnyuH’;

- Boolean type

Var isSuccess = true;

- Undefined type

Var age;

- Null type

Var isNull = null;

- Symbol type

Var id = Symbol(‘id’);// unique: tính duy nhất

- Function type

Var myFunction = function(){

Alert(‘Hello!’);

};

- Object type

Var myObject = {

Name: ‘CudHnyuH’,

Age: 23,

myFunction: function(){}

};

Var myArray = [

‘Java’,

‘Python’

};

- Cách kiểm tra kiểu dữ liệu:

Console.log(typeof …);

VD:

var a = '1';

var b = 2;

var c = typeof a;

var d = typeof b;

var e = typeof d;

console.log(c, d, e) // Output: ?

## Giải thích

typeof của **1 số** sẽ trả về 'number'.

typeof của **1 chuỗi** sẽ trả về 'string'.

Chú ý: Kết quả trả về của typeof sẽ luôn là 1 chuỗi, vậy nên typeof của d sẽ là 'string'.

**14. Toán tử so sánh II**

===(3 dấu =): bằng tuyệt đối

!==:khác tuyệt đối

Var a = 1;

Var b = ‘1’;

Console.log(a == b);// true

Console.log(a === b);// false

**15. Truthy và Falsy là gì?**

**ruthy - to bool is true**

Bất cứ giá trị nào trong Javascript khi chuyển đổi sang kiểu dữ liệu boolean mà có giá trị true thì ta gọi giá trị đó là Truthy.

Các giá trị 1, ['BMW'], { name: 'Miu' } và 'hi' được đề cập trong ví dụ dưới đây là Truthy vì khi chuyển sang Boolean ta nhận được giá trị true.

Ví dụ:

console.log(Boolean(1)) // true

console.log(Boolean(['BMW'])) // true

console.log(Boolean({ name: 'Miu' })) // true

console.log(!!'hi') // true

!! là gì? Đơn giản thôi. Toán tử ! là toán tử not (phủ định) nên !! là 2 lần phủ định, mà 2 lần phủ định lại trở thành "khẳng định". Trong Javascript thì đây là một "tip" để convert (chuyển đổi) mọi kiểu dữ liệu khác sang Boolean.

Ví dụ:

console.log(!!1) // true

console.log(!!'f8') // true

console.log(!!['Mercedes']) // true

Thêm !! phía trước các giá trị truthy sẽ luôn trả về true.

**Falsy - to bool is false**

Bất cứ giá trị nào trong Javascript khi chuyển đổi sang kiểu dữ liệu boolean mà có giá trị false thì ta gọi giá trị đó là Falsy.

Trong Javascript có 6 giá trị sau được coi là Falsy:

1. false
2. 0 (số không)
3. '' or "" (chuỗi rỗng)
4. null
5. undefined
6. NaN

Ví dụ:

console.log(!!false) // false

console.log(!!0) // false

console.log(!!'') // false

console.log(!!null) // false

console.log(!!undefined) // false

console.log(!!NaN) // false

**Chú ý!**

Nội dung đã đề cập phía trên đã đầy đủ khi nói về Truthy và Falsy trong Javascript. Tuy nhiên mình vẫn cần nhấn mạnh lại với các bạn rằng:

Ngoài 6 giá trị đã đề cập tới ở phần Falsy thì toàn bộ các giá trị khác đều là Truthy, kể cả những giá trị sau:

1. '0' (một chuỗi chứa số không)
2. ' ' (một chuỗi chứa dấu cách)
3. 'false' (một chuỗi chứa từ khóa false)
4. [] (một array trống)
5. {} (một object trống)
6. function(){} (một hàm "trống")

Một số người chuyển từ ngôn ngữ khác sang rất có thể sẽ bị nhầm [] (mảng "rỗng") là falsy, bởi vì trong ngôn ngữ họ đã học trước đó [] là falsy.

Với những người hiểu nhầm [] là falsy sẽ gặp trường hợp khó hiểu sau:

Ví dụ:

var cars = [] // Dù là mảng "rỗng" vẫn là truthy

if (!cars) {

// Họ sẽ thắc mắc: "Tại sao lại không lọt vào đây?"

}

Vì [] là truthy nên !cars sẽ trả về false. Câu lệnh if sẽ nhận được kết quả của mệnh đề so sánh là false, vì vậy đoạn mã trong if trên sẽ không được lọt vào.

**Ngoại lệ? - document.all**

Trong Javascript (phía trình duyệt) sẽ có sẵn một đối tượng document, và khi bạn thử !!document.all sẽ trả về false. Chẳng lẽ document.all cũng là falsy hay sao?

Bản thân mình cũng thắc mắc điều này nên mình đã search Google "Why document.all is falsy?" và mình đã tìm được câu trả lời [tại đây](https://fullstack.edu.vn/external-url?continue=https%3A%2F%2Fstackoverflow.com%2Fquestions%2F10350142%2Fwhy-is-document-all-falsy).

**Tóm tắt câu trả lời:**

document.all là một ngoại lệ chính thức duy nhất theo đặc tả ECMA (phiên bản 5). Đặc tả này mô tả toàn bộ các object khi chuyển sang boolean sẽ là true. Tuy nhiên, document.all là một ngoại lệ.

**Cụ thể như sau:**

1. document.all chuyển sang boolean sẽ là false
2. document.all khi là toán hạng của toán tử so sánh == hoặc != sẽ là undefined
3. Khi typeof document.all sẽ trả về "undefined"

ECMA là đặc tả chi tiết kỹ thuật mà các ngôn ngữ theo đặc tả này phải tuân theo. Javascript là một ngôn ngữ tuân thủ đặc tả kỹ thuật ECMA.

VD:

var a = '';

var b = 0;

var c = [];

var d = 1 > 2;

var e = {};

var f = '0';

Những biến nào sau đây có giá trị là falsy? => a,b,d

## Giải thích

Các giá trị falsy: a **(chuỗi rỗng)**, b **(số 0)**, d **(false)**

Các giá trị truthy: c **(mảng rỗng)**, e **(object rỗng)**, f **(chuỗi)**

**16. Toán tử logical và câu lệnh điều kiện If**

Var a=1;

Var b=2;

Var result = a<b;

Console.log(‘result’, result);

If(a<b) {

Console.log(‘A<B’);

} else {

Console.log(‘A>B’);

};

Var result = ‘A’ && ‘B’ && ’C’;

Console.log(result);// C

Var result = ‘A’ || ‘B’ || ‘C’;

Console.log(result);// A

**II. Làm việc với hàm**

**1. Khái niệm hàm (function)**

Function showDialog() {

// code

Alert(‘Hello!’);

};

showDialog();

VD:

function showMessage(message) {

console.log(message);

}

showMessage("Hi anh em F8!");

Chọn câu trả lời đúng.

Message là tham số(parameter).

**2. Tham số trong hàm**

function writeLog(message) {

console.log(message);

};

writeLog(‘Test message’);

function writeLog() {

console.log(arguments)

};

writeLog(‘Log1’, ‘Log2’, ‘Log3’);

function writeLog() {

for (var param of arguments) {

console.log(param)

}

};

writeLog(‘Log1’, ‘Log2’, ‘Log3’);

VD1:

Hãy tạo 1 hàm có tên là sum có 2 tham số:

* Tham số thứ 1 là a
* Tham số thứ 2 là b

Function sum(a,b) {};

**3. Từ khóa return trong hàm(return: trả về)**

Var isConfirm = confirm(‘Message?’);

Console.log(isConfirm);

Function cong(a,b) {

Return a+b;

}

Var result = cong(2,8);

Console.log(result);

**4. Các loại hàm**

a. Declaration function

function showMessage() {}

b. Expression function

var showMessage2 = function() {}

setTimeout(function() {});

var myObject = {

myFunction: function() {

}

}

VD:

Tạo 2 hàm declarationFunction và expressionFunction thỏa mãn các yêu cầu sau đây:

1. declarationFunction là 1 **declaration function**
2. expressionFunction là 1 **expression function**
3. function declarationFunction() {};
4. var expressionFunction = function() {};

**III. Làm việc với chuỗi**

**1. Kiểu dữ liệu chuỗi (string)**

Có 2 cách tạo chuỗi:

Var fullName = ‘CudHnyuH’;// nên dùng

Var fullName = new String(‘CudHnyuH’);

Var fullName = ‘CudHnyuH is \‘the best\’’;// backslash in javascript

Console.log(fullName.length);// kiểm tra độ dài 1 chuỗi

Var firstName = ‘Cud’;

Var lastName = ‘HnyuH’;

Console.log(`Toi la: ${firstName} ${lastName}`);

**2. Làm việc với chuỗi**

// Javascript string methods

Var myString = ‘CudHnyuH’;

Console.log(myString.length);// Length

Console.log(myString.indexOf(‘ABC’));//Find index, lastIndexOf: trả về vị trí cuối cùng

Console.log(myString.slice(4,6));//Cut String

Console.log(myString.replace(‘u’, ‘U’));// Replace

Console.log(myString.toUpperCase());// Convert to upper case

Console.log(myString.toLowerCase());// Convert to lower case

Console.log(myString.trim()); // Trim

Console.log(myString.split(‘’));// Split

Console.log(myString.charAt(0));//Get a character by index

VD:

Hãy tạo hàm getUpperCaseName có 1 tham số là name, hàm này sẽ trả về phiên bản chữ viết hoa của giá trị mà nó nhận được từ name.

function getUpperCaseName(name) {

   return name.toUpperCase();

};

// Expected results:

console.log(getUpperCaseName("Nguyen van a")) // "NGUYEN VAN A"

console.log(getUpperCaseName("nGuyen vAn C")) // "NGUYEN VAN C"

**IV. Làm việc với số**

**1. Kiểu dữ liệu số(number)**

***Kiểu số - Number***

Trong Javascript có 2 loại số:

1. Các số thông thường trong Javascript được lưu trữ dưới dạng 64 bit IEEE-754 (còn được gọi là các số phẩy động). Trong hầu hết các trường hợp khi làm việc với Javascript là chúng ta sử dụng loại số này.
2. Số BigInt là loại số sử dụng để biểu thị số nguyên có độ dài tùy ý.

Vì mức độ thông dụng nên ở đây chúng ta chỉ tìm hiểu về loại số thông thường.

***Cách khai báo***

Cách thông thường khi ta khai báo một số. Ví dụ là: 1000000 (một triệu)

var million = 1000000;

Cũng là khai báo số 1000000 nhưng có cách viết khác. Bạn có thể thêm chữ e vào sau số 1 và chỉ định số số không phía sau chữ e như sau:

var million = 1e6; // tương tự: 1000000

// hoặc

var billion = 2e9; // tương tự: 2000000000 (hai tỏi à nhầm hai tỉ)

***Đối tượng Number***

Đối tượng Number trong Javascript là đối tượng giúp chúng ta định nghĩa số và làm việc với số. Chúng ta thường sử dụng các phương thức sau của đối tượng Number khi làm việc với số trong Javascript:

| **Phương thức** | **Vai trò** |
| --- | --- |
| Number.isFinite() | Xác định xem giá trị đã cho có phải là số hữu hạn hay không. Trả về boolean |
| Number.isInteger() | Xác định xem giá trị đã cho có phải là số nguyên hay không. Trả về boolean |
| Number.parseFloat() | Chuyển đổi chuỗi đã cho thành một số dấu phẩy động |
| Number.parseInt() | Chuyển đổi chuỗi đã cho thành một số nguyên |
| Number.prototype.toFixed() | Chuyển đổi và trả về chuỗi đại diện cho số đã cho, có số chữ số chính xác sau dấu thập phân |
| Number.prototype.toString() | Chuyển đổi và trả về số đã cho dưới dạng chuỗi |

Ví dụ:

Number.isFinite(2 / 0); // false

Number.isFinite(20 / 5); // true

Number.isFinite(0 / 0); // false

Number.isInteger(999999999); // true

Number.isInteger(0.2); // false

Number.isInteger(Math.PI); // false

Number.parseFloat('10') // 10

Number.parseFloat('10.00') // 10

Number.parseFloat('238,21') // 238

Number.parseFloat('237.22') // 237.22

Number.parseFloat('34 56 78') // 34

Number.parseFloat(' 37 ') // 37

Number.parseFloat('18 is my age') // 18

Number.parseInt('10') // 10

Number.parseInt('10.00') // 10

Number.parseInt('238,21') // 238

Number.parseInt('237.22') // 237

Number.parseInt('34 56 78') // 34

Number.parseInt(' 37 ') // 37

Number.parseInt('18 is my age') // 18

var numberObject = 1234.56789;

numberObject.toFixed(); // '1235'

numberObject.toFixed(1); // '1234.6'

numberObject.toFixed(6); // '1234.567890'

(11).toString(); // '11'

(18).toString(); // '18'

(17.3).toString(); // '17.3'

**2. Số và làm việc với số**

Var age = 23;// nên dùng

Var PI = 3.14;

Var otherNumber = new Number(23);

Var result = 10/’a’;

Console.log(isNaN(result));

Console.log(age.toString());// int to string

Console.log(PI.toFixed());// To Fixed

VD:

* Tạo hàm isNumber, hàm này dùng với mục đích kiểm tra xem một giá trị có phải là 1 số hay không
* Hàm isNumber có 1 tham số, hãy đặt tên nó là value
* Khi value có kiểu số, hàm sẽ trả về true, ngược lại trả về false

function isNumber(value) {

   return Number.isFinite(value)

}

// Expected results:

console.log(isNumber(999)); // true

console.log(isNumber('abc')); // false

console.log(isNumber('100')); // false

function isNumber(value) {

   return typeof value  === 'number';

}

// Expected results:

console.log(isNumber(999)); // true

console.log(isNumber('abc')); // false

console.log(isNumber('100')); // false

**V. Làm việc với mảng**

**1. Mảng(Array)**

Var languages = [

‘Javascript’,

‘PHP’,

‘Ruby’

];

Console.log(languages);

Var languages = new Array(

‘Javascript’,

‘PHP’,

‘Ruby’

);

Console.log(typeof languages);

Console.log(Array.isArray(…));

Console.log(languages.length);// Length( Độ dài mảng)

Console.log(languages[0]);//Lấy phần tử theo index

**2. Làm việc với mảng**

Console.log(languages.toString()); // To String

Console.log(languages.join(‘ - ’)); // Join

Console.log(languages.pop()); // Pop: Xóa phần tử cuối mảng và trả về phần tử đã xóa.

Console.log(languages.push(‘Dart’)); // Push: Thêm phần tử và cuối mảng và trả về độ dài.

Console.log(languages.shift()); // Shift: Xóa phần tử đầu mảng và trả về phần tử đã xóa.

Console.log(languages.unshift()); // Unshift: Thêm 1 hoặc nhiều phần tử và đầu mảng.

Console.log(languages.splice()); //Splicing: Xóa, chèn

Console.log(languages.concat(languages2));// Concat: Nối mảng

Console.log(languages.slice(1, 2)); // Slice: Cắt phần tử của mảng

*array*.slice(start, end)

VD:

Cho trước hàm joinWithCharacter có 2 tham số là array và charactor, hãy hoàn thành phần nội dung hàm sao cho hàm trả về 1 chuỗi là kết quả của việc nối từng phần tử của mảng với charactor.

function joinWithCharacter(array, charactor) {

return array.join(charactor)

}

// Ví dụ khi sử dụng

var cars = ['Honda', 'Mazda', 'Mercedes'];

var result = joinWithCharacter(cars, ' - ');

console.log(result); // Expected: "Honda - Mazda - Mercedes"

Để vượt qua thử thách này, hãy tạo hàm getLastElement có 1 tham số (F8 sẽ gọi hàm này và luôn truyền đối số là 1 array), hàm này sẽ trả về phần tử cuối cùng trong mảng.

// Viết hàm tại đây

function getLastElement(animals) {

    return animals.slice(-1)

};

// Ví dụ sử dụng

var animals = ['Monkey', 'Tiger', 'Elephant'];

var result = getLastElement(animals);

console.log(result); // Expected: "Elephant"

console.log(animals); // Expected: ['Monkey', 'Tiger', 'Elephant']

**VI. Làm việc với Object**

**1. Object là gì?**

Var myInfo = {

Name: ‘CudHnyuH’,

Age: 23,

Address: ‘VN’

};

myInfo.email = ‘phamhuynhduc805@gmail.com’;

Console.log(myInfo);

Console.log(myInfo.Name); // Kết quả trả về CudHnyuH.

Var myKey = ‘address’;

Console.log(myInfo[myKey]);

Delete myInfo.age; // Xóa key age trong Object.

Var emailKey = ‘email’;

Var myInfo = {

Name: ‘CudHnyuH’,

Age: 23,

Address: ‘VN’,

[emailKey]: ‘phamhuynhduc805@gmail.com’,

getName: function() {

return this.Name;

}

};

Console.log(myInfor.getName());// Kết quả trả về CudHnyuH.

// Function 🡪 Phương thức( method)

// Others 🡪 Thuộc tính( property)

**2. Object constructor( xây dựng đối tượng)**

Function User(firstName, lastName, avatar) {

This.firstName = firstName;

This.lastName = lastName;

This.avatar = avatar;

This.getName = function () {

Return `${this.firstName} ${this.lastName}`

}

};

Var author = new User(‘Cud’, ‘HnyuH’, ‘Avatar’);

Var user = ner User(‘Huynh’, ‘Duc’, ‘Avatar’);

Author.title = ‘hihihaha’;

User.comment = ‘abc’;

Console.log(author);

Console.log(user);

VD:

Hãy tạo object constructor cho nhóm Animal có 3 thuộc tính là name, leg và speed. Trong đó, name là để lưu tên động vật, leg là số lượng chân và speed là tốc độ di chuyển của động vật đó.

Sau đó, hãy tạo một đối tượng từ object constructor trên để mô tả 1 con vẹt, lưu đối tượng vào biến parrot.

function Animal(name, leg, speed){

    this.name = name;

    this.leg = leg;

    this.speed = speed;

};

var parrot = new Animal('Cud', 2, 12);

**3. Object prototype( Đối tượng nguyên mẫu)**

User.prototype.className = ‘A1’;// Thêm thuộc tính từ bên ngoài vào hàm tạo

User.prototype.getClassName = function() {

Return this.className;

}

VD:

Để vượt qua thử thách này, bạn hãy tạo một object constructor Student gồm: firstName, lastName. Sau đó, định nghĩa thêm phương thức là getFullName, phương thức này sẽ trả về tên đầy đủ của sinh viên.

// Làm bài tại đây

function Student(firstName, lastName){

    this.firstName = firstName;

    this.lastName = lastName;

};

Student.prototype.getFullName = function() {

    return `${this.firstName} ${this.lastName}`

}

// Ví dụ khi sử dụng

var student = new Student('Long', 'Bui');

console.log(student.firstName);  // 'Long'

console.log(student.lastName);  // 'Bui'

console.log(student.getFullName());  // 'Long Bui'

**4. Đối tượng Date**

**// Javascript date object mozilla**

Var date = new Date();

Var year = date.getFullYear();

Var month = date.getMonth() + 1;// giá trị trả về từ 0🡪11

Var day = date.getDate();

Console.log(`${day}/${month}/${year}`);

VD:

Để vượt qua thử thách này, bạn hãy tạo hàm getNextYear, hàm này sẽ trả về năm kế tiếp. Ví dụ, năm nay là 2022, hàm sẽ trả về 2023 là kiểu số.

function getNextYear() {

    var date = new Date()

    return date.getFullYear()+1;

}

**5. Math Object**

Console.log(Math.PI);

Console.log(Math.round());// làm tròn số

Console.log(Math.abs());// giá trị tuyệt đối

Console.log(Math.ceil());// làm tròn trên

Console.log(Math.floor());// làm tròn dưới

Console.log(Math.random());// trả về số thập phân ngẫu nhiên nhỏ hơn 1

VD1:

Var random = Math.floor(Math.random() \* 5);

Var bonus = [

’10 coin’,

’20 coin’,

’30 coin’,

’40 coin’,

’50 coin’

];

Console.log(bonus[random]);

VD2:

Var random = Math.floor(Math.random() \* 100);

If (random <5) {

Console.log(‘Thanh cong’);

}

Console.log(Math.min(-100, 1, 2));// trả về -100 , tương tự với Math.max

**VII. Lệnh rẽ nhánh, toán tử 3 ngôi**

**1. Lệnh rẽ nhánh If else**

Var date = 2;

If (date === 2) {

Console.log(‘Hom nay la thu 2’);

} else if (date === 3) {

Console.log(‘Hom nay la thu 3’);

}else {

Console.log(‘Khong biet’);

};

VD:

Để vượt qua thử thách này, hãy hoàn thành hàm cho trước để đáp ứng các yêu cầu sau:

1. Khi a chia hết cho 3 thì return về 1
2. Khi a chia hết cho 5 thì return về 2
3. Khi a chia hết cho 15 thì return về 3
4. function run(a) {
5. if (a%15 === 0) {
6. return 3;
7. } else if(a%5 === 0) {
8. return 2;
9. } else if(a%3 === 0) {
10. return 1;
11. };
12. };
13. // Kỳ vọng
14. console.log(run(3)) // 1
15. console.log(run(5)) // 2
16. console.log(run(15)) // 3

**2. Lệnh rẽ nhánh Switch**

Var date = 2;

Switch(date) {

Case 2:

Console.log(‘Hom nay la thu 2’);

Break;

Default:

Console.log(‘Khong biet’);

};

function run(fruits) {

    var result;

    switch (fruits) {

        case "Banana":

            result = "This is a Banana";

            break;

        case "Apple":

            result = "This is an Apple";

            break;

        default:

            result = "No fruits";

    }

    return result;

}

**3. Toán tử 3 ngôi (Ternary operator)**

**Cách 1:**

Var course = {

Name: ‘Javascript’;

Coin: 250

};

If (course.coin > 0) {

Console.log(`${course.coin} Coins`);

} else {

Console.log(‘Mien phi’);

}

**Cách 2:**

Var course = {

Name: ‘Javascript’;

Coin: 250

};

Var result = course.coin > 0 ? `${course.coin} Coins` : ‘Mien phi’;

Console.log(result);

VD:

Để vượt qua thử thách này bạn cần tạo hàm getCanVoteMessage, hàm này có 1 tham số là age. Trong trường hợp từ 18 tuổi trở lên hàm sẽ trả về Bạn có thể bỏ phiếu, ngược lại hàm trả về Bạn chưa được bỏ phiếu.

Sử dụng toán tử 3 ngôi trong bài này, không sử dụng if/else bạn nhé.

// Làm bài

function getCanVoteMessage(age) {

    return age>=18 ? 'Bạn có thể bỏ phiếu' : 'Bạn chưa được bỏ phiếu';

}

// Kỳ vọng

console.log(getCanVoteMessage(18)) // 'Bạn có thể bỏ phiếu'

console.log(getCanVoteMessage(15)) // 'Bạn chưa được bỏ phiếu'

**VIII. Vòng lặp**

**1. Giới thiệu vòng lặp**

a. for – Lặp với điều kiện đúng

b. for/in – Lặp qua key của đối tượng

c. for/of – Lặp qua điều kiện đúng

d. while – Lặp khi điều kiện đúng

e. do/while – Lặp ít nhất 1 lần, sau đó lặp khi điều kiện đúng

**2. Vòng lặp For**

For (var i=1; i<=1000; i++) {

Console.log(i);

};

VD:

Hãy tạo hàm getRandNumbers có 3 tham số là min, max, length. Hàm này sẽ trả về một mảng gồm length phần tử, các giá trị trong mảng là số ngẫu nhiên, giá trị trong khoảng từ min tới max.

Gợi ý: Math.random() \* (max - min) + min là cách tạo ra 1 số ngẫu nhiên trong khoảng min - max.

function getRandNumbers(min, max, length) {

    var arr = []

    for (var i=0; i<length; i++){

        arr[i] = Math.random() \* (max - min) + min;

    }

    return arr

}

Cho trước mảng numbers, hãy viết hàm getTotal trả về tổng giá trị các phần tử của mảng.

function getTotal(arr) {

    var sum = 0

    for (var i=0; i<arr.length; i++){

        sum += arr[i];

    }

    return sum;

}

Var myArray = [

‘JS’,

‘PHP’

];

Console.log(myArray[0]);

// var arrayLength = myArray.length;

// for (var i=0; i<arrayLength; i++) {

// console.log(i);

//};

VD:

Cho trước mảng orders là danh sách chứa các khóa học, các mặt hàng này được thể hiện dưới dạng object và đều có 1 key là price để thể hiện giá trị của mặt hàng đó.

Bạn hãy hoàn thành hàm getTotal để tính được tổng giá trị của đơn hàng.

var orders = [

    {

        name: 'Khóa học HTML - CSS Pro',

        price: 3000000

    },

    {

        name: 'Khóa học Javascript Pro',

        price: 2500000

    },

    {

        name: 'Khóa học React Pro',

        price: 3200000

    }

]

function getTotal(orders) {

    var ArrLength = orders.length;

    var sum = 0;

    for (var i=0; i< ArrLength; i++) {

        sum += orders[i].price;

    }

    return sum;

}

// Expected results:

getTotal(orders) // Output: 8700000

**3. Vòng lặp For…in**

Var myInfo = {

Name: ‘CudHnyuH’,

Age: 23,

Address: ‘VN’

};

For (var key in myInfo) {

Console.log(key);

// console.log(myInfo[key]);

};

function run(object) {

    var arr = [];

    for(var key in object){

        arr.push(`Thuộc tính ${key} có giá trị ${object[key]}`)

    }

    return arr

}

// Expected results:

console.log(run({ name: 'Nguyen Van A', age: 16 }));

// Output:

// [

//     "Thuộc tính name có giá trị Nguyen Van A",

//     "Thuộc tính age có giá trị 16"

// ]

**4. Vòng lặp for…of**

For (var value of languages) {

Console.log(value);

};

Console.log(Object.keys(myInfor));

**5. Vòng lặp While**

Var i=0;

While (i<1000) {

i++;

console.log(i);

};

**6. Vòng lặp do…while**

Var i = 0;

Var isSuccess = false;

do {

i++;

console.log(‘Nap the lan ’ + i);

if(true) {

isSuccess = true;

};

} while(!isSuccess && i<3);

**7. Break và Continue trong vòng lặp**

For (var i=0; i<10; i++) {

Console.log(i);

If (i>=5) {

break;

};

};

For (var i=0; i<10; i++) {

If (i%2 !==0){

continue;

}

Console.log(i);

};

**8. Vòng lặp lồng nhau (Nested loop)**

Var myArray = [

[1,2],

[3,4],

[5,6]

];

For (var i=0; i<myArray.length;i++) {

For(var j=0; j< myArray[i].length; j++) {

Console.log(myArray[i][j]);

};

//Console.log(myArray[i]);

};

**9. Đệ quy**

**IX. Làm việc với mảng II**

**1. Làm việc với mảng - Phần 2**

Var courses = [

{

Id : 1,

Name: ‘Javascript’,

Coin: 250

},

{

Id : 2,

Name: ‘HTML, CSS’,

Coin: 0

},

{

Id : 3,

Name: ‘Ruby’,

Coin: 0

},

{

Id : 4,

Name: ‘PHP’,

Coin: 400

},

{

Id : 5,

Name: ‘ReactJs’,

Coin: 500

},

];

Courses.forEach(function(course, index) {

Console.log(index, course);

});

Var isFree = Courses.every(function(course, index) {

Return course.coin === 0 ;

});

Console.log(isFree);// Kiểm tra tất cả các phần tử phải thỏa mãn 1 điều kiện gì đó

Var isFree = Courses.some(function(course, index) {

Return course.coin === 0 ;

});

Console.log(isFree);// Kiểm tra 1 phần tử thỏa mãn điều kiện

Var course = Courses.find(function(course, index) {

Return course.name === ‘Ruby’ ;

});

Console.log(course);

Var listCourse = Courses.filter(function(course, index) {

Return course.name === ‘Ruby’ ;

});

Console.log(listCourse);

VD:

Để vượt qua thử thách này, hãy tạo hàm getMostFavoriteSport có 1 tham số (F8 sẽ gọi hàm này và luôn truyền đối số là 1 array). Hàm getMostFavoriteSport sẽ trả về các môn thể thao có điểm số yêu thích lớn hơn 5.

Gợi ý: Sử dụng phương thức filter.

const sports = [

    {

        name: 'Bóng rổ',

        like: 6

    },

    {

        name: 'Bơi lội',

        like: 5

    },

    {

        name: 'Bóng đá',

        like: 10

    },

]

function getMostFavoriteSport(arr) {

   return arr.filter(function(sports){

       return sports.like > 5

   })

}

// Kỳ vọng

console.log(getMostFavoriteSport(sports))

// Output: [{ name: 'Bóng rổ, like: 6 }, { name: 'Bóng đá, like: 10 }]

**2. Array map() method**

Var newCourses = courses.map(function courseHandler(course) {

Return {

Id: course.id,

Name: `Khoa hoc: ${course.name}`,

Coin: course.coin,

coinText: `Gia: ${course.coin}`,

index: index,

originArray: courses,

};

});

Console.log(newCourse);

**3. Array reduce() method**

Var totalCoin = 0;// Biến lưu trữ

// Lặp qua các phần tử

For (var course of courses) {

// Thực hiện việc lưu trữ

totalCoin += course.coin;

};

Console.log(totalCoin);

Function coinHandler(accumulator, currentValue) {

return accumulator + currentValue.coin;

};

Var totalCoin = courses.reduce(coinHandler, 0);

Console.log(totalCoin);

VD:

Tại SEA GAMES 31 vừa qua, đoàn thể thao Việt Nam đã đứng đầu bảng tổng sắp huy chương. Hãy tạo hàm getTotalGold có 1 tham số là mảng. Tính tổng số huy chương vàng mà đoàn thể thao Việt Nam đạt được trong kỳ SEA Game lần này.

var sports = [

    {

        name: 'Bơi lội',

        gold: 11

    },

    {

        name: 'Boxing',

        gold: 3

    },

    {

        name: 'Đạp xe',

        gold: 4

    },

    {

        name: 'Đấu kiếm',

        gold: 5

    },

]

function getTotalGold(arr) {

    return arr.reduce(function(accumulator, currentValue) {

        return accumulator + currentValue.gold;

    }, 0)

}

// Expected results:

console.log(getTotalGold(sports)) // Output: 23

var totalCoin = courses.reduce(function(total, course){

return total + course.coin;

},0);

Console.log(totalCoin);

// Flat – “Làm phẳng” mảng từ Depth array- “Mảng sâu”

Var depthArray = [1,2,[3,4],5,6,[7,8,9]];

Var flatArray = depthArray.reduce(function(flatOutput, depthItem){

Return flatOutput.concat(depthItem);

}, []);

Console.log(flatArray);

Var newCourses = topics.reduce(function(courses, topic) {

Return courses.concat(topic.courses);

}, []);

Console.log(newCourses);

VD:

Cho trước danh sách một số bộ phim, hãy viết hàm calculateRating để tính điểm trung bình IMDB của những bộ phim mà Christopher Nolan làm đạo diễn.

### Gợi ý

* Dùng phương thức filter để lấy ra những bộ phim do Christopher Nolan làm đạo diễn
* Dùng phương thức reduce để tính tổng điểm IMDB
* Tính điểm IMDB trung bình
* var watchList = [
* {
* "Title": "Inception",
* "Year": "2010",
* "Rated": "PG-13",
* "Released": "16 Jul 2010",
* "Runtime": "148 min",
* "Genre": "Action, Adventure, Crime",
* "Director": "Christopher Nolan",
* "Writer": "Christopher Nolan",
* "Actors": "Leonardo DiCaprio, Joseph Gordon-Levitt, Elliot Page, Tom Hardy",
* "Plot": "A thief, who steals corporate secrets through use of dream-sharing technology, is given the inverse task of planting an idea into the mind of a CEO.",
* "Language": "English, Japanese, French",
* "Country": "USA, UK",
* "imdbRating": "8.8",
* "imdbVotes": "1,446,708",
* "imdbID": "tt1375666",
* "Type": "movie",
* },
* {
* "Title": "Interstellar",
* "Year": "2014",
* "Rated": "PG-13",
* "Released": "07 Nov 2014",
* "Runtime": "169 min",
* "Genre": "Adventure, Drama, Sci-Fi",
* "Director": "Christopher Nolan",
* "Writer": "Jonathan Nolan, Christopher Nolan",
* "Actors": "Ellen Burstyn, Matthew McConaughey, Mackenzie Foy, John Lithgow",
* "Plot": "A team of explorers travel through a wormhole in space in an attempt to ensure humanity's survival.",
* "Language": "English",
* "Country": "USA, UK",
* "imdbRating": "8.6",
* "imdbVotes": "910,366",
* "imdbID": "tt0816692",
* "Type": "movie",
* },
* {
* "Title": "The Dark Knight",
* "Year": "2008",
* "Rated": "PG-13",
* "Released": "18 Jul 2008",
* "Runtime": "152 min",
* "Genre": "Action, Adventure, Crime",
* "Director": "Christopher Nolan",
* "Writer": "Jonathan Nolan (screenplay), Christopher Nolan (screenplay), Christopher Nolan (story), David S. Goyer (story), Bob Kane (characters)",
* "Actors": "Christian Bale, Heath Ledger, Aaron Eckhart, Michael Caine",
* "Plot": "When the menace known as the Joker wreaks havoc and chaos on the people of Gotham, the caped crusader must come to terms with one of the greatest psychological tests of his ability to fight injustice.",
* "Language": "English, Mandarin",
* "Country": "USA, UK",
* "imdbRating": "9.0",
* "imdbVotes": "1,652,832",
* "imdbID": "tt0468569",
* "Type": "movie",
* },
* {
* "Title": "Batman Begins",
* "Year": "2005",
* "Rated": "PG-13",
* "Released": "15 Jun 2005",
* "Runtime": "140 min",
* "Genre": "Action, Adventure",
* "Director": "Christopher Nolan",
* "Writer": "Bob Kane (characters), David S. Goyer (story), Christopher Nolan (screenplay), David S. Goyer (screenplay)",
* "Actors": "Christian Bale, Michael Caine, Liam Neeson, Katie Holmes",
* "Plot": "After training with his mentor, Batman begins his fight to free crime-ridden Gotham City from the corruption that Scarecrow and the League of Shadows have cast upon it.",
* "Language": "English, Urdu, Mandarin",
* "Country": "USA, UK",
* "imdbRating": "8.3",
* "imdbVotes": "972,584",
* "imdbID": "tt0372784",
* "Type": "movie",
* },
* {
* "Title": "Avatar",
* "Year": "2009",
* "Rated": "PG-13",
* "Released": "18 Dec 2009",
* "Runtime": "162 min",
* "Genre": "Action, Adventure, Fantasy",
* "Director": "James Cameron",
* "Writer": "James Cameron",
* "Actors": "Sam Worthington, Zoe Saldana, Sigourney Weaver, Stephen Lang",
* "Plot": "A paraplegic marine dispatched to the moon Pandora on a unique mission becomes torn between following his orders and protecting the world he feels is his home.",
* "Language": "English, Spanish",
* "Country": "USA, UK",
* "imdbRating": "7.9",
* "imdbVotes": "876,575",
* "imdbID": "tt0499549",
* "Type": "movie",
* }
* ];
* function calculateRating(watchList) {
* var e = watchList.filter(function(e){
* return e.Director === 'Christopher Nolan'
* })
* console.log(e)
* var totalRating = e.reduce(function(acc, cur)
* {
* return acc + Number(cur.imdbRating)
* },0)
* return totalRating/e.length;
* }
* // Expected results
* console.log(calculateRating(watchList)); // Output: 8.675

**4. String/Array includes() method**

Var title = ‘Responsive web design’;

Console.log(title.includes(‘web’));

**X. Callback**

**1. Callback?**

- Là hàm

- Được truyền qua đối số

- Được gọi lại (trong hàm nhận đối số)

Function myFunction(param) {

If (typeof param === ‘function’){

param(‘Test’);

}

};

Function myCallback(value) {

Console.log(‘Value: ’, value);

};

myFunction(myCallback);

var courses = [

‘Javascript’,

‘PHP’,

‘Ruby’

];

Courses.map(function(course) {

Console.log(course)

});

Array.prototype.map2 = function() {

Var output = [];

Var arrayLength = this.length;

For (var i=0; i < arrayLength; i++) {

Var result = callback(this[i],i);

output.push(result);

};

Return output;

};

Courses.map2();

VD:

Array.prototype.myMap = function(cb) {

    var output = [];

    var arrLength = this.length

    for(var i=0; i< arrLength; i++){

        output[i] = cb(this[i], i)

    }

    return output

}

// Expected results

// const numbers = [1, 2, 3];

// console.log(numbers.myMap(function (number) {

//     return number \* 2;

// })) // Output: [2, 4, 6]

// console.log(numbers.myMap(function (number, index) {

//     return number \* index;

// })) // Output: [0, 2, 6]

**2. Empty elements of array?**

Var courses = [

‘Javascript’,

‘PHP’

];

Courses.length = 10;

For (var index in courses) {

Console.log(courses[index]);

}

**3. My forEach() method**

Array.prototype.forEach2 = function(callback) {

For (var index in this) {

Console.log(‘index: ’, index)

}

}

**4. My filter() method**

Array.prototype.filter2 = function(callback) {

For (var index in this) {

If (this.hasOwnProperty(index)) {

Callback(this[index], index, this);

}

}

}

Var courses = [

{

Name: ‘Javascript’,

Coin: 680

},

{

Name: ‘PHP’,

Coin: 860

},

{

Name: ‘Ruby’,

Coin: 980

},

];

Var filterCourses = courses.filter2(function(course, index, array) {

Return course.coin > 700;

});

Console.log(filterCourses);

VD:

Hãy tạo ra phương thức myFilter hoạt động tương tự phương thức filter2 và filter nhé.

Array.prototype.myFilter = function(cb) {

    var arr = [];

    for (var index in this) {

        if (this.hasOwnProperty(index)) {

            var result = cb(this[index], index, this);

            if (result) {

                arr.push(this[index]);

            }

        }

    }

    return arr;

}

**5. My some() method**

Array.prototype.some2 = function(callback) {

For (var index in this) {

If (this.hasOwnProperty(index)) {

Callback(this[index], index, this);

If (Callback(this[index], index, this)) {

Return true;

}

}

}

Return false;

}

Var courses = [

{

Name: ‘Javascript’,

Coin: 680

isFinish: true,

},

{

Name: ‘PHP’,

Coin: 860,

isFinish: false,

},

{

Name: ‘Ruby’,

Coin: 980,

isFinish: false,

},

];

Var result = courses.some2(function(course, index, array){

Return course.isFinish;

});

Console.log(result);

VD:

Hãy tạo ra phương thức mySome hoạt động tương tự phương thức some2 và some

Array.prototype.mySome = function(cb) {

    for (var index in this) {

        if (this.hasOwnProperty(index)) {

            var result = cb(this[index], index, this)

            if(result) {

                return true;

            }

        }

    }

    return false;

}

**6. My every() method**

Array.prototype.every2 = function(callback) {

Var output = true;

For (var index in this) {

If (this.hasOwnProperty(index)) {

Var result = Callback(this[index], index, this);

If (!result) {

Output = false;

Break;

}

}

}

Return output;

}

Var courses = [

{

Name: ‘Javascript’,

Coin: 680

isFinish: true,

},

{

Name: ‘PHP’,

Coin: 860,

isFinish: false,

},

{

Name: ‘Ruby’,

Coin: 980,

isFinish: false,

},

];

Var result = courses.every2(function(course, index, array){

Return course.isFinish;

});

Console.log(result);

VD: Hãy tạo ra phương thức myEvery hoạt động tương tự phương thức every2 và every

Array.prototype.myEvery = function(cb) {

    var output = true;

    for (var index in this) {

        if(this.hasOwnProperty(index)) {

            var result = cb(this[index], index, this)

            if (!result) {

                output = false;

                break;

            }

        }

    }

    return output;

}

**XI. HTML DOM**

**1. HTML DOM là gì?**

- Doccument Object Model



The HTML DOM Tree of Object

HTML DOM có 3 thành phần:

a. Element (< >)

b. Attribute ( “ href”)

c. Text ( “ ”)

**2. HTML DOM vs DOM API**

**a. HTML DOM**  
Là quy chuẩn mô tả mô hình của các thành phần trong tài liệu HTML được đưa ra bởi W3C.

**b. DOM API**  
Là bộ API nằm trong Web API có mặt trên những môi trường hỗ trợ duyệt web - như trên trình duyệt. DOM API cung cấp các đối tượng và phương thức hỗ trợ truy xuất, chỉnh sửa các đối tượng / thành phần trong DOM.

**3. Document object**

Document.write(‘Hello world!’);

**4. Get element methods**

Element: ID, class, tag, CSS selector, HTML collection

a. ID

Var headingNode = Document.getElementById(‘heading’);

Console.log({

Element: headingNode

});

b. Class

document.getElementsByClassName(‘heading’);

c. Tag

document.getElementsByTagName(‘h1’);// select qua thẻ element

d. CSS selector

document.querySelector(‘.heading’);//

document.querySelectorAll(‘’);

e. HTML collection

document.forms[…];// đặt tên form k đúng quy chuẩn

document.forms.testForm;

document.anchors;

VD:

Cho trước file HTML, các bạn hãy sử dụng các phương thức truy vấn đến các element trong DOM được học ở bài trước để lấy ra các element sau:

1. productsListElement: thẻ div có class là products-list.
2. firstProductElement: thẻ div đầu tiên có class là product.
3. buttonElements: tất cả các thẻ button.

var productsListElement = document.querySelector('.products-list')

var firstProductElement = document.querySelector('.product')

var buttonElements = document.querySelectorAll('button')

var listItemNodes = document.querySelectorAll(‘.box-1 li’);// thẻ li trong class box-1

var boxNode = document.querySelector(‘.box-1’);

boxNode.getElementByTagName(‘li’);

**5. Attribute node & Text node**

***a. DOM attribute***

var headingElement = document.querySelector(‘h1’);

headingElement.title = ‘Heading’;

headingElement.id = ‘…’;

headingElement.className = ‘…’;

headingElement.href = ‘…’; href là 1 attribute của thẻ a.

headingElement.setAttribute(‘class’, ‘heading’);// Thêm 1 class heading vào DOM

console.log(headingElement.getAttribute(‘class’));// Lấy attribute trong element

VD:

1. Thêm thuộc tính title có giá trị "F8 - Học lập trình để đi làm" cho thẻ h1.

var headingElement = document.querySelector('h1');

headingElement.title = 'F8 - Học lập trình để đi làm';

1. Thêm thuộc tính data-title có giá trị "F8 - Học lập trình để đi làm" cho thẻ h1.

headingElement.setAttribute('data-title', 'F8 - Học lập trình để đi làm');

1. Thêm thuộc tính href có giá trị "https://fullstack.edu.vn/" cho thẻ a.

var headingElement1 = document.querySelector('a');

headingElement1.href = 'https://fullstack.edu.vn/';

1. Thêm thuộc tính target có giá trị "\_blank" cho thẻ a.

headingElement1.setAttribute('target','\_blank');

***b. InnerText vs textContent Property***

var headingElement = document.querySelector(‘.heading’);

console.log(headingElement.innerText);

console.log(headingElement.textContent);

headingElement.innerText = ‘New Heading’;

innerText;// Hiển thị những gì nhìn thấy trên Website

textContent;// Hiển thị tất cả elementNode và TextNode ẩn trong Website

VD: Các bạn hãy thay đổi nội dung của thẻ h1 thành F8 - Học lập trình để đi làm.

var headingElement = document.querySelector('h1');

headingElement.innerText = 'F8 - Học lập trình để đi làm'

***c. InnerHTML vs OuterHTML Property***

var boxElement = document.querySelector(‘.box’);

boxElement.innerHTML = ‘<h1>Heading</h1>’;

boxElement.innerHTML = ‘<h1 title=”Heading”>Heading</h1>’;

boxElement.innerHTML;// Lấy output tính từ thẻ h1

boxElement.outerHTML; // Lấy output tính từ thẻ div class

VD1: Hãy viết hàm render nhận vào 1 tham số là html, hàm render sẽ có nhiệm vụ chèn giá trị của html vào trong thẻ ul đã cho trước.

function render(html) {

    var boxHeading = document.querySelector('ul');

    boxHeading.innerHTML = html;

    return boxHeading;

}

VD2: Hãy viết hàm render có 1 tham số courses, hàm render sẽ thêm các item của mảng courses để tạo thành 1 danh sách các khóa học trên giao diện.

Ví dụ: Với mảng var courses = ['ReactJS', 'AngularJS', 'VueJS'] sẽ thu được kết quả:

* ReactJS
* AngularJS
* VueJS

Gợi ý:

* Sử dụng phương thức map kết hợp với join để tạo chuỗi HTML từ mảng courses.
* Gán chuỗi HTML vừa tạo vào thuộc innerHTML của thẻ ul giống bài trước nhé.

var courses = ['HTML & CSS', 'Javascript', 'PHP', 'Java']

function render(courses) {

    var course = courses.map(function(arr) {

        return `<li>${arr}</li>`;

    }).join('');

    var listCourses = document.querySelector('ul');

    listCourses.innerHTML = course;

    return listCourses;

}

**6. Node properties**

**7. DOM CSS**

// DOM Style

Var boxElement = document.querySelector(‘.box’);

Console.log(boxElement.style);

boxElement.style.width = ‘100px’;

boxElement.style.height = ‘200px’;

boxElement.style.backgroundColor = ‘red’;

Object.assign(boxElement.style, {

Width: ‘200px’,

Height: ‘100px’,

backgroundColor: ‘green’

});// Không thường dùng để CSS cho đối tượng

Console.log(boxElement.style.width);// Lấy ra thuộc tính không phải giá trị

VD: Hãy thay đổi màu nền của thẻ div có class là .red thành màu #f00 và màu chữ của thẻ p thành màu #f05123.

var result = document.querySelector('.red');

result.style.backgroundColor = '#f00';

var result1 = document.querySelector('p');

result1.style.color = '#f05123'

**8. ClassList property**

// add

// contains

// remove

// toggle

Var boxElement = document.querySelector(‘.box’);

boxElement.classList.value;// trả về chuỗi trong class box

boxElement.classList.add(‘red’);

boxElement.classList.contains(‘red’);// kiểm tra class red có tồn tại hay không?

boxElement.classList.remove(‘red’);// xóa class red

boxElement.classList.toggle(‘red’);// Phương thức toggle dùng để thêm class được chỉ định vào element nếu class đó chưa xuất hiện và xóa bỏ class đó nếu đã xuất hiện.

.replace(*searchValue, newValue*); //  Thay thế class đã có trên phần tử bằng class khác.

VD1:

* Thêm class title vào thẻ h1
* Thay thế class sub-title ở thẻ p bằng class content

var result = document.querySelector('h1');

result.classList.add('title');

var result1 = document.querySelector('p');

result1.classList.replace('sub-title','content');

VD2:

Cho trước file HTML có các thẻ div, các bạn hãy thêm class box vào các thẻ div này nhé.

Gợi ý: Bạn có thể sử dụng forEach lặp qua các phần tử div, nhưng trước hết các bạn phải lấy được danh sách các phần tử div ra nhé.

Note: Hãy sử dụng kiến thức học được ở bài ClassList Property, không được sử dụng thuộc tính className nhé.

var result = document.querySelectorAll('div');

result.forEach(function(boxElement) {

    return boxElement.classList.add('box');

});

**9. DOM events**

*a. Attribute events*

<h1 onclick=”console.log(Math.random())”>DOM events</h1>

<h1 onmouseout=”console.log(Math.random())”>DOM events</h1>

<h1 onclick=”console.log(this.innerText)”>DOM events</h1>

*b. Assign event using the element node*

var h1Element = document.querySelector(‘h1’);

h1Element.onclick = function() {

console.log(Math.random());

}

var h1Elements = document.querySelector(‘h1’);

for(var i=0; i<h1Elements.length;i++) {

h1Element[i].onclick = function(e) {

console.log(e.target);

}

}

h1Element.onclick = function(e) {

console.log(e.target);

};//MouseEvent

VD: Cho trước thẻ button, các bạn hãy viết code JS sao cho khi click vào button sẽ đổi màu chữ button sang màu #fff.

var buttonElement = document.querySelector('button');

buttonElement.onclick = function() {

    buttonElement.style.color = '#fff';

};

**10. PreventDefault and StopPropagation**

Var aElement = document.querySelectorAll(‘a’);// document.links

For(var i=0; i<aElement.length; i++) {

aElement[i].onclick = function(e) {

if (!e.target.href.startsWith(‘https://fullstack.edu.vn’)) {

e.preventDefault();

}

}

}

e.stopPropagation();// ngăn chặn nổi bọt

**11. Event listener**

Btn.addEvenListener(‘click’, function(e) {

Console.log();

});

**XII. JSON, Fetch, Postman**

**1. JSON là gì?**

- Là 1 định dạng dữ liệu (chuỗi).

- JavaScript Object Notation

- JSON: Number, String, Boolean, Null, Array, Object

- Mã hóa / Giải mã (Encode / Decode)

- Stringify: Từ Javascript types => JSON

- Parse: Từ JSON => Javascript types

Var json = ‘…’;// 1, “abc”, true, null, [“…”,”…”], {“name”:”Cud”,”age”:”23”}.

Var a = ‘…’;

Console.log(JSON.parse(a));

Console.log(JSON.stringify());// chuyển sang string type

**2. Promise (sync, async)**

- Sync: Đồng bộ

- Async: Bất đồng bộ (setTimeout, setInteral, fetch, XMLHttpRequest, file reading javascript, request animation frame) => Callback

=> Sử dụng Callback để xử lý các thao tác bất đồng bộ async.

VD:

setTimeout(function() {

    console.log('Dòng này sẽ in ra sau')

}, 1000)

// setTimeout là tác vụ bất động bộ (async)

console.log('Dòng này sẽ in ra trước') // Đây là tác vụ đồng bộ (sync)

*a. Promise (pain)*

// Callback hell



// Pyramid of DOM

*b. Promise (concept)*

var promise = new Promise(

// Excutor

function (resolve, reject) {

// Logic

// Thành công: resolve()

// Thất bại: reject()

}

);

Promise

.then(function () {// Thành công})

.catch(function () {// Thất bại})

.finally(function () {// Done})

Promise có 3 trạng thái:

* Pending
* Fulfilled
* Rejected

*c. Promise (chain)*

var promise = new Promise(

// Excutor

function (resolve, reject) {

resolve();

}

);

Promise

.then(function () {

Return 1;

});

.then(function (data) {

Console.log(data);

Return 2;

});

.catch(function () {// Thất bại})

VD: Đây là ví dụ về callback hell trong truyền thuyết, và cách sử dụng Promise để giải quyết nó.

Các bạn có thể thấy khi tạo ra 1 đoạn code callback hell sẽ dẫn đến khó đọc code, thay vì viết như thế, chúng ta có thể áp dụng tính chất chuỗi (chain) của Promise để tạo ra 1 đoạn code dễ nhìn hơn mà vẫn đảm bảo đúng logic.

function hell(value, cb) {

    cb(value);

}

// Không sử dụng Promise dẫn đến tạo ra callback hell

hell(1, function (valueFromA) {

    hell(valueFromA + 1, function (valueFromB) {

        hell(valueFromB + 1, function (valueFromC) {

            hell(valueFromC + 1, function (valueFromD) {

                console.log(valueFromD + 1);

            });

        });

    });

});

// Sử dụng Promise sẽ tạo ra đoạn code dễ đọc hơn và vẫn đảm bảo đúng logic

function notHell(value) {

    return new Promise(function (resolve) {

        resolve(value);

    });

}

notHell(1)

    .then(function (value) {

        return value + 1;

    })

    .then(function (value) {

        return value + 1;

    })

    .then(function (value) {

        return value + 1;

    })

    .then(function (value) {

        console.log(value + 1);

    });

**3. Promise methods (resolve, reject, all)**

- Khi thực thi resolve gặp reject thì chương trình sẽ bị ngắt đoạn và thực hiện .catch và show ra lỗi.

// Promise.resolve

Var promise = new Promise (

Function (resolve, reject) {

Resolve(‘Success!’);

}

);// Hoặc Promise.resolve(‘Success!’);

Promise

.then(function (result) {

Console.log(‘Result: ’, result);

});

// Promise.reject

Var promise = new Promise (

Function (resolve, reject) {

Reject(‘Error!’);

}

);// Hoặc Promise.reject(‘Error!’)

Promise

.catch(function (err) {

Console.log(‘Err: ’, err);

});

// Promise.all

Var promise1 = new Promise (

Function (resolve, reject) {

setTimeout (function () {

resolve([1]);

}, 2000);

}

);

Var promise2 = new Promise (

Function (resolve, reject) {

setTimeout (function () {

resolve([2, 3]);

}, 5000);

}

);

Promise.all([promise1, promise2])

.then(function (result) {

Var result1 = result[0];

Var result1 = result[1];

Console.log(result1.concat(result2));

});

var users = [

{

id: 1,

name: 'Kien Dam'

},

{

id: 2,

name: 'Son Dang'

},

{

id: 3,

name: 'Hung Dam'

}

];

var comments = [

{

id: 1,

user\_id: 1,

content: 'Anh son chua ra video :('

},

{

id: 2,

user\_id: 2,

content: 'Vua ra xong em oi'

},

];

function getComments() {

return new Promise(function(resolve) {

setTimeout(function() {

resolve(comments)

}, 1000);

});

};

function getUsersByIds(userIds) {

return new Promise(function(resolve) {

var result = users.filter(function(user) {

return userIds.includes(user.id)

});

setTimeout(function(){

resolve(result)

}, 1000);

});

};

getComments()

.then(function(comments){

var userIds = comments.map(function(comment) {

return comment.user\_id;

});

return getUsersByIds(userIds)

.then(function(users) {

return {

users: users,

comments: comments,

};

});

});

.then(function(data) {

var commentBlock = document.getElementById('comment-block');  
 var html = '';  
 data.comments.forEach(function(comment) {  
 var user = data.users.find(function(user) {  
 return user.id === comment.user\_id;  
 });  
 html += `<li>${user.name}: ${comment.content}</li>`;  
 });  
 commentBlock.innerHTML = html;  
  
 });

**4. Fetch**

// API => Application programing interface

// Cổng giao tiếp giữa các phần mềm

// Backend => API => Fetch => JSON/XML

// => JSON.parse => Javascript types => Render ra giao diện với HTML

Var postApi = ‘https://jsonplaceholder.typicode.com/posts’;

Fetch(postApi)

.then(function (response) {

Return response.json();

// JSON.parse: JSON => Javascript types

});

.then(function (posts) {

Console.log(posts);

});

fetch('https://jsonplaceholder.typicode.com/todos/1')

.then(response => response.json())

.then(json => console.log(json))

{

"userId": 1,

"id": 1,

"title": "delectus aut autem",

"completed": false

}

**5. JSON server**

// Fake API server

<https://github.com/typicode/json-server>

**6. Sử dụng Postman làm việc với REST API**

- CRUD:

- Create: Tạo mới => POST

- Read: Lấy dữ liệu => GET

- Uppdate: Chỉnh sửa => PUSH

- Delete: Xóa => DELETE

**XIII. ECMAScript 6+**

**1. ECMAScript 6 là gì?**

// Let, const

// Template Literals

// Multi-line String

// Arrow function

// Classes

// Default parameter values

// Destructuring

// Rest parameters

// Spread

// Enhanced object literals

// Tagged template literal

// Modules

**2. Let & Const**

// Var || Let, Const: Khác nhau phạm vi truy cập (Scope), Hosting

// Const || Var, Let: Assigment

// Code block: if else, loop, {}, …

{

Var course = ‘Javascript basic!’;

}

Console.log(course);// Truy cập được và trả về dữ liệu

{

let course = ‘Javascript basic!’;

}

Console.log(course);// Không truy cập được và biến course => Lỗi

{

const course = ‘Javascript basic!’;

}

Console.log(course);// Không truy cập được và biến course => Lỗi

=> Let & Const truy cập được khi ở bên trong 1 block {…}, và ưu tiên trong phạm vi block gần nhất.

- Hosting: Đưa lên trên.

a=1;

var a;

console.log(a);// Đưa ra giá trị a

// Khi khai báo giá trị rồi khai báo biến với kiểu let và const thì sẽ xảy ra lỗi khi biên dịch

// Code thuần: var

// Babel: Const, Let

// - Khi định nghĩa biến và không gán lại biến đó: Const

// - Khi cần gán lại giá trị cho biến: Let

**3. Template literals**

Const courseName = ‘JavaScript’;

Const description = ‘Course name: ’ + courseName;

// const description = `Course name: ${courseName}`;

Console.log(description);

**4. Arrow function**

Function logger(log) {

Console.log(log);

}

logger(‘Message…’);

⬄

const logger= (log) => {

Console.log(log);

}

logger(‘Message…’);

const sum = (a, b) => {

return a+b;

}

Console.log(sum(2, 2));

const sum = (a, b) => a+b;

Console.log(sum(2, 2));

Const sum = (a, b) => ({a:a, b:b});

Const logger = log => console.log(log);

logger(‘Message…’);

const obj = {

name: ‘Javascript basic!’,

getName: function() {

return this.name;

}

};

Console.log(course.getName());

Const Course = function(name, price) {

this.name = name;

this.price = price;

}

Const jsCourse = new Course(‘Javascript’, 1000);

Console.log(jsCourse);

**5. Classes**

Const Course = function(name, price) {

this.name = name;

this.price = price;

}

Const phpCourse = new Course(‘PHP’, 1000);

Const jsCourse = new Course(‘Javascript’, 1200);

Console.log(phpCourse);

Console.log(jsCourse);

Class Course {

Constructor(name, price) {

This.name = name;

This.price = price;

getName() {

return this.name;

}

getPrice() {

return this.price;

}

Run() {

Const isSuccess = false;

If (…) {

isSuccess = true;

}

}

}

};// Thay thế cụm function

VD: Tạo class Person có thuộc tính name và age

class Person {

    constructor (name, age) {

        this.name = name;

        this.age = age;

    }

}

// Expected results

const person = new Person('Long', 22);

console.log(`Tên: ${person.name}, tuổi: ${person.age}`);

// Output: 'Tên: Long, tuổi: 22'

**6. Default parameter values**

Function logger(log) {

If (typeof log === ‘underfined’) {

log = ‘Gia tri mac dinh!’;

}

Console.log(log);

}

logger();

Function logger(log = ‘Gia tri mac dinh!’) {

Console.log(log);

}

logger();

**7. Enhanced object literals**

// Định nghĩa key: value cho object

// Định nghĩa method cho object

// Định nghĩa key cho object dưới dạng biến

Var name = ’Javascript’;

Var price = 1000;

Var course = {

Name: name,

Price: price

};

Console.log(course);

Var name = ’Javascript’;

Var price = 1000;

Var course = {

name,

price,

getName: function () {

return name;

}

};

Console.log(course.getName());

Var filedName = ‘name’;

Var filedPrice = ‘price’;

Const course = {

[filedName]: ‘Javascript’,

[filedPrice]: 1000

};

Console.log(course);

**8. Destructuring, Rest**

// Phân rã, sử dụng với array và object

a. Array

var array = [‘JS’, ‘PHP’, ‘Ruby’];

var [a, b, c] = array;

console.log(a, b, c);

var [a, …rest] = array;// Toán tử còn lại, lấy ra phần tử còn lại

console.log(rest);// Kết quả: PHP, Ruby

b. Object

var course = {

name: ‘Javascript’,

price: 1000

};

var { name, price} = course;

console.log(name, price);

var course = {

name: ‘Javascript’,

price: 1000

};

var { name, …rest} = course;

console.log(rest);

var { name, …newObject} = course;

console.log(rest);// Thủ thuật xóa 1 key mà không dùng delete.

var course = {

name: ‘Javascript’,

price: 1000

children: {

name: ‘ReactJS’

}

};

var { name: parentName, children: { name} } = course;

console.log(rest);// name: parentName: đổi tên biến name => parentName

function logger(a, …params) {

};//Toán tử Rest

logger(1,2,3,4);// 2,3,4

**9. Spread**

var array = [‘JS’, ‘PHP’, ‘Ruby’];

var array2 = [‘ReactJS’, ‘Dart’];

var array3 = […array2, …array1];// [“ReactJS”, “Dart”, “JS”, “PHP”, “Ruby”]

// Sử dụng nối mảng

**10. Tagged template literals**

Function highLight ([first,…strings], …values) {

Return values.reduce(

(acc, curr) => […acc, `<span>${curr}</span>`, strings.shift()],

[first]

).join(‘’);

};

Var brand = ‘F8’;

Var course = ‘Javascript’;

Const html = highLight `Học lập trình ${course} tại ${brand}`;

**11. Modules**

// Import

// Export

export default logger => export default, mỗi file js chỉ export default duy nhất 1 cái. export type => export thường, có thể export nhiều cái trong file

* Để import default: import [tên\_đối\_tượng\_cần\_import] from [địa\_chỉ\_file\_js]
* Để import các export thường: Ta sử dụng destructuring: `import { [tên\_đối\_tượng\_cần\_import] } from [địa\_chỉ\_file\_js].
* Đối với import dạng thường, ta có thể import nhiều đối tượng. `import {typelog, typewarn, typeerror} from './constants.js

***a.Module***

* Một module có thể hiểu là một file, trong đó sẽ chứa những đoạn code đảm nhiệm một chức năng / nghiệp vụ cụ thể nào đó.

***b.Tại sao lại sử dụng Module***

* Để code dễ đọc hơn, và dễ bảo trì, thay đổi hơn, chúng ta cần chia chúng thành các thành phần nhỏ hơn.

***c.Làm sao để sử dụng module***

* Để sử dụng module, ta cần set attribute type = module cho thẻ script

<script type="module" src="./main.js"></script>

***d.Export***

* Để xuất ra cho module khác để sử dụng.
* Lưu ý: **Một module chỉ có duy nhất 1 export default**

// Example File demo.js

const TYPE\_LOG = 'log'

const TYPE\_WARN = 'warn'

const TYPE\_ERROR = 'error'

const logger = 'logger'

export { TYPE\_LOG, TYPE\_WARN, TYPE\_ERROR }

export default logger

***e.Import***

* Để nạp module ta sử dụng từ khoá import.
* Để nạp **export default** của một module **ta import bình thường**.

// Import từ export default< của file logger.js

import logger from './logger.js'

* Để nạp các **export thường** (không phải export default) **ta sử dụng thuật toán destructuring**.

// Import các export của file type\_logger.js

import { TYPE\_LOG, TYPE\_WARN, TYPE\_ERROR } from './demo.js'

* Để nạp tất cả các export ta sử dụng import \* as [Tên\_Muốn\_Thay\_Thế] from ' ... '. **Trả về 1 object dạng module chứa các export**.

import \* as constants from './demo.js'

console.log( constants.TYPE\_LOG ) // log

console.log( constants.TYPE\_WARN ) // error

console.log( constants.default ) // logger

// File index.js

// Viết chi tiết

import logger from './logger.js'

export default logger

// ===== ===== //

// Viết rút gọn

export { default } from './logger.js'

// Import

import logger from './index.js'

**12. Optional chaining (?.)**

**XIV. Các bài thực hành**

**1. Xây dựng Tabs UI**

***a. HTML***

<link rel="preconnect" href="https://fonts.gstatic.com">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.2/css/all.min.css" integrity="sha512-HK5fgLBL+xu6dm/Ii3z4xhlSUyZgTT9tuc/hSrtw6uzJOvgRr2a9jyxxT1ely+B+xFAmJKVSTbpM/CuL7qxO8w==" crossorigin="anonymous" />

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600&display=swap" rel="stylesheet">

<div>

<!-- Tab items -->

<div class="tabs">

<div class="tab-item active">

<i class="tab-icon fas fa-code"></i>

React

</div>

<div class="tab-item">

<i class="tab-icon fas fa-cog"></i>

Angular

</div>

<div class="tab-item">

<i class="tab-icon fas fa-plus-circle"></i>

Ember

</div>

<div class="tab-item">

<i class="tab-icon fas fa-pen-nib"></i>

Vue.JS

</div>

<div class="line"></div>

</div>

<!-- Tab content -->

<div class="tab-content">

<div class="tab-pane active">

<h2>React</h2>

<p>React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.</p>

</div>

<div class="tab-pane">

<h2>Angular</h2>

<p>Angular is an application design framework and development platform for creating efficient and sophisticated single-page apps.</p>

</div>

<div class="tab-pane">

<h2>Ember</h2>

<p>Ember.js is a productive, battle-tested JavaScript framework for building modern web applications. It includes everything you need to build rich UIs that work on any device.</p>

</div>

<div class="tab-pane">

<h2>Vue.js</h2>

<p>Vue (pronounced /vjuː/, like view) is a progressive framework for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable. </p>

</div>

</div>

</div>

***b. CSS***

\* {

padding: 0;

margin: 0;

box-sizing: border-box;

}

body {

display: flex;

font-family: "Poppins", sans-serif;

}

body > div {

margin: 5% auto 0;

max-width: 720px;

}

.tabs {

display: flex;

position: relative;

}

.tabs .line {

position: absolute;

left: 0;

bottom: 0;

width: 0;

height: 6px;

border-radius: 15px;

background-color: #c23564;

transition: all 0.2s ease;

}

.tab-item {

min-width: 80px;

padding: 16px 20px 11px 20px;

font-size: 20px;

text-align: center;

color: #c23564;

background-color: #fff;

border-top-left-radius: 5px;

border-top-right-radius: 5px;

border-bottom: 5px solid transparent;

opacity: 0.6;

cursor: pointer;

transition: all 0.5s ease;

}

.tab-icon {

font-size: 24px;

width: 32px;

position: relative;

top: 2px;

}

.tab-item:hover {

opacity: 1;

background-color: rgba(194, 53, 100, 0.05);

border-color: rgba(194, 53, 100, 0.1);

}

.tab-item.active {

opacity: 1;

}

.tab-content {

padding: 28px 0;

}

.tab-pane {

color: #333;

display: none;

}

.tab-pane.active {

display: block;

}

.tab-pane h2 {

font-size: 24px;

margin-bottom: 8px;

}

***c. Javascript***

const $ = document.querySelector.bind(document);

const $$ = document.querySelectorAll.bind(document);

const tabs = $$(".tab-item");

const panes = $$(".tab-pane");

const tabActive = $(".tab-item.active");

const line = $(".tabs .line");

// SonDN fixed - Active size wrong size on first load.

// Original post: https://www.facebook.com/groups/649972919142215/?multi\_permalinks=1175881616551340

requestIdleCallback(function () {

line.style.left = tabActive.offsetLeft + "px";

line.style.width = tabActive.offsetWidth + "px";

});

tabs.forEach((tab, index) => {

const pane = panes[index];

tab.onclick = function () {

$(".tab-item.active").classList.remove("active");

$(".tab-pane.active").classList.remove("active");

line.style.left = this.offsetLeft + "px";

line.style.width = this.offsetWidth + "px";

this.classList.add("active");

pane.classList.add("active");

};

});

***2. Code Music Player***

- Render songs

- Scroll top

- Play/ Pause/ Seek

- CD rotate

- Next/ Prev

- Random

- Next/ Repeat when ended

- Active song

- Scroll active song into view

- Play song when click