



JAVASCRIPT SIÊU TỐC

FULL



MỞ ĐẦU

Phương pháp học tập



<http://Tuhoc.CC>

0

Welcome

❑ Các seri đã ra mắt trên <http://tuhoc.cc> :

1. *Csharp*: <http://csharp.tuhoc.cc>
2. *Android kotlin*: <http://kotlin.tuhoc.cc>
3. *Python* : <http://python.tuhoc.cc>
4. *Java* : <http://java.tuhoc.cc>
5. *C ++* : <http://c.tuhoc.cc>

6. *Lập trình web* : <http://web.tuhoc.cc>
7. *Javascript* : <http://js.tuhoc.cc>

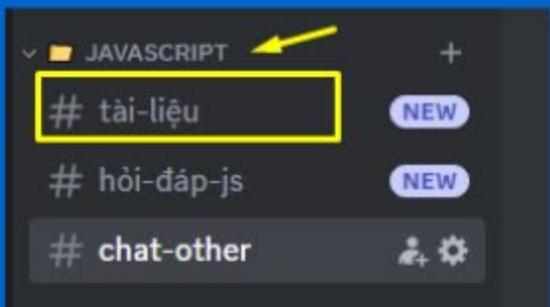


Front End

The screenshot shows the YouTube channel page for 'Gà Lại Lập Trình'. The banner features a megaphone, a rocket, and the text 'Video mới mỗi ngày!' and 'Tài liệu bài giảng tại http://dc.tuhoc.cc'. The channel name 'Gà Lại Lập Trình' is prominently displayed with a 'Creator' badge. Below the banner, there's a profile picture of a person wearing headphones and playing a game. A 'SUBSCRIBE' button with a bell icon is visible. The main content area shows a grid of video thumbnails, including titles like 'TUTORIAL TỰ HỌC JAVASCRIPT TỪ SỐ 0', 'Photoshop Beta AI 2023 - Hướng dẫn cách cài đặt Photoshop...', 'Tự học lập trình web từ số 0', 'ĐĂNG KÝ SỚM OPEN AI', 'LẬP TRÌNH C++ cơ bản 2023 | Tự học lập trình C++ siêu dễ hiểu...', and 'THÁT BẠI CỦA AI'. At the bottom, there are sections for 'Danh sách phát đã tạo' and 'Sắp xếp theo'.

TÀI LIỆU KHÓA HỌC

<http://dc.tuhoc.cc>



<http://js.tuhoc.cc>





LÀM GÌ
KHI GẶP KHÓ KHĂN
CODE KHÔNG HOẠT ĐỘNG

Tips

1. Dựa vào thông báo lỗi
2. So sánh với code mẫu
3. Check với ChatGPT
4. Trao đổi: <http://dc.tuhoc.cc>

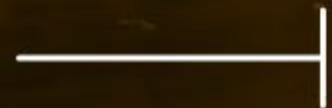


Phương pháp học tập

1. **Code theo, Cố gắng hiểu, không
học thuộc**
2. **Note lại thông tin**
3. **Note ngay dưới video**
4. **Ôn tập ngắt quãng
(Đường cong lãng quên)**



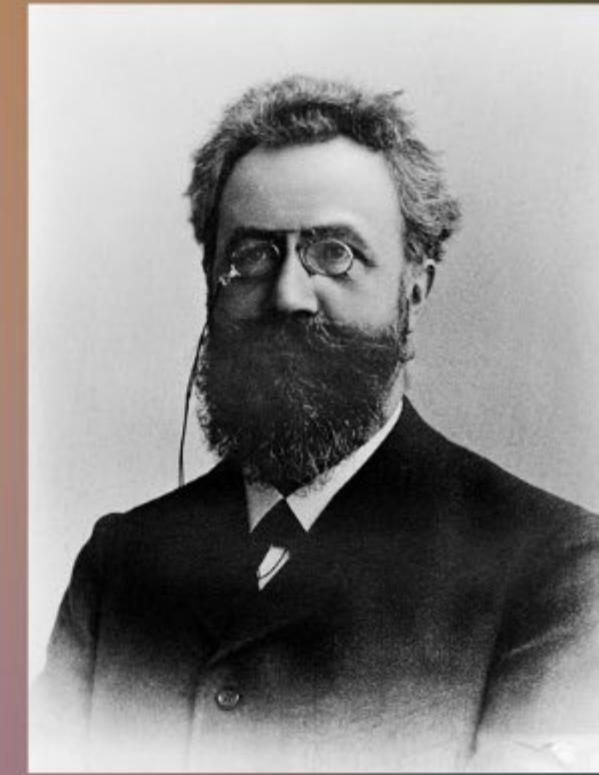
SUBSCRIBE



Cooking Tutorial

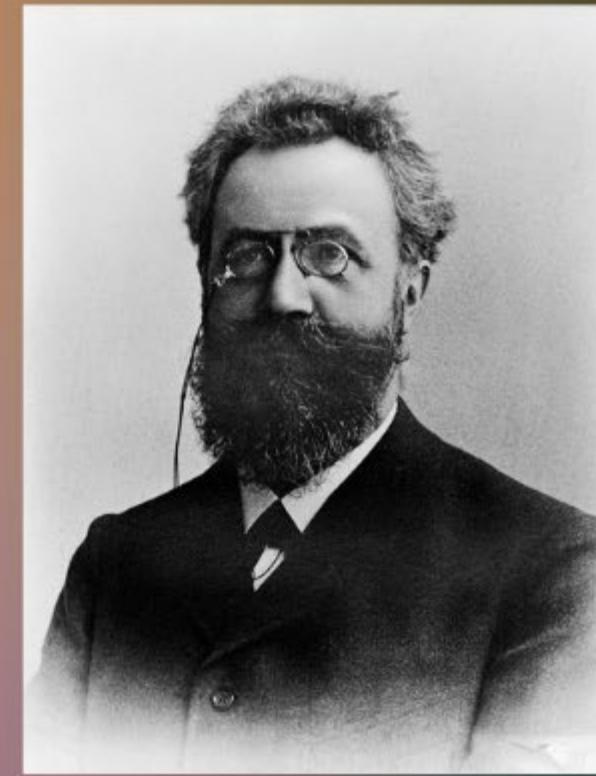
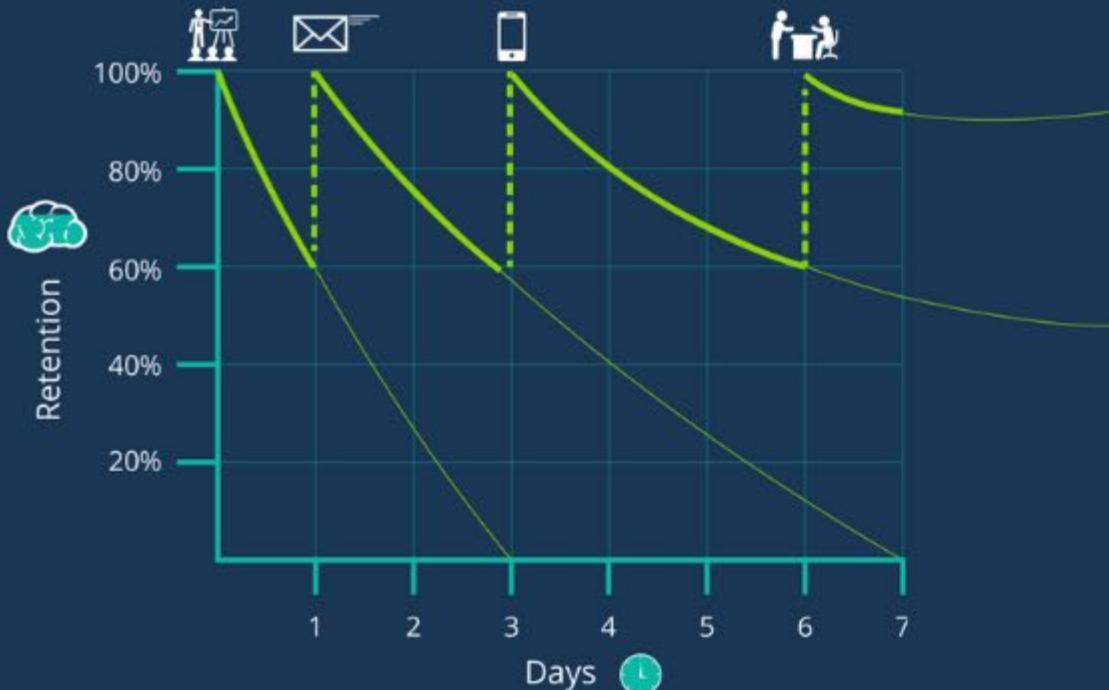


THE FORGETTING CURVE



Hermann Ebbinghaus
24 /1 /1850
26 /2 /1909)

COMBATING THE FORGETTING CURVE



Hermann Ebbinghaus
24 /1 /1850
26 /2 /1909)

LỜI KẾT

SUPPORT
FOR
US





JAVASCRIPT SIÊU TỐC

FULL



INTRODUCTION

Tổng quan Javascript



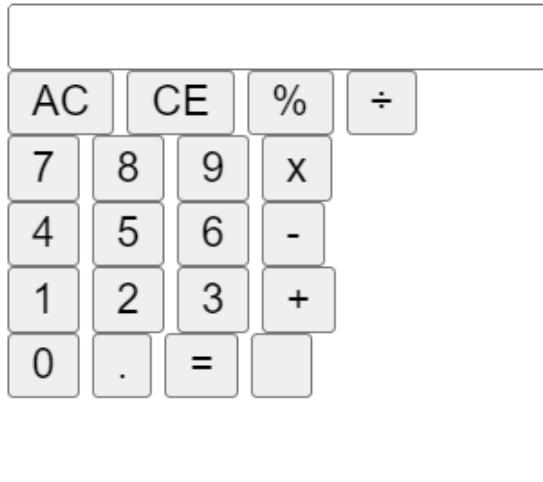
<http://Tuhoc.CC>

1

Tổng quan JS

<https://codepen.io/galailaptrinh/pen/dyqJwOE>

Tuhoc.CC Demo Calculator



HTML



HyperText Markup Language

Ngôn ngữ đánh dấu siêu văn bản

CSS



Cascading Style Sheets

Ngôn ngữ định dạng trang web (tô màu, in đậm, in nghiêng...)

JS



JavaScript

Các thao tác trên văn bản:Ẩn, hiện khi click, thay đổi nội dung hiển thị ...

<https://web.tuhoc.cc>

HTML



CSS



<https://js.tuhoc.cc>

JS



3 công nghệ phổ biến để lập trình web

Front End



JAVASCRIPT HISTORY

SUBSCRIBE NOW

2

Lịch sử JS

Google

js history w3



W3Schools

<https://www.w3schools.com> › js_history · Dic

[JavaScript History](#)

Google

ECMA wiki



Wikipedia

<https://vi.wikipedia.org> › wiki › Ecma_International

[Ecma International – Wikipedia tiếng Việt](#)

ECMAScript 1

ECMAScript 3

ECMAScript 5

1997

1998

1999

2009

2015

ECMAScript 2

ECMAScript 4

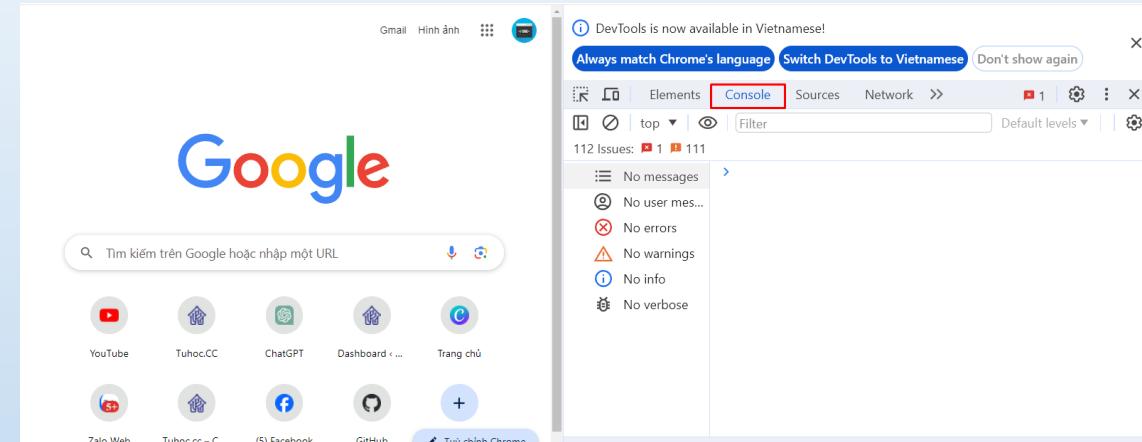
ECMAScript 6

3

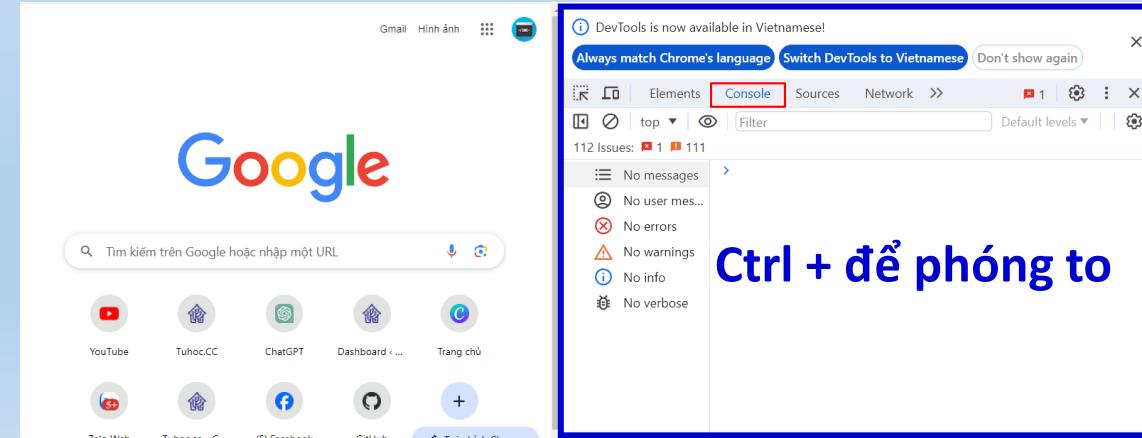
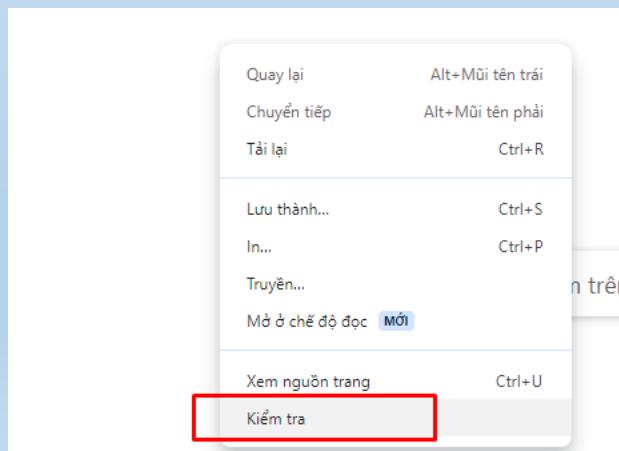
Hello world in JS



Chrome



Cách 1: Ctrl + Shift + J



Ctrl + để phóng to

Cách 2: Right mouse >> Kiểm tra



JAVASCRIPT SIÊU TỐC

FULL



3

CONSOLE.LOG - ALERT

Running JavaScript snippets



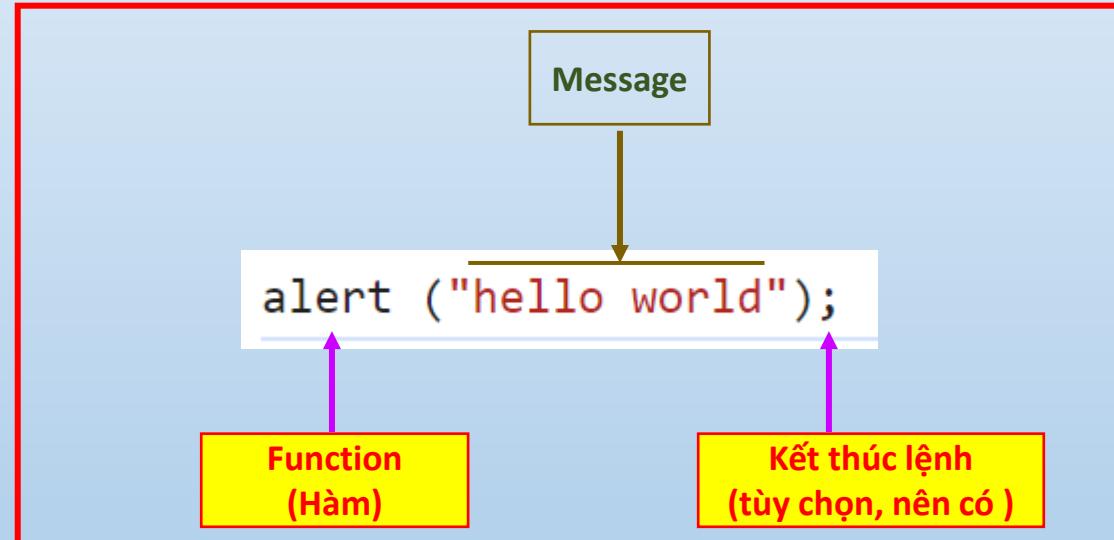
<http://Tuhoc.CC>

1

alert

❑ Hàm *alert()*

- ✓ Là một hàm trong JavaScript được sử dụng để hiển thị một thông báo dạng popup trên trình duyệt của người dùng. Cú pháp :



Bản chất :

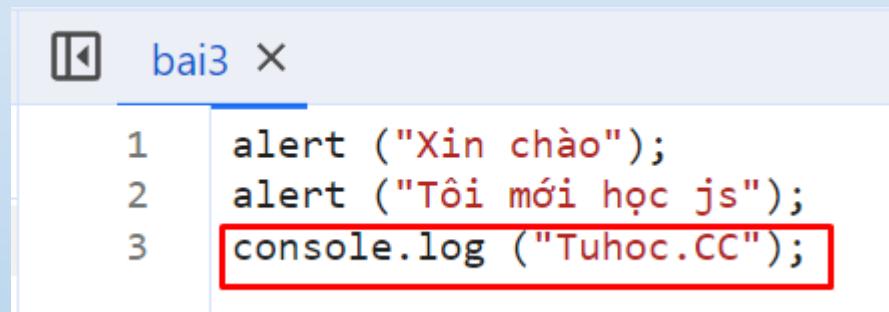
Khi dùng hàm *alert* chúng ta muốn thông báo cho trình duyệt biết rằng, ta muốn hiển thị 1 popup bật lên, với nội dung truyền vào trong “”

2

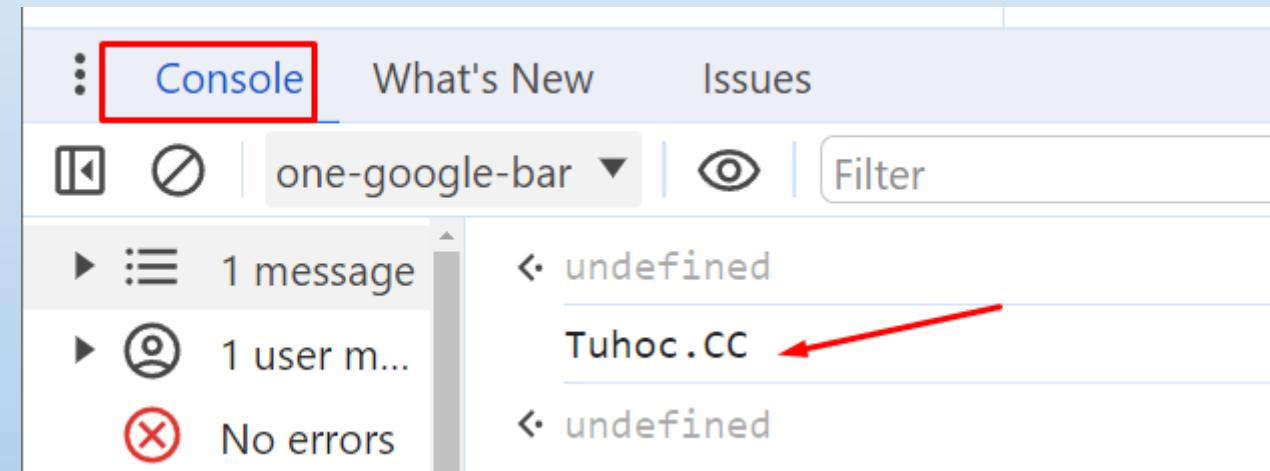
Console.log

 Hàm *console.log*

- ✓ Là một hàm được sử dụng để xuất thông tin ra màn hình điều khiển (console) của trình duyệt
- ✓ Sử dụng để hiển thị thông tin debug, giúp phát hiện và sửa lỗi trong mã nguồn.



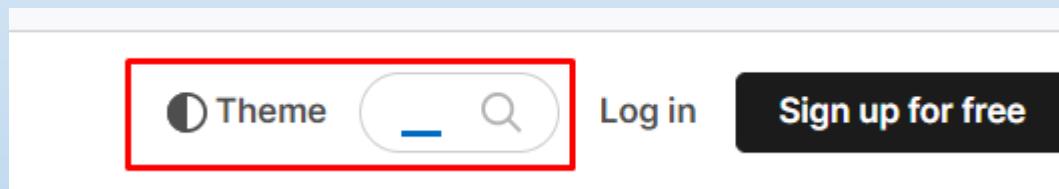
```
1 alert ("Xin chào");
2 alert ("Tôi mới học js");
3 console.log ("Tuhoc.CC");
```



3

Document

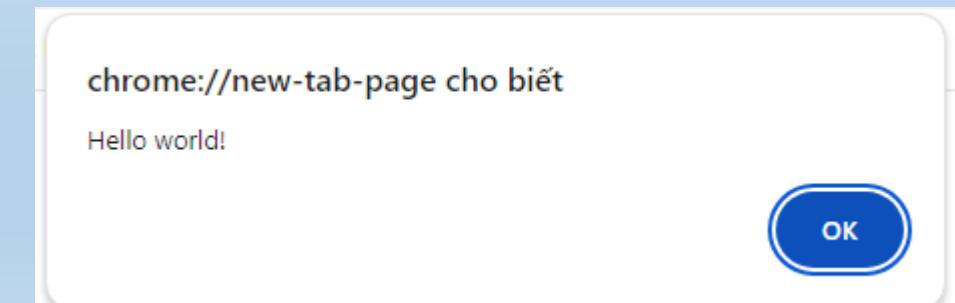
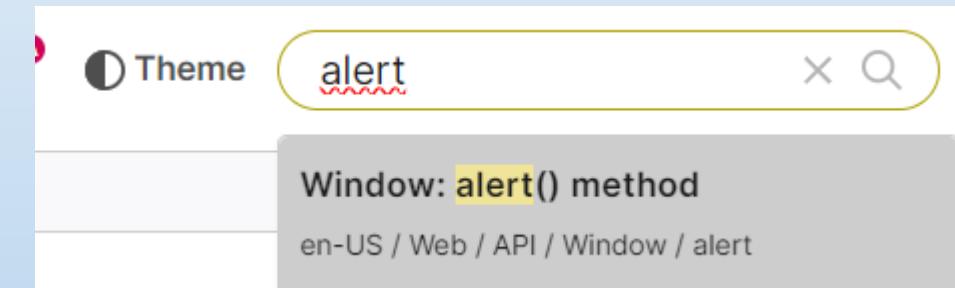
Google js mdn



```
JS

window.alert("Hello world!");
alert("Hello world!");
```

Cả 2 đều cho kết quả

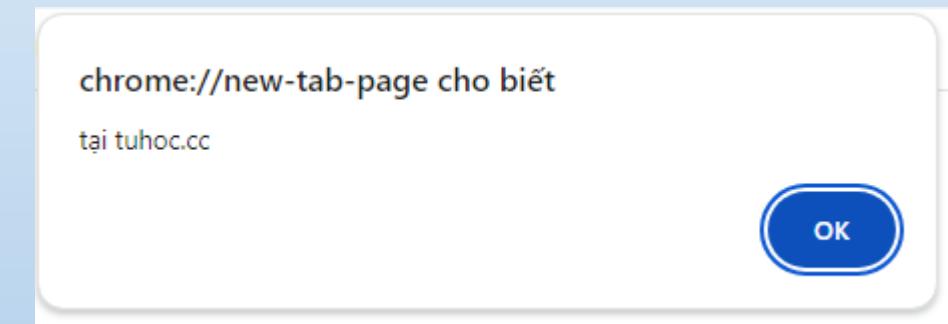
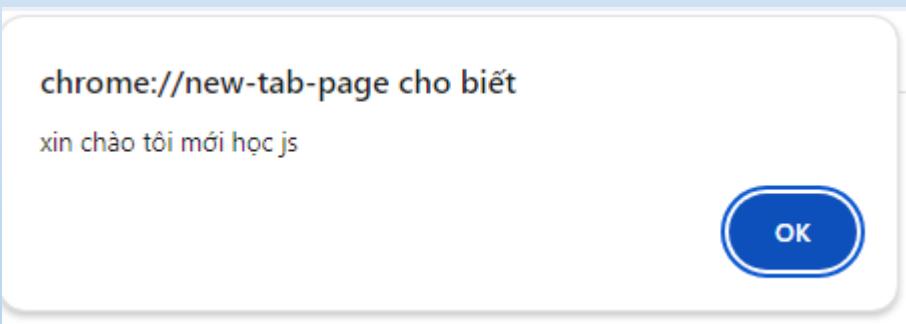


3

Multi line console

```
> alert("xin chào tôi mới học js");
  alert("tại tuhoc.cc");
```

Shift + Enter

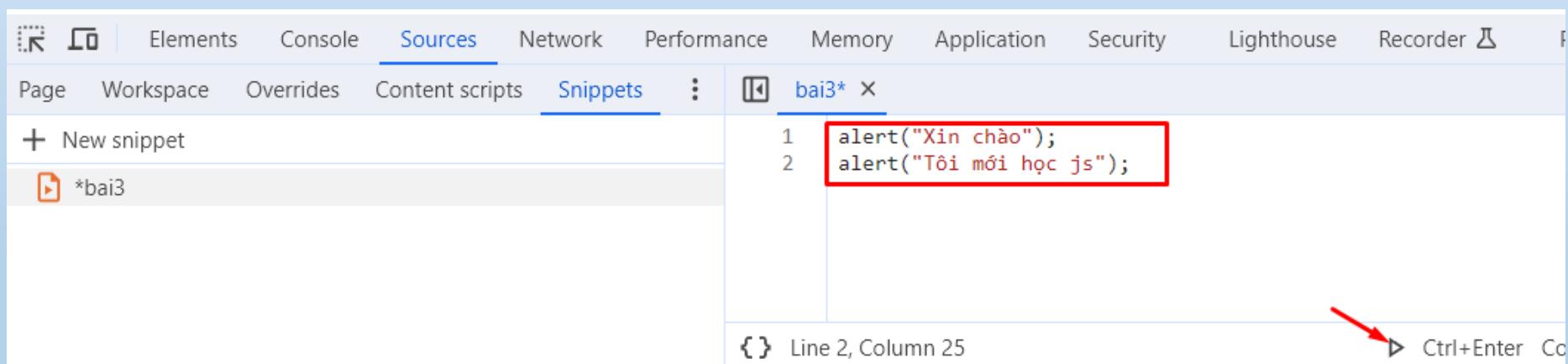
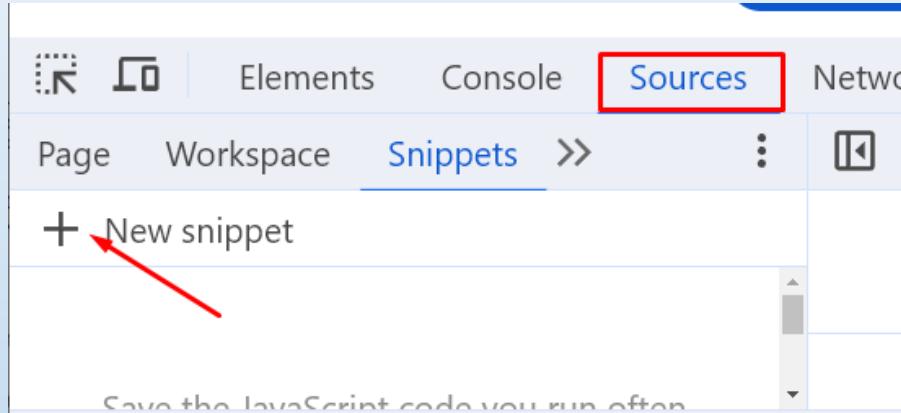


Hạn chế:

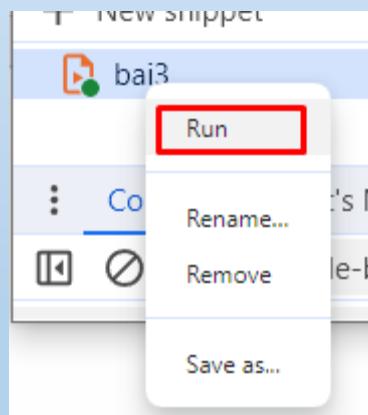
Nếu vô tình quên nhấn Shift, toàn bộ code sẽ được thực hiện, và phải gõ lại từ đầu

4

Running JavaScript snippets



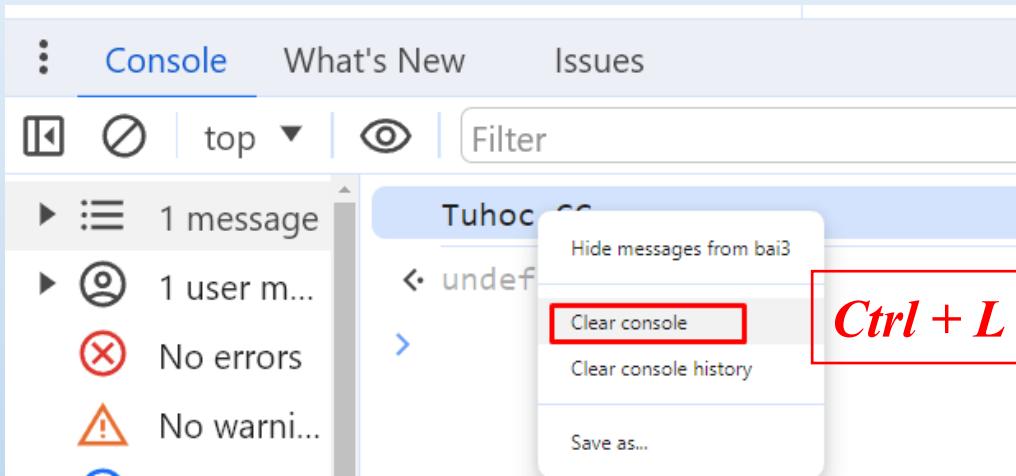
```
1 alert("Xin chào");
2 alert("Tôi mới học js");
```



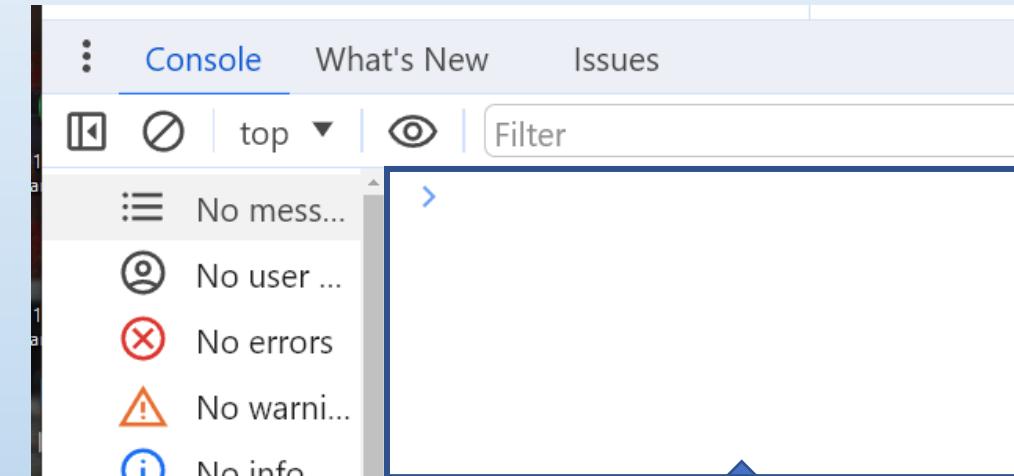
Sau khi soạn thảo xong có thể nhấn phím tắt Ctrl + Enter hoặc click run

5

Clear console



Ctrl + L



Kết quả log
console sau khi xóa



JAVASCRIPT SIÊU TỐC

FULL



4

THÊM MÃ JS VÀO HTML



<http://Tuhoc.CC>

1

Thêm JS vào HTML

```
<body>
    <!-- các thành phần khác của HTML -->
    <!-- Đoạn mã js đặt trước thẻ đóng thẻ body -->
    <script>
        // Mã JavaScript sẽ được thực thi ở đây
        alert("Xin chào, đây là một thông báo từ JavaScript!");
    </script>
</body>
```

Nhược điểm:

- Làm file HTML trở nên cồng kềnh

1

Thêm JS vào HTML

Ưu điểm:

- Code JS tách biệt với HMTL, dễ quản lý



JAVASCRIP SIÊU TỐC

FULL



5.1

JAVASCRIPT VARIABLE

Khai báo - Khởi tạo biến



<http://Tuhoc.CC>

1

Đặt vấn đề

2

Biến và Khai báo biến

2

Biến và Khai báo biến

2

Biến và Khai báo biến

```
// thử Thay đổi giá trị sau khai báo
myName = "supper man";
yourName = "Mai Siêu Phong";

// xuất giá trị của biến
console.log(myName);
console.log(yourName);

doSoi = 50; // báo lỗi, do không thể thay đổi giá trị của hằng
```

Trần Như Nhộn

Mãi Văn Dốt

100

supper man

Mai Siêu Phong

✖ ► Uncaught TypeError: Assignment to constant variable.
at main.js:31:7

3

Bài tập vận dụng 1



JAVASCRIP SIÊU TỐC

FULL



5.2

JAVASCRIPT VARIABLE

Quy tắc đặt tên biến



<http://Tuhoc.CC>

1

Bài tập vận dụng 1

1

Bài tập vận dụng 1

2

Quy tắc đặt tên biến



JAVASCRIPT SIÊU TỐC

FULL



PRIMITIVE DATA TYPES

Các kiểu dữ liệu nguyên thủy



2

Primitive Data Types trong JS

| No | Kiểu dữ liệu | Mô tả | Ví dụ |
|----|--------------|--|--|
| 1 | String | Kiểu chuỗi : Kiểu dữ liệu để lưu trữ chuỗi ký tự | let myString = "Xin chào"; |
| 2 | Number | Kiểu số : Kiểu dữ liệu để lưu trữ số, có thể là số nguyên hoặc số thực. | let soNguyen = 10; let soThuc = 3.14; |
| 3 | Boolean | Kiểu luận lý, chỉ có thể nhận giá trị true hoặc false | let.isTrue = true; let.isFalse = false; |
| 4 | Undefined | Biến chưa được gán giá trị sẽ có kiểu dữ liệu là undefined | let undefinedVariable; |
| 5 | Null | Biến được gán giá trị là null. Thường dùng để reset biến, gán giá trị mặc định... | let nullValue = null; |

```
> let myName;
  console.log(myName);
  undefined
```

```
console.log("Số nguyên an toàn tối đa:", Number.MAX_SAFE_INTEGER);
console.log("Số nguyên an toàn tối thiểu:", Number.MIN_SAFE_INTEGER);
```

Số nguyên an toàn tối đa: 9007199254740991

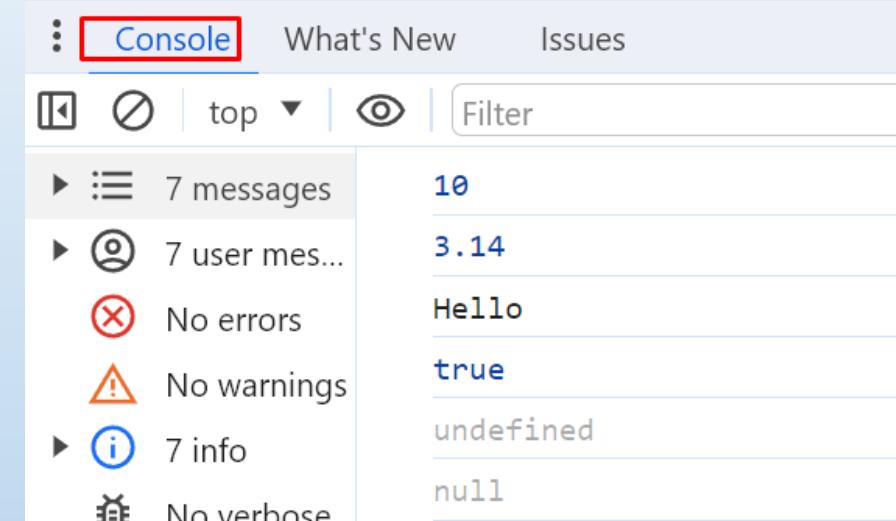
Giá trị từ -($2^{53} - 1$) to $2^{53} - 1$

Số nguyên an toàn tối thiểu: -9007199254740991

2

Primitive Data Types trong JS

```
1 // kiểu number
2 let soNguyen = 10;
3 let soThuc = 3.14;
4 // Kiểu String - chuỗi
5 let myString = "Hello";
6 // kiểu boolean
7 let check = true;
8 // Kiểu Undefined
9 let myName; //khai báo nhưng không gán giá trị
10 // Kiểu null
11 let score = null;
12
13 // Xuất các biến
14 console.log(soNguyen);
15 console.log(soThuc);
16 console.log(myString);
17 console.log(check);
18 console.log(myName);
19 console.log(score);
```



The screenshot shows the browser's developer tools Console tab. The tab bar has 'Console' highlighted with a red box. Below the tab bar, there are icons for refresh, top, filter, and a list of messages. The main area displays the following log entries:

| Message Type | Content |
|---------------|-----------|
| 7 messages | 10 |
| 7 user mes... | 3.14 |
| No errors | Hello |
| No warnings | true |
| 7 info | undefined |
| No verbose | null |

2

Toán tử `typeof`

```
// Kiểm tra kiểu loại của variable
console.log("Kiểm tra kiểu dữ liệu của biến: ");
console.log(typeof soNguyen);
console.log(typeof soThuc);
console.log(typeof myString);
console.log(typeof check);
console.log(typeof myName);
console.log(typeof score);
```

Kiểm tra kiểu dữ liệu của biến:
number
number
string
boolean
undefined
object

Google

why `typeof null` is object



JAVASCRIPT SIÊU TỐC

FULL



Xuất Dữ Liệu với Biến

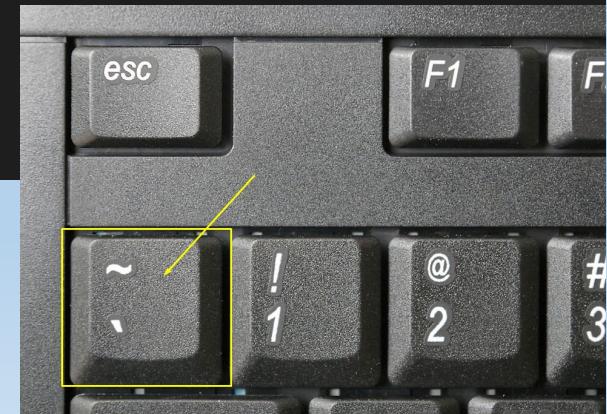


<http://Tuhoc.CC>

1

Các cách xuất dữ liệu với biến

```
// 7. các cách xuất dữ liệu với biến trong js
let soA = 25;
let soB = 5;
// Cách 1:
console.log("Căn bậc hai của " + soA + " là: " + soB);
// Cách 2:
console.log("Căn bậc hai của", soA, "là: ", soB);
// Cách 3:
console.log(`Căn bậc hai của ${soA} là: ${soB}`); // dùng phím ` dưới ESC
// Cách 4:
console.log(`Căn bậc hai của %s là %s:`, soA, soB);
```



2

Luyện tập

```
// luyện tập
let a = 4,
    b = 3,
    kq = a * b;
// xuất kết quả ra màn hình console theo 4 cách đã học
```

Kết quả: 4 nhân với 3 bằng 12

```
// Cách 1:
console.log("Kết quả: " + a + " nhân với " + b + " bằng " + kq);
// Cách 2:
console.log("Kết quả: ", a, " nhân với ", b, " bằng ", kq);
// Cách 3:
console.log(`Kết quả: ${a} nhân với ${b} bằng ${kq}`);
// Cách 4:
console.log(`Kết quả: %s nhân với %s bằng %s`, a, b, kq);
```



JAVASCRIPT SIÊU TỐC

FULL



7.2

Các phép toán cơ bản

+ - * / %



<http://Tuhoc.CC>

1

Các phép toán cơ bản

```
let a = 5;
let b = 2;

let tong = a + b; // tong = 7
let hieu = a - b; // hieu = 3
let tich = a * b; // tich = 10
let thuong = a / b; // thuong = 2.5
let soDu = a % b; // dư = 1

// xuất kết quả
console.log("tổng 2 số: " + tong);
console.log("hiệu 2 số: " + hieu);
console.log("tích 2 số: " + tich);
console.log("thương 2 số: " + thuong);
console.log("dư của phép chia a cho b : " + soDu);
```

* Giống với toán học thông thường, nhân chia trước, cộng trừ sau

* Đối với các phép toán phức tạp, dùng ngoặc() để thể hiện sự ưu tiên



JAVASCRIPT SIÊU TỐC

FULL



7.3

JS Operator Precedence

Quy tắc thứ tự ưu tiên



<http://Tuhoc.CC>

1

js operator precedence mdn



js operator precedence mdn

```
// Ưu tiên ()
let diemToan = 9.5;
let diemVan = 8.75;

let dtb = (diemToan + diemVan) / 2;
console.log(dtb);

// right to left
let x = (y = 25);
// left to right
let c = 25 - 7 + 8 - 1;
```



JAVASCRIPT SIÊU TỐC

FULL



ÉP KIỂU DỮ LIỆU



<http://Tuhoc.CC>

1

Tại sao cần ép kiểu

- Mặc định dữ liệu nhập vào sử dụng hàm **prompt()** có kiểu string
- Đôi khi ta cần thực hiện tính toán với dữ liệu nhập vào từ bàn phím của người dùng
→ **Cần ép kiểu trước khi thực hiện tính toán**

```
// xuất thông báo cho người dùng nhập vào số a
let numberA = prompt("Mời cụ nhập vào numberA: ");
// check thử kiểu loại biến numberA
console.log(typeof numberA);  string
// thử thực hiện tính toán ( nếu chưa ép kiểu )
let numberB = 5;
console.log(typeof numberB);
// cộng A với B
let kq = numberA + numberB;
console.log(`kết quả A + B = ${kq}`);
console.log(typeof kq);  string
// vd a = 8, kết quả 85 --> sai, js hiểu là cộng chuỗi
```

1

Tại sao cần ép kiểu

```
console.log(numberA + numberB);
// vd a = 8, kết quả 85 --> sai, js hiểu là cộng chuỗi
```

- trong JS, khi nhập dữ liệu từ bàn phím bằng hàm ***prompt()***, mặc định kiểu dữ liệu là ***string***
 1. **Đối với phép cộng → JS sẽ hiểu là cộng chuỗi**
 2. **Các phép tính khác → JS sẽ cố gắng chuyển đổi thành kiểu số trước khi thực hiện tính toán → Nếu không chuyển được sang số để tính toán, kết quả sẽ là *NaN - not a number* ☺**

```
console.log(numberA * numberB);
console.log(numberA / numberB);
console.log(numberA % numberB);
```

| |
|-----|
| 40 |
| 1.6 |
| 3 |

➔ Cần ép kiểu để xử lý tính toán

2

Ép kiểu dữ liệu

// ép sang kiểu nguyên

```
let numberC = parseInt(prompt("Mời cụ nhập numberC: "));  
// kiểm tra kiểu dữ liệu của c  
console.log("kiểu dữ liệu của numberC: " + typeof numberC);  
console.log(numberB + numberC);
```

// ép sang kiểu số thực Float

```
let numberD = parseFloat(prompt("Mời cụ nhập numberD: "));  
// kiểm tra kiểu dữ liệu của numberD  
console.log("kiểu dữ liệu của numberD: " + typeof numberD);  
console.log(numberB + numberD);
```

// Hoặc đơn giản dùng hàm Number() để ép --> Kiểu dữ liệu number

```
let numberE = Number(prompt("Mời cụ nhập numberE: "));  
// kiểm tra kiểu dữ liệu của numberE  
console.log("kiểu dữ liệu của numberE: " + typeof numberE);  
console.log(numberB + numberE);
```



JAVASCRIPT SIÊU TỐC

FULL



TOÁN TỬ GÁN

 $= + = - =$ $* = / = \% =$ <http://Tuhoc.CC>

1

Toán tử gán

| Phép tính | Giải thích | Công thức | Ý nghĩa |
|-----------|-----------------|-----------|--------------|
| = | Gán bằng | $x = 1$ | $x = 1$ |
| += | Gán cộng | $x += y$ | $x = x + y$ |
| -= | Gán trừ | $x -= y$ | $x = x - y$ |
| *= | Gán nhân | $x *= y$ | $x = x * y$ |
| /= | Gán chia | $x /= y$ | $x = x / y$ |
| %= | Gán chia lấy dư | $x \%= y$ | $x = x \% y$ |

1

Toán tử gán

```
// 1. Toán Tử Gán (=) Dùng để gán giá trị cho biến.  
let x = 10; // Biến x được gán giá trị 10  
let y = 5; // Biến y được gán giá trị 5  
let z = x + y; // Biến z được gán giá trị là tổng của x và y  
console.log(x);  
console.log(y);  
console.log(z);
```

1

Toán tử gán

| Phép tính | Giải thích | Công thức | Ý nghĩa |
|-----------|-----------------|-----------|--------------|
| = | Gán bằng | $x = 1$ | $x = 1$ |
| += | Gán cộng | $x += y$ | $x = x + y$ |
| -= | Gán trừ | $x -= y$ | $x = x - y$ |
| *= | Gán nhân | $x *= y$ | $x = x * y$ |
| /= | Gán chia | $x /= y$ | $x = x / y$ |
| %= | Gán chia lấy dư | $x \%= y$ | $x = x \% y$ |

```
// Gán +=, -=, *=, /=, %=
console.log("Kết quả gán +=, -=, *=, /=, %= ");
let m = 10;
m += 5; // Tương đương với m = m + 5;
console.log(m); // In ra 15

let n = 8;
n -= 3; // Tương đương với n = n - 3;
console.log(n); // In ra 5

let p = 6;
p *= 2; // Tương đương với p = p * 2;
console.log(p); // In ra 12

let q = 9;
q /= 3; // Tương đương với q = q / 3;
console.log(q); // In ra 3

let r = 4;
r %= 3; // Tương đương với r = r % 3;
console.log(r); // In ra 1
```

2

Bài tập JS 03

Bài tập 3 : Thực hiện các phép toán gán và in ra kết quả

let a = 7;

a . Thực hiện phép toán gán mở rộng để tăng giá trị của a lên 3 và in ra kết quả

Gợi ý: sử dụng toán tử +=

a += 3;

console.log("a =", a);

b) b = 15, b -= 6

*c) c = 5; c *= 4*

d) d = 12; d /= 2

e) e = 5, f= 2 rút gọn biểu thức e = e-(f+1)



JAVASCRIPT SIÊU TỐC

FULL



10

TOÁN TỬ TĂNG GIẢM

++

--

<http://Tuhoc.CC>

1

Toán tử tiền tố, hậu tố (prefix, postfix)

| Ký hiệu | Giải thích | Cách biểu đạt | Kết quả |
|-----------|--------------------|---------------|---------|
| ++ | Tăng giá trị lên 1 | a=1, a++ | a = 2 |
| -- | Giảm giá trị đi 1 | a= 1, a-- | a = 0 |

```
let a = 99;
let b = 10;
let c = 77;
let d = 88;
a++; // tương đương với a = a + 1; --> 100
b--; // tương đương với b = b - 1; --> 9
++c; // tương đương với c = c + 1; --> 78
--d; // tương đương với = a + 1; --> 87
console.log(a);
console.log(b);
console.log(c);
console.log(d);
```

*Phép tính đơn lẻ, viết ++, --
trước hay sau đều được*

1

Toán tử tiền tố, hậu tố (prefix, postfix)

Trường hợp biểu thức phức tạp thì tuân theo quy tắc sau:

□ Quy tắc viết dấu ++, --

a++, a-- (viết phía sau biến) => Postfix

++a, --a (viết trước biến) => Prefix

□ Ưu tiên tính toán Postfix, Prefix

Step 1 . Prefix

Step 2. Các phép toán còn lại

Step 3. Gán giá trị cho biến ở bên trái dấu bằng

Step 4. Tính postfix

```
let x = 1;
let y = 2;
let z = x++ - ++y + 1;
console.log(x); // 2
console.log(y); // 3
console.log(z); // -1
```

x = 1

y = 2

z = x++ - ++y + 1

Step 1: ++y => y = 3

Step 2: x=1, y = 3 => 1-3+1 = -1

Step 3: z = -1

Step 4: x++ => x = 2

int z = 2 - 3 + 1 = 0

Phép tính sai



JAVASCRIPT SIÊU TỐC

FULL



TOÁN TỬ LOGIC

Bài tập JS 04 - 06



<http://Tuhoc.CC>

1

Toán tử logic **&&** || !

| Ký hiệu | Giải thích | Cách biểu đạt | Kết quả |
|-------------------|--|--|------------------------|
| && | Toán tử logic AND (và) giữa 2 giá trị , trả về True khi cả 2 đều đúng (and nhiều giá trị tương tự) | x = True, y = True x= False , y = True x= True , y = False | True False False |
| | Toán tử logic OR (hoặc) giữa 2 giá trị , trả về False khi cả 2 đều sai (Or nhiều giá trị tương tự) | x = True, y = Fasle x = Fasle, y = True x = Fasle, y = False | True True False |
| ! | NOT - Phép phủ định (Đảo ngược giá trị) | X=True , !x X = Fasle , !x | False True |

```
let i = 7;
// kiểm tra xem i >0 và i < 10 không?
console.log(i > 0 && i < 10); //true
// kiểm tra i<0 hoặc i<10 không?
console.log(i > 0 || i < 10); //true
// phủ định
console.log(!(i > 0 || i < 10)); //false
```

2

Bài tập vận dụng 04 - 06

 Bài tập 04 : Tính chu vi, diện tích hình tròn

Viết chương trình nhập vào từ bàn phím bán kính r của đường tròn , in ra kết quả

a. Chu vi = ?

b. Diện tích = ?

Gợi ý :

$$\text{chu vi} = 2 * \text{PI} * r$$

$$\text{dientich} = \text{PI} * r * r$$

```
let PI = Math.PI;
```

Nhập bán kính của đường tròn:

5|

Chu vi = 31.41592653589793

Diện tích = 78.53981633974483

2

Bài tập vận dụng 04 - 06

 Bài tập 05: Tính chu vi, diện tích hình chữ nhật

1. *Viết chương trình nhập vào 2 số thực dương a, b từ bàn phím a, b là chiều dài và chiều rộng của hình chữ nhật.*
2. *In ra màn hình chu vi và diện tích của hình chữ nhật đó.*

Gợi ý :

$$\text{diện tích } s = a * b ,$$

$$\text{chu vi } p = (a+b) * 2$$

Nhập chiều dài của hình chữ nhật:

4

Nhập chiều rộng của hình chữ nhật:

3

Chu vi = 14

Diện tích = 12

2

Bài tập vận dụng 04 - 06

- Bài tập 06:** *Viết chương trình nhập vào điểm ba môn toán, văn, anh của 1 học sinh, tính điểm trung bình và xuất kết quả làm tròn 2 chữ số sau dấu phẩy*

```
// Xuất kết quả làm tròn 2 chữ số sau dấu phẩy  
dtb = 8.21456;  
dtb = dtb.toFixed(2);
```

Điểm trung bình = 8.21



JAVASCRIPT SIÊU TỐC

FULL



TOÁN TỬ LOGIC

Giải bài tập JS 04



<http://Tuhoc.CC>

1

Bài tập vận dụng 04

Bài tập 04 : Tính chu vi, diện tích hình tròn

Viết chương trình nhập vào từ bàn phím bán kính r của đường tròn , in ra kết quả

a. Chu vi = ?

b. Diện tích = ?

Gợi ý :

$$\text{chu vi} = 2 * \text{PI} * r$$

$$\text{dientich} = \text{PI} * r * r$$

```
let PI = Math.PI;
```

Nhập bán kính của đường tròn:

5|

Chu vi = 31.41592653589793

Diện tích = 78.53981633974483

1

Bài tập vận dụng 04

```
// Yêu cầu người dùng nhập bán kính từ bàn phím
let r = Number(prompt("Nhập bán kính của đường tròn:"));

// Tính chu vi và diện tích
let PI = Math.PI;
let chuVi = 2 * PI * r;
let dienTich = PI * r * r;

// In ra kết quả
console.log("Chu vi =", chuVi);
console.log("Diện tích =", dienTich);
```

```
let PI = Math.PI;
```

Nhập bán kính của đường tròn:

5|

Chu vi = 31.41592653589793
Diện tích = 78.53981633974483



JAVASCRIPT SIÊU TỐC

FULL



TOÁN TỬ LOGIC

Giải bài tập JS 05



2

Bài tập vận dụng 04 - 06

☐ Bài tập 05: Tính chu vi, diện tích hình chữ nhật

1. *Viết chương trình nhập vào 2 số thực dương a, b từ bàn phím a, b là chiều dài và chiều rộng của hình chữ nhật.*
2. *In ra màn hình chu vi và diện tích của hình chữ nhật đó.*

Gợi ý:

$$\text{diện tích } s = a * b, \\ \text{chu vi } p = (a + b) * 2$$

Nhập chiều dài của hình chữ nhật:
4

Nhập chiều rộng của hình chữ nhật:
3

Chu vi = 14

Diện tích = 12

```
// Yêu cầu người dùng nhập chiều dài và chiều rộng từ bàn phím
let a = Number(prompt("Nhập chiều dài của hình chữ nhật:"));
let b = Number(prompt("Nhập chiều rộng của hình chữ nhật:"));

// Tính chu vi và diện tích
let chuVi = (a + b) * 2;
let dienTich = a * b;

// In ra kết quả
console.log("Chu vi =", chuVi);
console.log("Diện tích =", dienTich);
```



TOÁN TỬ LOGIC

Giải bài tập JS 06



2

Bài tập vận dụng 06

- ❑ **Bài tập 06:** *Viết chương trình nhập vào điểm ba môn toán, văn, anh của 1 học sinh, tính điểm trung bình và xuất kết quả làm tròn 2 chữ số sau dấu phẩy*

```
// Yêu cầu người dùng nhập điểm từ bàn phím
let diemToan = Number(prompt("Nhập điểm môn Toán:"));
let diemVan = Number(prompt("Nhập điểm môn Văn:"));
let diemAnh = Number(prompt("Nhập điểm môn Anh:"));
// Tính điểm trung bình
let dtb = (diemToan + diemVan + diemAnh) / 3;
// Xuất kết quả làm tròn 2 chữ số sau dấu phẩy
dtb = dtb.toFixed(2);

// In ra kết quả
console.log("Điểm trung bình =", dtb);
```



JAVASCRIPT SIÊU TỐC

FULL



12

CÁC PHÉP SO SÁNH

> >= < <=

== !=



<http://Tuhoc.CC>

1

Các phép so sánh

| Phép so sánh | Giải thích | Ví dụ | Kết quả |
|--------------|--|------------------------|----------------|
| > | Lớn hơn | $1 > 2$ | False |
| < | Nhỏ hơn | $2 < 1$ | False |
| \geq | Lớn hơn hoặc bằng | $2 \geq 1$ | True |
| \leq | Nhỏ hơn hoặc bằng | $2 \leq 2$ | True |
| $==$ | Bằng nhau Không quan tâm kiểu dữ liệu | $1 == 1$ “1” == 1 | True True |
| $==$ | Bằng nhau So sánh giá trị và kiểu dữ liệu | $1 === 1$ “1” === 1 | True False |
| $!=$ | Khác nhau Không quan tâm kiểu dữ liệu | $1 != 1$ “1” != 1 | False False |
| $!==$ | Khác nhau So sánh giá trị và kiểu dữ liệu | $1 !== 1$ “1” !== 1 | False True |

1

Sự khác nhau giữa == và ===

| Đặc Trưng | == | === |
|------------------------------------|--|--|
| Chức năng | Bằng nhau So sánh giá trị Không quan tâm kiểu dữ liệu | Bằng nhau So sánh giá trị và kiểu dữ liệu |
| Coercion ép buộc | Tuân theo coercion (ép kiểu ngầm định trong JS) | Không tuân theo coercion (ép kiểu ngầm định trong JS) |
| Type Conversion Chuyển đổi loại | Ngầm chuyển đổi kiểu dữ liệu trước khi so sánh | Không chuyển đổi kiểu dữ liệu ban đầu của chúng |

```
let a = 3;
let b = 3;
let c = "3";
```

```
console.log(a == b);
console.log(c == b);
```

$3 == 3 \rightarrow true$
 $"3" == 3 \rightarrow "3" \rightarrow$ **Bị ép kiểu ngầm định** $\rightarrow 3 \rightarrow 3 == 3 \rightarrow true$

```
console.log(a === b);
console.log(c === b);
```

$3 === 3 \rightarrow true$
 $"3" === 3 \rightarrow false$



JAVASCRIPT

SIÊU TỐC



12.2

COERCION IN JS

Làm rõ 1 số phép toán với string





JAVASCRIPT SIÊU TỐC

FULL



13

CÁC HÀM TOÁN HỌC



<http://Tuhoc.CC>

1

Các hàm toán học thông dụng

```
// 1. Hàm `Math.sqrt()`  
let soA = 25;  
let soB = Math.sqrt(soA);  
console.log(`Căn bậc hai của ${soA} là: ${soB}`);  
  
// 2. Hàm `Math.pow(base, exponent)`  
let soC = 2;  
let soD = 3;  
let result = Math.pow(soC, soD);  
console.log(`${soC} mũ ${soD} là: ${result}`);  
  
//3. Hàm `Math.abs()`  
let soE = -10;  
let absoluteValue = Math.abs(soE);  
console.log(`Giá trị tuyệt đối của ${soE} là: ${absoluteValue}`);
```

1

Các hàm toán học thông dụng

```
// 4. Hàm `Math.ceil()` và `Math.floor()`  
let decimalNumber = 4.01;  
let ceilValue = Math.ceil(decimalNumber);  
let floorValue = Math.floor(decimalNumber);  
console.log(`Làm tròn lên: ${ceilValue}, Làm tròn xuống: ${floorValue}`);  
  
// 5. Hàm `Math.round()` , từ 0.5 làm tròn thành 1  
let floatingNumber = 7.49;  
let roundedValue = Math.round(floatingNumber);  
console.log(`Làm tròn số ${floatingNumber} là: ${roundedValue}`);
```

1

Các hàm toán học thông dụng

```
//6. làm tròn x chữ số phần đơn vị
let x = 3.14159;
let xRounded = x.toFixed(2);
console.log(`x sau khi làm tròn: ${xRounded}`);
// lưu ýtoFixed sẽ trả về kết quả là string
console.log(`kiểu dữ liệu của xRounded: ${typeof xRounded}`);
// Muốn là số thì cần ép kiểu sang number
let xRounded2 = parseFloat(x.toFixed(2));
console.log(`kiểu dữ liệu của xRounded2: ${typeof xRounded2}`);
console.log(`x sau khi làm tròn: ${xRounded2}`);

// 7. Hàm `Math.min()` và `Math.max()`
let num1 = 8,
    num2 = 12,
    num3 = 5;
let minValue = Math.min(num1, num2, num3);
let maxValue = Math.max(num1, num2, num3);
console.log(`Giá trị nhỏ nhất là: ${minValue}, Giá trị lớn nhất là: ${maxValue}`);
```



JAVASCRIPT SIÊU TỐC

FULL



14

CÁC HÀM TOÁN HỌC

Number

isNaN và Number.isNaN

<http://Tuhoc.CC>

1

Hàm Number()

Number(value)

→ Chuyển value sang số

→ Nếu không chuyển được trả về *Nan*

```
var str = "123";
var num = Number(str);
console.log(`num, kiểu dữ liệu ${typeof num}`); // Kết quả: 123

var invalidStr = "abc";
var invalidNum = Number(invalidStr);
console.log(invalidNum);
// Kết quả: NaN (Not a Number) do chuỗi không thể chuyển đổi thành số
```

2

Hàm isNaN

isNaN(value) : kiểm tra xem giá trị (hoặc biểu thức) truyền vào:

Kiểm tra value **không phải định dạng số**: **NaN** (Not a Number) , hoặc không thể chuyển sang định dạng số.

Step 1: Cố gắng chuyển đổi về kiểu number

Step 2:

→ Trả về **true**, nếu giá trị sau chuyển đổi không phải là kiểu số(number)

→ Trả về **false**, nếu giá trị sau chuyển đổi là kiểu số(number)

2

Hàm isNaN

```
console.log(isNaN("abc")); // true
console.log(isNaN("123")); //false --> Do string "123" --> Chuyển đổi được thành số
console.log(isNaN(123)); // false
console.log(isNaN("NaN")); // true
console.log(isNaN(undefined)); // true
console.log(isNaN({})); // true
console.log(isNaN("blabla")); // true
console.log(isNaN(true)); // false, this is coerced to 1
// note thử chuyển true sang số
let convertTrue = Number(true);
console.log(`giá trị sau chuyển là: ${convertTrue}`);
console.log(isNaN(null)); // false, this is coerced to 0
console.log(isNaN ""); // false, this is coerced to 0
console.log(isNaN(" ")); // false, this is coerced to 0
```

3

Hàm Number.isNaN

Number.NaN: được giới thiệu trong ECMAScript 2015 (ES6).

* Phương thức này chỉ trả về `true` nếu value hoặc biểu thức truyền vào có giá trị NaN.

```
console.log(Number.isNaN(NaN));
// <- true, NaN is NaN
console.log(Number.isNaN("pony" / "foo"));
// <- true, 'pony'/'foo' is NaN, NaN is NaN
let check = "pony" / "foo";
console.log(`giá trị của biến check: ${check}`);

console.log(Number.isNaN(0 / 0)); // true
console.log(Number.isNaN(Number.NaN)); // true
console.log(Number.isNaN({})); // <- false, {} is not NaN
console.log(Number.isNaN("ponyfoo")); // <- false, 'ponyfoo' is not NaN
```



JAVASCRIPT SIÊU TỐC

FULL

[Xem toàn bộ bài giảng](#)**15**

CÁC HÀM TOÁN HỌC

Math.random()

<http://Tuhoc.CC>

1

Hàm isNaN

```
//1. Random trong [0-> 1)
let randomValue = Math.random();
console.log(`Số ngẫu nhiên từ 0 đến sát 1 là [0-1): ${randomValue}`);

// 2. Random số lớn hơn 1
let random2 = Math.random() * 10;
console.log(`Số ngẫu nhiên từ 0 đến sát 10 là : ${random2}`);

// 3.Random số nguyên
let random3 = parseInt(Math.random() * 10);
console.log(`Số nguyên ngẫu nhiên từ 0 đến sát 10 là : ${random3}`);
```

Số ngẫu nhiên từ 0 đến sát 1 là [0-1): 0.7709890518024098

Số ngẫu nhiên từ 0 đến sát 10 là : 9.264840746535052

Số nguyên ngẫu nhiên từ 0 đến sát 10 là : 2



JAVASCRIPT SIÊU TỐC

FULL

**16.1**

CÂU LỆNH IF ELSE

**Debug js in visual studio
with chrome**

<http://Tuhoc.CC>

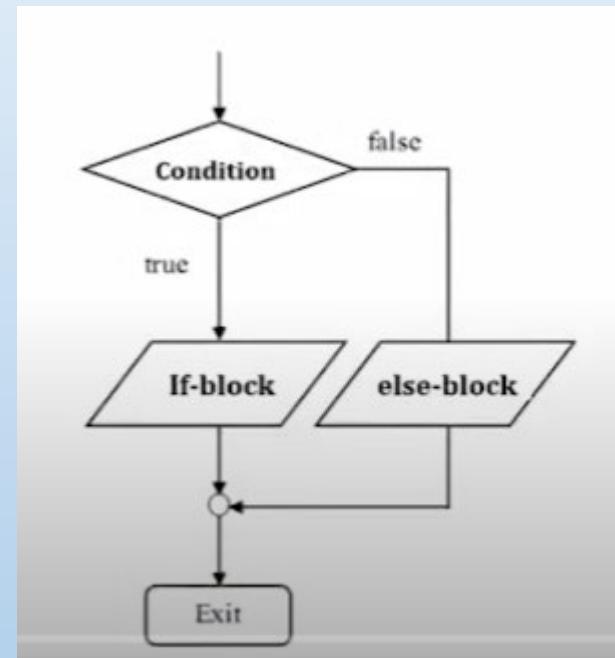
1

If else trong JS

□ Cú pháp:

```
if (Điều_Kiện)
    <Khối lệnh khi Điều_Kiện đúng>
[else
    <Khối lệnh khi Điều_Kiện sai>]
```

```
// xuất thông báo cho người dùng nhập điểm
let dtb = parseFloat(prompt("Mời cụ nhập điểm: "));
// kiểm tra điều kiện
if (dtb >= 5.0) {
    console.log("Bạn đã đỗ");
} else {
    console.log("Bạn đã tạch");
}
```





JAVASCRIPT SIÊU TỐC

FULL



16.2

CÂU LỆNH IF - ELSE IF - ELSE

Bài tập JS 07 -12



<http://Tuhoc.CC>

2

If - else if - else

Nhập vào điểm TB, in ra xếp loại của học sinh

Giỏi: $dtb \leq 10$ và $dtb \geq 8$

Khá : $8 > dtb \geq 6.5$

TB : $6.5 > dtb \geq 5$

Yếu: $0 \leq dtb < 5$

```
let score = parseFloat(prompt("Mời cụ score: "));
//kiểm tra điều kiện
if (score <= 10 && score >= 8) {
    console.log("Học sinh giỏi");
} else if (score < 8 && score >= 6.5) {
    console.log("học sinh khá");
} else if (score < 6.5 && score >= 5) {
    console.log("Học sinh trung bình");
} else if (score >= 0 && score < 5) {
    console.log("Học sinh yếu");
} else {
    console.log("Bạn nhập tào lao");
}
```

3

Bài tập vận dụng 07- 12

Bài tập 07: Tìm x, y khi biết tổng và hiệu của chúng

case test : Tong = 14 ,hieu = 4 => $x=9, y = 5$

case 2 : Tong = 8 hieu = 5 => $x=6.5, y = 1.5$

Gợi ý: $x + y = 14$

$x - y = 4$

Nhập vào tổng 2 số:

14

Nhập vào hiệu 2 số:

4

Giá trị x cần tìm là: 9

Giá trị y cần tìm là: 5

Nhập vào tổng 2 số:

8

Nhập vào hiệu 2 số:

5

Giá trị x cần tìm là: 6.5

Giá trị y cần tìm là: 1.5

3

Bài tập vận dụng 07- 12

Bài tập 08: Viết chương trình nhập vào chiều cao, cân nặng, tính BMI và xuất ra thông báo

$BMI < 15$: Thân hình quá gầy

$BMI \geq 15$ and $BMI < 16$: Thân hình gầy

$BMI \geq 16$ and $BMI < 18.5$: Thân hình hơi gầy

$BMI \geq 18.5$ and $BMI < 25$: Thân hình bình thường

$BMI \geq 25$ and $BMI < 30$: Thân hình hơi béo

$BMI \geq 30$ and $BMI < 35$: Thân hình béo

$BMI \geq 35$: Thân hình quá béo

Gợi ý cách tính : $BMI = canNang / (chieuCao ^ 2)$

Nhập vào chiều cao (m):
1.67

Nhập vào cân nặng (kg):
68

BMI của bạn = 24.382374231741
Thân hình bình thường

3

Bài tập vận dụng 07- 12

□ **Bài tập 09:** *Viết chương trình nhập vào 1 năm dương lịch, kiểm tra năm đó có phải năm nhuận hay không .*

□ **Gợi ý :** Năm nhuận là năm
(chia hết cho 4, và không chia hết cho 100) hoặc (chia hết cho 400)
⇒ *((nam %4 ==0) && (nam %100 !=0)) || (nam %400 ==0)*

□ **Case test :**

Năm nhuận : 2004, 2008, 2012, 2016, 2020, 2024

Năm không nhuận : 1900, 2005

3

Bài tập vận dụng 07- 12

Bài tập 10: *Viết chương trình cho người dùng nhập vào 1 tháng bất kỳ từ 1 – 12 => Cho biết tháng đó có bao nhiêu ngày ?*

Gợi ý :

- _ Tháng 1,3,5,7,8,10,12 có 31 ngày
- _ Tháng 4,6,9,11 có 30 ngày
- _ Nếu tháng 2 thì yêu cầu nhập thêm năm:
 - + nếu năm nhuận thì tháng 2 có 29 ngày
 - + năm không nhuận thì tháng 2 có 28 ngày

3

Bài tập vận dụng 07- 12

- Bài tập 11:** Viết chương trình giải phương trình bậc 2 : $ax^2 + bx + c = 0$

Phương trình bậc 2

$$\begin{aligned} ax^2 + bx + c &= 0 \\ \Delta &= b^2 - 4ac \end{aligned}$$

Bước 1: Tính $\Delta = b^2 - 4ac$ **Bước 2:** So sánh Δ với 0

- $\Delta < 0 \Rightarrow$ phương trình (1) vô nghiệm
- $\Delta = 0 \Rightarrow$ phương trình (1) có nghiệm kép $x_1 = x_2 = -\frac{b}{2a}$
- $\Delta > 0 \Rightarrow$ phương trình (1) có 2 nghiệm phân biệt, ta dùng công thức nghiệm sau:

$$x_1 = \frac{-b + \sqrt{\Delta}}{2a} \text{ và } x_2 = \frac{-b - \sqrt{\Delta}}{2a}$$

1. case1 : $a=1, b=2, c=-3$

\Rightarrow Pt có 2 nghiệm $x_1=1$ $x_2 = -3$

2. case2 : $a=1, b=2, c=1$

\Rightarrow Pt có nghiệm kép $x_1=x_2 = -1$

3. case3 : $a=1, b=1, c=1$

\Rightarrow Pt vô nghiệm

3

Bài tập vận dụng 07- 12

- Bài tập 12:** *Viết chương trình nhập vào tháng trong năm, cho biết tháng đó thuộc quý mấy*
- Gợi ý :**

1 năm có 4 quý, mỗi quý 3 tháng :

- + Quý 1 : tháng 1,2,3*
- + Quý 2 : tháng 4,5,6*
- + Quý 3 : tháng 7,8,9*
- + Quý 4 : tháng 10,11,12*



CÂU LỆNH IF ELSE

Giải bài tập JS 07





JAVASCRIPT SIÊU TỐC

FULL



16.4

CÂU LỆNH IF ELSE

Giải bài tập JS 08 Tính BMI



<http://Tuhoc.CC>



JAVASCRIPT SIÊU TỐC

FULL



16.5

CÂU LỆNH IF ELSE

Giải bài tập JS 09
Kiểm tra năm nhuận



<http://Tuhoc.CC>



JAVASCRIPT SIÊU TỐC

FULL



16.6

CÂU LỆNH IF ELSE

Giải bài tập JS 10
Kiểm tra số ngày trong tháng



<http://Tuhoc.CC>



CÂU LỆNH IF ELSE

Giải bài tập JS 11
Giải ptb2





JAVASCRIPT SIÊU TỐC

FULL



16.8

CÂU LỆNH IF ELSE

Giải bài tập JS 12 Kiểm tra Quý

<http://Tuhoc.CC>



JAVASCRIPT SIÊU TỐC



17

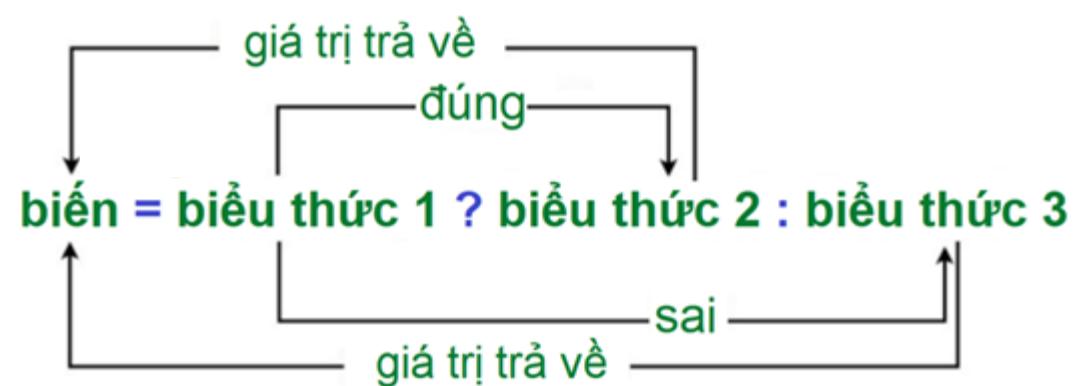
TOÁN TỬ 3 NGÔI

Ternary operator



1

Toán tử 3 ngôi

Cú Pháp:

- Ktra điều kiện, nếu **đúng** thì thực hiện **Bthuc 2**, ngược lại thì thực hiện **3**
- Bản chất rút gọn lệnh **if.. else**

```
let number = 10;
let message = number >= 0 ? "Số dương" : "Số âm";
console.log(message); // Kết quả: "Số dương"

let number2 = 11;
let message2 = (number % 2 === 0) ? "Số chẵn" : "Số lẻ";
console.log(message); // Kết quả: "Số lẻ"
```

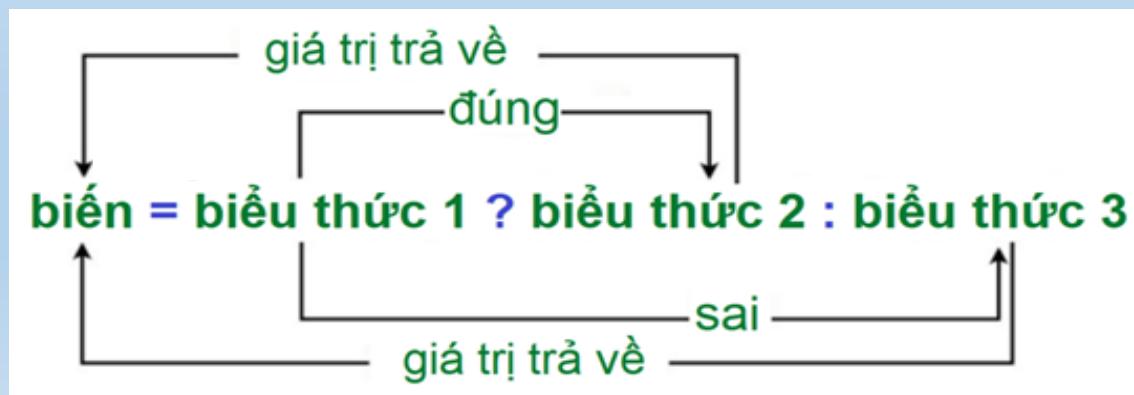
2

Bài tập js 13

Bài tập 13: Viết chương trình nhập vào điểm trung bình, và xuất ra kết quả xếp loại của học sinh theo tiêu chuẩn sau: (dùng toán tử 3 ngôi)

- ✓ *Giỏi: Nếu Điểm >= 8*
- ✓ *Khá: Nếu 8 > Điểm >= 6.5*
- ✓ *Trung bình: Nếu 6.5 > Điểm >= 5*
- ✓ *Yếu: Nếu Điểm < 5*

Biến = biểu_thức_1 ? Biểu thức 2 : (biểu thức 1' ? Biểu thức 2' : biểu thức 3)





JAVASCRIPT SIÊU TỐC

FULL



TOÁN TỬ 3 NGÔI

Giải bài tập JS 13

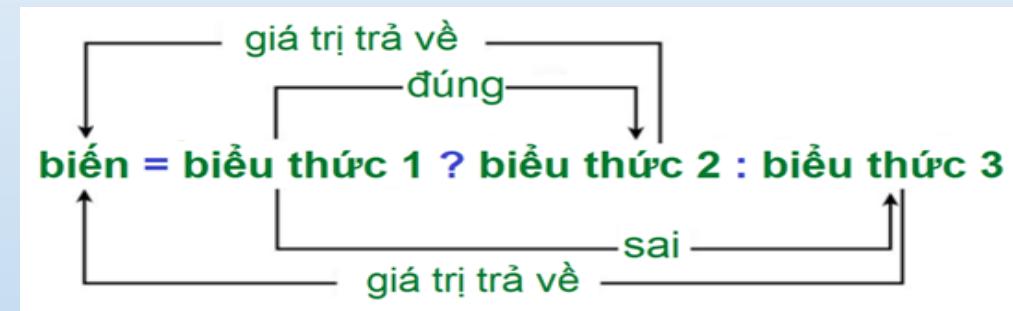


1

Giải Bài tập js 13

Bài tập 13: Viết chương trình nhập vào điểm trung bình, và kết quả xếp loại của học sinh theo tiêu chuẩn sau: (dùng toán tử 3 ngôi)

- ✓ **Giỏi:** Nếu Điểm ≥ 8
- ✓ **Khá:** Nếu $8 > \text{Điểm} \geq 6.5$
- ✓ **Trung bình:** Nếu $6.5 > \text{Điểm} \geq 5$
- ✓ **Yếu:** Nếu Điểm < 5



Biến = biểu_thức_1 ? Biểu_thức_2 : (biểu_thức_1' ? Biểu_thức_2' : biểu_thức_3)

```

let dtb = parseFloat(prompt("Mời cụ nhập điểm:"));

let xepLoai =
| dtb >= 8 ? "Giỏi" : dtb >= 6.5 ? "Khá" : dtb >= 5 ? "Trung bình" : "Yếu";
console.log(`Xếp loại của học sinh là: ${xepLoai}`);
  
```



JAVASCRIPT SIÊU TỐC



18

TRUTHY AND FALSY VALUES



1

Truthy and Falsy Values

- ✓ Chúng ta đã bàn đến ép kiểu dữ liệu từ **chuỗi sang số** dùng:
Number, parseInt, parseFloat
- ✓ Chúng ta cũng có thể ép ngược lại từ **số → chuỗi**

```
let a = 12345;
let b = String(a);
console.log(b);
console.log(typeof b);
```

| |
|--------|
| 12345 |
| string |

- ✓ Tuy nhiên đối với kiểu boolean (kiểu luận lý true false thì cần chú ý)

1

Truthy and Falsy Values

Trong **JavaScript**, một số giá trị được coi là "**falsy**" (giá trị sai)

Những giá trị này khi được chuyển đổi thành kiểu Boolean → sẽ mang giá trị **false**

Ngược lại, các giá trị không phải là "**falsy**" được coi là "**truthy**" (giá trị đúng).

1. **false**: Giá trị Boolean **false** luôn là "**falsy**".
2. **0**: Số không (**0**) được coi là "**falsy**".
3. **-0**: Số âm không (**-0**) cũng là "**falsy**".
4. **0n**: Số **BigInt** không (**0n**) cũng là "**falsy**".
5. **""**: Chuỗi rỗng (không có ký tự nào) cũng là "**falsy**".
6. **null**: Giá trị **null** được coi là "**falsy**".
7. **undefined**: Giá trị **undefined** cũng là "**falsy**".
8. **NaN**: Giá trị **NaN** (Not-a-Number) được coi là "**falsy**".

1

Truthy and Falsy Values

```
// Ví dụ kiểm tra 1 biến đã được gán giá trị hay undefined
let salary;
console.log(salary);
if (salary === true) {
    console.log("salary is defined");
} else {
    console.log("salary is undefined");
}
```



JAVASCRIPT

FULL

SIÊU TỐC



19.1

SWITCH CASE

bài tập JS 14

<http://Tuhoc.CC>

1

switch case js

☐ Cú pháp:

```
switch (biểu_thức) {  
    case giá_trị1:  
        // Đoạn mã được thực thi nếu biểu_thức có giá trị là giá_trị1  
        break;  
    case giá_trị2:  
        // Đoạn mã được thực thi nếu biểu_thức có giá trị là giá_trị2  
        break;  
    // Các case khác ở đây...  
    default:  
        // Đoạn mã được thực thi nếu không có trường hợp nào khớp  
}
```

```
let number = 2;  
switch (number % 2) {  
    case 0:  
        console.log("Số chẵn");  
        break;  
    case 1:  
        console.log("Số lẻ");  
        break;  
    default:  
        console.log("Không phải số");  
}
```

2

switch case biến thể

□ Cú pháp:

```
switch (biểu_thức) {  
    case giá_trị1:  
    case giá_trị2:  
    case giá_trị3:  
        // Câu lệnh được thực hiện nếu biểu_thức có giá trị là  
        // giá_trị1, giá_trị2, hoặc giá_trị3  
        break;  
    // Các case khác ở đây...  
    default:  
        // Đoạn mã được thực thi nếu không có trường hợp nào  
        // khớp  
}
```

□Chú ý:

Ta có thể gom nhóm các case nếu chúng cùng thực hiện 1 công việc

```
let month = parseInt(prompt("Nhập vào một tháng (1 - 12):"));  
switch (month) {  
    case 1: // Tháng 1  
    case 3: // Tháng 3  
    case 5: // Tháng 5  
    case 7: // Tháng 7  
    case 8: // Tháng 8  
    case 10: // Tháng 10  
    case 12: // Tháng 12  
        console.log("Tháng có 31 ngày.");  
        break;  
  
    case 4: // Tháng 4  
    case 6: // Tháng 6  
    case 9: // Tháng 9  
    case 11: // Tháng 11  
        console.log("Tháng có 30 ngày.");  
        break;  
  
    case 2: // Tháng 2  
        console.log("Tháng có 28 hoặc 29 ngày.");  
        break;  
  
    default:  
        console.log("Tháng không tồn tại.");  
}
```

3

Bài tập js 14

 Bài tập 14: Áp dụng switch case

Viết chương trình khung tìm kiếm: cho người dùng nhập vào lựa chọn

 1. tìm theo tên 2. tìm theo tác giả 3. tìm theo nhà xuất bản 4. tìm theo tiêu đề Thoát nếu phím bấm không hợp lệ

Chọn cách tìm kiếm:

1. Tìm theo tên
2. Tìm theo tác giả
3. Tìm theo nhà xuất bản
4. Tìm theo tiêu đề

1|

127.0.0.1:5500 cho biết

Bạn đã chọn tìm theo tên.

console.log(

'

Chọn cách tìm kiếm:

1. Tìm theo tên
2. Tìm theo tác giả
3. Tìm theo nhà xuất bản
4. Tìm theo tiêu đề

'

);



JAVASCRIPT SIÊU TỐC



19.2

SWITCH CASE

Giải bài tập JS 14



<http://Tuhoc.CC>

3

Bài tập js 14

Bài tập 14: Áp dụng switch case

Viết chương trình khung tìm kiếm:

cho người dùng nhập vào lựa chọn

1. tìm theo tên

2. tìm theo tác giả

3. tìm theo nhà xuất bản

4. tìm theo tiêu đề

Thoát nếu phím bấm không hợp lệ

```
let choice = Number(  
  prompt(  
    `Chọn cách tìm kiếm:  
    1. Tìm theo tên  
    2. Tìm theo tác giả  
    3. Tìm theo nhà xuất bản  
    4. Tìm theo tiêu đề`  
  ))  
  
switch (choice) {  
  case 1:  
    alert("Bạn đã chọn tìm theo tên.");  
    break;  
  case 2:  
    alert("Bạn đã chọn tìm theo tác giả.");  
    break;  
  case 3:  
    alert("Bạn đã chọn tìm theo nhà xuất bản.");  
    break;  
  case 4:  
    alert("Bạn đã chọn tìm theo tiêu đề.");  
    break;  
  default:  
    alert("Lựa chọn không hợp lệ.");  
    break;  
}
```



JAVASCRIPT
SIÊU TỐC

FULL



WHILE LOOP

Vòng lặp while



<http://Tuhoc.CC>

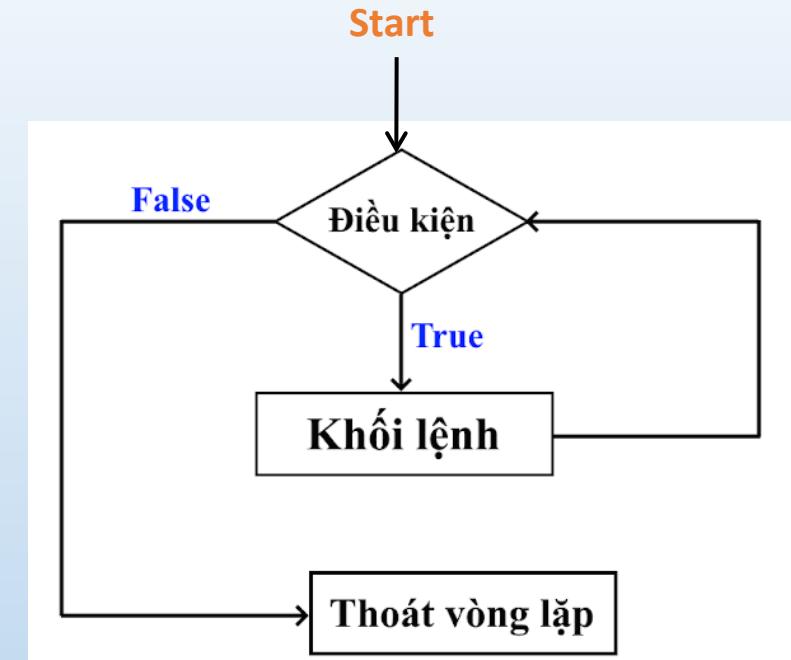
1

while JS

```
while (condition) {  
    // Code sẽ được thực thi trong vòng lặp  
  
    // Khi condition trở thành false, vòng lặp sẽ  
    dừng  
}
```

```
let i = 1;  
while (i <= 5) {  
    console.log(i);  
    i++;  
}
```

```
1  
2  
3  
4  
5
```



```
2 let i = 1;  
● 3 while (i <= 5) {  
● 4     console.log(i);  
● 5     i++;  
6 }
```

Có thể dùng debug để hiểu rõ hơn

1

while JS

*Ví dụ thực hành: Viết chương trình nhập vào số n từ bàn phím.
n phải là số nguyên nằm trong khoảng từ 1 đến 99
Nhập sai bắt nhập lại*

```
let n = prompt("Nhập vào số n (từ 1 đến 99):");
console.log(n);
// Có thể dùng n % 1 !== 0 để kiểm tra xem có phải số nguyên 0
while (isNaN(n) || n < 1 || n > 99 || n % 1 !== 0) {
    n = Number(
        prompt("Số bạn nhập không hợp lệ. Vui lòng nhập lại (từ 1 đến 99):")
    );
}

alert("Bạn đã nhập số n: " + n);
```



JAVASCRIPT



SIÊU TỐC

**DO - WHILE****Vòng lặp do-while**

1

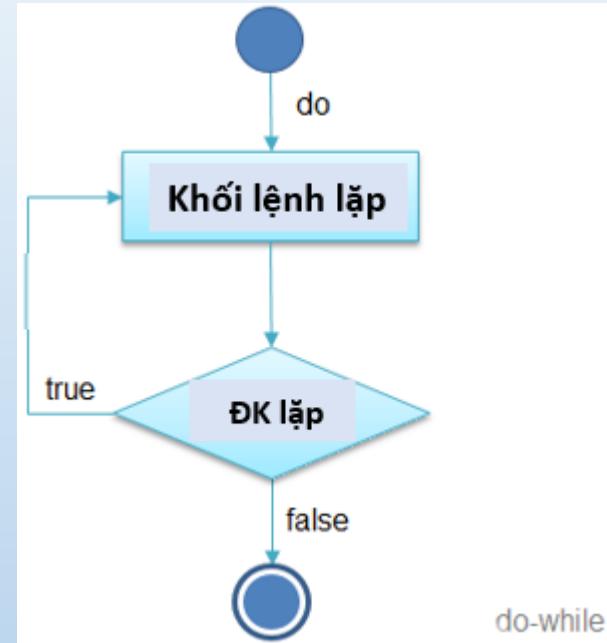
do while JS

❑ Cú pháp:

```
do {
    // Mã lệnh được thực thi ít nhất một lần
    // sau đó vòng lặp sẽ kiểm tra điều kiện
} while (điều_kiện);
```

```
let i = 1;
do {
    console.log(i);
    i++;
} while (i <= 5);
```

| |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |



```
let i = 7;
do {
    console.log(i);
    i++;
} while (i <= 5);
```

7



JAVASCRIPT
SIÊU TỐC

FULL



22

WHILE (TRUE)

Vòng lặp vô tận



<http://Tuhoc.CC>

2

while true C++

❑ **while (true)** để tạo ra một vòng lặp vô tận.

Khi sử dụng vòng lặp này, mã lệnh bên trong nó sẽ được thực thi liên tục mà không cần kiểm tra bất kỳ điều kiện nào.

```
// Tăng n lên cho đến khi n=10
let i = 0;
while (true) {
    i++;
    console.log(i);
    if (i === 10) {
        break; // Sẽ thoát khỏi vòng lặp nếu i = 10
    }
}
```



JAVASCRIPT
SIÊU TỐC

FULL



23

FOR LOOP



1

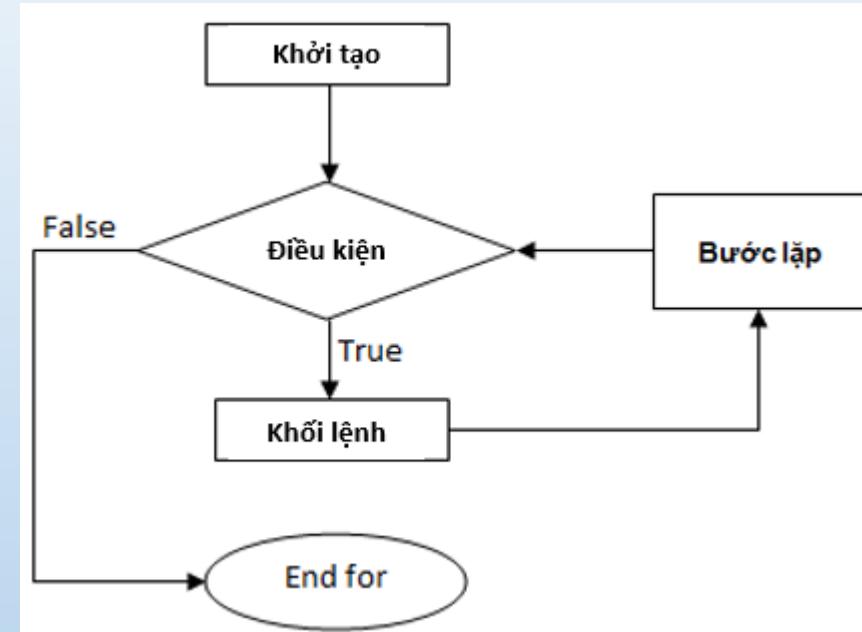
for loop JS

□ Cú pháp:

```
for ([Khởi_tạo] ; [Điều_kiện] ; [Bước_lặp])
{
    < Khối_lệnh>
}
```

```
// ví dụ 1:
for (let i = 0; i < 5; i++) {
    console.log(i); // 0 1 2 3 4
}
```

```
// ví dụ 2: Xuất các số chẵn từ 0 đến 10
for (let i = 0; i <= 10; i += 2) {
    console.log(i); //2 4 6 8 10
}
```



```
//ví dụ 3: tính Tổng các số chẵn từ 0 đến 10
let tong = 0;
for (let i = 0; i <= 10; i += 2) {
    tong += i; //tong = tong +i ;
}
console.log("Tổng các số chẵn từ 0 đến 10 là: " + tong);
```



JAVASCRIPT SIÊU TỐC

FULL



24.1

CONTINUE , BREAK

Bài tập JS 15 - 21



1

continue , break

□ Cách dùng :

- **break;** thường được dùng để thoát khỏi 1 vòng lặp
- **continue;** dùng để bỏ qua 1 giá trị trong vòng lặp

```
for (let i = 1; i <= 10; i++) {  
  if (i % 2 !== 0) {  
    continue; // Bỏ qua các số lẻ.  
  } else {  
    console.log(i); // 2 4 6 8 10  
  }  
}
```

```
let n = 0;  
while (n < 100) {  
  n++; // Tăng giá trị của n lên 1  
  if (n === 4) {  
    break; // Dừng vòng lặp khi n = 4  
  }  
}  
  
console.log(`Giá trị cuối cùng của n là: ${n}`); //4
```

2

Bài tập JS 15 - 21

❑ Bài tập JS 15:

Viết chương trình nhập vào số nguyên n, in ra kết quả n!

- ✓ *Dùng vòng lặp for*
- ✓ *Dùng vòng lặp while*

```
mời nhập vào số nguyên n:  
4  
kết quả 4! = 24
```

❑ Bài tập JS 16:

Viết chương trình nhập nhập số a từ bàn phím,

- ✓ *Nếu a chẵn thì tính tổng các số chẵn từ 0 đến a*
- ✓ *Nếu a lẻ thì in ra dòng chữ “tôi o tính tổng số lẻ, bye bye ” và thoát chương trình*

```
Moi nhap vao so nguyen a: 4  
Tong cac so chan tu 0 den 4 la: 6
```

```
Moi nhap vao so nguyen a: 15  
Toi khong tinh tong so le, bye bye
```

2

Bài tập JS 15 - 21

Bài tập JS 17:

Viết chương trình tính tổng các số lẻ từ 1 đến n, n nhập từ bàn phím

- ✓ Nhập $n = 7$, Bỏ qua không cộng tổng với số 3 => in ra kết quả
(gợi ý đáp án : $1+5+7 =13$)
- ✓ Thủ break khi vòng lặp chạy đến giá trị $n=3$

```
Moi nhap vao so nguyen n: 7
tong cac so le tu 1 - 7 ngoai tru 3 : 13
```

Bài tập JS 18:

Viết chương trình :

```
Nhung so chia het cho 3 tu 10-50 la: 12 15 18 21 24 27 30 33 36 39 42 45 48
```

- ✓ Tìm những số chia hết cho 3 từ 10 đến 50

Bài tập JS 19:

Viết chương trình :

- ✓ Tính tổng $S = 1! + 2! + 3! + \dots + 10!$

```
Tong can tinh la : 4037913
```

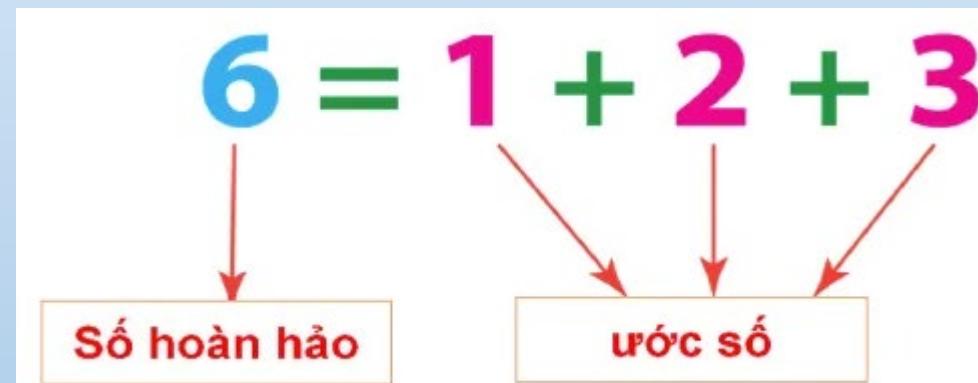
2

Bài tập JS 15 - 21

 Bài tập JS 20:

Số hoàn thiện (hay còn gọi là số hoàn chỉnh, số hoàn hảo hoặc số hoàn thành) là một số nguyên dương mà tổng các ước nguyên dương chính thức của nó (số nguyên dương bị nó chia hết ngoại trừ nó) bằng chính nó.

✓ Tìm tất cả những số hoàn thiện trong phạm vi từ 1-1000



So hoan hao trong pham vi tu 1-1000:

6 28 496

2

Bài tập JS 15 - 21

Bài tập JS 21:

Viết chương trình nhập số nguyên $a > 0$ từ bàn phím

Cho biết đó có phải số tố

(số tố là số > 1 , và chỉ chia hết cho 1 và chính nó)

✓ Kết thúc chương trình hỏi người dùng: Bạn có muốn tiếp tục sử dụng phần mềm không? Nếu chọn không thì thoát cctrinh

```
Moi nhap vao so a: -9
Vui long nhap so nguyen a > 0
Moi nhap vao so a: -15
Vui long nhap so nguyen a > 0
Moi nhap vao so a: 15
15 khong phai la so nguyen to
Ban co muon tiep tục khong?
bam n/N de thoat
1
Moi nhap vao so a: 3
3 la so nguyen to
Ban co muon tiep tục khong?
bam n/N de thoat
```



JAVASCRIPT SIÊU TỐC

FULL



Number.isInteger

Giải bài tập JS 15: Tính n!

<http://Tuhoc.CC>



JAVASCRIPT SIÊU TỐC

FULL



Giải bài tập js 16

Tính tổng số chẵn, lẻ

<http://Tuhoc.CC>



JAVASCRIPT SIÊU TỐC

FULL



Giải bài tập js 17

Tính tổng số lẻ từ 1 đến n

<http://Tuhoc.CC>



JAVASCRIPT SIÊU TỐC

FULL



Giải bài tập js 18

Tìm số chia hết cho 3



<http://Tuhoc.CC>



JAVASCRIPT SIÊU TỐC

FULL

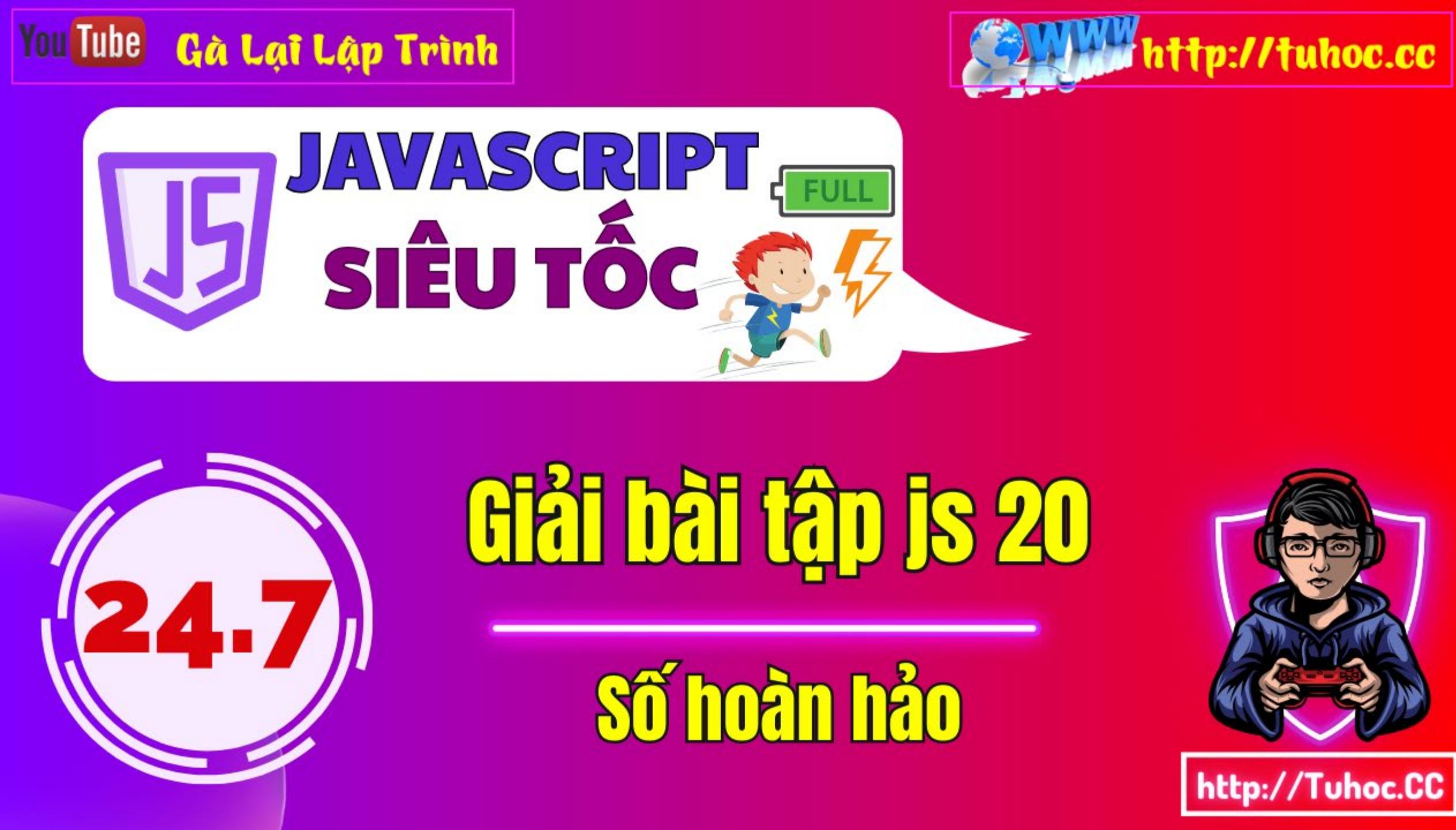


Giải bài tập js 19

Tính $S = 1! + 2! + 3! + \dots + 10!$



<http://Tuhoc.CC>



YouTube Gà Lại Lập Trình

WWW <http://tuhoc.cc>

JAVASCRIPT SIÊU TỐC

FULL



Giải bài tập js 20

Số hoàn hảo

24.7



<http://Tuhoc.CC>



JAVASCRIPT SIÊU TỐC

FULL



Giải bài tập js 21

Kiểm tra số nguyên tố



<http://Tuhoc.CC>



JAVASCRIPT

SIÊU TỐC

FULL



Function

Tổng quan



1

Function trong js

1. Khái quát về hàm :

Khi muốn thực thi một đoạn code nào nó nhiều lần, thay vì phải copy đi copy lại đoạn code đó, dẫn đến chương trình bị trùng lặp code

=> Khi đó ta sử dụng hàm

Hàm là 1 khối lệnh thực hiện 1 công việc hoàn chỉnh (module)

Hàm còn được gọi là chương trình con

Công dụng :

1. Chia nhỏ phân việc của dự án

2. Tái sử dụng: khi cần chỉ cần gọi lại chương trình con mà o cần phải viết lại

2

Khai báo Function

```
// Bước 1: Khai báo
function tenFunction() {
    // Mã lệnh thực hiện một nhiệm vụ nào đó
}
```

```
// Bước 2: Gọi hàm - khi cần sử dụng
tenFunction();
```

```
// 2. Ví dụ đơn giản
function xinChao() {
    console.log(`Chào mừng đến với tuhoc.cc`);
}
```

```
// Gọi hàm
xinChao();
xinChao();
```

Chào mừng đến với tuhoc.cc

Chào mừng Mãi vẫn dốt đến với tuhoc.cc

3

function với tham số(parameters)

```
let inputName = "Mãi vẫn dốt";
function xinChao2(name) {
    console.log(`Chào mừng ${name} đến với tuhoc.cc`);
}

// Gọi hàm
xinChao2(inputName);
xinChao2("Trần Như Nhộn");
```

Chào mừng Mãi vẫn dốt đến
với tuhoc.cc

Chào mừng Trần Như Nhộn
đến với tuhoc.cc

```
// Ví dụ 2 về parameters
function tinhTong(a, b) {
    // let ketQua = a + b;
    // return ketQua;
    return a + b;
}
```

4

Gán biến cho hàm

```
// gán biến cho hàm (lưu giá trị trả về vào biến)
let diemToan = 8;
let diemVan = 9.5;
let tongDiem = tinhTong(diemToan, diemVan);
console.log(`tongDiem = ${tongDiem}`);
console.log(`DTB = ${tongDiem / 2}`);
```

- Về cơ bản `console.log(value)` cũng là 1 function được xây dựng sẵn bởi js
→ `console.log(value)` sẽ hiển thị value mà người dùng truyền vào hiện lên cửa sổ `console`
- `Number(value)`; 1 ví dụ khác , cũng là 1 function

5

Giá trị mặc định parameters

```
function tinhTong3(a = 0, b = 0) {  
    return a + b;  
}  
console.log(tinhTong3(10));
```

Chú ý : Thân hàm nếu không có return thì giá trị trả về là undefined



JAVASCRIPT
SIÊU TỐC

FULL



Function

Function Declaration
Function Expression

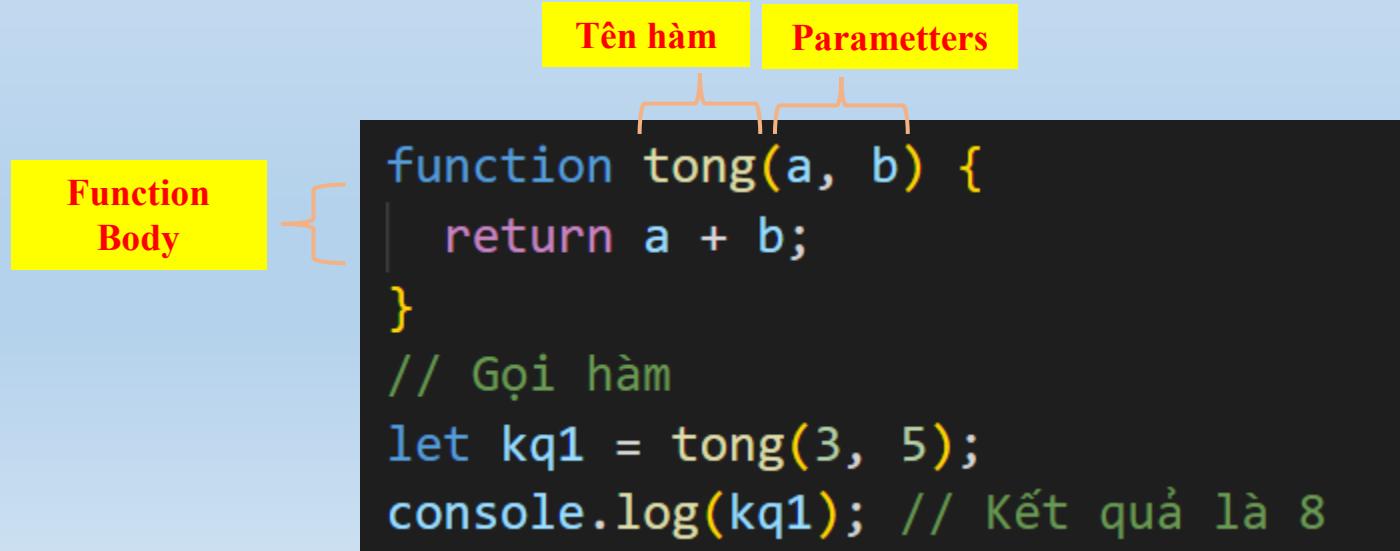
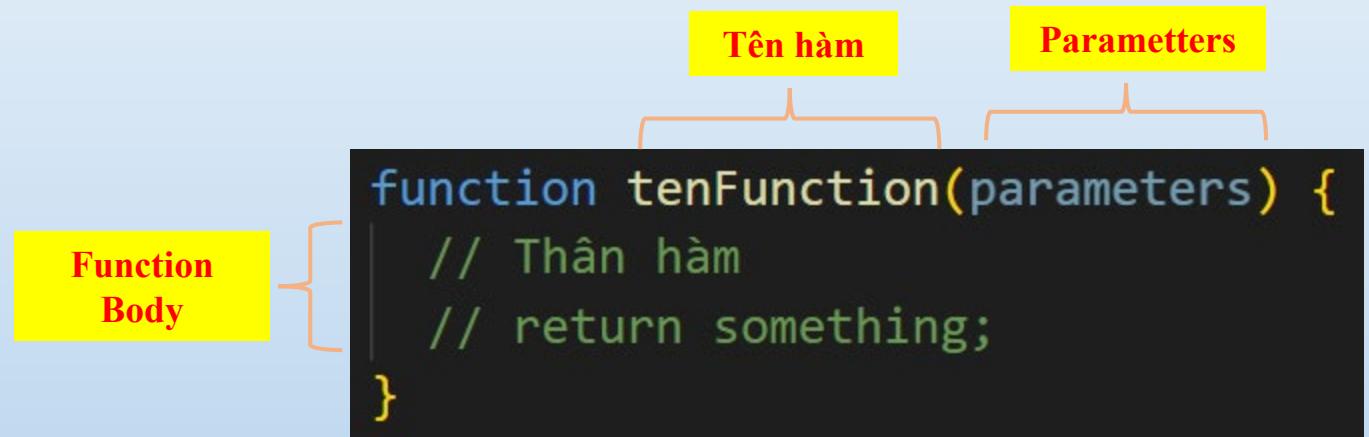


<http://Tuhoc.CC>

1

Các cách khai báo function JS

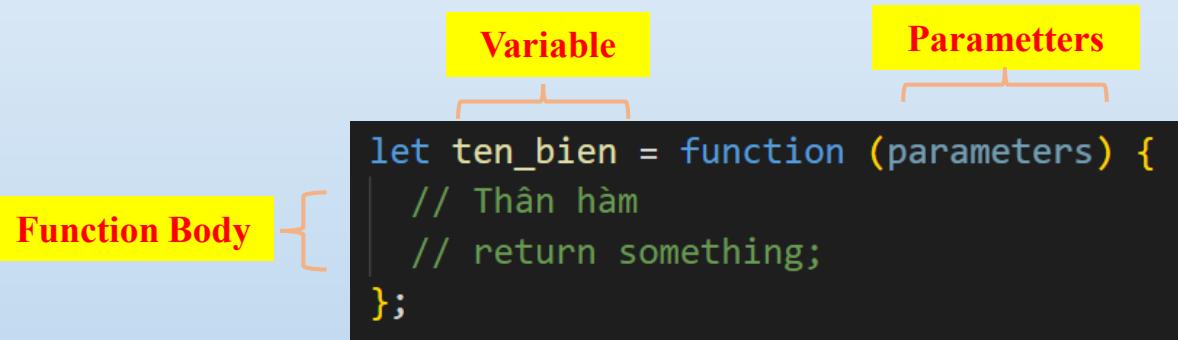
1. Function declaration



1

Các cách khai báo function JS

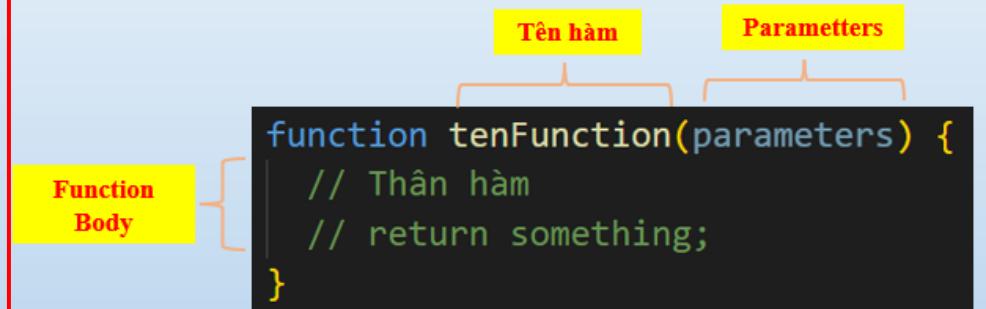
2. Function expression



```
let tich = function (a, b) {
  return a * b;
};
```

```
// Gọi hàm
let kq2 = tich(4, 6);
console.log(kq2); // Kết quả là 24
```

1. Function declaration



```
function tong(a, b) {
  return a + b;
}
```

```
// Gọi hàm
let kq1 = tong(3, 5);
console.log(kq1); // Kết quả là 8
```



JAVASCRIPT

SIÊU TỐC

FULL



Function

Arrow Function

<http://Tuhoc.CC>

1

Các cách khai báo function JS

3. Arrow function

```
// Arrow function (hàm mũi tên)
let multiplyArrow = (a, b) => a * b;
```

Với các biểu thức đơn giản,
Cú pháp sẽ ngắn gọn hơn nhiều mà không
cần phải dùng keyword return

2. Function expression

```
// Function Expression
let multiply = function (a, b) {
  return a * b;
};
```

1

Các cách khai báo function JS

3. Arrow function

Với trường hợp hàm phức tạp, cần thêm khối {}

```
let ten_bien = (parameters) =>{
    // Thân hàm
    // Return something
}
```

```
// Arrow function
let multiplyAndAddArrow = (a, b) => {
    let product = a * b;
    let sum = a + b;
    return product + sum;
};
```

```
// Gọi hàm
console.log(multiplyAndAdd(3, 5)); // 23
console.log(multiplyAndAddArrow(3, 5)); // 23
```

2. Function expression

```
let multiplyAndAdd = function (a, b) {
    let product = a * b;
    let sum = a + b;
    return product + sum;
};
```



JAVASCRIPT

SIÊU TỐC

FULL



Function

Functions Calling Other Functions



1

Functions Calling Other Functions

Trong JavaScript, "Functions Calling Other Functions" là một khái niệm mà một hàm có thể gọi một hoặc nhiều hàm khác để thực hiện các tác vụ cụ thể. Điều này giúp chia nhỏ chương trình thành các phần nhỏ, dễ quản lý và tái sử dụng.

```
// Hàm thực hiện phép cộng
function cong(a, b) {
    return a + b;
}

// Hàm thực hiện phép nhân
function nhan(x, y) {
    return x * y;
}
```

```
function congNhan(num1, num2, num3) {
    // Gọi hàm cong để tính tổng
    let sum = cong(num1, num2);
    // Gọi hàm nhan để tính tích
    let product = nhan(sum, num3);
    // Trả về kết quả
    return product;
}

// Sử dụng hàm congNhan
let result = congNhan(2, 3, 4);
console.log(result); // Output: 20
```

2

Bài tập JS 22 – 23

☐ Bài tập JS 22: (*n! cách bình thường chưa ở bài 24.2*)

Sử dụng function - Viết chương trình nhập vào số nguyên n, in ra kết quả n!

✓ Dùng vòng lặp for

✓ Dùng vòng lặp while

```
mời nhập vào số nguyên n:  
4  
kết quả 4!= 24
```

☐ Bài tập JS 23: (*PTB2 cách bình thường chưa ở bài 16.7*)

Sử dụng function Viết chương trình giải phương trình bậc 2 : $ax^2 + bx + c = 0$

a. $a=1, b=2, c=-3$

\Rightarrow Pt có 2 nghiệm $x_1=1 x_2 = -3$

b. $a=1, b=2, c=1$

\Rightarrow Pt có nghiệm kép $x_1=x_2 = -1$

c. case3 : $a=1, b=1, c=1$

\Rightarrow Pt vô nghiệm



JAVASCRIPT

FULL

SIÊU TỐC



Function

Giải bài tập JS 22
Tính n!

25.5

<http://Tuhoc.CC>



JAVASCRIPT

FULL

SIÊU TỐC



Function

Giải bài tập JS 23

Giải ptb2



<http://Tuhoc.CC>



JAVASCRIPT
SIÊU TỐC

FULL



JavaScript Scope

Global, Function và
Block Scope



1

JavaScript Scope

Phạm vi sử dụng (Scope):

Xác định nơi mà một biến có thể được truy cập hoặc sử dụng.

1. **global scope** : Phạm vi toàn cục, truy xuất được ở mọi nơi
2. **function-scope**: phạm vi của hàm.
3. **block-scope** : phạm vi của khối (block)
chẳng hạn như trong một if statement hoặc vòng lặp.

2

Global Scope

Phạm vi toàn cục (Global Scope):

Biến được khai báo bên ngoài cùng tệp js

```
9 // Global Scope
10 let a = 1;
11 // Có thể truy xuất biến global ở ngoài hàm
12 console.log(a);

13

14 function viDu_1() {
15     // Có thể truy xuất biến global ở trong hàm
16     console.log(a); → Truy xuất được trong function
17     if (a % 2 === 0) {
18         console.log(` ${a} là số chẵn`); → Truy xuất được trong {} block
19     } else {
20         console.log(` ${a} là số lẻ`);
21     }
22 }
23 viDu_1();
```

1. *global scope : phạm vi toàn cục*
2. *function-scope: phạm vi của hàm.*
3. *block-scope : phạm vi của khối (block)*

3

Function Scope

*Phạm vi hàm (Function Scope):
Biến được khai báo bên trong thân hàm*

```
// 2. Function Scope:  
function viDu_2() {  
    // Function Scope  
    let b = 2;  
    // Biến chỉ có thể truy xuất từ bên trong hàm  
    console.log(b); → Truy xuất được trong function (cùng cấp)  
    if (b % 2 === 0) {  
        console.log(` ${b} là số chẵn`); → Truy xuất được trong {}(cấp bên trong)  
    } else {  
        console.log(` ${b} là số lẻ`);  
    }  
}  
viDu_2();  
// console.log(b);  
// Lỗi - Không thể truy xuất b từ bên ngoài hàm
```

4

Block Scope

Phạm vi block {}(Block Scope):

Biến được khai báo bên trong khối {}

```
// 3. Block Scope: biến được khai báo trong {}
let n = 3;
if (n === 3) {
    let m = 1;
    console.log("m= " + m); → Truy xuất được do cùng cấp
}
console.log("m= " + m);
// báo lỗi nếu cố tình truy xuất từ bên ngoài
```

- *Phạm vi sử dụng là một cách quan trọng để tránh xung đột tên biến,*

- *Note :*

Với const, let thì biến truy xuất được, nếu cùng cấp hoặc từ phạm vi cấp thấp hơn



JAVASCRIPT SIÊU TỐC



FULL

JavaScript Hoisting

27

So sánh
var vs let / const

<http://Tuhoc.CC>

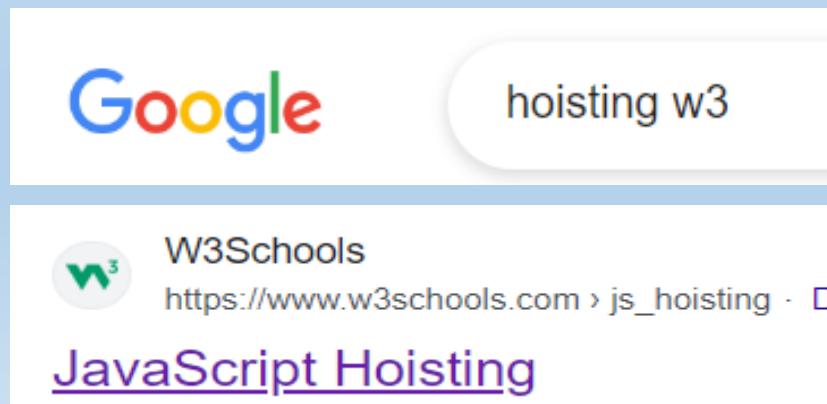
1

JavaScript Hoisting

Hoisting (Nâng cao): là hành vi của JS nhằm di chuyển tất cả các khai báo lên đầu phạm vi hiện tại (lên đầu tập lệnh hiện tại, hoặc hàm hiện tại, lên đầu block hiện tại)

1. **var:** Được nâng cao (hoisted) - lên đầu tập lệnh hiện tại, hoặc hàm hiện tại, có thể sử dụng trước khi nó được khai báo trong mã lệnh.

2. **let, const:** Cũng được nâng cao (hoisted) lên đầu phạm vi block scope tuy nhiên biến sẽ nằm trong vùng temporal dead zone - không thể sử dụng biến trước khi nó được khai báo.



2

So sánh var > let

1. Hoisting :

- a. **var:** Được nâng cao (hoisted) - có thể sử dụng trước khi nó được khai báo trong mã lệnh.
- b. **let, const:** Cũng được nâng cao (hoisted)- tuy nhiên biến sẽ nằm trong **temporal dead zone** - không thể sử dụng biến trước khi nó được khai báo.

```
console.log(c); → Truy xuất biến trước khi nó được khai báo → OK  
// undefined , a được đẩy lên khai báo ở đầu tập lệnh  
//( chú ý chỉ đẩy khai báo chứ không đẩy được giá trị)  
var c = 9;  
// nếu đổi thành let sẽ báo lỗi
```

2

So sánh var >< let

2. Phạm vi sử dụng (Scope):

a. var: Có phạm vi là function-scope :

- Nếu một biến được khai báo bằng var, phạm vi của nó sẽ là phạm vi của hàm (function)
- Biến sẽ được đẩy lên đầu phạm vi Function Scope

b. let, const: có phạm vi block scope

```
function exampleFunctionScope() {  
    if (true) {  
        var y = 20;           Khai báo sử dụng var  
        console.log(y); // Có thể truy cập y từ bên trong if block  
    }  
    console.log(y); // Có thể truy cập y từ bên ngoài if block  
}
```



```
function exampleFunctionScope() {  
    var y;           Đẩy khai báo y lên đầu phạm vi function scope  
    if (true) {  
        y = 20;  
        console.log(y); // Có thể truy cập y từ bên trong if block  
    }  
    console.log(y); // Có thể truy cập y từ bên ngoài if block  
}
```

Trình biên dịch js hiểu

2

So sánh var >< let

2. Phạm vi sử dụng (Scope):

a. **var**: Có phạm vi là *function-scope*:

b. **let, const (ES6)** : Có phạm vi là *Block Scope*:

→ Nếu một biến được khai báo bằng **let, const**

phạm vi của nó sẽ là phạm vi block

→ Biến sẽ được đẩy lên đầu phạm vi **BlockScope**

```
function exampleBlockScope() {  
    if (true) {  
        let b = 40;  
        console.log(b); // Có thể truy cập b trong cùng phạm vi block  
    }  
    // console.log(b); // Không thể truy cập b từ bên ngoài if, sẽ gây lỗi  
}
```

2

So sánh var >< let

3. Re-declaration | Khai báo lại :

var: Có thể khai báo lại một biến sử dụng var mà không gây ra lỗi.

let: Không thể khai báo lại một biến sử dụng let trong cùng một phạm vi.

```
var diemVan = 9;  
var diemVan = 5;  
let diemToan = 8.5;  
let diemToan = 9.2;
```

Khai báo lại cùng phạm vi → OK

Khai báo lại cùng phạm vi → Error

Cannot redeclare block-scoped variable 'diemToan'. ts(2451)

let diemToan: number

[View Problem \(Alt+F8\)](#) No quick fixes available



JAVASCRIPT SIÊU TỐC

FULL



JavaScript Hoisting

Hoisting in function



1

Hoisting in function

1. Function Declaration:

Có hoisting. Bạn có thể gọi hàm trước khi nó được định nghĩa.

```
hoistedFunction(); // Hoạt động
function hoistedFunction() {
  console.log("Hello");
}
```

Gọi hàm trước khi hàm được định
nghĩa → OK

2. Function Expression , Arrow Function

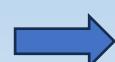
Không có hoisting. Phải gọi hàm sau khi đã đã định nghĩa

```
// nonHoistedFunction(); // Lỗi
let nonHoistedFunction = function () {
  console.log("Hello");
};
```

```
// arrowFunction(); // Lỗi
let arrowFunction = () => {
  console.log("Hello");
};
```



Function Expression js mdn



Function expression hoisting

Function expressions in JavaScript are not hoisted, unlike [function declarations](#). You can't use function expressions before you create them:



JAVASCRIPT SIÊU TỐC



JS

28

Độ Quy



1

Đệ Quy trong JS

- ❑ *Đệ quy là cách dùng hàm để tự gọi lại chính nó*
- ❑ *Để giải bằng đệ quy cần 2 điều kiện :*
 - 1.Điểm dừng của bài toán
 2. Quy luật của bài toán

Ví dụ 1: tính $N!=N*(N-1)!..1$

✓ *Quy luật:* $5! = 5 * 4!$
 $4! = 4 * 3! \Rightarrow n! = n * (n-1)!$

✓ *Điểm dừng:* $n=0$, hoặc $n=1$ giải thừa luôn bằng 1

```
// tính N!=N*(N-1)!..1
function gaiThua(N) {
    if (N === 0 || N === 1) {
        return 1;
    } else {
        return N * gaiThua(N - 1);
    }
}

// Gọi hàm tính giải thừa
var ketQua = gaiThua(5);
console.log(ketQua); //120
```

1

Đệ Quy trong JS

Ví dụ 2: Dãy Fibonacci : $F_1=1, F_2=1, F_n=F(n-1) + F(n-2)$

- ✓ *Quy luật: $F_n=F(n-1) + F(n-2)$*
- ✓ *Điểm dừng: $n \leq 2$ thì $F(n) = 1$*

$$F(n) := \begin{cases} 1, & \text{khi } n = 1; \\ 1, & \text{khi } n = 2; \\ F(n - 1) + F(n - 2) & \text{khi } n > 2. \end{cases}$$

| F_1 | F_2 | F_3 | F_4 | F_5 | F_6 | F_7 | F_8 | F_9 | F_{10} | F_{11} | F_{12} | F_{13} | F_{14} | F_{15} | F_{16} | F_{17} | F_{18} |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 | 55 | 89 | 144 | 233 | 377 | 610 | 987 | 1597 | 2584 |

```
function f(n) {
    if (n <= 2) {
        return 1;
    } else {
        return f(n - 1) + f(n - 2);
    }
}

// Gọi hàm tính Fibonacci
let ketQua2 = f(10);
console.log(ketQua2); //55
```



JAVASCRIPT SIÊU TỐC



29.1

String JS

index , length



<http://Tuhoc.CC>

1

String JS – Kiểu chuỗi

□ 1. Khái niệm :

✓ Trong js string(Chuỗi) là tập hợp các ký tự, ví dụ như chữ cái, số, hoặc ký tự đặc biệt:

```
console.log('xin chào, abc 123 !@#$.' );
console.log("xin chào, abc 123 !@#$.");
console.log(`xin chào, abc 123 !@#$.`);
```



□ 2. Khởi tạo chuỗi :

```
let s1 = `Hồi đó tôi chê mồm e rộng
Không tin hai đứa chập mồm đo`;
```

Hồi đó tôi chê mồm e rộng
Không tin hai đứa chập mồm đo

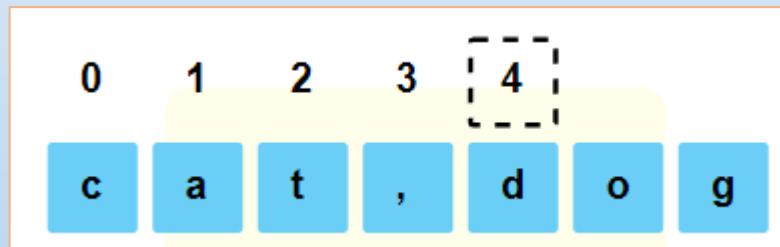
```
let s2 = "Hồi đó tôi chê mồm e rộng\nKhông tin hai đứa chập mồm đo";
console.log(s1);
console.log(s2);
console.log(typeof s1); //string
```

1

String JS – Kiểu chuỗi

 3. Quy tắc về index trong String

- ✓ Mỗi ký tự trong chuỗi có một vị trí số được gọi là index.
Index bắt đầu từ 0



Chú ý index text bắt đầu từ 0

```
let ten = "John";
console.log(ten[0]); // 'J'
```

1

String JS – Kiểu chuỗi

□ 4. (property) thuộc tính length : Kiểm tra chiều dài chuỗi

```
let s3 = "Chưa có bao giờ đẹp như hôm nay";
//tạo 1 biến lưu giá trị độ dài chuỗi
let s3Length = s3.length;
// xuất độ dài chuỗi
console.log("độ dài chuỗi s3: " + s3Length); //31
console.log("độ dài chuỗi s3: " + s3.length); //31
```



□ Ví dụ vận dụng : Nhập vào 1 tin nhắn, giới hạn ký tự nhập tối đa 140, nếu quá báo số ký tự vượt quá

```
let message = prompt("please input message");
alert(
    "bạn đã nhập: " +
    message.length +
    "ký tự, số ký tự còn lại: " +
    (140 - message.length)
);
```



JAVASCRIPT

SIÊU TỐC

FULL



29.2

String JS

slice () vs substring ()

<http://Tuhoc.CC>

2

Các phương thức cắt chuỗi - String JS

□ 5.1 5.2 Slice, substring – cắt chuỗi con

| STT | Method – Phương thức | Ý nghĩa |
|-----|---|---|
| 1 | <code>name.slice(beginIndex, [endIndex]);</code> | <i>Trích lọc chuỗi con từ chuỗi ban đầu, cắt từ beginIndex → endIndex -1</i> <i>endIndex : Không nhập → cắt đến cuối chuỗi</i> |
| 2 | <code>name.substring(startIndex, [endIndex])</code> | <i>Trích lọc chuỗi con từ chuỗi ban đầu, cắt từ startIndex → endIndex -1</i> <i>endIndex : Không nhập → cắt đến cuối chuỗi</i> |

2

Các phương thức cắt chuỗi - String JS

□ 5.1 string. slice(beginIndex, [endIndex]);

- ✓ beginIndex : Vị trí bắt đầu cắt chuỗi
- ✓ endIndex: (tùy chọn) Vị trí sát endIndex, không bao gồm endIndex.
- ✓ endIndex: Nếu không nhập sẽ ngầm định là cắt đến cuối chuỗi

```
let s4 = "0123456789";
// cắt từ vị trí index 1 đến sát 4 (4-1 = 3)
let s5 = s4.slice(1, 4); // cắt từ index 1-3
console.log("chuỗi sau cắt: " + s5); //123
```

chuỗi sau cắt: 123

□ Ví dụ vận dụng : chỉ cho phép tin nhắn dài tối đa 20 ký tự, nếu quá thì tự cắt phần thừa, và xuất chuỗi sau xử lý

2

Các phương thức cắt chuỗi - String JS

□ 5.1 string. slice(beginIndex, [endIndex])

□ **Ví dụ vận dụng :** chỉ cho phép tin nhắn dài tối đa 20 ký tự, nếu quá thì tự cắt phần thừa, và xuất chuỗi sau xử lý

```
let s6 = prompt("Mời nhập vào chuỗi");
console.log(`Bạn đã nhập ${s6.length} ký tự`);
if (s6.length > 20) {
    console.log(`Bạn đã nhập quá ${s6.length - 20} ký tự`);
    let s7 = s6.slice(0, 20);
    console.log("Chuỗi sau xử lý: " + s7);
}

// one line:
console.log(
    "chuỗi sau xử lý : " + prompt("please input some text").slice(0, 20)
);
```

2

Các phương thức cắt chuỗi - String JS

5.2 string.substring(startIndex, [endIndex]);

- ✓ **beginIndex**: Vị trí bắt đầu cắt chuỗi
- ✓ **endIndex**: (tùy chọn) Vị trí sát **endIndex**, không bao gồm **endIndex**.
- ✓ **endIndex**: Nếu không nhập sẽ ngầm định là cắt đến cuối chuỗi

```
let s8 = "0123456789";
// cắt từ vị trí index 1 đến sát 4 (4-1 = 3)
let s9 = s8.substring(1, 4); // cắt từ index 1-3
console.log("Chuỗi cắt substring: " + s9); // Kết quả: '123' Chuỗi cắt substring: 123
```

```
let s4 = "0123456789";
// cắt từ vị trí index 1 đến sát 4 (4-1 = 3)
let s5 = s4.slice(1, 4); // cắt từ index 1-3
console.log("chuỗi sau cắt: " + s5); //123
```

chuỗi sau cắt: 123

substring khá giống với slice() ?

2

Các phương thức cắt chuỗi - String JS

*** So sánh substring() vs slice()

1. Đối số truyền vào :

- ✓ **substring(start, end):** Nhận vào hai tham số là vị trí bắt đầu và kết thúc trích xuất. Nếu **end < start**, chúng sẽ được đảo ngược - Tự động hiểu số nhỏ hơn là vị trí start
- ✓ **slice(start, end):** Cũng nhận vào hai tham số là vị trí bắt đầu và kết thúc trích xuất. Nếu **end < start**, chuỗi sẽ được xem như rỗng.

```
let s10 = "0123456789";
console.log("Sử dụng substring");
console.log(s10.substring(1, 4)); // "123"
console.log(s10.substring(4, 1)); // "123" ←
// "123" (đảo ngược vị trí start và end)

console.log("Sử dụng slice");
console.log(s10.slice(1, 4)); // "123"
console.log(s10.slice(4, 1)); // "" ←
// "" (chuỗi rỗng, không đảo ngược vị trí start và end)
```

2

Các phương thức cắt chuỗi - String JS

*** So sánh substring() vs slice()

2. Xử lý với số âm:

- ✓ `substring(start, end)`: *substring sẽ chuyển số âm thành 0 và nếu end < start sau khi chuyển số âm thì nó sẽ đảo ngược chúng.*
- ✓ `slice(start, end)`: *cho phép sử dụng số âm để đếm từ cuối chuỗi. Số âm sẽ được hiểu là đếm từ cuối chuỗi về phía trước.*

```
s10 = "0123456789";
console.log("Sử dụng substring với đối số âm");
console.log(s10.substring(-3, -1));
// "" (chuyển số âm thành 0)
// s10.substring(0, 0) -> "" (chuỗi rỗng)

// TH2: substring nếu end < start đảo ngược vị trí start và end)
console.log("Trường hợp substring có 1 đối số âm, end < start ");
console.log(s10.substring(4, -5)); // 0123
// s10.substring(4, -5) -> s10.substring(4, 0) -> s10.substring(0, 4)

console.log("Sử dụng slice với đối số âm");
console.log(s10.slice(-3, -1));
// 78 do: lấy từ index -3 đến sát -1 (tức là -2)
```



JAVASCRIPT

SIÊU TỐC

FULL



29.3

String JS

trim - trimEnd

trimStart

<http://Tuhoc.CC>

2

Các phương thức cắt chuỗi - String JS

 5.3 5.4 5.5 trim, trimEnd, trimStart – xóa khoảng trắng dư thừa

| STT | Method – Phương thức | Ý nghĩa |
|-----|----------------------|---|
| 3 | trim(); | Xóa toàn bộ khoảng trắng ở cả hai đầu chuỗi |
| 4 | trimEnd(); | Xóa toàn bộ khoảng trắng ở cuối chuỗi |
| 5 | trimStart() | Xóa toàn bộ khoảng trắng ở đầu |

2

Các phương thức cắt chuỗi - String JS

□ 5.3 trim() – xóa khoảng trắng dư thừa ở cả 2 đầu chuỗi

```
//5.3 string.trim();
/*
loại bỏ các khoảng trắng ở cả hai đầu chuỗi.
trả về một chuỗi mới sau khi thực hiện
*/
let s11 = "____He      llo world____";
console.log(s11.length);
let s12 = s11.trim();
console.log(s12); // Kết quả: "He      llo world"
console.log(s12.length);
```

2

Các phương thức cắt chuỗi - String JS

□ 5.4 5.5 trimEnd / trimStart – xóa khoảng trắng dư thừa ở cuối / đầu chuỗi

```
//5.4 string.trimEnd();
//loại bỏ tất cả các khoảng trắng ở cuối chuỗi.
let s13 = "    He    llo world";
let s14 = s13.trimEnd();
console.log(s14); // Kết quả: "He    llo world"

//5.5 string.trimStart()
//loại bỏ tất cả các khoảng trắng ở đầu chuỗi.
let s15 = "    He    llo world    ";
let s16 = s15.trimStart();
console.log(s16); // Kết quả: "He    llo world    "
```



JAVASCRIPT

FULL

SIÊU TỐC



String JS

29.4

concat - charAt
toUpperCase - toLowerCase

<http://Tuhoc.CC>

3

Các phương thức String JS

□ 5.6 → 5.9: concat(), toUpperCase(), toLowerCase(), charAt()

| STT | Method – Phương thức | Ý nghĩa |
|-----|--|--|
| 6 | <code>let newString = str1.concat(str2, str3, ..., strN);</code> | <i>Nối các chuỗi str2, 3, ... n vào chuỗi str1</i> |
| 7 | <code>toUpperCase()</code> | <i>Chuyển toàn bộ chuỗi sang in hoa</i> |
| 8 | <code>toLowerCase()</code> | <i>Chuyển toàn bộ chuỗi sang in thường</i> |
| 9 | <code>charAt(index)</code> | <i>Trả về chuỗi nằm ở vị trí index được truyền vào</i> |

3

Các phương thức String JS

□ 5.6 concat nối chuỗi

□ Cú pháp : let newString = string1.concat(string2, string3, ..., stringN);

→ Nối chuỗi string 1 với str2, 3, → Trả về chuỗi mới

```
let firstName = "John";
let lastName = "Doe";

let fullName = firstName.concat(" ", lastName);
console.log(fullName); // Kết quả: 'John Doe'
```

3

Các phương thức String JS

□ 5.7 **toUpperCase()**: Chuyển toàn bộ chuỗi sang in hoa

```
// 5.7 toUpperCase() : chuyển toàn bộ sang viết hoa
let s18 = "Chào bạn, Tôi học tại TUHoc.CC";
console.log(s18.toUpperCase()); → CHÀO BẠN, TÔI HỌC TẠI TUHOC.CC
```

□ 5.8 **toLowerCase()** : Chuyển toàn bộ chuỗi sang in thường

```
// 5.8 toLowerCase() chuyển toàn bộ chuỗi sang viết thường
console.log(s18.toLowerCase()); → chào bạn, tôi học tại tuhoc.cc
```

3

Các phương thức String JS

□ 5.9 **charAt(index)**: Trả về chuỗi nằm ở vị trí index được truyền vào

```
//string.charAt(index);
let s19 = "Hello";
console.log(s19.charAt(1)); // Kết quả: 'e'
```



JAVASCRIPT



SIÊU TỐC



String JS

29.5

replace (oldValue, newValue)
repeat (count)



3

Các phương thức String JS

□ 5.10 → 5.11: replace(), repeat()

| STT | Method – Phương thức | Ý nghĩa |
|-----|---|--|
| 10 | <code>let newString = replace(oldValue, newValue);</code> | <i>Thay thế giá trị (oldValue) đầu tiên được tìm thấy trong chuỗi bằng giá trị mới (newValue).</i> |
| 11 | <code>string.repeat(count);</code> | <i>Lặp chuỗi count lần count: Số lần lặp lại chuỗi</i> |

3

Các phương thức String JS

- 5.10 replace(oldValue, newValue): *Thay thế giá trị (oldValue) đầu tiên được tìm thấy trong chuỗi bằng giá trị mới (newValue).*

```
let s14 = "hoc học nữa học mãi";  
console.log(s14.replace("hoc", "ngù"));
```

ngù học nữa học mãi

- ✓ Để thay thế tất cả dùng biểu thức chính quy (regular expression)

```
console.log(s14.replace(/hoc/g, "ngù"));
```

ngù ngù nữa ngù mãi

/: Ký tự dấu gạch chéo (forward slash) bắt đầu biểu diễn chính quy.

hoc: Đây là chuỗi cần tìm kiếm và thay thế.

/: Ký tự dấu gạch chéo kết thúc phần chuỗi cần tìm kiếm.

g: Cờ global. Khi sử dụng cờ này → js sẽ tìm kiếm toàn bộ chuỗi

3

Các phương thức String JS

5.11 repeat(count): *Lặp chuỗi count lần*

count: Số lần lặp lại chuỗi

```
let s23 = "Hello, ";
let s24 = s23.repeat(3);

console.log(s24); → Hello, Hello, Hello,
```



JAVASCRIPT SIÊU TỐC

FULL



String JS

indexOf
lastIndexOf, includes



<http://Tuhoc.CC>

4

Các phương thức tìm kiếm String JS

□ 5.12 → 5.14: indexOf (), lastIndexOf (), includes ()

| STT | Method – Phương thức | Ý nghĩa |
|-----|---------------------------------------|---|
| 12 | indexOf(searchValue, [fromIndex]); | <i>Tìm kiếm (searchValue) xuất hiện lần đầu trong chuỗi.</i> <i>Nếu tồn tại trả về vị trí index, 0 tồn tại trả về -1</i> |
| 13 | lastIndexOf(searchValue, [endIndex]); | <i>Tìm kiếm (searchValue) xuất hiện lần cuối trong chuỗi</i> <i>Nếu tồn tại trả về vị trí index, 0 tồn tại trả về -1</i> |
| 14 | includes(searchValue, [fromIndex]); | <i>Kiểm tra chuỗi con</i> <i>fromIndex (tùy chọn): giống mục 12</i> |

4

Các phương thức tìm kiếm String JS

5.12 string.indexOf(searchValue, [fromIndex]);

Tìm kiếm (*searchValue*) xuất hiện lần đầu tiên trong chuỗi gốc.

Nếu giá trị tìm kiếm không được tìm thấy, phương thức trả về **-1**

fromIndex (tùy chọn): Vị trí bắt đầu tìm kiếm trong chuỗi.

Nếu bỏ trống, tìm kiếm sẽ bắt đầu từ đầu chuỗi.

```
let s25 = "abcdef abcdef";  
  
let s26 = s25.indexOf("c");  
console.log(s26); // Kết quả: 2  
// tìm "c" nhưng bắt đầu từ index 3  
console.log(s25.indexOf("c", 2)); // -1 do không tìm thấy c  
let notFoundIndex = s25.indexOf("g");  
console.log(notFoundIndex); // Kết quả: -1
```

4

Các phương thức tìm kiếm String JS

5.13 string.lastIndexOf(searchValue, [endIndex]);

Tìm kiếm giá trị xuất hiện **cuối cùng** của chuỗi (**searchValue**) trong chuỗi gốc.

Nếu giá trị tìm kiếm **không được tìm thấy**, phương thức trả về -1

endIndex (tùy chọn): Tìm kiếm trong khoảng từ vị trí index 0 → endIndex

Nếu bỏ trống, tìm kiếm toàn bộ chuỗi.

```
let s27 = "abcdef abcdef";  
  
let s28 = s27.lastIndexOf("a");  
console.log(s28); // Kết quả: 7  
// Tìm với tham số endIndex (tìm trong đoạn từ index 0 -> endIndex)  
console.log(s27.lastIndexOf("a", 7)); //tìm trong abcdef a --> 7  
console.log(s27.lastIndexOf("a", 6)); //tìm trong abcdef --> 0
```

4

Các phương thức tìm kiếm String JS

5.14 string.includes(searchValue,[fromIndex]);

Tìm kiếm (*searchValue*) có trong chuỗi gốc hay không ?

Nếu có trả về *true*

Không có trả về *false*

fromIndex (tùy chọn): Vị trí bắt đầu tìm kiếm trong chuỗi.

Nếu bỏ trống, tìm kiếm toàn bộ chuỗi

```
let s29 = "abcdef abcdef";
console.log(s29.includes("a"));
console.log(s29.includes("g"));
// Tìm a bắt đầu từ vị trí index 8 (tìm trong bcdef)
console.log(s29.includes("a", 8));
```



JAVASCRIPT

SIÊU TỐC

FULL



String JS

startsWith endsWith

<http://Tuhoc.CC>

4

Các phương thức tìm kiếm String JS

□ 5.15 → 5.16: startsWith(), endsWith()

| STT | Method – Phương thức | Ý nghĩa |
|-----|---|--|
| 15 | <code>startsWith(searchValue,[startIndex])</code> | <i>Kiểm tra chuỗi gốc có bắt đầu bằng searchValue không? Đúng trả về true, sai trả về false</i> <i>startIndex (tùy chọn): Vị trí bắt đầu tìm kiếm trong chuỗi. Nếu bỏ trống: tìm kiếm sẽ bắt đầu từ đầu chuỗi.</i> |
| 16 | <code>endsWith(searchValue,[endIndex]);</code> | <i>Kiểm tra chuỗi gốc có kết thúc bằng searchValue không? Đúng trả về true, sai trả về false</i> <i>endIndex (tùy chọn): Vị trí kết thúc tìm kiếm trong chuỗi gốc. Nếu bỏ trống: tìm kiếm đến hết chuỗi gốc</i> |

4

Các phương thức tìm kiếm String JS

5.15 string. startsWith(searchValue,[startIndex]);

Kiểm tra chuỗi gốc có bắt đầu bằng searchValue không?

Đúng trả về true, sai trả về false

startIndex (tùy chọn): Vị trí bắt đầu tìm kiếm trong chuỗi.

Nếu bỏ trống: tìm kiếm sẽ bắt đầu từ đầu chuỗi.

```
let s30 = "abcdef abcdef";
console.log(s30.startsWith("a")); // Kết quả: true
console.log(s30.startsWith("b")); // Kết quả: false
// Tìm từ vị trí index số 1 xem có phải bắt đầu bằng chuỗi bc?
console.log(s30.startsWith("bc", 1)); // Kết quả: true
```

4

Các phương thức tìm kiếm String JS

5.16 string.endsWith(searchValue,[endIndex]);

Kiểm tra chuỗi gốc có **kết thúc bằng** *searchValue* không?

Đúng trả về **true**, sai trả về **false**

endIndex (tùy chọn): Vị trí kết thúc tìm kiếm trong chuỗi gốc = (*endIndex* -1)

Nếu bỏ trống: tìm kiếm đến hết chuỗi gốc

```
let s31 = "01234567890";  
  
console.log(s31.endsWith("0")); // Kết quả: true  
//endPosition =10, chuỗi được kiểm tra từ index 0 -> 10-1 = 9  
console.log(s31.endsWith("9", 10)); // Kết quả: true  
console.log(s31.endsWith("5")); // Kết quả: false
```

❑ Ví dụ kiểm tra xem tên tệp âm thanh có kết thúc .mp3?

4

Các phương thức tìm kiếm String JS

5.16 string.endsWith(searchValue,[endIndex]);

□ Ví dụ kiểm tra xem tên tệp âm thanh có kết thúc .mp3?

```
// Tên của tệp tin cần kiểm tra
let tenTepTin = "nhac.mp3";
// Kiểm tra xem tệp tin có kết thúc bằng ".mp3" không
if (tenTepTin.endsWith(".mp3")) {
    console.log("Tệp tin là file âm thanh MP3.");
} else {
    console.log("Tệp tin không phải là file âm thanh MP3.");
}
```



JAVASCRIPT

FULL

SIÊU TỐC



29.8

String JS

Split ()

<http://Tuhoc.CC>

5

Tách chuỗi thành các phần tử mảng

5.17 string.split(separator, [limit])

Tách chuỗi trả về mảng (mảng sẽ học chuyên sâu sau nhé ☺)

1. string: *Chuỗi gốc cần chia.*

2. separator: *chuỗi để xác định vị trí thực hiện tách.*

Nó có thể là một ký tự hoặc một biểu thức chính quy.

3. limit (tùy chọn): *Giới hạn số lượng phần tử trong mảng kết quả.*

```
let fruits = "apple,orange,banana,grape";
// Tách dựa trên dấu ,
let fruitArray = fruits.split(",");
console.log(fruitArray);
// Output: ["apple", "orange", "banana", "grape"]
// Giới hạn phần tử mảng
let fruitArray2 = fruits.split(",", 3);
console.log(fruitArray2);
// Output: ['apple', 'orange', 'banana']
```

5

Tách chuỗi thành các phần tử mảng

5.17 string.split(separator, [limit])

1. string: Chuỗi gốc cần chia.

2. separator: chuỗi để xác định vị trí thực hiện tách.

Nó có thể là một ký tự hoặc một biểu thức chính quy.

3. limit (tùy chọn): Giới hạn số lượng phần tử trong mảng kết quả.

```
// Nếu nhập vào "" thì sẽ tách rời từng ký tự -> Mảng
let str = "abcdefgh";
let arr = str.split("");
console.log(arr);
// Out put: ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
```



JAVASCRIPT

SIÊU TỐC

FULL



String JS

30.1

Bài tập vận dụng 24- 28



1

Bài tập js 24 - 28

Bài tập js 24:

Nhập vào 1 chuỗi từ bàn phím ,

1. đếm xem có bao nhiêu ký tự thường
2. bao nhiêu in hoa
3. bao nhiêu số
4. bao nhiêu space

```
case test : 123AbcDD
Số ký tự thường: 2
Số ký tự in hoa: 3
Số chữ số: 3
Số khoảng trắng: 0
```

1

Bài tập js 24 - 28

Bài tập js 25:

a. *Viết chương trình kiểm tra tính hợp lệ của mật khẩu:*

1. *mật khẩu hợp lệ khi có ít nhất 6 ký tự*
2. *chứa ít nhất 1 chữ cái viết hoa*
3. *chứa ít nhất 1 chữ cái viết thường*
4. *chứa ít nhất 1 chữ số*

b. *Cho người dùng nhập vào mật khẩu để login / so sánh, nếu đúng mở cửa, sai quá 5 lần khóa đăng nhập, thoát chương trình*

// Ví dụ mật khẩu hợp lệ : Abc123

1

Bài tập js 24 - 28

❑ Bài tập js 26:

Viết chương trình chuyển tin nhắn sang mật mã theo bảng :

const a = "abcdefghijklmnopqrstuvwxyz"

const b = "zxcvbnmasdfghjklqwertyuiop"

Nhập tin nhắn cần mã hóa:

abcxyz|

Tin nhắn đã được mã hóa: zxciop

❑ Bài tập js 27:

const a = "tôi chăm học tôi chịu khó tôi đẹp zai";

Đếm từ tôi trong string a trên ?

Số lần xuất hiện của từ "tôi" trong chuỗi là: 3

❑ Bài tập js 28:

Viết chương trình tách số và chữ từ chuỗi nhập vào thành 2 chuỗi :

** ví dụ nhập vào : abc123 sẽ tách và in ra thành 2 chuỗi abc và 123*

Nhập chuỗi:

abc123|

Chuỗi chữ: abc

Chuỗi số: 123



JAVASCRIPT SIÊU TỐC

FULL



String JS

Giải bài tập js 24



<http://Tuhoc.CC>



JAVASCRIPT SIÊU TỐC

FULL



30.3

String JS

Giải bài tập js 25



<http://Tuhoc.CC>



JAVASCRIPT SIÊU TỐC

FULL



String JS

Giải bài tập js 26



<http://Tuhoc.CC>



JAVASCRIPT SIÊU TỐC

FULL



30.5

String JS

Giải bài tập js 27



<http://Tuhoc.CC>



JAVASCRIPT SIÊU TỐC

FULL



30.6

String JS

Giải bài tập js 28



<http://Tuhoc.CC>



JAVASCRIPT SIÊU TỐC

FULL



Date & time



31.1

<http://Tuhoc.CC>

1

Giới thiệu về date time trong JS

- ✓ *Thời gian và ngày tháng là một phần quan trọng trong phát triển web.*
- ✓ *Trong JavaScript, có một đối tượng mạnh mẽ là **Date** giúp bạn xử lý và hiển thị thời gian và ngày tháng.*
- ✓ *Chúng ta sẽ tìm hiểu cách tạo, hiển thị, và thực hiện các thao tác khác liên quan đến **Date object***

1. Tạo đối tượng date : Sử dụng **new Date() để tạo một đối tượng Date mới.**

```
let currentDate = new Date();
```

Mặc định : Đối tượng Date sẽ tự động lấy thời gian hiện tại khi tạo

```
console.log(currentDate);
```



Fri Feb 09 2024 07:42:49 GMT+0700 (Giờ Đông Dương)

2

Các phương thức lấy thông tin thời gian

Fri Feb 09 2024 07:42:49 GMT+0700 (Giờ Đông Dương)

```
// Lấy năm, tháng, ngày, giờ, phút, giây
const year = currentDate.getFullYear();
const month = currentDate.getMonth() + 1; // Tháng bắt đầu từ 0
const day = currentDate.getDate();
const hours = currentDate.getHours();
const minutes = currentDate.getMinutes();
const seconds = currentDate.getSeconds();

console.log("Năm hiện tại: " + year);
console.log("Tháng hiện tại: " + month);
console.log("ngày hiện tại: " + day);
console.log("Giờ hiện tại: " + hours);
console.log("Phút hiện tại: " + minutes);
console.log("Giây hiện tại: " + seconds);
```

| |
|--------------------|
| Năm hiện tại: 2024 |
| Tháng hiện tại: 2 |
| ngày hiện tại: 2 |
| Giờ hiện tại: 11 |
| Phút hiện tại: 13 |
| Giây hiện tại: 22 |

3

Timestamp là gì ?

Timestamp là một đại diện cho một điểm cụ thể trong thời gian, tính bằng millisecond từ mốc 0
Mốc thời gian 0:

00:00:00 ngày 1 tháng 1 năm 1970

```
// Xuất thời gian tại mốc 0
const timestamp1 = new Date(0);
console.log(timestamp1);
//Thu Jan 01 1970 07:00:00 GMT+0700 (Giờ Đông Dương)
```

```
// sử dụng getTime(); để lấy timestamp hiện tại
const currentTimeStamp = new Date().getTime();
// timestamp tính bằng mili giây
console.log("Timestamp hiện tại:", currentTimeStamp);
```

Timestamp hiện tại: 1706849013409



JAVASCRIPT SIÊU TỐC

FULL



Date & time

31.2

Định dạng và hiển thị
ngày tháng



4

Set ngày tháng theo ý muốn

```
//Cách 1 : new Date(year, monthIndex, day, hours, minutes, seconds, milliseconds)
const myDate1 = new Date(2024, 1, 14); // Tháng 1 là 0, Tháng 2 là 1, ...
console.log(myDate1);
console.log(myDate1.toLocaleDateString()); // Output: 2/14/2024

//Cách 2 : new Date(dateString)
const myDate2 = new Date("2024-02-15T12:30:45");
console.log(myDate2.toLocaleDateString());
/*
"2022-02-15" là ngày 14 tháng 2 năm 2022.
"T" là ký tự phân tách.
"12:30" là thời gian 12 giờ 30 phút.
*/
```

4

Set ngày tháng theo ý muốn

```
// Cách 3: Sử dụng setFullYear, setMonth, setDate
let myDate3 = new Date();
myDate3.setFullYear(2024);
myDate3.setMonth(1); // Tháng 2 - do index month chạy từ 0
myDate3.setDate(16);
console.log(myDate3); // Fri Feb 16 2024 16:23:02 GMT+0700 (Giờ Đông Dương)
```

5

Xuất ngày tháng năm

☐ Cách 1: Định dạng date time sử dụng phương thức **toLocaleDateString()**

```
myDate3.setFullYear(2024);
myDate3.setMonth(1); // Tháng 2 - do index month chạy từ 0
myDate3.setDate(16);
```

```
console.log(myDate3.toLocaleDateString()); // Output: 16/2/2024
```



toLocaleDateString js

Syntax

JS

```
toLocaleDateString()
toLocaleDateString(locales)
toLocaleDateString(locales, options)
```



MDN Web Docs

<https://developer.mozilla.org> › ... › Date · Dịch trang này[Date.prototype.toLocaleDateString\(\) -](#)

5

Xuất ngày tháng năm

☐ Cách 2: Định dạng Date time tự code

```
// Xuất ngày tháng năm theo định dạng mong muốn(tự code):
console.log(
  `Ngày ${myDate3.getDate()}/Tháng ${
    myDate3.getMonth() + 1
  } /Năm ${myDate3.getFullYear()}`
);

// Thêm số 0 để xuất ngày tháng dạng 01, 02 ...
let prefixDate = myDate3.getDate() < 10 ? "0" : "";
let prefixMonth = myDate3.getMonth() < 9 ? "0" : "";
console.log(
  `Ngày ${prefixDate}${myDate3.getDate()}/Tháng ${prefixMonth}${
    myDate3.getMonth() + 1
  } /Năm ${myDate3.getFullYear()}`
);
```

Ngày 16/Tháng 2 /Năm 2024

Ngày 16/Tháng 02 /Năm 2024



JAVASCRIPT SIÊU TỐC

FULL



Date & time

setTimeout ()



31.3



1

SetTimeout

❑ *setTimeout* được sử dụng để **thực hiện một hàm sau một khoảng thời gian delay nhất định.**

❑ Cú pháp:

setTimeout(function, milliseconds, param1, param2, ...)

- **function:** Hàm mà chúng ta muốn thực hiện sau khoảng thời gian (Xem lại 25.2 & 25.3)
- **delay:** Thời gian chờ trước khi hàm được gọi, được đo bằng mili giây.

1. Function declaration

```
Tên hàm      Parametters  
function tenFunction(parameters) {  
    // Thân hàm  
    // return something;  
}
```

2. Function expression

```
Variable      Parametters  
let ten_bien = function (parameters) {  
    // Thân hàm  
    // return something;  
};
```

3. Arrow function

```
let ten_bien = (parameters) => {  
    // Thân hàm  
    // Return something  
}
```

1

SetTimeout

❑ Cú pháp:

`setTimeout(function, milliseconds, param1, param2...)`

1. setTimeout với 3 cách khai báo hàm đã học

```
// a. với Arrow function
let helloArrow = () => {
  console.log("Hello");
};

setTimeout(helloArrow, 3000);
```

// b. với function declaration

```
function xinChao() {
  console.log("Hello in function declaration");
}

setTimeout(xinChao, 3000);
```

// c. với function expression

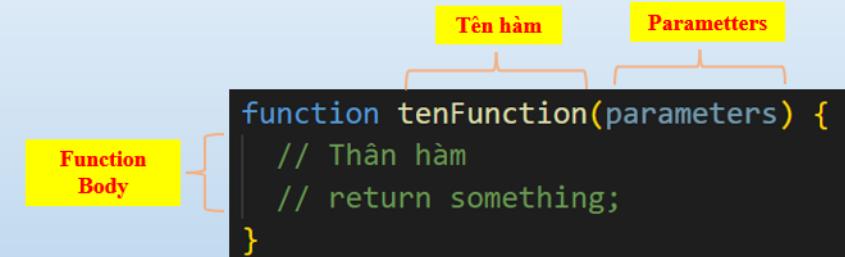
```
let helloExpression = function () {
  console.log("Hello in function expression");
};

setTimeout(helloExpression, 3000);
```

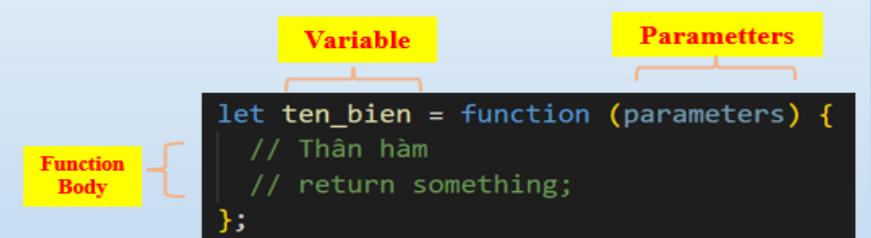
3. Arrow function

```
let ten_bien = (parameters) =>{
  // Thủ tục
  // Return something
}
```

1. Function declaration



2. Function expression



1

SetTimeout

Cú pháp:

`setTimeout(function, milliseconds, param1, param2...)`2. *setTimeout* : Truyền trực tiếp function

```
setTimeout(function xinChao() {
    console.log("hello in function declaration");
}, 3000);
```

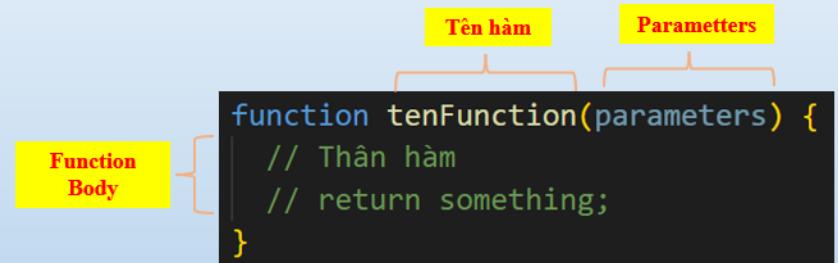
//b. setTimeout với function expression

```
setTimeout(function () {
    console.log("hello in function expression");
}, 3000);
```

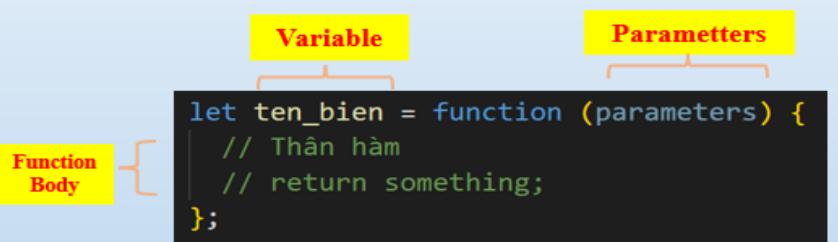
//c. setTimeout với arrow function

```
setTimeout(() => {
    console.log("Hello");
}, 3000); //delay 3s
```

1. Function declaration



2. Function expression



3. Arrow function

```
let ten_bien = (parameters) =>{
    // Thủ hàm
    // Return something
}
```

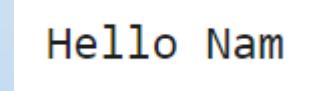
1

SetTimeout

❑ Cú pháp:

`setTimeout(function, milliseconds, param1, param2...)`3. **setTimeout** : Trường hợp hàm có tham số truyền vào

```
// Ví dụ: với Arrow function
let helloArrow2 = (yourName) => {
  console.log("Hello" + yourName);
};
setTimeout(helloArrow2, 3000, " Nam");
```

4 **setTimeout** : Huỷ thực hiện hàm

Khi bạn sử dụng **setTimeout** để tạo một đợi (delay) trong việc thực thi một hàm, **setTimeout** sẽ trả về một **ID** của timeout

→ Gán **ID** của timeout vào một biến, để tham chiếu đến nó và hủy bỏ thực hiện hàm nếu cần

```
// 4. Khi muốn hủy quá trình thực hiện hàm
// step 1: Gán setTimeout vào 1 biến để lưu giá trị
let timeOut = setTimeout(helloArrow2, 3000, "Giang");
// step 2: Sử dụng clearTimeout để huỷ thực hiện hàm
clearTimeout(timeOut);
```



JAVASCRIPT SIÊU TỐC

FULL



Date & time

setInterval()

Bài tập js 29- 31



1

setInterval

□ *setInterval* giúp thực hiện một function **lặp đi lặp lại sau một khoảng thời gian cố định**

□ Cú pháp:

setInterval (**function**, **milliseconds**, **param1**, **param2**, ...)

- **function**: Hàm mà chúng ta muốn thực hiện **lặp lại sau mỗi 1 khoảng thời gian**
- **milliseconds** : Thời gian hàm sẽ tự động được gọi lại lần tiếp theo

□ *Cách sử dụng: Tương tự với setTimeout có thể gọi hàm trực tiếp hoặc thông qua tên hàm, tên biến*

1

setInterval

Cú pháp:

`setInterval(function, milliseconds, param1, param2...)`

```
//Ví dụ với function declaration
function showNotification() {
  console.log("Bạn có xxx tin chưa đọc");
}
// In ra "thông báo" sau 2 giây (2000ms)
setInterval(showNotification, 2000);
```

4 Bạn có xxx tin chưa đọc

```
function updateTime() {
  let currentTime = new Date();
  console.log(currentTime);
}
setInterval(updateTime, 1000);
```

Sat Feb 10 2024 07:20:01
GMT+0700 (Giờ Đông Dương)

Sat Feb 10 2024 07:20:02
GMT+0700 (Giờ Đông Dương)

```
let counter = 0;
```

```
// 1. function declaration
function count() {
  console.log(counter++);
}
setInterval(count, 1000);
```

```
// 2. function expression
let count = function () {
  console.log(counter++);
};
setInterval(count, 1000);
```

```
//3. arrow function
let count = () => {
  console.log(counter++);
};
setInterval(count, 1000);
```

1

setInterval

□ Cú pháp:

`setInterval(function, milliseconds, param1, param2...)`

□ Huỷ lặp

intervalID là một giá trị duy nhất được trả về bởi hàm `setInterval`

→ Gán *intervalID* vào một biến, để tham chiếu đến nó và **hủy lặp** nếu cần

```
let count = () => {
    console.log(counter++);
    if (counter === 5) {
        // step 2: Dừng lặp dùng clearInterval
        clearInterval(stopInterval);
    }
};

//step 1: Gán biến để nhận giá trị trả về
let stopInterval = setInterval(count, 1000);
```

2

Bài tập js 29 - 31

 Bài tập js 29:

Viết chương trình nhập vào năm sinh và in ra số tuổi,

Kiểm tra điều kiện dữ liệu năm sinh nhập vào phải là số nguyên, phải lớn hơn 0

Nhập năm sinh của bạn:

1987

Tuổi của bạn là: 37 tuổi

Nhập năm sinh của bạn:

abc123|

Năm sinh không hợp lệ. Năm sinh là số nguyên lớn hơn 0

Nhập năm sinh của bạn:

1985.62

Năm sinh không hợp lệ. Năm sinh là số nguyên lớn hơn 0

2

Bài tập js 29 - 31

Bài tập js 30:

Viết chương trình đếm ngược thời gian theo từng giây(countdown)

Ví dụ thời gian làm bài là 45 phút nếu chạy về 0 thì thông báo hết thời gian

Mời nhập thời gian làm bài

Thời gian của bạn làm bài là: 45 phút

OK

45:00
44:59
44:58
44:57

2

Bài tập js 29 - 31

□ Bài tập js 31:

Viết chương trình có tên timeSince, thông báo người dùng offline x phút ...

ví dụ: bạn đang chat với bạn A, sau đó bạn A offline

→ Yêu cầu: update hiển thị thời gian A offline 'x giây trước' 'x phút trước', 'x ngày trước', 'x tháng trước', 'x năm trước'

```
const now = new Date();
console.log("Xem timestamp hiện tại: " + now.getTime());
const timeDifference = now - timestamp;
const seconds = Math.floor(timeDifference / 1000);
const minutes = Math.floor(seconds / 60);
const hours = Math.floor(minutes / 60);
const days = Math.floor(hours / 24);
const months = Math.floor(days / 30);
const years = Math.floor(months / 12);
```

Xem timestamp hiện tại: 1707387891820

Online 2 ngày trước

```
// Ví dụ sử dụng
```

```
const timestamp = 1707206532854; // Thời điểm bạn A offline
```



Date & time

Giải Bài tập js 29



Date & time

Giải Bài tập js 30



JAVASCRIPT SIÊU TỐC

FULL



Date & time

Giải Bài tập js 31

31.7



<http://Tuhoc.CC>



JAVASCRIPT SIÊU TỐC

FULL



32.1

JavaScript Array

Tổng quan về mảng



<http://Tuhoc.CC>

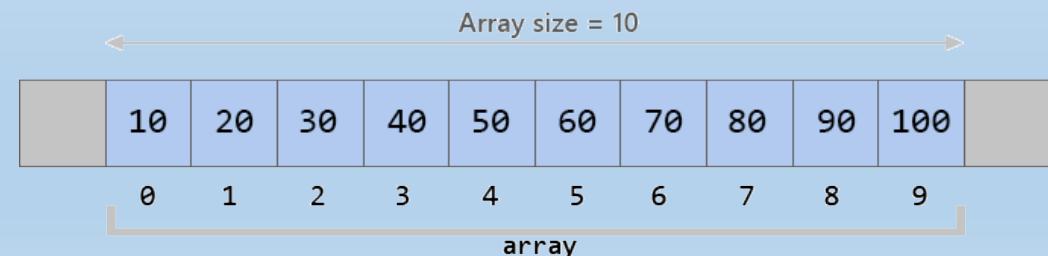
1

Mảng – array JS

 1 . Tại sao phải dùng mảng:

Ví dụ : Chúng ta có khoảng 20 người bạn cần lưu tên, nếu không dùng mảng thì chúng ta phải khai báo 20 biến

- Quản lý và tổ chức dữ liệu một cách hiệu quả
- Truy cập và thay đổi dữ liệu dễ dàng.

 2. Khái niệm : Mảng là tập hợp các phần tử có thể cùng hoặc khác kiểu dữ liệu
(Number, string, array, object ,boolean ...)

Mảng có index bắt đầu từ 0

1

Mảng – array JS

□ 3. Khai báo mảng :

Cách 1 (Thường dùng): Sử Dụng Cặp Dấu Ngoặc Vuông []:

```
// Khai báo mảng rỗng
let arr1 = [];
console.log(arr1);
// Khai báo mảng chứa nhiều loại giá trị
let arr2 = [1, 2, "three", true];
console.log(arr2);
```

▶ []

▶ (4) [1, 2, 'three', true]

1

Mảng – array JS

□ 3. Khai báo mảng :

Cách 2: Sử Dụng Array và Constructor:

```
// Khai báo mảng rỗng sử dụng Array constructor
let arr3 = new Array();
console.log(arr3);
// Khai báo mảng chứa nhiều loại giá trị sử dụng Array constructor
let arr4 = new Array(1, 2, "three", true);
console.log(arr4);
// Tạo mảng với độ dài xác định, các phần tử là undefined
let arrayWithLength = new Array(5);
console.log(arrayWithLength);
```

▶ []

▶ (4) [1, 2, 'three', true]

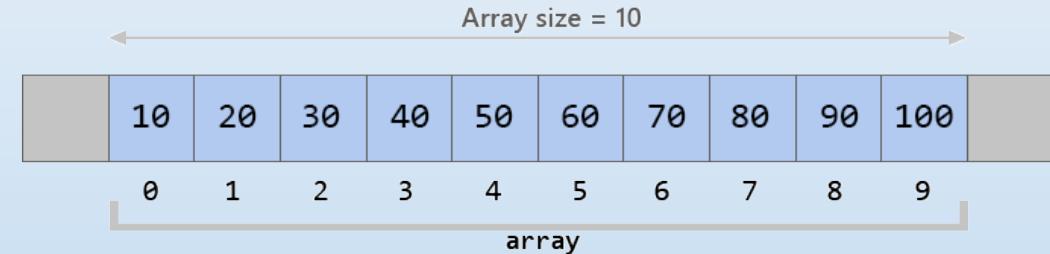
▶ (5) [empty × 5]

1

Mảng – array JS

□ 4. Truy xuất phần tử của mảng :

```
//4. Truy xuất phần tử của mảng qua vị trí index  
let arr5 = [10, 20, 30, 40];  
console.log(arr5[0]); //10  
console.log(arr5[2]); //30
```



□ 5. Thuộc tính length : trả về số phần tử của mảng (chiều dài mảng, bắt đầu từ 1)

```
console.log(arr5.length); // 4
```

□ 6. Gán, thay đổi giá trị cho mảng qua index

```
let arr6 = [5, 6, 7];  
console.log("arr6 trước khi thay đổi: ");  
console.log(arr6); // 5 6 7  
arr6[1] = 10;  
console.log("arr6 trước sau đổi: ");  
console.log(arr6); // 5 10 7
```

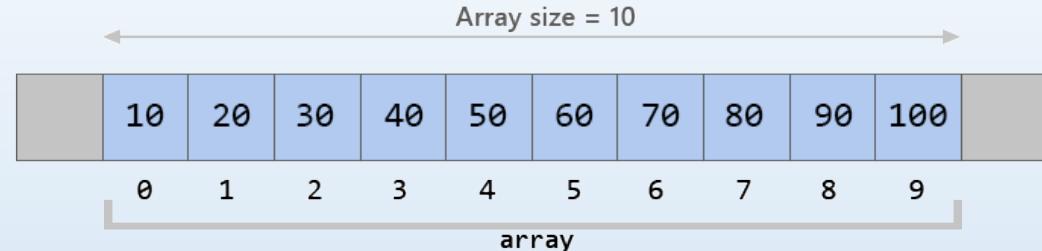
1

Mảng – array JS

□ 7. Duyệt mảng :

```
// 7.1 Cách 1 dùng vòng lặp for (có thể xem và sửa giá trị mảng)
let arr7 = [1, 2, 3, 4, 5];
console.log(arr7); //Output : [1, 2, 3, 4, 5]
for (let i = 0; i < arr7.length; i++) {
    // Sửa giá trị của phần tử tại chỉ số i
    arr7[i] = arr7[i] * 2;
}
console.log(arr7); //Output : [2, 4, 6, 8, 10]

// 7.2 for...of : Chỉ có thể xem, không sửa được giá trị của mảng
let count = 0;
let arr8 = [1, 2, 3, 4, 5];
for (let element of arr8) {
    // Xem được
    console.log(element);
    //Check điều kiện với từng pt, Đếm số phần tử chia hết cho 2
    if (element % 2 === 0) {
        count++;
    }
}
console.log("Số phần tử chia hết cho 2 của mảng: " + count);
```





JAVASCRIPT SIÊU TỐC

FULL



JavaScript Array

Array Methods

01

<http://Tuhoc.CC>

2

Các phương thức array JS

| STT | Phương thức | Nội dung |
|-----|-------------|---|
| 1 | concat () | <i>Nối mảng hiện tại với mảng khác và trả về một mảng mới.</i> |
| 2 | push () | <i>Thêm một hoặc nhiều phần tử vào cuối mảng gốc.</i> |
| 3 | unshift () | <i>Thêm một hoặc nhiều phần tử vào đầu mảng gốc.</i> |
| 4 | pop () | <i>Loại bỏ phần tử cuối cùng của mảng và trả về phần tử đã bị loại bỏ.</i> |
| 5 | shift () | <i>Loại bỏ phần tử đầu tiên của mảng và trả về phần tử đã bị loại bỏ.</i> |
| 6 | slice () | <i>Tạo một bản sao của mảng → lưu sang 1 vùng nhớ mới</i> |
| 7 | includes () | <i>Kiểm tra xem một mảng có chứa một giá trị cụ thể hay không. Trả về true nếu có và false nếu không.</i> |

2

Các phương thức array JS

- 1. **concat()**: Nối mảng hiện tại với mảng khác và trả về một mảng mới.

```
let arr1 = [1, 2, 3];
let arr2 = [4, 5, 6];
let newArr = arr1.concat(arr2);
console.log(newArr); // Output: [1, 2, 3, 4, 5, 6]
```

- 2. **push()**: Thêm một hoặc nhiều phần tử vào cuối mảng gốc.

```
let arr3 = [1, 2, 3];
arr3.push(4);
console.log(arr3); // Output: [1, 2, 3, 4]
```

- 3. **unshift()**: Thêm một hoặc nhiều phần tử vào đầu mảng gốc.

```
let arr4 = [2, 3];
arr4.unshift(1);
console.log(arr4); // Output: [1, 2, 3]
```

2

Các phương thức array JS

- 4. *pop()*: Loại bỏ phần tử cuối cùng của mảng (*làm thay đổi mảng gốc*) và trả về phần tử đã bị loại bỏ.

```
let arr5 = [1, 2, 3];
let removedElement = arr5.pop();
console.log(arr5); // Output: [1, 2]
console.log(removedElement); // Output: 3
```

- 5. *shift()*: Loại bỏ phần tử đầu tiên của mảng và trả về phần tử đã bị loại bỏ.

```
let arr6 = [1, 2, 3];
let removedElement2 = arr6.shift();
console.log(arr6); // Output: [2, 3]
console.log(removedElement2); // Output: 1
```

2

Các phương thức array JS

6. ***slice(sratIndex, endIndex)***: Tạo một bản sao của mảng → lưu sang 1 vùng nhớ mới
Cắt chuỗi gốc, lấy từ ***startIndex*** đến sát ***endIndex*** chuỗi gốc (không bao ***endIndex***).
Bỏ trống sê hieu sao chép lấy toàn bộ mảng gốc sang mảng mới

```
let arr7 = [1, 2, 3, 4, 5];
let slicedArr = arr7.slice(1, 4); //cắt từ index 1 đến sát 4
console.log(slicedArr); // Output: [2, 3, 4]
```

7. ***includes()***: Kiểm tra xem một mảng có chứa một giá trị cụ thể hay không.
Trả về ***true*** nếu có và ***false*** nếu không.

```
let arr8 = [1, 2, 3, 4, 5];
let isPresent = arr8.includes(3);
console.log(isPresent); // Output: true
```



JAVASCRIPT SIÊU TỐC

FULL



JavaScript Array

Array Methods

02

<http://Tuhoc.CC>

2

Các phương thức array JS

| STT | Phương thức | Nội dung |
|-----|-----------------|---|
| 8 | indexOf() | <p>Trả về vị trí index của phần tử xuất hiện lần đầu tiên trong mảng nếu tìm thấy.</p> <p>Trả về -1 nếu không tìm thấy.</p> |
| 9 | lastIndexOf() | <p>Trả về vị trí index của phần tử xuất hiện lần cuối cùng trong mảng nếu tìm thấy.</p> <p>Trả về -1 nếu không tìm thấy.</p> |
| 10 | reverse() | <p>Đảo ngược thứ tự của các phần tử trong mảng gốc.</p> <p>Nó thay đổi mảng gốc và không tạo ra mảng mới</p> |
| 11 | join(separator) | <p>Nối các phần tử trong mảng thành chuỗi, theo ký tự phân tách 'separator'</p> <p>separator : Nếu bỏ trống sẽ mặc định là dấu ,</p> <p>Phương thức này không thay đổi mảng gốc.</p> |

2

Các phương thức array JS

- 8. *indexOf()*: *Trả về vị trí index của phần tử xuất hiện lần đầu tiên trong mảng nếu tìm thấy.*
Trả về -1 nếu không tìm thấy.

```
let arr9 = [1, 2, 3, 4, 2, 5];
let index = arr9.indexOf(2);
console.log(index); // Output: 1
```

- 9. *lastIndexOf()*:

Trả về vị trí index của phần tử xuất hiện lần cuối cùng trong mảng nếu tìm thấy.
Trả về -1 nếu không tìm thấy.

```
let arr10 = [1, 2, 3, 4, 2, 5];
var lastIndex = arr10.lastIndexOf(2);
console.log(lastIndex); // Output: 4
```

2

Các phương thức array JS

- 10. **reverse()**: đảo ngược thứ tự của các phần tử trong mảng gốc.
Nó **thay đổi mảng gốc và không tạo ra mảng mới**.

```
let arr11 = [1, 3, 5, 7, 9];
arr11.reverse();
console.log(arr11); // Output: [9, 7, 5, 3, 1]
```

Nếu bạn muốn **giữ nguyên mảng gốc** và tạo ra một bản sao đảo ngược, bạn có thể sử dụng **phương thức slice()** để tạo một bản sao và sau đó sử dụng **reverse()** trên bản sao đó

```
let arr12 = [1, 3, 5, 7, 9];
let arr13 = arr12.slice().reverse();
console.log(arr12); // Output: [1, 3, 5, 7, 9]
console.log(arr13); // Output: [9, 7, 5, 3, 1]
```

Sao chép toàn bộ mảng gốc

2

Các phương thức array JS

11. array.**join(separator);**

Nối các phần tử trong mảng thành chuỗi, theo ký tự phân tách 'separator'

separator : Nếu bỏ trống sẽ mặc định là dấu ,

Phương thức này không thay đổi mảng gốc.

```
// Không truyền separator
let arr14 = ["Nam", "Trung", 1, 2, 3];
let string1 = arr14.join();
console.log(string1); // Output "Nam,Trung,1,2,3"
console.log(typeof string1); // string

// Có truyền separator
let arr16 = ["Nam", "Trung", 1, 2, 3];
let string2 = arr16.join(" + ");
console.log(string2); // Output "Nam + Trung + 1 + 2 + 3"
```



JAVASCRIPT SIÊU TỐC

FULL



32.4

JavaScript Array

splice ()



<http://Tuhoc.CC>

2

Các phương thức array JS

□ 12. *splice()* là một công cụ mạnh mẽ cho việc thay đổi cấu trúc của mảng

array.**splice(start, deleteCount, item1, item2, ...);**

- ✓ *start:* Chỉ định vị trí bắt đầu thay đổi mảng. Nếu là một số âm, nó sẽ được tính từ cuối mảng.
- ✓ *deleteCount:* Số nguyên chỉ định số lượng phần tử sẽ bị loại bỏ từ mảng, bắt đầu từ vị trí start. (*Nếu deleteCount là 0, không có phần tử nào bị loại bỏ.*)
- ✓ *item1, item2, ...:* Các phần tử mới sẽ được thêm vào mảng từ vị trí start

```
// 12.1 Loại bỏ phần tử từ mảng:  
let arr11 = [1, 2, 3, 4, 5];  
arr11.splice(2, 1); // Loại bỏ 1 phần tử từ vị trí 2  
console.log(arr11); // Output: [1, 2, 4, 5]
```

2

Các phương thức array JS

□ 12. *splice()* là một công cụ mạnh mẽ cho việc thay đổi cấu trúc của mảng

array.**splice(start, deleteCount, item1, item2, ...);**

- ✓ *start:* Chỉ định vị trí bắt đầu thay đổi mảng. Nếu là một số âm, nó sẽ được tính từ cuối mảng.
- ✓ *deleteCount:* Số nguyên chỉ định số lượng phần tử sẽ bị loại bỏ từ mảng, bắt đầu từ vị trí start. (*Nếu deleteCount là 0, không có phần tử nào bị loại bỏ.*)
- ✓ *item1, item2, ...:* Các phần tử mới sẽ được thêm vào mảng từ vị trí start

//12.2 Thay thế phần tử trong mảng:

```
let arr12 = [1, 2, 3, 4, 5];
arr12.splice(2, 1, 6); // Loại bỏ 1 phần tử từ vị trí 2 và thêm phần tử 6 vào vị trí đó
console.log(arr12); // Output: [1, 2, 6, 4, 5]
```

//12.3 Thêm phần tử vào mảng:

```
let arr13 = [1, 2, 3, 4, 5];
arr13.splice(2, 0, 6); // Không loại bỏ phần tử nào và thêm phần tử 6 vào vị trí 2
console.log(arr13); // Output: [1, 2, 6, 3, 4, 5]
```



JAVASCRIPT
SIÊU TỐC

FULL



32.5

JavaScript Array

Toán Tử Spread



<http://Tuhoc.CC>

3

Toán tử spread (...)

□ 13. **Toán tử spread (...)** là một toán tử mới được thêm vào từ phiên bản ES6. Spread cho phép duyệt qua lần lượt các phần tử → Và qua đó ta có thể thực hiện :

- a. Tạo bản sao (clone) của mảng:
- b. Truyền đối số vào hàm:
- c. Kết hợp mảng (nối mảng):
- d. Tạo mảng mới với thêm phần tử:
- e. Chuyển đổi iterable (có thể duyệt qua: string, arr) thành mảng:

a. Tạo bản sao (clone) của mảng:

Tạo ra mảng mới nằm trên ô nhớ mới, có phần tử giống hệt mảng gốc

```
let M1 = [1, 2, 3];
let M2 = [...M1];
console.log(M2); // Output: [1, 2, 3]
// Mảng M2 là clone của M1 ,
// và thay đổi giá trị của M2 không liên quan đến M1
M2[0] = 99;
console.log("Mảng M2 = " + M2); // Mảng M2 = 99,2,3
console.log("Mảng M1 = " + M1); // Mảng M1 = 1,2,3
```

3

Toán tử spread (...)

b. Truyền đối số vào hàm

```
function sum(a, b, c) {  
  return a + b + c;  
}  
  
let numbers = [1, 2, 3];  
let result = sum(...numbers);  
console.log(result); // Output: 6
```

c. Kết hợp mảng

```
let arr1 = [1, 2, 3];  
let arr2 = [4, 5, 6];  
let combinedArray = [...arr1, ...arr2];  
console.log(combinedArray); // Output: [1, 2, 3, 4, 5, 6]
```

3

Toán tử spread (...)

- d. Tạo mảng mới với thêm phần tử:

```
let M3 = [1, 2, 3];
let M4 = [...M3, 4];
console.log(M4); // Output: [1, 2, 3, 4]
```

- e. Chuyển đổi iterable thành mảng:

```
let myName = "jacky";
let chars = [...myName];
console.log(chars); // Output: ['j', 'a', 'c', 'k', 'y']
```



YouTube Gà Lại Lập Trình

WWW <http://tuhoc.cc>

JAVASCRIPT SIÊU TỐC

FULL



JavaScript Array

sort ()

32.6



<http://Tuhoc.CC>

4

Phương thức sort()

□ 14. *sort(...)*

Cú pháp: *array.sort([compareFunction])*

array: Mảng cần được sắp xếp.

compareFunction (Tùy chọn): Hàm so sánh được sử dụng để xác định thứ tự sắp xếp.

Nếu không được cung cấp, sort() sẽ sắp xếp các phần tử dưới dạng chuỗi Unicode.

Hàm So sánh (Compare Function):

1. Nếu compareFunction được cung cấp, nó sẽ nhận hai đối số, thường được gọi là *a* và *b*.
2. Nếu *compareFunction(a, b)* trả về một giá trị < 0 , *a* sẽ được đặt trước *b*.
3. Nếu *compareFunction(a, b)* trả về 0 , thứ tự giữa *a* và *b* không thay đổi.
4. Nếu *compareFunction(a, b)* trả về một giá trị > 0 , *b* sẽ được đặt trước *a*.

4

Phương thức sort()

Cú pháp: `array.sort([compareFunction])`

- 14.1 : Khi không truyền `compareFunction` : So sánh lần lượt từ ký tự đầu, đến các ký tự phía sau (Nếu các ký tự khác nhau sẽ dùng so sánh → Sắp xếp theo thứ tự tăng dần dựa theo thứ tự trong bảng mã UNICODE)

```
let M1 = ["b", "a", "c"];
// Để xem giá trị unicode dùng charCodeAt()
console.log("b".charCodeAt());
// Duyệt mảng để xem
for (let element of M1) {
  console.log(` ${element} có mã unicode là: ${element.charCodeAt()}`);
}
/*
Đối với mảng này, giá trị Unicode của các ký tự là:
"b": 98
"a": 97
"c": 99
*/
let sortedM1 = M1.sort();
console.log(sortedM1); // Output: ["a", "b", "c"]
```

4

Phương thức sort()

Cú pháp: `array.sort([compareFunction])`

- 14.1 : Khi không truyền `compareFunction` : So sánh lần lượt từ ký tự đầu, đến các ký tự phía sau (Nếu các ký tự khác nhau sẽ dùng so sánh → Sắp xếp theo thứ tự tăng dần dựa theo thứ tự trong bảng mã UNICODE)

```
// Trường hợp 2: phần tử có nhiều ký tự: So sánh các ký tự đầu để xếp
// Nếu ký tự giống nhau, so tiếp đến ký tự phía sau, mã unicode thấp hơn xếp trước
let M2 = ["baa", "a", "c"];
let sortedM2 = M2.sort();
console.log(sortedM2);
//Output : ['a', 'baa', 'c']
```

// Ví dụ với ký tự số (vì xếp theo unicode nên kq không như mong muốn)

```
let M3 = [10000, 1, 9];
let sortedM3 = M3.sort();
console.log(sortedM3); // [1, 10000, 9] → Kết quả không như mong đợi
console.log("1".charCodeAt()); // 49
console.log("9".charCodeAt()); // 57
console.log("0".charCodeAt()); // 48
```

4

Phương thức sort()

Cú pháp: `array.sort([compareFunction])`□ 14.2 Sử dụng với hàm so sánh `compareFunction`**Hàm So sánh (Compare Function):**

1. Nếu `compareFunction` được cung cấp, nó sẽ nhận hai đối số, thường được gọi là `a` và `b`.
2. Nếu `compareFunction(a, b)` trả về một giá trị `< 0`, `a` sẽ được đặt trước `b`.
3. Nếu `compareFunction(a, b)` trả về `0`, thứ tự giữa `a` và `b` không thay đổi.
4. Nếu `compareFunction(a, b)` trả về một giá trị `> 0`, `b` sẽ được đặt trước `a`.

```
let M4 = [9, 3, 5, 4];
// Trước khi sắp xếp
console.log(M4); // Output: [9, 3, 5, 4]
// Sau khi sắp xếp tăng dần
let sortedM4 = M4.sort((a, b) => a - b);
console.log(sortedM4); // Output: [3, 4, 5, 9]

// Sau khi sắp xếp giảm dần
let sortedM5 = M4.sort((a, b) => b - a);
console.log(sortedM5); // Output: [9, 5, 4, 3]
```



JAVASCRIPT SIÊU TỐC

FULL



JavaScript Array

reduce ()

<http://Tuhoc.CC>

5

Phương thức reduce()

15. Phương thức *reduce()*

Dựa trên một hàm xử lý → tính toán và trả về một giá trị duy nhất sau tính toán

Cú pháp: `array.reduce(function , [initialValue])`

1. *function*: Một hàm để thực thi cho từng phần tử trong mảng
2. *initialValue* : Giá trị khởi tạo

`function(accumulator, currentValue, [currentIndex], [array])`

Hàm được gọi với những đối số sau:

1. *accumulator*: Giá trị tích lũy, được cập nhật sau mỗi lần gọi hàm .
2. *currentValue*: Giá trị hiện tại đang xử lý trong mảng.
3. *currentIndex*: (Tùy chọn) Chỉ số của phần tử đang xử lý.
4. *array*: (Tùy chọn) Mảng đang được reduce.

5

Phương thức reduce()

// Bài toán tính tổng các phần tử của mảng sử dụng vòng lặp for

```
let M1 = [1, 2, 3];
// Thực hiện tính tổng các phần tử trong mảng
// 1. Giá trị khởi tạo ban đầu
let sum = 0;
for (let element of M1) {
    // Giá trị lưu trữ: cộng dồn
    sum += element;
}
console.log("Tổng các phần tử của mảng:" + sum); // 6
```

5

Phương thức reduce()

Cú pháp: array.reduce(function , [initialValue])

// Bài toán tính tổng các phần tử của mảng dùng reduce()

```
let sum2 = M1.reduce(  
    // Tham số thứ 1: function  
    (accumulator, currentValue, currentIndex, array) => {  
        return currentValue + accumulator;  
    },  
    0  
    /*  
     * Tham số thứ 2: initialValue giá trị khởi tạo của accumulator ban đầu,  
     * 1. Nếu bỏ trống initialValue:  
         a. accumulator sẽ lấy giá trị đầu tiên trong mảng làm giá trị khởi tạo,  
             và sẽ bắt đầu thực hiện từ phần tử thứ 2  
         b. Nếu mảng trống mà 0 có giá trị initialValue --> reduce() Sẽ báo lỗi  
    */  
);
```

```
// Rút gọn lại  
let sum3 = M1.reduce(  
    (accumulator, currentValue) => accumulator + currentValue,  
    0  
);  
console.log(sum3);
```



reduce() js mdn



6

Phương thức filter()

16. Phương thức **filter()** : Trích lọc các phần tử thỏa mãn điều kiện của hàm

Dựa trên một hàm xử lý → để tạo ra một mảng mới từ một mảng đã cho, chỉ chứa các phần tử thỏa mãn một điều kiện nhất định được xác định bởi hàm

Cú pháp: **array.filter(function)**

1. **function:** Một hàm để thực thi cho từng phần tử trong mảng

function(currentValue, [currentIndex] , [array])

Hàm được gọi với những đối số sau:

- a. **currentValue:** Giá trị hiện tại đang xử lý trong mảng.
- b. **currentIndex:** (Tùy chọn) Chỉ số của phần tử đang xử lý.
- c. **array:** (Tùy chọn) Mảng đang được duyệt.

6

Phương thức filter()

Cú pháp: array.filter(function)

1. **function:** Một hàm để thực thi cho từng phần tử trong mảng

function(currentValue, [currentIndex] , [array])

Hàm được gọi với những đối số sau:

- a. **currentValue:** Giá trị hiện tại đang xử lý trong mảng.
- b. **currentIndex:** (Tùy chọn) Chỉ số của phần tử đang xử lý.
- c. **array:** (Tùy chọn) Mảng đang được duyệt.

```
let numbers = [1, 2, 3, 4, 5];
// Tìm những số chẵn trong mảng
let evenNumbers = numbers.filter(function (num) {
  return num % 2 === 0;
});
console.log(evenNumbers); // Output: [2, 4]
```



JAVASCRIPT SIÊU TỐC

FULL

**33.1**

JavaScript Array

Bài tập thực hành

32 - 33<http://Tuhoc.CC>

1

Bài tập js 32 - 33

Bài tập js 32:

Viết chương trình tạo 1 mảng 1 chiều gồm các phần tử là số nguyên có n phần tử, n do người dùng nhập từ bàn phím

Bài tập js 33:

- ✓ 1. *Viết chương trình tạo 1 mảng 1 chiều gồm các phần tử là số nguyên, có n phần tử ngẫu nhiên, n do người dùng nhập từ bàn phím*
- ✓ 2. *Xuất các giá trị trong mảng*
- ✓ 3. *Đảo ngược mảng, và xuất mảng sau khi đảo ngược*
- ✓ 4. *Sắp xếp mảng tăng dần*
- ✓ 5. *Tính tổng các phần tử trong mảng*
- ✓ 6. *Cho người dùng nhập 1 số bất kỳ, kiểm tra số đó có tồn tại trong mảng hay không, nếu có xuất ra vị trí index của số đó trong mảng*



JAVASCRIPT SIÊU TỐC

FULL



JavaScript Array

Giải bài tập JS 32







JAVASCRIPT SIÊU TỐC

FULL



34.1

JavaScript Objects

Literal syntax



<http://Tuhoc.CC>

1

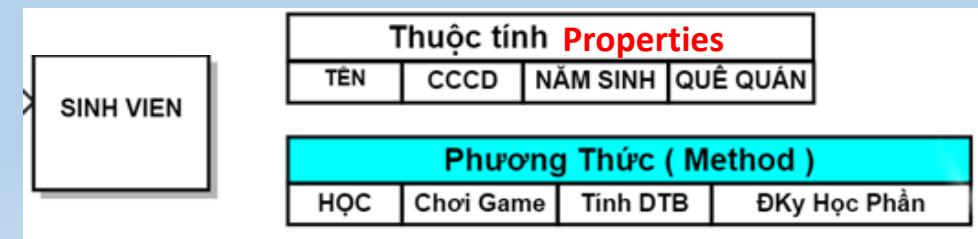
Tổng quan về object

- ✓ Chúng ta đã được học về **biến** để lưu trữ dữ liệu cụ thể: **number, string, boolean,..**
- ✓ **array** để lưu trữ 1 tập hợp phần tử, ví dụ: 20 giá trị là tên của học sinh → **Bản chất array có kiểu dữ liệu là object, do JS đã xây dựng sẵn**
- ✓ Ta cũng có thể tự xây dựng object theo quy tắc do mình đặt ra → Ví dụ 1 trường học có 1000 sinh viên, mỗi sinh viên lại có các thông tin: **Tên, ngày tháng năm sinh, Quê Quán, số căn cước, Điểm tốt nghiệp,...**
→ **Object** ra đời để đảm nhiệm việc này
- ✓ **Object** là một cấu trúc dữ liệu mạnh mẽ, cho phép bạn lưu trữ và tổ chức thông tin dưới dạng cặp **key-value**.
- ✓ **Đối tượng (object)** trong lập trình hướng đối tượng giống như 1 đối tượng cụ thể trong thế giới thực

Mỗi đối tượng có thuộc tính và phương thức riêng

+ **Thuộc tính :** Đặc điểm của đối tượng

+ **Phương thức:** Hành vi của đối tượng



2

Khai báo object

1. Literal syntax:

// Sử dụng cặp dấu {} để định nghĩa 1 đối tượng mới với các cặp key value

```
let student = {
    // Thuộc tính (ta có thể nói student có 4 thuộc tính)
    // key có quy tắc giống với quy tắc đặt tên biến
    fullName: "Tran Nhu Nhong",
    birthYear: 2005,
    address: {
        city: "Hanoi",
        country: "Vietnam",
    },
    scores: [8, 9, 10],
    // Phương thức
    // VD1: Phương thức để lấy tuổi
    getAge: function () {
        return 2024 - this.birthYear;
    },
    // VD2. Phương thức để tính điểm trung bình
    diemTrungBinh: function () {
        // Tính tổng điểm
        sumScores = this.scores.reduce((a, b) => a + b, 0);
        // Trả về điểm trung bình
        return sumScores / 3;
    },
};
```

- ✓ *Mỗi đối tượng có thuộc tính và phương thức riêng*
 - + *Thuộc tính : Đặc điểm của đối tượng (Biến)*
 - + *Phương thức: Hành vi của đối tượng (Hàm)*

1

Khai báo object

1. Literal syntax:

Có thể khai báo key dưới dạng chuỗi

(key có thể chứa các ký tự đặc biệt bất kỳ, thậm chí có thể vi phạm nguyên tắc đặt tên biến)

```
let teacher = {  
    "1 fullName": "Jacob",  
    "@address": {  
        city: "Hanoi",  
        country: "Vietnam",  
    },  
};
```



JAVASCRIPT



SIÊU TỐC



34.2

JavaScript Objects

Dot Notation và Bracket Notation

<http://Tuhoc.CC>

2

Truy Cập Thuộc Tính :

Dot Notation . và Bracket Notation []

2.1 Sử dụng dot (.) - Dot Notation:

```
console.log(student.fullName); // "Tran Nhu Nhong"
console.log(student.scores); // [8, 9, 10]
console.log(student.scores[1]); // 9
```

2.2 Sử dụng [] - Bracket Notation - và truyền vào giá trị của key

Bracket Notation dùng khi key đặc biệt - có khoảng trắng, vi phạm quy tắc đặt tên biến....

```
console.log(teacher["1 fullName"]);
console.log(student["fullName"]); // "Tran Nhu Nhong"
```

2

Truy Cập Thuộc Tính :

Dot Notation . và Bracket Notation []

2.3 Linh động sử dụng [] - Bracket Notation với biến:

```
// Linh động sử dụng [] với biến
let inputKey = prompt("Mời nhập key: (address,scores)");
// let inputKey = "address";
console.log("Truy xuất đến key dùng biến + []");
console.log(student[inputKey]);
// Nếu truy xuất đến 1 key không tồn tại sẽ trả về undefined
// Ví dụ nhập vào key : job --> undefined
```

```
// Ví dụ : Kiểm tra xem nếu nhập key không tồn tại thì báo lỗi
// Xem lại bài 18: Truthy and Falsy Values
if (student[inputKey]) {
    console.log(student[inputKey]);
} else {
    console.log("inputKey bạn nhập không tồn tại");
}
```



JAVASCRIPT SIÊU TỐC

FULL



34.3

JavaScript Objects

Truy cập phương thức



<http://Tuhoc.CC>

3

Truy Cập Phương Thức:

3.1 Truy cập phương thức :

```
// 3.1 Truy cập phương thức
console.log("Tuổi:");
console.log(student.getAge());
console.log(student.diemTrungBinh());
```

3

Truy Cập Phương Thức:

3.2 Vấn đề: Giả sử trong chương trình bạn cần gọi **nhiều lần hàm getAge**
→ Chương trình của bạn sẽ **mất nhiều lần tính toán** do hàm bị gọi đi gọi lại

```
console.log(student.getAge());
console.log(student.getAge());
console.log(student.getAge());
console.log(student.getAge());
```

→ Để giảm tải, tăng tốc cho chương trình, sử dụng mèo gán giá trị trả về vào thuộc tính
→ Hàm chỉ cần gọi 1 lần

```
// Phương thức viết lại để trả về 1 thuộc tính
getAge2: function () {
    // sử dụng this.ten_thuoc_tinh để thêm thuộc tính vào đối tượng
    this.age = 2024 - this.birthYear;
    return this.age;
},
```

```
// Tuổi chỉ thực hiện tính 1 lần --> gọi hàm 1 lần
student.getAge2();
// Xuất giá trị của thuộc tính age
console.log(student.age);
console.log(student.age);
console.log(student.age);
```



JAVASCRIPT SIÊU TỐC

FULL



34.4

JavaScript Objects

Thêm và xóa thuộc tính



<http://Tuhoc.CC>

4

Thêm, xóa thuộc tính

```
// 4. Thêm, sửa, xóa thuộc tính (thêm cặp key value)
student.email = "tuhoc.cc@gmail.com"; // Thêm mới thuộc tính
student["website"] = "http://tuhoc.cc"; // Thêm mới thuộc tính

// Xuất đối tượng để xem
console.log(student);
console.log(student.email);
console.log(student.website);

// Xóa bỏ thuộc tính
delete student.email;
console.log(student); //đã mất thuộc tính email

// Sửa thuộc tính
student.website = "giá trị đã bị sửa";
console.log(student);
```