



JavaScript



Buổi 2: Làm quen với JavaScript (tiếp).

Nội dung buổi 2

- Giá trị Truthy và Falsy.
- Boolean Logic. Nullish Coalescing.
- Short Circuiting.
- Toán tử ba ngôi.
- switch/case.
- Hàm (Function).
- Làm việc với kiểu dữ liệu số.

Truthy và Falsy



Truthy và Falsy

- Tất cả các giá trị trong JavaScript đều có thể quy về kiểu Boolean (true và false).
- Có thể kiểm tra một giá trị là Truthy hay Falsy qua mệnh đề điều kiện hoặc hàm Boolean().

```
// TRUTHY
Boolean(1); // true
Boolean(-1); // true
Boolean("abc"); // true
Boolean([1, 2, 3]); // true
Boolean([]); // true
Boolean({ type: "Object" }); // true
Boolean({}); // true
```

```
// FALSY
Boolean(0); // false
Boolean(""); // false
Boolean(undefined); // false
Boolean(null); // false
Boolean(NaN); // false
```

Boolean Logic



Boolean Logic

So sánh đồng thời nhiều điều kiện với cú pháp “&&”:

```
const age = 27;
const motorLicense = true;

if (age >= 18 && motorLicense > 8) {
  console.log("Được phép lái xe máy");
}
```

So sánh theo kiểu hoặc có điều kiện này, hoặc có điều kiện kia:

```
const math = 7;
const literature = 8;
const averageGrade = 8.2;

if (averageGrade >= 8 && (math >= 8 || literature >= 8)) {
  console.log("Đạt học sinh giỏi");
}
```

Chú ý: Khi sử dụng đồng thời “&&” và “||” thì điều kiện với “&&” sẽ được ưu tiên hơn, do đó cần phải bọc các vế của điều kiện hoặc trong cặp ngoặc đơn để đảm bảo tính ưu tiên của điều kiện hoặc.

Boolean Logic

Phủ định giá trị với cú pháp “!”:

```
!true; // false  
!false; // true  
!"abc"; // false  
!23; // false  
!0; // true  
!undefined; // true
```

2 cách để quy đổi một giá trị về true hoặc false:

```
Boolean(undefined); // false  
!!undefined; // false
```

Short Circuiting

Short Circuiting

Cú pháp “&&”: Nếu tất cả các điều kiện đều thỏa mãn, trả về kết quả là giá trị ở vị trí cuối cùng của chuỗi quan hệ. Nếu có giá trị trả về giá trị kiểu Falsy thì lập tức trả về giá trị tại vị trí đó và không tiếp tục kiểm tra các điều kiện ở phía sau.

```
true && "abc"; // "abc"
true && "abc" && 3; // 3
false && "abc"; // false
true && 1 > 2 && 3; // false
true && 0 && 3; // 0
undefined && 3; // undefined
```

Cú pháp “||”: Nếu gặp giá trị trả về giá trị kiểu Truthy, lập tức trả về kết quả đó và ngừng kiểm tra. Nếu gặp giá trị kiểu Falsy, tiếp tục kiểm tra giá trị ở vị trí tiếp theo. Nếu tất cả giá trị đều trả về giá trị kiểu Falsy thì trả về giá trị ở vị trí cuối cùng của chuỗi quan hệ.

```
3 || "2"; // 3
0 || "2"; // "2"
undefined || 2 > 1 || null; // true
0 || undefined || null; // null
```

Nullish Coalescing Operator

Cú pháp “??” hoạt động tương tự như “||” nhưng nó cho phép nhận cả giá trị 0 và “”.

```
0 || 2; // 2
"" || "abc"; // "abc"
0 ?? 2; // 0
"" || "abc"; // ""
undefined ?? "defined"; // undefined
null ?? "not-null"; // null
```

Bài thực hành số 1



Bài thực hành số 1

Kết quả của những biểu thức sau?

1. `20 > 10 && 10 < 9;`

2. `20 > 10 || 10 < 9;`

3. `false || "abc";`

4. `"abc" && true;`

5. `20 || !false;`

6. `!!undefined && "correct";`

7. `!null || !"correct";`

8. `20 || 30 || "abc";`

9. `true && "" && "abc";`

10. `10 > 20 && ("a" === "a" || "b" === "c");`

11. `1 !== 2 && false || "abc";`

12. `1 !== 2 && (false || "abc");`

13. `1 !== 2 && !false || "abc";`

14. `1 !== 2 && false || !"abc";`

Toán tử ba ngôi



Toán tử ba ngôi

Cú pháp “??” hoạt động tương tự như “||” nhưng nó cho phép nhận cả giá trị 0 và “”.

```
const age = 10;  
const underAge = age >= 18 ? "Trưởng thành" : "Vị thành niên";  
// underAge === "Vị thành niên"
```

```
const underAge = age >= 18 ? "Trưởng thành" : "Vị thành niên";
```

Kiểm tra giá trị
Truthy, Falsy

Giá trị trả về nếu về
đầu tiên trả về Truthy

Giá trị trả về nếu về
đầu tiên trả về Falsy

Có thể sử dụng lồng các toán tử ba ngôi ở bên trong nhau.

```
const alcohol = true;  
const rate = 4.5;  
  
const baverage = alcohol ? (rate <= 6 ? "Beer" : "Wine") : "Pepsi";
```

switch/case



switch/case

```
const day = 0;
let result = "";

switch (day) {
  case 6:
    result = "Today is Saturday";
    break;
  case 0:
    result = "Today is Sunday";
    break;
  default:
    result = "Its not weekend yet!";
}
// result === "Today is Sunday"
```

```
const point = 9;
let result = "";

switch (true) {
  case point >= 9:
    result = "A";
    break;
  case point >= 7:
    result = "B";
    break;
  case point >= 4:
    result = "C";
    break;
  default:
    result = "Failed";
}
// result === "A"
```

Bài thực hành số 2



Bài thực hành số 2

- Bài 1: Viết lại cách tính chỉ số BMI và trả kết quả bằng switch/case.
- Bài 2: Trong bài tập so sánh tuổi trong buổi 1, hãy thêm 2 biến mới là "olderPerson" và "youngerPerson" để lưu giá trị tuổi theo tên người lớn tuổi nhất và người nhỏ nhất, sử dụng toán tử ba ngôi để gán giá trị cho 2 biến này. Kết quả chỉ cần báo là ai lớn hơn ai bao nhiêu tuổi. Không xét đến trường hợp bằng tuổi.
- Bài 3: Quy ước giá trị từ 0 - 6 đại diện cho các ngày trong tuần (0 - Chủ Nhật, 1 - Thứ Hai,...). Tạo một biến chứa giá trị số trong khoảng trên, sử dụng switch/case và mệnh đề điều kiện để in ra cửa sổ console giá trị theo biến trên theo dạng "Thứ Tư - Ngày trong tuần", "Thứ Bảy - Ngày cuối tuần"...

Hàm (Function)

Hàm (Function)

- Hàm (hay còn gọi là Function) là một khối lệnh (một nhóm các lệnh) được viết để nhằm hoàn thành một nhiệm vụ nào đó.
- Hàm chỉ được thực thi khi chúng ta tiến hành gọi hàm (call function).

```
function sayHello() {  
  console.log("Hello everyone!");  
}
```

Block được thực thi
nếu hàm được gọi

```
sayHello(); —————> Gọi hàm
```

```
// Cửa sổ console in giá trị "Hello everyone!"
```

Hàm (Function) - Khai báo hàm

Declarations:

```
sayHello(); → Có thể gọi hàm  
trước khi khai báo  
  
function sayHello() {  
  console.log("Hello everyone!");  
}
```

Expressions:

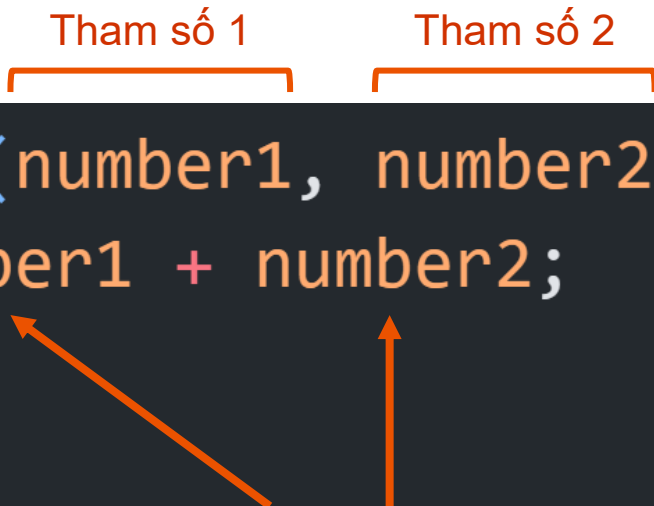
```
const sayHello = function () {  
  console.log("Hello everyone!");  
};  
  
sayHello(); → Chỉ được gọi hàm  
sau khi khai báo
```

Arrow function:

```
const sayHello = () => {  
  console.log("Hello everyone!");  
};  
  
sayHello(); → Chỉ được gọi hàm  
sau khi khai báo
```

Hàm (Function) - Tham số, return

- Hàm (hay còn gọi là Function) có thể nhận nhiều tham số, tên tham số không được trùng.
- Hàm không trả về (return) giá trị thì mặc định nó nhận giá trị là undefined.
- Khi hàm được gọi mà tham số nó nhận vào không đủ thì tham số bị thiếu đó sẽ có giá trị là undefined.



The diagram illustrates the mapping of arguments to parameters. Above the code, two orange brackets are labeled "Tham số 1" and "Tham số 2". Below, the function definition `const sum = (number1, number2) => { return number1 + number2; };` and the function call `const result = sum(1, 2);` are shown. An orange arrow points from the argument `1` in the call to the parameter `number1` in the definition. Another orange arrow points from the argument `2` in the call to the parameter `number2` in the definition.

```
const sum = (number1, number2) => {  
  return number1 + number2;  
};  
  
const result = sum(1, 2);  
// result === 3
```

Hàm (Function) - Tham số, return

- Trong một hàm, khối code ở phía sau return sẽ không được thực thi.

```
const player = "John";
let positon, number;

const assignNewMember = (playerName) => {
  if (!playerName) {
    return "There is no player";
  }

  positon = "striker";
  number = 8;
  return `${playerName} is a new ${positon} in number ${number} jersey`;
};
const result = assignNewMember(player);
// result === "John is a new striker in number 8 jersey"
```

Không thỏa mãn, bỏ qua block code điều kiện

```
const player = null;
let positon, number;

const assignNewMember = (playerName) => {
  if (!playerName) {
    return "There is no player";
  }

  positon = "striker";
  number = 8;
  return `${playerName} is a new ${positon} in number ${number} jersey`;
};
const result = assignNewMember(player);
// result === "There is no player"
```

Trả kết quả và bỏ qua nội dung code còn lại trong block code của hàm

Hàm (Function) - Nhận tham số là một hàm

```
function convertAndSumNumber(string1, string2, sumFunction) {  
  const number1 = Number(string1);  
  const number2 = Number(string2);  
  
  return sumFunction(number1, number2);  
}  
  
function sum(num1, num2) {  
  return num1 + num2;  
}  
  
const result = convertAndSumNumber("1", "2", sum);  
// result === 3
```

Bài thực hành số 3



Bài thực hành số 3

- Bài 1: Viết lại 3 bài tập ở Bài thực hành số 2 và áp dụng tham số thay cho các biến khai báo cố định hiện tại.
- Bài 2: Viết một hàm nhận tham số là giá trị nhiệt độ theo độ C và trả về kết quả là giá trị đó quy đổi sang độ F.
- Bài 3: Viết một hàm nhận 2 tham số với giá trị số, hàm trả về số có giá trị lớn hơn.
- Bài 4: Viết một hàm nhận một tham số bất kỳ và kiểm tra xem tham số đó là giá trị Truthy hay Falsy. Nếu là Truthy trả về giá trị của tham số đó, nếu là Falsy trả về undefined.

Bài thực hành số 3

- Bài 5: Viết một hàm nhận một tham số bất kỳ. Nếu tham số là một số, trả về kết quả là số đó. Nếu tham số không phải là số, quy đổi nó về dạng số và trả về kết quả là số đó. Nếu không thể quy đổi về số, trả kết quả về 0.
- Bài 6: Viết một hàm nhận 3 tham số là các giá trị số. Hàm sẽ kiểm tra tham số thứ ba có phải là tổng bình phương của tham số thứ nhất và thứ hai hay không. Trả về true hoặc false.

Làm việc với kiểu dữ liệu số



Quy đổi về giá trị số

Quy đổi thông thường:

```
Number("23"); // 23  
+"23"; // 23
```

Quy đổi về số nguyên:

```
Number.parseInt("15"); // 15  
Number.parseInt("15px"); // 15  
Number.parseInt("15.5px"); // 15  
Number.parseInt("px"); // NaN
```

Quy đổi về số thập phân:

```
Number.parseFloat("15"); // 15  
Number.parseFloat("15px"); // 15  
Number.parseFloat("15.5px"); // 15.5  
Number.parseFloat("px"); // NaN
```

Math

Làm tròn số:

```
// Làm tròn đúng  
Math.round(5.6); // 6  
Math.round(5.4); // 5  
  
// Làm tròn lên  
Math.ceil(1.1); // 2  
  
// Làm tròn xuống  
Math.floor(1.9); // 1
```

Làm tròn theo số thập phân:

```
(2.2).toFixed(); // '2'  
(2.2).toFixed(1); // '2.2'  
(2.2).toFixed(2); // '2.20'
```

Math

Tạo ra một số ngẫu nhiên:

```
Math.random(); // Trả về giá trị số từ 0 - < 1 ~ 0 - 0.9999...  
  
// Cách tạo ra một số ngẫu nhiên nằm trong khoảng từ 0 - 10.  
Math.round(Math.random() * 10);
```

Tìm giá trị lớn nhất, nhỏ nhất:

```
Math.min(1, 2, 3, 4); // 1  
Math.max(1, 2, 3, 4); // 4
```

Toán tử lấy phần dư sau khi chia:

```
6 % 2; // 0  
5 % 2; // 1  
7 % 5; // 2
```

Bài thực hành số 4



Bài thực hành số 4

- Bài 1: Viết một hàm nhận tham số là một số. Hàm trả về kết quả báo số đó là số chẵn hay số lẻ.
- Bài 2: Viết một hàm tính diện tích của hình tròn, tham số truyền vào là bán kính của hình tròn đó.
- Bài 3: Viết một hàm nhận giá trị là một số hoặc một chuỗi. Hàm chuyển đổi tham số về giá trị số (nếu cần) và trả về kết quả dưới dạng "8px", "10px"... Nếu tham số không thể chuyển đổi về dạng số, trả về "0px".
- Bài 4: Viết một hàm trả về một số ngẫu nhiên từ 3 - 8.

Hoàn thành JavaScript – Buổi 2

Good job!

