

# BÀI 02: KIỂU DỮ LIỆU, BIỂU THỨC VÀ CÁC TOÁN TỬ TRONG JAVASCRIPT

**Giảng Viên: ThS. Giang Hào Côn**

## 2.1./ Kiểu dữ liệu (Data Type) là gì?

- Trong lập trình, **kiểu dữ liệu** là khái niệm ám chỉ sự phân loại dữ liệu (**loại = kiểu**). Mỗi kiểu dữ liệu sẽ có chung một hay nhiều đặc điểm nhất định.
- **Trong JavaScript**, mỗi giá trị dữ liệu sẽ thuộc một nhóm kiểu dữ liệu nhất định.
- Ví dụ:

```
var a = "Lập Trình Web"; //giá trị của biến a thuộc kiểu dữ liệu string  
var b = 1993; //giá trị của biến b thuộc kiểu dữ liệu number  
var c = true; //giá trị của biến c thuộc kiểu dữ liệu boolean  
var d = false; //giá trị của biến d thuộc kiểu dữ liệu boolean  
var e = {name:"Nhân", age:24}; //giá trị của biến e thuộc kiểu dữ liệu object
```

## 2.1./ Kiểu dữ liệu (Data Type) là gì?

- Ví dụ:

```
<!DOCTYPE html>
<html>
<body>
  <script>
    var a = 22;
    var b = "22";
    var c = a + b;
    alert("Kết quả : " + c);
  </script>
</body>
</html>
```

An embedded page on this page says

Kết quả : 2222

OK

Việc hiểu rõ **bản chất một giá trị thuộc kiểu dữ liệu nào** và cách sử dụng kiểu dữ liệu đó ra sao là điều hết sức quan trọng. Vì trong JavaScript, ta phải thường xuyên thực hiện những biểu thức giữa các giá trị, việc nhầm lẫn kiểu dữ liệu sẽ khiến kết quả không như mong đợi.

## 2.1./ Kiểu dữ liệu (Data Type) là gì?

### **Có bao nhiêu kiểu dữ liệu trong Javascript ?**

Trong JavaScript, các kiểu dữ liệu được chia thành những loại cơ bản như sau:

- string
- number
- boolean
- object
- undefined
- array (đây là một trường hợp đặc biệt của kiểu dữ liệu object)

## 2.1.1/ Dữ liệu kiểu string

Trong JavaScript, các dữ liệu thuộc kiểu **string** (hay còn được gọi là "**chuỗi**") là **một tập hợp gồm các ký tự**, chúng được viết bên trong **cặp dấu nháy kép** hoặc **cặp dấu nháy đơn**.

### Ví dụ:

```
var a = "x"; // Dữ liệu của biến a được xếp vào kiểu string
```

```
var b = "1234"; // Dữ liệu của biến b được xếp vào kiểu string
```

```
var c = 'Hello world!'; // Dữ liệu của biến c được xếp vào kiểu string
```

```
const d = "Hôm nay tôi học lập trình. Thực ra học lập trình cũng rất thú vị chứ không khô khan như mọi người thường nghĩ! Ahihi  
-))))"; // Dữ liệu của hằng d được xếp vào kiểu string
```

## 2.1.1/ Dữ liệu kiểu string

### Đặc điểm nổi bật của dữ liệu kiểu string:

- Luôn được đặt trong cặp dấu nháy đôi ("...") hoặc nháy đơn ('...')
- Không thể dùng để tính toán cho dù giá trị có là chữ số
- Có thể dùng để nối với một dữ liệu khác cùng thuộc kiểu string.

### Ví dụ:

```
<!DOCTYPE html>
<html>
<body>

  <p id="demo1"></p>

  <p id="demo2"></p>

  <script>
    var a = "Tài liệu học HTML";
    var b = 'Lập Trình Web';
    document.getElementById("demo1").innerHTML = a;
    demo2.innerHTML = b;
  </script>

</body>
</html>
```

Tài liệu học HTML  
Lập Trình Web

## 2.1.2/ Dữ liệu kiểu number

Trong JavaScript, các dữ liệu thuộc kiểu **number** (hay còn được gọi là "**số**") là một tập hợp của các con số (0 - 9) không chứa dấu khoảng trắng và có thể chứa dấu trừ (-) nằm ở đầu để đại diện cho số âm.

**Ví dụ:**

```
var a = 100; // Dữ liệu của biến a được xếp vào kiểu number  
var b = -25.5; // Dữ liệu của biến b được xếp vào kiểu number
```

### **Đặc điểm nổi bật của dữ liệu kiểu number**

- Có thể dùng để tính toán (cộng, trừ, nhân, chia)

## 2.1.3/ Dữ liệu kiểu boolean

- Trong JavaScript, các dữ liệu thuộc kiểu **boolean** chỉ có thể nhận một trong hai giá trị, đó là: **true (đúng)**, **false (sai)**.
- Có hai cách để nhận giá trị kiểu boolean, đó là:
  - Gán giá trị trực tiếp.
  - Nhận được từ một điều kiện.

- **Ví dụ:**

```
<script>
  var a = true;
  var b = false;
  var c = 6 > 2;
  var d = 6 > 10;
</script>
```

- Giá trị của biến a là true.
- Giá trị của biến b là false.
- Giá trị của biến c là true, vì điều kiện  $(6 > 2)$  là đúng.
- Giá trị của biến d là false, vì điều kiện  $(6 > 10)$  là sai.



## 2.1.4/ Dữ liệu kiểu object

- Trong JavaScript, các dữ liệu thuộc kiểu **object** (hay còn được gọi là "**đối tượng**") là một tập hợp gồm **những cái tên** và **mỗi cái tên** sẽ chứa đựng một **giá trị dữ liệu**.
- Lưu ý: Những cái tên còn được gọi là "**thuộc tính**" của đối tượng, giá trị của những cái tên còn được gọi là "**giá trị thuộc tính của đối tượng**".

- **Ví dụ:**

```
<!DOCTYPE html>
<html>
<body>

<script>
  var SinhVien = { name:"Giang Hào Côn", gender:"Nam", year:1971 }

  document.write("Giá trị của thuộc tính name là: " + SinhVien.name);
  document.write("<hr>");
  document.write("Giá trị của thuộc tính gender là: " + SinhVien.gender);
  document.write("<hr>");
  document.write("Giá trị của thuộc tính year là: " + SinhVien.year);
</script>
</body>
</html>
```

Giá trị của thuộc tính name là: Giang Hào Côn

Giá trị của thuộc tính gender là: Nam

Giá trị của thuộc tính year là: 1971

## 2.1.4/ Dữ liệu kiểu array

- Trong JavaScript, **array** còn được gọi là mảng, nó là một trường hợp đặc biệt của đối tượng. (Thật ra, mảng có kiểu dữ liệu là object)
- Mảng là một loại biến đặc biệt có thể lưu trữ nhiều giá trị đồng thời, mỗi giá trị được gọi là một phần tử mảng.

- **Ví dụ:**

```
<!DOCTYPE html>
<html>
<body>

  <script>
    var mobile = ["HTC", "Nokia", "SamSung"];
    document.write(mobile[0] + "<hr>");
    document.write(mobile[1] + "<hr>");
    document.write(mobile[2] + "<hr>");
  </script>

</body>
</html>
```



HTC
Nokia
SamSung

## 2.1.4/ Dữ liệu kiểu undefined

- Trong JavaScript, khi **một biến được khai báo mà không gán giá trị** thì biến đó sẽ có giá trị là **undefined** và kiểu dữ liệu cũng là undefined.

- Ví dụ:**

```
<!DOCTYPE html>
<html>
<body>

  <script>
    var myName;
    document.write(myName);
  </script>

</body>
</html>
```

undefined

## 2.1.5/ Cách xác định kiểu của dữ liệu

- Để xác định kiểu của một dữ liệu nào đó thì ta sử dụng toán tử **typeof**.

- **Ví dụ:**

```
<script>
var a = typeof "";
var b = typeof "Lập Trình Web";
var c = typeof 1993;
var d = typeof true;
var e = typeof false;
var f = typeof {name:"Nhân", gender:"Nam", year:1993};
var g = typeof undefined;
var h = typeof ["HTC","Nokia","SamSung"];

document.write(a + "<hr>");
document.write(b + "<hr>");
document.write(c + "<hr>");
document.write(d + "<hr>");
document.write(e + "<hr>");
document.write(f + "<hr>");
document.write(g + "<hr>");
document.write(h + "<hr>");
</script>
```

string
string
number
boolean
boolean
object
undefined
object

## 2.2/ Biểu thức là gì ?

- Trong toán học, **biểu thức** là một tập hợp gồm các chữ số và các phép toán.
- Trong biểu thức, các chữ số được gọi là **toán hạng** còn các phép toán thì được gọi là **toán tử**
- Ví dụ: **10 + 5 - 2** là một biểu thức, trong đó **10, 5, 2** là toán hạng còn **+, -** là toán tử.
- Biểu thức trong JavaScript cũng là một tập hợp gồm các toán hạng và các toán tử.

## 2.2.1./ Toán hạng ?

- Điểm khác nhau giữa toán hạng trong JavaScript và toán hạng trong toán học chính là toán hạng trong JavaScript thì không nhất thiết phải là một chữ số, nó có thể là **một giá trị** hoặc **một biến** hoặc **một hàm**.
- Ví dụ, ở kịch bản bên dưới, giá trị của biến b là một biểu thức (trong đó biến a là một toán hạng).

```
<script>  
  var a = 50;  
  var b = a + 20; //Biến b sẽ có giá trị là 70  
</script>
```

## 2.2.2./ Toán tử ?

- Các toán tử trong JavaScript có chức năng giống với toán tử trong toán học.
- Tuy nhiên, một vài toán tử trong JavaScript có cách viết khác so với cách viết toán tử trong toán học.
- Bên dưới là danh sách những toán tử cơ bản nhất trong JavaScript:

Toán tử	Tên gọi	Ví dụ	Kết quả
+	Phép cộng	10 + 4	14
-	Phép trừ	10 - 4	6
*	Phép nhân	10 * 4	40
/	Phép chia	10 / 4	2.5
%	Phép chia lấy phần số dư	10 % 4	2

## 2.2.2./ Toán tử ?

Phép cộng trong JavaScript tương đối khác so với phép cộng trong toán học.  
Trong JavaScript:

- Số có thể cộng số => cho ra số.
- Số có thể cộng chuỗi (hoặc chuỗi có thể cộng số) => cho ra chuỗi.
- Chuỗi có thể cộng chuỗi => cho ra chuỗi.

12
2000 Vinh Kiệt
Vinh Kiệt 1993
Nguyễn Thị Bạch Liễu

```
<!DOCTYPE html>
<html>
<body>

  <script>
    var a = 5 + 7;
    var b = 2000 + " Vinh Kiệt ";
    var c = "Vinh Kiệt " + 1993;
    var d = "Nguyễn" + " Thị Bạch Liễu";
    document.write(a);
    document.write("<hr>");
    document.write(b);
    document.write("<hr>");
    document.write(c);
    document.write("<hr>");
    document.write(d);
  </script>

</body>
</html>
```



## 2.2.2./ Toán tử ?

### Độ ưu tiên của các toán tử.

Mức độ ưu tiên	Toán tử		
1	()		
2	*	/	%
3	+		-

Trong JavaScript, dấu khoảng trắng giữa các toán hạng và toán tử là không quan trọng (có cũng được, không có cũng không sao)

- Ví dụ, thứ tự thực thi của biểu thức  $7 + 8 * (10 - 2) + 3 * 4$  như sau:

◦  $7 + 8 * (10 - 2) + 3 * 4 \Rightarrow 7 + 8 * 8 + 3 * 4$

◦  $7 + 8 * 8 + 3 * 4 \Rightarrow 7 + 64 + 3 * 4$

◦  $7 + 64 + 3 * 4 \Rightarrow 7 + 64 + 12$

◦  $7 + 64 + 12 \Rightarrow 71 + 12$

◦  $71 + 12 \Rightarrow 83$

## 2.2.3./ Toán tử so sánh

Toán tử so sánh là loại toán tử dùng để so sánh hai giá trị với nhau, ví dụ như:

- So sánh xem giá trị này có lớn hơn giá trị kia hay không.
- So sánh xem giá trị này có nhỏ hơn giá trị kia hay không.
- So sánh xem giá trị này có bằng với giá trị kia hay không.


Toán tử	Ý nghĩa của toán tử
>	Lớn hơn
>=	Lớn hơn hoặc bằng
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
==	Bằng giá trị ( <i>không phân biệt kiểu dữ liệu</i> )
===	Bằng giá trị ( <i>phải có chung kiểu dữ liệu</i> )
!=	Khác giá trị
!==	Khác giá trị hoặc khác kiểu dữ liệu

## 2.2.3./ Toán tử so sánh

Lưu ý: Biểu thức dùng để so sánh hai giá trị với nhau, được gọi là **biểu thức so sánh**. Biểu thức so sánh sẽ trả về một trong hai giá trị:

- **true** (nếu biểu thức so sánh đó là đúng)
- **false** (nếu biểu thức so sánh đó là sai)

```
var result_1 = 5 > 3;  
var result_2 = 5 > 7;  
var result_3 = 5 >= 3;  
var result_4 = 5 >= 5;  
var result_5 = 5 >= 7;  
var result_6 = 5 < 7;  
var result_7 = 5 < 3;  
var result_8 = 5 <= 7;  
var result_9 = 5 <= 5;  
var result_10 = 5 <= 3;  
var result_11 = 5 == "5";  
var result_12 = 5 === "5";  
var result_13 = 5 != 7;  
var result_14 = 5 !== "7";
```



true
false
true
true
false
true
false
true
false
true
true
false

## 2.2.4./ Toán tử logic

- Toán tử logic là loại toán tử dùng để xác định mối quan hệ logic giữa các giá trị logic (true, false)
- Toán tử logic được chia làm ba loại: **&& || !**

=> Toán tử && (and) sẽ trả về giá trị:

- true: nếu hai giá trị là true
- false: nếu hai giá trị là false, hoặc một giá trị là true và một giá trị là false

=> Toán tử || (or) sẽ trả về giá trị:

- true: nếu hai giá trị là true, hoặc một giá trị là true và một giá trị là false
- false: nếu hai giá trị là false

=> Toán tử ! (not) sẽ trả về giá trị logic ngược lại.

## 2.2.4./ Toán tử logic

Một số ví dụ:

true && true	Trả về: true
true && false	Trả về: false
false && true	Trả về: false
false && false	Trả về: false
true    true	Trả về: true
true    false	Trả về: true
false    true	Trả về: true
false    false	Trả về: false
!true	Trả về: false
!false	Trả về: true

## 2.2.4./ Toán tử logic

```
var result_1 = (5 > 3) && (5 < 7);  
var result_2 = (5 > 3) && (5 > 7);  
var result_3 = (5 < 3) && (5 > 7);  
var result_4 = (5 > 3) || (5 < 7);  
var result_5 = (5 > 3) || (5 > 7);  
var result_6 = (5 < 3) || (5 > 7);  
var result_7 = !(5 < 3);  
var result_8 = !(5 > 3);  
var result_9 = (5 > 3) && (5 > 7) || (7 > 5);
```

## 2.3./ Sử dụng hàm (Function) trong Javascript

- **Hàm** là một tập hợp gồm nhiều câu lệnh, các câu lệnh này được sắp xếp theo một thứ tự xác định để xây dựng thành một chức năng cụ thể
- Mỗi hàm sẽ có một cái tên và hàm chỉ thực thi khi nó được gọi đến tên.

```
<!DOCTYPE html>
<html>
<body>
  <input type="button" id="Gioithieu" value="Giới Thiệu"
    onclick="var name='Hào Côn';
            var year=1971;
            document.write('Tôi là ' + name + ' sinh năm ' + year)">
</body>
</html>
```



## 2.3./ Sử dụng hàm (Function) trong Javascript

```
<!DOCTYPE html>
<html>
<head>
  <script>
    function gioithieu()
    {
      var name = "Hào Côn";
      var year = 1971;
      document.write("Họ tên : " + name + " Sinh năm : " + year);
    }
  </script>
</head>
<body>
  <input type="button" id="gioithieu" value="Giới Thiệu" onclick="gioithieu()">
</body>
</html>
```



## 2.3./ Sử dụng hàm (Function) trong Javascript

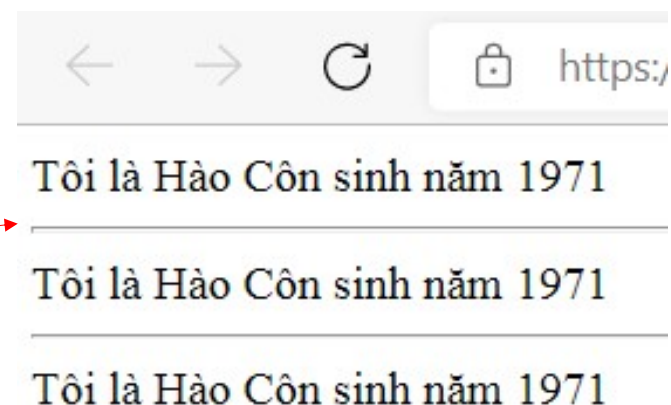
### Phân loại hàm

- Hàm được chia làm hai loại cơ bản: **hàm không có tham số** & **hàm có tham số**. Hàm không có tham số là hàm mà kết quả thực thi của nó luôn luôn không thay đổi.

```
<!DOCTYPE html>
<html>
<body>

  <script>
    function GioiThieuBanThan()
    {
      var name = "Hào Côn";
      var year = 1971;
      document.write("Tôi là " + name + " sinh năm " + year);
    }

    GioiThieuBanThan();
    document.write("<hr>");
    GioiThieuBanThan();
    document.write("<hr>");
    GioiThieuBanThan();
  </script>
</body>
</html>
```



## 2.3./ Sử dụng hàm (Function) trong Javascript

### Phân loại hàm

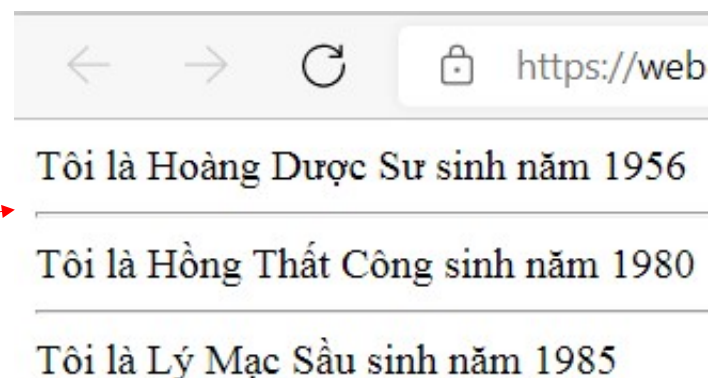
- **Hàm có tham số** là loại hàm mà khi gọi hàm ta phải truyền giá trị vào cho nó. Tùy vào giá trị được truyền mà hàm sẽ thực thi và cho ra kết quả khác nhau..

```
<!DOCTYPE html>
<html>
<body>

  <script>
    function GioiThieuBanThan(x,y){
      var name = x;
      var year = y;
      document.write("Tôi là " + name + " sinh năm " + year);
    }

    GioiThieuBanThan("Hoàng Dược Sư", 1956);
    document.write("<hr>");
    GioiThieuBanThan("Hồng Thất Công", 1980);
    document.write("<hr>");
    GioiThieuBanThan("Lý Mạc Sầu", 1985);
  </script>

</body>
</html>
```



## 2.3./ Sử dụng hàm (Function) trong Javascript

### Lệnh return

dùng để trả về cho hàm một giá trị. (Sau khi thực thi xong, hàm sẽ có một giá trị, lúc đó nó có thể được sử dụng giống như một biến).

```
<!DOCTYPE html>
<html>
<body>

  <script>
    function number()
    {
      return (10*10 - 50);
    }
    var result_01 = number();
    var result_02 = 7 + number() - 30;
    var result_03 = "Hello: " + number();

    document.write(result_01 + "<hr>");
    document.write(result_02 + "<hr>");
    document.write(result_03 + "<hr>");

  </script>
</body>
</html>
```



## 2.3./ Sử dụng hàm (Function) trong Javascript

### Lệnh return

dùng để trả về cho hàm một giá trị. (Sau khi thực thi xong, hàm sẽ có một giá trị, lúc đó nó có thể được sử dụng giống như một biến).

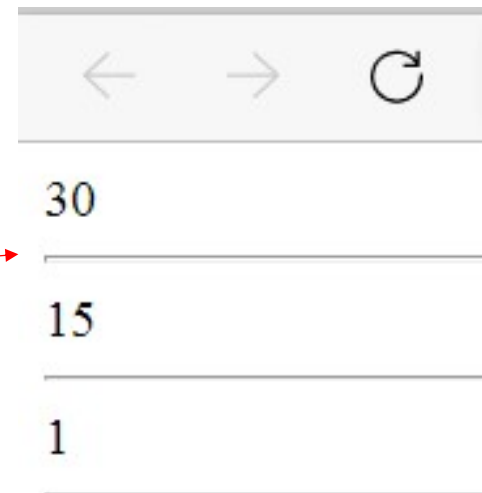
```
<!DOCTYPE html>
<html>
<body>

  <script>
    function number(a,b){
      return (a+b)*2;
    }

    var result_01 = number(5,10);
    var result_02 = number(2,8) - 5;
    var result_03 = 5*number(1,4) - 49;

    document.write(result_01 + "<hr>");
    document.write(result_02 + "<hr>");
    document.write(result_03 + "<hr>");

  </script>
</body>
</html>
```



30

15

1

## 2.3./ Sử dụng hàm (Function) trong Javascript

### Lệnh return

dùng để trả về cho hàm một giá trị. *(Sau khi thực thi xong, hàm sẽ có một giá trị, lúc đó nó có thể được sử dụng giống như một biến).*

```
<!DOCTYPE html>
<html>
<body>
<script>
    function ThongTin(name, year){
        var hoten = "Họ tên: " + name;
        var namsinh = "Năm sinh: " + year;
        var thongtin = hoten + "<br>" + namsinh;
        return thongtin;
    }

    var SinhVien = "SINH VIÊN<hr>" + ThongTin("Giang Hào Côn", 1971);
    document.write(SinhVien);
</script>
</body>
</html>
```

