Kiểu dữ liệu (Data type)

Có 2 kiểu chính: Kiểu nguyên thủy và object.

Kiểu nguyên thủy (Primitive)

- Number: 1, 6, 10.2, ...
- String: 'du thanh duoc', ...
- Boolean: true/false
- Null
- Undefined
- Symbol (ES6)

Sự khác nhau giữa `null` và `undefined`

`null` và `undefined` bằng nhau về mặt giá trị nhưng khác nhau về kiểu

```
typeof undefined // undefined
typeof null // object. Dù là object nhưng nó vẫn được xếp vào kiểu nguyên thủy nhé
null === undefined // false
null == undefined // true
```

`undefined` nghĩa là không có giá trị. `undefined` xuất hiện ở

```
// Biến chưa khởi tạo giá trị
let a
console.log(a) // undefined
// Thuộc tính không tồn tại trong object
let people = { name: 'Duoc' }
console.log(people.age) // undefined
// Thiếu param
function sum(a, b) {
  console.log(b)
}
sum(1) // log ra b là undefined
// function không return hoặc return undefined
function handle() {
  alert('XdevClass')
}
console.log(handle()) // undefined
```

`null` nghĩa là rỗng (khác với `''` nghĩa là chuỗi rỗng), giá trị của nó là rỗng. `null` thường được dùng cho object mà không có giá trị như mong đợi.

Không thể truy cập thuộc tính của `null` và `undefined`

```
function getName(value) {
  return value.name
}
// Lỗi
getName(undefined) // TypeError
getName(null) // TypeError
```

```
// Symbol (ES6)
let sym = Symbol()
console.log(sym)
```

Kiểu Object

Tất cả những kiểu không thuộc kiểu dữ liệu nguyên thủy được coi là kiểu Object

```
// Plain Object
var people = { name: 'Du Thanh Duoc', age: 24 }
// Array
var products = ['Iphone', 'Samsung', 'Xiaomi']
// Regular Expression
var regex = /ab+c/
// Function cũng được coi là 1 object
function sum(a, b) {
  return a + b
}
// Nhưng
typeof sum === 'function' // true
// null không được coi là object mặc dù
typeof null === 'object' // true
```

Với cách khai báo kiểu dữ liệu nguyên thủy như trên thì ta đang khai báo kiểu **constructor**. Ngoài ra ta có thể tạo một object cho boolean, number, string bằng các **wrapper object** như `String`, `Number`, `Boolean`.

```
var str = new String('Duoc')
console.log(typeof str) // object
console.log(str) // String {"Duoc"}
str === 'Duoc' // false
var num = new Number(1996)
var b = new Boolean(true)
```

Để lấy giá trị nguyên thủy từ các object trên thì ta dùng hàm `valueOf()`

```
var str = new String('Duoc')
str.valueOf() // 'Duoc'
```

Chúng có thể đóng vai như một function để chuyển đổi về giá trị nguyên thủy

```
String(1996) // '1996'
Number('2020') // 2020
Boolean(0) // false
```

Tham trị và tham chiếu

Mình đã có một bài đầy đủ và chi tiết tại tại đây: Bạn đã thực sự hiểu về tham trị và tham chiếu Javascript?

Tham trị

Khi giá trị thuộc kiểu dữ liệu nguyên thủy, biến sẽ chứa giá trị của biến đó.

```
// So sánh bằng giá trị
'abc' === 'abc' //true
24 === 24 // true
// Luôn luôn bất biến (immutable)
var num = 1
// Thêm thuộc tính name
num.name = 'XdevClass'
// Không hiệu nghiệm
console.log(num.name) // undefined
```

Tham chiếu

Khi gán hoặc sao chép dữ liệu thuộc kiểu object thì biến đó chỉ lưu địa chỉ của giá trị đó trên vùng nhớ. Nó không lưu giá trị được gán.

```
// So sánh bằng tham chiếu
{} === {} // false
var names1 = ['Duoc']
var names2 = ['Duoc']
names1 === names2 // false
// Có thể thay đổi (mutate)
var people1 = {name: 'Jack'}
var people2 = people1
people2.name = 'Huge'
console.log(people1) // {name: 'Huge'}
```

Truthy và Falsy

Truthy và falsy là những giá trị mà Javascript khi ép về kiểu boolean sẽ cho ra `true` hoặc `false`. Giống như dùng `Boolean(value)` để ép kiểu vậy.

Các giá trị được cho là **truthy**: Chuỗi khác rỗng, số khác `0` và tất cả các object.

Vì thế: `[]` hay `{}` vẫn được cho là truthy

```
Boolean({}) // true
Boolean([]) // true
```

Các giá trị được cho là falsy: `undefined`, `null`, `false`, `0`, `NaN`, `''`