

Types de Javascript

Jean-Luc Falcone

Août 2019

Combien de types en JS ?

Combien de types en JS ?

- Environ 7

Valeurs logiques

Le type boolean n'a que deux valeurs: true et false.

```
> let x = true;
```

```
undefined
```

```
> let y = false;
```

```
undefined
```

```
> x && y
```

```
false
```

```
> x || y
```

```
true
```

```
> x && !y
```

```
true
```

Nombres

Le type number représente les nombres à virgule flottante. Il n'existe pas de nombre entier en JS.

```
> let x = 12;
```

```
> let y = 0;
```

```
> x + y
```

```
12
```

```
> x / y
```

```
Infinity
```

```
> Math.log(-1)
```

```
NaN
```

```
> 2 ** 3
```

```
8
```

Chaînes de caractères

Le type string représente les chaînes de caractères.

```
> let x = "Hello";
```

```
> let y = 'world';
```

```
> x + ", " + y + " !"
'Hello, world !'
```

```
> 'Il a dit: "Bonjour".'
```

```
'Il a dit: "Bonjour".'
```

```
> "Il a dit: 'Bonjour'."
```

```
'Il a dit: \'Bonjour\'. '
```

Chaînes de caractères: Méthodes

Le type string comprend plusieurs méthodes pratiques:

```
> let s = "Balalaika";  
> s.length;  
9  
> s.slice( 1, 6 );  
'alala'  
> s.replace("lalaik","nan");  
'Banana'  
> s.indexOf("k")  
7  
> s.charAt(6);  
'i'
```

"Non défini"

Le type `undefined` représente les valeurs non-définies. Il n'a qu'une seule valeur nommée également `undefined`.

```
> let x = undefined;
```

```
> x;
```

```
undefined;
```

```
> x = 3;
```

```
> x;
```

```
3
```

null

Il existe aussi une valeur `null` dont la sémantique est similaire mais différente. Il est déconseillé de l'utiliser.

Tableaux (1)

Le type array représente des tableaux dynamiques.

```
> let n = [ 'Alice', 'Bob', 'Charles' ];
```

```
> n.length;
```

```
3
```

```
> n[0] + " et " + n[2];
```

```
'Alice et Charles'
```

```
> n.push("Dora")
```

```
4
```

```
> n
```

```
[ 'Alice', 'Bob', 'Charles', 'Dora' ]
```

```
> n[1] = 'Burt'
```

```
'Burt'
```

```
> n
```

```
[ 'Alice', 'Burt', 'Charles', 'Dora' ]
```

Tableaux (2)

Le type array représente des tableaux dynamiques.

```
> n[25] = 'Zoe'
```

```
'Zoe'
```

```
> n
```

```
[ 'Alice', 'Burt', 'Charles', 'Dora', , , , ,  
  , , , , , , , , , , , , , , , , 'Zoe' ]
```

Tableaux: Méthodes

Le type array comprend plusieurs méthodes pratiques:

```
> let n = [ 'Alice', 'Burt', 'Charles', 'Dora' ];  
[ 'Alice', 'Burt', 'Charles', 'Dora' ]  
> n.pop()  
'Dora'  
> n  
[ 'Alice', 'Burt', 'Charles' ]  
> n.shift()  
'Alice'  
> n  
[ 'Burt', 'Charles' ]  
> n.concat( ['Dora', 'Eugen'] )  
[ 'Burt', 'Charles', 'Dora', 'Eugen' ]  
> n  
[ 'Burt', 'Charles' ]
```

Objets (1)

Le type `object` représente les objets. Il s'agit de tableaux associatifs.

```
> let a = { name: "Alice", age: 34 };  
> a  
{ name: 'Alice', age: 34 }  
> a.name  
'Alice'  
> a.age = 35  
35  
> a  
{ name: 'Alice', age: 35 }
```

Objets (2)

```
> a["name"] = "Alysse"
'Alysse'
> a
{ name: 'Alysse', age: 35 }
> a.address
undefined
> Object.keys(a)
[ 'name', 'age' ]
```

Imbrication

On peut représenter toutes les structures de données en utilisant les types de base:

```
let a = {  
  name: {  
    firstnames: [ 'Alice', 'Pleasance' ],  
    surname: 'Hargreaves',  
    born: 'Lidell'  
  },  
  birthday: { date: "1852-05-04", place: "Oxford" },  
  address: undefined  
};
```

Fonctions

Les fonctions Javascript sont considérées comme des valeurs à part entière. Elles sont représentées par le type `function`.

```
> function f(x) {  
    return x + 1;  
}
```

```
> f(3)
```

```
4
```

```
> f("Numéro ")  
'Numéro 1'
```

```
> let incr = f;
```

```
> incr( 1 )
```

```
2
```

Fonction anonyme

```
> let decr = function( x ) {  
    return x - 1;  
}  
> incr( decr( incr( decr( 2 ) ) ) ) );  
2
```


Fonctions membre des objets

Comme une fonction est une valeur comme les autres, on peut ajouter des fonctions à un objet:

```
> let Mathematique = {  
  abs: function(x) {  
    if( x < 0 ) {  
      return -x;  
    } else {  
      return x;  
    }  
  },  
  racineCarrée: Math.sqrt  
};
```

Fonctions membres des objets (usage)

```
> Mathematique.racineCarrée( 100 );  
10  
> Mathematique.abs( -2 );  
2  
> Mathematique.abs  
[Function]
```

Fonctions et construction d'objet

Un objet étant une valeur comme une autre, on peut les retourner depuis une fonction:

```
> function point2D( x, y ) {  
  return {  
    dim: 2,  
    x: x,  
    y: y  
  }  
}  
  
> point2D( 2, -3 )  
{ dim: 2, x: 2, y: -3 }
```

Opérateur typeof

L'opérateur typeof permet d'obtenir le type d'une valeur. Il retourne une chaîne de caractères correspondant au nom du type:

```
> typeof true;  
'boolean'
```

```
> typeof 12;  
'number'
```

```
> typeof "12";  
'string'
```

```
> typeof undefined;  
'undefined'
```

```
> typeof {};  
'object'
```

```
> typeof null;  
'object'
```

```
> typeof Math.sqrt;  
'function'
```

```
> typeof [1,2,3,4];  
//???
```