

TABLEAUX, OBJETS ET ITÉRATION

Objectifs

- Créer des tableaux et accéder à leurs valeurs
- Comprendre le concept de copie par référence
- Découvrir les méthodes de tableau basiques
- Travailler avec des tableaux imbriqués
- Créer des objets littéraux

On peut créer un tableau avec `[x,y]`
Les tableaux en JavaScript peuvent
mélanger les types de valeurs.

```
let students = [];  
let couleurs = ["rouge", "bleu", "jaune"];  
let numsGagnants = [19, 22, 56, 12];  
let mixed = [true, null, 42, "chat", NaN];
```

On peut accéder à la longueur d'un tableau avec la propriété **length**.

On peut accéder à un élément en particulier avec la syntaxe **array[n]**

```
let mixed = [true, null, 42, "chat", NaN];  
mixed.length; // 5  
mixed[0]; // true  
mixed[5]; // undefined
```

Méthodes de tableau basiques

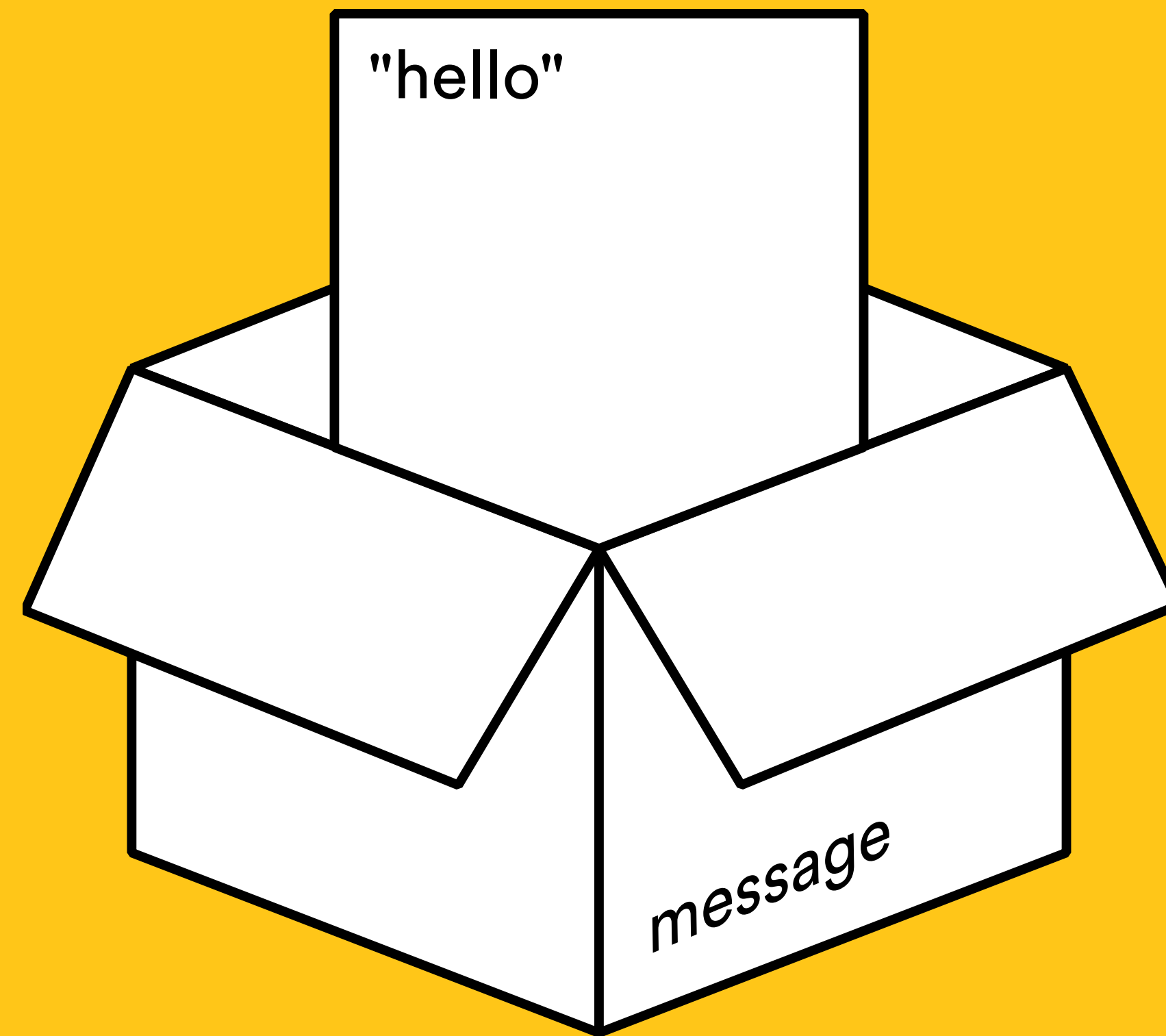
- **Push** – ajouter à la fin
- **Pop** – supprimer de la fin
- **Shift** – Enlever du début
- **Unshift** – Ajouter au début
- **Concat** – Fusionne des tableaux
- **Includes** – Cherche une valeur
- **indexOf** – comme `str.indexOf`
- **Join** – Créé une string à partir du tableau
- **Reverse** – Inverse un tableau
- **Slice** – copie une portion du tableau
- **Splice** – enlève / remplace des éléments
- **Sort** – classe notre tableau

```
let grille = [  
  ["0", null, "X"],  
  [null, "X", "0"],  
  ["X", "0", null]  
]  
  
grille[1]  
[2]; // "0"
```

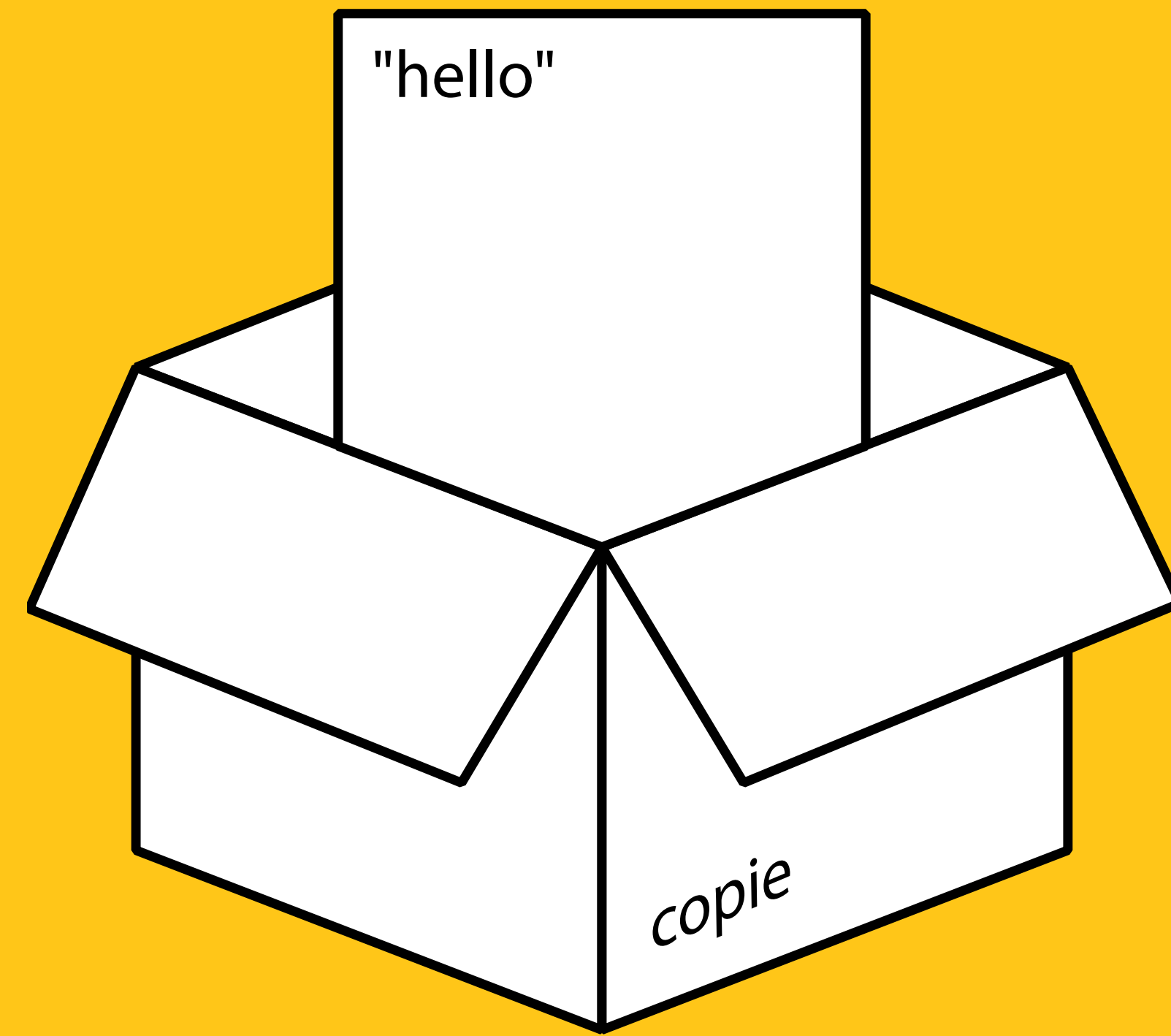
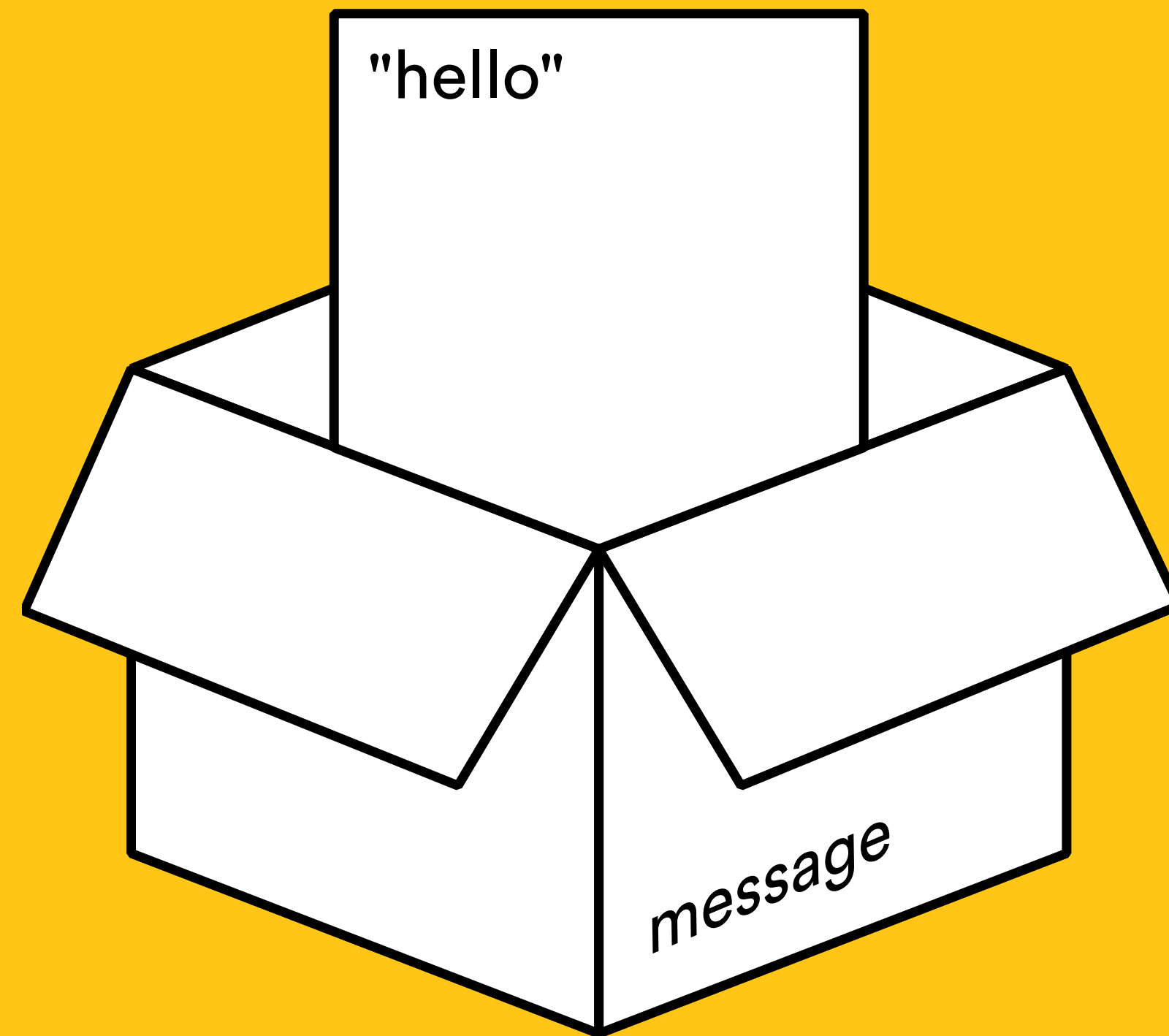
**Vous pouvez imbriquer
des tableaux dans des
tableaux pour créer
des matrices.**



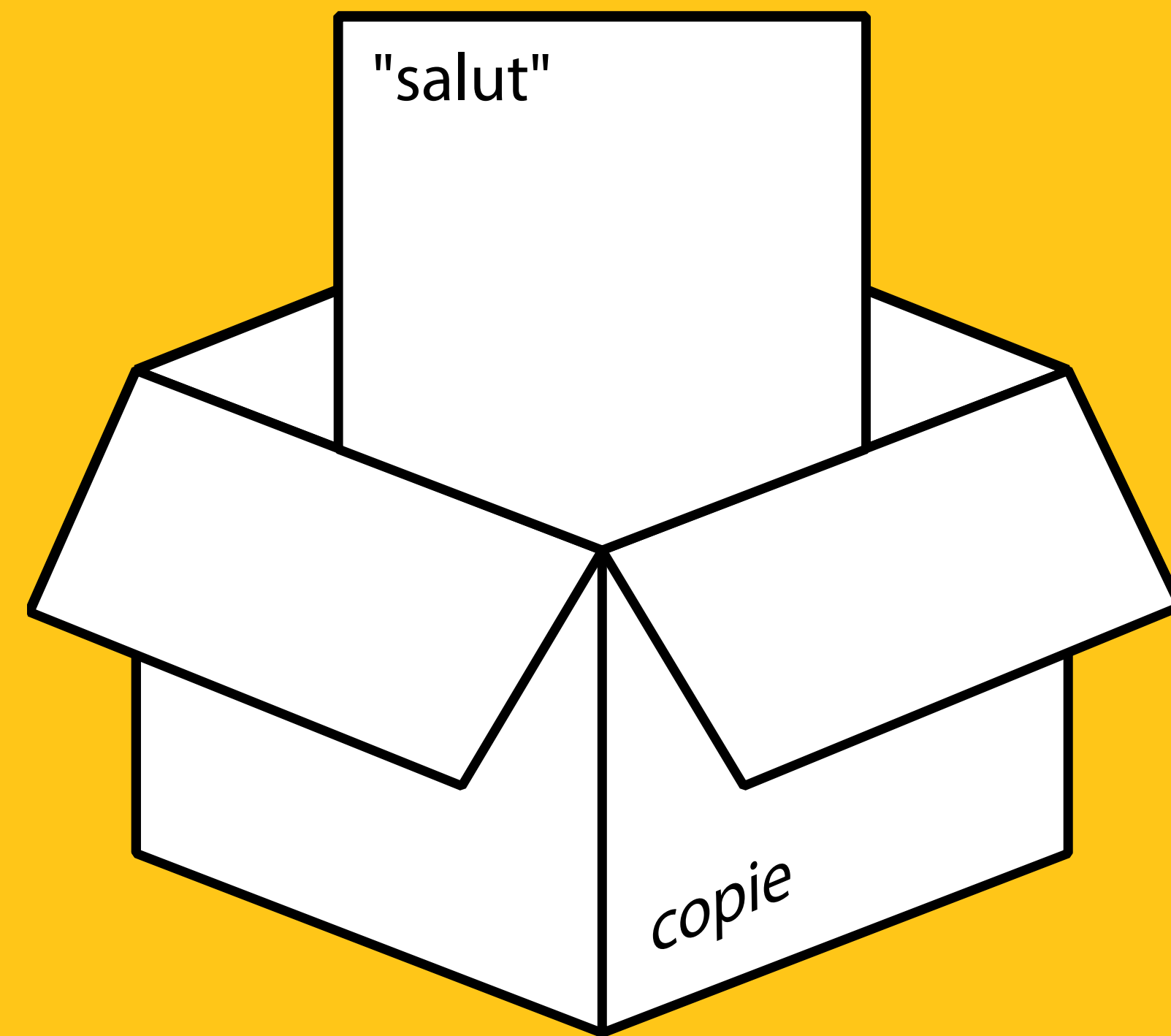
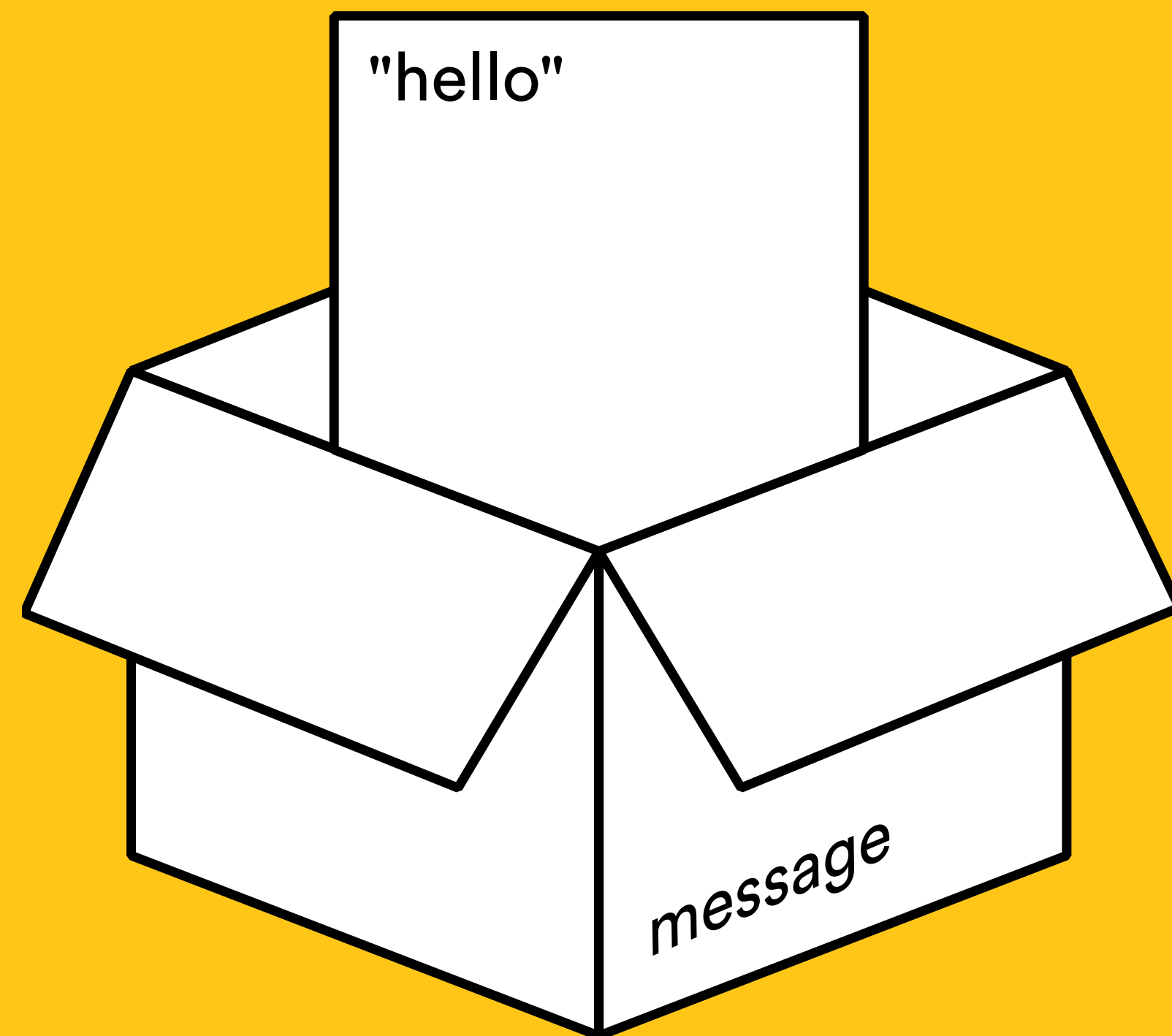
Contrairement aux valeurs primitives, les tableaux (et les objets) sont stockés et copiés par référence.



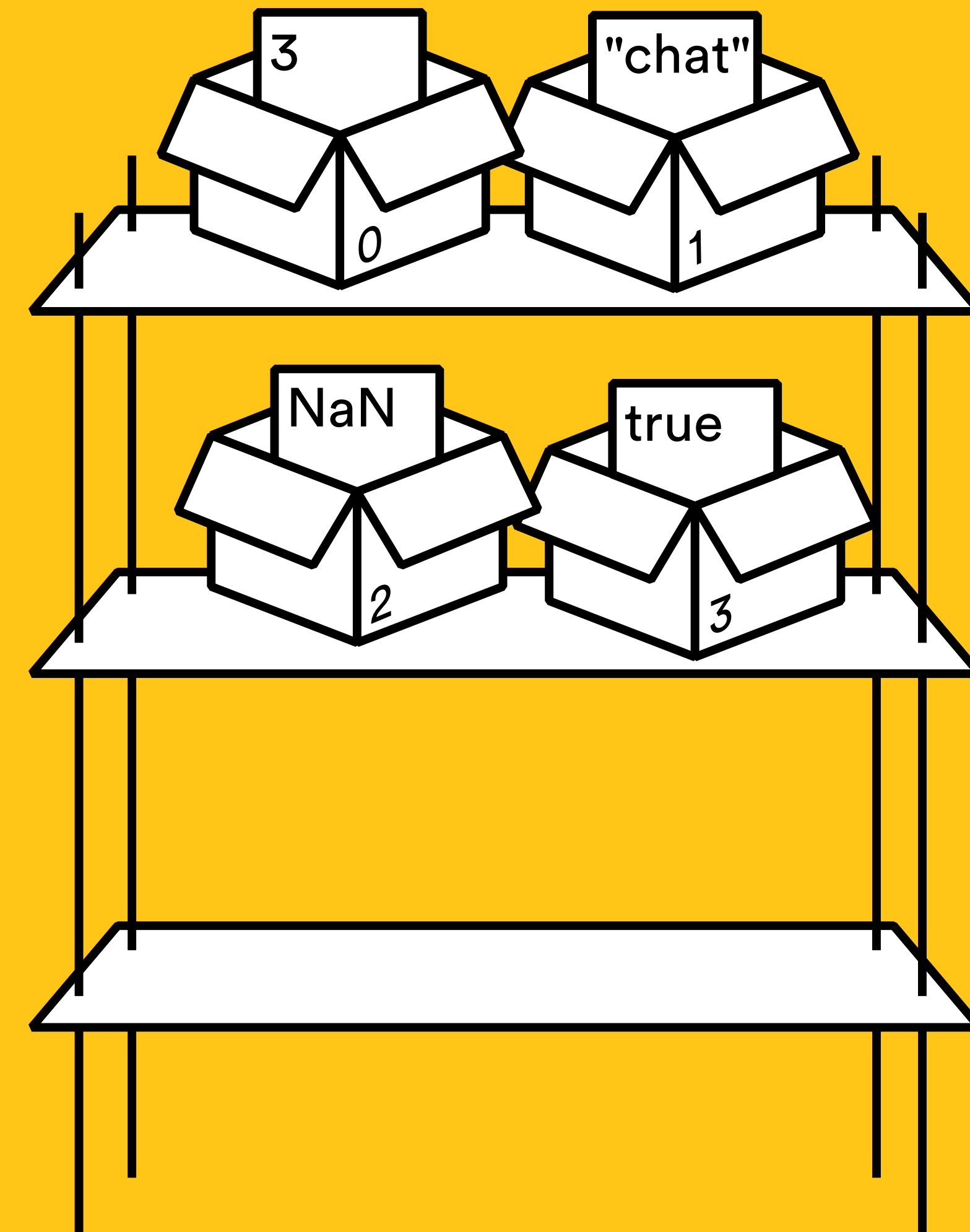
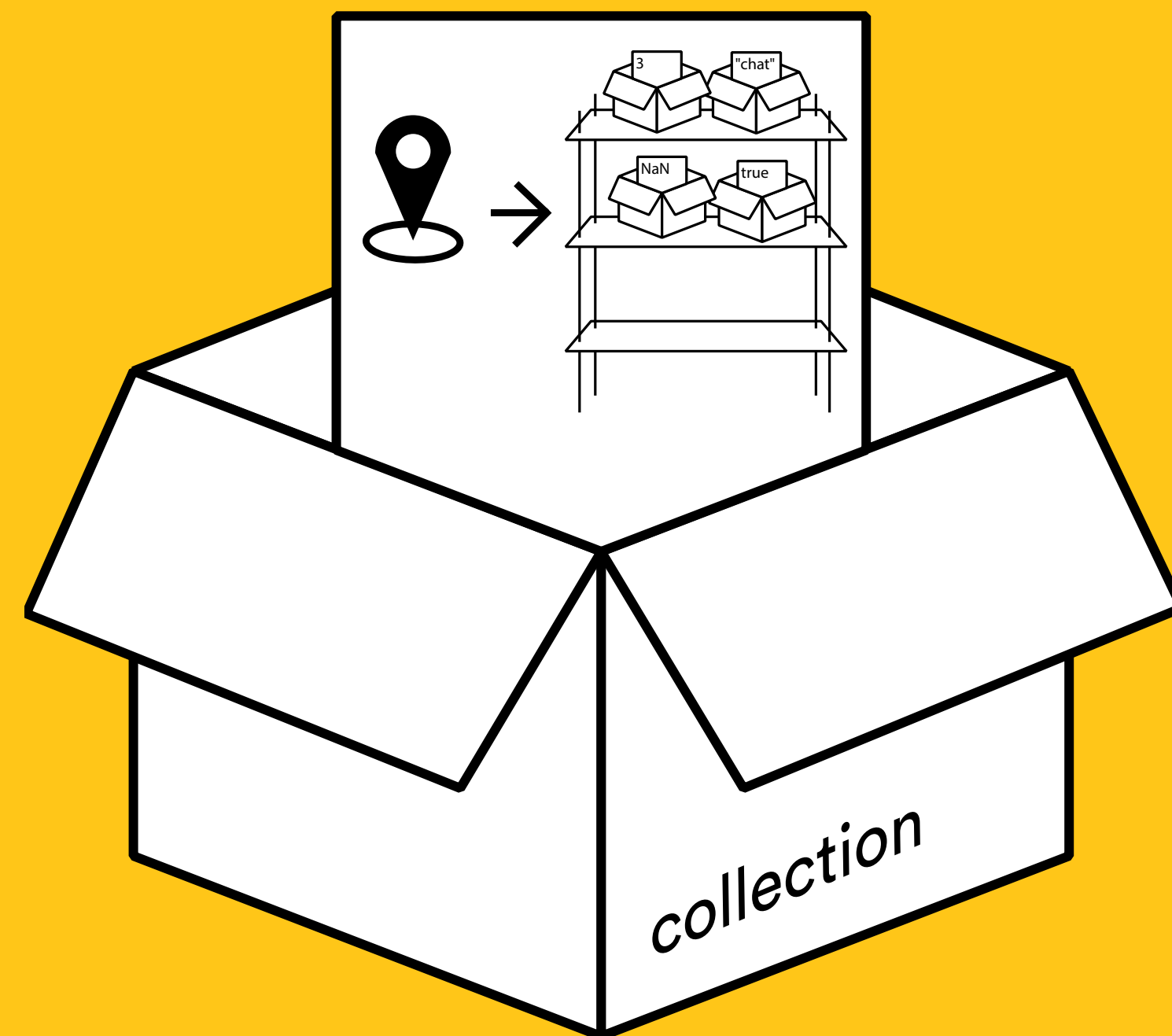
```
let message = "hello";
```

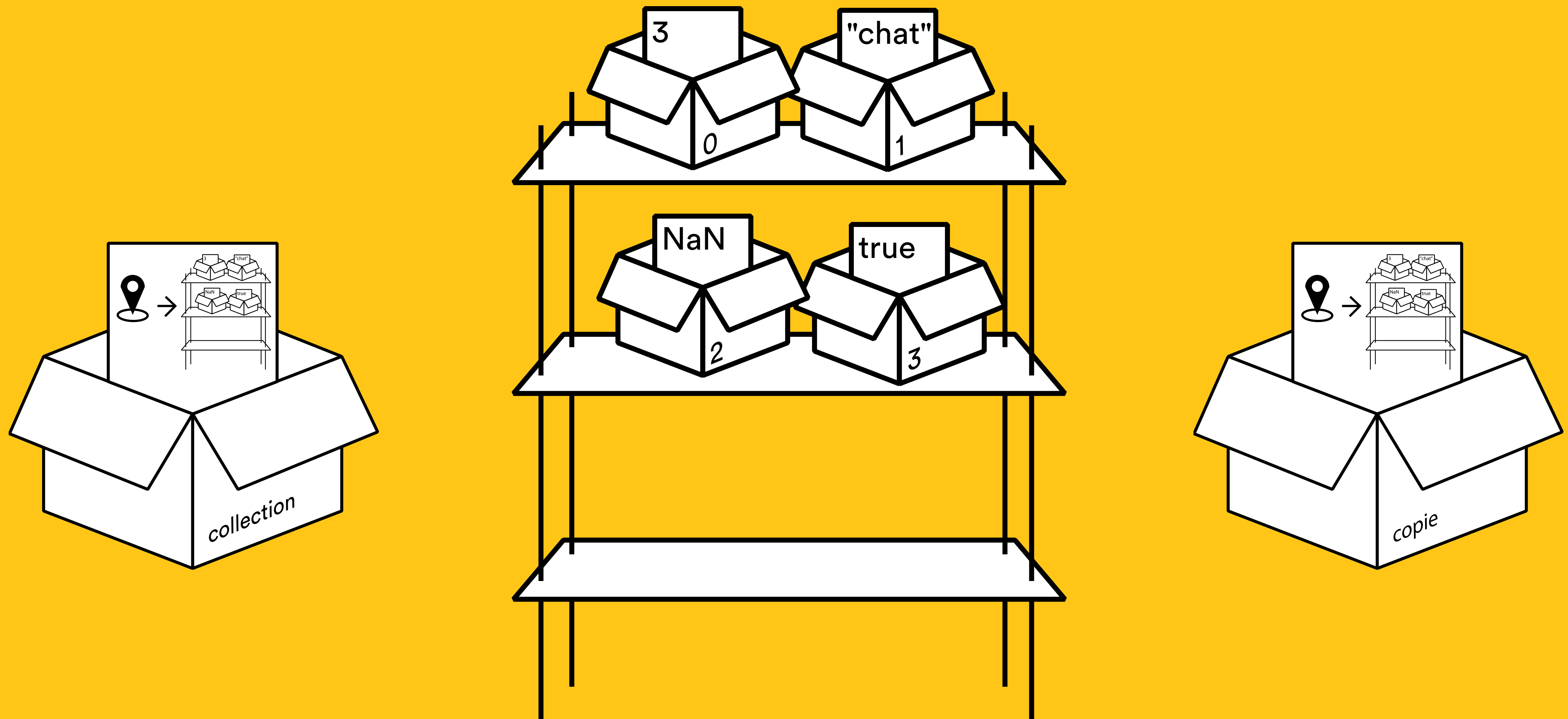
```
let message = "hello";  
let copie = message;
```



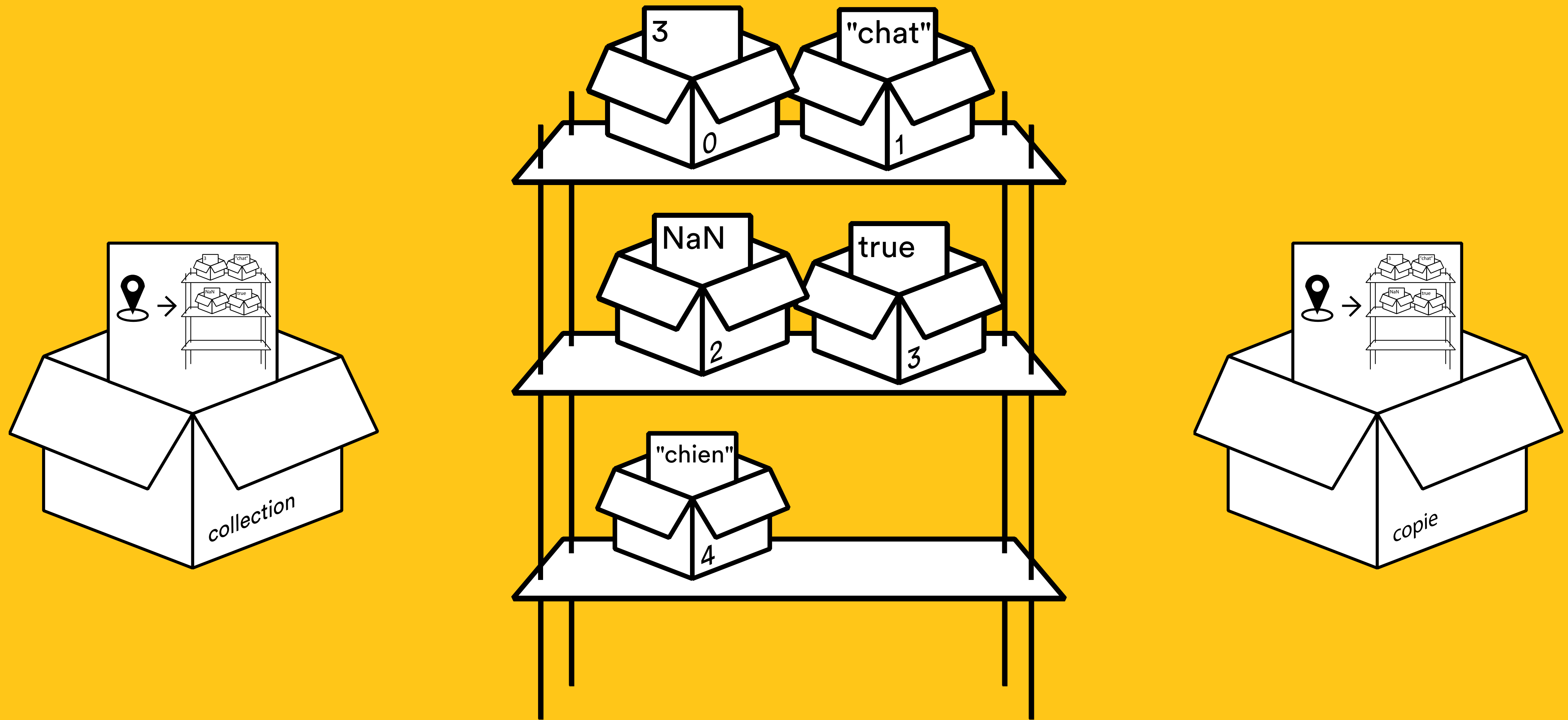
```
let message = "hello";  
let copie = message;  
copie = "salut";
```



```
const collection = [3, "chat", NaN, true];
```



```
const collection = [3, "chat", NaN, true];  
const copie = collection;
```



```
const collection = [3, "chat", NaN, true];  
const copie = collection;  
copie.push("chien");
```

**Pour éviter les bugs,
stockez vos tableaux dans
une constante!**



Shohei Ohtani

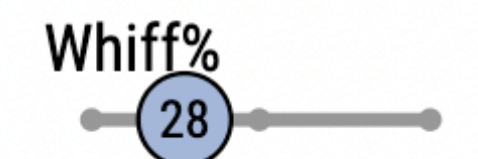
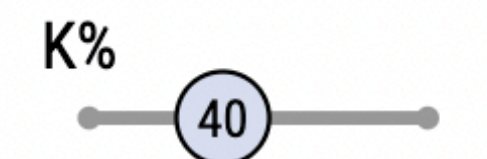
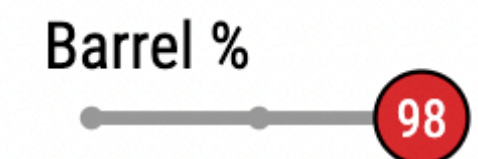
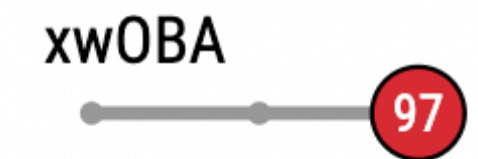
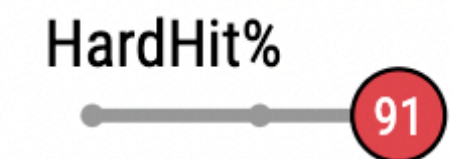
TWP | B/T: L/R | 6' 4" 210LBS | Age: 28

Player Apps

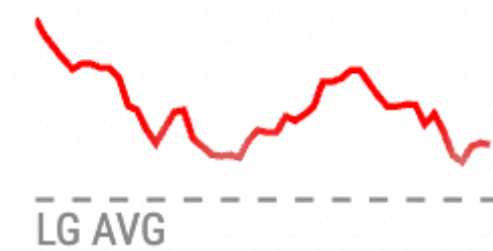
[Pitch Highlighter](#) | [Player Similarity](#) | [Shifts](#) | [Swing Take Profile](#) | [illustrator](#) | [Zone Swing Profile](#) | [Player Comparison](#) | [Transaction History](#)

Show Random Video

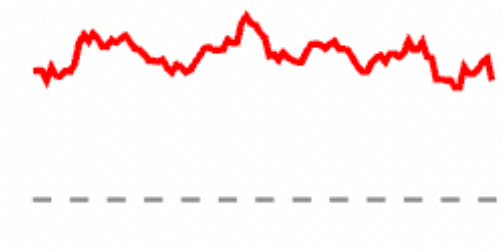
2022 MLB Percentile Rankings



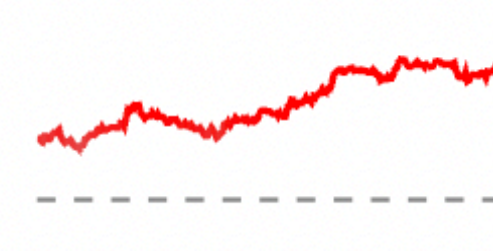
xwOBA



Past 50 PA



Past 100 PA



Past 250 PA

```
const player = [96, 100,  
91, 97, 87, 98, 98, 40,  
89, 28, 69, 78]
```



```
const int array[9] = {0, 0, 0, 0, 0, 0, 0, 0, 0},  
  array2[9] = {91, 97, 98, 99, 100, 101, 102, 103, 104},  
  array3[9] = {89, 28, 7, 78, 1, 2, 3, 4, 5}
```



```
const user = {}; // objet vide
```

```
const player = {  
  firstName: "Shohei",  
  lastName: "Ohtani",  
  size: `6' 4"`,  
  weight: "210LBS",  
  age: 28,  
}
```

```
player.age // 28
```

```
player.isActive = true; // active: true
```

```
delete player.age;
```

Les objets en JavaScript peuvent être créés sans constructeur (nous parlerons des classes plus tard). Il s'agit simplement d'une collection de données caractérisée par des propriétés sous forme de paires "clé : valeur".

La syntaxe `{ }` crée un objet littéral.

Contrairement au tableau, l'ordre des propriétés d'un objet n'est pas important.

On crée et accède une propriété avec la notation par point.

Le mot-clé `delete` permet de se débarrasser d'une propriété.

JSON

JAVASCRIPT OBJECT NOTATION

```
const nums = [1,2,3];  
const moreNums = [1,2,3];  
  
nums === moreNums // false  
nums = moreNums // false
```

Vous ne pouvez pas utiliser les opérateurs de comparaison pour comparer le contenu de deux tableaux ou deux objets. (Stockage par référence.)

```
// for loop
for (let i = 0; i ≤ 10; i++) {
  console.log(i);
}
```

```
// while loop
let j = 0;

while(j ≤ 10) {
  console.log(j);
  j++;
}
```

Vous pouvez créer des boucles for et while classiques.

Attention aux boucles infinies!

```
for (variable of iterable) {  
}
```

```
const animaux = ["chat", "chien", "rat"];

for (let i = 0; i < animaux.length; i++) {
    console.log(animaux[i]);
}

for (animal of animaux) {
    console.log(animal);
}

// chat
// chien
// rat
```

Si vous n'avez pas besoin de l'index, l'utilisation d'une boucle **for...of** est plus propre.

Cette syntaxe est disponible pour toutes les structures itérables (arrays, strings, Map, Set, sur-mesure...).

Attention ces boucles ne fonctionnent pour itérer sur les propriétés d'un objet.