programmation web en javascript

3/6

les tableaux

- un tableau est 1 séquence d'éléments accessibles grâce à un indice entier. C'est une *sorte* d'objet définissant des propriétés entières par défaut
- création
- accès
- modification
- propriétés

```
let t = [ 'a', 'b', 'c', 'd'] ;

>t[0]
'a'


>t[0] = 'x' ;
>t
['x','b','c','d']


>t.length
4
>t.length = 3 ;
>t
['a', 'b', 'c']
```

- on peut mettre toutes sortes de choses dans un tableau :

```
let t= ['a', 'b', 'c' ]
let tab = [1,2,3, {}, 'a', (x)=>x+2, t] ;

console.log( tab[0] ) ;          // 1

tab[ 3 ].nom = 'marcel';         // tab[3] : objet
console.log( tab[3] ) ;          // { nom : "marcel" }

let v=tab[ 5 ](8) ;              // tab[5] : fonction
console.log( v ) ;               // 10

let l = tab[6][1] ;              // tab[6] : tableau
console.log( l ) ;               // "b"
```

quelques méthodes sur les tableaux

```
let t = ['a', 'b', 'c', 'd', 'e'] ;

// ajoute 1 élément
> t.push('z')
> t
['a', 'b', 'c', 'd', 'e','z']

// retire le dernier
> t.pop()
> t
['a', 'b', 'c', 'd', 'e']

// retire le premier
> t.shift()
> t
['b','c','d','e']

// extrait les éléments entre 2 indices
t.slice(1,3)
['c','d']
```

parcourir des tableaux

- avec un **`for`** :

```
let t = ['a', 'b', 'c', 'd', 'e'] ;
let i;

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

- avec **forEach** : exécute 1 fonction passée en paramètre sur chaque élément du tableau

```
let t = ['a', 'b', 'c', 'd', 'e'] ;

let logElement = (element) => { console.log(element); }

t.forEach(logElement) ;
```

filter()

- **`filter()`** : reçoit un callback et l'applique à chaque élément d'un tableau ; retourne un tableau contenant les éléments du tableau d'origine pour lesquels le callback retourne `true`
  - le *callback* prend en paramètre un `element` et renvoie un `boolean`

```
let t = [0,1,1,2,3,5,8] ;

let isGreaterThan4 = (element) => element > 4 ;

let result = t.filter(isGreaterThan4) ;

console.log(result) ;    // [5, 8]
console.log(t) ;         // [0,1,1,2,3,5,8]
```

map()

- **`map()`** : reçoit un callback et l'applique à chaque élément d'un tableau ; retourne un tableau formé des images des éléments du tableau par le callback passée en paramètre :
  - la *callback* prend en paramètre un `element` et renvoie une nouvelle valeur, qui peut être d'un type différent

```
let t = [0,1,1,2,3,5,8] ;

let multBy2 = (element) => element * 2 ;

let result = t.map(multBy2) ;

console.log(result) ;    // [0, 2,2,4,6,10,16]
console.log(t) ;         // [0,1,1,2,3,5,8]
```

on peut bien sur les combiner

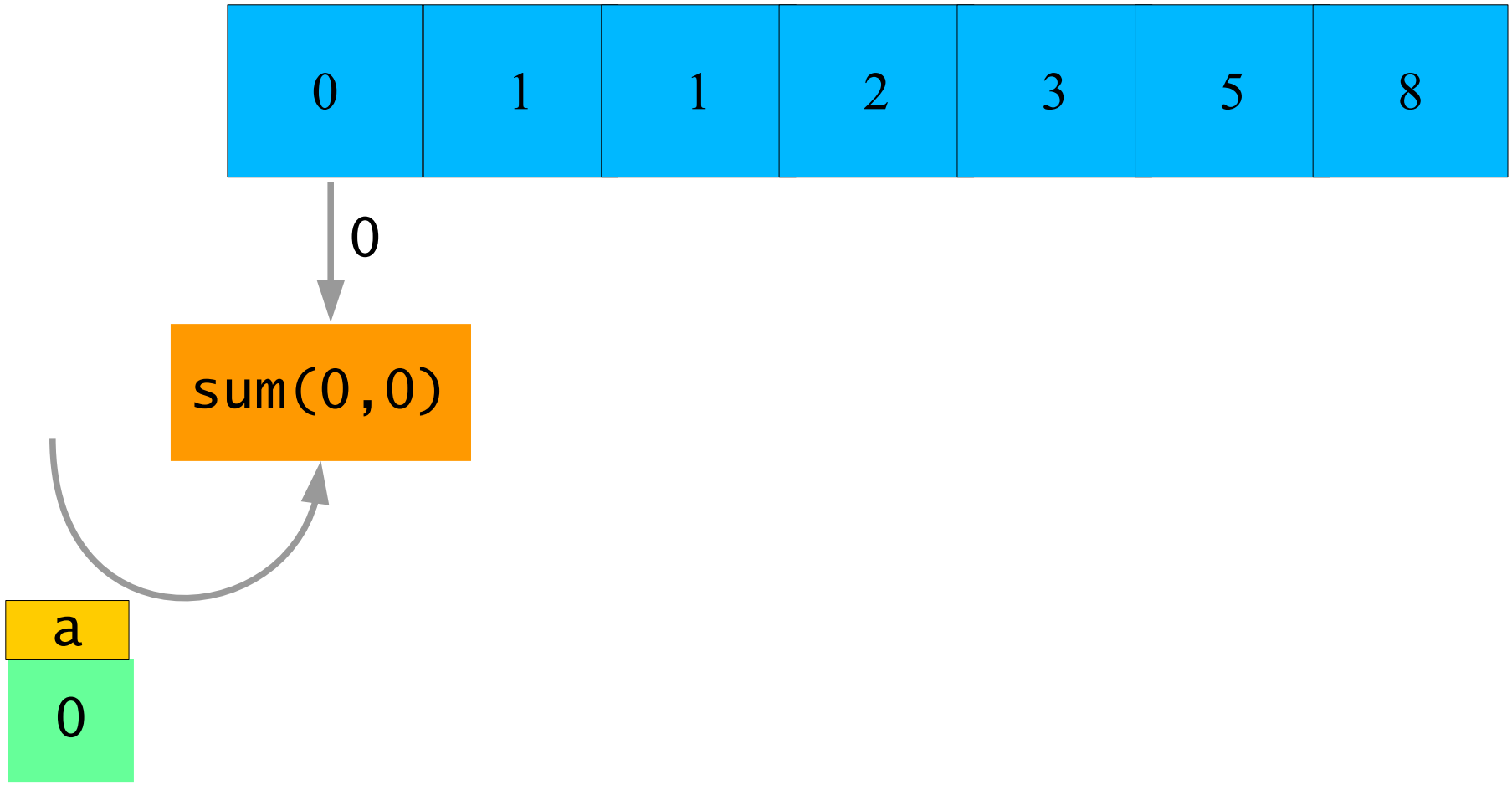
```
let t = [0,1,1,2,3,5,8] ;


let result = t.map( (e) => e * 2 )
              .filter( (e) => e > 4 );

console.log(result) ;    // [6,10,16]
console.log(t) ;         // [0,1,1,2,3,5,8]
```

reduce

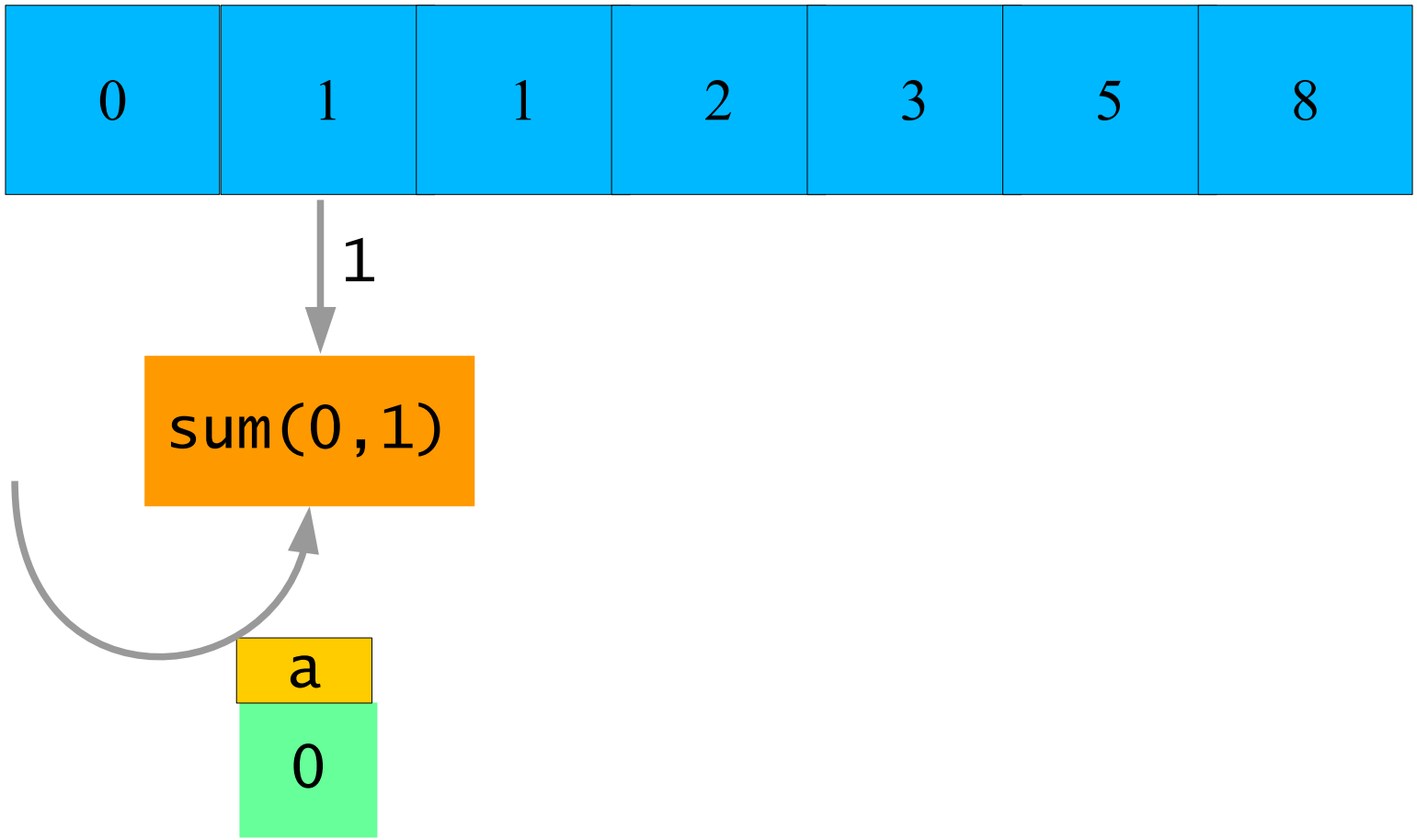
- **`reduce()`**: retourne **une valeur** calculée à partir d'une valeur initiale et de chaque élément du tableau (de gauche à droite)
  - le *callback* prend en paramètre un `accumulateur`, un `element` et renvoie une nouvelle valeur
  - la valeur retournée sert comme `accumulateur` pour le prochain calcul

```
let t = [0,1,1,2,3,5,8] ;

let sum = ( acc, elem ) => acc + elem ;

let result = t.reduce(sum, 0) ;

console.log(result) ;    // 20
console.log(t) ;         // [0,1,1,2,3,5,8]
```

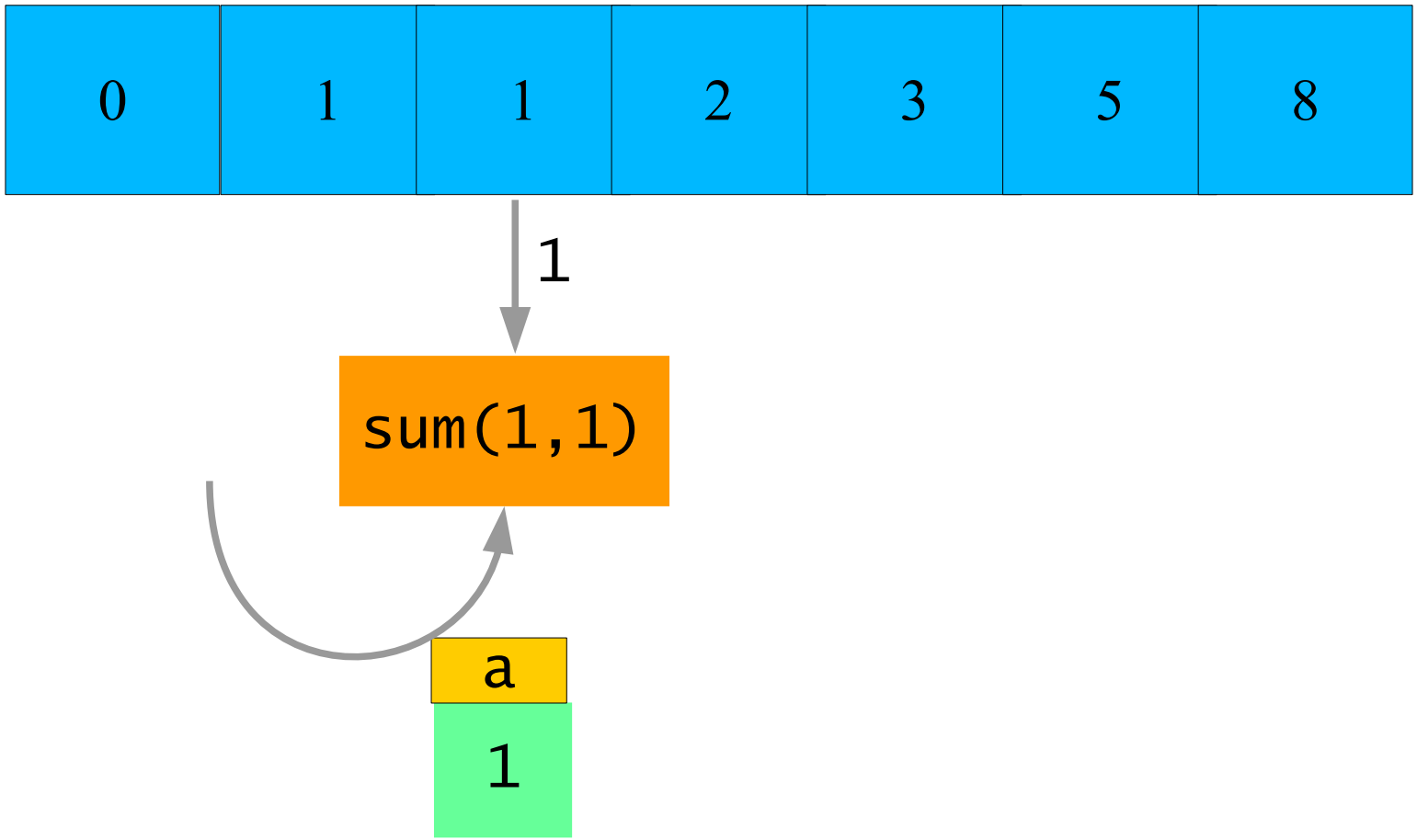
reduce(sum, 0)



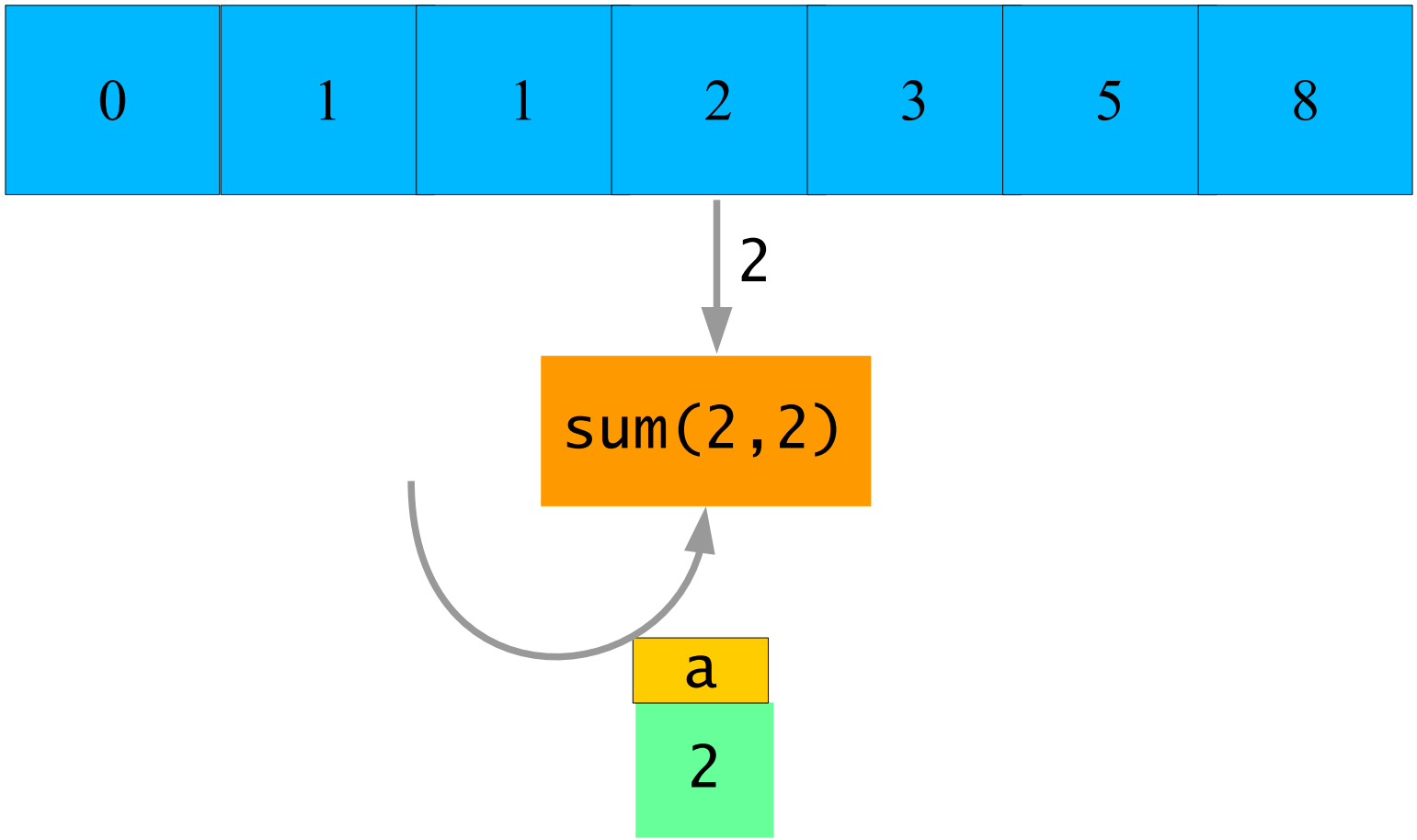
reduce(sum, 0)



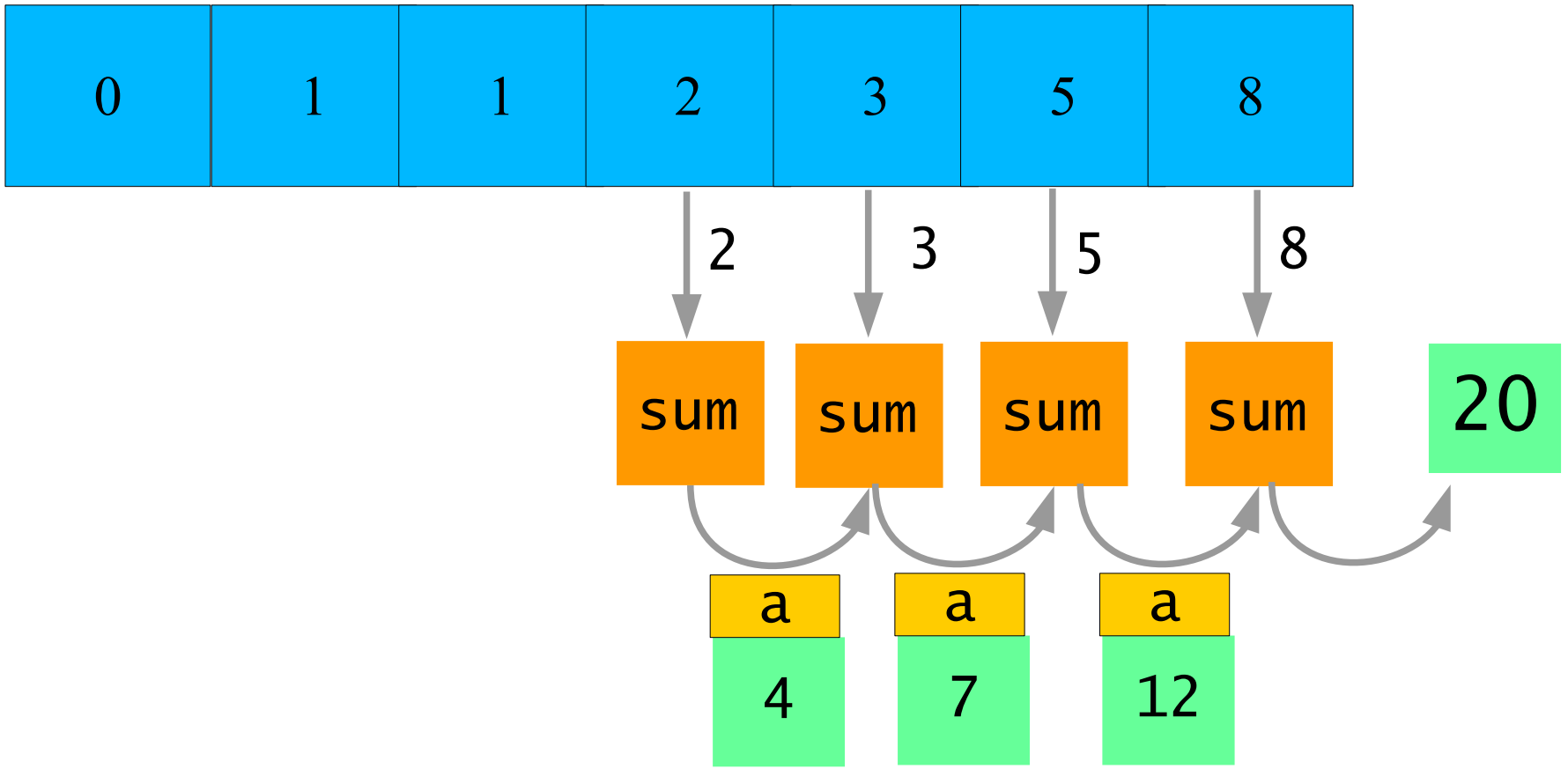
reduce(sum, 0)



reduce(sum, 0)



reduce(sum, 0)



reduce : autre exemple

- extraire le *min* d'un tableau
  - `accumulateur` a comme valeur par défaut le 1er élément du tableau

```
let t = [0,1,1,2,3,5,8] ;

let min = ( acc, elem ) => {
     if (acc < elem) return acc ;
     return elem ;
    }

let result = t.reduce( min ) ;

console.log(result) ;     // 0
console.log(t) ;          // [0,1,1,2,3,5,8]
```

combinaison

- Possibilité de combiner ces fonctions
  - `accumulateur` a comme valeur par défaut le 1er élément du tableau

```
let t = [0,1,1,2,3,5,8] ;


let result = t.filter( (e) => e > 4 )     // [ 5,8 ]
              .map( (e) => e * 2 )        // [ 10, 16 ]
              .reduce( (a,e) => a+e )     // 26

console.log(result) ;     // 26
console.log(t) ;          // [0,1,1,2,3,5,8]
```