ISEN
ALL IS DIGITAL!
OUEST
yncréa

JavaScript
- Tableaux -

Groupe des étudiants : CIR1

Généralités

- Un tableau est une collection de données indexée par un entier

- Pour initialiser un tableau, on indique des valeurs entre []

- En JavaScript, un tableau est un type particulier d'objet.

```
let t = [1,2,3];
console.log(typeof t);          // => object
```

Accès aux éléments

- Pour accéder au contenu d'un élément via son indice, on utilise t[i]
- Les indices commencent par 0

```
let t = ["foo", "bar", "qux"];
console.log(t[1]);          // => bar
```

- En JavaScript, les dépassements d'indice ne provoquent pas d'erreur

Lecture

```
let t = [1,2,3];
let v = t[10];
console.log(v); // => undefined
```

Ecriture

```
let t = [1,2,3];
t[10] = 42;
console.log(t); // => [1, 2, 3, empty × 7, 42]
```

Accès aux éléments

- **Solution 1** : utilisation des indices entiers (pareil que C)

```
let fruits = ["pomme", "banane", "orange", "poire"];

for(let i=0; i < fruits.length; i++){

    let fruit = fruits[i];
    console.log(fruit);
}
```

Retourne la longueur du tableau

```
/* =>
pomme
banane
orange
poire
*/
```

- **Solution 2** : for each (utilisée en lecture seule)

```
let fruits = ["pomme", "banane", "orange", "poire"];

for(let fruit of fruits){
    console.log(fruit);
}
```

```
/* =>
pomme
banane
orange
poire
*/
```

Accès aux éléments

- **Solution 1** : utilisation des indices entiers (pareil que C)

```
let fruits = ["pomme", "banane", "orange", "poire"];

for(let i=0; i < fruits.length; i++){

    let fruit = fruits[i];
    console.log(fruit);
}
```

Retourne la longueur du tableau

```
/* =>
pomme
banane
orange
poire
*/
```

- **Solution 2** : for each (utilisée en lecture seule)

```
let fruits = ["pomme", "banane", "orange", "poire"];

for(let fruit of fruits){
    console.log(fruit);
}
```

in

```
/* =>
0
1
2
3
*/
```

Test d'égalité

- En JS, si t1 et t2 sont deux tableaux, alors t1 == t2 ou t1 === t2 comparent l'adresse des deux tableaux et non leur contenu.

```
let t1 = [1,2,3];
let t2 = [1,2,3];

console.log(t1 == t1);     // => true
console.log(t1 == t2);     // => false
```

⇒ Pour comparer, correctement, deux tableaux, il faut comparer les cases une par une

Affectation

- L'affectation d'un tableau à une variable ne provoque pas la recopie de son contenu. Uniquement l'adresse de celui-ci est copiée.
  Conséquences :
  - Si t1 contient un tableau, la conséquence de l'affectation t2=t1 est que ces deux variables pointent sur le même tableau.

```
let t1 = [1,2,3];
let t2 = t1;  // recopie de l'adresse, pas du contenu

t1[1] = 42;
console.log(t1); // => [1, 42, 3]
console.log(t2); // => [1, 42, 3]
```

Affectation

- L'affectation d'un tableau à une variable ne provoque pas la recopie de son contenu. Uniquement l'adresse de celui-ci est copiée.
  Conséquences :
  - Si t1 contient un tableau, la conséquence de l'affectation t2=t1 est que ces deux variables pointent sur le même tableau.
  - Les tableaux sont systématiquement passés par adresse dans les fonctions.

```
function f(t){
    t[1] = 42;
}
```

```
let t = [1,2,3];
console.log(t); // => [1, 2, 3]
f(t);
console.log(t); // => [1, 42, 3]
```

Méthodes de manipulation

- Ajout/Suppression des éléments :

	Ajout	Suppression
Au début	`unshift()`	`shift()`
A la fin	`push()`	`pop()`

```
let t = ["foo", "bar", "qux"];

let a = t.shift();
console.log(a); // => foo
console.log(t); // => ["bar", "qux"]

let b = t.pop();
console.log(b); // => qux
console.log(t); // => ["bar"]

t.unshift("corge");
console.log(t); // => ["corge", "bar"]

t.push("grault");
console.log(t); // => [corge", "bar", "grault"]
```

- Duplication d'une portion d'un tableau t : t.slice(start, end)

```
let t = [1,2,3,4];
console.log(t.slice(3));      // => [4]
console.log(t.slice(1,3));    // => [2, 3]
console.log(t.slice(1,-2));   // => [2]
console.log(t);               // => [1, 2, 3, 4]
```

- slice sans argument "slice()" permet, aussi, de cloner un tableau :

```
let t1 = [1,2,3];
let t2 = t1.slice();  // t2 est une copie de t1

t1[1] = 42;
console.log(t1);    // => [1, 42, 3]
console.log(t2);    // => [1, 2, 3]
```

- Remplacement d'une portion d'un tableau t :

t.splice(start, nbElements, ...newElements)

```
let t1 = [1,2,3,4];
let t2 = t1.splice(1, 2, 404, 42, 1337)
console.log(t1);     // => [1, 404, 42, 1337, 4]
console.log(t2);     // => [2,3]
```

```
let t = ["tableau", "de", "chaînes", "de", "caractères"];
t.splice(2, 1); // supprime l'élément à l'indice 2
console.log(t); // => ["tableau", "de", "de", "caractères"]
```

- Recherche d'un élément dans un tableau t (avec la convention -1 = non trouvé) : t.indexOf(e)

```
let t = ["pomme", "banane", "orange", "poire"];

console.log(t.indexOf("orange")); // => 2
console.log(t.indexOf("kiwi"));   // => -1
```

Transformation tableau ↔ chaîne

- split() permet de découper une chaîne selon un séparateur. Le résultat du découpage est renvoyé sous la forme d'un tableau de chaînes

```
let sentence = "cette phrase va être découpée";
let words = sentence.split(" ");
console.log(words);     //  => ["cette",
"phrase", "va", "être", "découpée"]
```

- join() fusionne toutes les chaînes fournies dans un tableau en une chaîne unique. Les différentes parties sont délimitées par le séparateur fourni (virgule par défaut)

```
let numbers = [1,2,3,4];
let mergedNumbers = numbers.join("-");
console.log(mergedNumbers); // => 1-2-3-4
```

Caractéristiques communes entre chaînes et tableaux

- for..of et indexOf() peuvent être utilisées pour les chaînes de caractères

```
let word = "abc";

for(let letter of word){
    console.log(letter);
}

/* =>
a
b
c
*/

console.log(word.indexOf("c")); // => 2
console.log(word.indexOf("z")); // => -1
```