IFT1015 Programmation 1

Opérations sur les tableaux

Marc Feeley

Université de Montréal

Méthodes sur les tableaux vues à date

- *t*`.length` longueur du tableau
- *t*`.push(`*x*`)` ajoute l'élément *x* à la fin
- *t*`.pop()` retire le dernier élément
- *t*`.fill(`*x*`)` remplace les éléments par *x*
- *t*`.concat(`*tableau2*`)` concaténation des tableaux
- *t*`.slice(`*début*`,`*fin*`)` extraction d'un sous-tableau
- *t*`.map(`*fn*`)` appèle *fn* sur chaque élément
- *t*`.forEach(`*fn*`)` appèle *fn* sur chaque élément
- *t* est modifié par **`push`**, **`pop`** et **`fill`** mais pas par **`concat`**, **`slice`** et **`map`** (créent un nouveau tableau)

`push`, **`pop`** et **`fill`** sont des méthodes **"modifiantes"**

Généralisation de push

- On a vu l'appel simple : *t*`.push(`*x*`)`
- **`push`** accepte un nombre quelconque de paramètres qui sont tous ajoutés à la fin du tableau :

```
var t1 = [11,22];

t1.push(33);
t1.push(44);
t1.push(55);
print(t1);

var t2 = [11,22];

t2.push(33,44,55);
print(t2);
```

```
11,22,33,44,55
```

```
11,22,33,44,55
```

Généralisation de concat

- On a vu l'appel simple : *t*`.concat(`*tableau2*`)`
- **`concat`** accepte un nombre quelconque de paramètres qui sont tous concaténés au tableau pour construire le tableau résultant :

```
var t = [11,22];

print(t.concat(t));                          11,22,11,22

print(t.concat(t).concat([99]).concat(t));

print(t.concat(t,[99],t));                   11,22,11,22,99,11,22
```

Généralisation de slice

- On a vu l'appel simple : *t*`.slice(`*début*`,`*fin*`)`
- `slice` accepte deux, un ou aucun paramètres
- Si *fin* n'est pas fourni c'est comme si on avait fourni *t*`.length` (c'est-à-dire la fin du tableau)
- Si *début* n'est pas fourni c'est comme si on avait fourni 0 (c'est-à-dire le début du tableau)
- Si un **entier négatif** est fourni comme *début* ou *fin*, alors c'est une position relative à la fin du tableau (par exemple `-1` signifie l'index du dernier élément, `-2` signifie l'index de l'avant dernier élément, etc)

Généralisation de slice

```
var t = ["a","b","c","d","e"];
//        0   1   2   3   4

print(t.slice(1,3));
```

b,c

```
print(t.slice(1));
```

b,c,d,e

```
print(t.slice());
```

a,b,c,d,e

```
print(t.slice(1,-1));
```

b,c,d

```
print(t.slice(-2,-1));
```

d

```
print(t.slice(-2));
```

d,e

Autres méthodes modifiantes sur les tableaux

- *t*`.reverse()`
  - Modifie le tableau *t* pour que les éléments soient dans l'ordre inverse et retourne *t*
- *t*`.shift()`
  - Retire le **premier** élément du tableau *t* et le retourne (la longueur de *t* sera donc réduite de 1)
- *t*`.unshift(`*x*`)`
  - Ajoute l'élément *x* **au début** du tableau *t* (dont la longueur est donc augmentée de 1) et cette nouvelle longueur est retournée par `unshift`
  - Accepte un nombre quelconque de paramètres

Exemple de reverse, unshift, shift

```
var t = [11,22,33];

t.reverse();

print(t);

t.unshift(44);
t.unshift(55);
print(t);

var x = t.shift();
print(t);
print(x);

t.unshift(66,77);
print(t);
```

```
33,22,11

55,44,33,22,11

44,33,22,11
55

66,77,44,33,22,11
```

Méthode modifiante splice

- *t*`.splice(`*début*`,`*nbRetirer*`,`*val1*`,`*val2*`,`*val3*`...)`
  - Les *nbRetirer* éléments à partir de l'index *début* sont **retirés** du tableau *t* et **remplacés** par les éléments *val1*, *val2*, *val3*… (un nombre quelconque de valeurs, possiblement aucunes)
    - Cette méthode retourne un nouveau tableau contenant **les valeurs retirées** de *t*
    - Le paramètre *début* peut être un entier négatif (position relative à la fin du tableau)
- Cette méthode permet d'**ajouter** et **retirer** des éléments à **n'importe quel endroit** dans le tableau

Exemples de splice

```
var t = ["a","b","c","d","e"];
//        0   1   2   3   4


print(t.splice(2,2));
print(t);
```

```
c,d
a,b,e
```

```
print(t.splice(2,0,"C","D"));
print(t);
```

(tableau vide)

```
a,b,C,D,e
```

```
print(t.splice(1,2,"X"));
print(t);
```

```
b,C
a,X,D,e
```

```
print(t.splice(-1,1,"Y"));
print(t);
```

```
e     (tableau de longueur 1)
a,X,D,Y
```

Méthode non-modifiante d'ordre supérieur reduce

- *t* **.reduce(***fn* **,***valInitiale***)**
  - La fonction *fn* est utilisée pour combiner les éléments du tableau *t*
  - Cette opération c'est la **réduction**
  - Par exemple si *t* contient 3 éléments alors le calcul effectué est :

    fn **(***fn* **(***fn* **(***valInitiale* **,***t***[0]) ,***t***[1]) ,***t***[2])**
  - Si *valInitiale* n'est pas fourni, le calcul est :

    fn **(***fn* **(***t***[0] ,***t***[1]) ,***t***[2])**

Exemples de reduce

```
var t = [1,1,0,1];

var add = function (x,y) {
    return x+y;
};


print(t.reduce(add, 5));

print(t.reduce(add));

print(t.reduce(function (x,y) { return x*10+y; }));

print(t.reduce(function (x,y) { return x*2+y; }));

print(t.reduce(add, "z"));
```

8

3

1101

13

z1101
(texte)

Exemples de reduce

```
var somme = function (tab) {
    var s = 0;
    for (var i=0; i<tab.length; i++) {
        s += tab[i];
    }
    return s;
};
```

```
var somme = function (tab) {
    return tab.reduce(function (x,y) { return x+y; }, 0);
};
```

Exemple : renverser un tableau

- **Spécification** : écrire une fonction **`renverser`** qui prend un tableau *t* en paramètre et qui retourne **un nouveau tableau** contenant les éléments de *t* en ordre inverse
- Le tableau *t* ne doit pas être modifié (donc on ne peut pas utiliser directement la méthode **`reverse`**)

Version #1 : for et push

```
var renverser = function (t) {
    var resultat = [];
    for (var i=t.length-1; i>=0; i--) {
        resultat.push(t[i]);
    }
    return resultat;
};

var tab = [11,22,33];

print(renverser(tab));   // 33,22,11

print(tab);              // 11,22,33
```

Version #2 : for et unshift

```
var renverser = function (t) {
    var resultat = [];
    for (var i=0; i<t.length; i++) {
        resultat.unshift(t[i]);
    }
    return resultat;
};

var tab = [11,22,33];

print(renverser(tab));   // 33,22,11

print(tab);              // 11,22,33
```

Version #3 : forEach et unshift

```
var renverser = function (t) {
    var resultat = [];
    t.forEach(function (x) { resultat.unshift(x); });
    return resultat;
};

var tab = [11,22,33];

print(renverser(tab));   // 33,22,11

print(tab);              // 11,22,33
```

Version #4 : slice et reverse

```
var renverser = function (t) {
    return t.slice().reverse();
};

var tab = [11,22,33];

print(renverser(tab));   // 33,22,11

print(tab);              // 11,22,33
```

Version #5 : slice, pop et push

```
var renverser = function (t) {
    var resultat = [];
    var copie = t.slice();
    while (copie.length > 0) {
        resultat.push(copie.pop());
    }
    return resultat;
};

var tab = [11,22,33];

print(renverser(tab));  // 33,22,11

print(tab);             // 11,22,33
```

Version #6 : slice, shift et unshift

```
var renverser = function (t) {
    var resultat = [];
    var copie = t.slice();
    while (copie.length > 0) {
        resultat.unshift(copie.shift());
    }
    return resultat;
};

var tab = [11,22,33];

print(renverser(tab));  // 33,22,11

print(tab);             // 11,22,33
```

Version #7 : slice et splice

```
var renverser = function (t) {
    var resultat = [];
    var copie = t.slice();
    while (copie.length > 0) {
        resultat.splice(0,0,copie.splice(0,1)[0]);
    }
    return resultat;
};

var tab = [11,22,33];

print(renverser(tab));  // 33,22,11

print(tab);             // 11,22,33
```

Version #8 : reduce et concat

```
var renverser = function (t) {
    return tab.reduce(function (x,y) { return [y].concat(x); }, []);
};

var tab = [11,22,33];

print(renverser(tab));   // 33,22,11

print(tab);              // 11,22,33
```

Exemple : file d'attente

- **Spécification** : écrire les fonctions **`arriver`** et **`servir`** qui simulent le comportement d'une file d'attente, c'est-à-dire :

```
var arriver = function (x) { … };
var servir = function () { … };

arriver("julie");
arriver("martin");

print(servir()); // imprime: julie

arriver("paul");

print(servir()); // imprime: martin
print(servir()); // imprime: paul
```