Cours 7½

Tableaux d'IDs, push, pop

Intro. à la programmation - Aut. 2021

Menu du jour

- Révision
- Push et Pop
  - Ajouter / Retirer un élément d'un tableau
- Tableaux et IDs

❖ Tableaux

♦ Déclarer une variable qui contient un tableau

```
let monTableau = [valeur1, valeur2, valeur3, ...];
```

♦ Accéder à une valeur dans un tableau

```
monTableau[index]
```

- **index** est une valeur située entre **0** et **monTableau.length - 1**

♦ Modifier une valeur dans un tableau

```
monTableau[index] = nouvelleValeur;
```

❖ Tableaux et boucles

- ♦ Les boucles qui parcourent un tableau ont habituellement la forme suivante :

```
for(let index = 0; index < monTableau.length; index++){
      // Faire quelque chose avec monTableau[index]
}
```

monTableau // monTableau.length vaut 4

Index ->	0	1	2	3
Valeur ->	"Pomme"	"Banane"	"Cerise"	"Orange"

❖ Constantes

- ♦ Exactement comme une variable, mais interdit de modifier sa valeur
  - ○ Déclaration : **`const NOM_CONSTANTE = valeur;`**

```
const PI = 3.1415;


PI = 3.14;
PI = PI + 1;
```

```
let PI = 3.1415;


PI = 3.14;
PI = PI + 1;
```

Push et Pop

- Push() +
  - Permet d'ajouter un élément à la fin d'un tableau

```
let couleurs = ["Bleu", "Rouge", "Jaune", "Vert"];
// couleurs.length vaut 4

couleurs.push("Violet");
// couleurs vaut ["Bleu", "Rouge", "Jaune", "Vert", "Violet"]
// couleurs.length vaut 5
```

```
["Bleu", "Rouge", "Jaune", "Vert"] -> ["Bleu", "Rouge", "Jaune", "Vert", "Violet"]
```

Push et Pop

❖ Pop() 

- ♦ Permet de retirer un élément à la fin du tableau

```
let notes = [68, 71, 93, 78];
// notes.length vaut 4

notes.pop();
// notes vaut [68, 71, 93]
// notes.length vaut 3
```

[68, 71, 93, 78] -> [68, 71, 93]

Push et Pop

❖ Pop() 

- ♦ Ce n'est pas tout ! On peut aussi récupérer l'élément qui vient d'être retiré.
  - ○ Retirer un élément **sans le récupérer** :

```
let notes = [68, 71, 93, 78];
notes.pop();
// notes vaut [68, 71, 93]
```

  - ○ Retirer un élément **et le récupérer** :

```
let notes = [68, 71, 93, 78];
let noteRetiree = notes.pop(); // noteRetiree vaut 78
// note vaut [68, 71, 93]
```

Tableaux et IDs

❖ Boucles et IDs

- ♦ À la semaine 4, nous avions vu comment profiter des **boucles** pour faire des opérations **DOM** sur des éléments avec des *IDs similaires* ...

```
for(let index = 1; index < 4; index++){
    document.getElementById("daenerys" + index).classList.add("image");
}
```

```
<img id="daenerys1" src="images/daenerys1.png" alt="Daenerys contente">
<img id="daenerys2" src="images/daenerys2.png" alt="Daenerys fâchée">
<img id="daenerys3" src="images/daenerys3.png" alt="Daenerys écoeurée">
```



```
<img id="daenerys1" class="image" src="images/daenerys1.png" alt="Daenerys contente">
<img id="daenerys2" class="image" src="images/daenerys2.png" alt="Daenerys fâchée">
<img id="daenerys3" class="image" src="images/daenerys3.png" alt="Daenerys écoeurée">
```

❖ Boucles et IDs

- ♦ Cette astuce était possible car la seule différence entre les **IDs** étaient un nombre qui pouvait être représenté par l'**index** de la boucle.

```
document.getElementById("daenerys1").classList.add("image");

document.getElementById("daenerys2").classList.add("image");

document.getElementById("daenerys3").classList.add("image");
```

```
for(let index = 1; index < 4; index++){
    document.getElementById("daenerys" + index).classList.add("image");
}
```

❖ Boucles et IDs

- ♦ Toutefois, si nos **IDs** sont trop différents... cette astuce ne fonctionne pas 😭

```
document.getElementById( elementId: "chat").classList.add("cacher");
document.getElementById( elementId: "chien").classList.add("cacher");
document.getElementById( elementId: "perruche").classList.add("cacher");
```

```
for(let index = 0; index < 3; index++){
    document.getElementById(/* ??? */).classList.add("cacher");
}
```

❖ Tableaux et IDs

- ♦ C'est là que les **tableaux** viennent à la rescousse ! 🦸✊
  - Il faut commencer par créer un **tableau** qui contient les **IDs** des éléments pour lesquels nous souhaitons faire des opérations similaires :

```
let tableauIds = ["chat", "chien", "perruche"];
```

  - Ensuite, on prépare une **boucle** qui servira à parcourir le **tableau** :

```
for(let index = 0 ; index < tableauIds.length; index++){
    // Instruction ici
}
```

❖ Tableaux et IDs

- Finalement, on crée les instructions qui permettront de faire les modifications similaires à nos éléments :

```
let tableauIds = ["chat", "chien", "perruche"];

for(let index = 0 ; index < tableauIds.length; index++){
    document.getElementById(tableauIds[index]).classList.add("cacher");
}
```

`tableauIds[index]` vaudra, sucessivement, `"chat"`, puis `"chien"`, puis finalement `"perruche"`

❖ Tableaux et ID

♦ En résumé :

```
document.getElementById( elementId: "chat").classList.add("cacher");
document.getElementById( elementId: "chien").classList.add("cacher");
document.getElementById( elementId: "perruche").classList.add("cacher");
```

```
let tableauIds = ["chat", "chien", "perruche"];

for(let index = 0 ; index < tableauIds.length; index++){
    document.getElementById(tableauIds[index]).classList.add("cacher");
}
```

- Bien entendu, avec un **tableau** qui contient plus d'**IDs**, ça devient encore plus pertinent !