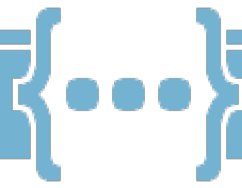


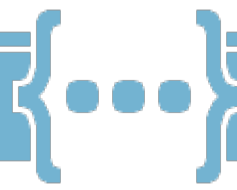
# Cours 4

DOM (classes et attributs) et boucles

Intro. à la programmation - Aut. 2021



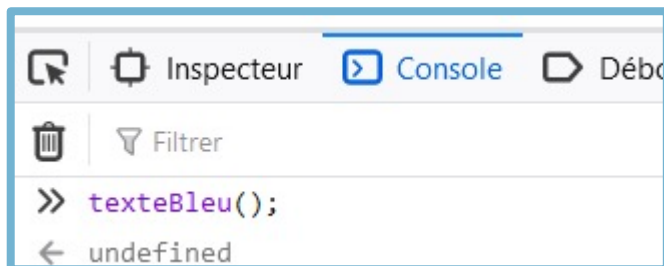
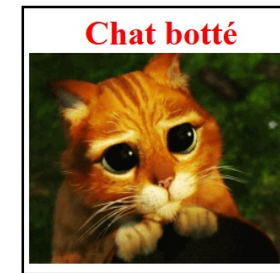
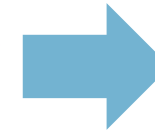
- ❖ Révision
- ❖ DOM
  - ◆ Classes
  - ◆ Attributs
  - ◆ Astuce avec DOM
- ❖ Boucles (Répétition)



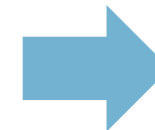
## ❖ Changer un **style** avec DOM

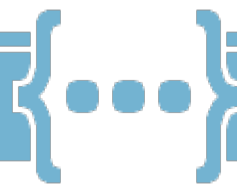
◆ `document.getElementById("id").style.propriété = "valeur";`

```
<h1 id="titre">Chat botté</h1>
```



```
<h1 id="titre">Dragon</h1>
```





## ❖ Variables globales / locales

- ◆ **Variable locale** : la variable « **couleur** » est déclarée dans une fonction. Elle ne peut être utilisée que dans cette fonction !

```
function texteBleu(){  
  let couleur = "blue";  
  document.getElementById("titre").style.color = couleur;  
}  
  
function fondBleu(){  
  document.getElementById("titre").style.backgroundColor = couleur;  
}
```

- ◆ **Variable globale** : la variable « **couleur** » est déclarée en dehors de toute fonction, au début du code **js**.
- ◆ Elle est donc utilisable n'importe où dans les **fonctions** qui suivent.

```
<> index.html  JS script.js  X  
js > JS script.js > ...  
1  let couleur = "blue";  
2  
3  function texteBleu(){  
4    document.getElementById("titre").style.color = couleur;  
5  }  
6  
7  function fondBleu(){  
8    document.getElementById("titre").style.backgroundColor = couleur;  
9  }
```



## ❖ Événements

`document.getElementById("id").addEventListener('type', nom_fonction)`

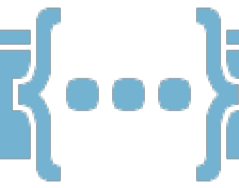
click, mouseover ou  
mouseout

Élément associé à l'événement

Déclencheur et fonction



```
function fondRouge(){  
    document.getElementById( elementId: "bouton").style.backgroundColor = "red";  
}
```



### ❖ Mot-clé **this**

- ◆ **this** représente n'importe quel **élément HTML** qui appelle la **fonction** suite au déclenchement d'un **événement**.

```
function vert(){  
    this.style.color = "limegreen";  
}
```

- Disons que c'est l'élément avec l'id « **piano** » qui appelle la fonction **vert()**, c'est comme si on avait ...

```
function vert(){  
    document.getElementById("piano").style.color = "limegreen";  
}
```

**this** « remplace » ceci



❖ DOM permet de changer le style d'un élément, mais également :

- ◆ Ajouter une classe
  - ◆ Retirer une classe
  - ◆ Vérifier si un élément possède une classe
- 
- ◆ Ajouter un attribut
  - ◆ Retirer un attribut
  - ◆ Modifier un attribut



## ❖ Classes des éléments HTML

- ◆ Les éléments **HTML** possèdent parfois une ou plusieurs **classes**

```
<div id="container" class="spongebob"> ... </div>
```

```
<div class="spongebob patrick"> ... </div>
```

- Notez que lorsqu'un élément HTML a plus d'une classe, elles sont séparées par des **espaces**.

- ◆ Les **classes** permettent d'appliquer un groupe de styles à plusieurs éléments.

```
<div class="spongebob">Bob</div>
<div class="spongebob">Patrick</div>
<div class="spongebob">Carlos</div>
```

```
.spongebob{
  color: ■ yellow;
  background-color: ■ lightskyblue;
}
```

Bob  
Patrick  
Carlos





## ❖ Ajouter une classe :

```
document.getElementById("id").classList.add("nouvelle_classe")
```



```

```



```

```



## ❖ Supprimer une classe :

```
document.getElementById("id").classList.remove("ancienne_classe")
```



```

```



```

```

Le morceau de code **HTML** `class=""` reste malgré tout présent, mais ce n'est pas grave. La **classe** est bel et bien retirée.

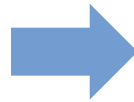


## ❖ « Basculer » la présence d'une classe dans un élément

- ◆ Donc si elle est présente, la retire. Si elle est absente, l'ajoute.
- ◆ Syntaxe : `document.getElementById("id").classList.toggle("classe")`



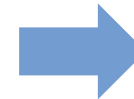
``



``

OU

``



``



## ❖ « Vérifier » si un élément **possède une classe**

- ◆ `document.getElementById("id").classList.contains("nom_classe")`
- ◆ Si l'élément possède la classe, le résultat est « **true** », sinon « **false** ».
  - Exemple

```
<li id="un" class="allo">Réchauffer des petits plats.</li>
```

```
let a = document.getElementById("un").classList.contains("allo");  
// a contient « true »
```

```
let b = document.getElementById("un").classList.contains("bye");  
// b contient « false »
```



- ❖ Les **éléments HTML** possèdent parfois un ou plusieurs **attributs**
  - ♦ Ils sont situés dans la **balise ouvrante**.

```
<p>Pas d'attributs</p>
```

Attribut

Sa valeur

```
<p title="Titre">Un attribut</p>
```

```
<p id="banniere" title="Bannière">Deux attributs</p>
```

```
<p id="banniere" class="titre" title="Bannière">Trois attributs</p>
```



## ❖ Ajouter un attribut à un élément HTML :

```
document.getElementById("id").setAttribute("nomAttribut", "valeur");
```

- Ex : `document.getElementById("babyshark").setAttribute("title", "doodoo");`

```
<div id="babyShark"> ... </div>
```



```
<div id="babyShark" title="doodoo"> ... </div>
```

## ❖ Modifier un attribut pour un élément HTML : (Même syntaxe)

- Ex : `document.getElementById("babyshark").setAttribute("title", "Baby shark doo doo doo doo");`

```
<div id="babyShark" title="texte qui va changer"> ... </div>
```



```
<div id="babyShark" title="Baby shark doo doo doo doo"> ... </div>
```






### ❖ Retirer un attribut à un élément HTML :

```
document.getElementById("id").removeAttribute("nomAttribut");
```

○ Exemple : `document.getElementById("babyShark").removeAttribute("style");`

`<div id="babyShark" style="color:yellow;"> ... </div>`  `<div id="babyShark"> ... </div>`



## ❖ « Obtenir » la **valeur** d'un attribut

```
document.getElementById("id").getAttribute("nomAttribut");
```

○ Exemple : `let image = document.getElementById("babyshark").getAttribute("src");`

```

```

- La variable **image** contient donc la valeur "images/DooDooDoo.png".





❖ Donc **DOM** permet, entre autres, ...

- ◆ D'ajouter, modifier et retirer des **classes**
- ◆ D'ajouter, modifier et retirer des **styles**
- ◆ D'ajouter, modifier et retirer **n'importe quel attribut**, en fait.

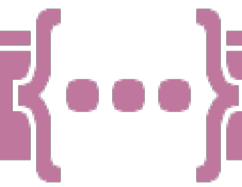
◆ D'autres exemples

- daddyShark.setAttribute("style", "color:blue; font-size:15px;");

`<div id="daddyShark"> ... </div>`  `<div id="daddyShark" style="color:blue; font-size:15px;"> ... </div>`

- mamaShark.setAttribute("src", "images/pinkShark.png");

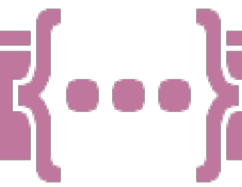
`<img id="mamaShark" alt="Mama shark">`  ``



### ❖ Vous devez manipuler fréquemment des éléments HTML avec DOM...

```
document.getElementById("renard").textContent = "Blablala";  
document.getElementById("renard").style.color = "orange";  
document.getElementById("renard").style.borderWidth = "5px";  
document.getElementById("renard").classList.add("animal");  
document.getElementById("renard").setAttribute("style", "fontSize:13px");
```

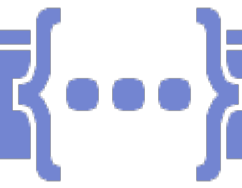
◆ Constamment réécrire `document.getElementById("...")` vous épuise ?



- ❖ Vous pouvez « ranger » une expression qui permet d'accéder à un **élément HTML** dans une variable :

```
function modifierRenard(){  
    let idRenard = document.getElementById("renard");  
  
    idRenard.textContent = "Blablala";  
    idRenard.style.color = "orange";  
    idRenard.style.borderWidth = "5px";  
    idRenard.classList.add("animal");  
    idRenard.setAttribute("style", "fontSize:13px");  
}
```

- ◆ Pas besoin de réécrire `document.getElementById("...")` à chaque fois pour l'élément **renard** !
  - Vous pouvez le faire avec n'importe quel **élément HTML** dès que vous comptez l'utiliser avec **DOM** à plusieurs reprises.



## ❖ Répéter des bouts de code similaires...

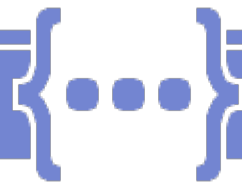
```
let nbAmis = 2;
let idJournal = document.getElementById("journal");

nbAmis++; // nbAmis = 3
idJournal.textContent += "J'ai maintenant " + nbAmis + " amis !";

nbAmis++; // nbAmis = 4
idJournal.textContent += "J'ai maintenant " + nbAmis + " amis !";

nbAmis++; // nbAmis = 5
idJournal.textContent += "J'ai maintenant " + nbAmis + " amis !";
```

◆ Il doit bien y avoir moyen de répéter le code sans le réécrire en entier ?



## ❖ Boucles

◆ Permettent de **répéter** des bouts de code !

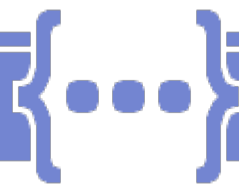
◆ Syntaxe :

```
for(initialisation; condition d'exécution; incrémentation) {  
    // Code à répéter  
}
```

◆ Exemple :

- Cette boucle se répète 2 fois

```
          Initialisation      Condition      Incrémentation  
for(let index = 1; index < 3; index++){  
    // Code à répéter  
}
```



## ❖ Boucles

### ◆ Fonctionnement

#### Initialisation

On crée une **variable locale** (qui n'existe que pour la durée de la boucle) avec une **valeur initiale**.

```
int index = 1
```

#### Condition d'exécution

La boucle n'est pas répétée (et est donc quittée) lorsque cette **condition** devient fausse.

```
index < 3
```

#### Incrémentation

À chaque fois qu'on répète la boucle, la valeur de cette **variable locale** évolue.

```
index++
```

```
for(let index = 1; index < 3; index++){  
    // Code à répéter  
}
```

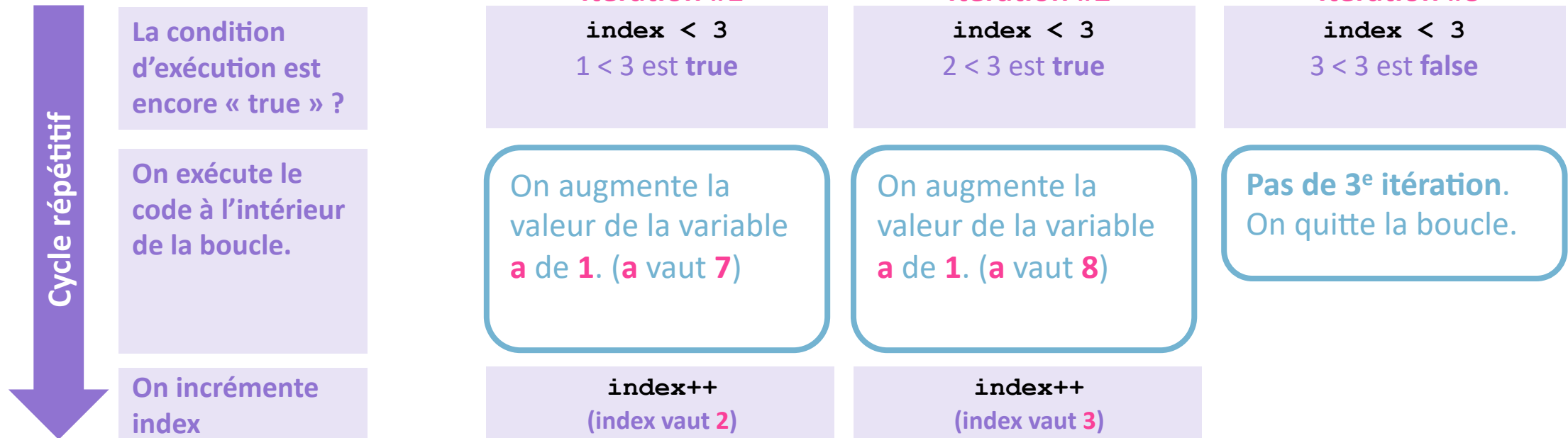


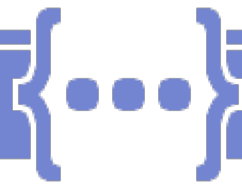
## ❖ Boucles

### ◆ Exemple de déroulement pour la **boucle** ci-droit.

- La variable **index** commence avec la valeur **1**.

```
let a = 6;  
  
for(let index = 1; index < 3; index++){  
    a++;  
}
```





## ❖ Boucles

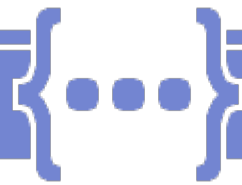
### ◆ Exemple 1

- Cette boucle fait **4 itérations**
- À chaque itération, on incrémente la variable « **valeur** » avec la valeur de l'**index**.

```
let valeur = 0;  
  
for(let index = 0; index < 4; index++){  
    valeur = valeur + index;  
}  
  
alert("Valeur finale : " + valeur);
```

- La valeur finale est : 0 + 0 + 1 + 2 + 3 (Donc 6)





## ❖ Boucles

### ◆ Exemple 2

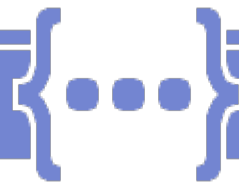
- Cette boucle fait **9 itérations**.
- On se sert de la variable **index** à chaque itération pour ajouter du texte.

```
let idNombres = document.getElementById("nombres");  
  
for(let index = 1; index < 10; index++){  
  idNombres.textContent = idNombres.textContent + " " + index;  
}
```

<div id="nombres"></div>



<div id="nombres"> 1 2 3 4 5 6 7 8 9</div>



## ❖ Boucles

### ◆ Exemple 3

- Cette boucle fait **3 itérations**. Elle ajoute donc la classe **image** à 3 éléments HTML.



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

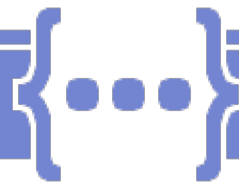
```
  
  

```



```
  
  

```



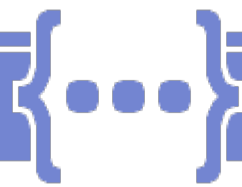
## ❖ Boucles

### ◆ Construire une boucle

- Commencez par analyser un **code répétitif** et trouvez les **différences**.

```
document.getElementById("daenerys1").classList.add("image");  
document.getElementById("daenerys2").classList.add("image");  
document.getElementById("daenerys3").classList.add("image");
```

- La seule chose qui varie dans ces **3 instructions**, c'est le **numéro** à la fin de l'**id** ...
  - On a donc besoin d'une **boucle** où l'**index** vaudra...
    - **1** pour la première itération
    - **2** pour la deuxième itération
    - **3** pour la troisième itération



## ❖ Boucles

### ◆ Construire une boucle

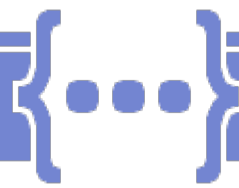
- On a besoin d'une **boucle** où l'**index** vaudra...

- **1** pour la première itération
- **2** pour la deuxième itération
- **3** pour la troisième itération

```
for(let index = 1; index < 4; index++){  
    // ...  
}
```

- Il reste à intégrer le code et à se servir de la variable **index** pour remplacer la partie qui doit varier d'itération en itération

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



## ❖ Boucles

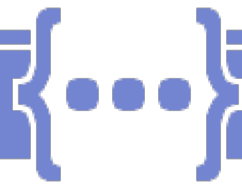
### ◆ Construire une boucle

```
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");  
}
```

- Si jamais on ajoute 2 images supplémentaires avec les id « **daenerys4** » et « **daenerys5** », il suffira de changer la **condition d'exécution** pour **index < 6**



## ❖ Boucles

### ◆ Exemple 4

- Une **boucle** ne s'incrmente pas forcément de 1 à **chaque itération**

```
for(let i = 1; i < 7; i = i + 2){  
    // Code quelconque  
}
```

### ◆ Combien d'**itérations** fera cette boucle ?

- 3 itérations !
  - 1<sup>ère</sup> itération : **i** vaut **1**.
  - 2<sup>e</sup> itération : **i** vaut **3**.
  - 3<sup>e</sup> itération : **i** vaut **5**.
  - Pas de 4<sup>e</sup> itération car **i** vaut **7** et cela viole la condition d'exécution.