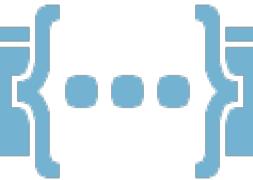


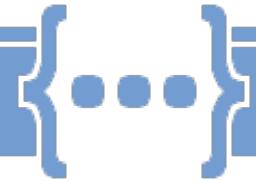
Semaine 7

DOM (classes et attributs) et boucles

Intro. à la programmation - Aut. 2022



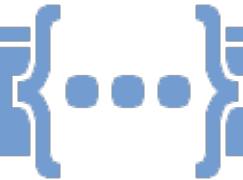
- ❖ Révision
- ❖ DOM
 - ◆ Classes
 - ◆ Attributs
 - ◆ Astuce avec DOM
- ❖ Boucles (Répétition)



❖ 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>
```

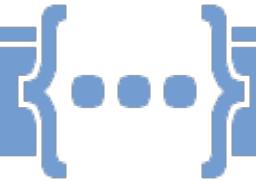
- Notez que lorsqu'un élément HTML a plus d'une classe, elles sont séparées par des **espaces**.

- ◆ 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.querySelector("#id").classList.add("nouvelle_classe")
```

```
document.querySelector("#smudge").classList.add("cat");
```

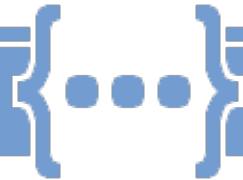
```

```



```

```



❖ Supprimer une classe :

```
document.querySelector("#id").classList.remove("ancienne_classe")
```

```
document.querySelector("#smudge").classList.remove("cat");
```

```

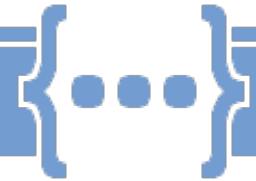
```



```

```

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



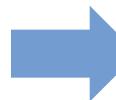
❖ « Basculer » la présence d'une classe dans un élément

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

```
document.querySelector("#smudge").classList.toggle("cat");
```

```

```



```

```

OU

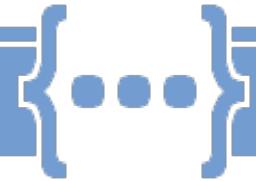
```

```



```

```



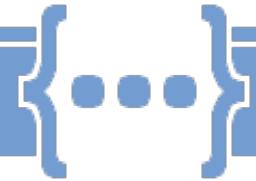
❖ « Vérifier » si un élément **possède** une classe

- ◆ `document.querySelector("#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.querySelector("#un").classList.contains("allo");  
// a contient « true »
```

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



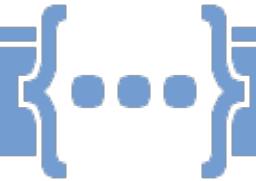
❖ « Vérifier » si un élément **possède** une classe

- ◆ `document.querySelector("#id").classList.contains("nom_classe")`
- ◆ Exemple de fonction qui exploite `classList.contains()` :

```
<p id="texte" class="sobre">Ton thé t'a-t-il ôté ta toux ?</p>
```

Si l'élément `#texte` possède la classe `sobre`, son texte devient "Je possède la classe sobre 😊". Sinon, son texte devient "Je ne possède pas la classe sombre 😢".

```
function verifierClasse(){  
  if(document.querySelector("#texte").classList.contains("sobre") == true){  
    document.querySelector("#texte").textContent = "Je possède la classe sobre 😊";  
  }  
  else{  
    document.querySelector("#texte").textContent = "Je ne possède pas la classe sobre 😢";  
  }  
}
```

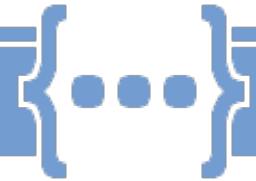


- ❖ Les éléments HTML possèdent parfois un ou plusieurs **attributs**
 - ◆ Ils sont situés dans la **balise ouvrante**.

<p>Pas d'attributs</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.querySelector("#id").setAttribute("nomAttribut", "valeur");
```

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

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



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

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

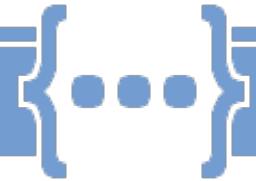
- Ex : `document.querySelector("#babyshark").setAttribute("title", "Baby shark doo doo doo doo");`

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



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



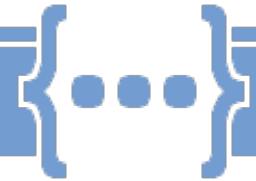


❖ Retirer un attribut à un élément HTML :

```
document.querySelector("#id").removeAttribute("nomAttribut");
```

- Exemple : `document.querySelector("#babyShark").removeAttribute("style");`

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



❖ « Obtenir » la valeur d'un attribut

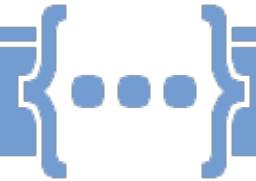
```
document.querySelector("#id").getAttribute("nomAttribut");
```

```

```

- Exemple : let **image** = document.querySelector("#babyShark").**getAttribute**("src");
 - La variable **image** contient donc la valeur "images/doodoo.png".

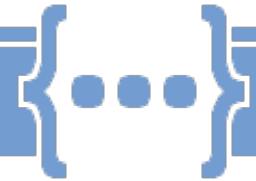
```
function verifierImage(){  
  if(document.querySelector("#babyShark").getAttribute("src") == "images/doodoo.png"){  
    alert("C'est la bonne image !");  
  }  
  else{  
    alert("Ce n'est pas la bonne image 😞");  
  }  
}
```



❖ Résumé de toutes les modifications qu'on peut faire sur le DOM

```
document.querySelector("#id") . . .
```

- ◆ Obtenir / modifier / ajouter du **contenu textuel**
 - `.textContent`
- ◆ Modifier les **styles**
 - `.style.propriété = "valeur"`
- ◆ Ajouter un **événement** (Rendre interactif un élément)
 - `.addEventListener("type", fonction)`
- ◆ Modifier / obtenir des **classes**
 - `.classList.add("maClasse")`
 - `.classList.remove("maClasse")`
 - `.classList.toggle("maClasse")`
 - `.classList.contains("maClasse")`
- ◆ Modifier / obtenir des **attributs**
 - `.setAttribute("attribut", "valeur")`
 - `.getAttribute("attribut")`
 - `.removeAttribute("attribut")`



❖ Usage de this

- ◆ Notez qu'on peut très bien utiliser **this** pour modifier les classes et attributs !
 - Cela permet de modifier les classes ou attributs de l'élément avec lequel on vient d'interagir (clic, survol ou fin du survol)
- ◆ Modifier / obtenir des **classes**
 - `this.classList.add("maClasse")`
 - `this.classList.remove("maClasse")`
 - `this.classList.toggle("maClasse")`
 - `this.classList.contains("maClasse")`
- ◆ Modifier / obtenir des **attributs**
 - `this.setAttribute("attribut", "valeur")`
 - `this.getAttribute("attribut")`
 - `this.removeAttribute("attribut")`

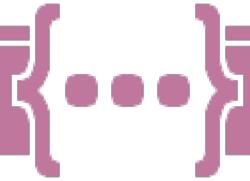


- ❖ Vous devez manipuler fréquemment des éléments HTML avec DOM...



```
document.querySelector("#mario").textContent = "Mario brosse 🍷";  
document.querySelector("#mario").style.color = "red";  
document.querySelector("#mario").style.borderWidth = "5px";  
document.querySelector("#mario").classList.add("goomba");  
document.querySelector("#mario").classList.toggle("mushroom");  
document.querySelector("#mario").setAttribute("title", "Plombier");
```

- ◆ Constamment réécrire `document.querySelector("#... ")` vous épuise ? 😔

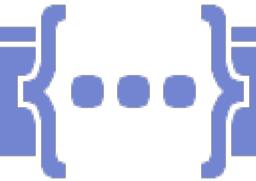


- ❖ Vous pouvez « ranger » une expression qui permet d'accéder à un élément HTML dans une variable 



```
function modifierMario(){  
  
    let elementMario = document.querySelector("#mario");  
  
    elementMario.textContent = "Mario brosse 🍷";  
    elementMario.style.color = "red";  
    elementMario.style.borderWidth = "5px";  
    elementMario.classList.add("goomba");  
    elementMario.classList.toggle("mushroom");  
    elementMario.setAttribute("title", "Plombier");  
  
}
```

- ◆ Pas besoin de réécrire `document.querySelector("#...")` à chaque fois pour l'élément `#mario` !
 - Vous pouvez le faire avec n'importe quel élément HTML dès que vous comptez le modifier à plusieurs reprises.



❖ Répéter des bouts de code similaires...

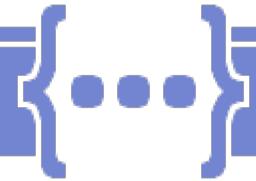
```
let nbAmis = 2;
let elementJournal = document.querySelector("#journal");

nbAmis += 1; // Vaut 3
elementJournal.textContent = "J'ai maintenant " + nbAmis + " amis !";

nbAmis += 1; // Vaut 4
elementJournal.textContent = "J'ai maintenant " + nbAmis + " amis !";

nbAmis += 1; // Vaut 5
elementJournal.textContent = "J'ai maintenant " + nbAmis + " amis !";
```

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



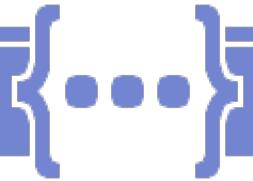
❖ Boucles

- ◆ Permettent de **répéter** des bouts de code !
- ◆ **Syntaxe :**

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

- ◆ **Exemple :**
 - Cette boucle se répète 2 fois

Initialisation	Condition d'exécution	Incrémantation
for(let index = 1; index < 3; index += 1){		
// 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**.

```
let index = 1
```

Condition d'exécution

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

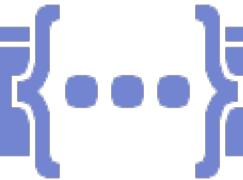
```
index < 3
```

Incrémantation

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

```
index += 1
```

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

La condition d'exécution est encore « true » ?

On exécute le code à l'intérieur de la boucle.

On incrémente index

Itération #1

index < 3
1 < 3 est **true**

On augmente la valeur de la variable **a** de **1**. (**a** vaut **7**)

index += 1
(**index** vaut **2**)

Itération #2

index < 3
2 < 3 est **true**

On augmente la valeur de la variable **a** de **1**. (**a** vaut **8**)

index += 1
(**index** vaut **3**)

Itération #3

index < 3
3 < 3 est **false**

Pas de 3^e itération.
On quitte la boucle.

Cycle répétitif



❖ Boucles

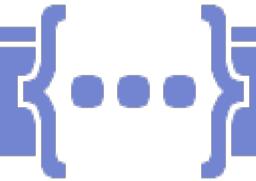
- ◆ Exemple de déroulement pour la **boucle** ci-droit.
 - La variable **index** commence avec la valeur **1**.

```
let a = 6;

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

```
let a = 6;

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



❖ Boucles

◆ Exemple 1

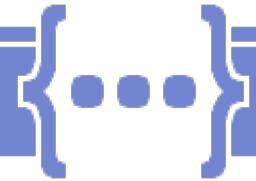
- Cette boucle fait **4 itérations**
- À chaque itération, on incrémente la variable « **valeur** » avec la valeur de l'**index**.

```
let valeur = 10;

for(let index = 0; index < 4; index += 1){
    valeur += index;
}

alert("Valeur finale : " + valeur);
```

- La valeur finale est : $10 + 0 + 1 + 2 + 3$ (Donc 16)



❖ Boucles

◆ Exemple 2

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

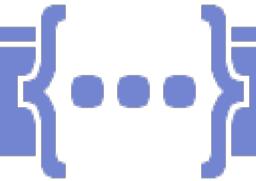
```
let elementNombres = document.querySelector("#nombres");

for(let index = 1; index < 10; index += 1){
    elementNombres.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 += 1){  
  document.querySelector("#daenerys" + index).classList.add("image");  
}
```

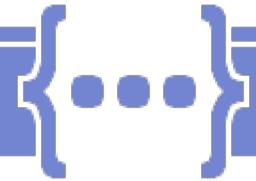
```
  
  

```



```
  
  

```



❖ Boucles

◆ Construire une boucle

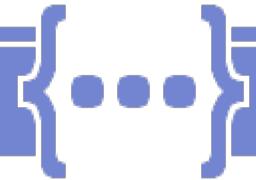
- Commencez par analyser un **code répétitif** et trouvez les **différences**.

```
document.querySelector("#daenerys1").classList.add("image");
```

```
document.querySelector("#daenerys2").classList.add("image");
```

```
document.querySelector("#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



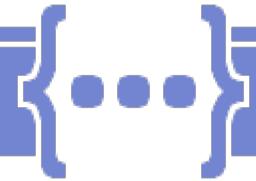
❖ Boucles

◆ 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
- 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 += 1){  
    // ...  
}
```

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



❖ Boucles

◆ Construire une boucle

```
document.querySelector("#daenerys1").classList.add("image");

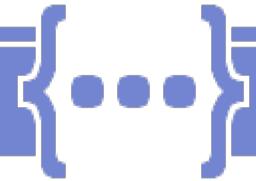
document.querySelector("#daenerys2").classList.add("image");

document.querySelector("#daenerys3").classList.add("image");
```



```
for(let index = 1; index < 4; index += 1){
  document.querySelector("#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'incrémente 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^{ère} itération : **i** vaut **1**.
 - 2^e itération : **i** vaut **3**.
 - 3^e itération : **i** vaut **5**.
 - Pas de 4^e itération car **i** vaut **7** et cela viole la condition d'exécution.