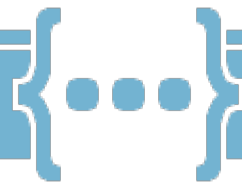


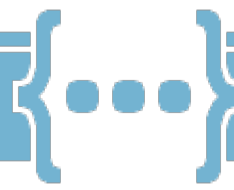
# Semaine 9

Événements clavier, planificateurs, attribut data-

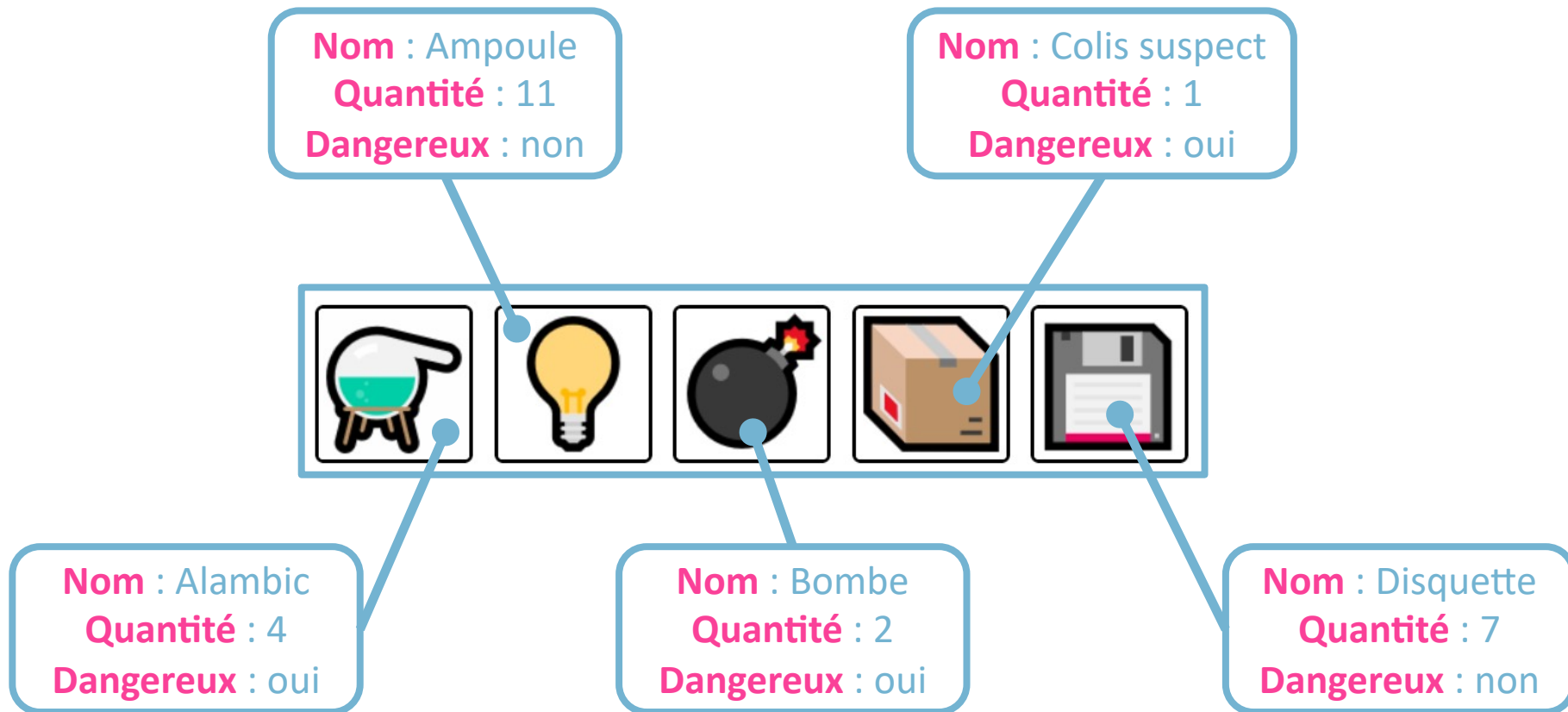
**Intro. à la programmation**

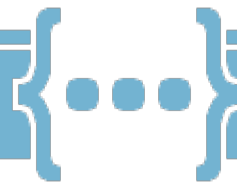


- ❖ **Attribut data-**
  - ◆ `parseInt()` et `parseFloat()`
- ❖ **Planificateurs**
  - ◆ `setTimeout`
  - ◆ `setInterval`
- ❖ **Événements clavier**
  - ◆ `keydown`



- ❖ Parfois, on veut stocker des données pour les éléments d'une page
  - ◆ Exemple : (Images dans une page Web)





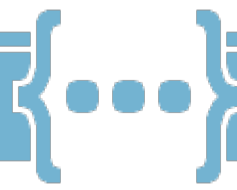
### ❖ Où stocker ces informations ?

- ◆ Dans des **tableaux** ?



```
let gNoms = ["Alambic", "Ampoule", "Bombe", "Colis suspect", "Disquette"];  
let gQuantites = [4, 11, 2, 1, 7];  
let gDangereux = ["oui", "non", "oui", "oui", "non"];
```

- ◆ Pas toujours pratique... si on change l'ordre des images dans la page où on retire un objet, ça devient vite embêtant. 😞
- ◆ Les informations sont classées par **type d'information** plutôt que par **objet**... Il faut garder à l'esprit que "Alambic", 4 et "oui" vont ensemble... par exemple.



## ❖ Attribut **data-**

- ◆ Permet de ranger n'importe quelle information dans un élément HTML

**data-nomDeLaPropriété="valeur"**



**Nom** : Alambic  
**Quantité** : 4  
**Dangereux** : Oui

```

```

- ◆ Comme les données sont stockées dans le HTML, pas besoin de les ranger dans des **tableaux** ou des **variables** dans ce cas-ci. 🥳




## ❖ Comment récupérer ces données ?

- ◆ On sait déjà comment 😎

```

```



```
let nom = document.querySelector("#alambic").getAttribute("data-nom");  
// nom contient "Alambic"
```

**getAttribute("nomAttribut")** permet de récupérer la valeur d'un attribut de notre choix. On peut, par exemple, stocker cette valeur dans une variable.



## ❖ Comment **ajouter / modifier** des données ?

- ◆ On sait déjà comment aussi 💪

```
document.querySelector("#alambic").setAttribute("data-quantite", 4);
```

```

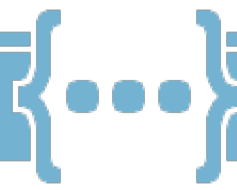
```



```

```

`setAttribute("nomAttribut", valeur)` permet d'**ajouter** ou de **modifier** un **attribut** de notre choix dans un **élément HTML**.



## ❖ Utiliser les attributs data-

◆ Exemple 1 : Alerter le nom d'un objet en cliquant dessus.

```

```

Cette fonction est appelée lorsqu'on clique sur l'image

```
function alerterNom(){  
    // Obtenir la valeur de data-nom  
    let nom = document.querySelector("#alambic").getAttribute("data-nom");  
  
    // Faire une alerte qui affiche la valeur de nom  
    alert(nom);  
}
```







## ❖ Utiliser les attributs data-

◆ Exemple 2 : Afficher toutes les données d'un objet en cliquant dessus.

```

```

Cette fonction est appelée lorsqu'on clique sur une des images

```
function afficherInfo(){  
  
    // Obtenir les données data-  
    let nom = this.getAttribute("data-nom");  
    let quantite = this.getAttribute("data-quantite");  
    let danger = this.getAttribute("data-danger");  
  
    // Afficher les données dans la page  
    document.querySelector("#nom").textContent = nom;  
    document.querySelector("#quantite").textContent = quantite;  
    document.querySelector("#danger").textContent = danger;  
}
```



Nom de l'objet : Colis suspect

Quantité : 1

Dangereux : Oui



## ❖ Utiliser les attributs data-

### ◆ Exemple 3 : Mettre une **bordure rouge** si l'objet est dangereux

```
function rougeSiDanger(){  
    let elementBombe = document.querySelector("#bombe");  
  
    if(elementBombe.getAttribute("data-danger") == "Oui"){  
        elementBombe.style.borderColor = "red";  
    }  
}
```

```

```





❖ Les **chaînes de caractères** contiennent parfois des **nombre**s.

◆ Exemples : "2", "7", "1.5", "43"

◆ Si on tente de les **additionner** sous cette forme ... ils se **concatènent** ...

"2" + "7" -> "27" 😞

◆ On peut transformer une **chaîne de caractères** en **nombre** !

○ De **chaîne de caractères** à nombre **entier** :

`parseInt("2") -> 2`

○ De **chaîne de caractères** à nombre à **virgule** :

`parseFloat("1.5") -> 1.5`



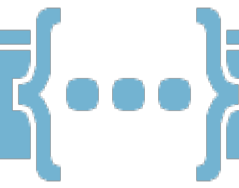
❖ Pour **additionner des nombres** qui sont sous forme de **chaîne de caractères**, il faut donc commencer par les convertir en nombre.

◆ Exemple : Additionner "5" et "2.5"

```
let x = "5";  
let y = "2.5";
```

```
x + y // vaut "52.5" 😡
```

```
parseInt(x) + parseFloat(y) // vaut 7.5 😊
```



### ❖ Utiliser les attributs data-

- ◆ Exemple 4 : Calculer la somme de la quantité de bombes et d'ampoules.

```
  

```

### ◆ Petit problème !

- Quand nous allons faire `getAttribute("data-quantite")` pour ces éléments... nous allons obtenir `"2"` plutôt que `2`, par exemple. (Une chaîne de caractères plutôt qu'un nombre)
- Nous aurons donc besoin de `parseInt(...)` pour convertir les valeurs obtenues en nombres.



## ❖ Utiliser les attributs data-

◆ Exemple 4 : Calculer la **somme** de la quantité de bombes et d'ampoules.

```
function calculerSomme(){  
  "2" → let quantiteBombe = document.querySelector("#bombe").getAttribute("data-quantite");  
  "11" → let quantiteAmpoule = document.querySelector("#ampoule").getAttribute("data-quantite");  
  
  let somme = parseInt(quantiteBombe) + parseInt(quantiteAmpoule); // 2 + 11  
  
  alert("Nombre de bombes ou d'ampoules : " + somme);  
}
```

file://

Nombre de bombes ou d'ampoules : 13

OK



## ❖ Planifier... quoi ? 🤔

- ◆ L'appel de fonctions !

## ❖ setTimeout ⌚

- ◆ Permet d'appeler une fonction ... dans x millisecondes.

- ◆ Syntaxe :

`setTimeout(maFonction, tempsEnMillisecondes)`

- Exemple : `setTimeout(afficherNom, 3000)` appellera la fonction `afficherNom()` dans 3 secondes.

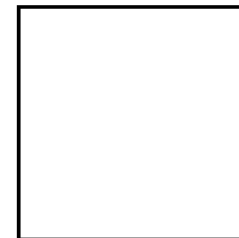
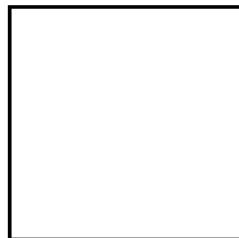


## ❖ setTimeout ⌚

◆ Exemple : Afficher puis cacher une image brièvement.

```
function boo(){  
  
    // Afficher boo dans 2 secondes  
    setTimeout(afficherBoo, 2000);  
  
    // Cacher boo dans 4 secondes  
    setTimeout(cacherBoo, 4000);  
  
}
```

```
function afficherBoo(){  
    document.querySelector("#boo").style.display = "block";  
}  
  
function cacherBoo(){  
    document.querySelector("#boo").style.display = "none";  
}
```



Visible pendant 2 secondes





## ❖ setInterval

- ◆ Permet d'appeler une fonction ... toutes les x millisecondes.
- ◆ Syntaxe :

`setInterval (maFonction, tempsEnMillisecondes)`

- Exemple : `setInterval (afficherAlerte, 3000)` appellera la fonction `afficherAlerte()` toutes les 3 secondes ! 🤖



## ❖ setInterval

- ◆ Exemple : Afficher puis cacher une image continuellement
  - La fonction **basculerVisibilite()** sera appelée **toutes les secondes**.



```
function intervalCrewmate(){  
  setInterval(basculerVisibilite, 1000);  
}  
  
function basculerVisibilite(){  
  document.querySelector("#crewmate").classList.toggle("cacher");  
}
```

```
.cacher{  
  display:none;  
}
```



## ❖ Et si on veut mettre fin à **setInterval** ?

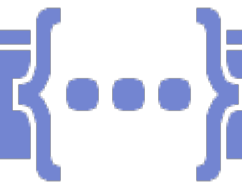
- ◆ Il existe la fonction **clearInterval()** pour arrêter un planificateur !
  - Par contre, il va falloir suivre quelques étapes pour pouvoir l'utiliser.
- ◆ **Étape 1** : Quand on utilise **setInterval()**, il faut « **stocker le planificateur à intervalle** » dans une **variable globale**.

```
function alternerCrewmate(){  
  |   gPlanificateur = setInterval(toggleCacher, 1000);  
  |  
  |  
  |  
  }  
}
```

- ◆ **Étape 2** : Quand on souhaite **arrêter le planificateur**, on utilise **clearInterval()**.

```
function stopCrewmate(){  
  |   clearInterval(gPlanificateur);  
  |  
  |  
  }  
}
```

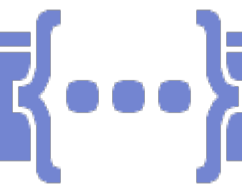
Dans ce cas-ci, on a un bouton qui permet d'appeler **stopCrewmate()**, ce qui arrête le planificateur qu'on a rangé dans la variable globale **gPlanificateur**



- ❖ Il existe un **écouteur d'événements** qui permet de savoir quand l'utilisateur appuie sur une **touche** de son **clavier**.
  - ◆ Pour pouvoir utiliser ce genre d'**événement**, il faut ajouter ceci dans notre fonction **init()** :

```
function init(){  
    // Écouteur d'événements clavier  
    document.addEventListener("keydown", toucheClavier);  
}
```

- Remarquez que cet événement n'est pas attaché à un élément HTML en particulier. Seulement au « **document** » ! (C'est-à-dire la page Web en entier)



## ❖ Comment peut-on savoir sur **quelle touche** l'utilisateur a **appuyé** ?

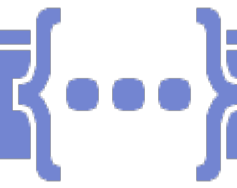
```
function init(){  
    // Écouteur d'événements clavier  
    document.addEventListener("keydown", toucheClavier);  
}
```

- ◆ D'abord, on s'assure de créer une **fonction** qui sera appelée par cet événement. Dans ce cas-ci, c'est **toucheClavier()**.
- ◆ Pour « savoir » quelle **touche** a été **appuyée**, nous allons stocker ceci dans une **variable** :

Exceptionnellement, on devra glisser une variable spéciale (sans **let**) ici.

```
function toucheClavier(event){  
    // On obtient et stocke la touche appuyée dans la variable touche  
    let touche = event.key;
```

# Événements clavier

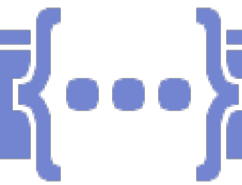


- ◆ Une fois qu'on a préparé le début de la fonction qui gère l'événement clavier comme ceci ...

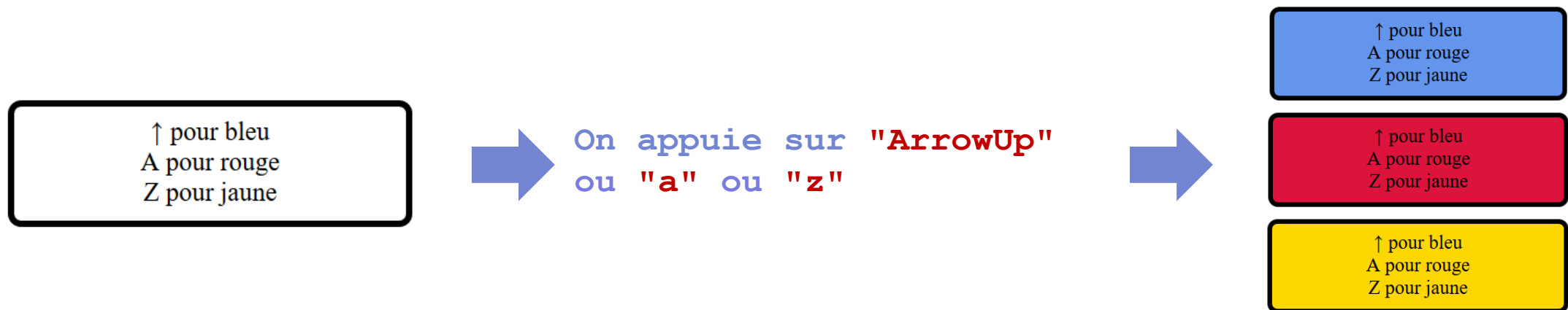
```
function toucheClavier(event){  
    // On obtient et stocke la touche appuyée dans la variable touche  
    let touche = event.key;
```

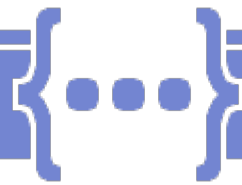
- Que contient la variable **touche** ? Ça dépend de la **touche** qui a été appuyée :





- ❖ On peut donc :
  - ◆ Savoir lorsqu'une **touche est appuyée**
  - ◆ Obtenir **quelle touche** a été appuyée
- ❖ Il nous reste à savoir ce qu'on veut faire avec ces informations !
  - ◆ Exemple : Quand on **appuie** sur une **touche**, cela change la **couleur de fond** d'un **élément HTML** :





❖ Exemple : Quand on **appuie** sur une **touche**, cela change la **couleur de fond** d'un **élément HTML** :

- ◆ Étape 1 : On ajoute notre **écouteur d'événement clavier** dans **init()**
  - Il appelle la fonction **toucheClavier()**

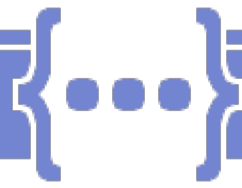
```
document.addEventListener("keydown", toucheClavier);
```

◆ Étape 2 :

- La fonction **toucheClavier(event)** va pouvoir obtenir la **touche** qui a été **appuyée** grâce à l'expression **event.key** (**event** aurait pu être nommé simplement **e** par exemple)

```
function toucheClavier(event){  
    // On obtient et stocke la touche appuyée dans la variable touche  
    let touche = event.key;
```





## ❖ Étape 3 : Que veut-on faire avec la touche appuyée ?

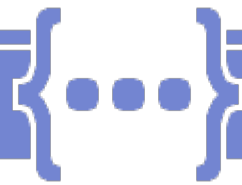
- ◆ Dans ce cas-ci, nous allons modifier la propriété backgroundColor du style de l'élément avec l'id "#clavierFond".

```
function toucheClavier(event){  
  
    let touche = event.key; // Quelle est la touche appuyée ?  
  
    let element = document.querySelector("#clavierFond");  
  
    if(touche == "ArrowUp"){  
        element.style.backgroundColor = "cornflowerblue";  
    }  
    else if(touche == "a"){  
        element.style.backgroundColor = "crimson";  
    }  
    else if(touche == "z"){  
        element.style.backgroundColor = "gold";  
    }  
  
}
```

↑ pour bleu  
A pour rouge  
Z pour jaune

↑ pour bleu  
A pour rouge  
Z pour jaune

↑ pour bleu  
A pour rouge  
Z pour jaune



### ❖ Déplacer un élément dans la page

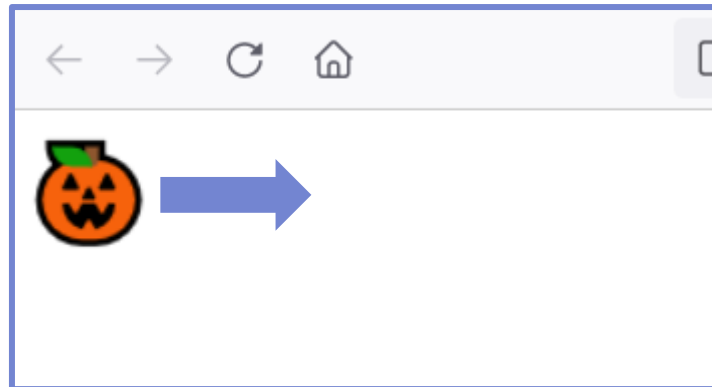
- ◆ Par exemple, on a cet élément dans la page :

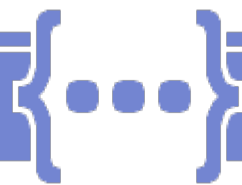
```

```



- ◆ On aimerait, quand on appuie sur "**ArrowRight**" sur le clavier, déplacer l'image de citrouille vers la droite dans la page.





## ❖ Déplacer un élément dans la page

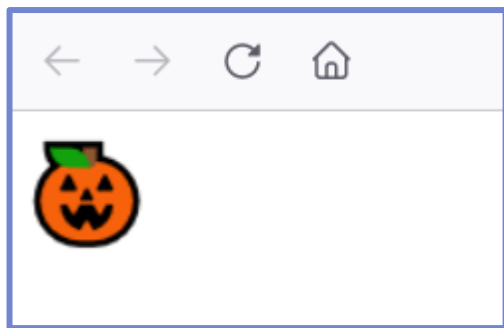
- ◆ Rappel sur les styles **left** et **top**

```

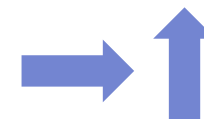
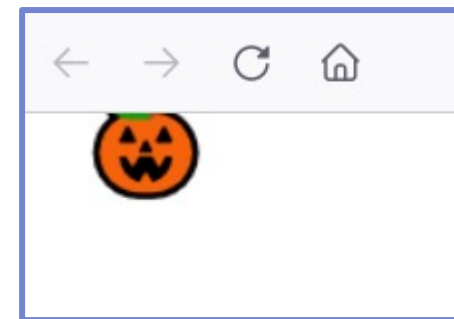
```



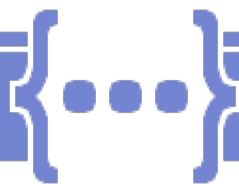
- ◆ **left** : Nombre de pixels d'espacement à gauche de l'élément.
- ◆ **top** : Nombre de pixels d'espacement en haut de l'élément.



`style="left:0px; top:0px;"`



`style="left:20px; top:-20px;"`



## ❖ Déplacer un élément dans la page

### ◆ Changer la valeur du style **left** (ou **top**)

- **Étape 1** : Ranger la valeur actuelle du style **left** dans une variable

```
let valeurLeft = document.querySelector("#pumpkin").style.left; // vaut "0px"
```

---

- **Étape 2** : Se débarrasser de "px" et ne garder qu'une valeur numérique

```
valeurLeft = parseInt(valeurLeft); // "0px" -> 0
```

---

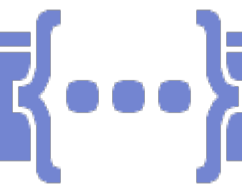
- **Étape 3** : Augmenter / réduire la valeur numérique

```
valeurLeft += 5; // 0 -> 5
```

---

- **Étape 4** : Changer le style **left** de l'élément avec la nouvelle valeur sans oublier de remettre le "px" après le nombre !

```
document.querySelector("#pumpkin").style.left = valeurLeft + "px"; // "5px"
```



## ❖ Déplacer un élément dans la page

### ◆ Changer la valeur du style **left** (ou **top**)

Utilisez la méthode que vous préférez !

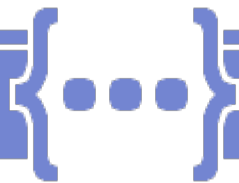
```
let valeurLeft = document.querySelector("#pumpkin").style.left; // "0px"
valeurLeft = parseInt(valeurLeft); // "0px" -> 0
valeurLeft += 5; // 0 -> 5
document.querySelector("#pumpkin").style.left = valeurLeft + "px"; // "5px"
```

### ◆ On peut aussi le faire de manière plus compacte

```
let valeurLeft = document.querySelector("#pumpkin").style.left;
document.querySelector("#pumpkin").style.left = parseInt(valeurLeft) + 5 + "px";
```

### ◆ Encore plus compacte

```
let element = document.querySelector("#pumpkin");
element.style.left = parseInt(element.style.left) + 5 + "px";
```



## ❖ Déplacer un élément dans la page

### ◆ Avec un événement clavier

```
function toucheClavier(e){  
  
    let touche = e.key; // Quelle est la touche appuyée ?  
    let element = document.querySelector("#pumpkin");  
  
    if(touche == "ArrowRight"){  
        element.style.left = parseInt(element.style.left) + 5 + "px";  
    }  
    if(touche == "ArrowLeft"){  
        element.style.left = parseInt(element.style.left) - 5 + "px";  
    }  
}
```

- Appuyer sur "ArrowRight" déplace l'élément de 5 pixels à droite.
- Appuyer sur "ArrowLeft" déplace l'élément de 5 pixels à gauche.