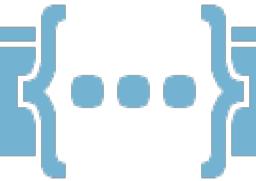


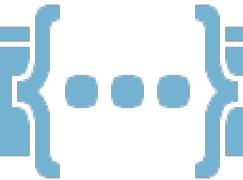
Semaines 12 et 13

Fonctions : paramètres et valeurs de retour

[Intro. à la programmation](#)



- ❖ Fonctions avec paramètres
- ❖ Fonctions avec valeur de retour
- ❖ Fonctions avec paramètres et retour



❖ Fonctions sans/avec paramètres

Fonctions sans paramètre

```
monTableau.pop();  
  
supprimerImage();
```

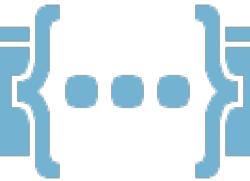
pop() et **supprimerImage()** sont des fonctions sans paramètre.

Fonctions avec paramètre(s)

```
monTableau.push("chat");  
  
let element = document.querySelector("#imageBalle");  
  
element.setAttribute("src", "images/balle.png");  
  
element.classList.add("rouge");
```

push(...), **querySelector(...)**, **setAttribute(...)** et **add(...)** sont des fonctions avec paramètre(s).

Lorsqu'une **fonction** requiert une ou plusieurs **données** dans ses **parenthèses** lorsqu'elle est appelée, on dit que c'est une **fonction avec paramètre(s)**.



❖ Fonctions avec paramètres

- ◆ Si une fonction requiert un ou plusieurs paramètres et qu'on ne lui fournit pas, ça ne fonctionne pas !

✓ `monTableau.push("chat");`

✗ `monTableau.push();`

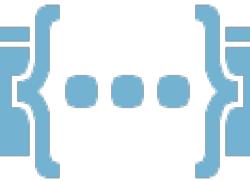
Ceci n'est pas valide. On veut « ajouter une donnée à la fin du tableau », mais on ne dit pas quelle donnée ! La fonction `push()` ne peut pas fonctionner sans savoir ce qu'on veut ajouter dans le tableau.

✓ `element.setAttribute("src", "images/balle.png");`

✗ `element.setAttribute("src");`

Ceci n'est pas valide. Il manque un **paramètre**. Certes, on indique qu'on souhaite changer la valeur de l'attribut `src`, mais si on n'indique pas la **nouvelle valeur** qu'on veut lui donner : ça ne fonctionnera pas !

Fonctions avec paramètres



❖ Déclarer une fonction avec paramètre(s)

- ◆ Dans les **parenthèses**, il faut ajouter un ou plusieurs « **paramètres** »

Déclaration de la fonction

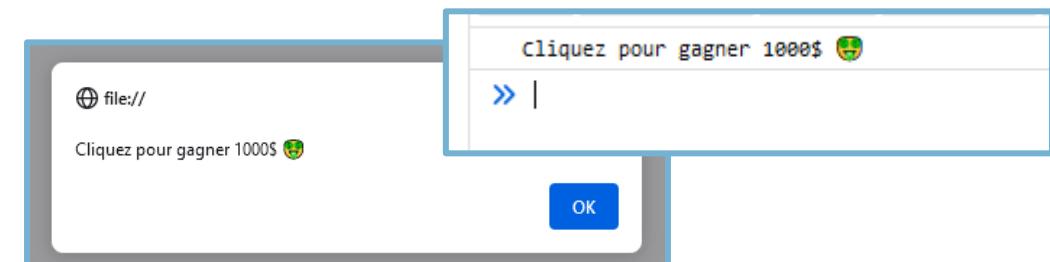
```
function alerteEtConsole(message){  
  alert(message);  
  console.log(message);  
}
```

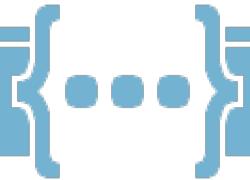
message est le nom du **paramètre** de la fonction. La valeur qui est donnée à **message** lorsque la fonction est **appelée** va déterminer l'alerte et l'affichage dans la console.

Appel de la fonction

```
alerteEtConsole("Cliquez pour gagner 1000$ 😎");
```

Ici, on appelle la fonction **alerteEtConsole(...)** et on donne la valeur "Cliquez pour gagner 1000\$ 😎" au paramètre **message**. Cela signifie que c'est cette phrase qui apparaîtra dans l'**alerte** et dans la **console**.





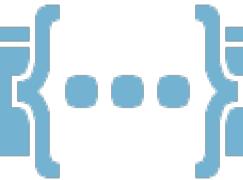
❖ Pourquoi des paramètres ?

- ◆ Exemple : ces trois fonctions permettent de changer la **couleur de texte et de bordure** de l'élément `#texte` :

```
function texteEtBordureRouge(){
    document.querySelector("#texte").style.color = "red";
    document.querySelector("#texte").style.borderColor = "red";
}

function texteEtBordureBleu(){
    document.querySelector("#texte").style.color = "blue";
    document.querySelector("#texte").style.borderColor = "blue";
}

function texteEtBordureVert(){
    document.querySelector("#texte").style.color = "green";
    document.querySelector("#texte").style.borderColor = "green";
}
```



❖ Pourquoi des paramètres ?

- ◆ Même exemple, mais on utilise un **paramètre**  pour choisir la couleur :

```
function texteEtBordure( couleurChoisie ){
  document.querySelector("#texte").style.color = couleurChoisie;
  document.querySelector("#texte").style.borderColor = couleurChoisie;
}
```

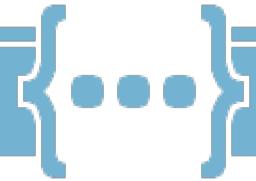
Pas besoin de créer une fonction par couleur !

```
texteEtBordure("pink");
```

Cet appel rendra le texte rose.

```
texteEtBordure("crimson");
```

Cet appel rendra le texte cramoisi.



❖ Pourquoi des paramètres ?

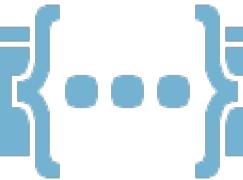
- ◆ Même exemple, mais on utilise **deux paramètres** 😱💥 : un pour choisir la **couleur** et un pour choisir l'**id** de l'élément à modifier !

```
function texteEtBordure( id , couleurChoisie ){  
  
    document.querySelector( id ).style.color = couleurChoisie ;  
    document.querySelector( id ).style.borderColor = couleurChoisie ;  
  
}
```

Pas besoin de créer une fonction par élément et par couleur !

```
texteEtBordure( "#description" , "crimson" );
```

```
texteEtBordure( "#texte" , "pink" );
```

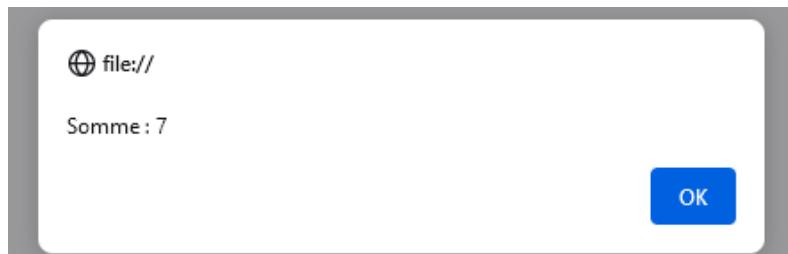


❖ Autres exemples

On peut utiliser des valeurs rangées dans des variables sans problème pour les paramètres. Ici `n1` vaudra 3 et `n2` vaudra 4.

```
let x = 3;  
let y = 4;  
alerterSomme(x, y);
```

```
function alerterSomme(n1, n2){  
  
    let somme = n1 + n2;  
    alert("Somme : " + somme);  
}
```



On peut passer un tableau en paramètre à une fonction. Ici, `alerterSommeTableau(...)` lance une alerte après avoir calculé la somme des nombres du tableau reçu en paramètre.

```
let nombres = [4, 7, 1, 3, 2, 8];  
alerterSommeTableau(nombres);
```

```
function alerterSommeTableau(tab){  
    let total = 0;  
    for(let t of tab){  
        total += t;  
    }  
    alert("Total : " + total);  
}
```



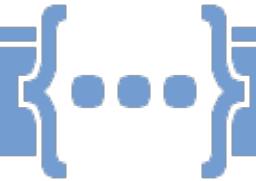


❖ Créer une fonction avec une valeur de retour

- ◆ À la fin de la fonction, on met le mot-clé « **return** » avec la valeur de notre choix.

```
function valeurPi(){  
    let pi = 3.14159265359;  
    return pi;  
}
```

Ici, on voit que la fonction **valeurPi()** va retourner la valeur 3.14159265...



❖ Appeler une fonction avec une valeur de retour

```
function valeurPi(){  
  let pi = 3.14159265359;  
  return pi;  
}
```

valeurPi() retourne la valeur 3.1415...

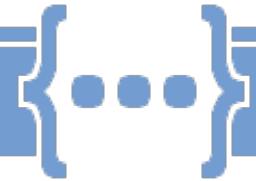
Voici ce qui se passe lorsqu'on appelle valeurPi() :

```
let diametre = 3;  
let perimetreCercle = diametre * valeurPi();
```

Ceci va se transformer en la valeur **retournée** par la fonction valeurPi(), c'est-à-dire 3.1415...

```
let perimetreCercle = 3 * 3.14159265359;
```

Au final, le calcul se servira des valeurs ci-dessus



❖ Point de non-retour !

- ◆ Notez que dès que l'instruction **return** est exécutée, on met fin à la fonction !

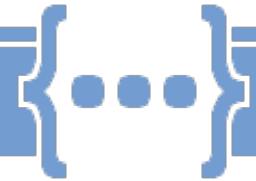
Ceci ne sera jamais exécuté, car
l'instruction **return** est atteinte avant.
gNombre1 continue de valoir 3.

```
let gNombre1 = 3;

function test(){
  let x = 2;
  return x + 1; // On retourne 3
  gNombre1 = 4;
}

gNombre1 + test() // 3 + 3
```





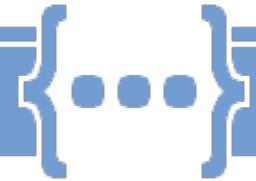
❖ Point de non-retour !

- ◆ S'il y a plusieurs **return**, la fonction est interrompue dès qu'on atteint un de ceux-ci.
 - Dans ce cas précis, il y a un "chat" dans le tableau, alors c'est "Il y a un chat 😍" qui sera retourné. Nous n'allons jamais atteindre l'autre **return**.

```
let gAnimaux = ["chien", "oiseau", "chat", "poisson"];

function chercherChat(){
    for(let a of gAnimaux){
        if(a == "chat"){
            return "Il y a un chat 😍";
        }
    }
    return "Pas de chat 🙁";
}

alert(chercherChat());
```

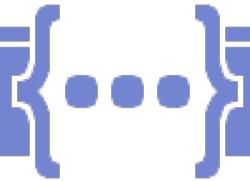


- ❖ On peut retourner n'importe quel type de données !
 - ◆ nombre, chaîne de caractères, booléen, tableau, etc.

```
let x = 3;
let y = 2;
let z = 8;

function getNumbers(){
  let numbers = [x, y, z, 1, 4]; // [3, 2, 8, 1, 4]
  return numbers;
}
```

Ci-dessus, la fonction `getNumbers()` crée un **tableau** (en pigeant des valeurs dans les variables globales et en intégrant également de nouvelles valeurs) et le **retourne** !

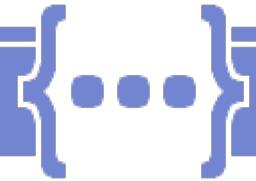


- ❖ En utilisant des **paramètres** ET une **valeur de retour**, on exploite le **plein potentiel** des fonctions ! 

```
function maximum(x, y){  
  if(x > y){  
    return x;  
  }  
  else{  
    return y;  
  }  
  
let nombre = maximum(2, 4); // nombre contient 4
```

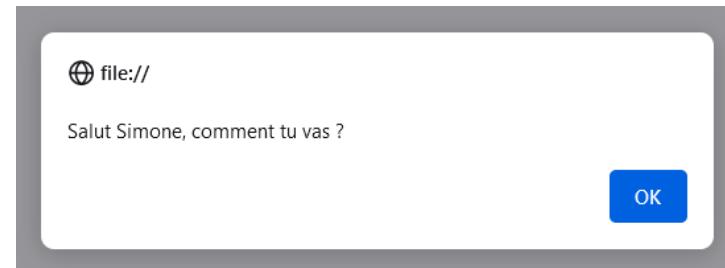
Cette fonction prend **2 paramètres** (deux nombres) et **retourne** la valeur la plus élevée.

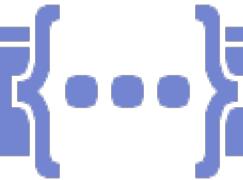
Fonctions avec paramètres et retour



On envoie un nom (chaîne de caractères) en **paramètre** et la **fonction** nous **retourne** un message (chaîne de caractères) qu'on peut utiliser dans une **alerte**, par exemple.

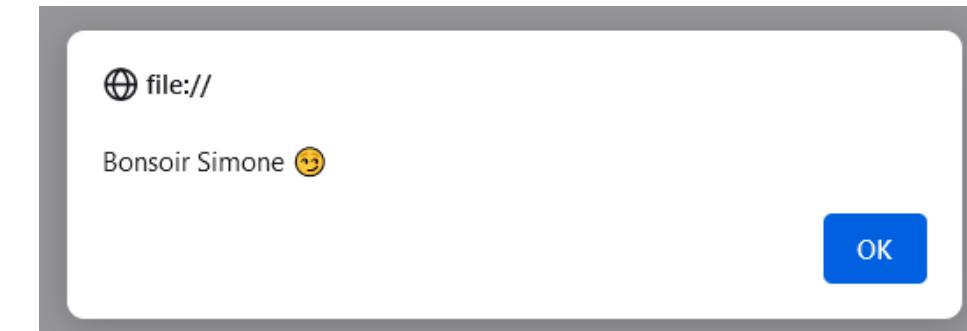
```
function saluer(nom){  
    return "Salut " + nom + ", comment tu vas ?";  
}  
  
alert(saluer("Simone"));
```

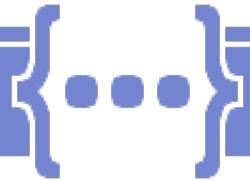




❖ Même fonction, en un peu plus complexe

```
function saluer(moment, nom){  
    if(moment == "jour"){  
        return "Bonjour " + nom + " ☀";  
    }  
    else{  
        return "Bonsoir " + nom + " 😊"  
    }  
}  
  
alert(saluer("soir", "Simone"));
```



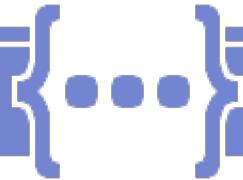


- ❖ On peut prendre un tableau en paramètre 😱

```
function sommeTableau(tab){  
    let total = 0;  
    for(let t of tab){  
        total += t;  
    }  
    return total;  
}  
  
let nombres = [1, 2, 3, 4];  
let somme = sommeTableau(nombres); // Vaut 10
```

Ici, la fonction `sommeTableau()` prend un **tableau de nombres** en paramètre et **retourne** la somme de toutes les valeurs du tableau.

Fonctions avec paramètres et retour



Pour chaque  dans le **tableau** fourni en **paramètre**, le **score** augmente de **3**. Pour chaque  dans le **tableau** fourni en **paramètre**, le **score** diminue de **5**. Cette fonction calcule le **score** et **retourne** le résultat final, qu'on met dans la variable **gScore**.

```
function calculerScore(tableau){  
  let score = 0;  
  for(let t of tableau){  
    if(t == "★"){  
      score += 3;  
    }  
    else if(t == "●"){  
      score -= 5;  
    }  
  }  
  return score;  
}  
  
let gItems = ["★", "★", "●", "★"];  
let gScore = calculerScore(gItems);
```