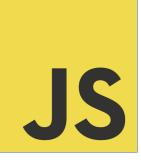


Web Developer Scripts Clients

06 – Les fonctions



Plan du chapitre

- 1.Généralités
- 2.Les arguments
- 3. Portée et variables globales
- 4. Valeur de retour
- 5.ES6

1. Généralités

- Bloc d'instructions pouvant être appelé à n'importe quel endroit dès lors qu'il y est défini
- Avantages :
 - Réutilisable
 - Maintenance plus aisée
 - Robustesse
 - Clarté
 - Meilleure découpe des programmes
- Mot-clé: function

1. Généralités

• Exemple :

```
function bonjour(){
  console.log('Bonjour');
}

// Appel de la fonction bonjour
bonjour();
```



2. Arguments

- Paramètres passés à la fonction afin de lui fournir des informations
- Se placent entre les parenthèses et sont séparés par des virgules
- Réels et formels
 - Formels : ce sont les paramètres qui sont utilisés dans la définition de la fonction
 - Réels : ce sont les valeurs qui sont passées lors de l'appel à la fonction

2. Arguments

• Exemple :

```
function bonjour(prenom){ // paramètre formel
  console.log('Bonjour ' + prenom);
}

// Appel de la fonction bonjour
bonjour('Bombur'); // paramètre réel
```



JS 2. Arguments

• Il est également possible de définir des valeurs par défaut pour les paramètres:

```
function bonjour(prenom='Big', nom='Red'){
  console.log('Bonjour ' + prenom + ' ' + nom);
bonjour();
bonjour('Little');
bonjour('Little', 'Blue');
```



2. Arguments

• Il existe un paramètre appelé <u>paramètre du reste</u>, qui permet de formaliser un nombre inconnu de paramètres sous forme d'un tableau (attention, toujours le placer en dernière position, sinon, une SyntaxError sera levée):

```
function test(p1, ...autres_params){
  console.log(p1);
  console.log(autres_params);
}
test(10);
test(10, 11);
test(10, 11, 12);
test(10, 11, 12, 'treize');
```

- Portée : domaine de validité d'une variable et de la valeur mémorisée
- Variable globale : variable dont la portée est égale à la totalité du programme

• Exemple d'une fonction avec let (idem avec const) :

```
function test(){
  let texte = 'test 1';
  if(true){
   let texte = 'test 2';
    console.log('Dans if (avec let) : ' + texte);
  console.log('Après if (avec let) : ' + texte);
test();
console.log('Hors test (avec let) : ' + texte);
// ReferenceError
```

 Exemple d'un bloc autre qu'une fonction avec let (idem avec const):

```
if(true){
  let texte = 'test 2';
  console.log('Dans if (avec let) : ' + texte);
}
console.log('Après if (avec let) : ' + texte);
// ReferenceError
```

• Exemple d'une fonction avec var :

```
function test(){
 var texte = 'test 1';
  if(true){
   var texte = 'test 2';
    console.log('Dans if (avec var) : ' + texte);
  console.log('Après if (avec var) : ' + texte);
test();
console.log('Hors test (avec var) : ' + texte);
// ReferenceError
```

• Exemple d'un bloc autre qu'une fonction avec var :

```
if(true){
  var texte = 'test 2';
  console.log('Dans if (avec var) : ' + texte);
}
console.log('Après if (avec var) : ' + texte);
// La valeur est affichée sans erreur
```



3. Portée – Hoisting et TDZ

• Le hoisting est le mécanisme de JS qui permet d'utiliser une variable avant sa déclaration avec var (la référence est remontée, pas la valeur) :

```
console.log('var = ' + texte);
// ReferenceError car identifiant texte non-déclaré
```

3. Portée – Hoisting et TDZ

```
console.log('var = ' + texte);
// undefined car l'identifiant est déclaré plus bas
var texte = 'test 2';
```



3. Portée – Hoisting et TDZ

• Le hoisting ne fonctionne pas avec const ou let. Le délai entre l'utilisation et la déclaration est appelée Temporal Dead Zone.

```
// Code exemple 1: avec let
console.log('let = ' + texte); // ReferenceError
let texte = 'test 2';
```

```
// Code exemple 2 : avec const
console.log('const = ' + texte); // ReferenceError
const texte = 'test 2';
```

JS 4. Retour

Valeur retournée par la fonction

• Mot-clé: return

4. Retour

• Exemple :

```
function somme(a, b){
  let result = a + b;

  return result;
}
let param1 = 4;
let param2 = 5;
let resultat = somme(param1, param2);

console.log(param1 + '+' + param2 + '=' + resultat);
```

JS 5. ES6

- ES6 introduit un nouveau type de fonction : les fonctions fléchées
- Elles sont plus concises et souvent utilisées comme fonctions anonymes
- Elles s'écrivent :

```
(param1, [param2,...]) => { // code }
// OU, si un seul paramètre
param1 => { // code }
```

JS 5. ES6

• Exemples :

```
// JS avant ES6 :
let tab = [9, 3, 4, 2, 11, 1, 12];
tab.sort(function(a, b) { return a-b; });
console.log(tab);
// ES6 :
let tab = [9, 3, 4, 2, 11, 1, 12];
tab.sort((a, b) \Rightarrow a-b);
console.log(tab);
```

JS 5. ES6

• Exemples :

```
// ES6 avec retour explicite :
let tab = [9, 3, 4, 2, 11, 1, 12];
tab.sort((a, b) => { return a-b; });
console.log(tab);
```

6. Compléments

- **Déclaration** function f(){}:
 - hoistée (appel possible avant et après)
- Expression const f = () => {}:
 - non hoistée (appel seulement possible après)