Bachelor CDWFS
Applications web dynamiques et interactives
2025-2026

Javascript

Les fonctions

Notions clefs

Appel de fonction

fonctions du langages

prompt()

fonctions globales

Math.round(), Math.floor()

méthodes d'instance

.endsWith()
.indexOf()
.join()
.split()
console.log()

Paramètres

Chaque fonction nécessite un nombre déterminé de paramètres. Chaque paramètre doit être renseigné. L'ordre des paramètres est important.

```
let text = "Please visit Microsoft!";
let newText = text.replace("Microsoft", "Purple");
```

Valeur de retour

FONCTION	TYPE DE RETOUR
Math.floor()	number
.join()	string
.split()	tableau

.startWith()	Booleen
console.log()	« Pas de retour »

Création de fonctions personnalisées

Déclaration de fonction

Sans valeur de retour

Déclaration

```
function bonsoir(){
 console.log("bonsoir");
}
```

Appel

```
bonsoir()
```

Avec valeur de retour

Déclaration

```
function bonsoir(){
 return "bonsoir";
}
```

Appel

```
let message = bonsoir()
console.log(message)
```

Avec passage de paramètre

Déclaration

```
function bonsoir(name){
      return `bonsoir ${name}`;
}
```

Appel

```
let message = bonsoir('Jean')
console.log(message)
```

Avec plusieurs paramètres

Déclaration

```
function bonsoir(name, role){
      return `bonsoir ${name}(${role})`;
}
```

Appel

```
let message1 = bonsoir('Jean', 'Admin')
let message2 = bonsoir('Pierre', 'User')
```

Avec un paramètre par défaut

Déclaration

```
function bonsoir(name, role='admin'){
      return `bonsoir ${name}(${role})`;
}
```

Appel

```
let message = bonsoir('Jean', 'Admin')
console.log(message)
```

Syntaxes alternatives

Expression de fonction

```
const average = function(param1, param2){
      return (param1 + param2)/2
}
```

Fonction fléchée

```
const area = (width, height) => {
      return width * height
}
```

Cas particuliers : 1 seul parametre : pas de parenthèses

```
const square = param => {
      return param**2
}
```

Cas particuliers : 1 seule instruction : pas d'accolades

```
const cube = param => param**3
```

Les fonctions de rappel (« callback »)

4 types de syntaxe

Avec une fonction anonyme

```
const values = [4, 10, 5, 1, 3];
const result = values.filter(function(value){
    return value > 6
});
console.log(result);
```

```
const values = [4, 10, 5, 1, 3];
const result = values.filter(value => value > 6);
console.log(result);
```

Ave une fonction nommée

```
const values = [4, 10, 5, 1, 3];
const result = values.filter(upToSix);
function upToSix(value){
    return value > 6
}
console.log(result);
```

```
const values = [4, 10, 5, 1, 3, 8];
const upToSix = (value) => value > 6
const result = values.filter(upToSix);
console.log(result);
```

Cas d’utilisation

Filtrer les éléments d’un tableau

cf ci-dessus.

Trier (numériquement) un tableau

La méthode .sort() permet de trier alphabétiquement les valeurs d'un tableau. Pour trier numériquement, on utilise une fonction de rappel :

```
var nombres = [4, 10, 5, 1, 3];
nombres.sort(function(a, b) {
    return a - b;
});
console.log(nombres)
```

Répéter des instructions à intervalles régulier

Syntaxe utilisant une fonction nommée :

```
let i= 0
let countdown = setInterval(timer, 1000);
function timer(){
   console.log(i++)
}
```

NB : pour interrompre la répétition, on utilise la fonction **clearInterval**()

Exemple :

```
let countdown = setInterval(timer, 1000);
let i= 3
function timer(){
     if(i==0){
        clearInterval(countdown)
    }
    console.log(i--)
}
```

Implémenter un écouteur d'évènement

```
const target = document.getElementById('img1');
target.addEventListener('click', action);
function action(event) {
    alert("image cliquée");
}
```

Pour aller plus loin

Le « callback hell » : l'enfer du callback

https://fr.javascript.info/callbacks