Bachelor CDWFS
Applications web dynamiques et interactives
2025-2026

Javascript

Objets littéraux

Un objet littéral JavaScript est une structure de données, composé de paires clé-valeur. C'est un moyen simple et efficace de créer et de structurer des données. Ils peuvent servir à stocker l'état d'un composant, ou définir son comportement.

Propriétés et Méthodes

Déclaration

```
const game = {
    player: "alexis",
    start: function(){console.log('go')}
}
```

Syntaxe

Chaque couple est séparé par une **virgule**.

Les **clefs** sont des chaînes de caractères (avec ou sans quote – double ou simple).

Les **valeurs** peuvent être de tous les types Javascript :

- nombre
- chaîne
- booléen
- tableau
- objet
- fonction
- null

Accès aux données

Notation pointée

La notation pointée donne accès aux propriétés et aux méthodes

```
game.start()
```

```
console.log(game.player)
```

Exercice

Créer un objet voiture avec une propriété marque (Bentley), une propriété modele (Continental) et un nombre de porte (2). Afficher ses propriétés

Exercice

Retrouver l'objet littéral utilisé dans ce code. Quelles propriétés et quelles méthodes ?
https://bitbucket.org/Sileax/jeu-de-morpion/src/master/

Notation « crochet »

Tableaux

Pour accéder aux éléments d'un tableau, on utilise la notation « crochet ».

Exemple :

```
const inventaire = {
  fruits : [ "pommes", "poires", "cerises"],
  legumes : [ "tomates", "oignons"]
}
console.log(inventaire.fruits[0]           // Affiche le premier fruit
```

Clefs dynamiques

La notation crochet donne aussi accès aux clés lorsqu'elles sont représentées par des vaiables.

```
  const type = 'start' ;
  const styles = {
    'start' : {'color': '#B2DFDB'},
    'stop': {'color': '#FFCDD2'},
    'pause': {'color': '#FFCC80'},
    'play': {'color': '#B2DFDB'}
  }
 console.log(styles[type].color)           // Affiche #B2DFDB
```

Modifications

Modification de valeurs

```
game.player = "max"
```

On affecte une nouvelle valeur à la propriété player

Ajout de propriétés

```
game.level = 2
```

ou bien, avec le spread opérator :

```
game = {... game, level : 2}
```

Tips

Si la variable level est définie :

```
const level = 2
```

On peut écrire :

```
game = {... game, level}
```

Mot-clef « this »

Le mot-clef « this » fait référence à l'objet lui-même.

```
        let paris = {
            lat:46,
            lon: 50,
            show: function() {return this.lat + ":" + this.lon},
            isNorthern: function(){return true}
        }

        console.log(paris.show())
```

this = objet

Dans l'exemple this.lat équivaut à paris.lat

Tableau d’objets littéraux

Syntaxe

```
const inventory = [
  { name: "apples", quantity: 2 },
  { name: "bananas", quantity: 0 },
  { name: "cherries", quantity: 5 },
];
```

Manipulations fréquentes

```
// Parcourir le tableau
 inventory.forEach((item) => {
     console.log(item.name + " : " + item.quantity)
 })

 // Ajouter un objet
 inventory.push({name: "dates", quantity: 10})

 // Modifier un objet
 inventory[0].quantity = 5

 // Trouver un objet
 let found = inventory.find((item) => item.name === "bananas")

 // Filtrer un objet
 let results = inventory.filter((item) => item.quantity > 2)

 // « Supprimer » un objet
 let newInventory = inventory.filter((item) => item.name !== "apples")

 // Trier
 inventory.sort((a,b) => a.quantity - b.quantity)
```

Pour aller plus loin

Grouper

```
 const data = [
     { name: "John", age: 30, city: "New York" },
     { name: "Jane", age: 25, city: "London" },
     { name: "Mike", age: 32, city: "Chicago" },
```

```
            { name: "Emily", age: 28, city: "New York" },
            { name: "David", age: 35, city: "London" }
        ];
        // Grouper par ville
        const groupByCity = data.reduce((acc, person) => {
            if (!acc[person.city]) {
                acc[person.city] = [];
            }
            acc[person.city].push(person);
            return acc;
        }, {});
        console.log(groupByCity)
```

Destructuration d’objets

Exemple 1 et 2

```
        let student = {
            id:20,
            name:'Alex',
            age:20
        }

        // Exemple 1 : déclaration de variables
        let {name,age} = student
        console.log(name)
        console.log(age)

        // Exemple 2 : déclaration de paramètres
        function display({name,age}){
            console.log(`L'étudiant ${name} a ${age} ans`)
        }
        display(student)
```

Exemple 3

```
const inventory = [
  { name: "apples", quantity: 2 },
  { name: "bananas", quantity: 0 },
  { name: "cherries", quantity: 5 },
];
// Sans destructuration
const result = inventory.find((fruit) => {
  return fruit.name === "cherries";
```

```
});
// Avec Destructuration
const result = inventory.find(({ name }) => name === "cherries");
```