Les objets littéraux en JavaScript

> Les **objets littéraux** sont l'une des façons les **plus simples** et **courantes** de créer des objets.
> Ils permettent de **regrouper des données** et des **fonctionnalités** sous une forme **structurée et lisible**.

1. Comprendre les objets littéraux

Un **objet littéral** est une collection de **paires clé-valeur**.
Les **clés**, appelées aussi **propriétés**, sont des chaînes ou des symboles, et les **valeurs** peuvent être **de tout type** : nombres, chaînes, fonctions, ou même d'autres objets.

Exemple :

```javascript
const personne = {
  nom: "Alice",
  age: 25,
  profession: "Développeuse"
};
console.log(personne.nom); // "Alice"
console.log(personne["age"]); // 25
```

2. Syntaxe des objets littéraux

> La syntaxe d'un objet littéral utilise des accolades `{}` pour **entourer** les paires clé-valeur **séparées par des virgules**.

Exemple :

```javascript
const voiture = {
  marque: "Tesla",
  modele: "Model S",
  electrique: true
};
```

- **Clés :** Identifiants ou chaînes (par défaut, les clés sont converties en chaînes).
- **Valeurs :** Les données associées à chaque clé.

3. Les objets imbriqués

> Un objet littéral peut contenir d'autres objets comme valeurs.
> Cela permet de structurer des données hiérarchiques.

Exemple :

```javascript
const utilisateur = {
  nom: "Marie",
  contact: {
    email: "marie@example.com",
    telephone: "0123456789"
  }
};
console.log(utilisateur.contact.email); // "marie@example.com"
```

4. Manipuler les objets littéraux

4.1. Accéder aux propriétés

> Utilisez la notation pointée `.` ou les crochets `[]`.
>
> On préfèrera la notation pointée pour les clés valides et la notation entre crochets pour les clés dynamiques ou contenant des espaces.

Exemple :

```javascript
console.log(utilisateur.nom); // "Marie"
console.log(utilisateur["contact"]["telephone"]); // "0123456789"

// Clé contenant un espace (non valide pour la notation pointée et non recommandée)
const objetComplexe = {
  "clé avec espace": "valeur"
};

console.log(objetComplexe["clé avec espace"]); // "valeur"
```

Une clé dynamique est une clé qui peut changer en fonction de la logique de votre programme :

```javascript
const cleDynamique = "age";
const utilisateur = {
  nom: "Marie",
  age: 30
};
console.log(utilisateur[cleDynamique]); // 30
```

4.2. Ajouter des propriétés

> On peut ajouter des propriétés d'un objet après sa création.

Exemple :

```javascript
const animal = {
  espece: "Chien"
};
console.log(animal); // { espece: "Chien" }

animal.nom = "Rex"; // Ajout d'une nouvelle propriété
console.log(animal); // { espece: "Chien", nom: "Rex" }
```

4.3. Modifier des propriétés

> Il est possible de modifier la valeur d'une propriété existante.

Exemple :

```javascript
const livre = {
  titre: "JavaScript pour les nuls",
  auteur: "John Doe"
};

livre.auteur = "Jane Doe"; // Modification de la valeur de la propriété auteur
console.log(livre); // { titre: "JavaScript pour les nuls", auteur: "Jane Doe" }
```

4.4. Supprimer des propriétés

> La commande `delete` permet de supprimer une propriété d'un objet.

Exemple :

```javascript
delete utilisateur.contact.telephone;
console.log(utilisateur.contact); // { email: "marie@example.com" }
```

4.5. Parcourir les propriétés

> La boucle `for...in` permet de parcourir les clés d'un objet.

Exemple :

```javascript
const cellphone = {
  marque: "Apple",
  modele: "iPhone 13",
  prix: 999
};

for (const propriete in cellphone) {
  console.log(`${propriete}: ${cellphone[propriete]}`);
}

// Affiche :
// marque: Apple
// modele: iPhone 13
// prix: 999
```

4.6. Copier un objet

Il est important de noter que la copie d'un objet avec `=` ne crée pas une nouvelle instance, mais une référence à l'objet original.

Pour créer une copie superficielle, utilisez l'opérateur spread `...`.

Cela permet de cloner un objet sans affecter l'original.
Cela signifie que les modifications apportées à la copie n'affecteront pas l'objet d'origine.

Note : Pour une copie profonde, il est recommandé d'utiliser des méthodes comme `JSON.parse(JSON.stringify(obj))` ou des bibliothèques comme Lodash.

On appelle copie superficielle (shallow copy) une copie qui ne copie que les propriétés de premier niveau tandis qu'une copie profonde (deep copy) copie également les propriétés imbriquées.
En d'autres termes, une copie superficielle ne copie pas les objets imbriqués, tandis qu'une copie profonde le fait.

Exemple :

```javascript
const obj1 = {
  nom: "Test",
  age: 25,
  info: {
    tel: "123",
    mail: "abc@email.com"
  }
};

// Copie par référence
const copie = obj1;

copie.nom = "Demo"; // Modifie le nom de la référence
console.log(obj1.nom); // "Demo" (affecté)
console.log(copie.nom); // "Demo" (affecté)

copie.info.tel = "1234"; // Modifie le tel de la référence
console.log(obj1.info.tel); // "1234" (affecté)
console.log(copie.info.tel); // "1234" (affecté)

// Copie superficielle
const obj2 = { ...obj1 };

obj2.nom = "Demo"; // Modifie le nom de la copie
console.log(obj1.nom); // "Test" (pas affecté)

obj2.info.tel = "456"; // Modifie le tel de la copie
console.log(obj1.info.tel); // "456" (affecté)

// Pour une copie profonde
const obj3 = JSON.parse(JSON.stringify(obj1)); // Copie profonde

obj3.info.tel = "789"; // Modifie le tel de la copie profonde
console.log(obj1.info.tel); // "456" (pas affecté)
console.log(obj3.info.tel); // "789" (affecté)
```

5. À RETENIR

- Un objet littéral est défini avec des `{}` et contient des paires clé-valeur.
- Les propriétés peuvent être lues, ajoutées, modifiées ou supprimées dynamiquement.
- Les objets imbriqués permettent de structurer des données complexes.
- Utilisez des outils comme `for...in`, `delete`, ou le `spread operator` (opérateur de propagation) `...` pour manipuler vos objets efficacement.
- La copie superficielle et la copie profonde sont des concepts importants à comprendre lors de la manipulation d'objets.