



Робота з об'єктами

www.t.me/shvets_js

Створення об'єкта з початковими значеннями за допомогою літеральної нотації

 `let obj = { key1: value1, key2: value2, ... };`

Створює новий об'єкт `obj` з заданими властивостями та їхніми значеннями за допомогою літеральної нотації.

Приклад:

```
let obj = { name: 'John', age: 30 };
```



Робота з об'єктами

www.t.me/shvets_js

Створення об'єкта з використанням конструктора

🔨 `let obj = new Object();`

Створює новий порожній об'єкт `obj` за допомогою конструктора `Object`.

Приклад:

```
let obj = new Object();
```



Робота з об'єктами

www.t.me/shvets_js

Отримання ключів об'єкта

🔑 `Object.keys(obj)`

Повертає масив з ключами (властивостями) об'єкта `obj`.

Приклад:

```
let obj = { name: 'John', age: 30 };
```

```
console.log(Object.keys(obj)); // Виведе ['name', 'age']
```



Робота з об'єктами

www.t.me/shvets_js

Отримання значень об'єкта



`Object.values(obj)`

Повертає масив зі значеннями властивостей об'єкта `obj`.

Приклад:

```
let obj = { name: 'John', age: 30 };  
console.log(Object.values(obj)); // Виведе ['John', 30]
```



Робота з об'єктами

www.t.me/shvets_js

Отримання пар ключ-значення об'єкта

🔗 `Object.entries(obj)`

Повертає масив масивів з парами ключ-значення об'єкта `obj`.

Приклад:

```
let obj = { name: 'John', age: 30 };  
console.log(Object.entries(obj)); // Виведе [['name',  
'John'], ['age', 30]]
```



Робота з об'єктами

www.t.me/shvets_js

Повне копіювання об'єкта за допомогою `Object.assign()`



```
let newObj = Object.assign({}, obj);
```

Створює повну копію об'єкта `obj` у новому об'єкті `newObj`. При цьому копіюється лише перший рівень властивостей об'єкта. Якщо об'єкт містить вкладені об'єкти чи масиви, вони будуть посиланнями на оригінал.

Приклад:

```
let obj = { name: 'John', age: 30 };  
let newObj = Object.assign({}, obj);
```




Робота з об'єктами

www.t.me/shvets_js

Копіювання об'єкта за допомогою `Object.create()`

🔑 `let newObj = Object.create(obj);`

Створює неповну копію об'єкта `obj` у новому об'єкті `newObj` за допомогою метода `Object.create()`, який має ті ж властивості, що і оригінальний об'єкт (`obj`), але буде мати `obj` як прототип. Зміни в `newObj` можуть впливати на `obj`, оскільки вони використовують спільний прототип.

Приклад:

```
let obj = { name: 'John', age: 30 };  
let newObj = Object.create(obj);
```



Робота з об'єктами

www.t.me/shvets_js

Глибоке копіювання об'єкта за допомогою JSON

🔍 `let newObj = JSON.parse(JSON.stringify(obj));`

Створює глибоку копію об'єкта `obj` у новому об'єкті `newObj` за допомогою серіалізації JSON.

Приклад:

```
let obj = { name: 'John', age: 30, address: { city: 'New York' }
};
```

```
let newObj = JSON.parse(JSON.stringify(obj));
```




Робота з об'єктами

www.t.me/shvets_js

Повне копіювання об'єкта за допомогою `_.cloneDeep()`



```
let newObj = _.cloneDeep(obj);
```

Створює повну глибоку копію об'єкта `obj` у новому об'єкті `newObj`. Всі вкладені об'єкти та масиви також глибоко копіюються, забезпечуючи незалежність від оригінального об'єкта.

Приклад:

```
let _ = require('lodash');
```

```
let obj = { name: 'John', address: { city: 'New York' } };
```

```
let newObj = _.cloneDeep(obj);
```



Робота з об'єктами

www.t.me/shvets_js

Створення об'єкта Map



```
let map = new Map();
```

Створює новий порожній об'єкт Map, який представляє собою колекцію ключ-значення, де ключі можуть бути будь-якого типу.

Приклад додавання ключ-значення до Map:

```
map.set('key1', 'value1');
```


```
map.set(2, 'value2');
```



Робота з об'єктами

www.t.me/shvets_js

Створення об'єкта Set

 `let set = new Set();`

Створює новий порожній об'єкт Set, який представляє собою колекцію унікальних значень, де кожен елемент може зустрічатися тільки один раз.

Приклад додавання значень до Set:

```
set.add('value1');
```

```
set.add(2);
```