ОСНОВИ РОБОТИ З МАСИВАМИ ТА ОБ'ЄКТАМИ

План:

1. Масиви: правила створення, елементи, ключи.
2. Функції для масивів.

Масиви: правила створення, елементи, ключи.

У програмуванні часто потрібно зберігати список різних значень, наприклад, всі дні тижня або всі місяці. Замість того, щоб створювати окрему змінну для кожного значення, можна використовувати масив – спеціальний тип даних, який зберігає список значень.

Щоб створити масив, потрібно використати квадратні дужки [ ].

Наприклад, так можна створити масив з днями тижня:

```
var arr = ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];
```

У масив можна зберігати різні значення, які поділяються між собою комою. Якщо значення є рядком, то його потрібно брати у лапки, а якщо числом – не потрібно. Наприклад, так можна створити масив з різними значеннями:

```
//У масиві можна зберігати як рядки, так і числа:
var arr = ['пн', 256, 'ср', 34, 38, 'сб', 95];
```

Для того, щоб вивести на екран конкретне значення з масиву, потрібно використати порядковий номер елемента масиву в квадратних дужках. При цьому нумерація починається з нуля.

Наприклад, так можна вивести третє значення з масиву з днями тижня:

Приклад:

```
//Виведемо слово 'ср':
var arr = ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];
alert(arr[2]);
```

У попередньому прикладі ми говорили про масив, який містить кілька елементів. Щоб звернутися до конкретного елементу в масиві, ми вказували його номер у квадратних дужках. Цей номер називається ключем масиву.

У JavaScript ключі для елементів масиву можуть бути вказані явно за допомогою об'єктів. Об'єкти в інших мовах програмування називаються асоціативними масивами. Об'єкт створюється за допомогою фігурних дужок {}, всередині яких ми можемо вказати ключ та його значення у форматі ключ: значення.

Наприклад, щоб понеділок мав ключ 1, а не нуль, як було раніше (і всім іншим дням додаємо одиницю):

```
// Вкажемо ключі в явному вигляді:
var obj = {1: 'пн', 2: 'вт', 3: 'ср', 4: 'чт', 5: 'пт', 6: 'сб', 7: 'вс'};
alert(obj[1]); //виведе 'пн'
```

Синтаксис тут такий: ключ, потім йде двокрапка :, а потім значення.

Ключі не обов'язково повинні бути числами, вони можуть бути і рядками.

Приклад. Зробити масив, в якому ключами будуть імена працівників, а елементами-їх зарплати

```
//Масив (об’єкт) робітників:
var obj = {'Коля': 200, 'Вася': 300, 'Петя': 400};
```

Дізнаємося зарплату Васі:

```
var obj = {'Коля': 200, 'Вася': 300, 'Петя': 400};
alert(obj['Вася']); //виведе 300
```

Крім того, лапки навколо строкових ключів можна і не ставити:

```
var obj = {key1: 200, key2: 300, key2: 400};
alert(obj['key1']); //виведе 200
```

Однак, на такі ключі накладаються обмеження: вони не повинні починатися з цифри, не повинні мати дефіс або символ пробілу всередині.

Якщо є такий ключ–його беруть в лапки.

Властивості об’єкта

Є ще один спосіб отримати доступ до елементів об'єкта – використовуючи точку, а не квадратні дужки: не obj['ключ'], а obj.ключ. Це називається звернення до властивості об'єкта. Отже, якщо ми маємо об'єкт з ключами та значеннями, ми можемо отримати значення, вказавши назву ключа після крапки. Наприклад, у наступному коді ми створюємо об'єкт з трьома ключами та їхніми значеннями:

Приклад:

```
var obj = {key1: 200, key2: 300, key2: 400};
alert(obj.key1); //виведе 200
```

Однак, є обмеження на назви ключів. Назви ключів не можуть починатися з цифри, мати дефіс або символ пробілу всередині. Якщо є такий ключ – потрібно використовувати звернення через квадратні дужки. Також замість фігурних дужок {} та квадратних дужок [] можна використовувати функції-конструктори Object() та Array() відповідно (це просто альтернативний синтаксис).

Приклад. Створення об'єкта другим синтаксисом

```
var obj = newObject(key1: 200, key2: 300, key2: 400);
```

```
alert(obj['key1']); //виведе 200
```

Створимо масив другим синтаксисом:

```
var arr = Array('пн', 256, 'ср', 34, 38, 'сб', 95);
```

Створення об'єкта через { } і через Object() еквівалентно.

В масивах ж різниця між ними проявляється в разі, якщо створюється масив, що складається з одного елементу, який буде цілим числом, ось так:

```
var arr = Array(10);
```

У цьому випадку отримаємо не такий масив:

```
var arr = [10];
```

А ось такий:

```
var arr = [,,,,,,,,,];
```

Це буде масив, що складається з 10-ти порожніх елементів (їх значення undefined), а не масив з одного елемента 10.

Заповнення масиву

Масив - це колекція даних, яку можна заповнювати після його створення. Наприклад, ми можемо створити порожній масив і потім додавати елементи до нього пізніше. Ось приклад:

```
var arr = [ ]
arr[0] = 1;
arr[1] = 2;
arr[2] = 3;
alert(arr) //за допомогою alert виводимо вміст масиву
```

Ми також можемо створити масив з об'єктами. В цьому випадку ми можемо додавати нові елементи до масиву, використовуючи назви властивостей як ключі. Ось приклад:

```
var obj = {};
obj['Коля'] = 100;
obj['Вася'] = 200;
```

```
obj['Петя'] = 300;
```

Багатовимірний масив

Масиви можуть також містити інші масиви, створюючи таким чином багатовимірний масив. Ось приклад, де ми створюємо багатовимірний масив студентів, який містить два підмасиви: студентів чоловічої статі і жіночої.

```
//Багатовимірний масив студентів:
var students = {
  'boys': ['Коля', 'Вася', 'Петя'],
  'girls': ['Даша', 'Маша', 'Лена'],
};
```

Щоб отримати доступ до елементу багатовимірного масиву, потрібно використовувати дві пари квадратних дужок. Ось приклад, де ми виводимо елемент 'Коля' з багатовимірного масиву:

```
students['boys'][0]; //виведе 'Коля'.
```

Функції для масивів

Метод concat() дозволяє об'єднати декілька масивів у один новий масив. Щоб злити масиви, потрібно викликати метод concat() на початковому масиві і передати в якості аргументів інші масиви, які потрібно додати до першого. Метод повертає новий масив, не змінюючи початковий.

Наприклад, є три масиви, які необхідно злити разом:

Приклад. 3 масиву зіллються в один:

```
var arr1 = ['Один', 'Два'];
var arr2 = ['Три', 'Чотири'];
var arr3 = ['П’ять', 'Шість'];
result = arr1.concat(arr2, arr3);
```

```
console.log(result);
```

Результат виконання коду: ['Один', 'Два', 'Три', 'Чотири', 'П’ять', 'Шість']

Приклад. Метод concat() можна також використовувати для додавання окремих значень до масиву:

```
var arr1 = ['Один', 'Два'];
var arr2 = ['Три', 'Чотири'];
result = arr1.concat(arr2, 'П’ять', 6, 7);
console.log(result);
```

Результат виконання коду: ['Один', 'Два', 'Три', 'Чотири', 'П’ять', 6, 7]

Метод reverse змінює порядок елементів в масиві на зворотний.

Метод змінює вихідний масив (він стане перевернутим) і повертає також перевернутий масив.

Синтаксис. масив.reverse();

Пример. Перевернем массив:

```
var arr = ['a', 'b', 'c'];
arr.reverse();
console.log(arr);
```

Результат виконання коду: ['c', 'b', 'a']

Приклад. В змінну newArr запишеться перевернутий масив:

```
var arr = ['a', 'b', 'c'];
var newArr = arr.reverse();
console.log(newArr);
```

Результат виконання коду: ['c', 'b', 'a']

Метод push додає необмежену кількість елементів в кінець масиву. Елементи передаються параметром методу.

Метод змінює вихідний масив. Повертає нову довжину масиву.

Синтаксис: масив.push(елемент, елемент, елемент...);

Приклад. У вихідний масив додано 2 нових елементи і виведено вміст нового масиву:

```
var arr = ['a', 'b', 'c'];
arr.push('d', 'e');
console.log(arr);
```

Результат виконання коду: ['a', 'b', 'c', 'd', 'e']

Приклад. У вихідний масив додано 2 нових елементи і виведена нова довжина масиву:

```
var arr = ['a', 'b', 'c'];
var length = arr.push('d', 'e');
document.write(length);
```

Результат виконання коду: 5

Метод unshift додає будь-яку кількість нових елементів в початок масиву.

Метод змінює вихідний масив. Повертає нову довжину масиву.

Синтаксис. масив.unshift(елемент, елемент,е...);

Приклад. У початок вихідного масиву було додано ще 2 нових елементи і виведений вже новий, змінений масив:

```
var arr = ['a', 'b', 'c'];
arr.unshift('d', 'e');
console.log(arr);
```

Результат виконання коду: ['d', 'e', 'a', 'b', 'c']

Приклад. У початок вихідного масиву було додано ще 2 нових елементи і виведена нова довжина масиву:

```
var arr = ['a', 'b', 'c'];
var length = arr.unshift('d', 'e');
document.write(length);
```

Результат виконання коду: 5

Метод shift видаляє перший елемент масиву.

Метод змінює вихідний масив. Повертає віддалений елемент.

Синтаксис: массив.shift();

Приклад. Видаляється перший елемент з масиву:

```
var arr = ['a', 'b', 'c', 'd', 'e'];
arr.shift();
console.log(arr);
```

Результат виконання коду: ['b', 'c', 'd', 'e']

Приклад. Видаляється перший елемент з масиву і виводиться його на екран:

```
var arr = ['a', 'b', 'c', 'd', 'e'];
document.write(arr.shift());
```

Результат виконання коду: a

Метод pop видаляє останній елемент масиву.

Метод змінює вихідний масив. Повертає віддалений елемент.

Синтаксис: масив.pop();

Приклад. З масиву arr видалено останній елемент:

```
var arr = ['a', 'b', 'c', 'd', 'e'];
arr.pop();
console.log(arr);
```

Результат виконання коду : ['a', 'b', 'c', 'd']

Приклад. Виведений останній елемент, який був видалений з вихідного масиву:

```
var arr = ['a', 'b', 'c', 'd', 'e'];
var elem = arr.pop();
document.write(elem);
```

Результат виконання коду : e

Метод slice() дозволяє вирізати (витягати) частину масиву і повернути її як новий масив. Ви можете вказати номер першого і останнього елемента, які потрібно вирізати. Якщо другий параметр не вказано, вирізані

елементи візьмуться від першого до кінця масиву. Ви також можете вказати від'ємні значення для другого параметра, щоб відлік елементів починався з кінця масиву.

Метод не змінює початковий масив.

Синтаксис. масив.slice(звідки відрізати, [докуда відрізати]);

Приклад. Вирізається з масиву елементи з нульового по другій не включно (другий не виріжеться):

```
var arr = ['a', 'b', 'c', 'd', 'e'];
console.log(arr.slice(0, 2));
```

Результат виконання коду : ['a', 'b']

Приклад. Вирізається з першого елемента до кінця масиву. Для цього другий параметр не задається:

```
var arr = ['a', 'b', 'c', 'd', 'e'];
console.log(arr.slice(1));
```

Результат виконання коду : ['b', 'c', 'd', 'e']

Приклад. Вирізаються елементи з другого по передостанній (-1 вказує на останній елемент і він не включиться в витягнуту частина):

```
var arr = ['a', 'b', 'c', 'd', 'e'];
console.log(arr.slice(1, -1));
```

Результат виконання коду : ['b', 'c', 'd']

Перевага такого підходу в тому, що вирізатися завжди буде частина масиву, не включаючи останній елемент, незалежно від розміру масиву.

Метод splice дозволяє додавати або видаляти елементи з масиву. Він може лише видаляти елементи, лише додавати їх або робити і те й інше одночасно. Це досить складний метод для розуміння, але дуже універсальний.

Метод змінює сам масив і повертає масив віддалених елементів. Перший параметр вказує номер елемента масиву, який потрібно видалити, а другий параметр вказує, скільки елементів потрібно видалити. Якщо другий параметр встановлено на 0, то елементи не будуть видалені, але будуть додані нові елементи.

Якщо другий параметр встановлено на 0, елементи будуть вставлені в масив, починаючи з позиції, яку вказано в першому параметрі методу. Перший параметр може мати негативне значення, щоб відлік позиції починався не з початку масиву, а з кінця. Останній елемент має номер '-1', передостанній - '-2' і так далі.

Метод повертає масив віддалених елементів.

Синтаксис. масив.splice(звідки видаляємо, скільки елементів видаляємо, [вставити елемент], [вставити елемент]...);

Приклад. З масиву будуть видалені елементи з другого, 3 штуки (це 'c', 'd' і 'e'):

```
var arr = ['a', 'b', 'c', 'd', 'e', 'f'];
arr.splice(2, 3);
console.log(arr);
```

Результат виконання коду : ['a', 'b', 'f']

Приклад. З масиву будуть видалені з другого, 3 штуки (це 'c', 'd' і 'e') і записані в змінну del:

```
var arr = ['a', 'b', 'c', 'd', 'e', 'f'];
var del = arr.splice(2, 3);
console.log(del);
```

Результат виконання коду : ['c', 'd', 'e']

Приклад. Спочатку буде видалено 3 елемент, а потім замість нього вставлено ще 2 нових елемента ('1' і '2'):

```
var arr = ['a', 'b', 'c', 'd'];
arr.splice(2, 1, '1', '2');
console.log(arr);
```

Результат виконання коду : ['a', 'b', '1', '2', 'd']

Приклад. Після 3-го елемента будуть додані ще 2 елемента (так як другим параметром вказано 0, то видалення не буде):

```
var arr = ['a', 'b', 'c', 'd'];
arr.splice(2, 0, '1', '2');
console.log(arr);
```

Результат виконання коду : ['a', 'b', 'c', '1', '2', 'd']

Приклад. З масиву буде видалений передостанній елемент ' e ' (його номер -2):

```
var arr = ['a', 'b', 'c', 'd', 'e', 'f'];
arr.splice(-2, 1);
console.log(arr);
```

Результат виконання коду : ['a', 'b', 'c', 'd', 'f']

Метод sort виробляє сортування масиву в лексикографічному порядку.

Також можна вказати параметром власну функцію для сортування, але це необов'язково.

Синтаксис: масив.sort([функція]);

Приклад. У масиві була проведена сортування за алфавітом і виведений відсортований масив:

```
var arr = ["Банан", "Апельсин", "Яблоко", "Манго"];
console.log(arr.sort());
```

Результат виконання коду : [Апельсин, Банан, Манго, Яблоко]

Приклад. Проведено сортування відповідно до лексикографічного порядку:

```
var arr = [43,-3, 10, 100];
console.log(arr.sort());
```

Результат виконання коду : [-3, 10, 100, 43]

Функція Object.keys дозволяє витягти Ключі об'єкта у вигляді масиву.

Синтаксис: Object.keys(объект);

Приклад. Отримано Ключі об'єкта у вигляді масиву:

```
var obj = {a: 1, b: 2, c: 3};
console.log(Object.keys(obj));
```

Результат виконання коду : ['a', 'b', 'c']