Lesson 7

План заняття

- Основи роботи з масивами;
- Варіанти створення масивів;
- Методи для роботи з масивами;
- Методи перебору масиву;
- Видалення елементів з масивів;

Array

Для зберігання впорядкованих колекцій існує спеціальна структура даних, яка називається масив, Array. Масив (Array) у JavaScript є глобальним об'єктом, який використовується для створення масивів; які є високорівневими списконодібними об'єктами. У масиві можуть зберігатися елементи будь-якого типу.

Массив є об'єктом і, слідуючи, веде себе як об'єкт.

Масив є прояв структури даних stack. Елементи масиву зберігаються безперервно у пам’яті. Звернання до елементу масиву можно оцінити як O(1)

Властивості:

1. Пам’ять виділяється статично. Відбувається послідовне заповнення пам’яті та наповнення масиву.
2. Отримання елементу відбувається за індексом, це дуже швидке рішення для машини.
3. Добавання елементу відбувається повільно. Це тому що пам’ять виділяєьтся статично.
4. Структура масиву може бути однонаправленим або мати декілька напрямків.

Array

Масив з довжиною у 5. елементів

Індекси елементів масиву

Чого не треба робити

1. Додавати нечислову властивість. Наприклад arr.example = 5;
2. Створювати "порожнини". Наприклад array[0] = 1 => array[100] = 100;
3. Заповнювати масив у зворотному порядку. Це призведе то втрати пам’яті та зайвим операціям під капотом.
4. Не порівнюйте масиви за допомогою == ;

```
0 == [] // true
'0' == [] // false
```

Правила порівнянь:

- Два об'єкти рівні один одному == тільки в тому випадку, якщо вони посилаються на той самий об'єкт.
- Якщо один із аргументів == є об'єктом, а інший – примітивом, то об'єкт перетворюється на примітив, як описано в розділі Перетворення об'єктів у примітиви.
- …За винятком null і undefined, які рівні == один одному і нічого більше.

Висяча кома

Список елементів масиву, як і список властивостей об'єкта, може закінчуватися комою:

```
let arr = [
  "el",
  "el2",
  "el3",
];
```

«Висяча кома» спрощує процес додавання/видалення елементів, оскільки всі рядки стають ідентичними.

Array

Синтаксис

let arr = new Array([value]);

let arr = [];

Отримання елементу за індексом

arr[index]

Отримання довжини масиву. Початкове значення якого дорівнює +0, а атрибути: { [[Writable]]: true, [[Enumerable]]: false, [[Configurable]]: false }.

arr.length

Array

Отримання останнього елементу:

arr[arr.length - 1] або arr.at(-1)

arr[-1] - не працює, це помилка. Воно поверне undefined.

Array and for

Перебор масиву

```
for ( let i = 0; i < arr.length; i++) {
    console.log(arr[i]);
}

for ( let element of arr) {
    console.log(element);
}
```

Array методи

- .shift() - метод видаляє елемент із нульовим індексом і зміщує значення з послідовними індексами вниз, а потім повертає вилучене значення. Якщо властивість length дорівнює 0, повертається undefined.
- .unshift(element1, element2, /* …, */ elementN) - метод вставляє задані значення на початок об’єкта, схожого на масив.
- .push(element1, ..., elementN) - додає один або більше елементів до кінця масиву і повертає нову довжину масиву.
- .pop() - видаляє останній елемент з масиву та повертає його значення. Якщо масив порожній то поверне undefinned.
- .splice(start, deleteCount, item1, item2, /* …, */ itemN) - метод змінює array починаючи з позиції start: видаляє deleteCount елементів і вставляє elem1, ..., elemN на їх місце. Повертається масив з видалених елементів. Можно вставляти елементи без будь-яких видалень. Для цього нам потрібно встановити значення 0 для deleteCount. Тут і в інших методах масиву допускаються від’ємні індекси. Вони дозволяють почати відлік елементів з кінця. Пам’ятайте що цей метод метує вхідний масив!
- .slice([begin[, end]]) - повертає новий масив, копіюючи до нього всі елементи від індексу start до end (не включаючи end). І start, і end можуть бути від’ємними. В такому випадку відлік буде здійснюватися з кінця масиву.

Array методи

- .concat(value1, value2, /* ..., */ valueN) - створює новий масив, в який копіює дані з інших масивів та додаткові значення;
- .forEach(callbackFn, thisArg) - дозволяє запускати функцію для кожного елемента масиву. Його значення, що повертається, відкидається => return = udefined

  Callback => (element, index, array) => { ... }
- .indexOf(item[, from]) - шукає item, починаючи з індексу from, і повертає індекс, на якому був знайдений шуканий елемент, в іншому випадку -1. використовує суворе порівняння === ;
- .includes(item[, from]) - шукає item, починаючи з індексу from, і повертає true, якщо пошук успішний;
- .find(callbackFn, thisArg) - функція викликається по черзі для кожного елемента масиву. Якщо функція повертає true, пошук припиняється, повертається item. Якщо нічого не знайдено, повертається undefined;
- .findIndex/findLastIndex(callbackFn, thisArg) - по суті, те ж саме, але повертає індекс, на якому був знайдений елемент, а не сам елемент, і -1, якщо нічого не знайдено;

Array методи

- .sort([callbackFn]) - на місці сортує елементи масиву та повертає відсортований масив. Сортування не обов'язково стійке. Порядок сортування за замовчуванням відповідає порядку кодових точок Unicode. Пам’ятайте що вона амутує вхідний масив.

callbackFn(a, b) необов'язковий параметр. Вказує функцію, яка визначає порядок сортування. Якщо опущений, масив сортується відповідно до значень кодових точок кожного символу Unicode, отриманих шляхом перетворення кожного елемента в рядок.

Якщо функцію порівняння compareFunction надано, елементи масиву сортуються відповідно до її значення, що повертається. Якщо порівнюються два елементи a і b, то:

- Якщо compareFunction(a, b) менше 0, сортування поставить a меншим індексом, ніж b, тобто, a йде першим.
- Якщо compareFunction(a, b) поверне 0, сортування залишить a та b незмінними по відношенню один до одного, але відсортує їх по відношенню до всіх інших елементів. Зверніть увагу: стандарт ECMAscript не гарантує цю поведінку, і її слідують не всі браузери (наприклад, версії Mozilla принаймні, до 2003 року).
- Функция compareFunction(a, b) должна всегда возвращать одинаковое значение для определённой пары элементов a и b. Если будут возвращаться непоследовательные результаты, порядок сортировки будет не определён.

Array методи

- .map(function callback(currentValue[, index[, array]]) {

  // повернє новий елемент у новий масив

}[, thisArg]) - викликає функцію для шкірного елемента масиву і повертає масив результатів виконання цієї функції.

- .reverse() - змінює порядок елементів в arr на зворотний. Мутує вхідний масив.
- .split([separator[, limit]]) - він розбиває рядок на масив по заданому роздільнику separator. Limit - обмеження на кількість елементів в масиві;
- .join(separator) - він створює рядок з елементів масиву, вставляючи separator між ними.
- .isArray() - повертає true, якщо value – це масив, інакше false.

Array методи

- .reduce/right(callbackFn, initialValue) - використовується для обчислення якогось одного значення на основі всього масиву. Функція застосовується по черзі до всіх елементів масиву і «переносить» свій результат на наступний виклик.

reduce() є центральною концепцією функціонального програмування, де неможливо змінити будь-яке значення, тому, щоб накопичувати всі значення в масиві, потрібно повертати нове значення накопичувача на кожній ітерації. Ця угода поширюється на JavaScript-редукції(): ви повинні використовувати розповсюдження або інші методи копіювання, де це можливо, щоб створити нові масиви та об'єкти як накопичувач, а не змінювати існуючий. Якщо ви вирішили змінити акумулятор замість того, щоб його копіювати, пам'ятайте, що все одно повертайте змінений об'єкт у зворотному виклику, інакше наступна ітерація отримає undefined.

callback(accumulator, element, index, array) => { ... }

- accumulator – результат попереднього виклику цієї функції, дорівнює initial при першому виклику (якщо переданий initial),
- item – черговий елемент масиву,
- index – його індекс,
- array – сам масив.

При виконанні функції результат її виклику на попередньому елементі масиву передається як перший аргумент.

Зрозуміти простіше, якщо думати про перший аргумент як «збирач» результатів попередніх викликів функції. Після закінчення він стає результатом reduce.

thisArgs

Значення параметра thisArg стає this для func. Тобто ми можеме встановити контекст виклику для функції. Про це ми поговоримо пізніше.

Дякую за увагу