Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебная дисциплина «Программные средства создания Internet-приложений»

Инструкция
по выполнению лабораторной работы
«Работа с массивами в JavaScript. Применение методов работы с массивами»

Минск
2020

Лабораторная работа № 18

Тема работы: Работа с массивами в JavaScript. Применение методов работы с массивами

1. Цель работы

Формирование умений работы с массивами в JavaScript и использования стандартных методов для работы с массивами.

2. Задание

Выполнить задания в соответствии с порядком выполнения лабораторной работы.

3. Оснащение работы

ПК, редактор исходного кода, браузер.

4. Основные теоретические сведения

В JavaScript массивы можно создавать следующими способами:

1) с помощью литерала;

2) с помощью вызова конструктура Array();

3) С помощью метода Array.of().

Первый способ создания массива - с помощью литерала, который представляет собой простой список разделенных запятыми элементов массива в квадратных скобках. Значения в литерале массива не обязательно должны быть константами - это могут быть любые выражения, в том числе и литералы объектов:

```
let empty = [];                         // Пустой массив
let numbers = [2, 3, 5, 7, 11];         // Массив с пятью числовыми элементами
let misc = [ 1.1, true, "a", ];         // 3 элемента разных типов + завершающая запятая

let base = 1024;
let table = [base, base+1, base+2, base+3];           // Массив с переменными

let arrObj = [[1,{x:1, y:2}], [2, {x:3, y:4}]];       // 2 массива внутри, содержащие
                                                      объекты
```

Другой способ создания массива состоит в вызове конструктора **Array()**. Вызвать конструктор можно тремя разными способами:

1) вызвать конструктор без аргументов:

```
let arr = new Array();
```

2) вызвать конструктор с единственным числовым аргументом, определяющим длину массива:

```
let arr = new Array(10);
```

1) явно указать в вызове конструктора значения первых двух или более элементов массива или один нечисловой элемент:

```
let arr = new Array(5, 4, 3, 2, 1, "тест");
```

Третий способ создания массива с помощью метода **Array.of()**, который создаёт новый экземпляр массива Array из произвольного числа аргументов, вне зависимости от числа или типа аргумента.

```
Array.of(7);           // [7]
Array.of(1, 2, 3);     // [1, 2, 3]
```

Чтобы узнать длину массива, используется свойство length.

```
let myArray = new Array();

...
// узнать количество элементов
alert(myArray.length)
```

Обратите внимание, что значение Array.length на единицу больше номера последнего элемента массива, т.к. нумерация в массиве начинается с нуля, а свойство length показывает общее количество элементов.

Добавление элементов производится простой инициализацией соответствующего элемента.

```
let myArray = new Array();
myArray[0] = 'Питер';
```

Удалять элементы массива можно с помощью оператора delete, как обычные свойства объектов:

```
let arr = [1,2,'three'];
delete arr[2];
2 in arr;              // false, индекс 2 в массиве не определен
arr.length;            // 3: оператор delete не изменяет свойство length массива
```

Добавлять элементы в массив, равно как и удалять их, можно также, как и обычные свойства любых других объектов. С тем лишь отличием, что при добавлении числовых свойств может изменятся свойство length, а при изменении свойства length могут удаляться числовые свойства. В общем случае алгоритм установки свойств у массивов примерно следующий:

1) при добавлении несуществующего числового свойства i, если length меньше или равен i, то length устанавливается равным i + 1;
2) при изменении свойства length:

- если присваиваемое значение меньше 0, то бросается RangeError;
- удаляются все числовые свойства, индексы которых больше и равны новому length.

```
let a = [0, 1, 2];
alert(a.length);  // 3
a[5] = 5;
alert(a.length);  // 6 добавление элемента в крнец массива
delete a[5];
alert(a.length);  // 6: при удалении элементов length не изменяется.
a.length = 1;
alert([0 in a, 1 in a, 2 in a]);    // true,false,false: элементы с индексами 1 и 2 удалены
try {
    a.length = -1;
} catch (e) {
    alert(e.message);  // RangeError
```

Наиболее правильной конструкцией для перебора элементов массива является.

```
for (let i = 0, length = a.length; i < length; i++) {
    if (i in a) {
        // делаем что-то с элементом массива.
    }
}
```

Но для массивов возможен и другой вариант цикла, for..of:

```
let fruits = ["Яблоко", "Апельсин", "Слива"];
```

```
// проходит по значениям
for (let fruit of fruits) {
  alert( fruit );
}
```

Метод **toString()** преобразует массив в строку значений массива (разделенных запятыми).

Метод **arr.join()** преобразует все элементы массива в строки, объединяет их и возвращает получившуюся строку. В необязательном аргументе методу можно передать строку, которая будет использоваться для отделения элементов в строке результата. Если строка-разделитель не указана, используется запятая.

Метод **arr.push** добавляет переданные элементы в конец массива.

Метод **arr.pop** возвращает последний элемент массива и удаляет его.

Метод **arr.unshift** добавляет переданные элементы в начало массива, при этом существующие элементы в массиве смещаются в позиции с более высокими индексами. Элементы будут располагаться в том же порядке, в каком они были переданы.

Метод **arr.shift** возвращает первый элемент массива и удаляет его.

Метод **arr.splice()** - это универсальный метод, выполняющий вставку или удаление элементов массива. В отличие от методов slice() и concat(), метод splice() изменяет исходный массив, относительно которого он был вызван.

Метод splice() может удалять элементы из массива, вставлять новые элементы или выполнять обе операции одновременно. Он изменяет значение свойства length и сдвигает элементы массива с более низкими или высокими индексами по мере необходимости, чтобы после вставки или удаления образовывалась непрерывная последовательность.

Первый аргумент метода splice() определяет позицию в массиве, начиная с которой будет выполняться вставка и/или удаление. Если первый аргумент отрицательный, то индекс, с которого начнётся удаление, будет равен **«length + первый аргумент»**. Второй аргумент определяет количество элементов, которые должны быть удалены (вырезаны) из массива. Если второй аргумент опущен, удаляются все элементы массива от указанного до конца массива. За этими аргументами может следовать любое количество дополнительных аргументов, определяющих элементы, которые будут вставлены в массив, начиная с позиции, указанной в первом аргументе.

Метод splice() возвращает массив удаленных элементов или (если ни один из элементов не был удален) пустой массив.

Метод **arr.slice()** возвращает фрагмент, или подмассив, указанного массива. Два аргумента метода определяют начало и конец возвращаемого фрагмента. Возвращаемый массив содержит элемент, номер которого указан в первом аргументе, плюс все последующие элементы, вплоть до (но не включая) элемента, номер которого указан во втором аргументе.

Если указан только один аргумент, возвращаемый массив содержит все элементы от начальной позиции до конца массива. Если какой-либо из аргументов имеет отрицательное значение, он определяет номер элемента относительно конца массива. Так, аргументу -1 соответствует последний элемент массива, а аргументу -3 - третий элемент массива с конца.

Метод **arr.concat()** создает и возвращает новый массив, содержащий элементы исходного массива, для которого был вызван метод concat(), и значения всех аргументов, переданных методу concat(). Если какой-либо из этих аргументов сам является массивом, его элементы добавляются в возвращаемый массив.

Метод **arr.some** проверяет, удовлетворяет ли какой-либо элемент массива условию, заданному в передаваемой функции (callback). Он вернет значение true, если хотя бы один элемент совпадет с проверяемой функцией, и значение false – если нет.

Метод **arr.every** проверяет, удовлетворяют ли все элементы массива условию, заданному в передаваемой функции (callback). Он вернет значение true, если каждый элемент совпадет с проверяемой функцией, и значение false – если нет.

Метод **arr.reduce** используется для последовательной обработки каждого элемента массива с сохранением промежуточного результата.

Метод **arr.map** используется для *трансформации* массива. Он создаёт новый массив, который будет состоять из результатов вызова callback(item, i, arr) для каждого элемента arr.

Метод **arr.flat()** принимает в качестве аргумента массив массивов и сглаживает вложенные массивы в массив верхнего уровня.

Метод **arr.filter** используется для фильтрации массива через функцию. Он создаёт новый массив, в который войдут только те элементы arr, для которых вызов callback(item, i, arr) возвратит true.

Метод **forEach** используется для перебора массива. Он для каждого элемента массива вызывает функцию callback.

Метод **arr.find()** принимает функцию в качестве аргумента и в дальнейшем применяет ее к массиву. Он возвращает значение элемента, найденного в массиве, если элемент удовлетворяет условию проверяющей функции. В противном случае оно возвращается со значением undefined.

Метод **arr.findIndex()** принимает функцию в качестве параметра и в дальнейшем применяет ее к массиву. Он возвращает индекс найденного элемента, если элемент удовлетворяет условию проверяющей функции, переданной в качестве аргумента. Если не удовлетворяет, возвращается –1.

Метод **arr.sort()** сортирует элементы в исходном массиве и возвращает отсортированный массив. Если метод sort() вызывается без аргументов, сортировка выполняется в алфавитном порядке (для сравнения элементы временно преобразуются в строки, если это необходимо). Неопределенные элементы переносятся в конец массива.

Метод **arr.fill()** заполняет все элементы массива одинаковым значением, от начального индекса (по умолчанию 0) до конечного индекса (по умолчанию array.length).

Метод **arr.includes()** возвращает значение true, если массив содержит определенный элемент, и значение false – если нет.

Метод **arr.reverse()** меняет порядок следования элементов в массиве на обратный (первый элемент становится последним, а последний – первым) и возвращает переупорядоченный массив. Перестановка выполняется непосредственно в исходном массиве, т.е. этот метод не создает новый массив с переупорядоченными элементами, а переупорядочивает их в уже существующем массиве.

Метод **arr.flatMap()** применяет функцию к каждому элементу массива, а затем сглаживает результат в новый массив. Он объединяет метод **flat()** и метод **map()** в одну функцию.

Метод **Array.from()** создаёт новый экземпляр Array из массивоподобного или итерируемого объекта:

5. Порядок выполнения работы

1. Создайте три массива произвольной длины, содержащие произвольные элементы, разными способами.

2. Реализуйте создание четвертого массива, состоящего из элементов первых трех массивов. С помощью диалогового окна выведите длину массива и количество элементов в полученном массиве.

3. Используя метод arr.splice() выполните удаление 2-ух произвольных элементов из середины массива. Затем реализуйте добавление трех элементов в указанный массив начиная с 4-ой позиции. С помощью диалогового окна выведите длину массива и количество элементов в полученном массиве.

4. Выполните удаление последнего элемента в массиве двумя разными способами.

5. С помощью оператора delete выполните удаление 3-х произвольных элементов

массива.

6. Выполните вывод полученного массива: необходимо вывести длину массива, количество элементов в нем, все элементы с указанием индекса элемента массива, при этом должны выводиться только инициализированные элементы массива.

7. Выполните вывод всех элементов массива Рассчитайте среднее значение элементов массива и выполните проверку, содержится ли в массиве число, соответствующее среднему значению.

8. Очистить массив двумя разными способами.

9*. Реализуйте следующий алгоритм: пользователь вводит многозначное число. Напишите код, который вставляет двоеточие (:) между двумя нечетными числами. Например, если вводится число 55639217, то на выходе должно быть 5:563:921:7.

6. Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____

Фамилия, инициалы учащегося _____

Дата выполнения работы _____

Тема работы: _____

Цель работы: _____

Оснащение работы: _____

Результат выполнения работы: _____

_____

_____

7. Контрольные вопросы и задания

1. Как методы push/pop, shift/unshift влияют на быстродействие JavaScript?
2. Для чего служит свойство length?
3. Как можно создать многомерный массив в JavaScript?
4. Какие методы выполняю преобразование массива? В чем их назначение.
5. Перечислите методы, которые позволяют добавлять или удалять элементы в массиве.

8. Рекомендуемая литература

1. **JAVASCRIPT.RU** [Электронный ресурс] / Современный учебник JavaScript – 2007—2020 Илья Кантор. – Режим доступа: https://learn.javascript.ru. – Дата доступа: 04.03.2020.

2. **Никсон, Р.** Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон. 4-е изд. – СПб.: Питер, 2018.

3. **Симпсон, К.** ES6 и не только / К. Симпсон. – СПб.: Питер, 2017.

4. **Хавербеке, М.** Выразительный JavaScript. Современное веб-программирование / М. Хавербеке – СПб.: Питер, 2019.