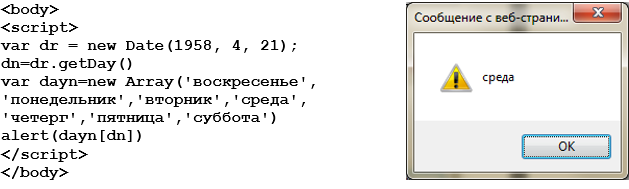
**Объект Array (продолжение)**

Объект **Array** предназначен для хранения массивов данных. Массив - это упорядоченный набор элементов. Доступ к отдельному элементу производится по имени массива и индексу (номеру) в этом массиве. Нумерация элементов начинается с нуля.

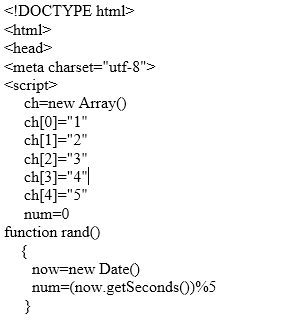
Выведем в окно названия дня недели для даты 21 мая 1958 года. (раньше вы это уже делали с помощью условных операторов) Для хранения названий дней недели создадим **dayn** – экземпляр объекта **Array**. В качестве индекса для выбора из массива названия дня недели используется вычисленное значение переменной **dn**.

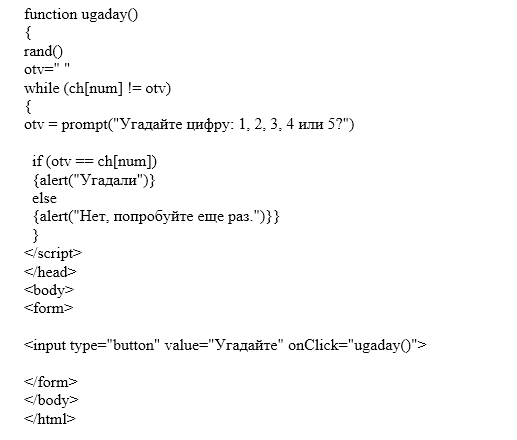


И еще один пример решения пройденного задания с помощью массива

Пользователю нужно угадать цифру. Запрос повторяется до тех пор, пока пользователь не угадает. При новой попытке, загаданное значение изменяется автоматически.

(не забываем преобразовывать классическую запись массивов к укороченной)





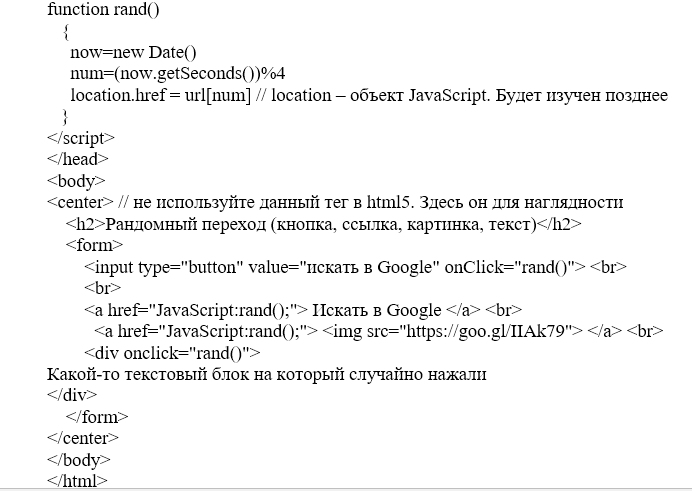
Как видите, цикл повторяется до тех пор, пока пользователь не угадает. Не совсем удобно, ведь пользователь может захотеть завершить игру. Разместите где требуется оператор breack; для выхода из цикла.

Дополните скрипт, чтобы пользователь мог угадывать цифру от 0 до 9.

После того, как проверили работоспособность сценария, вынесите его в отдельный файл и подключите на странице перед закрывающим тегом body.

Доводилось ли вам встречать ситуацию, когда переход по одной ссылке приводил вас на совершенно разные сайты? Разумеется, попав не туда, вы нажимали в браузере кнопку «Назад», снова нажимали на ссылку и…снова попадали не туда. И так делали несколько раз, пока наконец-то не попадали на нужную страницу. Благодаря вашему упорству вы увеличивали количество посещений для ряда сайтов. А может, на каком-то из них и оставались, забыв о начальной цели.





Итак, одномерный числовой массив (Array) – изучили. Как уже стало понятным, свойствами массива являются его элементы. Перейдем к изучению стандартных методов.

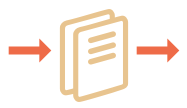
**Методы объекта Array**

В JavaScript, как и в др. языках программирования, кроме формализованных типов данных существуют еще и [абстрактные типы данных](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Абстрактные типы позволяют программистам лучше представить, а следовательно и организовать, процессы доступа к хранимым данным и алгоритмы их обработки.

В JavaScript работа с числовым массивом Array, может быть организована в виде абстрактных типов данных – [очередь](https://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) и [стек](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA).

***(Внимательно ознакомьтесь с кратким теоретическим материалом!)***

В программировании под очередью понимают упорядоченную коллекцию элементов, такую что элементы добавляются в конец, а обрабатываются – с начала.



В реальной жизни эта структура данных встречается очень часто. Например, очередь сообщений, которые надо показать на экране.

Стек - это такая коллекция элементов, в которой новые элементы добавляются в конец и берутся с конца.



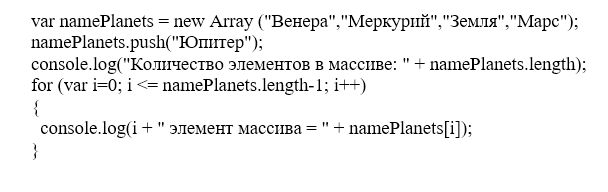
Например, стеком является колода карт, в которую новые карты кладутся сверху, и берутся – тоже сверху.

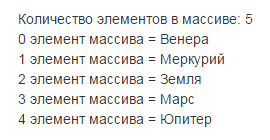
Для того, чтобы реализовывать эти структуры данных, и просто для более удобной работы с массивами существуют специальные методы.

Объект Array содержит следующие методы для работы с элементами массива: **pop(), push(), shift(),unshift(), slice(), splice(), join(), split(), reverse()**, **sort(); concat(); indexOf();**

**Добавление элемента в конец массива - push**

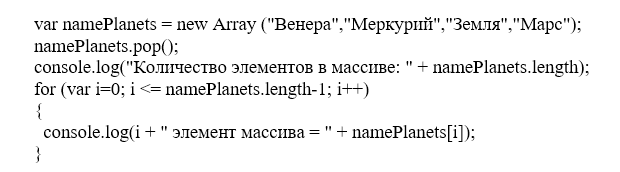
Метод push предназначен для добавления элемента в конец массива. Значение этого элемента указывается в качестве параметра данного метода. В качестве результата метод push возвращает количество элементов в массиве с учётом добавленного.

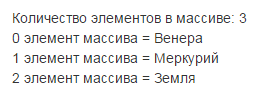




**Удаление последнего элемента массива -pop**

Метод pop предназначен для удаления последнего элемента массива. У данного метода нет параметров. В качестве результата он возвращает значение последнего (удалённого) элемента массива.

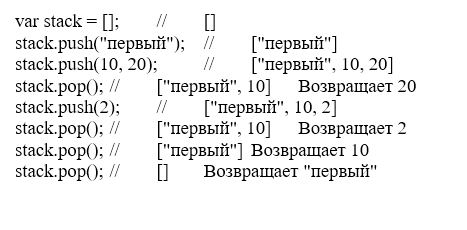




Если вы при работе с объектом Array используете методы push и pop, то вы реализует структуру данных – Стек. Тем самым вы исключаете использование других методов, характерных для структуры – Очередь.

Реализуйте маленький пример:

Убедитесь, что понимаете принцип работы методов и структуру Стека.

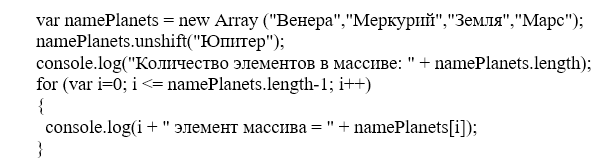


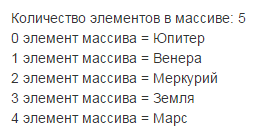
Мы посмотрели классическую реализацию Стека. Она встречается во множестве языков программирования.

JavaScript позволяет реализовать и обратный Стек, где добавление и удаление элементов осуществляется в начале массива.

**Добавление элемента в начало массива - unshift**

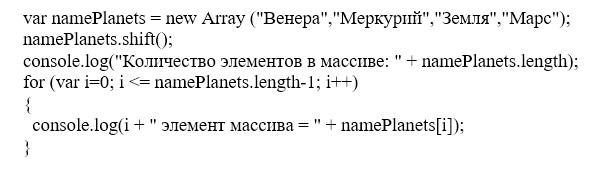
Метод unshift предназначен для добавления элемента в начало массива. Значение этого элемента указывается в качестве параметра данного метода. В качестве результата данный метод возвращает количество элементов в массиве с учётом добавленного.

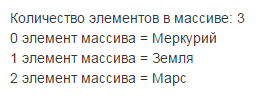




**Удаление первого элемента массива - shift**

Метод shift предназначен для удаления первого элемента из массива, т.е. элемента, имеющего индекс 0. Все остальные элементы массива смещаются к началу, т.е. у них уменьшается индекс на 1. Данный метод возвращает в качестве результата выполнения значение удалённого элемента массива.





Вызов **unshift()** добавляет аргументы (в указанном порядке) в начало массива, сдвигая существующие элементы на места с высшими индексами и соответственно увеличивая значение свойства length массива.

Например,

var myArray = [345, 78, 2];

myArray.unshift(4,"fun");

alert(myArray) ;

добавляет два элемента в начало массива.

Вызов shift() удаляет первый элемент массива, возвращает его, перемещает оставшиеся элементы влево на один индекс и уменьшает значение length. Можете думать о shift() как о сдвиге всех элементов массива влево на один индекс, в результате чего первый элемент выталкивается и возвращается. Поэтому, используя предыдущий пример, если вызвать

myArray.shift() ;

мы получим в результате массив, содержащий "fun", 345, 78 и 2. Как и в случае pop (), вызов shift ) для массива возвратит значение, которое можно использовать. Например, можно сохранить вытолкнутое из массива значение в переменной:

var х = myArray.shift();

**Гибридный Стек или Очередь**

Комбинируя методы **push(); pop(); shift(); unshift();** можно добиться интересных результатов.

Заполнение массива с обоих концов

var num = [3];

num.push(4, 5);

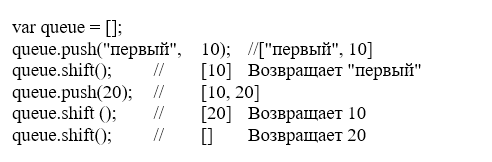
num.unshift(1, 2);

alert(num); // результат: [1, 2, 3, 4, 5]

**Используйте push() и shift() для реализации очереди.**

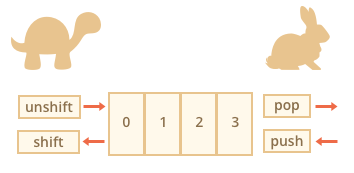
***(или pop(); и unshift();)***

Мы помещаем новые данные в конец массива и извлекаем данные, исключая элемент с нулевым индексом.



[**Быстродействие**](https://learn.javascript.ru/array#влияние-на-быстродействие)

Методы push/pop выполняются быстро, а shift/unshift – медленно.

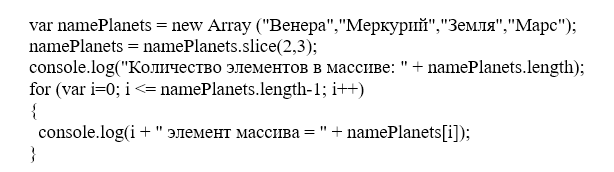


**Копирование элементов массива - slice**

Метод slice предназначен для копирования указанных элементов массива. Данный метод не изменяет исходный массив, а возвращает в качестве результата новый массив, состоящий из скопированных элементов.

Метод slice имеет 2 параметра:

* 1 параметр (обязательный) - предназначен для указания индекса элемента, с которого необходимо начинать копировать элементы;
* 2 параметр (необязательный)- предназначен для указания индекса элемента, **ДО** которого необходимо копировать (сам он не включается). Если его не указать, то будут скопированы элементы до конца указанного массива.



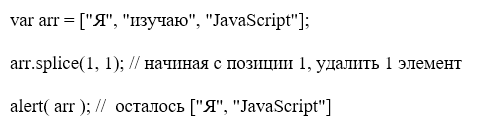
JavaScript - Метод slice

**Добавление и (или) удаление элементов - splice**

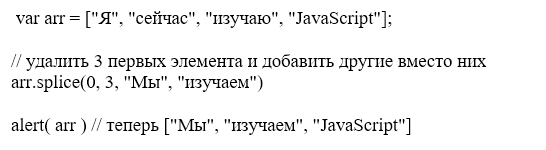
Метод splice – это универсальный метод. С его помощью можно: удалять элементы(в начале и в конце), добавлять элементы( в начало и в конец), заменять элементы.

Сперва, рассмотрим следующие примеры:

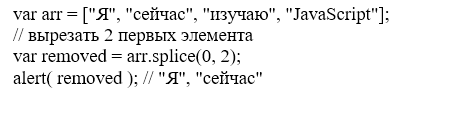
1. **Удаление**



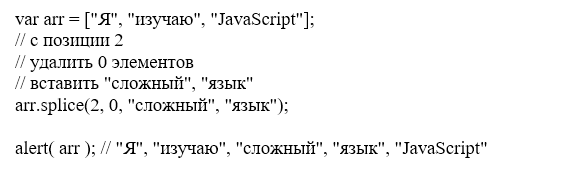
1. **Замена**



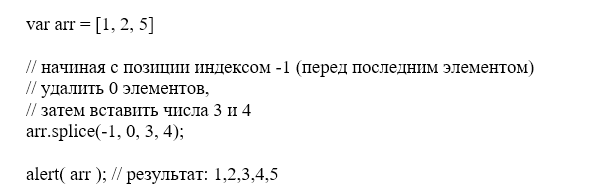
1. **Вырезать**



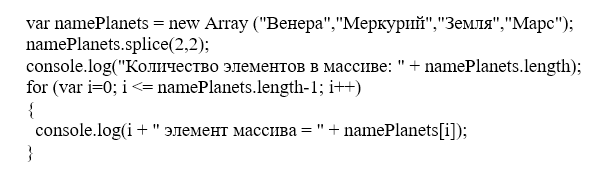
1. **Вставка без удаления** (второй параметр = 0)

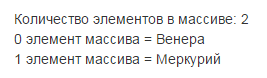


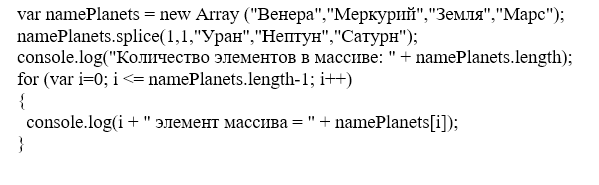
1. **Допускается использование отрицательного номера позиции, которая в этом случае отсчитывается с конца:**

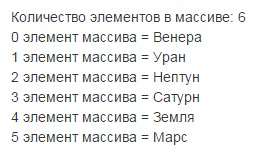


**Реализуйте следующие примеры:**





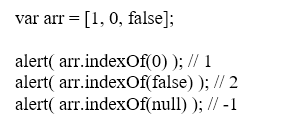




**Поиск элемента в массиве - indexOf**

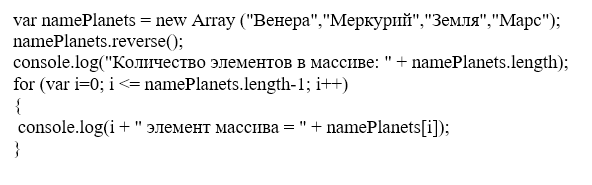
В начале знакомства с объектом Array вы уже реализовывали поиск элемента в массиве и возврат его индекса в случаи нахождения и -1 если искомый элемент отсутствует. Делали вы это с помощью цикла for.

Операция с поиском индекса относится к разряду часто используемых. Поэтому знакомый вам цикл «обернут» в стандартный метод indexOf



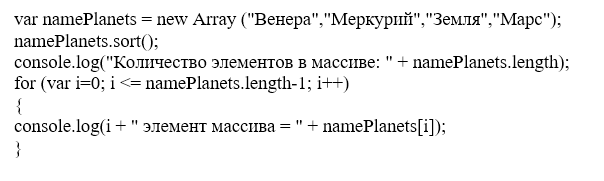
**Обратный порядок элементов -reverse**

Вы уже осуществляли смену порядка элементов с помощью переменной и поняли принцип работы. В дальнейшем, для таких целей используйте данный метод.



**Сортировка элементов массива - sort**

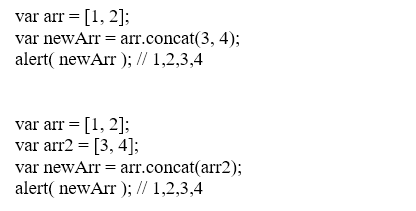
Метод sort предназначен для сортировки элементов массива. **Будьте осторожны!** По умолчанию данный метод сортирует массив как строки.



Реализуйте сортировку массива чисел с помощью данного метода.

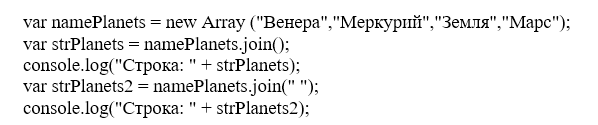
**Слияние двух массивов – concat**

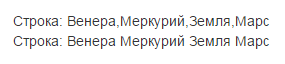
Метод создаёт новый массив, в который копируются элементы из существующего массива или значения, переданные в качестве параметров.



**Преобразование массива в строку - join**

Метод join предназначен для преобразования массива в строку. Данный метод имеет параметр, который используется как соединительная строка между элементами массива при их преобразовании строку. Если в качестве параметра ничего не указать, то для соединений элементов массива будет использоваться запятая (",").





**Преобразование строки в массив - split**

Метод split предназначен для преобразования строки в массив. Данный метод имеет один параметр, в качестве которого вы можете указать строку, на основании которой данная строчка будет разбита на массив строк.

