Неделя 16

Ответы на вопросы:

1. Какими способами можно объявлять массивы в JS?

Обычно массив создается с помощью квадратных скобок:

```
let items = [item1, item2...];
```

Ещё один вариант синтаксиса для создания массива (очень редко используется):

```
let items = new Array(item1, item2...);
```

2. Приведите 3 примера, из тех, которые не были озвучены в материалах, что могло бы быть массивом?

2.1) список работников на портале организации
2.2) перечень продуктов ежедневного рациона в дневнике питания
2.3) список просмотренных фильмов/планируемых к просмотру в онлайн-кинотеатре

3. Каким еще способом, кроме pop и shift можно удалять элементы из массивов?

Ручное уменьшение длины массива "укоротит" его, то есть удалит элементы.
Но на практике так делать абсолютно не рекомендуется.
Можно также перезаписать определенный элемент, заменить на другой с помощью обычного присваивания. то есть это не совсем удаление, а именно замена.

4. Можно ли пропускать части for? Что получится, если написать for(;;)?

```
for (инициализация; условие; завершающая операция) {
    // тело цикла
    }
```

(инициализация; условие; завершающая операция)- необязательные выражения.

Инициализация
Обычно используется, чтобы инициализировать счётчик. Это выражение может опционально объявлять новые переменные. но переменная может быть объявлена и ранее, отдельно.

Условие
Выражение, выполняющееся на каждой итерации цикла. Если выражение истинно, цикл выполняется.
Условие не является обязательным. Если его нет, условие всегда считается истиной.
Если выражение ложно, выполнение переходит к первому выражению, следующему за for.
Нужно иметь ввиду, что, если условие пропущено, необходимо в теле цикла прервать его (break), иначе будет создан бесконечный цикл.

Завершающая операция
Выражение, выполняющееся в конце итерации цикла. Можно пропустить, но также, как и при пропуске условия, нужно убедиться, что цикл прерван в теле, чтоб закончить цикл, а также изменить счётчик, так, чтобы условие для break было истинно в нужный момент.

5. Самостоятельно разберитесь, как работает цикл while и приведите два примера кода с его использованием.

```
while (условие) {
    //тело цикла
}
```

Создаёт цикл, выполняющий заданную инструкцию, пока истинно проверяемое условие. Логическое значение условия вычисляется перед исполнением тела цикла.

Цикл выполняется до тех пор, пока i меньше 10:

```
var i = 0;
while (i < 10) {
console.log(i);
i++
}
```

У пользователя запрашивается ввод числа от 1 до 10. Если пользователь введет неправильное число, будет запрошен новый ввод, и повторная проверка, соблюдено ли условие (введено ли число от 1 до 10). Код:

```
var theNumber = prompt("Пожалуйста, введите число от 1 до 10.");

while (theNumber < 1 || theNumber > 10 || isNaN(theNumber)) {
theNumber = prompt("Введено неправильное значение, пожалуйста, введите число от 1 до 10!");
}
alert("Отлично! Вы ввели число: " + theNumber);
```

6. Какой получится массив, если создать его вот так new Array(5)?

Получится пустой массив с пятью элементами. (`[empty × 5]`)

7. Как вывести чётные числа от 2 до 10 при помощи цикла for?

```
for (let i = 2; i <= 10; i = i + 2) {
    console.log(i)
}
```

// Результат работы цикла
// 2
// 4
// 6
// 8
// 10

8. Каков будет результат выполнения этого кода? Почему?

Ответ в комментариях к коду

```
let arr = ["a", "b"];       // создан массив arr с двумя элементами
arr.push(function() {     // в массив (в конец) добавляется функция (в массиве могут храниться элементы любого типа, и функция тоже)
```

```
    alert( arr );
})

arr[2]();          // получить элемент с индексом 2 (функция) и выполнить её - то
есть вывести в алерт массив.
```

Поэтому результат выполнения кода это:

```
a, b, function(){
alert( arr )
}     - список элементов массива
```

9. Три основных способа перебора элементов массива?

Цикл for (традиционный)
Цикл for ... of
Метод forEach (появился в ES5)

10. Как можно выбрать все инпуты из вашей формы регистрации из прошлого ДЗ с помощью querySelector*?

```
const inputs = document.querySelectorAll('input');
```

11. Самостоятельно разберитесь, как можно проще всего сделать сортировку в массиве на JS? Пожалуйста, не усложняйте ответ на этот вопрос 😸

Для начала нужно понять, для чего сортировка: оптимальность алгоритма сортировки тесно зависит от типа списков/массивов, которые необходимо сортировать, и даже от модели ЭВМ. Чем больше информации в распоряжении, тем более точным будет выбор.

Например, bubble sort (сортировка пузырьком) работает медленно на тестах, в которых маленькие элементы стоят в конце, потому что такой элемент на каждом шаге алгоритма будет двигаться всего на 1 шаг. Поэтому в данном случае оптимальнее будет шейкерная сортировка (Shaker sort), когда мы идем по массиву не только слева направо, но и справа налево. Или сортировка расчёской (Comb sort), когда первоначально задается большое расстояние между элементами (длина массива), и по мере упорядочивания оно сужается.

12. Как можно принудительно остановить выполнение цикла?

С помощью директивы break.

Практическое задание 💎

1. Напишите функцию `sumInput()`…

   Функция прописана в файле avto.js, как событие на кнопку «нажми меня».

2. Напишите калькулятор расчета стоимости автомобиля в зависимости от комплектации.

   В репозитории.

3. Задачи с массивами являются одним из часто встречаемых в работе программиста и, как следствие, на собеседованиях. Поэтому давай потренируемся использовать изученные методы.

Код по следующим заданиям закомментирован в файле avto.js, привожу его также ниже:

1. Дан массив `['js', 'css', 'html']`. Выведите на экран первый элемент.

```
let example = ['js', 'css', 'html'];
alert( example[0] );
```

2. Отфильтруйте массив `[0, 1, false, 2, undefined, '', 3, null]` от нежелательных значений, таких как `false`, `undefined`, пустые строки, `0`, `null` с помощью метода `filter`. Ожидаемый результат: `[1, 2, 3]`.

```
let onlyNumbers = [0, 1, false, 2, undefined, '', 3, null];
let filteredNumbers = onlyNumbers.filter(Boolean);
alert(filteredNumbers);
```

3. Дан массив `[[1,2], [1,2,3], [1,2,3,4]]`. Найдите индекс массива, длина которого `> 3`. Ожидаемый результат: `2`

```
let numb = [[1,2], [1,2,3], [1,2,3,4]];
const el = numb.findIndex(el=>el>3);
alert(el);       //   вернется -1, так как такого массива нет
```