1. Какими способами можно объявлять массивы в JS?

- Пустой массив можно легко описать с помощью квадратных скобок:

```
let items = [];
```

Можно также сразу создать массив со значениями элементов:

- new Array()

2. Приведите 3 примера, из тех, которые не были озвучены в материалах, что могло бы быть массивом?
   Как может выглядеть ваш ответ: например, список книг в приложении "Библиотека" - это массив. Список людей в паспортном столе - это массив. Список полей в форме регистрации - это массив.

- *Список номеров телефонов в записной книжки*
- *Список ФИО учеников в классе*

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

- delete
- arr.splice(str)
- arr.slice[]

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

Любая часть for может быть пропущена.

- Мы можем пропустить начало если нам ничего не нужно делать перед стартом цикла:

```
let i = 0; // мы уже имеем объявленную i с присвоенным значением

for (; i < 3; i++) { // нет необходимости в "начале"
  alert( i ); // 0, 1, 2
}
```

- Можно убрать и шаг:

```
let i = 0;

for (; i < 3;) {
  alert( i++ );
}
```

Это сделает цикл аналогичным `while (i < 3)`.

- Можно и вообще убрать всё, получив бесконечный цикл:

```
for (;;) {
  // будет выполняться вечно
}
```

При этом сами точки с запятой ; обязательно должны присутствовать, иначе будет ошибка синтаксиса.

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

Цикл while имеет следующий синтаксис:

```
while (condition) {
  // код
  // также называемый "телом цикла"
}
```

Код из тела цикла выполняется, пока условие condition истинно.

Например, цикл ниже выводит i, пока i < 3:

```
let i = 3;
while (i) { // когда i будет равно 0, условие станет ложным, и цикл остановится
  alert( i );
  i--;
}
```

Одно выполнение тела цикла по-научному называется *итерация*. Цикл в примере выше совершает три итерации.

Если бы строка i++ отсутствовала в примере выше, то цикл бы повторялся (в теории) вечно. На практике, конечно, браузер не позволит такому случиться, он предоставит пользователю возможность остановить «подвисший» скрипт, а JavaScript на стороне сервера придётся «убить» процесс.

Любое выражение или переменная может быть условием цикла, а не только сравнение: условие while вычисляется и преобразуется в логическое значение.

Например, while (i) – более краткий вариант while (i != 0):

```
let i = 3;
while (i) { // когда i будет равно 0, условие станет ложным, и цикл остановится
  alert( i );
  i--;
}
```

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

На месте элементов пустота , length = 5

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

```
for (let i = 2; i <= 10; i++) {
  if (i % 2 == 0) {
    alert( i );
  }
}
```

8. Каков будет результат выполнения этого кода? Почему?

```
let arr = ["a", "b"];

arr.push(function() {
alert( arr );
```

```
})

arr[2](); // a,b function
```

У массива в итоге 3 элемента: сначала их было 2, плюс функция.

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*`?

document.querySelectorAll('input');

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

реальном коде массивы сортируют, используя уже готовые функции стандартной библиотеки.
В JavaScript сортировка выполняется с помощью метода sort() массивов
Самый популярный для обучения — пузырьковая сортировка (bubble sort).
Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на свое место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива

12. Как можно принудительно остановить выполнение цикла?

`break/continue`