Array (Массивы):

Массивы в JavaScript представляют упорядоченные списки элементов.

```
let numbers = [1, 2, 3, 4, 5];
let fruits = ["apple", "orange", "banana"];


// Доступ к элементам массива
let firstNumber = numbers[0]; // 1
let secondFruit = fruits[1]; // "orange"


// Длина массива
let length = numbers.length; // 5


// Добавление элемента в конец массива
fruits.push("grape");


// Удаление последнего элемента
let lastFruit = fruits.pop();
```

Циклы (For и While):

Цикл `for`:

```
for (let i = 0; i < 5; i++) {
  console.log(i);
```

```
}
```

Цикл `while`:

```
let count = 0;


while (count < 5) {
  console.log(count);
  count++;
}
```

Методы строк (String Methods):

Строки в JavaScript имеют множество встроенных методов для работы с текстом.

```
let text = "Hello, World!";


// Длина строки
let length = text.length; // 13


// Преобразование в верхний/нижний регистр
let upperCase = text.toUpperCase(); // "HELLO, WORLD!"
let lowerCase = text.toLowerCase(); // "hello, world!"


// Получение подстроки
let substring = text.substring(0, 5); // "Hello"


// Поиск подстроки
let indexOfWorld = text.indexOf("World"); // 7
```

Объект Math:

Объект `Math` предоставляет математические методы и константы.

```
// Округление числа вверх
let roundedUp = Math.ceil(4.2); // 5


// Округление числа вниз
let roundedDown = Math.floor(4.8); // 4
```

```
// Возведение в степень
let power = Math.pow(2, 3); // 8


// Генерация случайного числа от 0 до 1
let random = Math.random(); // (например, 0.738496...)
```

Разница между массивами в js и java

Хотя массивы в JavaScript (JS) и Java имеют схожие названия и обеспечивают похожую функциональность, они имеют некоторые существенные различия:

1. **Динамическая типизация в JavaScript:**
    - **JS:** Переменные в JavaScript могут хранить значения разных типов, и массивы не ограничены определенным типом данных. Вы можете хранить в массиве значения различных типов, таких как числа, строки, объекты и другие массивы.
    - **Java:** В Java массивы строго типизированы. Это означает, что при создании массива вы должны указать тип данных, который он будет содержать, и массив будет принимать только значения этого типа.
2. **Динамическое изменение размера в JavaScript:**
    - **JS:** Массивы в JavaScript динамически изменяют свой размер. Вы можете добавлять или удалять элементы в массиве без явного указания размера.
    - **Java:** В Java размер массива определяется при его создании и не может быть изменен. Если вам нужно изменить размер массива, вам придется создать новый массив.
3. **Интерфейс и методы:**
    - **JS:** В JavaScript массивы предоставляют богатый набор методов для работы с данными, таких как `push`, `pop`, `shift`, `unshift`, `splice` и другие.
    - **Java:** В Java массивы предоставляют ограниченный набор методов, и большинство операций с массивами выполняются с использованием стандартных циклов.

Примеры:

JavaScript:

```
let jsArray = [1, "two", { three: 3 }, [4, 5]];
jsArray.push(6); // Добавление элемента в конец массива
let element = jsArray.pop(); // Удаление последнего элемента
```

Java:

```
int[] javaArray = {1, 2, 3, 4, 5};
// Размер массива неизменен, и мы не можем добавить/удалить элементы
// без создания нового массива с другим размером.
```

Таким образом, важно учитывать различия в подходах к массивам в JavaScript и Java, особенно в контексте динамической типизации и динамического изменения размера массивов в JavaScript.

Методы `push`, `pop`, `shift` и `unshift`

Методы `push`, `pop`, `shift` и `unshift` предоставляют удобные способы изменения содержимого массива в JavaScript.

1. **`push()`**: Добавляет один или несколько элементов в конец массива и возвращает новую длину массива.

```
let fruits = ['apple', 'banana'];
let length = fruits.push('orange', 'pear');
// fruits теперь ['apple', 'banana', 'orange', 'pear']
// length теперь 4
```

2. **`pop()`**: Удаляет последний элемент из массива и возвращает его. Массив укорачивается на один элемент.

```
let fruits = ['apple', 'banana', 'orange'];
let lastFruit = fruits.pop();
// fruits теперь ['apple', 'banana']
// lastFruit теперь 'orange'
```

3. **`shift()`**: Удаляет первый элемент из массива и возвращает его. Массив укорачивается на один элемент.

```
let fruits = ['apple', 'banana', 'orange'];
let firstFruit = fruits.shift();
// fruits теперь ['banana', 'orange']
// firstFruit теперь 'apple'
```

4. **`unshift()`** : Добавляет один или несколько элементов в начало массива и возвращает новую длину массива.

```
let fruits = ['banana', 'orange'];
let length = fruits.unshift('apple', 'pear');
// fruits теперь ['apple', 'pear', 'banana', 'orange']
// length теперь 4
```

Эти методы являются мощными инструментами для манипуляции массивами в JavaScript и используются для добавления, удаления и изменения элементов в начале или конце массива.