JS

JAVASCRIPT ARRAYS

Shaik Ahmad Nawaz

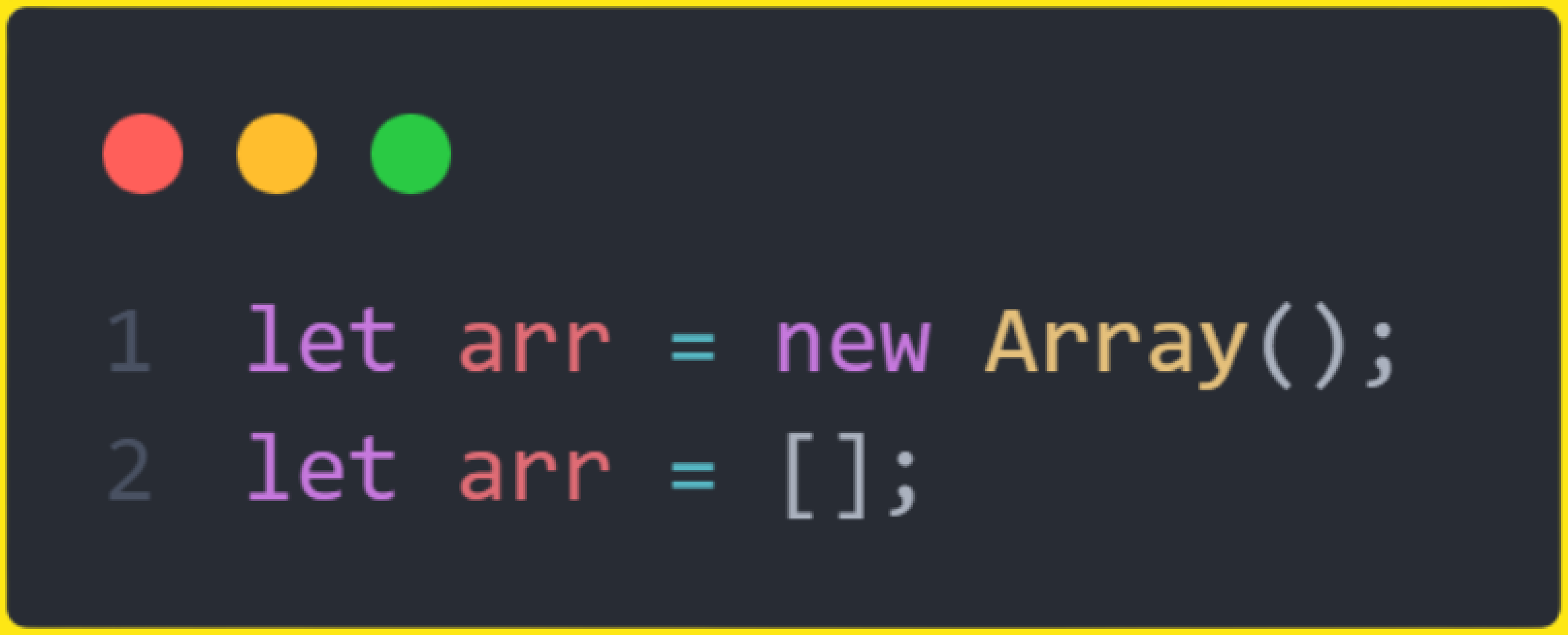
SWIPE

Arrays

- The array is a **collection** of data items of the **same type**.
- In simple terms, it is a variable that contains multiple values.

Declaration:

There are **two** syntaxes for creating an empty array:



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

SWIPE

- Array elements are numbered, starting with **zero**.
- We can get an element by its number in square brackets:

```
let fruits = ["Apple", "Orange", "Plum"];

alert(fruits[0]); // Apple
alert(fruits[1]); // Orange
alert(fruits[2]); // Plum
```

We can **replace** an element or add a **new one** to the array:

```
fruits[2] = "Pear"; // now ["Apple", "Orange", "Pear"]
fruits[3] = "Lemon"; // now ["Apple", "Orange", "Pear", "Lemon"]
```

SWIPE

Methods that work with the **end** of the array:

pop

Extracts the **last element** of the array and returns it:

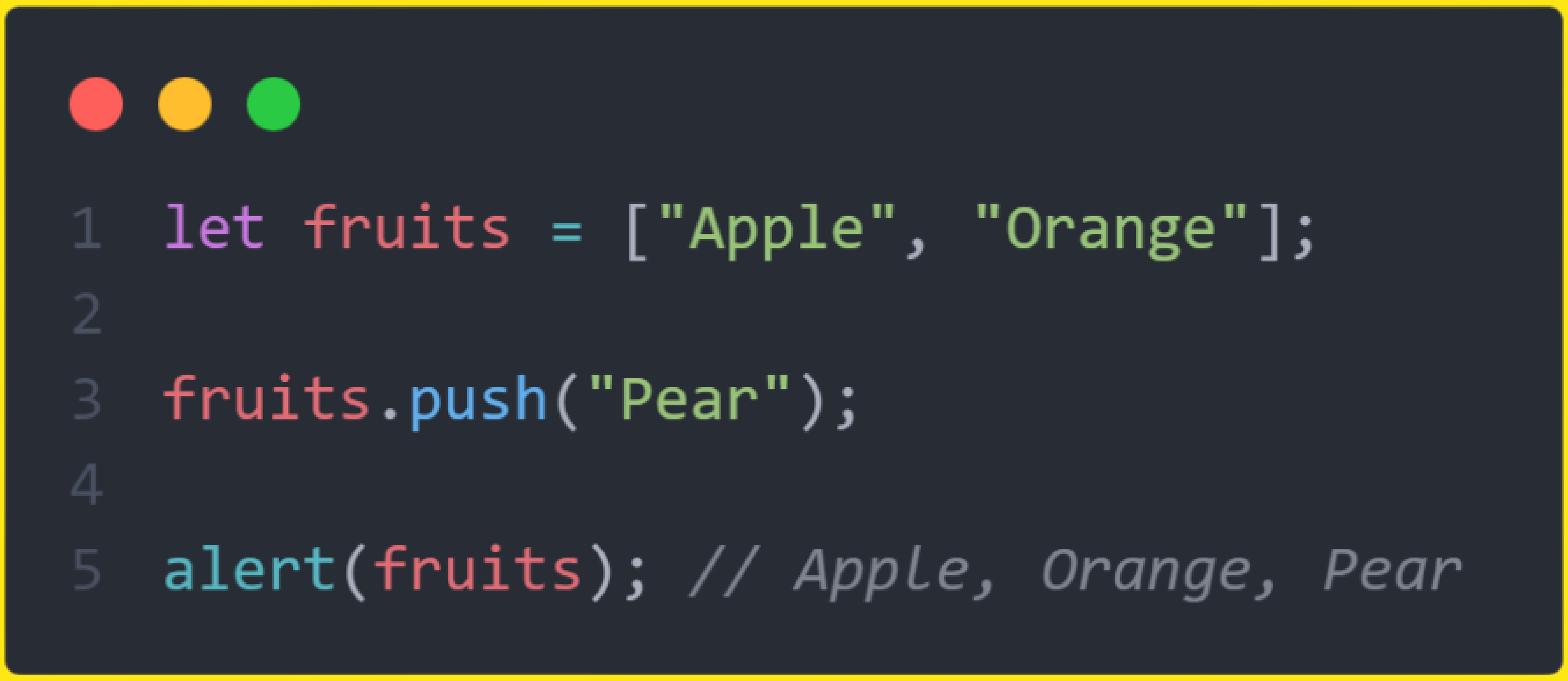
```javascript
let fruits = ["Apple", "Orange", "Pear"];

alert(fruits.pop()); // remove "Pear" and alert it

alert(fruits); // Apple, Orange
```

Both **fruits.pop()** and **fruits.at(-1)** return the **last** element of the array, but **fruits.pop()** also modifies the array by removing it.

SWIPE

push

Append the element to the **end** of the array:

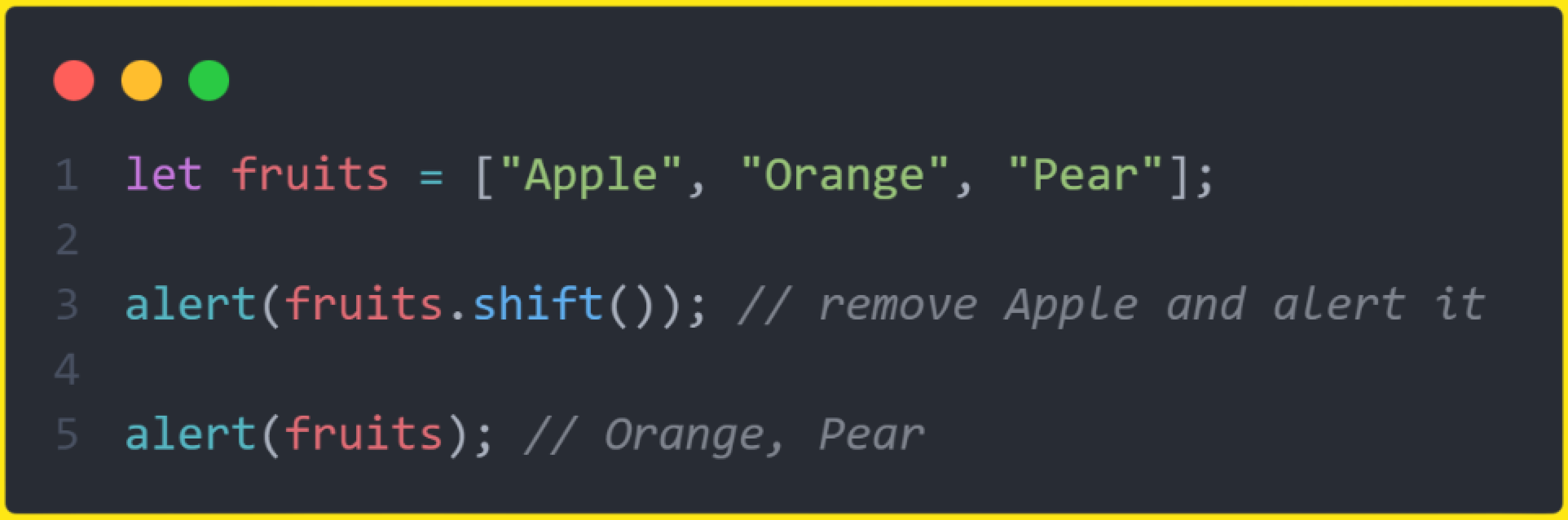


```javascript
let fruits = ["Apple", "Orange"];

fruits.push("Pear");

alert(fruits); // Apple, Orange, Pear
```

Methods that work with the **beginning** of the array:

shift

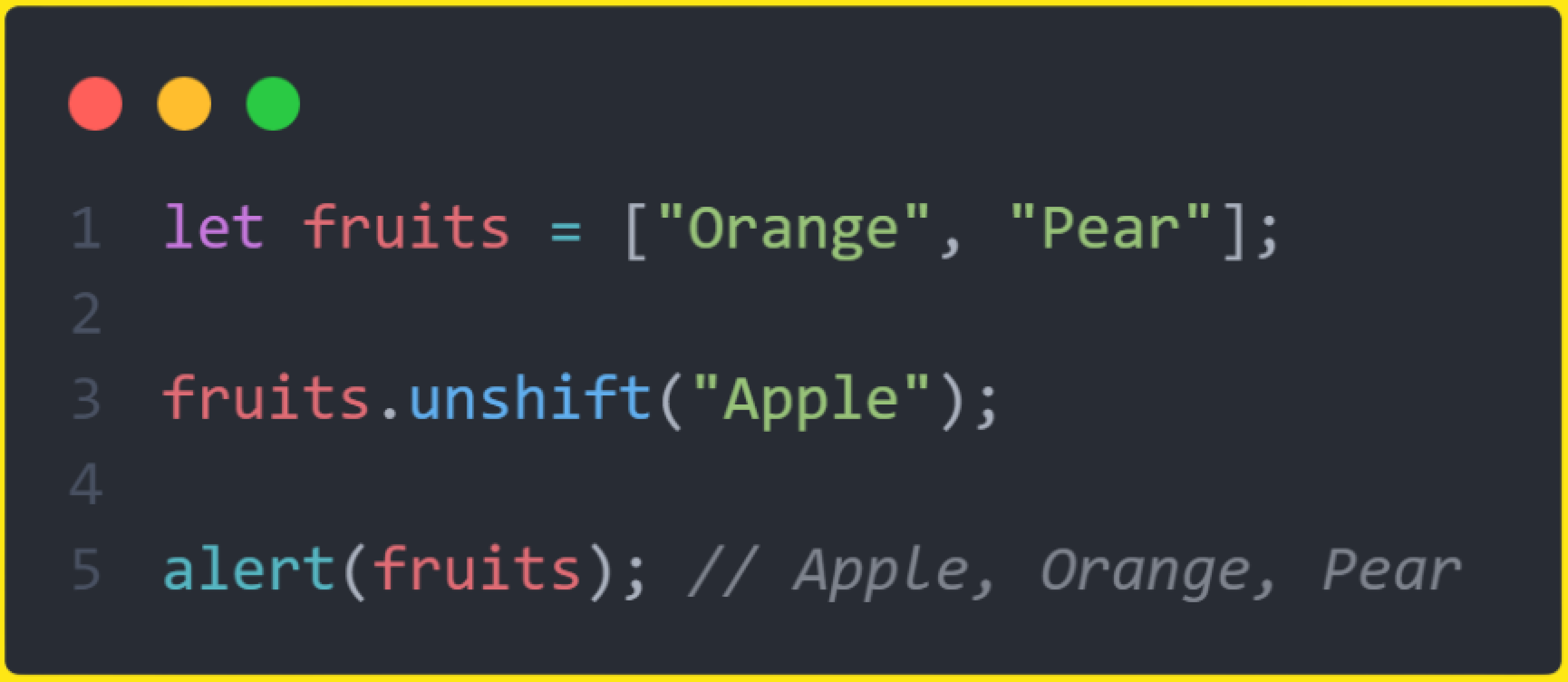
Extracts the **first** element of the array and returns it:



```javascript
let fruits = ["Apple", "Orange", "Pear"];

alert(fruits.shift()); // remove Apple and alert it

alert(fruits); // Orange, Pear
```

unshift

Add the element to the **beginning** of the array:

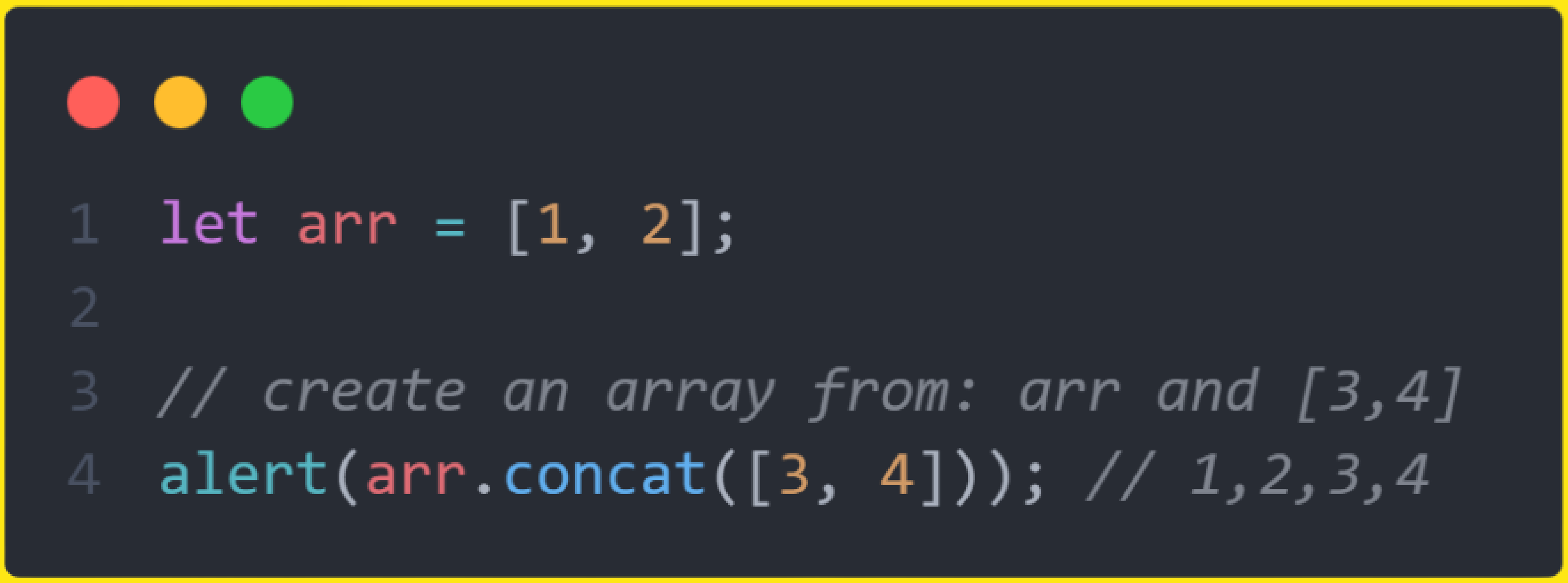


```javascript
let fruits = ["Orange", "Pear"];

fruits.unshift("Apple");

alert(fruits); // Apple, Orange, Pear
```

SWIPE

concat

The method **arr.concat** creates a new array that includes values from other arrays and additional items.



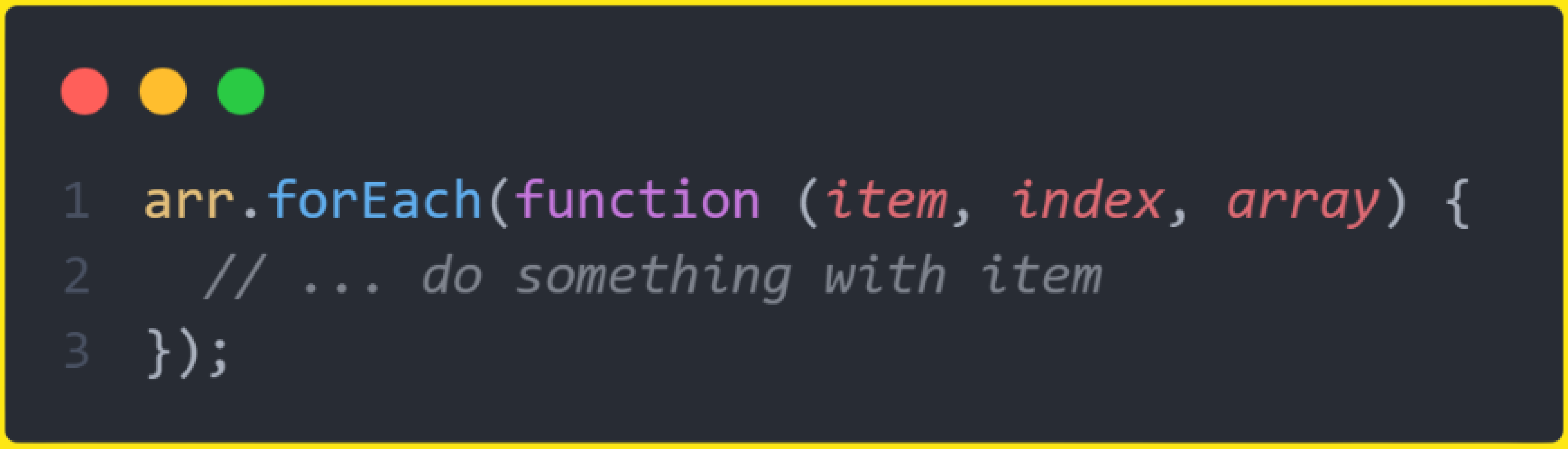
```javascript
let arr = [1, 2];

// create an array from: arr and [3,4]
alert(arr.concat([3, 4])); // 1,2,3,4
```

Iterate: forEach

The **arr.forEach** method allows to **run a function for every element** of the array.

SWIPE

The syntax:



```
arr.forEach(function (item, index, array) {
  // ... do something with item
});
```

A **cheat sheet** of array methods:.

- **slice(start, end) –** creates a new array, copies elements from index start till end (not inclusive) into it.
- **indexOf/lastIndexOf(item, pos) –** look for item starting from position pos, return the index or -1 if not found.

- **map(func) –** creates a new array from results of calling func for every element.
- **sort(func) –** sorts the array in-place, then returns it.
- **reverse() –** reverses the array in-place, then returns it.
- **split/join –** convert a string to array and back.
- **reduce/reduceRight(func, initial) –** calculate a single value over the array by calling func for each element and passing an intermediate result between the calls.

SWIPE

Did You Find It Useful?

Shaik Ahmad Nawaz

Follow For More!