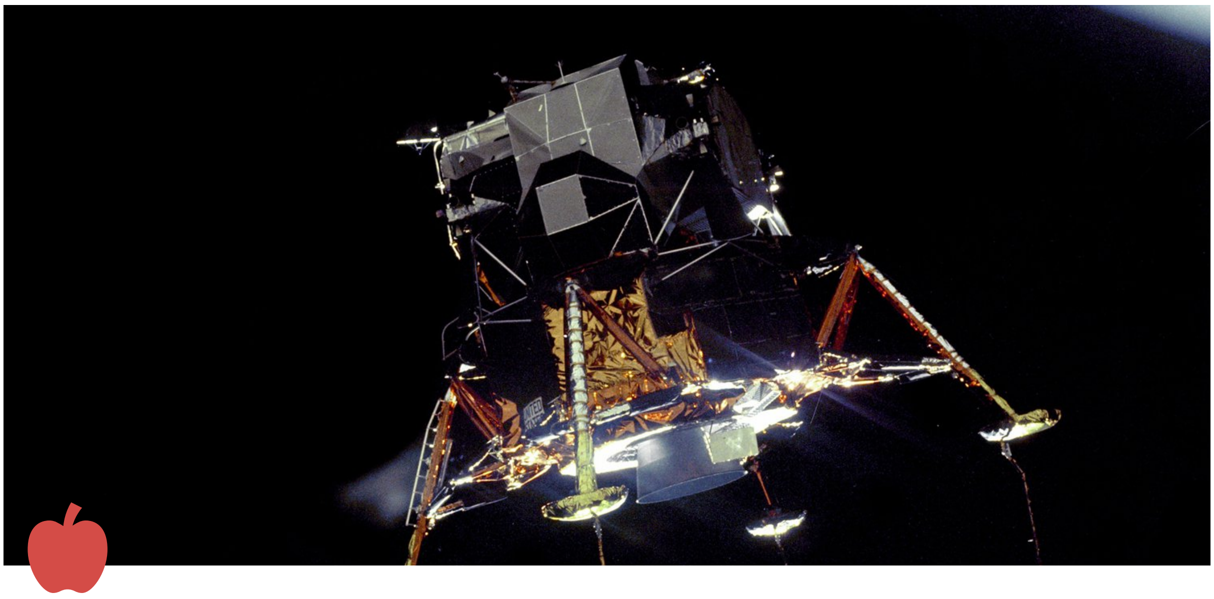
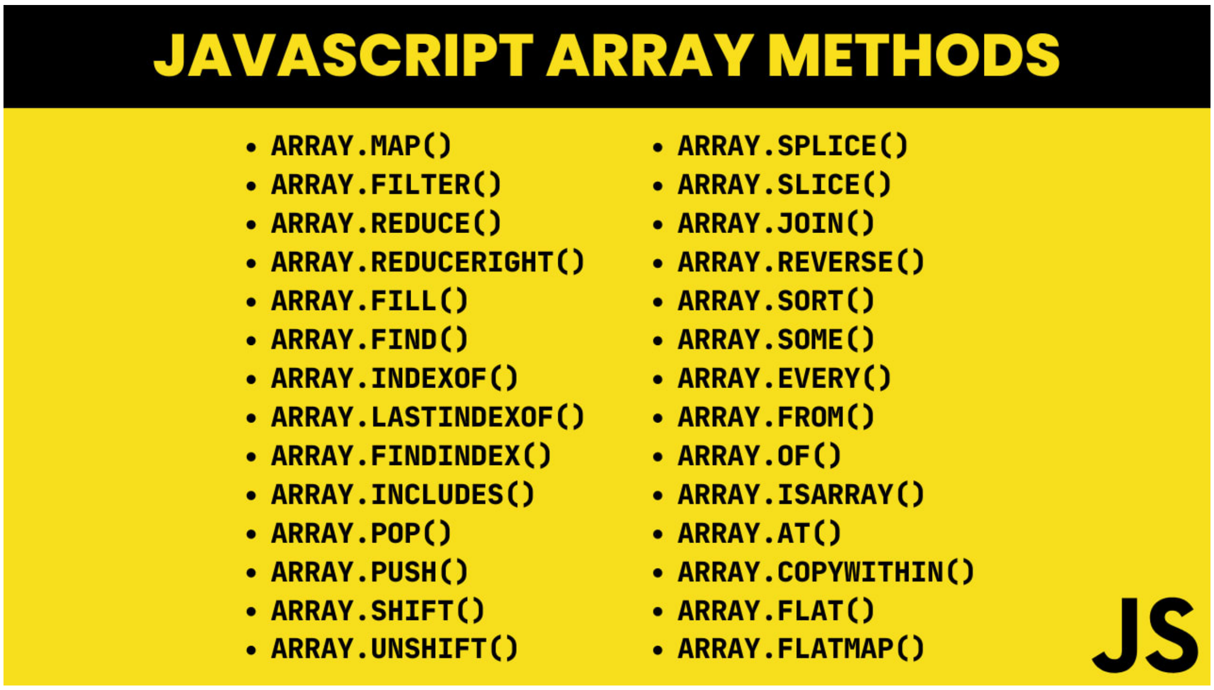


JavaScript Higher Order Functions

Baraneetharan Ramasamy

JavaScript Array Methods

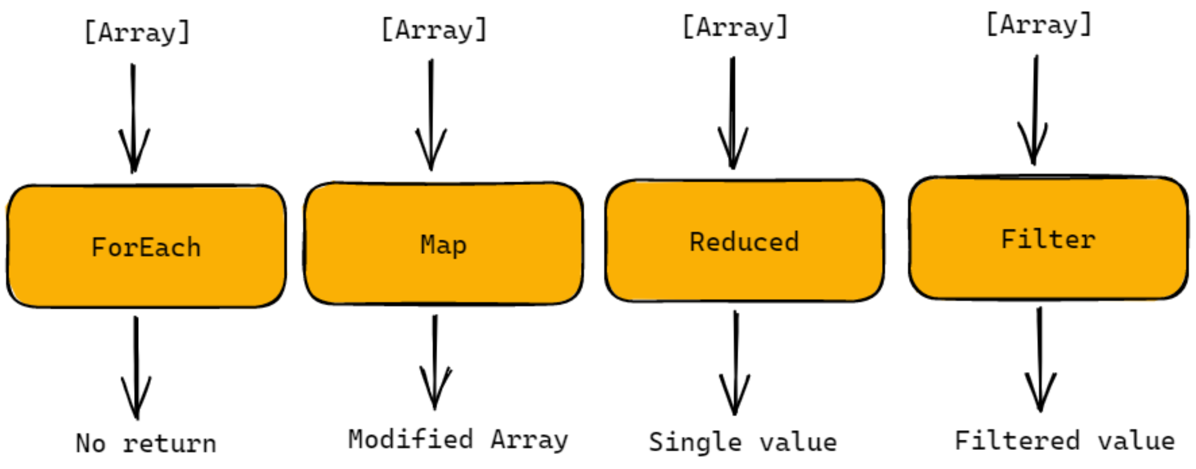


JAVASCRIPT ARRAY METHODS

- ARRAY.MAP()
- ARRAY.FILTER()
- ARRAY.REDUCE()
- ARRAY.REDUCERIGHT()
- ARRAY.FILL()
- ARRAY.FIND()
- ARRAY.INDEXOF()
- ARRAY.LASTINDEXOF()
- ARRAY.FINDINDEX()
- ARRAY.INCLUDES()
- ARRAY.POP()
- ARRAY.PUSH()
- ARRAY.SHIFT()
- ARRAY.UNSHIFT()
- ARRAY.SPLICE()
- ARRAY.SLICE()
- ARRAY.JOIN()
- ARRAY.REVERSE()
- ARRAY.SORT()
- ARRAY.SOME()
- ARRAY.EVERY()
- ARRAY.FROM()
- ARRAY.OF()
- ARRAY.ISARRAY()
- ARRAY.AT()
- ARRAY.COPYWITHIN()
- ARRAY.FLAT()
- ARRAY.FLATMAP()

JS

Higher Order Functions



```
const students = [
    { name: 'Alice', age: 20, grade: 'A' },
    { name: 'Bob', age: 22, grade: 'B' },
    { name: 'Charlie', age: 21, grade: 'A' },
    { name: 'David', age: 19, grade: 'C' },
    { name: 'Eve', age: 20, grade: 'B' }
];

// Using map to transform objects in the array
const studentNames = students.map(student => student.name);
console.log(studentNames);
// Output: ['Alice', 'Bob', 'Charlie', 'David', 'Eve']

// Using filter to filter objects in the array
const studentsWithGradeA = students.filter(student => student
console.log(studentsWithGradeA);
// Output: [{ name: 'Alice', age: 20, grade: 'A' }, { name: '

// Using reduce to aggregate information from objects in the
const totalAge = students.reduce((total, student) => total +
console.log(totalAge);
// Output: 102
```

```javascript
// Using reduce to create an object with aggregated informati
const gradeCounts = students.reduce((counts, student) => {
    if (!counts[student.grade]) {
        counts[student.grade] = 1;
    } else {
        counts[student.grade]++;
    }
    return counts;
}, {});

console.log(gradeCounts);
// Output: { A: 2, B: 2, C: 1 }

// Using forEach to perform an action for each object in the
students.forEach(student => {
    console.log(`Student: ${student.name}, Age: ${student.age
});
// Output:
// Student: Alice, Age: 20, Grade: A
// Student: Bob, Age: 22, Grade: B
// Student: Charlie, Age: 21, Grade: A
// Student: David, Age: 19, Grade: C
// Student: Eve, Age: 20, Grade: B

// Chaining map, filter, and reduce
const gradeACount = students
    .filter(student => student.grade === 'A')
    .map(student => student.name)
    .reduce((count, name) => count + 1, 0);

console.log(gradeACount);
// Output: 2
```

`map`

- The `map()` function transforms each element of an array based on a given function.

- In your example, we use `map()` to extract the names of students from the `students` array.
- The result is an array of student names: `['Alice', 'Bob', 'Charlie', 'David', 'Eve']`.

`filter`

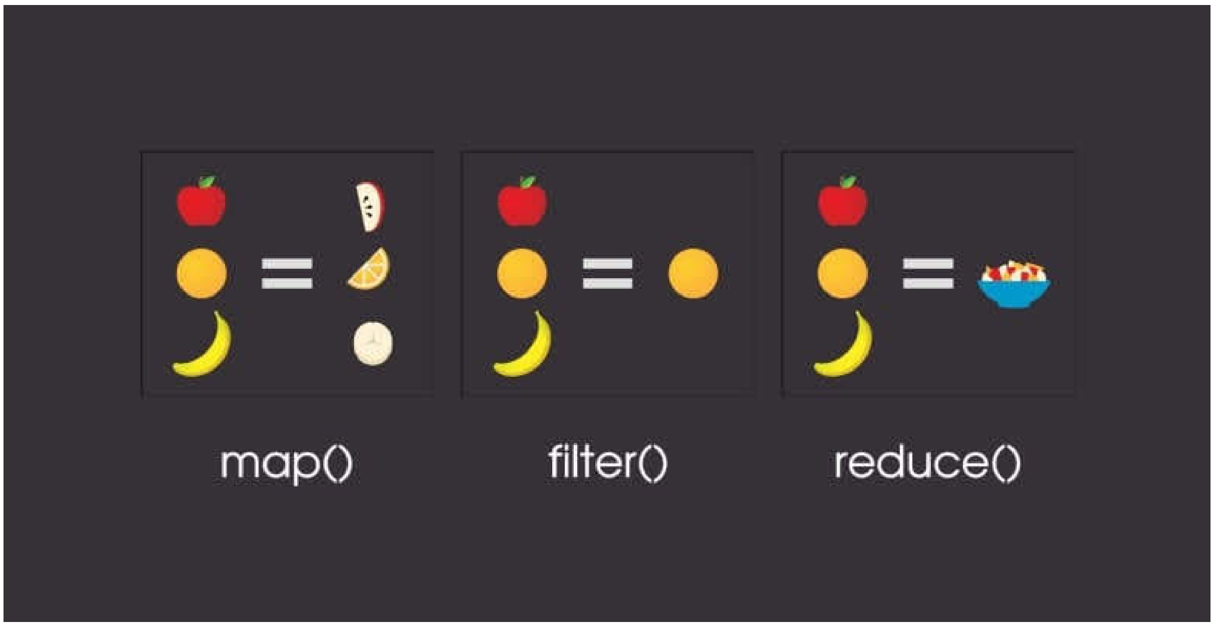
- The `filter()` function creates a new array containing elements that satisfy a given condition.
- In your code, we filter students who have a grade of 'A'.
- The result is an array of students with grade 'A': `[{ name: 'Alice', age: 20, grade: 'A' }, { name: 'Charlie', age: 21, grade: 'A' }]`.

`reduce`

- The `reduce()` function aggregates information from an array into a single value.
- In your example, we calculate the total age of all students.
- The result is the sum of ages: `102`.

`forEach`

The `forEach` function is useful when you want to perform an action for each element in an array without creating a new array as a result.



Chaining

`map()`, `filter()`, and `reduce()`:

- You can chain these functions together to perform multiple operations in sequence.
- In your code, we first filter students with grade 'A', then map their names, and finally count the number of students.
- The result is `2`, representing the number of students with grade 'A'.