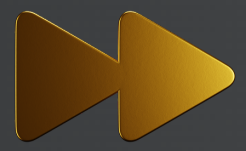
Saif Mujawar
@saifmujawar7

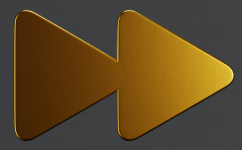


Some Important Array.reduce() Examples

1. Flatten Arrays.
2. Counting Instances
3. Grouping Objects by property
4. Function composition enabling piping
5. Function composition enabling piping

Part - 02

Saif Mujawar
@saifmujawar7

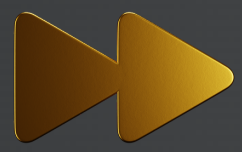


Flatten Arrays

Flattening array with reduce

```
const flattened = (arr) => {
  const someArr = arr.reduce(
  (previousValue, currentValue) => {
    if(Array.isArray(currentValue)){
        previousValue = previousValue.concat(flattened(currentValue))
    }else{
        previousValue.push(currentValue)
    };
      return previousValue
  },
  []
)
  return someArr
}

// flattened is [0, 1, 2, 3, 4, 5, 6, 8, 10, 45, 88]

console.log(flattened([[0,1], [2,[3]], [4,[5,6,8],[10,[45],[88]]]]))
```

Saif Mujawar
@saifmujawar7

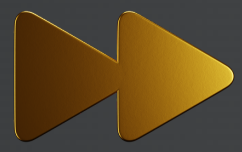


Counting instances of values in an object

Counting instances of values in an object

```
const names = ['Alice', 'Bob', 'Tiff', 'Bruce', 'Bob', 'Alice']

const countedNames = names.reduce((allNames, name) => {
  allNames[name] = allNames[name] ? allNames[name]+=1 : 1;
  return allNames
}, {})

// countedNames is:
// { 'Alice': 2, 'Bob': 2, 'Tiff': 1, 'Bruce': 1 }

console.log(countedNames)
```

Saif Mujawar
@saifmujawar7

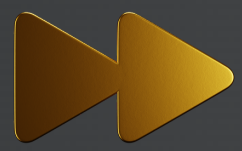


Grouping objects by a property

```
const people = [
  { name: 'Saif', age: 21 },
  { name: 'Ray', age: 20 },
  { name: 'John', age: 20 }
];

function groupBy(objectArray, property) {
  return objectArray.reduce(function (acc, obj) {
    let key = obj[property]
    if (!acc[key]) {
      acc[key] = []
    }
    acc[key].push(obj)
    return acc
  }, {})
}

let groupedPeople = groupBy(people, 'age')

// groupedPeople is:
// {
//   20: [
//     { name: 'Ray', age: 20 },
//     { name: 'John', age: 20 }
//   ],
//   21: [{ name: 'Saif', age: 21 }]
// }
```

Grouping objects by a property

Saif Mujawar
@saifmujawar7

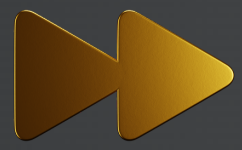


Map using reduce

Map using reduce

```
if (!Array.prototype.mapUsingReduce) {
  Array.prototype.mapUsingReduce = function(callback, initialValue) {
    return this.reduce(function(mappedArray, currentValue, currentIndex, array) {
      mappedArray[currentIndex] = callback.call(initialValue, currentValue, currentIndex,
array)
      return mappedArray
    }, [])
  }
}

[1, 2, , 3].mapUsingReduce(
  (currentValue, currentIndex, array) => currentValue + currentIndex + array.length
) // [5, 7, , 10]
```

Saif Mujawar
@saifmujawar7



Function composition enabling piping

Function composition enabling piping

```
// Building-blocks to use for composition
const double = x => x + x
const triple = x => 3 * x
const quadruple = x => 4 * x

// Function composition enabling pipe functionality
const pipe = (...functions) => initialValue => functions.reduce(
    (acc, fn) => fn(acc),
    initialValue
)

// Composed functions for multiplication of specific values
const multiply6 = pipe(double, triple)
const multiply9 = pipe(triple, triple)
const multiply16 = pipe(quadruple, quadruple)
const multiply24 = pipe(double, triple, quadruple)

// Usage
multiply6(6)   // 36
multiply9(9)   // 81
multiply16(16) // 256
multiply24(10) // 240
```

Turn on notifications 🔔

And That's it!!!

Did you find it useful?

Saif Mujawar
@saifmujawar7

Follow for more!!