Color Sorting Algorithm

By: Uzziel Kyle Ynciong

The Program:

Classes

- Color
- ColorsArray
  - ColorsArray.sort()

Functions

- randomColorsGenerator()

Color

```
class Color {
    constructor(color) {
        this.name = color
        this.visual = {
            'red': '❤️ ',
            'white': '⚪ ',
            'blue': '🔷 ',
            'green': '🍀 '
        }[color]
        this.value = {
            'red': 0,
            'white': 1,
            'blue': 2,
            'green': 3
        }[color]
    }
}
```

ColorsArray

```
class ColorsArray {
    constructor(length) {
        this.length = length
        this.colors = randomColorsGenerator(num=this.length)
    }

    sort() {
        //
        // codes here
        //
    }
}
```

randomColorsGenerator()

```
function randomColorsGenerator(num, min = 2) {
    let colors = [new Color('red'), new Color('white'), new Color('blue'), new Color('green')]

    if (num <= colors.length) {
        if (num < min) {
            num = min
        }

        shuffleArray(colors)

        randomColors = []
        for (let i = 0; i < num; i++) {
            randomColors.push(colors[i])
        }

        return randomColors
    }
```

randomColorsGenerator() - cont.

```
    randomColors = []
    colors.forEach(color => {
        randomColors.push(color)
    })

    let remainingSlots = num - randomColors.length
    for (let i=0; i < remainingSlots; i++) {
        let randomIndex = Math.floor(Math.random() * (colors.length-1))
        let newColor = colors[randomIndex]
        randomColors.push(newColor)
    }

    shuffleArray(randomColors)

    return randomColors
}
```

ColorsArray - sorting self method

```
class ColorsArray {
    ...
    sort()
}
```

THE SORTING ALGORITHM

ColorsArray.sort()

```
sort() {
        let this.colors = Array.from(this.colors)

        let sortedColors = []
        let num = 0
        while (sortedColors.length != this.length) {
            let groupedColors = []
            this.colors.forEach(color => {
                if (color.value == num) {
                    groupedColors.push(color)
                }
            })

            sortedColors.push(...groupedColors)
            num++
        }

        return sortedColors
    }
```

Unsorted array

```
this.color = [⚪ , ❤️ , ❤️ , ⚪ , 🔷 , 🍀 , 🍀 , 🔷 ]
```

Creating an empty array to store grouped colors by order of sorting

```
let sortedColors = []
```

```
for (num = 0; sortedColors.length != this.length; num++) {
    let groupedColors = []
    this.colors.forEach(color => {
        if (color.value == num) {
            groupedColors.push(color)
        }
    })
```

```
sortedColors.length = 0
this.length = 8
```

num = 0

```
groupedColors = []
this.colors = [⚪ , ❤️ , ❤️ , ⚪ , 🔷 , 🍀 , 🍀 , 🔷 ]
```

```
Searching for value == num

index 0 - [] || NO NEW ADDITION
index 1 - [❤️ ] || ADDED
index 2 - [❤️ , ❤️ ] || ADDED
index 3 - [❤️ , ❤️ ] || NO NEW ADDITION
index 4 - [❤️ , ❤️ ] || NO NEW ADDITION
index 5 - [❤️ , ❤️ ] || NO NEW ADDITION
index 6 - [❤️ , ❤️ ] || NO NEW ADDITION
index 7 - [❤️ , ❤️ ] || NO NEW ADDITION

groupedColors = [❤️ , ❤️ ]
```

Adding grouped color to sortedColors

```
sortedColors.push(...groupedColors)
```

```
# Output
sortedColors = [❤️ , ❤️ ]
```

```
for (num = 0; sortedColors.length != this.length; num++) {
    let groupedColors = []
    this.colors.forEach(color => {
        if (color.value == num) {
            groupedColors.push(color)
        }
    })
```

```
sortedColors.length = 2
this.length = 8
```

```
num = 1
groupedColors = []
this.colors = [⚪ , ❤️ , ❤️ , ⚪ , 🔷 , 🍀 , 🍀 , 🔷 ]
```

```
Searching for value == num

index 0 - [⚪ ] || ADDED
index 1 - [⚪ ] || NO NEW ADDITION
index 2 - [⚪ ] || NO NEW ADDITION
index 3 - [⚪ , ⚪ ] || ADDED
index 4 - [⚪ , ⚪ ] || NO NEW ADDITION
index 5 - [⚪ , ⚪ ] || NO NEW ADDITION
index 6 - [⚪ , ⚪ ] || NO NEW ADDITION
index 7 - [⚪ , ⚪ ] || NO NEW ADDITION

groupedColors = [⚪ , ⚪ ]
```

Adding grouped color to sortedColors

```
sortedColors.push(...groupedColors)
```

```
# Output
sortedColors = [❤️ , ❤️ , ⚪ , ⚪ ]
```

```
for (num = 0; sortedColors.length != this.length; num++) {
    let groupedColors = []
    this.colors.forEach(color => {
        if (color.value == num) {
            groupedColors.push(color)
        }
    })
```

```
sortedColors.length = 4
this.length = 8
```

```
num = 2
groupedColors = []
this.colors = [⚪ , ❤️ , ❤️ , ⚪ , 🔷 , 🍀 , 🍀 , 🔷 ]
```

```
Searching for value == num

index 0 - [] || NO NEW ADDITION
index 1 - [] || NO NEW ADDITION
index 2 - [] || NO NEW ADDITION
index 3 - [] || NO NEW ADDITION
index 4 - [🔷 ] || ADDED
index 5 - [🔷 ] || NO NEW ADDITION
index 6 - [🔷 ] || NO NEW ADDITION
index 7 - [🔷 , 🔷 ] || ADDED

groupedColors = [🔷 , 🔷 ]
```

Adding grouped color to sortedColors

```
sortedColors.push(...groupedColors)
```

```
# Output
sortedColors = [❤️ , ❤️ , ⚪ , ⚪ , 🔷 , 🔷 ]
```

```
for (num = 0; sortedColors.length != this.length; num++) {
    let groupedColors = []
    this.colors.forEach(color => {
        if (color.value == num) {
            groupedColors.push(color)
        }
    })
```

```
sortedColors.length = 6
this.length = 8
```

```
num = 2
groupedColors = []
this.colors = [⚪ , ❤️ , ❤️ , ⚪ , 🔷 , 🍀 , 🍀 , 🔷 ]
```

```
Searching for value == num

index 0 - [] || NO NEW ADDITION
index 1 - [] || NO NEW ADDITION
index 2 - [] || NO NEW ADDITION
index 3 - [] || NO NEW ADDITION
index 4 - [] || NO NEW ADDITION
index 5 - [🍀 ] || ADDED
index 6 - [🍀 , 🍀 ] || ADDED
index 7 - [🍀 , 🍀 ] || NO NEW ADDITION

groupedColors = [🍀 , 🍀 ]
```

Adding grouped color to sortedColors

```
sortedColors.push(...groupedColors)
```

```
# Output
sortedColors = [❤️ , ❤️ , ⚪ , ⚪ , 🔷 , 🔷 , 🍀 , 🍀 ]
```

```
for (num = 0; sortedColors.length != this.length; num++) {
    let groupedColors = []
    this.colors.forEach(color => {
        if (color.value == num) {
            groupedColors.push(color)
        }
    })
```

```
sortedColors.length = 8
this.length = 8
```

Condition Satisfied!

Returns the list of sorted colors

```
        return sortedColors
```

Final Output in the Terminal

```
*Enter number of colors: 8
Unsorted: [⚪ , ❤️ , ❤️ , ⚪ , 🔷 , 🍀 , 🍀 , 🔷 ]

Sorted: [❤️ , ❤️ , ⚪ , ⚪ , 🔷 , 🔷 , 🍀 , 🍀 ]
```

Worst Case Scenario

Time Complexity: O(n^2)

m = range of color values

- number of different colors to sort
- O(n)

n = length of `this.colors`

- number of colors of a given list
- O(n)

m * n is O(n2) if m is O(n).