GROUPBY IN JAVASCRIPT

Balaji Venkatraman
@balaji-venkatraman

Object.groupBy

Object.groupBy is a static method that runs through all the elements in the iterable(ex. arrays) and groups it based on the property/key returned by the provided callback function.

Object.groupBy function returns a **null-prototype object** which contains separate properties/keys for each associated group, containing the associated elements of each group(in array format).

Syntax

JS

```
Object.groupBy(iterable, callbackFn);
```

Balaji Venkatraman
@balaji-venkatraman

Example

JS

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
const output = Object.groupBy(numbers, (num) =>
      num % 2 === 0 ? 'Even' : 'Odd'
);
console.log(output);
```

```
{
  "Odd": [1, 3, 5, 7, 9],
  "Even": [2, 4, 6, 8, 10]
}
```

Balaji Venkatraman
@balaji-venkatraman

Example

JS

```
const webTech = [
  { name: 'React', type: 'frontend' },
  { name: 'NodeJS', type: 'backend' },
];
const output = Object.groupBy(webTech, (tech) =>
tech.type);
console.log(output);
```

```
{
 "frontend": [{ "name": "React","type": "frontend" }],
 "backend": [{ "name": "NodeJS", "type": "backend" }]
}
```

NOTE

The elements in the returned object and the original iterable are the same (not deep copies). Changing the internal structure of the elements will be reflected in both the original iterable and the returned object.

Balaji Venkatraman
@balaji-venkatraman

Map.groupBy

Map.groupBy is a static method that runs through all the elements in the iterable(ex. arrays) and groups it based on the associated object returned by the provided callback function.

Map.groupBy function returns a **Map** object with keys for each group, each assigned to an array containing the elements of the associated group.

Syntax

JS

```js
Map.groupBy(iterable, callbackFn);
```

Balaji Venkatraman
@balaji-venkatraman

Map.groupBy

The callback function returns an object for each element indicating the associated group of the element. This returned value of the callback function is used as the key for the associated elements. Each key has an associated array containing all the elements for which the callback returned the same value. This method is useful when you need to group information that is related to a particular object that might potentially change over time.

NOTE

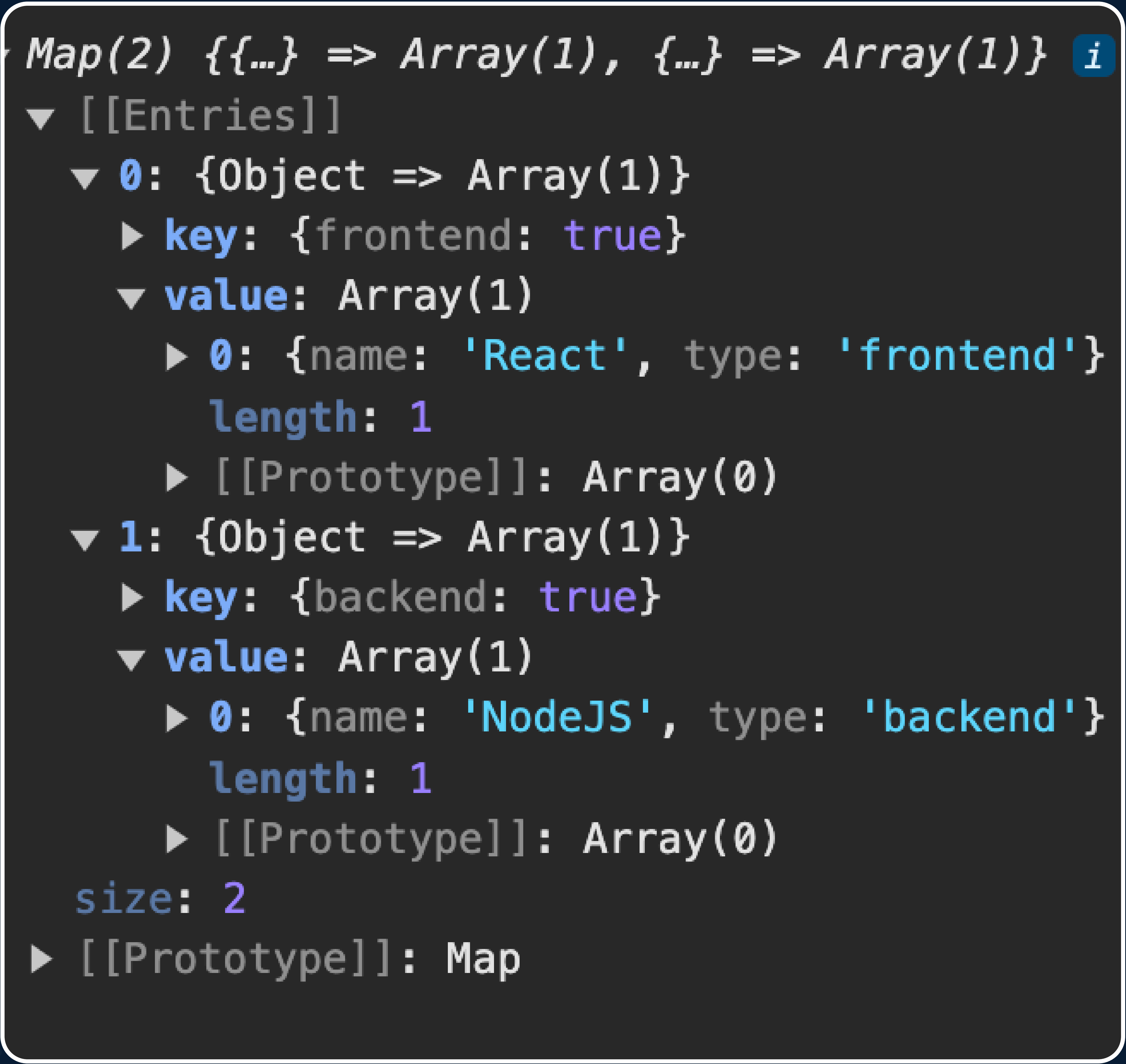
The elements in the returned Map and the original iterable are the same (not deep copies). Changing the internal structure of the elements will be reflected in both the original iterable and the returned Map.

Example

JS

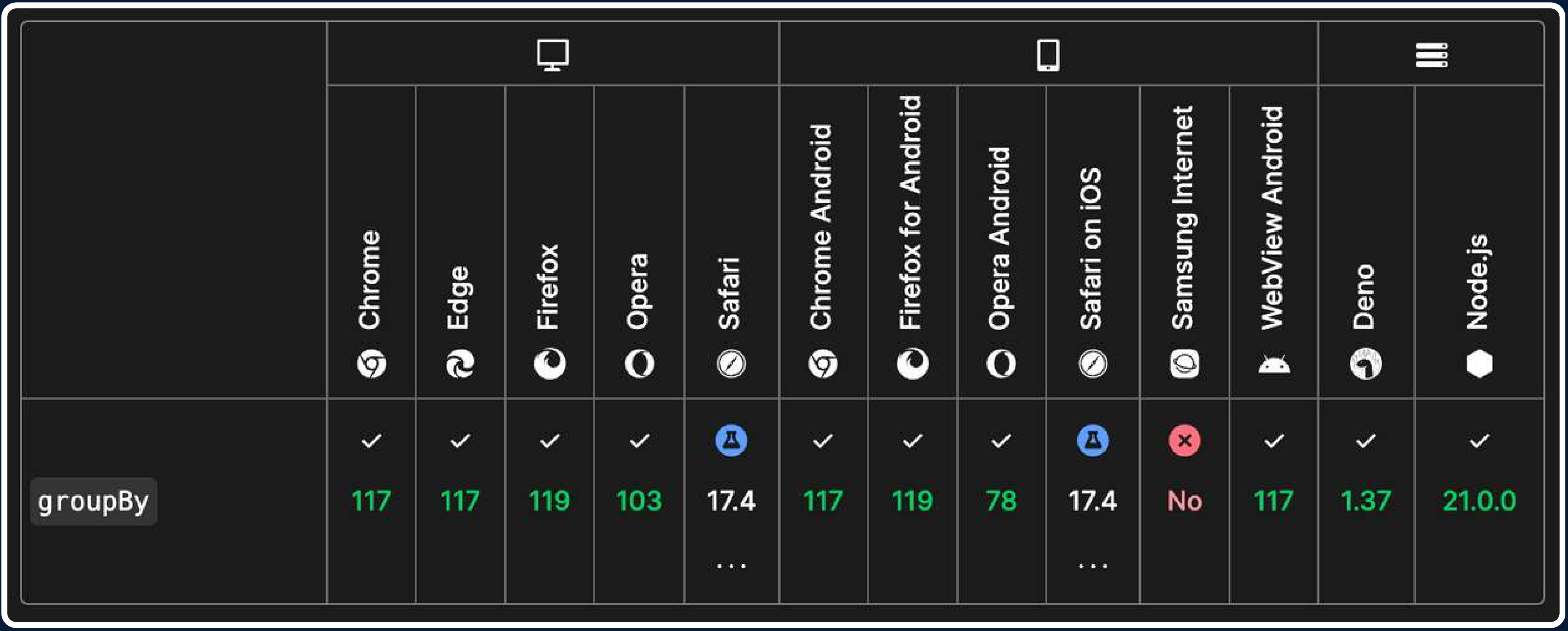
```javascript
const webTech = [
  { name: 'React', type: 'frontend' },
  { name: 'NodeJS', type: 'backend' },
];
const frontend = { frontend: true };
const backend = { backend: true };
const output = Map.groupBy(webTech, (tech) =>
 tech.type === 'frontend' ? frontend : backend
);
console.log(output);
```

Output



```
Map(2) {{…} => Array(1), {…} => Array(1)} i
▼ [[Entries]]
  ▼ 0: {Object => Array(1)}
    ▶ key: {frontend: true}
    ▼ value: Array(1)
      ▶ 0: {name: 'React', type: 'frontend'}
        length: 1
      ▶ [[Prototype]]: Array(0)
  ▼ 1: {Object => Array(1)}
    ▶ key: {backend: true}
    ▼ value: Array(1)
      ▶ 0: {name: 'NodeJS', type: 'backend'}
        length: 1
      ▶ [[Prototype]]: Array(0)
  size: 2
▶ [[Prototype]]: Map
```

Browser Compatability



	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android	Deno	Node.js
groupBy	✓ 117	✓ 117	✓ 119	✓ 103	17.4 …	✓ 117	✓ 119	✓ 78	17.4 …	✗ No	✓ 117	✓ 1.37	✓ 21.0.0

Since **groupBy()** method is still not supported by all browsers yet we can wrote polyfils to support the feature

Balaji Venkatraman
@balaji-venkatraman

Object.groupBy Polyfill

JS

```
class Object {
  static groupBy(items, callbackfn) {
     const grouped = {};
     for (let i = 0; i < items.length; i++) {
        const item = items[i];
        const key = callbackfn(item, i);
        if (!grouped[key]) {
           grouped[key] = [];
        }
        grouped[key].push(item);
     }
    return grouped;
  }
}
```

Map.groupBy Polyfill

JS

```
class Map {
  static groupBy(items, callbackfn) {
    const grouped = new Map();
    for (let i = 0; i < items.length; i++) {
      const item = items[i];
       const key = callbackfn(item, i);
       if (!grouped.has(key)) {
          grouped.set(key, []);
       }
       grouped.get(key).push(item);
     }
     return grouped;
  }
}
```

Balaji Venkatraman S
@balaji-venkatraman

Follow for more such posts

Save

Like

CODINOUSAR
TECH

https://www.codinousar.com/

COMMENT BELOW