

Problem 2631: Group By

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Write code that enhances all arrays such that you can call the

`array.groupBy(fn)`

method on any array and it will return a

grouped

version of the array.

A

grouped

array is an object where each key is the output of

`fn(arr[i])`

and each value is an array containing all items in the original array which generate that key.

The provided callback

`fn`

will accept an item in the array and return a string key.

The order of each value list should be the order the items appear in the array. Any order of keys is acceptable.

Please solve it without lodash's

`_.groupBy`

function.

Example 1:

Input:

```
array = [ { "id": "1" }, { "id": "1" }, { "id": "2" } ], fn = function (item) { return item.id; }
```

Output:

```
{ "1": [{ "id": "1" }, { "id": "1" } ], "2": [{ "id": "2" } ] }
```

Explanation:

Output is from `array.groupBy(fn)`. The selector function gets the "id" out of each item in the array. There are two objects with an "id" of 1. Both of those objects are put in the first array. There is one object with an "id" of 2. That object is put in the second array.

Example 2:

Input:

```
array = [ [1, 2, 3], [1, 3, 5], [1, 5, 9] ] fn = function (list) { return String(list[0]); }
```

Output:

```
{ "1": [[1, 2, 3], [1, 3, 5], [1, 5, 9]] }
```

Explanation:

The array can be of any type. In this case, the selector function defines the key as being the first element in the array. All the arrays have 1 as their first element so they are grouped

```
together. { "1": [[1, 2, 3], [1, 3, 5], [1, 5, 9]] }
```

Example 3:

Input:

```
array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] fn = function (n) { return String(n > 5); }
```

Output:

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

Explanation:

The selector function splits the array by whether each number is greater than 5.

Constraints:

$0 \leq \text{array.length} \leq 10$

5

fn

returns a string

Code Snippets

JavaScript:

```
/**
 * @param {Function} fn
 * @return {Object}
 */
Array.prototype.groupBy = function(fn) {

};

/**
```

```
* [1,2,3].groupBy(String) // {"1":[1],"2":[2],"3":[3]}
*/
```

TypeScript:

```
interface Array<T> {
  groupBy(fn: (item: T) => string): Record<string, T[]>
}

Array.prototype.groupBy = function(fn) {

}

/**
 * [1,2,3].groupBy(String) // {"1":[1],"2":[2],"3":[3]}
 */
```

Solutions

JavaScript Solution:

```
/**
 * Problem: Group By
 * Difficulty: Medium
 * Tags: array, string
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * @param {Function} fn
 * @return {Object}
 */
Array.prototype.groupBy = function(fn) {

};
```

```
/**
 * [1,2,3].groupBy(String) // {"1":[1],"2":[2],"3":[3]}
 */
```

TypeScript Solution:

```
/**
 * Problem: Group By
 * Difficulty: Medium
 * Tags: array, string
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

interface Array<T> {
  groupBy(fn: (item: T) => string): Record<string, T[]>
}

Array.prototype.groupBy = function(fn) {

}

/**
 * [1,2,3].groupBy(String) // {"1":[1],"2":[2],"3":[3]}
 */
```