JavaScript Course Schedule

Challenge

There is a total of `numCourses` courses you must take, labeled from `0` to `numCourses - 1`. You are given an array `prerequisites` where `prerequisites[i] = [$a_i$, $b_i$]` indicates that you must take course `$b_i$` first if you want to take course `$a_i$`.

For example, the pair `[0, 1]`, indicates that to take course `0` you must first take course `1`.

Return `true` if you can finish all courses. Otherwise, return `false`.

1st Example

```
Input: numCourses = 2, prerequisites = [[1,0]]
Output: true
Explanation: There are a total of 2 courses to take. To take
             course 1 you should have finished course 0. So,
             it is possible.
```

2nd Example

```
Input: numCourses = 2, prerequisites = [[1,0],[0,1]]
Output: false
Explanation: There are a total of 2 courses to take. To take
             course 1 you should have finished course 0, and
             to take course 0 you should also have finished
             course 1. So, it is impossible.
```

Constraints

- `1 <= numCourses <= 2000`
- `0 <= prerequisites.length <= 5000`
- `prerequisites[i].length == 2`
- `0 <= a_i, b_i < numCourses`
- All the pairs `prerequisites[i]` are unique.

Solution

```
const canFinish = (numCourses, prerequisites) => {
    const preMap  = {},
          visited = {};

    for (let i = 0; i < prerequisites.length; i++) {
        if (preMap[prerequisites[i][0]] === undefined) {
            preMap[prerequisites[i][0]] =
                  [prerequisites[i][1]];
        } else {
            preMap[prerequisites[i][0]]
            .push(prerequisites[i][1]);
        }
    }

    console.log(preMap);

    const dfs = (node) => {
        if (visited[node]){
            return false;
        }
```

Solution continues on next page...

```
        if (preMap[node] !==undefined) {
            if (preMap[node].length === 0) {
                return true;
            }

            visited[node] = true;

            for (let val of preMap[node]) {
                if (!dfs(val)) {
                    return false;
                }
            }

            visited[node] = false;

            preMap[node] = [];
        }

        return true;
    }

    for (let key in preMap) {
        if (!dfs(key)) {
            return false;
        }
    }

    return true;
};
```

Explanation

I've coded a function that checks if it is possible to finish all the given courses by considering the prerequisites. It returns `true` if it is possible to complete all courses, and `false` otherwise.

The function starts by initializing two objects: `preMap` and `visited`.

Next, it enters a loop that iterates through the `prerequisites` array. Within this loop, the function fills the `preMap` object with the prerequisites for each course. If a course does not exist in `preMap`, a new key-value pair is created with the course as the key and an array containing the prerequisite as the value. If the course already exists in `preMap`, the prerequisite is appended to the existing array.

The function then defines a recursive function called `dfs` (depth-first search). This function takes a node as an argument and checks if the node has already been visited. If the node has been visited, the function returns `false`. If the node does not have any prerequisites (ex. it has an empty array in `preMap`), the function returns `true`. If the node has prerequisites, the function recursively calls `dfs` for each prerequisite. If any of the prerequisites cannot be completed (ex. `dfs` returns `false`), the function also returns `false`. After checking all the prerequisites, the node is marked as unvisited and its array of prerequisites in `preMap` is emptied.

After defining the `dfs` function, the function enters another loop that iterates through each key in the `preMap` object. For each key (course), the function calls the `dfs` function. If the `dfs` function returns false for any course, indicating that it is not possible to complete the course, it returns `false`.

If the function completes the loop without returning `false`, it means that all courses can be completed. In this case, it returns `true`.

In summary, the function checks the prerequisites for each course

and determines if it is possible to complete all the courses. It uses a depth-first search approach to explore the prerequisites and returns `true` if all courses can be completed, and `false` otherwise.

Author: Trevor Morin