Kata 6 - SmartParking

Assignment

50 minutes

Status | Complete

Welcome back to Codeville! The citizens of Codeville seem pleased with all the upgrades your administration has been making to the local infrastructure, and they want more! The parking lot in the Codeville Devtropolis Shopping Mall needs an upgrade, and you've decided this is the perfect opportunity to install a smart parking system.

The system will use special parking sensors to keep track of all parking spots and monitor which ones are available. Every time a vehicle enters the parking lot, the system directs them to an available spot for their particular vehicle type, or notifies them that no spots are available.

> We need to write a function called `whereCanIPark()` that returns the coordinates of an available parking spot for the vehicle, or returns false if there is no available spot. Our function receives an array of arrays representing parking spots, and a string with type of the vehicle that is looking for a parking spot.

There are three kinds of possible vehicles: **regular** cars, **small** cars, and **motorcycles**.

- Regular cars can only park in **R** spots.
- Small cars can park in **R** or **S** spots.
- Motorcycles can park in **R**, **S**, or **M** spots.

In the array of parking spots, spots are written in both lower-case and upper-case. An upper-case letter means that the particular spot is **AVAILABLE**, while lower-case letters mean that the spot is **UNAVAILABLE**.

Our function must return an array with the coordinates of the spot as an [X, Y] pair. See the example input and output below for an illustration.

> **Note:** There may be multiple available spots for a particular vehicle. It does not matter which spot your function chooses, as long as the spot is available. And if there are no available spots, remember to return false.

Input

```
const whereCanIPark = function (spots, vehicle) {
  // Code here!
};

console.log(whereCanIPark(
  [
    // COLUMNS ARE X
    // 0    1    2    3    4    5
    ['s', 's', 's', 'S', 'R', 'M'], // 0 ROWS ARE Y
    ['s', 'M', 's', 'S', 'r', 'M'], // 1
    ['s', 'M', 's', 'S', 'r', 'm'], // 2
    ['S', 'r', 's', 'm', 'r', 'M'], // 3
    ['S', 'r', 's', 'm', 'r', 'M'], // 4
    ['S', 'r', 'S', 'M', 'M', 'S']  // 5
  ],
  'regular'
));

console.log(whereCanIPark(
  [
    ['M', 'M', 'M', 'M'],
    ['M', 's', 'M', 'M'],
    ['M', 'M', 'M', 'M'],
    ['M', 'M', 'r', 'M']
  ],
  'small'
));

console.log(whereCanIPark(
  [
    ['s', 's', 's', 's', 's', 's'],
    ['s', 'm', 's', 'S', 'r', 's'],
    ['s', 'm', 's', 'S', 'r', 's'],
    ['S', 'r', 's', 'm', 'r', 's'],
    ['S', 'r', 's', 'm', 'R', 's'],
    ['S', 'r', 'S', 'M', 'm', 'S']
  ],
  'motorcycle'
))
```

Expected Output

```
[4, 0]
false
[3, 1]
```

To go through a single array, we can use a single for-loop. To go through every element in an array of arrays, we can use two for-loops: one nested within the other. **See two of the Lighthouse Labs team pair program this challenge below:**

👉 Submit Your Work

- Browse to gist.github.com and create a new gist.
- Copy-and-paste your code into the form
- Name the gist and the file appropriately and click `Create secret gist`.
- Finally, mark this activity as completed (at the bottom of this page) and please copy/paste the *entire* browser URL for your gist (from *gist.github.com*) into the text field.

---

https://gist.github.com/sa ✖ Cancel Submission

← Previous
Kata 5 - Percent Encoded String

Next
Kata 7 - In the Air Tonight →

How well did you understand this content?

Thank you for your feedback

★ ★ ★ ★ ★

Totally got it!

Please give us some written insight into your feedback

Prep Work

› 1: Welcome

› 2: Dev Environment

› 3: Version Control

› 4: Programming Intro

› 5: The Browser

6: Katas

6 hrs *+ 29 hrs stretch* 🏆

- Katas ✓
- </> Kata 1 - Sum the Largest Numbers ✓
- </> Kata 2 - Conditional sums ✓
- </> Kata 3 - Vowels ✓
- </> Kata 4 - Instructors Names ✓
- </> Kata 5 - Percent Encoded String ✓
- </> Kata 6 - SmartParking ✓
- </> Kata 7 - In the Air Tonight ✓
- </> *Kata 8 - Repeating Numbers* ✓
- </> *Kata 9 - Case Maker* ✓
- </> *Kata 10 - Multiplication Table* ✓
- </> *Kata 11 - Bouncy Castles* ✓
- </> *Kata 12 - The Great Codeville Bake-off* ✓
- </> *Kata 13 - Talking Calendar* ✓
- </> *Kata 14 - Change Calculator* ✓
- </> *Kata 15 - Organizing Instructors* ✓
- </> *Kata 16 - Case Maker II* ✓
- </> *Kata 17 - JS Object From URL Encoded String* ✓
- </> *Kata 18 - Square Code* ✓
- </> *Kata 19 - Queen Threat Detector* ✓
- </> *Kata 20 - Taxicab Geometry* ✓
- </> *Kata 21 - Number Guesser* ✓

7: Stretch Project

8: The Lab Manual

9: Day One Prep

10: Collab Tools Setup

POWERED BY

LIGHTHOUSE LABS