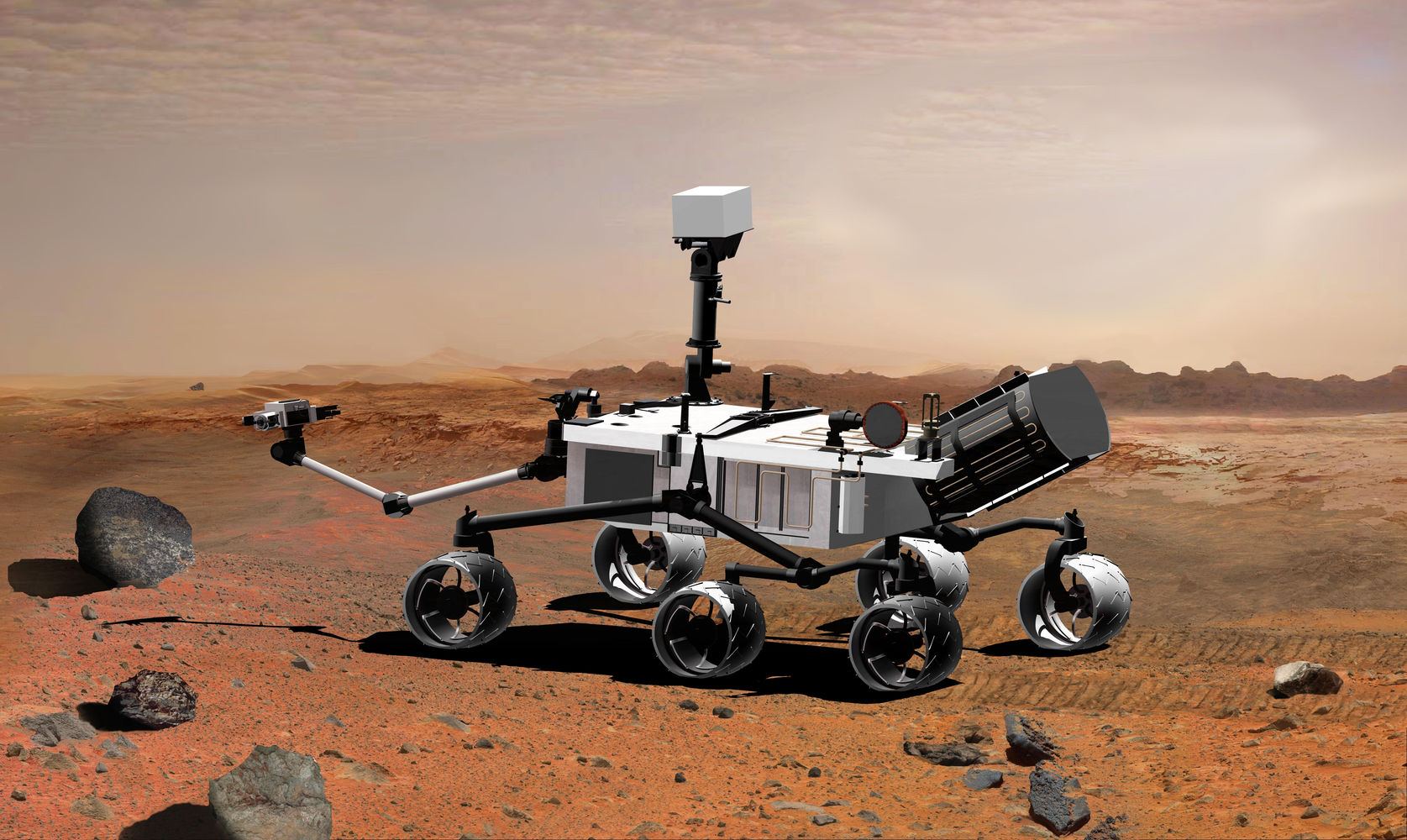
**Mars Rover Kata**

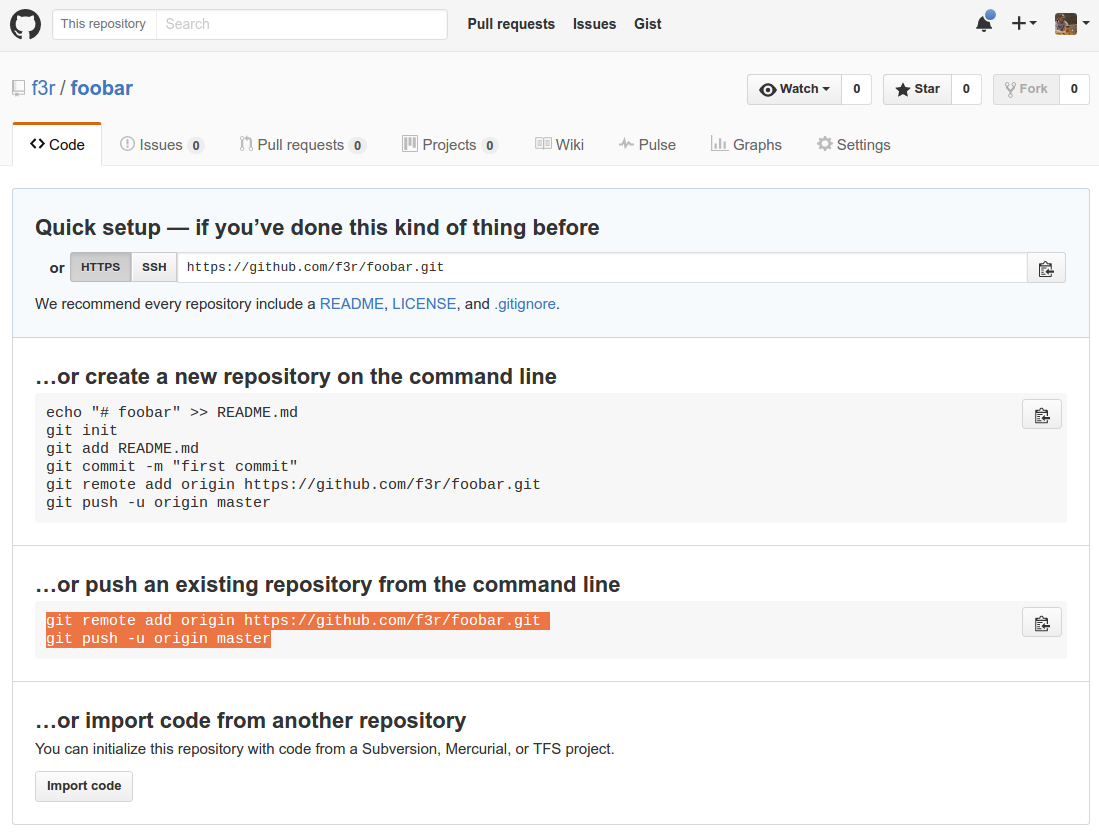
**Introduction**

We are sending a rover to Mars and we need to program its movements so that we can send it commands from Earth.



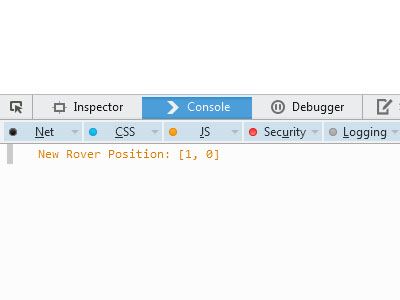
Requirements

* [Download these Mars Rover files](https://s3-eu-west-1.amazonaws.com/ih-platform/documents/mars_rover.zip) to ~/prework/code/labs.
* Open your terminal and run the following commands:
* cd ~/prework/code/labs
* unzip ih-prework-html-css.zip
* cd ih-prework-html-css
* git init
* git add .
* git commit -m"Just setting up my Github"
* [Create a new repo](https://github.com/new) on your Github account, and copy-paste the HTTPS URL (i.e. <https://github.com/>/ih-prework-html-css.git)



* Once you are done, run the following commands:
* git add remote origin https:*//github.com/<your-user-name>/ih-prework-html-css.git*
* git push origin master

Make sure your files have been uploaded into your Github repo before you start.

* Open the **index.html** file and then open your console. You should be able to see the results of any console.log() functions here. 
* Make sure you use data structures (basic data types, arrays & objects)
* Make sure you use boolean and conditionals
* Make sure you use loops to avoid code repetition
* Make sure you use functions and organize your code

Submission

* Upon completion run the following commands:
* git init
* git add .
* git **commit** -m"done"
* git push origin master
* Make sure that your Github repo is updated with your last changes
* Send your Github repo link to your TA's through Slack/Email.

**Basic Information**

* The rover will have initial starting point (x,y) coordinates (i.e. 0,0)
* The rover will have an initial direction (N,E,S,W) to where it is facing
* The rover is on a 10 x 10 grid
* Implement commands that go forward and backward (f,b)
* Implement commands to turn the rover left or right (l,r). The rover just change the direction it is facing when the user sets this command. It won't move right or left automatically. To make it change its position, the user needs to specify the change of direction and then the actual movement.

Iteration 1

1) Create an object to represent the rover that has position and direction attributes 2) Create a grid using arrays (hint: do a google search for two-dimensional arrays). 3) Write functions for the various commands 4) Try to call some of those functions and display the new position of the rover.

For example, you could start with this:

**var** Rover = {

position: [0,0],

direction: 'N'

}

**function** goForward(rover) {

**switch**(rover.direction) {

**case** 'N':

rover.position[0]++

**break**;

**case** 'E':

rover.position[1]++

**break**;

**case** 'S':

rover.position[0]--

**break**;

**case** 'W':

rover.position[1]--

**break**;

}

}

goForward(myRover);

**Iteration 2**

We want to be able to send a series of commands to the rover and execute them in sequence.

We will send an array of character commands, which can include:

* f for go forward
* b for go back
* r for turn right\*
* l for turn left\*

\*The rover just change its direction when we use this command. It won't move right or left automatically. To make it go right or left, the user needs to specify the change of direction and then the actual movement.

So we should be able to tell the rover, for example, ‘fffrfflfffbb’ and it would execute those movements and provide us with its new position.

Furthermore, we need to make sure that the rover never goes off the grid but rather, wraps from one edge of the grid to another (planets are spheres after all)

**Bonus**

* In your 10 x 10 grid, place some obstacles that the rover cannot cross or land on
* Implement obstacle detection. The rover should execute the given commands until it reaches an obstacle, then stop at the last possible position and report the obstacle.

**Extra hardcore super bonus**

Our rover is lonely! We need to send another one that can roam on the same grid and execute the same commands. Make sure the rovers don’t bump into each other.