****

**JavaScript testing assignment**

**Before you start you can take a look at JavaScript screencasts & useful links:**

### [JavaScript lecture 1](https://wiki.itechart-group.com/screencasts/trainings/Front-End%20Development%20Training/JavaScript/03-12-2013/JavaScript%20Course%20-%20Lecture%201.mp4)

### [JavaScript lecture 2](https://wiki.itechart-group.com/screencasts/trainings/Front-End%20Development%20Training/JavaScript/03-14-2013/JavaScript%20Course%20-%20Lecture%202.mp4)

### [JavaScript lecture 3](https://wiki.itechart-group.com/screencasts/trainings/Front-End%20Development%20Training/JavaScript/03-19-2013/JavaScript%20Course%20-%20Lecture%203.mp4)

### [JavaScript lecture 4](https://wiki.itechart-group.com/screencasts/trainings/Front-End%20Development%20Training/JavaScript/03-21-2013/JavaScript%20Course%20-%20Lecture%204.mp4)

### [JavaScript lecture 5](https://wiki.itechart-group.com/screencasts/trainings/Front-End%20Development%20Training/JavaScript/03-26-2013/JavaScript%20Course%20-%20Lecture%205.mp4)

### [JavaScript lecture 6](https://wiki.itechart-group.com/screencasts/trainings/Front-End%20Development%20Training/JavaScript/03-28-2013/JavaScript%20Course%20-%20Lecture%206.mp4)

### [JavaScript lecture 7](https://wiki.itechart-group.com/screencasts/trainings/Front-End%20Development%20Training/JavaScript/04-02-2013/JavaScript%20Course%20-%20Lecture%207.mp4)

### [Современный учебник Javascript](https://learn.javascript.ru/)

### [JavaScript Guide MDN](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide)

### [Introduction to JavaScript by freeCodeCamp](https://learn.freecodecamp.org/javascript-algorithms-and-data-structures/basic-javascript)

**When you are ready to implement:**

1) Create new account at [Github](https://github.com/) or [Bitbucket](https://bitbucket.org/) ( if you don't have it yet)

2) Create new repository and then create 3 branches in this repository for each JavaScript task.

3) Push your code for each task into related branch.

4) When you finish all tasks - ping your mentor via slack/skype/email and share repository link with him.

**1) Utils task**

**Search**

Implement a function that sort array of numbers and return sorted array.

**Element search**

Implement a function that returns the first element in array that satisfies given condition.

**Filter**

Implement a function that filters array based on callback result.

**Map**

Implement a function that creates new array with the results of calling a provided function on every element in this array.

**Average**

Given array of numbers, find average of even.

**2) Robot task**

**Robot toy**

Implement a robot toy class with move(), left(), right(), report() functions. Robot has 0, 0 as initial coordinates and sees on NORTH.

Method move() should change coordinate by one, left() and right() will rotate the robot 90 degrees in the specified direction without changing the position of the robot.

Coordinates can't be negative - ignore any move that change it on negative. report() should print robot coordinates and direction.

After calling next code:  
robot.move();   
robot.move();   
robot.move();   
robot.right();   
robot.move();   
robot.move();   
robot.report();  
your report() method should print result message: Coordinates: 2,3. Direction: EAST.

**3) Logging library (read some notes for this task below)**

**Design and implement client-side Logging Library in adherence to following user stories:**

— I want to have an ability to log information

— I want to have an ability to send log information to the console.

— I want to have an ability to send log information to an alert window.

— I want to have an ability to send log information to the current window.

— I want to have an ability to send log information to abstract Web API endpoint.

— I want to have an ability to select from built-in logging methods.

— I want to be able to extend the library with custom logging methods.

— I want to library to automatically log all unhandled client-side exceptions.

— I want to have an ability to extend library with my own instrumentation methods so that the library can monitor not only exceptions, but other events too.

— As a developer extending the library with my own logging or instrumentation methods I want to have established contracts in a form of base class of API documentation so that I can easily know how to extend the library.

— As a developer extending the library I want to see clear errors and warning in case if I violate the contract of logging or instrumentation methods.

**Significant implementation details & design goals:**

a) Architecture of the library is what matters most. Following SOLID principles is crucial to successfully manage dependencies in your code.

b) Extensibility matters.

c) Use your existing knowledge to design clean API for your library. But keep in mind that API design is always a tradeoff: API that good for everybody, good for no one.

d) Don’t take word “method” literally when reading user stories. In your implementation “method” can be mapped to construct of JavaScript language or OOP concept.