Hi there!

Thanks a lot for taking out the time to do our challenge.
This file will provide the instructions for solving the challenge.
##

Before that, we would like to state:
1. Feel free to use any libraries, frameworks or tools of your choice.
You can use any necessary tools that you are comfortable with.

2. We really appreciate a well commented code. We would like to understand your thought-process.
So please provide apt comments in your code to convey that.

3. If you have any doubts/questions/feedback, please feel free to approach us.

4. Do not worry about IE. Your code needs to run on latest Chrome & Firefox.

5. You should compress your solution-directory into a zip-file and email it to us.
Please provide clear instructions on how to run your solution e.g. a python-server or a Nodejs static file-server (if applicable).

6. Please double check your solution before submitting it to us.
Make sure it works properly.

7. Bonus points for making the solution responsive and adaptive to smaller devices.

8. Ideally it should not take you more than 4 hours to finish this challenge.
However, take as much time as you need. There is no time-restriction from our side.
##

Challenge:
At ixigo, one of the key responsibility is displaying real-time data collected from multiple sources.
This challenge simulates a similar situation (with reduced complexity).
We would like you to build a flight-results page, for sample flights between Delhi & Mumbai.

In this directory, you will find a file called "data.js".
This file holds the json-data for different flights between Delhi & Mumbai.

1. The "flightsData" array holds multiple-objects, where each object contains the information about a specific flight.
For example :
```
    {
        originCode : "DEL", //the origin airport code. You can get the corresponding name from the "airportMap" object.
        destinationCode : "MUM", //the destination airport code. You can get the corresponding name from the "airportMap" object.
        takeoffTime : "1388820600000", //the take-off time for the flight (measured in milliseconds since January 1, 1970, 00:00:00 UTC).
        landingTime : "1388831400000", //the landing time for the flight (measured in milliseconds since January 1, 1970, 00:00:00 UTC).
        price : "5600", //the cost of the flight in INR.
        airlineCode : "G8", //the airline-code. You can get the corresponding name from the "airlineMap" object.
        class : "Economy" // flight-class (economy vs business).
    }
```

2. The "airlineMap" object maps the different airline-codes with their corresponding names.

3. The "airportMap" object maps the different airport-codes with their corresponding names.

Using this data, create an interface that displays these flights, along with any other relevant information.
The interface needs to:

a. Display every flight clearly with relevant information like price, class, take-off/landing times, airport-names, flight-duration etc..
b. Support simple sorting options to sort the flights based on their price, take-off & landing times and filters to filter flight results based on airline code & class.

You need not build a pagination-scheme. However if you feel it compliments your solution, go ahead.

##
You can take a look at www.ixigo.com/flights to get an inspiration.
But please do not be restricted by that. Be creative about your solution.
Try making it work on smaller devices as well.