# Trip sorter

*Maximum amount of time allowed: 2 hours*

## Task

You are given a stack of boarding cards for various transportations that will take you from a point A to point B via several stops on the way. All of the boarding cards are out of order and you don't know where your journey starts, nor where it ends. Each boarding card contains information about seat assignment, and means of transportation (such as flight number, bus number etc).

Provide an API that let's you sort this kind of list and present back a description of how to complete your journey. For instance the API should be able to take an unordered set of boarding cards, provided in a format defined by you, and produce this list:

- Take train 78A from Madrid to Barcelona. Sit in seat 45B.
- Take the airport bus from Barcelona to Gerona Airport. No seat assignment.
- From Gerona Airport, take flight SK455 to Stockholm. Gate 45B, seat 3A. Baggage drop at ticket counter 344.
- From Stockholm, take flight SK22 to New York JFK. Gate 22, seat 7B. Baggage will we automatically transferred from your last leg.
- You have arrived at your final destination.

The list should be defined in a format that's compatible with the input format.

The API is to be a client JavaScript API, so it will only communicate with other parts of the code in the browser without the need of additional requests between the browser and the server. Use PHP/Java-doc to document the input and output your API accepts / returns.

## Requirements

- Use object oriented Javascript for the implementation.
- Do not use any 3rd party framework or library. Start all code from scratch.
- The structure of the code should be extendable to make building in support for any means of transportation / extra information required about a specific type of transportation easy.
- The implementation of your sorting algorithm should work with any set of boarding passes, as long as there is always an unbroken chain between all the legs of the trip. Ie. it's one continuos trip with no interruptions.
- The algorithm doesn't need to consider that departure / arrival are in the correct order. In fact there is no information about any such times on the boarding passes. It is just assumed that your next connection is waiting for you when you arrive at a destination.
- The algorithm should have the lowest possible order of complexity (Big O notation) you

could think of.

- Although we always produce all software in teams in Tuenti, and believe this working as a team is the best way to tackle problems, since this task is meant to be an assessment of your personal skills, please refrain from asking friends / cooperating with other people while solving this task.

# What we look at

This task is designed to give us an idea of how you think when faced with a very limited amount of time to solve a task of significant complexity. We are also interested in how you structure your code so that it's easily extendable, complies with best OO practices, and easy to modify / understand by others. We are also interested in seeing how efficient the sorting algorithm you implement is.