API directory:

* utils
  + [autocomplete.js](#y426rvlwndsb)
  + [network.js](#90bsppdddq69)
  + [profile.js](#rtellpw0c0ce)
  + [search.js](#jjqr5n0juaz)
  + [user.js](#msvno1fkg2ss)
  + [utils.js](#xg7lf6r43kni)

# autocomplete.js

Methods:

getDict() gets the dictionary (in text format)

autocomplete(dict, s) gets all possible words start with s in the dictionary dict

**getDict()**

parameters:

No parameters.

return:

dict array the dictionary with each entry being a word

description:

gets the dictionary (in text format)

**autocomplete(dict, s)**

parameters:

dict array the dictionary

s string the word that is entered

return:

No returns.

description:

gets all possible words start with s in the dictionary dict

# network.js

Methods:

group(users, id, size) forms a group of {size} for user {id}

**group(users, id, size)**

parameters:

users Map contains all information with the id as the key

id int id of the user

size int size of group

return:

list array the id of the students in the group

description:

forms a group of {size} for user {id}

# profile.js

Methods:

score(userA, userB, pref, filter) calculate the score of user B for user A

sharedAvailability(timeA, timeB) find the shared available time slots

dayCnt(time) count the length of time slots in a day

**score(userA, userB, pref, filter)**

parameters:

userA User the current user

userB User the target user

pref Pref user A's preference

filter Filter the search filter

return:

score int the score

description:

calculate the score of user B for user A

the score depends on availability and interests

a higher score indicates a more likely chance of working together

a score of 0 means that user B should not work with user A

**sharedAvailability(timeA, timeB)**

parameters:

timeA array:string avaiable time for user A

timeB array:string avaiable time for user B

return:

availability array:string(7) the share available time

description:

find the shared available time slots

**dayCnt(time)**

parameters:

time string the time slots

return:

length int the length of time

description:

count the length of time slots in a day

if a time segment is less than 30 minutes,

it will not be counted towards the total length of time

this is because a small segment of time cannot be enough for a meeting

# search.js

Methods:

search(users, id, pref, filter) gets a list of suggested users for user {id}

**search(users, id, pref, filter)**

parameters:

users Map contains all information with the id as the key

id int user id

filter Filter the search filter

return:

list array a sorted list of users

description:

gets a list of suggested users for user {id}

# user.js

Methods:

sampleUsers() gets a sample set of users

**sampleUsers()**

parameters:

No parameters.

return:

users Map the users with all information

description:

gets a sample set of users

# utils.js

Methods:

sharedElements(arrA, arrB) gets all shared elements in two lists

**sharedElements(arrA, arrB)**

parameters:

arrA array list A

arrB array list B

return:

filteredArray array the list of shared elements

description:

gets all shared elements in two lists