# TT-datamining

index.html

* Styles (needs work, extract to separate file styles.css)
* Display input buttons
  + Student data: calls *parseStudentData*
  + Log file: calls *parseCSV*
* Load javascript programs
  + parse-student-data.js
  + html-support.js
  + parse-file.js
  + utils.js
  + html-utils.js
  + analysis-tools.js
  + report-tools.js
  + PapaParse-4.1.2/papaparse.js
  + jquery.min.js (from Google)

parse-student-data.js

* *parseStudentData*: called by input button to read user-selected Student-anon data file (using *Papa.parse*) and load into ***studentDataObjs*** array

parse-file.js

* *parseCSV*: called by input button to read user-selected log file (using *Papa.parse*) and then:
  + load into ***rowObjs*** array of row objects
  + call *makeTeams* (in *utils.js*) to create ***teams*** array
  + call *analyze* (in *analysis-tools.js*) to create ***changes*** array
  + call *setupForm* to output table of choices for user

html-support.js

* *toggleSelectAll*(checkboxName) // insert check in all boxes if SelectAll chosen
* *setupForm*(teams) // create table for user to select teams, levels, actions, varRefs, summary reports, teacher reports, including buttons: “Submit query” (*generateReport*), “Clear screen” (*clearScreen*), “Actions File download” (*downloadLogCSV*), “Summary File download” (*downloadSummaryCSV*), and “Find guess and check” (*findGuessAndCheck*)
* *clearReport*() // remove report from screen
* *setUpActionsReport*(teams) // Sets up a table with three columns into which selected actions will be listed, one column per team member
* *addLevelRow*(team, level) // creates a row for a new level of a team
* *counter*(checkboxName) // define the count and reset functions for checkboxes
* *showData*(act) // reveal pop-up window (*addActionRow*) with mouse-down on time
* *hideData*() // close pop-up window with mouse-up
* *addActionRow*(act, content) // create contents of pop-up window (see *showData*)

utils.js

* *clearScreen* () // clears data and summary tables
* *initializeVarRefs* (level) // Initializes variable references for this level
* *scoreAction* (action) // scoring actions based on relevance (inclusion of varRefs)
* *highlightMessage* (act, text) // Highlights variable names/values in a message
* *getVarRefs* (action, text) // returns variable references contained in message
* *findVars* (act, numStr) // finds variables by matching their known values
* *makeTeams* (rowObjs) // parse logdata row objects looking for and populating teams
* *getLevelNumber*(levelName) // coping with various names for the levels, boil it down to level number 2, 3, 4, or 5
* *about* (num, target, tol) // compare number with known value
* *addTeam* (ro) // forming teams with students who join a group or pick username
* *addLevel* (myTeam, ro) // construct a new level from ro and add it to levels array.
* *addMember* (myTeam, ro) // add a new member to the array from ro
* *getLevel* (ro) // get Level of this ro
* *addLevelValues* (myLevel, ro) // update goal R and V values for this ro
* *findTeam* (teams, ro) // Check to see whether the team in this ro is in the teams array
* *getMemberDataObj* (userID) // returns the studentData object for this userID
* *findMember* (id) // return a team member for some team, if he exists
* *unixTimeConversion* (uTime) // return a Date object based on the timestamp
* *arrayToObjects* (rows) // convert array of rows to array of objects
* *getAlphabeticLabel* (index) // returns A for level 2, B for level 3, and so forth
* *testScore* (varStr) // temp debugging routine
* *saveData* (data) // save a file to disk (called by downloadCSV)
* *downloadLogCSV*(csvDataArray) // converts query summary of log data to csv format and saves it to …LOGS.csv
* *downloadSummaryCSV*(csvSummaryArray) // converts summary data in Message or Teacher reports to csv format and saves it to …SUMMARY.csv

html-utils.js

* *makeTeamTable*(team, title, levelData, type, arrMssgScores) // takes a team, title, data array, and type ("Total", "Number", or "Average") and returns a formatted report table for team, all levels
* *scoreActions* (level) // returns total number, score, and average score of **messages** at level

analysis-tools.js

* *analyze* (rowObjs) // identifying/tallying each row of logged data, calling the appropriate add function
* *addAction* (ro, type) // adding a new action of each type, saving common parameters
* *findRValues* (ro, oldR) // find resistance values from row; return resistance matrix.
* *findVValues* (E, R0, R) // returns V given E, R0 and current R values.
* *duplicate* (action) // detect duplicate actions, returns true/false
* *addActivitySettings*(ro) // add E and R0 activity values to *activity-settings* actions
* *addModelValues*(ro) // add V1, V2, V3, Goal R1, GoalR2, Goal R3 to *model-values* actions
* *addJoinedGroup* (ro) // add to joined-group actions
* *addOpenedZoom*(ro) // add to opened-zoom actions
* *addClosedZoom*(ro) // add to closed-zoom actions
* *addConnectLead* (ro) // add to connect-lead actions
* *addDisconnectLead* (ro) // add to disconnect-lead actions
* *addRChange* (ro) // add to resistorChange actions
* *addMessage* (ro) // add to message actions
* *addCalculation* (ro) // add to calculation actions
* *addMeasurement* (ro, i) // add to measurement actions
* *addSubmit* (ro) // add to submitClicked actions
* *addSubmitER* (ro) // add to submitER actions
* *addAttachProbe* (ro) // add to attach-probe actions
* *addDetachProbe* (ro, i) // // add to detach-probe actions
* *addMovedDial* (ro, i) // add to move-dial actions

reporting-functions.js

* *reportResistorChange*(act) // add content to actionRow re resistor change
* *reportCalculation*(act) // add content to actionRow re calculation
* *reportMessage*(act) // add content to actionRow re message contents
* *reportActivitySettings*(act) // add content to actionRow re E and R0 values
* *reportMeasurement*(act) // add content to actionRow re results of measurement, and current flow and probes status at the time
* *reportMovedDial*(act) // add content to actionRow re change of DMM dial
* *reportSubmitVoltages*(act) // add content to actionRow re use of We Got It button
* *reportSubmitER*(act) // add content to actionRow re use of Submit E (or ER) button
* *reportAttachProbe*(act) // add content to actionRow re DMM probes and current flow
* *reportDetachProbe*(act) // add content to actionRow re DMM probes and current flow
* *reportConnectLead*(act) // add content to actionRow re wire change and current flow
* *reportDisconnectLead*(act) // add content to actionRow re re wire change and current flow
* *reportJoinedGroup*(act) // add content to actionRow re a member joining the group
* *reportOpenedZoom*(act) // add content to actionRow re opening the zoom window
* *reportClosedZoom*(act) // add content to actionRow re closing the zoom window
* *reportCircuitState*(act) // calculates voltages and current
* *reportAllActions*(teams, act) // generates table that reports all actions for the team and level, selected by act (not used?)
* *openNewWindow*() // ??
* *resDist*(act) //Returns the number of legal resistor values between the current one and the one that overshoots or undershoots the goal voltage. It will be 0 if the current resistance gets the user as close as possible to the goal voltage on that side, 1 if there is only one legal resistance value that provides a voltage on the same side of the goal voltage, and so forth.

report-tools.js

* *generateReport* (teams) // driving function called by “submit query” button to create reports for selected teams, actions, etc
* *reportResults* (teams) // extract and list actions selected by user
* *reportVarRefs* (teams) // extract and list varRefs selected by user
* *variableInVarRef* (vrStr, vrArray) // return true if vrStr found in vrArray
* *reportSummary* (teams) // report on resistor changes by category, team member, level.
* *reportActions* (teams, type) // create tables for each team summarizing message scores, calling makeTeamTable for 1) total scores, 2) number of mssgs, and 3) average score
* *teacherReport* (teams) // create table for selected teacher reporting the attainments or lack thereof by each team and level

PapaParse-4.1.2/papaparse.js

jquery.min.js (from Google)