makeversion.pl
Perl script generating production versions of
MOSFECCS_vY_xxxxxx.html (source code)
MOSFECCS_vY_xxxxxxcc.html (main script compiled)
mostest.js (program for testing SMILES-generator and -parser)
msvg.js (batch reconversion of exam answers [SMILES->structural formula])
from development source MOSFECCS_vY_SVG_DEV_xxxxxx.html

MOSFECCS_vY_SVG_DEV_xxxxxx-z.html is the central source file used for development and bug-fixing of MOSFECCS (Y is the main version, xxxxxx is the subversion (date YYMMDD) with build=xxxxxx.z).

In addition to all features of the MOSFECCS Editor for use with Moodle quizzes (the "production version"), the development source also contains the code to generate an SVG graphics file of whatever is drawn inside the drawing area when the SVG button (specific to the development version) is clicked. This button triggers the display of the SVG graphics together with the xml-source text of the graphics in an additional pop-up window.

Parts of the code to generate SVG-graphics from the molecular data objects contained in the development version are also used in the test program mostest.js and in the program msvg.js that converts SMILES-codes entered as answers in exams back into structural formulae for the examinator during grading.

makeversion.pl uses the source file of the development version MOSFECCS_vY_SVG_DEV_xxxxxx-z.html to generate 4 different programs

1. Production version MOSFECCS_vY_xxxxxx.html source code (build=xxxxxx.z).
2. Production version MOSFECCS_vY_xxxxxxcc.html with the main script compiled by the google closure compiler in ADVANCED mode.
3. mostest_xxxxxx.js (the testing program, uncompiled)
4. msvg_xxxxxx.js (uncompiled program to reconvert all SMILES answers entered in an exam (extracted from the Results export .csv file of the exam,) back into graphics of structural formulae (SVG-format) for inspection by the examinator.

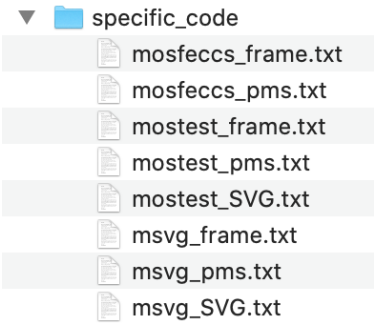
For a detailed description of the use of mostest.js (3. above) and msvg.js (4. above), see their manuals in the MOSTEST and MSVG directories.

MOSFECCS_vY_SVG_DEV_xxxxxx-x.html is subdivided into sections that are labeled and delimited by comment line TAGS as follows:

<!-- START mv_SVG FRAME -->
    HTML <head> tag
    script 1 dealing with GET parameters and setting global variables accordingly)
    parameter dependent loading of CSS. Preloading of graphics for editor GUI (icon tools).
    HTML </head> tag
    HTML <body> tag
<!-- END mv_SVG FRAME -->
<!-- START INTERACTIVE -->
//START CODE TO COMPILE
    main function drawCanvas() with all "global" variable declarations, event and button handlers and functions needed for user interaction and drawing on the 5 overlapping canvases.
//END INTERACTIVE
//START MV_SVG_DEV SVG
    handler for SVG button
    function drawMol_svg() that generates the SVG graphic
//END MV_SVG_DEV SVG
//START SVG
    functions drawBond_svg(), drawAtomLabel_svg(), drawArrow_svg(), drawRxnArrow_svg()
    that generate the elements of an SVG graphic
//END SVG
//START COMMON
    all functions common to all 4 programs, including the whole SMILES-generator [getsmiles()]
    constructors for all molecular data objects
//END COMMON
//START MV_SVG_DEV PMS
    variables and main of function parse_m_Smiles() [the multi-SMILES-parser]
//END MV_SVG_DEV PMS
//START PMS
    all functions internal to the multi-Smiles parser [parse_m_SMILES()],
    including the SMILES-parser for one structure (molecule) [parseSMILES()],
//END PMS
//START HTML-TERM
    storage of function drawCanvas in window['drawCanvas'] as external for compiler
//END CODE TO COMPILE
    script 3: call of main drawCanvas function with parameters as evaluated by script 1
        let drwCanv = window['drawCanvas'];
        drwCanv(pad,phone,app,arrows,helpWindow,svgWindow);
    HTML </body> tag

<!--END HTML-TERM-->

The sections in blue are specific to the MOSFECCS_vY_SVG_DEV_xxxxxx.html file
For the 4 production programs, makeversion.pl replaces them by the corresponding sections stored as text files in the folder specific_code:



Differences between the development source and the specific files:

A. *_frame.txt:

mosfeccs_frame.txt In script1, the development source loads the ***_full_rs.css files, whereas in the production MOSFECCS, the minified CSS files are loaded.

mostest_frame.txt contains the main IIFE function mostest() and auxiliary functions for reading the input files, the declarations of all variables "global" in the drawCanvas() context and the functions for output of LOG and SUMMARY files but neither script1 with CSS loading nor HTML. The main function mostest() calls the SMILES parser parse_m_SMILES() and the SMILES generator getsmiles().

msvg_frame.txt contains the main IIFE function msvg() and auxiliary functions for reading the input CDE file, the declarations of all variables "global" in the drawCanvas() context and the functions for output of HTML files (containg the SVGs for one question), the statistics (frequencies), LOG and SUMMARY files but neither script1 with CSS loading nor HTML. The main function msvg() calls the SMILES parser parse_m_SMILES() and the SMILES generator getsmiles().

B. *_pms.txt:

mosfeccs_pms.txt contains the parse_m_SMILES() main function calling the multismiles parser with mosfeccs-specific interactive error/failure/warnAtoms handling.
mostest_pms.txt contains the parse_m_SMILES_svg() main function calling the multismiles parser with mostest-specific error/failure/warnAtoms handling.
msvg_pms.txt contains the parse_m_SMILES_svg() main function calling the multismiles parser with msvg-specific error/failure/warnAtoms handling.

C. *_SVG.txt:

mostest_SVG.txt contains the functions clearSelection_svg(), get_mol_brects(),getboundrect_svg(), deleteAtom_svg(), clearMol_svg(), f1(ii), drawMol_svg().
msvg_SVG.txt contains the functions clearSelection_svg(), get_mol_brects(),getboundrect_svg(), deleteAtom_svg(), clearMol_svg(), f1(ii), drawMol_svg().
mostest_SVG.txt and msvg_SVG.txt differ only in the function drawMol_svg(), the other functions are identical.

Beware: It is up to the maintainer/contributor to adjust code in the specific files above accordingly, if bug fixes or new features introduced in the development source require it. makeversion.pl does not change code inside the specific files.

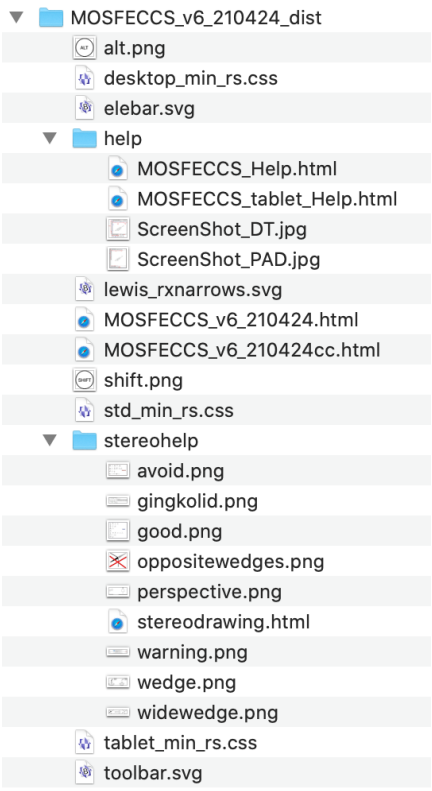
Compiled version MOSFECCS_vy_xxxxxxcc.html:
makeversion.pl generates mvY_xxxxxx.js, a copy of the section between the comment tags //START CODE TO COMPILE and //END CODE TO COMPILE
This file is then used as input for the compiler and the result (compiled code) is the file mvY_xxxxxxcc.js. MOSFECCS_vy_xxxxxxcc.html (the compiled production HTML file) is a copy of the production source file MOSFECCS_vY_xxxxxx.html in which the section between //START CODE TO COMPILE and //END CODE TO COMPILE has been replaced by the compiled code in mvY_xxxxxxcc.js.
The actual call of the google closure compiler by makeversion.pl is done by calling the shell script acc.sh, which contains the local full path to the compiler (and must be adjusted accordingly for the installation location of closure compiler on your computer).

script acc.sh

```
#!/bin/bash
# bash script for compilation of .js file with closure compiler in the ADVANCED mode
# USAGE: ./acc.sh infilename.js
# the first and only argument is the filename of the .js file to be compiled,
# which must reside in the current directory

input=$1

pat=".js"
subst="cc.js"
substerr="aerr.txt"

output=${input/$pat/$subst}
stderr=${input/$pat/$substerr}
/Users/bj/MOSFECCS/MOSFECCS_DEV/closure_compiler/compiler --compilation_level ADVANCED
$input > $output 2>$stderr
echo $input "compiled to " $output
```

Composing a new Distribution from the files generated by makeversion.pl

The listing below shows all files that are needed for a production release (distribution for standalone local installation or installation in Moodle).
After running makeversion.pl with a changed MOSFECCS_vY_SVG_DEV_xxxxxx-z.html development source file as input, replace the files MOSFECCS_vY_ xxxxxx.html and MOSFECCS_vY_ xxxxxxcc.html with the newly created ones. The files in the help folder might need updating as well (.html and annotated screen shot .png).
If the CSS files were changed (std_full_rs_css, tablet_full_rs.css and desktop_full_rs.css), make new minified versions (std_min_rs_css, tablet_min_rs.css and desktop_min_rs.css) with the CSS minfier (https://cssminifier.com) and exchange the corresponding files in the Distribution.
Keep the new mostest_xxxxxx.js in a safe place for testing purposes. Always use the version xxxxxx of mostest_xxxxxx.js to test MOSFECCS_vY_xxxxxx.html.
If the new Distribution shall be used in a Moodle online exam, consider using the new msvg_xxxxxx.js program with node.js to reconvert the SMILES answers into human readable structural formulae for non-automatic grading by the examiners (*see* the manual "MOODLE on-line Exams: Steps Needed to Generate the Tables for Visual Correction of Non-Automatically Graded Answers").



B. Jaun April 2021