```
/* --- FM-biblio.jsx ------ UTF-8 ------------------------------------------------
Menu items
  - Documentation of the process
  - Collect temporary citations into a new document, which is then saved as RTF
  - Instruct user how to deal with the RTF in EndNote resp. Citavi
  - Read the output-RTF of Citavi/EN and resolve the temp. citations to final citations, place bibliography.
  - Alternatively expand temp. citations in footnotes (or elsewhere) to bibliographic references.

  2015-04-09: In the user document temp. citations are in braces, such as  {Dante, #712}
              previously they were enclosed by double brackets [[Dante, #712]]
              Hence no special insertion (with pgm ctrl-y) is necessar anymore
  2015-11-17  Be careful with app.Displaying . It is stack-based
              See https://forums.adobe.com/message/8084988#8084988 by Ric Quatro
  ==>  11-18  Temporary (for breakpoint use) //?? outcomment messages
       12-01  gbDoingBook is only read from FM-biblio.ini;
              Hence for tests with independent steps it must be set correctly there.
------------------------------------------------------------------------------- */

#target framemaker

// --- global definitions ------------------------
var giLvlTrace      = 0;                          // level of function nesting
var gasFndCitations = [];                         // array of found citation texts (excluding braces/brackets)
var gsCollectPath   = "";                         // same as current book/file path
var gasFmtCitsRaw   = [];                         // left column in processed RTF
var gasFmtCitsFmt   = [];                         // right column in processed RTF
var gaoBibliography = [];                         // bibliography paragraphs from processed RTF
var gasIniVals = [], gasIniTexts  = [];           // values from ini and cfg file
var goCurrentDoc, goRtfDoc;                       // keep goRtfDoc open (keep the para -IDs intact in gaoBibliography)

FMbiblioSetup ();                                 // get data from the ini file
var gsFMbiblioname    = gasIniVals[0];            // name for message titles
var gsFMbiblioversion = gasIniVals[1];            // current version of script
var gbIsDebug      = StringToBool(gasIniVals[2]); // Debug output (tur/false)
var gbListItems    = StringToBool(gasIniVals[3]); // Debug output to separate file gsListFile
var gbDoingBook    = StringToBool(gasIniVals[4]); // Process started in a book
var gsTemplate     = gasIniVals[5];               // located in script-dir
var gsCollectFile  = gasIniVals[6];               // file name; collected items
var gsBibAppName   = gasIniVals[7];               // name of the bibliographic application
var gsBibAppLoc    = gasIniVals[8];               // path where gsBibAppName is located
var gsListFile     = gasIniVals[9];               // debug file in %appdata%\...\FrameMaker

var giCitUnique, giCitUnResolved, giBibEntries;   // statistics for final message

FMbiblioExpand (); // direct call - temp for test only


ReportInfo();

DefineMenus();

// --- Set up the menu
```

```
function DefineMenus () {
  giLvlTrace += 1; ZTrace ("DefineMenus");
  var sMenuMain   = gasIniTexts[0];
  var sMenuDocu   = gasIniTexts[1];
  var sMenuColl   = gasIniTexts[2];
  var sMenuBibapp = gasIniTexts[3];
  var sMenuRes1   = gasIniTexts[4];
  var sMenuRes2   = gasIniTexts[5];

// There is no menu Special in books => FormatMenu used instead for both cases
// --- Menu definition for documents
  var menuLocation = app.GetNamedMenu("FormatMenu");
  var FMbiblioMenu  = menuLocation.DefineAndAddMenu("!FMbibliomain", sMenuMain);
  FMbiblioMenu.DefineAndAddCommand(1,"docFMbiblioDocu",    sMenuDocu, "");
  FMbiblioMenu.DefineAndAddCommand(2,"docFMbiblioCollect", sMenuColl, "");
  FMbiblioMenu.DefineAndAddCommand(3,"docFMbiblioBibapp",  sMenuBibapp, "");
  FMbiblioMenu.DefineAndAddCommand(4,"docFMbiblioResolve", sMenuRes1, "");
  FMbiblioMenu.DefineAndAddCommand(5,"docFMbiblioExpand",  sMenuRes2, "");
// --- Menu definition for books
  menuLocation = app.GetNamedMenu("BookFormatMenu");
  var bkFMbiblioMenu  = menuLocation.DefineAndAddMenu("!bkFMbibliomain", sMenuMain);
  bkFMbiblioMenu.DefineAndAddCommand(1,"bookFMbiblioDocu",    sMenuDocu, "");
  bkFMbiblioMenu.DefineAndAddCommand(2,"bookFMbiblioCollect", sMenuColl, "");
  bkFMbiblioMenu.DefineAndAddCommand(3,"bookFMbiblioBibapp",  sMenuBibapp, "");
  bkFMbiblioMenu.DefineAndAddCommand(4,"bookFMbiblioResolve", sMenuRes1, "");
  bkFMbiblioMenu.DefineAndAddCommand(5,"bookFMbiblioExpand",  sMenuRes2, "");

  UpdateMenus();                                  // refresh and actually make the new menu appear

  giLvlTrace -= 1;
} // ---  end DefineMenus

function Command (cmd) {
  giLvlTrace += 1; ZTrace ("Command");
  switch (cmd) {
    case 1:  OpenHelpFile();                      // OK 2013-10
      break;
    case 2:  FMbiblioCollect();                   // OK 2015-02-18, 2015-04-14
      break;
    case 3:  FMbiblioCallBibapp();                // OK 2015-02-26
      break;
    case 4:  FMbiblioResolve();                   // OK 2015-03-20, 2015-04-14
      break;
    case 5:  FMbiblioExpand();                    // work in progress
      break;
  }
  giLvlTrace -= 1;
} // --- end Command

function OpenHelpFile () {
// ---------------------------------------------- Menu item "Documentation"
// --- display the pdf with same name as this script ($.filename includes extension)
  giLvlTrace += 1; ZTrace ();
  var helpFile;

  helpFile = File($.fileName.replace (/\.jsx(?:bin)?$/i , ".pdf"));
  if (helpFile.exists) {
    helpFile.execute();
  }
  giLvlTrace -= 1;
} // --- end OpenHelpfFile
```

```

function FMbiblioCollect () {
// ---------------------------------------------- Menu item "Collect temp. Citations"
  giLvlTrace += 1; ZTrace ("FMbiblioCollect");
  var docFile, newDoc;
  if ((app.ActiveDoc.ObjectValid() === 0)  && (app.ActiveBook.ObjectValid() === 0)){
//??    Message (true, "info", gasIniTexts[10], "FMbiblioCollect");   // please open document or book
    return;
  }
  gasFndCitations = [];                           // clear array from previous run
  if (app.ActiveBook.ObjectValid()) {
    docFile = app.ActiveBook.Name;
    gbDoingBook = true;
//??    Message ("true", "info", gasIniTexts[22], "FMbiblioCollect", docFile);
    gsCollectPath = GetDocPath ();                // path of active book
    CollectInBook(app.ActiveBook);
  } else {
    docFile = app.ActiveDoc.Name;
    gbDoingBook = false;
//??    Message ("true", "info", gasIniTexts[23], "FMbiblioCollect", docFile);
    gsCollectPath = GetDocPath ();                // path of active document
    CollectInDoc(app.ActiveDoc);
  }
// --- Insert found citation in new doc and save it as RTF
  if (gasFndCitations.length === 0) { // no citations found in doc or book
//??    Message (true, "warning", gasIniTexts[13], "FMbiblioCollect");
    giLvlTrace -= 1;
    return;
  }  if (gbIsDebug) {ReportFoundCitations ();}
  newDoc = CreateBibDoc ();                       // write cites to RTF
  SaveFileRTF (newDoc, gsCollectFile, gsCollectPath);
  Message ("true", "info", gasIniTexts[25], "FMbiblioCollect");
  giLvlTrace -= 1;
} // --- end FMbiblioCollect

function FMbiblioCallBibapp () {
// ---------------------------------------------- Menu item "Call Biblio App"
  giLvlTrace += 1; ZTrace ("FMbiblioCallBibapp");
  var msgNr, bibAppPgm;
  if (gsBibAppName.toLowerCase() === "citavi") {
    msgNr = 14;
  } else {
    msgNr  = 15;
  }
  bibAppPgm = File (gsBibAppLoc);
  if (bibAppPgm.exists) {
//??    Message (false, "info", gasIniTexts[msgNr], "FMbiblioCallBibapp", gsBibAppName); // The bibliographic application will be started ...
    gsCollectPath = GetDocPath ();                // path for active document
    CopyToClipboard (gsCollectPath + "\\" + gsCollectFile);
    bibAppPgm.execute();
  } else {     // The bibliographic application %01 not found at %02
//??    Message (true, "info", gasIniTexts[16], "FMbiblioCallBibapp", gsBibAppName, gsBibAppLoc);
    giLvlTrace -= 1;
    return;
  }
  giLvlTrace -= 1;
} // --- end FMbiblioCallBibapp

function FMbiblioResolve () {
```

```
// ---------------------------------------------- Menu item "Resolve temp. citations"
  giLvlTrace += 1; ZTrace ("FMbiblioResolve");
  var inputFile, prompt, docFile;

  if (IsBookRequiredQ ()) {
//??    Message (true, "error", gasIniTexts[17], "FMbiblioResolve");      // --- You collected citations from a book, but its not active now
    giLvlTrace -= 1;
    return;
  }
  prompt = gasIniTexts [18];                        // --- Select the RTF file with the formatted citations
  inputFile = ChooseFile (prompt, GetDocPath (), "*.rtf", Constants.FV_ChooseSelect);
  if (inputFile === null) {
//??    Message (true, "error", gasIniTexts[20], "FMbiblioResolve");      // --- File selection cancelled, step aborted
    giLvlTrace -= 1;
    return;
  }
  ReadFileRTF (inputFile);                         // --- get the modified RTF file and fill global arrays
  giCitUnique     = gasFmtCitsRaw.length;
  giBibEntries    = gaoBibliography.length;
  giCitUnResolved = giCitUnique - giBibEntries;

  if (gbIsDebug) {ReportResolvedCitations (giCitUnique, giBibEntries)};

  app.Displaying = false;                          // avoid screen flicker

  if (app.ActiveBook.ObjectValid()) {              // replace in book
//??    Message ("true", "info", gasIniTexts[22], "FMbiblioResolve", app.ActiveBook.Name)
    ReplaceInBook (app.ActiveBook);
  } else {                                         // replace in document
//??    Message ("true", "info", gasIniTexts[23], "FMbiblioResolve", app.ActiveDoc.Name)
    ReplaceInDoc (app.ActiveDoc);
  }
  app.Displaying = true;                           // screen refresh on again

  Message (true, "warning", gasIniTexts[21], "FMbiblioResolve", giCitUnique, giCitUnResolved, giBibEntries);
  giLvlTrace -= 1;
} // --- end FMbiblioResolve

function ReplaceInBook (book) {
  giLvlTrace += 1; ZTrace ("ReplaceInBook");
  var oDoc, docNme, bookComp = book.FirstComponentInBook;

  while(bookComp.ObjectValid()) {                  // process only FM docs
    if (bookComp.BookComponentFileType === Constants.FV_BK_FM) {
      book.StatusLine = "Processing " + File(bookComp.Name).displayName;
      docName = bookComp.Name;
      if (gbIsDebug) {Console("docName: "+ docName)};
//  doc = SimpleOpen (docName, false);             // Does not handle open errors automatically
      openParams = GetOpenParams ();
      openReturnParams =  new PropVals();
      oDoc = Open (docName, openParams, openReturnParams);
      ReplaceInDoc(oDoc);
    }
    bookComp = bookComp.NextBookComponentInDFSOrder;
```