```
<!DOCTYPE HTML> <!-- StudyPlan.htm - GENERATE A STUDY PLAN

MODULE INDEX
NAME            CONTENTS
<body>          HTML document body
showError       Show an error message
resetError      Reset the error message
readScalar      Read scalar parameters
createTable     Create the subject entry table
getSubVal       Get subject table value
isSubBlank      Test if the subject fields are blank
isExamBlank     Test if the exam fields are blank
readSub         Read the subject fields
readExam        Read the exam fields
readTable       Read the contents of the subject entry table
writeTable      Write the contents of the subject entry table
addLines        Add lines to the subject entry table
saveParms       Save the parameters to a file
loadParms       Load the parameters from a file
loadDayCnt      Determine the number of days and periods in the schedule
genPlan         Generate a study plan
evalPlan        Evaluate the objective equation for a study plan
emitPlan        Emit a study plan
runModel        Run the model
-               Main line

MAINTENANCE HISTORY
DATE            PROGRAMMER AND DETAILS
28-09-17        JS      Original
09-10-17        YGL     Added footnote

------------------------------------------------------------------------------->

<html>
<head>
        <title>Generate a Study Plan</title>
</head>
<style>
        #subTable, #subCode, #basKnow, #basPerKnow, #basVolKnow,
        #incKnow, #incPerKnow, #incVolKnow, #volHalf, #weight,
        #exam, #examDate, #examPrd {
        border: 1px solid grey;
        border-collapse: collapse;
        }
        #subCode { max-width: 10em; }
        #volHalf { max-width: 4em; }
        #weight { max-width: 5em; }
        #resTable, #resRow, #resDate {
        border: 1px solid grey;
        border-collapse: collapse;
        }
        #resSub {
        border-left: 1px solid grey;
        width: 10em;
```

```
        }
        #resVal { width: 4em; }
</style>
<body>
        <h1>Generate a Study Plan</h1>
        <table>
                <tr>
                        <td>Start date</td>
                        <td><input id="startDate" type="date"/></td>
                </tr>
                <tr>
                        <td>Study periods per day</td>
                        <td><input id="prdPerDay" type="number"
                        min="1" max="12" step="1"/></td>
                </tr>
                <tr>
                        <td>Number of cycles</td>
                        <td><input id="cycleCnt" type="number"
                        min="1" max="100000000" step="1"/></td>
                </tr>
        </table>
        <table id="subTable"></table>
        <p>
        <button id="addLinesBut">Add lines</button>
        <table>
                <tr>
                        <td style="vertical-align: text-top">Note:</td>
                        <td style="max-width: 40em">
                        if a subject has more than one exam, put the
                        dates and times of the additional exams on
                        otherwise blank lines below the subject line.
                        </td>
                </tr>
        </table>
        <p>
        <table>
                <tr>
                        <td>File name</td>
                        <td><input id="fileName" pattern="[A-Za-z][A-Za-z0-9]*"
                        type="text"/></td>
                        <td><button id="saveBut">Save</button>
                        <td><button id="loadBut">Load</button>
                </tr>
        </table>
        <table id="errTable"></table>
        <p>
        <button id="runBut">Run</button>
        <table id="resTable"></table>
        <p id="e2"></p>
        <table>
                <tr>Copyright 2017 Jonathan Searcy</tr>
                <tr></tr>
                <tr>
                        <td style="max-width: 50em">
```

```
                                This program is free software: you can redistribute it and/or
                                modify
                                it under the terms of the GNU General Public License as
                                published by
                                the Free Software Foundation, either version 3 of the License, or
                                (at your option) any later version.
                        </td>
                </tr>
                <tr></tr>
                <tr>
                        <td style="max-width: 50em">
                                This program is distributed in the hope that it will be useful,
                                but WITHOUT ANY WARRANTY; without even the implied warranty of
                                MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
                                GNU General Public License for more details.
                        </td>
                </tr>
                <tr></tr>
                <tr>
                        <td style="max-width: 50em">
                                You should have received a copy of the GNU General Public License
                                along with this program.  If not, see
                                <a
                                href="http://www.gnu.org/licenses/">http://www.gnu.org/licenses/<
                                /a>
                        </td>
                </tr>
        </table>
</body>
<script>

//-----------------------------------------------------------------------------

// DEFINITIONS

var     PRD_UNUSED = 0;         // Unused period
var     PRD_STUDY = 1;          // Study period
var     PRD_EXAM = 2;           // Exam period
var     CHANGE_PROB = 0.03;     // Change probability

//-----------------------------------------------------------------------------

// GLOBAL VARIABLES

var     subTabSize;             // Current subject table size
var     subArr;                 // Subject array
var     startDate;              // Model start date
var     prdPerDay;              // Periods per day
var     cycleCnt;               // Number of model cycles

//-----------------------------------------------------------------------------

// SHOW AN ERROR MESSAGE
```

```
function showError (errMsg) {
        var errBody = "";
        errBody += "<tr>";
        errBody += "<td style=\"vertical-align: text-top; " +
                "color: red;\">Sorry:</td>";
        errBody += "<td style=\"vertical-align: text-top; max-width: 40em; " +
                "color: red;\">" + errMsg + "</td>";
        errBody += "</tr>";
        document.getElementById ("errTable").innerHTML = errBody;
}

//----------------------------------------------------------------------------

// RESET THE ERROR MESSAGE

function resetError () {
        document.getElementById ("errTable").innerHTML = "";
}

//----------------------------------------------------------------------------

// READ SCALAR PARAMETERS

function readScalar () {
        if (document.getElementById ("startDate").value == "")
                throw ("missing start date");
        startDate = Date.parse(document.getElementById("startDate").value);
        if (document.getElementById ("prdPerDay").value == "")
                throw ("missing periods per day");
        prdPerDay = parseInt(document.getElementById("prdPerDay").value);
        if (document.getElementById ("cycleCnt").value == "")
                throw ("missing number of cycles");
        cycleCnt = parseInt(document.getElementById("cycleCnt").value);
}

//----------------------------------------------------------------------------

// CREATE THE SUBJECT ENTRY TABLE

function createTable (lineCnt) {
        var     tableBody;      // Table body buffer
        var     i;              // General purpose index

        tableBody = "";
        tableBody += "<tr>";
        tableBody += "<td rowspan=\"2\" id=\"subCode\">Subject code</td>";
        tableBody += "<td colspan=\"2\" id=\"basKnow\">Base Knowledge %</td>";
        tableBody += "<td colspan=\"2\" id=\"incKnow\">" +
                "Incremental Knowledge %</td>";
        tableBody += "<td rowspan=\"2\" id=\"volHalf\">" +
                "Volatile half-life (days)</td>";
        tableBody += "<td rowspan=\"2\" id=\"weight\"> " + "Subject weight</td>";
        tableBody += "<td colspan=\"2\" id=\"exam\">Exam</td>";
        tableBody += "</tr>";
```

```
        tableBody += "<tr>";
        tableBody += "<td id=\"basPerKnow\">Permanent</td>";
        tableBody += "<td id=\"basVolKnow\">Volatile</td>";
        tableBody += "<td id=\"incPerKnow\">Permanent</td>";
        tableBody += "<td id=\"incVolKnow\">Volatile</td>";
        tableBody += "<td id=\"examDate\">Date</td>";
        tableBody += "<td id=\"examPrd\">Period</td>";
        for (i = 0; i < lineCnt; i++) {
                tableBody += "<tr>";
                tableBody += "<td id=\"subCode\">" +
                        "<input id=\"subCode_" + i + "\" " + "style=\"border: none;\" " +
                        //"type=\"text\" maxlength=\"4\" size=\"4\"/></td>";
                        "type=\"text\" size=\"10\"/></td>";
                tableBody += "<td id=\"basPerKnow\">" +
                        "<input id=\"basPerKnow_" + i + "\" " + "style=\"border: none;\" " +
                        "type=\"number\" min=\"0\" max=\"100\" " + "step=\"0.1\"/></td>";
                tableBody += "<td id=\"basVolKnow\">" +
                        "<input id=\"basVolKnow_" + i + "\" " + "style=\"border: none;\" " +
                        "type=\"number\" min=\"0\" max=\"100\" " + "step=\"0.1\"/></td>";
                tableBody += "<td id=\"incPerKnow\">" +
                        "<input id=\"incPerKnow_" + i + "\" " + "style=\"border: none;\" " +
                        "type=\"number\" min=\"0\" max=\"100\" " + "step=\"0.1\"/></td>";
                tableBody += "<td id=\"incVolKnow\">" +
                        "<input id=\"incVolKnow_" + i + "\" " + "style=\"border: none;\" " +
                        "type=\"number\" min=\"0\" max=\"100\" " + "step=\"0.1\"/></td>";
                tableBody += "<td id=\"volHalf\">" +
                        "<input id=\"volHalf_" + i + "\" " + "style=\"border: none;\" " +
                        "type=\"number\" min=\"1\" max=\"999\" " + "step=\"1\"/></td>";
                tableBody += "<td id=\"weight\">" +
                        "<input id=\"weight_" + i + "\" " + "style=\"border: none;\" " +
                        "type=\"number\" min=\"0\" max=\"100\" " + "step=\"0.1\"/></td>";
                tableBody += "<td id=\"examDate\">" +
                        "<input id=\"examDate_" + i + "\" " + "style=\"border: none;\" " +
                        "type=\"date\"/></td>";
                tableBody += "<td id=\"examPrd\">" +
                        "<input id=\"examPrd_" + i + "\" " + "style=\"border: none;\" " +
                        "type=\"number\" min=\"1\" max=\"12\" " + "step=\"1\"/></td>";
                tableBody += "</tr>";
        }
        document.getElementById ("subTable").innerHTML = tableBody;
}

//---------------------------------------------------------------------------

// GET SUBJECT TABLE VALUE

function getSubVal (colName, lineNo) {
        return document.getElementById (colName + "_" + lineNo).value;
}

//---------------------------------------------------------------------------

// TEST IF THE SUBJECT FIELDS ARE BLANK
```

```
function isSubBlank (lineNo) {
        return getSubVal ("subCode", lineNo) == "" &&
                getSubVal ("basPerKnow", lineNo) == "" &&
                getSubVal ("basVolKnow", lineNo) == "" &&
                getSubVal ("incPerKnow", lineNo) == "" &&
                getSubVal ("incVolKnow", lineNo) == "" &&
                getSubVal ("volHalf", lineNo) == "" &&
                getSubVal ("weight", lineNo) == "";
}

//---------------------------------------------------------------------------

// TEST IF THE EXAM FIELDS ARE BLANK

function isExamBlank (lineNo) {
        return getSubVal ("examDate", lineNo) == "" &&
                getSubVal ("examPrd", lineNo) == "";
}

//---------------------------------------------------------------------------

// READ THE SUBJECT FIELDS

function readSub (lineNo) {
        this.subCode = getSubVal ("subCode", lineNo);
        this.basPerKnow = parseFloat(getSubVal ("basPerKnow", lineNo));
        this.basVolKnow = parseFloat(getSubVal ("basVolKnow", lineNo));
        this.incPerKnow = parseFloat(getSubVal ("incPerKnow", lineNo));
        this.incVolKnow = parseFloat(getSubVal ("incVolKnow", lineNo));
        this.volHalf = parseFloat(getSubVal ("volHalf", lineNo));
        this.weight = parseFloat(getSubVal ("weight", lineNo));
        this.examArr = [];
}

//---------------------------------------------------------------------------

// READ EXAM FIELDS

function readExam (lineNo) {
        this.examDate = Date.parse(getSubVal ("examDate", lineNo));
        this.examPrd = parseInt(getSubVal ("examPrd", lineNo));
}

//---------------------------------------------------------------------------

// READ THE CONTENTS OF THE SUBJECT ENTRY TABLE

function readTable () {
        var     lineNo;         // Line number in the table
        var     newSub;         // New subject structure

        lineNo = 0;
        subArr = [];
        for (;;) {
```

```
                while (lineNo < subTabSize && isSubBlank(lineNo) &&
                        isExamBlank(lineNo)) lineNo ++ ;
                if (lineNo >= subTabSize) break;
                if (
                        getSubVal ("subCode", lineNo) == "" ||
                        getSubVal ("basPerKnow", lineNo) == "" ||
                        getSubVal ("basVolKnow", lineNo) == "" ||
                        getSubVal ("incPerKnow", lineNo) == "" ||
                        getSubVal ("incVolKnow", lineNo) == "" ||
                        getSubVal ("volHalf", lineNo) == "" ||
                        getSubVal ("weight", lineNo) == "" ||
                        getSubVal ("examDate", lineNo) == "" ||
                        getSubVal ("examPrd", lineNo) == ""
                ) throw ("missing field in subject header line");
                newSub = new readSub (lineNo);
                newSub.examArr.push (new readExam (lineNo));
                lineNo ++ ;
                for (;;) {
                        while (lineNo < subTabSize && isSubBlank(lineNo) &&
                                isExamBlank(lineNo)) lineNo ++ ;
                        if (lineNo >= subTabSize || ! isSubBlank(lineNo)) break;
                        if (
                                getSubVal ("examDate", lineNo) == "" ||
                                getSubVal ("examPrd", lineNo) == ""
                        ) throw ("missing field in exam extension line");
                        newSub.examArr.push (new readExam (lineNo));
                        lineNo ++ ;
                }
                subArr.push (newSub);
        }
}

//---------------------------------------------------------------------------

// WRITE THE CONTENTS OF THE SUBJECT ENTRY TABLE

function writeTable () {
        var lineNo = 0;
        for (var i = 0; i < subArr.length; i++) {
                document.getElementById ("subCode_"+lineNo).value =
                        subArr[i].subCode;
                document.getElementById ("basPerKnow_"+lineNo).value =
                        subArr[i].basPerKnow;
                document.getElementById ("basVolKnow_"+lineNo).value =
                        subArr[i].basVolKnow;
                document.getElementById ("incPerKnow_"+lineNo).value =
                        subArr[i].incPerKnow;
                document.getElementById ("incVolKnow_"+lineNo).value =
                        subArr[i].incVolKnow;
                document.getElementById ("volHalf_"+lineNo).value =
                        subArr[i].volHalf;
                document.getElementById ("weight_"+lineNo).value =
                        subArr[i].weight;
                for (var j = 0; j < subArr[i].examArr.length; j++) {
```

```
                        if (j != 0) lineNo ++ ;
                        document.getElementById ("examDate_"+lineNo).value =
                                new Date (subArr[i].examArr[j].examDate)
                                        .toISOString().substring(0,10);
                        document.getElementById ("examPrd_"+lineNo).value =
                                subArr[i].examArr[j].examPrd;
                } lineNo ++ ;
        }
}

//-----------------------------------------------------------------------------

// ADD LINES TO THE SUBJECT ENTRY TABLE

function addLines () {
        try {
                resetError ();
                readTable ();
                subTabSize += 5;
                createTable (subTabSize);
                writeTable ();
        }
        catch (errMsg) {
                showError (errMsg);
        }
}

//-----------------------------------------------------------------------------

// SAVE THE PARAMETERS TO A FILE

function saveParms () {
        var     parms;          // Parameters structure
        var     fileName;       // File name under which to save the parameters

        try {
                resetError ();
                if (typeof(Storage) === "undefined")
                        throw "no web storage support";
                fileName = document.getElementById ("fileName").value;
                if (fileName == "")
                        throw "file name required";
                readScalar ();
                readTable ();
                parms = {
                        startDate: startDate,
                        prdPerDay: prdPerDay,
                        cycleCnt: cycleCnt,
                        subArr: subArr
                };
                localStorage.setItem (fileName, JSON.stringify(parms));
        }
        catch (errMsg) {
                showError (errMsg);
```

```
        }
}

//---------------------------------------------------------------------------

// LOAD THE PARAMETERS FROM A FILE

function loadParms () {
        var     parms;          // Parameters structure
        var     fileName;       // File name from which to load the parameters

        try {
                resetError ();
                if (typeof(Storage) === "undefined")
                        throw "no web storage support";
                fileName = document.getElementById ("fileName").value;
                if (fileName == "")
                        throw "file name required";
                if (localStorage.getItem(fileName) === null)
                        throw "file does not exist";
                parms = JSON.parse (localStorage.getItem (fileName));
                document.getElementById ("startDate").value =
                        new Date(parms.startDate).toISOString().substring(0,10);
                document.getElementById ("prdPerDay").value = parms.prdPerDay;
                document.getElementById ("cycleCnt").value = parms.cycleCnt;
                subArr = parms.subArr;
                subTabSize = 0;
                for (var i = 0; i < subArr.length; i++)
                        subTabSize += subArr[i].examArr.length < 1 ?
                                1 : subArr[i].examArr.length;
                subTabSize = (subTabSize + 4) / 5 * 5;
                if (subTabSize < 10) subTabSize = 10;
                createTable (subTabSize);
                writeTable ();
        }
        catch (errMsg) {
                showError (errMsg);
        }
}

//---------------------------------------------------------------------------

// DETERMINE THE NUMBER OF DAYS AND PERIODS IN THE SCHEDULE

function loadDayCnt () {
        var minExamDate;        // Minimum exam date
        var maxExamDate;        // Maximum exam date

        minExamDate = subArr[0].examArr[0].examDate;
        maxExamDate = subArr[0].examArr[0].examDate;
        for (var i = 0; i < subArr.length; i++) {
                for (var j = 0; j < subArr[i].examArr.length; j++) {
                        if (subArr[i].examArr[j].examDate < minExamDate)
                                minExamDate = subArr[i].examArr[j].examDate;
```

```
                        if (subArr[i].examArr[j].examDate > maxExamDate)
                                maxExamDate = subArr[i].examArr[j].examDate;
                }
        }
        if (minExamDate < startDate)
                throw ("exam before start date");
        dayCnt = (maxExamDate - startDate)/(24*60*60*1000) + 1;
        prdCnt = dayCnt * prdPerDay;

        return true;
}

//-----------------------------------------------------------------------------

// GENERATE A STUDY PLAN

function genPlan (bestPlan) {
        var     newPlan;        // New study plan
        var     actSubSet;      // Active subject set
        var     actSubArr;      // Active subject array
        var prd;                // Period number
        var     i;              // General purpose index
        var     j;              // General purpose index

        // Initialise the new study plan

        newPlan = [];
        for (prd = 0; prd < prdCnt; prd++)
                newPlan.push ({prdState: PRD_UNUSED, prdSub: 0, prdResult: 0});

        // Load the exams

        actSubSet = [];
        for (i = 0; i < subArr.length; i++) {
                actSubSet[i] = false;
                for (j = 0; j < subArr[i].examArr.length; j++) {
                        prd = (subArr[i].examArr[j].examDate - startDate)
                                / (24*60*60*1000) * prdPerDay
                                + subArr[i].examArr[j].examPrd;
                        newPlan[prd].prdState = PRD_EXAM;
                        newPlan[prd].prdSub = i;
                }
        }

        // Randomly select subjects for each period

        actSubArr = [];
        prd = prdCnt;
        while (prd != 0) {
                --prd;
                switch (newPlan[prd].prdState) {
                case PRD_UNUSED:
                        if (actSubArr.length != 0) {
                                if (
```

```
                                        bestPlan === null ||
                                        Math.random() < CHANGE_PROB
                                ) {
                                        i = Math.floor(Math.random()*actSubArr.length);
                                        newPlan[prd].prdState = PRD_STUDY;
                                        newPlan[prd].prdSub = actSubArr[i];
                                } else {
                                        newPlan[prd] = bestPlan[prd];
                                }
                        }
                        break;
                case PRD_EXAM:
                        if ( ! actSubSet[newPlan[prd].prdSub]) {
                                actSubSet[newPlan[prd].prdSub] = true;
                                actSubArr.push (newPlan[prd].prdSub);
                        }
                        break;
                }
        }

        return newPlan;
}

//-----------------------------------------------------------------------------

// EVALUATE THE OBJECTIVE EQUATION FOR A STUDY PLAN

function evalPlan (plan) {
        var     subStateArr;    // Subject state array
        var     sub;            // Subject reference
        var     subState;       // Subject state reference
        var     redVolKnow;     // Reduced volatile knowledge
        var     totKnow;        // Total knowledge
        var     newPerKnow;     // New permanent knowledge
        var     newVolKnow;     // New volatile knowledge
        var     e2;             // Sum of the errors squared
        var     i;              // General purpose index
        var     prd;            // Period index

        // Initialise the subject state array

        subStateArr = [];
        for (i = 0; i < subArr.length; i++) {
                subStateArr[i] = {
                        lastPrd: 0,
                        perKnow: subArr[i].basPerKnow,
                        volKnow: subArr[i].basVolKnow
                };
        }

        // Process the effect of each period

        e2 = 0;
        for (prd = 0; prd < prdCnt; prd++) {
```

```
                sub = subArr[plan[prd].prdSub];
                subState = subStateArr[plan[prd].prdSub];
                redVolKnow = subState.volKnow * Math.exp (
                        -(prd - subState.lastPrd) * Math.LN2
                        / (prdPerDay * sub.volHalf));
                totKnow = subState.perKnow + redVolKnow;
                switch (plan[prd].prdState) {
                case PRD_STUDY:
                        newPerKnow = sub.incPerKnow * (100.0-totKnow) / 100.0;
                        newVolKnow = sub.incVolKnow * (100.0-totKnow) / 100.0;
                        subState.lastPrd = prd;
                        subState.perKnow += newPerKnow;
                        subState.volKnow = redVolKnow + newVolKnow;
                        break;
                case PRD_EXAM:
                        plan[prd].prdResult = totKnow;
                        e2 += sub.weight * ((100-totKnow) * (100-totKnow))
                                / sub.examArr.length;
                        break;
                }
        }

        return e2;
}

//---------------------------------------------------------------------------

// EMIT A STUDY PLAN

function emitPlan (plan) {
        var resTab = "";
        var i;
        var j;
        var localDate;
        var prdPlan;

        resTab += "<tr id=\"resRow\">";
        resTab += "<td id=\"resDate\">Date</td>";
        for (j = 0; j < prdPerDay; j++)
                resTab += "<td colspan=\"2\" id=\"resSub\">Period "+j+"</td>";
        resTab += "</tr>";

        for (i = 0; i < dayCnt; i++) {
                resTab += "<tr id=\"resRow\">";
                localDate = new Date (startDate + i*24*60*60*1000
                        + new Date().getTimezoneOffset());
                resTab += "<td id=\"resDate\">" + localDate.toDateString() +
                        "</td>";
                for (j = 0; j < prdPerDay; j++) {
                        prdPlan = plan[i*prdPerDay + j];
                        switch (prdPlan.prdState) {
                        case PRD_UNUSED:
                                resTab += "<td id=\"resSub\">----</td>";
                                resTab += "<td id=\"resVal;\"></td>";
```

```
                        break;
                case PRD_STUDY:
                        resTab += "<td id=\"resSub\">" +
                                subArr[prdPlan.prdSub].subCode+"</td>";
                        resTab += "<td id=\"resVal\"></td>";
                        break;
                case PRD_EXAM:
                        resTab += "<td id=\"resSub\">" +
                                subArr[prdPlan.prdSub].subCode+"*</td>";
                        resTab += "<td id=\"resVal\">" +
                                prdPlan.prdResult.toFixed(2) + "</td>";
                        break;
                }
        }
        resTab += "</tr>";
    }
    document.getElementById ("resTable").innerHTML = resTab;
}

//-----------------------------------------------------------------------------

// RUN THE MODEL

function runModel () {
    var     bestPlan;       // Best study plan
    var     bestE2;         // Best sum of errors squared
    var     expPlan;        // Experimental study plan
    var     expE2;          // Experimental sum of errors squared
    var     cycle;          // Cycle index

    // Process rejections

    try {
        // Reset the error display and results

        resetError ();
        document.getElementById ("resTable").innerHTML = "";

        // Read the scalar parameters

        readScalar ();

        // Read the subject table

        readTable ();
        if (subArr.length == 0) throw ("no subject data");

        // Determine the number of days and periods in the schedule

        loadDayCnt ();

        // Search for the best study plan

        bestPlan = genPlan (null);
```

```
                bestE2 = evalPlan (bestPlan);
                for (cycle = 1; cycle < cycleCnt; cycle++) {
                        expPlan = genPlan (bestPlan);
                        expE2 = evalPlan (expPlan);
                        if (expE2 < bestE2) {
                                bestPlan = expPlan;
                                bestE2 = expE2;
                        }
                }

                // Emit the best plan

                emitPlan (bestPlan);
                document.getElementById ("e2").innerHTML =
                        "Value of the objective equation = " +
                        bestE2.toFixed(2);
        }
        catch (errMsg) {
                showError (errMsg);
        }
}

//------------------------------------------------------------------------

// MAIN LINE

subTabSize = 10;
createTable (subTabSize);
document.getElementById ("addLinesBut").addEventListener ("click", addLines);
document.getElementById ("saveBut").addEventListener ("click", saveParms);
document.getElementById ("loadBut").addEventListener ("click", loadParms);
document.getElementById ("runBut").addEventListener ("click", runModel);

//------------------------------------------------------------------------

</script>
</html>
```