```
/*
Copyright © 2005, Apple Computer, Inc.  All rights reserved.
NOTE:  Use of this source code is subject to the terms of the Software
License Agreement for Mac OS X, which accompanies the code.  Your use
of this source code signifies your agreement to such license terms and
conditions.  Except as expressly granted in the Software License Agreement
for Mac OS X, no other copyright, patent, or other intellectual property
license or right is granted, either expressly or by implication, by Apple.
*/



//
// Terminology:
//
// The "selected" date is the one that is currently being browsed and has the yellow highlight;
// it may or may not be the same as today's date.
//
// The "collapsed view" state is the smaller size, with only the two tiles showing.
// The "grid view" state is the larger size, with the calendar day grid array (setupMonth)
showing.
//
// A "date id" is an html tag ID of the form "0102" (mmdd) that is used to tag and refer to the
// dynamically-generated selectable calendar days in the grid.
//



var resizeAnimation = {startTime:0, duration:250, positionFrom:0, positionTo:0, positionNow:0,
frameFrom:0, frameTo:0, frameNow:0, timer:null, element:null, image:null, background:null,
onfinished:null, initialDateToShow:null};
var date = new Date();
var currentMonth = date.getMonth();
var currentYear = date.getFullYear();
var currentDate = date.getDate();
var selectedDateID = idFromMonthDate (currentMonth, currentDate);
var isCollapsed = false;
var updateDateTimer = null;



document.addEventListener("keypress", keyPressed, true);



function keyPressed(e)
{
    if (!isCollapsed)
    {
        if (!currentDate)
            currentDate = 1;

        var mo = new Date(currentYear, currentMonth, currentDate);
        var dateChanged = false;
        var monthYearView = false;

        switch (e.keyIdentifier) {
```

```
        case "Up":
            if (e.metaKey) {
                mo.setFullYear(currentYear+1);
                monthYearView = true;
            } else
                mo.setDate(currentDate-7);
            dateChanged = true;
            break;
        case "Down":
            if (e.metaKey) {
                mo.setFullYear(currentYear-1);
                monthYearView = true;
            } else
                mo.setDate(currentDate+7);
            dateChanged = true;
            break;
        case "Right":
            if (e.metaKey) {
                currentMonth++;
                if (currentMonth > 11) {
                    currentMonth = 0;
                    currentYear++;
                }
                mo.setMonth(currentMonth);
                mo.setFullYear(currentYear);
                monthYearView = true;
            } else
                mo.setDate(currentDate+1);
            dateChanged = true;
            break;
        case "Left":
            if (e.metaKey) {
                currentMonth--;
                if (currentMonth < 0) {
                    currentMonth = 11;
                    currentYear--;
                }
                mo.setMonth(currentMonth);
                mo.setFullYear(currentYear);
                monthYearView = true;
            } else
                mo.setDate(currentDate-1);
            dateChanged = true;
            break;

        case "PageUp":
            currentMonth--;
            if (currentMonth < 0) {
                currentMonth = 11;
                currentYear--;
            }
            mo.setMonth(currentMonth);
            mo.setFullYear(currentYear);
            monthYearView = true;
            dateChanged = true;
            break;
```

```
            case "PageDown":
                currentMonth++;
                if (currentMonth > 11) {
                    currentMonth = 0;
                    currentYear++;
                }
                mo.setMonth(currentMonth);
                mo.setFullYear(currentYear);
                monthYearView = true;
                dateChanged = true;
                break;

            case "Home":
                var today = new Date();

                currentMonth = today.getMonth();
                mo.setMonth(currentMonth);
                currentYear = today.getFullYear();
                mo.setFullYear(currentYear);
                currentDate = today.getDate();
                mo.setDate(currentDate);
                dateChanged = true;
                break;

            default:
                break;
        }

        if (dateChanged)
        {
            // create the new calendar
            currentMonth = mo.getMonth();
            currentYear = mo.getFullYear();
            currentDate = mo.getDate();
            setupMonth(currentMonth, currentYear);

            hiliteToday();

            if (!monthYearView)
                hiliteDate(idFromMonthDate(currentMonth, currentDate));
            else
                hiliteDate("");

            if (monthYearView)
                currentDate = 0; // note this makes the id of the form mm00yyyy which doesn't
correspond to an HTML id

            updateTiles();

            selectedDateID = idFromMonthDate (currentMonth, currentDate);

            e.stopPropagation();
            e.preventDefault();
        }
    }

}
```

```
function limit_3 (a, b, c)
{
    return a < b ? b : (a > c ? c : a);
}



function computeNextFloat (from, to, ease)
{
    return from + (to - from) * ease;
}



function animate ()
{
    var T;
    var ease;
    var time  = (new Date).getTime();
    var yLoc;
    var frame;

    T = limit_3(time-resizeAnimation.startTime, 0, resizeAnimation.duration);
    ease = 0.5 - (0.5 * Math.cos(Math.PI * T / resizeAnimation.duration));

    if (T >= resizeAnimation.duration)
    {
        yLoc = resizeAnimation.positionTo;
        clearInterval (resizeAnimation.timer);
        resizeAnimation.timer = null;

        if (resizeAnimation.onfinished)
            setTimeout ("resizeAnimation.onfinished();", 0); // call after the last frame is
drawn
    }
    else
        yLoc = computeNextFloat(resizeAnimation.positionFrom, resizeAnimation.positionTo,
ease);

    // convert float to integer; old parseInt was like a round-down (floor), now Math.round
rounds up
    resizeAnimation.positionNow = Math.round(yLoc);
    resizeAnimation.element.style.height = resizeAnimation.positionNow;
}



function toggleGridViewAnimFinished()
{
    updateTiles();

    if (window.widget)
    {
        if (isCollapsed)
```

```
        {
            window.resizeTo (177, 114);
        }
        widget.setPreferenceForKey (isCollapsed ? "true" : "false", createKey("collapsed"));
    }
}



function dayClassFromDayIndex (inDay)
// specifying this part of the css class on each day (just to clear:left .sun)
// may not be needed anymore
{
    switch (inDay)
    {
        case 0: return "sun";  break;
        case 1: return "mon";  break;
        case 2: return "tue";  break;
        case 3: return "wed";  break;
        case 4: return "thur"; break;
        case 5: return "fri";  break;
        case 6: return "sat";  break;
    }
}



function idFromMonthDate(month, dateCount)
{
    return ((month<10) ? "0" + month : month.toString()) +
           ((dateCount<10) ? "0" + dateCount : dateCount.toString());
}



function setupWeekHeader ()
//
// Get localized names of the narrow days of the week from the plugin and populate the Grid
View header:
//
{
    for (var i = 0;   i <= 6;   i++)
    {
        document.getElementById ("dayofweek" + i).innerText = fetchIPrefNarrowDayOfWeekName
(i);
    }
}



// JavaScript's Date.getDay() is Sunday-based, so we have to fix it up.
function getDayFromDate(date)
{
    var dayIndex = date.getDay() - fetchIPrefFirstWeekday();
    if (dayIndex < 0)
        dayIndex += 7;
    return dayIndex;
```

```
}



function setupMonth(month, year)
{
    var mo = new Date(year, month, 1);
    var firstDay = getDayFromDate(mo);
    var monthDiv = document.getElementById("calendarDetail");
    var dayOfWeek = 0;
    var dateCount = 1;
    var rowCount = 0;

    // show the names of the days of the week
    setupWeekHeader ();

    // clear out the old
    if (monthDiv.hasChildNodes()) {
        var children = monthDiv.childNodes;
        var count = children.length;
        for(var j=0; j < count; j++) {
            monthDiv.removeChild(children[0]);
        }
    }

    // create the month
    for(var i=0; i< 48; i++)
    {
        var dateSpan = document.createElement("span");

        if (dayOfWeek < firstDay) {
            dateSpan.innerHTML = " ";
        } else {
            dateSpan.innerHTML = dateCount;
            dateSpan.setAttribute("id", idFromMonthDate(month, dateCount));
            dateCount++;
            mo.setDate(dateCount);
        }

        dateSpan.setAttribute("onmousedown", "selectDate(event, this.id);");
        dateSpan.setAttribute("onkeydown", "selectDate(event, this.id);");
        dateSpan.setAttribute("class", dayClassFromDayIndex(dayOfWeek%7));
        monthDiv.appendChild(dateSpan);

        // increment the number of rows
        if (dayOfWeek%7 == 0)
            rowCount++;

        // stop after the number of days in this month
        if (mo.getMonth() != month)
            break;

        dayOfWeek++;
    }

    // tweak the row divider artwork
    var calDiv1 = document.getElementById("calDiv1");
```

```
    var calDiv2 = document.getElementById("calDiv2");

    if (rowCount == 5) {
        calDiv1.style.visibility = "visible";
        calDiv2.style.visibility = "hidden";
    } else if (rowCount == 6) {
        calDiv1.style.visibility = "visible";
        calDiv2.style.visibility = "visible";
    } else {
        calDiv1.style.visibility = "hidden";
        calDiv2.style.visibility = "hidden";
    }
}



function changeMonth(inEvent, inMoveNext)
{
    if (inMoveNext) {
        currentMonth++;
        if (currentMonth > 11) {
            currentYear++;
            currentMonth = 0;
        }
    } else {
        currentMonth--;
        if (currentMonth < 0) {
            currentYear--;
            currentMonth = 11;
        }
    }

    // create the new calendar
    setupMonth(currentMonth, currentYear);

    // set date selections as necessary
    hiliteToday();

    hiliteDate(""); // clear selected date

    selectedDateID = idFromMonthDate (currentMonth, currentDate); // note: will be invalid if
currentDate is 0

    currentDate = 0; // clear the selected date

    updateTiles();
}



function hiliteToday() {
    var hilite = document.getElementById("todayHilite");

    if ((currentMonth == date.getMonth()) && (currentYear == date.getFullYear())) {
        var c = document.getElementById("calendar");
        var today = document.getElementById(idFromMonthDate(currentMonth, date.getDate()));
```

```
        today.setAttribute("class", dayClassFromDayIndex(getDayFromDate(date)) + " today");

        hilite.style.top = (today.offsetTop+c.offsetTop)-1+"px";
        hilite.style.left = today.offsetLeft+0+"px";
        hilite.style.visibility = "visible";
    }
    else {
        hilite.style.visibility = "hidden";
    }
}



function hiliteDate (inDate)
{
    var hilite = document.getElementById("dateHilite");

    if (inDate == "")
    {
        currentDate = 0;    // clear the selected date; don't parse empty string
    }
    else
    {
        currentDate = parseInt(inDate.substr(2,2),10);  // set the selected date
    }

    if ((idFromMonthDate(date.getMonth(), date.getDate()) == inDate) || (inDate==""))
    {
        // the hilite is over today's date.  hide it
        hilite.style.visibility = "hidden";
    }
    else
    {
        var c = document.getElementById("calendar");
        var aDate = document.getElementById(inDate);

        hilite.style.top = (aDate.offsetTop+c.offsetTop)+"px";
        hilite.style.left = aDate.offsetLeft+1+"px";
        hilite.style.visibility = "visible";
    }

    updateTiles();
}



function jumpToToday (inEvent)
{
    date = new Date();
    currentMonth = date.getMonth();
    currentYear = date.getFullYear();
    setupMonth(currentMonth, currentYear);
    hiliteToday();
    hiliteDate(""); // clear selected date
    currentDate = date.getDate();
    updateTiles();
    selectedDateID = idFromMonthDate (currentMonth, currentDate);
```

```
}



function selectDate (inEvent, inID)
{
    hiliteDate (inID);

    selectedDateID = inID;

    inEvent.stopPropagation ();
    inEvent.preventDefault ();
}



function fetchIPrefFirstWeekday () // returns integer
//
// Fetch data from this widget's plugin, from International Prefs.  Do fallback work if no
plugin available.
//
// Get the first-day-of-week, 0-based index.
//
{
    if (window.Calendar)    // plugin available
    {
        return Calendar.firstWeekday ();
    }
    else                    // fallback case
    {
        return 0;
    }
}



function fetchIPrefNarrowDayOfWeekName (inDayOfWeekIndex) // returns string
//
// Fetch data from this widget's plugin, from International Prefs.  Do fallback work if no
plugin available.
//
// Get the (localized customized in International Pref Pane) "narrow" day-of-week, which is
usually a single
// character such as "S" for Sunday, etc.  The inDayOfWeek input is zero-based and follows USA
first-day-of-week
// conventions, even if the first-day-of-week is not Sunday for the current localized settings.
Thus: 0=Sunday,
// 1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday, and 6=Saturday, in a
single-character format.
//
{
    var returnString = "x";

    if (window.Calendar)    // plugin available
    {
        returnString = Calendar.localizedNarrowDayOfWeekNameForIndex (inDayOfWeekIndex);
    }
```

```
    else                    // fallback case
    {
        switch (inDayOfWeekIndex)
        {
            case 0:   returnString = "S";   break;
            case 1:   returnString = "M";   break;
            case 2:   returnString = "T";   break;
            case 3:   returnString = "W";   break;
            case 4:   returnString = "T";   break;
            case 5:   returnString = "F";   break;
            case 6:   returnString = "S";   break;
        }
    }

    return returnString;
}



function fetchIPrefShortMixedcaseDayOfWeekName (inDayOfWeekIndex) // returns string
//
// Fetch data from this widget's plugin, from International Prefs.  Do fallback work if no
plugin available.
//
// Get the (localized customized in International Pref Pane) mixed-case short day name,
// zero-based.  For example, 0="Sun.", 1="Mon.", etc.
//
{
    var returnString = "x";

    if (window.Calendar)    // plugin available
    {
        returnString = Calendar.localizedShortDayOfWeekNameForIndex (inDayOfWeekIndex);
    }
    else                    // fallback case
    {
        switch (inDayOfWeekIndex)
        {
            case 0:   returnString = "Sun";   break;
            case 1:   returnString = "Mon";   break;
            case 2:   returnString = "Tue";   break;
            case 3:   returnString = "Wed";   break;
            case 4:   returnString = "Thu";   break;
            case 5:   returnString = "Fri";   break;
            case 6:   returnString = "Sat";   break;
        }
    }

    return returnString;
}



function fetchIPrefShortCapitalizedMonthName (inMonthIndex) // returns string
//
// Fetch data from this widget's plugin, from International Prefs.  Do fallback work if no
plugin available.
```

```
//
// Get the (localized customized in International Pref Pane) short month name, zero-based.
// For example, 0="JAN", 1="FEB", etc.
//
{
    var returnString = "x";

    if (window.Calendar)    // plugin available
    {
        // get string from plugin
        returnString = Calendar.localizedShortMonthNameForIndex (inMonthIndex);

        // get locale from plugin, and if US English, then return a version forced to all-caps
        if (Calendar.currentLocaleLanguageCode () == "en")
            returnString = returnString.toUpperCase ();
    }
    else                    // fallback case
    {
        switch (inMonthIndex)
        {
            case 0:    returnString = "JAN";   break;
            case 1:    returnString = "FEB";   break;
            case 2:    returnString = "MAR";   break;
            case 3:    returnString = "APR";   break;
            case 4:    returnString = "MAY";   break;
            case 5:    returnString = "JUN";   break;
            case 6:    returnString = "JUL";   break;
            case 7:    returnString = "AUG";   break;
            case 8:    returnString = "SEP";   break;
            case 9:    returnString = "OCT";   break;
            case 10:   returnString = "NOV";   break;
            case 11:   returnString = "DEC";   break;
        }
    }

    return returnString;
}


function setFontSizeToFit (inElement, inStartingFontSize, inStartingTop, inMaxWidth,
inMaxHeight)
//
// Sets the font size of the given element, while trying to honor that the string contained it
fit to the
// given size.
//
// The starting font size and top position parameters are given by the caller so that the
setting always
// proceeds from an absolute basis.  If the requested font size cannot be honored, then the
font size and
// top position in the element are repeatedly adjusted until the element's inner text fits (up
to a limit).
//
// NOTE: The CSS declaration for the given element must not have explicitly specified a "width"
property.
// NOTE: The height parameter is useful even for "single line" cases to prevent wrapping to an
```

```
undesired
// additional line.
//
{
    var measuredWidth;
    var measuredHeight;
    var newFontSize = inStartingFontSize;
    var newTop = inStartingTop;

    while (1)
    {
        inElement.style.setProperty ("font-size", newFontSize + "px", "important");
        inElement.style.setProperty ("top", newTop + "px", "important");
        measuredWidth = parseInt (document.defaultView.getComputedStyle (inElement,
null).getPropertyValue ("width"), 10);
        measuredHeight = parseInt (document.defaultView.getComputedStyle (inElement,
null).getPropertyValue ("height"), 10);
        if ((measuredWidth > inMaxWidth) || (measuredHeight > inMaxHeight))
        {
            if (newFontSize <= 6)
            {
                break;
            }
            newFontSize -= 1;
            newTop += 0.75;
        }
        else
        {
            break;
        }
    }
}



function updateTiles()
{
    var leftTopElement      = document.getElementById ("day");
    var leftBottomElement   = document.getElementById ("month");
    var rightElement        = document.getElementById ("date");
    var changeLeftTopStr    = undefined;
    var changeLeftBottomStr = undefined;
    var changeRightStr      = undefined;

    if (isCollapsed || (!currentDate && (currentMonth == date.getMonth()) && (currentYear ==
date.getFullYear())))
    {
        var today = date.getDate();
        changeLeftTopStr = fetchIPrefShortMixedcaseDayOfWeekName(getDayFromDate(date));
        changeLeftBottomStr = fetchIPrefShortCapitalizedMonthName(date.getMonth());
        changeRightStr = today;
        rightElement.setAttribute("class", "tileCommon "+(((today>9) &&
(today<20))?"tileDateTight":"tileDate"));
    }
    else
    {
        if (currentDate)
```

```
        {
            var aDate = new Date(currentYear, currentMonth, currentDate);
            changeLeftTopStr = fetchIPrefShortMixedcaseDayOfWeekName(getDayFromDate(aDate));
            changeLeftBottomStr = fetchIPrefShortCapitalizedMonthName(currentMonth);
            changeRightStr = currentDate;
            rightElement.setAttribute("class", "tileCommon "+(((currentDate>9) &&
(currentDate<20))?"tileDateTight":"tileDate"));
            // NOTE: in the above line currentDate may be invalid at this time
        }
        else
        {
            changeLeftBottomStr = fetchIPrefShortCapitalizedMonthName(currentMonth);
            changeRightStr = currentYear;
            rightElement.setAttribute("class", "tileCommon tileSmallYear");
        }
    }

    if (changeLeftTopStr != undefined)          // typically the day name or nothing
    {
        leftTopElement.style.visibility = "hidden";
        leftTopElement.innerText = changeLeftTopStr;
        setFontSizeToFit (leftTopElement, 18, 18, 52, 40);
        leftTopElement.style.visibility = "visible";
    }
    else
    {
        leftTopElement.style.visibility = "hidden";
    }

    if (changeLeftBottomStr != undefined)       // typically the month name
    {
        leftBottomElement.style.visibility = "hidden";
        leftBottomElement.innerText = changeLeftBottomStr;
        setFontSizeToFit (leftBottomElement, 27, 34, 53, 40);
        leftBottomElement.style.visibility = "visible";
    }

    if (changeRightStr != undefined)            // either the date-in-month number or
year-number
    {
        rightElement.innerText = changeRightStr;
        // we will not do a squeeze-to-fit adjust on the right tile's date number (beyond what
was done
        // by maybe picking a different style, above); squeezeFontAsNeeded will not work with
explictly
        // specified width in the css anyway
    }
}



function mouseDown(event, tag)
{
    event.target.src = "Images/"+tag+"_pressed.png";
    event.stopPropagation();
    event.preventDefault();
}
```

```
function mouseUp (inEvent, inTag)
{
    inEvent.target.src = "Images/"+inTag+".png";

    if (inTag == "bar_left")
    {
        changeMonth(inEvent, false);
    }
    else if (inTag == "bar_right")
    {
        changeMonth(inEvent, true);
    }
    else    // inTag == bar_middle
    {
        jumpToToday (inEvent);
    }

    inEvent.stopPropagation();
    inEvent.preventDefault();
}



function mouseOut (event, tag)
{
    event.target.src = "Images/"+tag+".png";
}



function toggleGridView(doItSlow)
{
    var midDiv = document.getElementById("calMid");
    var timeNow = (new Date).getTime();
    var multiplier = (doItSlow ? 10 : 1); // enable slo-mo
    var startingSize = parseInt(midDiv.clientHeight,10);

    resizeAnimation.element = midDiv;
    if (resizeAnimation.timer != null) // it is moving... change to new size
    {
        clearInterval(resizeAnimation.timer);
        resizeAnimation.timer = null;
        resizeAnimation.duration -= (timeNow - resizeAnimation.startTime);
        resizeAnimation.positionFrom = resizeAnimation.positionNow;
    }
    else
    {
        resizeAnimation.duration = 250 * multiplier;
        resizeAnimation.positionFrom = startingSize;
    }

    var resizeTo = isCollapsed ? 210 : 80;  // ack!!!!
    if (isCollapsed && window.widget)
        window.resizeTo (177, 244);
```

```
    resizeAnimation.positionTo = parseInt(resizeTo, 10); // lots of hard coding, yum...
    resizeAnimation.startTime = timeNow - 13; // set it back one frame.
    resizeAnimation.onfinished = toggleGridViewAnimFinished;

    resizeAnimation.element.style.height = startingSize;
    resizeAnimation.timer = setInterval ("animate();", 13);
    animate();
    isCollapsed = !isCollapsed;
}



function onRequestChangeGridView (inEvent)
{
    toggleGridView (inEvent.shiftKey);
}



function setupUpdateTimer ()
{
    if (updateDateTimer != null)
    {
        clearTimeout (updateDateTimer);
        updateDateTimer = null;
    }

    // create a new timer for when we need to update the date
    var hourMillis = ((23 - date.getHours()) * 3600000);
    var minsMillis = ((59 - date.getMinutes()) * 60000);
    var secsMillis = ((60 - date.getSeconds()) * 1000); // overshoot a second
    var millisRemaining = hourMillis + minsMillis + secsMillis;

    updateDateTimer = setTimeout("updateDate();", millisRemaining);
}



function updateDate() {
    var currentEqualDate = date.getMonth() == currentMonth &&
                           date.getFullYear() == currentYear &&
                           date.getDate() == currentDate;
    date = new Date();

    if (currentEqualDate)
    {
        // only change the current vars if they matched the date
        // before it was updated
        currentMonth = date.getMonth();
        currentYear = date.getFullYear();
        currentDate = date.getDate();
    }

    setupMonth(currentMonth, currentYear);
    hiliteToday();
    updateTiles();
```

```
    // reset the timer
    setupUpdateTimer();
}



function onshow () {
    // refresh the date, in case it's changed
    updateDate();
}



function onhide () {
    if (updateDateTimer) {
        clearTimeout(updateDateTimer);
        updateDateTimer = null;
    }
}



function onLoaded()
{
    // Register for notifications:
    //
    if (window.widget)
    {
        widget.onhide = onhide;
        widget.onshow = onshow;
    }

    // Obtain persistent pref settings; default to Grid View if no pref defined:
    //
    var viewSize = "grid";
    if (window.widget)
    {
        var pref = widget.preferenceForKey (createKey("collapsed"));
        if (pref != undefined)
        {
            if (pref == "true")
            {
                viewSize = "collapsed";
            }
        }
    }

    // Set up the global "isCollapsed", the midDiv size, and the total window size
    // on startup:
    //
    switch (viewSize)
    {
    case "collapsed":
        isCollapsed = true;
        document.getElementById ("calMid").style.height = "80px";
        window.resizeTo (177, 114);
```

```
    break;

    case "grid":
        isCollapsed = false;
        document.getElementById ("calMid").style.height = "210px";
        // window.resizeTo (177, 400);  Do not resize like the collapsed case above, not only
        // because it is not needed, but also because we do not want to interrupt the possibly-
        // going-on-right-now ripple animation.
    break;
    }

    // Draw things:
    //
    setupMonth (currentMonth, currentYear);
    updateTiles ();
    hiliteToday ();
}



function onremove ()
{
    widget.setPreferenceForKey (null, createKey("collapsed"));
}



function createKey (key)
{
    return widget.identifier + "-" + key;
}



if (window.widget)
{
    widget.onremove = onremove;
}
```