/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function to calculate dates for holidays.

Input date and whether you need actual

or observed dates that fall on weekends.

~~Comment out specific floating & fixed holidays~~

~~Below to remove from consideration in JS. To remove Easter~~

~~& relational Holidays see instructions in comments~~

~~in respective sections.~~

**No longer need to edit JS code. Use case module to set/clear holidays that are derived from the JS code.**

**The 3 modules, the case, set, and clear modules can be moved to parent IVRs for cases where varying**

**BU’s have different holidays (i.e. integrators, etc). Test Holiday is now fixed at 12/25/2022 so you only**

**need to set the testmode to 1 to test functionality.**

The observed value can now be set to 3 condistions.

0 – Only actual holidays are set

1 – Actual and observed holidays are set (Friday prior to Saturday and Monday following Sunday).

2 – only Monday following Sunday is set to accommodate customer that are open on Saturdays and  
 and don’t observe on Fridays.

Originally written by Greg Thomas Jr. Enhancements and

maintenance by Greg Thomas Sr (greg.thomassr@five9.com)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

We need to be able to get the week number of the month

This does that.

\*/

Date.prototype.getMonthWeek = function(){

var firstDayOfWeek = new Date(this.getFullYear(), this.getMonth(), 1).getDay();

return Math.ceil((this.getDate() + firstDayOfWeek)/7);

}

/\*

find what date the nth occurrence of y day falls on

\*/

function nthWeekdayOfMonth(weekday, n, idate) {

var imonth = idate;

var idate = new Date(date.getFullYear(), imonth, 1),

add = (weekday - idate.getDay() + 7) % 7 + (n - 1) \* 7;

// don't leave current month! Related to occurrence value of 5 below

do {

idate.setMonth(imonth);

idate.setDate(1 + add);

add -= 7;

} while (idate.getMonth() != imonth);

idate.setDate(idate.getDate());

idate.setFullYear(year);

return idate;

}

/\*

Variable declarations -- don't change.

\*/

var date = dateInput;

var week = date.getMonthWeek();

var day = date.getDay();

var month = date.getMonth();

var year = date.getFullYear();

var currentDate = date.getDate();

var isHoliday = false;

// variables used for observedInput

var observed = observedInput;

var thisdate=dateInput;

var thisday=thisdate.getDay();

/\*

Floating Holidays: Count month from 0, occurrence from 1, dayOfWeek from 0.

Month is the month that the holiday occurrs in

Occurrence is the nth occurrence of the day of week

dayOfWeek is the day of the week, 0=sunday 6=saturday.

IE, for the 3rd Monday in January, month=0, occurrence=3, dayOfWeek=1

For last occurrence of a day in a month, use a number >= 5

\*/

var floatingHolidays = {

"holiday": [

{"name":"MLK Day", "month":"0", "occurrence":"3","dayOfWeek":"1"},

{"name":"Fathers Day", "month":"5", "occurrence":"3", "dayOfWeek":"0"},

{"name":"Presidents Day", "month":"1","occurrence":"3","dayOfWeek":"1"},

{"name":"Mothers Day","month":"4","occurrence":"2","dayOfWeek":"0"},

{"name":"Memorial Day","month":"4","occurrence":"5","dayOfWeek":"1"},

{"name":"Parents Day","month":"6","occurrence":"3","dayOfWeek":"0"},

{"name":"Grandparents Day","month":"8","occurrence":"2","dayOfWeek":"0"},

{"name":"Labor Day","month":"8","occurrence":"1","dayOfWeek":"1"},

{"name":"Columbus Day","month":"9","occurrence":"2","dayOfWeek":"1"},

{"name":"Thanksgiving","month":"10","occurrence":"4","dayOfWeek":"4"},

{"name":"Thanksgiving Eve","month":"10","occurrence":"4","dayOfWeek":"3"},

]

};

/\*

For fixed holidays, the key is month,daynumber with month counting from zero and day counting from 1

Value of the key-value pair is the name of the holiday

For example, February 28th - fake holiday - would be input as "1,28":"fake holiday"

\*/

var fixedHolidays = {

"0,1": "New Years Day",

"1,14": "Valentines Day",

"3,15": "Tax Day",

"5,19": "Juneteenth",

"6,4": "Independence Day",

"10,11": "Veterans Day",

"11,24": "Christmas Eve",

"11,25": "Christmas Day",

"11,31": "New Years Eve",

// "1,1": "Test Holiday"

}

/\*

These values are hard coded

\*/

var relationalHolidays = {

"holiday": [

{"name":"Black Friday","relatedTo":"Thanksgiving","offset":"1"},

{"name":"Good Friday","relatedTo":"Easter","offset":"-2"},

]

}

// Attach dates to objects

function setHolidayDates() {

var i;

for (i = 0; i < Object.keys(floatingHolidays.holiday).length; i++) {

var hdate = nthWeekdayOfMonth(floatingHolidays.holiday[i].dayOfWeek,floatingHolidays.holiday[i].occurrence,floatingHolidays.holiday[i].month);

floatingHolidays.holiday[i].date = hdate;

if (floatingHolidays.holiday[i].name == "Thanksgiving") {

var rdate = new Date(hdate.valueOf());

rdate.setDate(rdate.getDate()+1);

relationalHolidays.holiday[0].date = rdate;

}

}

}

// Detect if input date is holiday

function getHoliday(month, week, day, currentDate) {

setHolidayDates();

var easterDate = getEaster(year);

//

// \*\*\*\*\* Comment out below line to disable Easter \*\*\*\*\*

//

floatingHolidays.holiday.push({"name":"Easter","date":easterDate});

var gfDate = new Date(easterDate.valueOf());

gfDate.setDate(gfDate.getDate()-2);

relationalHolidays.holiday[1].date = gfDate;

var i;

for (i = 0; i < Object.keys(floatingHolidays.holiday).length; i++) {

var tempVar = JSON.stringify(floatingHolidays.holiday[i]);

var holidayMonth = floatingHolidays.holiday[i].date.getMonth();

var holidayDate = floatingHolidays.holiday[i].date.getDate();

// console.log(`${tempVar} ${holidayMonth}`);

if (holidayMonth == month && holidayDate == currentDate) {

// console.log(`Floating holiday detected! ${floatingHolidays.holiday[i].name}`);

isHoliday = true;

return {"isHoliday":"true","holiday":floatingHolidays.holiday[i].name};

break;

}

// console.log(`${tempVar}`);

}

if (isHoliday == false) {

for (i = 0; i < Object.keys(relationalHolidays.holiday).length; i++) {

// console.log(relationalHolidays.holiday[i].date);

var holidayMonth = relationalHolidays.holiday[i].date.getMonth();

var holidayDate = relationalHolidays.holiday[i].date.getDate();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\* Comment out below "if" to remove relational Holidays \*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*No Relational Holidays!!\*/

if (holidayMonth == month && holidayDate == currentDate) {

return {"isHoliday":"true","holiday":relationalHolidays.holiday[i].name};

// console.log(`Relational Holiday Detected! ${relationalHolidays.holiday[i].name}`);

}

}

}

if (fixedHolidays[month + "," + currentDate] != null) {

// console.log(`Static holiday detected! ${fixedHolidays[month + "," + currentDate]}`);

isHoliday = true;

return {"isHoliday":"true","holiday":fixedHolidays[month + "," + currentDate]};

}

/\*

check if Observed set to 1 or 2.If 1, and not a holiay and current day is Friday, set date to tomorrow, if Observed

set, current day is Monday, set date to yesterday. If 2 and not a holiay and current day is Friday,do not set date to tomorrow, if Observed

set, current day is Monday, set date to yesterday.

\*/

if (isHoliday == false) {

if (observed==1) {

if (thisday==5) {

currentDate++

}

if (thisday==1) {

currentDate--

}

}else if (observed==2) {

if (thisday==1) {

currentDate--

}

}

if (fixedHolidays[month + "," + currentDate] != null) {

// console.log(`Static holiday detected! ${fixedHolidays[month + "," + currentDate]}`);

isHoliday = true;

return {"isHoliday":"true","holiday":fixedHolidays[month + "," + currentDate]};

}

}

if (isHoliday == false) {

return {"isHoliday":"false"};

}

}

/\*

This calculates easter.

\*/

function getEaster(year) {

var f = Math.floor,

// Golden Number - 1

G = year % 19,

C = f(year / 100),

// related to Epact

H = (C-f(C/4)-f((8\*C+13)/25)+19\*G+15)%30,

// number of days from 21 March to the Paschal full moon

I = H - f(H/28) \* (1 - f(29/(H + 1)) \* f((21-G)/11)),

// weekday for the Paschal full moon

J = (year + f(year / 4) + I + 2 - C + f(C / 4)) % 7,

// number of days from 21 March to the Sunday on or before the Paschal full moon

L = I - J,

month = 3 + f((L + 40)/44),

day = L + 28 - 31 \* f(month / 4);

month -= 1;

var easterDate = new Date(year,month,day);

return easterDate;

}

// Execute code

var currentHoliday = getHoliday(month, week, day, currentDate);

if (typeof currentHoliday.holiday == "undefined") currentHoliday.holiday = "No Holiday";

//console.log(currentHoliday);

//process.exit;

return currentHoliday;