

## all files / solar-calc/lib/ sun.js

94.66% Statements 195/206 73.21% Branches 41/56 100% Functions 32/32 95.26% Lines 181/190

```

1      "use strict";
2
3      2x    var _createClass = (function () { function defineProperties(target, props) { for (var key in props) { var prop = props[key]; prop.configura
4
5      30x    var _classCallCheck = function (instance, Constructor) { 1 if (!(instance instanceof Constructor)) { throw new TypeError("Cannot call a cl
6
7      1x    var Sun = (function () {
8      1x    function Sun(date, latitude, longitude) {
9      30x    _classCallCheck(this, Sun);
10
11     30x    this.date = date;
12     30x    this.latitude = latitude;
13     30x    this.longitude = longitude;
14
15     30x    this.julianDate = getJD(date);
16     }
17
18     1x    _createClass(Sun, {
19     solarNoon: {
20     get: function () {
21     2x    return calcSolNoon(this.julianDate, this.longitude, this.date);
22     }
23     },
24     timeAtAngle: {
25     value: function timeAtAngle(angle, rising) {
26     24x    return calcSunriseSet(rising, angle, this.julianDate, this.date, this.latitude, this.longitude);
27     }
28     }
29     });
30
31     1x    return Sun;
32   })();
33
34     1x    var formatDate = function formatDate(date, minutes) {
35     14x    var seconds = (minutes - Math.floor(minutes)) * 60;
36     14x    return new Date(Date.UTC(date.getFullYear(), date.getMonth(), date.getDate(), 0, minutes, seconds));
37   };
38
39     1x    function calcTimeJulianCent(jd) {
40     1112x    var T = (jd - 2451545) / 36525;
41     1112x    return T;
42   }
43
44     1x    function isLeapYear(yr) {
45     12x    return yr % 4 === 0 && yr % 100 !== 0 || yr % 400 === 0;
46   }
47
48     1x    function calcDoyFromJD(jd) {
49     12x    var z = Math.floor(jd + 0.5);
50     12x    var f = jd + 0.5 - z;
51     12x    var A;
52     12x    if (z < 2299161) {
53     A = z;
54   } else {
55     12x    var alpha = Math.floor((z - 1867216.25) / 36524.25);
56     12x    A = z + 1 + alpha - Math.floor(alpha / 4);
57   }
58     12x    var B = A + 1524;
59     12x    var C = Math.floor((B - 122.1) / 365.25);
60     12x    var D = Math.floor(365.25 * C);
61     12x    var E = Math.floor((B - D) / 30.6001);
62     12x    var day = B - D - Math.floor(30.6001 * E) + f;
63     12x    var month = E < 14 ? E - 1 : E - 13;
64     12x    var year = month > 2 ? C - 4716 : C - 4715;
65
66     12x    var k = isLeapYear(year) ? 1 : 2;
67     12x    var doy = Math.floor(275 * month / 9) - k * Math.floor((month + 9) / 12) + day - 30;
68     12x    return doy;
69   }
70
71     1x    function radToDeg(angleRad) {
72     3328x    return 180 * angleRad / Math.PI;
73   }
74
75     1x    function degToRad(angleDeg) {
76     16648x    return Math.PI * angleDeg / 180;
77   }
78
79     1x    function calcGeomMeanLongSun(t) {
80     2220x    var L0 = 280.46646 + t * (36000.76983 + t * 0.0003032);
81     2220x    while (L0 > 360) {
82     34998x    L0 -= 360;
83   }
84     2220x    while (L0 < 0) {
85     L0 += 360;

```

```

86     }
87     2220x    return L0; // in degrees
88 }
89
90     1x    function calcGeomMeanAnomalySun(t) {
91     2220x        var M = 357.52911 + t * (35999.05029 - 0.0001537 * t);
92     2220x        return M; // in degrees
93     }
94
95     1x    function calcEccentricityEarthOrbit(t) {
96     1112x        var e = 0.016708634 - t * (0.000042037 + 1.267e-7 * t);
97     1112x        return e; // unitless
98     }
99
100    1x    function calcSunEqOfCenter(t) {
101    1108x        var m = calcGeomMeanAnomalySun(t);
102    1108x        var mrad = degToRad(m);
103    1108x        var sinm = Math.sin(mrad);
104    1108x        var sin2m = Math.sin(mrad + mrad);
105    1108x        var sin3m = Math.sin(mrad + mrad + mrad);
106    1108x        var C = sinm * (1.914602 - t * (0.004817 + 0.000014 * t)) + sin2m * (0.019993 - 0.000101 * t) + sin3m * 0.000289;
107    1108x        return C; // in degrees
108    }
109
110    1x    function calcSunTrueLong(t) {
111    1108x        var l0 = calcGeomMeanLongSun(t);
112    1108x        var c = calcSunEqOfCenter(t);
113    1108x        var l = l0 + c;
114    1108x        return l; // in degrees
115    }
116
117    1x    function calcSunApparentLong(t) {
118    1108x        var o = calcSunTrueLong(t);
119    1108x        var omega = 125.04 - 1934.136 * t;
120    1108x        var lambda = o - 0.00569 - 0.00478 * Math.sin(degToRad(omega));
121    1108x        return lambda; // in degrees
122    }
123
124    1x    function calcMeanObliquityOfEcliptic(t) {
125    2220x        var seconds = 21.448 - t * (46.815 + t * (0.00059 - t * 0.001813));
126    2220x        var e0 = 23 + (26 + seconds / 60) / 60;
127    2220x        return e0; // in degrees
128    }
129
130    1x    function calcObliquityCorrection(t) {
131    2220x        var e0 = calcMeanObliquityOfEcliptic(t);
132    2220x        var omega = 125.04 - 1934.136 * t;
133    2220x        var e = e0 + 0.00256 * Math.cos(degToRad(omega));
134    2220x        return e; // in degrees
135    }
136
137    1x    function calcSunDeclination(t) {
138    1108x        var e = calcObliquityCorrection(t);
139    1108x        var lambda = calcSunApparentLong(t);
140
141    1108x        var sint = Math.sin(degToRad(e)) * Math.sin(degToRad(lambda));
142    1108x        var theta = radToDeg(Math.asin(sint));
143    1108x        return theta; // in degrees
144    }
145
146    1x    function calcEquationOfTime(t) {
147    1112x        var epsilon = calcObliquityCorrection(t);
148    1112x        var l0 = calcGeomMeanLongSun(t);
149    1112x        var e = calcEccentricityEarthOrbit(t);
150    1112x        var m = calcGeomMeanAnomalySun(t);
151
152    1112x        var y = Math.tan(degToRad(epsilon) / 2);
153    1112x        y *= y;
154
155    1112x        var sin2l0 = Math.sin(2 * degToRad(l0));
156    1112x        var sinm = Math.sin(degToRad(m));
157    1112x        var cos2l0 = Math.cos(2 * degToRad(l0));
158    1112x        var sin4l0 = Math.sin(4 * degToRad(l0));
159    1112x        var sin2m = Math.sin(2 * degToRad(m));
160
161    1112x        var Etime = y * sin2l0 - 2 * e * sinm + 4 * e * y * sinm * cos2l0 - 0.5 * y * y * sin4l0 - 1.25 * e * e * sin2m;
162    1112x        return radToDeg(Etime) * 4; // in minutes of time
163    }
164
165    1x    function calcHourAngle(angle, lat, solarDec) {
166    1108x        var latRad = degToRad(lat);
167    1108x        var sdRad = degToRad(solarDec);
168    1108x        var HAarg = Math.cos(degToRad(90 + angle)) / (Math.cos(latRad) * Math.cos(sdRad)) - Math.tan(latRad) * Math.tan(sdRad);
169    1108x        var HA = Math.acos(HAarg);
170    1108x        return HA; // in radians (for sunset, use -HA)
171    }
172
173    1x    function isNumber(inputVal) {
174    1084x        var oneDecimal = false;
175    1084x        var inputStr = "" + inputVal;
176    1084x        for (var i = 0; i < inputStr.length; i++) {
177    1479x            var oneChar = inputStr.charAt(i);

```

```

178 1479x 1 if (i === 0 && (oneChar === "-" || oneChar === "+")) {
179      continue;
180  }
181 1479x  if (oneChar === "." && !oneDecimal) {
182    24x    oneDecimal = true;
183    24x    continue;
184  }
185 1455x  if (oneChar < "0" || oneChar > "9") {
186    1060x    return false;
187  }
188  }
189    24x  return true;
190 }
191
192 function getJD(date) {
193     30x var year = date.getFullYear();
194     30x var month = date.getMonth() + 1;
195     30x var day = date.getDate();
196
197     30x var A = Math.floor(year / 100);
198     30x var B = 2 - A + Math.floor(A / 4);
199     30x var JD = Math.floor(365.25 * (year + 4716)) + Math.floor(30.6001 * (month + 1)) + day + B - 1524.5;
200     30x return JD;
201 }
202
203 1x function calcSolNoon(jd, longitude, date) {
204     2x var tnoon = calcTimeJulianCent(jd - longitude / 360);
205     2x var eqTime = calcEquationOfTime(tnoon);
206     2x var solNoonOffset = 720 - longitude * 4 - eqTime; // in minutes
207     2x var newt = calcTimeJulianCent(jd + solNoonOffset / 1440);
208     2x eqTime = calcEquationOfTime(newt);
209     2x var solNoonLocal = 720 - longitude * 4 - eqTime; // in minutes
210     2x while (solNoonLocal < 0) {
211         solNoonLocal += 1440;
212     }
213     2x while (solNoonLocal >= 1440) {
214         solNoonLocal -= 1440;
215     }
216     2x return formatDate(date, solNoonLocal);
217     // return timeString(solNoonLocal, 3);
218 }
219
220 1x function dayString(jd) {
221     12x 1 if (jd < 900000 || jd > 2817000) {
222         return "error";
223     } else {
224         12x var z = Math.floor(jd + 0.5);
225         12x var f = jd + 0.5 - z;
226         12x var A;
227         12x 1 if (z < 2299161) {
228             A = z;
229         } else {
230             12x var alpha = Math.floor((z - 1867216.25) / 36524.25);
231             12x A = z + 1 + alpha - Math.floor(alpha / 4);
232         }
233         12x var B = A + 1524;
234         12x var C = Math.floor((B - 122.1) / 365.25);
235         12x var D = Math.floor(365.25 * C);
236         12x var E = Math.floor((B - D) / 30.6001);
237         12x var day = B - D - Math.floor(30.6001 * E) + f;
238         12x var month = E < 14 ? E - 1 : E - 13;
239         12x var year = month > 2 ? C - 4716 : C - 4715;
240         12x return new Date(Date.UTC(year, month - 1, day, 0, 0, 0));
241     }
242 }
243
244 1x function calcSunriseSetUTC(rise, angle, JD, latitude, longitude) {
245     1108x var t = calcTimeJulianCent(JD);
246     1108x var eqTime = calcEquationOfTime(t);
247     1108x var solarDec = calcSunDeclination(t);
248     1108x var hourAngle = calcHourAngle(angle, latitude, solarDec);
249     //alert("HA = " + radToDeg(hourAngle));
250     1108x if (!rise) hourAngle = -hourAngle;
251     1108x var delta = longitude + radToDeg(hourAngle);
252     1108x var timeUTC = 720 - 4 * delta - eqTime; // in minutes
253     1108x return timeUTC;
254 }
255
256 1x function calcSunriseSet(rise, angle, JD, date, latitude, longitude)
257 // rise = 1 for sunrise, 0 for sunset
258 {
259     24x var timeUTC = calcSunriseSetUTC(rise, angle, JD, latitude, longitude);
260     24x var newTimeUTC = calcSunriseSetUTC(rise, angle, JD + timeUTC / 1440, latitude, longitude);
261     24x if (isNaN(newTimeUTC)) {
262
263         12x return formatDate(date, newTimeUTC);
264     } else {
265         // no sunrise/set found
266         12x var doy = calcDoyFromJD(JD);
267         12x var jdy;
268         12x 1 if (latitude > 66.4 && doy > 79 && doy < 267 || latitude < -66.4 && (doy < 83 || doy > 263)) {
269             //previous sunrise/next sunset

```

```
270 12x    jdy = calcJDofNextPrevRiseSet(!rise, rise, angle, JD, latitude, longitude);
271 12x    return dayString(jdy);
272      } else {
273          //previous sunset/next sunrise
274          jdy = calcJDofNextPrevRiseSet(rise, rise, angle, JD, latitude, longitude);
275          return dayString(jdy);
276      }
277  }
278 }
279
280 1x function calcJDofNextPrevRiseSet(next, rise, type, JD, latitude, longitude) {
281 12x     var julianday = JD;
282 12x     var increment = next ? 1 : -1;
283
284 12x     var time = calcSunriseSetUTC(rise, type, julianday, latitude, longitude);
285 12x     while (!isNumber(time)) {
286 1048x         julianday += increment;
287 1048x         time = calcSunriseSetUTC(rise, type, julianday, latitude, longitude);
288     }
289
290 12x     return julianday;
291 }
292
293 1x module.exports = Sun;
```

Code coverage generated by [istanbul](#) at Mon Oct 17 2016 15:34:35 GMT-0400 (EDT)