| PROPERTY | DESCRIPTION |
|---|---|
| E | Euler's constant *e*, which is the base of a natural logarithm; this value is approximately 2.7182818284590452354 |
| LN10 | The natural logarithm of 10, which is approximately 2.302585092994046 |
| LN2 | The natural logarithm of 2, which is approximately 0.6931471805599453 |
| LOG10E | The base-10 logarithm of *e*, the base of the natural logarithms; this value is approximately 0.4342944819032518 |
| LOG2E | The base-2 logarithm of *e*, the base of the natural logarithms; this value is approximately 1.4426950408889634 |
| PI | A constant representing the ratio of the circumference of a circle to its diameter, which is approximately 3.1415926535897932 |
| SQRT1_2 | The square root of 1/2, which is approximately 0.7071067811865476 |
| SQRT2 | The square root of 2, which is approximately 1.4142135623730951 |

Table 7-7: `Math` class properties

As an example of how to use the properties of the `Math` object, the following code shows how to use the `PI` property to calculate the area of a circle based on its radius. The code also uses the `pow()` method to calculate the radius raised to the second power, and the `round()` method to round the value returned to the nearest whole number.

```
var radius = 25;
var area = Math.PI * Math.pow(radius, 2);
var roundedArea = Math.round(area); // returns 1963
```

The design of the Outer Orbits reservation page includes a countdown timer like those traditionally used for space launches, which calculates the days, hours, minutes, and seconds until the selected flight takes off. You'll assign the current date and time to a variable, and the date and time of the selected launch to another variable. You'll then use the `Math.floor()` method to determine the whole number of days, hours, minutes, and seconds between the two.

**To calculate the days, hours, minutes, and seconds until launch:**

1. Return to the **orbits.js** document in your text editor, and then above the `createEventListeners()` function, add the following code to create the `updateCountdown()` function and declare its variables:

```
1   function updateCountdown() {
2       var dateToday = new Date();
3       var dateFrom = Date.UTC(dateToday.getFullYear(),↵
```

```
            dateToday.getMonth(), dateToday.getDate(),↵
            dateToday.getHours(), dateToday.getMinutes(),↵
            dateToday.getSeconds());
    var dateTo = Date.UTC(dateObject.getFullYear(),↵
            dateObject.getMonth(), dateObject.getDate(),↵
            19, 0, 0); // all launches at 8:00pm UTC
}
```

The first statement sets the value of `dateToday` to the current date and time. The second creates a `dateFrom` variable containing the current year, month, date, hours, minutes, and seconds. The third statement creates a `dateTo` variable with a value containing the year, month, and date selected by the user, along with an hour of 19, a minute of 0, and a second of 0 to reflect that all launches take place at exactly 8:00 pm UTC.

**2.** Within the `updateCountdown()` function, add the following code:

```
// days
var daysUntil = Math.floor((dateTo - dateFrom) / 86400000);
document.getElementById("countdown").innerHTML = daysUntil;
```

Both the `dateTo` and `dateFrom` values are expressed in a value in milliseconds. The first statement calculates the difference in milliseconds between these two time values, and then divides it by 86400000 (the number of milliseconds in a day) to calculate the number of days between the two dates. The statement uses the `Math.floor()` method to convert the value to only the whole number portion of the difference. The second statement places the `daysUntil` value in the countdown element on the page.

**3.** Below the statements you entered in the previous step, add the following code:

```
// hours
var fractionalDay = (dateTo - dateFrom) % 86400000;
var hoursUntil = Math.floor(fractionalDay / 3600000);
if (hoursUntil < 10) {
   hoursUntil = "0" + hoursUntil;
}
document.getElementById("countdown").innerHTML +=↵
   ":" + hoursUntil;
```

The `fractionalDay` variable uses the modulus (`%`) operator to find the remainder from calculating the number of days. To calculate the `hoursUntil` variable, the `fractionalDay` variable is divided by 3600000 (the number of microseconds in an hour), and the `Math.floor()` method again provides just the whole number portion

of the result. The final `if` statement checks if the number of minutes is a single digit and if so, appends a 0 to the start of the number; this is to represent the format generally seen in digital clocks. The final statement appends the `hoursUntil` value to the existing days value in the countdown element, with a colon between the two.

**4.** Below the statements you entered in the previous step, add the following code:

```
// minutes
var fractionalHour = fractionalDay % 3600000;
var minutesUntil = Math.floor(fractionalHour / 60000);
if (minutesUntil < 10) {
   minutesUntil = "0" + minutesUntil;
}
document.getElementById("countdown").innerHTML +=↵
   ":" + minutesUntil;
// seconds
var fractionalMinute = fractionalHour % 60000;
var secondsUntil = Math.floor(fractionalMinute / 1000);
if (secondsUntil < 10) {
   secondsUntil = "0" + secondsUntil;
}
document.getElementById("countdown").innerHTML +=↵
   ":" + secondsUntil;
```

This code uses the same sets of statements used to calculate the hours, dividing by 60000 (the number of microseconds in a minute) to calculate the remaining difference in minutes, and 1000 (the number of microseconds in a second) to calculate the remaining difference in seconds. Your `updateCountdown()` function should match the following:

```
function updateCountdown() {
   var dateToday = new Date();
   var dateFrom = Date.UTC(dateToday.getFullYear(),↵
         dateToday.getMonth(), dateToday.getDate(),↵
         dateToday.getHours(), dateToday.getMinutes(),↵
         dateToday.getSeconds());
   var dateTo = Date.UTC(dateObject.getFullYear(),↵
         dateObject.getMonth(), dateObject.getDate(),↵
         19, 0, 0); // all launches at 8:00pm UTC
   // days
```

```
11      var daysUntil = Math.floor((dateTo - dateFrom) / 86400000);
12      document.getElementById("countdown").innerHTML = daysUntil;
13      // hours
14      var fractionalDay = (dateTo - dateFrom) % 86400000,
15          hoursUntil = Math.floor(fractionalDay / 3600000);
16      if (hoursUntil < 10) {
17         hoursUntil = "0" + hoursUntil;
18      }
19      document.getElementById("countdown").innerHTML +=↵
20         ":" + hoursUntil;
21      // minutes
22      var fractionalHour = fractionalDay % 3600000,
23          minutesUntil = Math.floor(fractionalHour / 60000);
24      if (minutesUntil < 10) {
25         minutesUntil = "0" + minutesUntil;
26      }
27      document.getElementById("countdown").innerHTML +=↵
28         ":" + minutesUntil;
29      // seconds
30      var fractionalMinute = fractionalHour % 60000,
31          secondsUntil = Math.floor(fractionalMinute / 1000);
32      if (secondsUntil < 10) {
33         secondsUntil = "0" + secondsUntil;
34      }
35      document.getElementById("countdown").innerHTML +=↵
36         ":" + secondsUntil;
37   }
```

**5.** At the top of the document, in the global variables section, add the following statement:

```
var countdown;
```

**6.** Within the `selectDate()` function, just before the closing `}`, enter the following statements:

```
countdown = setInterval(updateCountdown, 1000);
document.getElementById("countdownSection").style.display =↵
   "block";
```

The first statement uses the `setInterval()` method of the `Window` object to call the `updateCountdown()` function when a user selects a valid date, and to repeatedly call the function every second (every 1000 milliseconds). This simulates the behavior of a digital timer. The `setInterval()` method is set as the value of the global countdown variable so it can be cancelled later. The second statement makes the web page element containing the counter visible.

**7.** Within the `updateCountdown()` function, just before the `// days` comment, enter the following code:

```
1   if ((dateTo - dateFrom) < 1000) { // time will be less than 0↵
2      when setInterval runs next
3      clearInterval(countdown);
4      document.getElementById("countdownSection").style.↵
5         display = "none";
6   }
```

This `if` statement checks if the time left will be less than 0 on the next update. If so, it uses the `clearInterval()` method to clear the interval referenced by the `countdown` variable.

**8.** Save your changes to **orbits.js**, shift-refresh or shift-reload **booktrip.htm** in your browser to clear the form, click the **Pick a date** box, and then click a future date. As Figure 7-11 shows, the countdown timer is displayed on the right side of the page, and the time value changes once per second.


Figure 7-11: Countdown timer