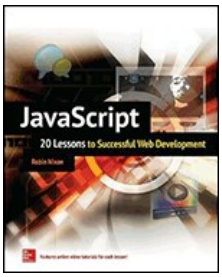

JavaScript: 20 Lessons to Successful Web Development

by Robin Nixon

Appendix B: Common JavaScript Functions

This appendix lists the most common functions and properties in JavaScript. The list is not fully comprehensive, however, as there are still a number of complex functions and properties that are beyond the scope of this book, due to requiring advanced programming techniques.

If you are interested in seeing what they are, though, you can download the addedbytes.com JavaScript cheat sheet at the following URL to act as a good starting point: tinyurl.com/jsfuncs.

Arithmetic Functions

- **Math.abs(a)** Returns a as a positive number (or 0).
- **Math.acos(a)** Returns the arc cosine of a.
- **Math.asin(a)** Returns the arc sine of a.
- **Math.atan(a)** Returns the arc tangent of a.
- **Math.atan2(a, b)** Returns the arc tangent of a / b.
- **Math.ceil(a)** Rounds up to return the integer closest to a.
- **Math.cos(a)** Returns the cosine of a.
- **Math.exp(a)** Returns the exponent of a (Math.E to the power a).
- **Math.floor(a)** Rounds down to return the integer closest to a.
- **Math.log(a)** Returns the log of a base e.
- **Math.max(a,b)** Returns the maximum of a and b.
- **Math.min(a,b)** Returns the minimum of a and b.
- **Math.pow(a,b)** Returns a to the power b.
- **Math.random()** Returns a pseudo-random number with a value of 0 or greater, but less than 1.
- **Math.round(a)** Rounds up or down to return the integer closest to a.
- **Math.sin(a)** Returns the sine of a.
- **Math.sqrt(a)** Returns the square root of a.
- **Math.tan(a)** Returns the tangent of a.

Array Functions

- **array.concat(a2[, a3 …])** Returns a new array comprising array joined with a2 and optionally more arrays.
- **array.every(c[, o])** Tests all elements of array using the callback function c, optionally using the object o as this when executing the callback.
- **array.filter(c[, o])** Creates a new array with all elements in array that pass the test implemented by the function c, optionally using the object o as this when executing the callback.
- **array.forEach(c[, o])** Calls callback function c for all elements in array, optionally using the object o as this when executing the callback.
- **array.indexOf(s[, i])** Returns the first element of array that matches s, optionally starting at element index i (otherwise starting at element 0).

- **array.join(s)** Returns a string comprising all elements in array, optionally joined to each other with the separator in string s (otherwise separated with a comma).
- **array.lastIndexOf(s[, i])** Returns the last element of array that matches s, optionally working backward from element index i (otherwise starting at the end and working backward).
- **array.map(c[, o])** Returns a new array comprising each element of array being passed to callback function c, optionally using the object o as this when executing the callback.
- **array.pop()** Pops off the last element from array and returns it.
- **array.push(e1[, e2 …])** Pushes element e1 (and optionally additional elements) to the end of array.
- **array.reduce(c[, i])** Returns a single value determined by applying callback function c sequentially to each pair of elements in array, optionally using i as the initial argument to the first call of the callback. For example, if the callback function performs an addition and the array's contents are [1,2,3], then the result returned will be 6 (1 + 2 + 3).
- **array.reduceRight(c[, i])** Same as reduce(), but array is processed in the opposite direction.
- **array.reverse()** Returns array in reversed element order.
- **array.shift()** Removes the first element from array and returns it.
- **array.slice(s[, e])** Returns a new array comprising a selection of elements from array starting at index s and optionally ending at index e (otherwise ending at the array end).
- **array.some(c[, o])** Tests whether at least one element of array passes the test in callback function c, optionally using the object o as this when executing the callback.
- **array.toSource()** Returns a string representing the source code of array—not currently available in Internet Explorer or Safari.
- **array.sort(f)** Returns array sorted using optional function f; otherwise, the array is sorted in case-sensitive, ascending alphabetical order.
- **array.splice(i, n[, e1 …])** Returns an array extracted from array starting at index i, containing n elements from the array, and optionally adding the element e1 (or more new elements).
- **array.toString()** Returns a string representing the elements in array separated with commas.
- **array.unshift(e1[, e2 …])** Adds e1 (and optionally more elements) to the start of array, returning the new length of the array.

Boolean Functions

- **Boolean(n)** Returns number n as a Boolean number.
- **object.toSource()** Returns a string representing the source code of object—not currently available in Internet Explorer or Safari.
- **object.toString()** Returns a string of either true or false depending on the value of object.
- **object.valueOf()** Returns the primitive value of object.

Date Functions

- **Date()** Returns a new Date object.
- **date.getDate()** Returns the day of the month for date according to local time, between 1 and 31.
- **date.getDay()** Returns the day of the week for date according to local time, between 0 for Sunday and 6 for Saturday.
- **date.getFullYear()** Returns the year for date according to local time, as a four-digit number for years 1000–9999, such as 2018.

- **date.getHours()** Returns the hour from date according to local time, between 0 and 23.
- **date.getMilliseconds()** Returns the milliseconds from date according to local time, between 0 and 999.
- **date.getMinutes()** Returns the minutes from date according to local time, between 0 and 59.
- **date.getMonth()** Returns the month from date according to local time, between 0 for January and 11 for December.
- **date.getSeconds()** Returns the seconds from date according to local time, between 0 and 59.
- **date.getTime()** Returns the time in milliseconds since January 1, 1970, as 00:00:00 UTC.
- **date.getTimezoneOffset()** Returns the difference, in minutes, between UTC and local time (can be negative, zero, or positive).
- **date.getUTCDate()** Returns the day of the month from date according to Universal Time, between 1 and 31.
- **date.getUTCDay()** Returns the day of the week from date according to Universal Time, between 0 for Sunday and 6 for Saturday.
- **date.getUTCFullYear()** Returns the year from date according to Universal Time as a four-digit number for years 1000–9999, such as 2018.
- **date.getUTCHours()** Returns the hour from date according to Universal Time, between 0 and 23.
- **date.getUTCMilliseconds()** Returns the milliseconds from date according to Universal Time, between 0 and 999.
- **date.getUTCMonth()** Returns the month from date according to Universal Time, between 0 for January and 11 for December.
- **date.getUTCSeconds()** Returns the seconds from date according to Universal Time, between 0 and 59.
- **date.getYear() (Deprecated—use getFullYear() instead)** Returns the year from date as a two-digit number.
- **date.setDate(d)** Sets the day of the month in date according to local time, where d is between 1 and 31.
- **date.setFullYear(y[, m[, d]])** Sets the year in date according to local time as a full year, such as 2018 in y, optionally passing the month between 0 and 11 in m, and the day between 1 and 31 in d.
- **date.setHours(h[, m[, s[, ms]]])** Sets the hour in date according to local time, between 0 and 23 in h, optionally passing the minutes between 0 and 59 in m, the seconds between 0 and 59 in s, and the milliseconds between 0 and 999 in ms.
- **date.setMilliseconds(ms)** Sets the milliseconds in date according to local time, where ms is between 0 and 999.
- **date.setMinutes(m[, s[, ms]])** Sets the minutes in date according to local time, between 0 and 59 in m, optionally passing the seconds between 0 and 59 in s, and the milliseconds between 0 and 999 in ms.
- **date.setMonth(m[, d])** Sets the month in date according to local time in m (from 0 for January to 11 for December), optionally passing the day (from 1 to 31) in d.
- **date.setSeconds(s[, ms])** Sets the seconds in date according to local time, between 0 and 59 in s, optionally passing the seconds in milliseconds in ms.
- **date.setTime(t)** Sets the time in date in milliseconds since January 1, 1970, at 00:00:00 UTC in t.
- **date.setUTCDate(d)** Sets the day of the month in date according to Universal Time, between 1 and 31 in d.
- **date.setUTCFullYear(y[, m[, d]])** Sets the year in date according to Universal Time, as a full year, such as 2018 in y, optionally passing the month between 0 and 11 in m, and the day between 1 and 31 in d.
- **date.setUTCHours(h[, m[, s[, ms]]])** Sets the hour in date according to Universal Time, between 0 and 23 in h, optionally passing the minutes between 0 and 59 in m, the seconds between 0 and 59 in s, and the milliseconds between 0 and 999 in ms.
- **date.setUTCMilliseconds(ms)** Sets the milliseconds in date according to Universal Time, between 0 and 999 in ms.

- **date.setUTCMinutes(m[, s[, ms]])** Sets the minutes in date according to Universal Time, between 0 and 59 in m, optionally passing the seconds between 0 and 59 in s, and the milliseconds between 0 and 999 in ms.
- **date.setUTCMonth(m[, d])** Sets the month in date according to Universal Time, between 0 for January and 11 for December in m, optionally passing the day between 1 and 31 in d.
- **date.setUTCSeconds(s[, ms])** Sets the seconds in date according to Universal Time, between 0 and 59 in s, optionally passing the milliseconds between 0 and 999 in ms.
- **date.setYear(**y**) (Deprecated—use setFullYear() instead)** Sets the year to a value from 1900 to 1999, where y is a value between 0 and 99.
- **date.toDateString()** Returns the date from date according to local time, in human readable form in American English.
- **date.toGMTString() (Deprecated—use toUTCString() instead)** Returns the date from date according to local time, using Internet GMT conventions.
- **date.toLocaleDateString()** Returns the date from date according to local time, using the locale's conventions of the operating system.
- **date.toLocaleFormat(f)** Returns the date from date according to local time, using the formatting specified in f (in the same format expected by the strftime() function in C)—not currently available in Internet Explorer or Safari.
- **date.toLocaleString()** Returns the date from date according to local time, using the locale's conventions of the operating system.
- **date.toLocaleTimeString()** Returns the time portion of the date from date according to local time, using the current locale's conventions.
- **date.toSource()** Returns a string representing the source of date—not currently available in Internet Explorer or Safari.
- **date.toString()** Returns the date (as a string) from date.
- **date.toTimeString()** Returns the time portion of a date (as a string) from date.
- **date.toUTCString()** Returns the date from date in the UTC time zone.
- **date.valueOf()** Returns the primitive value of date as the number of milliseconds since midnight on January 1, 1970, UTC.

DOM Functions

- **document.createElement(t)** Creates a new element with the tag name t.
- **document.getElementById(i)** Returns the DOM object of the element with the id of i.
- **document.getElementsByTagName(t)** Returns all elements matching the tag name in t.
- **document.write(s)** Writes the value(s) in s to the browser; does not work in XHTML documents.

Global Functions

- **clearInterval(h)** Clears the regular interrupts created by setInterval() using the handle in h.
- **clearTimeout(h)** Clears the interrupt created by setTimeout() using the handle in h.
- **decodeURI(u)** Returns the URI encoded string u as an unencoded string.
- **decodeURIComponent(u)** Returns the URI component encoded string u as an unencoded string.
- **encodeURI(u)** Returns the string u in URI encoded form without encoding URI reserved characters that have special meaning.
- **encodeURIComponent(u)** Returns the string u in URI encoded form and encodes any characters that have special

meaning in URIs, such as ,, /, ?, :, @, &, =, +, $, and #.

- **escape(s) (Deprecated—use encodeURI() or encodeURIComponent() instead)** Returns s encoded by escaping special characters.
- **eval(e)** Returns the result of evaluating the expression in e.
- **function.call(a1[, a2 …])** Calls the function function passing any number of optional arguments.
- **isFinite(v)** Returns true if v is a finite, legal number, or false if it is infinite or NaN.
- **isNaN(v)** Returns true if the value v is not a number; otherwise, returns false.
- **parseFloat(s)** Returns the string s as a floating point number.
- **parseInt(s)** Returns the string s as an integer.
- **setInterval(c, m)** Sets up repeating interrupts calling the code in c every m milliseconds; returns a handle that can be used to clear the interrupts.
- **setTimeout(c, m)** Sets up a single interrupt to call the code in c in m milliseconds; returns a handle that can be used to clear the interrupt.
- **unescape(s) (Deprecated – use decodeURI() or decodeURIComponent() instead)** Returns the escaped string s as an unescaped string.

Number Functions

- **Number(s)** Returns string s as a number.
- **number.constructor()** Returns the function that created this instance of number; by default, this is the Number object.
- **number.toExponential(n)** Returns a string representing number in exponential notation with n representing the number of digits after the decimal point.
- **number.toFixed(n)** Formats number with n digits to the right of the decimal point.
- **number.toLocaleString()** Returns a string value version of number in a format that may vary according to a browser's locale settings.
- **number.toPrecision(n)** Returns a string representing number to the specified precision n.
- **number.toString(n)** Returns a string representation of number in the specified radix (base) in n.
- **number.valueOf()** Returns the primitive value of number.

Regular Expression Functions

- **regex.exec(s)** Returns an array of matches found in the string s using the regular expression regex, or returns null if no matches were made.
- **regex.test(s)** Tests the string s using the regular expression regex, returning true if a match is found, otherwise false.
- **regex.toSource()** Returns a string representing the source code of regex—not currently available in Internet Explorer or Safari.
- **regex.toString()** Returns a string representation of regex in the form of a regular-expression literal.

String Functions

- **String(n)** Returns number n as a string.
- **string.charAt(n)** Returns the character in string at index n.

- **string.charCodeAt(n)** Returns a number indicating the Unicode value of string at index n.
- **string.concat(s1[, s2 …])** Concatenates string with s1 (and more strings if passed) and returns a new single string.
- **string.indexOf(s[, i])** Returns the index in string of the search string s, optionally starting at i.
- **string.lastIndexOf(s[, i])** Returns the index in string of the last occurrence of search string s, optionally starting at i.
- **string.localeCompare(s)** Returns a number indicating whether string comes before or after (or is the same) as s in sort order.
- **string.match(e)** Returns one or more matches (depending on whether the g modifier is used in the expression) for the regular expression e in the string string.
- **string.replace(e, s)** Finds a match between regular expression e and string string.
- **string.search(e)** Returns the index of the first location of regular expression e in string string.
- **string.slice(s[, e])** Extracts a section of string starting at index s, and optionally ending at e (otherwise ending at the string end).
- **string.split(s[, l])** Returns an array comprising sections of string split at separator s, optionally limited to the number of occurrences specified in l.
- **string.substr(s[, l])** Returns a section of string starting from index s, and optionally limited to the number of characters in l; otherwise, all characters to the end of the string are returned.
- **string.substring(f, t)** Returns a substring of string starting from index f and ending with the character immediately preceding index t.
- **string.toLocaleLowerCase()** Returns a lowercase version of string according to the current locale.
- **string.toLocaleUpperCase()** Returns an uppercase version of string according to the current locale.
- **string.toLowerCase()** Returns a lowercase version of string.
- **string.toString()** Returns a string representing string.
- **string.toUpperCase()** Returns an uppercase version of string.
- **string.valueOf()** Returns the primitive value of string.

Window Functions

- **alert(v)** Pops up an alert window displaying the value(s) in v.
- **confirm(t)** Pops up an alert window displaying the value in t and supplying two options: “OK” that returns true if clicked, and “Cancel” that returns false if clicked.
- **window.blur()** Removes focus from window.
- **window.close()** Closes window.
- **window.focus()** Gives focus to window.
- **window.open(u, n[, f[, r]])** Opens a new window using the URL in u and giving it the name in n. Optionally, the string f defines features such as height and width using a range of key/value pairs, and (optionally) either true or false in r if the window is to replace the current value in the browser’s history (ignored by some browsers).
- **window.print()** Opens a dialog for printing window to a printer.
- **window.scroll(x, y)** The same as scrollTo() (see later).
- **window.scrollBy(x, y)** Scrolls the window by x pixels horizontally and y pixels vertically.

- **element.scrollIntoView(a)** Scrolls the element element into view, aligned to the window top if the optional argument a is true, otherwise aligned to the bottom.
- **window.scrollTo(x, y)** Scrolls the window to the offset from the top left of the page of x pixels horizontally and y pixels vertically.