JavaScript and AJAX
2nd Edition

Tutorial 2

Working with Operators and Expressions

Objectives

- Work with event handlers
- Insert a value into a Web form field
- Create and work with date objects
- Extract information from date objects
- Work with arithmetic, unary, conditional, and logical operators
- Understand the properties and methods of the Math object
- Understand how JavaScript works with numeric values
- Run time-delayed and timed-interval commands

Introducing onevent Processing

- An **event** is an action that occurs within a Web browser or Web document.
- An **event handler** is a statement that tells browsers what code to run in response to the specified event.
  - Script

```
<input type = "button" value = "Total Cost" onclick = "calcTotal()" />
```


Introducing onevent Processing

Category	Event Handler	Description
Window and document event handlers	onload	The browser has completed loading the document.
	onunload	The browser has completed unloading the document.
	onerror	An error has occurred in the JavaScript program.
	onmove	The user has moved the browser window.
	onresize	The user has resized the browser window.
	onscroll	The user has moved the scroll bar within the browser window.
Form event handlers	onfocus	The user has entered an input field.
	onblur	The user has exited an input field.
	onchange	The content of an input field has changed.
	onselect	The user has selected text within an input or text area field.
	onsubmit	The user has submitted the Web form.
	onreset	The user has reset the Web form.
Mouse and keyboard event handlers	onkeydown	The user has pressed a key.
	onkeypress	The user has pressed and released a key.
	onclick	The user has clicked the mouse button.
	ondblclick	The user has double-clicked the mouse button.
	onmousedown	The user has pressed the mouse button.
	onmouseup	The user has released the mouse button.
	onmousemove	The user has moved the pointer over the element.
	onmouseout	The user has moved the pointer off of the element.

Introducing onevent Processing

- To insert an event handler as an element attribute, use the syntax

  `<element onevent = "script"> ...`

  where *element* is the Web page element, *event* is the name of an event associated with the element, and *script* is a command to be run in response to the event

Working with Date Objects

- **Date objects** contain information about a specified date and time
- To store a date and time in a variable, use the JavaScript command

  variable = new Date("month day, year hours:minutes:seconds")

  where variable is the name of the variable that contains the date object, and month, day, year, hours, minutes, and seconds indicate the date and time to be stored in the object.
- Time values are entered in 24-hour time

Working with Date Objects

- To store a date and time using numeric values, use the JavaScript command

  variable = new Date(year, month, day, hours, minutes, seconds)

  where year, month, day, hours, minutes, and seconds are the values of the date and time, and the month value is an integer from 0 to 11, where 0 = January, 1 = February, and so forth.
- Time values are entered in 24-hour time.

Working with Date Objects

- To create a date object containing the current date and time, use the following JavaScript command:

  variable = new Date()

Working with Date Objects

- **Date methods** can be used to retrieve information from a date object or to change a date object's value

Method	Retrieves	Value (when the thisDate variable stores the date object for "June 15, 2011 14:35:28")
thisDate.getSeconds()	Retrieves the seconds value	28
thisDate.getMinutes()	Retrieves the minutes value	35
thisDate.getHours()	Retrieves the hours value (in 24-hour time)	14
thisDate.getDate()	Retrieves the day of the month value	15
thisDate.getDay()	Retrieves the day of the week value (0 = Sunday, 1 = Monday, 2 = Tuesday, 3 = Wednesday, 4 = Thursday, 5 = Friday, 6 = Saturday)	3
thisDate.getMonth()	Retrieves the month value (0 = January, 1 = February, 2 = March, etc.)	5
thisDate.getFullYear()	Retrieves the four-digit year value	2011
thisDate.getTime()	Retrieves the time value, as expressed in milliseconds, since January 1, 1970	1,308,166,505,000

Working with Date Objects

Method	Description
DateObject.setSeconds(*value*)	Sets the seconds value of *DateObject* to *value*
DateObject.setMinutes(*value*)	Sets the minutes value of *DateObject* to *value*
DateObject.setHours(*value*)	Sets the hours value of *DateObject* to *value*
DateObject.setDate(*value*)	Sets the day of the month value of *DateObject* to *value*
DateObject.setMonth(*value*)	Sets the month number of *DateObject* to *value* (0 = January, 1 = February, etc.)
DateObject.setFullYear(*value*)	Sets the four-digit year value of *DateObject* to *value*
DateObject.setTime(*value*)	Sets the time of *DateObject* in milliseconds since January 1, 1970

Working with Operators and Operands

- An **operator** is a symbol used to act upon an item or a variable within a JavaScript expression.
- The variables or expressions that operators act upon are called **operands.**

Operator	Description	Example
+	Combines or adds two items	Men = 20; Women = 25; Total = Men + Women;
–	Subtracts one item from another	Income = 1000; Expense = 750; Profit = Income – Expense;
*	Multiplies two items	Width = 50; Length = 20; Area = Width * Length;
/	Divides one item by another	Persons = 50; Cost = 200; CostPerPerson = Cost / Persons;
%	Calculates the remainder after dividing one value by another	TotalEggs = 64; CartonSize = 12; EggsLeft = TotalEggs % CartonSize;

Working with Operators and Operands

- **Binary operators** work with two operands in an expression.
- **Unary operators** work with one operand.
- **Increment operators** increase the value of the operand by 1.
  - x++;
- **Decrement operators** decrease the value of the operand by 1.
  - x--;

Working with Operators and Operands

- **Assignment operators** assign values to items.
  - Equal sign (=)
    - x = x + y
  - A common use of the += operator is to concatenate strings or add a value to an existing value of a variable

```
quote = "To be or not to be: ";
quote += "That is the question. ";
quote += "Whether 'tis nobler in the mind to suffer ";
quote += "the slings and arrows of outrageous fortune, ";
quote += "Or to take arms against a sea of troubles";
quote += "And by opposing end them.";
...
```


Working with Operators and Operands

Operator	Description	Example	Equivalent To
=	Assigns the value of the expression on the right to the expression on the left	x = y	
+=	Adds two expressions	x += y	x = x + y
-=	Subtracts the expression on the right from the expression on the left	x -= y	x = x – y
*=	Multiplies two expressions	x *= y	x = x * y
%=	Calculates the remainder from dividing the expression on the left by the expression on the right	x %= y	x = x % y

Working with the Math Object and Math Methods

- The **Math object** is an object that can be used for performing mathematical tasks and storing mathematical values.
  - **Math methods** store functions used for performing advanced calculations and mathematical operations such as:
    - Generating random numbers
    - Extracting square roots
    - Calculating trigonometric values

Working with the Math Object and Math Methods

Math Method	Returns
Math.abs(*x*)	the absolute value of *x*
Math.acos(*x*)	the arc cosine of *x* in radians
Math.asin(*x*)	the arc sine of *x* in radians
Math.atan(*x*)	the arc tangent of *x* in radians
Math.atan2(*x*, *y*)	the angle between the x-axis and the point (*x*, *y*)
Math.ceil(*x*)	*x* rounded up to the next highest integer
Math.cos(*x*)	the cosine of *x*
Math.exp(*x*)	e^x
Math.floor(*x*)	*x* rounded down to the next lowest integer
Math.log(*x*)	the natural logarithm of *x*
Math.max(*x*, *y*)	the larger of *x* and *y*
Math.min(*x*, *y*)	the smaller of *x* and *y*
Math.pow(*x*, *y*)	x^y
Math.random()	a random number between 0 and 1
Math.round(*x*)	*x* rounded to the nearest integer
Math.sin(*x*)	the sine of *x*
Math.sqrt(*x*)	the square root of *x*
Math.tan(*x*)	the tangent of *x*

Working with the Math Object and Math Methods

- The Math object also stores numeric values for mathematical constants.

Math Constant	Description
Math.E	The natural logarithm base, *e* (approximately 2.7183)
Math.LN10	The natural logarithm of 10 (approximately 2.3026)
Math.LN2	The natural logarithm of 2 (approximately 0.6931)
Math.LOG10E	The base 10 logarithm of *e* (approximately 0.4343)
Math.LOG2E	The base 2 logarithm of *e* (approximately 1.4427)
Math.PI	The value π (approximately 3.1416)
Math.SQRT1_2	The value of 1 divided by the square root of 2 (approximately 0.7071)
Math.SQRT2	The value of the square root of 2 (approximately 1.4142)

Controlling How JavaScript Works with Numeric Values

- Some mathematical operations can return results that are not numeric values.
  - You cannot divide a number by a text string
    - Returns “NaN”
  - You cannot divide a number by zero
    - Returns “Infinity”
- The isNaN function is a Boolean function that tests a value to see whether it is numeric or not.
- The isFinite function is a Boolean function that checks for the value of Infinity.

Controlling How JavaScript Works with Numeric Values

Function or Method	Description
isFinite(*value*)	Returns a Boolean value indicating whether *value* is finite and a legal number
isNaN(*value*)	Returns a Boolean value, which has the value true if *value* is not a number
parseFloat(*string*)	Extracts the first numeric value from a text string
parseInt(*string*)	Extracts the first integer value from a text string
value.toExponential(*n*)	Returns a text string displaying *value* in exponential notation with *n* digits to the right of the decimal point
value.toFixed(*n*)	Returns a text string displaying *value* to *n* decimal places
value.toPrecision(*n*)	Returns a text string displaying *value* to *n* significant digits either to the left or to the right of the decimal point

Working with Conditional, Comparison, and Logical Operators

- A **conditional operator** is a ternary operator that tests whether a certain condition is true or not.
- A **comparison operator** is an operator that compares the value of one expression to another.

Operator	Description	Example
==	Returns true if the values are equal	x == y
!=	Returns true if the values are not equal	x != y
>	Returns true if the value on the left is greater than the value on the right	x > y
<	Returns true if the value on the left is less than the value on the right	x < y
>=	Returns true if the value on the left is greater than or equal to the value on the right	x >= y
<=	Returns true if the value on the left is less than or equal to the value on the right	x <= y

Working with Conditional, Comparison, and Logical Operators

- **Logical operators** allow you to connect several expressions

Operator	Description	Example (when x = 20 and y = 25)	Value
&&	Returns true when both expressions are true	(x == 20) && (y == 25)	TRUE
||	Returns true when at least one expression is true	(x == 20) || (y < 10)	TRUE
!	Returns true if the expression is false and false if the expression is true	!(x == 20)	FALSE

Running Timed Commands

- A **time-delayed command** is a JavaScript command that is run after a specified amount of time has passed.

```
setTimeout("command", delay);
clearTimeout();
```

- A **time-interval command** instructs the browser to run the same command repeatedly at specified intervals.

```
setInterval ("command", interval);
clearInterval();
```