```
// Anything following double slashes is an English-language comment.
// Read the comments carefully: they explain the JavaScript code.

// variable is a symbolic name for a value.
// Variables are declared with the var keyword:
var x; // Declare a variable named x.

// Values can be assigned to variables with an = sign
x = 0; // Now the variable x has the value 0
x // => 0: A variable evaluates to its value.

// JavaScript supports several types of values
x = 1; // Numbers.
x = 0.01; // Just one Number type for integers and reals.
x = "hello world"; // Strings of text in quotation marks.
x = 'JavaScript'; // Single quote marks also delimit strings.
x = true; // Boolean values.
x = false; // The other Boolean value.

x = null; // Null is a special value that means "no value".
x = undefined; // Undefined is like null.

// JavaScript's most important data type is the object.
// An object is a collection of name/value pairs, or a string to value map.
var book = { // Objects are enclosed in curly braces.
topic: "JavaScript", // The property "topic" has value "JavaScript".
fat: true // The property "fat" has value true.
}; // The curly brace marks the end of the object.

// Access the properties of an object with . or []:
book.topic // => "JavaScript"
book["fat"] // => true: another way to access property values.
book.author = "Flanagan"; // Create new properties by assignment.
book.contents = {}; // {} is an empty object with no properties.

// JavaScript also supports arrays (numerically indexed lists) of values:
var primes = [2, 3, 5, 7]; // An array of 4 values, delimited with [ and ].
primes[0] // => 2: the first element (index 0) of the array.
primes.length // => 4: how many elements in the array.
primes[primes.length-1] // => 7: the last element of the array.
primes[4] = 9; // Add a new element by assignment.
primes[4] = 11; // Or alter an existing element by assignment.
var empty = []; // [] is an empty array with no elements.
empty.length // => 0

// Arrays and objects can hold other arrays and objects:
var points = [ // An array with 2 elements.
{x:0, y:0}, // Each element is an object.
{x:1, y:1}
];

var data = { // An object with 2 properties
trial1: [[1,2], [3,4]], // The value of each property is an array.
trial2: [[2,3], [4,5]] // The elements of the arrays are arrays.
};

// Operators act on values (the operands) to produce a new value.
// Arithmetic operators are the most common:
3 + 2 // => 5: addition
```



```
3 - 2 // => 1: subtraction
3 * 2 // => 6: multiplication
3 / 2 // => 1.5: division
points[1].x - points[0].x // => 1: more complicated operands work, too
"3" + "2" // => "32": + adds numbers, concatenates strings

// JavaScript defines some shorthand arithmetic operators
var count = 0; // Define a variable
count++; // Increment the variable
count--; // Decrement the variable
count += 2; // Add 2: same as count = count + 2;
count *= 3; // Multiply by 3: same as count = count * 3;
count // => 6: variable names are expressions, too.

// Equality and relational operators test whether two values are equal,
// unequal, less than, greater than, and so on. They evaluate to true or false.
var x = 2, y = 3; // These = signs are assignment, not equality tests
x == y // => false: equality
x != y // => true: inequality
x < y // => true: less-than
x <= y // => true: less-than or equal
x > y // => false: greater-than
x >= y // => false: greater-than or equal
"two" == "three" // => false: the two strings are different
"two" > "three" // => true: "tw" is alphabetically greater than "th"
false == (x > y) // => true: false is equal to false

// Logical operators combine or invert boolean values
(x == 2) && (y == 3) // => true: both comparisons are true. && is AND
(x > 3) || (y < 3) // => false: neither comparison is true. || is OR
!(x == y) // => true: ! inverts a boolean value

// Functions are parameterized blocks of JavaScript code that we can invoke.
function plus1(x) { // Define a function named "plus1" with parameter "x"
return x+1; // Return a value one larger than the value passed in
} // Functions are enclosed in curly braces

plus1(y) // => 4: y is 3, so this invocation returns 3+1
var square = function(x) { // Functions are values and can be assigned to vars
return x*x; // Compute the function's value
}; // Semicolon marks the end of the assignment.
square(plus1(y)) // => 16: invoke two functions in one expression

// When functions are assigned to the properties of an object, we call
// them "methods". All JavaScript objects have methods:
var a = []; // Create an empty array
a.push(1,2,3); // The push() method adds elements to an array
a.reverse(); // Another method: reverse the order of elements
// We can define our own methods, too. The "this" keyword refers to the object
// on which the method is defined: in this case, the points array from above.
points.dist = function() { // Define a method to compute distance between points
var p1 = this[0]; // First element of array we're invoked on
var p2 = this[1]; // Second element of the "this" object
var a = p2.x-p1.x; // Difference in X coordinates
var b = p2.y-p1.y; // Difference in Y coordinates
return Math.sqrt(a*a + // The Pythagorean theorem
b*b); // Math.sqrt() computes the square root
};
points.dist() // => 1.414: distance between our 2 points
```