1 Expressions and Operators

JavaScript

2 Primary Expressions

- These primary expressions are reserved words:
    - true—literal that evaluates to boolean true
    - false—literal that evaluates to boolean false
    - null—literal that usually means an object has not yet been instantiated
    - this—the current object's reference to itself

3 Array Initializers

- An array initializer is an array literal in square [brackets], e.g.:
    - [] is an empty array with no elements
    - [3, 7] is an array with two elements
    - [[1, 2, 3], [4, 5, 6], [7, 8, 9]] is a 3 x 3 2-dimensional array
    - [1, , , , 5] is an array with five elements in which three of the elements are undefined

4 Invocation Expressions

- Invocations expressions are JavaScript's format for calling (invoking) a method or function
- The method or function simply is named with any arguments inside the required parentheses:
    - myFunction(5);
    - Math.max(3, 5, 7);
    - myArray.sort();

5 The Keyword new

- The keyword new followed by a *constructor method* is used to instantiate an object:
    - new Object();
    - new Car();

6 Assignment Operators

- Also known as op equals operators
- Assigns an *updated* value to a variable

Operator	Example	Explanation
+=	ctr += 1;	ctr = ctr + 1;
-=	ctr -= 17;	ctr = ctr - 17;
*=	ctr *= 8;	ctr = ctr * 8;
/=	ctr /= 5;	ctr = ctr / 5;

7 Unary Operators (Page 1)

- Unary operators update variables values by adding (increment operator) or subtracting (decrement operator) value of 1 to (or from)

Operator	Example	Explanation
++	ctr++;	ctr = ctr + 1; (post)
++	++ctr;	ctr = ctr + 1; (pre)
--	ctr--;	ctr = ctr - 1; (post)
--	--ctr;	ctr = ctr - 1; (pre)

8 □ **Unary Operators (Page 2)**

- If operator is a *prefix*, the value is returned *after* it is increased or decreased:
  - ▫ When the variable ctr = 5:
    newVar = ++ctr; // newVar will be 6
- If operator is a *suffix*, the value is returned *before*:
  - ▫ When the variable ctr = 5:
    newVar = ctr++; // newVar will be 5
- Final value of ctr in *both* cases will be 6

9 □ **The eval() Method**

- The eval() method evaluates a string to see if it is a valid JavaScript statement:
  - ▫ eval("var a = 17");

10 □ **The typeof Operator**

- A *unary* operator that returns a string that evaluates what is the type of any expression:
  - ▫ typeOf null // returns object
  - ▫ typeOf true // returns boolean
  - ▫ typeOf("hello") // returns string
    - May be placed inside (parentheses) to look like a method call

11 □ **Order of Evaluation**

- Operations of equal precedence in an expression are executed in left to right order
- A partial list of the order of precedence is:
  - ▫ All operations in (parentheses) first
  - ▫ Arithmetic multiply (*), divide (/), modulo (%)
  - ▫ Arithmetic add (+), subtract (-)
  - ▫ Conditional operators equal (==), not equal (!=), greater than (>), greater than or equal to (>=), less than (<), less than or equal to (<=)
  - ▫ Logical OR (||)
  - ▫ Logical AND (&&)