**Chapter 1 – Values, Types, and Operators**

* Types of Values
  + **Numbers – numeric values**
    - Fractional numbers are written using a dot (e.g. “9.81)
    - Scientific notation/exponential numbers can be written by adding an “e” followed by the exponent of the number (e.g. “2.998e8” = 2.990 x 10^8 = 299,800,000)
    - Arithmetic – same rules apply. MDAS
    - Special Numbers – three of them exist in JS. JS treats these as numbers but don’t behave like normal numbers:
      * Infinity (positive infinite)
      * -Infinity (negative infinite)
      * NaN (not a number)
  + **Strings – are represented by text**
    - “\” indicates a newline (e.g. “This is line one\And this is line two”)
    - In some situations where you want a backslash to be just a backslash, you put double backlashes (“\\”)
      * Example conde: console.log ("A newline character is written like \"\\n\".");
      * //A newline character is written like “\n”
    - Strings can be concatenated by using the “+” operator
    - Unary operators - some values are written as words. The “typeof” operator produced a string value naming the type of the value you give it.
    - This is in contract to binary operators which uses two
  + **Boolean Values – like yes/no, true/false, up/down, etc**
    - Comparisons – is one way to produce Boolean values
    - In strings uppercase letters are always considered “lesser than lowercase letters
    - Logical Operators - JS supports three logical operators: *and/&&*, *or/*||, and *not*/!
      * And operator is “&&” – binary operator; result is true if both values are true
      * Or operator is “||” – result is true if either values given are true
      * Not operator is “!” – is a unary operator that flips the value given to it
      * Precedence of operators - || has lowest, then &&, then comparison operators, and then the rest
      * Ternary logical operator – e.g. “console.log(true ? 1: 2);”
        + Value on left of the question picks which of the other two values comes out.
        + If true, middle value is choson
        + If false, rightmost value is chosen
  + **Undefined Values**
    - null
    - undefined
  + **Automatic Type conversion** 
    - JS does “type coercion”
      * console.log(null == undefined); // → true
      * console.log(null == 0); // → false
      * That last piece of behavior is often useful. When you want to test whether a value has a real value instead of null or undefined, you can simply compare it to null with the == (or !=) operator.
  + **Short circuiting of logical operators**
    - The logical operators && and || handle values of different types in a peculiar way. They will convert the value on their left side to Boolean type in order to decide what to do, but depending on the operator and the result of that conversion, they return either the original left-hand value or the right-hand value.
    - The || operator, for example, will return the value to its left when that can be converted to true and will return the value on its right otherwise. This conversion works as you’d expect for Boolean values and should do something analogous for values of other types.
    - console.log(null || "user") // → user console.log("Karl" || "user") // → Karl
    - This functionality allows the || operator to be used as a way to fall back on a default value. If you give it an expression that might produce an empty value on the left, the value on the right will be used as a replacement in that case.
    - The && operator works similarly, but the other way around. When the value to its left is something that converts to false, it returns that value, and otherwise it returns the value on its right.
    - Another important property of these two operators is that the expression to their right is evaluated only when necessary. In the case of true || X, no matter what X is—even if it’s an expression that does something terrible—the result will be true, and X is never evaluated. The same goes for false && X, which is false and will ignore X. This is called short-circuit evaluation.
    - The conditional operator works in a similar way. The first expression is always evaluated, but the second or third value, the one that is not picked, is not.