Building Blocks: Variables and Operators

2.1 What is a Variable?

Memory Locations

- A variable is a name for a storage location in the computer’s memory
- Like a mailbox
- The size of the mailbox corresponds to the datatype of the variable

Variable Names: Rules

- Cannot start with a numeral
- Cannot contain an operator (arithmetic, relational, or logical)
- Cannot use any punctuation
- No spaces…ever
- Cannot use JavaScript keywords
- Is case sensitive

Variable Names: Conventions

- Some begin with an indication of the datatype
  - `intNumber`
  - `stringName`
- CamelBack (or CamelCase) notation uses uppercase for first letter of a new word
  - `yourAge`
  - `myCar`
- Use underscores to separate words
  - `your_age`
  - `my_car`

Variable Names: Tips

- Be sure your variable name is not too long
  - You have to retype it!
  - `theFirstPlayerInTheGame` or
  - `playerOne`
- Use CamelBack or underscores for readability
  - `number1` or `number_1`
- Make variable names meaningful
  - `abc` or `first_name`

Declaring Variables

- Use the keyword `var`
- No need to define the datatype
- Can declare more than one variable on a single line
- Examples:
  - `var burger, chips, soda;`

  is same as:
  - `var burger;`
  - `var chips;`
  - `var soda;`

2.2 Datatypes

A Loosely Typed Language

- Strongly typed
  - Like C++ or Java
  - Static typing: type checking occurs when program is compiled
  - Once declared as a type, variable retains all properties of that type
- Loosely typed
  - Like JavaScript or PHP
  - Dynamic typing: type is checked as the program runs
  - The type can change as the program runs

Numbers

- All JavaScript numeric variables are stored as floats
- To create a numeric variable, use `var` keyword and give the variable a numeric value initially
- Examples:
  - `var x = 3;`
  - `var y = -2.983`

Strings and Characters

- A character is any of the keyboard characters
- A string is a group of characters
- To create a string variable, use var keyword and give the variable a string or character for an initial value. Enclose the value in quotes
- Examples:
  - `var name = “Joey”;`
  - `var name = ‘Joey’;`
  - `var age = “3”;`

Quotes

Nest quotes if your string contains quotes but…

use different quotes (double or single)

Examples:

(a) `var Joe = ‘Joe says, “Go team!” ’;`

display : `Joe says, “Go team!”`

(b) `var Joe = “Joe says, ‘Go team!’ “;`

display: `Joe says, ‘Go team!’`

(c) `var Joe = “Joe says, “Go team!” “;`

display: Joe says,

JavaScript sees first double quote as the beginning of the variable’s value and the next double quote is assumed to be the end

(d) `var Joe = “Joe says, \“Go team!\” ”;`

or `var Joe = 'Joe says, \'Go team!\' ';`

Named Constants

- Use for a value that will not change throughout program but may need to be changed at some future time
- Allows one change to affect whole program
- Usually variable names for named constants are all uppercase
- Example:

  Set a tax rate:

```
var TAX = 0.065;
```

2.3 Arithmetic Operators and Important Functions

The Modulus Operator

- Written as `x % y`
- Read as `x mod y`
- Means the integer remainder after dividing `x` by `y`
- Examples:
  - `15 % 2 = 1`
  - `23 % 7 = 2`
  - `18 % 3 = 0`

The Hierarchy of Operations or The Order of Precedence

1. Parentheses
2. Exponents
3. Multiplication, division, and modulus in order from left to right
4. Addition and subtraction in order from left to right

Note:

- Division is same as multiplying by the inverse 6/3 is same as 6 * 1/3
- Subtraction is same as adding the negative of a number 8 – 5 is same as 8 + (-5)

The Concatenation Operator

- Joins strings of text
- Is represented by + sign
- Example:

```
      var username = “lizzy”;
      var school = “myschool”;
      var domain = “edu”;
      var email = username + “@” + school + “.”
+ domain;
      document.write email;
```

Output is: <u>`lizzy@myschool.edu`</u>

Parsing Integers and Floating Point Numbers

- Numeric input always stored initially as text
- `parseInt()` changes a numeric text value to an integer
- `parseFloat()` changes a numeric text value to a float
- If input is not numeric, result is `NaN`

Examples : User enters `4.5, 6.7,` and `“hello”`

```
var num1 = prompt(“enter a number);
var num2 = prompt(“enter a number);
var num3 = prompt(“enter a number);
document.write(parseInt(num1) + “, “);
document.write(parseFloat(num2) + “, “);
document.write(parseFloat(num3);
```

Output is: `4, 6.7, NaN`

2.4 Relational Operators

ASCII Code

- ASCII: American Standard Code for Information Interchange
- Associates a character with a number from 0 - 127
- Examples:

  `“A”` in ASCII is

  `65` `“9”` in

  ASCII is `57`

  `“ ”` (space) in ASCII is

  `32` `var` **`name`** `=`

  `“Sam”`

  Stored in ASCII as `83, 97, 109`

Relational Operators

- `>` greater than
- `<` less than
- `>=` greater than or equal to
- `<=` less than or equal to
- `==` is the same as
- `!=` is not the same as

Result of using relational operators is always either

`true` or `false`

The Comparison Operator

- `==` is NOT the same as `=`
- `==` is a comparison operator. It asks, “is this thing the same as this other thing?”
- `=` is an assignment operator. It says “set the value of this variable to this value.”
- Example:

  `x` `= 5` sets the value of **`x`** to `5`

  `x` `== 5` asks if **`x`** has the value of `5`
- Example:

```
var x = 5;
document.write(“The value of x = 5 is “ + x + ”<br />”);
document.write(“The value of x == 8 is “ + (x == 8));
```

The output is:

```
The value of x = 5 is 5
The value of x == 8 is false
```

2.5
Logical Operators and the Conditional Operator

Logical Operators

- The result of an expression with logical operators is always either `true` or `false`
- The `AND` operator is written as `&&`
  - Always results in `false` unless both sides of the expression are `true`
- The `OR` operators is written as `||`
  - Always results in `true` unless both sides of the expression are `false`
- The `NOT` operator is written as `!`
  - It is `true` if the expression is `false` and `false` if the expression is `true`

Truth Table for `AND`, `OR`, and `NOT` Operators

X	Y	X || Y	X && Y	! X
true	true	true	true	false
true	false	true	false	false
false	true	true	false	true
false	false	false	false	true

Order of

	Description	Symbol
Arithmetic Operators are evaluated first in the order listed		
	1st: Parentheses	( )
	2nd: Exponents	^
	3rd: Multiplication / Division / Modulus	*, /, %
	4th: Addition / Subtraction	+ –
Relational Operators are evaluated second and all relational operators have the same precedence		
	Less than	<
	Less than or equal to	<=
	Greater than	>
	Greater than or equal to	>=
	The same as, equal to	==
	Not the same as	!=
Logical Operators are evaluated last in the order listed		
	1st: NOT	!
	2nd: AND	&&
	3rd: OR	||

Examples: Are the expressions `true` or `false`?

1. Given: **x** `= 6,` **y** `= 4,` **z** `= 18`

```
 (x * y) > z && (z > (x + y)
(6 * 4) > 18 && (18 > (6 + 4)

        24 > 18 &&  18 > 10
           True   &&   True results in True
```

2. Given: **x** `= 6,` **y** `= 4,` **z** `= 18`

```
!(2 * x + y == z - 2)
!(2 * 6 + 4 == 18 - 2)

 !( 16  == 16)
  !  (True) results in False
```

The Conditional Operator

This asks: Is one thing the same as another thing? If yes, do something. If not, do something else or do nothing.

Written like this:

```
varName = (condition) ? value_1 :
value_2;
```

The Conditional Operator: Examples

```
  1.   var password = “Open Sesame”;
       var userpw = prompt(“enter password:”);
       var message = (password == userpw) ? “You may
enter!” : “No entry permitted.”;
       document.write message;
  2.   var x = 3; var y = 15; var z = 5;
       var answer = prompt(“What is “ + y + “ divided
by “ + x + “?”);
       answer = parseFloat(answer);
       var message = (y/x == answer) ? “Correct” :
“Wrong”;
       document.write message;
```

The `charAt()` Function

- Returns the character at any specific location in a string
- Example:

```
var transport = “4-wheeler”;

   characters: 4      -  w  h  e  e  l  e  r
      index:0         1  2  3  4  5  6  7  8

transport.charAt(0) = “4”
transport.charAt(1) = “-”
transport.charAt(7) = “e”
```