JavaScript (Variables)

- Variable – A memory location that can store a value. In JavaScript variables are declared using **var**

  var temperature;
- Variables names must start with a letter, underscore or dollar sign and can be followed by any number of letters, underscores, dollar signs or digits.
- Variables must be declared before they are used.
- A variable can hold different type of values
- Values we can assign to variables
  - Integer – 0, 10, 40, 6, -7
  - Floating-point – 3.2, .67, 1.48E-20
  - String literals – “hello”, “goodbye”
- Operators
- Assignment operator (=)
  - Typical arithmetic operators (+, -, *, /)
- **Example:** Variables.html

Reserved Words

- Reserved words – words you cannot use as identifiers
- Some of them are:
  - break
  - do
  - If
  - catch

Spaces, Semicolons and Comments

- JavaScript ignores spaces, tabs, and newlines between tokens
- Use spaces to create nicely indented code
- The rules are usually one tab for indentation or three spaces. You need to satisfy this requirement in programming assignments.
- A semicolon is generally used to mark the end of a statement and is optional when a statement appears on a separate line. For example, the following two set of statements are equivalent

```
x = 1;
y = 2;

x = 1
y = 2
```

- In this course we will always use a semicolon to mark the end of a statement.

Comments

- Comments in JavaScript
  - Used to provide information to the programmer
  - Used to identify sections in your code
  - Ignore by the JavaScript interpreter
- Two types of comments
  - Inline comment - // This is a comment until the end of the line
  - Block comment –

    /* The following is a comment
    that spans several
    lines */
  - We can use a block comment for as a single-line comment
  - Block comments cannot be nested.

JS Program Template

- **Example:** TemplateJS.html

JavaScript (Dialog Boxes)

- We can perform input and output via dialog boxes.
- Input via **prompt**.
- **Example:** InputOutput.html
  - Notice we can define several variables at the same time.
  - ***prompt*** is a function that displays a dialog box with the specified title. It can be used to read any data.
  - You can read numbers and strings via prompt.
- **prompt** – returns a string.
- If you need to perform some mathematical computation you might need to explicitly convert the value read it into a number.

Strings

- You can use ‘ ‘ or “ “ for strings although we will use “ “ in this class.
- You can determine the number of characters in a string by accessing the length value.

  var s = “Hello”;

  var x = s.length;
- Some functions you can use with strings:
  - **toLowerCase()**
  - **toUpperCase()**
  - **substr(start, length)**- Copies segment of the source string beginning at start and continuing for length characters

Conversions

- In JavaScript you don't specify the type of variables.
- Most of the time implicit transformations will take care of transforming a value to the expected one.

  Example:

  ```
  var age = 10;
  var s = "John Age: " + age;  // age will be transformed into a string
  ```
- Sometimes you might need to explicitly transform a value.
- Mechanism to transform values:
  - **Converting number to string**

    ```
    var stringValue = String(number);
    ```
  - **Converting string to number**
    - var number = Number(stringValue);
    - var number = parseInt(stringValue);
    - var number = parseFloat(stringValue);
  - **Shortcuts**
    - Subtract zero from a string to convert it into a number
    - Add the empty string ("") to convert number into a string
- **Example:** Conversions1.html, Conversions2.html

Math Functions/Constants

- Some mathematical functions and constants you can use while working with numbers
  - Math.abs() – Absolute value
    - Example: Math.abs(-10)
  - Math.max() – Maximum of two values
    - Example: Math.max(10, 20)
  - Math.sqrt() – Square root
    - Example: Math.sqrt(4)
  - Math.random() – Random value between 0 and 1.
    - Example: Math.random()
  - Constants
    - Math.PI – Mathematical constant pi

Boolean Type

- We have seen integer, float, and string values
- New type: boolean type
- Assumes the value *true* or *false*
- Variable declaration and initialization

```
var found = true;
var attending = false;
```

JavaScript (Comparisons)

- You can compare values by using the following operators.
  - != → Returns true if the values are different, false otherwise (Example: x != y)
  - === → Return true if the values are equal, false otherwise (Example: x === y)
  - == → Not as strict as the previous equality operator
  - Relational Operators
    - < → Less than Returns true if left value is less than right value (Example: x < y)
    - > → Greater than
    - <= → Less than or equal
    - >= → Greater than or equal
- **Example:** Comparison1.html, Comparison2.html

JavaScript (If Statement)

- If statement – Control statement that allow us to make decisions.
- **First Form**

  if (expression)

  statement // executed if expression is true

- **Example:** IfStm1.html

- **Second Form**

  if (expression)

  statement1 // executed if expression is true

  else

  statement2 // executed if expression is false

- To execute more than one statement use a set of { }
- **Example:** IfStm2.html

JavaScript (Logical Operators)

- Used with comparison operators to create more complex expressions.
- Operators
  - Logical and (&&) – expr1 && expr2
    - The whole expression is true if and only if both expressions are true otherwise is false.
    - **Example:** LogicalOp1.html
  - Logical or (||) – expr1 || expr2
    - The whole expression is false if and only if both expressions are false otherwise is true.
    - **Example:** LogicalOp2.html
  - Logical Not (!) – !expr
    - Inverts the boolean value of the expression

Precedence/Associativity

- Remember you can use parenthesis to impose a particular order for the evaluation of an expression

Cascaded If Statement Idiom

- You can combine if statements to handle different cases.
- This approach to organize if statements to handle different cases is called the ***Cascaded If*** Statement.
- Cascaded If statement general form

```
If (expr1)
    // Statement is executed if expr1 is true
else if (expr2)
    // Statement is executed if  expr2 is true
else if (expr3)
    // Statement is executed if expr3 is true
else
    // If none of the above expressions is true
```

- Notice it is not a JavaScript statement.
- Once one of the cases is executed no other case will be executed.
- You can use { } to enclose more than one statement.
- **Example:** See CascadedIf.html