Announcements

- Check class announcements daily
Designing Using Pseudocode

- So far we have focused on the syntax and semantics of JavaScript.
- As the complexity of problems increase you need a design strategy to solve such problems.
- Several alternatives exist to come up with a solution to a problem. A popular one is Pseudocode.

**Pseudocode**: English-like description of the set of steps required to solve a problem.

- When you write pseudocode you focus on determining the steps necessary to solve a problem without worrying about programming language syntax issues.
Pseudocode Example

**Pseudocode for finding the minimum value**

1. Read number of values to process (call this value n)
2. Repeat the following steps until the n input values have been processed
   a. Read next value into x
   b. If (x is the first value read) {
      
      currentMinimum = x
   } else {
      
      if (x < currentMinimum)
         
         currentMinimum = x
      
   }
3. Print currentMinimum value
Pseudocode Elements

- When writing pseudocode you need the following constructs:
  - Input
  - Output
  - Assignments
  - Repetition Structures
  - Conditionals

- To help you with the design of pseudocode you can use the following syntax to represent the above constructs
**Pseudocode Elements**

- **Input**
  
  \[ \text{variable} = \text{read()} \]
  
  e.g., \( x = \text{read()} \)

- **Output**
  
  \( \text{print}(\text{variable}) \)
  
  e.g., \( \text{print}(x) \)

- **Assignment**
  
  \( x = \text{<value>} \)
  
  e.g., \( x = 20, s = \text{“Bob”} \)

- **Repetition**
  
  \[
  \text{while (expression)} \{ \text{OR} \quad \text{do} \{ \\
  \text{stmts} \\
  \} \text{while (expression)}
  \]

- Notice the above constructs look like JavaScript code but they are not JavaScript code
Pseudocode Elements

**Conditional (1)**
if (expression) {
    stmts
}

**Conditional (2)**
if (expression) {
    stmts
} else {
    stmts
}

**Conditional (3)**
if (expression1) {
    stmts
} else if (expression2) {
    stmts
    ...
} else if (expressionN) {
    stmts
} else {
    stmts
}

- For comparisons use: `===, <, >, <=, >=`
- Notice the above constructs look like JavaScript code but they are not JavaScript code
How Good Is Your Pseudocode

- Your code does not use language constructs that are particular to a programming language.
- Anyone receiving the pseudocode will not need to ask you questions in order to transform the pseudocode into code (no matter what the target programming language is).
do while Statement

- do while statement – Allows repetition of a set of statements

**Basic Form**

```
do {
    statement // executed as long as expression is true
} while (expression);
```

- Notice the semicolon after the expression in parenthesis
- Executes the statement at least once
- You don’t need the {} if you only need to execute one statement

**Example:** DoWhileNumbers.html

**Example:** DoWhile.html

- Any type of statements (including do whiles) in a do while
- When to use a do while?
- When to use a while?
Empty Statement

- Empty statement: represented by a semicolon
- It does nothing
- Example:

```javascript
if (x === 100)
    ;
else {
    // task
}
```

- Notice we don’t want to have code like the one above. We could rewrite it as follows

```javascript
if (x !== 100)
    // task
```

- If you add a semicolon to a while statement (not a do while) you may generate an infinite loop. Be careful
Creating a Bookmark in a Document

- Use name attribute to define the target

  `<a name="Sample_Output"></a>`

- Use `<a href …>` to jump to location

  `<a href="#Sample_Output">Web Site Snapshot/Video</a>`

- Online Example:
  http://www.cs.umd.edu/class/spring2009/cmsc122/projects/p2/p2Description.html#Sample_Output
Coding Example (Password Reading)

- Let’s write a program that reads a password and allows a maximum of two attempts.
- We can expand the program to have a menu of options (e.g., withdrawals, deposits) once the appropriate password has been provided.
- Using “\n” to define menu entries.
Increment/Decrement Operators

- `++` → increases value by one
  - `x++` → `x = x + 1`
- `--` → decreases value by one
  - `x--` → `x = x - 1`

Pre/post version

- `+++x` vs. `x++`
- `--x` vs. `x--`