Announcements

- Class Web Site:
  - http://www.cs.umd.edu/projects/passport/Classes/Spring2008/
  - You can find this link at the end of the main passport site
    - http://www.cs.umd.edu/projects/passport/webPage/

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, NetworkSelection.html

while Statement

- **while statement** – Control statement which allows JavaScript to repeat a set of statements.
- **Basic Form**

  while (expression)

  statement // executed as long as expression is true

- If you want to execute more than one statement then use a set of { } to enclose the statements.
- You can have other types of statements (including whiles) in a while.
- **Example:** SqrtTable.html

Combination of Statements

- Keep in mind that you can have any combination of conditionals, and iteration (while) statements.
- For example:
  - Conditionals insides of loops.
  - Conditionals inside conditionals.
  - Loops inside conditionals.
  - Loops inside of loops.

Trace Tables

- Mechanism to keep track of values in a program
- Allows you to understand the program behavior
- We could create a trace table for SqrtTable.html

Infinite Loops

- An infinite loop occurs when the expression controlling the loop never becomes false.
- **Example1**

```
int x = 30;
while(x > 0)
   document.writeln("<li>Element</li>");
```

- **Example2**

```
int x = 7;  // how about x = 8
while (x != 0)  {
   document.writeln("<li>Element</li>");
   x=x – 2;
}
```

- How can we detect infinite loops?

Programming Errors

- **Syntax Error**: (Compile-time error) The program violates the language's grammar.
- **Semantic Error:** The program fails to accomplish what we want.
- **Debugging:** The process of finding and fixing errors. Extremely hard for large software systems. Tools for debugging:
  - Trace tables
  - Output statements
  - Debuggers
- **Analogy:**

  Taco tom ate.→ Syntactically therefore semantically incorrect.

  A taco ate tom.→ Syntactically correct however semantically incorrect.

  Tom ate a taco → Syntactically and semantically correct (what we want!)

Designing Using Pseudocode

- So far we have focus on the syntax and semantics.
- As the complexity of problems increases 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 JavaScript 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 has been processed
    a. Read next value into x
    b. If (x is the first value read)
        currentMinimum = x
       else {
        if (x < currentMinimum)
            currentMinimum = x
       }
    c. Read next value into x
3. Print currentMinimum value

Pseudocode Elements

- When writing pseudocode you need the following fundamentals 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**

  variable = read() e.g., x = read()

- **Output**

  print(variable) e.g., print(x)

- **Assignment**

  x = <value> e.g., x = 20, s = “Bob”

- **Repetition**

```
while (expression) {        OR    do {
    stmts                           stmts
}                                 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 is the target programming language)

Suggestions for Solving Problems Using a Programming Language

- **Pseudocode -** Make sure you have written pseudocode. Try to verify (e.g., trace tables) that your pseudocode is correct.
- **Do not wait until the last minute** – Code implementation could be unpredictable
- **Incremental code development** – Fundamental principle in computer programming. Write a little bit of code, and make sure it works before you move forward
- **Don't make assumptions** – If you are not clear about a language construct write a little program to familiarize yourself with the construct
- **Good Indentation** – From the get-go use good indentation as it will allow you to understand your code better

Suggestions for Solving Problems Using a Programming Language

- **Good variable names –** Use good variable names from the get-to
- **Testing –** Test your code with simple cases first
- **Keep backups –** As you make significant progress in your development, make the appropriate backups
- **Trace your code**
- **Use a debugger**
- **Take breaks –** If you cannot find a bug take a break and come back later

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 parenthesis
- Executes the statement at least once
- If you want to execute more than one statement { }
- **Example:** DoWhile.html
- Any type of statements (including do whiles) in a do while.
- alert – Used to generate a dialog box
- When to use a do while?
- When to use a while?

JavaScript Lint

- How it can help us?
- http://www.javascriptlint.com/online_lint.php

Firefox Error Console

- How it can help us?
- Access the console via:
  - Tools → Error Console

Introduction to Debugging

- How to debug your code?