Welcome to CMSI 185
Computer Programming

Week 01

B.J. Johnson

Course Syllabus

- Syllabus is online at:

  http://bjohnson.lmu.build/cmsi185web/

- Textbook:

  Programming with Javascript

  John David Dionisio and Ray Toal

  ISBN-13: 978-0-7637-8060-9

  ISBN-10: 0-7637-8060-X

About the Instructor ☺

- Software Engineer/Architect
- Worked at Boeing, Raytheon, Hughes, and IBM
- Over 36 years' work experience
- Teaching at LMU since Fall 2005
- Focus on automated satellite test and operations
- Working with computers since 1969; as a full-time career since 1981

Generally:

- Homework about every other week
  - Due on Thursday, at *start of class*
  - Graded/returned the following Tuesday
- Test 1: Thursday, 2016-09-27
- Test 2: Thursday, 2016-11-01
- Final exam: Tuesday, 2016-12-13, 11:00
- Class grade determined by:
  - 10% class participation
  - 20% test #1, another 20% test #2
  - 20% homework (5% each, X 4 assignments)
  - 30% final exam

Class Order of Operations

- Intro, syllabus, history, tools (1)
- Fundamental concepts (3)
- Loops and decisions (2)
- Functions and closures (3)
- Events and User Interfacing (2)
- Software Construction (3)
- Review for Final Exam (1)

This Week's Agenda

- Course Syllabus
- Introduction to the introduction
- Chapter one and two
- Tools and such you'll need and use
- Programming languages
- What is computer science
- Introductory stuff: people, parts, etc.
- Definition of an algorithm
- Program modeling: flowcharts, UML, PDL/pseudocode

OK, Soooooooooo……

- Without further ado, let’s get into it!

Syllabus

- It's online at:

http://bjohnson.lmu.build/cmsi185web/

- Pretty much everything is on line…
- Let's go there now for a drive-by

A Few Basic Terms to Know:

- Algorithm: finite set of steps to perform computation that always ends
- Program: a representation or an implementation of an algorithm
- Abstraction: represents an algorithm in a "language independent" way
- Bug: a fault, flaw, or error in a program
- Grammar: a method of specifying a language

…And a Few More……

- Script: another name for program
- Binary: on or off, one or zero, yes or no, true or false (lit or dark)
- Crash: premature/unexpected ending of a program/script
- Server: a program that continuously executes to handle requests
- Client: program that makes requests

Some of My Favorites:

- Evil: code containing serious violation of proper coding practices
- Hairy: code which is so complicated it's impossible to understand
- Cruft: old code that is left over but isn't used or executed
- CRUD: Create, Retrieve, Update, Delete (a database term)
- DRY: Don't Repeat Yourself

Early Computing Machines

- Abacus (example)
- Jacquard's Loom
- Babbage's Difference Engine (video)
- Slide Rule (example)
- Pocket Calculator (1971 cost $400)
- Colossus (picture)
- Mark 1 (picture)
- ENIAC (picture)

Famous Computer People

- Alan Turing: Father of Computer Science
- Alonzo Church: Father of Lambda Calculus
- John Von Neumann: fetch, decode, execute
- Grace Hopper: invented COBOL, coined the term “bug”
- Brian Kernighan & Dennis Ritchie: developed the “C” language
- Richard Stallman: father of GNU
- Mary Lou Jepsen: Head of Display Division at Google X Lab
- Linus Torvalds: created Linux
- Edsgar Dijkstra: Structured programming (death of the GOTO)
- Bjarne Stroustrup: developed C++ language
- Frances E. Allen: pioneered optimizing compilers
- Noam Chomsky: formalized grammar rules
- Ada Lovelace: first programmer
- Douglas Engelbart: invented the computer mouse
- Alan Kay: developed windowing computer interface
- Hedy Lamarr: invented frequency hopping
- John Seely Brown: led Xerox PARC
- Fred Brooks: “Mythical Man Month”, chief architect of IBM 360

What *IS* Computer Science?

- It's like calling surgery "knife science"
- Basically the study of algorithms
  - Decompose a solution to its steps
  - Analyze algorithms
    - Time complexity
    - Number of steps
    - Efficiency
  - Create new and interesting ways to solve new and interesting problems
- Related to many fields

Central Questions of CS

- Which problems can be solved by algorithmic processes?
- How can algorithm discovery be made easier?
- How can techniques of representing and communicating algorithms be improved?
- How can characteristics of different algorithms be analyzed and compared?
- How can algorithms be used to manipulate information?
- How can algorithms be applied to produce intelligent behavior?
- How does the application of algorithms affect society?

What We'll Learn……

- Computer programming (duh!)
- How to program in JavaScript
- How to program in Python
- How to make cool stuff on web pages
- Programming fundamentals which you can apply to *any* language
- Foundations of the "Art" and "Craft" of writing computer code

We'll Also Learn a Bit About:

- How your computer works
  - Computer hardware/organization
  - Bits, bytes, data representations
  - Operating systems (a little)
  - Web pages, style sheets
  - Software coding and debugging tools
- How to know what goes on inside
- Ethical things to think about

Things You'll Need Here

- Concepts of basic algebra
  - Operations
  - Variables and assignments
  - Symbols
- Critical thinking skills
- Willingness to ask questions
- Willingness to fail *and learn from the failure*
- Willingness to try new things

I Need a Volunteer ~

- Think of an algorithm!
  - Think of something you do or a problem you must solve
  - What are the steps?
  - Write them down
  - Is the problem solved?
  - Is the "something" completed?
- If so, you've just made an algorithm
- In fact . . .

For Thursday ~

- Read chapters one and two
- Visit these web sites:
  - http://www.w3schools.com/js/default.asp
  - http://jsfiddle.net/
  - http://javascript.cs.lmu.edu/runner/
  - http://www.jslint.com/
  - http://nodejs.org/
- Take some time to "investigate" on these sites


That's all Folks!