* Project Management
  + 2 september - 8 september lecture 3
  + What makes a project a project?
    - A series of tasks w/ distinct beginning and end.  Bound by time, cost, and quality. Requires resources, solves problems, stems from a larger strategy/vision, has a customer and/or sponsor, distinct from a process
    - Lifecycle: inception, conceptualization, planning, execution, termination
  + What makes a project successful?
    - On time, on budget, meets objectives
  + Triple constraint model
    - Time, cost, scope (think like pick three triangle)
  + Managing scope
    - Undefined, poorly defined, or shifting requirements (scope creep) cause scope failure
  + Waterfall
    - One step at a time, no product availability until project is finished.
      * Problems with scope will extend deadlines/cost
      * Project requirements defined and solidified early
      * Detailed work structure from beginning
  + Agile, Scrums
    - Agile Manifesto
      * Focused on software dev projects
      * Value:
        + individuals /interactions over processes and tools
        + Working software over comprehensive documentation
        + Customer collab. Over contract negotiation
        + Responding to change over following a plan
      * Priorities:  early/continuous delivery of software, accepting change, shorter timescales, collaboration, face-to-face, working software is primary measure of progress, sustainable development, attention to good design, simplicity, self-reflection
    - Daily Scrums
      * 3 questions: what did you complete, what will you complete by next meeting, what is a roadblock/obstacle
      * 15 minutes (should not exceed this time)
      * All stand
    - Scrum/Agile Roles
      * Product owner
        + Acts on behalf of customer, prioritizes feature backlog, keeps team focused on priority objectives, final authority on requirements, decides when a feature is complete
      * Team
        + Builds feature, 5-9 members, self-organizing, no internal roles, give input to sprint planning
      * Scrum master
        + Facilitator
        + Resolves issues
        + Keeps team on process
        + “coach “ not “manager”
* Full Stack Development
  + Full Stack Developer - someone who can work on both the front-end and back-end portions of an application
  + Stack elements
    - Front end - presentation, what user interacts with
    - Back end - underlying business logic, database, etc.
* Defining Requirements
  + ‘Use Case’
    - Describes how a user will interact with a solution to achieve a specific goal
    - Sum of use cases are ‘user requirements’ for a solution
  + Functional Requirement
    - “what” – (Users’ view)
    - Documented by use case
    - what the system does for them
  + Non-functional Requirement
    - “how” – (Developers’ view)
    - Describes HOW the system should behave
    - “If a document is flagged as private, only the user who created it can see it.”

* Linux Shell
  + Navigating the shell – using directories, files
    - Pwd-Present working directory (tells you where you are)
    - Mkdir-make directory (mkdir nameOfFile)
    - Cd-change directory, cd with no file instruction returns you to home directory
    - Ls-list directory contents
    - Cp-copy files
    - Rm-remove files
    - Mv-move files
    - Rmdir-remove directory
    - Others?
  + Referring to files
    - Absolute path from root
    - Relative path from current directory
    - “/” root directory, “.” current directory, “..” parent directory
      * /home/user/docs/Letter.txt
      * ./inthisdir
      * ../../greatgrandparent
  + Processing Files
    - Cat - copy contents to the screen
    - More - display file contents in user friendly manner
    - Head - copy the first lines of a file to the screen
    - Tail - copy the last lines of a file to the screen
    - Wc - count the number of lines, words, characters in a file (-l, -w for first two) wc -l file
    - Grep - globally search a regular expression and print (pattern file) grep include controller.cpp
    - Diff - compare two files diff file1 file 2
    - Sort - sort lines of text files (-n on numerical data file, -k# by column) sort -k2 data.txt
    - Find - find a file in a directory tree find . -name filename -print
  + Running Programs
    - Redirecting output - can tell shell to pass something from a file to a process as input, and to place output in a file instead of displaying it
    - Pipes - input to output between working process
    - Ps process status
  + History of Commands Typed
    - ‘Linux keeps a record of prev. commands, that you can scroll through with the up/down arrows from the same command line

* Shell Scripting, Regex
  + Regex
    - Regular expressions-used to find strings
    - Regex used to check user data input\
    - ^.\*\.txt$ equiv. Command to find all text files in a directory (look for any file name that ends in txt)
    - 2 september - 8 september lecture 2
    - <https://regex101.com/> online test ref
    - \b[A-Za-z]+@[A-Za-z]+\.[A-Za-z]+ approximately finds an email address
      * + means one or more of previous thing
  + #!
  + Shebang
  + Chmod
  + Variables
  + comments
* Awk & Sed
  + 2 september - 8 september lecture 2
  + Awk
    - Processing files and records using regex
    - a full-blown programming language (more used)
    - good for searching input files/fields and reporting (esp. When there are field separators)
    - Used when the text is in “file/delimited field format”
    - awk ‘/\*RegEx\*/’ testfile -a command to use a RegEx with awk (no stars)
    - Field - a unit of data in a line
    - Field separator - some syntax to separate fields
    - Record - the collection of fields in a line
    - Data file is made up of records
    - $1 $2… references a specific field in a record, $0 shows the whole record
  + Sed
    - a less robust language
    - good for modifying a streamed file
* Git & Version Control
  + Init - create new local repository with the same name
  + Status -
  + Diff - the set of changes between two revisions
  + Working copy - project’s files
  + staged file - change for next commit
  + Local repo - .git folder
  + Commit - write/save changes to a repository
  + Checkout - switches to the specified branch and updates working directory
  + Head -
  + merge -  combine one branch into current local branch
* HTML & CSS
  + HTML
    - uses pre-defined tags
    - Markup language rendered by the browserCSS
    - Cascading Style Sheets
    - 3 places to apply style
      * In tag, in header, in external CSS file
    - Hierarchy of applying style
      * Highest to lowest priority: tag, header, external
    - Reference an external style sheet
      * <head>
      * <link rel="stylesheet" href="styles.css">
      * </head>
* SQL & Databases
  + SQL - standard language for creating and managing a database
  + Tables, Rows, Columns
    - All info in a SQL database is maintained in tables
    - Rows or records are specific info on one instance (user, order, etc)
    - Columns or features are a type of info specific to a row (username, order #)
  + Primary key - a column or set of columns that uniquely identifies rows of a table
  + Foreign key - a key used to link two tables together, (ex. A userID may be a primary key in a user table, and a user order table may have userID as one of the columns.  The second userID is not the primary key in the order table, but it can be used to link rows of orders to rows of users in the first table.)
  + Data model
  + Select statement
    - Where clause conditions (like, in)
    - Group functions,
      * Group By - sorts data in a column into groups that can then be parsed together.  Ex. Select buyerID, max(price); from orders; group by buyerID. Shows the max price order for each buyer
      * Having - sorts an aggregate, Select buyerID, max(price) from orders; having price > 1000 filters orders below 1000.
    - Subquery and joins
      * SubQuery - does complicated stuff, adds commands to queries
      * Join -
      * Outer Join - The outer join is used when a join query is "united" with the rows not included in the join, and are especially useful if constant text "flags" are included.
  + DDL – create, alter, drop
    - Data Definition Language
  + DML – insert, update, delete
    - Data Manipulation Language
* Web Services
  + Http
    - Request/response protocol
    - stateless
  + Xml
    - Extensible Markup Language
    - “Tag” based, like HTML
    - Tags are user-defined
    - Represents data in key:value pair format
  + Json
    - Java Script Object Notation
    - More compact than XML
  + api
* REST & SOAP
  + REST is an architectural style
  + Representative state transfer
    - Stateless, layered, cacheable, scalable
    - Uniform interface methods (http), data format (json, xml)
    - ~70% of public APIs
  + SOAP is a protocol
    - Protocols pass messages and requests from one layer to another
    - Relies on XML
    - Requires WSDL
    - Less flexible than RESTful web services
* Extra
  + A programming language used to develop and deploy server-side scripts on a web server. → NodeJS