Code Review Nov. 8th. 2017 Wed.

Reading 1 (Mit) Summary:

Code Review: Code Review is careful, systematic study of source code by people who are not the original author of the code. It's analogous to proofreading a term paper.

Two purposes:
- Improving the code
- Improving the programmer

Style Standards

self-consistent, but there are some rules that are quite sensible:

don't repeat yourself: duplicated code is a risk to safety. If you have identical or very similar code in two places, then the fundamental risk is that there's a bug in both copies, and some maintainer fixes the bug in one place but not the other.

comments where needed: ① specification, which appears above a method or above a class and documents the behavior of the method or class.
② specify the provenance or source of a piece of code that was copied or adapted from elsewhere.
③ comment the obscure code

fail fast: code should reveal its bugs as early as possible.

avoid magic number: declare the number as a constant with a good, explanatory name (e.g. FEBRUARY)

one purpose for each variable: you will confuse your reader if a variable that used to mean one thing suddenly starts meaning something different a few lines down.

② method parameters should generally be left unmodified. (for java)

use good names: in python, classes – CAPITALIZED
variables – lowercase, words_are_separated_by_

use whitespace to help the reader: use 4 spaces, not tab

don't use global variables: A global variable is:
- a variable, a name whose meaning can be changed.
- that is global. accessible and changeable from anywhere in the program.

methods should returns results, not print them: in general, only the highest-level parts of a program should interact with the human user or the console.

3 key properties of good software: 1. Safe from bugs
2. Easy to understand
3. Ready for change

Reading 2 (Evoke Technologies) Summary:

1. Code formatting
   a) Use alignments (left margin), proper white space
   b) Proper naming conventions
   c) Code should fit in the standard 14 inch laptop screen
   d) Remove the commented code (can be obtained from Source Control, if required)

2. Architecture
   a) The code should follow the defined architecture
      1. Separation of Concerns followed
         - Split into multiple layers and tiers as per requirements
         - Split into respective files (HTML, Javascript and CSS)
      2. Code is sync with existing code patterns/technologies
      3. Use appropriate design pattern

3. Coding best practices
   1. No hard coding (embed an input or configuration data directly into the source code)
   2. Group similar values under an enumeration (enum)
   3. Comments, why you are doing, mention pending tasks
   4. Avoid multiple if/else blocks
   5. Use framework features

4. Non Functional requirements
   a) Maintainability
      1. Readability
      2. Testability
      3. Debuggability
      4. Configurability
   b) Reusability
      1. DRY (Do not Repeat Yourself) principle
      2. Consider reusable services, functions and components
      3. Consider generic functions and classes

c) Reliability :

Exception handling and cleanup (dispose) resources.

d) Extensibility

e) Security

f) Performance

1. Use a data type that best suits
2. Lazy loading, asynchronous and parallel processing
3. Caching and session/application cluster

g) Scalability

h) Usability

5. Object-Oriented Analysis and Design (OOAD) Principles

1. Single Responsibility Principle (SRS)
2. Open Closed Principle
3. Liskov substitutability principle
4. Interface segregation
5. Dependency Injection