Itential IDEV110 Training

* Chapter 1: Platform Logic Division
  + Layers offer benefits including
    - Reducing change propagation
    - Improved code reliability
    - Better code reuse
    - Improved code testing
    - Faster code updates
    - Rapid innovation and new technologies
  + Adapters
    - Transform data from external systems.
    - Abstract external system nomenclature
    - Abstract external system APIs
    - Design efficiency
* Chapter 2: Itential Product Packages
  + IAP products are Node.js packages that extend features of the Itential platform.
  + Custom IAP products allow for access to disparate systems.
  + IAP products combine server side business logic and views to solve specific business cases.
  + Products use business logic to:
    - Transform data
    - Modify data
    - Add new, unique business logic
  + IAP products have:
    - Npm package configuration, package.json
    - IAP configuration file, pronghorn.json
    - JSON schema that defines parameters
    - Node.js module
    - Node.js module that defines a class with public methods and export an object
  + Best Practices
    - IAP product should be multiple node.js modules
    - The main Node.js module defines class, public methods, and exports an object
    - Helper functions defined in separate modules
    - Main module imports private functions from separate modules
* Chapter 3: NPM
  + IAP products leverage the Node.js Package Manager (NPM) to automatically manage downloading and installing dependencies from the public npm registry.
  + Node.js = server-side JavaScript
  + A package is a file or directory that is described by a package.json file
  + A package.json file:
    - Lists the packages dependencies
    - Specifies compatible dependent package versions through semantic versioning rules.
    - Makes the build reproducible
  + Package.json file must contain name and version fields
  + Semantic Versioning (semver)
    - First release 1.0.0
    - Patch release 1.0.1
    - Minor release 1.1.0
    - Major release 2.0.0
  + Npm commands
    - Npm init
      * Initialize a package
    - Npm install
      * Install package dependencies
    - Npm run
      * Execute scripts against a package
    - Npm version
      * Increment package versions
* Chapter 4: JSDoc
  + JSDoc is an API documentation generator JavaScript
  + Block tags begin with @
  + Inline tags, begin with @, require braces {}
* Chapter 5: Node.js Modules
  + A node.js module is a text file containing node.js programming code
  + Modules can define and export variables, functions, classes and objects that other modules can import and use.
  + Modules can be bundled into node.js packages that make sharing easy
  + Module programming is the process of dividing a computer program into sub programs.
  + Benefits of modular programming are
    - Improve maintainability. Code is shorter and simpler
    - Errors are more easily identified
    - Variable scoping is more easily controlled
    - Less code is required, as subsections can be reused.
    - The same code can be used in many applications.
    - Allows many programmers to collaborate on the same application.
    - Teams can develop modules separately
    - Programs can be designed more easily.
  + Modules are implemented with *require(‘module’)*
* Chapter 6: Node.js Callbacks
  + IAP public functions are expected to take a callback function as its last parameter
  + IAP callbacks follow a data-first convention.
* Chapter 7: IAP Packages
  + IAP products are Node.js packages with specific, required characteristics.
    - Must have an npm configuration file, package.json in the top level directory.
    - Must have Itential platform configuration file, pronghorn.json in the top level directory
    - The main module must define a class, instantiate an object from the class and export the object.
    - One or more JSON schema files that describe, validate and align methods parameters and returned data. Must be located in a json.schema directory.
  + IAP package.json should include
    - Name
    - Version
    - Description
    - Main
    - Scripts
    - Author
    - License
    - Dependencies
    - devDependencies
  + pronghorn.json should include:
    - id
    - summary
    - title
    - export
    - type
    - src
    - roles
    - methods -should contain
      * name
      * description
      * input should include
        + name
        + description
        + type
        + schema
      * output should include
        + name
        + description
        + type
        + schema
      * roles
    - views
  + JSON schema
    - $id property
    - Title
    - Description
    - Type and description
    - Default and examples
    - Pattern
  + Common module start up problems
    - A missing file
    - JSON syntax error
    - Comments in JSON files return errors
    - The module is not found
    - If the file name and pronghorn.json src do not match, it won’t load
    - Syntax errors in the node module
  + Log messages
    - Error – runtime problem detected
    - Warn – a problem didn’t occur, but something was detected
    - Info – informational purposes
    - Debug – debugging messages
    - Trace – adds copies of passed arguments and returned data to debugs
    - Spam – adds copies when data is large
* IAP API’s
  + Node.js APIs
  + HTTP APIs
  + Job tasks
  + IAP products can use all three apis
  + IAP brokers can use node.js apis or job tasks.
  + IAP adapters can be called by node.js apis or job tasks.
  + Node api cogs call products
* Runtime Problems
  + Caused by code errors, external service disruptions, upgrades with unintended effects, and other changes.
  + Can use **tail -f** to monitor IAP logs.
  + In node.js, callback functions are primarily used for locating runtime problems.