inedo

KEVIN GRIFFIN

• • •

SWIFT KICK

• • •

INEDO TRAINER

LUNCH AND LEARN

Universal Packaging Essentials

What and Why Packages?

Swift Kick

Kevin Griffin

December 12th, 2018

About the Instructor

Kevin Griffin

Owner, Swift Kick

Swift Kick is a software training and consulting company based out of Southeast Virginia. Our primary focus is Microsoft-based technologies, such as .NET and Microsoft Azure, and web-based software development.

kevin@swiftkick.in

https://kevgriffin.com

https://swiftkick.in

InedoCon 2019

- May 22nd and 23rd
- Portland, OR

- Keynotes from CEO Aaron Jensen, Continuous Delivery Architect at WebMD Health and the creator of Carbon

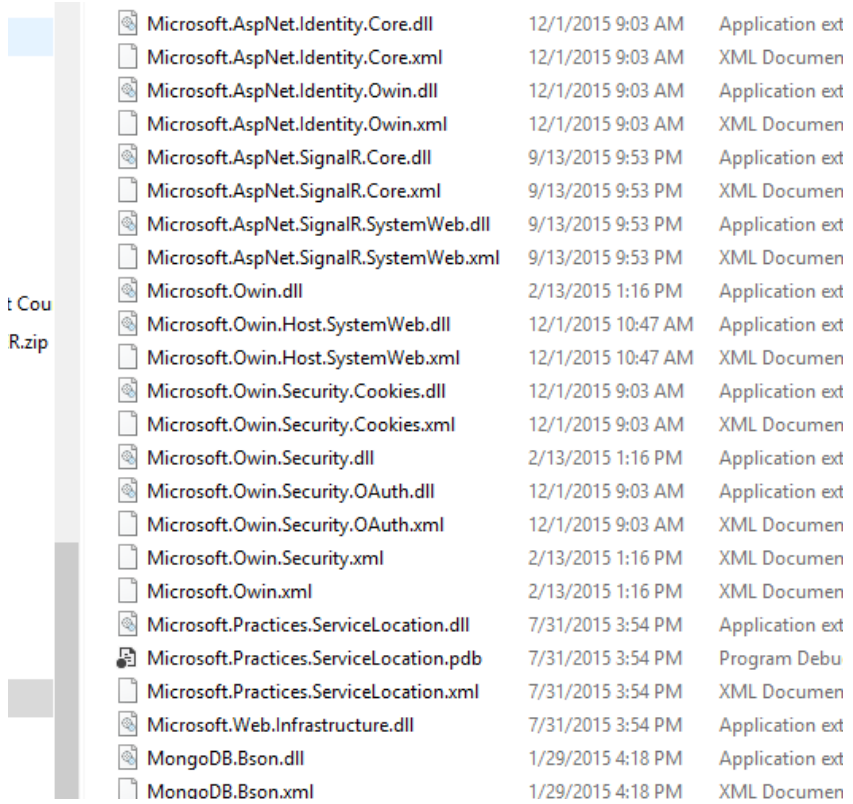
- Tickets are 40% off until Jan 4th, 2019 ($150)
- Regular price tickets will be $250

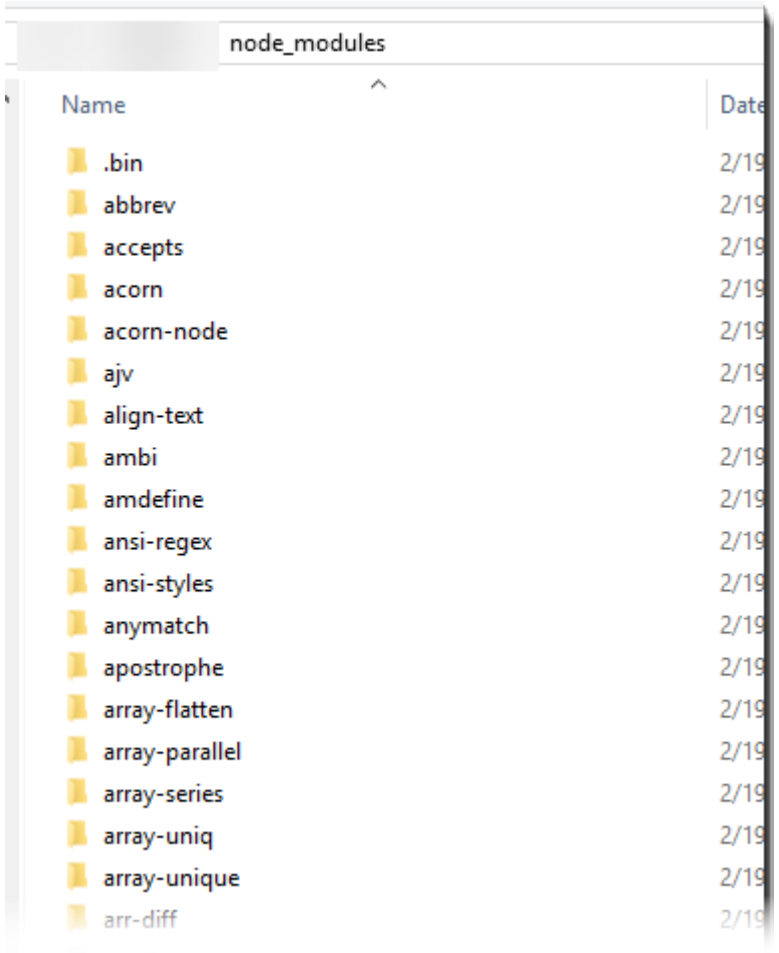
- **`https://inedocon.inedo.com`**

Agenda

1. Applications are complicated – and we make it worse
2. What is a package?
3. What goes inside a package?
4. Application design with packages
5. Package versioning
6. Package discovery

Applications are complicated






Okay, step 42. Go download SwiftKick.Utilities.dll from the share drive.
Right click, Add Reference….
Teams make it more complicated

All projects have dependencies

OurCompany.Utilities

- First-party dependencies are libraries built in house.

  Some are specific to an action, like talking to databases or writing to a log aggregator.

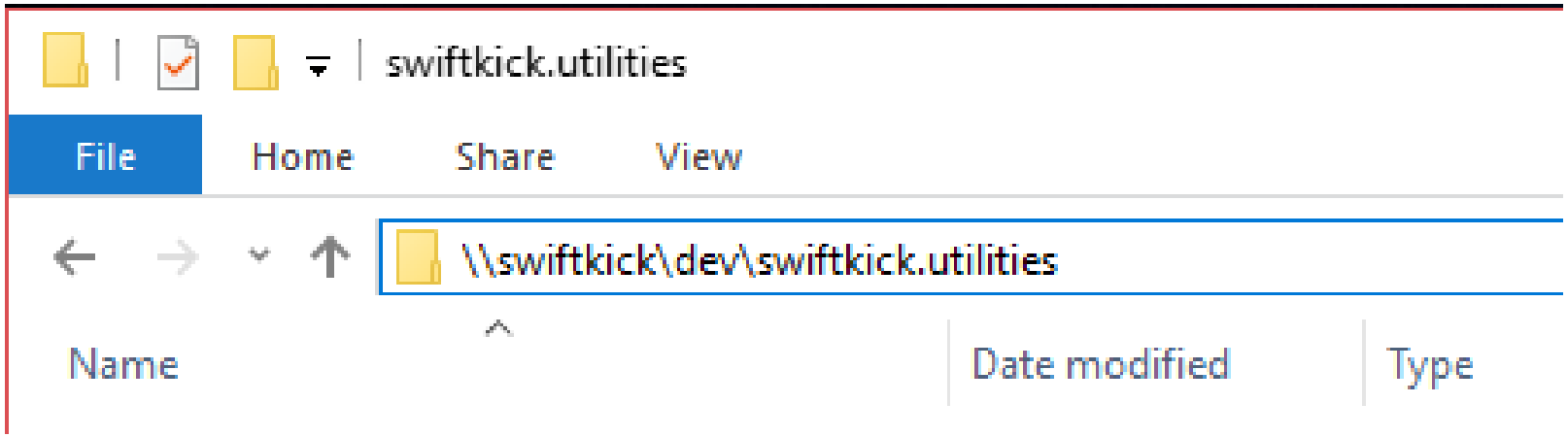
  Others are “kitchen sink” libraries that do everything.

All projects have dependencies

OurCompany.Utilities

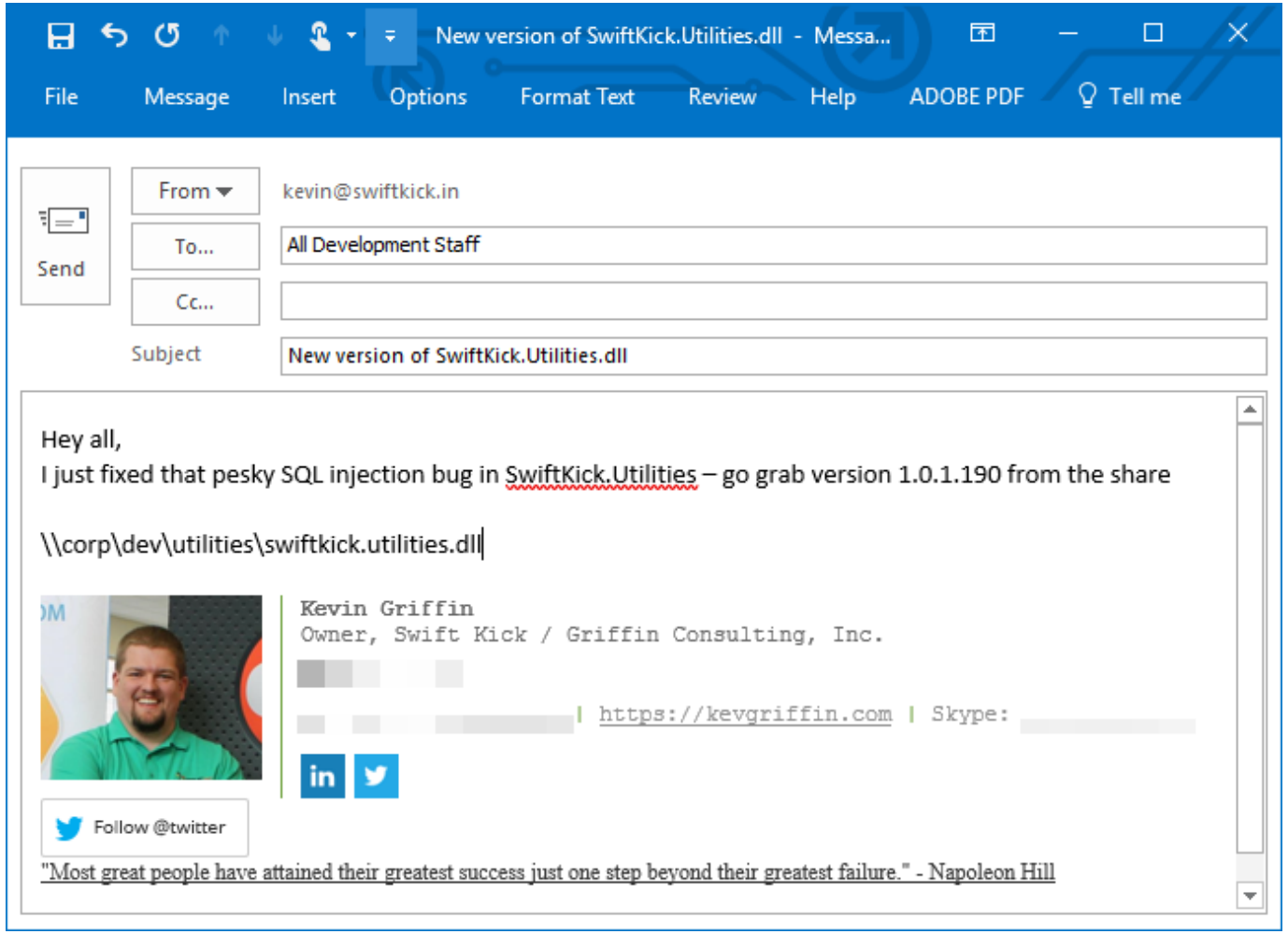
- Third party dependencies are libraries developed outside the organization, and are commonly used to help teams from “reinventing the wheel”.
  - Reading/writing JSON
  - Uploading files to S3
  - Processing credit cards
  - And more!

How do you distribute or consume these libraries?




No no no! I released version 1.0.1.190 an hour ago!
Go get that version.
How do you version the kitchen sink?

How do you manage updates?




There should be
a better way…

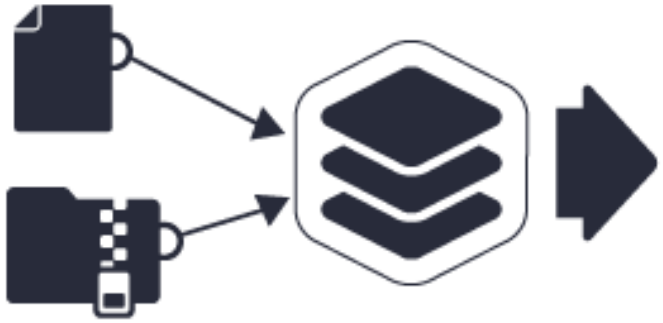
Swift Kick
SOFTWARE TRAINING AND CONSULTING

Packages

What is a package?

Swift Kick

- A package is a container file (think .zip files) that contains files, scripts, and a manifest for installing or adding an external dependency.
- Packages are used by library or tool creators to help with distribution.
- Packages are hosted or stored in centralized repositories (feeds).



Swift Kick
SOFTWARE TRAINING AND CONSULTING

Types of Packages

Three types of packages

- Developer Packages
  - 3rd party libraries used by developers
  - Popular examples
    - jQuery
    - JSON.NET
    - Angular/React/Vue
  - Deployed via public package managers
    - NPM
    - NuGet
    - And others…
- Application/Component Packages
  - Internal packages
  - Not available for public consumption
  - Deployed via private package managers, such as ProGet
  - If packages don't fit existing models, UPack is a great solution!
- Machine Packages
  - Contains tools or utilities
  - Used by admins
  - Examples:
    - Chocolatey
    - PowerShell
    - Debian

Machine packages?

Swift Kick

- This might seem like a new concept, but you've been using them for a long time!

- App Stores
  - Apple
  - Google
  - Microsoft
  - And others!

- Any "app" you have installed is a self-contained package of binary executables.

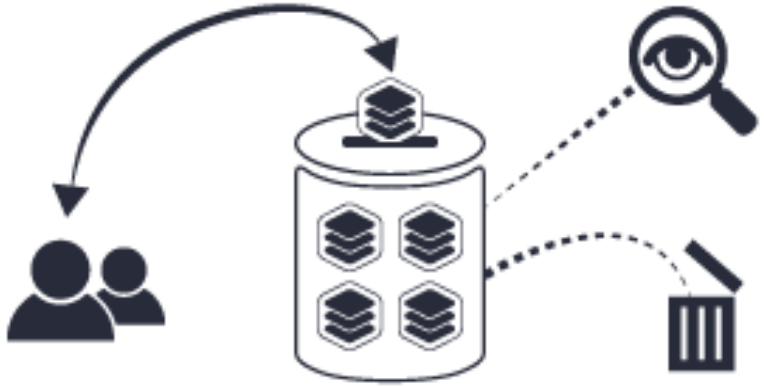
Machine packages?

- To go back even further, machine packages have been growing more popular with computer professionals.
- Packages solve a fundamental problem: how do you quickly and efficiently install software and utilities?

- Linux distros use package managers to make installing software easier.
  - Debian uses apt.
  - Red Hat uses yum or rpm.
  - Arch uses pacman.
  - Gentoo uses portage.
- Chocolatey is becoming increasingly popular on Windows.

Machine packages?

- By using machine packages:
  - We can track dependencies and automatically install them
  - Check for updates as they are available
  - Instantly remove and cleanup remnants of a package



All major platforms have package feeds

- .NET (NuGet)
- Node.JS (NPM or Yarn)
- Python (PIP)
- Ruby (Gems)
- Go (gopm)
- And more…

Swift Kick
SOFTWARE TRAINING AND CONSULTING

What is inside a package?

What types of files go in a packages?

- Packages can contain everything to add a dependency or install a tool.

- Executable files or library binaries
- Scripts to update the environment as needed by the dependency
- Supporting content
  - Images
  - Stylesheets
  - Text files/configuration
- A manifest that tells the installer what is in the package, and how to properly install the dependency

Manifests

- Every packaging system uses a different type of manifest to describe how a package should be installed.

- Two examples:
  - NuSpec is used by NuGet (Visual Studio and .NET-based applications).
  - Package.json is used by NPM (Node.js)

Example: NuGet

Swift Kick

```
<?xml version="1.0" encoding="utf-8"?>
<package xmlns="http://schemas.microsoft.com/packaging/2013/05/nuspec.xsd">
  <metadata minClientVersion="2.12">
    <id>Newtonsoft.Json</id>
    <version>11.0.2</version>
    <title>Json.NET</title>
    <authors>James Newton-King</authors>
    <owners>James Newton-King</owners>
    <requireLicenseAcceptance>false</requireLicenseAcceptance>
    <licenseUrl>https://raw.github.com/JamesNK/Newtonsoft.Json/master/LICENSE.md</licenseUrl>
    <projectUrl>https://www.newtonsoft.com/json</projectUrl>
    <iconUrl>https://www.newtonsoft.com/content/images/nugeticon.png</iconUrl>
    <description>Json.NET is a popular high-performance JSON framework for .NET</description>
    <copyright>Copyright © James Newton-King 2008</copyright>
    <tags>json</tags>
    <repository type="git" url="https://github.com/JamesNK/Newtonsoft.Json.git" />
    <dependencies>
```

Example: NuGet

Swift Kick

```
<dependencies>
  <group targetFramework=".NETFramework2.0" />
  <group targetFramework=".NETFramework3.5" />
  <group targetFramework=".NETFramework4.0" />
  <group targetFramework=".NETFramework4.5" />
  <group targetFramework=".NETPortable0.0-Profile259" />
  <group targetFramework=".NETPortable0.0-Profile328" />
  <group targetFramework=".NETStandard1.0">
    <dependency id="Microsoft.CSharp" version="4.3.0" exclude="Build,Analyzers" />
    <dependency id="NETStandard.Library" version="1.6.1" exclude="Build,Analyzers" />
    <dependency id="System.ComponentModel.TypeConverter" version="4.3.0" exclude="Build,Analyzers" />
    <dependency id="System.Runtime.Serialization.Primitives" version="4.3.0" exclude="Build,Analyzers" />
  </group>
  <group targetFramework=".NETStandard1.3">
    <dependency id="Microsoft.CSharp" version="4.3.0" exclude="Build,Analyzers" />
    <dependency id="NETStandard.Library" version="1.6.1" exclude="Build,Analyzers" />
    <dependency id="System.ComponentModel.TypeConverter" version="4.3.0" exclude="Build,Analyzers" />
    <dependency id="System.Runtime.Serialization.Formatters" version="4.3.0" exclude="Build,Analyzers" />
    <dependency id="System.Runtime.Serialization.Primitives" version="4.3.0" exclude="Build,Analyzers" />
    <dependency id="System.Xml.XmlDocument" version="4.3.0" exclude="Build,Analyzers" />
  </group>
```

Example: NPM

Swift Kick

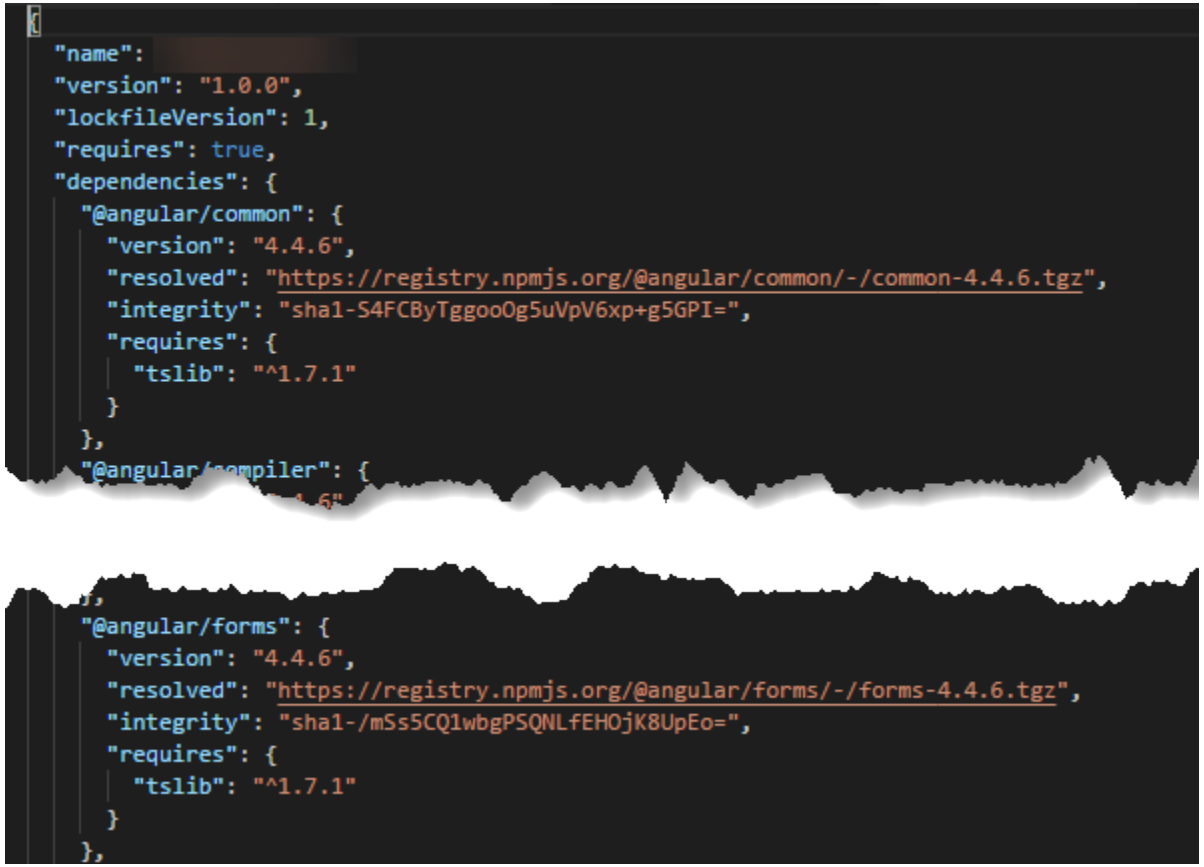
```
{
  "dependencies": {
    "@angular/common": "4.4.6",
    "@angular/compiler": "4.4.6",
    "@angular/core": "4.4.6",
    "@angular/forms": "4.4.6",
    "@angular/http": "4.4.6",
    "@angular/platform-browser": "4.4.6",
    "@angular/platform-browser-dynamic": "4.4.6",
```

```
    "vinyl-paths": "^2.1.0",
    "zone.js": "^0.8.18"
  },
  "devDependencies": {
    "@types/jasmine": "^2.5.43",
    "@types/jquery": "^3.2.16",
    "@types/lodash": "^4.14.80",
    "@types/moment-timezone": "^0.5.2",
    "@types/node": "8.0.47",
    "@types/toastr": "^2.1.35",
    "angular2-template-loader": "^0.6.2",
    "awesome-typescript-loader": "^3.1.2",
    "css-loader": "^0.28.7",
    "eslint": "^4.19.1",
```

```
    "uglifyjs-webpack-plugin": "1.0.0",
    "webpack": "^3.10.0",
    "webpack-bundle-analyzer": "^2.4.0",
    "webpack-dev-server": "^2.4.1",
    "webpack-merge": "^4.0.0"
  },
  "name":
  "private": true,
  "version": "1.0.0",
  "scripts": {
    "prod": "npm run clean & webpack --config webpack/webpack.prod.js --progress --profile --bail",
    "dev": "npm run clean & webpack --config webpack/webpack.dev.js --progress --profile --bail --watch",
    "clean": "rmdir wwwroot\\dist /s /q"
  }
```

How do you verify packages?

Swift Kick



Swift Kick
SOFTWARE TRAINING AND CONSULTING

Application Design with Packages

Application Design with Packages

- Packages prevent us from having to “re-invent the wheel”.

- Applications have a common set of needs, and it’s likely your application will have similar needs and requirements of other applications.
- Because of shared needs, a package of a library or utility might already exist.

- Packages allow developers to quick inject functionality into their applications without having to manually copy files or create configuration.
- “Plug and Play”

Application Design with Packages

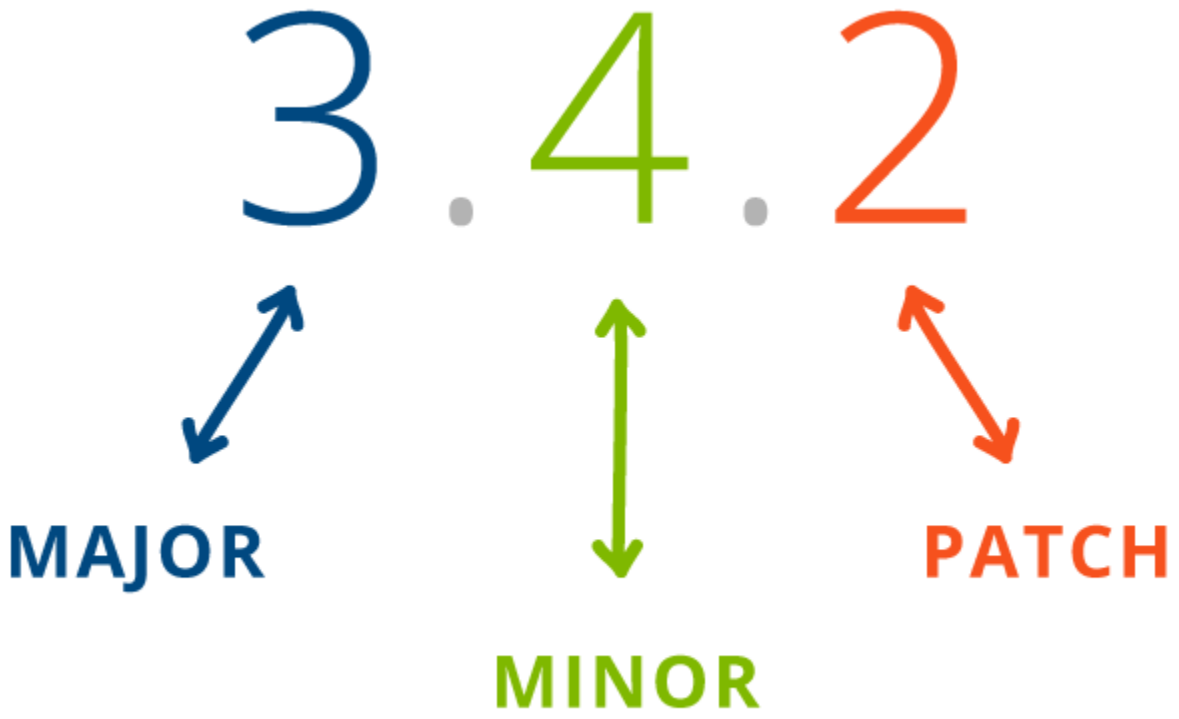
- Manifests, such as we saw with NPM and NuGet, are human readable files.

- These can be committed into source control, and provide some great benefits:
  - Package updates can be versioned and tracked.
  - Binaries for packages don't get committed, since developers can pull the exact package and version at a later time.

Swift Kick
SOFTWARE TRAINING AND CONSULTING

Package Versioning

Versioning (Semver)



Versioning (Semver)

- Major
  - Used for major updates. Typically used when backwards compatibility is not possible.
- Minor
  - Used for new features and updates that do not break existing functionality.
- Patch
  - Used for bug fixes and other minor changes

Prerelease Qualifiers

Swift Kick

- If a package is in prerelease, a suffix can be added to the version.

1.0.0-alpha
1.0.0-beta
1.0.0-rc.1

- A suffixed version is considered lower than the non-suffixed version.

1.0.0 < 1.1.0-beta < 1.1.0

Packages Should Be Immutable

- When you download a package from a feed, you need to have a certain level of trust that the package will not change.
- Some types of feeds prohibit package developers from overwriting a version of a package after it has been published.
- Using Semver gives package developers the ability to patch packages when overwriting would be the obvious solutions.
- Some feed types, such as Universal Packages, allow for repackaging which allows package developers to change the version of a package without altering the content of the package.

Swift Kick
SOFTWARE TRAINING AND CONSULTING

Package Discovery

Package Discovery

- There are multiple ways to share packages

- Direct copy
  - Since most packages are self-contained files with manifests, you can copy packages to file shares, email, or other direct methods.
  - It is difficult to maintain versions of packages with this approach.

- Feeds
  - Services like NuGet or NPM provide feeds that will track packages and versions of packages through several iterations.
  - Much easier to install and manage older versions of packages.
  - Private package hosting is available through tools like ProGet

Recap

- Managing dependencies the hard way requires manual processes and human interaction.
- Packages solve these issues by encapsulating dependencies, and providing a manifest for tools to use for installation and configuration.
- Most major software platforms have public feeds available
- Semantic versioning helps humans and package managers know what the current version of a package is.
- Package managers aid in the discovery of new packages and ensuring the correct version of a package is installed.