Coding Standards Addendum

- All classes and interfaces will be properly documented
    - Comments at the class/interface level
    - Descriptions at the function/method level
    - Inline-comments when necessary
    - User standard VS comment format
    - Deprecated classes and methods should be avoided in all cases
- Sonar Cube must be used and all issues addressed before pull request is issued
    - SonarQube Analysis
- Classes and Methods should be single purpose
    - One objective
    - If you are using #region you are probably packing too much responsibility into a code unit and definitely needs refactoring
    - Short and readable code
    - Favor readability over language simplifications
- Use packages and folders
    - No new Controllers, Services or Repositories should go into the general collection of existing ones (we should use packages)
- Every public method should have a Unit Test
    - Depending on the importance private methods can have unit tests as well
    - Database Unit tests should not rely on existing data
        - Create, insert/update and clean up your unit test data
- Every code change must be associated with a Jira
    - Commit comments should alwasy be in this format:
      <Jira #> <Jira Title>
      - <Comment>

```
* CR-4207 International Concept on COG, Projects, Filter Frame, PFSG filtering for Peers - Defined Facto
- added support to iterate over multiple connection strings
- now obtaining HTTPContext in every repo class via DI
- implemented ToString() in GetRequest class for better logging
- made Domicile accessible to all child repository classes
- removed EmailID related implementation from UserRepository
```

- Please squash/collapse GIT commits to represent logical commits before raising a pull request
    - https://stackoverflow.com/questions/6934752/combining-multiple-commits-before-pushing-in-git

Front-End

- Custom component selectors should be prefixed with *app*-
- Use custom components when they structurally make sense, but don't create separations more than is necessary (a good question to ask is, "is this component likely to be reused somewhere else?" If the answer is no, save breaking out the component for later)
- Unsubscribe subscriptions in ngOnDestroy
- Don't abuse/overuse ngOnChanges
- Separate markup and styling semantically (no inline styling)
- Use BEM (Block Element Modifier) syntax for class names, prefixed with bfa_ (Broadridge Fiduciary Analytics)
- Don't use elements or ids in SCSS - rely on class names (this makes cascading and overriding much easier)
- Avoid deep SCSS nesting