# Code Development Standards

* All development on the codebases should follow the ‘Development Workflow’ as described in the Team Guidelines document

# Angular

Angular code and application architecture should follow the angular style guide (<https://angular.io/guide/styleguide>). The Angular team recommends these practices with good reason.

As a team, we also follow the default rules in the angular lint and editorconfig files with the following modifications:

* Max line length = 175
* Variable names are allowed to begin with an underscore (private class-level variables)

In order to automate code standard reviews, developers should enable their linter in the preferred IDE and use .tslint as the config. In VS Code, installing the ts-lint plugin will take care of this. If feasible, the ng lint command should pass with no errors before pull requests can be merged.

Recommendations for all developers:

* Use Visual Studio Code or JetBrains WebStorm;
* Maintain compatibility with Eclipse IDE;
* Install the ts-lint extension to enforce linting rules in VS Code;
* Install the EditorConfig for VSCode extension to enforce code rules;
* Follow styles and conventions in .editorconfig (this should be automatic in VS Code and WebStorm);
* Don’t change the .editorconfig or tsconfig files without first consulting the entire development team first (this could be done in the form of a PR)

Other considerations:

* Be mindful of using any in TypeScript. It’s best to define classes/interfaces to model an object. Embrace type checking where possible!
* In the event a lot of function chaining is necessary, put each chained function on a newline. Example:

this.\_httpClient.get(‘somepage/asdf’)

.takeUntil(this.\_unsubscribe)

.subscribe((val: Type) => {

// …

});

* Long arrays/objects should use newlines to make it easier to read. Example:

const routes: Route[] = [

{

Path: ‘asdf’,

redirectTo: ‘/feature

},

{

Path: ‘qwerty’,

redirectTo: ‘/anotherfeature’

},

]

* Don’t leave console.log’s in the code when making a PR to merge with develop.

# Java

The client has provided the team with the current CheckStyle files that are used by PC^2.   
Our team will use these provided files as the Java coding standards. This file can be found in the pc2v9 project.

**Checkstyle** is a static code analysis tool used in software development for checking if Java source Code compiles with coding rules. <https://en.wikipedia.org/wiki/Checkstyle>

Checkstyle is built in a JAR file which can run inside a Java VM or as an Apache Ant task. It can also integrate into an IDE or other tools.  
  
A Checkstyle plug-in can provide new functionalities, like:  
  
 • overload syntax coloring or decorations in code editor;  
 • decorate the project explorer to highlight problem-posing resources;  
 • add warnings and errors outputs to the outputs.

Thus, the developer can directly access the code parts highlighted by Checkstyle.

*Follow this guide to install and start using checkstyle:*

<https://examples.javacodegeeks.com/desktop-java/ide/eclipse/eclipse-checkstyle-plugin-example/>

You can right click a file and select “Checkstyle > Check code with checkstyle” and it will highlight and display what does not match with the given style sheet.   
  
If there is not a specified checkstyle file located within the project, the plugin will use the google standards checkstyle instead.