# Leveraging Lint to Create Better Android

**Patrick Hammond**
patrick@madebyatomicrobot.com

@patrickhammond
+PatrickHammond

Creative Commons
Attribution-ShareAlike 3.0 Unported

# For your convenience...

# For your convenience...

Sample code and presentation available here:
https://github.com/patrickhammond/Presentation-CustomAndroidLintChecks

# Who am I?

# Who am I?

# Who am I?

- 4+ amazing years developing on Android

# Who am I?

- 4+ amazing years developing on Android
- Atomic Robot founding partner

# Who am I?

- 4+ amazing years developing on Android
- Atomic Robot founding partner
- Husband, Dad, Geek, Insomniac, Cook, Capsaicin Lover, Craft Beer Fan

# Disclaimer

# Disclaimer

- **Warning:** prepare for jumping between this, coding, and demos. Take your Dramamine now!

# Disclaimer

- **Warning:** prepare for jumping between this, coding, and demos. Take your Dramamine now!
- Sample code is aimed to drive discussion and show concepts only. It is far from polished so use it with caution.

# Outline

# Outline

- Why should I care about Lint?

# Outline

- Why should I care about Lint?
- How can Lint help me?

# Outline

- Why should I care about Lint?
- How can Lint help me?
- Advanced Lint: building your own checks!

# Outline

- Why should I care about Lint?
- How can Lint help me?
- Advanced Lint: building your own checks!
- Discussion

# Why should I care?

Saves you from fighting common issues.

# Why should I care?

Saves you from fighting common issues.

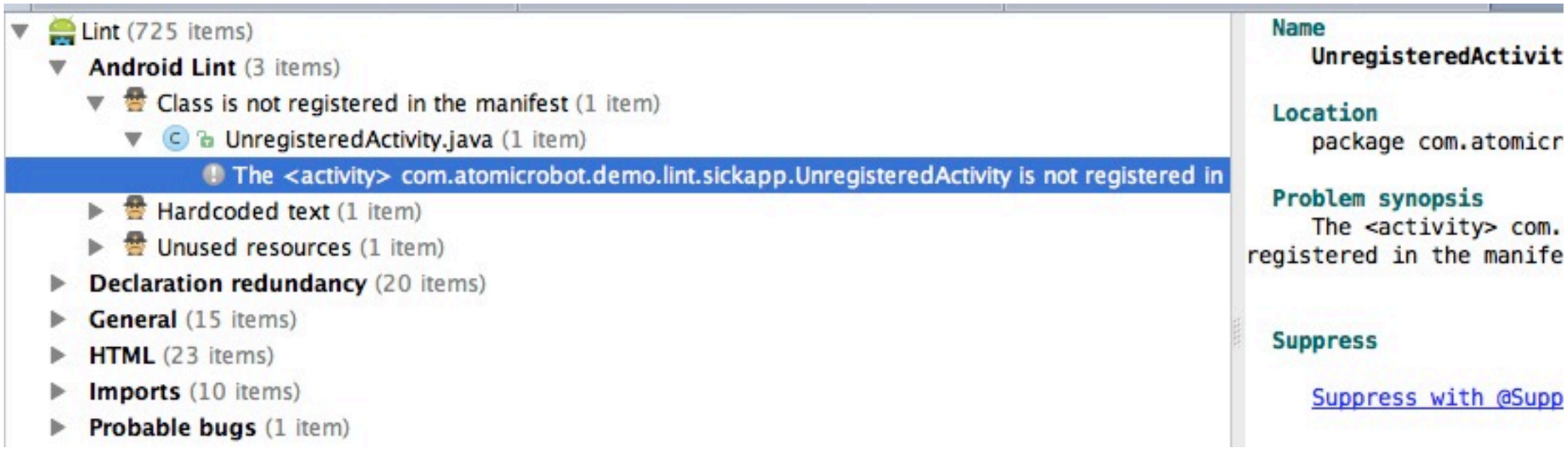
```
11-18 21:58:18.024    1603-1603/com.atomicrobot.demo.lint.sickapp E/AndroidRuntime: FATAL EXCEPTION: main
    android.content.ActivityNotFoundException: Unable to find explicit activity class {com.atomicrobot.demo.lint.sickapp/
 have you declared this activity in your AndroidManifest.xml?
            at android.app.Instrumentation.checkStartActivityResult(Instrumentation.java:1541)
            at android.app.Instrumentation.execStartActivity(Instrumentation.java:1515)
            at android.app.Activity.startActivityFromFragment(Activity.java:3850)
            at android.app.Activity.startActivityFromFragment(Activity.java:3825)
            at android.app.Fragment.startActivity(Fragment.java:996)
            at android.app.Fragment.startActivity(Fragment.java:975)
            at com.atomicrobot.demo.lint.sickapp.MainActivity$PlaceholderFragment.launchUnregisteredActivity(MainActivit
            at com.atomicrobot.demo.lint.sickapp.MainActivity$PlaceholderFragment.access$000(MainActivity.java:54)
            at com.atomicrobot.demo.lint.sickapp.MainActivity$PlaceholderFragment$1.onClick(MainActivity.java:67)
            at android.view.View.performClick(View.java:4084)
            at android.view.View$PerformClick.run(View.java:16966)
            at android.os.Handler.handleCallback(Handler.java:615)
            at android.os.Handler.dispatchMessage(Handler.java:92)
            at android.os.Looper.loop(Looper.java:137)
            at android.app.ActivityThread.main(ActivityThread.java:4745)
            at java.lang.reflect.Method.invokeNative(Native Method)
```

# Why should I care?

Saves you from fighting common issues.

# Why should I care?

Saves you from fighting common issues.



# Why should I care?

Saves you from fighting easy to miss issues.

# Why should I care?

Saves you from fighting easy to miss issues.

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- values/strings.xml -->
<resources>

    <string name="app_name">Lint</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>

</resources>
```

# Why should I care?

Saves you from fighting easy to miss issues.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- values/strings.xml -->
<resources>

    <string name="app_name">Lint</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>

</resources>
```

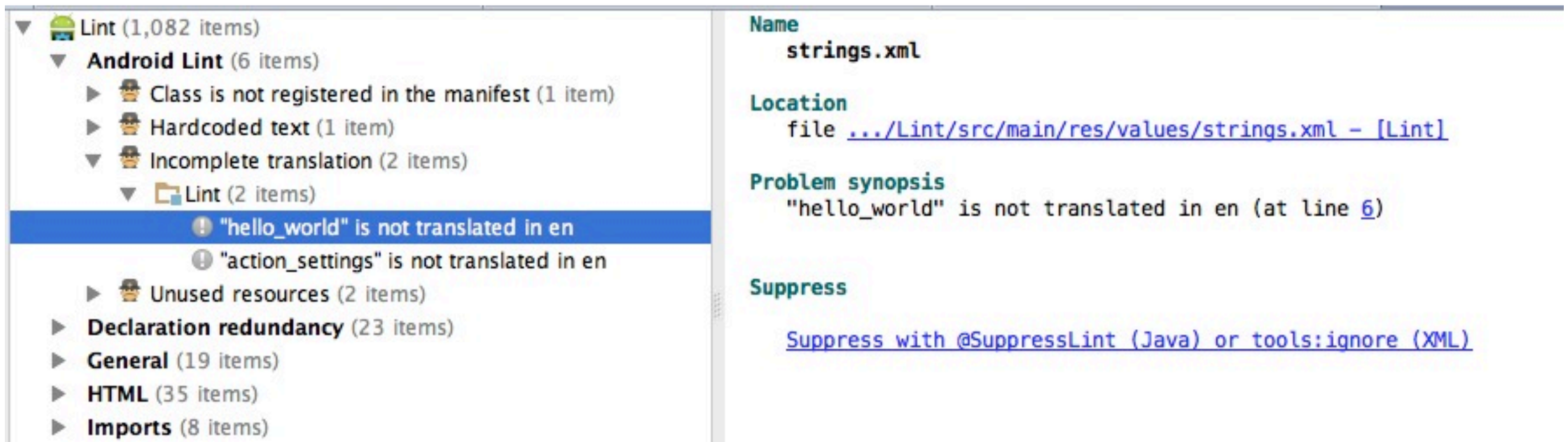
```
<?xml version="1.0" encoding="utf-8"?>
<!-- values-en/strings.xml -->
<resources>

    <string name="app_name">Lint</string>

</resources>
```

# Why should I care?

Saves you from fighting easy to miss issues.

# Why should I care?

Saves you from fighting easy to miss issues.



# Why should I care?

Saves you from fighting insidious issues.

# Why should I care?

Saves you from fighting insidious issues.

```
private void setBackgroundToRed(View view) {
    view.setBackgroundColor(R.color.red);
}
```

# Why should I care?

Saves you from fighting platform issues.

# Why should I care?

Saves you from fighting platform issues.

## How to Leak a Context: Handlers & Inner Classes

Consider the following code:

```
1  public class SampleActivity extends Activity {
2
3    private final Handler mLeakyHandler = new Handler() {
4      @Override
5      public void handleMessage(Message msg) {
6        // ...
7      }
8    }
9  }
```

While not readily obvious, this code can cause cause a massive memory leak. Android Lint will give the following warning: *"In Android, Handler classes should be static or leaks might occur."* But where exactly is the leak and how might it happen? Let's determine the source of the problem by first documenting what we know:

http://www.androiddesignpatterns.com/2013/01/inner-class-handler-memory-leak.html

# Why should I care?

Promotes Android best practices.

# Why should I care?

Promotes Android best practices.

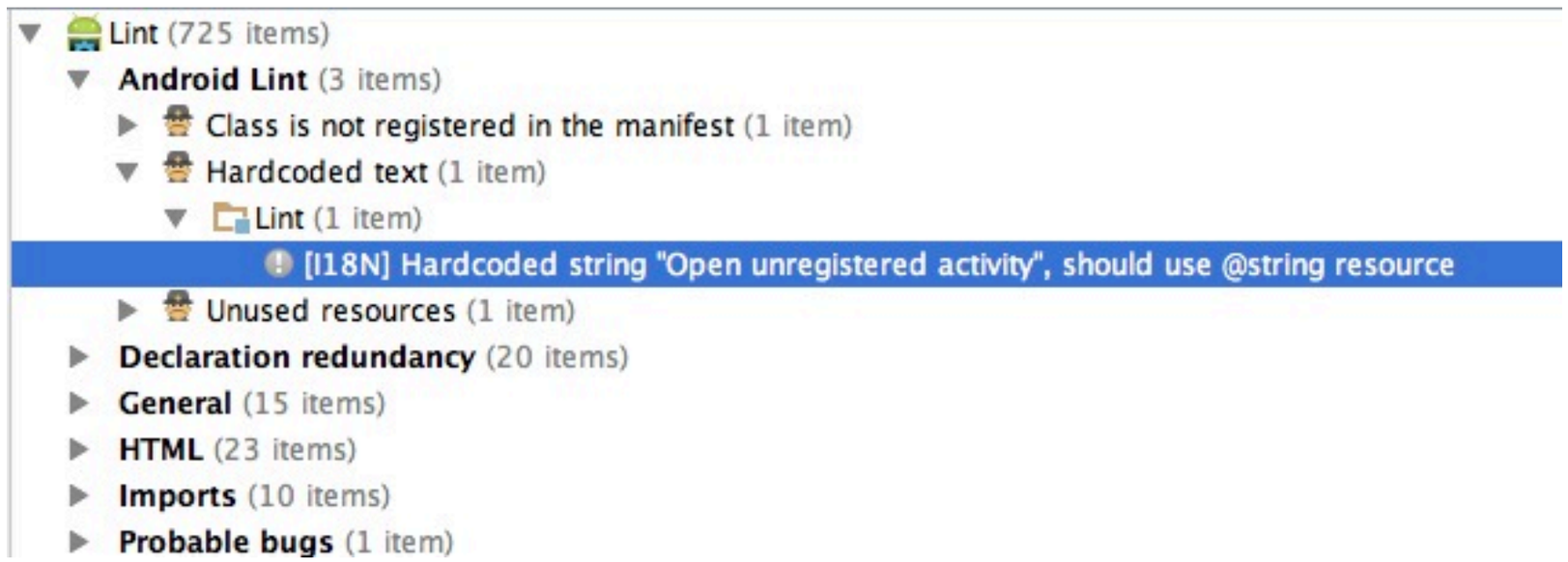
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/androic
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="64dp"
    android:paddingRight="64dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    tools:context=".MainActivity$PlaceholderFragment">

    <Button
        android:id="@+id/action_view_unregistered_activity"
        android:text="Open unregistered activity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

# Why should I care?

Promotes Android best practices.

# Why should I care?

Promotes Android best practices.



# Why should I care?

Easier for you to create great apps!

**Don't ignore Lint warnings!**

In an Android app that I'm working on, we have a Google map in one pane and a ListView of POIs in right pane when the app is landscape mode for tablets. Then in portrait mode (and on smaller devices), we just have a single pane where you can toggle back and forth between the Google Map and the ListView.

I was noticing that in portrait mode performance was fine, but then in landscape mode, with the two panes, the Google Map pane was stuttering and warning about dropped frames in logcat. After spending a lot of time checking my overlays and Android lifecycle methods to make sure I wasn't leaking anything, I finally came across a Lint warning in my layout file. Lint suggested that I use android:baselineAligned="false" on my LinearLayout for better performance. This fixed the performance issues.

I was amazed that one tiny change could have so much effect on the app performance. After much Googling I didn't find a real good reason for why this fixed my performance problem. Best I can figure is that with baselineAligned off, there are less CPU cycles being used to try to align the components when it really isn't needed -- especially when you use layout weights on the children. Maybe one of the Google Android folks (**+Tor Norbye** or **+Romain Guy** or **+Reto Meier**) can explain to me (or point me to an article) that explains why **not** turning off baselineAligned was killing my performance?

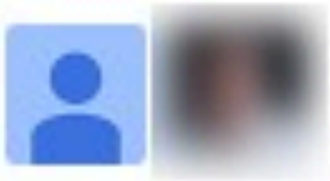
I guess the moral of the story and my lesson learned is *Don't ignore Lint warnings!*

+3 9 Add a comment...

at one tiny change could have so much effect on the app
er much Googling I didn't find a real good reason for why this fixed
problem. Best I can figure is that with baselineAligned off, there are
being used to try to align the components when it really isn't needed
n you use layout weights on the children. Maybe one of the Google
**or Norbye** or **+Romain Guy** or **+Reto Meier**) can explain to me (or
ticle) that explains why **not** turning off baselineAligned was killing
?

l of the story and my lesson learned is *Don't ignore Lint warnings!*

Add a comment...



# How do I start?

# How do I start?

- Command line
  - `lint --html lint.html .`

# How do I start?

- Command line
  - `lint --html lint.html .`
- Gradle
  - `gradlew lint`

# How do I start?

# How do I start?

- IDE: Android Studio/IntelliJ
  - Analyze -> Inspect Code

# How do I start?

- IDE: Android Studio/IntelliJ
  - Analyze -> Inspect Code
- IDE: Eclipse
  - First, Window -> Show View -> Other
  - Then, Android -> Lint Warnings

# Managing Lint

## XML: tools:ignore="HardcodedText"

Need to also add

`xmlns:tools="http://schemas.android.com/tools"`

to the root element of the XML resource.

# Managing Lint

## XML: tools:ignore="HardcodedText"

Need to also add
`xmlns:tools="http://schemas.android.com/tools"`
to the root element of the XML resource.

```
<Button
    android:id="@+id/action_view_unregistered_activity"
    android:text="Open unregistered activity"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    tools:ignore="HardcodedText" />
```

# Managing Lint

Java: @SuppressLint(“ResourceAsColor”)

# Managing Lint

Java: @SuppressLint("ResourceAsColor")

```
private void setBackgroundToRed(View view) {
    view.setBackgroundColor(R.color.red);
}
```

# Managing Lint

Java: @SuppressLint("ResourceAsColor")

```
private void setBackgroundToRed(View view) {
    view.setBackgroundColor(R.color.red);
}
```

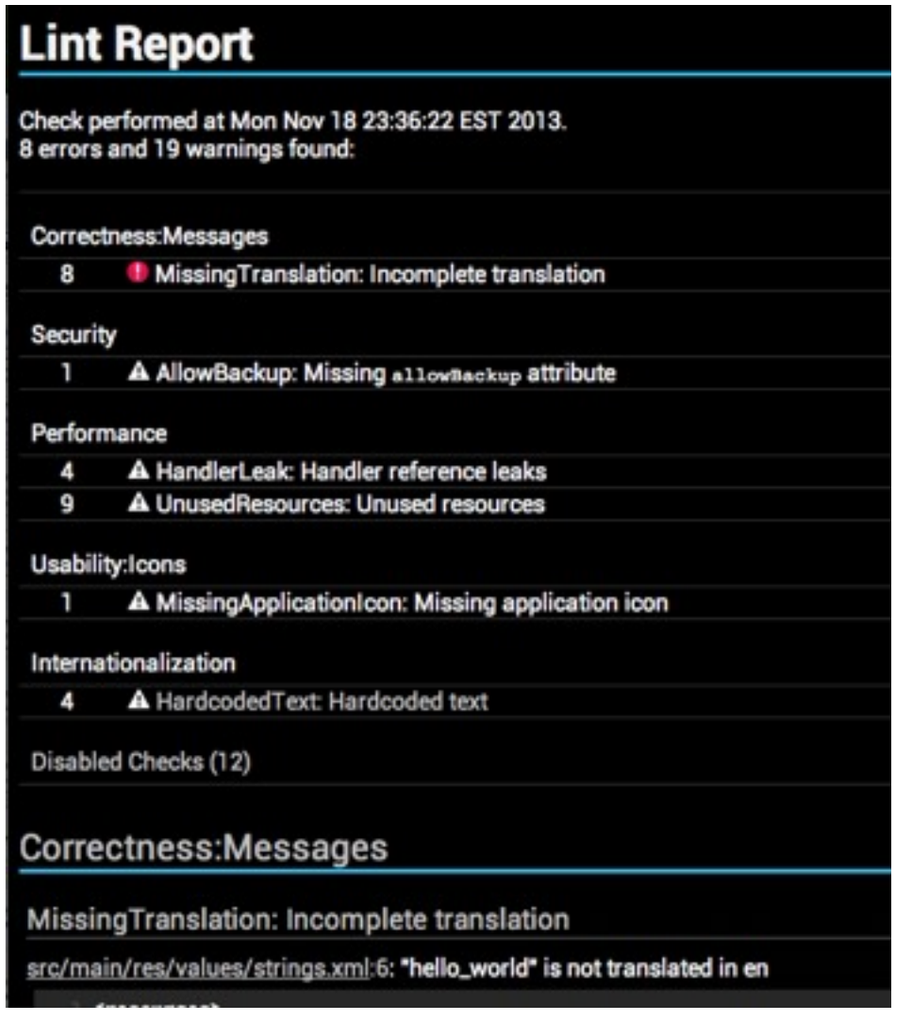
```
// I 'know' better...even though it is very wrong (don't do this)!
@SuppressLint("ResourceAsColor")
private void setBackgroundToRed(View view) {
    view.setBackgroundColor(R.color.red);
}
```

# Managing Lint

Project: lint.xml

# Managing Lint

Project: lint.xml




- Details at bottom of lint HTML report
- 11/18/13 - Bug: Appears to be ignored by Android Studio?

# Custom Lint Checks

- Manifest, resources, Java sources, Java classes, ProGuard
- Look at existing checks!!!
  https://android.googlesource.com/platform/tools/base/+/master/lint/libs/lint-checks/src/main/java/com/android/tools/lint/checks
- Distribution still awkward/undocumented, but recent commits will help.

# Resources

- http://developer.android.com/tools/help/lint.html
- http://developer.android.com/tools/debugging/improving-w-lint.html
- http://tools.android.com/tips/lint
- http://tools.android.com/tips/lint-checks
- http://tools.android.com/tips/lint/writing-a-lint-check
- http://tools.android.com/recent/ignoringlintwarnings


madebyatomicrobot.com

# Questions?

Thank you and happy clean coding!

Email: patrick@madebyatomicrobot.com
Twitter: @patrickhammond
Google Plus: +PatrickHammond