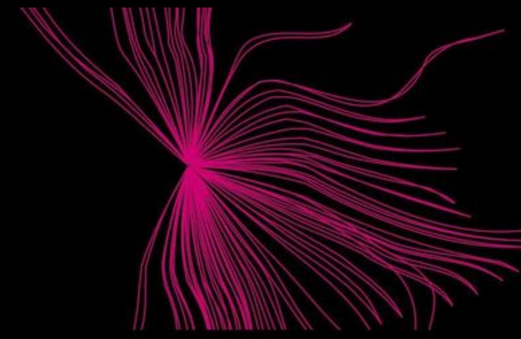
Testing and Code Readability

Fernando Castor



UNIVERSITY
OF TWENTE.

Basic assumption:

You have already checked out the videos and slides

Tests and Readability

Tests are about **finding bugs**.

Readability is about **making bugs less likely**.

But they are also about more than that.

Great programmers worry about this as much as or even more than just writing working code.

Testing

What to test?
(Quick refresher)

- **Unit testing** – testing individual software units
- **Integration testing** –different software components (that may have been tested individually) work together
- **System testing** – an entire system, before handing it over to end users
- **Acceptance testing** – the user accepts the final product
- **Regression tests** – run regularly to check for new bugs
  - Orthogonal to the previous ones

Big companies really like tests

@Google*

*“**Unit Testing is strongly encouraged and widely practiced at Google**. All code used in production is expected to have unit tests, and the code review tool will highlight if source files are added without corresponding tests. Code reviewers usually require that **any change which adds new functionality should also add new tests** to cover the new functionality.”*

*Fergus Henderson. Software Engineering at Google. Feb. 2017. https://arxiv.org/abs/1702.01715

Big companies really like tests
@Facebook*

*"At Facebook, **engineers conduct unit tests for their newly developed code**. In addition, the code **must pass all the accumulated regression tests**, which are administered automatically as part of the commit and push process."*

*Dror G. Feitelson, Eitan Frachtenberg, Kent L. Beck. Fergus Henderson. Development and Deployment at Facebook. IEEE Internet Computing 17(4). July. 2013. https://research.facebook.com/publications/development-and-deployment-at-facebook/

What characteristics do you think (unit) tests should have?

The F.I.R.S.T. principles of testing*

Fast

Independent

Repeatable

Self-checking

Timely

*Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship. Pearson, 1st ed. 2008.

Most importantly, of course, tests should find bugs!!!

(But that would not have fit into the acronym)

```
// Part of class java.util.ArrayList (with a small change)
public void add(int index, E element) {
    rangeCheckForAdd(index);
    modCount++;
    final int s;
    Object[] elementData;
    if ((s = size) > (elementData = this.elementData).length)
        elementData = grow();
    System.arraycopy(elementData, index,
                     elementData, index + 1,
                     s - index);
    elementData[index] = element;
    size = s + 1;
}
```

```java
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class CustomArrayListTest {
   private ArrayList<Integer> list;

    @BeforeEach
    void setUp() {
        list = new ArrayList<>();
    }
    @Test
    void testAddAtStart() {
        list.add(0, 10);
        assertEquals(1, list.size());
        assertEquals(10, list.get(0));
    }
}
```

```
java.lang.ArrayIndexOutOfBoundsException
	at java.lang.System.arraycopy(Native Method)
	at ArrayList.add(ArrayList.java:484)
	at CustomArrayListTest.testAddAtStart(CustomArrayListTest.java:19)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:497)
	at org.junit.platform.commons.util.ReflectionUtils.invokeMethod(ReflectionUtils.java:727)
	at org.junit.jupiter.engine.execution.MethodInvocation.proceed(MethodInvocation.java:60)
	at org.junit.jupiter.engine.execution.InvocationInterceptorChain$ValidatingInvocation.proceed(InvocationInterceptorChain.java:131)
	at org.junit.jupiter.engine.extension.TimeoutExtension.intercept(TimeoutExtension.java:156)
	at org.junit.jupiter.engine.extension.TimeoutExtension.interceptTestableMethod(TimeoutExtension.java:147)
	at org.junit.jupiter.engine.extension.TimeoutExtension.interceptTestMethod(TimeoutExtension.java:86)
	at org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor$$Lambda$125/748658608.apply(Unknown Source)
	at org.junit.jupiter.engine.execution.InterceptingExecutableInvoker$ReflectiveInterceptorCall.lambda$ofVoidMethod$0(InterceptingExecutableInvoker.java:103)
	at org.junit.jupiter.engine.execution.InterceptingExecutableInvoker$ReflectiveInterceptorCalls$$Lambda$126/1641808846.apply(Unknown Source)
	at org.junit.jupiter.engine.execution.InterceptingExecutableInvoker.lambda$invoke$0(InterceptingExecutableInvoker.java:93)
	at org.junit.jupiter.engine.execution.InterceptingExecutableInvoker$$Lambda$292/640363654.apply(Unknown Source)
	at org.junit.jupiter.engine.execution.InvocationInterceptorChain$InterceptedInvocation.proceed(InvocationInterceptorChain.java:106)
	at org.junit.jupiter.engine.execution.InvocationInterceptorChain.proceed(InvocationInterceptorChain.java:64)
	at org.junit.jupiter.engine.execution.InvocationInterceptorChain.chainAndInvoke(InvocationInterceptorChain.java:45)
	at org.junit.jupiter.engine.execution.InvocationInterceptorChain.invoke(InvocationInterceptorChain.java:37)
	at org.junit.jupiter.engine.execution.InterceptingExecutableInvoker.invoke(InterceptingExecutableInvoker.java:92)
	at org.junit.jupiter.engine.execution.InterceptingExecutableInvoker.invoke(InterceptingExecutableInvoker.java:86)
	at org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor.lambda$invokeTestMethod$7(TestMethodTestDescriptor.java:217)
	at org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor$$Lambda$313/1413246829.execute(Unknown Source)
	at org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCollector.java:73)
	at org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor.invokeTestMethod(TestMethodTestDescriptor.java:213)
	at org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor.execute(TestMethodTestDescriptor.java:138)
	at org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor.execute(TestMethodTestDescriptor.java:68)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$6(NodeTestTask.java:151)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$233/706197430.execute(Unknown Source)
	at org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCollector.java:73)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$8(NodeTestTask.java:141)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$232/1568059495.invoke(Unknown Source)
	at org.junit.platform.engine.support.hierarchical.Node.around(Node.java:137)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$9(NodeTestTask.java:139)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$231/1518864111.execute(Unknown Source)
	at org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCollector.java:73)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.executeRecursively(NodeTestTask.java:138)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.execute(NodeTestTask.java:95)
	at org.junit.platform.engine.support.hierarchical.SameThreadHierarchicalTestExecutorService$$Lambda$237/473581465.accept(Unknown Source)
	at java.util.ArrayList.forEach(ArrayList.java:1249)
	at org.junit.platform.engine.support.hierarchical.SameThreadHierarchicalTestExecutorService.invokeAll(SameThreadHierarchicalTestExecutorService.java:41)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$6(NodeTestTask.java:155)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$233/706197430.execute(Unknown Source)
	at org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCollector.java:73)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$8(NodeTestTask.java:141)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$232/1568059495.invoke(Unknown Source)
	at org.junit.platform.engine.support.hierarchical.Node.around(Node.java:137)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$9(NodeTestTask.java:139)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$231/1518864111.execute(Unknown Source)
	at org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCollector.java:73)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.executeRecursively(NodeTestTask.java:138)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.execute(NodeTestTask.java:95)
	at org.junit.platform.engine.support.hierarchical.SameThreadHierarchicalTestExecutorService$$Lambda$237/473581465.accept(Unknown Source)
	at java.util.ArrayList.forEach(ArrayList.java:1249)
	at org.junit.platform.engine.support.hierarchical.SameThreadHierarchicalTestExecutorService.invokeAll(SameThreadHierarchicalTestExecutorService.java:41)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$6(NodeTestTask.java:155)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$233/706197430.execute(Unknown Source)
	at org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCollector.java:73)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$8(NodeTestTask.java:141)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$232/1568059495.invoke(Unknown Source)
	at org.junit.platform.engine.support.hierarchical.Node.around(Node.java:137)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$9(NodeTestTask.java:139)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$231/1518864111.execute(Unknown Source)
	at org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCollector.java:73)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.executeRecursively(NodeTestTask.java:138)
	at org.junit.platform.engine.support.hierarchical.NodeTestTask.execute(NodeTestTask.java:95)
	at org.junit.platform.engine.support.hierarchical.SameThreadHierarchicalTestExecutorService.submit(SameThreadHierarchicalTestExecutorService.java:35)
	at org.junit.platform.engine.support.hierarchical.HierarchicalTestExecutor.execute(HierarchicalTestExecutor.java:57)
	at org.junit.platform.engine.support.hierarchical.HierarchicalTestEngine.execute(HierarchicalTestEngine.java:54)
	at org.junit.platform.launcher.core.EngineExecutionOrchestrator.execute(EngineExecutionOrchestrator.java:147)
	at org.junit.platform.launcher.core.EngineExecutionOrchestrator.execute(EngineExecutionOrchestrator.java:127)
	at org.junit.platform.launcher.core.EngineExecutionOrchestrator.execute(EngineExecutionOrchestrator.java:90)
	at org.junit.platform.launcher.core.EngineExecutionOrchestrator.lambda$execute$0(EngineExecutionOrchestrator.java:55)
	at org.junit.platform.launcher.core.EngineExecutionOrchestrator$$Lambda$175/13326370.accept(Unknown Source)
	at org.junit.platform.launcher.core.EngineExecutionOrchestrator.withInterceptedStreams(EngineExecutionOrchestrator.java:102)
	at org.junit.platform.launcher.core.EngineExecutionOrchestrator.execute(EngineExecutionOrchestrator.java:54)
	at org.junit.platform.launcher.core.DefaultLauncher.execute(DefaultLauncher.java:114)
	at org.junit.platform.launcher.core.DefaultLauncher.execute(DefaultLauncher.java:86)
	at org.junit.platform.launcher.core.DefaultLauncherSession$DelegatingLauncher.execute(DefaultLauncherSession.java:86)
	at org.junit.platform.launcher.core.SessionPerRequestLauncher.execute(SessionPerRequestLauncher.java:53)
	at com.intellij.junit5.JUnit5IdeaTestRunner.startRunnerWithArgs(JUnit5IdeaTestRunner.java:57)
	at com.intellij.rt.junit.IdeaTestRunner$Repeater$1.execute(IdeaTestRunner.java:38)
	at com.intellij.rt.execution.junit.TestsRepeater.repeat(TestsRepeater.java:11)
	at com.intellij.rt.junit.IdeaTestRunner$Repeater.startRunnerWithArgs(IdeaTestRunner.java:35)
	at com.intellij.rt.junit.JUnitStarter.prepareStreamsAndStart(JUnitStarter.java:232)
	at com.intellij.rt.junit.JUnitStarter.main(JUnitStarter.java:55)


Process finished with exit code 255
```

```
// Part of class java.util.ArrayList (with a small change)
public void add(int index, E element) {
    rangeCheckForAdd(index);
    modCount++;
    final int s;
    Object[] elementData;
    if ((s = size) > (elementData = this.elementData).length)
        elementData = grow();
    System.arraycopy(elementData, index,
                     elementData, index + 1,
                     s - index);
    elementData[index] = element;
    size = s + 1;
}
```

```
// Part of class java.util.ArrayList (with a small change)
public void add(int index, E element) {
    rangeCheckForAdd(index);
    modCount++;
    final int s;
    Object[] elementData;
    if ((s = size) == (elementData = this.elementData).length)
        elementData = grow();
    System.arraycopy(elementData, index,
                     elementData, index + 1,
                     s - index);
    elementData[index] = element;
    size = s + 1;
}
```

Is this test good enough?

It even found a bug…

A reliable approach to find every bug in a program: test every possible program behavior!

Obviously not possible.
Unreasonably expensive.

How to know what is sufficient and relevant?

Selecting meaningful input

Leverage **equivalence classes**

- Break the input into groups that are likely to trigger the same behavior

Table 1: Brackets for wage tax/national insurance contributions 2022

Grade	Annual wages	Payroll taxes contribution	
		Up to state pension age	State pension age and older, born in 1946 or later
1a	up to and including € 35,472	37.07%	19.17%
1b	€ 35,473 - € 69,398	37.07%	37.07%
2	€ 69,399 or more	49.50%	49.50%

Selecting meaningful input

Leverage **equivalence classes**

- Break the input into groups that are likely to trigger the same behavior

Target the **limit values** of different classes

Establishing equivalence classes may be **hard**

- For example, in a compiler, we must test individual language features and their combinations. Similarly for games.

```
...

class CustomArrayListTest {
    ...
    @Test
    void testAddAtStart() { ... }

    @Test
    void testAddAtMiddle() {
        list.add(0, 10);
        list.add(1, 20);
        list.add(1, 15); // Add at the middle
        assertEquals(3, list.size());
        ...
    }
    @Test
    void testAddAtEnd() { ... }
}
```

Test coverage

Measuring test code comprehensiveness

A useful metric to keep track of **how much of your code is tested**.

The percentage of the code that is run through a (unit) test.

Different **criteria**: statement, branch, line, etc.

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
org.jacoco.core		97%		91%	148	1,546	125	3,654	19	747	2	147
org.jacoco.examples		58%		64%	24	53	97	193	19	38	6	12
org.jacoco.agent.rt		75%		83%	32	130	75	344	21	80	7	22
jacoco-maven-plugin		90%		82%	35	193	49	465	8	116	1	23
org.jacoco.cli		97%		100%	4	109	10	275	4	74	0	20
org.jacoco.report		99%		99%	4	572	2	1,345	1	371	0	64
org.jacoco.ant		98%		99%	4	162	8	428	3	110	0	19
org.jacoco.agent		86%		75%	2	10	3	27	0	6	0	1
Total	1,438 of 29,017	95%	188 of 2,402	92%	253	2,775	369	6,731	75	1,542	16	308

Source: https://www.jacoco.org/jacoco/trunk/coverage/

Code coverage @ Google

Data from 2019*

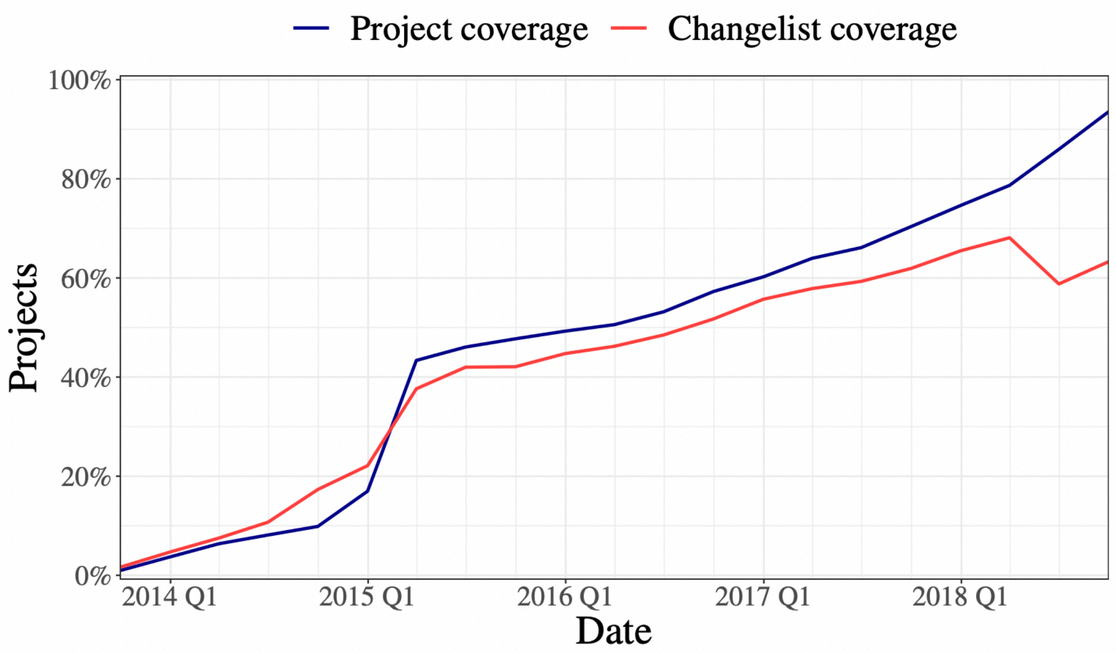


Figure 7: Projects actively using coverage automation.
If a project that has coverage automation enabled does not actively use it (e.g., no daily project coverage computation is successful), it does not contribute towards the active count.

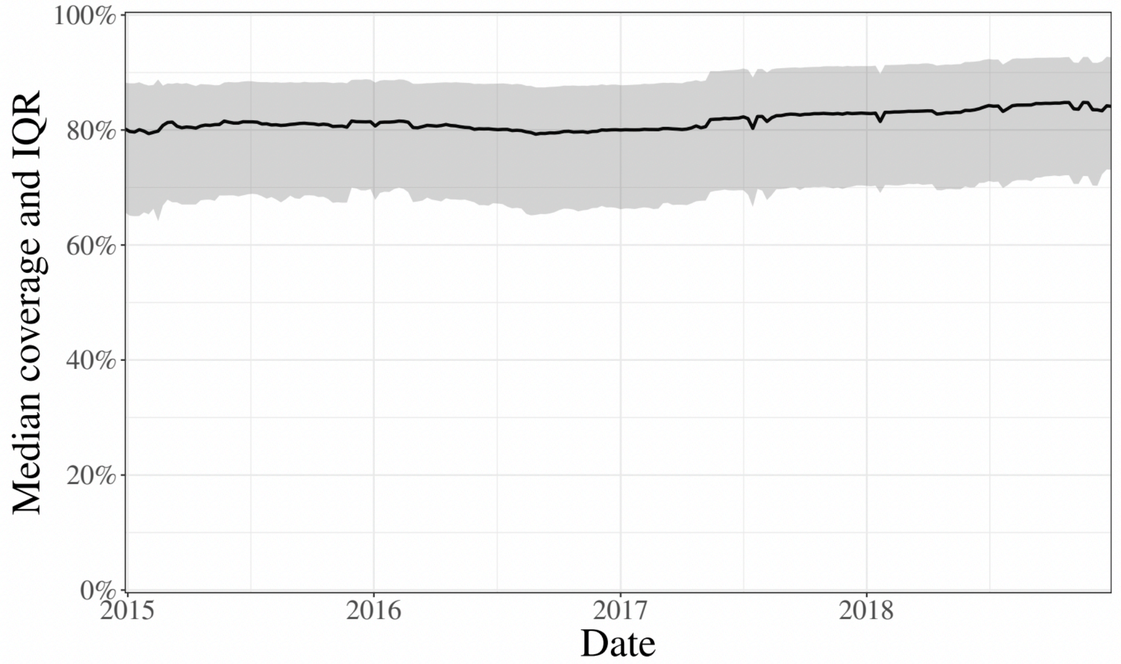


Figure 8: Median weekly project coverage.
The weekly project coverage is the median of the daily project coverages in a given week. If a project had no successful coverage computation in a given week, it is excluded from the aggregation for that week. Note that we exclude data older than 2015 because fewer than 20% of projects were using coverage before 2015 (see Figure 7).

*Marko Ivanković, Goran Petrović, René Just, and Gordon Fraser. 2019. Code coverage at Google. In Proc. 27th ESEC/FSE, 2019, pp 955–963. ACM.

```
// Additional situations beyond just partitioning the input
// Now we are thinking about situations internal to the data structure
// Somewhat covered by adding at the middle.
...
class CustomArrayListTest {
    ...
    // The following also checks whether elements are shifted correctly.
    // The important thing is to consider this additional internal aspect.
    @Test
    void testAddAtMiddle() { ... }

    @Test
    void testAddWhenArrayGrows() { ... }
}
```

We should also test exceptional cases!

The scenarios where the element under test is expected to fail

The goal is to verify that it reacts as expected

- i.e., that the correct exceptions/errors are produced

```
...

class CustomArrayListTest {
    ...
    @Test
    void testAddWithIndexOutOfBounds() {
        assertThrows(IndexOutOfBoundsException.class,
                     () -> list.add(1, 10)); // Invalid idx for empty list
        list.add(0, 10);
        assertThrows(IndexOutOfBoundsException.class,
                     () -> list.add(2, 20)); // Index greater than size
    }


}
```

A little bit on readability

You should write readable code

Linux kernel coding style

This is a short document describing the preferred coding style for the linux kernel. Coding style is very personal, and I won't **force** my views on anybody, but this is what goes for anything that I have to be able to maintain, and I'd prefer it for most other things too. Please at least consider the points made here.

First off, I'd suggest printing out a copy of the GNU coding standards, and NOT read it. Burn them, it's a great symbolic gesture.

Anyway, here goes:

1) Indentation

Tabs are 8 characters, and thus indentations are also 8 characters. There are heretic movements that try to make indentations 4 (or even 2!) characters deep, and that is akin to trying to define the value of PI to be 3.

Common C# code conventions

Article • 08/01/2023 • 12 contributors

Feedback

In this article

Coding conventions are essential for maintaining code readability, consistency, and collaboration within a development team. Code that follows industry practices and established guidelines is easier to understand, maintain, and extend. Most projects enforce a consistent style through code conventions. The dotnet/docs and dotnet/samples projects are no exception. In this series of articles, you learn our coding conventions and the tools we use to enforce them. You can take our conventions as-is, or modify them to suit your team's needs.

We chose our conventions based on the following goals:

github.com/airbnb/javascript

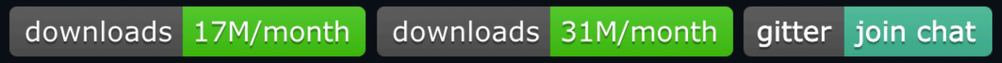
QUANTUM | You searched for... | Read-Along: Struc...

README | Code of conduct | MIT license | Security

Airbnb JavaScript Style Guide() {

A mostly reasonable approach to JavaScript

> **Note**: this guide assumes you are using Babel, and requires that you use babel-preset-airbnb or the equivalent. It also assumes you are installing shims/polyfills in your app, with airbnb-browser-shims or the equivalent.



This guide is available in other languages too. See Translation

Other Style Guides

- ES5 (Deprecated)
- React
- CSS-in-JavaScript
- CSS & Sass
- Ruby

google.github.io/styleguide/

QUANTUM

Yead-Along: Struc...

READM

styleguide

Google Style Guides

Every major open-source project has its own style guide: a set of conventions (sometimes arbitrary) about how to write code for that project. It is much easier to understand a large codebase when all the code in it is in a consistent style.

"Style" covers a lot of ground, from "use camelCase for variable names" to "never use global variables" to "never use exceptions." This project (google/styleguide) links to the style guidelines we use for Google code. If you are modifying a project that originated at Google, you may be pointed to this page to see the style guides that apply to that project.

- AngularJS Style Guide
- Common Lisp Style Guide
- C++ Style Guide
- C# Style Guide
- Go Style Guide
- HTML/CSS Style Guide
- JavaScript Style Guide
- Java Style Guide
- JSON Style Guide
- Markdown Style Guide
- Objective-C Style Guide
- Python Style Guide
- R Style Guide
- Shell Style Guide
- Swift Style Guide
- TypeScript Style Guide
- Vim script Style Guide

It’s not just a matter of conventions

When you write code …

- you are writing instructing the computer on how to behave
- but you are also **teaching** something to a developer

Understanding programs is just learning*

That developer may be **you**!

*Delano Oliveira, Reydne Bruno, Fernanda Madeiral, Fernando Castor. Evaluating Code Readability and Legibility: An Examination of Human-centric Studies. ICSME 2020: 348-359

```
# I am really not proud of this code snippet I wrote. It’s too complex.
readingsPostIdentifier:List[str] = \
    [" ".join(filterEmptyStr([paren1, pp1, pp2.format(len(parameters)), pl, parS, paren2, postS]))
     for paren1 in parenthesis for pp1 in parPrefix1
     for pp2 in parPrefix2 for pl in parameterLists for parS in parSuffix
     for paren2 in parenthesis
     for postS in postSuffix
     if not(paren1!="" and pp1 != "") and not(paren1!="" and pp2 != "")
     and not (paren2!="" and paren1=="")
     and (pp2=="" or pp1!="") and (parS=="" or pp1!="") and not(parS!=""
     and pp2!="") # A => B == not A or B
     and not(pp1=="" and pp2=="" and paren1=="" and len(parameters) != 0)
     and not(len(parameters) > 0 and (pp1 == "no" or pp1 == "without" or pp1 == "zero"))
     and not(len(parameters) == 0 and (not ((pp1 == "no" or pp1 == "without" or pp1 == "zero")
     or pp2 == "") or paren1==""))
    ]
```

```
# I am really not proud of this code snippet I wrote. It's too complex.
readingsPostIdentifier:List[str] = \
    [" ".join(filterEmptyStr([paren1, pp1, pp2.format(len(parameters)), pl, parS, paren2, postS]))
     for paren1 in parenthesis for pp1 in parPrefix1
     for pp2 in parPrefix2 for pl in parameterLists for parS in parSuffix
     for paren2 in parenthesis
     for postS in postSuffix
     if not(paren1!="" and pp1 != "") and not(paren1!="" and pp2 != "")
     and not (paren2!="" and paren1=="")
     and (pp2=="" or pp1!="") and (parS=="" or pp1!="") and not(parS!=""
     and pp2!="") # A => B == not A or B
     and not(pp1=="" and pp2=="" and paren1=="" and len(parameters) != 0)
     and not(len(parameters) > 0 and (pp1 == "no" or pp1 == "without" or pp1 == "zero"))
     and not(len(parameters) == 0 and (not ((pp1 == "no" or pp1 == "without" or pp1 == "zero")
     or pp2 == "") or paren1==""))
    ]
```

Understanding code is really **important**

*“On average, developers spend **70%** of their time performing program comprehension”*[1]

*“On average developers spend **~58%** of their time on program comprehension activities”*[2]

*“[…] participants spend almost **30%** of their in-IDE time on code editing and execution and another **22%** navigating documents and source code.”*[3]

[1] R. Minelli, A. Mocci and M. Lanza, "I Know What You Did Last Summer - An Investigation of How Developers Spend Their Time," 2015 IEEE 23rd International Conference on Program Comprehension, Florence, Italy, 2015, pp. 25-35
[2] X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan and S. Li, "Measuring Program Comprehension: A Large-Scale Field Study with Professionals," in IEEE Transactions on Software Engineering, vol. 44, no. 10, pp. 951-976
[3] S. Amann, S. Proksch, S. Nadi and M. Mezini, "A Study of Visual Studio Usage in Practice," 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Osaka, Japan, 2016, pp. 124-134

```
// Follow the naming convention, or your program will be hard to read!
public void Add(int InDeX, E Element) {
    range_check_for_add(InDeX);
    mod_COUNT++;
    final int S;
    Object[] ELEMENT_DATA;
    if ((S = size) == (ELEMENT_DATA = this.elementData).length)
        ELEMENT_DATA = grow();
    System.arraycopy(ELEMENT_DATA, InDeX,
                     ELEMENT_DATA, InDeX + 1,
                     S - InDeX);
    ELEMENT_DATA[InDeX] = Element;
    size = S + 1;
}
```

```java
// This adheres to the Java standard. Better.
public void add(int index, E element) {
    rangeCheckForAdd(index);
    modCount++;
    final int s;
    Object[] elementData;
    if ((s = size) == (elementData = this.elementData).length)
        elementData = grow();
    System.arraycopy(elementData, index,
                     elementData, index + 1,
                     s - index);
    elementData[index] = element;
    size = s + 1;
}
```

```java
// A LotR-themed ArrayList class. Poor variable naming is really bad.
public void gandalf(int frodo, E sauron) {
    rangeCheckForGandalf(frodo);
    modCount++;
    final int legolas;
    Object[] sauronData;
    if ((legolas = size) == (sauronData = this.sauronData).length)
        sauronData = grow();
    System.arraycopy(sauronData, frodo,
                     sauronData, frodo + 1,
                     legolas - frodo);
    sauronData[frodo] = sauron;
    size = legolas + 1;
}
```

```
// complex control flow/code structure is another no-no
readingsPostIdentifier:List[str] = \
    [" ".join(filterEmptyStr([paren1, pp1, pp2.format(len(parameters)), pl, parS, paren2, postS]))
     for paren1 in parenthesis for pp1 in parPrefix1
     for pp2 in parPrefix2 for pl in parameterLists for parS in parSuffix
     for paren2 in parenthesis
     for postS in postSuffix
     if not(paren1!="" and pp1 != "") and not(paren1!="" and pp2 != "")
     and not (paren2!="" and paren1=="")
     and (pp2=="" or pp1!="") and (parS=="" or pp1!="") and not(parS!=""
     and pp2!="") # A => B == not A or B
     and not(pp1=="" and pp2=="" and paren1=="" and len(parameters) != 0)
     and not(len(parameters) > 0 and (pp1 == "no" or pp1 == "without" or pp1 == "zero"))
     and not(len(parameters) == 0 and (not ((pp1 == "no" or pp1 == "without" or pp1 == "zero")
     or pp2 == "") or paren1==""))
    ]
```

Seven levels of nesting!

Back to the Linux style guide

1) Indentation

Tabs are 8 characters, and thus indentations are also 8 characters. There are heretic movements that try to make indentations 4 (or even 2!) characters deep, and that is akin to trying to define the value of PI to be 3.

Rationale: The whole idea behind indentation is to clearly define where a block of control starts and ends. Especially when you've been looking at your screen for 20 straight hours, you'll find it a lot easier to see how the indentation works if you have large indentations.

Now, some people will claim that having 8-character indentations makes the code move too far to the right, and makes it hard to read on a 80-character terminal screen. The answer to that is that if you need more than 3 levels of indentation, you're screwed anyway, and should fix your program.

One last word: Testing

Tests are **executable code documentation**

- The best type. Never gets outdated!
- Don’t forget: the tests should be 100% automated

Another strong reason to write tests, besides finding bugs: they **explain** the code