Code Review Best Practice

Trisha Gee (@trisha_gee)
Developer Advocate & Java Champion, JetBrains

```java
/**
 * Returns a set containing the names of all collections in this database.
 * @return the names of collections in this database
 * @throws MongoException
 */
public Set<String> getCollectionNames(){

    DBCollection namespaces = getCollection("system.namespaces");
    if (namespaces == null)
        throw new RuntimeException("this is impossible");

    Iterator<DBObject> i = namespaces._find(new BasicDBObject(), null, 0, 0, 0, getOptions(), getReadPreference(), null);
    if (i == null)
        return new HashSet<String>();

    List<String> tables = new ArrayList<String>();

    for (; i.hasNext();) {
        DBObject o = i.next();
        if ( o.get( "name" ) == null ){
            throw new MongoException( "how is name null : " + o );
        }
        String n = o.get("name").toString();
        int idx = n.indexOf(".");

        String root = n.substring(0, idx);
        if (!root.equals(_name))
            continue;

        if (n.indexOf("$") >= 0)
            continue;

        String table = n.substring(idx + 1);

        tables.add(table);
    }

    Collections.sort(tables);

    return new LinkedHashSet<String>(tables);
}
```

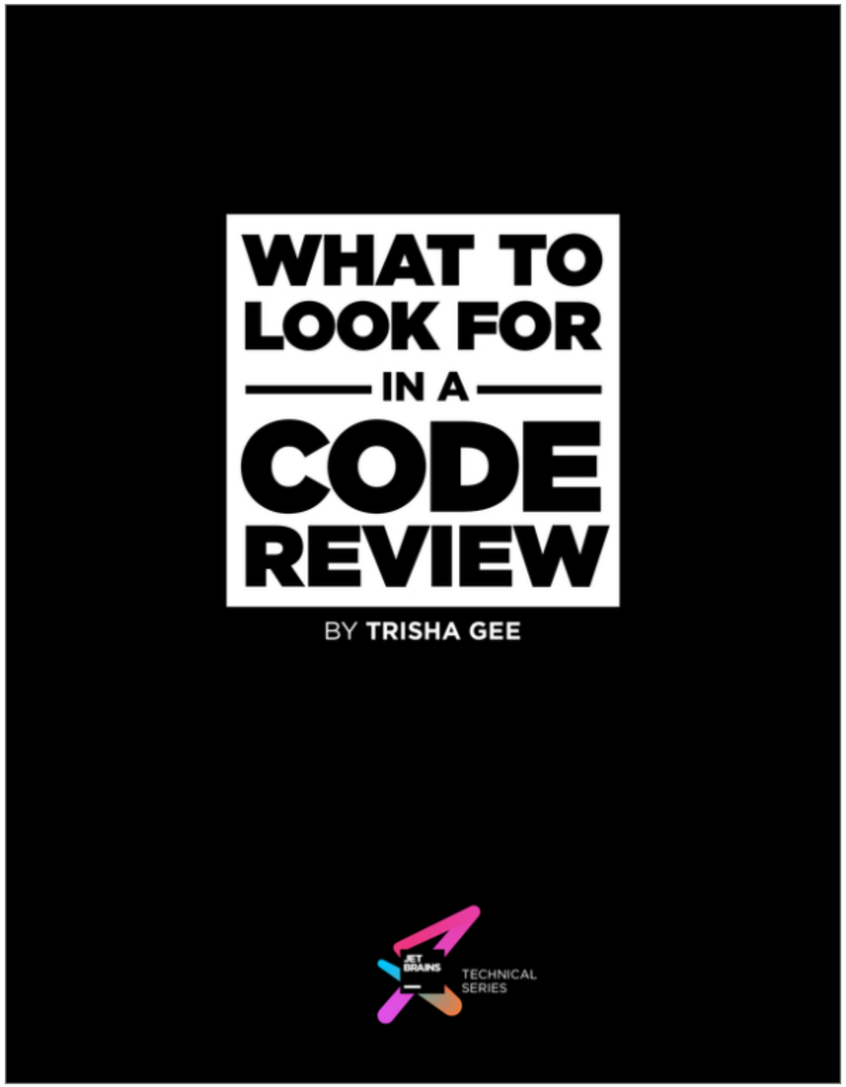
This code works

Having Opinions On Code Is An Occupational Hazard

Having Opinions On Code Is An Occupational Requirement

Are we harder on other people's code than our own?

What should we be looking for in a Code Review?



http://jb.gg/book/codereview

Different workflows

- Gateway Reviews
- Knowledge Sharing
- Early Design Feedback

https://blog.jetbrains.com/upsource/tag/code-review-workflows/

What should you look for when reviewing code?

It Depends

My First Code Review

My job is to **Find Problems**

Anti-Pattern

Nit picking

Anti-Pattern

Design changes when the code works

Anti-Pattern

Inconsistent feedback

Anti-Pattern

The Ghost Reviewer

Anti-Pattern

Ping pong reviews

Developers hate code reviews

Code Reviews are a Massive Waste of Time

Take a step back...

1. Why?

Ensure code meets standards

Find bugs

Ensure code does what it’s supposed to

Check code is understandable

Share knowledge

Collaborate on design

Evolve application code

2. When?

When do you review?

- During implementation?
- When it’s ready to merge?
- After it’s been merged?

When is the review complete?

- When everyone agrees?
- When a gatekeeper agrees?
- When all comments are addressed?

3. Who?

Who reviews the code?

Who signs it off?

4. Where?

Pairing

Showing code to a colleague at a computer

Mob reviewing in a conference room

Remote screen-sharing

In the IDE, checking out a commit or branch

Using code review software

5. What?

Requires you to know:

1. Why
2. When
3. Who
4. Where

What to look for

- Fit with the overall architecture
- SOLID principles, Domain Driven Design, Design Patterns or other paradigms of choice
- New code follows team's current practices
- Code is in the right place
- Code reuse
- Over-engineering
- Readable code and tests
- Testing the right things
- Exception error messages
- Subtle bugs
- Security
- Regulatory requirements
- Performance
- Documentation and/or help files been updated
- Spelling, punctuation & grammar on user messages

https://blog.jetbrains.com/upsource/tag/what-to-look-for/

Human reviewers should be doing what cannot be automated

Understand the constraints

Why: Knowledge Sharing

Purpose isn't to reject the code

Why: Knowledge Sharing

Focus is on learning what the code does and why

When: At the End

Too late for design

When: At the End

Should have set of checks

6. How?

Automate Everything You Can

Submitting for review

Reviews should be small

Submitting for review

Annotate your code

Reviewing

Should be clear **Who** is reviewing

Reviewing

Respond in a timely fashion

Reviewing

Checklist of **What** to look for

Comments

Bear in mind **Why**, **When** and **What**

Comments

Be constructive

Comments

Be specific

Accept or Reject

Accept or Raise Concern

Next steps should be clear

Making changes

Respond in a timely fashion

Making changes

Respond to comments

Resolving

The goal is to accept the review

Resolving

Should be clear **Who** signs it off

Resolving

...and **When**

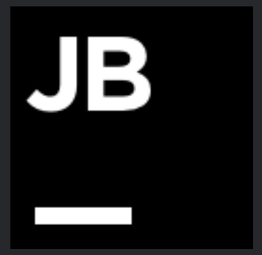
Code Reviews Suck Less When...

The process is clear

1. Why
2. When
3. Who
4. Where
5. What
6. How

The Goal Is To Ship The Code

Not to prove how clever you are



http://bit.ly/CRGood