Code reviews

Code reviews are what we are doing now: a run through of someone's code to check for correctness and code quality. They are useful because another pair of eyes can catch bugs and problems easier than the person writing the code.

Some considerations during code reviews:

1) Know what you are reviewing. Knowing the code context is important.
2) Look for bad code smells. Refer to the refactoring book in the description for this.
3) Factors for good code: readability, correctness, performance.
4) If you are unsure, do not sign off the review.
5) Expect tests backing up the author's claims.

Things to keep in mind:

1) Look for problems. Only *suggest* solutions.
2) Don't have long discussions of solutions with other reviewers here. Set a meeting or make a separate report. (Link it to the review later)
3) Don't try and score points during the review. Don't mention irrelevant or complex solutions to sound cool. (Especially for the fresher engineers)
4) Don't be nitpicky to sign-off. Fix the code yourself if you have a lot of nitpicky issues. Trust your teammates to improve themselves as they spend time looking at your code.
5) Offload general issues to the build manager. Styling, formatting, conventions etc… can be parser checks. Test coverage must be compulsory before sending a pull request, etc…

Some tricks for readability:

1) Use an IDE when doing the review.
2) Move magic numbers and strings to constants and Enums.
3) Complex maps should be objects. Nested loops should be broken into functions or graph searches.
4) Name variables and functions by what they do instead of how or why they do it.
5) Move *common* object conversions, validations and other pieces of code to common Utils.

Some tricks for performance:

1) Use singletons and resource pools where you can.
2) Constructing and Configuring a client is tricky. There should be dependency injection here.
3) IO calls are expensive. Use caching, request collapsing and batch queries to reduce load.

Some tricks for resource management:

1) Make sure resources are being released.
2) Make sure that race conditions are dealt with.
3) Logging is critical when handling resources. Log with context.