General Principles of API Design

Functionality

- F1. API should do one thing and do it well
- F2. API should be as small as possible but no smaller–when in doubt, leave it out
- F3. Make it easy to do what’s common & possible to do what’s less so

Usability (of the API) - Reminder: Naming has enormous impact on usability

- U1. Don’t make users do anything library could do for them
- U2. Be consistent
- U3. Use existing standards and APIs where appropriate
- U4. Conversely, new API is a chance to start fresh
- U5. An API must be appropriate to its audience
- U6. APIs should be approachable
- U7. APIs should be discoverable
- U8. Monitor complexity constantly

Quality (of client code that uses the API)

- Q1. Minimize mutability
- Q2. Minimize accessibility
- Q3. Make it easy for user to do what’s preferable
  - Q3a. If you support strict and lenient behaviors, strict should be default (“tough love”)
- Q4. Prevent failure, or fail quickly, predictably, and informatively (“fail fast”)
- Q5. Handle boundary conditions (edge cases, corner cases) gracefully
- Q6. Don’t place arbitrary limits on sizes
- Q7. Don’t force users to pay for functionality they don’t want or need
- Q8. Consider Performance Consequences of API Design Decisions
- Q9. Avoid leaky abstractions
- Q10. Some things should be left to experts–don’t write your own crypto protocols or APIs

Longevity (aging gracefully)

- L1. Design for evolution
- L2. Implementation should not impact API
- L3. Hyrum’s Law (“With a sufficient number of users, it does not matter what you promise in the API’s contract: all observable behaviors of your system will be depended on by somebody.”)
- L4. Beware of fads