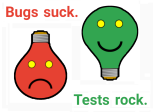

What's in a Name?

by Adam Raider

"There are only two hard things in computer science: cache invalidation and naming things." —Phil Karlton

Have you ever read an identifier only to realize later it doesn't do what you expected? Or had to read the implementation in order to understand an interface? These indirections eat up our cognitive bandwidth and make our work more difficult. We spend far more time reading code than we do writing it; **thoughtful names can save the reader (and writer) a lot of time and frustration.** Here are some naming tips:

- **Spend time considering names—it's worth it.** Don't default to the first name that comes to mind. The more public the name, the more expensive it is to change. Past a certain scale, names become infeasible to change, especially for APIs. Pay attention to a name in proportion to the cost of renaming it later. If you're feeling stuck, consider running a new name by a teammate.
- **Describe behavior.** Encourage naming based on *what* functions do rather than *when* the functions are called. Avoid prefixes like "handle" or "on" as they describe *when* and provide no added meaning:

`button.listen('click', handleClick)`	`button.listen('click', addItemToCart)`

- **Reveal intent with a contextually appropriate level of abstraction**:
  - High-abstraction functions describe the *what* and operate on high-level types.
  - Lower-abstraction functions describe the *how* and operate on lower-level types.

  For example, `logout` might call into `clearUserToken`, and `recordWithCamera` might call into `parseStreamBytes`.
- **Prefer unique, precise names.** Are you frequently asking for the `UserManager`? `Manager`, `Util`, and similar suffixes are a common but imprecise naming convention. What does it do? It manages! If you're struggling to come up with a more precise name, consider splitting the class into smaller ones.
- **Balance clarity and conciseness—use abbreviations with care.** Commonly used abbreviations, such as HTML, i18n, and RPC, can aid communication but less-known ones can confuse your average readers. Ask yourself, "Will my readers immediately understand this label? Will a reader five years from now understand it?"
- **Avoid repetition and filler words.** Or in other words, don't say the same thing twice. It adds unnecessary visual noise:

`userData.userBirthdayDate`	`user.birthDate`

- **Software changes—names should, too.** If you see an identifier that doesn't aptly describe itself—fix it!

Learn more about identifier naming in this post: IdentifierNamingPostForWorldWideWebBlog.

More information and archives: `testing.googleblog.com`