Choose Specific Words

The main idea is that it's crucial to pick clear and meaningful names when writing code. Words like "get" or unclear names like "Size()" can make it hard to understand what a function does. It's better to use names that explain exactly what the function does, like "FetchPage()", "NumNodes()", or "Pause()". You can use a thesaurus or ask others for help finding the right words. But remember, it's most important to be clear and precise, even if that means the names aren't fancy.

Avoid Generic Names Like tmp and retval

This part highlights that using clear and meaningful names for variables in code is important. Names like "tmp" or simple letters like "i," "j," "k" are easy, but they don't tell much about what the variable does. Instead, choosing names that explain the variable's purpose, like "sum_squares" instead of "retval," or specific loop names like "club_i," "members_i," and "users_i," can make the code easier to understand and less likely to have mistakes. However, there are times when simple names are okay, like for temporary things. The main idea is to use simple names wisely and prefer descriptive names to make code clearer and easier to manage.

Prefer Concrete Names over Abstract Names

The main idea here is to give things specific names that clearly show what they do. When you name variables, functions, and other parts of your code in a way that describes their purpose, it helps people read and understand the code better. Google's coding rules show that using vague or confusing names can cause problems. For example, changing the name of a code part from DISALLOW_EVIL_CONSTRUCTORS to DISALLOW_COPY_AND_ASSIGN makes it easier to understand. Also, using names like --extra_logging and --use_local_database instead of --run_locally makes the code clearer, even if it means using more words. So, it's better to prioritize clarity in naming, even if it makes the code longer.

Attaching Extra Information to a Name

The passage says it's important to use clear names for variables in code. It suggests adding helpful words like "_ms" for milliseconds or "_secs" for seconds to make it easier to understand what the variable represents. It also recommends adding words like "untrusted" or "unsafe" to warn about potential risks, especially for security. This approach, called "English Notation," is different from "Hungarian Notation," which uses prefixes to show variable types. English Notation is more flexible and casual. Overall, using clear and descriptive variable names helps people understand and manage code better.

How Long Should a Name Be?

This passage says it's important to balance how long a name is with how clear it is. Short names are okay for small things, but for bigger things, longer names are needed to make sure people understand what it's about. Also, nowadays, text editors help with longer names by

suggesting words as you type. If you're shortening names with abbreviations or acronyms, they should make sense to everyone on the project. And try to keep names simple without extra words, so they're easy to understand but still short.

Use Name Formatting to Convey Meaning

This part talks about how important it is to pick good names when writing code in languages like C++ and JavaScript. It says you should use things like underscores, dashes, and capital letters to make names easier to understand. For example, in C++, variables are named like "lower_separated", constants like "kConstantName", and class member variables end with an underscore. In JavaScript, you capitalize constructors and start normal functions with lowercase letters. Adding a $ before jQuery results helps too. In HTML/CSS, IDs use underscores and classes use dashes. It's important to stick to the same naming style throughout a project to keep things clear.