IdentifierNamingPostForWorldWideWebBlog

It's easy to get carried away creating long identifiers. Longer names often make things more readable. But **names that are *too* long can decrease readability**. There are many examples of variable names longer than 60 characters on GitHub and elsewhere. **In 58 characters, we managed this haiku for you to consider**:

> **Name variables**
> **Using these simple guidelines**
> **Beautiful source code**

Names should be two things: *clear* (know what it refers to) **and** *precise* (know what it does *not* refer to). Here are some guidelines to help:

• Omit words that are obvious given a variable's type declaration.

```
// Bad, the type tells us what these variables are:
String nameString; List<DateTime> holidayDateList;
// Better:
String name; List<DateTime> holidays;
```

• Omit irrelevant details.

```
// Overly specific names are hard to read:
Monster finalBattleMostDangerousBossMonster; Payments nonTypicalMonthlyPayments;
// Better, if there's no other monsters or payments that need disambiguation:
Monster boss; Payments payments;
```

• Omit words that are clear from the surrounding context.

```
// Bad, repeating the context:
class AnnualHolidaySale {int annualSaleRebate; boolean promoteHolidaySale() {...}}
// Better:
class AnnualHolidaySale {int rebate; boolean promote() {...}}
```

• Omit words that could apply to any identifier.
You know the usual suspects: *data, state, amount, number, value, manager, engine, object, entity, instance, helper, util, broker, metadata, process, handle, context*. Cut them out.

There are some exceptions to these rules; use your judgment. Names that are too long are still better than names that are too short. However, **following these guidelines, your code will remain unambiguous and be much easier to read**. Readers, including "future you," will appreciate how clear your code is!

More information, discussion, and archives:
testing.googleblog.com