Principles of API design

Jayesh Kawli

API is an integral part of any programming language. It takes long time and efforts to flawlessly design and code an API. It is especially important as any API thus designed will be used by many users across the world and even a smallest bug will cause big trouble. Designing an API is very much similar to designing a language both of which can be compared to the task of developing a website
Since API is an integral part of the language, it is pretty much similar to designing and coding language libraries.

Here are some of the guidelines to be noted about design decisions

- Should be difficult to misuse
- Reusable
- Appropriate to use for users who are actually going to work on it

Following are the steps in successful API development lifecycle

- **API design**
- **Class design**
- **Method design**
- **Exception design**
- **API improvement (Optional)**

Keep it short and simple
In order to make design compact, keep its specification short which will allow making future additions easier. Depending upon the task developer has to decide what kind of errors can be expected from API execution and decide what kind of exceptions to be added in design. Initial design decisions are pivotal as earlier mistakes may lead to bigger problems in the final development stage. In fact, APIs users are bound to find bugs in it in the future.

Extensibility
In order to make sure API is extendable, most APIs provide SPI (Service Provider Interface) which acts as an interface to use services provided by specific owner of that SPI (e.g. (JCE) or Java Cryptography Extension)

Code size
While designing an API, size matters – developer should not keep too much redundant stuff and try to make it minimal. If skeptical about adding specific functionality, it is better to avoid it as it is obvious that in the later stages of development it is easier to add things rather than removing them out of the API.

Abstraction
APIs should avoid leaking too much information and in most cases make use of default parameters. E.g. Hash function – it should just expose the functionality and not the inner details such as its working, kind of algorithms used.

Documentation
Documentation plays another important part. It provides every detail of API. Since it will be used for very long time it is useful to make its specifications clear so that user will not have any ambiguity understanding what exactly the specific API does.

Immutability
While designing APIs make classes immutable. Though they are simpler, thread safe and easier to use, it creates memory issues as for each change, new object is created.

Concise and clear
In order to make code compact, avoid boilerplate code which tends to be repetitive. Repetition is undesirable in the sense that it is prone to mistakes and they propagate rapidly across different places. Also, change in one section can be time consuming to propagate the same change to other places.

Summary
To summarize, API design is very meticulous task. Programmer has to pay attention to many details before changes go to the production stage. Besides those mentioned above, there are other minute factors that decide final design decision. These considerations are pivotal as APIs are generally used worldwide in Universities, corporations, real world projects and hence paying attention to good API design principles is an indication of software development maturity.

Reference: Based on talk "How to Design a Good API and why it Matters" taken from http://www.youtube.com/watch?v=aAb7hSCtvGw