PWL-BSB # 12:
"**The Little Manual of API Design**"
Jasmin Blanchette

Papers We Love Brasília

Presenter: Alessandro Leite

November 14, 2018

What is an API?

It's all about abstraction

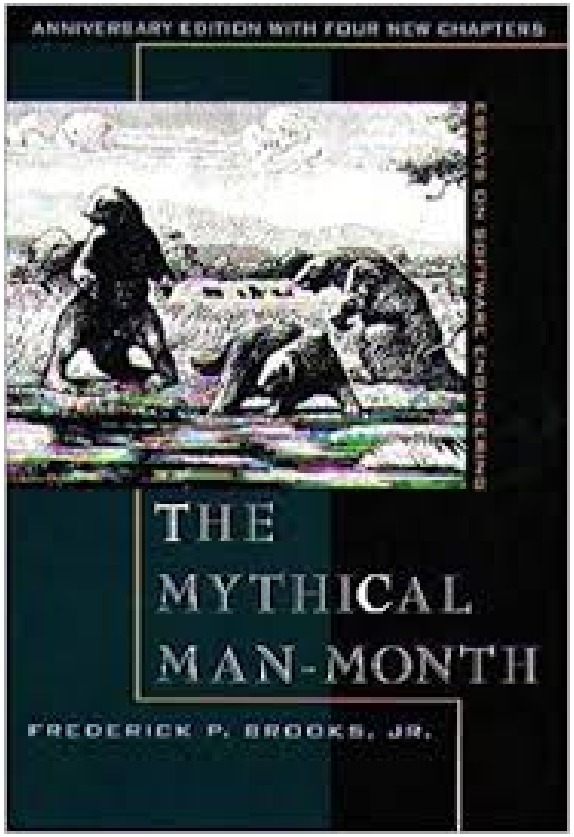
"Software is built on abstractions. Pick the right ones, and programming will flow naturally from design; modules will have small and simple interfaces; and new functionality will more likely fit in without extensive reorganization. Pick the wrong ones, and programming will be a series of nasty surprises.".

— Daniel Jackson, Software Abstractions, MIT Press, 2012.

A good API has 5 major characteristics

- A good API is:
  - Easy to learn and memorize
  - Leads to readable code
  - Hard to misuse
  - Easy to extend
  - Complete

An inconsistent design will lead to an API that will be hard to learn, to memorize, to extend



*"I contend that conceptual **integrity** is the **most important** consideration in **system design**. It is better to have a system omit certain anomalous features and improvements, but to reflect one set of design ideas, than to have one that contains many good but independent and uncoordinated ideas"*

— Frederick Brooke

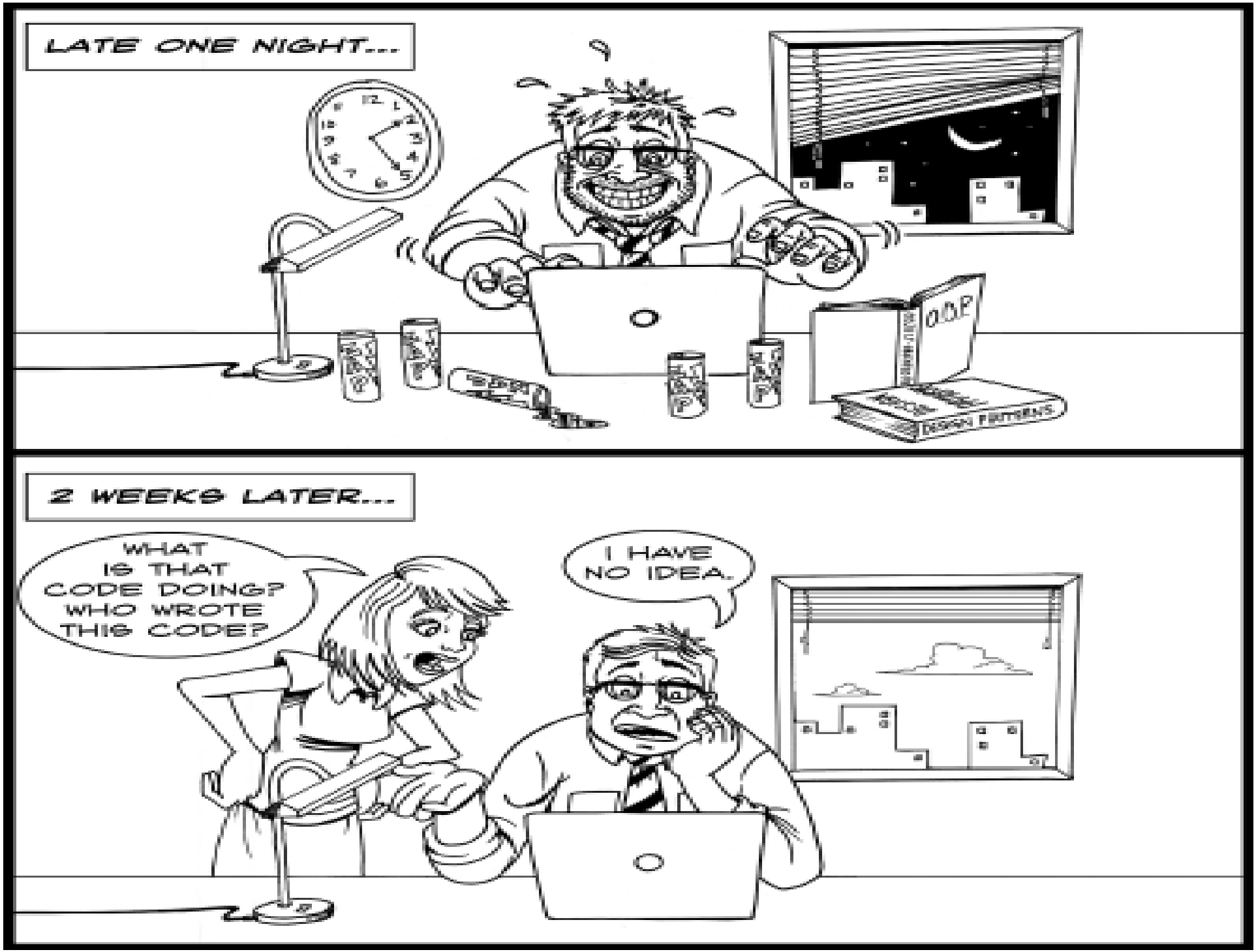
Easy to learn and memorize

- An API is easy to learn when:
  - It follows an consistent naming convention
  - It uses the same name for the same concepts, and different names for different concepts
  - It's enable us to reuse what we already learned when using one part of the API on another one
- An API is not only the names of the classes and methods that compose it, but also their intended semantics.

```
addItem(int, item)
addItem(item)
```

“Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”

– Martin Fowler


LATE ONE NIGHT...
OOP
DESIGN PATTERNS
2 WEEKS LATER...
WHAT IS THAT CODE DOING? WHO WROTE THIS CODE?
I HAVE NO IDEA.

Leads to readable code

- A code is usually written once, but read several times by different developers during its whole lifetime
- Readable code is easy to document and to maintain
- Readable code also makes bugs visible
- For instance, in this code, it's difficult to see that the initial value 6 is outside the defined volume range ([8,128])

```
slider = new QSlider(8, 128, 1, 6, Qt::Vertical,
    0,"volume")
```

- But not in this version

```
slider = new QSlider(Qt::Vertical);
slider->setRange(8, 128);
slider->setObjectName("volume");
slider->setValue(6);
```

Leads to readable code (2)

- Readable code does not mean concise code
- It's always a question of finding the correct level of abstraction
  - Don't hide important things nor force the developers to specify irrelevant information

- Qt Jambi code

```
QGridLayout layout = new QGridLayout;
layout.addWidget(slider, 0, 0);
layout.addWidget(spinBox, 0, 1);
layout.addWidget(resetButton, 2, 1);
layout.setRowStretch(1, 1);
setLayout(layout);
```

- Swing code

```
GridBagLayout layout = new GridBagLayout();
GridBagConstraints constraint = new
    GridBagConstraints();
constraint.fill = GridBagConstraints.
    HORIZONTAL;
constraint.insets = new Insets(10, 10, 10,
    0);
constraint.weightx = 1;
layout.setConstraints(slider, constraint);
constraint.gridwidth = GridBagConstraints.
    REMAINDER;
constraint.insets = new Insets(10, 5, 10,
    10);
constraint.weightx = 0;
layout.setConstraints(spinner, constraint);
constraint.anchor = GridBagConstraints.
    SOUTHEAST;
constraint.fill = GridBagConstraints.
    REMAINDER;
constraint.insets = new Insets(10, 10, 10,
    10);
constraint.weighty = 1;
layout.setConstraints(resetButton,
    constraint);
JPanel panel = new JPanel(layout);
panel.add(slider);
panel.add(spinner);
panel.add(resetButton);
```

Hard to misuse

- Well-designed APIs encourage good program practices
- It makes hard to write incorrect code
- It does not force the developers to be aware of side effects
- HTML design may lead the developers to write "incorrect code"

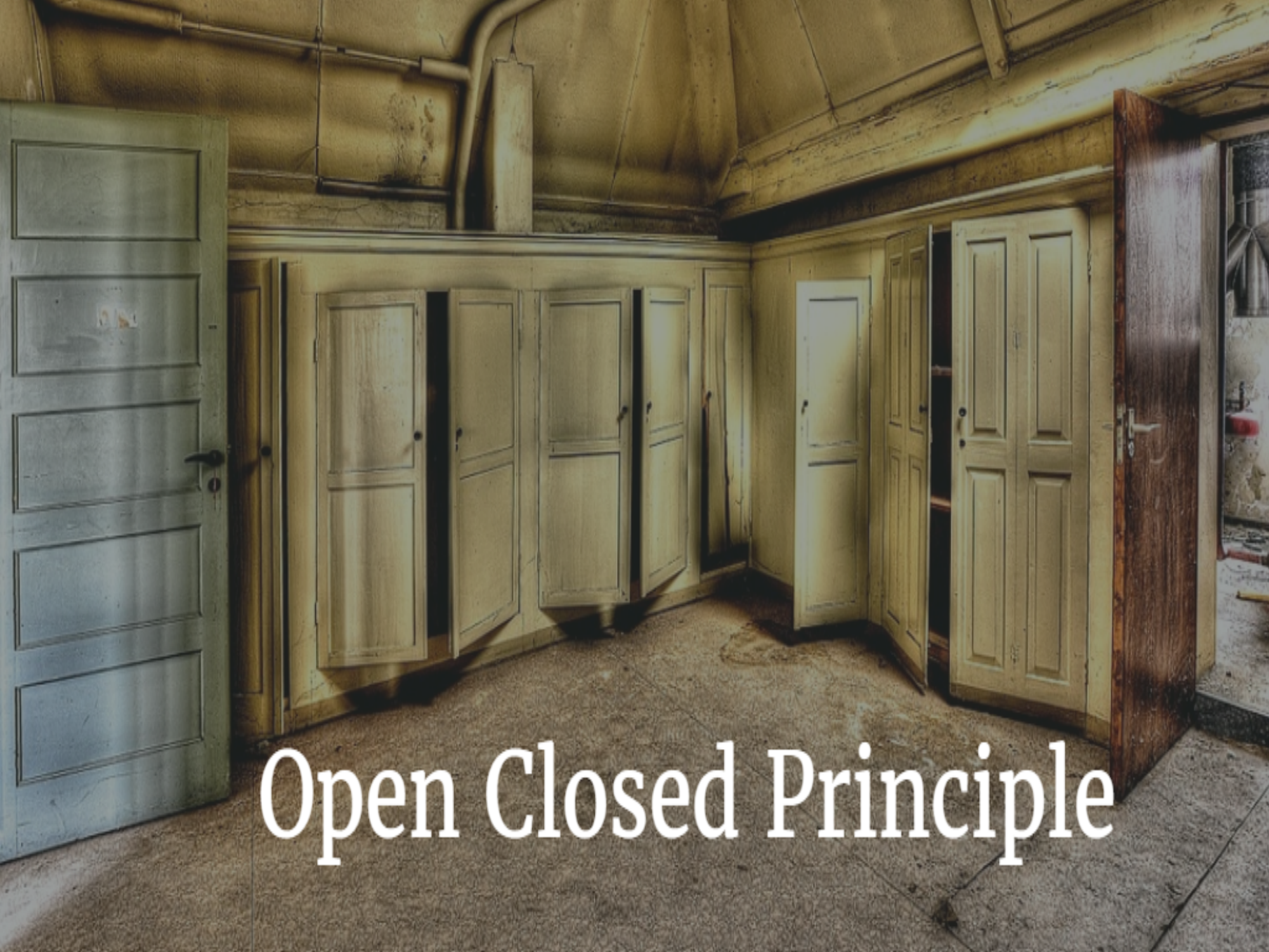
```
the <b>goto <u>label</b></u> statement
```

- It eliminates redundancy

```
obj.addItem(yksi);
obj.addItem(kaksi);
obj.addItem(kolme);
```

- Redundant code may encourage wrong code

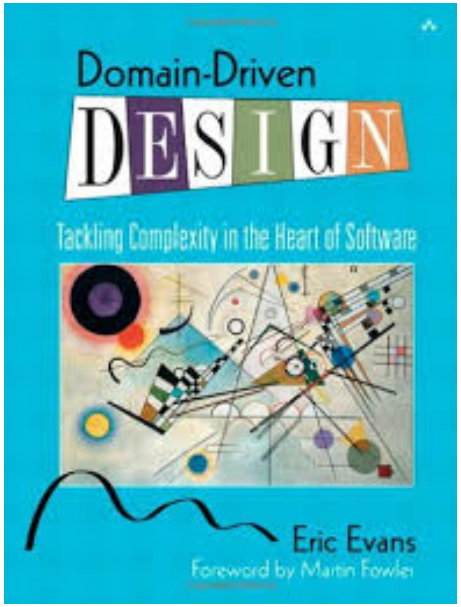
```
obj.addItem(0, yksi);
obj.addItem(1, kaksi);
obj.addItem(3, kolme);
```


Open Closed Principle

Easy to extend

- "Classes, modules, functions, should be open for extension, but closed for modification"

Complete



- A good API has well-defined boundaries
- Each API model lives in its context

Naming

- Choose self-explanatory names and signatures
- Choose unambiguous names for related things
- Beware of false consistency
- Avoid abbreviations
- Prefer specific names to general names
- Don't be a slave of an underlying API's naming practices

Semantics

- Choose good defaults
- Avoid making your APIs overly clever
- Pay attention to edge cases


That's all Folks!