

# JavaScript API Design Principles

@ariyahidayat

Feb 25, 2014  
Atlanta, GA

DEVNEXUS

**/usr/bin/whoami**

**SH=PE**

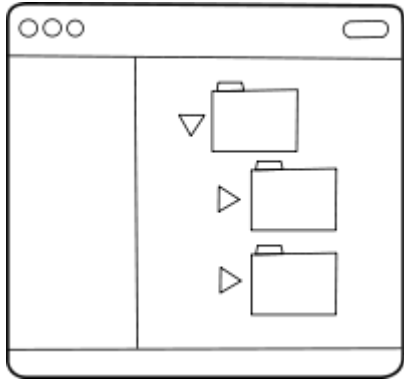
**shapesecurity.com**



PhantomJS



Esprima



```
treeItem.setState(true, false);
```



## **“How to Design a Good API and Why It Matters”**

Joshua Bloch

## **“Qt API Design Principles”**

<http://qt-project.org/wiki/API-Design-Principles>

Matthias Ettrich, Jasmin Blanchette



**Ariya Hidayat**

@AriyaHidayat

This is your annual reminder that Java is to JavaScript as horse is to horseradish.

↩ Reply 🗑 Delete ★ Favorite ⋮ More

RETWEETS

73

FAVORITES

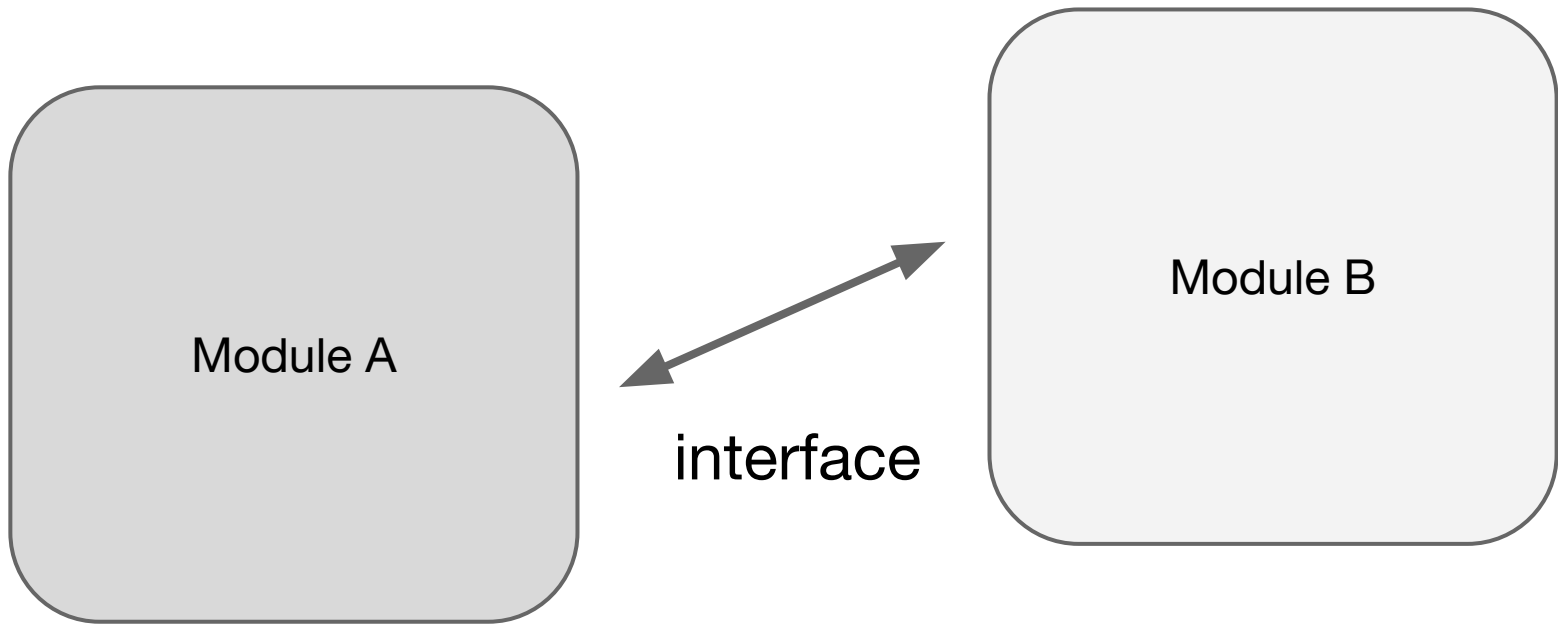
39



8:39 PM - 30 Jan 2014

<https://twitter.com/AriyaHidayat/status/429111596627918848>

# Modular Application Architecture



# WHY?

**Nobody** reads API documentation

Code is written **once**, read **many times**



Readable, consistent,  
crystal clear

Thank you, Captain Obvious!



Using **static polymorphism**  
to ensure consistency

Preventing **dangerous convenience**  
such as Boolean trap

Avoiding unreadable code  
due to **confusing semantics**

# **Static Polymorphism**

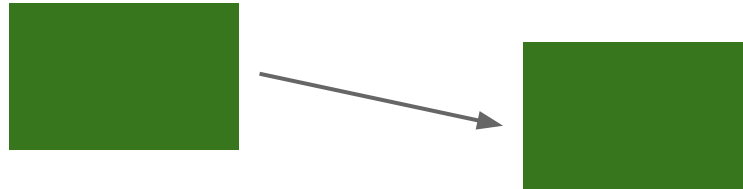
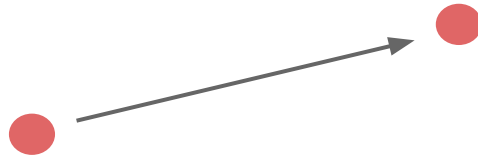
- ☒ Rare
- ☐ Medium
- ☐ Well done
- ☒ Fries

Order

```
X1.value = 'Rare';  
X2.value = 'Medium';  
X3.value = 'Well done';  
Y.option = 'Fries';  
Z.caption = 'Order';
```

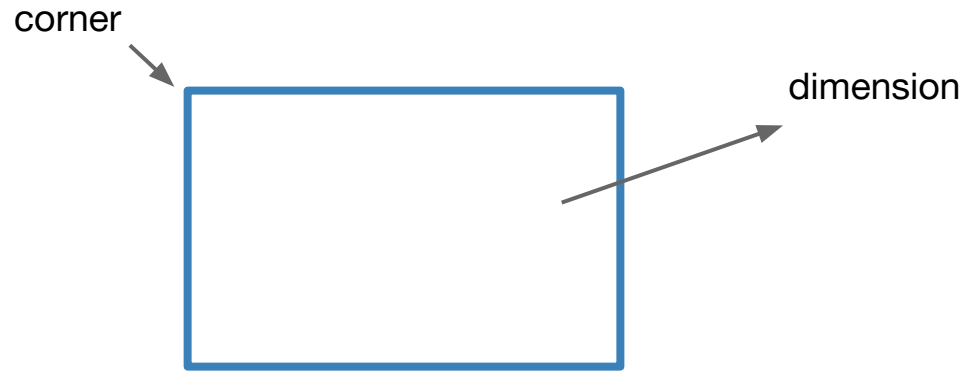
```
X1.value = 'Rare';  
X2.value = 'Medium';  
X3.value = 'Well done';  
Y.value = 'Fries';  
Z.value = 'Order';
```

Progress Bar vs Slider



```
point.translate(14, -3)  
rect.translateBy(26, 4)
```

```
point.translate(14, -3)  
rect.translate(26, 4)
```



```
corner = new Point(10, 10);  
dim = new Size(70, 50);  
R = new Rect(corner, dim);
```

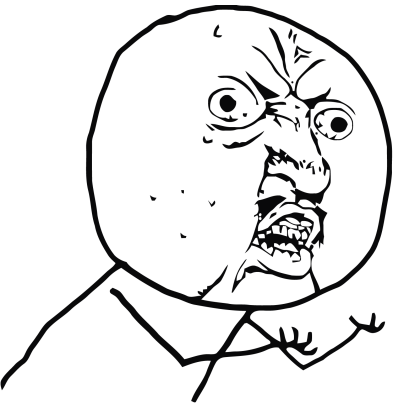
```
Q = new Rect(10, 10, 80, 60);
```

x1, y1, x2, y2

```
corner = new Point(10, 10);  
dim = new Size(70, 50);  
R = new Rect(corner, dim);
```

```
Q = new Rect(10, 10, 70, 50);
```

x, y, w, h



```
var x = NaN;
```

```
isNaN(x)      true
```

```
x === NaN     false
```



**Ariya Hidayat**  
@AriyaHidayat

This is your annual reminder that NaN stands for "Not a NaN".

↩ Reply 🗑 Delete ★ Favorite ⋮ More

RETWEETS  
51

FAVORITES  
12



8:22 AM - 23 Oct 2013

<https://twitter.com/AriyaHidayat/status/393034774245171200>

# **Convenience & Traps**

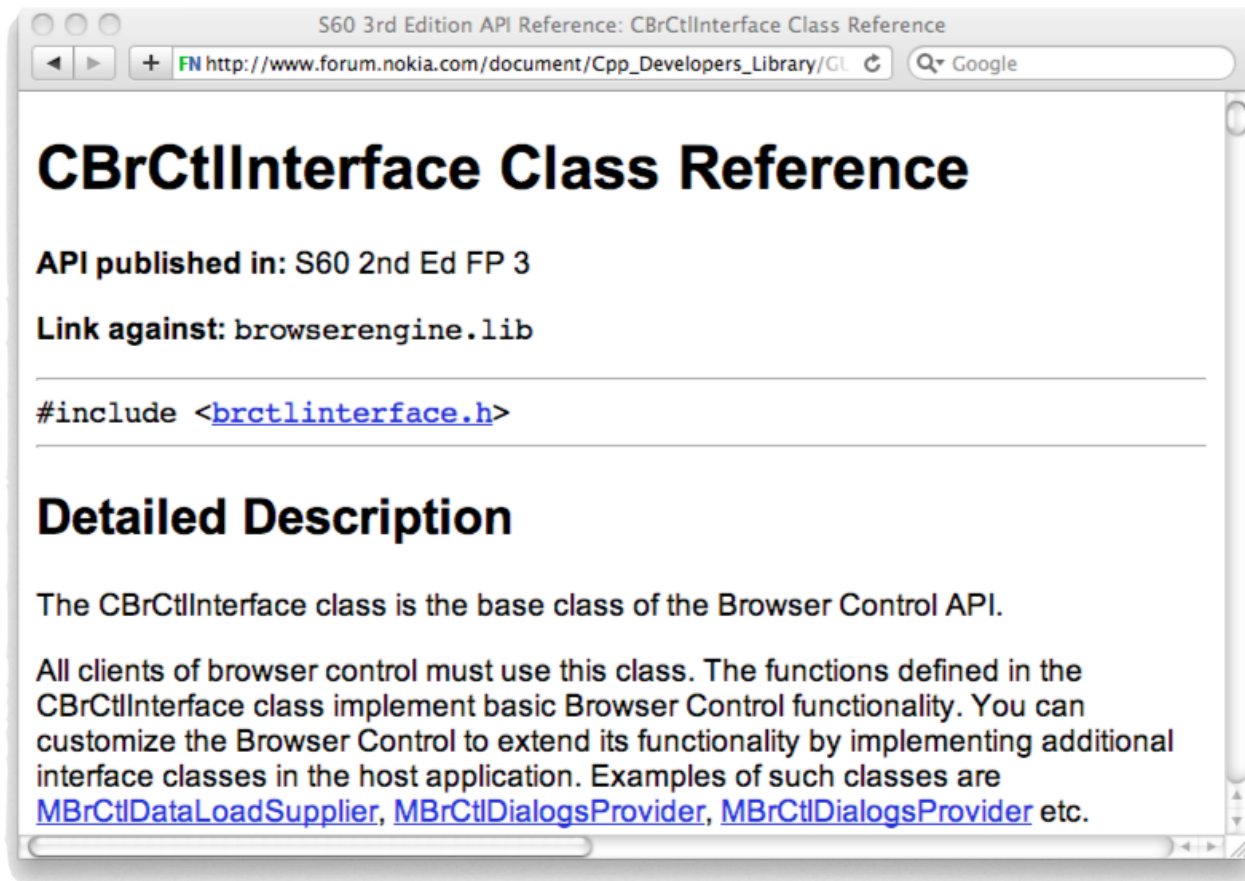
# Romeo and Juliet: Act II. Scene II

*What's in a name?*  
that which we call a rose  
by any other name  
would smell as sweet...





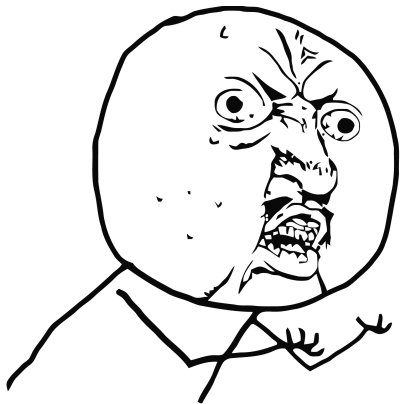
# Can We Have Some Vowels, Please?



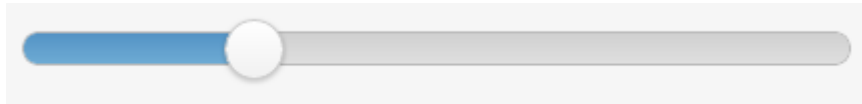
**BrCtl** →  
WebView

# DOM Event

```
event.initKeyEvent("keypress", true, true,  
null, null, false, false, false, false, 9, 0);
```



# Boolean Trap



```
// Horizontal
```

```
s1 = new Slider(true);
```

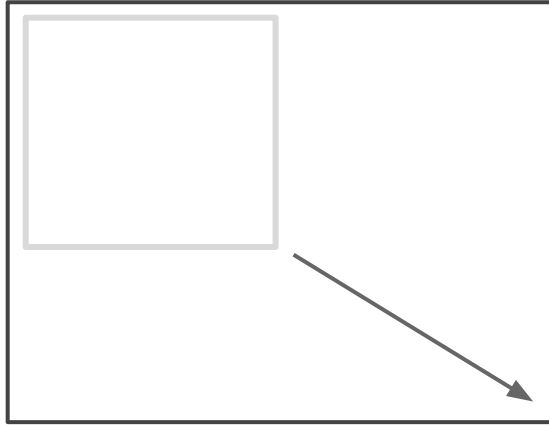
```
// Vertical
```

```
s2 = new Slider(false);
```

```
s1 = new Slider({  
    orientation: 'horizontal'  
});
```

```
s2 = new Slider({  
    orientation: 'vertical'  
});
```

# Boolean Modifier Trap



// Animate the resize

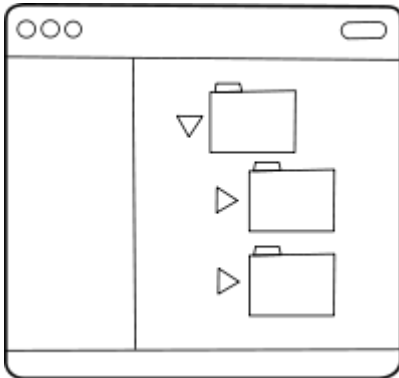
```
w.resize(250, 150, true);
```

// Do not animate the resize

```
w.resize(250, 150, false);
```

```
w.resize(250, 150, {  
    animate: true  
}));
```

# Ping-pong, Anyone?



```
// Expand + animate
```

```
treeItem.setState(true, true);
```

```
// Expand + don't animate
```

```
treeItem.setState(true, false);
```

```
// Collapse + animate
```

```
treeItem.setState(false, true);
```

# Ping-pong, Anyone?



```
// Expand + animate
```

```
treeItem.setState(true, true);
```

```
// Expand + don't animate
```

```
treeItem.setState(true, false);
```

```
// Collapse + animate
```

```
treeItem.setState(false, true);
```



**John Carmack** ✓  
@ID\_AA\_Carmack



+ Follow

I nearly fell into the "boolean trap"  
[ariya.ofilabs.com/2011/08/hall-o...](http://ariya.ofilabs.com/2011/08/hall-o...)  
three times while writing one new  
code file.

↩ Reply ↻ Retweet ★ Favorite ... More

RETWEETS  
**203**

FAVORITES  
**128**



8:19 PM - 13 Aug 2013

[https://twitter.com/ID\\_AA\\_Carmack/status/367485612627984385](https://twitter.com/ID_AA_Carmack/status/367485612627984385)

**Ambiguity Begets Insanity**



# Double Negatives

```
X.disabled = false;
```

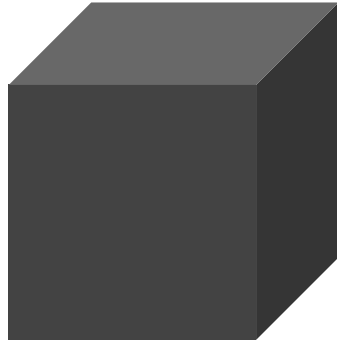
```
X.enabled = true;
```

```
Y.setHidden(true);
```

```
Y.setVisible(false);
```

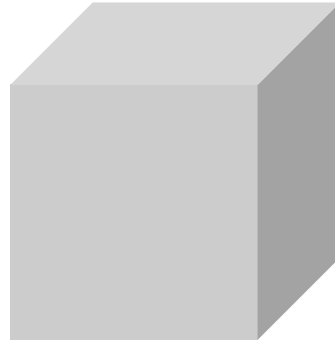
```
filter.caseInsensitive = true;
```

```
filter.caseSensitive = false;
```



20% translucent

```
cube.translucency = 0.2;
```



80% opaque

```
cube.opacity = 0.8;
```

# Nonverbal Actions

```
status.message("Ready");
```

```
status.showMessage("Ready");
```

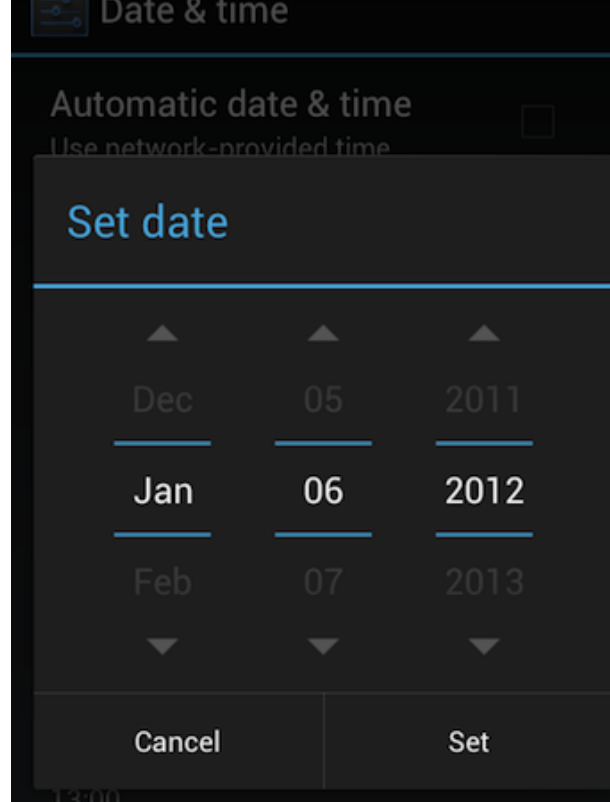
```
page.forward();
```

```
page.goForward();
```

```
page.backward();
```

```
page.goBackward();
```





```
picker.yearFrom = 1980;           picker.minimumYear = 1980;
```

```
picker.yearTo = 2020;             picker.maximumYear = 2020;
```



```
this.callMe = "Adam";
```

```
this.name = "Adam";
```

```
flight.from = SFO;
```

```
flight.departure = SFO;
```

```
flight.to = JFK;
```

```
flight.destination = JFK;
```

# String Extraction

`'devnexus2014'.substr(3, 9)`      `"nexus2014"`

`'devnexus2014'.substr(9, 3)`      `"014"`

`'devnexus2014'.substring(3, 9)`      `"nexus2"`

`'devnexus2014'.substring(9, 3)`      `"nexus2"`

`'devnexus2014'.slice(3, 9)`      `"nexus2"`

`'devnexus2014'.slice(9, 3)`      `""`

# Array Extraction

```
var a = [14, 3, 77]
```

a [14, 3, 77]

```
var b = a.slice(1, 2)
```

b [3]

## Immutable vs Mutable

```
var a = [14, 3, 77]
```

a [14]

```
var b = a.splice(1, 2)
```

b [3, 77]

# Verb = Action

```
var p = new Point(14, 3);  
p.translate(4, 4);
```

```
var s = ' devnexus ';  
s.trim();
```

Does this change my string or  
return a fresh new string?



# Explicit (Im)mutability

```
var p = new Point(14, 3);  
p.translate(4, 4);  
p.translated(4, 4);
```



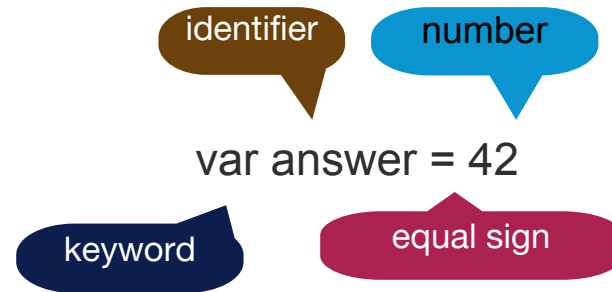
p does not change  
Returns a translated version of p

```
var s = ' devnexus ';  
s.trim();  
s.trimmed();
```

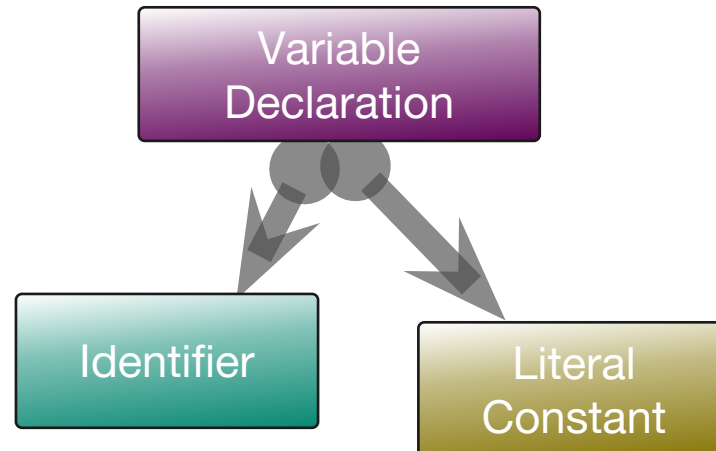
# **API Detective**

# Syntax Parser

**Tokenization** →  
Tokens



**Parsing** → Syntax Tree



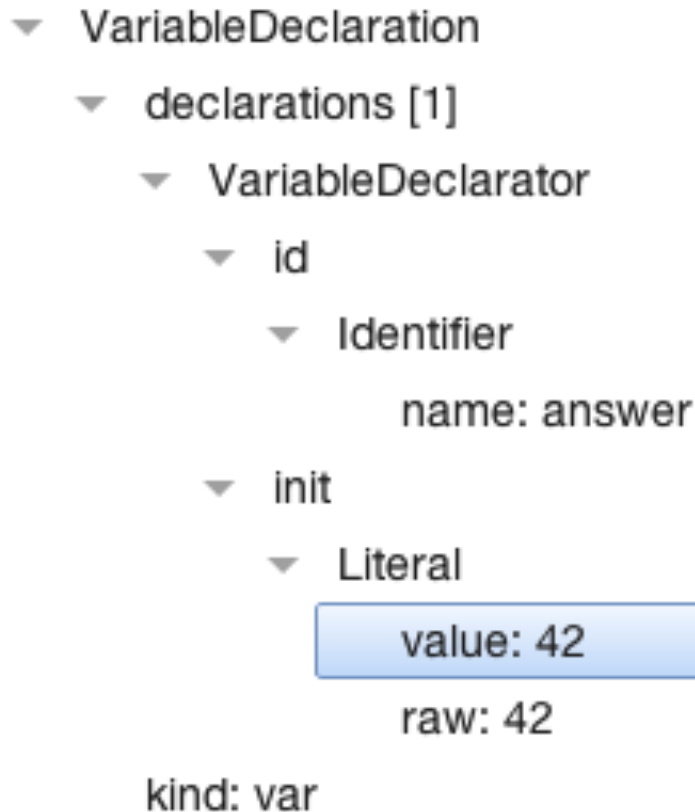
# Syntax Tree

```
var answer = 42;
```

**Terms** → ECMAScript 5.1 Specification

```
{
  type: "Program",
  body: [
    {
      type: "VariableDeclaration",
      declarations: [
        {
          type: "VariableDeclarator",
          id: {
            type: "Identifier",
            name: "answer"
          },
          init: {
            type: "Literal",
            value: 42,
            raw: "42"
          }
        }
      ]
    }
  ],
  kind: "var"
}
```

# Syntax Visualization



# Module Structure

```
MyApp.create('MyApp.Person', {  
  name: 'Joe Sixpack',  
  age: 42,  
  
  constructor: function(name) {},  
  walk: function(steps) {},  
  run: function(steps) {}  
});
```



Metadata

```
{  
  objectName: 'MyApp.Person',  
  functions: ['walk', 'run'],  
  properties: ['name', 'age']  
}
```

Meta-object

# Collecting Member Expression

foo.bar

- ▼ MemberExpression
  - computed: false
  - ▼ object
    - ▼ Identifier
      - name: foo
  - ▼ property
    - ▼ Identifier
      - name: bar

```
syntax = esprima.parse(code);
traverse(syntax, function (node) {
  if (node.type === 'MemberExpression') {
    if (node.property) {
      console.log(node.property.name);
    }
  }
});
```

# Detecting Boolean Traps

`reload(x, y, false)`

- ▼ CallExpression
  - ▼ callee
    - ▼ Identifier
      - name: reload
  - ▼ arguments [3]
    - ▼ Identifier
      - name: x
    - ▼ Identifier
      - name: y
    - ▼ Literal
      - value: false
      - raw: false

```
function checkLastArgument(node) {  
  var args = node['arguments'];  
  
  if (args.length < 2) {  
    return;  
  }  
  
  if (typeof args[args.length - 1].value === 'boolean') {  
    report(node, 'Dangerous Boolean literal');  
  }  
}
```



# Blacklisting Function Names

```
setDisabled(false)
```

- ▼ CallExpression
  - ▼ callee
    - ▼ Identifier
      - name: setDisabled
  - ▼ arguments [1]
    - ▼ Literal
      - value: false
      - raw: false

This block is left **intentionally** blank.

**Exercise** for the brave reader!

# What About API Usage?

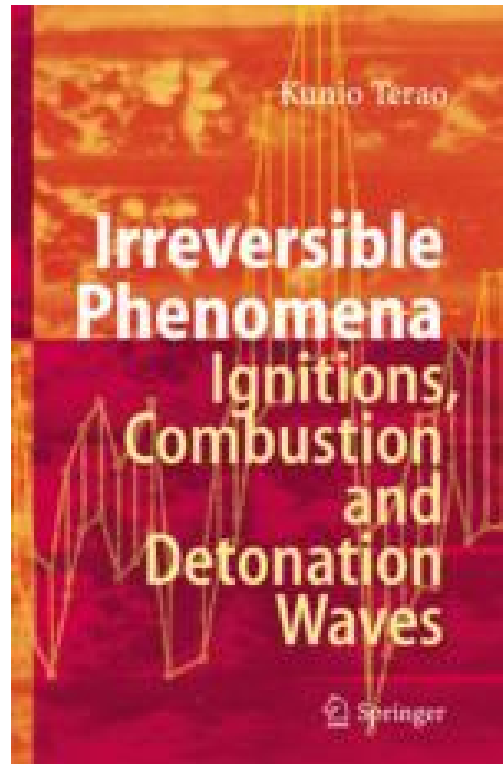
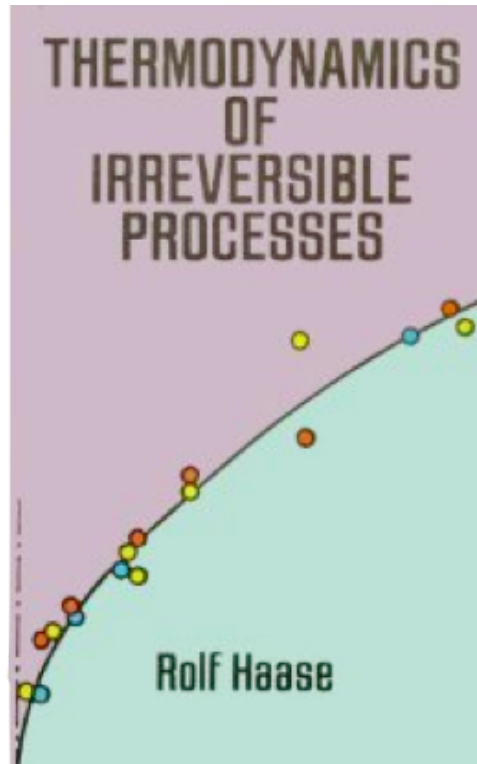
“You call this **immutable** function but you never assign its return value.”

“You check the return value of this function 2000 times. **Why don't** you do it this time (line = ....)?”



# **Best Practices**

# #1: Private by Default



Public = Irreversible

# SHStripMnemonic

“Why is the function  
SHStripMnemonic misspelled?”

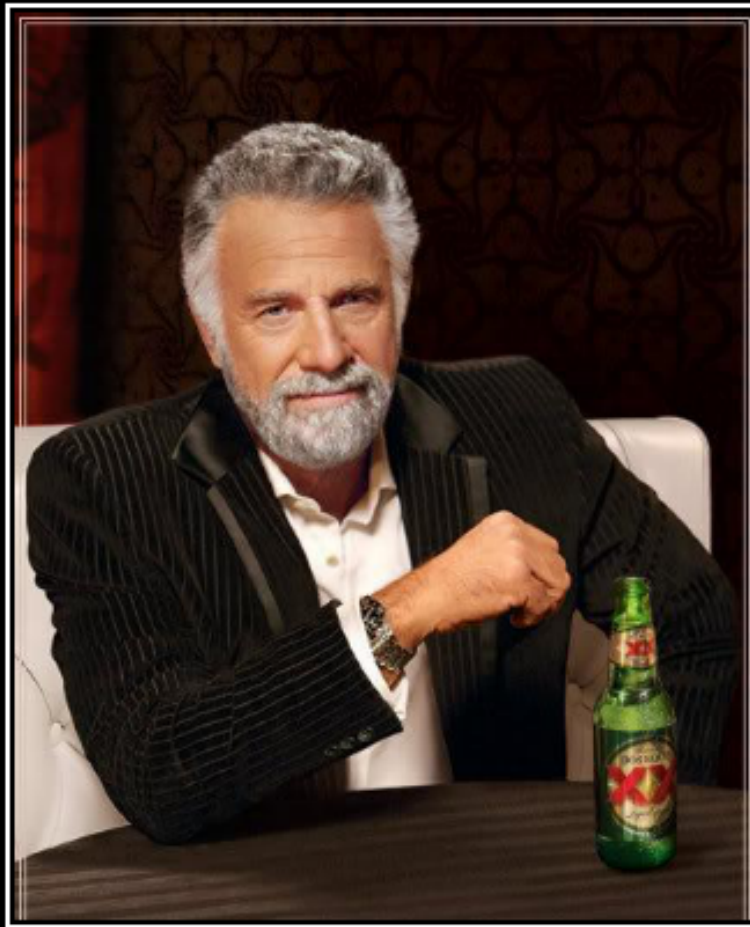
Raymond Chen, Microsoft

<http://blogs.msdn.com/b/oldnewthing/archive/2008/05/19/8518565.aspx>



## **#2: Public by Justification**

“Write 3 examples which  
use the API”

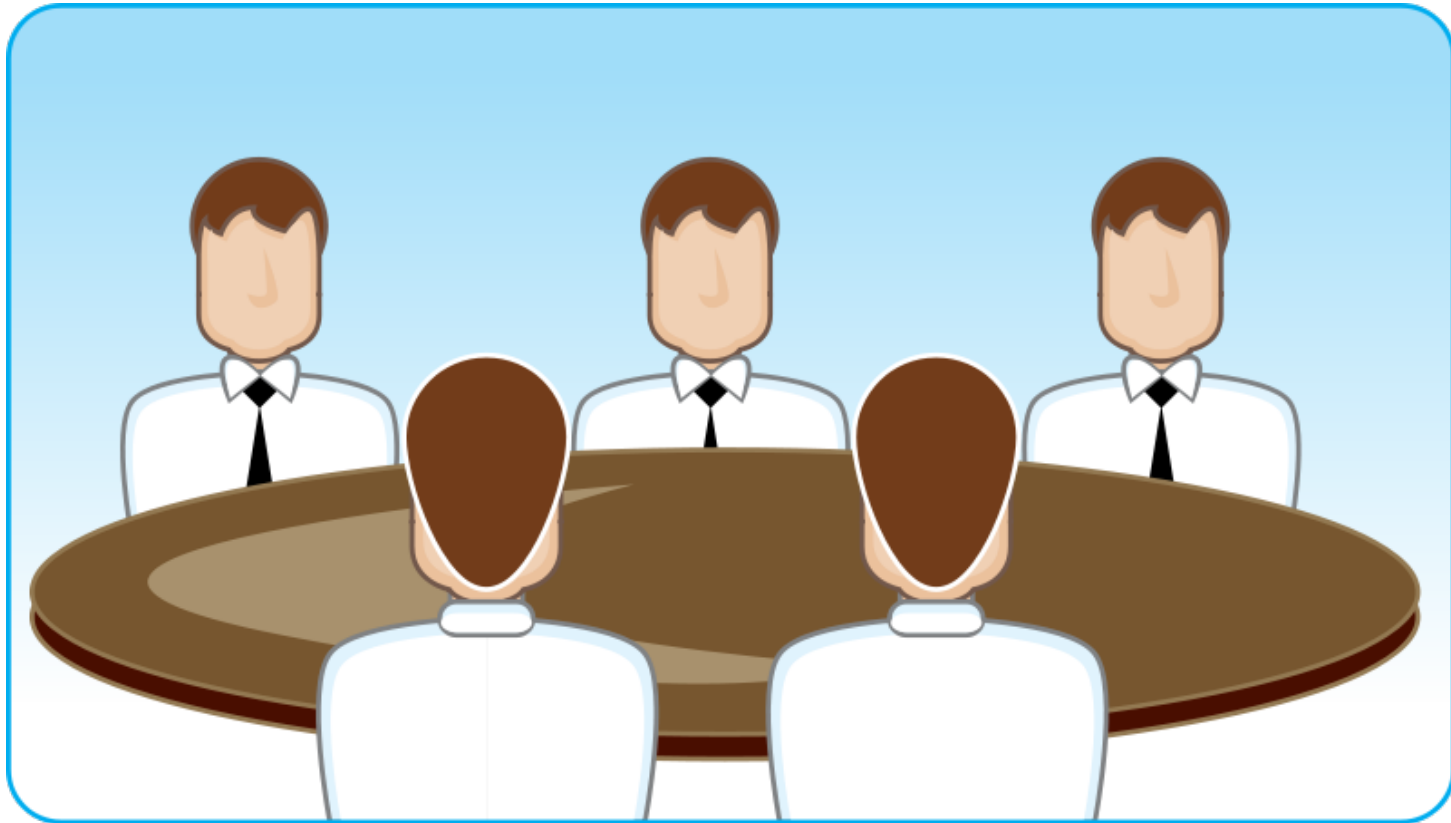


# I DON'T ALWAYS CODE

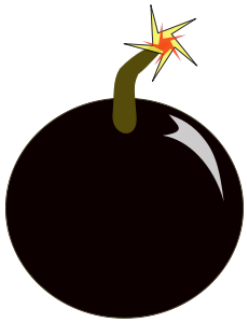
but when I do, it comes with 3 examples



## #3: Mandatory API Review



“You’d fail your first  
**two** attempts anyway.”



-- yours truly



“YOU SHALL NOT PASS!”

— *Darth Vader*

## #4: API Tools



**Tools separate us**

T D D

B D D



P D D

# Final Words

Practice the rituals:

- Apply static polymorphism
- Judge every convenient shortcut
- Read aloud (and often)

Implement + tweak API tools



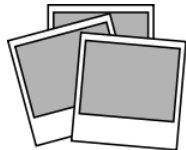
# Thank You



@ariyahidayat



[ariya.ofilabs.com/highlights](http://ariya.ofilabs.com/highlights)



[speakerdeck.com/ariya](http://speakerdeck.com/ariya)

