Readable Code

Companion slides of
The art of readable code

by Dustin Boswell, Trevor Foucher
O'Reilly, 2012

Code should be easy to understand.

Code should be written to minimize the time it would take for someone else to understand it.

Packing Information into Names

Choosing specific words

Avoiding generic names

Using concrete names instead of abstract names

Attaching extra information to a name

Deciding how long a name should be

Using name formatting to pack extra information

Choose specific words

```
public String getPage(String url) { ... }

class BinaryTree {
  public int getSize() { ... }
}
```

getHeight
getNoNodes ?
getMemorySize

Avoid generic names

```
public double euclideanNorm(double[] v) {
  double retValue = 0;
  for (int i=0; i < v.length; i++)
    retValue += v[i] * v[i];
  return Math.sqrt(retValue);
}
```

? retValue does not pack much information

! sumSquares is much better!

Avoid generic names

```
if (right < left} {
  tmp = right;
  right = left;
  left = tmp;
}
```

```
String tmp = user.name();
tmp += " " + user.phone();
tmp += " " + user.email();

template.set("user_info", tmp);
```

! Use tmp when being short-lived and temporary is the most important fact.

Attaching extra information to a name

```
String id = "af84e";
String hexId = "af84e";
```

```
Date start, stop;...
long secElapsed = stop.getTime() - start.getTime();
System.out.println("Elapsed seconds " + secElapsed);
```

? getTime() returns time in milliseconds!

How long should a name be?

Shorter names are ok for shorter scope!

BEManager ? BackEndManager

Would a new team member understand what does the name mean?

convertToString ? toString()

Omit meaningless words!

Names that can't be misconstructed

? What other meanings could someone interpret from this name?

Filter ?

```
results = Database.all_objects.filter("year <= 2011")
```

? Objects whose year is <= 2011

? Objects whose year is not <= 2011

! select or exclude are better names

min and max ?

```
public static final int CART_TOO_BIG_LIMIT = 10;
...
if(shopping_cart.num_items() >= CART_TOO_BIG_LIMIT)
    return;
```

? CART_TOO_BIG_LIMIT - up to OR up to and including

! MAX_ITEMS_IN CART = 10;

Inclusive/Exclusive Ranges

```
int start = 2, stop = 4;
print_integers(start, stop);
```

? [2, 3] OR [2, 3, 4]

! int first = 2, last = 4;

```
print_events_in_range("OCT 16 12:00am","OCT 17 12:00am");
print_events_in_range("OCT 16 12:00am",
                      "OCT 16 11:59:59.0000pm");
```

! int begin = 2, end = 4; (for Inclusive/Exclusive Ranges)

Naming Booleans

```
private boolean readPassword = true;
```

? we need to read a password
OR
password has already been read

```
! private boolean needPassword = true;
  private boolean userIsAuthenticated = true;

? private boolean disableSSL = false;
! private boolean enableSSL = true;
```


get*()

```
public class StatisticsCollector {
    public void addSample(double x) {
        ...
    }
    public double getMean() {
        // iterate through all elements and return total / size
        ....
    }
}
```

! computeMean() is better

size()

? return a pre-calculated value

OR iterate through the elements

```
while(list.size() > ... ) {
} //Time performance is important !
```

Evaluate Multiple Name Candidates

experimentId: 100

description: "font size 14pt"

[more 15 parameters given as key/values pairs]

experimentId: 101

? theOtherExperimentIdIWantToReuse: 100

? better name: template, reuse, copy, inherit ?

[change any properties as needed]

Comments

! Comments kelps the reader know as much as the writer did.

Worthless comments

```
//the class definition for Account
public class Account {
   //constructor
   public void Account() { ... }
   //set the profit member to a new value
   public void setProfit(double profit) { ... }
   //return the profit from this account
   public double getProfit() { ... }
}
```

Don't comment on facts that can be derived quickly from the code itself.

Worthless comments

```
// find the node in the given subtree, with the given name, using the given depth
Node findNodeInSubtree(Node subtree, String name, int depth) { ... }

! // find a node with the given name or return null
  // if depth <= 0, only subtree is inspected
  // if depth == N, only subtree and N levels below are inspected
Node findNodeInSubtree(Node subtree, String name, int depth) { ... }
```

Don't comment bad names - Fix the names instead

// release the handle for this key.

// This doesn't modify the actual registry.

? public void deleteRegistry(RegistryKey key);

! public void releaseRegistryHandle(RegistryKey key);

! key.releaseRegistryHandle();

Director Commentary

A Director Commentary tracks where the filmmakers give their insights and tell stories to help you understand how the film was made.

// This heuristic might miss a few words

// This class is getting messy....

Comment on your constants

//as long as it is >= 2 * processor number, it is good enough

public static final int NUM_THREADS = 8;

//users thought 0.72 gave the best size/quality tradeoff

public static final float IMAGE_QUALITY = 0.72;

Put yourself in the reader's shoes.

! Imagine what your code looks like to an outsider.

! Making control flow easy to read.

The order of arguments in conditionals

? if (length >= 10) OR if (10 <= length)

? while (bytesReceived < bytesExpected)

OR

while (bytesExpected > bytesReceived)

! Left-hand side
the expression being interrogated, whose value is more in the flux

! Right-hand side
the expression being compared against, whose value is more constant

The order of if/else blocks

```
? if (a==b) {
    // Case One ...
} else {
    // Case Two ...
}
? if (a!=b) {
    // Case Two ...
} else {
    // Case One ...
}
```

Deal with positive case first. if (debug) instead of if (!debug)

Deal with the simpler case first.

Deal with the more interesting case first.

The order of if/else blocks

```
if (!url.hasQueryParameter("expand_all")) {
    response.render(items);   ...
} else {
    for (int i = 0; i < items.size(); i++) {
        items[i].expand();
    } ...
}
```

```
if (url.hasQueryParameter("expand_all")) {
    for (int i = 0; i < items.size(); i++) {
        items[i].expand();
    } ...
} else {
    response.render(items);   ...
}
```

The ?: conditional expression

```
String time = (hour >= 12) ? "pm" : "am";
```

=

```
String time;
if (hour >= 12)
    time = "pm";
else
    time = "am";
```

Use ?: only for the simplest cases.

Avoid do/while loops

```
// search through the list, starting at node, for the given name
// don't consider more than maxLength nodes
public boolean listHasNode(Node node, String name, int maxLength) {
    do {
        if (node.getName().equals(name))
            return true;
        node = node.next();
    } while ((node != null) && (--maxLength > 0));
    return false;
}
```


Avoid do/while loops

```
// search through the list, starting at node, for the given name
// don't consider more than maxLength nodes
public boolean listHasNode(Node node, String name, int maxLength) {
    while ((node != null) && (maxLength-- > 0)) {
        if (node.getName().equals(name))
            return true;
        node = node.next();
    }
    return false;
}
```


Minimize nesting

```
if (user_result == SUCCESS) {
  reply.writeErrors("");
} else {
  reply.writeErrors(user_result);
}
reply.done();
```

Minimize nesting

```
if (user_result == SUCCESS) {
  if (permission_result != SUCCESS) {
    reply.writeErrors("error writing permissions");
    reply.done();
    return;
  }
  reply.writeErrors("");
} else {
  reply.writeErrors(user_result);
}
reply.done();
```


Minimize nesting

```
if (user_result != SUCCESS) {
  reply.writeErrors(user_result);
  reply.done();
  return;
}
if (permission_result != SUCCESS) {
    reply.writeErrors("error writing permissions");
    reply.done();
    return;
  }
reply.writeErrors("");
reply.done();
```


Breaking down giant statements

```
var update_highlight = function (message_num) {
    if ($("#vote_value" + message_num).html() === "Up") {
        $("#thumbs_up" + message_num).addClass("highlighted");
        $("#thumbs_down" + message_num).removeClass("highlighted");
    } else if ($("#vote_value" + message_num).html() === "Down") {
        $("#thumbs_up" + message_num).removeClass("highlighted");
        $("#thumbs_down" + message_num).addClass("highlighted");
    } else {
        $("#thumbs_up" + message_num).removeClass("highighted");
        $("#thumbs_down" + message_num).removeClass("highlighted");
    }
};
```

(JavaScript sample)

Breaking down giant statements

```
var update_highlight = function (message_num) {
    var thumbs_up = $("#thumbs_up" + message_num);
    var thumbs_down = $("#thumbs_down" + message_num);
    var vote_value = $("#vote_value" + message_num).html();
    var hi = "highlighted";

    if (vote_value === "Up") {
        thumbs_up.addClass(hi);
        thumbs_down.removeClass(hi);
    } else if (vote_value === "Down") {
        thumbs_up.removeClass(hi);
        thumbs_down.addClass(hi);
    } else {
        thumbs_up.removeClass(hi);
        thumbs_down.removeClass(hi);
    }
};
```

(JavaScript sample)

Towards variables

! Make your variable visible by as few lines of code as possible.


```
class Value{
    private int value;
    private String str;

    public Value() {
        value = 0;
    }

    public Value(int v) {
        value = v;
    }

    public void setValue(int v) {
        value = v;
    }

    public String toString() {
        str = "My value is " + value;
        return str;
    }
}
```

?