

Code Smells And How To Fix Them

A JavaScript Example by Teresa Holfeld

Code Smells



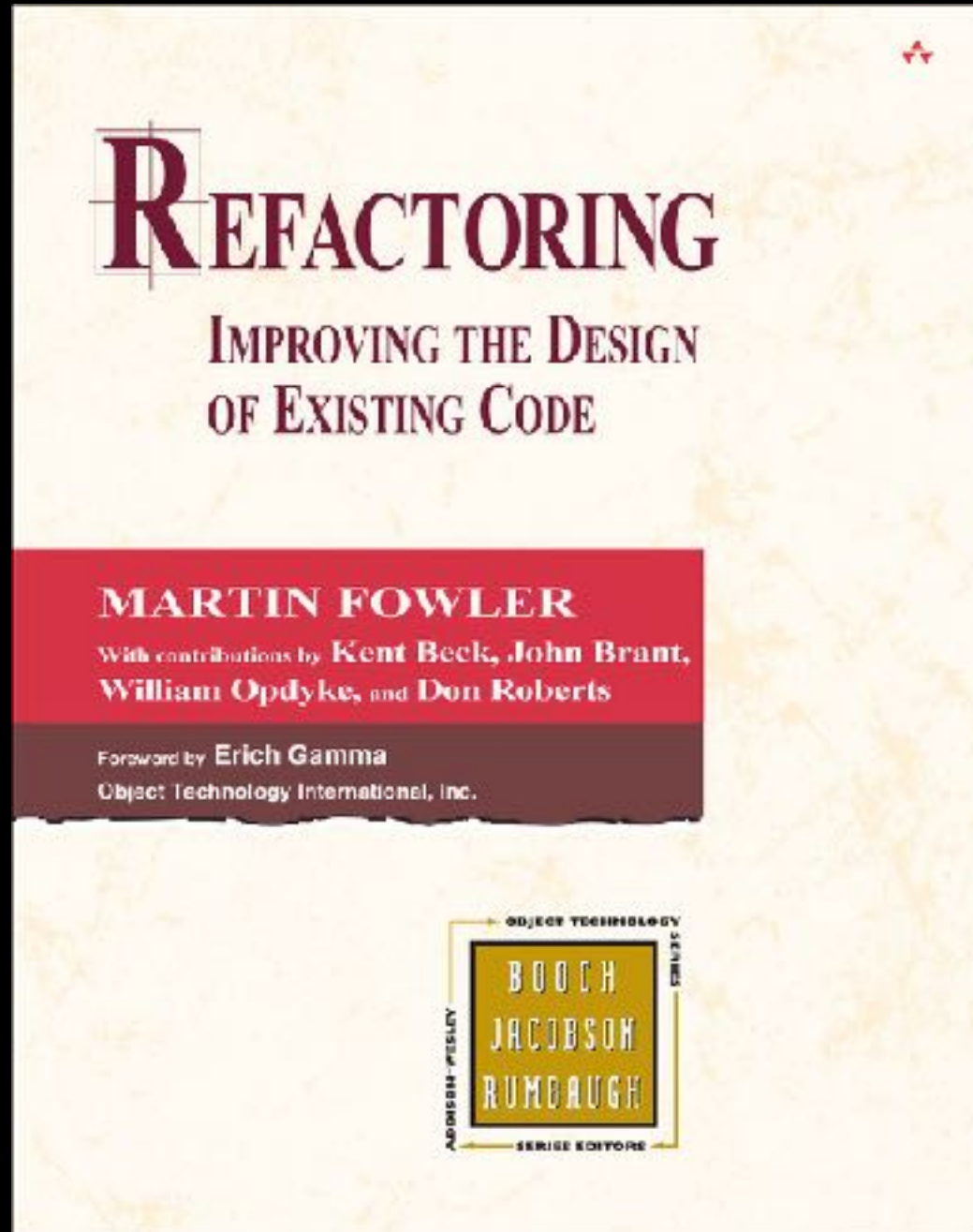
Code Smells

Code changes all the time.

The harder it gets to change it, the more expensive it is to maintain it.

How can we make it easy to change?

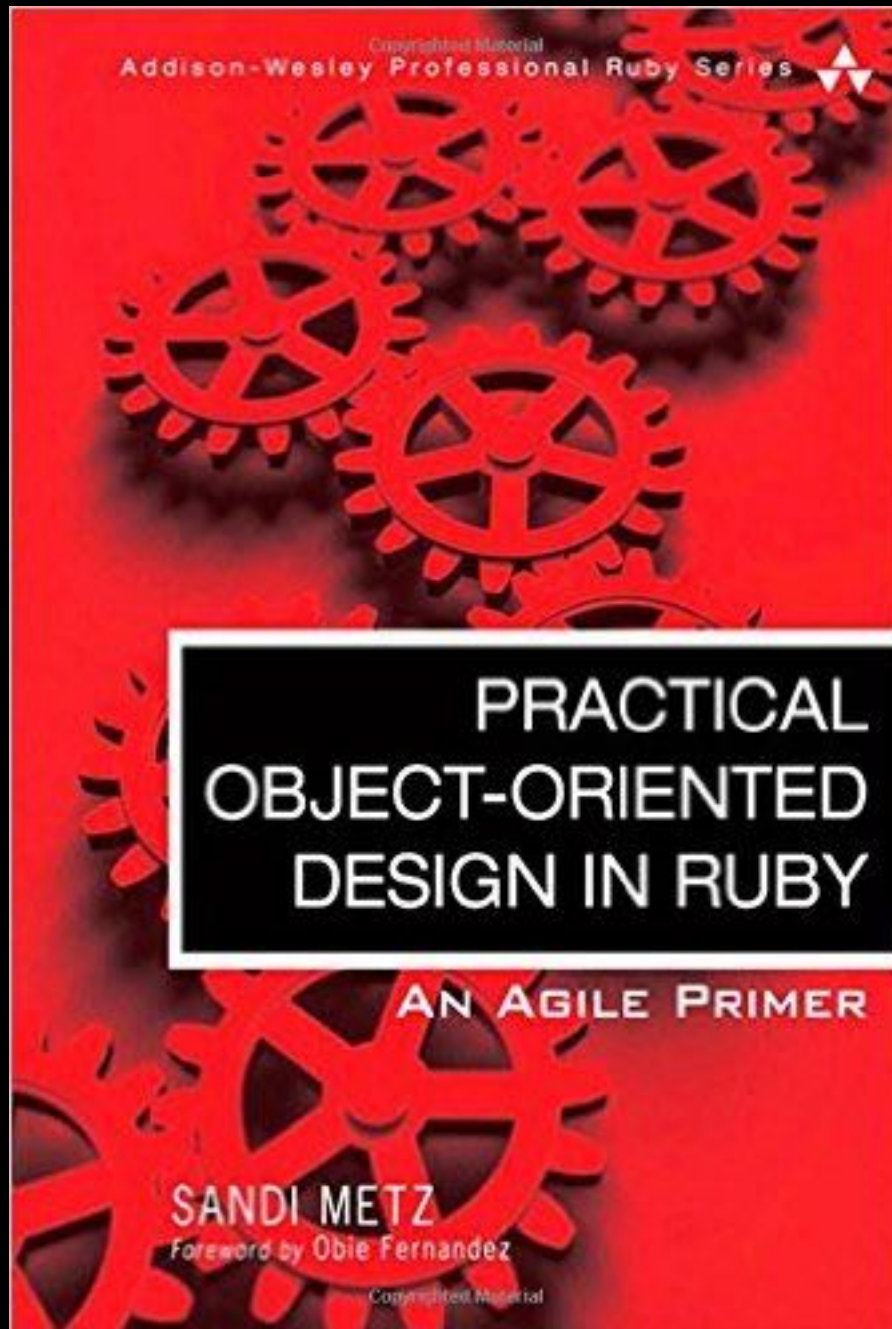
Code Smells



Martin Fowler, 1999:
Refactoring. Improving the
Design of Existing Code.



Code Smells



Sandi Metz, 2012:
Practical Object-Oriented
Design in Ruby.



Code Smells

22 Code Smells

For each smell, there is a set of refactoring steps to solve them.

Many of them can also be applied to JavaScript.

Code Smells

Long Method

Refactoring steps:

- Extract method
- Replace temp with query

Code Smells

Duplicated Code

Refactoring steps:

- Extract method
- Extract class

Code Smells

Large Class

Refactoring steps:

- Extract class
- Extract subclass
- Extract interface

Code Smells

Large File

Refactoring steps:

- Extract file
- Extract methods

Code Smells

Long Parameter List

Shotgun Surgery

Data Clumps

Switch Statements

Lazy Class

Temporary Field

Middle Man

Alternative Classes with Different
Interfaces

Data Class

Divergent Change

Feature Envy

Primitive Obsession

Parallel Inheritance Hierarchies

Speculative Generality

Message Chains

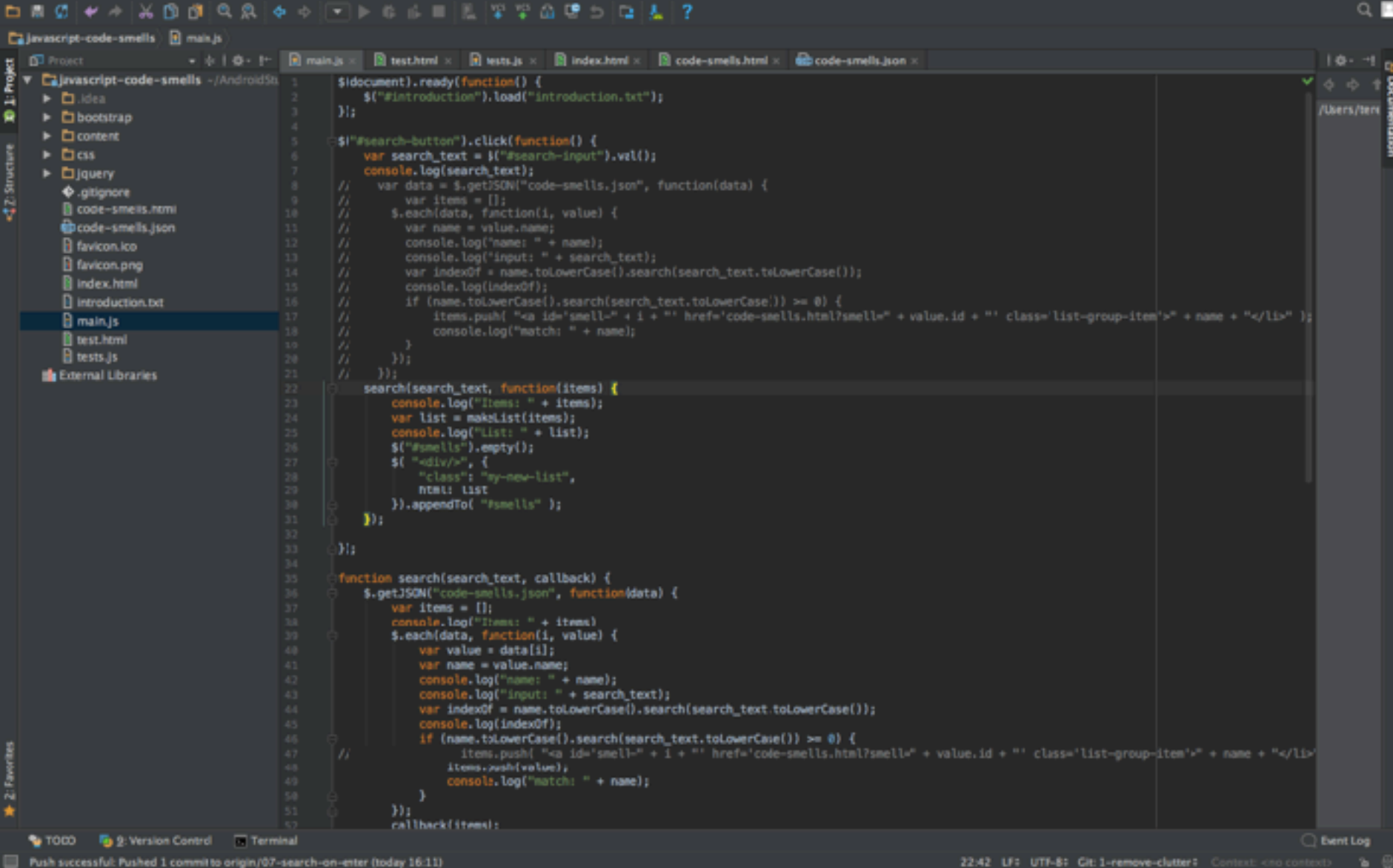
Inappropriate Intimacy

Incomplete Library Class

Refused Bequest

Comments

Let's see an example...



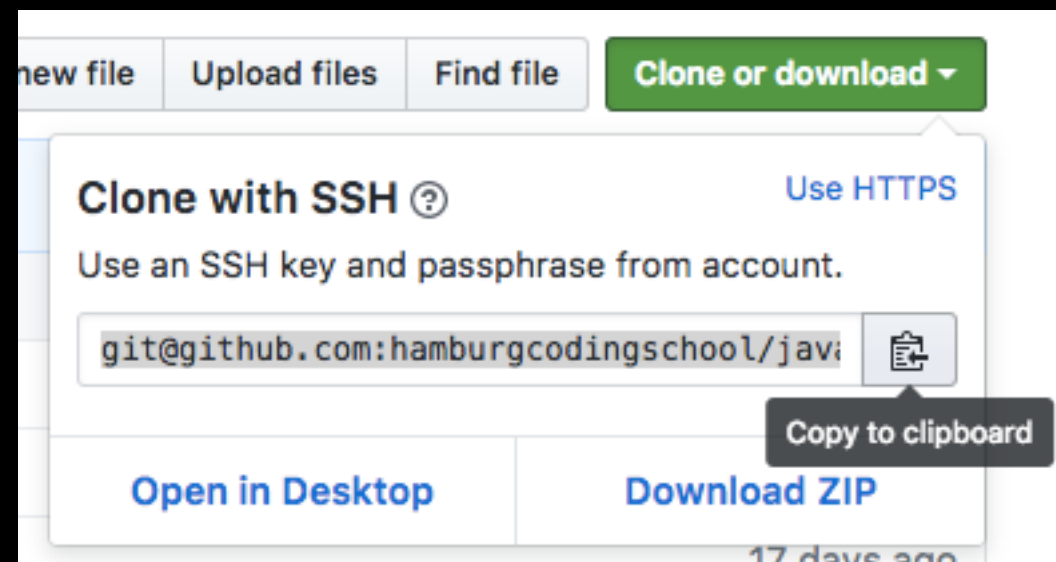
The screenshot shows an IDE with a project named "javascript-code-smells". The file explorer on the left lists files including "main.js", "test.html", "tests.js", "index.html", "code-smells.html", and "code-smells.json". The "main.js" file is open in the editor, showing a search function implemented with jQuery and JSON. The code includes a click event for a search button, a function to fetch data from "code-smells.json", and a function to render the search results as a list of items. The status bar at the bottom indicates a successful push to the origin repository.

```
1 $(document).ready(function() {
2   $("#introduction").load("introduction.txt");
3 });
4
5 $("#search-button").click(function() {
6   var search_text = $("#search-input").val();
7   console.log(search_text);
8   // var data = $.getJSON("code-smells.json", function(data) {
9   //   var items = [];
10  //   $.each(data, function(i, value) {
11  //     var name = value.name;
12  //     console.log("name: " + name);
13  //     console.log("input: " + search_text);
14  //     var indexOf = name.toLowerCase().search(search_text.toLowerCase());
15  //     console.log(indexOf);
16  //     if (name.toLowerCase().search(search_text.toLowerCase()) >= 0) {
17  //       items.push( "<a id='smell-" + i + "' href='code-smells.html?smell=" + value.id + "' class='list-group-item'" + name + "</li>" );
18  //       console.log("match: " + name);
19  //     }
20  //   });
21  // });
22  search(search_text, function(items) {
23    console.log("Items: " + items);
24    var list = makeList(items);
25    console.log("List: " + list);
26    $("#smells").empty();
27    $( "<div>", {
28      "class": "my-new-list",
29      html: list
30    }).appendTo( "#smells" );
31  });
32  });
33
34 function search(search_text, callback) {
35   $.getJSON("code-smells.json", function(data) {
36     var items = [];
37     console.log("Items: " + items);
38     $.each(data, function(i, value) {
39       var value = data[i];
40       var name = value.name;
41       console.log("name: " + name);
42       console.log("input: " + search_text);
43       var indexOf = name.toLowerCase().search(search_text.toLowerCase());
44       console.log(indexOf);
45       if (name.toLowerCase().search(search_text.toLowerCase()) >= 0) {
46         items.push( "<a id='smell-" + i + "' href='code-smells.html?smell=" + value.id + "' class='list-group-item'" + name + "</li>" );
47         items.push(value);
48         console.log("match: " + name);
49       }
50     });
51     callback(items);
52   });
53 }
```

Set Up the Project

Check out the project:

<https://github.com/hamburgcodingschool/javascript-code-smells>



Set Up the Project

Check out the project:

[https://github.com/hamburgcodingschool/
javascript-code-smells](https://github.com/hamburgcodingschool/javascript-code-smells)

```
git clone  
git@github.com:hamburgcodingschool/  
javascript-code-smells.git
```

Set Up the Project

Check out branch:

```
git checkout 3-large-class
```


Set Up the Project

We need to run a local web server.

Do we have python installed?

Mac

Open terminal.
Type:

```
python -v
```

Linux

Open shell.
Type:

```
which python
```

Windows

Command Line.
Type:

```
python
```

Set Up the Project

We need to run a local web server.

No? Then we need to install it:

<https://www.python.org/downloads/>

Set Up the Project

We need to run a local web server.

Once we know we have python installed, we can run our local web server.

Go to the directory where this tutorial project is located.

Set Up the Project

Go to the directory where this tutorial project is located.

Mac & Linux & Windows 7:

```
python -m SimpleHTTPServer
```

Windows 10:

```
py -m http.server
```

Set Up the Project

Serving HTTP on 0.0.0.0 port 8000 ..

Go to your browser and have a look:

<http://localhost:8000/>

<http://localhost:8000/test.html>

Tutorial Steps

1. Remove Clutter
2. Testing
3. Large Class
4. Long Method
5. Imprecise Names
6. Rinse and Repeat

Tutorial Steps

1. Remove Clutter

2. Testing

3. Large Class

4. Long Method

5. Imprecise Names

6. Rinse and Repeat

Tutorial Steps

Branches:

1-remove-clutter

2-testing

3-large-class

4-long-method

5-imprecise-names

6-repeat

7-search-on-enter

8-solution

Tutorial Steps

Branches:

1-remove-clutter

2-testing

3-large-class

4-long-method

5-imprecise-names

6-repeat

7-search-on-enter

8-solution

Tutorial Steps

Check out branch 3:

```
git fetch
```

```
git checkout 3-large-class
```

Testing

How do we make sure we are not breaking anything in the process of refactoring?

We use unit testing.

Testing

In our project:

JUnit

<http://api.qunitjs.com/>

Testing

```
QUnit.test("Search for 'class'", function(assert) {  
    ...  
});
```

```
QUnit.test("Create an <a> tag with makeList()",  
function(assert) {  
    ...  
});
```

```
QUnit.test("Create multiple <a> tags with makeList()",  
function(assert) {  
    ...  
});
```

Testing

If you go to

<http://localhost:8000/test.html>

You will see:

Testing

QUnit Example

☐ Hide passed tests ☐ Check for Globals ☐ No try-catch

Module:

Filter:

Go

QUnit 2.3.2; Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36

3 tests completed in 65 milliseconds, with 0 failed, 0 skipped, and 0 todo.
8 assertions of 8 passed, 0 failed.

- | | | |
|--|-------|-------|
| 1. Search for 'class' (6) | Rerun | 45 ms |
| 2. Create an <a> tag with makeList() (1) | Rerun | 0 ms |
| 3. Create multiple <a> tags with makeList() (1) | Rerun | 0 ms |

Large Class

“When a class is trying to do too much, it often shows up as too many instance variables. When a class has too many instance variables, duplicated code cannot be far behind.”

“A class with too much code is prime breeding ground for duplicated code, chaos, and death.”

~ Fowler 1999

Large Class

Refactoring steps:

- Extract class
- Extract methods
- Extract variables
- Remove duplicated code

Long Method

Check out branch 4:

```
git checkout 4-long-method
```

Long Method

“The Long method code smell is a sign that you possibly need to take some part of related functionality in your method and create a new method to hold this functionality.”

~ Fowler 1999

Long Method

Refactoring steps:

- extract method
- replace temp with query

Goal: method no longer than 5 lines of content.

Long Method

Extract method:

```
function search(search_text, callback) {
    $.getJSON("code-smells.json", function(data) {
        var items = [];
        $.each(data, function(i, value) {
            var value = data[i];
            var name = value.name;
            var index0f = name.toLowerCase().search(search_text.toLowerCase());
            if (name.toLowerCase().search(search_text.toLowerCase()) >= 0) {
                items.push(value);
            }
        });
        callback(items);
    });
}
```


Long Method

```
function search(search_text, callback) {  
    $.getJSON("code-smells.json", function(data) {  
        var items = [];  
        $.each(data, function(i, value) {  
            var value = data[i];  
            var name = value.name;  
            var indexOf = name.toLowerCase().search(search_text.toLowerCase());  
            if (name.toLowerCase().search(search_text.toLowerCase()) >= 0) {  
                items.push(value);  
            }  
        });  
        callback(items);  
    });  
}
```

Long Method

```
function search(search_text, callback) {  
    $.getJSON("code-smells.json", function(data) {  
        var items = [];  
        $.each(data, function(i, value) {  
            var value = data[i];  
            var name = value.name;  
            var index0f = name.toLowerCase().search(search_text.toLowerCase());  
            if (name.toLowerCase().search(search_text.toLowerCase()) >= 0) {  
                items.push(value);  
            }  
        });  
        callback(items);  
    });  
}
```

Long Method

```
function search(search_text, callback) {
    $.getJSON("code-smells.json", function(data) {
        var items = [];
        $.each(data, function(i, value) {
            var value = data[i];
            getMatchingItems(search_text, value, items);
        });
        callback(items);
    });
}

function getMatchingItems(search_text, value, items) {
    var name = value.name;
    var index0f = name.toLowerCase().search(search_text.toLowerCase());
    if (name.toLowerCase().search(search_text.toLowerCase()) >= 0) {
        items.push(value);
    }
}
```

Long Method

Replace temp with query:

```
function search(search_text, callback) {  
    $.getJSON("code-smells.json", function(data) {  
        var items = [];  
        $.each(data, function(i, value) {  
            var value = data[i];  
            getMatchingItems(search_text, value, items);  
        });  
        callback(items);  
    });  
}
```

Long Method

Replace temp with query:

```
function search(search_text, callback) {  
    $.getJSON("code-smells.json", function(data) {  
        var items = [];  
        $.each(data, function(i, value) {  
            var value = data[i];  
            getMatchingItems(search_text, value, items);  
        });  
        callback(items);  
    });  
}
```

Long Method

Replace temp with query:

```
function search(search_text, callback) {  
    $.getJSON("code-smells.json", function(data) {  
        var items = [];  
        $.each(data, function(i, value) {  
            getMatchingItems(search_text, data[i], items);  
        });  
        callback(items);  
    });  
}
```

Long Method

Refactoring steps:

- extract method
- replace temp with query

Goal: method no longer than 5 lines of content.

Imprecise Names

Check out branch 5:

```
git checkout 5-imprecise-names
```


Imprecise Names

Readability is key:

You read code more than you write - it costs more to read code than to write it.

- Rename variables and methods wherever you find a better matching name.

Imprecise Names

```
function search(search_text, callback) {  
    $.getJSON("code-smells.json", function(data) {  
        var items = [];  
        $.each(data, function(i, value) {  
            getMatchingItems(search_text, data[i], items);  
        });  
        callback(items);  
    });  
}  
  
function getMatchingItems(search_text, value, items) {  
    var name = value.name;  
    var index0f = name.toLowerCase().search(search_text.toLowerCase());  
    if (name.toLowerCase().search(search_text.toLowerCase()) >= 0) {  
        items.push(value);  
    }  
}
```

Imprecise Names

```
function search(search_text, callback) {
    $.getJSON("code-smells.json", function(data) {
        var items = [];
        $.each(data, function(i, value) {
            getMatchingItems(search_text, data[i], items);
        });
        callback(items);
    });
}

function getMatchingItems(search_text, value, items) {
    var name = value.name;
    var index0f = name.toLowerCase().search(search_text.toLowerCase());
    if (name.toLowerCase().search(search_text.toLowerCase()) >= 0) {
        items.push(value);
    }
}
```

Imprecise Names

```
function search(search_text, callback) {
    $.getJSON("code-smells.json", function(data) {
        var results = [];
        $.each(data, function(i, value) {
            getMatchingItems(search_text, data[i], results);
        });
        callback(results);
    });
}

function getMatchingItems(search_text, value, results) {
    var name = value.name;
    var indexOf = name.toLowerCase().search(search_text.toLowerCase());
    if (name.toLowerCase().search(search_text.toLowerCase()) >= 0) {
        results.push(value);
    }
}
```

Imprecise Names

```
function search(search_text, callback) {
    $.getJSON("code-smells.json", function(data) {
        var results = [];
        $.each(data, function(i, value) {
            getMatchingItems(search_text, data[i], results);
        });
        callback(results);
    });
}

function getMatchingItems(search_text, value, results) {
    var name = value.name;
    var index0f = name.toLowerCase().search(search_text.toLowerCase());
    if (name.toLowerCase().search(search_text.toLowerCase()) >= 0) {
        results.push(value);
    }
}
```

Imprecise Names

```
function search(search_text, callback) {
    $.getJSON("code-smells.json", function(data) {
        var results = [];
        $.each(data, function(i, value) {
            getMatchingItems(search_text, data[i], results);
        });
        callback(results);
    });
}

function getMatchingItems(search_text, value, results) {
    var name = value.name;
    var indexOf = name.toLowerCase().search(search_text.toLowerCase());
    if (name.toLowerCase().search(search_text.toLowerCase()) >= 0) {
        results.push(value);
    }
}
```

Imprecise Names

```
function search(search_text, callback) {
    $.getJSON("code-smells.json", function(data) {
        var results = [];
        $.each(data, function(i, value) {
            addToResultsIfMatch(search_text, data[i], results);
        });
        callback(results);
    });
}

function addToResultsIfMatch(search_text, value, results) {
    var name = value.name;
    var index0f = name.toLowerCase().search(search_text.toLowerCase());
    if (name.toLowerCase().search(search_text.toLowerCase()) >= 0) {
        results.push(value);
    }
}
```

Imprecise Names

Readability is key:

You read code more than you write - it costs more to read code than to write it.

- Rename variables and methods wherever you find a better matching name.

Rinse and Repeat

Repeat until you are satisfied.

Feel free to browse other branches of the project.

8-solution



Teresa Holfeld

Android Developer at Moovel

Twitter: @TeresaHolfeld

GitHub: teresaholfeld



Hamburg Coding School

hamburgcodingschool.com

Twitter: @hhcodingschool

GitHub: [hamburgcodingschool](https://github.com/hamburgcodingschool)