Clean Code

Objectives

By the end of this lesson you will be able to:

— Explain why you should strive for clean code
— Identify why a snippet of code is not clean
— Choose two things you will focus on this week

X

```
var x,y,z;

x = 5;

if (x === 5) {
    console.log(y);
} else { return false; }
function Person (firstname, lastname, age, weight, isAlive, hairColor, height, city, state, country) {
    //Create a Person object
}
```

“Programming is the art of telling another human what one wants the computer to do.”

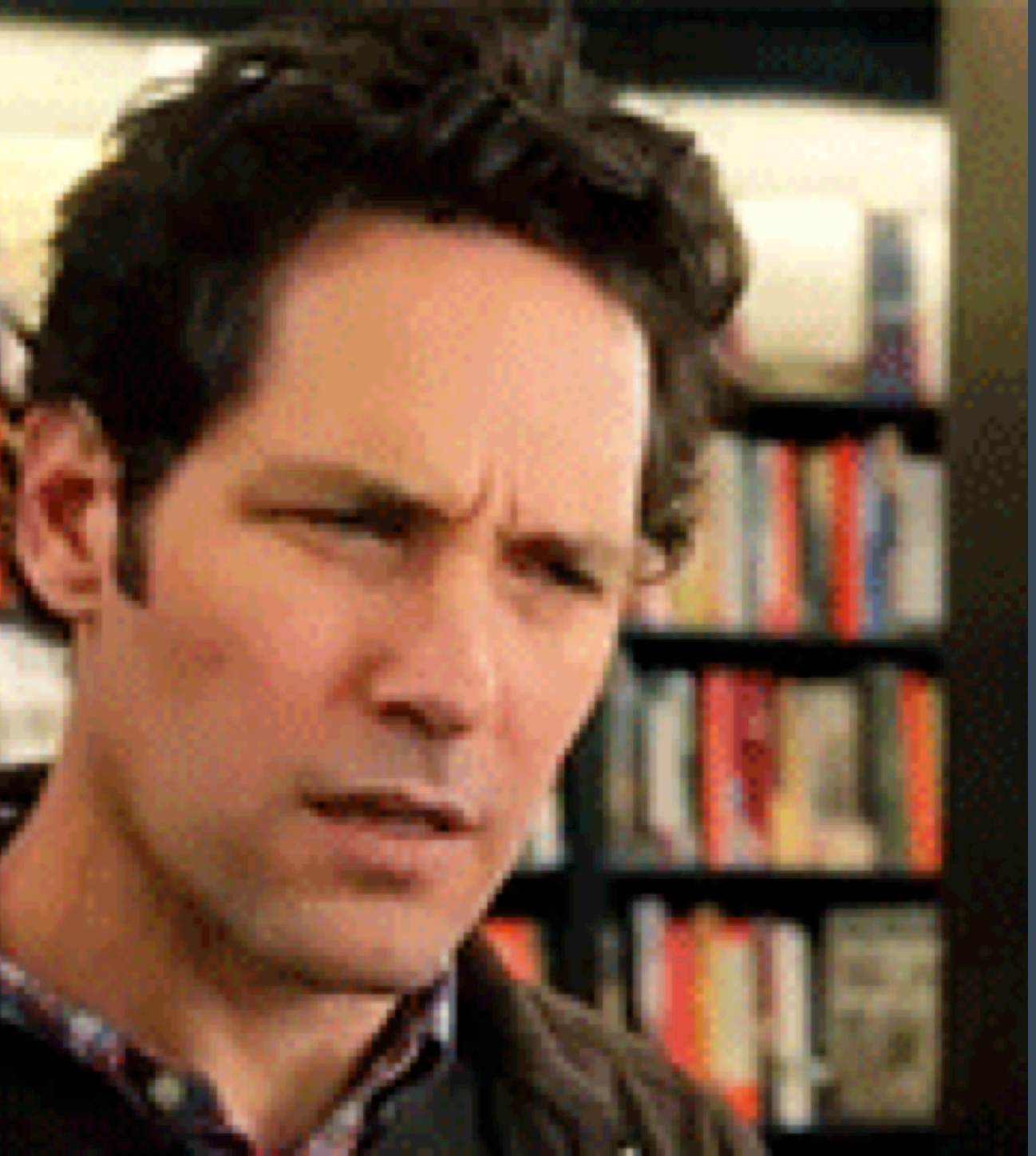
— Donald Knuth

Clean Code Helps You and Teammates:

— Read code
— Change code
— Keep the code working
— Reduce the cost of maintenance
— Fix bugs
— Reduce the time to modify code
— Enjoy programming more
— Make life easier

X

```
function Person (firstname, lastname, age, weight, isAlive, hairColor, height, city, state, country) {
    //Create a Person object
}
```



Many Parameters

√

```
// Can anyone remember the parameters and order of the many version?
function Person (spec) {
    //Create a Person object
    this.firstname = spec.firstname;
    this.isAlive = spec.isAlive;
}
```

X

```
function checkSystem (isReady, message) {var output; if (isReady) {
        output = 'all systems go! ' + message;
} else {
output = 'I can\'t let you do that.'}return output;}
```

Inconsistent Indentation

√

```
function checkSystem (isReady, message) {
    var output;
    if (isReady) {
        output = 'all systems go! ' + message;
    } else {
        output = 'I can\'t let you do that.';
    }
    return output;
}
```

X

```
function calculatePrice (retailPrice) {
    return retailPrice + retailPrice * 0.08 + retailPrice * 0.02;
}
```

Magic Numbers

√

```
function calculatePrice (retailPrice) {
    var salesTax = 0.08;
    var viceTax = 0.02;
    return retailPrice + retailPrice * salesTax + retailPrice * viceTax;
}
```

X

```
function sendMessage() {
    var message = $("#messageBox").val();
    var user = $("#user").val();
    var time = new Date();
    emitMessage({ message: message, user: user });
}
// Did the developer intentionally not use “time”,
// or did he accidentally leave it out and it is a bug?
```

Unused Variables

√

```
function sendMessage() {
    var message = $("#messageBox").val();
    var user = $("#user").val();
    emitMessage({ message: message, user: user });
}
```

X

```
function sendMessage() {
    var message = $("#messageBox").val();
    if (message.length <= 3) {
        throw new Error('message not long enough');
    }
    var user = $("#user").val();
    var message = { message: message, user: user };
    var server = getServer();
    server.connect();
    var result = server.send(message);
    if (!result.success) {
        throw new Error(result.message);
    }
}
```

Long Functions

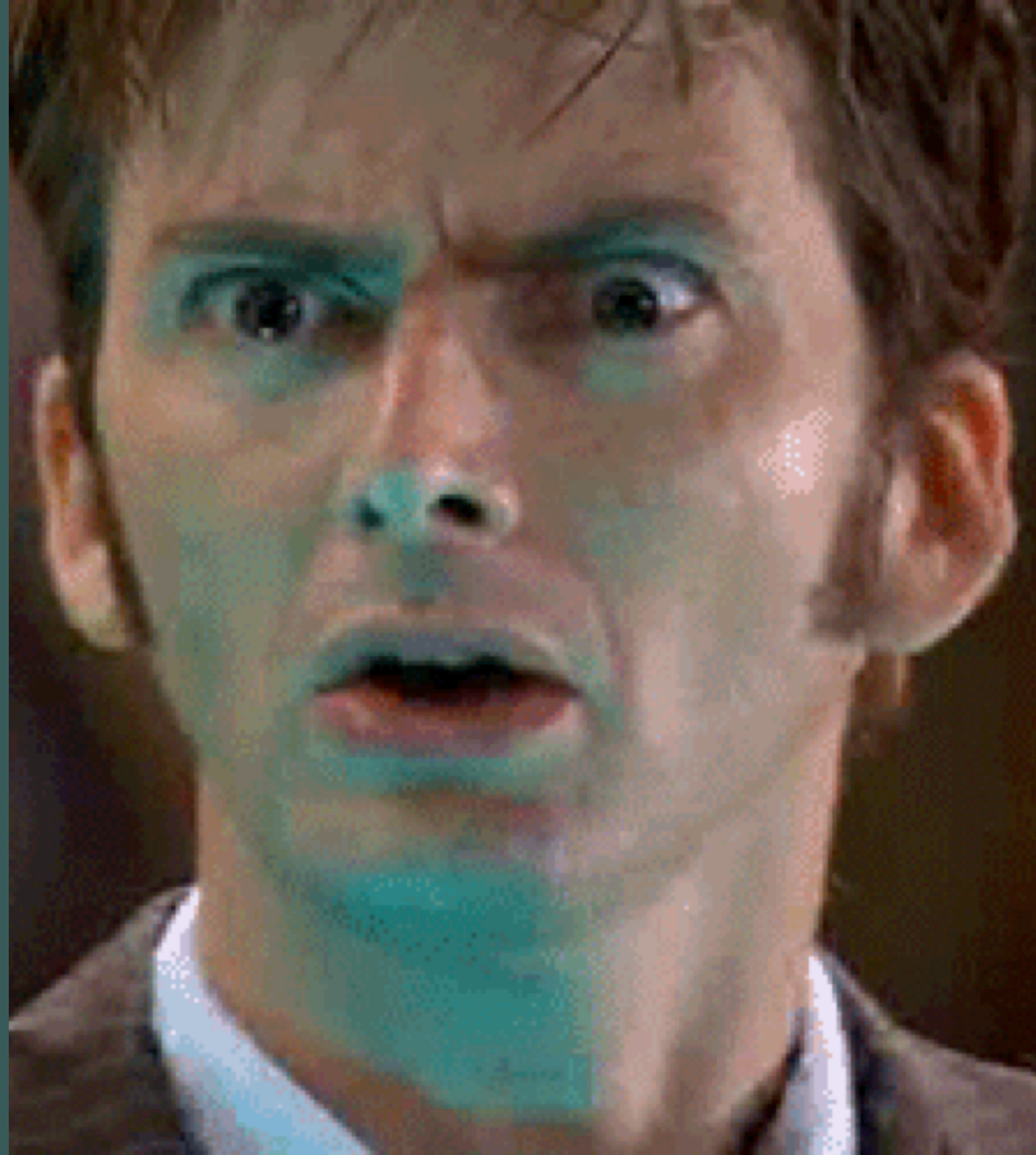
√

```
function sendMessage() {
    //handle errors in functions and handle complexity
    var message = getMessageObject();
    server.sendMessage(message);
}
```

X

```
function checkoutBook() {
    var maxBooks = 3;
    var book = getBook(), user = getUser(), booksOut = user.getBooksOut(), name = user.name;
    if (booksOut.length >= maxBooks) {
        throw new Error('Too many books checked out', user);
    }
    user.checkout(book);
    console.log('Thanks for using the library ' + name);
}
```

Long Lines



√

```
function checkoutBook() {
    var maxBooks = 3;
    var book = getBook();
    var user = getUser();
    var booksOut = user.getBooksOut();
    if (booksOut.length >= maxBooks) {
        throw new Error('Too many books checked out', user);
    }
    user.checkout(book);
    console.log('Thanks for using the library ' + user.name);
}
```

X

```
function p(b, e) {
    if (e === undefined) {
        e = 0;
    }
    var r = 1;
    for (var c = 0; c < e; c++) {
        r *= b;
    }
    return r;
}
```

Confusing

Names

√

```
function power(base, exponent) {
    if (exponent === undefined) {
        exponent = 0;
    }
    var result = 1;
    for (var count = 0; count < exponent; count++) {
        result *= base;
    }
    return result;
}
```

X

```
$('div').click(function () {})
$('div').css('background-color', 'red')
$('div').hide()
```

Bad jQuery Selectors

√

```
var $divs = $('div')
$divs.click(function () {})
$divs.css('background-color', 'red')
$divs.hide()
```

Not Just Javascript

X

```
div table tr td ul li #con, div div div div span #con {color:red;background-color:blue;margin:0 0 0 10px;
padding: 0 10px 10px 100px;}
```

Bad CSS

√

```
#con {
   color: red;
   background-color: blue;
   margin: 0 0 0 10px;
   padding: 0 10px 10px 100px;
}
```

X

```
<body><div><div><span>
puppies</span>
</div>
                </div><div>
                      kittens
</div></body>
```

Bad HTML

√

```
<body>
  <div>
    <div>
      <span>puppies</span>
    </div>
  </div>
  <div>kittens</div>
</body>
```

What will you focus on this week?

Better Readability

- ~~Inconsistent Indentation~~ Use proper indentation and spacing
- ~~Long Lines~~ Use line breaks to make code more readable
- ~~Confusing Names~~ Make names more descriptive

Simple Clean Ups

- ~~Many Parameters~~ Use fewer parameters
- ~~Magic Numbers~~ Create descriptive variables for numbers
- ~~Unused Variables~~ Remove unused variables

Refactor & Make More Efficient

- ~~Long Functions~~ Make functions shorter (do one thing well)
- ~~Bad jQuery Selectors~~ Name selectors and choose smart selectors