Programming Do’s and Don’ts

Monde Duinkharjav
In an ideal world...
In an ideal world...

- Fully tested
- No hacky code
- Nothing DRY
- etc.
But reality sucks… so you can’t have nice things

- Fully tested
  - Even Facebook and Google’s codebase test coverage is really bad

- No hacky code
  - 1. Javascript the language by nature is hacky
  - 2. You’re under a tight deadline. You will inevitably write hacky things

- Nothing DRY
  - If making non-DRY code requires a complete overhaul of your codebase structure, it’s often not worth it.
But most importantly…

1. You often don’t know what’s the best way to do something until you do it wrongly and realize what matters and what doesn’t

2. You often have better things to be spending your time on than writing tests for a helper function
Good Coding Practices

The Important Parts
I’m working on a lab assignment for a course.
I’ve been implementing a huge function for the last 3 hours.

“I’m really close to being done,” I think…
I have a nagging feeling none of the code I’ve written so far actually works but
the sunk cost of the last 3 hours is really stressing me out.
Bad Solution #1: What I did on some other occasion

I feel super lost, ask my teammate to debug my code and run for the hills…

That’s a cruel thing to ask of your teammate.
Bad Solution #2: What I did for this anecdote

“I’m almost done I’ll just finish this and then debug everything in sweep.”

I proceed to be met with 30+ errors in my code... the first of which begins with me having forgotten to close a bracket somewhere.
Decent Solution: What I should’ve done

Stop right there, take a bathroom / water break.

Sit back down, comment everything out and uncomment little by little and debug everything and incrementally get rid of my bugs.
Best Solution: What I would do now

Never be in that situation in the first place.
Lesson 1: Code in small chunks, make incremental changes
Code in small chunks, make incremental changes

The progression of the Catbook Workshops is a very good example of how you break down applications/features.

The biggest hurdle for making progress in a project is having a huge thing... that doesn’t work.
Lesson 2: Rush to get something that WORKS
Rush to get something that WORKS

Idea: Minimize the number of moving parts you’re keeping track of in your head.

When you’re in a rush to get some feature working, code quality can take a back seat for 10 minutes… if you do something hacky, jot down a TODO as a comment in your code and come back to it after you got your thing working.
A couple of warnings regarding variable names

1. If you find a need to rename a variable, do it IMMEDIATELY. Your text editor is your friend, use its find and replace feature!

   Incorrect variable names confuse your head when you’re trying to think about the underlying problem.

2. If you keep renaming a variable… something in your LOGIC is flawed. You’re not thinking about the problem correctly. Take a step back and think why.

3. Shape of your code matters. If your code looks ugly, it’s harder to debug it!
Personal Anecdote #2

We’re at a hackathon... it’s 5AM. My teammate wrote some code and left to take a nap this is what he left us...

```javascript
$("#song-details").html(totalOperation2);
results = document.getElementsByClassName("resultsList");
resultsTable = document.getElementsByClassName("result");

for (i = 0; i < results.length; i++) {
    results[i].id = 'result' + i;
    resultsTable[i].id = 'resultT' + i;
    if (i != 0) {
        $(results[i]).css('display', 'none')
    }
}
resultsIndex = -1;
var clicked = $('#invis');
$('.result').hover(function() {
    if (this != clicked) {
        $(this).toggleClass('invis');
    }
});
```
Personal Anecdote #3

I have code that I’m sure works. And I just added some new code and I’m fairly certain it should work but I haven’t actually tested it yet.

Before testing if my new addition works, I realize that I wanna do some housekeeping: e.g. refactor code into separate functions, move some functions into a different file, change some variable names, add some comments, delete print statements etc.
Bad Solution #1

I apply these changes and push to your repo without much thought. I mean WCGW right? I only made a small change and moved around some code... that’s how I crash my site.
Bad Solution #2: What I did

Apply all the changes and then try testing my new additions. Now I’m stuck with a hodge podge mess where you just don’t know what the cause of the bug is… is it my new code or all the other things I did…

Number of moving parts has increased >:(

Good Solution

Test my code make sure it works and then make the changes.
Better Solution: What I should’ve done

Test my code make sure it works... commit my changes and then do the housekeeping and make a follow up commit to apply the housekeeping.
Lesson 3: You will do hacky things when you’re desperate... make sure to keep track of them
Dealing with doing hacky things

If you’re doing something hacky, make sure to comment right next to it saying it’s a hack.

Hacks are the origin of many bugs. These comments give you a good heuristic for locating a bug.
Lesson 4: You don’t know how your code should be organized until it is disorganized
Code organization

There’s a fine balance to reach: Your code should have enough organization to avoid making it difficult to develop your application.

One side: You want to make helper functions and keep some level of organization so that you’re not always dealing with 100s of lines of code in a single function call.

The other side: You’re not writing Facebook’s backend. Don’t go over your head on code structure.
Summary

Code in small chunks

Make incremental changes

Be in a rush to get something that WORKS

Comment when you write bad code and you can’t help it

Don’t overthink code organization. Just do it as it comes
How to Debug Effectively

How to make the experience less painful
Personal Anecdote #3

I’m debugging like 200 lines of code cuz it just doesn’t work. How do I do it?
Solution

Never be in that place to begin with
Lesson 5: Test in small chunks. Test components in isolation.
Lesson 6: Make an environment where your debugging loop is as low as possible
What is a debugging loop

Test your app ←→ Make a change

This is important since

1. The few extra seconds per iteration really add up
2. The tedium of the long annoying debugging loop really gets on your nerves
How to address this?

Create debugging tools like systematic print statements

Halt the execution of your code. Use breakpoints in your chrome inspector.
Personal Anecdote #4

I have a really large system already and I need to add more modules and debug things as bugs pop up.
Bad Solution

Every time a bug pops up write new print statements to figure out what’s going on.
Good Solution:

Create a debugging helper function that shows key system states so you have a better grasp of what is going wrong. You can use this helper function in between code that changes your state and make sure everything happens as expected.
Various small things
Spot the bug!

```javascript
for (let i=0; i < width; i++) {
  for (let j=0; j < height; j++) {
    for (let k=0; k < depth; j++) {
      
    }
  }
}
```
Correct Implementation

```javascript
for (let i=0; i < width; i++) {
    for (let j=0; j < height; j++) {
        for (let k=0; k < depth; k++) {
        }
    }
}
```
Spot the bug!

We’re trying to change the contents of array arr to be the average of itself, and its left and right neighbors.

```javascript
for (let i=0; i < arr.length; i++) {
    const lo = max(i-1, 0);
    const hi = min(i+1, arr.length-1);
    arr[i] = (arr[lo] + arr[i] + arr[hi]) / 3;
}
```
Correct Implementation

We’re trying to change the contents of array arr to be the average of itself, and its left and right neighbors.

```javascript
let out = [];
for (let i=0; i < arr.length; i++) {
    const lo = max(i-1, 0);
    const hi = min(i+1, arr.length-1);
    out.push((arr[lo] + arr[i] + arr[hi]) / 3);
}
arr = out;
```
You’ll be implementing two types of functions

Observers:

Does not change the state of any objects. Only tells you what the current state is

Mutators:

Changes the state of an object.
Be VERY careful when reading documentation

Example

Extract parts of a string:

```javascript
var str = "Hello world!";
var res = str.substr(1, 4);
```

The result of `res` will be:

`ello`

Example

Extract characters from a string:

```javascript
var str = "Hello world!";
var res = str.substring(1, 4);
```

The result of `res` will be:

`ell`
Last Personal Anecdote

I had an array. I kept track of a variable called `maxIdx`.

This variable used to keep track of the index of the last element in the array.

At some point I realized it made more sense to keep track of the array length instead of the max index so I changed the implementation.

BUT not the variable name. This failure to notice and make this change came back to make me miserable.

I eventually changed my variable name to `arrLen`