Gmail

Dhaval Chheda <kiddo.dhaval@googlemail.com>

[Just JavaScript] 01. Mental Models
1 message

Dan Abramov <dan@overreacted.io> 27 May 2020 at 13:10
To: Dhaval <kiddo.dhaval@gmail.com>

Read this code:

```
let a = 10;
let b = a;
a = 0;
```

What are the values of `a` and `b` after it runs? Work it out in your head before reading further.

If you've been writing JavaScript for a while, you might object: "This snippet is much simpler than the code I'm writing every day. What's the point?"

The goal of this exercise isn't to introduce you to variables. We assume you're already familiar with them. Instead, it is to make you notice and reflect on your *mental model*.

What's a Mental Model?

Read the code above again with an intention to *really be sure* what the result is. (We'll see why this intention is important a bit later.)

While you're reading it for the second time, pay close attention to what's happening in your head, step by step. You might notice a monologue like this:

- `let a = 10;`
  - Declare a variable called `a`. Set it to `10`.

- `let b = a;`
    - Declare a variable called `b`. Set it to `a`.
    - Wait, what's `a` again? Ah, it was `10`. So `b` is `10` too.
- `a = 0;`
    - Set the `a` variable to `0`.
- So `a` is `0` now, and `b` is `10`. That's our answer.

Maybe your monologue is a bit different. Maybe you say "assign" instead of "set", or maybe you read it in a slightly different order. Maybe you arrived at a different result. Pay attention to how exactly it was different. Note how even this monologue doesn't capture what's really happening in your head. You might say "set `b` to `a`", but what does it even mean to *set* a variable?

You might find that for every familiar fundamental programming concept (like a variable) and operations on it (like setting its value), there is a set of deep-rooted analogies that you associated with it. Some of them may come from the real world. Others may be repurposed from other fields you learned first, like numbers from math. These analogies might overlap and even contradict each other, but they still help you make sense of what's happening in the code.

For example, many people first learned about variables as "boxes" into which you can put stuff. Even if you don't vividly imagine boxes anymore when you see a variable, they might still behave "boxy" in your imagination. These approximations of how something works in your head are known as "mental models". It may be hard if you've been programming for a long time, but try to notice and introspect your mental models. They're probably a combination of visual, spatial, and mechanical mental shortcuts.

These intuitions (like "boxiness" of variables) influence how we read code our whole lives. But sometimes, our mental models are wrong. Maybe a tutorial we read early on has sacrificed correctness for the ease of explaining. Maybe we incorrectly transferred an intuition about a particular language feature, like `this`, from another language we learned earlier. Maybe we inferred a mental model from some piece of code and never really verified if it was

mental model from some piece of code and never really verified if it was accurate.

Identifying and fixing these problems is what *Just JavaScript* is all about. We will gradually build (or, possibly, rebuild) your mental model of JavaScript to be accurate and useful. A good mental model will help you find and fix bugs faster, understand other people's code better, and feel confident in the code you write.

(By the way, `a` being `0` and `b` being `10` *is* the correct answer.)

Coding, Fast and Slow

"Thinking, Fast and Slow" by Daniel Kahneman is a widely popular non-fiction book. Its central thesis is that humans use two different "systems" when thinking.

Whenever we can, we rely on the "fast" system. We share this system with many animals, and that gives us amazing powers like walking without falling all the time. This "fast" system is good at pattern matching (necessary for survival!) and "gut reactions". But it's not good at planning.

Uniquely, thanks to the development of frontal lobe, humans also possess a "slow" thinking system. This "slow" system is responsible for complex step-by-step reasoning. It lets us plan future events, engage in arguments, or follow mathematical proofs.

Because using the "slow" system is so mentally draining, we tend to default to the "fast" one — even when dealing with intellectual tasks like coding.

Imagine that you're in the middle of a lot of work, and you want to quickly identify what this function does. Take a quick look at it:

```
function duplicateSpreadsheet(original) {
  if (original.hasPendingChanges) {
    throw new Error('You need to save the file before
```

```
you can duplicate it.');
  }
  let copy = {
    created: Date.now(),
    author: original.author,
    cells: original.cells,
    metadata: original.metadata,
  };
  copy.metadata.title = 'Copy of ' +
original.metadata.title;
  return copy;
}
```

You've probably noticed that:

- This function duplicates a spreadsheet.
- It throws an error if the original spreadsheet isn't saved.
- It prepends "Copy of" to the new spreadsheet's title.

What you might *not* have noticed (great job if you did though!) is that this function *also* accidentally changes the title of the original spreadsheet.

Missing bugs like this is something that happens to every programmer, every day. But now that you know a bug exists, will you read the code differently? If you've been reading code in the "fast" mode, it's likely you'll switch to the more laborious "slow" mode to find it.

In the "fast" mode, we guess what the code does based on naming, comments, and its overall structure. In the "slow" mode, we retrace what the code does step by step.

That's why having a correct mental model is so important. Simulating a computer in our heads is hard enough — and this effort is wasted with wrong mental models.

Don’t worry if you can’t find the bug at all. This means you’ll get the most out of this course! Over the next modules, we’ll rebuild our mental model of JavaScript together so that you see this bug plain as day.

In the next module, we'll start building mental models for some of the most fundamental JavaScript concepts — values and variables.

Please click here to send feedback and complete a short survey about this module.