THE INNER JSON EFFECT

by TJ Mott in Feature Articles on 2016-07-27

TJ Mott

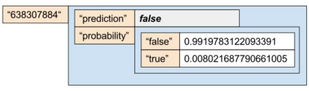
I've been a developer for the past seven years. Currently I'm in the aerospace industry and work with a variety of programming languages and operating systems.

Jake eagerly stepped into his new job, grateful for more experience and new challenges, craving to learn new software stacks and see what his new company had to teach him about the world of software.

They told him he'd be working on some websites, dealing with JavaScript, Node.js, JSON, and the like. It sounded pretty reasonable for web development, except for the non-technical interviewer's comment that it was all "built on top of Subversion" which he assumed was a simple misunderstanding.

Then he was thrust into a project using the company's custom "JSON-based Domain Specific Language", or JDSL. His boss told him to check out a copy of the project he'd be assigned to, and spend a week or two getting familiar with it. "Just ask anyone for help if you have questions, but you shouldn't have any trouble judging from your experience."

```
{
  "638307884": {
    "prediction": false,
    "probability": {
      "false": 0.9919783122093391,
      "true": 0.008021687790661005
    }
  }
}
```



So Jake began an SVN checkout... and long story short it took two days to complete. When he asked about it, his coworker Scott told him, "Oh that's normal. Just play Solitaire or something until it finishes."

Two days later he started poking around. He started with a seemingly-innocuous file called "customers.json" and stared in confusion at its contents:

```
{
    "File" : "Customers.json",
    "Class" : "Customers",
    "Author" : "redacted@redacted.com",
    "Purpose" : "",
    "Functions" : [
        568,
        899,
        900,
        901,
        902,
        1877,
        2880
    ]
}
```

The project was full of such files, along with some apparently-incomplete code files such as this one called "customers.js":

```
Customers.prototype.UpdateBillingInfo = function(info)
{
    this.cc = info.cc;
    this.type = info.type;
    this.name = info.name;
    this.expM = info.expM;
    this.expY = info.expY;
    this.ccv = info.ccv;

    /* snip a bunch of similar lines */
    this.saveToDatabase();
};
```

After a couple days of spelunking through the codebase, and not finding even a single code comment, Jake could make no sense of what he was seeing, and finally asked for help. A coworker named Scott was available and sat with him to walk through some things.

"Oh, you just don't get it yet," he began. "JDSL was written by Tom. He's a super-genius and wrote JDSL himself. So basically that customers.json is just metadata used to put together the Customers class." He waited for Jake to "get it".

He didn't. "So...how do I run it?" he asked.

Scott laughed. "You wouldn't want to 'just' run it. It takes a couple days for a new deployment to finish starting up. JDSL can be a little slow, but it's really powerful. *Really* powerful. Like I said, Tom is a genius."

Jake still wasn't "getting it." "So walk me through this metadata file. What does it do?"

Scott laughed again. "This is the genius part. See here where it says 'Class'?"

"Uh-huh."

"Well that's the class name. Now, see down here where it says 'functions'?"

"Yeah."

"Well those are subversions link to all the functions that make up the class!"

"...I still don't understand..." Jake responded. Inwardly he thought he started to understand, but prayed he was wrong.

"So you have 'customers.json' and 'customers.js'. The JSON file is the metadata and the JS file has all the code. So the list of functions in the JSON file tells JDSL to look up those revisions of the JS file to find what functions are available. In this case the actual code is in revisions 568, 899, 900, 901, and so on."

Jake blinked slowly, hoping he was just being hazed. "Um..."

"Each revision of "customers.js" has one function, so to add functions all you have to do is check in your new code and update the JSON metadata file with the new revisions!"

Jake's confusion turned into incredulity.

"Whenever something makes a function call on a Customer object, JDSL uses the list of function revisions to check out all the actual functions until it finds a match! Understand?"

"...I think so..."

"Like I said, Tom is a genius! This lets you track every function that has ever existed. You can add new functions by overwriting the JS file and adding a new revision to the JSON, and you can remove a function just by removing its revision number from the function list. And it's still there in history, inactive but never lost!" Scott stood. "Let me know if you have any more questions," he said as he left Jake's desk.

Armed with Scott's insight into JDSL, Jake slowly began to understand the system, checking out multiple revisions of each file so he could piece them together and see what was going on at runtime. He soon realized it was merely a web portal to allow customers to update their personal information, but thanks to the complexity of JDSL it took days to do coding work that should only take minutes.

As he went through the code, still familiarizing himself with it, he started checking in code comments to help him and his coworkers map together the convoluted mess, and even fixed a few obvious bugs he found just by reading the code. He did this one class at a time, and at the end of the week he updated and checked in all the JSON 'metadata' files to use the new function revisions.

Monday morning, he showed up to a virtual firestorm. Everyone was in a panic. "Something broke with JSDL and our customer database got scrambled!" Scott quickly explained as he passed Jake in the hallway.

"You!" a voice boomed.

Jake stopped and turned to face a tall, lanky, pale blond man who was obviously angry. "Are you Jake? The new guy?"

"Yes," he answered carefully.

"I'm Tom. You broke JDSL!"

"Uh, what?" Jake had only been looking at the customer portal. How could he have caused any problems?

"You broke JDSL!" he screamed. "I'm reporting you to the bosses and having you fired!" And Tom turned and stormed off, leaving Jake standing confused.

Shortly afterwards, Jake was summoned to a small conference room. Tom, an employee from HR, and a couple Vice Presidents waited for him. Tom looked like he was stewing and could boil over any minute.

"Tell us what you did to JDSL," one of the VP's asked.

"I don't *think* I did anything," Jake answered. "I've only been here two weeks, trying to learn JDSL and how the customer portal works. I don't even know how to deploy it!"

"You made a few commits to Subversion!" Tom shouted.

"Well, yes. I added a few code comments, trying to–"

"You *can't* use comments in JDSL!" Tom shouted. "THAT'S WHAT BROKE IT!!"

Jake stayed silent, trying to process how code comments could wipe out a customer database. Tom continued after a pause. "I haven't added comment support to JDSL, so the runtime executes comments like normal code! You must have had database updates in some comments?!"

"Well, yeah, I put a couple short syntax examples in a comment to clarify–"

Tom burst to his feet. "I knew it! You BROKE IT!" He turned to face the VPs. "I can't deal with coders who don't understand the system! You will either fire Jake...or I quit!" And he stormed out of the room.

The VPs turned to the HR representative and talked as if Jake wasn't even in the room. "I think our course of action is pretty clear. Tom's a programming virtuouso and our best resource, and Jake *did* delete the database. We have to fire Jake."

And so Jake moved on to greener pastures. *Much* greener pastures. Ones where production systems didn't do dozens of SVN file checkouts for each function call at runtime. Ones where production systems didn't automatically use the latest trunk. And ones that didn't come to a complete standstill because a newbie checked in a code comment.

View all 84 comments »

« It's Log, Log, Log

Exit Thread »