Bash is good for processing text, but nowadays a lot of stuff is JSON: configs, APIs, etc. And bash is horrible with manipulating JSON. Here we have various tool for processing JSON, and I think you heard or used one of them – jq.

But when I want to do something more complex with it I always fail as it has unfamiliar syntax, and, in fact, it is completely independent language. I wanted more simple and more familiar tool for working with JSON.

What is the best language to work with JSON? I think the answer is contained in its name – JavaScript Object Notation. So I created new tool 😁

In fact, two tools – in JavaScript and in Go:
- **fx** (JavaScript, Node.js)
- **xx** (Go, Otto, a JavaScript interpreter in Go)

Main idea what you can pipe any JSON in it and pass anonymous function as argument. That’s it. Nothing to remember. 😊

```bash
$ echo '{"foo": [{"bar": "value"}]}'] | fx 'x => x.foo[0
value

fx** has a lot of useful features. For example to simple get value from JSON start with . (dot), and you will get an unquoted, raw value:

```bash
$ echo '{"foo": "value"}'] | fx .foo
value

You can refer to passed JSON object by this keyword. And it’s possible to pass any number of anonymous functions for reducing JSON:

```bash
$ echo '{"foo": [{"bar": "value"}]}'] | fx '.foo' 'this[
value

One of nice features of **fx** is updating JSON using spread operator:

```bash
$ echo '{"count": 100}'] | fx '{...this, count: ++this.c
{
    "count": 101
}
```
Also **fx** and **xx** have standalone binaries (see releases pages on GitHub).

---

**Eat 🧀**

Also I’ve created tool called **eat**, which can “eat” everything including **json**, **yaml**, **toml**, **xml**, **ini** and output JSON for next processing with **fx**:

```bash
$ cat response.xml | eat | fx .Document.Title
```

---

**Benchmarks**

Let’s benchmark **fx/xx** with jq. For it we will be using awesome **hyperfine**. We take jq’s example from tutorial on crunching GitHub API commit response.

```bash
cat commits.json | jq "[.[] | {message: .commit.message,
parents: [.parents[] .html_url]]
```

```bash
cat commits.json | fx "this.map(x => ({message: x.commit.
name, parents: x.parents.map(p => p.html_url)}))"
```

```bash
cat commits.json | xx "this.map(function (x) { return {
```

---

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Description</th>
<th>Time (mean ± std)</th>
<th>Range (min – max)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>11.9 ms ± 1.4 ms</td>
<td>10.0 ms – 17.5 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>170.3 ms ± 4.4 ms</td>
<td>167.0 ms – 179.6 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13.0 ms ± 1.5 ms</td>
<td>11.5 ms – 19.2 ms</td>
</tr>
</tbody>
</table>
Node.js version was significantly slower than jq or xx. But this is due to big startup time of any node application. 😐 What if we benchmark on a really big file?

```
ll big.json
-rw-r--r-- 1 anton staff 36M 9 sep 00:07 big.json
```

![Benchmark Chart]

`fx` performs twice faster than `jq` written in C, and more than 3 times faster `xx`. But how JavaScript version can be faster? It's because v8 faster `jq`, and `otto`.

...  

So should we abandon `jq`? No, of course not. But I hope that these new tools can be useful for somebody. And they will find their users 😊.

Hi, I'm Anton. If you liked the post
If you found a typo edit post on GitHub.

← Immutable list implementation
How to write an expression engine? →