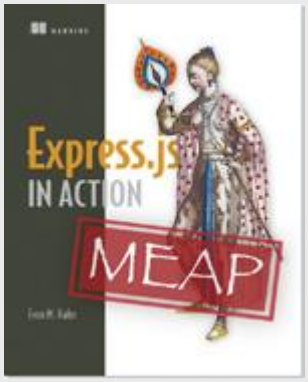

Express.js: How to version your API

By Evan Hahn

What if you decide down the road that you want to update your API without having all of the people who use your API have to update their code? In this article, excerpted from Express.js in Action, I'll talk about versioning your API.

Let me walk you through a scenario.

You want to create an API that lets developers get timezone information. Perhaps they put in a latitude and longitude and your API responds with the timezone in that location, or perhaps you've cooked up some other clever feature. In any case, you've built an API that handles timezones. You design a public API for your timezone app and it becomes a big hit. People all over the world are using it. People are using it to find times all across the globe. It's working well.

But, after a few years, you want to update your API. You've decided you want to change something, but there's a problem: if you change it, all of the people using your API will have to update their code. At this point, you might feel kinda stuck. What do you do? Do you make the changes you want to make and break old users, or does your API stagnate and never stay up-to-date?

There's a solution to all of this: version your API.

All you have to do is add some version information to your API. So a request that comes into this URL might be for version 1 of your API:

```
/v1/timezone
```

And a request coming into version 2 of your API might visit this URL:

```
/v2/timezone
```

This allows you to make changes to your API by simply making a new version! Now, if someone wants to upgrade to version 2, they'll do it by consciously changing their code, not having a version pulled out from under them.

Express makes this kind of separation pretty easy through its use of routers, which we saw in the previous chapter.

To create version 1 of your API, you can create a router that handles version 1 exclusively.

The file might be called api1.js and look like this:

Listing 1 Version 1 of your API, in api1.js

```
var express = require("express");

var api = express.Router();   #A

api.get("/timezone", function(req, res) {      #B
res.send("Sample response for /timezone");
});
api.get("/all_timezones", function(req, res) { #B
res.send("Sample response for /all_timezones");
});
module.exports = api;  #C
```

#A Create a new Router, a mini-application.
#B These are just some example routes. You can add whatever routes or middleware you want to these routers.
#C Export the router so that other files can use it.

Notice that "v1" doesn't appear anywhere in the routes. To use this router in your app, you'll create a full application and use the router from your main app code. It might look like Listing 2.

Listing 2 The main app code in app.js

```
var express = require("express");

var apiVersion1 = require("./api1.js");   #A

var app = express();

app.use("/v1", apiVersion1);  #A

app.listen(3000, function() {   console.log("App
started on port 3000"); });
```

#A Require and use the router, like we saw in the previous chapter.

Then, many moons later, you decide to implement version 2 of your API. It might live in api2.js. It'd also be a router, just like api1.js. It might look like Listing 3.

Listing 3 Version 2 of your API, in api2.js

```
var express = require("express");

var api = express.Router();

api.get("/timezone", function(req, res) {      #A
  res.send("API 2: super cool new response for /timezone");
});
module.exports = api;
```

#A Once again, notice that these are just some example routes.

Now, to add version 2 of your API to the app, simply require and use it just like version 1:

Listing 4 The main app code in app.js

```
var express = require("express");

var apiVersion1 = require("./api1.js"); var

apiVersion2 = require("./api2.js");   #A

var app = express();

app.use("/v1",  apiVersion1);app.use("/v2",
apiVersion2);  #A

app.listen(3000, function() {   console.log("App
started on port 3000"); });
```

#A Note the two new lines. It's just like using version 1 of the router!

You can try visiting these new URLs in your browser to make sure that the versioned API works.

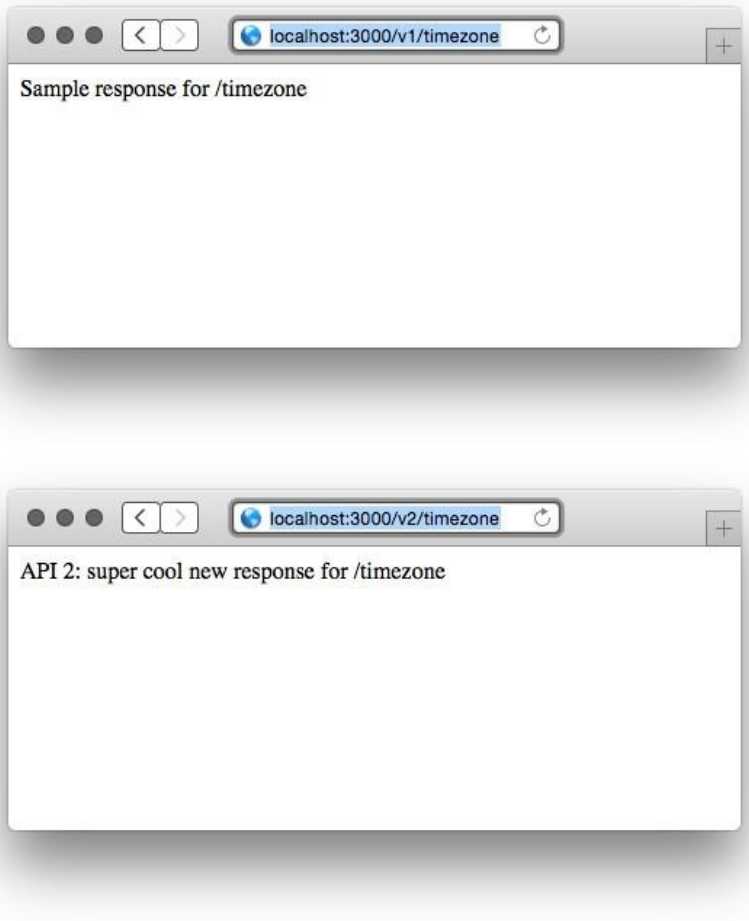

Figure 1 Testing the two API versions in your browser.

You can also use the cURL tool to test your app at the command line.

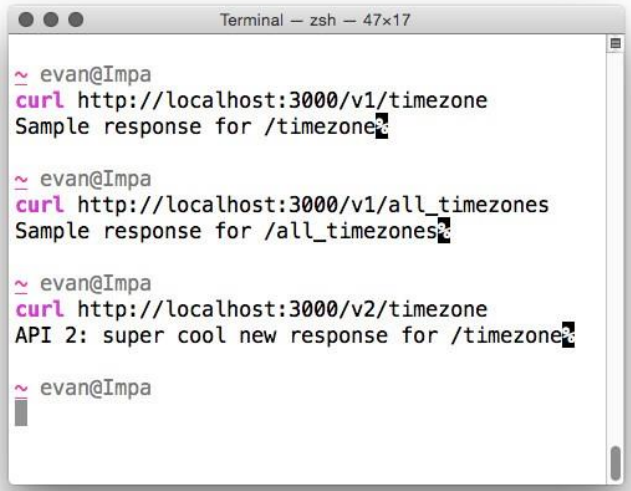

Figure 2 Testing your versioned API using the cURL command-line tool.