

At lesson 186. We're taught about the importance and how to version our API's.

*"There's no standardized way to version Node API's"*. Sure we can follow a course, a friend or a company way to version your API's, but it is not a standar way.

As we can see in the *Image 1*, we created a folder named **v1** which will hold all the routes from the Version 1 of our API, later on we added the file **router-api-v1.js** (see *Image 2* for reference) that imports all those routes from the version 1 (you can see this file is exactly like the **api.js** file in the course).

Image 1

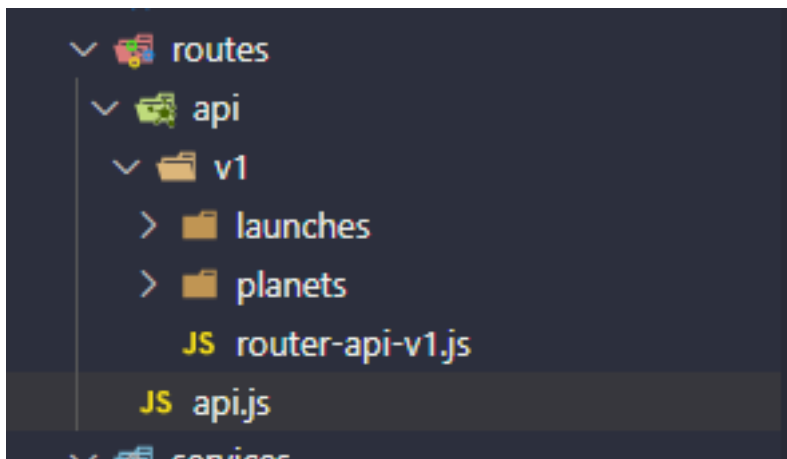


Image 2 (router-api-v1.js file)



Now we do still have the **api.js** file but we changed the way it works.

### Image 3 (api.js file)

A screenshot of a code editor window titled "JS api.js" with a close button. The editor shows the following JavaScript code:

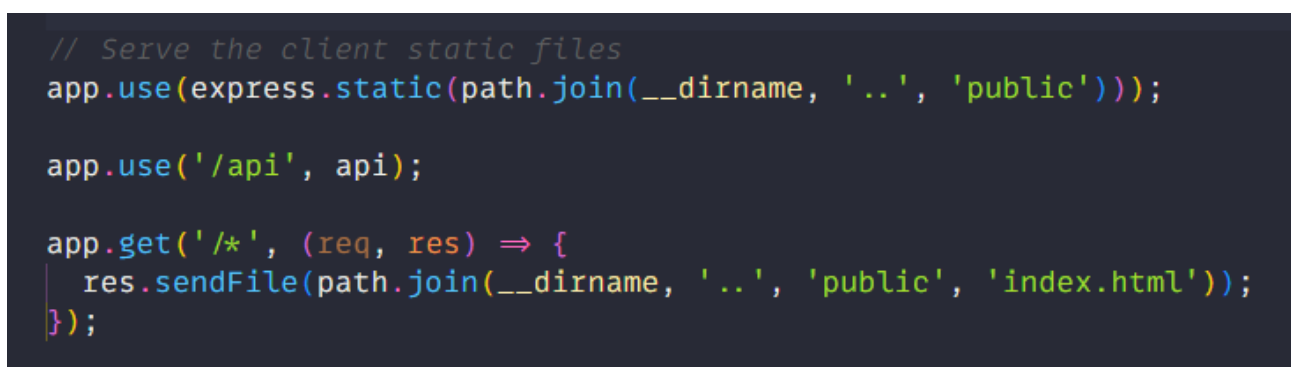
```
server > src > routes > JS api.js > ...
1  const express = require('express');
2  const routerApiV1 = require('./api/v1/router-api-v1');
3
4  const api = express.Router()
5
6  api.use('/v1', routerApiV1)
7
8  module.exports = api;
```

What **api.js** file do now is, it takes all the routes from <https://www.our-page-url.com/api> (ex: <http://localhost:4000/api> in this case) and depending on which version we ask for, it will return the respective router, in this case we just have v1 router, so if we have a request like <http://localhost:4000/api/v1> we will return the module **router-api-v1.js** file.

Now following this structure we can then have more versions. Let's say we now are on a version 2 of our API. We will then have a folder named **v2** with all the routes and code for that version, and in the **api.js** file we will listen to request from <http://localhost:4000/api/v2> and then return the module named **router-api-v2.js** file.

So to wrap this up, **the last change we made, is at the app.js file** where we changed the `app.use('/v1', api);` (from min 5:30 of the video) to `app.use('/api', api);` (see *Image 3 for reference*) where it explains itself, it just listen to any request to the route **/api** and "send it" to the **api.js** file we just discussed.

### Image 3

A screenshot of a code editor window showing the following JavaScript code:

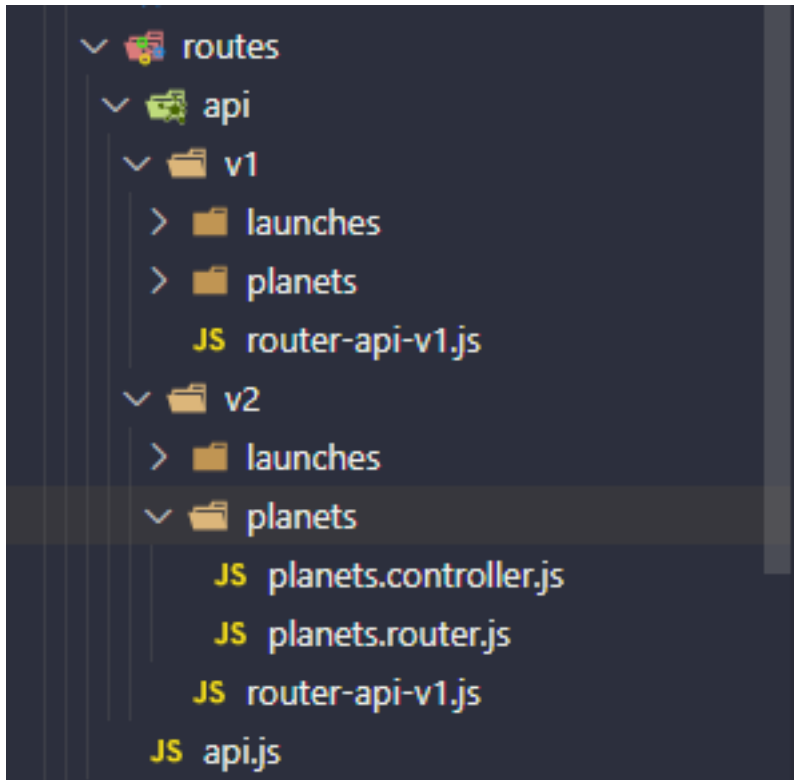
```
// Serve the client static files
app.use(express.static(path.join(__dirname, '..', 'public')));

app.use('/api', api);

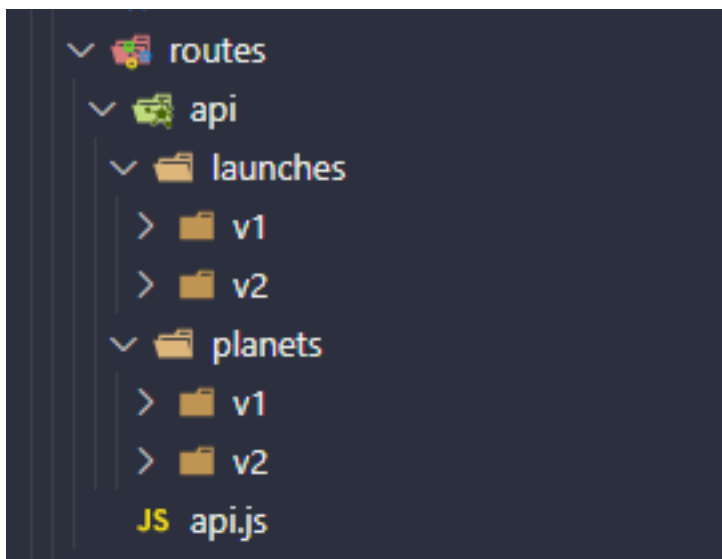
app.get('/*', (req, res) => {
  res.sendFile(path.join(__dirname, '..', 'public', 'index.html'));
});
```

Sometimes we may even want to individually version the endpoints themselves, in which case maybe we keep two versions of an endpoint maintained directly alongside each other. It's generally a good idea to have code that changes together live together, side by side.

Instead of this (what we have)



To do it this way



Now all the code is side by side.