Make the most of modern JavaScript with Fastify

@simonplend

What we're going to cover in this talk

- **What is Fastify?**
- **Promises and async / await in Node.js**
- **Node.js frameworks and async code**
  - Support matters
  - Framework support
- **Fastify and async code**
- **Hands on async code**
- **Out-of-the-box**
  - Plugins
  - Validation
  - Logging
- **Get started with Fastify**

What is Fastify?

"Fast and low overhead web framework, for Node.js"

- **Lean core** - Use plugins to add functionality.
- **Native support for async code** - Promises and `async` / `await`.
- **Two key features out-of-the-box** - Validation and logging.
- **Developer friendly** - Designed to be expressive, TypeScript support.
- **Fast (subtle clue in the name!)** - Fastest Node.js framework available today.
- **Production ready** - Being used at scale.

Matteo Collina
@matteocollina

Performance does not matter, until it absolutely does.

11:57AM · May 14, 2020 · Twitter Web App

245 Likes **46** Retweets

Promises and async / await in Node.js

Node.js has had support for:

- **Promises since v4** (released September 2015)
- **`async` / `await` since v7.6.0** (released February 2017)

Now widely used across the Node.js ecosystem.

Node.js frameworks and async code: Support matters

The Node.js framework you're using should have built-in support for async code.

If it doesn't you're at much greater risk of having unhandled promise rejections.

```
(node:168258) UnhandledPromiseRejectionWarning: Error: boom
    at getUser (file:///home/simonplend/dev/app.js:6:24)

(Use `node --trace-warnings ...` to show where the warning was created)

(node:168258) UnhandledPromiseRejectionWarning: Unhandled promise rejection. This error originated either by
throwing inside of an async function without a catch block, or by rejecting a promise which was not handled
with .catch(). To terminate the node process on unhandled promise rejection, use the CLI flag `--unhandled-
rejections=strict` (see https://nodejs.org/api/cli.html#cli_unhandled_rejections_mode). (rejection id: 2)


(node:168258) [DEP0018] DeprecationWarning: Unhandled promise rejections are deprecated. In the future,
promise rejections that are not handled will terminate the Node.js process with a non-zero exit code.
```

Node.js frameworks and async code: Support matters

Unhandled promise rejections…

- Can cause **memory leaks** in your application
- Will throw an error and **crash your application** from Node v15.0.0

Node.js frameworks and async code: Framework support

❌ **Express** - No native support.

❌ **Restify** - No native support yet. Support merged mid-2020, slated for a v9 release.

✅ **Koa** - Native support.

✅ **hapi** - Native support.

✅ **Sails.js (full stack)** - Native support.

✅ **Nest (full stack)** - Native support, but likely patching Express. Can use with Fastify.

Fastify and async code

- Fastify natively handles promises and supports `async` / `await` 🎉
- Routes will catch uncaught rejected promises for you.
- Allows you to write asynchronous code safely.
- Lets you do neat things (send return value as response body):

```
app.get("/user/:id", async (request) => await getUser(request.params.id));
```

Hands on async code

Let's look at how Express and Fastify behave with async code 👀

Out-of-the-box

Some of the other things which Fastify gives you:

- **Plugins**
- **Validation**
- **Logging**

Out-of-the-box: Plugins

Fundamental concept: Everything is a plugin.

- Plugins have their own **scope**.
- Plugins can contain **routes** or customise core Fastify objects with **"decorators"**.
- To use plugins you **register** them:

```
app.register(async (app, options) => {

  app.decorate("yolo", () => {
      return { yo: "lo" };
  });

  app.get("/yolo", async (request, reply) => {
      reply.send(app.yolo());
  });

});
```

Out-of-the-box: Validation

- **Request validation** - Uses Ajv (Another JSON schema validator).
- **Define validation rules with JSON Schema** - Compiled to ES6 code, helps make them very fast.

```
const schema = {
  body: {
      type: "object",
      required: ["first_name"],
      properties: {
          first_name: { type: "string", minLength: 1 }
      }
  }
};

app.post("/user", { schema }, async (request, reply) => {
  reply.send(request.body);
});
```

```
{
  "statusCode": 400,
  "error": "Bad Request",
  "message": "body should have required property 'first_name'"
}
```

Out-of-the-box: Logging

- **Logging often causes performance issues** - e.g. Serializing and transporting data elsewhere.
- **Logging fully integrated** - No need to spend time choosing and integrating a logger, Fastify uses a fast and flexible logger: `pino`

```
> node examples/fastify-app.js
[1614255468557] INFO (170038 on simon-xps13): Server listening at http://127.0.0.1:3000
[1614255472287] INFO (170038 on simon-xps13): incoming request
    req: {
      "method": "GET",
      "url": "/user/abc1",
      "hostname": "localhost:3000",
      "remoteAddress": "127.0.0.1",
      "remotePort": 40386
    }
    reqId: 1
[1614255472315] INFO (170038 on simon-xps13): request completed
    res: {
      "statusCode": 200
    }
    responseTime: 20.992334991693497
    reqId: 1
```

Get started with Fastify

- **Extensive documentation**
  - fastify.io/docs/latest/
- **Rich ecosystem of plugins**
  - fastify.io/ecosystem/
- **Example application** - Showing core Fastify concepts, best practices and recommendations.
  - github.com/delvedor/fastify-example
- **Plugin to help ease migration from Express**
  - fastify-express
- **Community Discord server**
  - discord.gg/D3FZYPy

THE END

Thank you for watching

I blog about Node.js at simonplend.com

I send out a weekly-ish newsletter
to help you build better Node.js applications.

Follow me on Twitter @simonplend