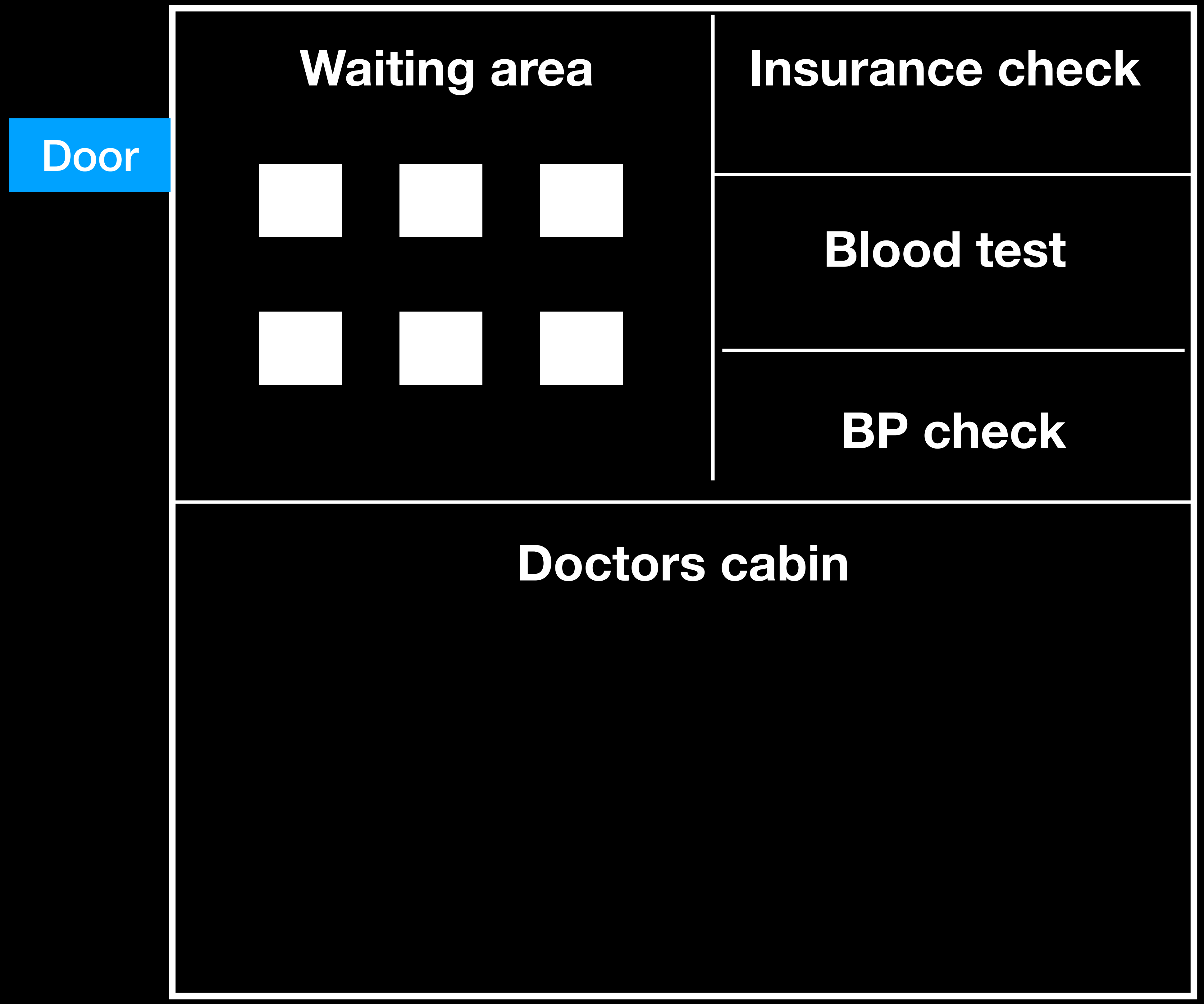
3.1
Middlewares, authentication, global catches, zod

Hospital (covid)



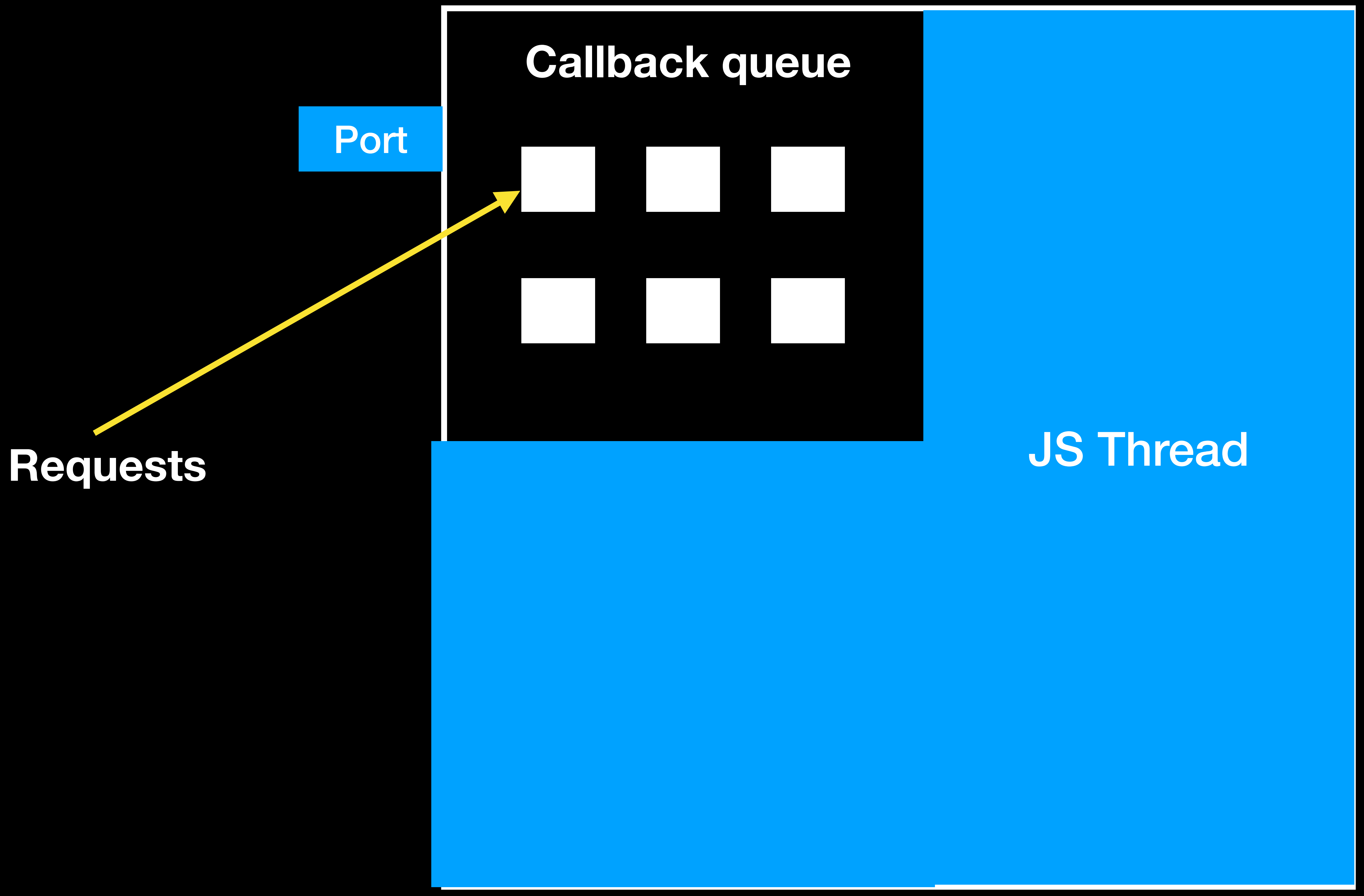
Before you reach the doctor

1. Your adhar/insurance info is taken. Only if you have insurance you proceed
2. Blood test is done, only if no STD does use proceed
3. BP is checked, only if BP is reasonable user proceeds

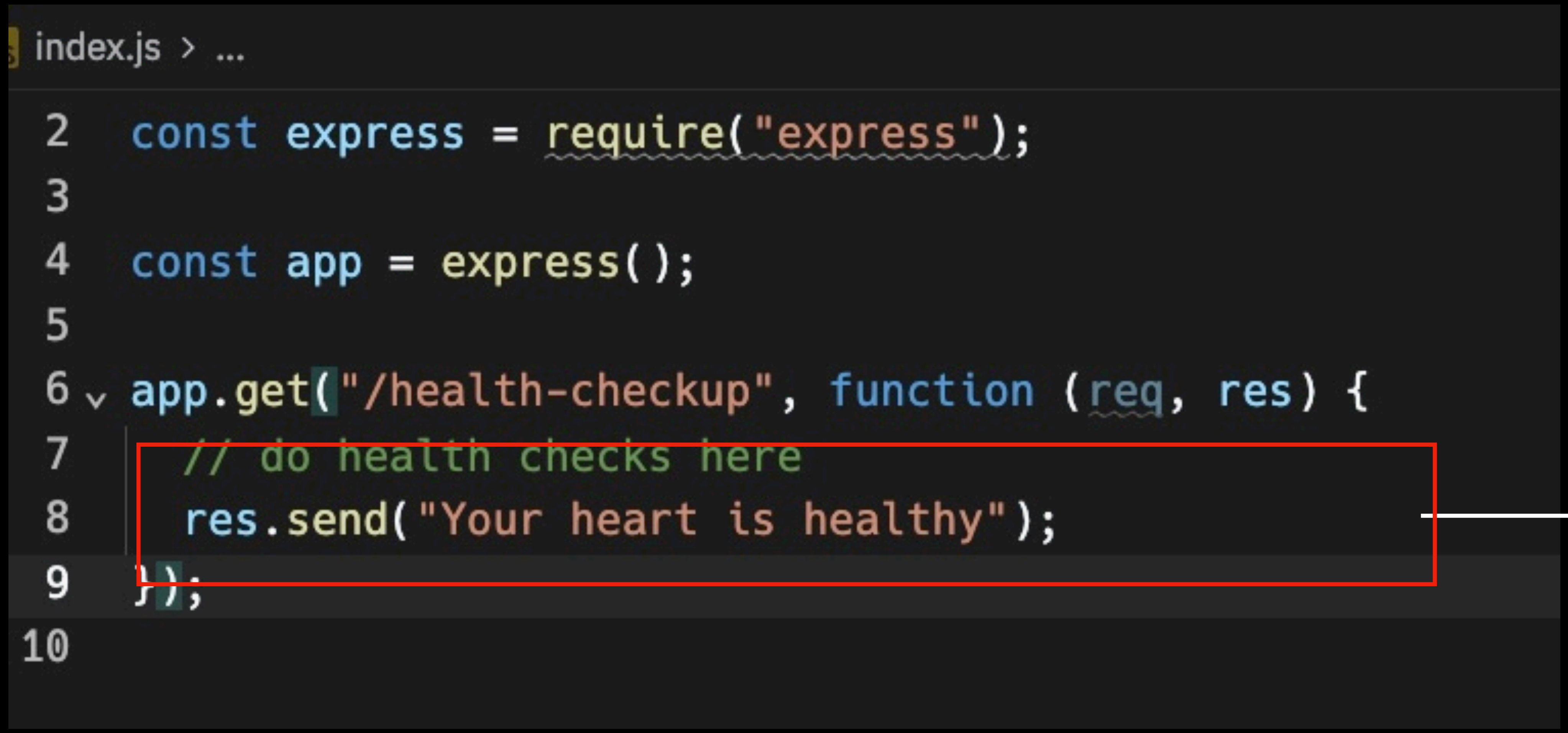
Hospital (covid)

Before you reach the doctor

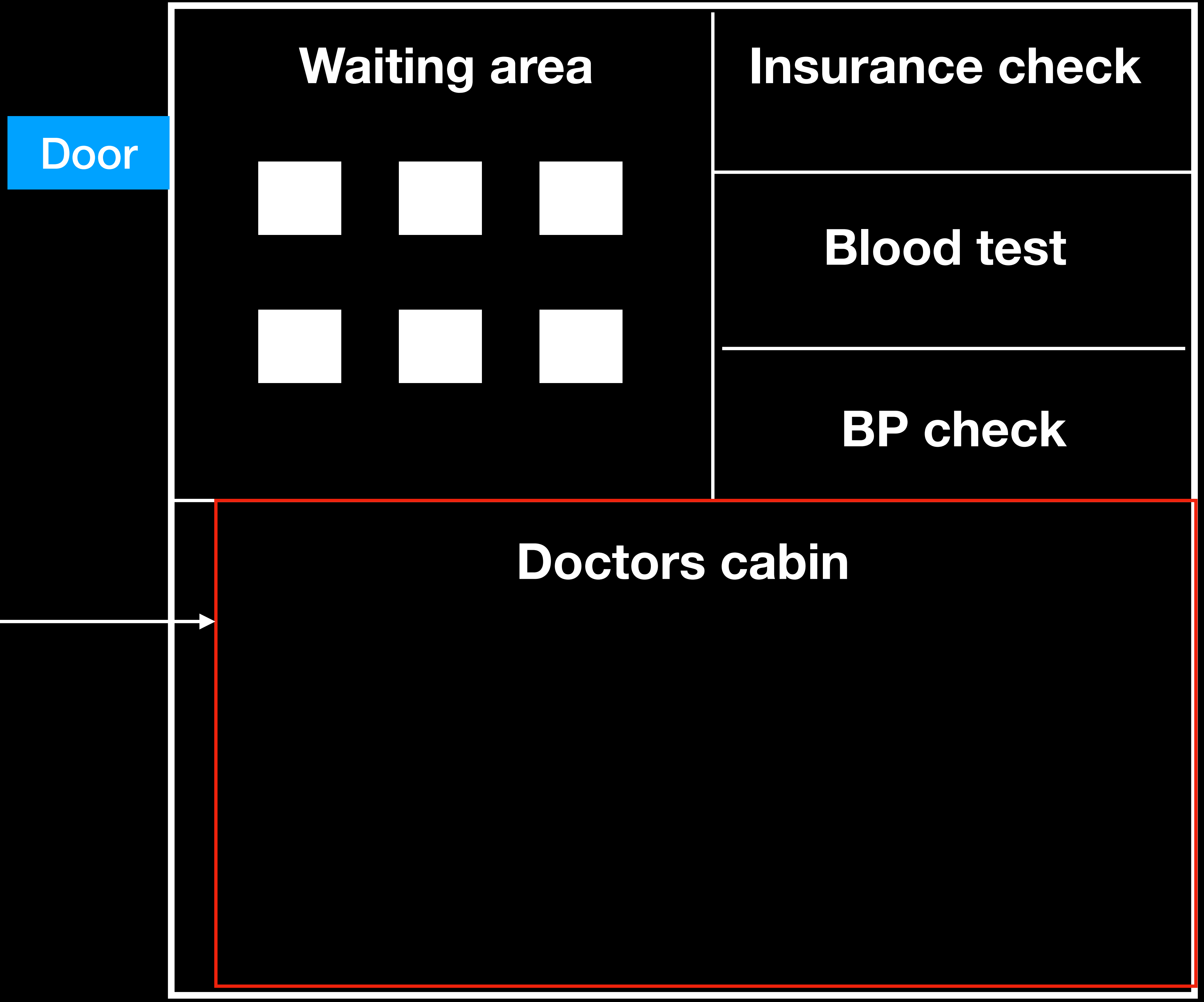
1. Your adhar/insurance info is taken
2. Blood test is done
3. BP is checked



Equivalent code



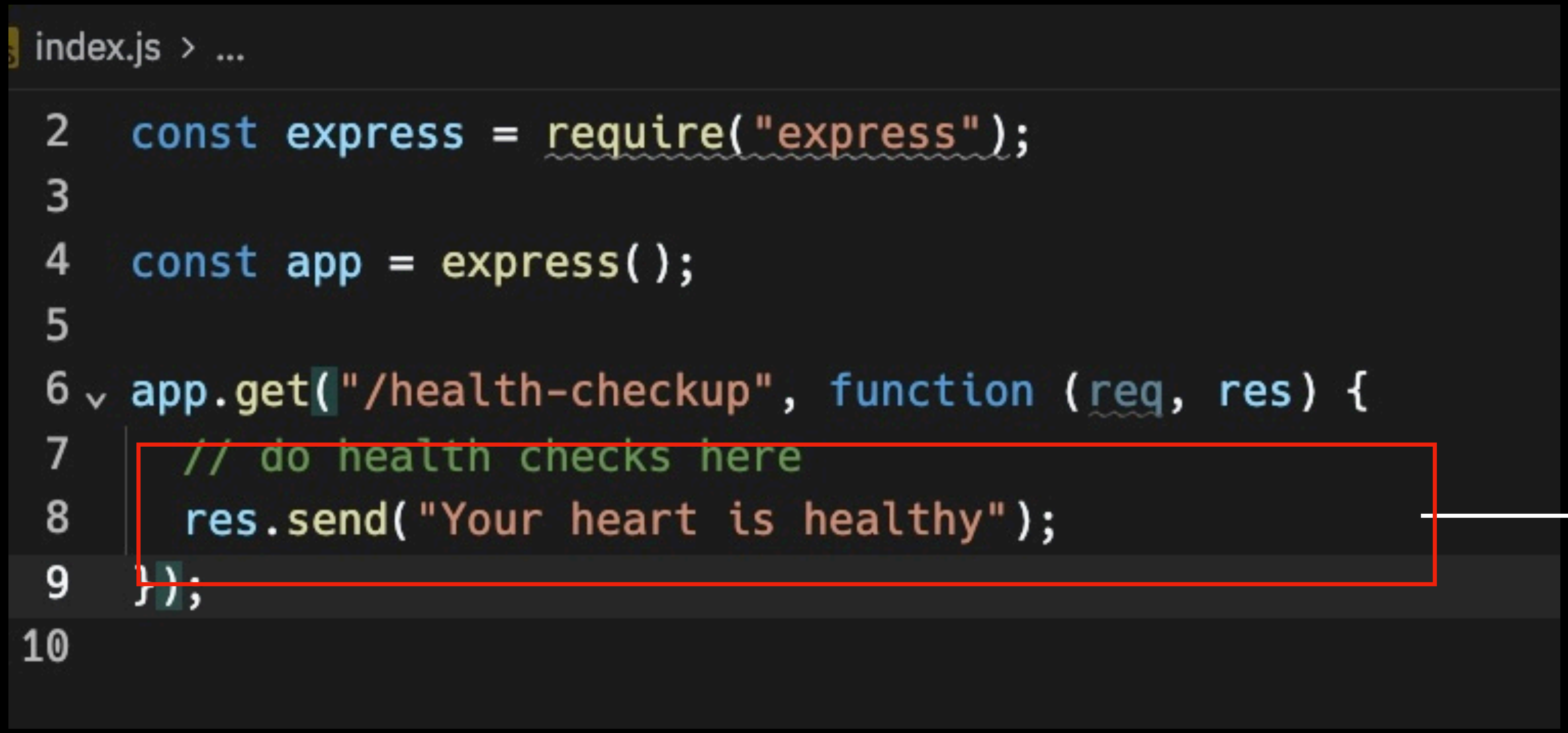
```
const express = require("express");

const app = express();

app.get("/health-checkup", function (req, res) {
  // do health checks here
  res.send("Your heart is healthy");
});
```



Questions

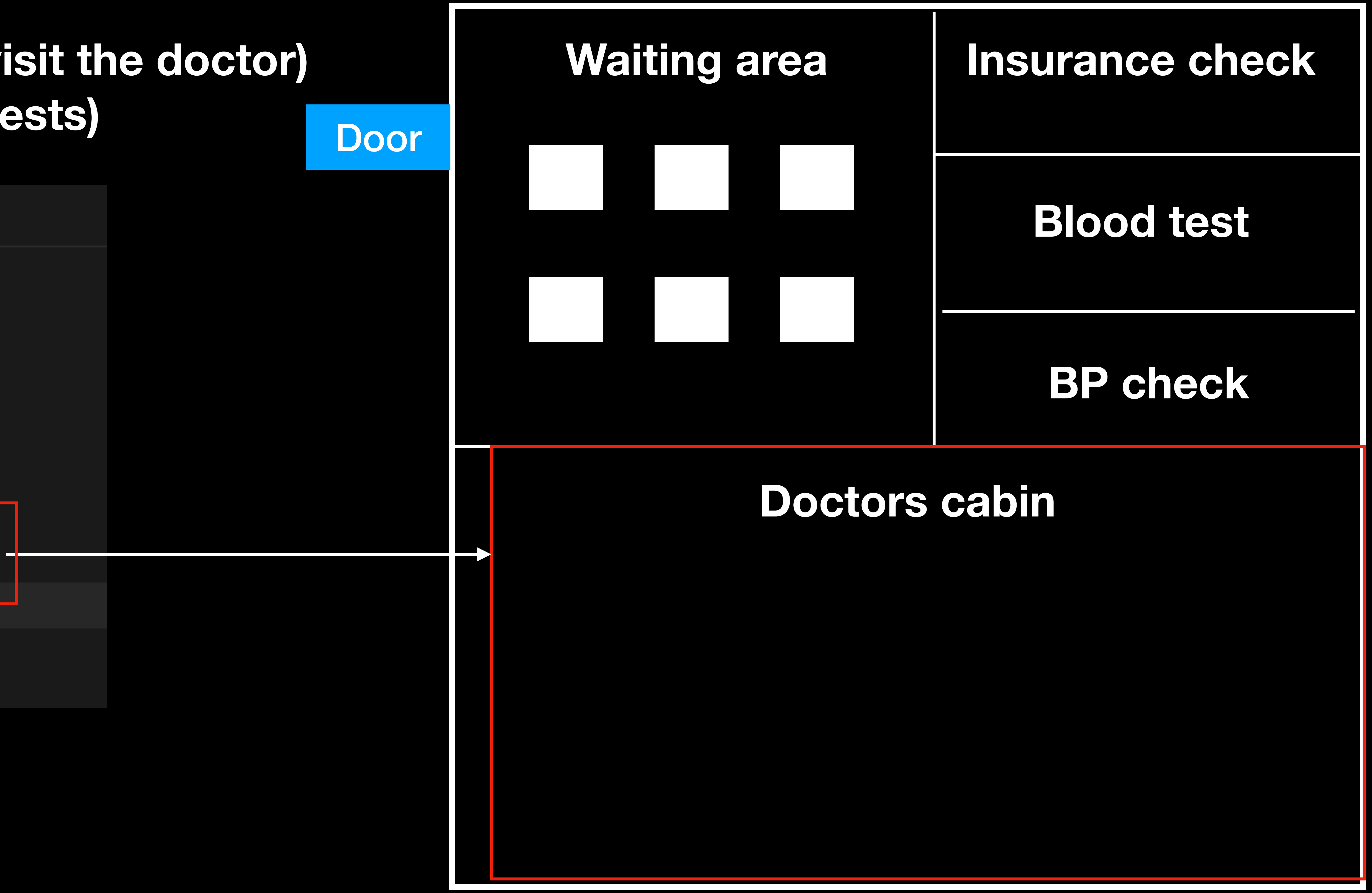
How do you do

1. Auth checks? (Does this user have funds to visit the doctor)
2. Ensure input by the user is valid (BP / blood tests)



index.js > ...

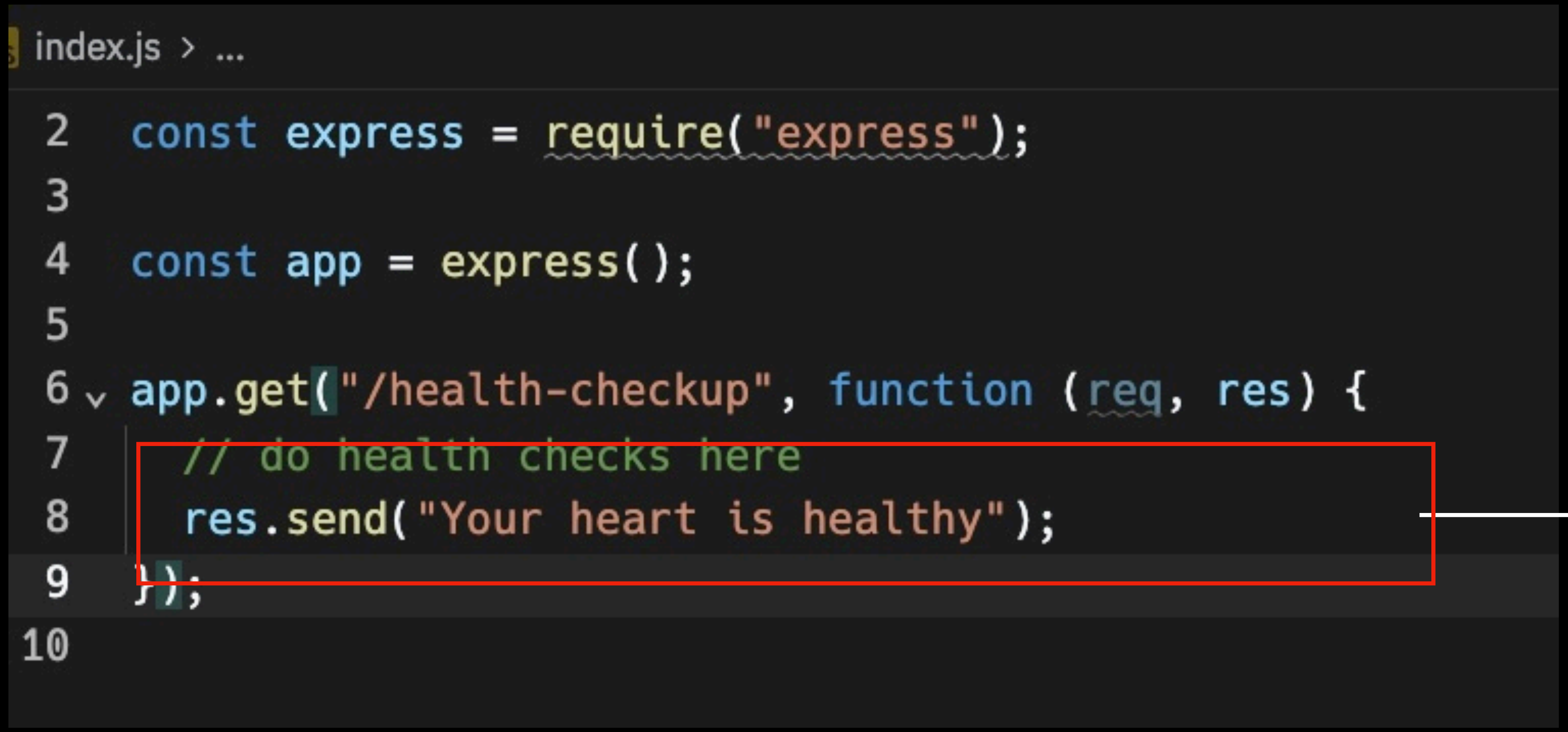
```
const express = require("express");

const app = express();

app.get("/health-checkup", function (req, res) {
  // do health checks here
  res.send("Your heart is healthy");
});
```



Answer - Middlewares

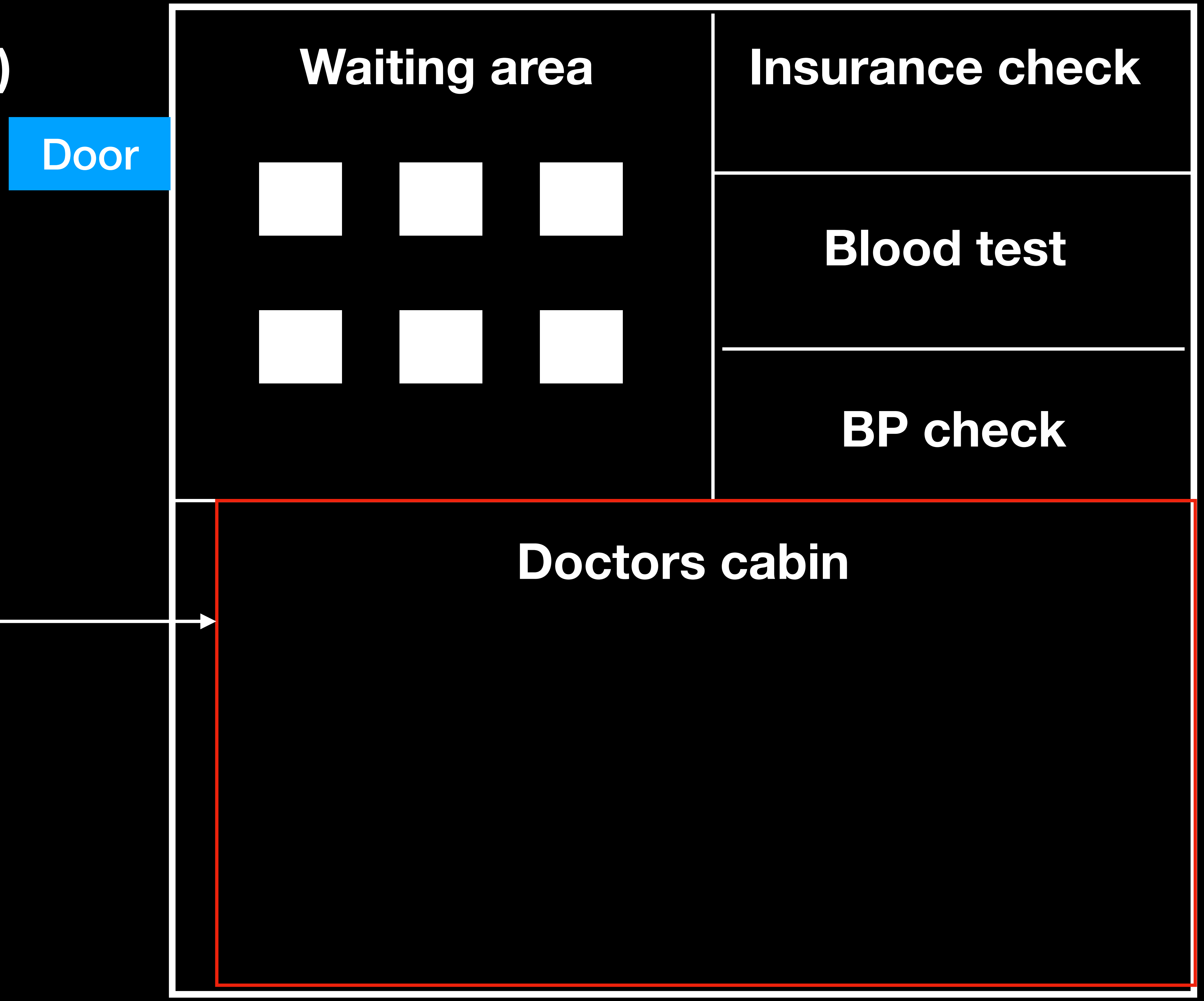
How do you do

1. Auth checks? (Does this user have funds to visit the doctor)
2. Ensure input by the user is valid (BP / blood tests)



index.js > ...

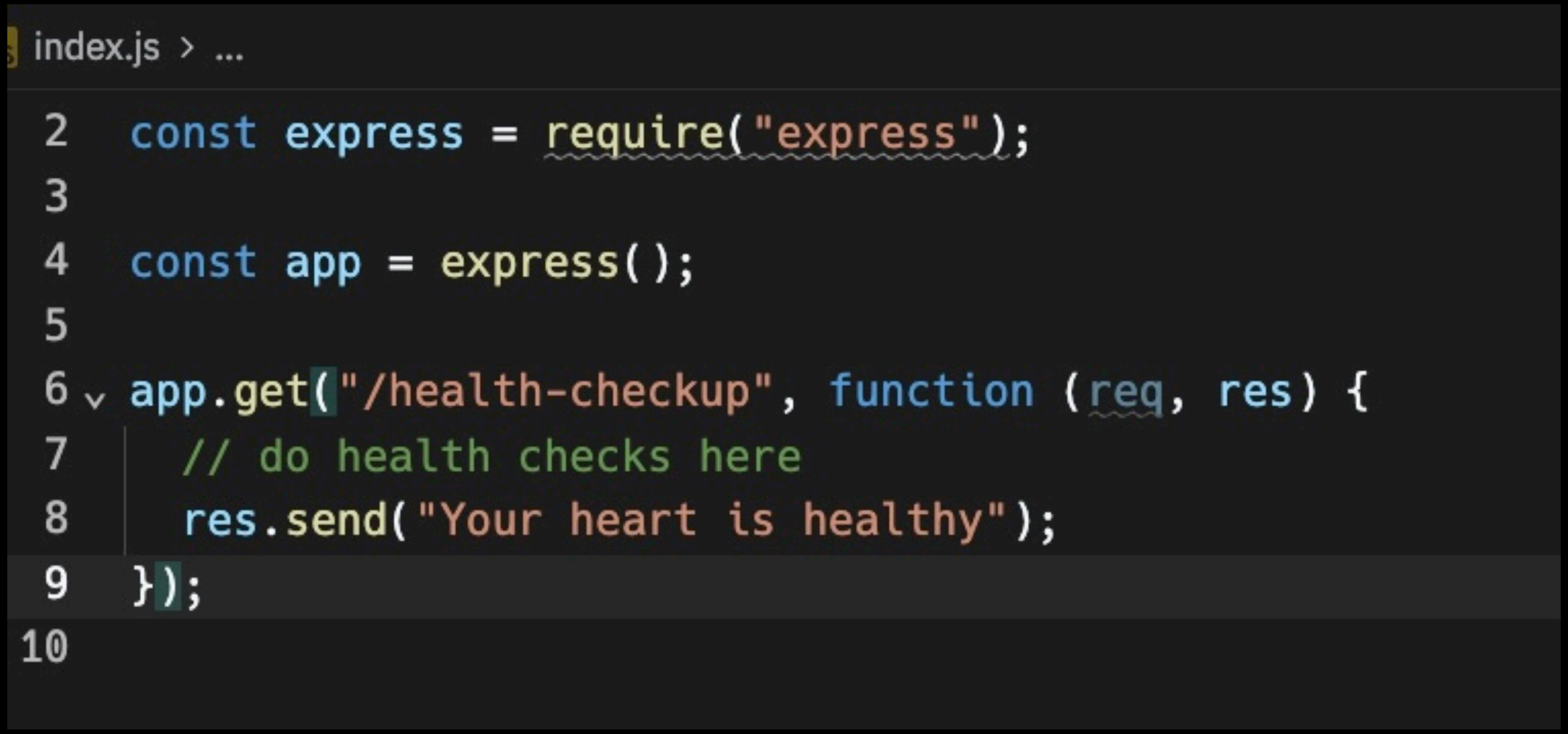
```
const express = require("express");

const app = express();

app.get("/health-checkup", function (req, res) {
  // do health checks here
  res.send("Your heart is healthy");
});
```



Answer - Middlewares

Before we proceed, lets add constraints to our route

1. **User needs to send a kidneyId as a query param which should be a number from 1-2 (humans only has 2 kidneys)**
2. **User should send a username and password in headers**



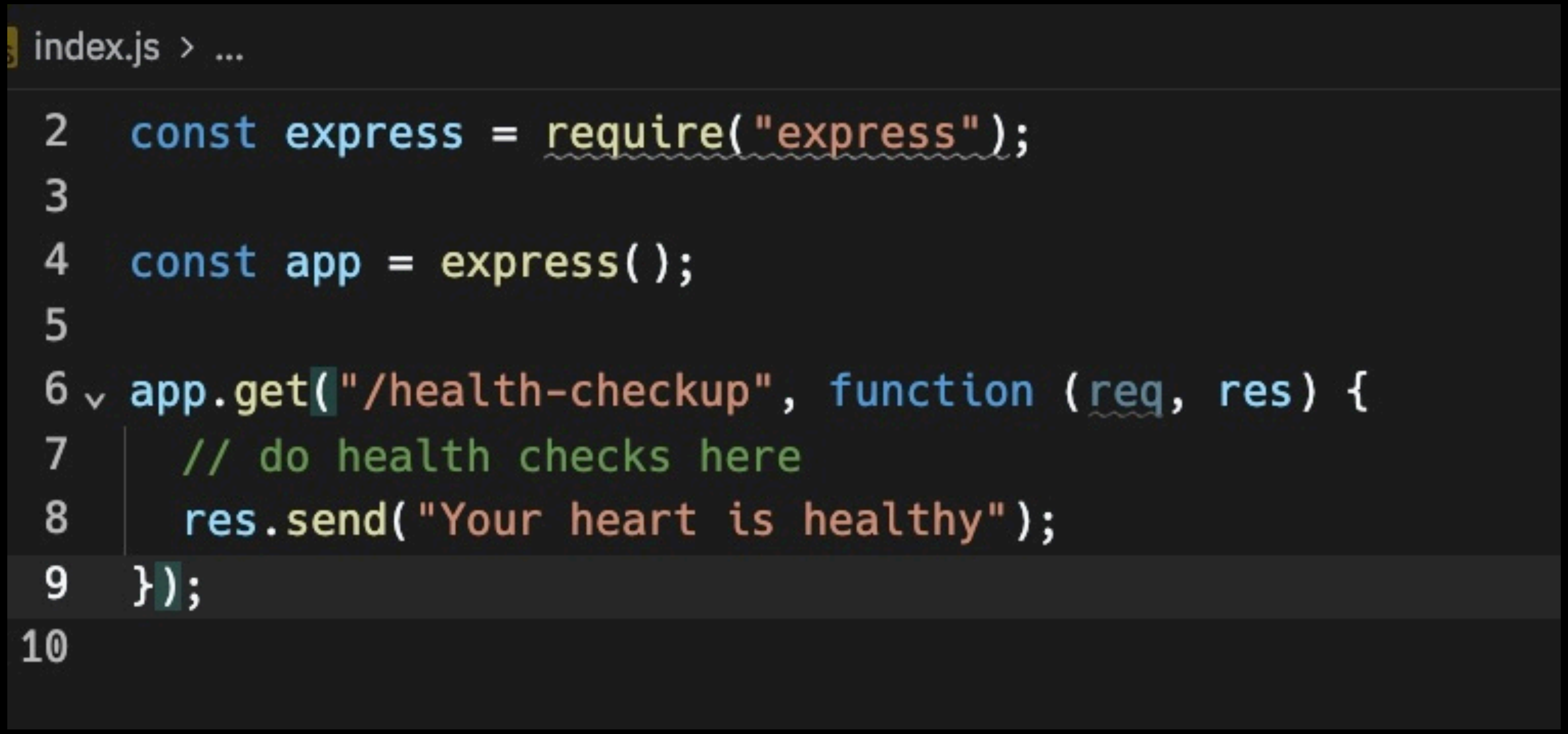
index.js > ...

```javascript
const express = require("express");

const app = express();

app.get("/health-checkup", function (req, res) {
  // do health checks here
  res.send("Your heart is healthy");
});

```

Answer - Middlewares

Before we proceed, lets add constraints to our route

1. **User needs to send a kidneyId as a query param which should be a number from 1-2 (humans only has 2 kidneys)**
2. **User should send a username and password in headers**



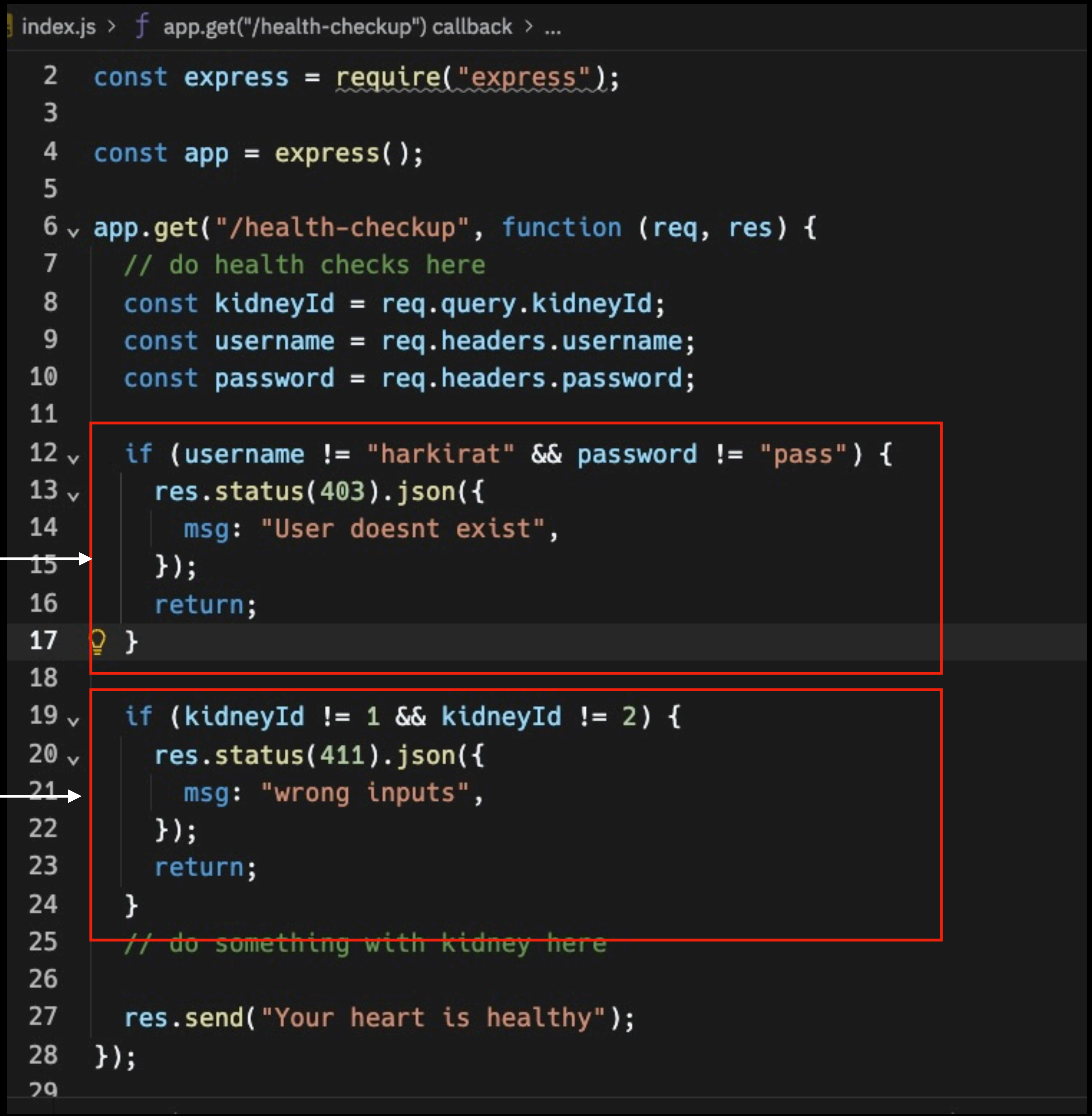
index.js > ...

```javascript
const express = require("express");

const app = express();

app.get("/health-checkup", function (req, res) {
  // do health checks here
  res.send("Your heart is healthy");
});

```

Answer - Middlewares

Before we proceed, lets add constraints to our route

1. **User needs to send a kidneyId as a query param which should be a number from 1-2 (humans only has 2 kidneys)**
2. **User should send a username and password in headers**



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

const app = express();

app.get("/health-checkup", function (req, res) {
  // do health checks here
  const kidneyId = req.query.kidneyId;
  const username = req.headers.username;
  const password = req.headers.password;

  if (username != "harkirat" && password != "pass") {
    res.status(403).json({
      msg: "User doesnt exist",
    });
    return;
  }

  if (kidneyId != 1 && kidneyId != 2) {
    res.status(411).json({
      msg: "wrong inputs",
    });
    return;
  }
  // do something with kidney here

  res.send("Your heart is healthy");
});
```

Username checks

Input validation

Answer - Middlewares

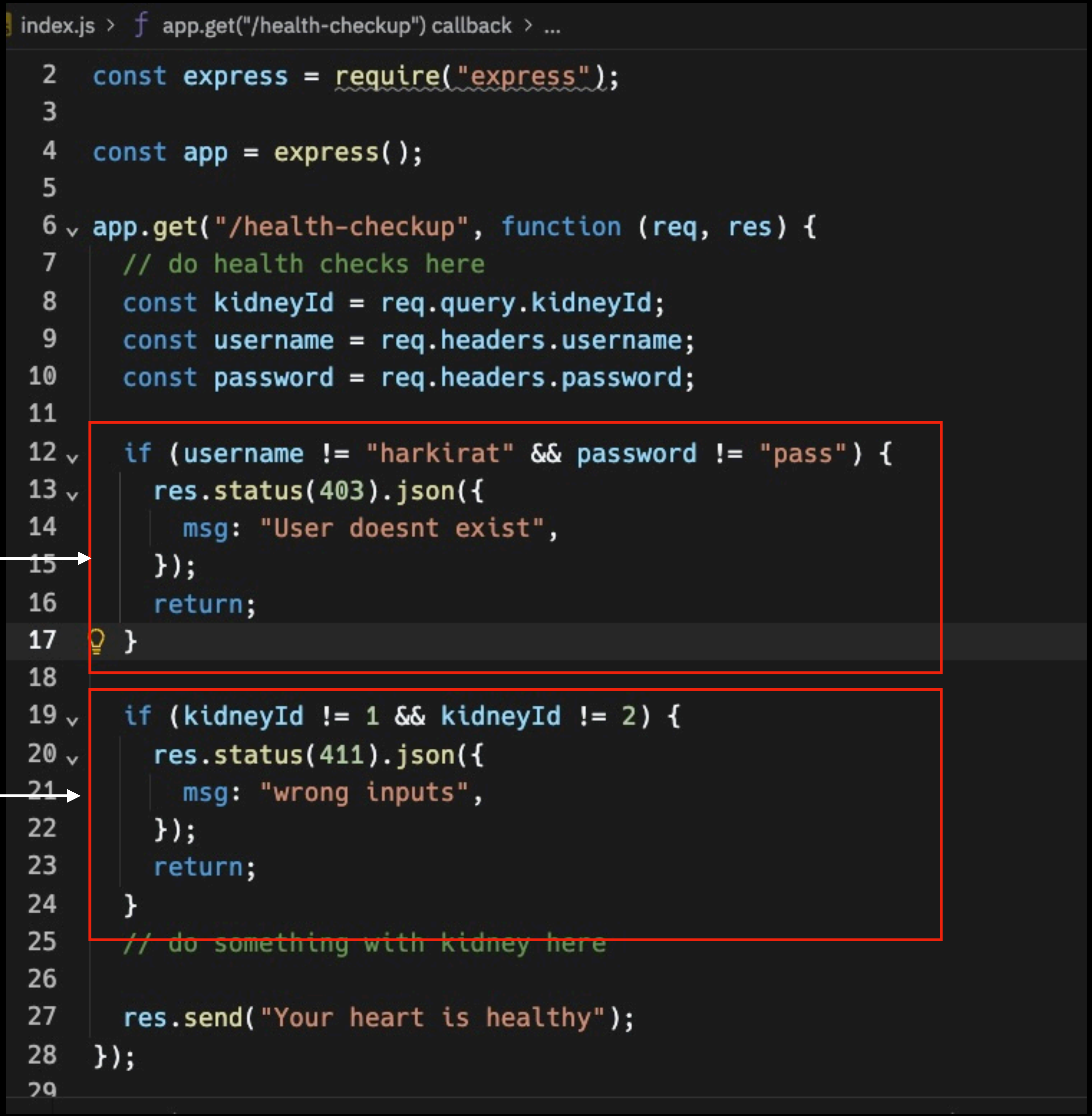
Before we proceed, lets add constraints to our route

1. **User needs to send a kidneyId as a query param which should be a number from 1-2 (humans only has 2 kidneys)**
2. **User should send a username and password in headers**

What if I tell you to introduce another route that does
Kidney replacement
Inputs need to be the same

Username checks

Input validation



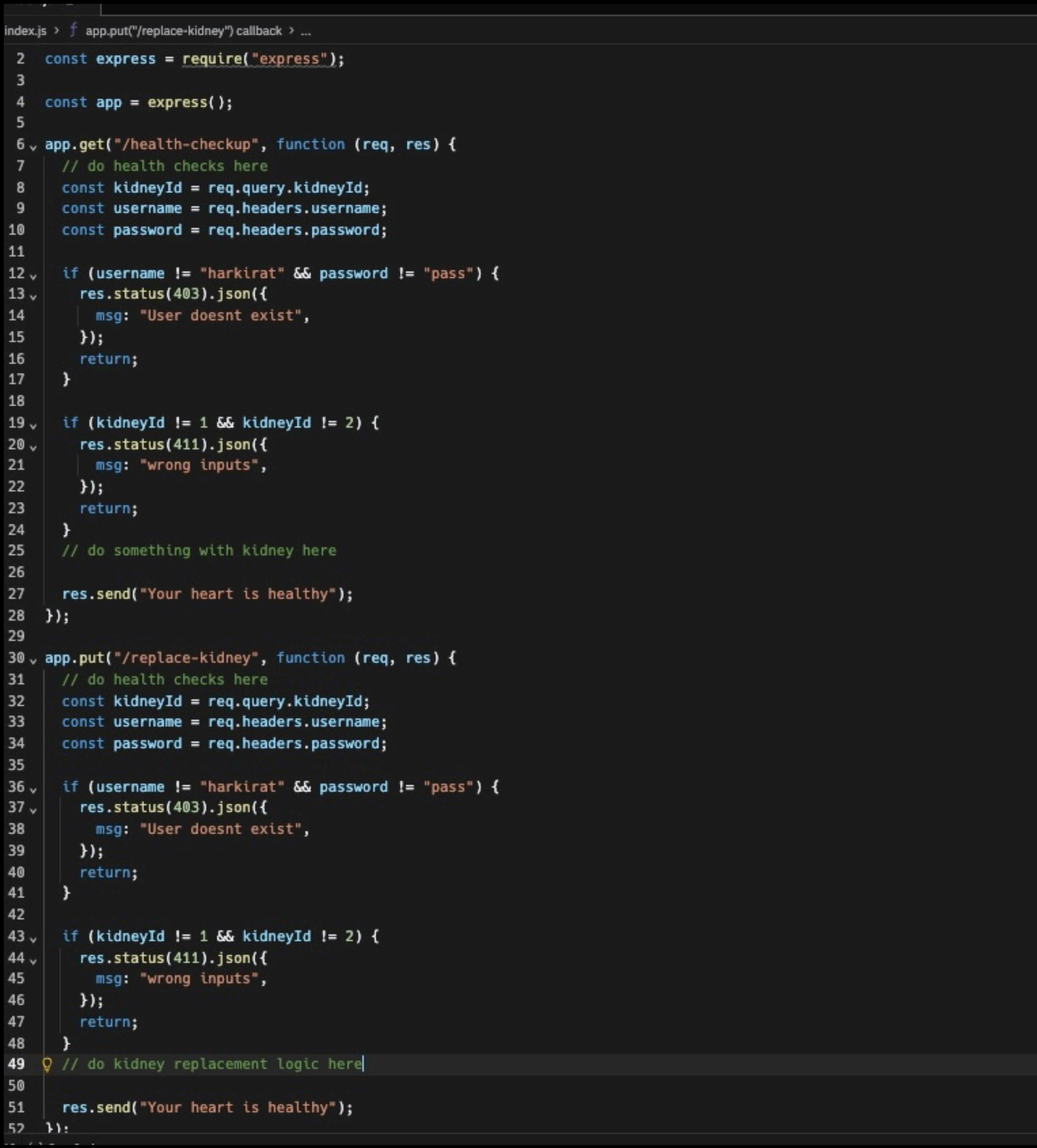
index.js > ƒ app.get("/health-checkup") callback > ...

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

const app = express();

app.get("/health-checkup", function (req, res) {
  // do health checks here
  const kidneyId = req.query.kidneyId;
  const username = req.headers.username;
  const password = req.headers.password;

  if (username != "harkirat" && password != "pass") {
    res.status(403).json({
      msg: "User doesnt exist",
    });
    return;
  }

  if (kidneyId != 1 && kidneyId != 2) {
    res.status(411).json({
      msg: "wrong inputs",
    });
    return;
  }
  // do something with kidney here

  res.send("Your heart is healthy");
});
```

Middlewares

What if I tell you to introduce another route that does
Kidney replacement
Inputs need to be the same

Ugly solution - Create a new route, repeat code

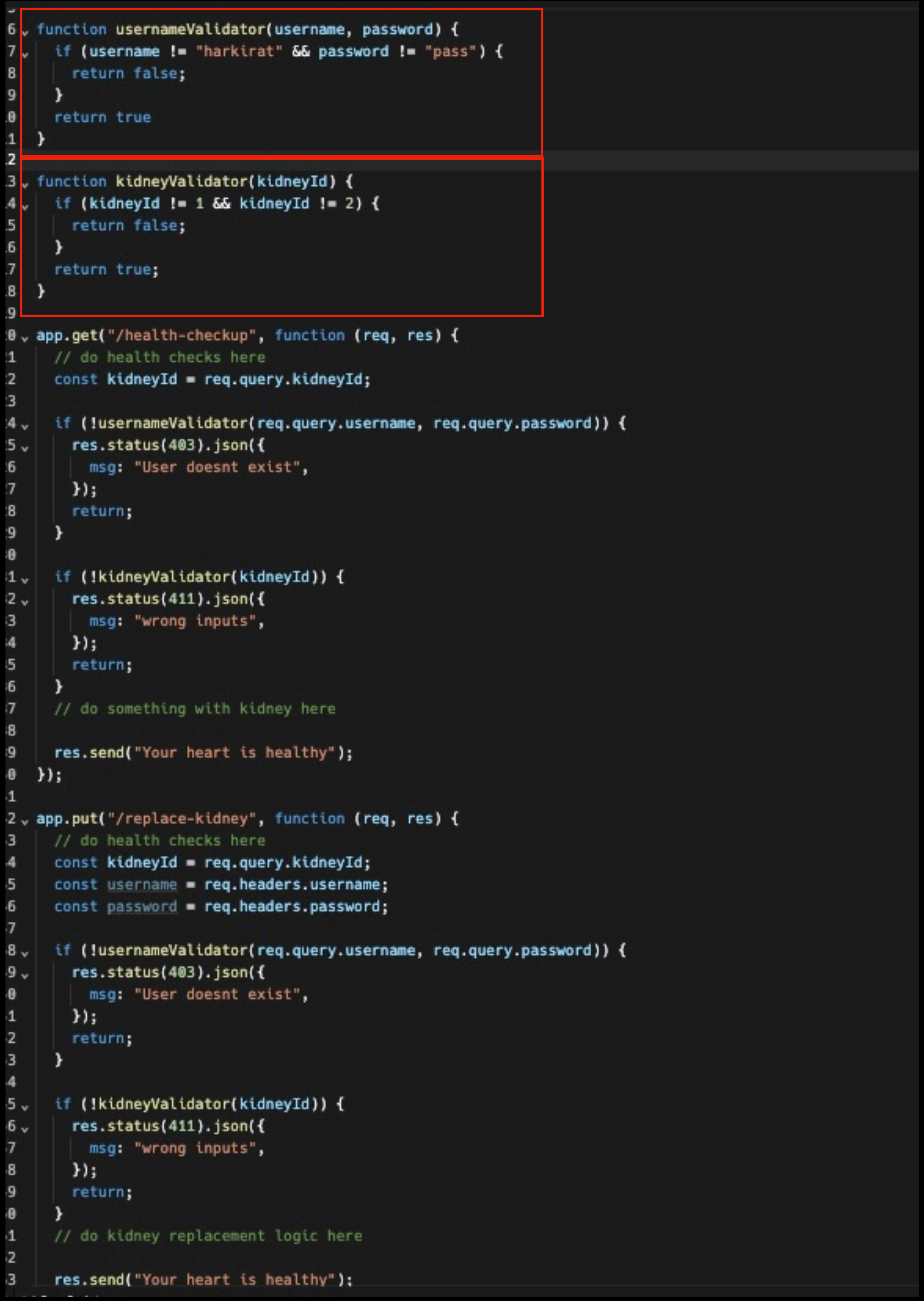

index.js > app.put("/replace-kidney") callback > ...

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

const app = express();

app.get("/health-checkup", function (req, res) {
  // do health checks here
  const kidneyId = req.query.kidneyId;
  const username = req.headers.username;
  const password = req.headers.password;

  if (username != "harkirat" && password != "pass") {
    res.status(403).json({
      msg: "User doesnt exist",
    });
    return;
  }

  if (kidneyId != 1 && kidneyId != 2) {
    res.status(411).json({
      msg: "wrong inputs",
    });
    return;
  }
  // do something with kidney here

  res.send("Your heart is healthy");
});

app.put("/replace-kidney", function (req, res) {
  // do health checks here
  const kidneyId = req.query.kidneyId;
  const username = req.headers.username;
  const password = req.headers.password;

  if (username != "harkirat" && password != "pass") {
    res.status(403).json({
      msg: "User doesnt exist",
    });
    return;
  }

  if (kidneyId != 1 && kidneyId != 2) {
    res.status(411).json({
      msg: "wrong inputs",
    });
    return;
  }
  // do kidney replacement logic here

  res.send("Your heart is healthy");
});
```

Middlewares

What if I tell you to introduce another route that does Kidney replacement
Inputs need to be the same

Slightly better solution - Create wrapper fns

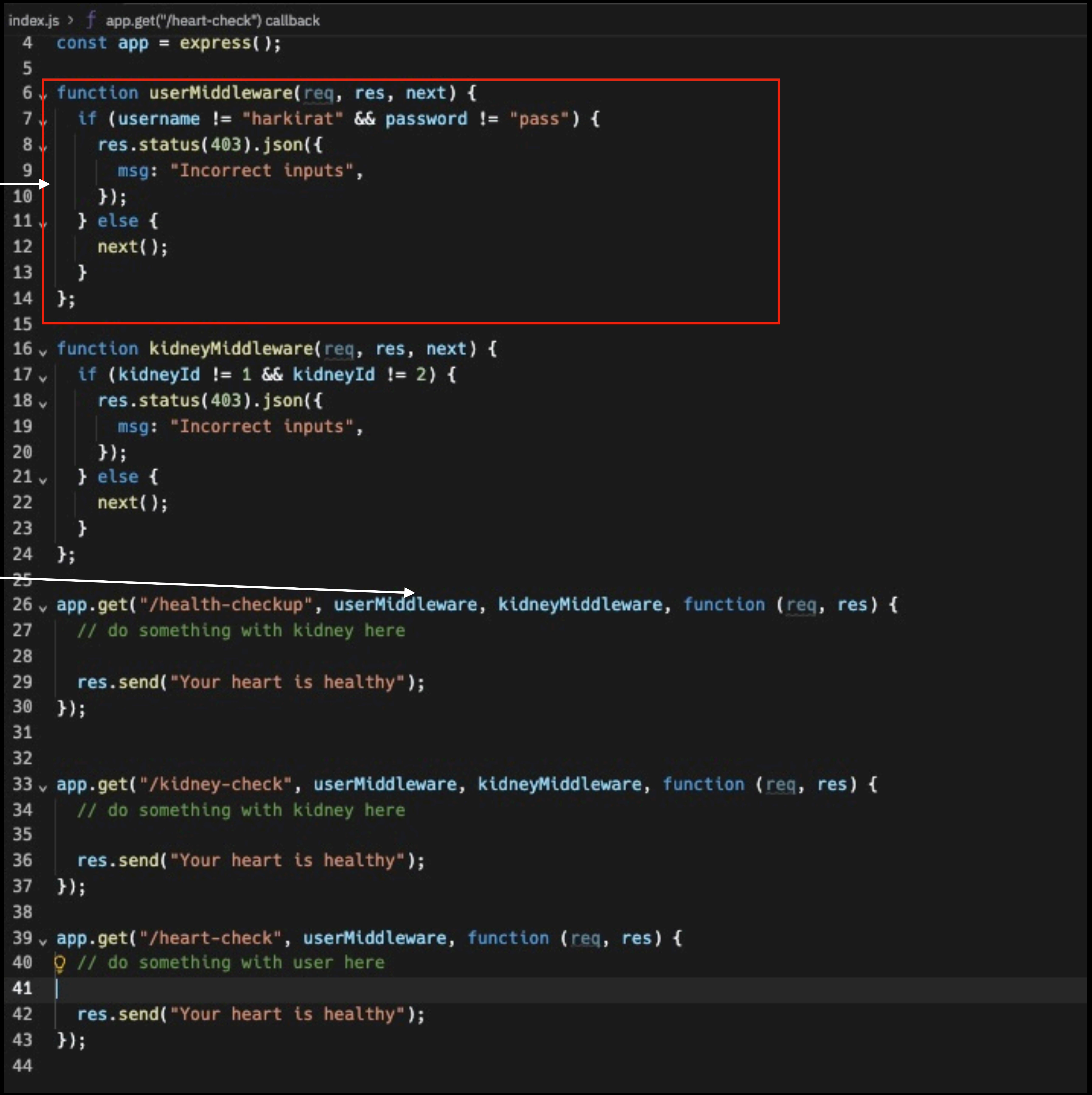


```javascript
function usernameValidator(username, password) {
  if (username != "harkirat" && password != "pass") {
    return false;
  }
  return true
}

function kidneyValidator(kidneyId) {
  if (kidneyId != 1 && kidneyId != 2) {
    return false;
  }
  return true;
}

app.get("/health-checkup", function (req, res) {
  // do health checks here
  const kidneyId = req.query.kidneyId;

  if (!usernameValidator(req.query.username, req.query.password)) {
    res.status(403).json({
      msg: "User doesnt exist",
    });
    return;
  }

  if (!kidneyValidator(kidneyId)) {
    res.status(411).json({
      msg: "wrong inputs",
    });
    return;
  }
  // do something with kidney here

  res.send("Your heart is healthy");
});

app.put("/replace-kidney", function (req, res) {
  // do health checks here
  const kidneyId = req.query.kidneyId;
  const username = req.headers.username;
  const password = req.headers.password;

  if (!usernameValidator(req.query.username, req.query.password)) {
    res.status(403).json({
      msg: "User doesnt exist",
    });
    return;
  }

  if (!kidneyValidator(kidneyId)) {
    res.status(411).json({
      msg: "wrong inputs",
    });
    return;
  }
  // do kidney replacement logic here

  res.send("Your heart is healthy");
```

Middlewares

Defining middleware (just another fn)

Using the middleware



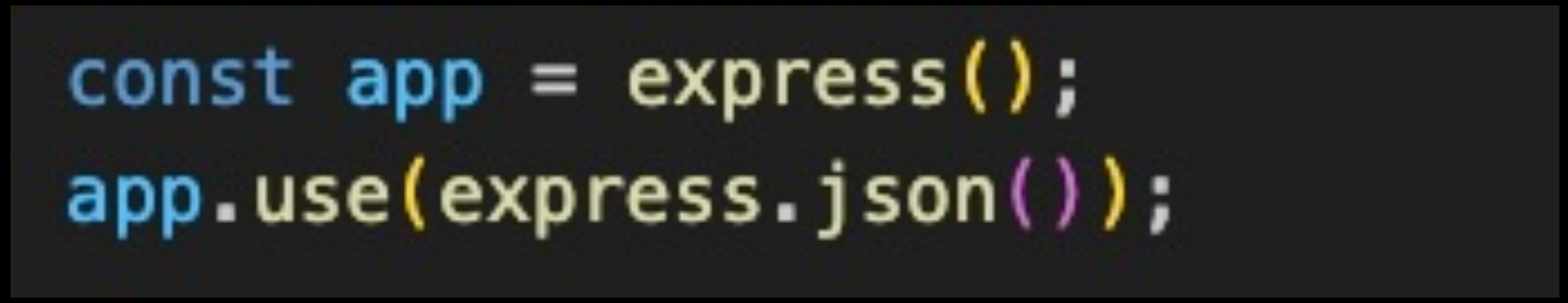
index.js > ƒ app.get("/heart-check") callback

```
const app = express();

function userMiddleware(req, res, next) {
  if (username != "harkirat" && password != "pass") {
    res.status(403).json({
      msg: "Incorrect inputs",
    });
  } else {
    next();
  }
};

function kidneyMiddleware(req, res, next) {
  if (kidneyId != 1 && kidneyId != 2) {
    res.status(403).json({
      msg: "Incorrect inputs",
    });
  } else {
    next();
  }
};

app.get("/health-checkup", userMiddleware, kidneyMiddleware, function (req, res) {
  // do something with kidney here

  res.send("Your heart is healthy");
});


app.get("/kidney-check", userMiddleware, kidneyMiddleware, function (req, res) {
  // do something with kidney here

  res.send("Your heart is healthy");
});

app.get("/heart-check", userMiddleware, function (req, res) {
  // do something with user here

  res.send("Your heart is healthy");
});

```

What if I tell you to introduce another route that does Kidney replacement
Inputs need to be the same

Best solution - middleware

Last thing in middleware - app.use



```
const app = express();
app.use(express.json());
```

Other use cases of middleware (assignment) =

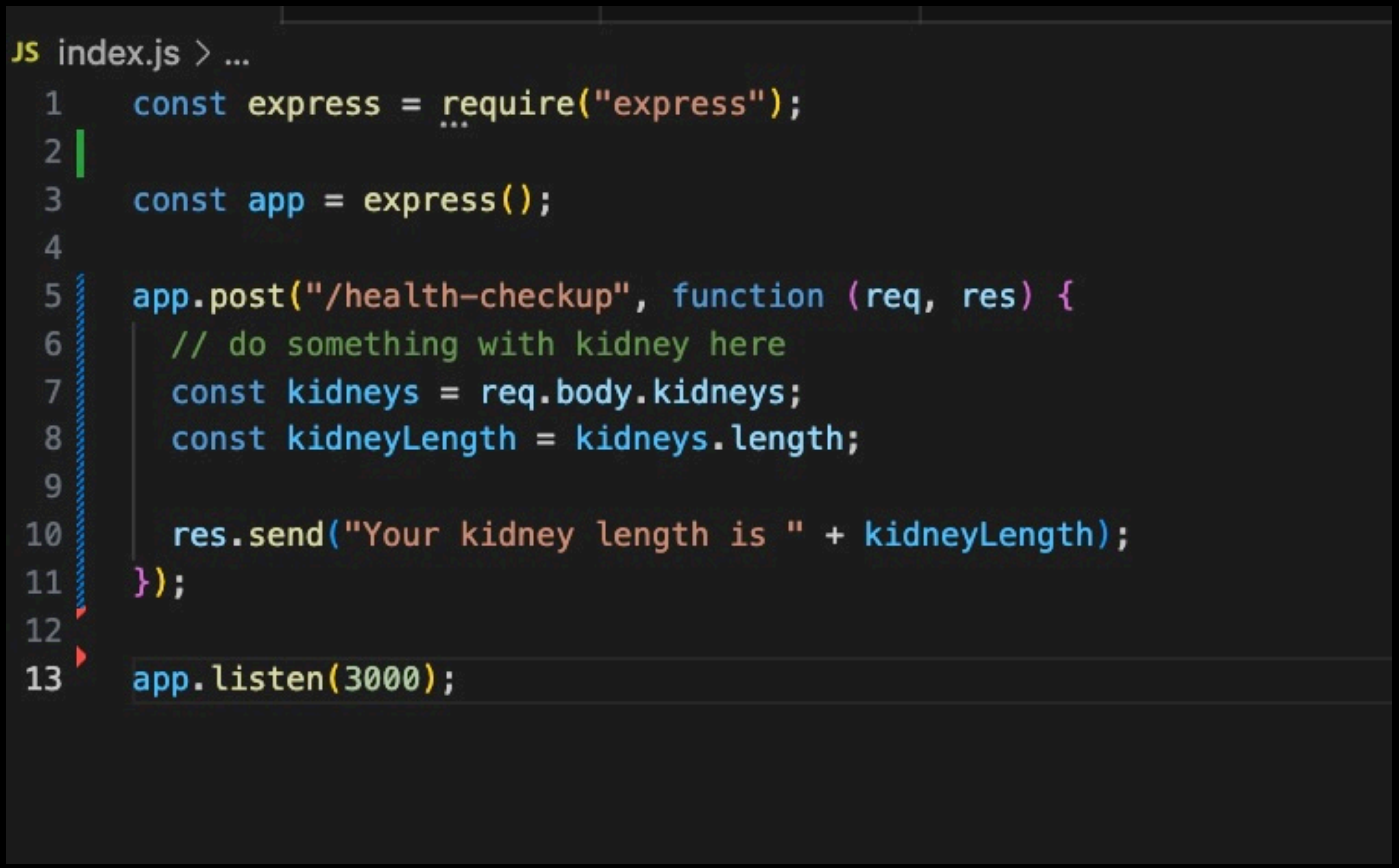
1. Count the number of requests
2. Find the average time your server is taking to handle requests

Take a breather

We understand middlewares

Why do you need input validation?
Lets see with an example

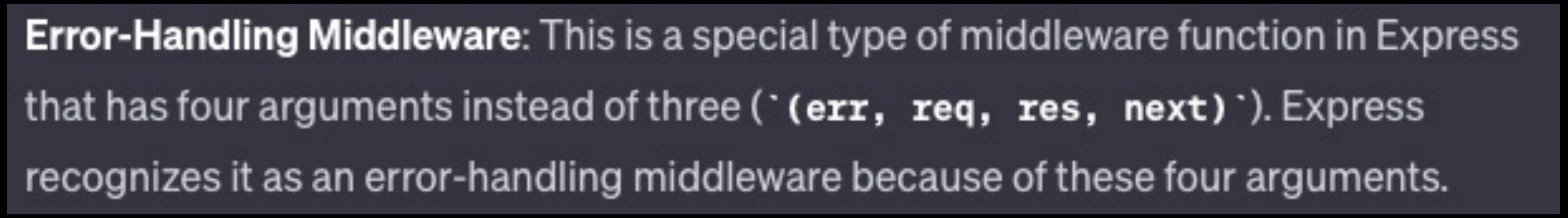
What if the user sends the wrong body?

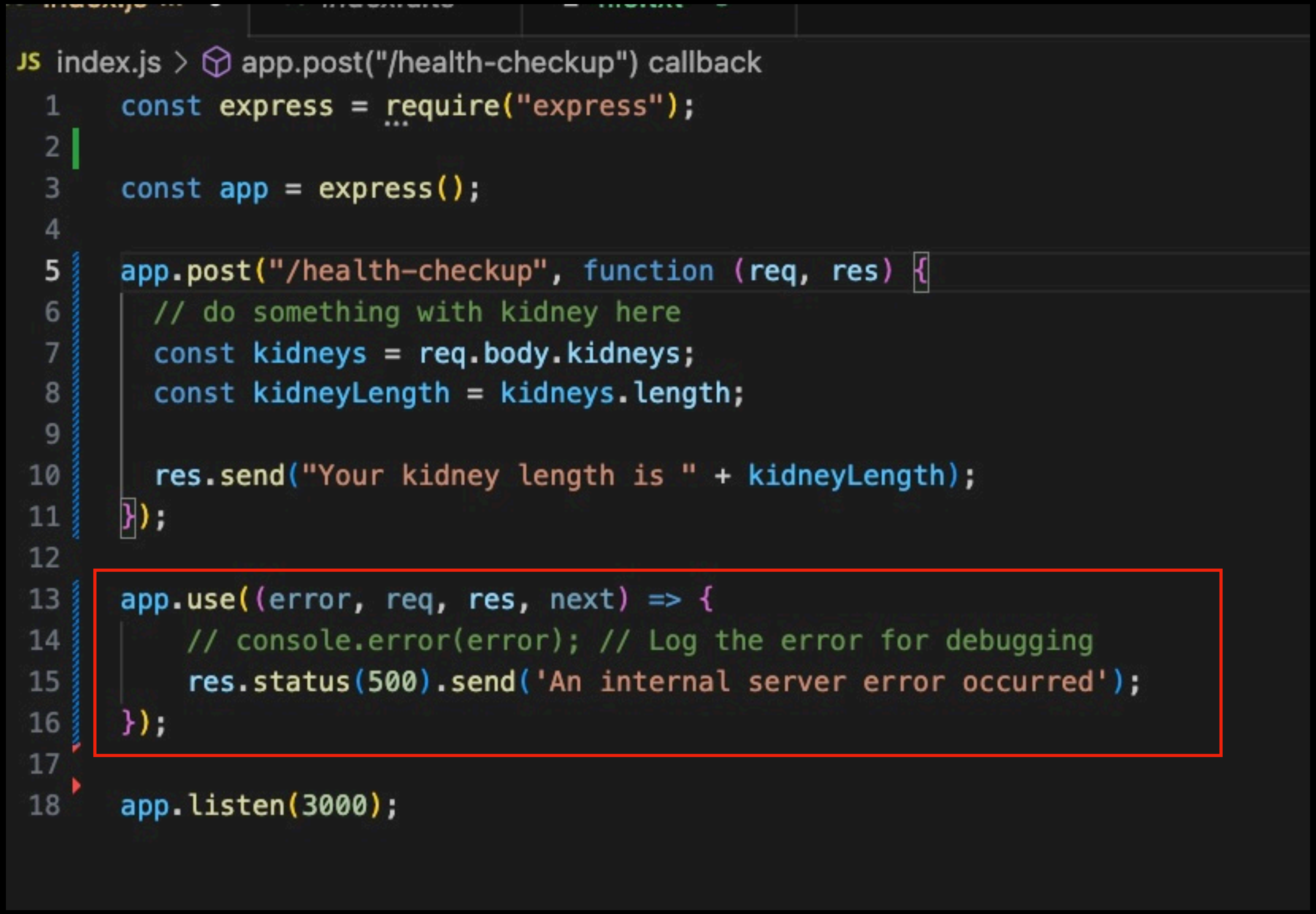

JS index.js > ...

```javascript
const express = require("express");

const app = express();

app.post("/health-checkup", function (req, res) {
  // do something with kidney here
  const kidneys = req.body.kidneys;
  const kidneyLength = kidneys.length;

  res.send("Your kidney length is " + kidneyLength);
});

app.listen(3000);
```

Why do you need input validation?
Lets see with an example

Global catches help you give the user a Better error message


Error-Handling Middleware: This is a special type of middleware function in Express that has four arguments instead of three (`(err, req, res, next)`). Express recognizes it as an error-handling middleware because of these four arguments.

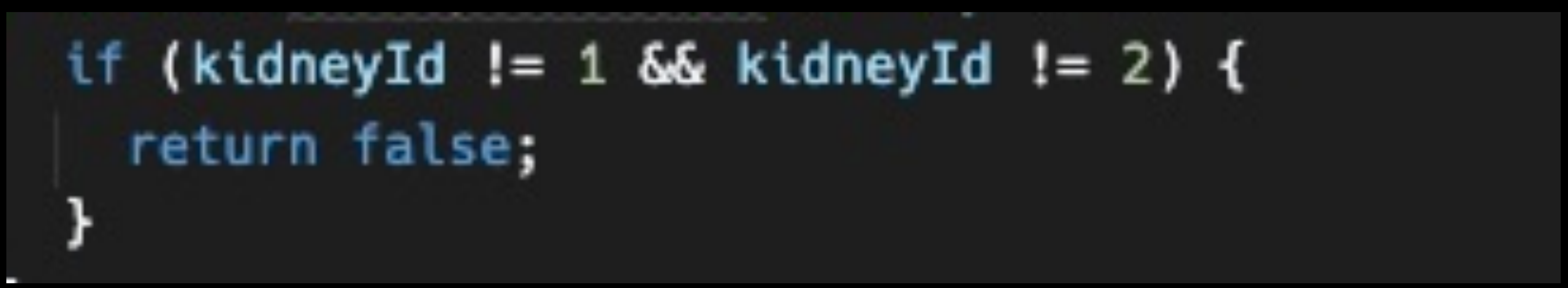

index.js > app.post("/health-checkup") callback

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

const app = express();

app.post("/health-checkup", function (req, res) {
  // do something with kidney here
  const kidneys = req.body.kidneys;
  const kidneyLength = kidneys.length;

  res.send("Your kidney length is " + kidneyLength);
});

app.use((error, req, res, next) => {
    // console.error(error); // Log the error for debugging
    res.status(500).send('An internal server error occurred');
});

app.listen(3000);
```

3.1
Middlewares, authentication, global catches, zod

How can you do better input validation?

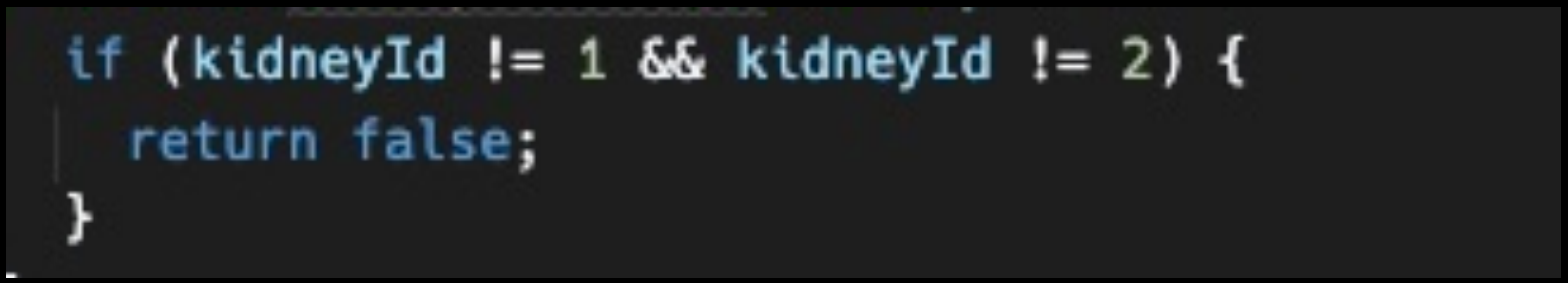
This is very hard to scale
What if you expect a complicated input?



```
if (kidneyId != 1 && kidneyId != 2) {
  return false;
}
```

How can you do better input validation?

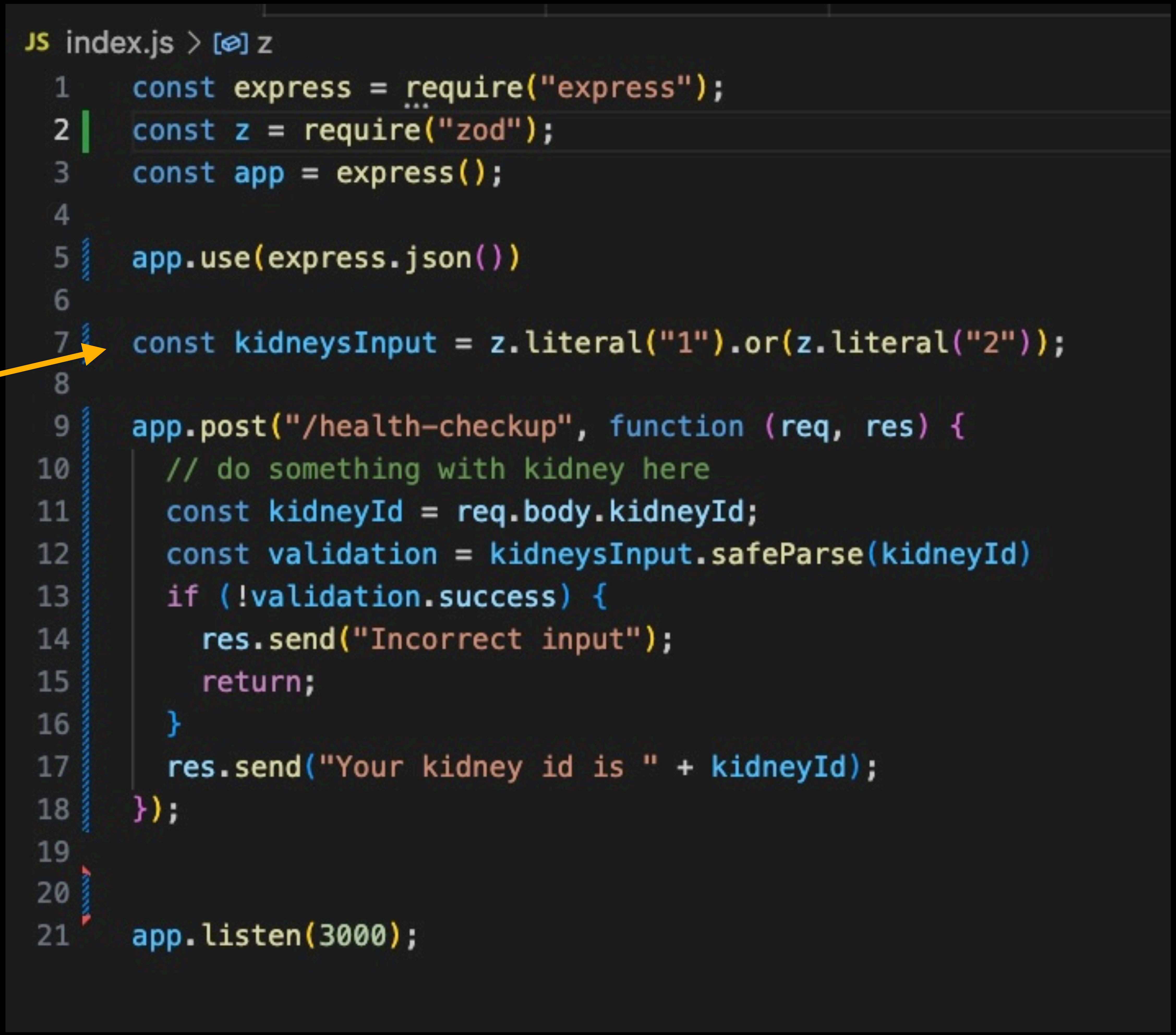
This is where zod comes into the picture



```
if (kidneyId != 1 && kidneyId != 2) {
  return false;
}
```

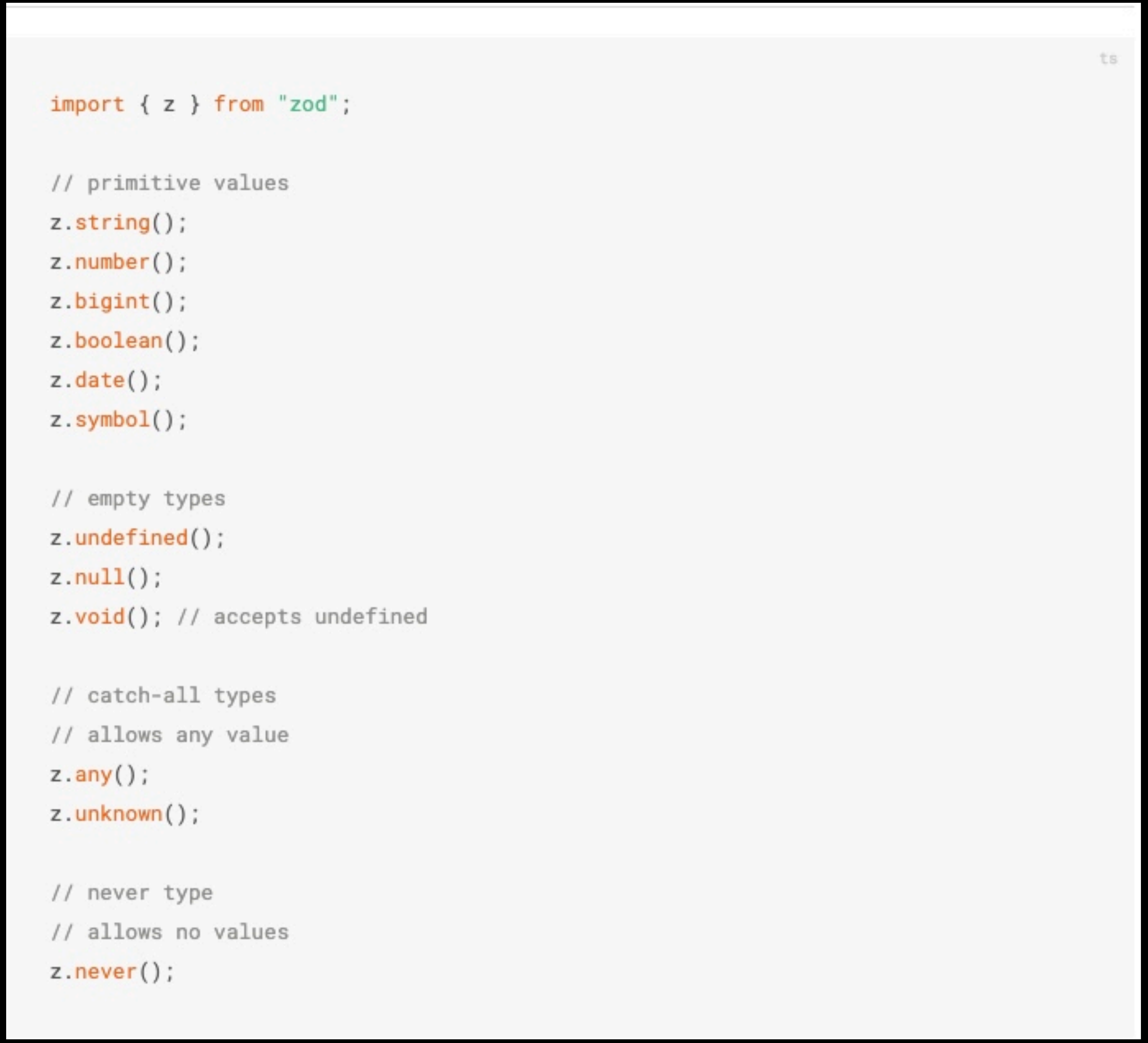
How can you do better input validation?

This is where zod comes into the picture

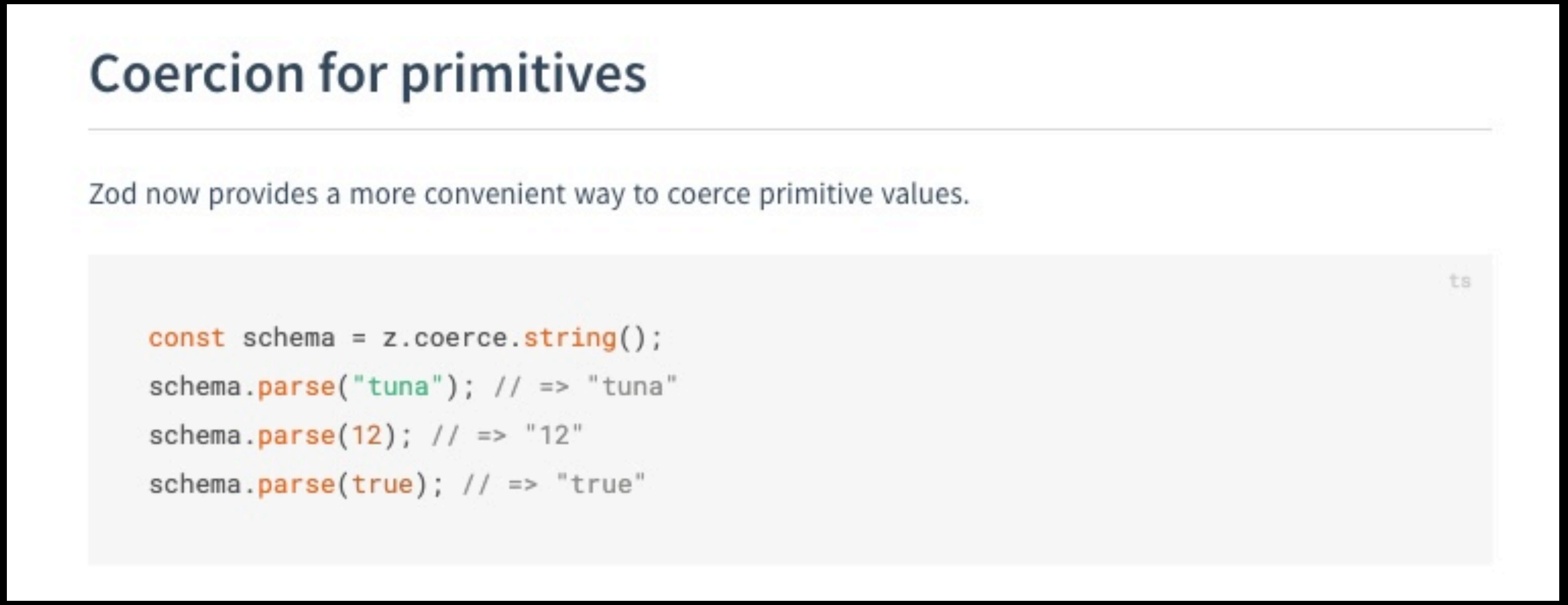


```js
const express = require("express");
const z = require("zod");
const app = express();

app.use(express.json())

const kidneysInput = z.literal("1").or(z.literal("2"));

app.post("/health-checkup", function (req, res) {
  // do something with kidney here
  const kidneyId = req.body.kidneyId;
  const validation = kidneysInput.safeParse(kidneyId)
  if (!validation.success) {
    res.send("Incorrect input");
    return;
  }
  res.send("Your kidney id is " + kidneyId);
});


app.listen(3000);
```

How can you do better input validation?



```ts
import { z } from "zod";

// primitive values
z.string();
z.number();
z.bigint();
z.boolean();
z.date();
z.symbol();

// empty types
z.undefined();
z.null();
z.void(); // accepts undefined

// catch-all types
// allows any value
z.any();
z.unknown();

// never type
// allows no values
z.never();
```



Coercion for primitives

Zod now provides a more convenient way to coerce primitive values.

```ts
const schema = z.coerce.string();
schema.parse("tuna"); // => "tuna"
schema.parse(12); // => "12"
schema.parse(true); // => "true"
```

https://zod.dev/?id=basic-usage

3.1
Middlewares, authentication, global catches, zod

Authentication

As you can tell by now, anyone can send requests to your backend
They can just go to postman and send a request
How do you ensure that this user has access to a certain resource?

Authentication

Dumb way - Ask user to send username and password in all requests as headers

Authentication

Slightly better way -

1. **Give the user back a token on signup/signin**
2. **Ask the user to send back the token in all future requests**
3. **When the user logs out, ask the user to forget the token (or revoke it from the backend)**

Authentication

Library we need to get comfortable in - jsonwebtokens