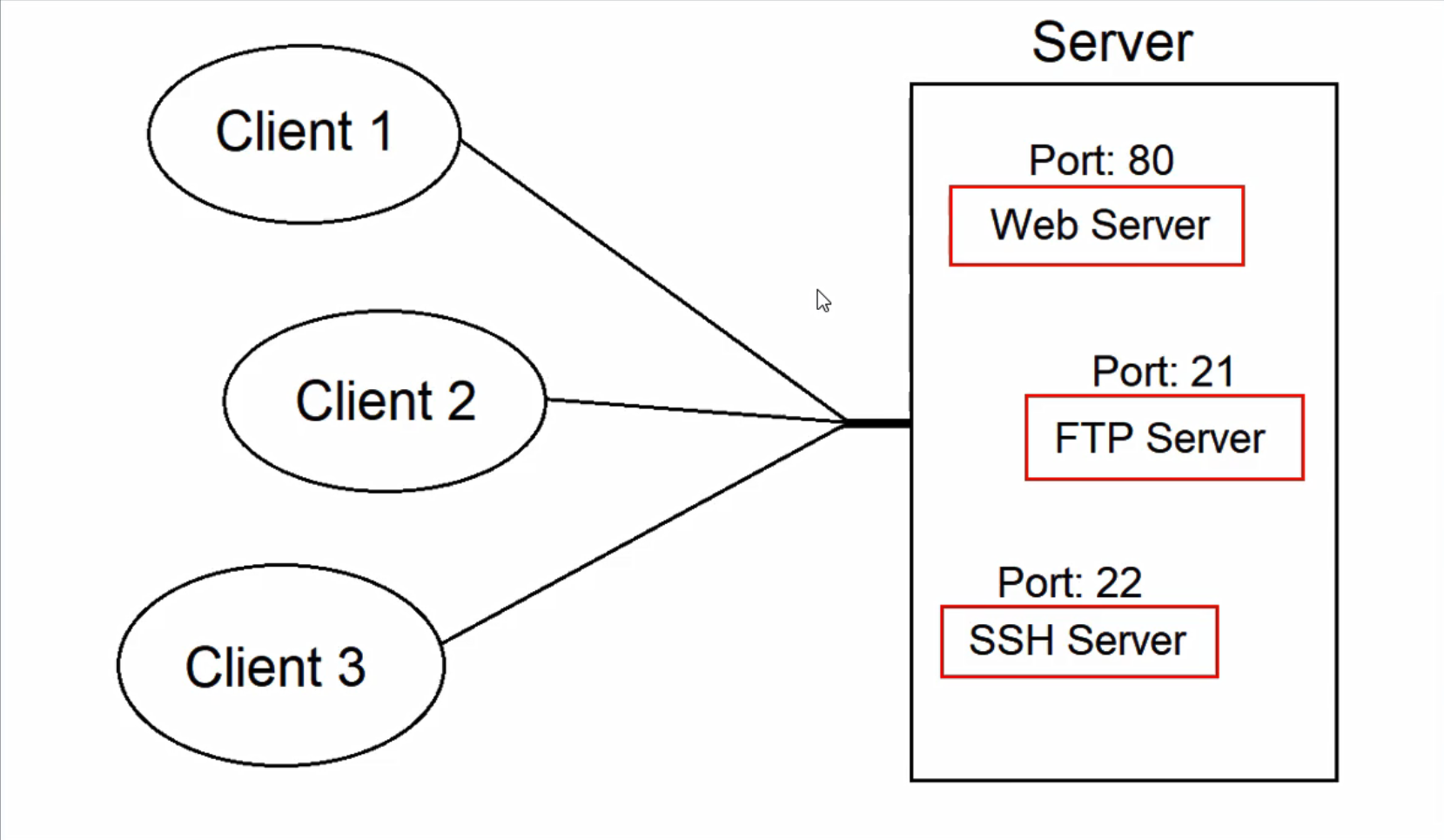
1- ساخت وب سرور با http



import http from "http";

const server = http.createServer(() => {

    console.log("A request received");

});

این کال بک زمانی اجرا میشه که رویداد request برای آبجکت server اتفاق بیفته .

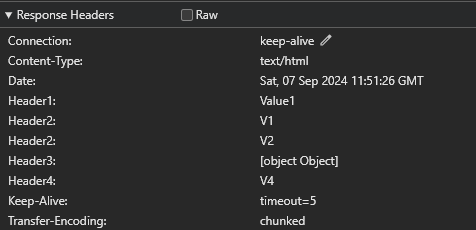
server.listen(3000, () => {

    console.log("server start on port " + 3000);

});

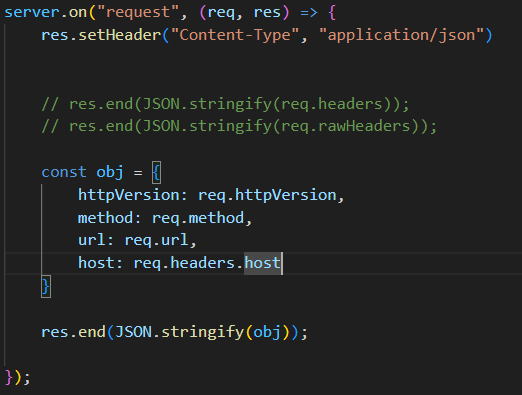
(<http://127.0.0.1:3000/> لوپ بک )





نکته : ست هدرو قبل متدهایی که write میکنن باید بنویسیم . تو پاسخ های اچ تی تی پی اول هدر ارسال میشه بعد بدنه . وقتی که داریم رایت میکنیم در واقع داریم توی بدنه مینویسیم . حالا وقتی بدنه رو داریم مینویسیم دیگه نمیتونیم هدر هارو تغییر بدیم چون قبلش هدر ها ارسال شدن .

نکته : res یک استریم رایتبل و req یک استریم ریدبل هست .







2 – شروع کار با express

const app = express();

app.use((req, res) => {

    console.log(req.url, req.method);

    res.send("aaaaa");

});

app.use("/", (req, res) => {

    console.log(req.url, req.method);

    res.send("bbbbbb");

});

app.use("/test", (req, res) => {

    console.log(req.url, req.method);

    res.send("ccccc");

});

app.listen(3000, () => {

    console.log("http://localhost:3000");

});

3-4-5- میدلویر ها

مت send خودش هدر Content-Type رو تشخیص میده و تو هدر ریسپانس ست میکنه و میفرسته.

از send بیشتر زمانی استفاده میکنیم که میخوایم یه رشته رو ارسال کنیم .

نکته : بعد از متد next() هم میتونیم دستوراتی داشته باشیم تو تابع . ولی وقتی رسید به نکست ، میره به میدلویر بعدی ، و کد های اونو اجرا میکنه و دوباره برمیگرده و کد های بعد از next() رو اجرا میکنه.

تو یکی از میدلویر ها باید یچی سند بشه

app.use("/route1", (req, res, next) => {

    next();

    console.log("Hello from first /route1");

});

app.use("/route1", (req, res, next) => {

    console.log("Hello from first /route1");

    // res.send("Hello from first /route1");

    next();

});

گفتیم که پاسخی که ب کلاینت ارسال توسط یکی از این میدلویر ها ارسال میشه در واقعا همون میدلویری که متد send رو داره استفاده میکنه و باقی میدلویر ها مستقیما چیزی رو برای کلاینت ارسال نمیکنه . خوب این میدلویر هایی که چیزی سند نمیکنن به چه دردی میخورن ؟   
اینا اکثرا به این صورتن که میان رو req یا res تغییراتی ایجاد میکنن .   
لاگ میگیرن و ...

نکته : آبجکت های req و res واحد هستن ینی اگه تغییراتی روش انجام بدیم ، اون تغییرات اعمال میشه و اون req و res با تغییرات انجام شده فرستاده میشه به میدلویر بعدی و ...

function logger(req, res, next) {

    const startTime = Date.now();

    next();

    const duration = Date.now() - startTime;

    console.log("Request duration: " + duration + " ms");

}

function addHeader(req, res, next) {

    res.setHeader("Custom-Header", "Custom Value");

    next();

}

app.use(logger);

app.use(addHeader);

app.use("/route1", (req, res, next) => {

    const array = [];

    for (let i = 0; i < 10000; i++) {

        array.unshift(i);

    }

    next();

    console.log("aaaaaaa");

});

app.use("/", (req, res) => {

    const array = [];

    for (let i = 0; i < 10000; i++) {

        array.unshift(i);

    }

    res.send(array);

});

نکته : تو متد هایی مثل get post patch و ... بر خلاف متد use آرگومان اول اجباری هست . در متد use ، وقتی route ای وارد کنیم ، شامل تمامی زیر مجموعه های اون روت هم میشد برای مثال اگه برای آرگومان اول روت "/" رو در نظر بگیریم ، کل زیر مجموعه های اون (که میشه تمامی روت ها) هم شامل میشه ولی تو متد های get post و ... حتما باید اون روت حتما باید دقیقا با روت ورودی مطابقت کنه ینی اگه "/" رو به عنوان آرگومان اول get بدیم ، دیگه زیر مجموعه هاشو شامل نمیشه و دقیقه باید روت ورودی "/" باشه که بره داخل اون میدلویر .

app.get("/", (req, res, next) => {

    console.log("A GET request received to middleware 3");

    next();

});

حالا اگه خواستیم برای همه انواع متد ها ، یک روت خاص رو در نظر بگیریم چیکار کنیم ؟ (چون یوز میومد تمام زیر مجمعه های روت رو هم در نظر میگیرفت ) در این صورت از متد All ساتفاده میکنیم

app.all("/", (req, res, next) => {

    console.log("A request received to middleware 5");

    next();

})

حالت های ختلف تعیین route :

“/abc” 🡪 abc

"/ab?c" 🡪 abc or ac

"/ab+c" 🡪 abc, abbc, abbbc, ...

"/ab\*c" 🡪 abc, abxc, abvgsthyc, ...

"/a(bc)+d" 🡪 abcd, abcbcc, ...

"/ab.c" 🡪 ab.c

"/ab\\dc" 🡪 ab1c, ab2c, ab9d, ...

/ab\*c/ 🡪 ac, abc, abbc, abbbc, ...

/ab+c/ 🡪 abc, abbc, abbbc, abbbbc, ...

نکته : تو ریجکس ی قسمتی از رشته مطابقت کنه اوکیه ولی تو اون حالت قبلی حتما باید کامل مطابقت کنه

/^\/ab\*c(\/)?$/ 🡪 /ac, /ac/, /abc, /abc/, /abbbc, /abbbc/, ...

/ab.c/ 🡪 abxc, abzc, ab8c, ...

/abc\/\d{2,3}/ 🡪 abc/12, abc/356, …

پارامتر ها :

"/user/:id" 🡪 /user/12, /user/ali

"/user/:id/book/:bookId" 🡪 /user/12/book/roman, /user/ali/book/47, ...

"/file/:name.:ext" 🡪 /file/records.csv,  ...

"/books/:from-:to" 🡪 /books/20-40

"/user/:id(\\d{1,3})" 🡪 /user/1, /user/31, /user/314,

معرفی متد route :

app.get("/book", (req, res) => {

    res.send("A GET request received.");

});

app.post("/book", (req, res) => {

    res.send("A POST request received.");

});

app.put("/book", (req, res) => {

    res.send("A PUT request received.");

});

ب جای بالایی ها میتونیم این پایینی رو بنویسیم.

App.route(“/book”)

    .get((req, res) => {

        res.send(“A GET request received.”);

    })

    .post((req, res) => {

        res.send(“A POST request received.”);

    })

    .put((req, res) => {

        res.send(“A PUT request received.”);

    });

کلاسه Router :

Home-router.js :

import express from "express";

const router = express.Router();

router.get(("/"), (req, res) => {

    res.send("A GET request received for /")

})

router.post("/", (req, res) => {

    res.send("A POST request received for /");

});

export default router;

User-router.js :

import express from "express";

const router = express.Router();

router.get(("/user"), (req, res) => {

    res.send("A GET request received for /user");

})

router.post("/user", (req, res) => {

    res.send("A POST request received for /user");

});

router.get("/user/route1", (req, res) => {

    res.send("A GET request received for /user/route1");

});

router.post("/user/route1", (req, res) => {

    res.send("A POST request received for /user/route1");

});

export default router;

admin-router.js:

import express from "express";

const router = express.Router();

router

    .route("/admin")

    .get((req, res) => {

        res.send("A GET request received for /admin");

    })

    .post((req, res) => {

        res.send("A POST request received for /admin");

    });

router

    .route("/admin/route1")

    .get((req, res) => {

        res.send("A GET request received for /admin/route!");

    })

    .post((req, res) => {

        res.send("A POST request received for /admin/route!");

    });

export default router;

app.js:

import express from "express";

import homeRoutes from "./home-routes.js";

import adminRoutes from "./admin-routes.js";

import userRoutes from "./user-routes.js";

const app = express();

app.use((req, res, next) => {

    console.clear();

    next();

});

app.use("/", homeRoutes);

app.use("/", adminRoutes);

app.use("/", userRoutes);

app.listen(3000, () => {

    console.log("http://localhost:3000");

});

نکته : وقتی داریم از فرم یک سری اطلاعات دریافت میکنیم ، باید میدلویر زیر رو ست کنیم تا بیاد اون مواردی ک دریافت شده رو داخل body ذخیره کنه .

app.use(express.urlencoded({extended: false}))

app.use(express.urlencoded({ extended: false }));

app.get("/", (req, res) => {

    const html = `

    <form action="/result-page" method="post">

        <label>3 + 4 = ?</label><br/>

        <input type="text" name="result" /><br />

        <button type="submit">Submit</button>

    </form>

    <style>

        \*{

            font-size: 24px;

            line-height: 2;

            margin-bottom: 10px;

        }

    </style>

    `;

    res.send(html);

});

app.route("/result-page")

    .post((req, res) => {

        console.log("asdasdasd");

        if (req.body.result) {

            if (req.body.result === "7") {

                res.send("Your answer is correct");

            } else {

                res.send("Ooops!, Your answer is incorrect");

            }

        } else {

            res.status(400).send("please send a value for result.");

        }

    })

    .get((req, res) => {

        res.redirect("/");

    });

app.use("/", (req, res) => {

    res.status(400).send("<h1>Page Not Found</h1>");

});

کوئری استرینگ :

app.get("/user/", (req, res) => {

    res.json(req.query)

})

<http://localhost:3000/user/?x=2&z=ali>

**{**"x": "2","z": "ali"**}**

میدلویر app.use(express.json()) :

وقتی از از طریق یه فرم اچ تی ام ال ی داده ای رو ارسال میکنیم ، داده ها با فرمت urlEncoded ارسال میشن و از میدلور قبلی ینی

app.use(express.urlencoded({ extended: false }));

استفاده میکنیم . حالا اگه این داده ها در قالب جیسون فرستاده شده باشن ، میاد اونارو به پراپرتی بادی از req اضافه میکنه.

برای معرفی فولدر پابلیک به عنوان استاتیک ، بهتره مسیر مطلق بدیم نه نسبی :

const \_\_dirname = path.dirname(fileURLToPath(import.meta.url));

app.use(

    express.static(path.join(\_\_dirname, "publick"))

);

نکته : میتونیم چن تا پوشه رو به عنوان استاتیک داشته باشیم مثلا همینجا میشد که یه پوشه دیگه به اسم publick2 داشته باشیم .

حالا اینجوری میتونیم مثلا از style.css در پابلیک که در پوشه styles هست استفاده کنیم :

app.get("/", (req, res) => {

    const html = `

    <link rel="stylesheet" href="styles/style.css">

    <form action="/result-page" method="post">

        <label>3 + 4 = ?</label><br/>

        <input type="text" name="result" /><br />

        <button type="submit">Submit</button>

    </form>

    `;

    res.send(html);

});