Basic Node JS API Implementation

This time we will learn how the API works on node JS

Create new folder and open with VS Code

Install required NPM package

```
npm i express body-parser cors dotenv
```

Update Allpackage (Run line by line)

```
npm i -g npm-check-updates
```

```
ncu -u
```

```
npm install
```

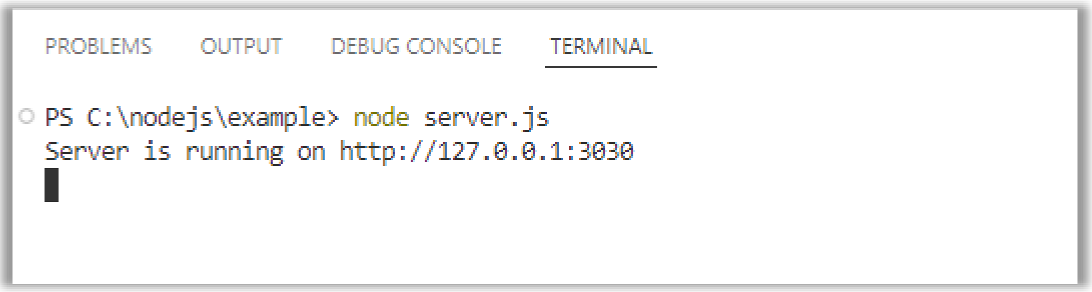
create **.env** file (environment file) and put this code:

```
PORT=3030
```

Create **server.js** file and put this code:

```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");
require("dotenv").config();

const app = express();
const PORT = process.env.PORT || 3000;

app.use(cors());
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static(__dirname));

//POST http://localhost:3030/concat
app.post("/concat", async (req, res) => {
    const { variableA, variableB } = req.body;
    try {
        var resultVar = variableA + variableB;
        res.status(200).json({
            success: true,
            result: resultVar,
        });
    } catch (error) {
        res.status(500).json({ success: false, message: error.message });
    }
});

// Start the server
app.listen(PORT, () => {
    console.log(`Server is running on http://127.0.0.1:${PORT}`);
});
```

Run server.js



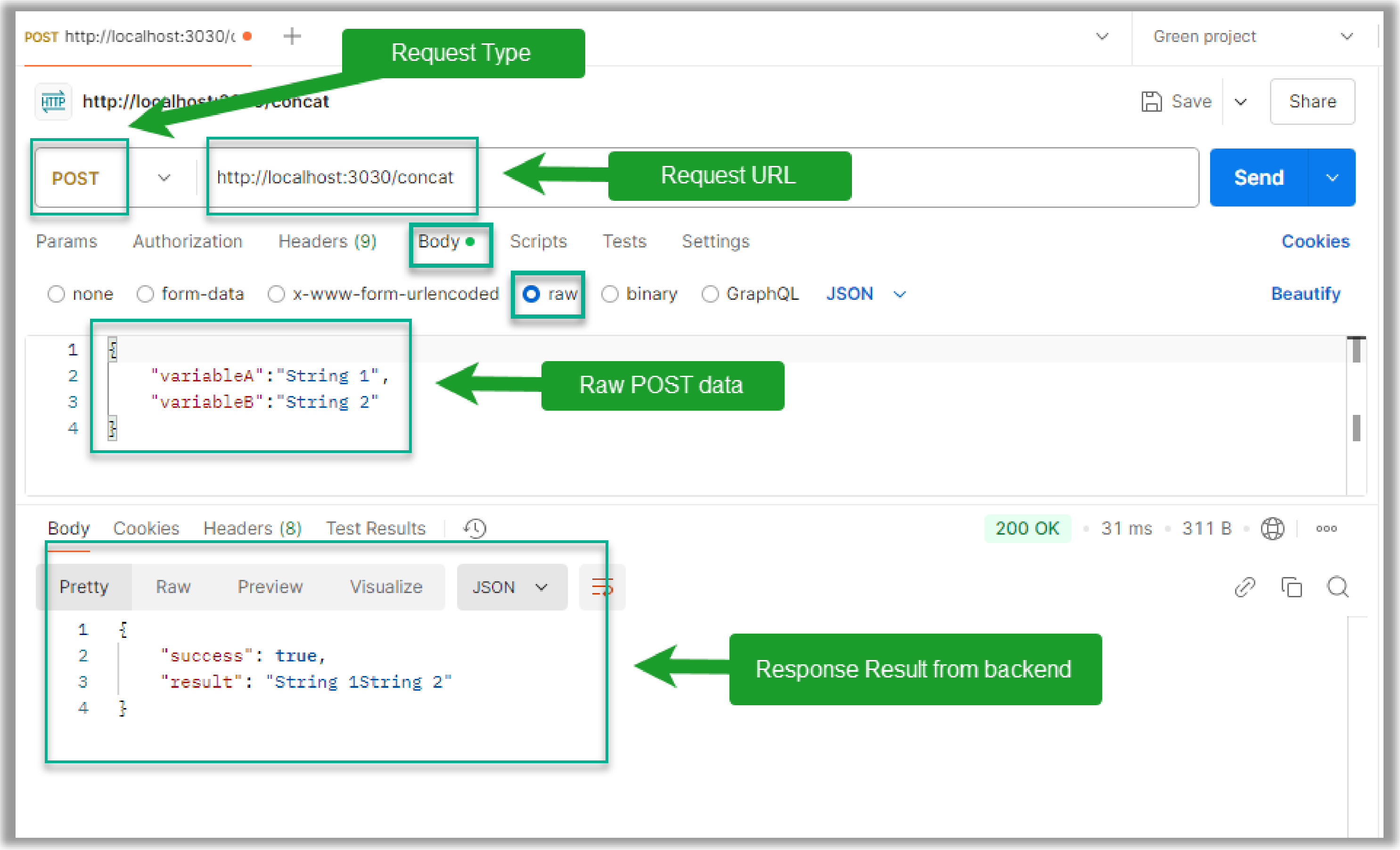
```
PS C:\nodejs\example> node server.js
Server is running on http://127.0.0.1:3030
```

Testing Backend API with postman

Open postman and send raw POST data:

```
{
    "variableA":"String 1",
    "variableB":"String 2"
}
```

To http://localhost:3030/concat

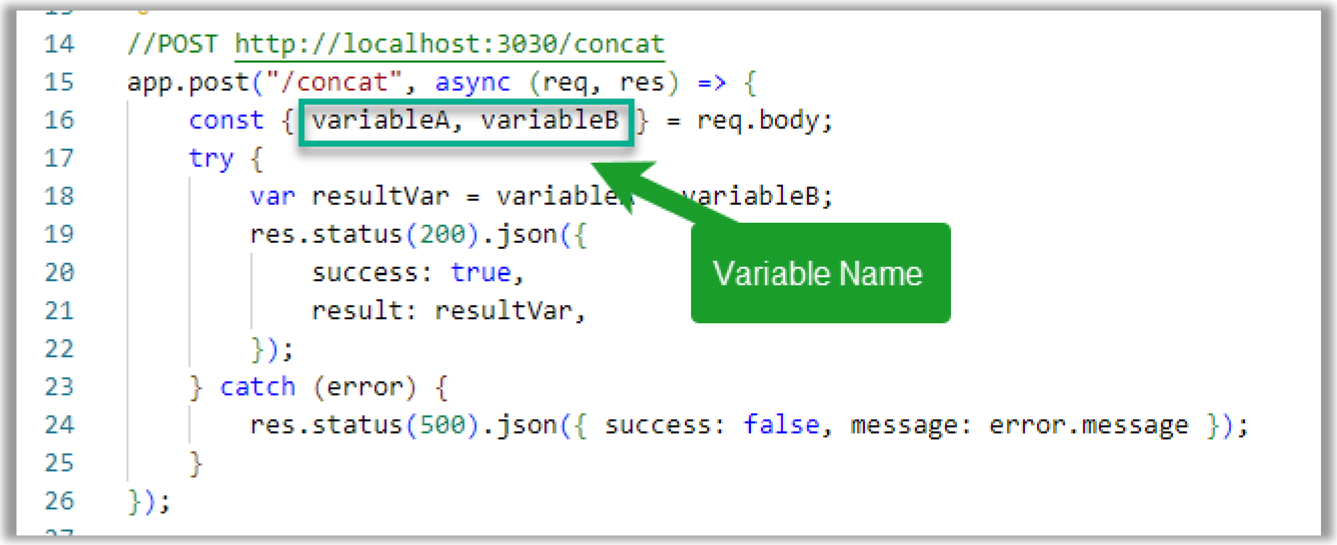


```
http://localhost:3030/concat
```

```
{
    "variableA":"String 1",
    "variableB":"String 2"
}
```

```
{
    "success": true,
    "result": "String 1String 2"
}
```

You can replace "**String 1**" and "**String 2**" with any string, But " **variableA**" and " **variableB**" must be the same as the variable names in the backend API



```
//POST http://localhost:3030/concat
app.post("/concat", async (req, res) => {
    const { variableA, variableB } = req.body;
    try {
        var resultVar = variableA + variableB;
        res.status(200).json({
            success: true,
            result: resultVar,
        });
    } catch (error) {
        res.status(500).json({ success: false, message: error.message });
    }
});
```

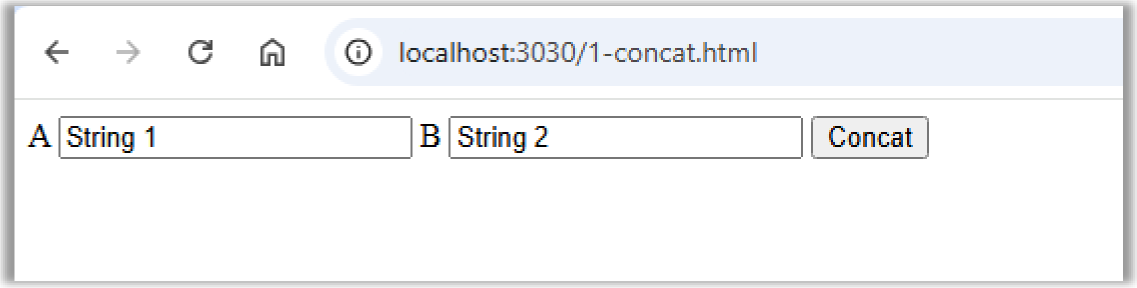
implementing backend API into frontend

Create **1-concat.html** as frontend file and put this code:

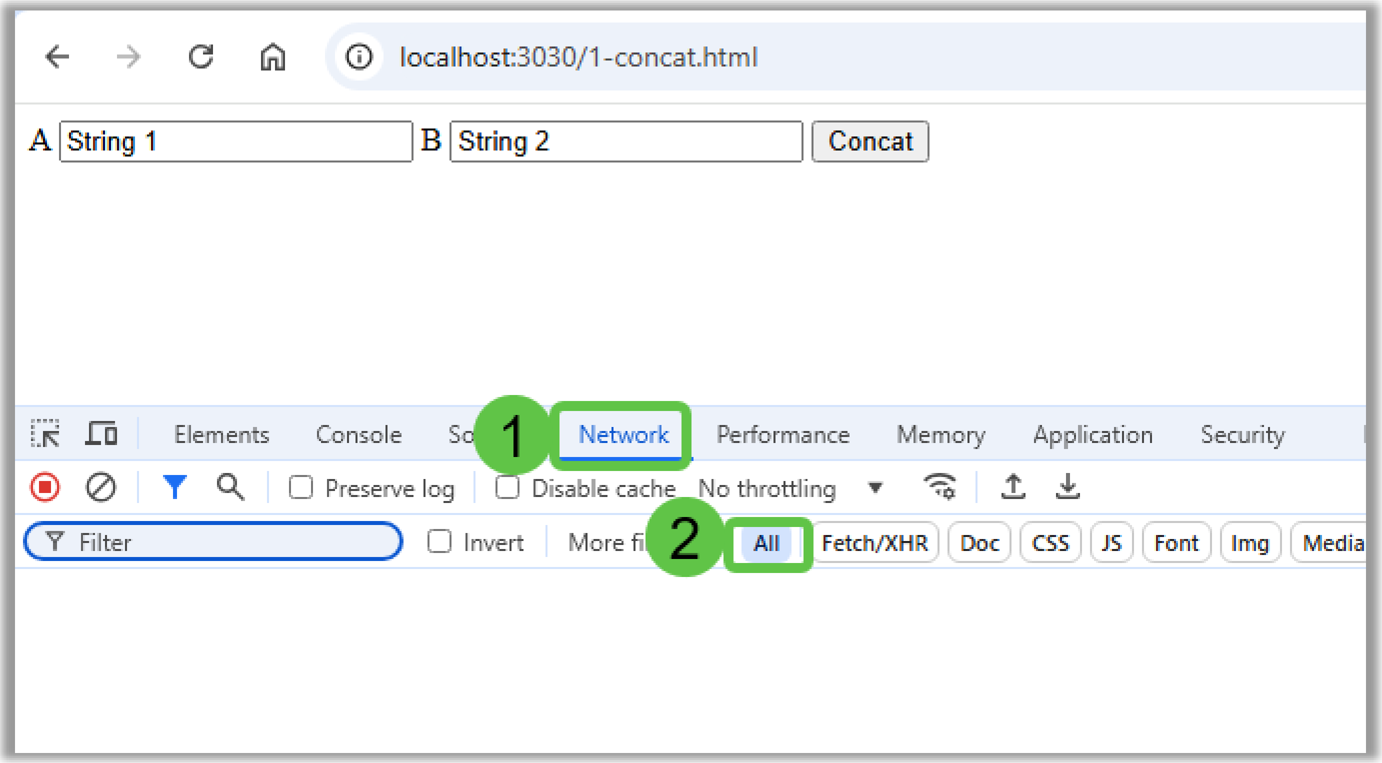
```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Concat</title>
        <script
            src="https://code.jquery.com/jquery-3.7.1.min.js"
            integrity="sha256-/JqT3SQfawRcv/BIHPThkBvs0OEvtFFmqPF/lYI/Cxo="
            crossorigin="anonymous"
        ></script>
    </head>

    <body>
        <form id="myForm">
            <label>A <input type="text" name="variableA" value="" /></label>
            <label>B <input type="text" name="variableB" value="" /></label>
            <button type="submit">Concat</button>
        </form>

        <script>
            $("#myForm").submit(function (event) {
                event.preventDefault();
                $.ajax({
                    type: "POST",
                    url: "http://localhost:3030/concat",

                    data: $(this).serialize(),
                    success: function (data) {
                        if (data.success) {
                            alert(data.result);
                        }
                    },
                    error: function (xhr, status, error) {
                        alert(xhr.responseJSON.message);
                    },
                });
            });
        </script>
    </body>
</html>
```

Note:
localhost is another name for 127.0.0.1
so, URL http://localhost:3030 is the same as http://127.0.0.1:3030

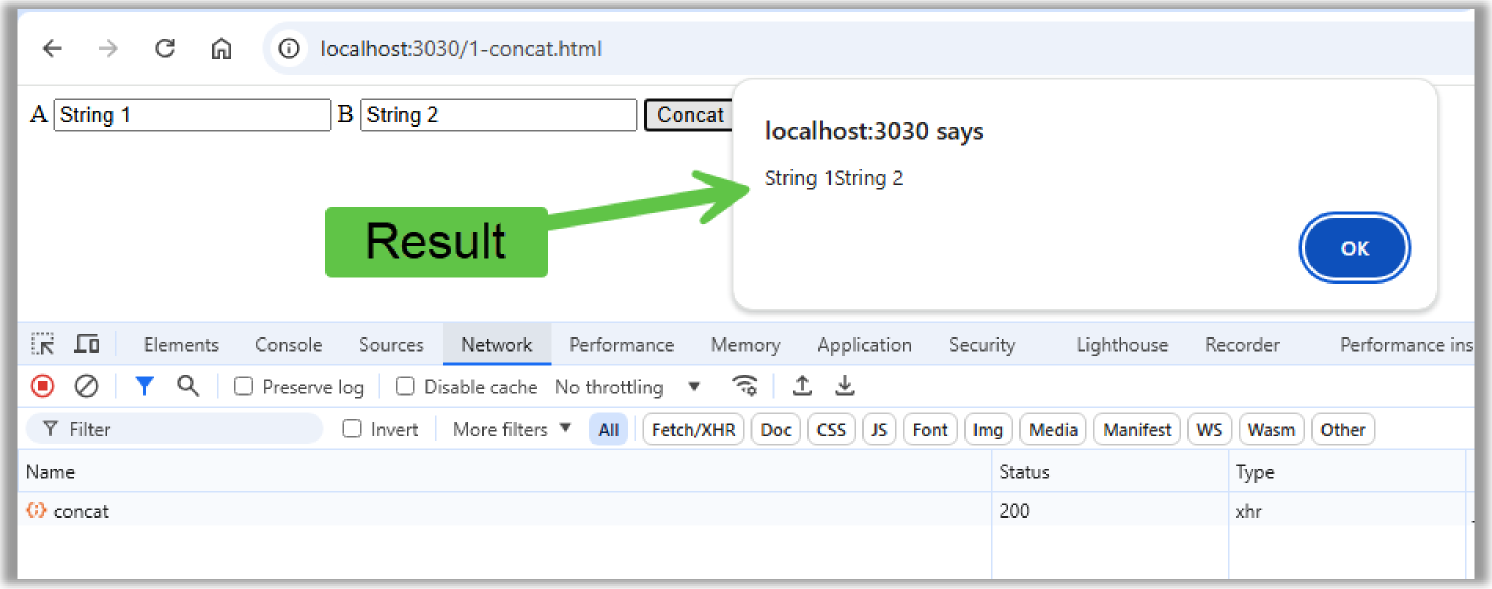
Open http://localhost:3030/1-concat.html in google chrome and fill all input text



Press F12 to open dev tool and select network tab



Click Concat button to get result

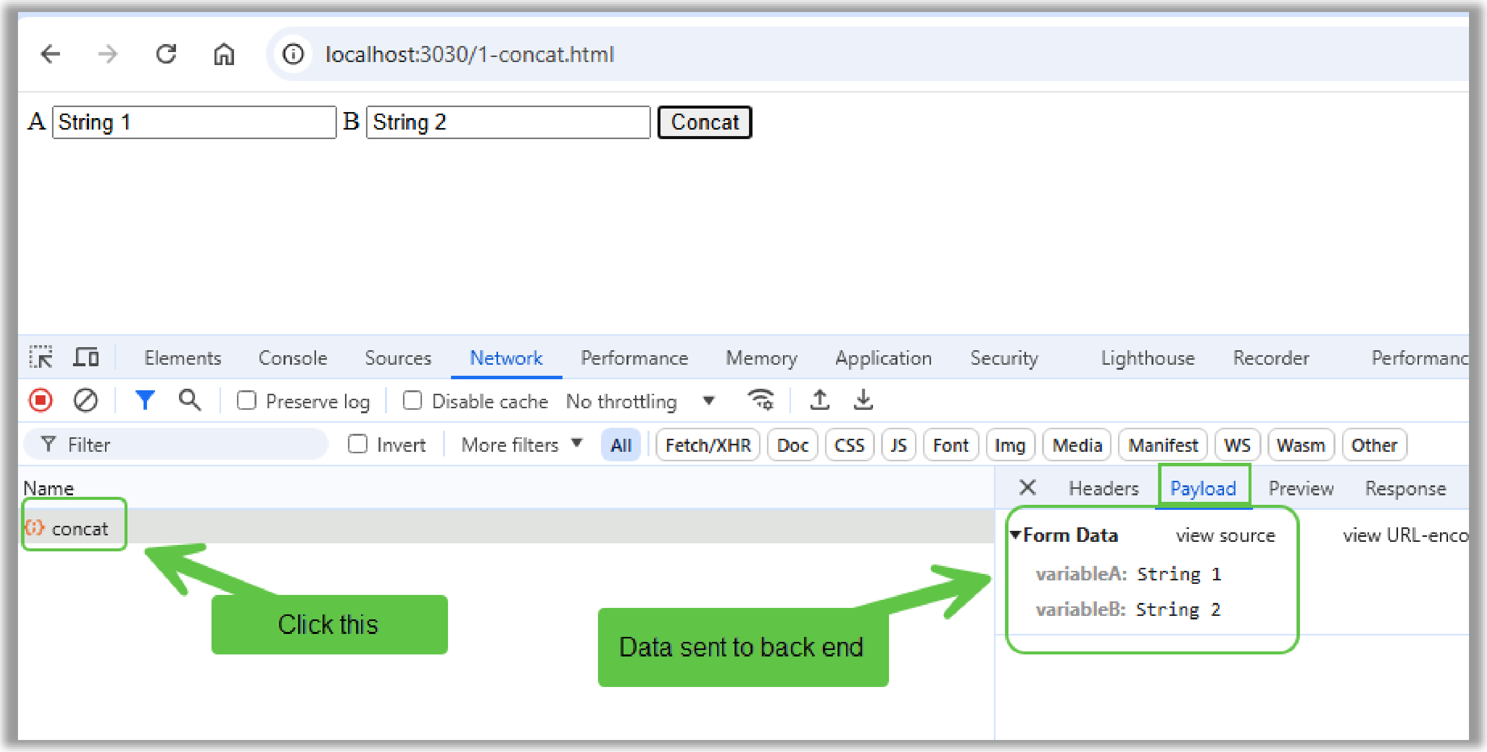


localhost:3030 says

String 1String 2

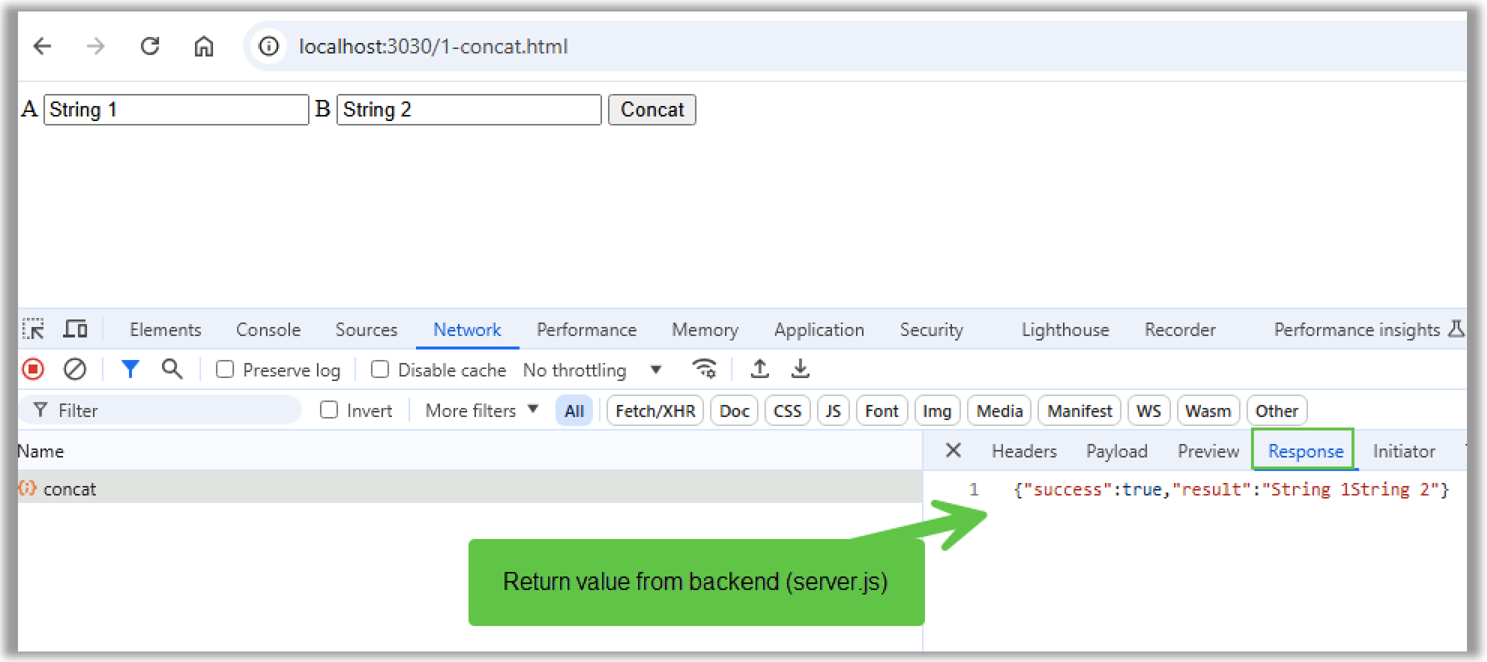
Name	Status	Type
concat	200	xhr

Press OK button to close alert and click concat in network tab. Select payload to view data sent to backend


localhost:3030/1-concat.html
A String 1 B String 2 Concat
Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performanc
Preserve log Disable cache No throttling
Filter Invert More filters All Fetch/XHR Doc CSS JS Font Img Media Manifest WS Wasm Other
Name
concat
Headers Payload Preview Response
Click this
Data sent to back end

Form Data view source view URL-enco
variableA: String 1
variableB: String 2

Click Response to view return value from backend


localhost:3030/1-concat.html
A String 1 B String 2 Concat
Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance insights
Preserve log Disable cache No throttling
Filter Invert More filters All Fetch/XHR Doc CSS JS Font Img Media Manifest WS Wasm Other
Name
concat
Headers Payload Preview Response Initiator
Return value from backend (server.js)

```
{"success":true,"result":"String 1String 2"}
```

Conclusion:

Basically, postman and browser do the same thing. But in the browser/frontend the user interface and user experience are created with html, css and javascript.