NodeJS

środowisko i technologia ServerSide

PAWEŁ ŁUKASZUK

Rok akademicki: 2023/2024, Semestr: 2, Wykład: 1

Server for Java Script apps

To access web pages of any web application, you need a web server.

The web server will handle all the http requests for the web application e.g IIS is a web server for ASP.NET web applications and Apache is a web server for PHP or Java web applications.

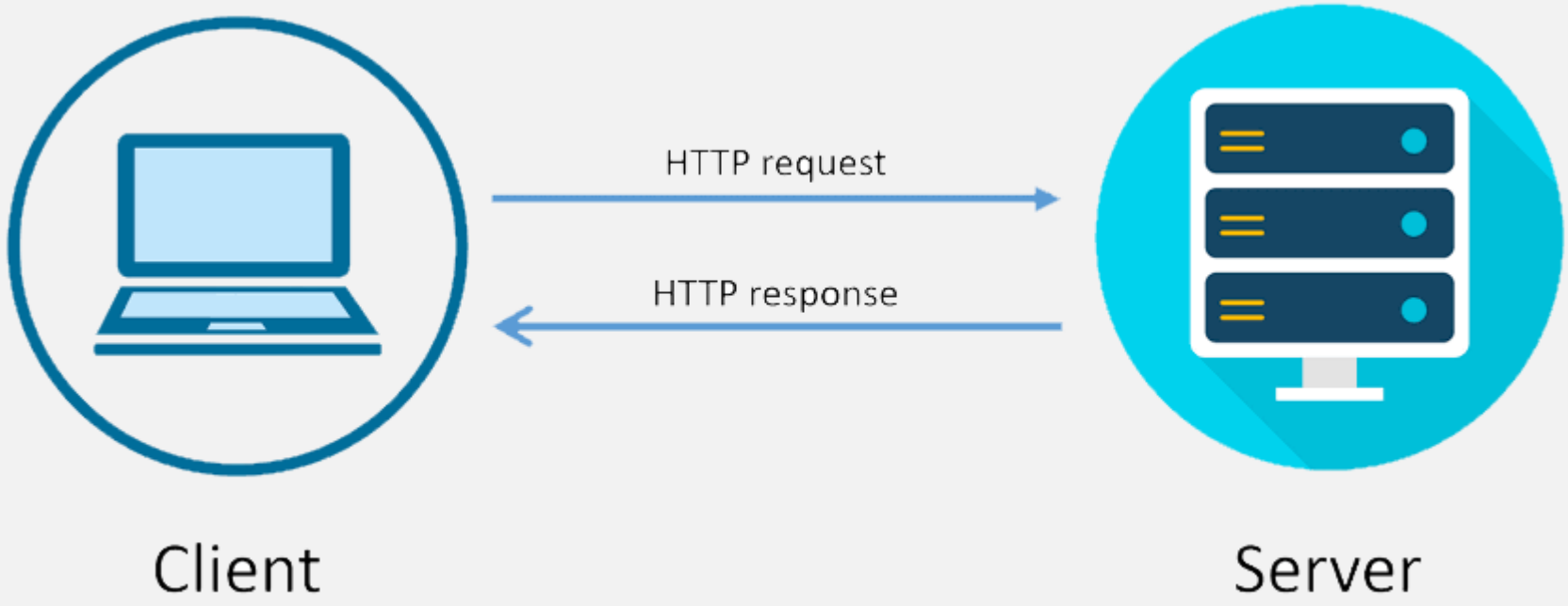
Node.js provides capabilities to create your own web server which will handle HTTP requests asynchronously. You can use IIS or Apache to run Node.js web application but it is recommended to use Node.js web server.

Server side Node.js

The web server passively waits for requests sent by clients. We have to send these requests ourselves - using a browser, Postman or other software.

Even in the absence of requests, the server should continue to work.

The end of operation should only occur as a result of an intentional shutdown or unhandled failure.



HTTP module

The http module is a core module of Node designed to support many features of the HTTP protocol.

It is most often used for two purposes:

- as a client - to send requests
- as a server - for receiving and handling requests

First scenario is less common because of more convenient approaches like native fetch or Axios library.

Basic HTTP server

```
const http = require('http');

http.createServer(
    function (req, res) {
        res.write('Hello World!');
        res.end();
}).listen(4700,
      console.log("server started");
```

Basic HTTP server

```
const http = require('http');                    //module import

http.createServer(                              // create server
    function (req, res) {                        // listener function which is
                                                    called when request is received
        res.write('Hello World!');               // write to reponse
        res.end();                               // send reponse

}).listen(4700,                                 // starts the HTTP server listening
                                                    for request on port 4700
    console.log("server started"));             // optional callback function
                                                    called after server starts
```

HTTP server - important #1

```
listen(4700);
```

Port number can be (almost) random, however

two different applications cannot use the same port

for the same communication protocol

Most common ports used by system services for HTTP are: 80 and 8080

https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

HTTP server - important #2

Every received request should be handled with

```
res.end();      // send reponse
```

Lack of this statement will cause client to wait.

HTTP server - important #3

You cannot send two responses for one request

neither change properties of already send response.

```
res.end();      // send reponse
res.end();      // send reponse
```

Results with error:

```
Error [ERR_HTTP_HEADERS_SENT]:
Cannot set headers after they are sent to the client
```

Listener function

```
function (req, res) {
      // ...
}
```

This function handles all request received by server.

It has two arguments:

- req – request, contains message sent by client
- res – response, contains message that will be send back to client

Listener - request

Important properties:

- method – HTTP method, GET, POST etc
- url – part of URL address used by client, (without protocol and host because server has no access to this data)
- headers – collection of headers
- httpVersion – HTTP protocol versions

Listener - response functions

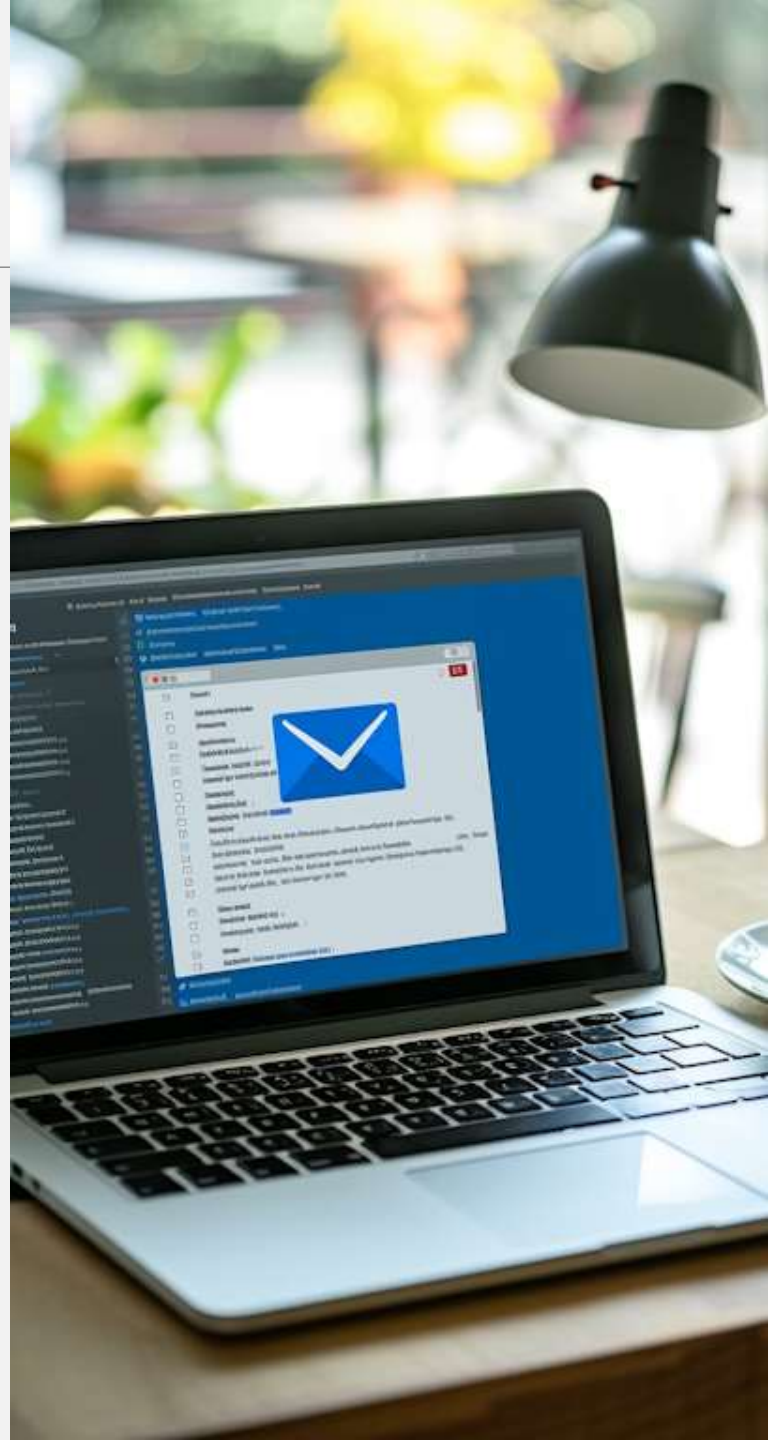
Important functions

- end() – signals server to send response
- getHeader("header") – returns the value of the specified header
- removeHeader("header") – removes the specified header
- setHeader("header", "value") – sets the specified header
- hasHeader("header") – return true when header is set
- write("message") – writes data in response body
- writeHead(200, {'Content-type':'text/plain’}) – writes data in response headers

Listener - response properties

Important properties

- statusCode – HTTP status code, as number
- statusMessage – message for HTTP status code



HTTP server

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

const app = http.createServer((request, response) => {
    if (request.method === "GET"){
        response.writeHead(200, {"Content-type":"text/plain"})
        response.write("Hello World from method GET");
    }

    response.end();
});

app.listen(4700);
```