HTTP Request

The HTTP server request object (do not confuse this with the client request object) has all the information about the incoming request to our server. Some examples include headers, URL, HTTP method names and of course the request body (payload). Here's the list of main properties:

- `request.headers`: Information about incoming requests headers such as Connection, Host, Authorization, etc (see list here)
- `request.method`: Information about the incoming requests methods such as GET, POST, PUT, DELETE, OPTIONS, HEAD, etc.
- `request.url`: Information about the incoming request URL, such as `/accounts`, `/users`, `/messages`, etc.

All values are accessible in the request handler callback. For example, you can print the values like this:

```
const http = require('http')
const port = 3000
http.createServer((request, response) => {
  console.log(request.headers)
  console.log(request.method)
  console.log(request.url)
  response.writeHead(200, {'Content-Type': 'text/plain'})
  response.end('Hello World\n')
}).listen(port)
```

The result of this script will depend on what requests are coming. Each request will trigger the output of its headers, method, and the URL.

Processing Incoming Request Body in the Server

To process the request body, use the same event emitter pattern as with the request client. Listen to the `data` event and collect the incoming payload using a buffer variable (`buff`). Take a look at `server-request-body.js`:

```
const http = require('http')
const port = 3000
http.createServer((request, response) => {
  console.log(request.headers)
  console.log(request.method)
  console.log(request.statusCode)
  console.log(request.url)
  if (request.method == 'POST') {
    let buff = ''
    request.on('data', function (chunk) {
      buff += chunk
    })
    request.on('end', function () {
      console.log(`Body: ${buff}`)
      response.end('\nAccepted body\n\n')
    })
  } else {
    response.writeHead(200, {'Content-Type': 'text/plain'})
    response.end('Hello World\n')
  }
}).listen(port)
```

The result of this program (`server-request-body.js`) would be a server which accepts POST requests and prints the request body in the server logs.