



javascript

 Copiar código

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

- Importa o módulo `http`, que permite criar um servidor web e lidar com requisições HTTP.


javascript

 Copiar código

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

- Importa o módulo `fs` (File System), que permite interagir com o sistema de arquivos, como ler e escrever arquivos.


javascript

 Copiar código

```
const server = http.createServer((req, res) => {
```

- Cria um servidor HTTP. A função recebe dois parâmetros: `req` (requisição) e `res` (resposta), que representam, respectivamente, a solicitação do cliente e a resposta que será enviada de volta.


javascript

 Copiar código

```
console.log(req.url, req.method);
```

- Exibe no console a URL da requisição e o método HTTP (como GET, POST, etc.) que foi usado.


javascript

 Copiar código

```
res.setHeader('Tipo-Conteudo', 'texto/html');
```

- Define um cabeçalho na resposta HTTP, especificando que o tipo de conteúdo é HTML. Porém, há um erro de digitação; o correto seria 'Content-Type'.


javascript

 Copiar código

```
let caminho = './views/';
```

- Inicializa uma variável `caminho` que armazena o diretório onde os arquivos HTML estão localizados.


javascript

 Copiar código

```
switch(req.url){
```

- Inicia uma estrutura `switch` para verificar a URL da requisição e decidir qual arquivo HTML retornar.


javascript

 Copiar código

```
case '/':  
  caminho += 'index.html';  
  res.statusCode = 200;  
  break;
```

- Se a URL for a raiz (`/`), o caminho é atualizado para `./views/index.html` e o código de status da resposta é definido como 200 (OK).


javascript

 Copiar código

```
case '/login':  
  caminho += 'login.html';  
  res.statusCode = 200;  
  break;
```

- Se a URL for `/login`, o caminho é atualizado para `./views/login.html` e o código de status é 200.


javascript

 Copiar código

```
case '/criarConta':  
  caminho += 'criarConta.html';  
  res.statusCode = 200;  
  break;
```

- Se a URL for `/criarConta`, o caminho é atualizado para `./views/criarConta.html` e o código de status é 200.

javascript


 Copiar código

```
case '/produtos':  
  caminho += 'produtos.html';  
  res.statusCode = 200;  
  break;
```

- Se a URL for `/produtos`, o caminho é atualizado para `./views/produtos.html` e o código de status é 200.




javascript

 Copiar código

```
case '/carrinho':  
  caminho += 'carrinho.html';  
  res.statusCode = 200;  
  break;
```

- Se a URL for `/carrinho`, o caminho é atualizado para `./views/carrinho.html` e o código de status é 200.


javascript

 Copiar código

```
default:  
  caminho += '404.html';  
  res.statusCode = 404;  
  break;
```

- Se a URL não corresponder a nenhum dos casos anteriores, o caminho é atualizado para `./views/404.html` e o código de status é definido como 404 (Não encontrado).

javascript

 Copiar código

```
fs.readFile(caminho, (err, data) => {
```

- Lê o arquivo HTML especificado pelo `caminho`. A função recebe dois parâmetros: `err` (erro, se ocorrer) e `data` (dados do arquivo).


javascript

 Copiar código

```
if(err){  
    console.log(err);  
    res.end();  
} else {  
    res.end(data);  
}
```

- Se ocorrer um erro ao ler o arquivo, ele é exibido no console e a resposta é finalizada sem enviar dados. Caso contrário, o conteúdo do arquivo é enviado na resposta.

javascript

 Copiar código

```
});  
  
server.listen(3000, 'localhost', () => {
```

- Faz o servidor escutar requisições na porta 3000 e no endereço 'localhost'. A função de callback é executada quando o servidor está pronto para receber requisições.

javascript

 Copiar código

```
console.log("Ouvindo requisição na porta 3000");  
});
```

- Exibe uma mensagem no console informando que o servidor está ouvindo na porta 3000.