ARQUIVO SERVIDOR:

```
//SISTEMA DE MÓDULOS:

//Entender o Sistema de Módulos é essencial para entender como uma aplicação Javascript funciona no Back-End usando o Node como runtime do Javascript.

//Falando um pouco sobre o funcionamento das aplicações Front-End, antigamente as aplicações Front-End eram feitas em um único arquivo Javascript com várias funções soltas que eram chamadas pelo arquivo HTML. Porém, hoje em dia não é mais feito assim, as aplicações são desenvolvidas em varios arquivos, onde cada arquivo tem sua funcionalidade, e esse arquivo é concatenado e minificado antes de ser enviado ao browser. Isso permite aplicações mais organizadas e mais leves, melhorando o desempenhos das nossas aplicações Front-End.

//A programação em Back-End já foge desse paradigma de compactação do arquivo, nós ainda temos que organizar nossos arquivos em pastas para manter a boa organização dos arquivos, porém, não precisamos compactá-los, pois eles serão interpretados no servidor, por uma linguagem que o servidor entende e browser desconhece. A partir daí o que será trafegado para o browser serão arquivos em formato JSON, que serão gerados no Servidor e interpretados pelo Javascript no browser.

//E como os arquivos são interpretados pelo Node?
//No Node, um arquivo é um módulo. E os módulos tem um comportamento especial que é não permitir que os códigos escritos dentro dele fiquem disponíveis para qualquer código externo. Essa ideia é bem parecida com o encapsulamento da POO. Para que tenhamos acesso aos códigos de um módulo ou para que possamos exportar códigos do nosso módulo para outras aplicações usamos o sistema "commonJS" que nada mais é do que a forma como o Node Importa e Exporta códigos de um módulo para uma aplicação. Vejamos como fazer a exportação e importação de arquivos...

//Para Exportar usamos o module.exports, que pode ser usado das seguintes formas:
this.ola = 'Fala pessoal' //É importante que saibamos que todo módulo é como se fosse um grande objeto e todos os elementos criados dentro dele são atributos desse objeto, o "this" é uma forma de criar um destes atributos, porém é uma forma menos comum...
exports.bemVindo = 'Bem vindo ao node' //O "exports" é uma forma abreviada de usar o "module.exports"...
module.exports.ateLogo = 'Até logo' //O "module.exports" é a forma mais comum de atribuir novos atributos ao módulo, quando queremos criar um valor de chave específica...
let naoExports = 'Não fui exportado' //Veja que valores que não foram exportados não ficaram visíveis em outro arquivo...

//****DESCOMENTE AQUI!!! Descomente aqui para ver que o "module.exports" atribuído diretamente sobrescreve quaisquer módulos criados anteriormente...
```

```
// module.exports = { //O "module.exports" com a atribuição direta é o módulo mais comum a ser criado, onde geramos chaves diretamente dentro do objeto. Nesse método de sobrescrita de todo o módulo temos que usar o nome completo "module.exports" se tentarmos usar "this" ou "exports" o nosso arquivo vai gerar erro ou undefined...
//     bomDia: 'Bom dia',
//     boaNoite() {
//         return 'Boa noite'
//     }
// }

//PARA VER COMO OS MÓDULOS SÃO IMPORTADOS CONSULTE A AULA 131-Sistemas_de_Modulos_Cliente.js...
```

ARQUIVO CLIENTE ...

```
//IMPORTAÇÃO DE MÓDULOS:
//Os módulos são importados por através do médoto "require()" do Node. Podemos atribuir todos os módulos exportados para uma variável a nossa escolha.
//OBS: Quaisquer variáveis e funções criadas dentro dos módulos que não estiverem usando "module.exports" não poderão ser importadas...

const modulo = require('./131-Sistema_de_Modulos') //O "./" indica o caminho relativo para que a nossa aplicação encontre o módulo, sempre que o nosso módulo estiver dentro do mesmo servidor, temos que percorrer todo o caminho á partir do "./"...

console.log(modulo.ola) //Perceba que para chamar os valores basta usarmos a notação de objeto que o valor virá naturalmente...
console.log(modulo.bemVindo)
console.log(modulo.ateLogo)
console.log(modulo) //Veja como muda quando o module.exports direto é descomentado, o module do arquivo é sobrescrito, e os atributos passados anteriormente são descontinuados, gerando valores undefined. Além disso, veja que a variável criada no módulo não pôde ser exportada por não usar "module.exports"...
console.log(modulo.boaNoite())
```

RESULTADO NO CONSOLE COM ARQUIVO SERVIDOR COMENTADO...

```
[Running] node "c:\Users\Almoxarifado\Documents\javascript\arquivos_das_aulas\131-Sistema_de_Modulos_Cliente.js"
Fala pessoal
Bem vindo ao node
Até logo
```

```
{
  ola: 'Fala pessoal',
  bemVindo: 'Bem vindo ao node',
  ateLogo: 'Até logo'
}
c:\Users\Almoxarifado\Documents\javascript\arquivos_das_aulas\131-Sistema_de_Modulos_Cliente.js:11
console.log(modulo.boaNoite())
                  ^

TypeError: modulo.boaNoite is not a function
    at Object.<anonymous> (c:\Users\Almoxarifado\Documents\javascript\arquivos_das_aulas\131-Sistema_de_Modulos_Cliente.js:11:20)
    at Module._compile (internal/modules/cjs/loader.js:1072:14)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1101:10)
    at Module.load (internal/modules/cjs/loader.js:937:32)
    at Function.Module._load (internal/modules/cjs/loader.js:778:12)
    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:76:12)
    at internal/main/run_main_module.js:17:47

[Done] exited with code=1 in 0.128 seconds
```

RESULTADO NO CONSOLE COM ARQUIVO SERVIDOR DESCOMENTADO...

```
[Running] node "c:\Users\Almoxarifado\Documents\javascript\arquivos_das_aulas\tempCodeRunnerFile.js"
undefined
undefined
undefined
{ bomDia: 'Bom dia', boaNoite: [Function: boaNoite] }
Boa noite

[Done] exited with code=0 in 0.253 seconds
```