

Módulos do Núcleo do Node



Rafael Hoffmann

[raelhoff@{gmail.com/edu.univali.br}](mailto:raelhoff@gmail.com/edu.univali.br)



Universidade do Vale do itajaí (UNIVALI)

Centro de Ciência e Tecnologia da Terra e do Mar (CTTMar)

Curso de Ciência da Computação - Campus São José

Núcleo do Node

Node tem um pequeno **grupo de módulos** que vem com ele por **padrão**.

Esses **módulos são público** e podem ser utilizado para escrever novas aplicações.

[Documentação](#)



Universidade do Vale do itajaí (UNIVALI)
Centro de Ciência e Tecnologia da Terra e do Mar (CTTMar)
Curso de Ciência da Computação - Campus São José

Sistema de Módulo



Universidade do Vale do itajaí (UNIVALI)
Centro de Ciência e Tecnologia da Terra e do Mar (CTTMar)
Curso de Ciência da Computação - Campus São José

Sistema de Módulos

- No Node.js o código é organizado por meio de módulos.
- Existe uma relação direta entre um **arquivo** e um **módulo**.
- Tudo que for definido dentro do módulo é **privado**
- Os módulos podem ser exportados utilizando **exports**, **this** ou **module.export**

Então vamos criar dois novos módulos **operacoes.js** e o **circulo.js**



Sistema de Módulos (operacoes.js)

```
// operacoes.js
function soma(a,b){
  return a + b;
}

function multiplicar(a,b){
  return a * b;
}

module.exports = {
  soma: soma,
  multiplicar: multiplicar
}
```

Sistema de Módulos (circulo.js)

```
// circulo.js  
var PI = Math.PI; // math biblioteca para javascript e node  
  
this.area = function (r) {  
    return PI * r * r;  
};  
  
exports.circunferencia = function (r) {  
    return 2 * PI * r;  
};
```

MATH

Sistema de Módulos - require

Como utilizar estes módulos?

Para utilizar basta chamar a função **require()**

A função **require** é responsável por retornar o que foi exportado de um outro módulo



Sistema de Módulos - require

```
//index.js
```

```
var operacoes = require('./operacoes');  
var circulo    = require('./circulo');
```

```
operacoes.soma(2,3);  
circulo.circunferencia(5);
```


Sistema de Módulos

Qual a diferença entre `module.exports`, `exports` e `this`?



Sistema de Módulos

```
console.log(module.exports === this); //true  
console.log(module.exports === exports); //true  
console.log(exports === this); //true
```

Todos eles apontam para a **mesma referência**, mas cuidado, apenas **module.exports** é retornado da função require



Arquivo dos Módulos

Um módulo com prefixo `` / `` será buscado no caminho absoluto sistema de arquivos. Por exemplo, `require('/home/node/exemplo.js')`

Um módulo com prefixo `` ./ `` será buscado no caminho relativo ao módulo que chamou a função `require()` . Por exemplo, `require('./circulo.js')`.

Já um módulo com prefixo `` ../ `` será buscado na pasta superior relativa ao módulo que chamou a função `require()` .



Arquivo dos Módulos

Caso o módulo requisitado não contenha o prefixo `` / `` ou `` ./ `` para indicar a localização do arquivo o módulo será considerado no **módulo do núcleo** ou um **módulo instalado na pasta node_modules**, gerenciada pela NPM .

Agora se o caminho do arquivo não existir, `require()` vai emitir um **Erro** com sua propriedade `code` igual a **'MODULE_NOT_FOUND'** .

Módulos Salvos em Cache

Os módulos são salvos em Cache após a primeira vez que eles são carregados.

Isto significa, entre outras coisas, que toda chamada `require('modulo')` vai retornar exatamente o mesmo objeto retornado na primeira chamada, se ela fosse importar o mesmo arquivo.



Módulos Salvos em Cache (Exemplo)

```
//msg.js  
console.log("Exemplo Módulos Salvos em Cache");  
  
module.exports = function(msg){  
    console.log(msg);  
}
```



Módulos Salvos em Cache (Exemplo)

```
//index.js
```

```
//Módulos Salvos em Cache  
var msg = require('./msg');  
var msg2 = require('./msg');
```

```
msg("Boa noite");  
msg2("Bem vindo");
```

Global Objects



Universidade do Vale do itajaí (UNIVALI)
Centro de Ciência e Tecnologia da Terra e do Mar (CTTMar)
Curso de Ciência da Computação - Campus São José

Global Objects

Cuidado, evite **poluir** o escopo global

Exibindo os Global Objects

```
console.log(global);
```



Global Objects

```
//main.js
```

```
var max = 1000;  
var app = require('./total');  
  
console.log(app());
```

Global Objects

```
//total.js
```

```
module.exports = function(){  
    return 5 * max;  
}
```



Global Objects

```
global.max = 10000;  
GLOBAL.max = 10000;  
root.max = 10000;
```

Criando uma variável global utilizando **global**, **GLOBAL** e **root**



Process

Um processo é uma **instância de um determinado programa em execução no sistema operacional.**



Sistema de Módulos

process: Um objeto que é associado ao presente processo em execução.

process.argv: Um array contendo os argumentos de linha de comando.

- O primeiro elemento será node
- O segundo elemento será o nome do arquivo JavaScript,
- Os próximos serão todos os argumentos de linha de comandos adicionais, caso sejam atribuídos

[Documentação](#)



Object.keys

Retorna um arrays cujo os elementos são strings correspondentes para a propriedade enumerável diretamente sobre o objeto.

```
var arr = ['a', 'b', 'c'];
console.log(Object.keys(arr)); // console: ['0', '1', '2']
// array com objeto

var obj = { 0: 'a', 1: 'b', 2: 'c' };
console.log(Object.keys(obj)); // console: ['0', '1', '2']

// array como objeto com ordenação aleatória por chave
var an_obj = { 100: 'a', 2: 'b', 7: 'c' };
console.log(Object.keys(an_obj)); // console: ['2', '7', '100']
```

Process

Process faz parte das variáveis globais?

```
Object.keys(global).forEach(function(value){  
  
    console.log(value);  
  
})
```



Process

O que tem disponível dentro de Process?

```
Object.keys(process).forEach(function(value){  
  
    console.log(value);  
  
});
```



UNIVALI

Universidade do Vale do itajaí (UNIVALI)
Centro de Ciência e Tecnologia da Terra e do Mar (CTTMar)
Curso de Ciência da Computação - Campus São José

cmd C:\Windows\system32\cmd.exe

```
pid
features
_needImmediateCallback
execPath
debugPort
_startProfilerIdleNotifier
_stopProfilerIdleNotifier
_getActiveRequests
_getActiveHandles
reallyExit
abort
chdir
cwd
umask
_kill
_debugProcess
_debugPause
_debugEnd
hrtime
cpuUsage
dlopen
uptime
memoryUsage
binding
_linkedBinding
_setupDomainUse
_events
_rawDebug
_eventsCount
domain
_maxListeners
_fatalException
_exiting
assert
config
emitWarning
nextTick
_tickCallback
_tickDomainCallback
stdout
stderr
stdin
openStdin
exit
kill
argv0
mainModule
```

Process

```
// Printing to console
process.stdout.write("Hello World!" + "\n");

// Reading passed parameter
process.argv.forEach(function(val, index, array) {
  console.log(index + ': ' + val);
});

// Getting executable path
console.log(process.execPath);

// Platform Information
console.log(process.platform);
```

Desafio 1

1 - Crie um módulo chamado index (index.js), ele será o nosso ponto de partida e deverá ser invocado da seguinte forma:

```
node index.js
```

2 - Dentro do módulo index, crie uma função construtora (aquela que utiliza o operador new) chamada Carro, contendo as propriedades **marca**, **modelo**, **ano**, **cor**, **versao** e **preco**;

3 - Crie também um array de carros, incluindo alguns carros diretamente dentro do array.

4 - Por fim, utilizando a função forEach, percorra o array de carros exibindo cada um deles.

Desafio 2 - (Sistema de Módulos)

- 1 - Crie uma pasta chamada domain e lá crie um módulo chamado carro(carro.js).
- 2 - Mova a função construtora Carro para lá, exportando-a na forma de uma função.
- 3 - Crie uma pasta chamada data e lá crie um módulo chamado carros (carros.js)
- 4 - Mova o array de carros para lá, utilizando a função require para importar a função construtora Carro.
- 5 - Crie uma pasta chamada service e dentro crie um módulo chamado carrosService (carrosService.js).
- 6 - Por fim, no módulo index, faça o require do módulo carrosService e invoque a função exibirCarros.

Timers



Universidade do Vale do itajaí (UNIVALI)
Centro de Ciência e Tecnologia da Terra e do Mar (CTTMar)
Curso de Ciência da Computação - Campus São José

Timers

Permite a execução de código em intervalos de tempo especificados.

Os dois principais métodos para usar com JavaScript são:

`setTimeout(função, milissegundos)`

- Executa uma função, depois de esperar um número especificado de milissegundos

`setInterval(função, milissegundos)`

- O mesmo que `setTimeout()`, mas repete a execução da função continuamente



Timers

```
//setTimeout
```

```
console.log('A' + new Date());  
var test = setTimeout(function(){  
    console.log('B' + new Date())  
}, 3000); //ms
```


Timers

```
//setInterval  
  
setInterval(function(){  
    console.log('I' + new Date());  
},1000);
```



Timers - Como parar a execução?

O método `clearTimeout()` interrompe a execução especificada em `setTimeout()`

O método `clearInterval()` para as execuções da função especificada no método `setInterval()`



Timers

```
var a = setTimeout(function(){  
    console.log('timer 1' + new Date);  
},3000);
```

```
var b = setTimeout(function(){  
    console.log('timer 2' + new Date);  
},3000);
```

```
clearTimeout(a);
```



Timers

```
var a = setTimeout(function(){  
    console.log('timer 1' + new Date);  
},3000);
```

```
var b = setTimeout(function(){  
    console.log('timer 2' + new Date);  
},3000);
```

```
clearTimeout(a);
```



Timers

```
var interval = function(callback, time){  
    setTimeout(function(){  
        callback();  
        interval(callback,time);  
    }, time);  
};
```

```
Interval(function(){  
    console.log('R' + new Date());  
},1000);
```



Desafio 3

- 1 - No módulo index, faça a leitura do teclado e imprima tudo que é digitado utilizando o módulo **readline**.
- 2 - Crie uma pasta chamada infra e crie um módulo chamado teclado (teclado.js), movendo a função de leitura do teclado para o módulo teclado, recebendo um callback que será executado sempre que algo for digitado.
- 3 - No módulo index, utilize a função require para importar o módulo teclado.
- 4 - No módulo carrosService, crie uma função para exibirCarrosPorModelo, utilizando o que foi digitado para realizar a busca.



Desafio 3

5 - Modifique o módulo `index` para invocar a função `exibirCarrosPorModelo` sempre que alguma linha for digitada

6 - Adicione a possibilidade de digitar `/q` para sair, utilizando a função `process.exit()`

7 - Para dar a impressão que a busca está sendo realizada, faça com que a função `exibirCarrosPorModulo` seja invocada com um atraso de 1000ms



Dgram (UDP)



Universidade do Vale do itajaí (UNIVALI)
Centro de Ciência e Tecnologia da Terra e do Mar (CTTMar)
Curso de Ciência da Computação - Campus São José

Dgram (UDP)

UDP (User Datagram Protocol) é um protocolo não orientado a conexão.

Este protocolo é muito simples já que **não fornece controle de erros**.

Como criar um servidor UDP?



Dgram (UDP) - Server

```
var PORT = 33333;  
var HOST = '127.0.0.1';  
var dgram = require('dgram');  
var server = dgram.createSocket('udp4');  
server.on('listening', function () {  
  var address = server.address();  
  console.log('UDP Server listening on ' + address.address + ":" +  
address.port);  
});
```

Dgram (UDP) - Server

```
server.on('message', function (message, remote) {  
  
  console.log(remote.address + ':' + remote.port + ' - ' + message);  
  
});  
  
server.bind(PORT, HOST);
```



Dgram (UDP) - Client

```
var PORT = 33333;  
var HOST = '127.0.0.1';  
var dgram = require('dgram');  
var message = new Buffer('Boa tarde!');  
var client = dgram.createSocket('udp4');  
client.send(message, 0, message.length, PORT, HOST, function(err,  
bytes) {  
    if (err) throw err;  
    console.log('UDP message sent to ' + HOST + ':' + PORT);  
    client.close();  
});
```

Dgram (UDP) - Client

Vamos criar um **chat simples (UDP)**



Dgram (UDP)

Vamos criar um módulo que facilite a transmitir mensagens UDP.

Dgram (UDP)

```
//retornoserve.js
var dgram = require('dgram');
module.exports = function(host, port, msg){
  var message = new Buffer(msg);
  var client = dgram.createSocket('udp4');
  client.send(message, 0, message.length, port, host,
function(err, bytes) {
  if (err) throw err;
  //console.log('UDP message sent to ' + host + ':' + port);
  client.close();
});
}
```

Dgram (UDP)

Vamos criar um módulo para obter os dados
informados pelo usuário/cliente

Dgram (UDP)

```
//teclado.js  
  
var readline = require('readline');  
  
var rl = readline.createInterface({  
    input:process.stdin,  
    output:process.stdout  
});
```

Dgram (UDP)

```
var aoDigital = function (callback) {  
    rl.on('line', function(answer){  
        var linha = (answer) ? answer.toString() : '';  
        linha = linha.replace(/\n/, '');  
        if (linha) callback(linha);  
    });  
};  
module.exports = {  
    aoDigital: aoDigital  
};
```

Core Modules - net

É possível interagir com diversos **protocolos baseado em TCP**, tais como **HTTP, SMTP, FTP, POP3, IRC** e muitos outros

O TCP, Transmission Control Protocol, é um protocolo da camada de transporte, orientado à conexão e **responsável por controlar o processo de transmissão de dados**.

Trata problemas como a **perda, duplicação** e ainda garante a **ordenação** dos pacotes

Core Modules - net

Como criar um servidor TCP?



Universidade do Vale do itajaí (UNIVALI)
Centro de Ciência e Tecnologia da Terra e do Mar (CTTMar)
Curso de Ciência da Computação - Campus São José

Core Modules - net (serve.js)

```
var net = require ( 'net');

var server = net.createServer (function (connection) {
  console.log ( 'cliente conectado');
  connection.on ( 'end', function () {
    console.log ( 'cliente desconectado');
  });

  connection.write ( 'Ola Mundo');
});

server.listen (5555, function () {
  console.log ( 'servidor esta escutando');
});
```



Core Modules - net (client.js)

```
var net = require ( 'net');

var client = net.connect ({port: 5555 }, function () {
    console.log ( 'ligado ao servidor!');
});

client.on ( 'data', function (data) {
    console.log (data.toString ());
    client.end ();
});

client.on ( 'end', function () {
    console.log ( 'conexao encerrada pelo servidor');
});
```

Desafio 4

1 - Crie um **chat simples** utilizando o módulo **net** (TCP)

