



NodeJS e Express

Guilherme Ferreira

Find me anywhere
@guilheeeeeerme



Node.js

Succinctly

by Agus Kurniawan

Roteiro

- ◆ Introdução
- ◆ Javascript
- ◆ Arrays e JSON
- ◆ Funções
- ◆ Módulos
- ◆ Eventos
- ◆ ExpressJS
- ◆ Socket.io
- ◆ Banco de Dados
- ◆ Projeto com AngularJS e Mongo

1.

Introdução



Instalação

Windows

- ◆ Baixar o instalador no site oficial
- ◆ Instalar o git
- ◆ Instalar o Mongo

Instalação

Linux

- ◆ `curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -`
- ◆ `sudo apt-get install -y nodejs`
- ◆ `sudo apt-get install -y git`
- ◆ `sudo apt-get install -y build-essential`

Ferramentas de Desenvolvimento

Sugestões

- ◆ Sublime
- ◆ Atom
- ◆ WebStorm

2.

Javascript



Declaração de Variáveis

DOS

- ◆ Nome descritivo
- ◆ Camel Case
- ◆ Letra minúscula para variável
- ◆ Letra maiúscula para classes
- ◆ Maiúscula e com underline para constantes
- ◆ Comentar as mais importantes

DONT'S

- ◆ Nome sem sentido
- ◆ Criar variáveis sem 'var'
- ◆ Variável no contexto errado ou inútil

break, case, catch, continue
debugger, default, delete, do
else, finally, for, new
package, private, protected, public
function, if, implements, in
instanceof, interface, return, static
switch, this, throw, try
typeof, var, void, while
with

Não pode ser nome de variável !!!

Aritmética

Operadores

- ◆ soma: +
- ◆ subtração: -
- ◆ divisão: /
- ◆ multiplicação: *
- ◆ comparação:
 - ==, ===, !=, !==
 - <, <=, >, >=
- ◆ lógico
 - ||, &&, !

Biblioteca Math

- ◆ Math.ceil(a)
 - arredonda pra cima
- ◆ Math.floor(a)
 - arredonda pra baixo
- ◆ Math.random()
 - Aleatório de 0 a 1
- ◆ Math.max(a, b)
 - maior entre A e B
- ◆ Math.min(a, b)
 - Menor entre B e A
- ◆ ...

Condicional

IF

```
if(condition){  
    //do_something_a;  
}else {  
    //do_something_b;  
}
```

Switch

```
switch (option) {  
    case option1:  
        // do option1 job  
        break;  
    case option2:  
        // do option2 job  
        break;  
}
```

3.

Array e JSON

`.push()`

Arrays

Criação

// Criando vazio

```
var array = [];
```

```
var array = new Array();
```

// Criando com valores

```
var array = [3,5,12,8,7];
```

Arrays

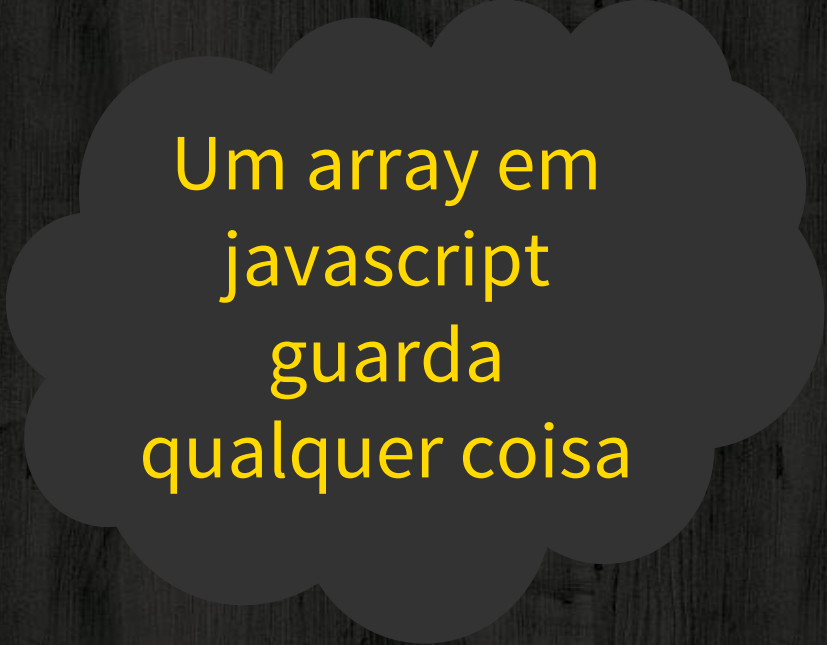
Inserindo

```
var array = [2,3,4];  
  
// inserir no final  
array.push(10);  
// [2, 3, 4, 10]  
  
// inserir no início  
array.unshift('Guilherme');  
// ['Guilherme', 2, 3, 4, 10]
```

Arrays

Inserindo

```
var array = [2,3,4];  
  
// inserir no final  
array.push(10);  
// [2, 3, 4, 10]  
  
// inserir no início  
array.unshift('Guilherme');  
// ['Guilherme', 2, 3, 4, 10]
```



Um array em
javascript
guarda
qualquer coisa

Arrays

Consultando

```
var array = ['a', 'b', 'c', new Date(), 10, [1,2,3] ];
```

```
// Manualmente
```

```
console.log(array[4]);
```

```
// Opção 1
```

```
for(var indice in array) {  
    console.log(indice, array[indice]);  
}
```

Arrays

Consultando

// Opção 2, estilo C

```
for(var indice = 0; indice < array.length; indice++){  
    console.log(indice, array[indice]);  
}
```

Arrays

Atualizando

```
var array = [2,3,4];
```

```
// atualizando o valor no índice 1
```

```
array[1] = ['a', 'b', 'c']
```

```
// [2, ['a', 'b', 'c'], 4]
```

Arrays

Removendo

```
var array = [0, 1, 2, 3, 4, 5];
```

```
// remover no final
```

```
var ultimo = array.pop();
```

```
// [0,1,2,3,4]
```

```
// remover no início
```

```
var primeiro = array.shift();
```

```
// [1,2,3,4]
```

Arrays

Removendo

```
var array = [0, 1, 2, 3, 4, 5];
```

```
// removendo em qualquer lugar
```

```
var valores = array.splice(3, 1);
```

```
// [0, 1, 2, 4, 5];
```

```
console.log(array, valores);
```

JSON

Criação

```
// Criando vazio  
var obj = {};
```

```
// Criando com valores  
var obj = {  
  name: 'Michael Z',  
  email: 'michael@email.com',  
  age: 35,  
  registeredDate: new Date()  
};
```



Conceito de
key-value

JSON

Inserindo/atualizando

```
var obj = {};
```

```
obj.name = 'guilherme';
```

```
// { 'name': 'guilherme' }
```

```
// dinamicamente
```

```
obj['idade'] = 23;
```

```
// { 'name': 'guilherme', 'idade': 23 }
```

JSON

Consultando

```
var obj = { /* criem um objeto */ };
```

```
// Manualmente
```

```
console.log(obj.<nome do campo>);
```

```
// Percorrendo todos
```

```
for(var key in obj) {  
    console.log(key, obj[key]);  
}
```

JSON

Removendo

```
var obj = { idade: 23 };
```

```
obj.idade = null;  
// { idade: null }
```

```
delete obj.idade;  
// {}
```

JSON

Extras

```
var obj = { idade: 23 };
```

```
// esse método checa se o campo existe
```

```
console.log(obj.hasOwnProperty('idade'));
```

```
console.log(obj.hasOwnProperty('name')); }
```

JSON

Extras

```
var obj = { idade: 23 };
```

```
// esse método checa se o campo existe
```

```
if (obj.idade){ /* o campo existe */ }
```

```
// checando se o campo NÃO existe no objeto
```

```
if (!obj.name){ /* se o campo não existe */ }
```

4.

Funções

```
function(){};
```

Funções

Uma função é um procedimento de JavaScript - um conjunto de instruções que executa uma tarefa ou calcula um valor. Para usar uma função, você deve defini-la em algum lugar no escopo do qual você quiser chamá-la.

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Functions>

Funções

(...) Uma função é um procedimento de JavaScript - um conjunto de instruções que executa uma tarefa ou calcula um valor (...)

```
var dataDeAgora = function() {  
  var agora = new Date();  
  console.log(agora);  
  return agora;  
}
```

```
function dataDeAgora () {  
  var agora = new Date();  
  console.log(agora);  
  return agora;  
}
```

O comando *return* é opcional

Funções

(...) Para usar uma função, você deve defini-la em algum lugar no escopo do qual você quiser chamá-la (...)

```
var oReturn = dataDeAgora()  
OU  
dataDeAgora()
```

A variável `oReturn` recebe o `return` da função. Caso não haja, recebe `undefined`

Funções

Passando parâmetros

```
/**  
 * Retorna um número entre min e max  
 */  
function getRandomArbitrary(min, max) {  
    return Math.random() * (max - min) + min;  
}  
  
console.log( getRandomArbitrary(0, 10) );
```

Funções

Lidando com assincronicidade (callbacks)

1. A função assíncrona

```
function buscarListaNoBanco(callback) {  
    /* ir buscar no banco */  
    /* quando voltar ...*/  
    callback(listaDoBanco);  
}
```

Funções

Lidando com assincronicidade (callbacks)

1. Usando a função assíncrona

```
buscarListaNoBanco(quandoaListaChegar);
```

```
function quandoaListaChegar(lista) {  
    // usar a lista  
}
```

5. Módulos

`.exports`

Definição

Um módulo encapsula e agrupa código similar em uma unidade atômica. Criar um módulo significa passar todo código de um determinado assunto para um único componente.

Módulos

Criando um módulo simples

```
// MyModule.js
```

```
function calculate (numA,numB){  
    return numA * numB + 10 * numB;  
}
```

```
exports.calculate = calculate;
```

Módulos

```
var add = function(numA,numB){  
    return numA + numB;  
}
```

```
var perform = function(){  
    // do something  
}
```

```
exports.add = add;  
exports.perform = perform;
```

Criando um módulo simples

... adicionando mais código

Módulos

```
// index.js
```

```
var myModule = require('./MyModule.js');
```

```
var result = myModule.calculate(20,10);
```

```
console.log(result);
```

Módulos (objetos)

```
// constructor  
var Account = module.exports = function() {  
    console.log('constructor');  
}  
  
Account.prototype.perform = function() {  
    console.log('perform');  
}
```

Módulos

```
var Account = require('./Account.js');
```

```
var account = new Account();
```

```
account.perform();
```

6. Eventos

.on () ;

Eventos

Começando

```
var EventEmitter =  
require('events').EventEmitter;  
var myEmitter = new EventEmitter;  
  
myEmitter.on('message', function(msg) {  
  console.log('message: ' + msg);  
});
```

Eventos

Módulo de Eventos

```
myEmitter.emit('message', 'this is  
the first message');
```

```
myEmitter.emit('message', 'this is  
the second message');
```

```
myEmitter.emit('message', 'welcome  
to nodejs');
```

Eventos

Once Event Listener

```
var EventEmitter = require('events').EventEmitter;
var myEmitter = new EventEmitter;
myEmitter.once('message', function(msg) {
  // do something
  console.log('message: ' + msg);
});

myEmitter.emit('message', 'this is the first message');
myEmitter.emit('message', 'this is the second message');
myEmitter.emit('message', 'welcome to nodejs');
```

Eventos

Removendo Eventos

```
function connection (id) {  
    // do something  
};
```

```
em.removeListener('connection', connection);
```

7.

ExpressJS

```
.get() ;
```

ExpressJS

Instalação

```
npm install express
```

ExpressJS

Criando Serviços

```
var express = require('express');  
var app = express();  
  
app.get('/', function(req, res){  
  res.send('Hello World Expressjs');  
});  
  
app.listen(8084);  
console.log('Server is running on port 8084');
```

ExpressJS

```
var express = require('express');
var app = express();

app.get('/', function(req, res){
    res.send('Hello World Expressjs');
});

app.get('/customer', function(req, res){
    res.send('customer page');
});

app.get('/admin', function(req, res){
    res.send('admin page');
});

app.listen(8084);
console.log('Server is running on port 8084');
```

ExpressJS

```
app.get('/admin', function(req, res){...
```

```
app.post('/admin', function(req, res){...
```

```
app.put('/admin', function(req, res){...
```

```
app.patch('/admin', function(req, res){...
```

```
app.delete('/admin', function(req, res){...
```

ExpressJS

Usar o body-parser para pegar o corpo das requisições.

```
npm install body-parser
```

ExpressJS

```
var express = require('express');
var app = express();
var bodyParser = require('body-parser')

app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: true }))

app.post('/admin', function(req, res){
  // req.body -> o copro
  res.send('post on admin');
});
```

ExpressJS

Servindo Pasta

```
var express = require('express');  
var app = express();  
  
// criando pasta publica  
app.use('/', express.static('view'))  
  
app.listen(8084);  
console.log('Server is running on port 8084');
```

8.

Socket.io

```
.emit();
```

Socket.io

Começando

```
npm install socket.io
```

Socket.io

```
var app = require('express')();  
var server = require('http').createServer(app);  
var io = require('socket.io')(server);  
  
io.on('connection', function(){ /* ... */ });  
  
server.listen(3000);
```

Socket.io

```
<script src='socket.io.js'></script>
<script>
  var socket = io('http://localhost:3000');
  socket.on('connect', function(){});
  socket.on('event', function(data){});
  socket.on('disconnect', function(){});
});
</script>
```

9.

Banco de Dados

```
.save() ;
```

Banco de Dados

Microsoft SQL Server,

<https://github.com/joyent/node/wiki/Modules#wiki-db-mssql>

PostgreSQL,

<https://github.com/joyent/node/wiki/Modules#wiki-db-pg>

MySQL,

<https://github.com/joyent/node/wiki/Modules#wiki-db-mysql>

Sqlite,

<https://github.com/joyent/node/wiki/Modules#wiki-db-sqlite>

Oracle,

<https://github.com/joyent/node/wiki/Modules#wiki-db-oracle>

NoSQL,

<https://github.com/joyent/node/wiki/Modules#wiki-db-nosql>

Banco de Dados

MongoDB

```
npm install mongoose
```

Banco de Dados

```
var mongoose = require('mongoose');  
mongoose.connect('mongodb://localhost/test');  
  
var Cat = mongoose.model('Cat', {  
  name: String  
});
```

Banco de Dados

```
var kitty = new Cat({ name: 'Zildjian' });
kitty.save(function (err) {
  if (err) {
    console.log(err);
  } else {
    console.log('meow');
  }
});
```