Certified Tech Developer
The Ultimate Degree

Grupo 5:

- Lucas Augusto Figueiredo Rosa
- Samir Libos
- Michel Picozzi

Objetos Literais

Como vimos, os objetos são uma das estruturas mais importantes da programação, tanto que há toda uma área dedicada à programação baseada neles, para entender um pouco mais por que eles são tão úteis e importantes, vamos realizar uma série de práticas para fortalecer melhor as particularidades e utilidades que vêm com eles.

Descrição do problema

A partir de um banco eles entram em contato conosco para criar um aplicativo que possa facilitar o manuseio de informações e facilitar as ações que ele precisa.
Nosso líder de tecnologia nos pede para pensar em como representar usuários, ou contas bancárias, em vez disso. Cada uma dessas contas tem as seguintes informações:

- Número da conta (somente números)
- Tipo de conta (conta corrente ou poupança em $)
- Saldo em $ (valor apenas)

- Titular da conta (nome completo)

Com essa informação em mente, somos solicitados a fazer o seguinte.

1. Pense na melhor forma de representar tais contas, e por quê? (discutir com a equipe sobre as diferentes estruturas e qual é a mais conveniente)

R: Representar com um tipo Objeto, onde:

- Número da conta (somente números) = **Número**
- Tipo de conta (conta corrente ou poupança em $) = **String**
- Saldo em $ (valor apenas) = **Número**
- Titular da conta (nome completo):
  i. Nome = **String**
  ii. Sobrenome = **String**

2. Uma vez decidido qual será o tipo de dados para representar as contas das contas, crie uma conta de teste e verifique se as propriedades mencionadas acima são criadas corretamente.

```
let Banco = {
    titular: 'Abigael Natte',
    // this.titular.sobreNome =
    numeroDaConta: 5486273622,
    tipoDaConta: 'Conta Corrente',
    saldo: 27771
};

console.log (Banco.titular)
```

3. Agora que sabemos como representar usuários e suas contas, o cliente nos forneceu uma lista de contas que devemos ser capazes de criar. Para isso, devemos gerar uma função construtora que facilite a criação das contas bancárias correspondentes mais rapidamente.

```
function Banco (titular, numeroDaConta, tipoDaConta, saldo) {
    this.titular = titular;
    // this.titular.sobreNome =
    this.numeroDaConta = numeroDaConta;
    this.tipoDaConta = tipoDaConta;
    this.saldo = saldo;
};
```

O líder tecnológico está muito feliz com nosso trabalho até agora. A essa altura já temos um banco de dados com 10 clientes do banco, provavelmente alojados em 10 variáveis diferentes (assumindo que cada variável é um objeto que foi criado com uma função construtora). A partir disso, eles nos pedem as seguintes características.

4. A partir dos 10 usuários, gere uma lista onde todos eles convergem (lista de objetos)

```
11  function Banco (titular, numeroDaConta, tipoDaConta, saldo) {
12      this.titular = titular;
13      // this.titular.sobreNome =
14      this.numeroDaConta = numeroDaConta;
15      this.tipoDaConta = tipoDaConta;
16      this.saldo = saldo;
17  };
18
19  let cliente1 = new Banco ('Abigael Natte',5486273622,'Conta Corrente',27771)
20  let cliente2 = new Banco ('Ramon Connell',1183971869,'Conta Poupança',8675)
21  let cliente3 = new Banco ('Jarret Lafuente',9582019689,'Conta Poupança',27363)
22  let cliente4 = new Banco ('Ansel Ardley',1761924656,'Conta Poupança',32415)
23  let cliente5 = new Banco ('Jacki Shurmer',7401971607,'Conta Poupança',18789)
24  let cliente6 = new Banco ('Jobi Mawtus',630841470,'Conta Corrente',28776)
25  let cliente7 = new Banco ('Thomasin Latour',4223508636,'Conta Corrente',2177)
26  let cliente8 = new Banco ('Lonnie Verheijden',185979521,'Conta Poupança',25994)
27  let cliente9 = new Banco ('Alonso Wannan',3151956165,'Conta Corrente',7601)
28  let cliente10 = new Banco ('Bendite Huggett',2138105881,'Conta Poupança',33196)
29
30  console.log (cliente1)
```

5. Também nos pedem a criação de um objeto literal chamado *banco* que terá uma propriedade chamada *clientes,* ele será composto pela lista de objetos gerados no ponto anterior.

```
47  function Banco (Titular, numeroDaConta, tipoDaConta, saldo) {
48      this.clientes = {
49          titular: {
50              nomeCompleto: Titular,
51              sobreNome: Titular.split(' ')[1],
52              nome: Titular.split(' ')[0],
53          },
54          numeroDaConta: numeroDaConta,
55          tipoDaConta: tipoDaConta,
56          saldo: saldo,
57      }
58  };
```

```
60  let banco = [{}]
61
62  let cliente = new Banco ('Abigael Natte',5486273622,'Conta Corrente',27771)
63  banco.push(cliente)
64  cliente = new Banco ('Ramon Connell',1183971869,'Conta Poupança',8675)
65  banco.push(cliente)
66  cliente = new Banco ('Jarret Lafuente',9582019689,'Conta Poupança',27363)
67  banco.push(cliente)
68  cliente = new Banco ('Ansel Ardley',1761924656,'Conta Poupança',32415)
69  banco.push(cliente)
70  cliente = new Banco ('Jacki Shurmer',7401971607,'Conta Poupança',18789)
71  banco.push(cliente)
72  cliente = new Banco ('Jobi Mawtus',630841470,'Conta Corrente',28776)
73  banco.push(cliente)
74  cliente = new Banco ('Thomasin Latour',4223508636,'Conta Corrente',2177)
75  banco.push(cliente)
76  cliente = new Banco ('Lonnie Verheijden',185979521,'Conta Poupança',25994)
77  banco.push(cliente)
78  cliente = new Banco ('Alonso Wannan',3151956165,'Conta Corrente',7601)
79  banco.push(cliente)
80  cliente = new Banco ('Bendite Huggett',2138105881,'Conta Poupança',33196)
81  banco.push(cliente)
```

6. o objeto *do banco* criará um método chamado *consultarCliente* que receberá um nome (titular) por parâmetro e deve pesquisar na lista de contas e retornar ao objeto do cliente que corresponde a esse nome inserido.

```
// ****************** tópico 6.
function Banco (Titular, numeroDaConta, tipoDaConta, saldo) {
    this.clientes = {
        titular: {
            nomeCompleto: Titular,
            sobreNome: Titular.split(' ')[1],
            nome: Titular.split(' ')[0],
        },
        numeroDaConta: numeroDaConta,
        tipoDaConta: tipoDaConta,
        saldo: saldo,

        consultarCliente: function (busca) {
            for (i=1; i<banco.length; i++) {
                if (banco[i].clientes.titular.nomeCompleto == busca) {
                    return i;
                }
            }
        }
    }
};
```

7. Crie outro método chamado *depósito* que receberá como parâmetros, o titular da conta e uma quantidade de dinheiro para depositar. O método deve chegar à conta correspondente e, em seguida, adicionar a quantidade de dinheiro para depositar o saldo da conta, então você deve dar um aviso pelo console com a mensagem *"Depósito realizado, seu novo saldo é: xxx"* .

```
142         consultarCliente: function (busca) {
143             for (i=1; i<banco.length; i++) {
144                 if (banco[i].clientes.titular.nomeCompleto == busca) {
145                     return i;
146                 }
147             }
148         },
149 
150         deposito: function (busca,valor) {
151             for (i=1; i<banco.length; i++) {
152                 if (banco[i].clientes.titular.nomeCompleto == busca) {
153                     banco[i].clientes.saldo += valor;
154                     console.log (banco[i].clientes.titular.nomeCompleto+ ", depósito realizado, seu novo saldo
155                     return banco[i].clientes.saldo;
156                 }
157             }
158         }
159     }
160 };
161 
```

8. Crie um último método chamado *saque* que também receberá dois parâmetros, o titular da conta e o valor a ser extraído. O método deve obter a conta correspondente e subtrair o valor do saldo atual. Caso o resultado seja inferior a 0, você deve imprimir uma mensagem através do console de *"Fundos insuficientes"*,caso contrário você deve imprimir *"Extração feita com sucesso, seu novo saldo é: xxx"*

```
207         deposito: function (busca,valor) {
208             for (i=1; i<banco.length; i++) {
209                 if (banco[i].clientes.titular.nomeCompleto == busca) {
210                     banco[i].clientes.saldo += valor;
211                     console.log (banco[i].clientes.titular.nomeCompleto+ ", depósito realizado, seu novo saldo
212                     return banco[i].clientes.saldo;
213                 }
214             }
215         },
216 
217         saque: function (busca,valor) {
218             for (i=1; i<banco.length; i++) {
219                 if (banco[i].clientes.titular.nomeCompleto == busca) {
220                     banco[i].clientes.saldo -= valor;
221                     console.log (banco[i].clientes.titular.nomeCompleto+ ", extração feita com sucesso,, seu n
222                     return banco[i].clientes.saldo;
223                 }
224             }
225         }
226     }
227 };
228 
```

Se você chegou aqui parabéns, a equipe de desenvolvimento e o líder técnico estão impressionados com o seu trabalho!

Bônus extra

Para que você não fique com o desejo ou que você possa continuar praticando se quiser, propomos um pouco (ou algum) exercício mais, tenha em mente que daqui os exercícios podem subir em dificuldade; como sempre dizemos, paciência, ignore a complexidade e tente resolvê-la com as ferramentas que você tem à sua disposição, você também pode procurar informações extras no google ou documentações que você conhece!!

Propriedade Única

Você deve criar uma função chamada *propriedadeUnica* que recebe uma array de objetos como parâmetro e uma string. Você deve devolver um novo array de objetos, tendo apenas a propriedade que foi passada como string.

exemplo:

let array = [{ nome: "Lean", idade: 27 }, { nome: "Eze", idade: 49}]

propriedadeUnica(array, "idade") deve retornar [{ idade: 27 }, { idade: 49 }]

propriedadeUnica(array, "nome") deve retornar [{ nome: "Lean"}, { nome: "Eze" }]

Calculadora de notas

Crie o objeto *do aluno,*que consistirá nas seguintes propriedades básicas:

- número
- Número do arquivo
- Lista de notas

Gostaríamos de calcular a média do aluno e se ela é aprovada com base em uma nota de aprovação que será dada. Para este exercício vamos deixar você pensar sobre todos os métodos e propriedades que podem ser necessários para que nosso programa funcione corretamente da maneira solicitada.