```
//FUNÇÕES FACTORY:
//Como o próprio nome sugere, funções factory são funções que tem o objetivo de fabricar. Fabricar o que? Fabricar objetos á partir d
a função, como se fosse uma classe ou funções construtoras.
//OBSERVAÇÕES: As funções factory possuem uma vantagem em relação as classes e funções construtoras criadas em Javascript, pois as cl
asses e funções construtoras precisam de ser instanciadas para que possam ser chamadas. Enquanto as factory não, podemos armazenar se
us valores dentro de variáveis sem a necessidade de instanciar...

//CRIANDO FUNÇÃO FACTORY SIMPLES:
function factory(nome, preco, descricao) { //Perceba que a criação de função factory é muito simples, nós a criamos assim como criamo
s uma função normal...
    return { //O que a identifica como uma factory é o retorno de um objeto...
        nome, //Repare que em nenhum momente usamos o "this" para instanciar os objetos que serão criados com á partir da função fact
ory...
        preco,
        desconto: 0.10, //Podemos também delimitar valores padrão, seja no objeto em si ou no campo de parâmetro...
        prodDescricao: () => `Descrição do produto: ${descricao}` //Perceba que podemos até colocar funções como se fossem métodos de
ntro do nosso objeto que para que sejam retornados por através da chamada do "método"....
    }
}

const notebook = factory('notebook', 2199.00, 'String qualquer que descreve o notebook'); //Aqui está o momento chave da criação de u
m objeto usando factory, veja que usamos só uma variável e atribuímos a ela a própria chamada do factory, passando os valores que des
ejamos armazenar...
const ipad = factory('ipad', 1999.00, 'String qualquer que descreve o ipad')

console.log("1) Objeto notebook criado com factory:")
console.log(notebook); //Veja o objeto completo apenas chamando pela variável...
console.log(notebook.prodDescricao()) //Podemos chamar uma função dentro do objeto como se fosse método...
console.log(typeof notebook); //Veja que realmente se trata de um objeto, como se fosse uma classe...

console.log("\n2) Objeto ipad criado com factory:")
console.log(ipad);
console.log(ipad.prodDescricao())
console.log(typeof ipad);
```

```
//COMPARANDO FACTORY COM CLASSES E FUNÇÕES CONTRUTORAS:
//Perceba logo abaixo que criamos uma classe como é comum em Javascript...
class Pessoa {

    constructor(nome) { //Na criação de classes é necessário o uso de um constructor para demarcar os atributos da classe...
        this.nome = nome; //E sempre usamos o this para referenciar a instância dáquela classe...
    }

    falar() {
        return `Olá meu nome é ${this.nome}`; //Nossos métodos também devem conter o "this" para instanciar corretamente o objeto...
    }

}

const joao = new Pessoa('João'); //instanciamos o joão á partir da classe a qual ele pertence...

console.log("\n3) Pessoa instanciada á partir de uma Classe:");
console.log(joao.falar());

//Classe Construtora em Javascript:
function PessoaConstrutora(nome) { //Criamos a mesma Classe, mas aqui como se fosse uma função construtora...

    this.nome = nome;//Também usamos "this" para instanciar ao objeto da classe...

    this.saudar = () => `Olá meu nome é ${this.nome}`; //Usamos também o "this" no método público...
}

const patricio = new PessoaConstrutora('Patrício'); //Para gerar um objeto do método construtor tivemos que instânciá-
lo por atravé da plavra reservada "new"...
console.log("\n4) Pessoa instanciada á partir de uma Função Construtora:");
console.log(patricio.saudar());

//Função Factory identica da Classe e Função Construtora anterior:
function PessoaFactory(nome) { //Perceba como a sintaxe de uma função factory é bem mais simples...
    return {
        apresentar: () => `Olá meu nome é ${nome}` //Lembre-se que ela não precisa de usar o this...
```

```
    }
}

const roberto = PessoaFactory('Roberto');
console.log("\n5) Pessoa instanciada á partir de uma Função Fábrica:");
console.log(roberto.apresentar()); //Por não usar o this a chamada de um objeto pertencente a uma função factory é sempre detro do co
ntexto do objeto, isso já é implícito, diferente das classes e funções construtoras, onde o this pode fazer com que o contexto da fun
ção varie...
```

RESULTADO NO CONSOLE...

```
[Running] node "c:\Users\User\Documents\JAVASCRIPT\ARQUIVOS_DAS_AULAS\073-Funcoes_Factory.js"
1) Objeto notebook criado com factory:
{
  nome: 'notebook',
  preco: 2199,
  desconto: 0.1,
  prodDescricao: [Function: prodDescricao]
}
Descrição do produto: String qualquer que descreve o notebook
object

2) Objeto ipad criado com factory:
{
  nome: 'ipad',
  preco: 1999,
  desconto: 0.1,
  prodDescricao: [Function: prodDescricao]
}
Descrição do produto: String qualquer que descreve o ipad
object

3) Pessoa instanciada á partir de uma Classe:
Olá meu nome é João

4) Pessoa instanciada á partir de uma Função Construtora:
```

```
Olá meu nome é Patrício

5) Pessoa instanciada á partir de uma Função Fábrica:
Olá meu nome é Roberto

[Done] exited with code=0 in 0.249 seconds
```