```
//OBJECTS:
//Objetos em Javascript são elementos que possuem um conjunto de chave e valor, onde as chaves servem como um identificador para um atributo daquele objeto, enquanto os valores são os atributos em si.
//OBSERVAÇÕES: No Javascript existem 2 tipos de object, para mais detalhes veja a aula 039;



//DECLARANDO OBJECT LITERALMENTE:
const prod1 = {}; //A declaração é feita á partir de um par de chaves...
console.log('1) ',typeof prod1);



//DECLARANDO OBJECTS LITERALMENTE COM AS NOVAS REGRAS DO ECMASCRIPT 2015:
//Com as novas regras do ecmascript, se tivermos um valor atrelado a uma variável e desejarmos jogar esses valores dentro de object, tendo como chave o mesmo nome da variável, não precisamos escrever o conjunto chave valor, podemos abreviar colocando somente os nomes das chaves, da seguinte forma:
const a = 1
const b = 2
const c = 3

const objAntigo = {a: 1, b:2, c:3} //Essa era a forma antiga, tínhamos que escrever chave e valor...
const objEcma = {a, b, c} //Perceba que colocamos somente os nomes das variáveis, e isso já basta...

console.log('\n2)',objAntigo)
console.log('2)',objEcma) //Perceba que mesmo sem termos explicitado o nome da chave, automáticamente o objeto pegou o nome da chave e o valor...



//ATRIBUINDO CHAVE E VALOR DINÂMICAMENTE:
prod1.nome = 'Celular Ultra Mega Power'; //perceba que criamos uma chave "nome" e atribuímos um valor para ela...
prod1.preco = 4998.90;
console.log('\n3) ',prod1); //Veja na impressão que temos conjuntos de par chave valor...
```

```
//ATRIBUINDO CHAVES COM ESPAÇO ENTRE AS PALAVRAS:
prod1['Desconto legal'] = 0.40; //Perceba que, para colocar palavras com espaçamento como chave, tivemos que colocar uma string
dentro de colchetes.
console.log('\n4) ', prod1);



//DECLARANDO OBJECT E ATRIBUINDO CHAVE A VALOR LITERALMENTE:
const prod2 = { //Perceba que abrimos a chave de dentro colocamos várias chaves e valores separados por vírgula
    nome: 'Camisa Polo', //Perceba que colocamos : para fazer atribuição
    preco: 79.90,
    estilo: { //Podemos ter objetos dentro de objetos, desde que eles estejam resguardados por chaves...
        cor: 'Vermelha',
        tamanhos: { //Podemos ter objetos dentro de objetos, seguindo o mesmo estilo de chave e valor...
            grande: 'G',
            media: 'M',
            pequeno: 'P'
        }
    }
}
    console.log('\n5) ',prod2);



//ATRIBUIÇÃO DE NOME DE CHAVE USANDO COLCHETES COM NOVO FORMATO DO ECMASCRIPT 2015:
//O EcmaScript 2015 possibilitou a criação de chaves á partir de valores existentes dentro de variáveis, podemos fazer isso usando os
colchetes, da seguinte forma:
const chaveEcma = 'Chave'
const valorEcma = 'Valor'

const objEcmaCol1 = {} //Perceba que já temos o objeto criado...
objEcmaCol1[chaveEcma] = valorEcma //Veja como usamos os colchetes para atribuir o valor da variável "chaveEcma" e usando diretamente
o nome da variável já pegamos o valor dela...
console.log('\n6)', objEcmaCol1)

//Podemos também usar esse formato...
```

```
const objEcmaCol2 = {[chaveEcma]: valorEcma} //Perceba que usando apenas uma linha conseguimos pegar o valor da chave e ainda o valor da variável...
console.log('6)', objEcmaCol2) //Veja que o resultado é o mesmo...



//RESGATANDO VALORES DE UM OBJECT:
console.log('\n7)');
console.log('Nome do produto 1: ', prod1.nome); //Para resgatar o valor chamamos pela chave...
console.log('Preço do produto 1: ', prod1.preco);
console.log('Desconto do produto 1: ', prod1["Desconto legal"]); //Se a chave tem nome composto usamos os colchetes...
console.log('\nNome do produto 2: ', prod2.nome);
console.log('Preço do produto 2: ', prod2.preco);
console.log('Cor do produto 2: ', prod2.estilo.cor); //Se a chave possuí uma subchave usamos a notação ponto mais uma vez...
console.log('Tamanhos do produto 2: ', prod2.estilo.tamanhos); //Chamando todos os valores de uma sub-chave...



//DECLARANDO UM OBJECT USANDO A FUNÇÃO OBJECT:
let produto = new Object; //Veja que podemos usar a função interna Object do javascript para criar objetos instânciando variáveis por através dessa função.
console.log(`\n8) ${typeof produto}`);



//CRIANDO OBJECTS COM ESTRUTURAS MAIS COMPLEXAS:
const carro = { //Veja que temos um carro de uma seguradora com uma série de informações dentro de um único objeto...
    modelo: 'A4',
    valor: 89000,
    proprietario: { //Dentro de um objeto colococamos outro objeto com informações do proprietário...
        nome: 'Raul',
        idade: 56,
        endereco: { //Algumas informações do proprietário também são objetos...
            logradouro: 'Rua ABC',
            numero: 123
        }
    },
```

```
    condutores: [{ //Veja que possuímos uma chave que contém um array com objetos dentro dele...
        nome: 'Júnior',
        idade: 19
    }, {
        nome: 'Ana',
        idade: 42
    }],
    calcularSeguro: function () { //Podemos ter até mesmo funções dentro dos nossos objetos...
        //...
    }
}

carro.proprietario.endereco.numero = 1000; //Veja que em casos onde o objeto é muito grande temos que fazer todo o caminho para
acessar...
carro['proprietario']['endereco']['logradouro'] = 'Av. Gigante'; //Podemos acessar também usando os colchetes e os nomes das chaves
em formato de string...

console.log(`\n9)`, carro)



//INSERINDO FUNÇÕES EM OBJECTS COM O NOVO ECMASCRIPT 2015:
const objFuncEcma = {
    funcao: function() {/*...*/} //Essa é a forma convencional de adicionar funções...
    ,
    funcao2() {/*...*/} //Essa é a forma nova de adicionar funções no ECMASCRIPT 6, o próprio nome da função já é a sua chave...
}

console.log('\n10)', objFuncEcma)



//ATENÇÃO COM O USO DO DELETE:
//Quando o usamos o delete em um object apagamos para sempre elementos de dentro de um objeto, e se estes elementos estiverem
encadeados, vamos acabar apagando tudo o que estiver encadeado á partir do ponto onde delimitamos o delete...
delete carro.proprietario.endereco
console.log(`\n11)`, carro) //Veja que o endereço foi deletado...
```

```
delete carro.calcularSeguro
console.log(`\n12)`, carro) //Agora a função foi deletada...



//POR QUE CONSEGUIMOS MUDAR OS VALORES DE OBJETOS MESMO ELES ESTANDO ARMAZENADOS EM CONSTANTES:
//Você deve ter percebido que, mesmo quando criamos objetos em constantes ainda assim é possível alterar os valores de suas chaves.
Como no exemplo abaixo:
const pessoa = { nome: 'Gabriel', idade: '30'}
pessoa.nome = 'Graziela' //Veja que mudamos o nome apesar do objeto estar armazenado numa constante...
console.log(`\n13) ${pessoa.nome} ${pessoa.idade}`)

//isso acontece por que as constante guardam na verdade um endereço de memória, quando referenciamos ao objeto pessoa, ele na verdade
é um endereço de memória que contém uma coleção de dados. Podemos mudar a coleção de dados se quisermos, mas não podemos mudar o
endereço de memória jamais...
pessoa = {nome: 'Gabriel'} //Perceba que temos um erro quando tentamos dar novos valores ao espaço de memória...



//CRIANDO CONSTANTES REALMENTE CONSTANTES:
//Podemos criar constantes que nunca poderão alterar suas chaves ou os valores delas usando um método built-in chamado "freeze()"...
const humano = Object.freeze({nome: 'Gabriel', idade: '30'}) //veja que freeze() deve ser utilizado sempre com conjunto com a função
"Object", para que possamos criar um objeto que jamais poderá ser modificado...

humano.nome = 'Graziela' //Veja que embora tenhamos mudado o nome como no exemplo mais acima, com o freeze() o nome sempre será igual
ao que foi declarado desde o ínicio...
console.log(`\n14) ${humano.nome} ${humano.idade}`)

delete humano.nome //Não podemos nem mesmo deletar nenhum valor...
console.log(`\n15)`, humano)
//O que for criado com freeze() só poderá ser deletado diretamente no código fonte...
```

RESULTADO NO CONSOLE...

```
[Running] node "c:\Users\Almoxarifado\Documents\JAVASCRIPT\arquivos_das_aulas\034-manipulando_Objects.js"
```

```
1)  object

2) { a: 1, b: 2, c: 3 }
2) { a: 1, b: 2, c: 3 }

3)  { nome: 'Celular Ultra Mega Power', preco: 4998.9 }

4)  {
  nome: 'Celular Ultra Mega Power',
  preco: 4998.9,
  'Desconto legal': 0.4
}

5)  {
  nome: 'Camisa Polo',
  preco: 79.9,
  estilo: {
    cor: 'Vermelha',
    tamanhos: { grande: 'G', media: 'M', pequeno: 'P' }
  }
}

6) { Chave: 'Valor' }
6) { Chave: 'Valor' }

7)
Nome do produto 1:  Celular Ultra Mega Power
Preço do produto 1:  4998.9
Desconto do produto 1:  0.4

Nome do produto 2:  Camisa Polo
Preço do produto 2:  79.9
Cor do produto 2:  Vermelha
Tamanhos do produto 2:  { grande: 'G', media: 'M', pequeno: 'P' }

8) object
```

```
9) {
  modelo: 'A4',
  valor: 89000,
  proprietario: {
    nome: 'Raul',
    idade: 56,
    endereco: { logradouro: 'Av. Gigante', numero: 1000 }
  },
  condutores: [ { nome: 'Júnior', idade: 19 }, { nome: 'Ana', idade: 42 } ],
  calcularSeguro: [Function: calcularSeguro]
}

10) { funcao: [Function: funcao], funcao2: [Function: funcao2] }

11) {
  modelo: 'A4',
  valor: 89000,
  proprietario: { nome: 'Raul', idade: 56 },
  condutores: [ { nome: 'Júnior', idade: 19 }, { nome: 'Ana', idade: 42 } ],
  calcularSeguro: [Function: calcularSeguro]
}

12) {
  modelo: 'A4',
  valor: 89000,
  proprietario: { nome: 'Raul', idade: 56 },
  condutores: [ { nome: 'Júnior', idade: 19 }, { nome: 'Ana', idade: 42 } ]
}

13) Graziela 30

14) Gabriel 30

15) { nome: 'Gabriel', idade: '30' }
```

```
[Done] exited with code=0 in 0.157 seconds
```

RESULTADO QUANDO TENTAMOS MODIFICAR O VALOR DE MEMÓRIA DE UM OBJECT CONSTANTE...

```
c:\Users\Almoxarifado\Documents\JAVASCRIPT\arquivos_das_aulas\034-manipulando_Objects.js:149
pessoa = {nome: 'Gabriel'} //Perceba que temos um erro quando tentamos dar novos valores ao espaço de memória...
       ^

TypeError: Assignment to constant variable.
    at Object.<anonymous> (c:\Users\Almoxarifado\Documents\JAVASCRIPT\arquivos_das_aulas\034-manipulando_Objects.js:149:8)
    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 0.106 seconds
```