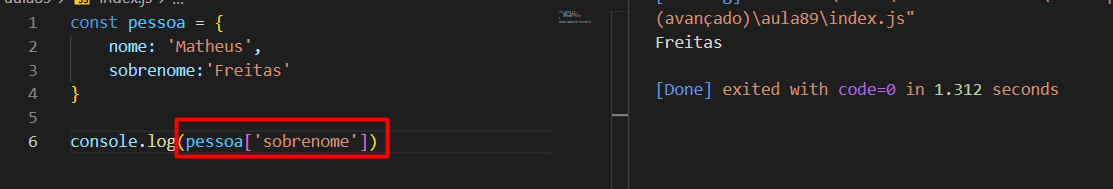
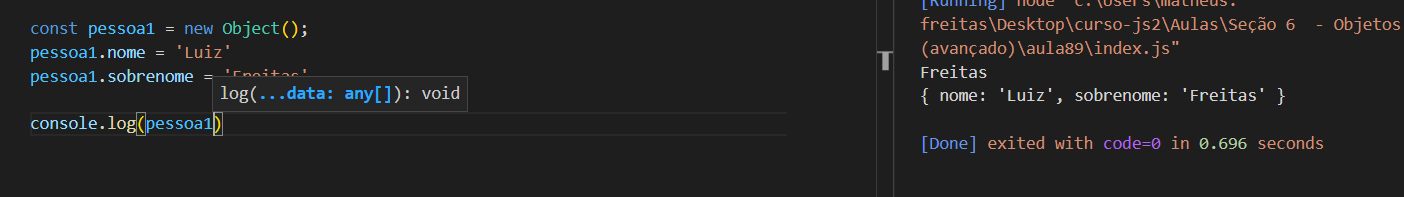
**AULA 89**

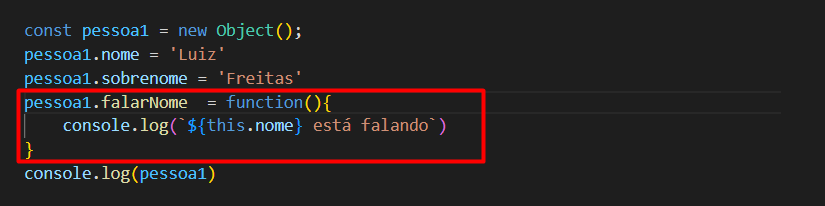
* Revisão de objetos
* Usasse as chaves para criar um objeto literal
* É formada por chaves, pares e valores
* Podemos acessar uma chave de um objeto utilizando colchetes



* Com os colchetes, podemos utilizar chaves dinâmicas (chaves armazenadas em variáveis que podem mudar de valor)
* Existem construtores de objetos



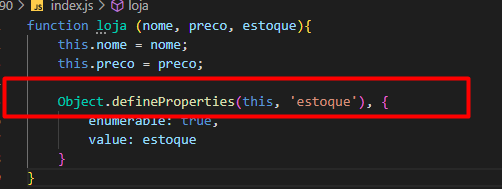
* Podemos excluir uma chave utilizando o comando delete
* Podem ter métodos (funções dentro de objetos ou array)



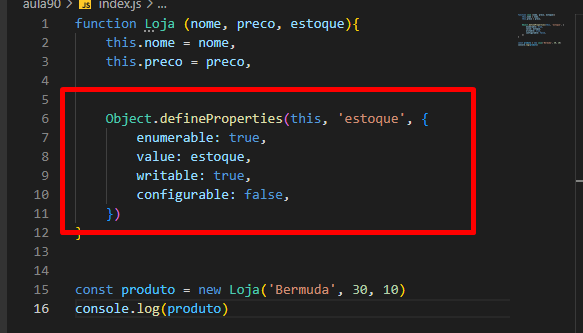
* Podemos utilizar o comando for para ler as chaves de um objeto
* Podemos utilizar funções fabricas e funções construtoras para criar objetos
* Object.freeze bloqueia alterações em um objeto ou array

**AULA 90**

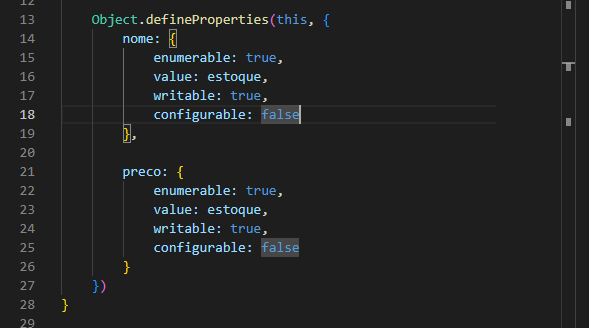
* Object.define.Property() e Object.defineProperties()
* São duas funções que são relacionadas as chaves dos objetos
* Utilizadas para congelar alguns valores de objetos
* Vimos um execicio onde criamos uma função construtora, que possui três chaves
  1. Nome
  2. Preço
  3. Estoque
* Utilizamos a propriedade Object.defineProperty para criar o atributo do estoque. Isso faz com que a propriedade estoque não seja exibida quando realizarmos alguma ação com objeto loja



* Criamos um Property Descrition que é um objeto que pode receber diversos valores (representado pelos colchetes na imagem acima) e utilizamos algumas propriedades:
  1. Enumerable – Torna o atributo visível para ser utilizado em outros métodos
  2. Value – Atribui o nome da chave
  3. Writable – Bloqueia ou permite a alteração do valor da chave
  4. Configurable – Bloqueia ou permite que a chave seja reescrita

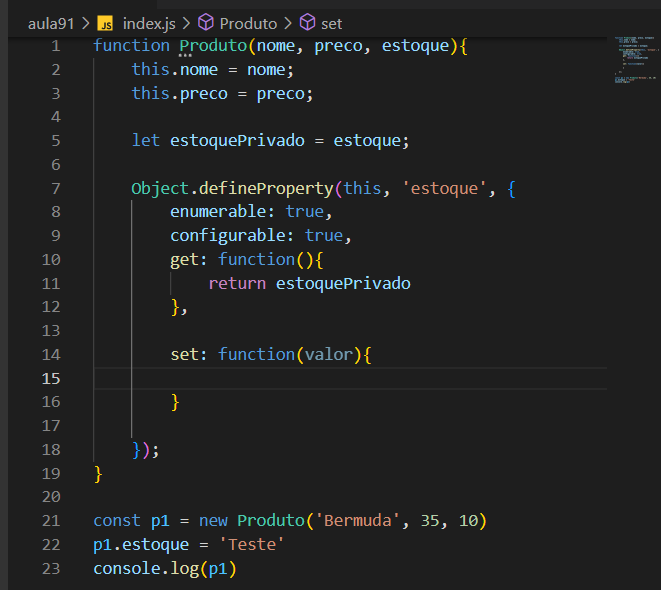


* Podemos utilizar métodos para atribuir o valor de value
* Object.defineProperties é utilizado para criar mais de uma chave
* Diferente do exemplo acima, esse recebe apenas os this como parâmetros e abre chaves de um objeto, onde cada “Chave” desse objeto na verdade é um outro objeto. Exemplo:

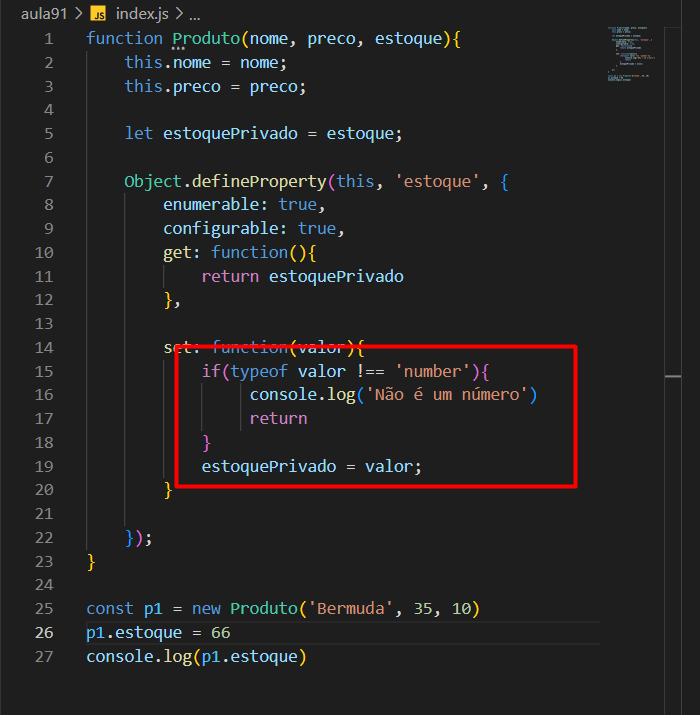


**AULA 91**

* Getters e setters
* É uma maneira de proteger alguma propriedade
* **Cri**amos um execicio onde criamos um objeto utilizando uma função construtora, utilizando o object.defineProperty para criar uma de suas chaves
* Criamos um método get para retornar o valor da chave e um método set para podermos modificar o valor da chave

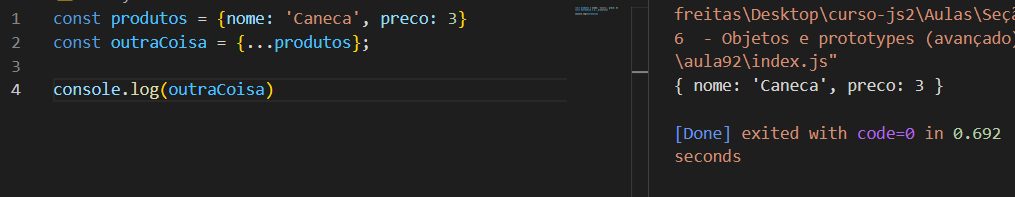


* Utilizamos o método If para validar se o valor atributo para a chave é um numero ou não

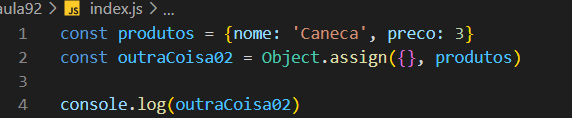


**AULA 92**

* Métodos uteis para objetos
* Objetos são valores passados por referencia
* Podemos copiar os valores de um objeto para outro, para podermos manipularmos os valores sem alterar o valor inicial
* Podemos copiar utilizando o spred operation (...)



* Podemos também copiar os valores de um objeto utilizando objetc.assign{}. Para isso, criamos uma variável onde é atribuída o método, dentro do método criamos abrimos e fechamos colchete e indicamos qual objeto desejamos copiar. Exemplo:



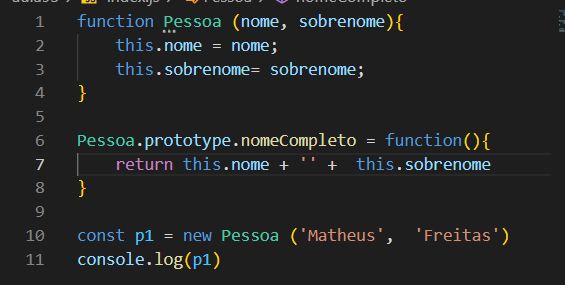
* Objetc.keys vai retornar as chaves de um objeto
* Object.freeze congela o valor do objeto
* Object.defineProperties definem chaves
* Object.defineProperty define uma chave
* O método objetc.getOwnPropertyDescriptor mostra a configuração de um objeto. Exemplo:



* Object.value retorna os valores das chaves de um objeto
* Objetc.entries retorna as chaves e os valores das chaves de um objeto em arrays

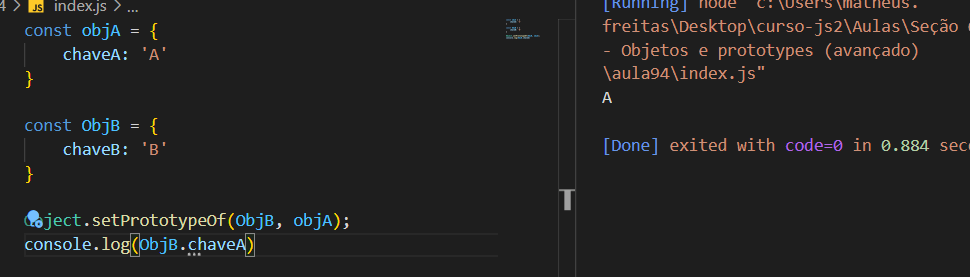
**AULA 93**

* Prototypes
* É a base que o JavaScript utiliza para trabalhar com heranças
* Quando se utiliza um método dentro de um objeto, o objeto primeiro vai verificar se aquele método está dentro dele, depois irá verificar se está fora em outro local do código
* Quando criamos um objeto através de uma função construtora, automaticamente o JavaScritp linka ao objeto \_proto\_
* \_proto\_ armazena alguns métodos dento de si, podendo ser acessado por mais de uma instancia criada a partir da mesma função, isso nos possibilita a economizar recurso na hora da exeução do software, pois assim, podemos criar um método dentro do prototype que será acessado por todos, ao invés de criar uma função por instancia
* Existe uma cadeia de acesso de recursos das funções
* Criamos um execicio onde criamos um método dentro do prototype da função

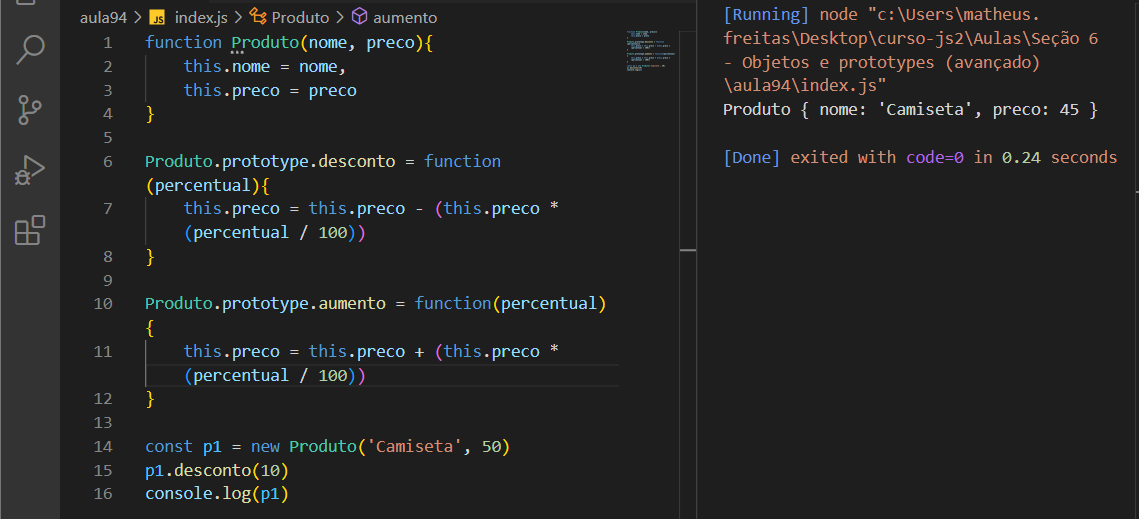


**AULA 94**

* Manipulação de protype
* Com o comando object.setPrototypeOf conseguimos alterar o objeto protótipo de um outro objeto, possibilitando acessar recursos do primeiro objeto



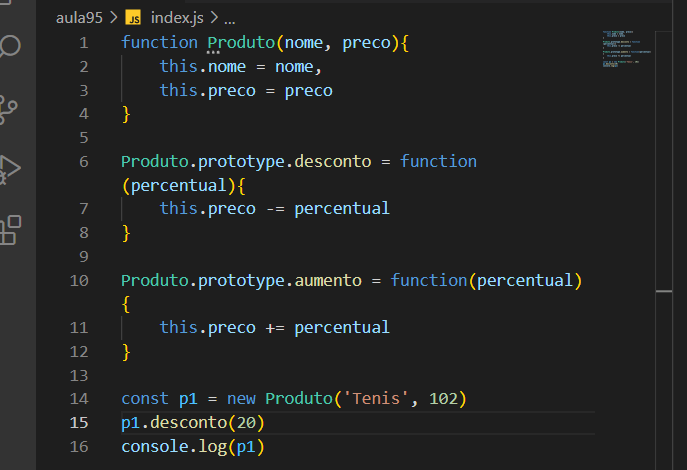
* Criamos um execicio onde, temos uma função de produtos e criamos métodos dentro do prototype para realizar o calculo desconto e aumento do preço



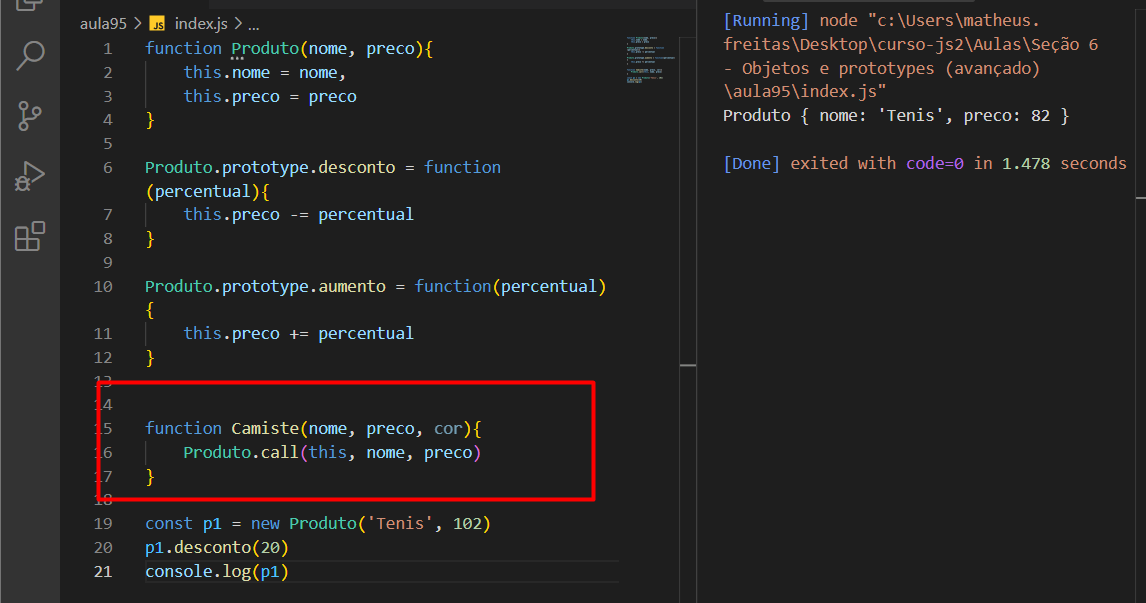
* Podemos utilizar o comadno object.create para atribuir o prototype de outro objeto no momento da criação do objeto

**AULA 95**

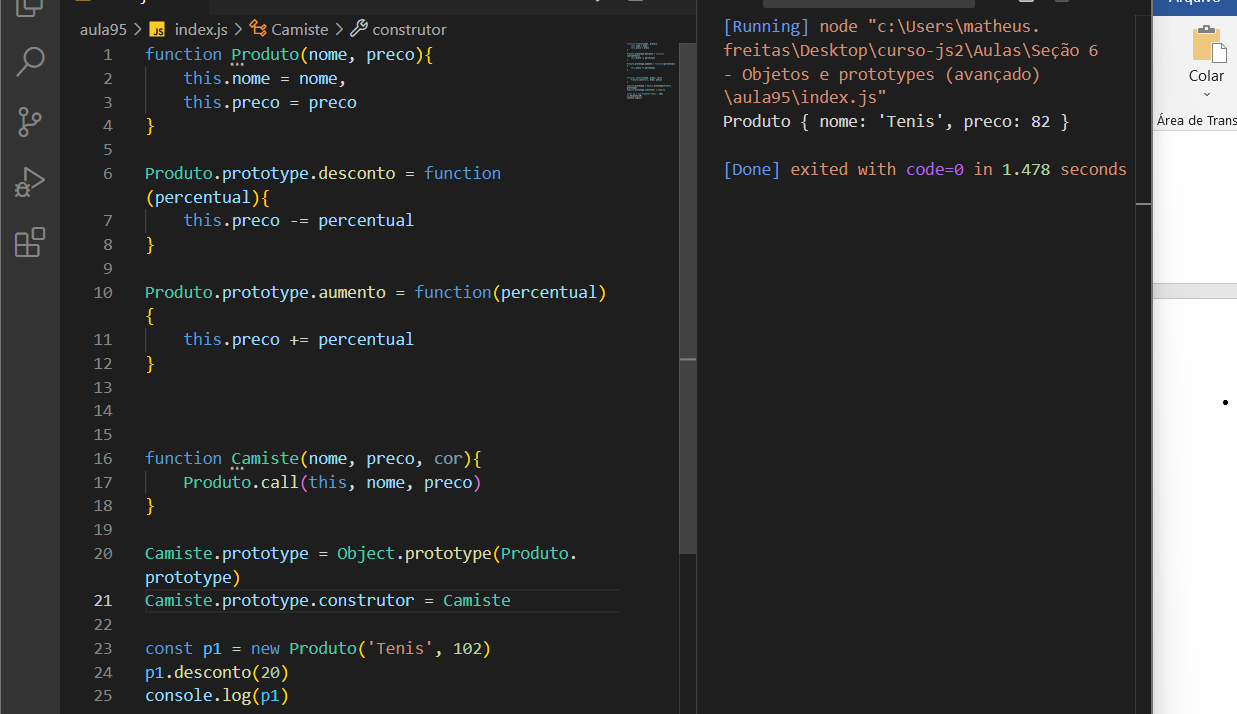
* Herança
* Vimos um exercício de especialização de função
* Criamos uma função construtora para a criação de produtos
* Nessa função, também criamos métodos no seu prototype para aumento e desconto no preço do produto



* Criamos uma função para a construção de um produto especifico, utilizando a função Produto que já possui algumas chaves que vamos utilizar utilizando o método .call

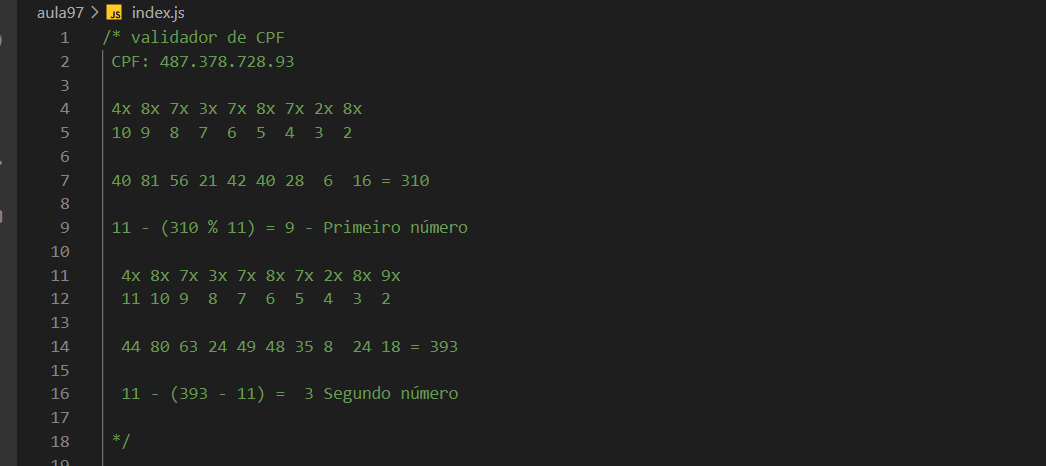


* Linkamos o prototype do produto a função construtora Produto, atribuindo o valor ao objeto vazio que referenciamos ao outro prototype



**AULA 97**

* Exercício validador de CPF
* Para validar um CPF precisamos verificar os dois últimos números
* O penúltimo número, é resultado da soma da multiplicação dos outros números, onde o primeiro número é multiplicado por 10, o segundo 9, o terceiro 8, até chegar ao número 2

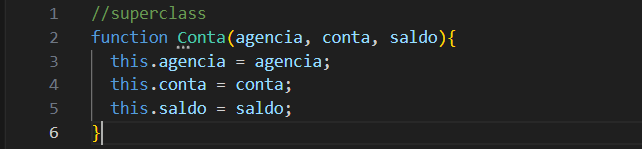


**AULA 98**

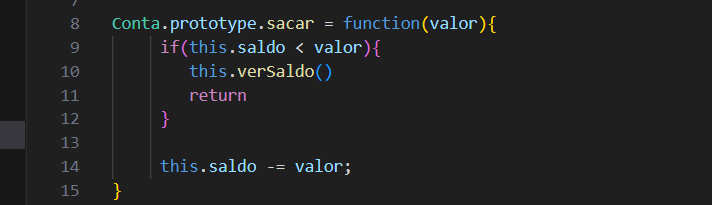
* Resolução do exercício da aula 97

**AULA 99**

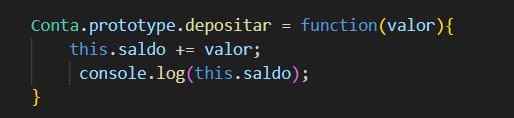
* Aula sobre Polimorfismo
* Possibilita um método se comportar de maneira diferente dentro de um mesmo objeto
* Classe = Função Construtora
* Criamos uma função construtora (Que na programação orientada a objetos, seria chamada de superclass) que representa uma conta bancaria. Como parâmetros, a função recebe ‘’agencia’’, ‘’conta’’ e ‘’saldo’’. Dentro da função criamos as variáveis utilizando os parâmetros



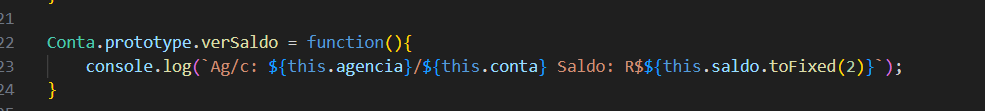
* Criamos três métodos dentro do prototype:
  1. Método para sacar algum valor



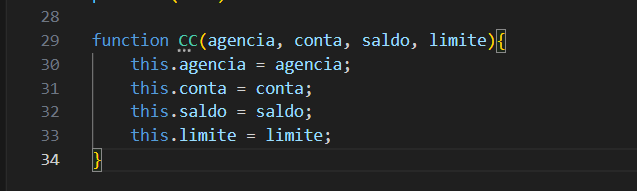
* 1. Método para depositar algum valor



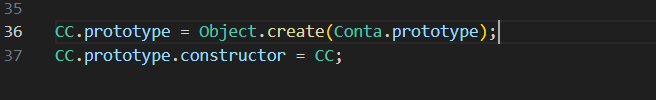
* 1. Método para visualizar o saldo



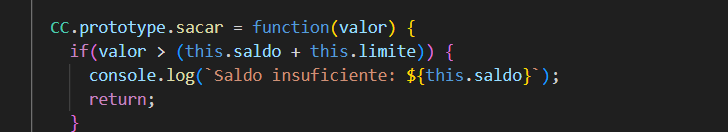
* Criamos uma função construtora que cria contas correntes



* Realizamos a conexão dos prototypes das duas funções

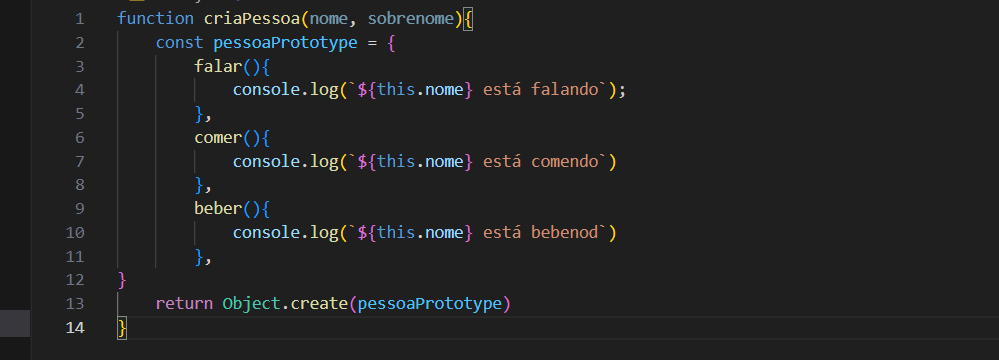


* Reescrevemos a função de sacar dinheiro, para ela se comportar de maneira diferente com contas correntes



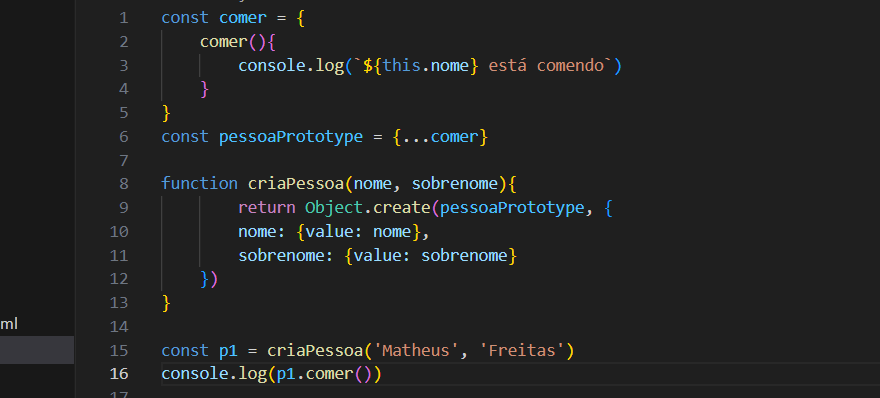
**AULA 100**

* Funções fabricas + Prototypes
* Podemos utilizar prototype junto de funções fabricas, mas de forma diferente
* **Para isso, precisamos construir um objeto que irá guardar as funções e métodos, ele será o nosso prototype**
* Para a função fabrica entender qual é o nosso prototype, utilizamos o comando Object.create
* Quando utilizamos o object.create, precisamos passar o objeto que será o nosso prototype como o primeiro parâmetro, fazendo isso o JavaScript já entende que o objeto é o prototype

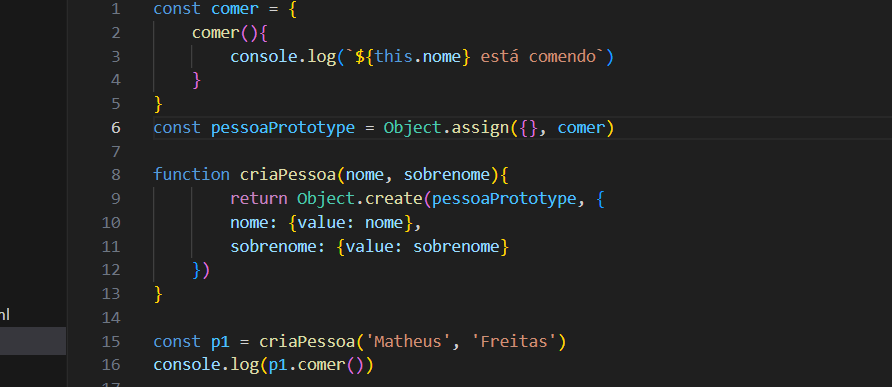


Desta forma, o objeto “pessoaPrototype” se torna o objeto prototype da função fabrica

* Podemos utilizar objetos fora da função fabrica para serem o objeto prototype da função (Que pode ser utilizada por mais de uma função). Exemplo:
  1. Utilizando sprad operation

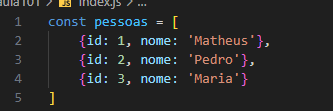


* 1. Utilizando Object.assing

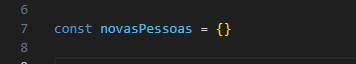


**AULA 101**

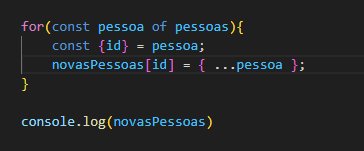
* Aula sobre Objeto Map()
* É uma estrutura de dados (como uma array)
* Criamos uma case
  1. Criamos uma array de objetos chamado pessoas, onde cada índice do array possui um objeto



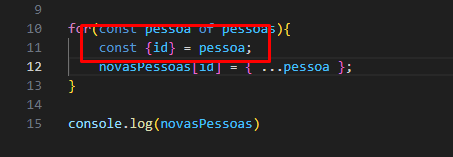
* 1. Criamos um objeto chamado “novasPessoas” vazio



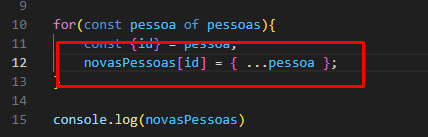
* 1. Utilizamos o comando for para inteirar os valores dos objetos dentro do array “pessoas”



* 1. Atribuímos o id dos objetos dentro da array pessoas para uma variável chamada pessoa



* 1. Atribuímos os valores dos objetos da array para o novo objeto novasPessoas



* 1. Quando utilizamos objetos, nossas chaves só podem ser símbolos ou strings, então para utilizar o número como chave do objeto, vamos utilizar o comando new Map()

