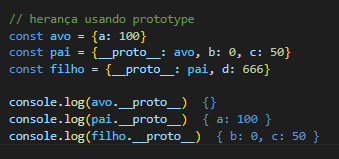
**Herança usando Prototype**

**Basta definir o nome do objeto utilizando a proprieade \_\_proto\_\_ na definição do objeto.**

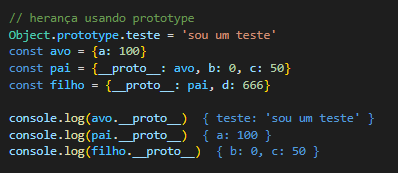
**Ex.:** o objeto avo não herda de ninguém; o objeto pai herda do objeto avo; o objeto filho herda do objeto pai.



**Observação 1:** se um objeto não herda explicitamente de um outro objeto, por padrão, ele irá herdar do objeto **Object.prototype**

**Object.prototype:** prototype é um objeto dentro do objeto Object

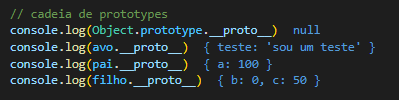
**Ex.:** perceba que o atributo definido em Object.prototype foi herdado ao atributo avo



**Cadeia de Prototypes**

Ao acessarmos a propriedade \_\_proto\_\_, estamos nos referindo ao objeto imediatamente anterior que serviu como prototype para objeto atual.

**Ex.:**

* O objeto ‘Object.prototype’ não tem nenhum outro objeto serviu como prototype
* O ‘avo’ tem o objeto ‘Object.prototype’ como protótipo
* O objeto ‘pai’ tem o objeto ‘avo’ como protótipo
* O objeto ‘filho’ tem o objeto ‘pai’ como protótipo

**Overiding de Métodos e Atributos com Prototype**

Os métodos e atributos da própria classe irão sobrepor os da super-classe.

**Object.setPrototypeOF(objeto, prototype):** define o protótipo de um objeto

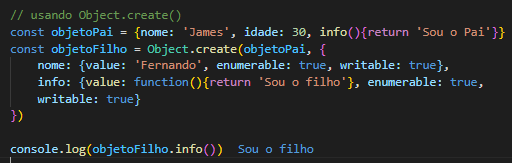
**Ex.:** a classe/objeto gato herdou da classe/objeto animal; o objeto/classe cachorro também herdou da classe/objeto animal

**Outra forma de definir um prototype**

**Object.create(objetoPrototype, {opções})**: método que retorna um objeto que terá protótipo o objeto deinido em objetoPrototype. Onde:

* **{opções}:** será um objeto contendo todo os atributos e métodos

Ex.:



**Métodos relacionados com prototypes**

**hasOwnPropery(string):** verifica se um objeto tem um propriedade própria.

**Ex.:** a propriedade ‘a’ é do próprio objeto teste1, mas a propriedade ‘d’ foi herdada

