```
//PROTOTYPE:
//Como sabemos, todo objeto em Javascript pertence a função interna Object do Javascript. E como uma função construtora, Object possuí propriedades, uma delas é o "prototype", essa propriedade é responsável por referenciar a um determinado objeto pai.
//Object é a função môr de todos os objetos, ou seja, ela não pode ter um pai, ela é o pai de todos, por isso, qualquer objeto criado á partir dele terá o Object como seu prototype.
//E isso vai acontecendo de função construtora para função construtora, por exemplo, se temos um objeto pai e desejamos criar um objeto filho que herde algumas características do pai, o objeto filho terá como prototype os atributos da função pai, e o pai por sua vez terá como prototype os atributos da função avô, e isso vai acontecendo até que chegue a função Object que é o último na lista de prototypes.
//Entender essa sequência de filiações é importante para saber referenciar heranças em Javascript.
//O prototype é uma propriedade privada do Object, por isso, para referenciá-la devemos usar o método de acesso: .__proto__
//Mas cuidado para não confundir, a propriedade [[prototype]] utilizada nas heranças de objetos é totalmente diferente da propriedade "prototype" que existe nas funções, uma coisa não tem a ver com a outra.
//OBS: Quando o assunto é Javascript, é sempre melhor priorizar o uso de composição no lugar de herança.


//ATRIBUTO PROTOTYPE DE OBJETC ESTÁ PRESENTE EM QUALQUER OBJETO:
//OBS: o método __proto__ não é o mais indicado, é preferível usarmos a função "setPrototypeOf()", falamos mais dela na lição "086"...
const teste = {atributo: 'oi'} //Perceba que temos um objeto comum
console.log('\n1)', teste.__proto__ === Object.prototype) //Ao fazer a comparação entre o seu prototype e o de Object temos o mesmo prototype...



//OBJECT NÃO POSSUÍ UM PROTOTYPE SUPERIOR A ELE:
console.log('\n2)', Object.prototype.__proto__) //Veja que o resultado é null, pois Object não possuí um prototype superior...



//UMA FUNCTION GERA UM PROTOTYPE PRÓPRIO DELA:
const funcao = () => {} //Perceba que o prototype de uma função é diferente do prototype do Object...
console.log('\n3)', funcao.prototype === Object.prototype)
```

RESULTADO NO CONSOLE...

```
[Running] node "c:\Users\Almoxarifado\Documents\javascript\arquivos_das_aulas\085-Prototype.js"
```

```
1) true

2) null

3) false

[Done] exited with code=0 in 0.128 seconds
```