FONTE

O JavaScript é frequentemente descrito como uma **linguagem baseada em protótipos** — para fornecer herança, os objetos podem ter um **objeto de protótipo**, que atua como um objeto de modelo do qual herda métodos e propriedades. O objeto de protótipo também pode ter um objeto de protótipo, do qual herda métodos e propriedades, e assim por diante. Isso geralmente é chamado de **cadeia de protótipos** e explica por que objetos diferentes têm propriedades e métodos definidos em outros objetos disponíveis para eles.

Para ser mais exato, as propriedades e os métodos são definidos na propriedade `prototype` nas funções construtoras dos Objetos, não nas próprias instâncias do objeto. Em JavaScript, é feito um link entre a instância do objeto e seu protótipo (sua propriedade `__proto__`, que é derivada da propriedade `prototype` no construtor), e as propriedades e os métodos são encontrados percorrendo a cadeia de protótipos.

É importante entender que há uma distinção entre o protótipo de um objeto (que está disponível por meio de `Object.getPrototypeOf(obj)`, ou por meio da propriedade `__proto__` ) e a propriedade `prototype` em funções construtoras. O primeiro é a propriedade em cada instância e o último é a propriedade no construtor. Ou seja, `Object.getPrototypeOf(new Foobar())` refere-se ao mesmo objeto que `Foobar.prototype`.

## A propriedade do construtor

Toda função de construtor possui uma propriedade prototype cujo valor é um objeto que contém uma propriedade `constructor`. Esta propriedade construtora aponta para a função construtora original. As propriedades definidas na propriedade nomeObjeto.prototype (ou, em geral, na propriedade prototype de uma função construtora, que é um objeto) tornam-se disponíveis para todos os objetos de instância criados usando Construtor Person().

Também é possível adicionar parênteses no final da propriedade do constructor, para criar outra instância de objeto com novos valores. O construtor é uma função depois de tudo, então pode ser chamado usando parênteses; só é preciso incluir a palavra-chave `new` para especificar que deseja usar a função como um construtor.

```
var nomeInstancia = new nomeInstanciaOriginal.constructor(Novos valores);
```