Object Oriented JavaScript

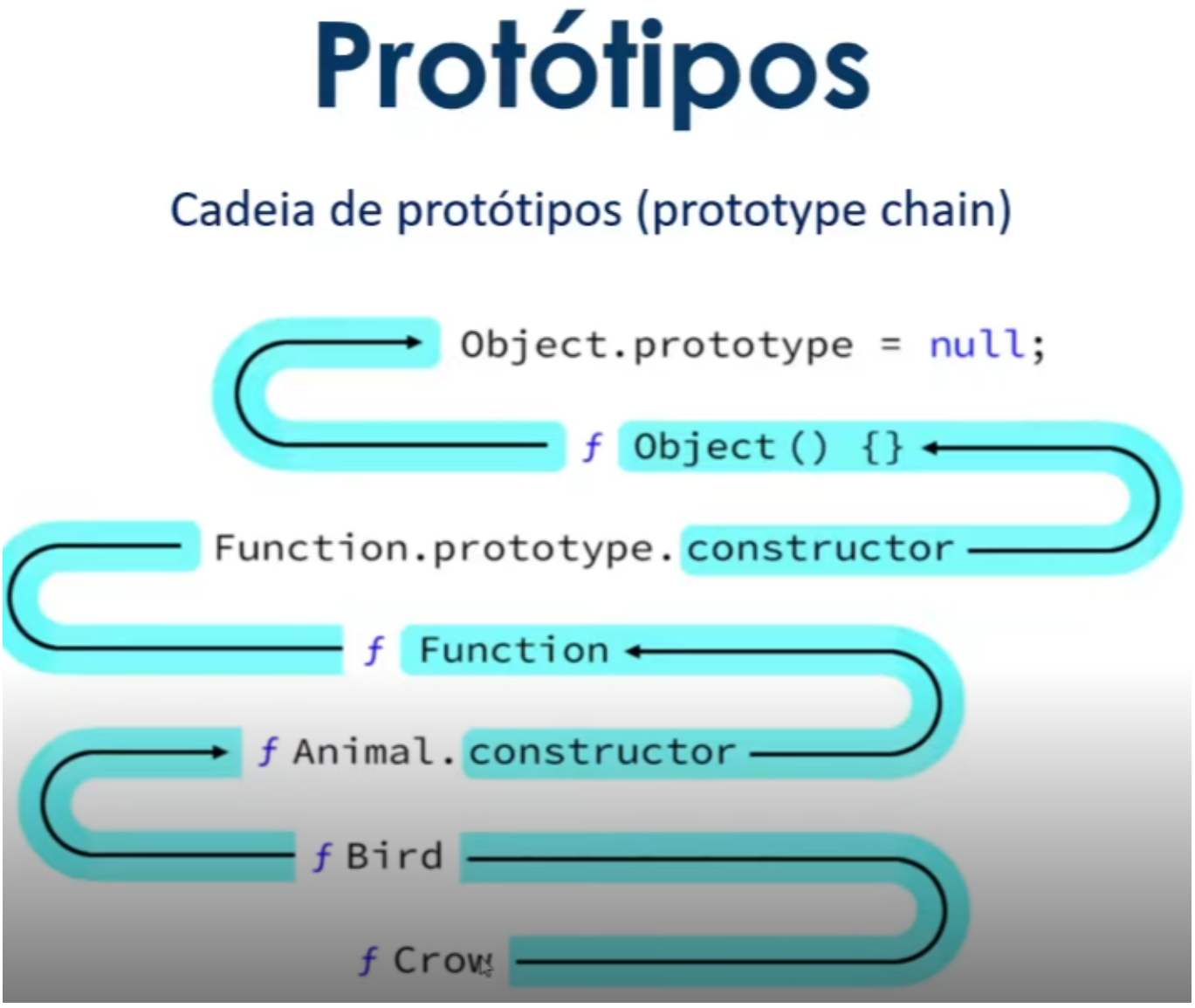
Os protótipos do Javascript são o esqueleto dos objetos, então todos os objetos do javascript vão herdar objeto do prototype.

Exemplo: objeto

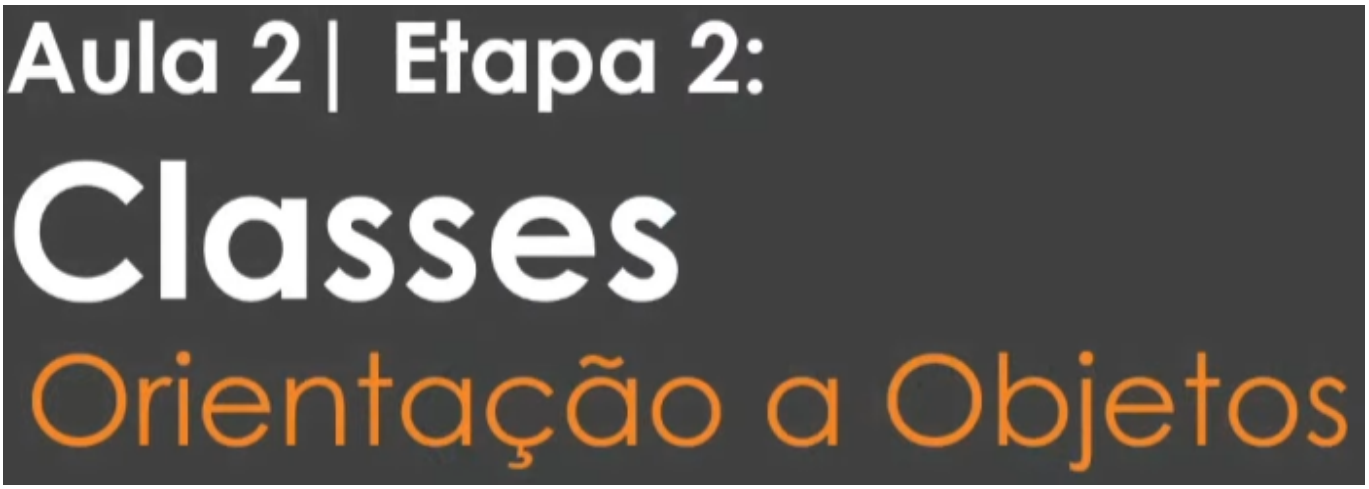
Sempre que temos uma constante do tipo não primitivo(complexo) ele vai ter a propriedade *_proto_* que vai ter uma serie de métodos e propriedades, então podemos utilizar o objeto hasOwnProperty para ver se o objeto tem alguma chave com aquele nome por conta do prototype, pois ele está na classe pai dele, podemos converter um objeto para toString pois a classe pai (prototype) tem essa função, então quando fazemos a chamada de um objeto usamos .toString se caso não existir ele seguiria a procura até o resultado fosse null

Exemplo array

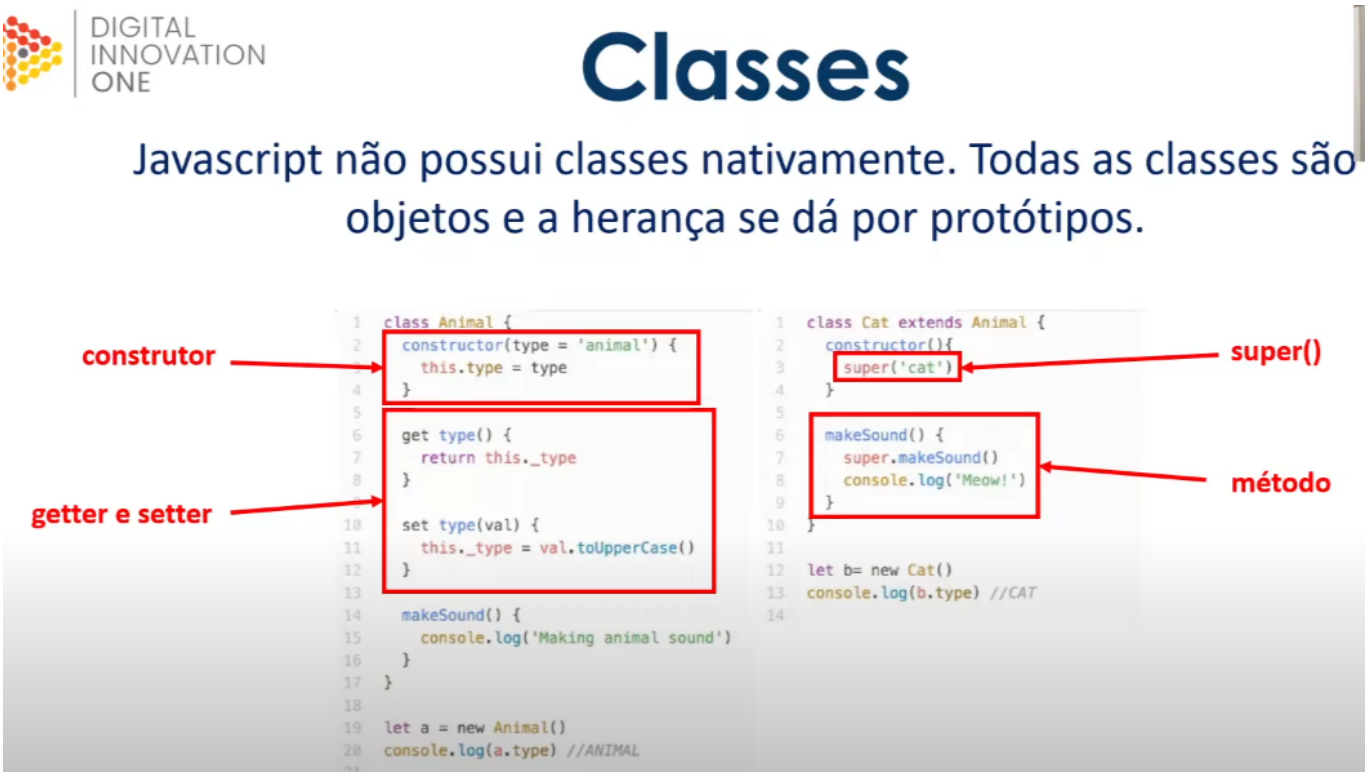
No *_proto_* do array existem todos os métodos do array pois é o protótipo que define os arrays no javascript.



O exemplo a seguir seria o pássaro que herdaria de animal que herdaria de function.prototype que herdaria do object.prototype assim a cadeia iria procurar sempre até chegar no null.



As classes no Javascript não existem nativamente são açucar sintético ou syntatic sugar, que é uma sintaxe para facilitar a escrita mas, o que acontece é usamos sempre objetos que têm protótipos mas com o javascript podemos fazer a sintaxe de classe mais parecida com outras linguagens que são feitas para acomodar o paradigma de orientação a objetos, não é o que está acontecendo e sim objetos e qualquer tipo de herança é feito por protótipos



Classe animal que tem a classe filha cat com a anatomia de uma classe em javascript que tem sempre um construtor que se for passado parametros ele atribui a certos valores dentro dessa classe.

O construtor é o animal, então se não for passado nenhuma informação o tipo vai ser animal se for passado outra coisa será passado de default.

Tem também os getters e setters para termos acesso aos parâmetros e propriedade do objeto utilizamos uma sintaxe para poder settar esse objetos, então o get type que é o tipo do animal vai retornar o tipo e o set para determinar o qual tipo tem outro valor então são métodos que podemos utilizar.

Também temos a classe filha do cat que tem um método chamado super que vai mandar para a função pai os parâmetros que estão ali, então quando digo supercat será construído o tipe ‘animal ’ mas mandando cat então o tipo sera cat O super serve para utilizar as propriedades do construtor que existem na classe pai.

O metodo makeSound é para sobrescrever métodos se quisermos que a classe filha tenha um metodo diferente podemos sobrescrever.