

Objetivos

1. Apresentar o conceito de protótipos e cadeia de protótipos
2. Apresentar a estrutura de classes em Javascript

Protótipos

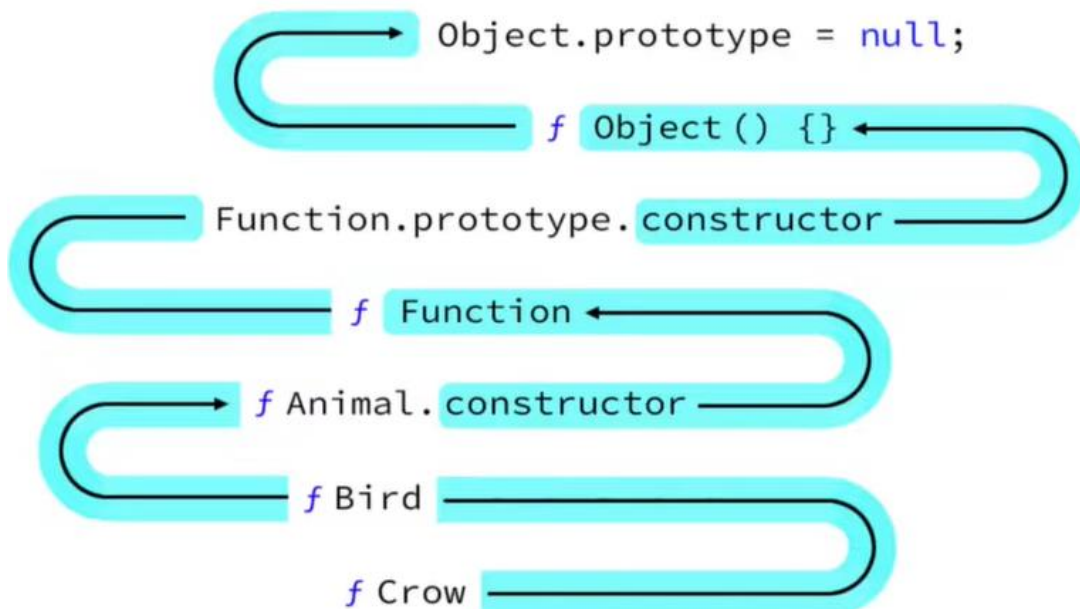
Todos os objetos Javascript herdam propriedades e métodos de um prototype.
O objeto Object.prototype está no topo desta cadeia.

```
> const objeto = {}
< undefined
> objeto
< { }
  __proto__: Object()
  constructor: f Object()
  hasOwnProperty: f hasOwnProperty()
  isPrototypeOf: f isPrototypeOf()
  propertyIsEnumerable: f propertyIsEnumerable()
  toLocaleString: f toLocaleString()
  toString: f toString()
  valueOf: f valueOf()
  __defineGetter__: f __defineGetter__()
  __defineSetter__: f __defineSetter__()
  __lookupGetter__: f __lookupGetter__()
  __lookupSetter__: f __lookupSetter__()
  get __proto__: f __proto__()
  set __proto__: f __proto__()

> array
< []
  length: 0
  __proto__: Array(0)
  concat: f concat()
  constructor: f Array()
  copyWithin: f copyWithin()
  entries: f entries()
  every: f every()
  fill: f fill()
  filter: f filter()
  find: f find()
  findIndex: f findIndex()
  flat: f flat()
  flatMap: f flatMap()
  forEach: f forEach()
  includes: f includes()
  indexOf: f indexOf()
  join: f join()
  keys: f keys()
  lastIndexOf: f lastIndexOf()
  length: 0
```

Protótipos

Cadeia de protótipos (prototype chain)



Classes

Syntatic sugar: uma sintaxe feita para facilitar a escrita



```
var Meal = function(food) {  
  this.food = food  
}  
  
Meal.prototype.eat = function() {  
  return '😋'  
}
```

✖ OLD

```
class Meal {  
  constructor (food) {  
    this.food = food  
  }  
  
  eat() {  
    return '😋'  
  }  
}
```

✔ NEW

🐦 samantha_ming samanthaming.com @samanthaming

Classes

Javascript não possui classes nativamente. Todas as classes são objetos e a herança se dá por protótipos.

The diagram illustrates JavaScript class syntax with two examples: `Animal` and `Cat`. Red boxes and arrows highlight specific features:

- construtor** (constructor): Points to the `constructor` function in the `Animal` class.
- getter e setter** (getter and setter): Points to the `get type()` and `set type(val)` methods in the `Animal` class.
- super()**: Points to the `super('cat')` call in the `Cat` class constructor.
- método** (method): Points to the `makeSound()` method in the `Cat` class.

```
1 class Animal {
2   constructor(type = 'animal') {
3     this.type = type
4   }
5
6   get type() {
7     return this._type
8   }
9
10  set type(val) {
11    this._type = val.toUpperCase()
12  }
13
14  makeSound() {
15    console.log('Making animal sound')
16  }
17 }
18
19 let a = new Animal()
20 console.log(a.type) //ANIMAL
21
```

```
1 class Cat extends Animal {
2   constructor(){
3     super('cat')
4   }
5
6   makeSound() {
7     super.makeSound()
8     console.log('Meow!')
9   }
10 }
11
12 let b= new Cat()
13 console.log(b.type) //CAT
14
```