ORIGAMID

OBJETOS

Prototype

Prototype

A propriedade prototype é um objeto adicionado a uma função quando a mesma é criada.

```
function Pessoa(nome, idade) {
  this.nome = nome;
  this.idade = idade;
}
const andre = new Pessoa('André', 28);

console.log(Pessoa.prototype); // retorna o objeto
console.log(andre.prototype); // undefined
```

funcao.prototype

É possível adicionar novas propriedades e métodos ao objeto prototype.

```
Pessoa.prototype.andar = function() {
  return this.nome + ' andou';
}
Pessoa.prototype.nadar = function() {
  return this.nome + ' nadou';
}
console.log(Pessoa.prototype); // retorna o objeto
```

Acesso ao Protótipo

O objeto criado utilizando o construtor, possui acesso aos métodos e propriedades do protótipo deste construtor. Lembrando, prototype é uma propriedade de funções apenas.

```
const andre = new Pessoa('André', 28);

andre.nome;
andre.idade;
andre.andar();
andre.nadar();
```

proto

O novo objeto acessa os métodos e propriedades do protótipo através da propriedade `__proto__`. É papel da engine fazer essa busca, não devemos falar com `__proto__` diretamente.

```
// Acessam o mesmo método
// mas __proto__ não terá
// acesso ao this.nome
andre.andar();
andre.__proto__.andar();
```

Herança de Protótipo

O objeto possui acesso aos métodos e propriedades do protótipo do construtor responsável por criar este objeto. O objeto abaixo possui acesso a métodos que nunca definimos, mas são herdados do protótipo de Object.

```
Object.prototype;
andre.toString();
andre.isPrototypeOf();
andre.valueOf();
```

Construtores Nativos

Objetos, Funções, Números, Strings e outros tipos de dados são criados utilizando construtores. Esses construtores possuem um protótipo com propriedades e métodos, que poderão ser acessadas pelo tipo de dado.

```
const pais = 'Brasil';
const cidade = new String('Rio');

pais.charAt(0); // B
cidade.charAt(0); // R

String.prototype;
```

É possível acessar a função do protótipo

É comum, principalmente em códigos mais antigos, o uso direto de funções do protótipo do construtor Array.

```
const lista = document.querySelectorAll('li');

// Transforma em uma array
const listaArray = Array.prototype.slice.call(lista);
```

> *Existe o método Array.from()*

Método do Objeto vs Protótipo

Nos objetos nativos existem métodos linkados diretamente ao Objeto e outros linkados ao protótipo.

```
Array.prototype.slice.call(lista);
Array.from(lista);

// Retorna uma lista com os métodos / propriedades
Object.getOwnPropertyNames(Array);
Object.getOwnPropertyNames(Array.prototype);
```

> `dado.constructor.name`, retorna o nome do construtor;

Apenas os Métodos do Protótipo são Herdados

```
[1,2,3].slice(); // existe
[1,2,3].from(); // não existe
```

Entenda o Que está Sendo Retornado

Os métodos e propriedades acessado com o `.` são referentes ao tipo de dados que é retornado antes desse `.`

```
const Carro = {
  marca: 'Ford',
  preco: 2000,
  acelerar() {
    return true;
  }
}

Carro // Object
Carro.marca // String
Carro.preco // Number
Carro.acelerar // Function
Carro.acelerar() // Boolean
Carro.marca.charAt // Function
Carro.marca.charAt(0) // String
```

> *Verifique o nome do construtor*

Exercícios

```
// Crie uma função construtora de Pessoas
// Deve conter nome, sobrenome e idade
// Crie um método no protótipo que retorne
// o nome completo da pessoa

// Liste os métodos acessados por
// dados criados com NodeList,
// HTMLCollection, Document

// Liste os construtores dos dados abaixo
const li = document.querySelector('li');

li;
li.click;
li.innerText;
li.value;
li.hidden;
li.offsetLeft;
```

```
// Qual o construtor do dado abaixo:
li.hidden.constructor.name;
```