
new

new_1.js — javascriptmasterclass

JS new_1.js

```
1   const person = {
2       name: "Linus Torvald",
3       city: "Helsinki",
4       year: 1969,
5       getAge() {
6           return ((new Date()).getFullYear() - this.year);
7       }
8   };
9   console.log(person);
10  console.log(person.getAge());
11
```

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node n
ew/new_1.js
{ name: 'Linus Torvald',
  city: 'Helsinki',
  year: 1969,
  getAge: [Function: getAge] }
49
rodrigobranas:javascriptmasterclass $
```

new_2.js — javascriptmasterclass

JS new_2.js

```
const person1 = {
    name: "Linus Torvald",
    city: "Helsinki",
    year: 1969,
    getAge() {
        return ((new Date()).getFullYear() - this.year);
    }
};
const person2 = {
    name: "Bill Gates",
    city: "Seattle",
    year: 1955,
    getAge() {
        return ((new Date()).getFullYear() - this.year);
    }
};
console.log(person1);
console.log(person1.getAge());
console.log(person2);
console.log(person2.getAge());
```

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node n
ew/new_2.js
{ name: 'Linus Torvald',
  city: 'Helsinki',
  year: 1969,
  getAge: [Function: getAge] }
49
{ name: 'Bill Gates',
  city: 'Seattle',
  year: 1955,
  getAge: [Function: getAge] }
63
rodrigobranas:javascriptmasterclass $
```

Como fazer para criar um objeto a partir da mesma estrutura?

A função fábrica, que é um tipo de padrão, retorna um novo objeto após ser invocada diretamente

new_3.js — javascriptmasterclass

JS new_3.js

```
1   const createPerson = function(name, city, year) {
2       return {
3           name,
4           city,
5           year,
6           getAge() {
7               return ((new Date()).getFullYear() - this.year);
8           }
9       }
10  };
11  const person1 = createPerson("Linus Torvald", "Helsinki", 19
12  const person2 = createPerson("Bill Gates", "Seattle", 1955);
13  console.log(person1);
14  console.log(person1.getAge());
15  console.log(person2);
16  console.log(person2.getAge());
17
```

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node n
ew/new_3.js
{ name: 'Linus Torvald',
  city: 'Helsinki',
  year: 1969,
  getAge: [Function: getAge] }
49
{ name: 'Bill Gates',
  city: 'Seattle',
  year: 1955,
  getAge: [Function: getAge] }
63
rodrigobranas:javascriptmasterclass $
```

O que fazer para eliminar a duplicação e reusar propriedades entre os objetos?

new_4.js — javascriptmasterclass

JS new_4.js

```
const personPrototype = {
    getAge() {
        return ((new Date()).getFullYear() - this.year);
    }
};
const createPerson = function(name, city, year) {
    const person = {
        name,
        city,
        year
    };
    Object.setPrototypeOf(person, personPrototype);
    return person;
};
const person1 = createPerson("Linus Torvald", "Helsinki", 19
const person2 = createPerson("Bill Gates", "Seattle", 1955);
console.log(person1);
console.log(person1.__proto__);
console.log(person1.getAge());
console.log(person2);
console.log(person2.__proto__);
console.log(person2.getAge());
console.log(person1.__proto__ === person2.__proto__);
```

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node n
ew/new_4.js
{ name: 'Linus Torvald', city: 'Helsinki', y
ear: 1969 }
{ getAge: [Function: getAge] }
49
{ name: 'Bill Gates', city: 'Seattle', year:
 1955 }
{ getAge: [Function: getAge] }
63
true
rodrigobranas:javascriptmasterclass $
```

A função construtora retorna um novo objeto ao ser invocada por meio do operador new

new_5.js — javascriptmasterclass

JS new_5.js

```
1   const Person = function(name, city, year) {
2       this.name = name,
3       this.city = city,
4       this.year = year,
5       this.getAge = function() {
6           return ((new Date()).getFullYear() - this.year);
7       }
8   };
9   const person1 = new Person("Linus Torvald", "Helsinki", 1969
10  const person2 = new Person("Bill Gates", "Seattle", 1955);
11  console.log(person1);
12  console.log(person1.__proto__);
13  console.log(person1.getAge());
14  console.log(person2);
15  console.log(person2.__proto__);
16  console.log(person2.getAge());
17  console.log(person1.__proto__ === person2.__proto__);
18
```

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node n
ew/new_5.js
Person {
  name: 'Linus Torvald',
  city: 'Helsinki',
  year: 1969,
  getAge: [Function] }
Person {}
49
Person {
  name: 'Bill Gates',
  city: 'Seattle',
  year: 1955,
  getAge: [Function] }
Person {}
63
true
rodrigobranas:javascriptmasterclass $
```

Toda função tem uma propriedade chamada prototype, que é vinculada ao __proto__ do objeto criado pelo operador new

new_6.js — javascriptmasterclass

JS new_6.js

```
1   const Person = function(name, city, year) {
2       this.name = name,
3       this.city = city,
4       this.year = year
5   };
6   Person.prototype.getAge = function() {
7       return ((new Date()).getFullYear() - this.year);
8   };
9   const person1 = new Person("Linus Torvald", "Helsinki", 1969
10  const person2 = new Person("Bill Gates", "Seattle", 1955);
11  console.log(person1);
12  console.log(person1.__proto__);
13  console.log(person1.getAge());
14  console.log(person2);
15  console.log(person2.__proto__);
16  console.log(person2.getAge());
17  console.log(person1.__proto__ === person2.__proto__);
18
```

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node n
ew/new_6.js
Person { name: 'Linus Torvald', city: 'Helsi
nki', year: 1969 }
Person { getAge: [Function] }
49
Person { name: 'Bill Gates', city: 'Seattle'
, year: 1955 }
Person { getAge: [Function] }
63
true
rodrigobranas:javascriptmasterclass $
```

new_7.js — javascriptmasterclass

JS new_7.js

```
const _new = function(fn, ...params) {
    const obj = {};
    Object.setPrototypeOf(obj, fn.prototype);
    fn.apply(obj, params);
    return obj;
};
const Person = function(name, city, year) {
    this.name = name,
    this.city = city,
    this.year = year
};
Person.prototype.getAge = function() {
    return ((new Date()).getFullYear() - this.year);
};
const person1 = _new(Person, "Linus Torvald", "Helsinki", 19
const person2 = _new(Person, "Bill Gates", "Seattle", 1955);
console.log(person1);
console.log(person1.__proto__);
console.log(person1.getAge());
console.log(person2);
console.log(person2.__proto__);
console.log(person2.getAge());
console.log(person1.__proto__ === person2.__proto__);

```

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node n
ew/new_7.js
Person { name: 'Linus Torvald', city: 'Helsi
nki', year: 1969 }
Person { getAge: [Function] }
49
Person { name: 'Bill Gates', city: 'Seattle'
, year: 1955 }
Person { getAge: [Function] }
63
true
rodrigobranas:javascriptmasterclass $
```

Não esqueça de utilizar o operador new quando utilizar funções construtoras