title: Javascript 原型链
date: 2019-10-21
tag: Javascript

# description: 理解 Javascript 原型链

## 对象划分

对象分为不同对象和函数对象。`Object`、`Function` 是 Javascript 自带的函数对象。

```
typeof Object // 'function'
typeof Function // 'function'

const obj = new Object()
typeof obj // 'object'

const fuc = new Function('log', 'console.log(log)')
typeof fuc // 'function'
```

### 总结

- 通过 `new Function()` 创建的都是函数对象；
- Object 是通过 new Function() 创建出来的。

## 构造函数 Constructor

定义一个 `Book` 构造函数：

```
function Book(name, author) {
  this.name = name;
  this.author = author;
  this.showName = function() {
    console.log('name', this.name)
  };
}
// es6 写法
class BookES2015 {
  constructor(name, author) {
    this.name = name;
    this.author = author;
    this.showName = function () {
      console.log('name', this.name);
    };
  }
}

// 调用
const book1 = new Book('Adada', 'Zsadsa Sad');
```

`book1` 是 Book 的**实例**，它拥有一个 `constructor` 属性切指向 Book。

```
// book1 是构造函数 Book 的实例
console.log(book1.constructor === Book); // true
```

### 总结

实例的构造函数属性指向构造函数。

# 原型对象 Prototype

对象中包含预定义的属性。**每个函数对象**都有 `prototype` 属性，这个属性指向函数的原型对象。

```
function PBook() {}
PBook.prototype.name = 'Adada';
PBook.prototype.author ='Zsadsa Sad'
PBook.prototype.showName = function () {
    console.log('name', this.name);
};


const pbook1 = new PBook();

console.log(pbook1.constructor === PBook); // true
// 可以把 PBook.prototype 视为 PBook 的实例
console.log(PBook.prototype.constructor == PBook); // true
```

## 总结

- 实例的构造函数属性（constructor）指向构造函数（PBook）；
- 原型对象（PBook.prototype）是构造函数（PBook）的一个实例。

原型对象其实就是普通对象。（`Function.prototype` 例外，是函数对象，但是它没有 prototype 属性）：

```
function Book(){}
console.log(typeof Book.prototype) // 'object'
console.log(typeof Object.prototype) // ‘object’
// 例外
console.log(typeof Function.prototype) // 'function'
console.log(typeof Function.prototype.prototype) // 'undefined'
```

## 使用

- 原型对象用于继承：

```
const PBook = function(name) {
  this.name = name; // 注意 this 指向
}
PBook.prototype.showName = function () {
    return this.name;  // 注意 this 指向
};
const pbook1 = new PBook('Mack');
```

`PBook.prototype` 设置了一个函数对象的属性， PBook 的实例 pbook1 继承了这个属性。

## 注意

- `typeof null === 'object'`。 `null` 代表空指针（0x00）， `Object.prototype.toString.call(null) === '[object Null]'`。

# proto

对象的内置属性，用于**指向创建它的构造函数的原型对象**。标准在 ES6 中定制。

```
PBook.prototype.constructor == PBook
pbook1.__proto__ == PBook.prototype
pbook1.constructor == PBook
```

## 问题以及解答

- `book1.__proto__`

book1 的构造函数（Book）的 prototype 属性，即 `Book.prototype`。

- `Book.__proto__`

Book 的构造函数（Function）的 prototype 属性，即 `Function.prototype`。

- `Book.prototype.__proto__`

`Book.prototype` 是原型对象（普通对象）的构造函数（Object）的 prototype 属性，即 `Object.prototype`。

- `Object.__proto__`
  Object 的构造函数（Function）的 prototype 属性，即 `Function.prototype`。

- `Object.prototype.__proto__`
  比较特殊， `Object.prototype.__proto__ === null`