

--	--	--	--	--

--	--

- _____
- _____
- prototype _____
- _____
- constructor _____
- instanceof _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____



A

--	--	--	--	--	--

B

--	--	--	--	--	--	--	--	--

B

```
" " class
```

JavaScript

class

“

--	--	--	--

```
" prototype[ ][ ][ ][ ][ ][ ]
```

JavaScript ☐ ☐ ☐ ☐ ☐ ☐

ES6 class

class []

[illegible]

--	--	--	--	--	--

--	--	--	--	--	--	--

JavaScript

A horizontal number line with arrows at both ends, labeled from 0 to 100 in increments of 10. A vertical tick mark is drawn at the number 10, and another vertical tick mark is drawn at the number 30.

```
function Cat (name, color) {
  this.name = name;
  this.color = color;
}

var cat1 = new Cat('黑', '黑');

cat1.name // '黑'
cat1.color // '黑'
```

The diagram illustrates the memory layout for a C program. It shows a stack frame for the `main` function, which contains two arrays: `cat` (10 slots) and `cat1` (20 slots). Additionally, it shows global variables `name` (3 slots) and `color` (10 slots) located in memory.

```
function Cat(name, color) {
  this.name = name;
  this.color = color;
  this.meow = function () {
    console.log('');
  };
}

var cat1 = new Cat('', '');
var cat2 = new Cat('', '');

cat1.meow === cat2.meow
// false
```

The diagram illustrates the memory layout for a C program. It shows a global array `meow` of 100 elements, followed by a function `meow` that takes a pointer to a 100-element array. The function's local variables include a pointer `meow` to the array, a `char` variable `cat1`, a `char` variable `cat2`, and a `meow` array of 10 elements. The diagram illustrates how the function's local variables are stored in memory, with the `meow` array being a copy of the global array.

JavaScript prototype

prototype 

```
function f() {}  
typeof f.prototype // "object"
```

```
function Animal(name) {
  this.name = name;
}

Animal.prototype.color = 'white';

var cat1 = new Animal('cat');
var cat2 = new Animal('cat');

cat1.color // 'white'
cat2.color // 'white'
```

```
Animal.prototype.color = 'yellow';

cat1.color // "yellow"
cat2.color // "yellow"
```

Diagram illustrating a memory layout with four rows of blocks and labels:

- Row 1: 8 blocks, `color` label, 4 blocks, `yellow` label, 4 blocks, `color` label.
- Row 2: 16 blocks, `color` label, 8 blocks, `color` label.
- Row 3: 32 blocks.
- Row 4: 8 blocks.
- Row 5: 32 blocks.

```

cat1.color = 'black';

cat1.color // 'black'
cat2.color // 'yellow'
Animal.prototype.color // 'yellow';

```

cat1 color black color
yellow

```

Animal.prototype.walk = function () {
  console.log(this.name + ' is walking');
};

```

Animal.prototype walk Animal

JavaScript prototype

" " prototype chain

Object.prototype Object prototype
 Object.prototype valueOf toString
 Object.prototype

Object.prototype Object.prototype null null
 null

```

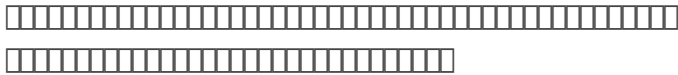
Object.getPrototypeOf(Object.prototype)
// null

```

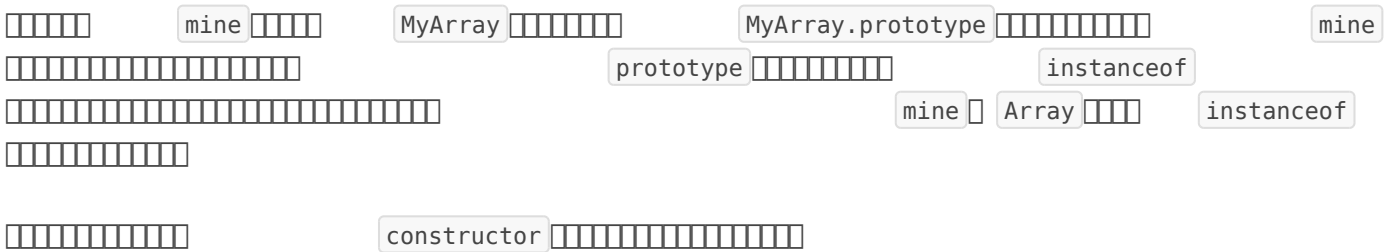
Object.prototype null null
Object.getPrototypeOf

JavaScript
 Object.prototype undefined

" "
overriding



```
var MyArray = function () {};  
  
MyArray.prototype = new Array();  
MyArray.prototype.constructor = MyArray;  
  
var mine = new MyArray();  
mine.push(1, 2, 3);  
mine.length // 3  
mine instanceof Array // true
```



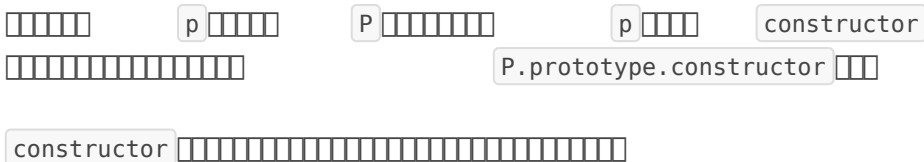
constructor



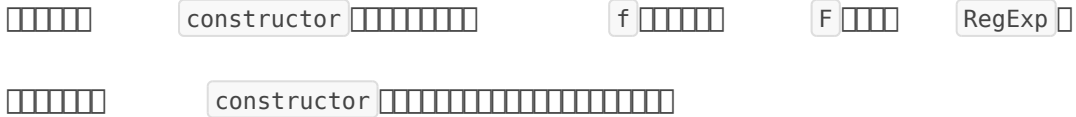
```
function P() {}  
P.prototype.constructor === P // true
```



```
function P() {}  
var p = new P();  
  
p.constructor === P // true  
p.constructor === P.prototype.constructor // true  
p.hasOwnProperty('constructor') // false
```



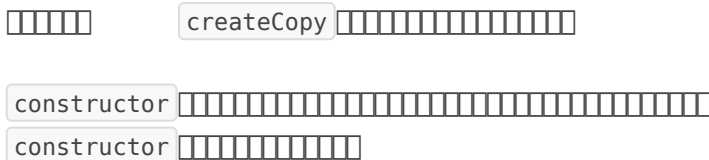
```
function F() {};  
var f = new F();  
  
f.constructor === F // true  
f.constructor === RegExp // false
```



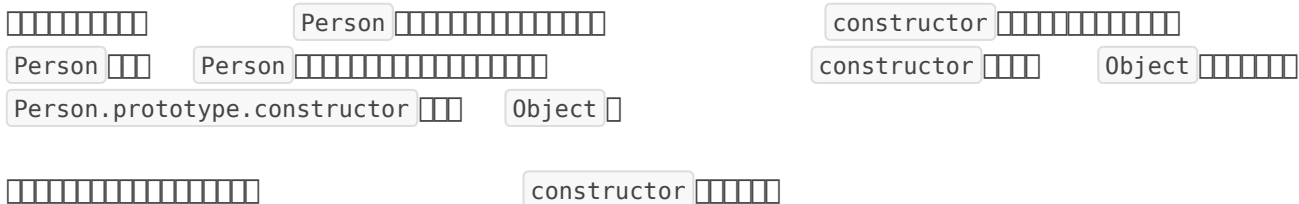
```
function Constr() {}  
var x = new Constr();  
  
var y = new x.constructor();  
y instanceof Constr // true
```



```
Constr.prototype.createCopy = function () {  
    return new this.constructor();  
};
```



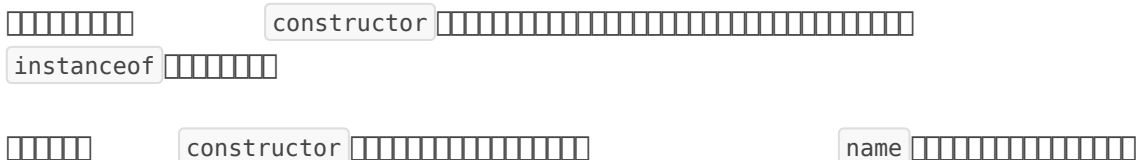
```
function Person(name) {  
    this.name = name;  
}  
  
Person.prototype.constructor === Person // true  
  
Person.prototype = {  
    method: function () {}  
};  
  
Person.prototype.constructor === Person // false  
Person.prototype.constructor === Object // true
```



```
// 初始化
C.prototype = {
  method1: function (...) { ... },
  // ...
};

// 设置
C.prototype = {
  constructor: C,
  method1: function (...) { ... },
  // ...
};

// 设置
C.prototype.method1 = function (...) { ... };
```



```
function Foo() {}
var f = new Foo();
f.constructor.name // "Foo"
```

instanceof

```
instanceof
```

```
var v = new Vehicle();
v instanceof Vehicle // true
```



instanceof []
[]

prototype

```
v instanceof Vehicle  
//  
Vehicle.prototype.isPrototypeOf(v)
```

[] Vehicle [] v [] Vehicle.prototype [] isPrototypeOf() []
JavaScript []

[] instanceof [] true []

```
var d = new Date();  
d instanceof Date // true  
d instanceof Object // true
```

[] d [] Date [] Object [] true []

[] null [] Object [] instanceof [] null
[]

```
var obj = { foo: 123 };  
obj instanceof Object // true  
  
null instanceof Object // false
```

[] null [] instanceof Object [] true []

instanceof [] prototype
[] null []
instanceof []

```
var obj = Object.create(null);  
typeof obj // "object"  
obj instanceof Object // false
```

[] Object.create(null) [] obj [] null [] Object.create()
[] Object [] prototype [] instanceof
[] obj [] Object [] instanceof [] null
[]

instanceof []

```
var x = [1, 2, 3];
var y = {};
x instanceof Array // true
y instanceof Object // true
```

 instanceof  x  y 

 instanceof 

```
var s = 'hello';
s instanceof String // false
```

 String  false 

 undefined  null  instanceof  false 

```
undefined instanceof Object // false
null instanceof Object // false
```

 instanceof  new 

```
function Fubar (foo, bar) {
  if (this instanceof Fubar) {
    this._foo = foo;
    this._bar = bar;
  } else {
    return new Fubar(foo, bar);
  }
}
```

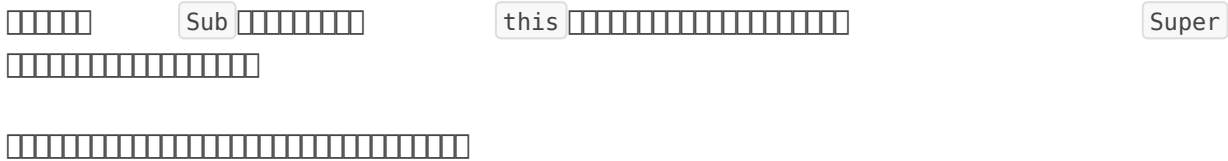
 instanceof  this  Fubar 
 new 



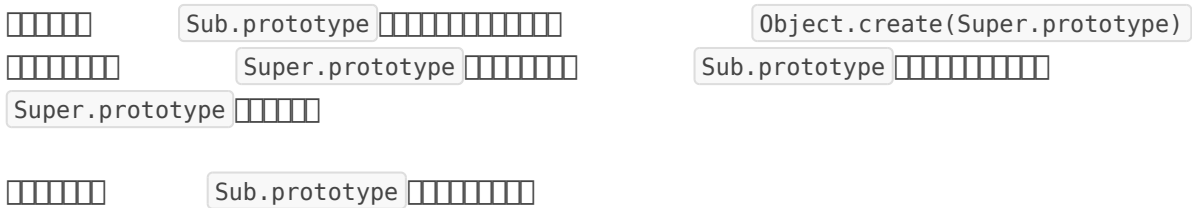



```
function Sub(value) {
  Super.call(this);
}
```

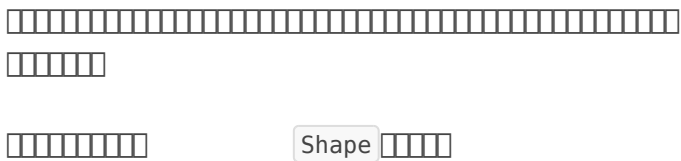
```
this.prop = value;
}
```



```
Sub.prototype = Object.create(Super.prototype);
Sub.prototype.constructor = Sub;
Sub.prototype.method = '...';
```



```
Sub.prototype = new Super();
```



```
function Shape() {
  this.x = 0;
  this.y = 0;
}

Shape.prototype.move = function (x, y) {
  this.x += x;
  this.y += y;
  console.info('Shape moved.');
```



```
// 初始化
function Rectangle() {
  Shape.call(this); // 初始化
}
```

```
// 测试
function Rectangle() {
  this.base = Shape;
  this.base();
}

// 测试原型链
Rectangle.prototype = Object.create(Shape.prototype);
Rectangle.prototype.constructor = Rectangle;
```

测试
instanceof
测试
true

```
var rect = new Rectangle();

rect instanceof Rectangle // true
rect instanceof Shape // true
```

测试

```
ClassB.prototype.print = function() {
  ClassA.prototype.print.call(this);
  // some code
}
```

测试
B
print
测试
A
print
测试
A

测试

JavaScript

测试

```
function M1() {
  this.hello = 'hello';
}

function M2() {
  this.world = 'world';
}
```

```
function S() {
  M1.call(this);
  M2.call(this);
}

// 创建 M1
S.prototype = Object.create(M1.prototype);
// 混入 M2
Object.assign(S.prototype, M2.prototype);

// 设置构造函数
S.prototype.constructor = S;

var s = new S();
s.hello // 'hello'
s.world // 'world'
```

原型链

S 原型链

M1 和 M2 原型链

Mixin 原型链



JavaScript “ ” JavaScript

.....

.....

JavaScript

.....

.....

JavaScript

ES6

“ ” “ ”

.....



.....

.....

```

var module1 = new Object({
  _count : 0,
  m1 : function (){
    //...
  },
  m2 : function (){
    //...
  }
});

```

|||||

m1

m2

|||||

module1

||||||||||||||||||||

```

module1.m1();

```

||||||||||||||||||||

|||

```

module1._count = 5;

```

|||||

|||||

```

function StringBuilder() {
  var buffer = [];

  this.add = function (str) {
    buffer.push(str);
  };

  this.toString = function () {
    return buffer.join('');
  };
}

```

|||||

buffer

||||||||||||||||||||

buffer

||||||||||||||||||||

||||||||||||||||||||

||||||||||||||||||||

[illegible]

```
function StringBuilder() {
  this._buffer = [];
}

StringBuilder.prototype = {
  constructor: StringBuilder,
  add: function (str) {
    this._buffer.push(str);
  },
  toString: function () {
    return this._buffer.join('');
  }
};
```

[illegible]

1

[illegible]

□□□□□□ "□□□□□" □ Immediately-Invoked Function Expression □ IIFE

[illegible]

```
var module1 = (function () {
    var _count = 0;

    var m1 = function () {
        //...
    };

    var m2 = function () {
        //...
    };

    return {
        m1 : m1,
        m2 : m2
    };
})();
```

```
_count
```

```
console.info(module1._count); //undefined
```

module1 JavaScript



augmentation

“ ”

```
var module1 = (function (mod){  
  mod.m3 = function () {  
    //...  
  };  
  return mod;  
})(module1);
```

module1 m3() module1

Loose augmentation

“ ” Loose augmentation

```
var module1 = (function (mod) {  
  //...  
  return mod;  
})(window.module1 || {});
```

“ ” “ ” “ ”



```
var module1 = (function ($, YAHOO) {  
  //...  
})(jQuery, YAHOO);
```

module1 jQuery YUI

module1



```
(function($, window, document) {  
  
    function go(num) {  
    }  
  
    function handleEvents() {  
    }  
  
    function initialize() {  
    }  
  
    function dieCarouselDie() {  
    }  
  
    //attach to the global scope  
    window.finalCarousel = {  
        init : initialize,  
        destroy : dieCarouselDie  
    }  
  
})( jQuery, window, document );
```

finalCarousel

init

destroy

go

handleEvents

initialize

dieCarouselDie



- [JavaScript Modules: A Beginner’s Guide](#), by Preethi Kasireddy