

Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.



We have 1 open job ♥

Love your work. Create an impact.

## How to get a JavaScript object's class?

Ask Question



536



134

I created a JavaScript object, but how I can determine the class of that object?

I want something similar to Java's `.getClass()` method.

javascript

oop

edited Dec 27 '14 at 13:04



Boann

36.7k

12

87

121

asked Aug 8 '09 at 18:11



DNB5brims

9,798

38

104

159

5 for example , I make a Person like this : `var p = new Person();` I have a Person Object that called "p", how can I use "p" to get back the Class name: "Person". – [DNB5brims](#)  
Aug 8 '09 at 18:20

7 [Duplicate](#) – [Casebash](#) May

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

TEAMS

Create Team

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

**does** have a  
class keyword  
and class  
syntax for  
creating  
prototypes in  
which the  
methods can  
more easily  
access super .  
—

[james\\_womack](#)  
Aug 1 '16 at 3:10

What about  
Object.className  
e? —  
[Paul Basenko](#)  
Jan 23 '17 at  
12:04

### 14 Answers



764



There's no exact  
counterpart to  
Java's [getClass\(\)](#)  
in JavaScript.

Mostly that's due  
to JavaScript  
being a [prototype-  
based language](#),  
as opposed to  
Java being a  
[class-based](#) one.

Depending on  
what you need  
[getClass\(\)](#) for,  
there are several  
options in  
JavaScript:

- [typeof](#)
- [instanceof](#)
- obj. [constructor](#)
- func. [prototype](#)

## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

```
typeof Foo;
typeof foo;

foo instanceof Foo
foo.constructor.name
Foo.name

Foo.prototype.isPrototypeOf(foo)

Foo.prototype.bar = function() {
  foo.bar(21);
}
```

Note: if you are compiling your code with Uglify it will change non-global class names. To prevent this, Uglify has a [--mangle](#) param that you can set to false is using [gulp](#) or [grunt](#).

edited Sep 16 '16 at 16:00



James L.

2,897 1 15 29

answered Aug 8 '09 at 18:20



earl

26.7k 4 43 53

---

5 That should probably be `func.prototype` (yes, functions are objects, but the prototype property is only relevant on function objects). – [Miles](#)  
Aug 8 '09 at 18:37

---

3 you might also want to mention `instanceof` / `isPrototypeOf()` and the non-standard `__proto__` – [Christoph](#) Aug 8 '09 at 18:37


---

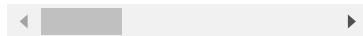
By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and [our Terms of Service](#).

## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

8 '09 at 18:52

1 Yes, clarkf, that's `Foo` pretty-printed. The comments don't indicate the return values, but equalities that hold for the return values. So the comment means that `foo.constructor` or `== Foo` holds, which will also be the case for you. – [earl](#)  
Oct 24 '10 at 21:12

4 **Warning:** don't rely on `constructor`. `name` if your code is being minified. The function name is going to change arbitrarily. – [igorsantos07](#)  
Mar 31 '16 at 21:26 



▲ `obj.constructor.name`  
220 works in most cases in modern browsers, despite `Function.name` not being officially added to the standard until ES6. If the object is instantiated with

## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

It will return "Number" for numbers, "Array" for arrays and "Function" for functions, etc. It seems to be quite reliable. The only cases where it fails are if an object is created without a prototype, via `Object.create( null )`, or the object was instantiated from an anonymously-defined (unnamed) function.

Arguably, `obj.constructor.name` is much more intuitive than `typeof`, and could be encapsulated in a function to handle the odd case where `constructor` isn't defined (and to handle null references).

**Note:** Another advantage to this method is it works intuitively across DOM boundaries versus comparing the constructor objects directly or using `instanceOf`. The reason that doesn't work as you might expect is there are actually different instances of the constructor

## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

won't work.

**Note 2:** Oddly enough, this method appears to return the name of the base-most function used in a prototype chain, which is unfortunately not intuitive. For example if `B` derives prototypically from `A` and you create a new instance of `B`, `b`, `b.constructor.name` returns "A"! So that feels totally backwards. It does work fine for single-level prototypes and all primitives, however.

ited Jul 20 '16 at 22:49

swered Jan 3 '12 at 16:36



[devios1](#)

19.2k 35 128 217

---


10 `Function.name` is not (yet) part of the JavaScript standard. It is currently supported in Chrome and Firefox, but not in IE(10). – [Halcyon](#) Nov 4 '13 at 16:44

---

```
Object.create  
(something).co
```

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and [our Terms of Service](#).

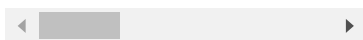
## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

all objects made with  
Object.create,  
no matter with  
or without a  
prototype. —  
[user2451227](#)  
Jul 22 '14 at  
10:55 

11 `obj.construct`  
`or.name` only  
works for *named*  
functions. I.e., if  
I define `var`  
`Foo =`  
`function() {}` ,  
then for `var`  
`foo = new`  
`Foo()` ,  
`foo.construct`  
`or.name` will  
give you empty  
string. — [KFL](#)  
Sep 1 '14 at  
8:02

15 **Warning:** don't  
rely on  
`constructor.n`  
`ame` if your code  
is being  
minified. The  
function name is  
going to change  
arbitrarily. —  
[igorsantos07](#)  
Mar 31 '16 at  
21:29

1 `Function.name`  
is part of ES6,  
see  
[developer.mozilla](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/...)  
[a.org/en-](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/...)  
[US/docs/Web/J](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/...)  
[avaScript/Refer](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/...)  
[ence/...](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/...) —  
[Janus Troelsen](#)  
Jun 20 '16 at  
15:02



This function  
returns either

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and [our Terms of Service](#).

## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

```

function getClass(o)
{
    if (typeof obj ==
        return "undefin
    if (obj === null)
        return "null";
    return Object.pro
        .match(/^\[obje
}

getClass("") ===
getClass(true) ===
getClass(0) ===
getClass([]) ===
getClass({}) ===
getClass(null) ===
// etc...

```

answered Aug 9 '09 at 5:53



**Eli Grey**

**27.7k** 12 61 87

---

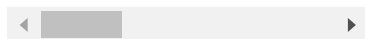
Object.prototype.  
getClass =  
function(){ using  
'this' instead of  
obj would be  
nice – [SparkK](#)  
Jan 24 '12 at  
17:20

---

2 of course then  
null and  
undefined would  
be uncheckable  
since only the  
Object would  
have the  
getClass method  
– [SparkK](#) Jan 24  
'12 at 17:25

---

5 This only works  
on native  
objects. If you  
have some kind  
of inheritance  
going you will  
always get  
"Object" . –  
[Halcyon](#) Nov 4  
'13 at 16:46



By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and [our Terms of Service](#).



## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

```
obj.constructor
```

assuming the  
 constructor is set  
 correctly when you  
 do the inheritance -  
 - which is by  
 something like:

```
Dog.prototype = new  

Dog.prototype.const
```

and these two  
 lines, together  
 with:

```
var woofie = new Do
```

will make  
 woofie.constructor  
 point to Dog . Note  
 that Dog is a  
 constructor  
 function, and is a  
 Function object.  
 But you can do if  
 (woofie.constructor  
 === Dog) { ... } .

If you want to get  
 the class name as  
 a string, I found the  
 following working  
 well:

<http://blog.magnetiq.com/post/514962277/finding-out-class-names-of-javascript-objects>

```
function getObjectC  

  if (obj && obj.  

    var arr = o  

    /functi  
  

    if (arr &&  

      return  

    }  

  }
```

```
return undefine
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

extract the name  
of the constructor  
function.

Note that

`obj.constructor.name`  
could have  
worked well, but it  
is not standard. It  
is on Chrome and  
Firefox, but not on  
IE, including IE 9  
or IE 10 RTM.

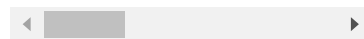
edited Oct 4 '12 at 21:39

answered Oct 4 '12 at 14:55



太極者無極而生

69.9k 98 371 621



You can get a  
reference to the  
constructor  
function which  
created the object  
by using the  
[constructor](#)  
[property](#):

```
function MyObject()  
{  
}
```

```
var obj = new MyObj  
obj.constructor; //
```

If you need to  
confirm the type of  
an object at  
runtime you can  
use the [instanceof](#)  
operator:

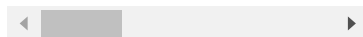
```
obj instanceof MyOb
```

edited Aug 8 '09 at 18:31

## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

doesn't it return the constructor function itself, like, you can call it again and create a new object of that type? – [Spark](#)  
Jan 24 '12 at 17:19

- 1 @Spark Yes, though you can still use this for a comparison so long as you are on the same DOM (you are comparing function objects). However it is much better practice to turn the constructor into a string and compare that, specifically because it works across DOM boundaries when using iframes. – [devios1](#) Feb 15 '12 at 16:00



- 4 In keeping with its unbroken record of backwards-compatibility, ECMAScript 6, JavaScript still doesn't have a `class` type (though not everyone understands this). It **does** have a `class` keyword as part of its `class` syntax for creating prototypes—but

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and [our Terms of Service](#).

## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

language.

Speaking of JS in terms of class is only either misleading or a sign of not yet grokking prototypical inheritance (just keeping it real).

That means

`this.constructor` is still a great way to get a reference to the constructor function. And `this.constructor.prototype` is the way to access the prototype itself. Since this isn't Java, it's not a class. It's the prototype object your instance was instantiated from. Here is an example using the ES6 syntactic sugar for creating a prototype chain:

```
class Foo {
  get foo () {
    console.info(th
    return 'foo'
  }
}

class Bar extends F
  get foo () {
    console.info('[
Object.getOwnProper
    console.info('[
Object.getOwnProper

    return `${super
  }
}

const bar = new Bar
console.dir(bar.foo
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

```
[SUPER] [Function:  
[Function: Bar] 'Ba  
'foo + bar'
```

There you have it!  
In 2016, there's a  
class keyword in  
JavaScript, but still  
no class type.

this.constructor  
is the best way to  
get the constructor  
function,

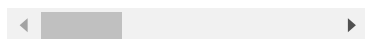
this.constructor.p  
rototype the best  
way to get access  
to the prototype  
itself.

swered Aug 1 '16 at 3:45



[james\\_womack](#)

7,068 4 44 65



4

i had a situation to  
work generic now  
and used this:



```
class Test {  
  // your class def  
}
```

```
nameByType = functi  
  return type.proto  
};
```

```
console.log(nameByT
```

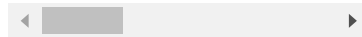
thats the only way i  
found to get the  
class name by type  
input if you don't  
have a instance of  
an object.

(written in ES2017)

dot notation also  
works fine

## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

Ah this is what I was looking for.  
If it's not instantiated you have to use 'prototype' to get the class name.  
Thanks a ton! –  
[Artokun](#) Jul 6 at 3:47



▲  
3  
▼

I find  
object.constructor  
.toString() return  
[object  
objectClass] in IE  
,rather than  
function  
objectClass () {}  
returned in chrome.  
So,I think the code  
in  
<http://blog.magnetiq.com/post/514962277/finding-out-class-names-of-javascript-objects>  
may not work well  
in IE.And I fixed  
the code as  
follows:

### code:

```
var getObjectClass
  if (obj &&
      /*
       *
       */
      if(
        }
        var
        /*
         *
         */
```

Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

```

    }
    if

    }
    return un

};

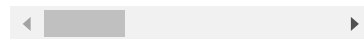
```

answered Dec 17 '12 at 12:34



[zzy7186](#)

46 4



▲  
3  
▼

For Javascript  
Classes in ES6  
you can use  
object.constructor  
. In the example  
class below the  
getClass()  
method returns the  
ES6 class as you  
would expect:

```

var Cat = class {
    meow() {
        console.log
    }
    getClass() {
        return this
    }
}

var fluffy = new Ca
...

var AlsoCat = fluff
var ruffles = new A

ruffles.meow();

```

If you instantiate  
the class from the

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and [our Terms of Service](#).

Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

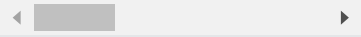
八 廿 巳 午 未 申 酉 戌 亥

swered Feb 2 '16 at 15:16



[Hugheth](#)

373 3 9



2

In javascript, there are no classes, but I think that you want the constructor name and `obj.constructor.toString()` will tell you what you need.

ited Oct 9 '12 at 14:08



[lucian.pantelimon](#)

3,047 3 22 45

swered Aug 8 '09 at 18:33



[Jikhan](#)

114 1

- 1 This will return the entire definition of the constructor function as a string. What you really want is `.name` – [devios1](#) Jan 3 '12 at 16:39
- 3 but `.name` is not defined even on IE 9 – [太極者無極而生](#) Oct 4 '12 at 14:47



1

Agree with dfa, that's why i consider the



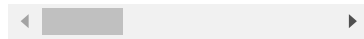
## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

or the one posted  
by Eli Grey, to  
match my way of  
mind

```
function what(obj){
  if(typeof(obj)=
  if(obj===null)r
  var res = Objec
  if(res==="Objec
    res = obj.c
    if(typeof(r
      if(obj
      if(obj
    return
  }
}
return res;
}
```

swered Oct 17 '14 at 16:18

 [Antkhan](#)  
27 1



JavaScript is a  
class-less  
languages: there  
are no classes that  
defines the  
behaviour of a  
class statically as  
in Java. JavaScript  
uses prototypes  
instead of classes  
for defining object  
properties,  
including methods,  
and inheritance. It  
is possible to  
simulate many  
class-based  
features with  
prototypes in  
JavaScript.

swered Aug 8 '09 at 18:21



[dfa](#)  
92.9k 28 172 218

Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

Update: AS of ECMAScript 6, JavaScript still doesn't have a `class` type. It **does** have a `class` keyword and `class` syntax for creating prototypes in which the methods can more easily access `super`.

—  
[james\\_womack](#)  
 Aug 1 '16 at 3:08



**Here's a  
 implementation of  
 and**

`getClass()`  
`getInstance()`

You are able to get a reference for an Object's class using `window`.

**From an  
 instance  
 context:**

```
function A() {
  this.getClass =
    return wind
}

this.getNewInst
return new
}

}

var a = new A();
console.log(a.getCl

// you can even:
var b = new a.getCl
b instanceof A; //
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

```

        return window[t
    }

    B.getInstance() {
        return new wind
    }

```

swered Dec 22 '15 at 22:13

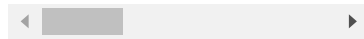


[Bruno Finger](#)

653 1 8 32

2 Why not just  
this.construc  
tor ? –  
[Solomon Ucko](#)  
Apr 22 '16 at  
16:42

I don't know, but  
if it is better, you  
can definitely  
edit the answer  
to improve it as  
you find better,  
after all this is a  
community. –  
[Bruno Finger](#)  
Mar 5 at 10:38



Question seems  
already answered  
but the OP wants  
to access the class  
of and object, just  
like we do in Java  
and the selected  
answer is not  
enough (imho).

With the following  
explanation, we  
can get a class of  
an object(it's  
actually called  
prototype in  
javascript).

```

var arr = new Array
var arr2 = new Arra

```

## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

```

    }
  });
  console.log(arr.last

```

But `.last` property will only be available to `'arr'` object which is instantiated from Array prototype. So, in order to have the `.last` property to be available for all objects instantiated from Array prototype, we have to define the `.last` property for Array prototype:

```

Object.defineProperty
  get: function(){
    return this[thi
  }
});
console.log(arr.last
console.log(arr2.la

```

The problem here is, you have to know which object type (prototype) the `'arr'` and `'arr2'` variables belongs to! In other words, if you don't know class type (prototype) of the `'arr'` object, then you won't be able to define a property for them. In the above example, we know `arr` is instance of the Array object, that's why we used `Array.prototype` to define a property for `Arrav`. But what

## Stack Overflow requires external JavaScript from another domain, which is blocked or failed to load.

```
Object.defineProperty  
  get: function(){  
    return this[thi  
  }  
});  
console.log(arr.las  
console.log(arr2.la
```

As you can see,  
without knowing  
that 'arr' is an  
Array, we can add  
a new property just  
by referring the  
class of the 'arr'  
by using  
'arr.\_\_proto\_\_'.

We accessed the  
prototype of the  
'arr' without  
knowing that it's an  
instance of Array  
and I think that's  
what OP asked.

answered Nov 12 '16 at 17:13



[Ramazan Polat](#)

2,378 23 40

