

JS

This

What is *this* ?

It's contextual based on the **conditions of the function's invocation**.

Call Site

The location in code where a **function is called**.

Call Stack

It's the stack of functions that have been called to **get us to the current moment in execution.**

Let's analyze

```
function baz() {  
  // call-stack is: `baz`  
  // so, our call-site is in the global scope  
  
  console.log( "baz" );  
  bar(); // <-- call-site for `bar`  
}  
  
function bar() {  
  // call-stack is: `baz` -> `bar`  
  // so, our call-site is in `baz`  
  
  console.log( "bar" );  
  foo(); // <-- call-site for `foo`  
}  
  
function foo() {  
  // call-stack is: `baz` -> `bar` -> `foo`  
  // so, our call-site is in `bar`  
  
  console.log( "foo" );  
}  
  
baz(); // <-- call-site for `baz`
```

How to find *this*

We're going to have 4 simple rules:

1. Default Binding.
2. Implicit Binding.
3. Explicit Binding.
4. New Binding.

Default Binding

Standalone function invocation.



```
function foo() {  
    console.log(this.a);  
}  
  
var a = 2;  
  
foo(); // 2
```

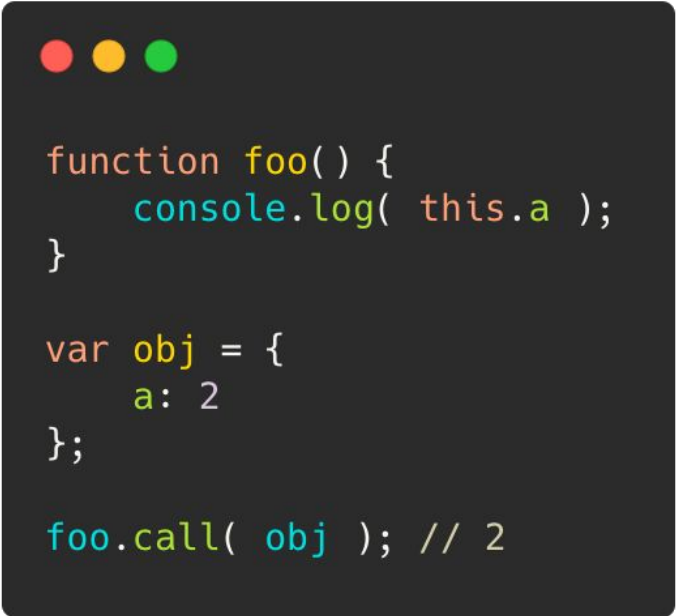
Implicit Binding

Call site has a context object.

```
function foo() {  
    console.log( this.a );  
}  
  
var obj1 = {  
    a: 2,  
    obj2: {  
        a: 42,  
        foo: foo  
    }  
};  
  
obj1.obj2.foo(); // 42
```


Explicit Binding

We call it explicitly.



```
function foo() {  
    console.log( this.a );  
}  
  
var obj = {  
    a: 2  
};  
  
foo.call( obj ); // 2
```

New Binding

Function is invoked with *new* keyword.



```
function foo(a) {  
    this.a = a;  
}  
  
var bar = new foo( 2 );  
console.log( bar.a ); // 2
```

Order of precedence

New Binding > Explicit Binding > Implicit Binding > Default Binding

Activity (optional)

Research about **implicit lost binding** & **Hard binding (pass-thru and ES5 utility)**.

You don't know JS

<https://github.com/getify/You-Dont-Know-JS>

JS