**What is ‘this’ referencing to in JavaScript?**

The JavaScript *this* keyword refers to all the objects that exist in the function’s call-site. A [call-site](https://en.wikipedia.org/wiki/Call_site) of a function is the location where the function is called.

When a function’s call-site is inside an object, *this* refers to the object where the function belongs to. When a function’s call-site is in the window’s object, this refers to the global objects.

If you are not familiar with *this*, the explanation above is probably not sufficient. Let’s have a look at actual examples.

**Example**

Can you tell the output of this code example?

<script src="https://gist.github.com/mydatahack/7e4d30c13dc1d465db63f5d710574860.js"></script>

**Output**

Here is the output from the code example above.

<pre>

hello

global

hello

hello

</pre>

**Explanation**

The function b’s call-site is the check object. Therefore, *this.a* refers to a that blongs to the check object.

In the function c, setTimeout’s call-site is the windows object (window.setTimeout). Therefore, *this* refers to the global object. Hence, *this.a* is from the global scope. If you delete a in the global namespace, *this.a* becomes undefined.

The function d uses the old school technique of reassigning *this* to \_*this*. By using \_*this*, setTimeout has access to the a variable within the check object.

ES6 arrow function introduced [lexical scope](https://en.wikipedia.org/wiki/Scope_(computer_science)#Lexical_scope_vs._dynamic_scope). Therefore, *this* refers to the check object in the function e as we are using the arrow function for the setTimeout callback function.

Hope this helps you to clarify *this*.

Example

<script>

  var a = 'global';

  var check = {

    a: 'hello',

    b: function() {

      console.log(this.a);

    },

    c: function() {

      setTimeout(function(){

        // this becomes global because setTimeout's call site is global

        // as setTimout is window.setTimeout

        console.log(this.a)

      }, 0)

    },

    d: function() {

      // we can reassign this to \_this

      var \_this = this;

      setTimeout(function() {

        console.log(\_this.a)

      })

    },

    e: function() {

// Arrow function brings lexical scope

      setTimeout(() => {

        console.log(this.a)

      })

    }

  }

  check.b();

  check.c();

  check.d();

  check.e();

  </script>