**What is this?**

In JavaScript, the this keyword refers to an **object**.

**Which** object depends on how this is being invoked (used or called).

The this keyword refers to different objects depending on how it is used:

|  |
| --- |
| In an object method, this refers to the **object**. |
| Alone, this refers to the **global object**. |
| In a function, this refers to the **global object**. |
| In a function, in strict mode, this is undefined. |
| In an event, this refers to the **element** that received the event. |
| Methods like call(), apply(), and bind() can refer this to **any object**. |

this is not a variable. It is a keyword. You cannot change the value of this.

**this in a Method**

When used in an object method, this refers to the **object**.

In the example on top of this page, this refers to the **person** object.

Because the **fullName** method is a method of the **person** object.

<script>

*const* person = {

        firstName: "John",

        lastName: "Doe",

        id: 5566,

        person1: {

          name: this.firstName,

          person2: {

            nested: this.name,

          },

        },

      };

      console.log(person);

    </script>

**FOR LOOP IN OBJECT**

**OBJ** are not iterateable so we cannot apply any method of loop except for in loop to access the keys and by using for in we can also access the value with the use of brackets[] and dot[.] operator is not working.

<script>

*const* obj = {

        name: "hamza",

        class: "bsiT",

        id: 77,

      };

      for (*let* i in obj) {

        console.log(i);

      }

      for (*let* i in obj) {

        console.log(obj[i]);

      }

// . operator is not working

     for (*let* i in obj) {

        console.log(obj.i);

      }

    </script>