@Code.Clash

JavaScript this

Next →

Hey Everyones 👋

In this post, you will learn about JavaScript's this keyword with the help of examples.

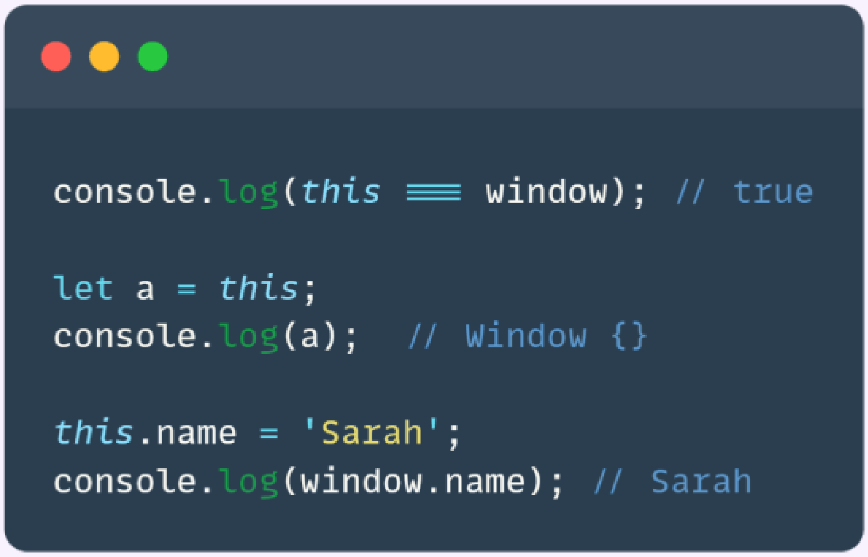
In JavaScript, this keyword refers to the object where it is called.

Do Like, save and Share This Post If You Found This Helpful.

NEXT →

this Inside Global Scope

- When this is used alone, this refers to the global object (window object in browsers).



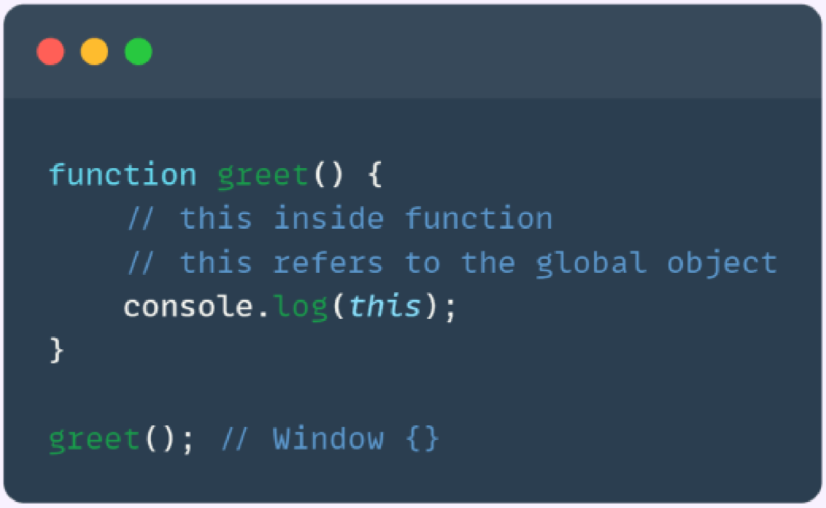
```
console.log(this === window); // true

let a = this;
console.log(a);  // Window {}

this.name = 'Sarah';
console.log(window.name); // Sarah
```

- Here, this.name is the same as window.name.

NEXT →

this Inside Function

- When this is used in a function, this refers to the global object (window object in browsers).

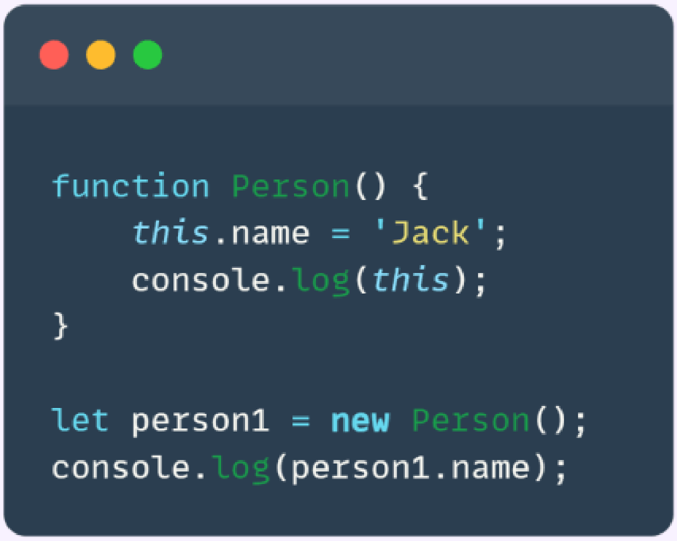


```
function greet() {
    // this inside function
    // this refers to the global object
    console.log(this);
}

greet(); // Window {}
```

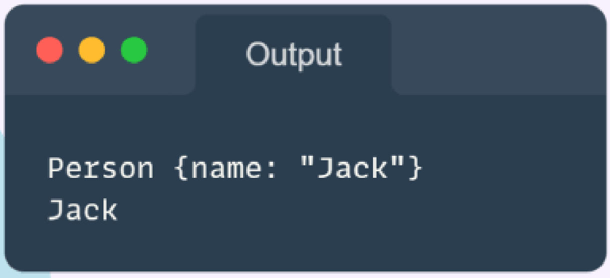
NEXT →

this Inside Constructor Function

- In JavaScript, constructor functions are used to create objects.
- When a function is used as a constructor function, this refers to the object inside which it is used.



```
function Person() {
    this.name = 'Jack';
    console.log(this);
}

let person1 = new Person();
console.log(person1.name);
```



```
Person {name: "Jack"}
Jack
```

NEXT →

this Inside Object Method

- When this is used inside an object's method, this refers to the object it lies within.

```
const person = {
    name : 'Jack',
    age: 25,

    // this refers to the object itself
    greet() {
        console.log(this);
        console.log(this.name);
    }
}
person.greet();
```

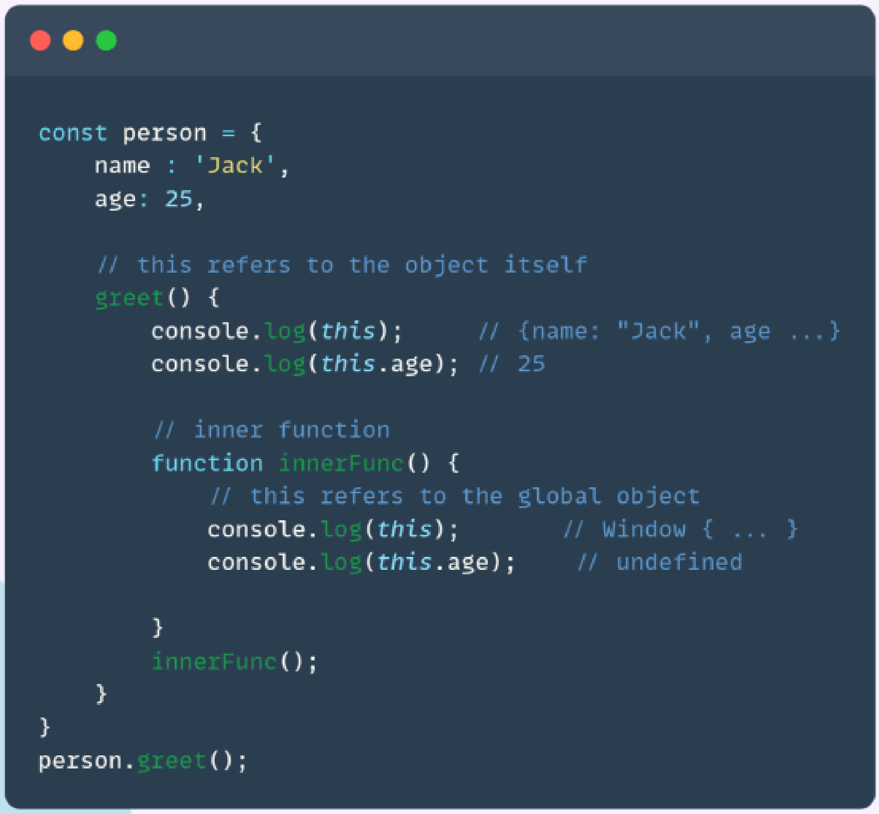
Output

```
{name: "Jack", age: 25, greet: ƒ}
Jack
```

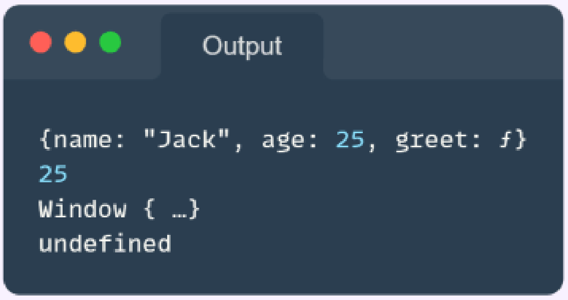
Next →

this Inside Inner Function

- When you access this inside an inner function (inside a method), this refers to the global object.



```
const person = {
    name : 'Jack',
    age: 25,

    // this refers to the object itself
    greet() {
        console.log(this);     // {name: "Jack", age ...}
        console.log(this.age); // 25

        // inner function
        function innerFunc() {
            // this refers to the global object
            console.log(this);       // Window { ... }
            console.log(this.age);    // undefined

        }
        innerFunc();
    }
}
person.greet();
```

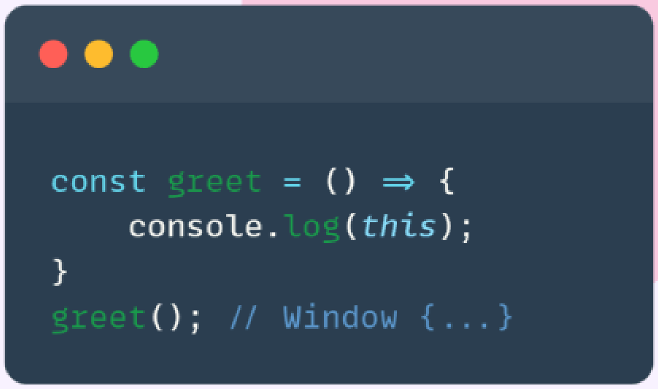


```
{name: "Jack", age: 25, greet: ƒ}
25
Window { …}
undefined
```

NEXT →

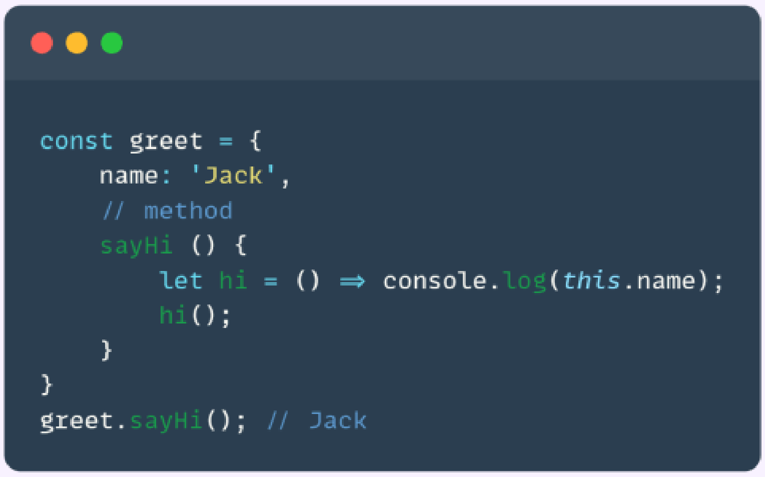
this **Inside Arrow Function**

- Inside the arrow function, this refers to the parent scope.



```
const greet = () => {
    console.log(this);
}
greet(); // Window {...}
```

- When you use this inside an arrow function, this refers to its parent scope object.

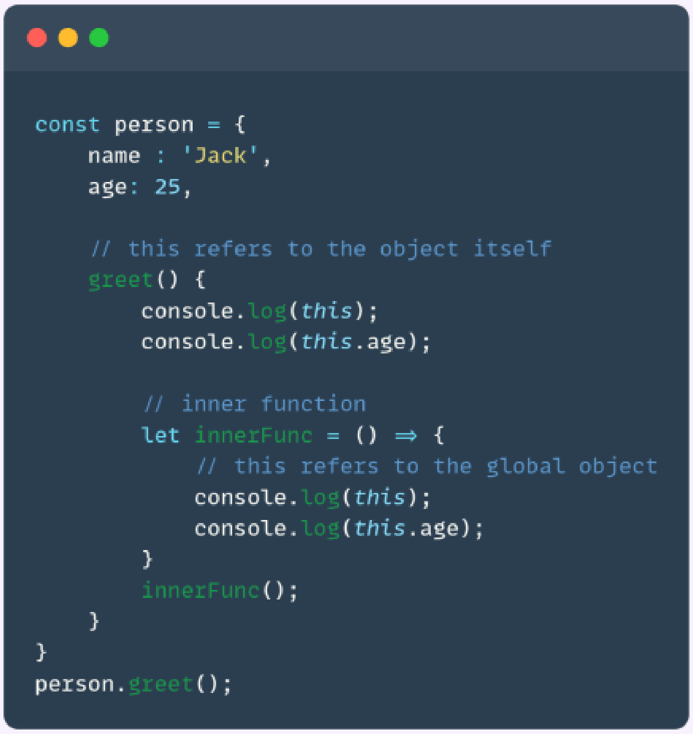


```
const greet = {
    name: 'Jack',
    // method
    sayHi () {
        let hi = () => console.log(this.name);
        hi();
    }
}
greet.sayHi(); // Jack
```

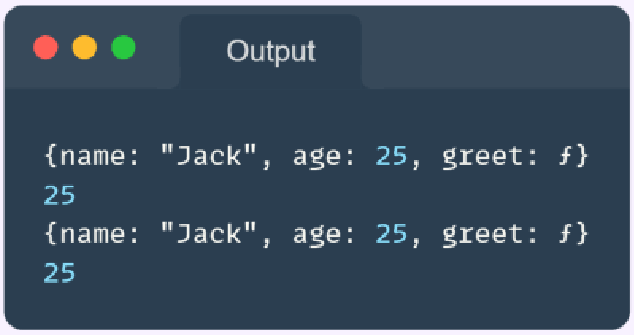
- Here, this.name inside the hi() function refers to the greet object.

NEXT →

- You can also use the arrow function to solve the issue of having undefined when using a function inside a method.



```
const person = {
    name : 'Jack',
    age: 25,

    // this refers to the object itself
    greet() {
        console.log(this);
        console.log(this.age);

        // inner function
        let innerFunc = () ⇒ {
            // this refers to the global object
            console.log(this);
            console.log(this.age);
        }
        innerFunc();
    }
}
person.greet();
```



```
{name: "Jack", age: 25, greet: ƒ}
25
{name: "Jack", age: 25, greet: ƒ}
25
```

- Here, innerFunc() is defined using the arrow function. It takes this from its parent scope. Hence, this.age gives 25.

@Code.Clash

THANKS FOR YOUR ATTENTION

Code.Clash

Imtiyaz Nandasaniya

LIKE AND SAVE IT

FOR LATER