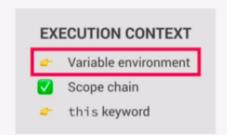
HOISTING IN JAVASCRIPT

Hoisting: Makes some types of variables accessible/usable in the code before they are actually declared. "Variables lifted to the top of their scope".



Before execution, code is scanned for variable declarations, and for each variable, a new property is created in the **variable environment object**.





TEMPORAL DEAD ZONE, LET AND CONST

```
const myName = 'Jonas';

If (myName === 'Jonas') {
    console.log(`Jonas is a ${job}`);
    const age = 2037 - 1989;
    console.log(age);

    const job = 'teacher';
    console.log(x);
}
TEMPORAL DEAD ZONE FOR job VARIABLE

** Different kinds of error messages:

ReferenceError: Cannot access 'job' before initialization

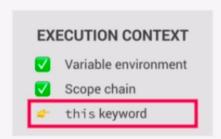
ReferenceError: x is not defined

**ReferenceError: x
```

46 people have written a note here.

HOW THE THIS KEYWORD WORKS

- this keyword/variable: Special variable that is created for every execution context (every function).
 Takes the value of (points to) the "owner" of the function in which the this keyword is used.
- this is **NOT** static. It depends on **how** the function is called, and its value is only assigned when the function **is actually called**.



```
Method this = <Object that is calling the method>
In strict mode! Otherwise: window (in the browser)

Simple function call this = undefined

Arrow functions this = <this of surrounding function (lexical this)>

Event listener this = <DOM element that the handler is attached to>

new, call, apply, bind
```

this does NOT point to the function itself, and also NOT the its variable environment!

Method example:

```
const jonas = {
  name: 'Jonas',
  year: 1989,
  calcAge: function()
  return 2037 - this.year
}

;
  jonas.calcAge(); // 48

calcAge  jonas  1989
is method
```

Way better than using jonas.year!

