Javascript-Behind the scence

JavaScript strict mode

- JavaScript is a loosely typed (dynamic) scripting language
- JavaScript allows strictness of code using "use strict" with ECMAScript 5 or later.
- Write "use strict" at the top of JavaScript code or in a function

```
"use strict";

var x = 1; // valid in strict mode
y = 1; // invalid in strict mode
```

```
<script>
    "use strict";
    y = 4;
    console.log(y);

    function add(a,a)
    {
        return a+a;
    }
    add(22,22);
</script>
```

SCOPE CHAIN

- **Scoping:** How our program's variables are **organized** and **accessed**. *"Where do variables live?"* or *"Where can we access a certain variable, and where not?"*;
- **Lexical scoping:** Scoping is controlled by **placement** of functions and blocks in the code;
- **Scope:** Space or environment in which a certain variable is **declared** (*variable environment in case of functions*). There is **global** scope, **function** scope, and **block** scope;
- **Scope of a variable:** Region of our code where a certain variable can be **accessed**.

TYPES OF SCOPE

GLOBAL SCOPE

```
const me = 'Jonas';
const job = 'teacher';
const year = 1989;
```

- Outside of **any** function or block
- Variables declared in global scope are accessible **everywhere**

FUNCTION SCOPE

```
function calcAge(birthYear) {
  const now = 2037;
  const age = now - birthYear;
  return age;
}

console.log(now); // ReferenceError
```

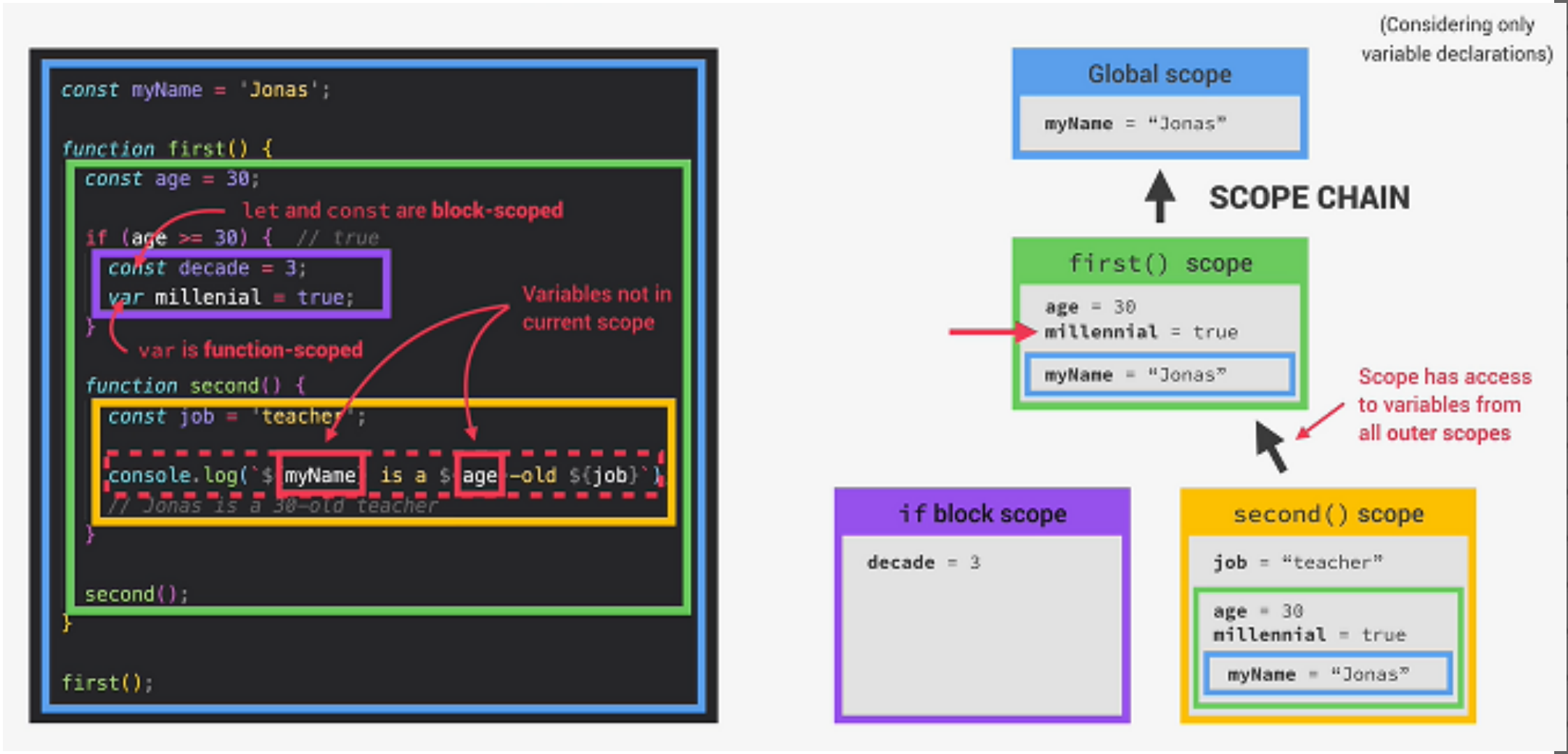
- Variables are accessible only **inside function**, **NOT** outside
- Also called local scope

BLOCK SCOPE (ES6)

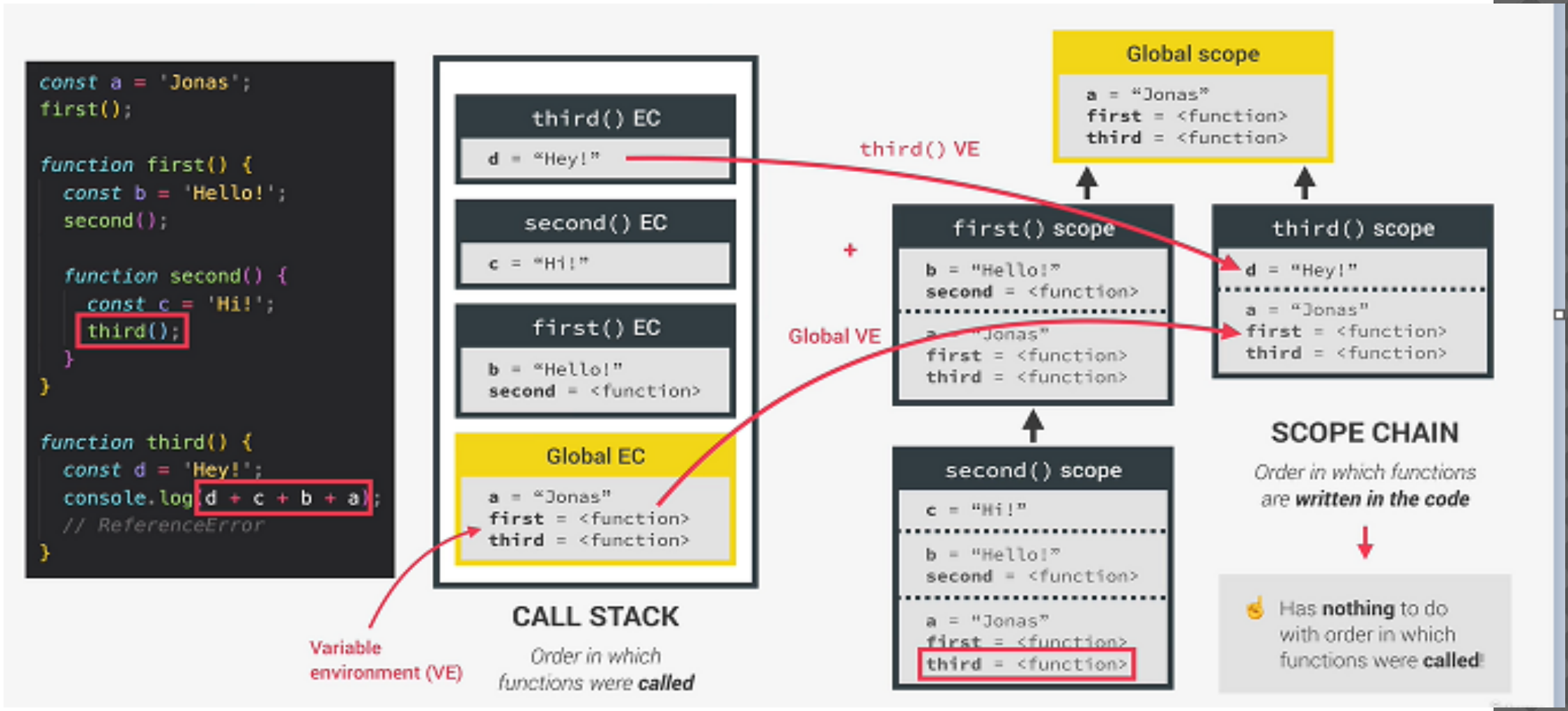
```
if (year >= 1981 && year <= 1996) {
  const millenial = true;
  const food = 'Avocado toast';
} ← Example: if block, for loop block, etc.

console.log(millenial); // ReferenceError
```

- Variables are accessible only **inside block** (block scoped)
- ⚠ **HOWEVER**, this only applies to **`let`** and **`const`** variables!
- Functions are **also block scoped** (only in strict mode)

SCOPE CHAIN


const myName = 'Jonas';
function first() {
const age = 30;
let and const are block-scoped
if (age >= 30) { // true
const decade = 3;
var millenial = true;
}
var is function-scoped
Variables not in current scope
function second() {
const job = 'teacher';
console.log(`${myName} is a ${age}-old ${job}`);
// Jonas is a 30-old teacher
}
second();
}
first();
(Considering only variable declarations)
Global scope
myName = "Jonas"
SCOPE CHAIN
first() scope
age = 30
millennial = true
myName = "Jonas"
Scope has access to variables from all outer scopes
if block scope
decade = 3
second() scope
job = "teacher"
age = 30
millennial = true
myName = "Jonas"

SCOPE CHAIN VS CALL STACK



SCOPE PRACTICE