SCOPE & SCOPE CHAIN.

PAGE NO.

* **Scoping :-** How our program's variables are organised & accessed.

* **Lexical scoping :-** Scoping is controlled by placement of functions & blocks in the code,
  e.g :- A function written inside another function has access to the variables of parent function.

* **Scope :-** space or environment in which certain variable is declared.

* **Scope of a Variable :-** Region of our code where a certain variable can be accessed.

* **Types of scope :-**

Global scope	function scope	Block scope (ES6)
• Outside of any function or block. • Variables declared in global scope are accessible everywhere.	• Variables are accessible only inside function, not outside. • Also called local scope.	• Variables are accessible only inside block. • let & const variables only. • functions are also block scoped. (only in strict mode)

Eg :-

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

E.g

```
function calcAge (birthYear) {
  const now = 2037;
  const age = now - birthYear;
  return age;
}
console.log(now); // Reference Error.
```

Eg of Block Scope (ES6)

```
if (year >= 1981 && year <= 1996) {
  const millenial = true;
  const food = 'Avocado toast';
}
```

↑ eg:- if block, for loop block etc.

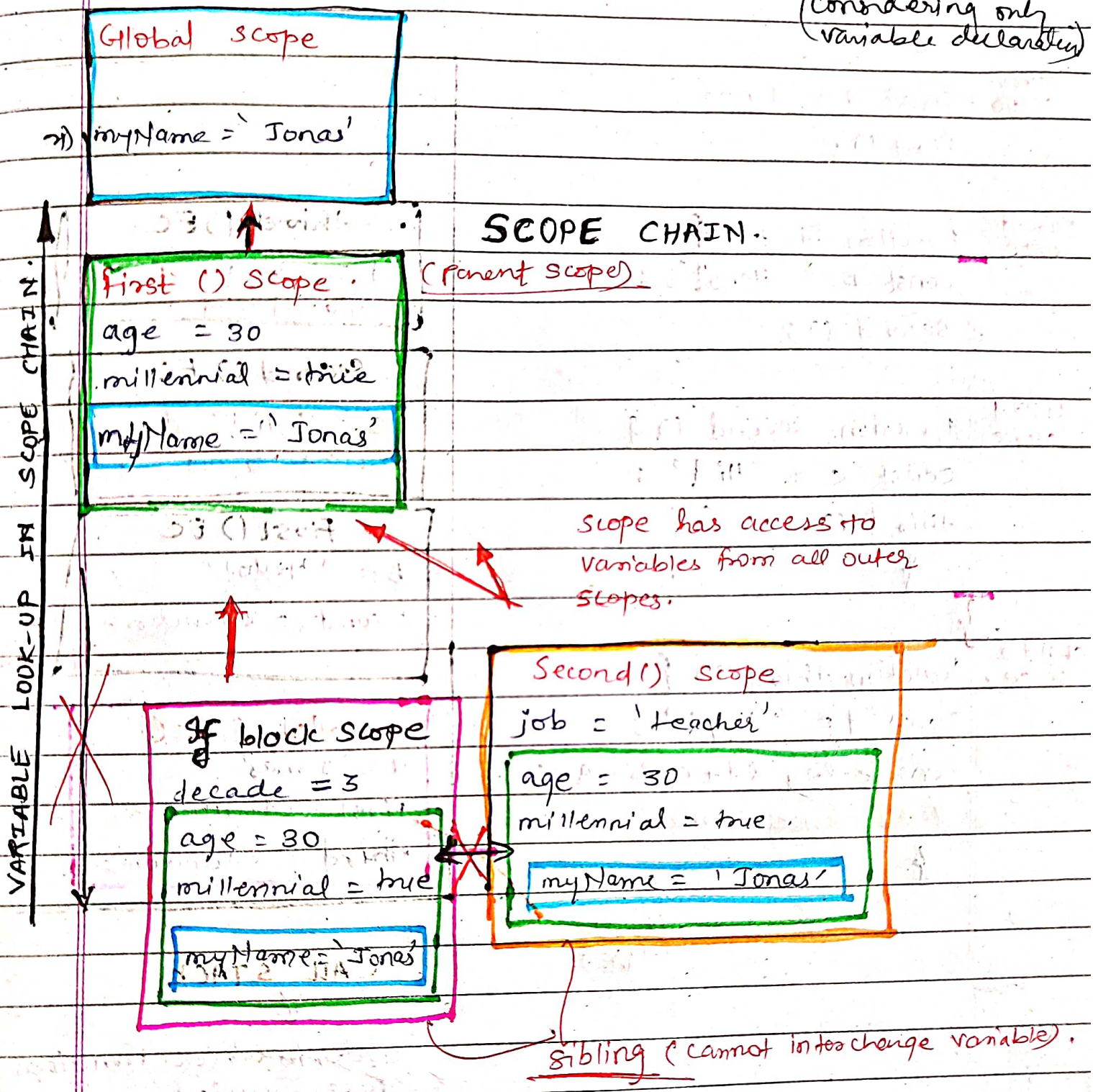
```
console.log (millenial); // Reference Error.
```

The SCOPE CHAIN :-

```
const myName = 'Jonas';
function first() {
  const age = 30;

  if (age >= 30) {
    const decade = 3;
    var millenial = true;
  }

  function second() {
    const job = 'teacher';
    console.log (`${myName} is a ${age}-old ${job}`);
    // Jonas is a 30-old teacher.
  }

  second();
}

first();
```

variables not in current scope.

(Considering only variable declaration)



SCOPE CHAIN VS CALL STACK.

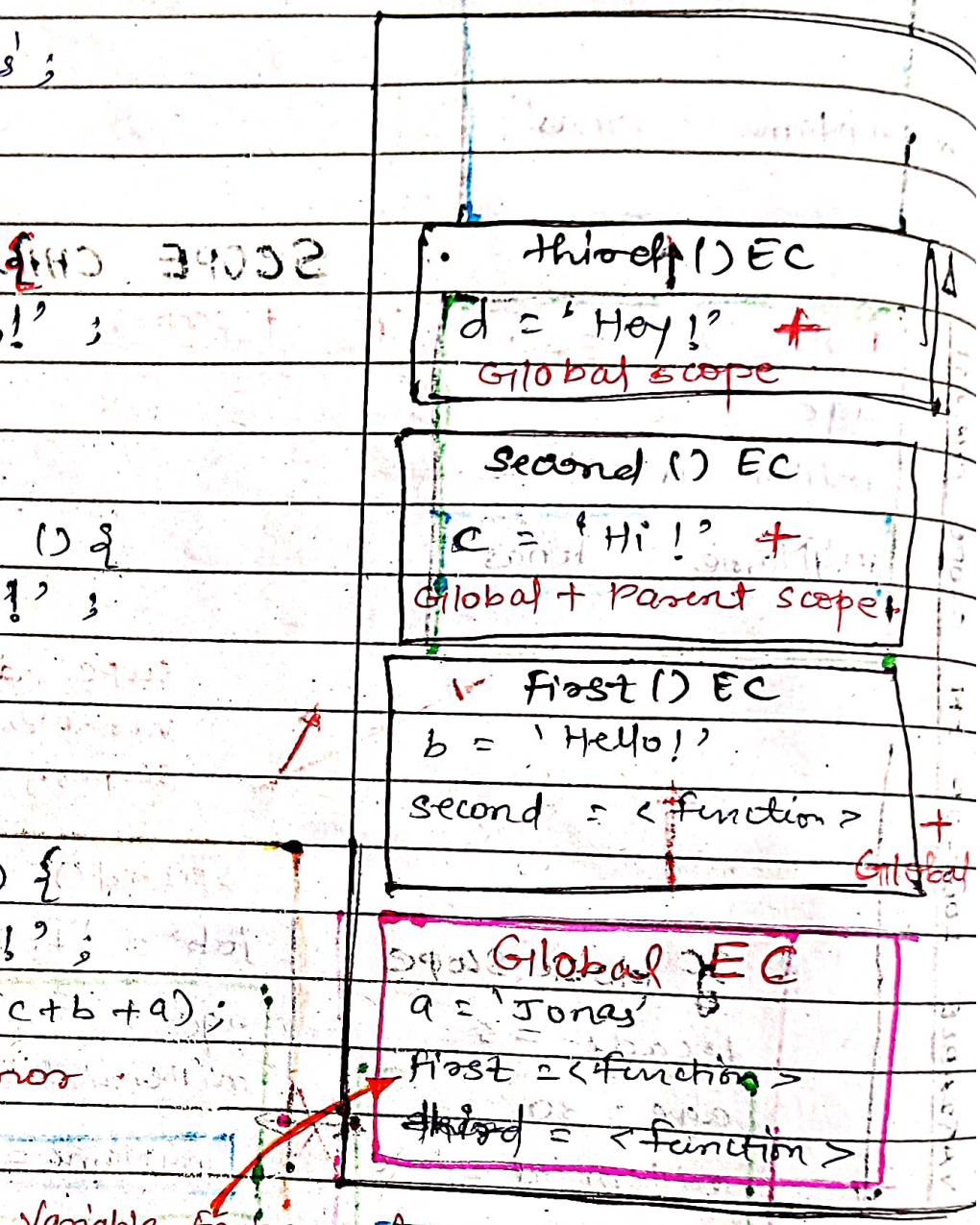
Global scope.

```
const a = 'Jonas';
first ();

function first () {
  const b = 'Hello!';
  second ();

  function second () {
    const c = 'Hi!';
    third ()
  }

  function third () {
    const d = 'Hey!';
    console.log (d+c+b+a);
    // Reference Error.
  }
}
```

Parent scope.

Child 1 scope.

Child 2 scope.



Variable Environment (VE)

CALL STACK

order in which functions were called.

Has nothing to do with order in which functions were called.

Pallavi · R. Sarkate

PAGE NO.

SUMMARY:

- Scoping – "where do variables live ? or where can we access a certain variable & where not ?".

- Three types of scope in JS :- Global, functions, Block scope ;
- Only Let & const variables are block-scoped . Variables declared with var end up in the closet function scope.
- Lexical scoping :- Rules of where we can access variables are based on exactly where in the code functions & blocks are written.
- Scope chain :- Every scope always has access to all the variables from all its outer scopes.

- When a variable is not in the current scope, the engine looks up in the scope chain until it finds the variable it's looking for. This is called Variable lookup.

- The scope chain is a one-way street : a scope will never, ever have access to the variables of an inner scope.

- The scope chain in a certain scope is equal to adding together all the variable environments of the all parent scopes.

- The scope chain has nothing to do with the order in which functions were called. It does not affect the scope chain at all!.