The scope chain.

→ Scope :- where can you ~~acess~~ access a particular variable or a function in our code.

·Lexical Environment.
whenever a excution context is created a lexical environment is also created.

Lexical Environment is the local memory along with lexical environment of its parent.

"Lexical":- means in a heirarchy or in a sequence

Scope chain :- The lexical environment chain is known as scope-chain.

Temporal dead zone :-

The time period from hoisting ~~to~~ till it get assign or it get some value. This is known as Temporal dead zone.

#LET

"Let" intialised variables are not stored in global space.

"Let" is little strict, you cannot re-intialize "Let variable again" you have to use another variable, otherwise the program won't even run, it will throw a syntax error.

e.g.)

```
let a = 10;
console.log(a);
let a = 100;
```

PROGRAM.

```
syntax error
```

OUTPUT

CONST

"const" is more strict then "let"

for example:- you intialize let variable and then you can input it's value anywhere in the program.

But in case of "const" you cannot do that you have to intialised at that point only otherwise the program will throw syntax ~~to~~ error.

e.g.)

```
let a;
const b;
b= 1000;
a= 10;
```

PROGRAM.

if we print console.log(a);

=> 10

if we print console.log(b);

=> syntax error: missing initalizer in const declaration.

⇒ If you try to re-intialise "const" then :-

e.g.)

```
let a = 1000;
const b = 1000;
 b = 100;
a = 10;
```

If console.log(a)

=) 10

If console.log(b)

=) TypeError: Assigned to constant variable.

③ index

Difference between syntax error and TypeError and Reference ~~er~~ error.

for syntax & Typeerror see above examples! (marked on ~~pg~~ button with index no.1)

Reference error :- this error comes the variable is in temporary dead zone.

e.g.)

```
console.log(a);   ] PROGRAM
let a = 1000;
```

Reference error] OUTPUT.

—————— × —————— × ——————

Ingeneral:- use const, and let.

If you want to use "var" then use it conciousey.

—————— × —————— × ——————

```
{
            ] → This is known as Block
}
```

This is use to club statements

e.g.) if(condition) [single statement].

↓

To add more statements we use { } these.