Top-right section:

Memory	Code

```
1) c1(x);
2) greet();
3) var x = 10
4) function greet(){
5)    console.log(Hello)
   }
1)
```

n: undefined
12
greet: f {
   }
hello: undefined

undefined
Hello
10

var hello = function(){
   c.l.("you guys rock")
}

Left-middle section:

```
1) function outer(){
2)    inner()
      function inner(){
         console.log(b);
3)
4)    }
5)
6)    }
      var b = 10;
      outer();
      console.log(b);
```

local scope

Memory creation | Code execution

1) Memory | 2) Code
inner: f {..}

Memory creation
outer: { }
b: undefined 10

10 | 10

CALL STACK

inner() EC
outer() EC
GEC

local mem + lexical env scope

Lexical Scope   lexical chaining

Right-middle section:

```
1) console.log(hello);
2)
3) var hello = function(){
4)    console.log("you guys rock");
5)
1) hello();
```

1) Mem   2) Code
hello: undefined
f {
...
}

undefined

you guys rock;

func expression → hello = 10 function
variable to store func

func
  └→ func   function (b,c){
                }

Yellow section (lower middle):

inner → local mem + parent at lexical env
outer at lexical env
→ grandpas
→ dads   outer at local memory
→ inner local memory   GEC at L.in
inner → inner local memory
outer at local mem   Scope

Scope chaining

Bottom-left section:

```
1) function outer(){
2)    inner();
3)    function inner(){
4)       console.log(b);
5)    }
6)    }
7)    var b = 10;
8)    outer();
9)    console.log(b);
```

Console
10 | 10

1) Mem   2) Code
{ x

inner at E.C
b) local mem + outer at Lexical env

1) Mem   Code
inner: f{..}
{ x

outer() at E.C.
local mem
i.e. + GEC

1) Mem   2) Code
outer: f{..}
b: undefined 10

GEC

GEC at Local mem