Homework

➤ Draw a lexical environment diagram for the right code and show:
  ➤ global lexical environment (LE)
  ➤ LE for makeArmy()
  ➤ LE for LE of the while loop
  ➤ LE for army[0]
  ➤ What will army[0] alert?
  ➤ Can you fix the code?
  ➤ How will the diagram change?

```
function makeArmy() {
    let shooters = [];
    let i = 0;
    while (i < 2) {
        let shooter = function() {
            alert(i);
        };
        shooters.push(shooter);
        i++;
    }
    return shooters;
}
let army = makeArmy();
army[0];
```
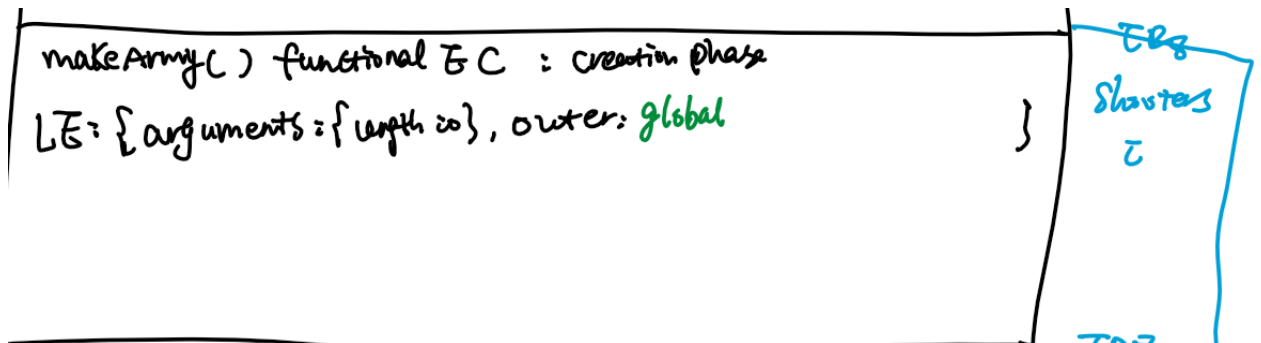
➤ global lexical environment (LE)

  ➤ creation phase


```
global Execution Context : creation phase
LE: { makeArmy: fn , outer = null,
this: window
}                                         army
```
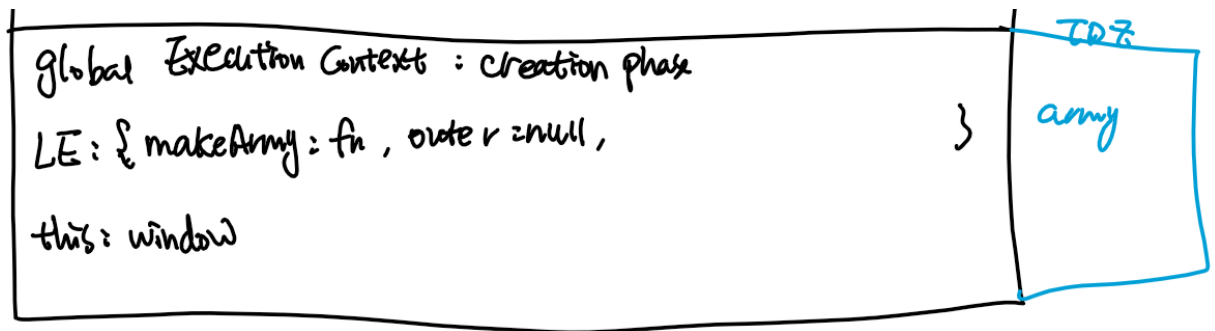
  ➤ execution phase

➤ LE for makeArmy()

  ➤ creation phase


```
makeArmy() functional EC : creation phase
LE: { arguments: { length :0}, outer: global
}                                    shooters
                                        i
```
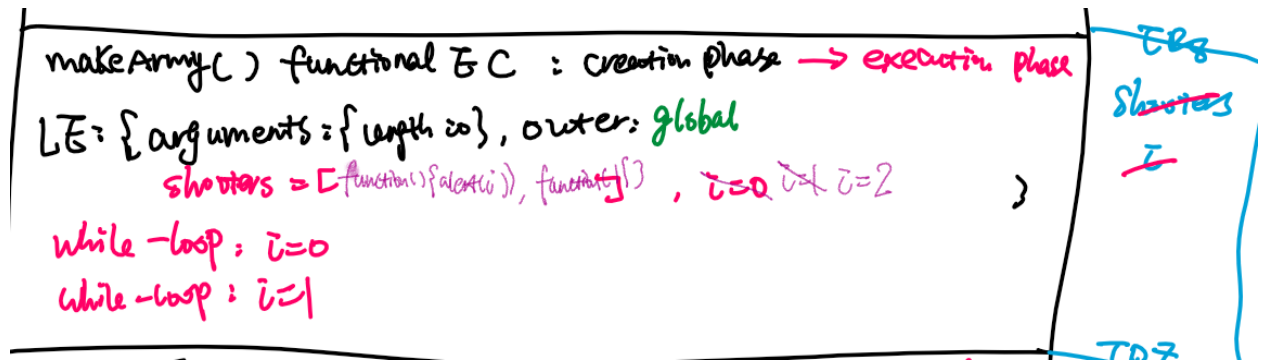
  ➤ executon phase

    ➤ After while-loop, changed the condition to while(i<2) to save time. The LE is being changed as below

makeArmy() functional EC : creation phase → execution phase

LE: { arguments : { length :0 }, outer: global
        shooters = [function () { alert(i) }, function () {} , i=0 :=1 :=2]
while -loop : i=0
while -loop : i=1

ERs
Shooters
I

TDZ

➢ LE for LE of the while loop

  ➢ Each iteration of while loop has own LE, only use i=0 as example here.

  ➢ creation phase

While -loop EC : creation phase

LE: { outer: makeArmy }

TDZ
Shooter

ERs
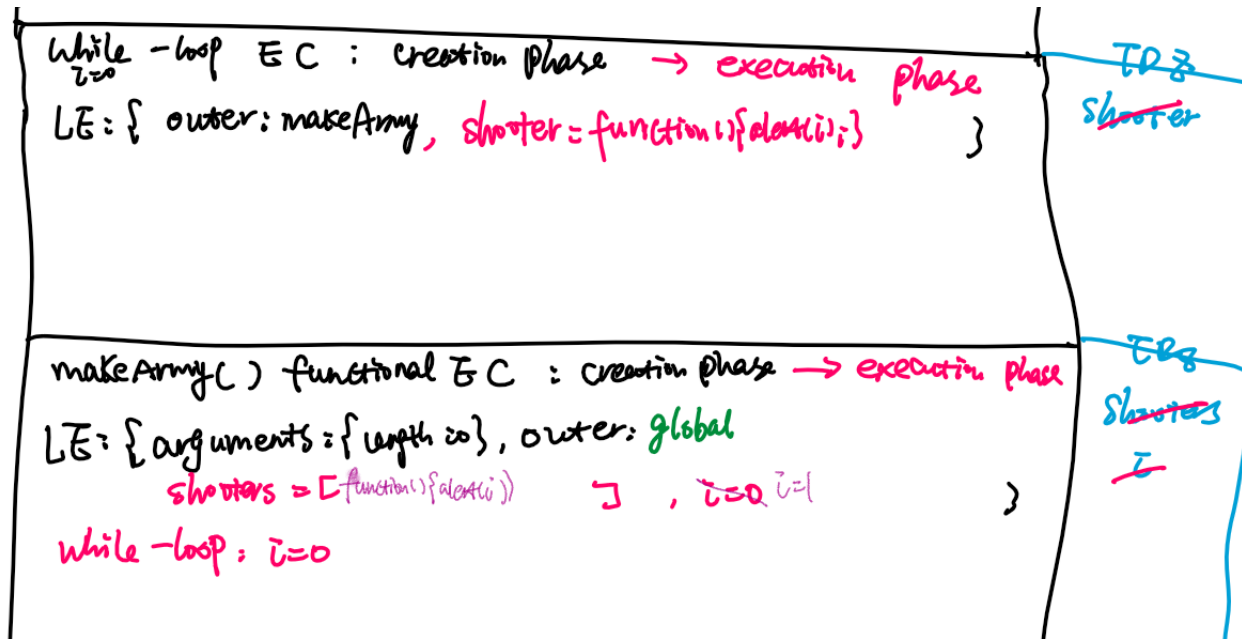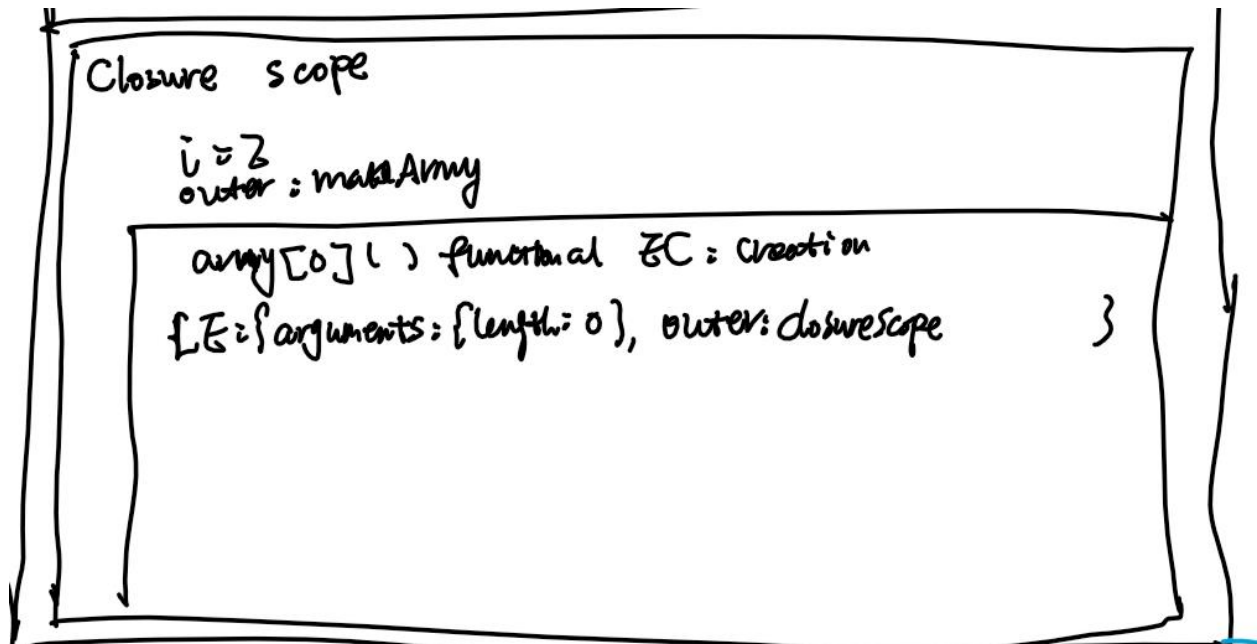
➢ executon phase

- shooters.push(shooter);
- i++;
  The two statements above will cause the changes in makeArmy() functional EC

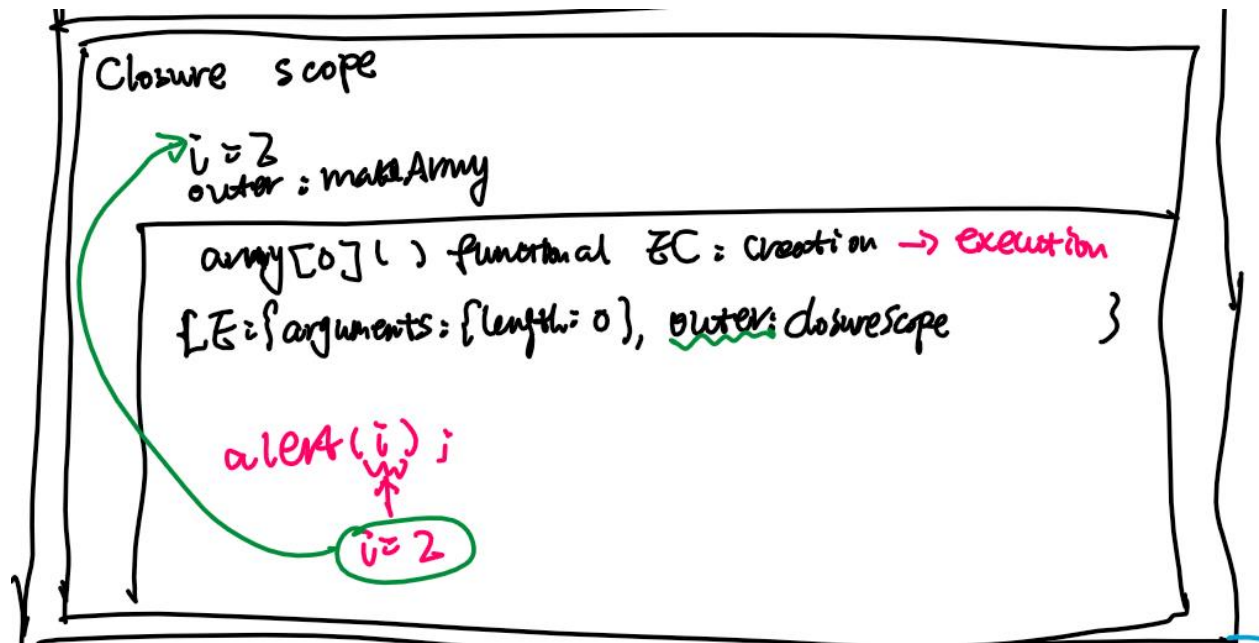while-loop EC : creation phase → execution phase
$z=0$
LE: { outer: makeArmy, shooter = function(){alert(i);} }

~~IR~~ ~~Shooter~~

makeArmy() functional EC : creation phase → execution phase
LE: { arguments: { length: 0 }, outer: global
shooters = [function(){alert(i)}] , $i=0$ $i=1$ }
while-loop: $i=0$

~~ER~~ ~~Shooters~~ ~~i~~

- ➤ LE for army[0]()

  - ➤ creation phase

Closure scope

$i = 3$
outer : makeArmy

army[0]() functional EC : creation
LE: { arguments: { length: 0 }, outer: closurescope }

  - ➤ executon phase

Closure scope

i = 2
outer : makeArmy

army[0]( ) functional EC : creation → execution
{EC:{arguments: {length: 0}, outer: closureScope   }

alert(i);

i = 2

➢ What will army[0]() alert?

2

➢ Can you fix the code?

```javascript
function makeArmy() {
    let shooters = [];
    let i = 0;
    while (i < 2) {
        let j = i;
        let shooter = function() {
            console.log(j);
        };
        shooters.push(shooter);
        i++;
    }
    return shooters;
}
let army = makeArmy();
army.forEach(f => f());
```

➢ How will the diagram change?

```
                                    while iteration
shooters = [                        LexicalEnvironment
   function () { alert(j); }, ➡    j: 0
   function () { alert(j); }, ➡    j: 1        makeArmy()
   function () { alert(j); }, ➡    j: 2    outer LexicalEnvironment
                                            ➡    ...
   ...

   function () { alert(j); } ➡      j: 10
];
```