* Creation phase

global Execution Context: creation phase

LE: { MakeArray: fn, outter: null, this: window

* execution phase

global EC : creation → execution

LE: { MakeArray: fn, outter: null, array: [ fn, fn ] }

makeArray()

array[0] = function{alert};

* creation phase

MakeArray() functional EC: creation phase

LE: { arguments: {length}, outer: global }

* execution phase After while-loop

makeArray() functional EC: creation phase
execution phase

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

4- what will army[0]() alert?

2

5 Fix the code?

```
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());
```

# LE for LE the while loop

while-loop EC: creation phase

LE: { outer: makeArray }

↑ ~~execution phase~~

~~wh~~:

* LE for Army[0]()

creation phase

closure scope

i = 2

outer: makeArmy

army[0]() functional EC:

creation

LE: { arguments: { length: 0 },

outer: closure scope

execution phase

i = 2

outer: makeArmy

array[0]() functional EC: creation → Execution

LE: { arguments: { length: 0 }, outer: closure scope

Q2

```
function printNumbers(from, to) {
  let current = from;
  let timerId = setInterval(function() {
    alert(current);
    if (current == to) {
      clearInterval(timerId);
    }
    current++;
  }, 1000);
}
printNumbers(5, 10);
```

Q3 setTimeout runs after the code below is finished

```
let i = 0
setTimeout(() => alert(i), 100);
for (let j = 0; j < 1000000000; j++) {
  i++;
}
```