Assignment 8: Closure and Execution context			
While-loop Execution Context	execution	TDZ	
LE: {this: window, Outer: makeArmy, arg: 0, length: 1	i=2; while(2!<2), shooters returned, while loop finished at resume stage i =2, => return is empty		shooters executed (returned)
While-loop Execution Context	Execution	TDZ	
LE: {this: window, Outer: makeArmy, arg: 0, length: 1	i=1; while(1<2), shooter = function(){alert(i);} shooters.push() = [],		Paused resumed
While-loop Execution Context	execution	TDZ	
LE: {this: window, Outer: makeArmy, arg: 0, length: 1}	i=0; while(0<2), shooter = function(){alert(i);} shooters.push() = [],	~~shooter~~	Paused resumed
makeArmy(): Execution context: Creation	execution	TDZ	paused
LE: {this: window, outer: global	Shooters = []; i= 0; while (i < 2)	~~Shooters, i~~	
Global execution Context: Creation	Execution	TD Z	paused
LE:{makeArray: fn, this:Window, Outer:null	army = makeArmy(); }	army	

Questions to complete execution stack:

1. Are there variables in the global lexical environment declared using 'var'?
2. Are there function declarations in the global lexical environment?
3. Are there variables and functions created using const and let?
4. What is the outer environment for the current lexical environment?
5. What does the 'this' binding refer to?
6. Are there arguments to functions?

Home Work Questions

Draw a lexical environment diagram for the right code and show:

➢ global lexical environment (LE) - The table above

➢ LE for makeArmy() – The table above

➢ LE for the while loop - The table above

➢ LE for army[0] - TDZ

➢ What will army[0] alert? - outputs f(){alert(i)};

➢ Can you fix the code? Code is fixed yellow shaded

➢ How will the diagram change? – As below using closure diagram

```
function makeArmy() {
let shooters = [];
let i = 0;
while (i < 2) {
let shooter = function() {
alert(j);
};
shooters.push(shooter);
i++;
} return shooters;
}
let army = makeArmy();
army[0] (call correction)
```

Closure scope

at the end of while loop we get i = 2,

outer global,

Army() – Execution context,

Creation:

LE: {outer: makeArmy() =>closure scope ,

Since makeArmy() is not there after shooters is returned, use closure scope

Execution: while I = 2 alert(i);

Code correction

```
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());
```

Shooters = [//while loop iteration lexical environment

function() { alerst(j);}, => j = 0

function() { alerst(j);}, => j = 1 =>outer makeArmy() lexical environment

];