```
        return 10
    })()
}
var result=fn()
result ?
```

- (●) 20
- ( ) 10
- ( ) 0
- ( ) Error

Ans: A

2.

array find method return value (if element there)?

- (●) element
- ( ) undefined

Ans: A

3.

document.getElementsByClassName return type

- ( ) number
- (●) list

- ch
- ""

Ans: A

5.

Array pop method take how many arguments

- exactly 1
- exactly 2
- exactly 3
- No args

Ans: D

6.

true+false+null

- 1
- 0
- 2
- None

Ans: A

8.

```
const obj=new Promise(function(a,b){
    a(100)
    b(20)
})
const res=await obj;
```

res ?

- 100
- 20
- Error
- undefined

Ans: A

9.

```
function fn(){
  return [10,20]
}
const [a,b]=fn()
```

a value ?

array

Ans: A

11.

```
var arr=[22,33,44,55]

arr.filter((val)=>{
  return val<10
})

?
```

- [] (selected)
- undefined
- null
- Error

Ans: A

12.

```
function fn(){
 console.log("fn called");
}
```

if(a){
console.log("statements");
}

statements printed or not ?

- (●) No
- ( ) Yes

Ans: A

14.

var o1={n1:10}

var o2={n2:20}

var o3={...o1,..o2,n1:100,n3:30}

o3 ?

- ( ) Invalid syntax
- ( ) {n1:100,n2:20}
- (●) {n1:100,n2:20,n3:30}
- ( ) n2 is not defined

Ans: C

16.

```
try{
    console.log(1);
    var loc;
    console.log(loc.length)
   console.log(2)
}catch(e){
  console.log("catch");
}
finally{
  console.log("finally")
}
```

output in console. ?

- 1 2 catch
- 1 catch finally
- 1 2 catch finally

Ans: B

18.

Below which data type referes functional scope ?

- var
- let
- const
- state

Ans: A

19.

```
function f1(){
   const n1=n2=100
}
f1()
console.log(n1)
```

output ?

- 0
- 100
- n1 is not defined
- ""

WRITE AGAIN