Javascript interview

1.variable , let , constant :

Variable :

Variable are used declare a data with type . It container for storing data(storing data values).

Example :

```
var a = " f" // datatype-  string
var num = 30; // datatype- number
var myData=undefined // datatype- undefined
// obj={} key:value
var m=[{ name:"dog"}] // datatype- object
var isavail= true //datatype-  boolean
var isavai= true||false //datatype- boolean
var intnum = null;  // datatype- object
var b= [ 1,2,2]//datatype-  string
```

Link to get code in github:
https://github.com/rashyandezhilan/js-topic--interview

Let :

Let are excuted in block level . let are can redeclare and can't assign in alread declare variable .

Example :

Let a = " happpy" // o / p happy
Let a = " hello " // o/p hello
Let a = " joy " // o/p a has already declared .

Link to get code in github:
https://github.com/rashyandezhilan/js-topic--interview

constant :

Constant are variable that are once declared can't be re-assign and can't be re-declared .

Examlpe :

Const a = " good" // o/p good
Const a = " bee " // error 'a' has already declared .

Link to get code in github:
https://github.com/rashyandezhilan/js-topic--interview

2. Closure

Closure are function can access all variable defined inside the function but also can access defined in outside in function.
Example :

```
function inti(){
    var nam= " moooo" // parent scope

function displayname(){ // child scope  call them
    console.log(nam);

}
displayname();

}
inti();
 //output  // mooooo
```

Link to get code in github:
https://github.com/rashyandezhilan/js-topic--interview

3. Hoisting :
Hoisting are refers to the appears to move in declaration , variable, class to top of the scope, prior to execution of code.
Example:

```
var a;
a=10;
{
    console.log(a)
}
//output
// 10
```

Link to get code in github:
https://github.com/rashyandezhilan/js-topic--interview

4.callback :
Callback function are excuted function are one by one . wait peroid are long time excuted particular fnction.

Link to get code in github:
https://github.com/rashyandezhilan/js-topic--interview

example:

```
   var space = " "
function getname(space){
    console.log("hi"+ " rashyand");

}
function callback ()
{
    console.log(" thank u")
}
callback(); // thank u
getname(space); // hi rashyand
```

5.promies :

Promies are function like callback but similary little different in promise function.
The promise are function can't waiting for other excution funtion.
It check reslove ,rejection that are reslove means ture , rejection means false
The promise are use two keywords reslove,rejection .

Example :

```
var weekendday = new Promise ((reslove,reject)=>{
    var sum = 20;
    if(sum == 20){
        sum= sum + 10 ;
        console.log("sum is" + sum );
        reslove(" sucesss");
    } else {
        console.log("better luck sometimes ")
        reject(" unsucess")
    }
})
.then((add)=>{
    console.log(add)

})
.catch((err)=>{
    console.log(err)
})
// output //
//sum = 20    =>  sum is 30  ,successs
// sum = 10   =>  better luck next time , unsuccess
```

Promise methods :

Promise method are 6 methods are reslove, rejection,all, allsettled,any, race .
 Example:

```
const car = new Promise((reslove, reject) => {
    setTimeout(() => {
        reslove(" car : volvo ")

    }, 100)
})
const bike = new Promise((reslove, reject) => {
    setTimeout(() => {
        reslove(" bike : h2r")

    }, 200)
})
const truck = new Promise((reslove, reject) => {
    setTimeout(() => {

        reslove(" truck : benz")

    }, 400)

})
Promise.allSettled([car, bike, truck]).then((res) => {
    console.log(res)
})
    .catch((error) => {
        console.log(error)
    })
    .finally(() => {
        console.log(" sucesss")
        console.log(" promies property")
    })
//1
//all - 0: " car : volvo "
      //1: " bike : h2r"
      //2: " truck : benz"
      // sucesss
       //promies property
```

```
//2. allsettled
//0: {status: 'fulfilled', value: ' car : volvo '}
//1: {status: 'fulfilled', value: ' bike : h2r'}
//2: {status: 'fulfilled', value: ' truck : benz'}
//length: 3[[Prototype]]: Array(0)
//  sucesss
//  promies property
//3.any
//car : volvo
//  sucesss
//  promies property
//4.race
//car : volvo
//  sucesss
//  promies property
```

Link to get code in github:
https://github.com/rashyandezhilan/js-topic--interview

6. Async, await function
   Async : it operator asynchornorus via event- loop . Async function are will return return value.
   Await: use async keyword should be use in await within the block of execution . await keyword is to assign a value or return   in assign a variable .
Example :

```
var hell = async()=>{
    var a = 10 ; // declare variable a
    var b = 10;  // eclare variable b
    var result = a + b ;
    if(result == 20){ // we use if conduction .
        result = a + b ;
        console.log(" the sum are is  " + result );
        result = " achieve";
    }
    else
    {
        result= " stop"
        console.log(" its wrong")
    }
    var add= await result; // result value assgin to variable add.
    return add;  // return  add
    }
hell().then((add)=>{
    console.log(add)
})
.catch((error)=>{
    console.log(error)
}) // we use finally - means its display the reslove sucessfully & then
display a msg.
```

Link to get code in github:
https://github.com/rashyandezhilan/js-topic--interview

8. Call , apply , blind fuction .
 Call () :
   Call function :
- pre-defined js method.
- used to invoke a method a with an owner object as an argument .

Example:

```
const country = {
    countryname: function () { // create fuction for name of country name
     return this.country + " "
    }
}
const place1 = {
    country: ' uk'


}
const place2 = {
    country  : ' ussr'


}
console.log(country.countryname.call(place1)); // we call place 1

```

Link to get code in github:
https://github.com/rashyandezhilan/js-topic--interview

apply() :
  apply() method are similary to call() method
  call() - take argument separately.
  apply() - take argument as an array
  Comparisipon apply are very useful than call() .
 example:

```
const country1 = {
    countryname: function (speed,color) { // argument (speed,color)
        return this.country + " " ;
    }
}
const place3 = {
    country: ' uk'
}
const place4 = {
    country  : ' ussr' // output  uk 5 red
}
console.log(country1.countryname.apply(place3,[5,'red']));
```

Link to get code in github:

https://github.com/rashyandezhilan/js-topic--interview

Blind () :
blind() method are an object can borrow a methiod from an object .

Example:

```
var inform = {
    company : " abbc" ,
    form:function(){
        return " company  is " + " " + this.company
  }
}
var inform1 = {
    company : " bbc",          // output company is bbc
}
var detail = inform.form.bind(inform1) // we blind inform to inform 1
var x  = console.log.bind(Document) // blind console.log()  in variable x
x(detail())
```

Link to get code in github:

https://github.com/rashyandezhilan/js-topic--interview

9. Class :

Class -keyword create a class .always add a method constructor() . declare variable to to create an object .

Example:

```
class Test {
    foo = 123;    // declare foo
    constructor() {
        this.foo = "234"; re- declare foo  it become override on them
    }
}
const test = new Test(); // create new object Test
console.log(test.foo); // o/p 234
```

Inheritance :

Inheritance are use another class property using keyword extends

Class -keyword create a class .always add a method constructor() . anothe class porperty use on them.

Example:

```
  class Profile
    constructor() {
        this.name = " rashyand"; // this -keyword refer a object
        this.class = " pg ";
        this.place = " coimbatore";
    }
    // method - methodname (detail)
    detail() {
        return " my name is " + this.name + " degree is " + this.class
    }
}  // output  my name is rashyand degree is pg

//concept - inhertiance
class Profile1 extends Profile {
    constructor(place) {
        super(place)
// super keyword - refer use in inhertiance  to access an anothe class
porperty.
        this.college = " Gasc"
    }
    add() {
        return " my profile update is " + this.college
    }
}
const profoloic = new Profile();
console.log(profoloic.detail()); //  class my name is  rashyand degree is
pg
const profoloic1 = new Profile1();
console.log(profoloic1.add());  // my profile update  is Gasc
```

Link to get code in github:
https://github.com/rashyandezhilan/js-topic--interview

10 . ternary operator :

Ternary operator - ?

It shorthand of if condtion - we use code witten in single line.

Example:

```
 //  ternatry operator condition ? ///
   var vi = 20 ;
   var outside = vi < 19? ' 20 is greater' : '20 is lesser' // condition ?
//  20 is lesser
   var outside = vi > 19? ' 20 is greater' : '20 is lesser' // 20 is
greater
  //console.log( outside)
```

Link to get code in github:
https://github.com/rashyandezhilan/js-topic--interview