1.  let randomValue={name:"birch"};

        randomValue=13;

        if(!typeof randomValue==="String"){

            console.log("not an String");

        }

        else{

            console.log("String");

        }

o/p-

String

2. const createMember=({email,address={}})=>{

            const valid=/.+\@.+\..+/.test(email);

            if(!valid){

                throw new Error("invalid email");

            }

                return {email,address:address?address:null};

        }

        const member=createMember({email:"my@email.com"})

        console.log(member);

o/p-

*{email: 'my@email.com', address: {…}}*

* 1. **address**: {}
  2. **email**: "my@email.com"
  3. [[Prototype]]: Object

3. const keys=["name","age"];

const values=["birch",12];

const method=/\* ?? \*/

Object[method](keys.map((\_,i)=>[keys[i],values[i]]));

o/p-

Task.html:39 Uncaught ReferenceError: Cannot access 'method' before initialization

at Task.html:39:8

(anonymous) @ Task.html:39

4. const promise1=Promise.resolve("first");

const promise2=Promise.resolve("second");

const promise3=Promise.resolve("third");

const promise4=Promise.resolve("fourth");

const runPromise= async ()=>{

    const result1=await promise1;

    const result2=await promise2;

    return [result1,result2];

}

runPromise().then(result=>console.log(result)).catch(err=>console.log(err));

o/p-(2)  ['first', 'second']

**0**: "first"

**1**: "second"

**length**: 2

[[Prototype]]: Array(0)

5. const user={

    email:"my@email.com",

    updateEmail:email=>{

        this.email=email;

}

}

user.updateEmail("new@email.com");

console.log(user);

1. o/p-*{email: 'my@email.com', updateEmail: ƒ}*
   1. **email**: "my@email.com"
   2. **updateEmail**: *email=>{ this.email=email; }*
   3. [[Prototype]]: Object

6. const fruit=["banana","orange","apple"];

fruit.slice(0,1);

fruit.splice(0,1);

fruit.unshift("grapes");

console.log(fruit);

o/p-(3) ['grapes', 'orange', 'apple']0: "grapes"1: "orange"2: "apple"length: 3[[Prototype]]: Array(0)

7.

const user={

    email:"e@email.com",

    password:"12345",

}

const updateUser=({email,password})=>{

if(email){

    Object.assign(user,{email});

}

if(password){

    user.password=password;

   }

   return user;

}

const newUser=updateUser({email:"new@email.com"});

console.log(newUser===user);

o/p-true

8. class Calc {

    constructor(){

        this.value=0;

    }

    increment(){

        this.value++;

    }

}

const calc=new Calc();

new Calc().increment();

console.log(calc.value);

o/p-0

9a. let count=0;

const nums=[0,1,2,3];

nums.forEach(num=>{

    if(num){

        count+=1;

    }

});

console.log(count);

o/p-3

9b. class Bird{

    constructor(){

        console.log("bird");

    }

}

class Flamingo extends Bird{

    constructor(){

        super();

        console.log("pink");

    }

}

const pet=new Flamingo();

o/p-bird

pink

10. const person={

    name:"birch",

    hobbies:["coding"],

}

function addHobby(hobby,hobbies=person.hobbies){

    hobbies.push(hobby);

    return hobbies;

}

addHobby("running",[]);

addHobby("dancing");

addHobby("baking",person.hobbies);

console.log(person.hobbies);

o/p- ['coding', 'dancing', 'baking']

11. const teams=[

    {name:'Team1',members:['birch','birch1']},

    {name:'Team2',members:['birch3','birch2']},

]

function\* getMembers(teams){

    for(let i=0; i<members.length; i++){

       yield members[i]

    }

}

function\* getTeams(){

    for(let i=0; i<teams.length; i++){

    }

}

const obj=getTeams(teams);

obj.next()

obj.next()

12. class counter{

    #number=10;

    increment(){

        this.#number++;

    }

    getNum(){

        return this.#number;

    }

}

const c=new counter();

c.increment();

console.log(c.#number);

o/p-error

13. const myPromise=Promise.resolve(Promise.resolve('Promise'));

function funcOne(){

    setTimeout(()=>{

        console.log('TimeOut 1!');

    },0);

    myPromise.then(res=>res).then(res=>{

        console.log(`${res} 1!`);

    });

    console.log('Last 1!');

}

async function funcTwo(){

    const res=await myPromise;

     setTimeout(()=>{

        console.log('TimeOut 2!');

    },0);

    console.log(`${res} 2!`);

    console.log('Last 2!');

}

funcOne();

funcTwo();

o/p-Last 1!

Promise 2!

Last 2!

Promise 1!

TimeOut 1!

TimeOut 2!

14a. const name="birch";

console.log(!typeof name==="Object");

console.log(typeof name==="String");

o/p-false,false

14b. const myMap=new Map();

const myFunc=()=>"greeting";

myMap.set(myFunc,"hello");

myMap.get("greeting");

myMap.get(myFunc);

myMap.get(()=>"greeting");

no o/p

15. const emoij=["star","vodka","simliy"]

emoij.map(x=>x+"star")

emoij.filter(x=>x!=="vodka");

emoij.find(x=>x!=="vodka");

emoij.reduce((acc,cur)=>acc+"star");

emoij.slice(1,2,"star");

emoij.splice(1,2,"star");

no o/p

16. const set = new Set();

set.add(1);

set.add('lydia');

set.add({ name: "ludia" });

for (let item of set) {

    console.log(item +2)

}

o/p-3

lydia2

[object Object]2

17. function getItems(fruitList, ...args, favoriteFruit) {

return [...fruitList, ...args, favoriteFruit]

}

getItems(["banana", "apple"], "pearl", "orange")

o/p-error

18. xport default () => "hello world";

export const name = "Lydia";

import \* as data from './LogicalProgram'

console.log(data)

o/p-error

19. const box = { x: 10, y: 20 };

Object.freeze(box);

const shape = box;

shape.x = 100;

console.log(shape)

o/p-{ x: 10, y: 20 }

20a. fetch('https://www.website.com/api/user/1')

.then(res => res.json())

.then(res => console.log(res))

o/p-error

20b. console.log(Number(2) === Number(2));

console.log(Boolean(false) === Boolean(false));

console.log(Symbol("foo") === Symbol("foo"));

o/p- true

true

false