

Question 1 : Answer The Following Questions [12 Points]

- 1- What do you understand by hoisting in JavaScript?
- 2- Why is super used in JavaScript?
- 3- What is let and const? And how it differs from var?
- 4- Discuss the Rest parameter in ES6 What is Arrow function?
What are all its uses? How it differs from normal function?
- 5- What is the difference between the readonly and disabled attributes for the <textarea> element
- 6- How do you specify units in the CSS?. What are the different ways to do it?
- 7- What property is used for changing the font face?
- 8- How to center align a div inside another div? [2 Ways]

Question2: True Or False [16 points]

- 1- Encapsulation is a mechanism which represent the essential features without including implementation details. **False**
- 2- Encapsulation lets you focus on what the object does instead of how it does it **True**
- 3- Abstraction means hiding the internal details or mechanics of how an object does something **True**
- 4- Overriding happens at compile-time
- 5- Overloading happens at runtime
- 6- Static binding is being used for overloaded methods and dynamic binding is being used for overridden/overriding method

7- binding object state(fields) and behavior(methods) together. If you are creating class, you are doing encapsulation.

8- Polymorphism is a object oriented programming feature that allows us to perform a single action in different ways.

Question 3 : Mcq [2 Points]

1- JavaScript is

- 1- synchronous, blocking, single-threaded language.
- 2- asynchronous, non-blocking, single-threaded language.
- 3- synchronous, blocking, multi-threaded language.
- 4- asynchronous, non-blocking, multi-threaded language

2- is the concept of object-oriented programming used to hide the internal representation, or state, of an object from the outside

- 1) Inheritance
- 2) Encapsulation
- 3) Abstraction
- 4) Inheritance

Question 4 : What is The Output [20 Points]

```
var p = new Promise((resolve, reject) => {  
  reject(Error('The Fails!'))  
})  
p.catch(error => console.log(error))  
p.catch(error => console.log(error.message))  
p.catch(error => console.log(error.message))
```

```
function Person(firstName, lastName) {  
  this.firstName = firstName;  
  this.lastName = lastName;  
}  
const member = new Person('Lydia', 'Hallie');  
Person.getFullName = function() {  
  return `${this.firstName} ${this.lastName}`;  
};  
console.log(member.getFullName());
```

Output:
Lydia Hallie

```
const SumBy = num1 => num2 => num1 + num2;  
const sumByTwo = SumBy(2);  
const sumByThree = SumBy(3);  
  
console.log(sumByTwo(4));  
console.log(sumByThree(5));
```

```
class Chameleon {  
  static colorChange(newColor) {  
    this.newColor = newColor;  
    return this.newColor;  
  }  
  
  constructor(newColor) {  
    this.newColor = newColor;  
  }  
}  
  
const freddie = new Chameleon('Purple');  
console.log(freddie.colorChange('orange'));
```

Output:
TypeError: freddie.colorChange is not a function

```
let age = parseFloat(prompt("Enter Your Age"));
let accessAllowed = age >= 18 ? true : false ;
console.log(typeof(accessAllowed));
```

```
function greeting(){
    return "Welcome All";
}
console.log(typeof(greeting()));
```

Output:
boolean
string

```
setTimeout(function(){
    setTimeout(function(){
        console.log(2);
        setTimeout(function(){
            console.log(3);
        } , 0 );
    } , 1000);
    setTimeout(function(){
        console.log(4);
    });
    console.log(1);
} , 2000);
console.log(0);
```

Output:
0
1
4
2
3

```
function counter(){  
  var i = 0 ;  
  return ++i;  
}  
console.log(i);
```

Output:
i is not defined

```
let obj = {  
  msg : "hello world",  
  x : 10  
}  
  
var x = "msg";  
  
console.log(obj[x]);  
console.log(obj["x"]);
```

```
const euros = [29.76, 41.85, 46.5];  
  
const doubled = euros.reduce((total, amount) => {  
  total.push(amount * 2);  
  return total;  
}, []);  
  
console.log(doubled);
```

Output:
[59.52, 83.7, 93]

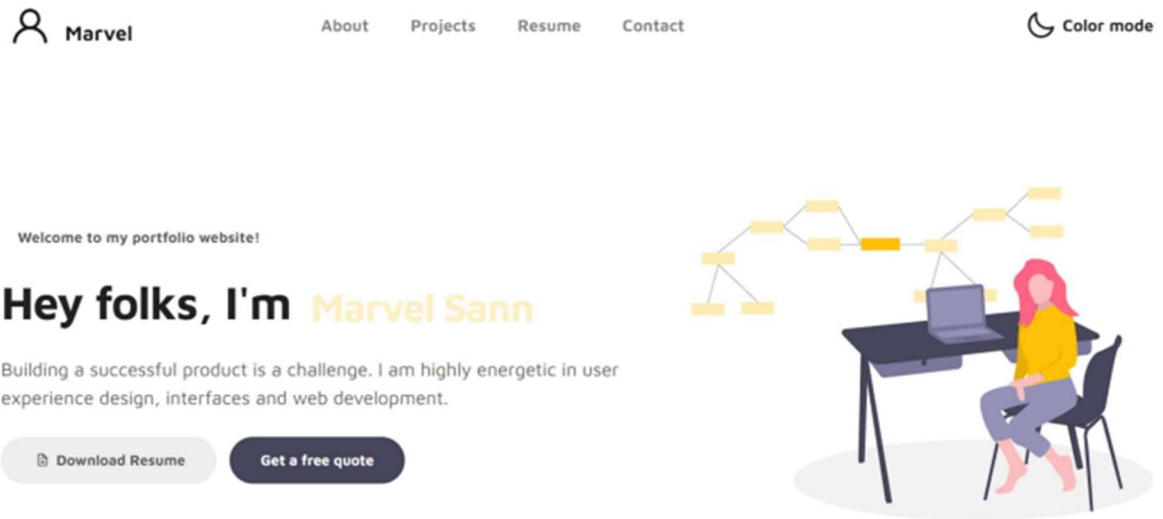
```
const names = ['Batman', 'Catwoman', 'Joker', 'Bane'];
const fromIndex = 1;
const removeCount = 2;
const newNames = [
  ...names.slice(0, fromIndex),
  ...names.slice(fromIndex + removeCount)
];
console.log(newNames);
```

Output: ['Batman', 'Bane']

Question 5 : [100 Points]

- 1- Write a function that returns the length of a string. Make your function recursive. [15 points]
- 2- Write a program that prints a multiplication table for numbers up to 12. Expected Output :
12 * 1 = 12 ===== 12 * 12 = 144 [5 points]
- 3- Write a function that returns the elements on odd positions in a list. [5 points]
- 4- Check If The Number Is Prime Or Not . [5 points]
- 5- Create Background Generator Using (html , css , js) . [8 points]
- 6- Write a short javascript function that counts the number of vowels in a given character string . [12 points]
- 7- Write a program with a mother class animal. Inside it define a name and an age variables, and set_value() function. Then create two base variables Zebra and Dolphin which write a message telling the age, the name and giving some extra information (e.g. place of origin). [5 points]

8- Using Html5 and css3 Make it possible [20 points]

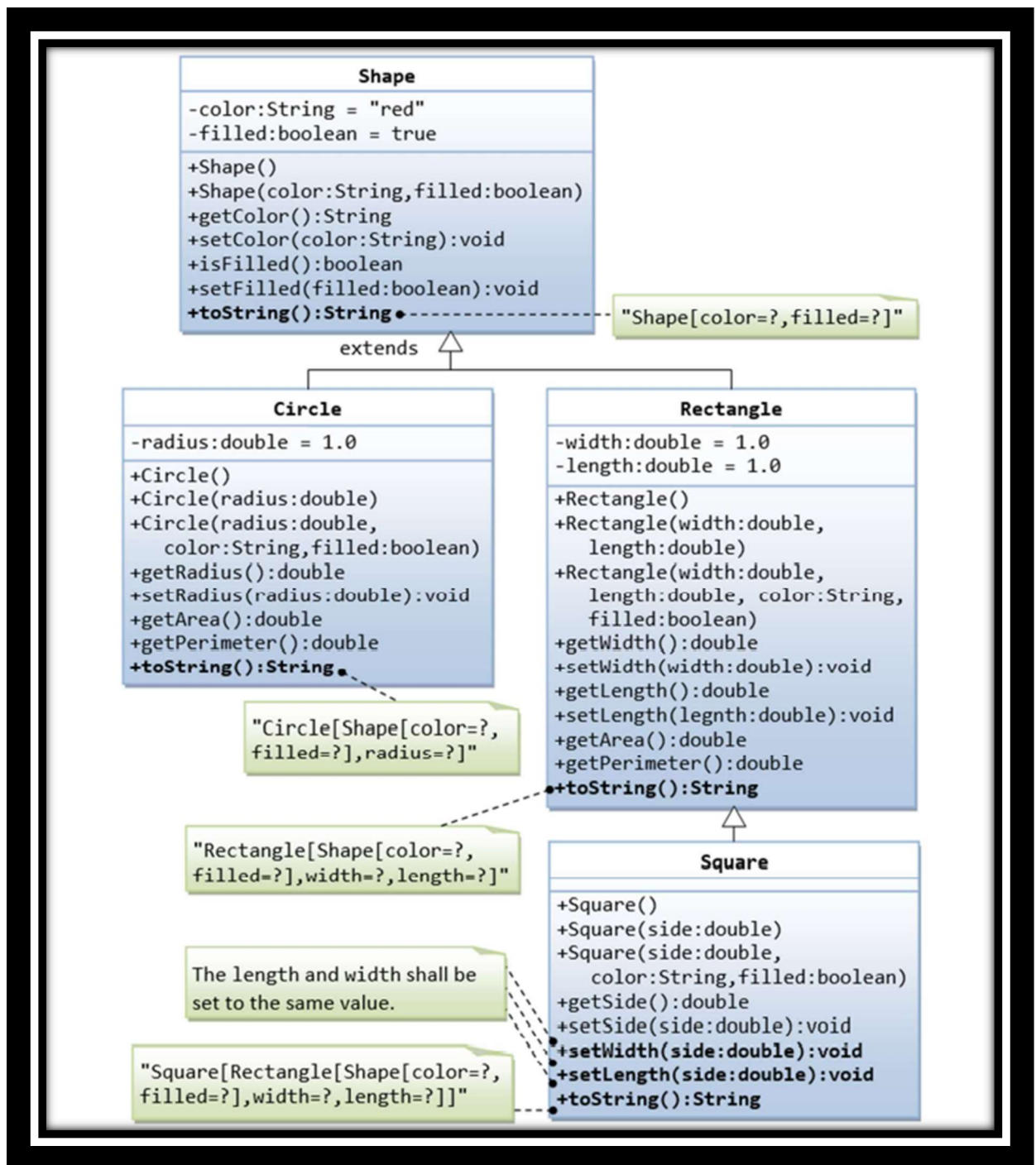


9- Provide one or more alternate implementations that will work as expected. [10 points]

```
// This Code Is Bad Script Performance
function MyObject(name, message) {
  this.name = name.toString();
  this.message = message.toString();
  this.getName = function() {
    return this.name;
  }

  this.getMessage = function() {
    return this.message;
  }
}
```

10- [15 points]



++ Youtube Clone Task [50 points] [html5 and css3 only]

Good Luck 😊

Eng: Hesham Mohamed