Javascript (ES5):

Functions:

1. Write a function square() that returns the square of a number passed to it. Use function declaration syntax to declare the function.
   Example::
   ```
   console.log( square( 3 ) ); // prints 9
   ```

   ```
   function square(x){
       return x*x;
   }
   ```

2. Write a function that accepts another function and calls the accepted function

   ```
   function f(g){
   console.log("I am f ");
   console.log(g);
   g();
   }
   function h(){
   console.log("I am h");
   }

   f(h);
   ```

3. Write a function *sum* that accepts 2 numbers (say *x* and *y*) and another function (say, *transform*) as arguments. The transform function should be a function that accepts a number and returns another number - for example, a function *square* that accepts a number and returns the square of a number. The sum() function applies the transform function on each of each and y and returns the sum of the results of calling transform - for example, sum() would return $x_2 + y_2$ if called as sum(x, y, square);

Example:

```
function square( x ) { return x * x };

function cube( x ) { return x * x *

x };


console.log( sum( 2, 3, square ) ); // prints 13
console.log( sum( 2, 3, cube ) ); // prints 35
function sum(a,b, fun)
{
        return fun(a)+fun(b);
}



function square(x){
    return x*x;
}

function cube(x){
    return x*x*x;
}

console.log(sum(2,3, square));
console.log(sum(1,2,cube));
```

4. Write a function *exponentFactory* that accepts a number, say x. Define 2 functions *square* and *cube* within it (which accept a number each, and return the square and cube respectively). If x is 2, exponentFactory returns the square function, if 3 it returns the cube function. For any other input it returns a function that returns the

number it accepts as such. Call the exponentFactory() function and then the returned function, and log the result.

Example:

```
var fn;


fn = exponentFactory( 2 );
console.log( fn( 5 ) ); // prints 25;


fn = exponentFactory( 3 );
console.log( fn( 5 ) ); // prints 125;


fn = exponentFactory( 4 );
console.log( fn( 5 ) ); // prints 5;



function exponentFactory( x ){
	function square(a){
			return a*a;
	}
	function cube(b){
		return b*b*b;
	}
	if(x==2)
	{
		return square(x);
	}
	else if(x==3)
	{
		return cube(x);
	}
	else
	{
		return x;
	}
}

var fn;
```

```
console.log(exponentFactory( 2
)  ); // prints 25;

fn = exponentFactory( 3 );
console.log( fn( 5 ) ); // prints
125;

fn = exponentFactory( 4 );
console.log( fn( 5 ) ); // prints 5;
```

5. Write a function *sumArray* that works like so.

```
console.log( sumArray( [ 1, 2, 3 ], square ) ); // prints 14

console.log( sumArray( [ 1, 2, 3 ],
cube ) ); // prints 36
```

```
function sumArray(ar, fun){
	let sum=0;
	for(let i=0; i<ar.length; i++)
	{
		sum = sum+ fun(ar[i]);
	}
	return sum;

}
function square(x){
	return x*x;
}function f(g){
	console.log("I am f ");
	console.log(g);
	g();
}
function h(){
	console.log("I am h");
}

f(h);
function cube(y){
```

```
        return y*y*y;
}

console.log( sumArray( [ 1, 2, 3 ], square ) ); // prints 14
console.log( sumArray( [ 1, 2, 3 ], cube ) ); // prints 36
```

FUNCTION CONTEXT

6. Declare a function *foo* and log its context
* Use bind() to create a new function where the context is the object { x: 1 } instead
* Call the bound function

```
function foo( x, y ) {
    console.log( `x = ${x}, y = ${y}` );
    console.log( `this = `, this );
}

foo();
foo({x:1},12,13)



const boundFoo = foo.bind( { x: 1 } );
boundFoo();
boundFoo( 12, 13 )
```

BUILT-IN CLASSES – STRINGS, ARRAYS

7. Given the following array, solve the questions that follow using appropriate array iterator methods (forEach, find, filter, map)

```
var days = [ 'Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday' ];



var days = [ 'Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday' ];

var name1 = days.forEach(function(item){
        console.log(item);
});

var name2 = days.map(function(item){
```

```
        return item.length;
});

console.log(name2);

var name3 = days.filter(function(item){

        var letterNumber = /^[S-Z]+$/;
        if(item[0].match(letterNumber))
        {
                return true;
        }
});

console.log(name3);


var name4 = days.filter(function(item){

                if(item.length == 6)
                {
                        return true;
                }
});

console.log(name4);
```

OBJECTS

8. Create 2 objects (that represents 2 persons, say John and Jane), each with 2 properties - name (a string), and age (a number).

* Print John's age.
* Increase Jane's age and print the Jane object.
* Add an address property to John and set it to an object with "first line" and "city" as properties (the values for these properties also need to be set).
* Print John's city name

* Add a new property spouse to each object. Set John's spouse property to Jane object, and Jane's spouse property to John object
* Add an emailids property to Jane. Set it to an array with 2 strings representing Jane's email ids.
* Print the second email id of Jane.
* Change the second email id of Jane and print it.
* Add a third email id for Jane and print the Jane object.
* Add a method celebrateBirthday() on John that adds 1 to the John's age. Call it on John to increase John's age.
* Add a method celebrateBirthday() on Jane that adds 1 to the Jane's age. Call it on Jane to increase Jane's age.
* Wouldn't it be nice to have a single celebrateBirthday() method shared by both John and Jane objects? Declare celebrateBirthday() as a global function and set it up as a method on both John and Jane objects. Call it to check it increases the age.

```
class Person {
   // no upfront data member definition
   constructor( name, age ) {
      // this; // {} - an newly created empty object
      this.name = name;
      this.age = age;
   }

   celebrateBirthday() {
      this.age++
   }
}

const john = new Person("divya", 12);
const jane = new Person("shubham", 11);
console.log(john);
console.log(jane);

john.celebrateBirthday();
jane.celebrateBirthday();
console.log(john.age);
console.log(jane.age);

john.address = {
        firstline : "jailroad",
        city : "nashik"
}
```

console.log(john.address.city);

Person.spouse = this.spouse;

CALCULATOR IMPLEMENTATION IN JAVASCRIPT

```
<!DOCTYPE html>
<html>
<head>
	<title>calculatorimplementation</title>
	<style>
.title{
border-radius: 12px;
margin-bottom: 14px;
text-align:center;
width: 212px;
color:#ff4456;
border: solid black 1px;
}
input[type="button"]
```

```
{
border-radius: 10px;
background-color:#ff4456;
color: black;
border-color:#ff4456 ;
width:100%;
}
input[type="text"]
{
border-radius: 10px;
text-align: right;
background-color:white;
border-color: black ;
width:100%
}
</style>
</head>
<body>
	<div class="title">Javascript Calculator !</div>
	<table border="2">
		<tr>
		<td><input type="button" onclick="clear()"></td>
		<td colspan="3"><input type="text" name="value" id="ans"></td>
	  </tr>
	  <tr>
	  <td><input type="button" value="+" onclick="dis('+')"/> </td>
		<td><input type="button" value="1" onclick="dis('1')"/> </td>
		<td><input type="button" value="2" onclick="dis('2')"/> </td>
		<td><input type="button" value="3" onclick="dis('3')"/> </td>
	  </tr>
	  <tr>
	  <td><input type="button" value="-" onclick="dis('-')"/> </td>
		<td><input type="button" value="4" onclick="dis('4')"/> </td>
		<td><input type="button" value="5" onclick="dis('5')"/> </td>
		<td><input type="button" value="6" onclick="dis('6')"/> </td>
	  </tr>
	  <tr>
	  <td><input type="button" value="*" onclick="dis('*')"/> </td>
		<td><input type="button" value="7" onclick="dis('7')"/> </td>
		<td><input type="button" value="8" onclick="dis('8')"/> </td>
		<td><input type="button" value="9" onclick="dis('9')"/> </td>
	  </tr>
	  <tr>
	  <td><input type="button" value="/" onclick="dis('/')"/> </td>
		<td><input type="button" value="." onclick="dis('.')"/> </td>
		<td><input type="button" value="0" onclick="dis('0')"/> </td>
		<td><input type="button" value="=" onclick="solve()"/> </td>
	  </tr>
	</table>
```

```
        <script>
//function for displaying values
function dis(val)
{
document.getElementById("ans").value+=val;}
//function for evaluation
function solve()
{
let x = document.getElementById("ans").value
let y = eval(x)
document.getElementById("ans").value = y
}
//function for clearing the display
function clear()
{
document.getElementById("ans").value = "";
}
</script>


</body>
</html>
```