- What is JavaScript?
  JavaScript is a scripting or programming language that allows you to implement complex features on web pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. — you can bet that JavaScript is probably involved. It is the third layer of the layer cake of standard web technologies, two of which (HTML and CSS) we have covered in much more detail in other parts of the Learning Area.

- What are the different data types in JavaScript
  Primitive data types: The predefined data types provided by JavaScript language are known as primitive data types. Primitive data types are also known as in-built data types.
  Boolean type-represents boolean value either false or true
  Null type-represents null i.e. no value at all
  Undefined type-represents undefined value
  Number type-represents numeric values e.g. 100
  String type-represents sequence of characters e.g. "hello"

  Non-primitive data types: The data types that are derived from primitive data types of the JavaScript language are known as non-primitive data types. It is also known as derived data types or reference data types.
  Object-represents instance through which we can access members
  Array-represents group of similar values
  RegExp-represents regular expression

- What are the three different ways of creating an object?
  Java New Operator:
  This is the most popular way to create an object in Java. A new operator is also followed by a call to constructor which initializes the new object. While we create an object it occupies space in the heap.
  Syntax:class_name object_name = new class_name()

  Java Class.newInstance() method:
  Java Class.newInstance() is the method of Class class. The Class class belongs to java.lang package. It creates a new instance of the class represented by this Class object. It returns the newly created instance of the class.
  Syntax:public T newInstance() throws IllegalAcccessException, InstantiationException
  It throws IllegalAccessException if the class or its nullary constructor is not accessible. It also throws InstantiationException, if the Class represents an abstract class, an interface, an array class, or a primitive type.

Java newInstance() method of Constructor class:
Java Constructor class also has a newInstance() method similar to newInstance() method of Class class. The newInstance() method belongs to java.lang.reflect.Constructor class. Both newInstance() method are known as reflective ways to create object. In fact the newInstance() method of Class class internally uses newInstance() method of Constructor class. The method returns a new object created by calling the constructor.
Syntax:public T newInstance(Objects...initargs)

- How to loop through an object?
  for…in loop:
  The most straightforward way to loop through an object's properties is by using the for...in statement. This method works in all modern and old browsers including Internet Explorer 6 and higher.
  Here is an example that uses the for...in loop to iterate over an object:

```
const user = {

    name: 'John Doe',

    email: 'john.doe@example.com',

    age: 25,

    dob: '08/02/1989',

    active: true
};

// iterate over the user object

for (const key in user) {

    console.log(`${key}: ${user[key]}`);
}

// name: John Doe

// email: john.doe@example.com

// age: 25

// dob: 08/02/1989

// active: true
```

Object.keys() Method:
It takes the object that you want to loop over as an argument and returns an array containing all properties names (or keys). After which you can use any of the array looping methods, such as forEach(), to iterate through the array and retrieve the value of each property.
Here is an example:

```
const courses = {
    java: 10,

    javascript: 55,

    nodejs: 5,

    php: 15
};

// convert object to key's array

const keys = Object.keys(courses);

// print all keys

console.log(keys);

// [ 'java', 'javascript', 'nodejs', 'php' ]

// iterate over object

keys.forEach((key, index) => {
    console.log(`${key}: ${courses[key]}`);
});

// java: 10

// javascript: 55

// nodejs: 5

// php: 15
```

Object.values() Method:
The Object.values() method was introduced in ES8 and it works opposite to that of Object.key(). It returns the values of all properties in the object as an array. You can then loop through the values array by using any of the array looping methods.
Let us look at an example:

```
const animals = {
```

```
    tiger: 1,

    cat: 2,

    monkey: 3,

    elephant: 4
};

// iterate over object values

Object.values(animals).forEach(val => console.log(val));

// 1
// 2
// 3
// 4
```

Object.entries() Method:

The Object.entries(), an other ES8 method can be used for traversing an array. . Object.entries() outputs an array of arrays, with each inner array having two elements. The first element being the property and the second element is the value

Here is an example:

```
const animals = {
    tiger: 1,

    cat: 2,

    monkey: 3,

    elephant: 4
};

const entries = Object.entries(animals);
console.log(entries);

// [ [ 'tiger', 1 ],

//   [ 'cat', 2 ],

//   [ 'monkey', 3 ],

//   [ 'elephant', 4 ] ]
```

- Write a program taking a real world instance on Inheritance & Encapsulation. Describe the same

```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
  present() {
    return 'I have a ' + this.carname;
  }
}

class Model extends Car {
  constructor(brand, mod) {
    super(brand);
    this.model = mod;
  }
  show() {
    return this.present() + ', it is a ' + this.model;
  }
}

let myCar = new Model("Ford", "Mustang");
document.getElementById("demo").innerHTML = myCar.show();
```

- Can we define variables with its data type in JS?
  JavaScript is a loosely typed language. It means it does not require a data type to be declared. You can assign any literal values to a variable, e.g., string, integer, float, boolean, etc.

- What is the output?

```
var i=0;

 i++;
Console.log(i); o/p:1

--i;
console.log(i); o/p:-1

i–;
console.log(i); o/p:-1

i=5;
console.log(i) o/p:5

i++;
console.log(i) o/p:6
```

```
var a={};
a[‘abcd’]=1234;
console.log(a); o/p: {abcd:1234}

var a={};
a[‘abcd’];
console.log(a); o/p:{}
```