Que:1: What are microtasks? What is a microtask queue? What is their role in Promises and how are they different from callbacks?

Ans:

**Microtask** is a short function which is executed after the function or program which created it exits and only if the Java execution stack is empty, but before returning control to the event loop being used by the user agent to drive the user agent script's execution environment.

**Microtask Queue** is like the callback queue, but Microtask Queue has**higher priority**. All the callback functions coming through [Promises](https://www.geeksforgeeks.org/javascript-promises/) and [Mutation Observer](https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver) will go inside the Microtask Queue. For example, in the case of [.fetch()](https://www.geeksforgeeks.org/javascript-fetch-method/), the callback function gets to the Microtask Queue. Promise handling always has higher priority so the JavaScript engine executes all the tasks from Microtask Queue and then moves to the Callback Queue.

Callback Queue gets the ordinary callback functions coming from theweb APIs like setTimeout() after the timer expires.

Callback Queue gets the callback functions coming through Promises and Mutation Observer.

Que: 2: Explain with examples how private, protected variables can be implemented in classes and how can they be used in subclasses?

Ans:

PROTECTED ---  
When you declare a method (function) or a property (variable) as Protected, those methods and properties can be accessed by :-  
**The same class that declared it.  
The classes that inherit the above declared class.**

Example:

<?php

class GrandPa

{

protected $name = 'Mark Henry';

}

class Daddy extends GrandPa

{

function displayGrandPaName()

{

return $this->name;

}

}

$daddy = new Daddy;

echo $daddy->displayGrandPaName(); // Prints 'Mark Henry'

$outsiderWantstoKnowGrandpasName = new GrandPa;

echo $outsiderWantstoKnowGrandpasName->name; // Results in a Fatal Error

for subclass:-

class Person {

protected String fname = "John";

protected String email = "john@doe.com";

}

class Student extends Person {

private int graduationYear = 2018;

public static void main(String[] args) {

Student myObj = new Student();

System.out.println("Name: " + myObj.fname + " " + myObj.lname);

System.out.println("Email: " + myObj.email);

System.out.println("Graduation Year: " + myObj.graduationYear);

}

}

Prints:

Name: John Doe

Email: john@doe.com

Graduation Year: 2018

PRIVATE ---

When you declare a method (function) or a property (variable) as private, those methods and properties can be accessed by :-  
**The same class that declared it.**

Outsider members cannot access those variables. Outsiders in the sense that they are not **object instances of the declared class itself** and even the classes that **inherit the declared class.**

Example:

class GrandPa

{

private $name = 'Mark Henry';

}

class Daddy extends GrandPa

{

function displayGrandPaName()

{

return $this->name;

}

}

$daddy = new Daddy;

echo $daddy->displayGrandPaName(); // Results in a Notice

$outsiderWantstoKnowGrandpasName = new GrandPa;

echo $outsiderWantstoKnowGrandpasName->name; // Results in a Fatal Error