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

Answer:

* As the name suggests, **microtask** is a small task/function which is created by some other function and gets executed after some event happens. So, the question is what are such events?
* As we know, the JS engine uses a call stack to keep track of multiple function calls. **event loop** is an endless loop which reads this stack and executes I/O operations in the stack.
* For ex, let's consider below code:
* If we run the above code, “code finished” will get executed first and then “promise done!” gets logged. Because-
  + Promise handlers **.then or .catch or .finally** are always asynchronous.
  + Microtask(here task is to log “promise done!”) gets created when asynchronous code gets executed.
  + Here, when a promise is ready, its **.then** handler is put into the queue; Items in the queue will not be picked/executed by event loop until the JS engine becomes free from the current code i.e. existing function code call stack gets emptied. Then it takes a task (i.e. **.then** handler) from the queue, puts it in the call stack and executes it.
  + The queue used in this scenario is called **microtask queue** also known as **PromiseJobs.**
  + This queue follows FIFO.
* How is it different from callbacks?
  + Callbacks are also small functions that are used by async functions and get executed after desired time/event.
  + These callbacks also use a queue (similar to microtask) called **Callback queue**.
  + The difference between microtask and callback are-
    - Callback queue gets the **ordinary callback functions** coming from apis like setTimeout/ setInterval after the timer expires. While, Microtask Queue gets the callback functions coming through **Promises**.
    - Another difference is the priority of queues. Let's say Promises and setTimeout are used in the same function then the microtask queue will take the higher priority than the callback queue of fetching the callback functions to Event Loop. Ex-

Once the call stack is empty, the event loop will log “code finished” then “promise done!”

1. Explain with examples how private, protected variables can be implemented in classes and how can they be used in subclasses?

Answer:

* Private: These members are only accessible within the class that instantiated the object.
* Protected: This keyword allows a little more access than private members but a lot less than the public. A protected member is accessible within the class (similar to private) and any object that inherits from it. It is not accessible by anybody else.
* Let's take an example:
* Privates should start with #. So, **#accountType** is a private variable.
* Protected properties are usually prefixed with an underscore **\_.** So, **\_balance** is a protected variable.
* Let’s assume the **Bank** allows only accounts of type **‘Saving’.** So, any account that inherits **Bank,** should not be able to change **accountType.**
* But **\_balance** should be changed on different bank activities.
* So above example shows how **#accountType** acts as a private variable and can not change from the external interface (gives SyntaxError if tried). And we are providing protected access to **\_balance** from inherited class (here **Account1**) and we can implement setters and getters on protected variables.