console.log("Start");

setTimeout(() => {

console.log("Timeout");

}, 0);

Promise.resolve().then(() => {

console.log("Promise");

});

console.log("End");

Output:

Code

**Start  
End  
Promise  
Timeout**

**Explanation:**

* **Event Loop:**

**JavaScript's event loop is a mechanism that manages the execution of code. It constantly checks if there are any tasks in the queue and executes them.**

* **Macrotasks and Microtasks:**

**The event loop has two main queues:**

* + **Macrotasks: Tasks like setTimeout and setInterval are considered macrotasks.**
  + **Microtasks: Tasks like promises and process.nextTick (in Node.js) are considered microtasks.**
* **Execution Order:**

**The event loop follows this order:**

* + **Execute the current synchronous code.**
  + **Execute all microtasks in the microtask queue.**
  + **Execute the next macrotask in the macrotask queue.**

**In the example above:**

* **"Start" is printed.**
* **setTimeout is scheduled, but its callback is added to the macrotask queue.**
* **The promise is resolved, and its then callback is added to the microtask queue.**
* **"End" is printed.**
* **The event loop checks the microtask queue, finds the promise callback, and executes it, printing "Promise".**
* **Finally, the event loop checks the macrotask queue and executes the setTimeout callback, printing "Timeout".**