1.  
**Microtasks** are usually scheduled for things that should happen straight after the currently executing script, such as reacting to a batch of actions, or to make something async without taking the penalty of a whole new task.

The microtask queue is processed after callbacks as long as no other JavaScript is mid-execution, and at the end of each task.

Once a promise settles, or if it has already settled, it queues a microtask for its reactionary callbacks. This ensures promise callbacks are async even if the promise has already settled.

2.

class Company{

protected name: string;

constructor(name: string) { this.name = name; }

}

class Apple extends Company{

private noOfModels: number;

constructor(name: string, noOfModels: number) {

super(name);

this. noOfModels = noOfModels;

}

public getIntro() {

return `Hello, This is ${this.name} phone and it has ${this. noOfModels } models.`;

}

}

let myPhone= new Apple ("COOL Car", 4);

console.log(myPhone.getIntro()); //Hello, This is Apple phone and it has 4 models.

console.log(myPhone.name); // Error!! , Property 'name' is protected and only accessible within class Company and its subclasses.