1. Micro Task: Micro tasks are the tasks which are executed after a normal code block completes, they are asynchronous in nature.  
   Tasks like Promises, Observables come under this category  
     
   Macro Task: Macro tasks are the tasks which are executed after all the queued up tasks are completed at the browser level, they are asynchronous in nature.  
   Tasks like settimeout, setinterval fall under this category.  
     
   Order of code execution:  
   Normal task -> Micro-task -> Macro-task  
     
   Eg:  
   setTimeout(() => {console.log('macro task');}, 0); // macro task

let promise = new Promise((resolve, reject) => { //micro task

resolve('micro task');

})

promise.then((res) => console.log(res));

console.log('main task'); // normal task

2. Private variable: Private methods and variables are one which are only accessible within the class, in order to use them in subclasses we can make use of getters/ public methods that return the private variable or execute the private function within themselves.

class Animal {

#type;

constructor(name) {

this.#type = 'tiger';

}

returnAnimalType() {

return this.#type;

}

#returnAMessage() {

return "You will do great things in life";

}

}

const myObject = new Animal("Adi");

// console.log(myObject.returnAnimalType()); // 42

// console.log(myObject["#type"]); // undefined

// console.log(myObject.#type); // SyntaxError

// console.log(myObject.#returnAMessage);

class abc extends Animal {

constructor() {

super();

}

}

const obj = new abc();

console.log('444', obj.returnAnimalType());

Protected: The concept of protected properties is a bit hard to replicate on the Javascript end, but to replicate the same I think the below example can be fine where we do have getter but we don’t have a setter, once we try to set the value manually we get error

class NameGenerator {

\_name;

constructor(name) {

this.\_name = name;

}

get name() {

return this.\_name;

}

}

let nameGenerator = new NameGenerator("Adi");

console.log(`My name is ${nameGenerator.name}`); // My name is John

nameGenerator.name = "Anmol"; // Cannot assign to 'name' because ii is private