SetTimeout: I guarantee that the function passed to me as a parameter is going to be executed **after** the timer duration that has been given as a second parameter to me.

Javascript is single threaded

```
console.log("Start");

setTimeout(function() {
  console.log("Hello");
}, 2000);

console.log("End");
```

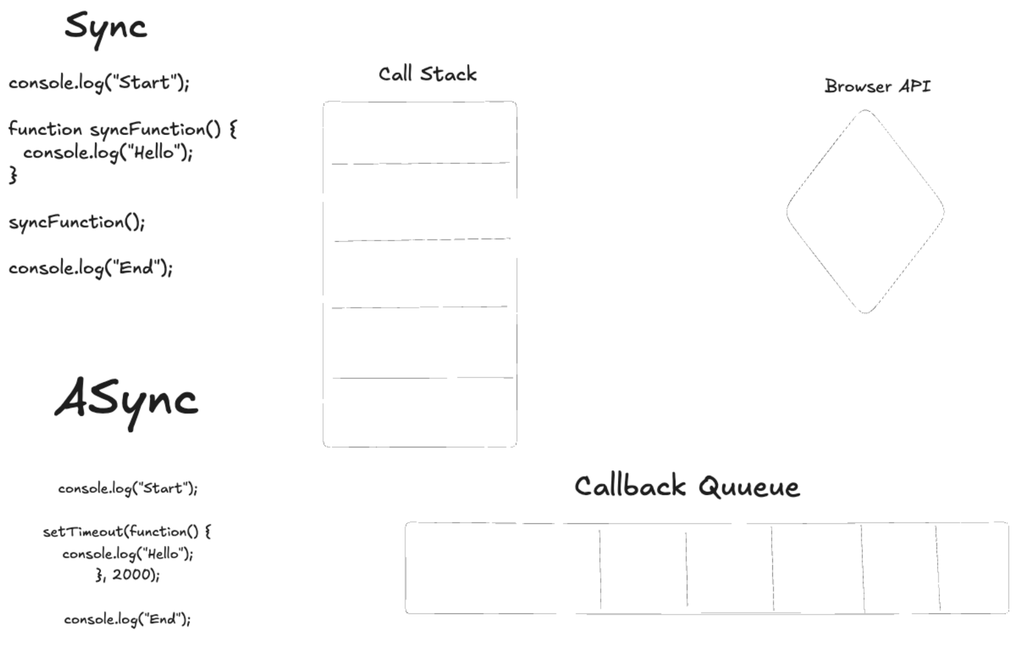
Event Loop:

Call stack
Callback Queue
Browser api’s

```
console.log("Start");

setTimeout(function() {
  console.log("Hello");
}, 2000);


 console.log("1");
```



```
console.log("Start");

function syncFunction() {
  console.log("Hello");
}

syncFunction();

console.log("End");
```

```
console.log("Start");

setTimeout(function() {
  console.log("Hello");
}, 2000);

console.log("End");
```

```
setTimeout(function(){
     console.log("world");
}, 0);

console.log("End");
```

<u>Promises:</u> In Javascript promises represent an object that is going to be run at event completions.
This is in order to support asynchronous behaviour of JS.

Promise: A promise can either be resolved or rejected.

```
let promise = new Promise(function(resolve, reject) {
	//executor function
	resolve();
	reject();
});
```

Promise states: Pending, Resolved(fulfilled), rejected

Pending

Resolved(fullfiled)   rejected.

Executor function would be immediately called.

coinTossPromise

There would be coint which would either result in heads or tails
If it is heads you consider it as win - resolve
Else you consider it as loss. - reject

```
let coinTossPromise = new Promise(function(resolve, reject) {
        setTimeout(function() {
        const isHeads = Math.random() > 0.5;

        if(isHeads) {
                resolve(‘win’);
        } else {
                reject(‘loss’);
        }
}, 1000)
})
```

Interview question

Question

```
var obj = {
	first: function() {
		console.log(“first”);
	},
	second: function() {
		console.log(“second”);
	},
	third: function() {
		console.log(“third”);
	}
}
```

Answer

```
var obj = {
	first: function() {
		console.log(“first”);
		return obj;
	},
	second: function() {
		console.log(“second”);
		return obj;
	},
	third: function() {
		console.log(“third”);
		return obj;
	}
}


obj.first().second().third();
first
second
third

// obj.second().third();
first
// obj.third()
 second
third


coinTossPromise.then(function(result) {
    console.log(result);
}).catch(function(error) {
    console.log(error);
})
```

Promise Chaining

Promises need to be chained.

1) Cleaning the room.
2) Removing the garbage.
3) Winning the icecream.

```
let cleanRoom = function() {
        return new Promise(function(resolve, reject) {
                resolve(‘Cleaned the room’);
        })
}


let removeGarbage = function(message) {
                return new Promise(function(resolve, reject) {
                resolve(message + ‘removed the garbage’);
        })
}

let winIcecream = function(message) {
          return new Promise(function(resolve, reject) {
                resolve(message + ‘got the icecream’);
        })
}

cleanRoom().then(function(result) {
        console.log(result);
        return removeGarbage(result);
}).then(function(result) {
        console.log(result);
        return winIcecream(result);
}).then(function(result){
        console.log(result);
})
```

Error scenario

```
let cleanRoom = function() {
  return new Promise(function(resolve, reject) {
    // 50% chance of success
    if (Math.random() < 0.5) {
      resolve('Cleaned The Room');
    } else {
        // 50% chance of failure
      reject('Failed to clean the room');
    }
  });
};

let removeGarbage = function(message) {
  return new Promise(function(resolve, reject) {
    // 50% chance of success
    if (Math.random() < 0.5) {
      resolve(message + ' then removed Garbage');
    } else {
        // 50% chance of failure
      reject('Failed to remove garbage');
    }
  });
};

let winIcecream = function(message) {
    return new Promise(function(resolve, reject) {
    resolve(message + ' then won Icecream');
  });
};



cleanRoom().then(function(result) {
        console.log(result);
        return removeGarbage(result);
}).then(function(result) {
        console.log(result);
        return winIcecream(result);
}).then(function(result){
        console.log(result);
}).catch(function(error) {
        console.log(error);
})
```