**NODEJS 04: BẤT ĐỒNG BỘ (ASYNCHRONOUS) TRONG JAVASCRIPT**

A – LÝ THUYẾT

I – XỬ LÝ ĐỒNG BỘ & XỬ LÝ BẤT ĐỒNG BỘ

**1. Phân biệt xử lý đồng bộ & bất đồng bộ**

*Xử lý đồng bộ là chương trình sẽ xử lý từ trên xuống dưới, câu lệnh nào trước sẽ thực thi trước, đến khi nào xong thì mới thực thi câu lệnh kế tiếp (dẫn tới tình trạng các câu lệnh sẽ phải chờ nhau để thực thi)*

*Xử lý bất đồng bộ là chương trình sẽ xử lý từ trên xuống dưới, tuy nhiên có thể nhẩy qua một số câu lệnh trước đó để ưu tiên thực thi các câu lệnh sau, có nghĩa là không nhất thiết lệnh nào vào trước sẽ thực thị trước.*

**2. Ví dụ minh họa cho xử lý đồng bộ & bất đồng bộ**

**a. Đồng bộ**

* Xử lý:

|  |
| --- |
| console.log('This is the first line');  console.log('This is the second line');  console.log('This is the last line'); |

* Kết quả:

|  |
| --- |
| This is the first line  This is the second line  This is the last line |

**b. Bất đồng bộ**

* Xử lý:

|  |
| --- |
| console.log('This is the first line');  setTimeout(function(){  console.log('This is the second line');  }, 1000);  console.log('This is the last line'); |

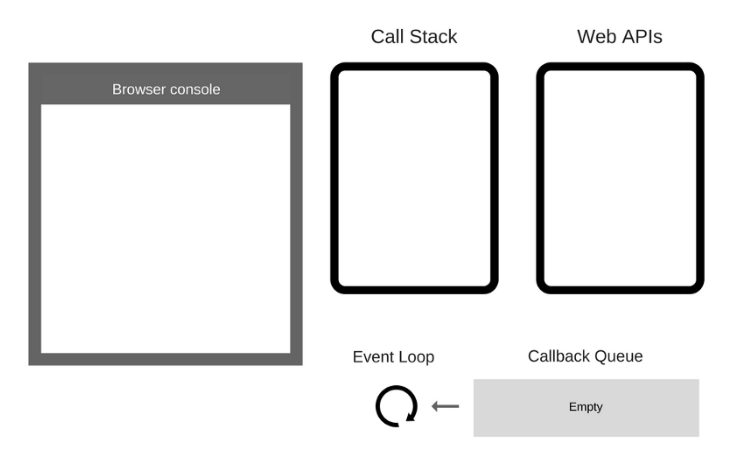
* Kết quả:

|  |
| --- |
| This is the first line  This is the last line  This is the second line |

**3. Giải thích về cơ chế bất đồng bộ**

Để giải thích cho cơ chế bất đồng bộ ở ví dụ trên chúng ta có các bước mô phỏng toàn bộ quá trình diễn ra như sau:

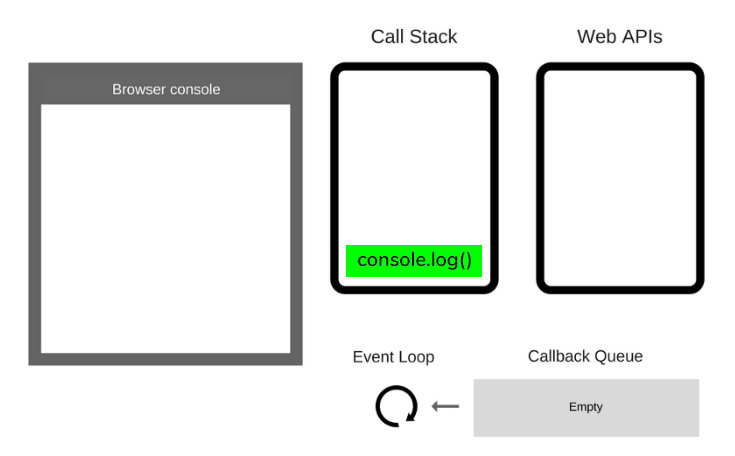
Cùng xem qua về các thành phần tham gia vào quá trình chạy code ở ví dụ trên



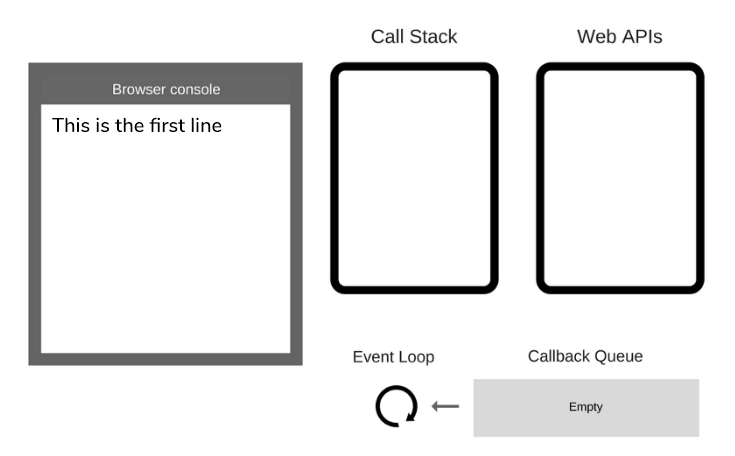
Trong đó:

* **Browser console:** Nơi hiển thị kết quả
* **Call Stack :** Một kiểu của hàng đợi, mỗi câu lên được thực hiện sẽ được xếp vào Call Stacktheo thứ tự thực thi trước vào trước (nằm trên cùng), và thực thi xong sẽ được chuyển ra khởi hàng đợi.
* **Web API :** Chính là các Function không truy cập trực tiếp mà chỉ có thể gọi đến nó (Callback Funtion)
* **Callback Queue :** Nơi tiếp nhận các Callback Funtion được đẩy xuống và hoạt động theo cơ chế, vào trước thực thi trước, vào sau thực thi sau.
* **Event Loop :** Giám sát Callback Queue và Call Stack

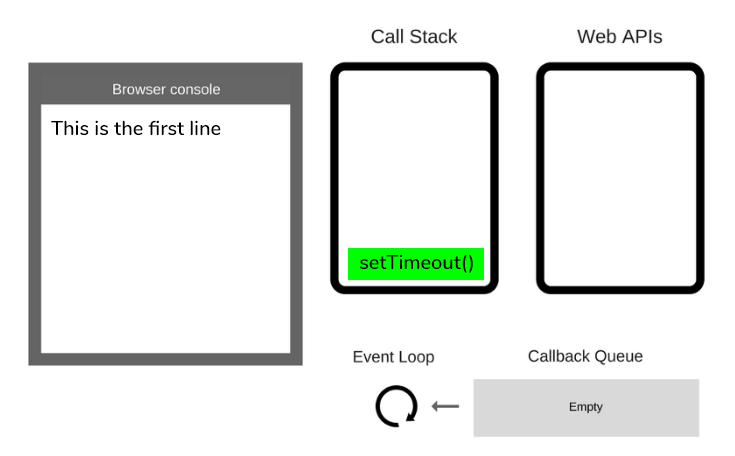
Khi chương trình bắt đầu thì câu lệnh đầu tiên chạy trước nên sẽ được đưa vào Call Stack để thực thi như hình dưới



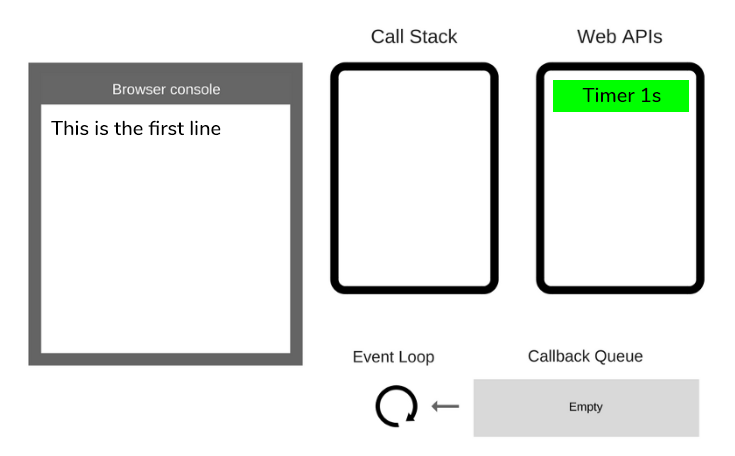
Câu lệnh này trả về kết quả This is the first line, đồng nghĩa nó đã thực thi xong và được đẩy ra khỏi Call Stack như hình dưới



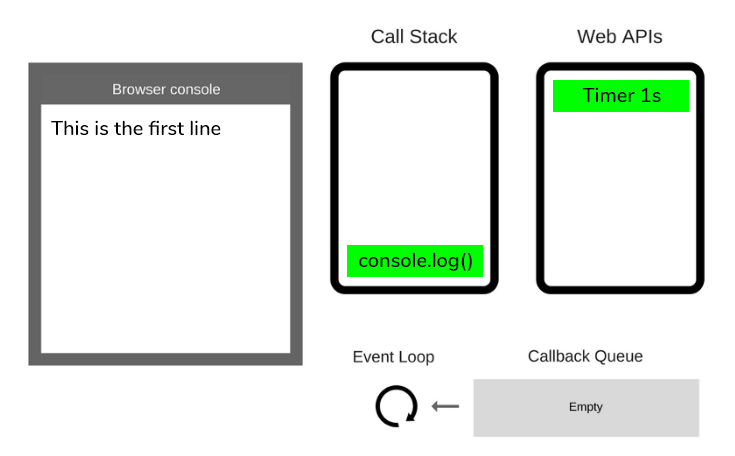
Và ngay lập tức câu lệnh tiếp theo được thực thi, và tương tự no được chuyển vào Call Stack như hình dưới



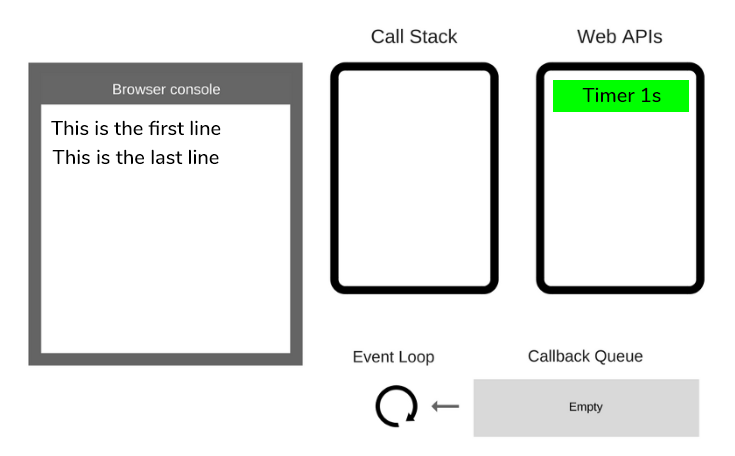
Do lệnh này chứa Callback Funtion nên thay vì thực thi ngay thì nó được chuyển vào Web APIs để đợi thực thi (và như ví dụ trên) nó phải đợi mất 1s mới được thực thi, thế nên được kết quả như hình dưới



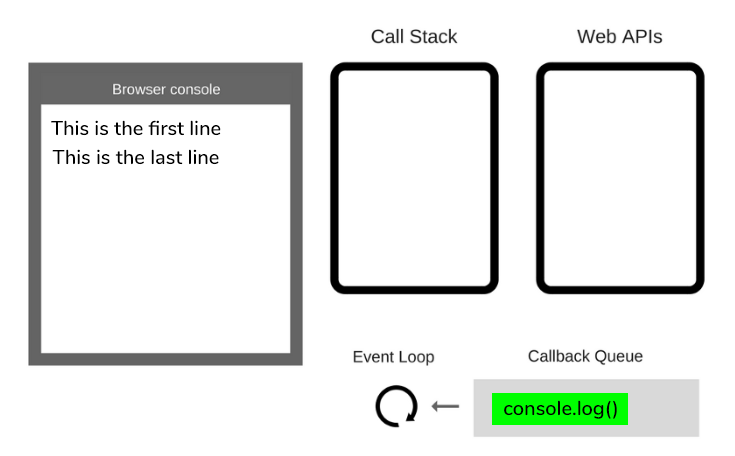
Khi đoạn code thứ 2 bị chuyển vào WEB APIs thì ngay lập tức đoạn code thứ 2 được chuyển vào Call Stack để thực thi như hình dưới



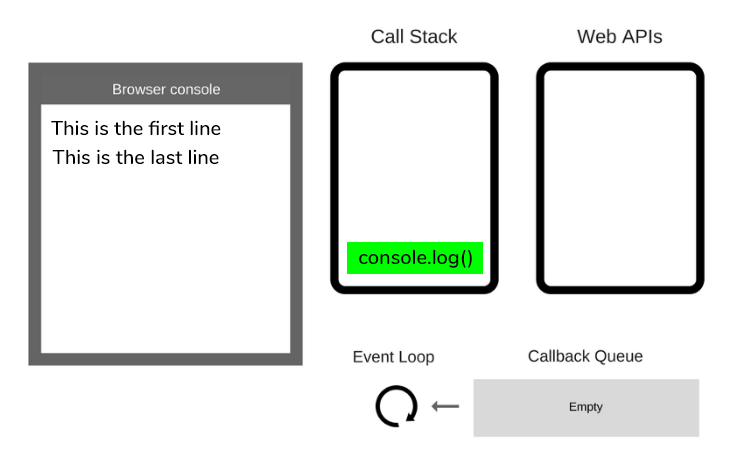
Và đoạn code này cũng nhanh chóng cho ra kết quả This is the last line và được chuyển ra khỏi Call Stack như hình dưới đây



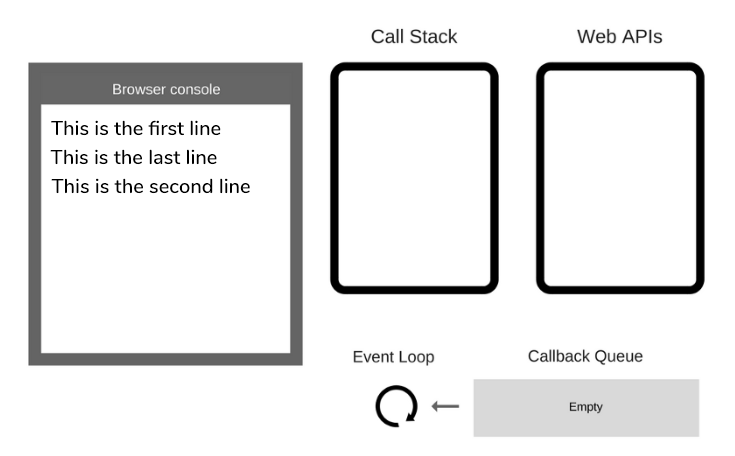
Lúc này, sau khi khoảng thời gian 1s đã trôi qua, cũng chính là thời điểm mà câu lệnh thức 2 đã thực thi xong, nó đẩy kết quả xuống Callback Queue như hình dưới



Tại đây Event Loop liên tục kiểm tra xem có gì trong Callback Queue không (điều kiện cần) và Call Stack có trống không (điều kiện đủ). Nếu thỏa mãn cả 2 điều kiện cần và đủ nó sẽ chuyển toàn bộ kết quả từ Callback Queue sang Call Stack như hình dưới



Và tại Call Stack, đoạn code được thực thi và in kết quả ra màn hình



II – XỬ LÝ BẤT ĐỒNG BỘ VỚI PROMISE

**1. Promise**

Promise trong Javascript ES6 được sử dụng để quản lý các kết quả của một hành động bất đồng bộ (Async)

**2. Khởi tạo Promise**

|  |
| --- |
| var promise = new Promise(function(resolve, reject){    }); |

Trong đó

* resolve: Là một hàm callback xử lý cho hành động thành công
* reject: Là một hàm callback xử lý cho hành động thất bại

**3. Promise trả về resolve**

* Khởi tạo Promise

|  |
| --- |
| var promise = new Promise(function(resolve, reject){    resolve('Success');  }); |

* Lấy giá trị trả về từ Promise

|  |
| --- |
| promise.then(function(resolve){  console.log(resolve);  }); |

**4. Promise trả về reject**

* Khởi tạo Promise

|  |
| --- |
| var promise = new Promise(function(resolve, reject){    reject('Error');  }); |

* Lấy giá trị trả về từ Promise

|  |
| --- |
| promise.then(  function(resolve){  console.log(resolve);  },  function(reject){  console.log(reject);  }  ); |

B – TRAINING DỰ ÁN

I – CÔNG VIỆC TRÊN LỚP

**1. Hoàn thiện chức năng Login với TK tồn tại trong CSDL**

**2. Hoàn thiện chức năng liệt danh sách sản phẩm**

II – CÔNG VIỆC VỀ NHÀ

**1. Hoàn thiện chức năng liệt kê danh sách thành viên**

* Làm tương tự như phần sản phẩm trên lớp

**2. Hoàn thiện chức năng liệt kê danh sách danh mục sản phẩm**

* Làm tương tự như phần sản phẩm trên lớp