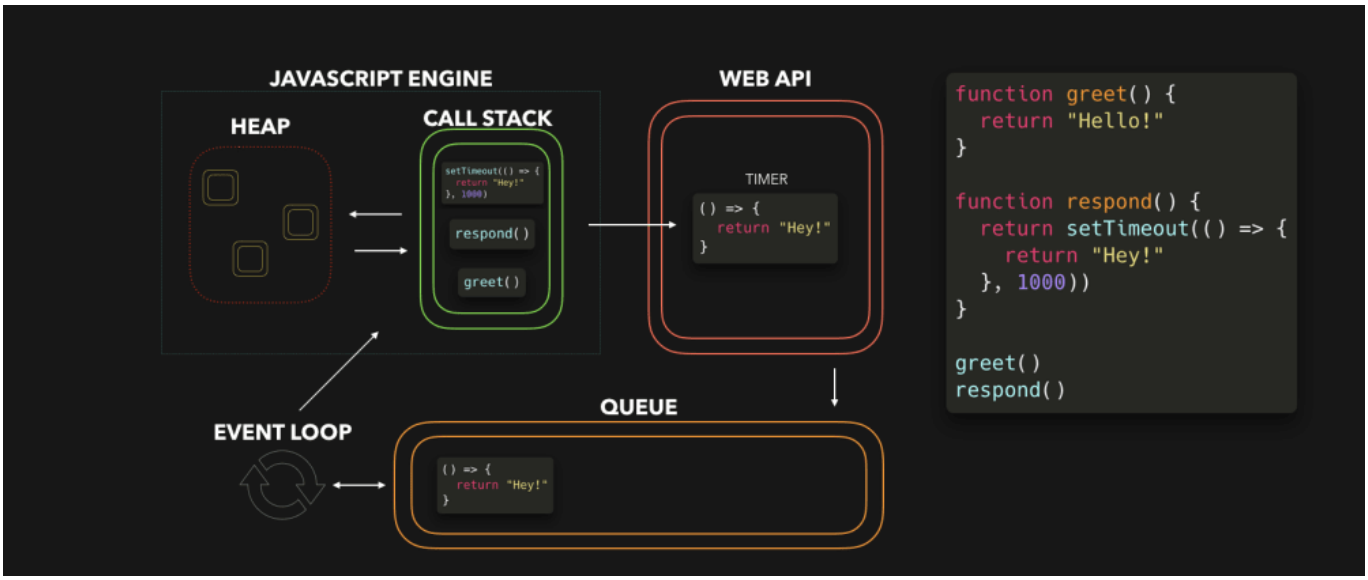
setTimeout(callback, timeout)



> Source: https://dev.to/lydiahallie/javascript-visualized-event-loop-3dif

1. setTimeout overview
2. clearTimeout
3. Late timeout

> Web api: https://developer.mozilla.org/en-US/docs/Web/API

Web API là gì?

Trong ngữ cảnh của trình duyệt web, "Web API" thường đề cập đến các giao diện lập trình ứng dụng (API) được hỗ trợ trực tiếp trong trình duyệt để cho phép các ứng dụng web tương tác với các tính năng của trình duyệt hoặc các tài nguyên mạng.

Các Web API trong trình duyệt cung cấp các phương thức và thuộc tính mà các trang web có thể sử dụng để thực hiện các nhiệm vụ như:

1. Truy cập vào thông tin trình duyệt và hệ thống: Bao gồm thông tin như loại trình duyệt, kích thước cửa sổ, vị trí địa lý, v.v.
2. Truy cập vào các thiết bị phần cứng: Đây bao gồm truy cập vào camera, microphone, và cảm biến khác của thiết bị.
3. Truy cập vào các tính năng của trình duyệt như đọc và ghi cookie, hoạt động về lịch sử duyệt web, v.v.
4. Truy cập vào các API mạng: Bao gồm các giao thức như HTTP, WebSocket, và các phương thức để thực hiện yêu cầu HTTP.
5. Truy cập vào các tính năng đồ hoạ và đa phương tiện: Bao gồm WebGL, Web Audio API, và các API khác để xử lý đồ hoạ và âm thanh trong trình duyệt.

Tóm lại, các Web API trong trình duyệt cung cấp cho các nhà phát triển web các công cụ và chức năng mạnh mẽ để tạo ra các ứng dụng web phức tạp và tương tác.

1. setTimeout overview

1. What: là một phương thức chung để đặt bộ hẹn giờ, sau đó thực hiện một chức năng khi bộ hẹn giờ hết hạn.
2. When: khi muốn chạy một chức năng sau một khoảng thời gian.
3. Where: browser + nodejs

```
const timeoutId = setTimeout(callback, timeout);
// callback is the function will be executed once the timer is expired
// timeout is the time in milliseconds, default to 0. 1 second = 1000 milliseconds
// timeoutId is a positive integer, used to identify
```

> Lưu ý truyền **hàm** vào setTimeout chứ không phải là gọi **hàm**

```
function sayHello() {
 console.log('Hi');
}
// say hello after 1 second
setTimeout(sayHello, 1000);
```

```
// say hello after 1 second
setTimeout(() => {
 console.log('Hi');
}, 1000);
```

```
// say hello as soon as you can (not immediately)
setTimeout(() => {
    console.log("Hi 1");
}, 0);

setTimeout(() => {
    console.log("Hi 2");
});

console.log("log 1");

setTimeout(() => {
    console.log("log 2");
});

console.log("log 3");
```

```
// log 1
// log 3
// Hi 1
// Hi 2
// log 2
```

```
console.log('Hello');
setTimeout(() => {
    console.log('Yup!');
}, 1000);
console.log('World');
```

```
// Real case: Redirect to another page after 3 seconds
setTimeout(() => {
 window.location.href = 'https://google.com';
}, 3000);
```

> Event Loop: http://latentflip.com/loupe/

2. clearTimeout

```
const timeoutId = setTimeout(() => {
 console.log('Tada!!!')
}, 5000);
clearTimeout(timeoutId); // cancel the timeout
// should be called before the timeout expired
```

3. Late timeout

Thời gian chờ cũng có thể kích hoạt muộn hơn dự kiến nếu trang (hoặc hệ điều hành/trình duyệt) đang bận thực hiện các tác vụ khác.

```
setTimeout(() => {
 console.log('Tada!!!')
});
let count = 1;
for (let i = 0; i < 1e9; i++) {
 count++;
}
console.log('done');
```

Tham khảo

- https://developer.mozilla.org/en-US/docs/Web/API
- https://developer.mozilla.org/en-US/docs/Web/API/setTimeout