

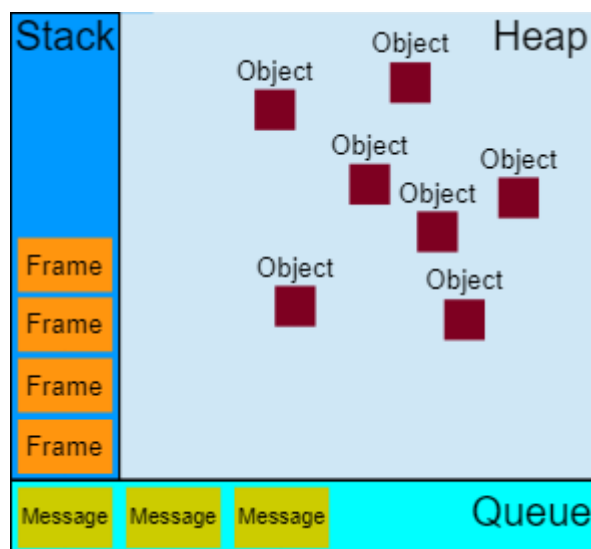


2023.05.21 Event Loop

Event Loop

자바스크립트는 **싱글 스레드** 기반의 언어입니다. 즉, **한 번에 하나의 작업만**을 진행할 수 있다는 의미입니다. 그런데 우리는 웹 브라우저를 이용할 때, 애니메이션 효과를 보면서 동시에 사용자 입력을 받아 처리할 수 있습니다. 어떻게 이런 **동시성**이 구현되는 것일까요? 바로 **이벤트 루프**를 이용한 비동기 처리 방식 때문입니다.

Javascript의 동작 과정



1. Javascript 엔진은 Stack과 Heap으로 구성되어 있습니다.
 - a. **Object**는 **Heap**에 할당됩니다. Heap은 단순히 메모리의 구조화되지 않은 큰 영역을 지칭합니다.
 - b. **호출된 함수들은 Stack에** 쌓입니다. 각 항목을 **Frame**이라고 합니다.
2. 브라우저 엔진에는 **Message Queue(또는 Event Queue)**가 존재하며, 여기에는 **비동기로 처리되는 Callback들이 저장**됩니다.
3. 코드가 실행되면 호출된 함수들이 Call Stack에 쌓이고, 가장 최근에 호출된 함수부터 (LIFO) 실행됩니다.
 - a. 이 때, 동기 함수들은 바로 실행됩니다.

- b. 비동기 함수들은 **Web API**(DOM, Ajax, Timeout 등)를 호출합니다.
- c. Web API는 비동기 함수의 Callback을 Message Queue에 넣습니다.
- 4. Call Stack의 함수 실행이 모두 완료되어 Call Stack이 빈 상태가 되면, **Event Loop가 Message Queue의 가장 오래된 Message의 Callback부터 Call Stack에** 쌓습니다. 이러한 Event Loop의 반복 과정을 **Tick**이라고 합니다.
- 5. Event Loop는 이러한 방식으로 Stack이 텅 빌 때까지 함수를 처리하고, 함수의 처리가 완료되면 Queue에서 다음 Message를 처리합니다. 대략적으로 아래와 같은 모양입니다.

```
while(queue.waitForMessage()) {  
  queue.processNextMessage();  
}
```

- a. processNextMessage는 현재 처리할 수 있는 Message가 존재하지 않으면 새로운 Message가 도착할 때까지 동기적으로 대기합니다.

예시

```
console.log('hello, visualforce');  
  
setTimeout(() => {  
  console.log('hello, aura');  
}, 2000);  
  
function sayHello() {  
  console.log('hello, lwc');  
}  
sayHello();
```

1. 위 코드가 실행되면, 가장 먼저 `console.log('hello, visualforce');`가 Call Stack에 쌓입니다.
2. 이는 동기적으로 동작하므로 console에 'hello, visualforce'를 출력하고 pop됩니다.
3. 다음으로, `setTimeout`이 Call Stack에 쌓입니다.
4. 이는 비동기적으로 동작하므로, Web API를 호출하고, Background에서 Timer 함수를 실행시킵니다.
5. 다음으로, Timer 함수의 종료를 기다리지 않고, `sayHello` 함수가 Call Stack에 쌓입니다.

6. `sayHello` 함수의 내부에서 실행되는 `console.log('hello, lwc');` 함수가 다시 Call Stack에 쌓입니다.
7. LIFO 순서를 따라 가장 최근에 실행된 함수인 `console.log('hello, lwc');` 가 실행되어 console에 'hello, lwc'가 출력된 후 pop 되고, `sayHello` 함수도 pop 됩니다.
8. 5~7의 과정이 동기적으로 처리되는 동안, Timer 함수의 처리가 끝나면 Web API는 Callback인 `console.log('hello, aura');` 를 Message Queue에 추가합니다.
9. 7까지의 과정이 종료되어 Call Stack이 빈 상태가 되면 , Event Loop에 의해 Message Queue에 있는 `console.log('hello, aura');` 이 Call Stack에 쌓이고, 'hello, aura'를 출력한 뒤 pop되며 실행이 종료됩니다.



interview 이정환

1. 이벤트 루프 짧게 설명해 주세요
 - a. 자바스크립트 동작 과정... ⇒ 컷
 - b. 비동기 작업을 도와주는 역할 ⇒ 어떻게 도와주는가?
 - i. 콜 스택이 비면 기준에 맞게 콜 스택에 넘겨 주는 역할 ⇒ 어디에 있던 것을 넘겨주나?
 - ii. 웹 API에서 비동기 처리, 셋 타임아웃 등의 작업, AJAX 등의 동작을 가져와서 콜스택
 - iii. 콜스택으로 보내겠죠? ⇒ 흠...
 - iv. 큐는 어디서 나온거죠? ⇒ 웹 API에서 발생하는 동작들이 콜백 큐에 추가된 뒤 이 상태를 보고 콜스택으로 넘어가는 구조이다.
 - v. 동작들이란? ⇒ 비동기, 셋타임아웃, AJAX 등입니다.
 - vi. 콜스택이 비지 않은 상태에서 셋타임아웃이 끝나면? ⇒ 타임이 지나기 전까지 다음 동작이 콜스택에 추가되어 실행되고 그 다음에 셋타임아웃 동작이 실행된다... ⇒ 콜스택이 비지 않은 경우에
2. 비동기로 동작하는 것은 JS때문인가요 브라우저 때문인가요? ⇒ 브라우저의 웹 API 덕분입니다. JS는 싱글 스레드이기 때문입니다.