* 예습
1. 6/5

41. 타이머

TABLE OF CONTENTS

#1. 호출 스케줄링

함수를 명시적으로 호출하면 함수가 즉시 실행된다. 만약 함수를 명시적으로 호출하지 않고 일정 시간이 경과된 이후에 호출되도록 함수 호출을 예약하려면 타이머 함수를 사용한다. 이를 호출 스케줄링(scheduling a call)이라 한다.

자바스크립트는 타이머를 생성할 수 있는 타이머 함수 setTimeOut과 setInterval, 타이머를 제거할 수 있는 타이머 함수 clearTimeOut과 clearInterval을 제공한다. 타이머 함수는 ECMAScript 사양에 정의된 빌트인 함수가 아니다. 하지만 브라우저 환경과 Node.js환경에서 모두 전역 객체의 메서드로서 타이머 함수를 제공하고 있다. 즉, 타이머 함수는 호스트 객체다.

타이머 함수 setTimeOut과 setInterval은 모두 일정 시간이 경과된 이후 콜백 함수를 호출하는 타이머를 생성한다.

setTimeOut 함수는 일정 시간이 경과하여 타이머가 만료되면 콜백 함수를 한번 호출한다. 즉, setTimeOut 함수는 콜백 함수를 단 한번 호출한다.

setInterval 함수는 일정 시간이 경과하여 타이머가 만료될 때마다 콜백 함수를 계속 호출한다. 즉, setInterval 함수는 일정 시간 간격으로 콜백 함수를 무한히 반복 호출한다

자바스크립트 엔진은 단 하나의 실행 컨텍스트 스택을 갖기 때문에 동시에 두가지 이상의 태스크를 동시에 실행할 수 없다. 즉, 자바스크립트 엔진은 싱글 스레드(single thread)로 동작한다. 이런 이유로 타이머 함수 setTimeOut과 setInterval는 **비동기적**(asynchronous)으로 동작한다. 이에 대해서는 "42. 비동기 프로그래밍"에서 자세히 살펴볼 것이다.

2. 타이머 함수

2.1. setTimeout / clearTimeout

setTimeout 함수는 두 번째 인수로 전달한 시간(ms, 1/1000초)이 경과한 이후에 첫 번째 인수로 전달한 콜백 함수를 단 한번 호출한다.

JAVASCRIPT

```
const timeoutId = setTimeout(func|code[, delay, param1, param2, ...]);
```

매개변수	설명
func	타이머가 만료된 이후 호출할 콜백 함수를 전달한다. 즉, 콜백 함수는 호출 스케줄링된다. * 콜백 함수 대신 코드를 문자열로 전달할 수 있다. 이때 코드 문자열은 타이머가 만료된 뒤 해석되고 실행된다. 이는 흡사 eval 함수와 유사하며 권장하지는 않는다.
delay	함수를 호출하기까지 지연할 시간을 밀리초(ms) 단위로 전달한다. 인수 전달을 생략한 경우 기본값 0이 지정된다. * delay에 전달한 지연 시간이 정확히 보장되지는 않는다. 태스크 큐에 콜백 함수를 등록하는 시간을 지연할 뿐이다. 태스크 큐는 "42. 비동기 프로그래밍"에서 자세히 살펴볼 것이다. * delay가 4ms 이하인 경우, 최소 지연 시간 4ms가 지정된다.
param1, param2, ...	호출 스케줄링된 콜백 함수에 전달하여야 할 인수가 존재하는 경우 세 번째 이후의 인수로 전달할 수 있다. * Internet Explorer 9 이하에서는 콜백 함수에 인수를 전달할 수 없다.

JAVASCRIPT

```
// 1초(1000ms) 이후 첫 번째 인수로 전달한 함수 호출
setTimeout(() => console.log('Hi!'), 1000);

// 1초(1000ms) 이후 첫 번째 인수로 전달한 함수에 인수를 전달하면서 호출
setTimeout(name => console.log(`Hi! ${name}.`), 1000, 'Lee');

// 지연 시간을 생략하면 기본값 0이 지정된다.
setTimeout(() => console.log('Hello!'));
```

setTimeout 함수는 일정 시간이 경과한 이후 전달받은 콜백 함수를 호출하는 타이머를 생성하고, 생성된 타이머를 식별할 수 있는 고유한 타이머 id 값을 반환한다. setTimeout 함수가 반환하는 타이머 id 값은 브라우저 환경인 경우 숫자이며 Node.js 환경인 경우 객체다.

setTimeout 함수가 반환한 타이머 id를 clearTimeout 함수의 인수로 전달하여 타이머를 취소할 수 있다. 즉, clearTimeout 함수는 호출 스케줄링을 취소한다.

JAVASCRIPT

```
// 1초(1000ms) 이후 첫 번째 인수로 전달한 함수 호출
const timeoutId = setTimeout(() => console.log('Hi!'), 1000);

// setTimeout 함수가 반환한 타이머 id를 인수로 전달하여 타이머를 취소
clearTimeout(timeoutId);
```

2.2. setInterval / clearInterval

setInterval 함수는 두 번째 인수로 전달한 시간(ms, 1/1000초)이 경과할 때 마다 첫 번째 인수로 전달한 콜백 함수를 타이머가 취소될 때까지 호출한다. setTimeout 함수는 일정 시간이 경과한 이후 콜백 함수를 단 한번 호출하지만 setInterval 함수는 일정 시간 간격으로 콜백 함수를 무한히 반복 호출한다. setInterval 함수에 전달할 인수는 setTimeout 함수와 동일하다.

JAVASCRIPT

```
const timerId = setInterval(func|code[, delay, param1, param2, ... ]);
```

setInterval 함수는 일정 시간이 경과할 때 마다 전달받은 콜백 함수를 호출하는 타이머를 생성하고, 생성된 타이머를 식별할 수 있는 고유한 타이머 id 값을 반환한다. setInterval 함수가 반환하는 타이머 id 값은 브라우저 환경인 경우 숫자이며 Node.js 환경인 경우 객체다.

setInterval 함수가 반환한 타이머 id를 clearInterval 함수의 인수로 전달하여 타이머를 취소할 수 있다. 즉, setInterval 함수는 호출 스케줄링을 취소한다.

JAVASCRIPT

```
let count = 1;

// 1초(1000ms)마다 첫 번째 인수로 전달한 콜백 함수를 호출
const timeoutId = setInterval(() => {
  console.log(count); // 1 2 3 4 5
  // count가 5이면 타이머를 취소
  if (count++ === 5) clearInterval(timeoutId);
}, 1000);
```