함수 심화학습

전역 객체

- 전역 객체를 사용하면 어디서나 사용 가능한 전역 변수나 함수 생성 가능
  - 단, 전역 변수, 함수는 **전역적으로 사용하는 경우에만 생성**해야 함 (보통은 그런 경우는 잘 없음)
- 브라우저 환경에선 전역 객체를 window라고 부르는데, 각 호스트 환경마다 부르는 이름은 다를 수 있음
- alert, prompt, confirm을 포함한 **빌트인 함수들은 모두 글로벌 객체 소속의 전역 함수**

```
alert("Hello");
// 위와 동일하게 동작합니다.
window.alert("Hello");
```

- 중요한 변수 또는 함수라서 모든 곳에서 사용할 수 있게 하려면 아래와 같이 **명시적으로 전역 객체에 직접 프로퍼티를 추가**

```
// 모든 스크립트에서 현재 사용자(current user)에 접근할 수 있게 이를 전역 객체에 추가함
window.currentUser = {
    name: "John"
};

// 아래와 같은 방법으로 모든 스크립트에서 currentUser에 접근할 수 있음
alert(currentUser.name);  // John

// 지역 변수 'currentUser'가 있다면
// 지역 변수와 충돌 없이 전역 객체 window에서 이를 명시적으로 가져올 수 있음
alert(window.currentUser.name); // John
```

- 전역 함수는 몰라도, 전역 변수는 되도록 사용하지 않는 것이 좋음
- 가급적 함수를 만들 땐 외부 변수나 전역 변수를 함수 내부에서 사용하는 것보다 **전달받은 입력값만을 활용하여 값을 반환**해야 테스트하기도 쉽고 재사용이 용이함

setTimeout과 setInterval을 이용한 호출 스케줄링

- **호출 스케줄링(scheduling a call)** => **일정 시간이 지난 후에 원하는 함수를 예약 실행(호출)**할 수 있게 하는 것
  - 호출 스케줄링을 구현하는 방법 두 가지
    - setTimeout, setInterval 함수 사용

setTimeout

- 일정 시간이 지난 후에 함수를 실행하기 위해서 사용

```
let timerId = setTimeout(func|code, [delay], [arg1], [arg2], ...)
```

- func => 함수
- delay => ms 단위 대기 시간
- arg1, arg2, ... => 추가 가변 인수

```
function sayHi() {
    alert('안녕하세요.');
}

// 함수 전달
setTimeout(sayHi, 1000);
// 아래와 같은 실수하지 않기 (함수 실행 결과가 아닌, 함수를 전달)
// setTimeout(sayHi(), 1000);
```

- setTimout 함수에 가변 인수를 전달 가능
    - 해당 **가변 인수들이 호출할 함수의 인수값으로 전달**됨

```
function sayHi(who, phrase) {
    alert( who + ' 님, ' + phrase );
}

// 1초 뒤에 sayHi("홍길동", "안녕하세요.") 호출
setTimeout(sayHi, 1000, "홍길동", "안녕하세요.");

// 위의 방식 말고 익명 함수 전달 방식도 가능
setTimeout(function() {
    sayHi("홍길동", "안녕하세요.");
}, 1000);
```

- setTimeout을 호출하면 **타이머 식별자(timer identifier)**가 반환
    - clearTimeout 함수에 반환받은 식별자를 전달하여 **스케줄링 취소 가능**

```
// 브라우저 환경이라면 timerId는 정수
let timerId = setTimeout(...);
// 스케쥴링 취소 가능
clearTimeout(timerId);
```

setInterval

- **일정 시간 간격을 두고 함수를 실행**하기 위해서 사용
    - setTimeout이 함수를 단 한 번만 실행하는 것과 달리 setInterval은 **함수를 주기적으로 실행**
- setInterval 메서드는 setTimeout과 동일한 문법을 사용 (함수 호출 인자가 같음)
- 함수 호출을 중단하려면 clearInterval 함수를 사용

```
let timerId = setInterval(() => console.log('째깍'), 2000);
setTimeout(() => {
    clearInterval(timerId);
}, 6000);
```