클로져

Scope란?

자바스크립트에서 객체나 함수는 모두 변수(variable)입니다.

변수의 유효 범위(scope)란 해당 변수가 접근할 수 있는 변수, 객체 그리고 함수의 집합을 의미합니다.

자바스크립트에서 변수는 유효 범위에 따라 다음과 같이 구분됩니다.

1. 지역 변수(local variable)
2. 전역 변수(global variable)

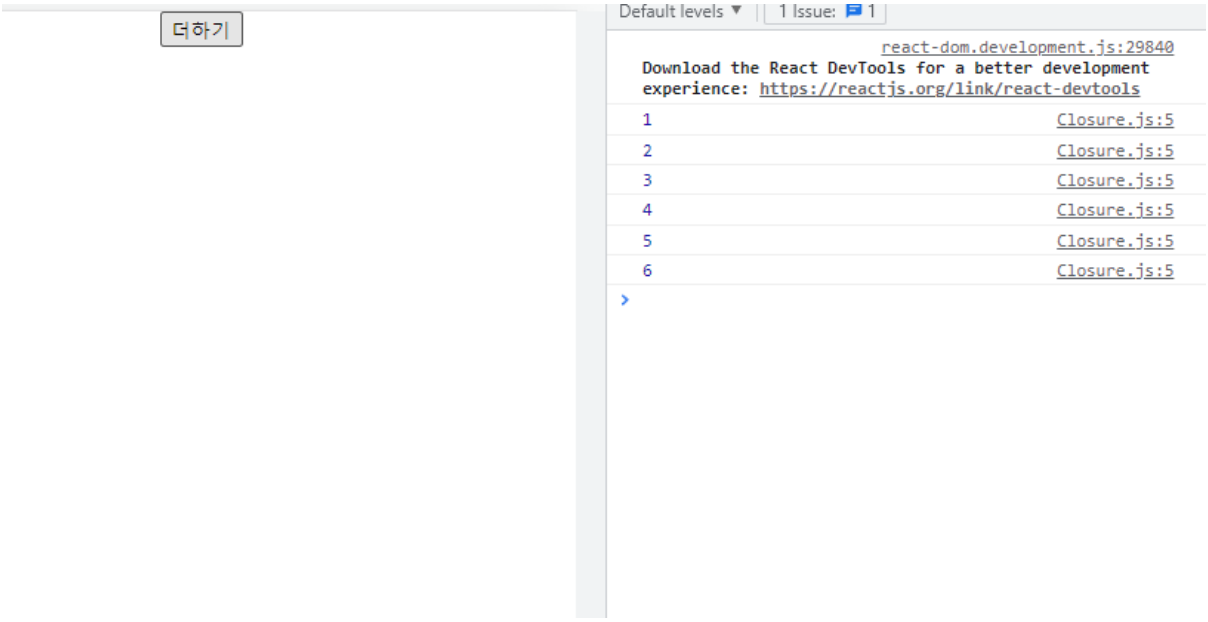
스코프 체인(scope chain)이란?

스코프 체인(Scope Chain)은 일종의 리스트로서 전역 객체와 중첩된 함수의 스코프의 레퍼런스를 차례로 저장하고, 의미 그대로 각각의 **스코프가 어떻게 연결(chain)되고 있는지 보여주는 것**을 말한다.

```jsx
export default function Closure() {
  const test1 = () =>{
    var a = 0
    return () => {
      console.log(++a)
    };
  }

  var test3 = test1()


  return (
    <div>
      <button onClick={test3}>더하기</button>
    </div>
  );
}
```



실행 컨텍스트(Execution context)

실행 가능한 코드를 형상화하고 구분하는 추상적인 개념이라고 정의하면 된다.쉽게 말하자면 코드들이 실행되기 위한 환경이라고 이해하면 될 것 같다.(코드가 실행된다면 `Execution Context` 내부에서 실행되고 있는 것이다.)

자바스크립트 엔진에서 코드를 실행하기 위해서는 실행에 필요한 정보를 알고 있어야한다.

- 변수 : 전역 변수, 지역 변수, 매개 변수, 객체의 프로퍼티
- 함수 선언
- 변수의 유효범위
- this

글로벌 실행 컨텍스트(Global Execution Context)란?

코드가 실행되기 전에 생성이 되며, 함수 내에 없는 코드는 모두 **전역 실행 컨텍스트 안에 존재**한다.

그렇기 때문에, 자바스크립트 엔진은 일부 자바스크립트 코드를 실행할 때마다 글로벌 실행 컨텍스트(Global Execution Context)를 작성한다.

글로벌 실행 컨텍스트의 특징으로는 무조건 **하나의 전역 실행 컨텍스트 만이 존재**하며, 애플리케이션이 종료될 때(웹 페이지에서 나가거나 브라우저를 닫을 때)까지 유지하는 것이다.

함수 실행 컨텍스트(Functional Execution Context)란?

전역 실행 컨텍스트가 생성된 후, 함수가 실행(ex 호출) 될 때마다 **새로운 실행 컨텍스트가 작성**된다.

클로져란?

클로저는 반환된 내부함수가 자신이 선언됐을 때의 환경(Lexical environment)인 스코프를 기억하여 자신이 선언됐을 때의 환경(스코프) 밖에서 호출되어도 그 환경(스코프)에 접근할 수 있는 함수를 말한다. 이를 조금 더 간단히 말하면 **클로저는 자신이 생성될 때의 환경(Lexical environment)을 기억하는 함수다**라고 말할 수 있겠다.

클로저에 의해 참조되는 외부함수의 변수 즉 test1 함수의 변수 a를 **자유변수(Free variable)**라고 부른다. 클로저라는 이름은 자유변수에 함수가 닫혀있다(closed)라는 의미로 의역하면 자유변수에 엮여있는 함수라는 뜻이다.

실행 컨텍스트의 관점에 설명하면, 내부함수가 유효한 상태에서 외부함수가 종료하여 외부함수의 실행 컨텍스트가 반환되어도, 외부함수 실행 컨텍스트 내의 **활성 객체(Activation object)**(변수, 함수 선언 등의 정보를 가지고 있다)는 **내부함수에 의해 참조되는 한 유효**하여 내부함수가 **스코프 체인**을 통해 참조할 수 있는 것을 의미한다.

즉 외부함수가 이미 반환되었어도 외부함수 내의 변수는 이를 필요로 하는 내부함수가 하나 이상 존재하는 경우 계속 유지된다. 이때 내부함수가 외부함수에 있는 변수의 복사본이 아니

라 실제 변수에 접근한다는 것에 주의하여야 한다.

클로저의 장점

1. 함수를 호출할 때마다 기존에 생성했던 값을 유지할 수 있다. 변수의 최신상태 유지에 용이
2. 외부에 해당 변수(참조하고 있는 변수)를 노출시키지 않는다. 전역적으로 노출 되고 있진 않지만, 전역적인 방식으로 사용될 수 있다는 것이다. 쉽게 말해 코드의 은닉성을 보장한다.

클로저의 단점

1. 클로저에 할당 된 변수는, 프로그램이 종료될 때 까지 메모리에 남아있다.

참고

https://ljtaek2.tistory.com/140

https://poiemaweb.com/js-closure