[모딥다 스터디 3주차] Scope

자바스크립트가 **참조의 연속** 이라고 하면,

Scope(유효 범위)는

- 사전적 의미 대로 : 유효한 범위
- 코드를 그대로 읽는 입장에서 : 컨텍스트 (문맥 → 글의 흐름)
- 자바스크립트 엔진의 입장에서 : 식별자(변수)가 무엇을 참조해야 할지 결정 → 식별자를 검색할 때 사용하는 규칙

- **ECMAScript Source Code 4가지 타입 → 각 Code에서 Scope를 형성**
  - Global Code : 전역에 영향을 주는 코드
    - not include : 대부분의 { } 를 쓰는 코드
  - Module Code : Module 내부의 코드, 그 안의
    - not include : 대부분의 { } 를 쓰는 코드
  - Function Code : function 내부의 코드, 중첩된 것 빼고
    - var 는 function scope, 함수 아니면 `{ }` 무시
    - let과 const는 block scope `{ }`
  - Eval Code : `eval` 을 절대 사용하지 말 것! → JSON 파싱 (문자열을 JavaScript 객체로)

- **코드의 실행 과정 : 표현 → 해석평가 → 실행 → 실행이 해석평가에 영향**
    i. 소스코드의 평가 : 무엇을 평가하나 - 문맥(의미 등)를 보고 순서, 범위를 정함
        a. 변수, 함수같은 선언하는 식별자들 먼저
        b. 식별자가 무엇을 참조해야 하는지 식별
    ii. Environment records in a lexical environment : 컴퓨터가 실행할 수 있게(1번에서 평가한 걸) 메모리에 저장
    iii. 소스코드의 실행 - 실행은 다시 2번에 영향을 주고 다시 실행

이해가 안되서 AI 선생님에게

1. **소스코드의 평가** : 이 단계에서는 프로그램이 실행될 준비를 합니다. 코드를 구성하는 각 요소들이 무엇을 의미하는지를 파악합니다. 이렇게 해석되는 요소들에는 변수, 함수, 클래스 등의 식별자가 포함되며, 이들의 사용 범위(scope)와 연관성(context)이 정해집니다. 즉, 어느 위치에서 어떤 식별자가 사용될 수 있는지, 그리고 그 식별자가 참조하는 대상은 무엇인지를 결정합니다.
2. **Environment Records in a Lexical Environment** : 이는 실행 환경을 의미합니다. 소스코드가 실행되면서 필요한 정보들(예: 변수의 값, 함수의 정의 등)을 저장하는 공간입니다. Lexical Environment는 실행할 코드의 문맥에 따라 변경되며, 실행 과정에서 현재 환경에 필요한 정보를 저장하고 조회하는 역할을 합니다.
3. **소스코드의 실행** : 평가 단계에서 준비된 정보를 바탕으로 실제로 코드를 실행하는 단계입니다. 이 단계에서 코드의 흐름에 따라 식별자들이 가진 값들이 바뀌고, 함수가 호출되며, 다양한 연산들이 수행됩니다. 이 과정에서 Lexical Environment는 계속 업데이트되며, 코드의 실행 상태를 최신으로 유지합니다. 이 단계에서 발생하는 모든 변화는 다시 평가 단계에 영향을 미칠 수 있습니다.

이 개념을 알아야

- callback function
- 나중에 나오는 closure

같은걸 잘 이해할 수 있지 않을까?

Scope의 Level 층의 영향력

넓은 범위의 참조 범위는 아랫 단계의 참조 단계들에게 영향을 줄 수 있지만

하위 단계는 또 그 아래만 영향을 줄 수 있고 반대는 안된다.

→ 아래에서는 위가 보이고, 위에서는 아래가 안보인다.

이렇게 위 아래로 연쇄적으로 검색하고 다니는 걸 스코프 체인이라고 하는거 같다

자바스크립트를

- 정의(만든시점)
- 호출(실행)

로 나누면 자바스크립트에서 Scope 범위는 정의(만든시점) 의 범위를 따른다.

이 원칙만 알면 된다