II. 실행 컨텍스트

1. 실행 컨텍스트란?

[참고 URL](https://gamguma.dev/post/2022/04/js_execution_context)

- 실행 컨텍스트 : 실행할 코드에 제공할 환경 정보들을 모아놓은 객체
- 어떤 실행 컨텍스트가 활성화될 때 자바스크립트 엔진은 해당 컨텍스트에 관련된 코드들을 실행하는 데 필요한 환경 정보들을 수집해서 실행 컨텍스트 객체에 저장합니다.
    - 이 객체는 자바스크립트 엔진이 활용할 목적으로 생성할 뿐 개발자가 코드를 통해 확인할 수는 없다.
    - **VariableEnvironment**
        - environmentRecord (snapshot)
        - outerEnvironmentReference (snapshot)
    - **LexicalEnvironment**
        - environmentRecord
        - outerEnvironmentReference
    - **ThisBinding**

2. VariableEnvironment

- VariableEnvironment에 담기는 내용은 LexicalEnvironment와 같지만 최초 실행 시의 스냅샷을 유지한다는 점이 다르다.
- 실행 컨텍스트를 생성하고 난 이후에는 LexicalEnvironment를 주로 활용하게 된다.

3. LexicalEnvironment

- environmentRecord
  - 현재 컨텍스트와 관련된 코드의 식별자 정보들(매개변수의 이름, 함수 선언, 변수명 등)이 저장된다.
- 호이스팅
  - 코드 해석을 좀 더 수월하게 하기 위해 environmentRecord의 수집 과정을 추상화한 개념이다.
  - 변수 선언과 값 할당이 동시에 이루어진 문장은 '선언부'만을 호이스팅하고, 할당 과정은 원래 자리에 남아있게 된다.
  - 여기서 함수 선언문과 함수 표현식의 차이가 발생한다.
- 스코프 체인
  - '식별자의 유효범위'를 안에서부터 바깥으로 차례로 검색해나가는 것
  - 이를 가능케 하는 것이 바로 outerEnvironmentReference
- outerEnvironmentReference
  - 현재 호출된 함수가 선언될 당시의 LexicalEnvironment를 참조한다.
  - 연결리스트의 형태를 띤다.
  - '선언 시점의 LexicalEnvironment를' 계속 찾아 올라가면 마지막엔 전역 컨텍스트의 LexicalEnvironment가 있을 것이다.
  - 스코프에서 동일한 식별자를 선언한 경우에는 **무조건 스코프 체인 상에서 가장 먼저 발견된 식별자에만 접근 가능**하게 된다.
- 전역변수 : 전역 컨텍스트의 LexicalEnvironment에 담긴 변수
  지역변수 : 그 밖의 함수에 의해 생성된 실행 컨텍스트의 변수들

4. this

- 실행 컨텍스트의 thisBinding에는 this로 지정된 객체가 저장된다.
- 실행 컨텍스트 활성화 당시에 this가 지정되지 않은 경우 this에는 전역 객체가 저장된다.