23장. 실행 컨텍스트

▼ 실행 컨텍스트란?

자바스크립트의 동작 원리를 담고 있는 핵심 개념

식별자(변수, 함수, 클래스 등의 이름)를 등록하고 관리하는 스코프와 코드 실행 순서 관리를 구현한 내부 메커니즘으로, 모든 코드는 실행 컨텍스트를 통해 실행되고 관리된다.

23.1 소스코드의 타입

▼ 전역 코드

전역 변수를 관리하기 위해 최상위 스코프인 전역 스코프를 생성

전역 코드가 평가되면 전역 실행 컨텍스트 생성

▼ 함수 코드

지역 스코프를 생성하고 지역 변수, 매개변수, arguments 객체를 관리

함수 코드가 평가되면 함수 실행 컨텍스트가 생성

▼ eval 코드

strict mode(엄격모드)에서 자신만의 독자적인 스코프를 생성

eval 코드가 평가되면 eval실행 컨텍스트가 생성

▼ 모듈 코드

모듈별로 독립적인 모듈 스코프를 생성

모듈 코드가 평가되면 모듈 실행 컨텍스트가 생성

23.2 소스코드의 평가와 실행

▼ 소스코드 과정

1. 소스코드의 평가 → 실행 컨텍스트를 생성하고 변수, 함수 등의 선언문만 먼저 실행

   생성된 변수나 함수 식별자를 키로 실행 컨텍스트가 관리하는 스코프에 등록

2. 소스코드의 실행 → 선언문을 제외한 소스코드가 순차적으로 실행되기 시작 [=런타임]

소스코드의 실행 결과는 다시 실행 컨텍스트가 관리하는 스코프에 등록

23.3 실행 컨텍스트의 역할

▼ 자바스크립트 엔진이 코드를 평가하고 실행하는 과정

1. 전역 코드 평가 - 전역 코드의 변수 선언문과 함수 선언문이 먼저 실행
2. 전역 코드 실행 - 전역 변수에 값이 할당되고 함수가 호출(이때, 함수 내부로 먼저 진입)
3. 함수 코드 평가 - 함수 내 매개변수와 지역 변수 선언문이 먼저 실행
4. 함수 코드 실행 - 매개변수와 지역 변수에 값이 할당되고 console.log 메서드 호출

23.4 실행 컨텍스트 스택

▼ 실행 컨텍스트 스택이란?

생성된 실행 컨테스트는 스택 자료구조로 관리 = 코드의 실행 순서를 관리

실행 컨텍스트 스택의 최상위에 존재하는 실행 컨텍스트는 현재 실행 중인 코드의 실행 컨텍스트

▼ 실행 중인 실행 컨텍스트란?

실행 컨텍스트 스택의 최상위에 존재하는 실행 컨텍스트

23.5 렉시컬 환경

▼ 렉시컬 환경이란?

식별자와 식별자에 바인딩된 값, 그리고 상위 스코프에 대한 참조를 기록하는 자료구조로

실행 컨텍스트를 구성하는 컴포넌트

실행 컨텍스트 스택 = 코드의 실행 순서를 관리

렉스컬 환경 = 스코프와 식별자 관리

렉시컬 환경은 스코프를 구분하여 식별자를 등록하고 관리하는 저장소 역할을 하는

렉시컬 스코프의 실체

▼ 렉시컬 환경 구성 컴포넌트

1. 환경 레코드(EnvironmentRecord) - 스코프에 포함된 식별자를 등록하고 등록된 식별자에 바인딩된 값을 관리하는 저장소
2. 외부 렉시컬 환경에 대한 참조(OuterLexicalEnvironmentReference) - 상위 스코프를 가리키며 상위 스코프를 통해 단방향 링크드 리스트인 스코프 체인을 구현한다.

   **상위 스코프 = 해당 실행 컨텍스트를 생성한 소스코드를 포함하는 상위 코드의 렉시컬 환경

23.6 실행 컨텍스트의 생성과 식별자 검색 과정

▼ 전역 객체 생성

전역 코드가 평가되기 이전에 생성

전역 객체도 Object.prototype을 상속받음 → 전역 객체도 프로토타입 체인의 일원

▼ 전역 코드 평가

소스코드가 로드되면 자바스크립트 엔진은 전역 코드를 평가

순서

1. 전역 실행 컨텍스트 생성
2. 전역 렉시컬 환경 생성

   2.1 전역 환경 레코드 생성

   2.1.1 객체 환경 레코드 생성

   2.1.2 선언적 환경 레코드 생성

   2.2 this 바인딩

   2.3 외부 렉시컬 환경에 대한 참조 결정

▼ 세부적인 생성 과정

1. **전역 실행 컨텍스트 생성**

비어있는 전역 실행 컨텍스트를 생성하여 실행 컨텍스트 스택에 푸시

→ 이때, 전역 실행 컨텍스트는 실행 컨텍스트 스택의 최상위(=실행 중인 실행 컨텍스트)가 된다.

2. **전역 렉시컬 환경 생성**

   전역 렉시컬 환경을 생성하고 전역 실행 컨텍스트에 바인딩한다.

   2.1 전역 환경 레코드 생성

   전역 변수를 관리하는 전역 스코프, 전역 객체의 빌트인 전역 프로퍼티와 빌트인 전역 함수, 표준 빌트인 객체를 제공한다.

   → let, const 키워드로 선언한 전역 변수는 전역 객체의 프로퍼티가 되지 않고 개념적인 블록 내에 존재(선언적 환경 레코드)

   2.1.1 객체 환경 레코드 생성

   전역 코드 평가 과정에서 var 키워드로 선언한 전역 변수와 함수 선언문으로 정의된 전역 함수는 전역 환경 레코드의 객체 환경 레코드에 연결된 BindingObject를 통해 전역 객체의 프로퍼티와 메서드가 된다.

   2.1.2 선언적 환경 레코드 생성

   let, const 키워드로 선언한 전역 변수는 선언적 환경 레코드에 등록되고 관리한다.

   2.2 this 바인딩

   전역 환경 레코드의 [[GlobalThisValue]] 내부 슬롯에 this가 바인딩된다.

   → this 바인딩은 전역 환경 레코드와 함수 환경 레코드에만 존재

   2.3 외부 렉시컬 환경에 대한 참조 결정

   현재 평가 중인 소스코드를 포함하는 외부 소스코드의 렉시컬 환경(=상위 스코프)

   → 이를 통해 단방향 링크드 리스트인 스코프 체인 구현 가능

▼ 전역 코드 실행

변수 할당문이 실행되어 변수에 값이 할당괴고 함수가 있다면 호출된다.

→ 먼저 변수 또는 함수 이름이 선언된 식별자인지 확인해야 함, 선언되지 않은 식별자는

할당/호출을 할 수 없기 때문

▼ 식별자 결정이란?

동일한 이름의 식별자가 다른 스코프에 여러 개 존재할 수 있기 때문에 어느 스코프의 식별자를 참조하면 되는지 결정할 필요를 가리킨다.

식별자 결정을 위해 식별자 검색할 때는 실행 중인 실행 컨텍스트에서 식별자를 검색하기 시작

▼ 함수 코드 평가

함수가 호출되면 전역 코드 실행을 일시 중단하고 함수 내부로 코드의 제어권이 이동하고 함수

코드 평가를 시작한다.

순서

1. 함수 실행 컨텍스트 생성
2. 함수 렉시컬 환경 생성

   2.1 함수 환경 레코드 생성

   2.2 this 바인딩

   2.3 외부 렉시컬 환경에 대한 참조 결정

▼ 세부적인 생성 과정

1. **함수 실행 컨텍스트 생성**

   함수 실행 컨텍스트를 생성하고 생성된 함수 실행 컨덱스트는 함수 렉시컬 환경이 완성된 다음 실행 컨텍스트 스택에 푸시된다.

   → 이때 함수 실행 컨텍스트는 실행중인 컨텍스트(실행 컨텍스트 스택의 최상위)가 된다.

2. **함수 렉시컬 환경 생성**

   렉시컬 환경을 생성하고 함수 실행 컨텍스트에 바인딩한다.

   2.1 함수 환경 레코드 생성

   매개변수, argements 객체, 함수 내부에서 선언한 지역 변수와 중첩 함수를 등록하고 관리한다.

   2.2 this 바인딩

함수 호출 방식에 따라 [[ThisValue]] 내부 슬롯에 바인딩될 객체가 결정된다.

2.3 외부 렉시컬 환경에 대한 참조 결정

함수 정의가 평가된 시점에 실행 중인 실행 컨텍스트의 렉시컬 환경의 참조가 할당된다.

▼ 함수 코드 실행

함수의 소스코드가 순차적으로 실행되기 시작한다. 매개변수에 인수가 할당되고, 변수 할당문이 실행되어 지역 변수에 값이 할당된다.

→ 이때 식별자 결정을 위해 실행 중인 실행 컨텍스트의 렉시컬 환경에서 식별자를 검색하기

시작한다.

23.7 실행 컨텍스트와 블록 레벨 스코프