Chap14 전역 변수의 문제점

14-1 변수의 생명주기

▼ 변수의 생명주기

변수는 선언에 의해 생성되고 할당을 통해 값을 가진 다음 언젠가 아무도 참조하지 않으면 소멸에 이르는 기간

▼ 지역 변수의 생명주기

```
function foo() {
  var x = 'local';
  console.log(x); // local
  return x;

}

foo();
console.log(x); // ReferenceError: x is not defined
```

- 지역변수 x는 foo가 호출되기 이전에는 생성되지 않았다.
- 지역변수 x는 foo가 호출되고 실행되는 런타임 직전에 미리 선언하고 undefined로 초기화 된다. (호이스팅)
- 함수의 생명주기가 끝나 더이상 참조하지 않으면 지역 변수 x의 생명주기는 끝난다. 따라서 함수 밖에서 x를 차조하려했을때 ReferenceError가 난다.
- 즉 함수의 생명주기와 지역변수의 생명주기는 일치한다.(다른 곳에서 지역변수를 참조하지 않고 있으면...)

▼ 전역 변수의 생명주기

- 명시적인 호출 없이 코드가 로드되자마자 곧바로 해석되고 실행된다.

- 따라서 전역 변수의 생명 주기는 전역 코드의 마지막 문이 실행되어 더 이상 실행할 문이 없을 때 종료되고 전역 변수도 소멸된다.
- `var` 키워드로 선언한 전역 변수는 전역 객체의 프로퍼티가 된다. 이는 전역 변수의 생명 주기는 전역 객체의 생명 주기와 일치한다는 것이다.
- 브라우저 환경에서는 `window`가 전역 객체이므로 `var` 키워드로 선언한 전역 변수는 `window` 객체의 프로퍼티이다. 브라우저 환경에서는 `var` 키워드로 선언한 전역 변수는 웹페이지를 닫을 때까지 유효하다.

▼ 전역객체

코드가 실행되기 이전 단계에 자바스크립트 엔진에 의해 어떤 객체보다도 먼저 생성되는 특수한 객체.

전역객체는 클라이언트 사이드 환경(브라우저)에서는 window, 서버 사이드 환경(Node.js)에서는 global 객체를 의미한다. 환경에 따라 전역 객체를 가리키는 다양한 식별자가 존재했으나 ES11에서 globalThis로 통일되었다.

14-2 전역 변수의 문제점

- 암묵적 결합 : 모든 코드가 전역변수를 참조,변경할 수 있음.변수의 유효 범위가 클수록 코드의 가독성은 나빠지고 의도치 않게 상태가 변경될 수 있는 위험성이 높아진다.
- 긴 생명주기
- 스코프 체인 상에서 종점에 존재 : 전역변수는 스코프 체인 종점에 존재하기때문에 가장 마지막에 검색되어 검색 속도가 가장 느리다.
- 네임스페이스 오염 : 다른 파일 내에서 동일한 이름으로 명명된 전역 변수나 전역 함수가 같은 스코프 내에 존재할 경우 예상치 못한 결과를 초래

14-3 전역 변수의 사용을 억제하는 방법

- 즉시 실행 함수 활용. 모든 코드를 즉시 실행 함수로 감싸면 모든 변수는 지역변수가 된다.
- 네임스페이스 객체
- 모듈 패턴
- ES6 모듈