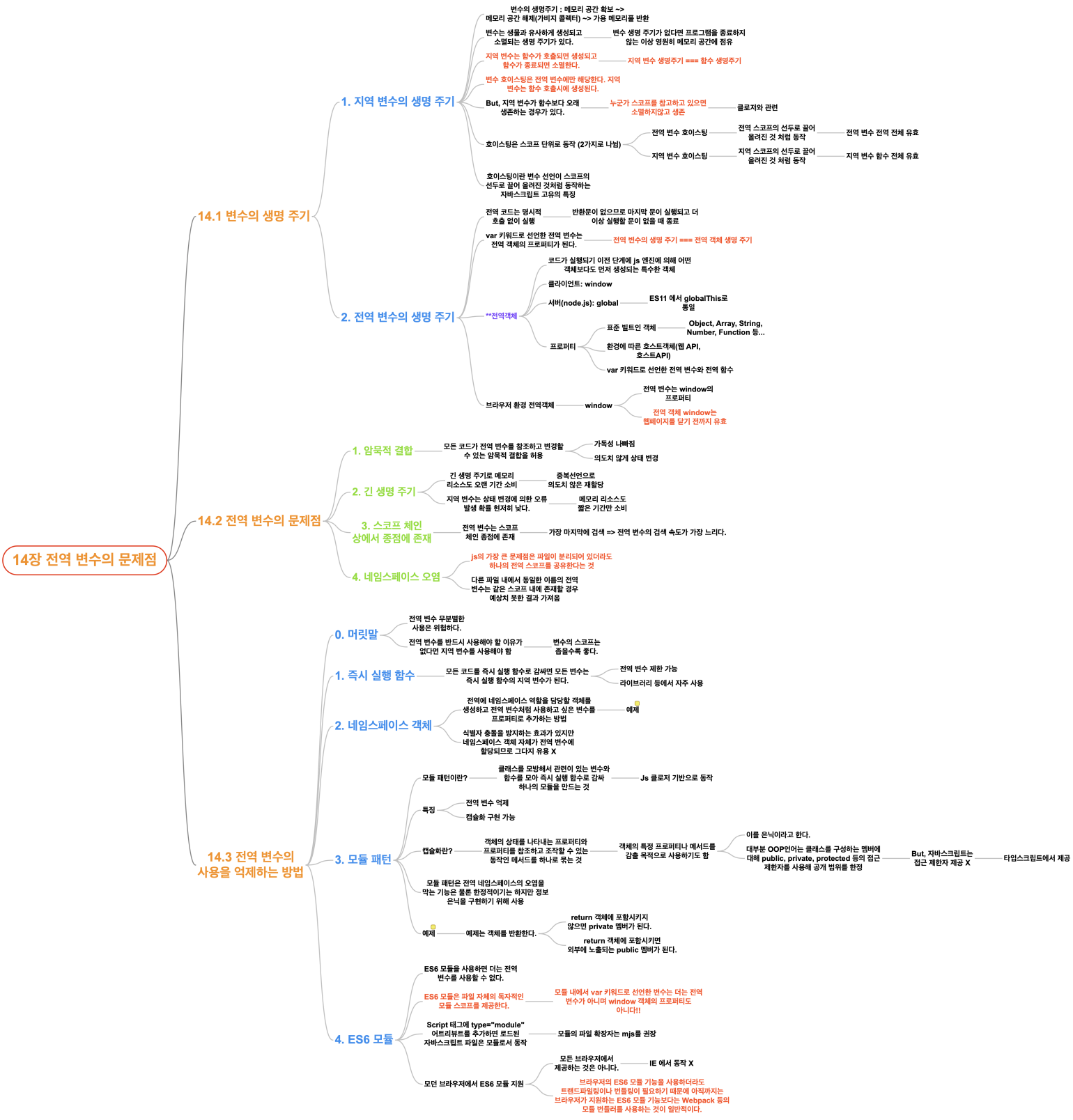
14장 전역 변수의 문제점
14.1 변수의 생명 주기
1. 지역 변수의 생명 주기
변수의 생명주기 : 메모리 공간 확보 ~> 메모리 공간 해제(가비지 콜렉터) ~> 가용 메모리풀 반환
변수는 생물과 유사하게 생성되고 소멸되는 생명 주기가 있다.
변수 생명 주기가 없다면 프로그램을 종료하지 않는 이상 영원히 메모리 공간에 점유
지역 변수는 함수가 호출되면 생성되고 함수가 종료되면 소멸한다.
지역 변수 생명주기 === 함수 생명주기
변수 호이스팅은 전역 변수에만 해당한다. 지역 변수는 함수 호출시에 생성된다.
But, 지역 변수가 함수보다 오래 생존하는 경우가 있다.
누군가 스코프를 참고하고 있으면 소멸하지않고 생존
클로저와 관련
호이스팅은 스코프 단위로 동작 (2가지로 나뉨)
전역 변수 호이스팅
전역 스코프의 선두로 끌어 올려진 것 처럼 동작
전역 변수 전역 전체 유효
지역 변수 호이스팅
지역 스코프의 선두로 끌어 올려진 것 처럼 동작
지역 변수 함수 전체 유효
호이스팅이란 변수 선언이 스코프의 선두로 끌어 올려진 것처럼 동작하는 자바스크립트 고유의 특성
2. 전역 변수의 생명 주기
전역 코드는 명시적 호출 없이 실행
반환문이 없으므로 마지막 문이 실행되고 더 이상 실행할 문이 없을 때 종료
var 키워드로 선언한 전역 변수는 전역 객체의 프로퍼티가 된다.
전역 변수의 생명 주기 === 전역 객체 생명 주기
**전역객체
코드가 실행되기 이전 단계에 js 엔진에 의해 어떤 객체보다도 먼저 생성되는 특수한 객체
클라이언트: window
서버(node.js): global
ES11 에서 globalThis로 통일
프로퍼티
표준 빌트인 객체
Object, Array, String, Number, Function 등...
환경에 따른 호스트객체(웹 API, 호스트API)
var 키워드로 선언한 전역 변수와 전역 함수
브라우저 환경 전역객체
window
전역 변수는 window의 프로퍼티
전역 객체 window는 웹페이지를 닫기 전까지 유효
14.2 전역 변수의 문제점
1. 암묵적 결합
모든 코드가 전역 변수를 참조하고 변경할 수 있는 암묵적 결합을 허용
가독성 나빠짐
의도치 않게 상태 변경
2. 긴 생명 주기
긴 생명 주기로 메모리 리소스도 오랜 기간 소비
중복선언으로 의도치 않은 재할당
지역 변수는 상태 변경에 의한 오류 발생 확률 현저히 낮다.
메모리 리소스도 짧은 기간만 소비
3. 스코프 체인 상에서 종점에 존재
전역 변수는 스코프 체인 종점에 존재
가장 마지막에 검색 => 전역 변수의 검색 속도가 가장 느리다.
4. 네임스페이스 오염
Js의 가장 큰 문제점은 파일이 분리되어 있더라도 하나의 전역 스코프를 공유한다는 것
다른 파일 내에서 동일한 이름의 전역 변수는 같은 스코프 내에 존재할 경우 예상치 못한 결과 가져옴
14.3 전역 변수의 사용을 억제하는 방법
0. 머릿말
전역 변수 무분별한 사용은 위험하다.
전역 변수를 반드시 사용해야 할 이유가 없다면 지역 변수를 사용해야 함
변수의 스코프는 좁을수록 좋다.
1. 즉시 실행 함수
모든 코드를 즉시 실행 함수로 감싸면 모든 변수는 즉시 실행 함수의 지역 변수가 된다.
전역 변수 제한 가능
라이브러리 등에서 자주 사용
2. 네임스페이스 객체
전역에 네임스페이스 역할을 담당할 객체를 생성하고 전역 변수처럼 사용하고 싶은 변수를 프로퍼티로 추가하는 방법
예제
식별자 충돌을 방지하는 효과가 있지만 네임스페이스 객체 자체가 전역 변수에 할당되므로 그다지 유용 X
3. 모듈 패턴
모듈 패턴이란?
클래스를 모방해서 관련이 있는 변수와 함수를 모아 즉시 실행 함수로 감싸 하나의 모듈을 만드는 것
Js 클로저 기반으로 동작
특징
전역 변수 억제
캡슐화 구현 가능
캡슐화란?
객체의 상태를 나타내는 프로퍼티와 프로퍼티를 참조하고 조작할 수 있는 동작인 메서드를 하나로 묶는 것
객체의 특정 프로퍼티나 메서드를 감출 목적으로 사용하기도 함
이를 은닉이라고 한다.
대부분 OOP언어는 클래스를 구성하는 멤버에 대해 public, private, protected 등의 접근 제한자를 사용해 공개 범위를 한정
But, 자바스크립트는 접근 제한자 제공 X
타입스크립트에서 제공
모듈 패턴은 전역 네임스페이스의 오염을 막는 기능은 물론 한정적이기는 하지만 정보 은닉을 구현하기 위해 사용
예제
예제는 객체를 반환한다.
return 객체에 포함시키지 않으면 private 멤버가 된다.
return 객체에 포함시키면 외부에 노출되는 public 멤버가 된다.
4. ES6 모듈
ES6 모듈을 사용하면 더는 전역 변수를 사용할 수 없다.
ES6 모듈은 파일 자체의 독자적인 모듈 스코프를 제공한다.
모듈 내에서 var 키워드로 선언한 변수는 더는 전역 변수가 아니며 window 객체의 프로퍼티도 아니다!!
Script 태그에 type="module" 어트리뷰트를 추가하면 로드된 자바스크립트 파일은 모듈로서 동작
모듈의 파일 확장자는 mjs를 권장
모던 브라우저에서 ES6 모듈 지원
모든 브라우저에서 제공하는 것은 아니다.
IE 에서 동작 X
브라우저의 ES6 모듈 기능을 사용하더라도 트랜드파일링이나 번들링이 필요하기 때문에 아직까지는 브라우저가 지원하는 ES6 모듈 기능보다는 Webpack 등의 모듈 번들러를 사용하는 것이 일반적이다.