**1. 함수 실행시 생성되는 실행컨텍스트란 무엇인가. 아는대로 쓰시오.**

실행 가능한 코드를 형상화하고 구분하는 추상적인 개념이라고 정의하면 된다. 쉽게 말하자면 코드들이 실행되기 위한 환경이라고 이해하면 될 것 같다.

전역 코드 : 전역 영역에 존재하는 코드  
Eval 코드 : eval 함수로 실행되는 코드  
함수 코드 : 함수 내에 존재하는 코드

자바스크립트 엔진은 코드를 실행하기 위하여 실행에 필요한 여러가지 정보를 알고 있어야 한다.

변수 : 전역변수, 지역변수, 매개변수, 객체의 프로퍼티  
함수 선언  
변수의 유효범위(Scope)  
this

이와 같이 실행에 필요한 정보를 형상화하고 구분하기 위해 자바스크립트 엔진은 실행 컨텍스트를 물리적 객체의 형태로 관리한다. 아래의 코드를 살펴보자.

var x = 'xxx';

function foo () {

  var y = 'yyy';

  function bar () {

    var z = 'zzz';

    console.log(x + y + z);

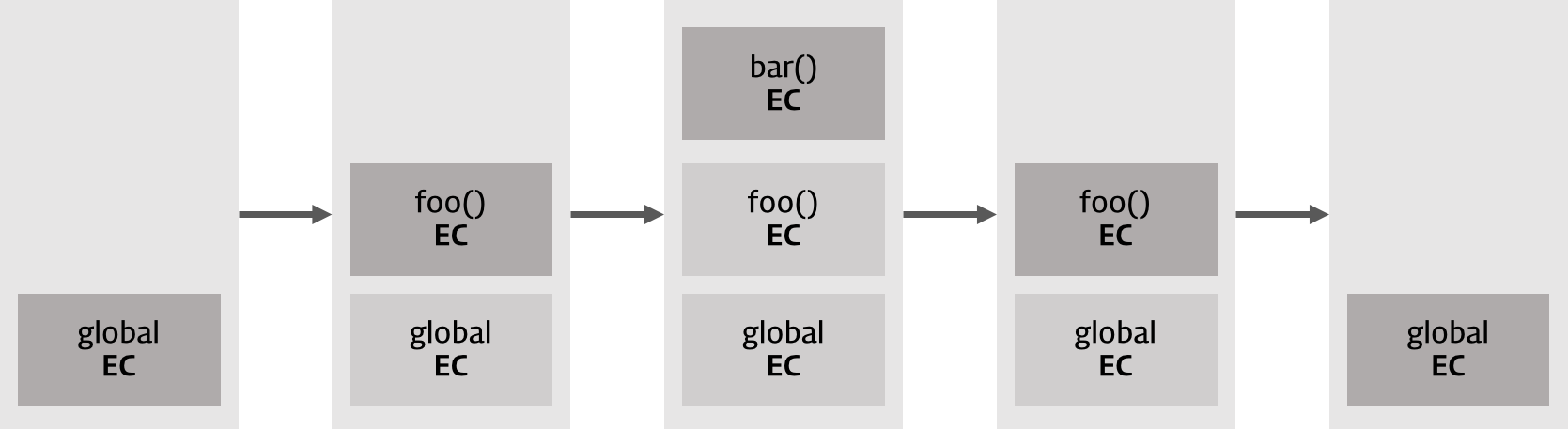
  }

  bar();

}

foo();

위 코드를 실행하면 아래와 같이 실행 컨텍스트 스택(Stack)이 생성하고 소멸한다. 현재 실행 중인 컨텍스트에서 이 컨텍스트와 관련없는 코드(예를 들어 다른 함수)가 실행되면 새로운 컨텍스트가 생성된다. 이 컨텍스트는 스택에 쌓이게 되고 컨트롤(제어권)이 이동한다.



논리적 스택 구조를 가지는 실행 컨텍스트 스택

컨트롤이 실행 가능한 코드로 이동하면 논리적 스택 구조를 가지는 새로운 실행 컨텍스트 스택이 생성된다. 스택은 LIFO(Last In First Out, 후입 선출)의 구조를 가지는 나열 구조이다.

전역 코드(Global code)로 컨트롤이 진입하면 전역 실행 컨텍스트가 생성되고 실행 컨텍스트 스택에 쌓인다. 전역 실행 컨텍스트는 애플리케이션이 종료될 때(웹 페이지에서 나가거나 브라우저를 닫을 때)까지 유지된다.

함수를 호출하면 해당 함수의 실행 컨텍스트가 생성되며 직전에 실행된 코드 블록의 실행 컨텍스트 위에 쌓인다. 함수 실행이 끝나면 해당 함수의 실행 컨텍스트를 파기하고 직전의 실행 컨텍스트에 컨트롤을 반환한다.

2. 실행 컨텍스트의 3가지 객체

실행 컨텍스트는 실행 가능한 코드를 형상화하고 구분하는 추상적인 개념이지만 물리적으로는 객체의 형태를 가지며 아래의 3가지 프로퍼티를 소유한다.

2.1 Variable Object (VO / 변수객체)

실행 컨텍스트가 생성되면 자바스크립트 엔진은 실행에 필요한 여러 정보들(변수/매개변수(parameter) / 인수 정보(arguments) /함수 선언(함수 표현식은 제외) ) 을 담을 객체를 생성한다.

2.2 Scope Chain (SC)

스코프 체인은 식별자 중에서 객체(전역 객체 제외)의 프로퍼티가 아닌 식별자, 즉 변수를 검색하는 메커니즘이다. 식별자 중에서 변수가 아닌 객체의 프로퍼티(물론 메소드도 포함된다)를 검색하는 메커니즘은 프로토타입 체인(Prototype Chain)이다.

2.3 this value

this 프로퍼티에는 this 값이 할당된다. this에 할당되는 값은 함수 호출 패턴에 의해 결정된다.

출처 : <https://poiemaweb.com/js-execution-context>