[js] 함수 호이스팅(Function Hoisting)

Created By 해연 김

Last Edited 10월 03, 2018 6:39 오후

Tags Empty

Property 1 Empty

Add a Property

Add a comment...

3가지의 정의 방식이 있는데, 정의 방식은 달라도 결국 Function() 생성자 함수를 통해 함수를 생성함. 함수 정의 방식은 동작 방식에 약간의 차이가 있음.

```
var res = add(1); function add(number) { return number+number;
}
```

JavaScript

위의 코드에서 함수 선언식으로 함수가 정의되기 이전에 함수 호출이 가능함.

함수의 선언 위치와는 상관없이 코드 내 어느 곳에서든지 호출이 가능한데, 이 것을 함수 호이스팅이라함.

자바스크립트는 ES6의 let, const를 포함하여 모든 선언(var, let, const, function, function*, class)를 호이스팅함.

호이스팅이란 var 선언문이나 function 선언문 등을 해당 scope의 맨 위로 옮기는 것을 말함.

즉 자바스크립트 코드는 코드를 실행하기 전에 var 선언문과 function 선언문을 해당 스코프의 맨 위로 옮김.

함수 선언식으로 정의된 함수는 자바스크립트 엔진이 스크립트가 로딩되는 시점에 바로 초기화

함고 이를 VO(variable object)에 저장함.
즉, 함수 선언, 초기화, 할당이 한번에 이루어짐.

그렇기 때문에 선언의 위치와는 상관없이 소스 내 어느 곳에서든지 호출가능함.

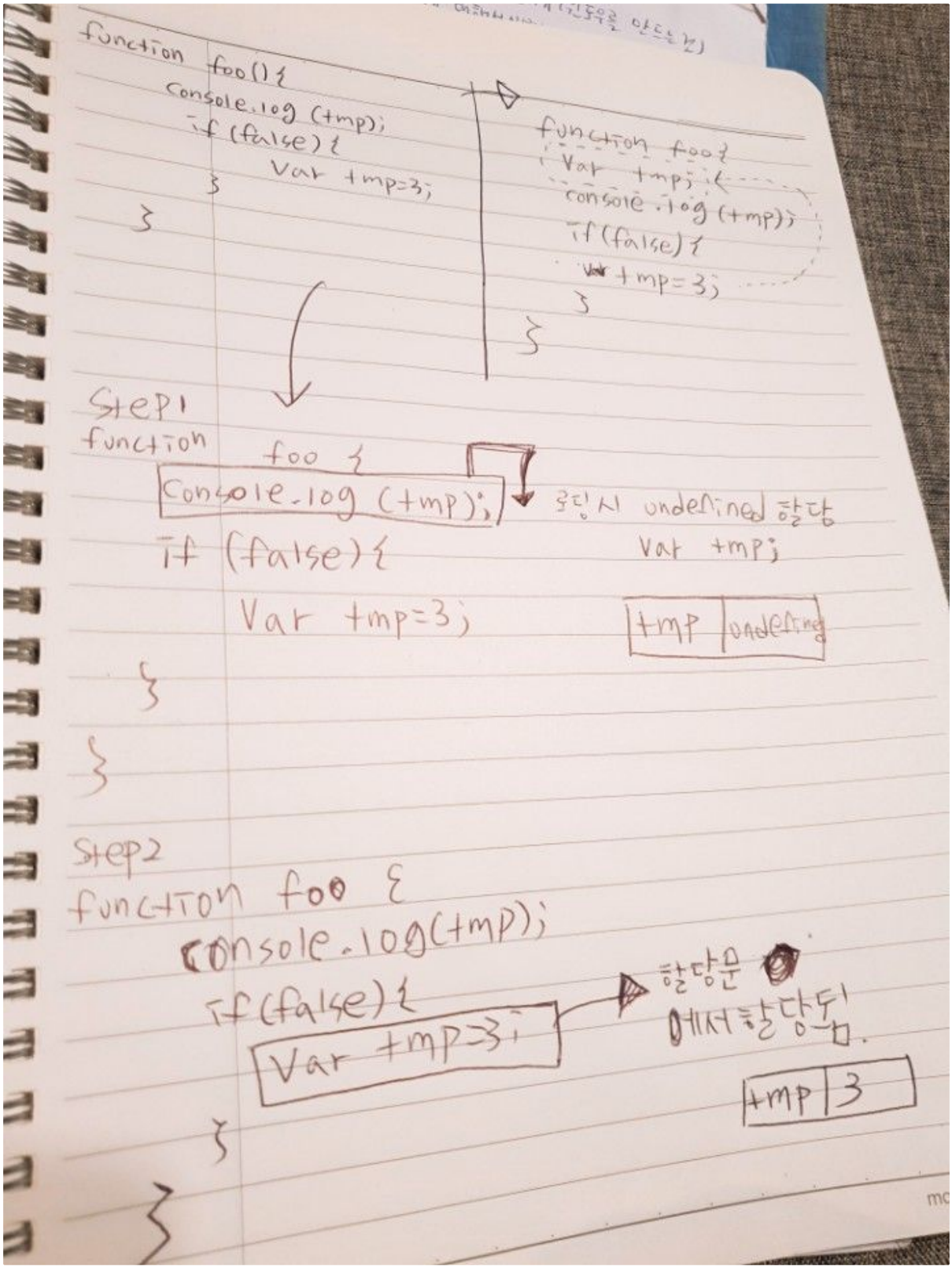
다음은 함수 표현식으로 함수를 정의한 경우임.

```
var res = add(1); var add = function(number) { return
number+number; }
```

JavaScript

함수 선언식의 경우와는 달리 TypeError하였음.

함수 표현식의 경우 함수 호이스팅이 아니라 변수 호이스팅이 발생함.



> 변수 호이스팅은 변수 생성과 할당이 분리되어 진행됨. 즉 , 변수 선언만 호이스팅되고 값의 초기화는 변수 선언문에서 이루어짐. 호이스팅된 변수에는 undefined가 우선 할당되고 실제값의 할당은 할당문에서 이루어짐.

함수 표현식은 함수 선언식과는 달리 스크립트 로딩 시점에 VO(Variable Object)에 함수를 저장하지 않고 runtime에 해석이 되므로 이 두가지를 구분하는 것은 중요함.

javascript : 더글러스 크락포드는 이와 같은 문제 때문에 함수 표현식만을 사용할 것을 권고하고 있음.

함수 호이스팅이 함수 호출 전 반드시 함수를 선언하여야 한다는 규칙을 무시하므로 코드의 구조를 엉성하게 만들 수 있다고 지적함.

함수 선어식으로 함수를 정의하면 사용하기에 쉽지만 대규모 어플리케이션을 개발하는 경우 인터프리터가 너무 많은 코드를 VO에 저장하므로 애플리케이션의 응답 속도는 현저히 떨어질 수 있으므로 주의해야 할 필요가 있음.