스코프 & 호이스팅

태그 이론

1. global variable(전역변수)

전역블록스코프(window)에 선언된 변수를 전역변수라고 한다. 전역변수는 어느누구든 접근가능하다. 전역변수의 경우 어느 누구든 접근이 가능하기도 하고 어플리케이션이 실행하고 끝날 때까지 변수가 선언된상태이기 때문에 항상 메모리가 탑재된 상태이기 때문에 전연변수를 남용하게 된다면 그만큼 메모리 낭비가 되는 것이며 성능적으로 좋지 못하다.

```
let globalName = 'global name';
{
  let name = 'ellie';
  console.log(name);
  name = 'hello';
  console.log(name);
  console.log(globalName);
}
console.log(name);
console.log(globalName);
```

2. local variable(지역변수)

블록스코프안에 선언된 변수를 지역변수라고 한다. 예를 들면, 선언된 함수 안에 선언된 변수를 지역변수라고 하며, 선언된 함수안의 코드내에서 지역변수를 사용가능하지만, 스코프를 벗어난 경우에는 스코프안에 변수는 접근 가능하지 않다.

3. block scope (블록스코프)

{} 중괄호로 묶어 놓은 것을 블록 스코프라고 하며 블록스코프안에 선언된 변수와 변수에 할당된 값은 블록스코프안에서만 접근이 가능하다.

let, const, var 변수할당 키워드

키워드 1) 재할당, 2) 재선언, 3) 초기값, 4) 호이스팅

세개 모두 변수를 선언하는 방식으로,

var 는 재선언, 재할당이 가능하고 초기값을 설정하지 않거나 호이스팅이 일어나도 오류가 나지 않습니다.

여기서 호이스팅은 함수안에 있는 선언들을 모두 끌어올려서 함수의 유효범위의 최 상단에 선언하는것을 말합니다.

똑같은 변수에 다른값을 계속적으로 넣어도 가장 최신에 할당된 값이 적용되기 때문에 예기치 못한 값을 반환합니다.

이것을 보완하기 위해 ES6 에 나온 변수 선언방식이 let, const 입니다.

let 은 재선언은 불가하나 재할당이 가능하고 초기값을 설정하지 않아도 됩니다.

const 는 재선언, 재할당 둘다 불가능하나 객체의 재할당은 가능합니다.

4. let, const

☝ let

mutable한 값을 변수에 할당할 때 쓰는 변수 키워드이다, 즉 할당된 값이 어떠한 조건에 의해 변경을 해야하는 경우 쓰여진다.

☝ const

immuable한 값을 변수에 할당할 때 쓰는 변수 키워드이다. 할당된 값이 어떠한 조건에도 변경이 되어서는 안되는 경우에 쓰여진다.

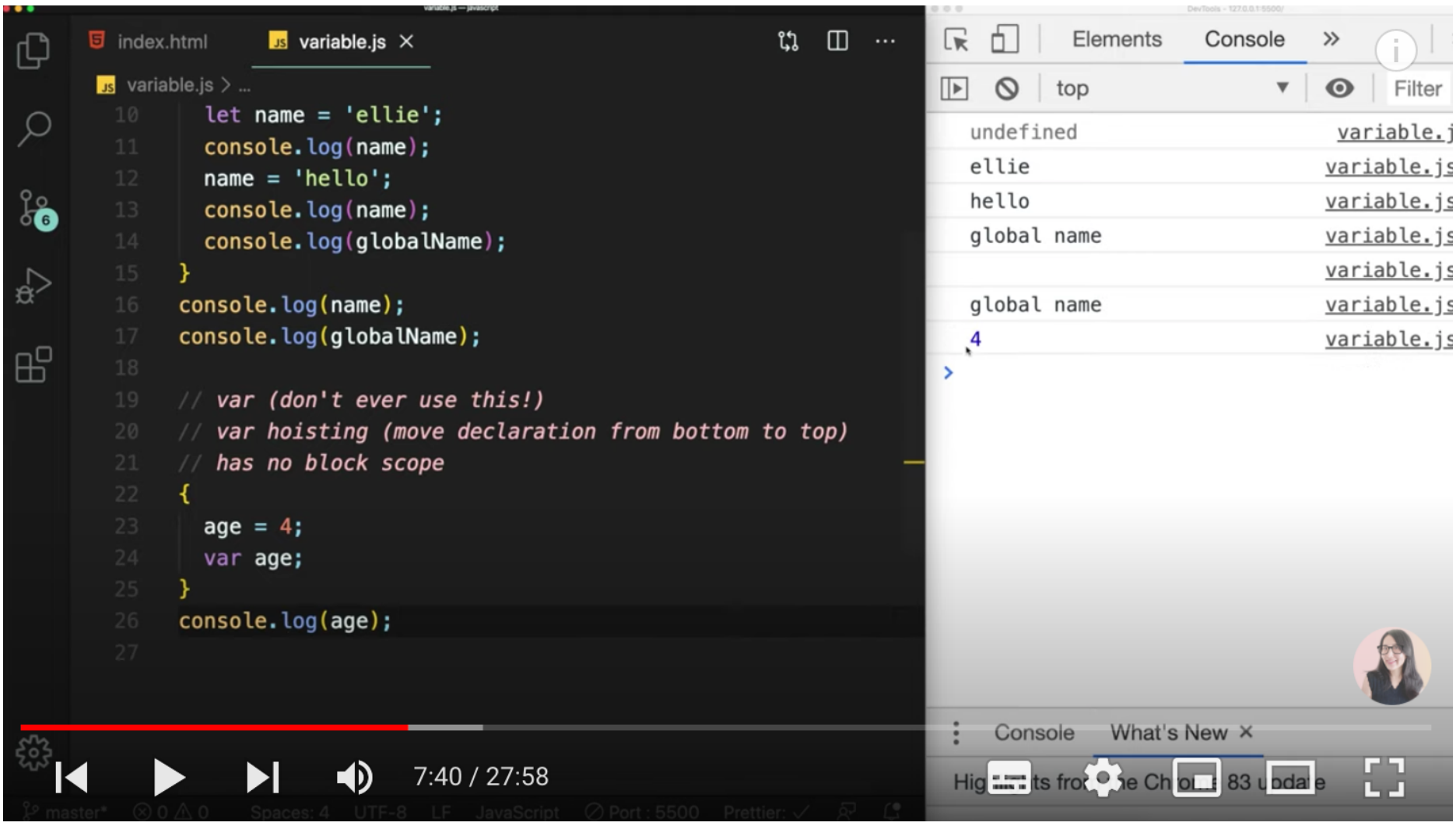
- let을 사용하여 변수를 선언하였을 때 발생할 수 있는 에러

5. var

재선언, 재할당이 가능한 변수키워드이다. var로 선언된 변수의 경우 hoisting이 일어나 선언된 위치에 상관없이 어느 곳에서 호출하여도 에러가 나지 않으며, undefined(값이 없는상태)로 출력이 되어진다. 그리고 var로 선언된 변수는 블록스코프를 무시한다.

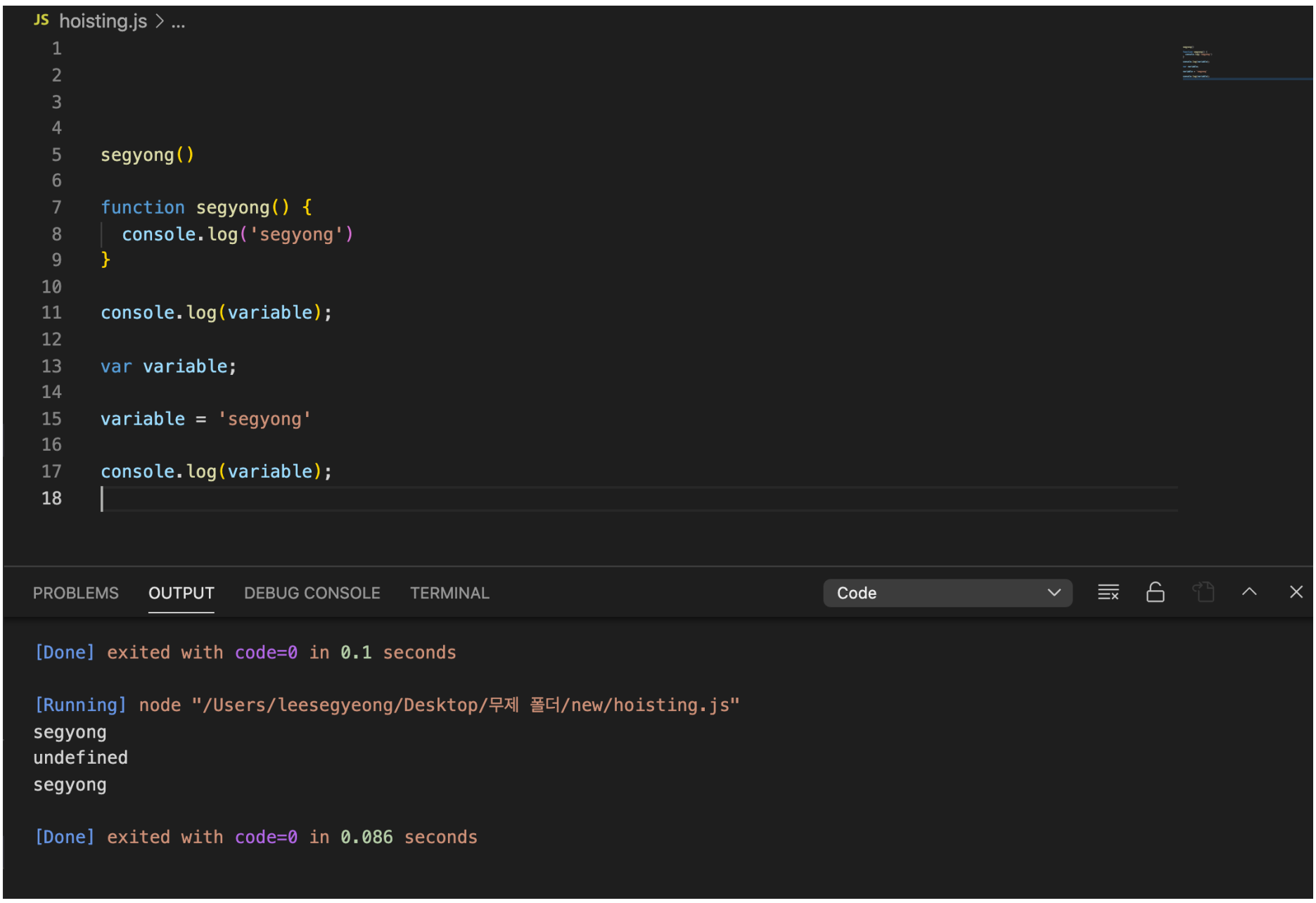
블록스코프에 var로 선언된 변수의 값을 블록스코프 밖에서 호출했을 경우에도 에러가 나지않는다.

이렇게 var키워드를 남용하게 된다면, 언제든 할당된 값이 바꿀수 있다는 점과 지역변수를 전역으로 사용이 되어진다는 점, 또한 선언된 위치와 상관없이 호출이 되어진다는 부분에서 코드의 많은 문제점이 발생하게 된다.



6. hoisting

선언된 시점을 무시하고 코드가 최상단으로 끌어올려지는 것을 말한다. 자바스크립트의 특성상 선언된 함수, var키워드는 선언된 시점을 무시하고 어디서 호출하여도 선언된 함수를 먼저 읽고, 코드를 실행이 되기 때문에 보통 코드를 위에서 아래로 읽어나가면서 실행되는 자바스크립트에서 혼돈의 카오스를 맞이 하게 된다.



호이스팅이란 변수나 함수의 호출 코드가 선언코드보다 아래쪽에 있음에도 불구하고 에러가 발생하지 않고, 선언코드를 위로 끌어올려서 함수 유효범위의 최상단에서 선언되어지는 것을 말합니다. 즉 함수 내에서 아래쪽에 존재하는 내용 중 필요한 것들만을 끌어올리는 것입니다. 호이스팅의 대상이 되는 것은 var변수선언과 함수 선언문에서만 호이스팅이 일어나게 된다.

함수 선언식으로 함수를 선언했다면 다른 코드보다 먼저 읽고 실행하는 특징이 있습니다. 그래서 함수선언식 보다 먼저 함수를 실행할 경우에도 오류가 발생하지 않습니다. 이러한 개념을 호이스팅이라고 합니다. 즉, 함수선언보다 함수실행을 먼저하여도 오류가 나지 않고 함수실행이 된다는 점입니다. 자바스크립트 안에서는 함수선언식을 먼저 읽고 실행하는 특징 있기 때문입니다. 하지만 함수 표현식으로 함수를 선언하고 해당 변수에 함수를 할당하는 형태라고 보면 함수 선언식과는 다르게 함수를 먼저 실행해 주지 않기 때문에 함수 선언 전 위에서 호출을 해주게 되면 오류가 발생하게 됩니다. 이때는 호이스팅이 되지 않아 함수 호출 시 오류가 발생하는 것입니다.

극명한 차이를 알 수 있는 예시가 바로 자바스크립트에서 함수를 생성하는 두가지 방법인 함수선언식과 함수표현식이 있다.

block scope: 블록 스코프

```
function getName(user) {
   return user.name;
}

let getAge = function(user) {
  return user.age;
}
```

function scope: 함수 스코프

```
if(true) {
  console.log('i am in the block');
}

for(let i = 0; i < 10; i++){
  console.log(i);
}

let getAge = user => {
  return user.age;
}
```