자바스크립트에서 var, let, const는 변수를 선언할 때 사용되지만, 각각의 특성과 스코프가 다릅니다. 이를 이해하면 코드의 가독성과 안정성을 높일 수 있습니다. 아래는 var, let, const의 비교와 예제입니다.

1. var

- **스코프**: 함수 스코프
    - 함수 내부에서 선언되면 함수 전체에서 유효하며, 함수 밖에서는 접근할 수 없습니다.
- **호이스팅**: 호이스팅이 되어 **선언이 함수 또는 스코프의 최상위로 끌어올려집니다**. 초기화는 선언 위치에서 이루어지기 때문에 초기화 전에는 undefined입니다.
- **재선언 가능**: 같은 스코프에서 여러 번 선언할 수 있습니다.
- **재할당 가능**: 언제든지 값을 변경할 수 있습니다.

예제:

javascript

코드 복사

function example() { console.log(a); *// undefined (호이스팅으로 선언은 되었으나 초기화는 안 됨)* var a = 10; var a = 20; *// 재선언 가능* a = 30; *// 재할당 가능* console.log(a); *// 30* } example();

2. let

- **스코프**: 블록 스코프
    - { }로 감싸진 블록 내에서만 유효합니다.
- **호이스팅**: 호이스팅이 되지만 **초기화 전에 접근할 수 없어** Temporal Dead Zone**(TDZ)이 발생**합니다.
- **재선언 불가**: 같은 스코프에서 다시 선언할 수 없습니다.
- **재할당 가능**: 선언 후 값을 변경할 수 있습니다.

예제:

javascript

코드 복사

function example() { console.log(b); *// ReferenceError: 초기화 전이므로 접근 불가 (TDZ)* let b = 10; *// let b = 20; // SyntaxError: 같은 스코프에서 재선언 불가* b = 30; *// 재할당 가능* console.log(b); *// 30* } example();

3. const

- **스코프**: 블록 스코프
    - { }로 감싸진 블록 내에서만 유효합니다.
- **호이스팅**: 호이스팅이 되지만 **초기화 전에 접근할 수 없어 TDZ가 발생**합니다.
- **재선언 불가**: 같은 스코프에서 다시 선언할 수 없습니다.
- **재할당 불가**: 값 변경이 불가능합니다. 선언 시 반드시 초기값이 필요합니다.
- **불변 객체 아님**: 객체나 배열의 경우 **참조는 변경할 수 없지만 내부 값은 수정할 수 있습니다**.

예제:

javascript

코드 복사

function example() { const c = 10; *// const c = 20; // SyntaxError: 같은 스코프에서 재선언 불가 // c = 30; // TypeError: 재할당 불가* const arr = [1, 2, 3]; arr.push(4); *// 가능 (배열 요소는 수정 가능)* console.log(arr); *// [1, 2, 3, 4]* } example();

요약 비교표

특성	var	let	const
스코프	함수 스코프	블록 스코프	블록 스코프
호이스팅	선언은 호이스팅됨	선언은 호이스팅됨 (TDZ)	선언은 호이스팅됨 (TDZ)
재선언	가능	불가능	불가능
재할당	가능	가능	불가능
초기화 필요 여부	필요 없음	필요 없음	반드시 초기화 필요

자바스크립트에서는 기본적으로 const를 우선 사용하고, 필요할 때만 let을 사용하는 것이 권장됩니다. var는 예기치 않은 오류를 일으킬 수 있어 거의 사용되지 않는 추세입니다.