ES6에서 추가된 let과 const 키워드는 var와 어떻게 다른가요?

ES5에서 ES6로 넘어오면서 let 과 const가 도입되면서 생긴 차이

ES5에서는 주로 `var`를 사용했지만, ES6에서는 `let`과 `const`가 도입되면서 변수 선언 및 할당에 더욱 명확하고 안전한 방식이 제공

▼ ES5와 ES6에서의 주요 차이점✔

1. 변수 선언 방식

- ES5: `var`를 사용하여 변수를 선언.
- ES6: `let`과 `const`가 도입되어 변수 선언 방식이 다양해졌으며, 각각의 특성에 따라 적절한 변수 선언을 선택할 수 있다.

2. 스코프

- ES5: `var`로 선언한 변수는 함수 스코프를 가진다. 이로 인해, 스코프 관리가 어렵고 실수로 인한 오류 발생이 쉽다.
- ES6: `let`과 `const`는 블록 스코프를 가지므로, 코드 블록 내에서만 변수를 사용할 수 있어 스코프 관리가 훨씬 쉬워졌다.

> 📖 스코프 (Scope)란?
> 스코프는 변수의 가시성과 생명주기를 결정하는 범위. 변수가 어디서 접근 가능한지, 언제 생성되고 소멸되는지를 결정하는 영역.
> JavaScript에서는 주로 세 가지 유형의 스코프가 있다.
>
> - 전역 스코프 (Global Scope): 코드 전체에서 접근 가능한 변수를 선언하는 영역. 전역 변수는 어디서든 참조하거나 수정할 수 있으므로, 사용 시 주의가 필요함.
> - 함수 스코프 (Function Scope): 함수 내부에서만 접근 가능한 변수를 선언하는 영역. 함수 내에서 선언된 변수는 해당 함수의 실행이 종료되면 메모리에서 해제 된다. `var` 키워드로 선언된 변수는 함수 스코프를 가짐.
> - 블록 스코프 (Block Scope): 중괄호 `{}`로 둘러싸인 코드 블록 내에서만 접근 가능한 변수를 선언하는 영역. 블록 스코프는 일반적으로 `if`, `for`, `while` 등의 제어문이나 코드 블록에 적용됩니다. `let`과 `const` 키워드로 선언된 변수는 블록 스코프를 가짐.

3. 호이스팅

- ES5: `var`로 선언한 변수는 호이스팅이 발생. 이로 인해 변수를 선언하기 전에 참조할 수 있게 되어, 의도치 않은 결과가 발생할 수 있다.

- ES6: `let`과 `const`는 호이스팅이 발생하지만, 초기화는 선언이 실행될 때까지 이루어지지 않아 일시적 사각지대(TDZ)가 발생. 이를 통해 변수 사용에 대한 실수를 방지할 수 있다.

> 📖 호이스팅 (Hoisting)
> 호이스팅은 변수와 함수의 선언을 코드의 최상단으로 끌어올리는 것을 의미. JavaScript 엔진은 실행 전에 변수와 함수의 선언을 먼저 처리함. 이 과정에서 변수와 함수의 선언이 해당 스코프의 최상단으로 끌어올려진다.
>
> - 변수 호이스팅: 선언과 초기화가 동시에 이루어지며, 최상단에서 변수가 `undefined`로 초기화. 이로 인해 변수를 선언하기 전에 참조할 수 있게 되어, 의도치 않은 결과가 발생할 수 있다.
> - 함수 호이스팅: 함수 선언식은 호이스팅이 발생. 함수 전체가 해당 스코프의 최상단으로 끌어올려지므로, 함수를 선언하기 전에 호출할 수 있다. 반면, 함수 표현식은 호이스팅이 발생하지 않는다.
> - `let`과 `const` 호이스팅: `let`과 `const`로 선언된 변수는 호이스팅이 되지만, 초기화는 선언이 실행될 때까지 이루어지지 않는다. 이로 인해 일시적 사각지대(TDZ, Temporal Dead Zone)가 발생하며, 선언 전에 참조하려고 하면 참조 에러(ReferenceError)가 발생. 이 특성은 의도치 않은 변수 참조를 방지하고 코드의 안전성을 높여준다.

4. 재할당 및 재선언

- ES5: `var`로 선언한 변수는 재할당 및 재선언이 모두 가능하므로, 의도치 않은 값 변경이나 변수 선언 오류가 발생하기 쉽다.
- ES6: `let`은 재할당이 가능하지만 재선언이 불가능하며, `const`는 재할당 및 재선언 모두 불가능. 이로 인해 코드 안정성이 향상되었다.

5. 코드의 간결성 및 가독성

- ES5: 변수 선언 및 할당에 대한 규칙이 비교적 느슨하고, 스코프 및 호이스팅 관련 문제로 인해 코드를 이해하고 유지 관리하기 어려울 수 있다.
- ES6: `let`과 `const`의 도입으로 변수 선언 및 할당에 대한 규칙이 더 엄격해져, 코드의 간결성 및 가독성이 향상되었다. 이를 통해 개발자들이 코드를 더 쉽게 이해하고 유지 관리할 수 있게 되었다.

6. 브라우저 지원

- ES5: 대부분의 브라우저에서 지원되므로 호환성 문제가 거의 없다.

- ES6: 최신 브라우저에서는 대부분의 ES6 기능을 지원하지만, 구형 브라우저에서는 일부 기능이 지원되지 않을 수 있다. 이 경우, Babel과 같은 트랜스파일러를 사용하여 ES6 코드를 ES5 코드로 변환해야 호환성을 보장할 수 있다.

---

var, let, const

▼ ES6 (ECMAScript 2015)에서 추가된 `let`과 `const` 키워드가 가지는 주요 차이점 ✓

1. `var`는 함수 스코프를 가지며 호이스팅이 발생하고, 재할당 및 재선언이 가능. 이로 인해 오류 발생 가능성이 높다.
2. `let`은 블록 스코프를 가지며 호이스팅은 발생하지만 TDZ로 인해 참조 에러가 발생하며, 재할당이 가능하지만 재선언은 불가능. 이로 인해 `var`보다 안전하다.
3. `const`는 블록 스코프를 가지며 호이스팅은 발생하지만 TDZ로 인해 참조 에러가 발생하며, 재할당 및 재선언이 모두 불가능. 상수를 사용하는 경우 가장 안전한 선택이다.
4. 일반적으로 변하지 않는 값을 사용할 때는 `const`를 값이 변경될 수 있는 경우에는 `let`을 사용.

▼ 자세한 차이점✔

1. 스코프 (Scope)
   `var`는 함수 스코프 (function scope)를 가지는 반면, `let`과 `const`는 블록 스코프 (block scope)를 가진다. 이 말은 `var` 변수가 함수 내 어디에서든 접근할 수 있지만, `let`과 `const` 변수는 선언된 블록 내에서만 접근할 수 있다는 말이다.
   - `var`: 함수 스코프 (Function scope)를 가진다. 함수 내부에서 선언된 변수는 해당 함수 전체에서 접근할 수 있다. 함수 외부에서 선언된