Warum du let und const anstatt var nutzen sollst

Die Verwendung von `var` in JavaScript wird heutzutage **nicht mehr empfohlen**, außer in sehr spezifischen Fällen. Stattdessen sollte `let` und `const` bevorzugt werden. Hier sind die Gründe und ein Vergleich der drei Schlüsselwörter:

1. Probleme mit `var`

a. Funktionale Scoping

- `var` ist **funktional** gescoped, nicht blockscoped.
- Variablen, die mit `var` deklariert werden, sind außerhalb von Block-Scope (z. B. in Schleifen oder if-Blöcken) zugänglich, was zu unerwartetem Verhalten führen kann.
- **Beispiel**:

```javascript
if (true) { var x = 10;} // `x` ist außerhalb dieses Blocks sichtbar
console.log(x); // 10
```

b. Hoisting

- `var` -Deklarationen werden beim **Hoisting** (der Verarbeitung vor der Codeausführung) an den Anfang ihrer Funktion oder des globalen Scopes verschoben.
- Dadurch kann eine `var` -Variable verwendet werden, bevor sie tatsächlich deklariert wurde.
- **Beispiel**:

```javascript
console.log(a); // undefined (keine Fehlermeldung, weil `var` gehostet wird)
var a = 5;
```

c. Risiko unbeabsichtigter globaler Variablen

- Wenn `var` versehentlich ohne `var` , `let` oder `const` verwendet wird, erstellt es eine **globale Variable**, was zu schwer nachvollziehbaren Fehlern führen kann.
- **Beispiel**:

```javascript
function test() { x = 5; // Erstellt eine globale Variable (schlecht!) }
test();
console.log(x); // 5
```

2. Vorteile von `let` und `const`

a. Block-Scoped

- `let` und `const` sind **block-scoped**, d. h. sie sind nur innerhalb des Blocks zugänglich, in dem sie deklariert wurden.
- **Beispiel**:

```javascript
if (true) { let y = 20; const z = 30; }
console.log(y); // Fehler: y is not defined
```

b. Kein unbeabsichtigtes Hoisting

- Während `let` und `const` technisch auch "gehostet" werden, können sie vor ihrer Deklaration nicht verwendet werden, da sie sich im **"temporal dead zone" (TDZ)** befinden.
- **Beispiel**:

```javascript
console.log(b); // Fehler: Cannot access 'b' before initialization let b = 10;
```

c. Unveränderliche Variablen mit `const`

- `const` erlaubt die Deklaration von **konstanten Variablen**, deren Wert nicht neu zugewiesen werden kann.
- **Beispiel**:

```javascript
const pi = 3.14;
pi = 3.14159; // Fehler: Assignment to constant variable.
```

---

3. Soll `var` jemals verwendet werden?

`var` sollte nur in folgenden seltenen Fällen in Betracht gezogen werden:

- Wenn du mit alten JavaScript-Versionen arbeiten musst, die `let` und `const` nicht unterstützen (z. B. vor ES6).
- Wenn du gezielt das funktionale Scoping von `var` nutzen möchtest (was selten sinnvoll ist).

Empfehlung

- **Verwende** `let` für Variablen, deren Wert sich ändern soll.
- **Verwende** `const` für Variablen, deren Wert nicht neu zugewiesen werden soll.
- **Vermeide** `var` in modernem JavaScript-Code. In den meisten modernen Codebases ist es obsolet.

Beispiel:

```javascript
const name = "Alice"; // Konstant
let age = 30; // Veränderliche Variable
age = 31;
```