Variablen in ECMA6/TypeScript

Kurzfassung:

Zur Deklaration von Variablen mit **var** gesellen sich die Deklaration mit **let** und **const**.

Letztere beide beachten Blockscope.

var-Keyword

```
<script type="text/javascript">
// Hoisting:
var test;                          Variable erreichbar, aber nicht belegt.

// ursprüngliche Deklaration:
var test = "Ein Test";             Variable erreichbar und belegt.

</script>
```

let-Keyword

```
<script type="text/javascript">
// Kein Hoisting!
```

Variable nicht erreichbar. **TDZ-Error!**

```
// Deklaration:
let test;
```

Variable erreichbar.

```
// Wertzuweisung:
test = "Ein Test";
```

Variable erreichbar und belegt.

```
</script>
```

const-Keyword

```
<script type="text/javascript">
// Kein Hoisting!
```

Variable nicht erreichbar. **TDZ-Error!**

```
// Deklaration:
const test = "Ein Test";
// überschreiben?
test = "Anderer Test";
```

Variable erreichbar und belegt.

(Überschreiben verboten!)

```
</script>
```

var-Keyword vs. let-Keyword

Im globalen Scope ist der Unterschied gering.

- Mit **var** vermeidet man Referenzfehler.
- Mit **let** riskiert man "temporal dead zone" Errors im Bereich *vor* der Deklaration.

Merke: Niemals eine **let**-Variable *vor* deren Deklaration anfassen! Die Existenz einer **let**-Variable ist NICHT TESTBAR (mit typeof)!

let-Keyword vs. const-Keyword

Im globalen Scope praktisch gleichwertig.

- Eine **let** -Variable darf neu belegt werden. Sie muss nicht initialisiert werden.
- Eine **const** -Variable *darf nicht* überschrieben werden und *muss* mit einem Wert initialisiert werden.

... betrachten wir **const** als einen "Kontrakt" mit der Laufzeit, dass der Wert gleichbleibt!

var-Keyword: Kein Blockscope

```
<script type="text/javascript">
// Hoisting:
 var i;                                  Variable i erreichbar, aber nicht belegt.


 // var ist nicht auf Block begrenzbar:
 for (var i = 0; i<5; i++) {             Variable i erreichbar und belegt.
        console.log( i );
 }
 console.log("i nach der Schleife:", i);
</script>
```

let-Keyword: Beachtet Blockscope

```
<script type="text/javascript">
// Kein Hoisting!
```

Variable j nicht erreichbar. **TDZ-Error!**

```
 // let-Variable GILT NUR im Block:
 for (let j = 0; j<5; j++) {
        console.log( j );
 }
 // console.log("j nach der Schleife:",  j );
</script>
```

Variable erreichbar und belegt.

TDZ-Error!

var-Keyword vs. let-Keyword

In Blöcken sind **let**-Variable sinnvoll:

- Als Schleifenzähler.
- Als (wirklich) temporäre Variablen.
- Zum Umspeichern (*value swap*)

Vorteil: Die **let**-Variable bleibt im Block, kann nach dem Block "abgeräumt" werden (keine Memory-Leaks.

const-Keyword: "Konstante" Funktionen

```
<script type="text/javascript">
// Kein Hoisting!
```

Variable nicht erreichbar. **TDZ-Error!**

```
// Initialisierung mit Funktionsobjekt:
const test = function () {
    console.log("Ich bin ein Test!");
}
test();
```

Nicht überschreibbare Funktion.

```
</script>
```

"Pro" const-Keyword:

Die Idee besteht darin, dass ein Speicherplatz für genau *ein* Konzept dient

Soll z.B. ein von diesem Konzept *abgeleiteter* Wert gespeichert werden, so wird ein neuer Speicherplatz verwendet.

Eine **const**-Variable verdeutlicht dies.

Globale Funktionen sollten als Konstante angelegt werden.