2025/11/20 06:35 1/5 LU02a - Variablen

## LU02a - Variablen

### **Einleitung**

Variablen sind Platzhalter für Daten (Kontainer), die im Programm gespeichert und später wiederverwendet werden können. Sie ermöglichen es, Werte wie Zahlen, Texte oder Objekte mit einem Namen zu versehen, anstatt sie direkt im Code mehrfach zu verwenden. Dadurch wird der Code flexibler, besser lesbar und leichter wartbar.

Variablen dienen in JavaScript dazu, Werte zu speichern und wiederzuverwenden. Sie können Zahlen, Texte, Objekte, Funktionen und vieles mehr aufnehmen.

### **Keywords**

Zur Deklaration von Variablen stehen drei Keywords zur Verfügung:

- var
- let
- const

#### var

Das älteste Schlüsselwort in JavaScript. Es ist funktion-skopiert (gültig innerhalb der Funktion) und nicht block-skopiert (gültig im gesamten Block) und wird beim Programmstart an den Anfang des Scopes verschoben.

**Hinweis:** Heutzutage wird var selten empfohlen, da es leicht zu unerwartetem Verhalten führt, ist also veraltet.

#### **Beispiel**

```
function demoVar() {
  if (true) {
    var name = "Volkan";
  }
  console.log(name); // funktioniert, obwohl innerhalb des if-Blocks
deklariert (scope innerhalb der Funktion
}
demoVar();
```

Last update: 2025/09/01 13:29

Wie in der Abbildung zu sehen ist, ist die Ausgabe der Variable *name* fehlerfrei, obwohl diese ausserhalb des if-Blocks definiert wurde. Der Scope (Gültigkeitsbereich) ist bei var die Funktion.

#### let

Seit ES6 (2015) eingeführt. let ist block-skopiert, d. h. die Variable existiert nur innerhalb des Blockes  $\{ \dots \}$ , in dem sie definiert ist.

#### **Beispiel**

```
function demoLet() {
  if (true) {
    let age = 42;
    console.log(age); // 42
  }
  // console.log(age); // Fehler: age ist hier nicht definiert
}
demoLet();
```

https://wiki.bzz.ch/ Printed on 2025/11/20 06:35

Wie in der Ausgabe zu sehen, wird *Markpoint 1:* fehlerfrei ausgegeben, *Markpoint 2:* liefer dann eine Fehlermeldung, weil let ausserhalb des IF-Blocks deklariert wurde.

#### const

**const** gibt es ebenfalls seit ES6. const verhält sich wie let (block-skopiert), aber der Bezeichner darf nicht neu zugewiesen bzw. überschrieben werden. D.h. der Inhalt wird einmal geladen und ist gültig bis der Server neu gestartet wird. Dies ist beispielsweise bei Mehrwersteuer für eine Land der Fall, da diese sich über lange Zeit nicht verändern.

**Wichtig:** Das bedeutet nicht, dass der gespeicherte Wert "eingefroren" ist – bei Objekten und Arrays lassen sich Inhalte weiterhin ändern.

#### **Beispiel 1 - einfache Varibale**

```
const id = "1234";
console.log("1: ", id); // Ausgabe des Inhalts
id = "3456"; // Versuch die Variable zu überschreiben
console.log("2: ", id); // 1234"
```

Last update: 2025/09/01 13:29

```
scripts > 15 test.js >
             console.log("1: ", id); // 1234
              console.log("2: ", id); // 1234
                                                                                                                                                                                                                              S zsh - scripts + ∨ □ 🖆 ··· | □ ×
                       OUTPUT DEBUG CONSOLE TERMINAL PORTS
  olkandemin@Mac scripts % node test.js
        rrs/volkandemir/Library/CloudStorage/OneDrive-BildungszentrumZürichsee/---- Latest Version ----/MZ88 - 2825HE/JS-Workspace_VS-Code/scripts/
     st.]s:5
= "3456"; // Versuch die Variable zu überschreiben
TypeError: Assignment to constant variable.

at Object.-anonymous> (/Users/volkandemir/Library/CloudStorage/OneDrive-Bildungszentru-Morkspace_VS-Code/scripts/test.js:5:4)
at Module._compile (node:internal/modules/cjs/loader:1688:14)
at Object..js (node:internal/modules/cjs/loader:1820:18)
at Module.load (node:internal/modules/cjs/loader:1423:32)
at Function._load (node:internal/modules/cjs/loader:1246:12)
at TracingChannel.traceSync (node:diagnostics_channel:322:14)
at wrapModuleload (node:internal/modules/cjs/loader:235:24)
at Function.executeUserEntryPoint (as runMain) (node:internal/modules/run_main:171:5)
at node:internal/main/run_main_module:36:49
      de.js v22.18.0
lkandemir@Mac scripts % ■
```

Die Konsolenausgabe bei Markpoint 1: zeigt, dass der ursprüngliche Wert der Variable name 1234 war. Der Versuch den Inhalt zu überschreiben führt zu einer Fehlermeldung. Markpoint 2: zeigt, dass der Inhalt der Variable sich nicht verändert hat.

#### **Beispiel 2 - Komplexe Variable (Objekt/Arra)**

```
const user = { name: "Doe",
               vorname: "Joan"
console.log("1: ", user); // { "name: "Doe", vorname: "Joan" }
user.name = "Muster"; // erlaubt, Objektinhalt änderbar
                       // erlaubt, Objektinhalt änderbar
user.vorname = "Max";
console.log("2: ", user); // { "name: "Doe", vorname: "Joan" }
user = {name: "Suter",
       vorname: "Thierry"
      }; // Fehler: Neuzuweisung nicht erlaubt
console.log("3: ", user); // { "name: "Doe", vorname: "Joan" }
{{:de:modul:m288:learningunits:lu02:lu02_04.png?600|Const mit assoziativen
Arrays}}
```

In der Abbildung ist zu sehen, dass der Versuch das Array neu zu schreiben eine Fehlermeldung herbeiführt. Hingegen kann der Inhalt des Arrays verändert werden.

https://wiki.bzz.ch/ Printed on 2025/11/20 06:35

# Zusatzmaterial

- W3School- JavaScript Varibles
- SelfHTML JS Variablen



From:

https://wiki.bzz.ch/ - BZZ - Modulwiki

Permanent link:

https://wiki.bzz.ch/de/modul/m288/learningunits/lu02/01?rev=1756726186

Last update: 2025/09/01 13:29

