
# JAVASCRIPT 01


# EINFÜHRUNG

# EINFÜHRUNG

- Einsatzzweck: dynamische Internetseiten erstellen
- Vielfältige Möglichkeiten um Aufbau, Layout und Inhalte zu verändern
- Zählt mittlerweile zu den etablierten Web-Technologien
- Aufgrund großer Beliebtheit hat sich Anwendungsbereich ausgeweitet
  - 2009 erschien bspw Node.js => Plattform um mit JavaScript Serveranwendungen programmieren zu können
  - Später auch Entwicklungsumgebungen wie Electron oder NW.js => möglich, Desktop-Anwendungen zu erstellen
- Browser-Anwendungen bleiben mit großem Abstand der häufigste Einsatzbereich von JS
- Läuft innerhalb einer Sandbox, dadurch kein Zugriff auf Funktionen des Betriebssystems, Hardware oder zu Netzwerken möglich

# ENTSTEHUNG

- Entstehung eng mit Entwicklung der Webbrowser verbunden
- Erste Version erschien 1995 unter der Bezeichnung „LiveScript“ und wurde im Netscape Navigator 2.0 umgesetzt
- Wenige Monate nach Ersterscheinung umbenannt in JavaScript (Marketingstrategie)

# Erste Schritte

- Früher musste innerhalb des `script`-Tags das Attribut `type="text/javascript"` stehen => heute nicht mehr nötig
- `alert`-Befehl erzeugt eine kleine Info-Box
- Semikolon beendet JS Befehle

```
<script type="text/javascript">

</script>
```

```
<script>
    alert("Hallo Welt!");
</script>
```

# KOMMENTARE

```
<script>
    // Ich bin ein einzeiliger Kommentar

    /*  Ich bin ein Kommentar
        der über mehrere Zeilen
        gehen kann */
</script>
```

# JS IN EINER EIGENEN DATEI

- HTML Datei

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>JS Seite</title>

    </head>
    <body>

        <script src="meinScript.js"></script>

    </body>
</html>
```

- JS Datei

```
alert('hallo Welt');
```

# „USE STRICT“

- Fehler und schlecht implementierte Funktionen wurden nicht verbessert
  Grund: Entwickler legen viel Wert auf abwärtskompatible Programmiersprache (sonst kann es passieren, dass entsprechende Funktionen plötzlich nicht mehr funktionsfähig sind)
- 2009 neue JavaScript-Version die alle bekannten Schwachstellen entfernen sollte => ECMA SCRIPT 5 (ES5) wurde veröffentlicht => nicht mehr abwärtskompatibel
- Mit `"use strict";` wird dem Browser mitgeteilt, dass er die neue Version verwenden soll, ansonsten wird ES5 vom Browser nicht berücksichtigt
- Befehl muss stets am Anfang eingefügt werden
- Befehl kann nicht rückgängig gemacht werden
- Empfehlenswert neue Programme in dieser moderneren Version zu verfassen

```
<script>
    "use strict";
    alert('hallo Welt!');
</script>
```

# Eine Eingabe des Anwenders aufnehmen

- Einfache Möglichkeit für Interaktion mit Nutzer ist der `prompt`-Befehl
- Erzeugt ein neues Fenster, ähnlich dem `alert`-Befehl

```
<script>
    "use strict";
    prompt("Wie alt bist du?", "Geben Sie das Alter an");
</script>
```

- der `prompt`-Befehl & `alert`-Befehl kombiniert

```
<script>
    "use strict";
    alert("Sie sind " + prompt("Wie alt bist du?", "Geben Sie das Alter an") + " Jahre alt.");
</script>
```

# KLEINE ÜBUNG

Schreibe ein Programm, das den Besucher zum Coding-Kurs begrüßt

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Übungen</title>
    </head>
<body>
    <script>
        "use strict";
        alert("willkommen zum JS-Kurs!");
    </script>
</body>
</html>
```

Frage den Besucher nach seinem Namen und erstelle eine personalisierte Begrüßung

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Übungen</title>
    </head>
<body>
    <script>
        "use strict";
        alert(prompt("Ihr Name:") + ", willkommen zu JavaScript");
    </script>
</body>
</html>
```

# VARIABLEN IN JAVASCRIPT

- Können unterschiedliche Werte aufnehmen (am häufigsten sind es Zahlen)
  Buchstaben, Zeichen, Wörter, längeren Text, Wahrheitswerte, …
- Jede Variable hat einen Typ
- In JS nicht notwendig den Variablentyp anzugeben
  Interpreter wählt während der Ausführung automatisch eine passende Möglichkeit aus
  auch möglich zuerst eine Zahl und dann einen String in die gleiche Variable abzulegen
- Einige Operationen lassen sich aber nur mit bestimmten Variablentypen durchführen (zB.: Division)

```
<script>
    let meineErsteVariable;
</script>
```

- Variable deklarieren => Initialisierung

```
meineErsteVariable = 5;
/* oder */
meineErsteVariable = "Ich bin ein Text";
```

- Der Variable einen Wert zuweisen

```
let meineVariable = 5, text = "Hallo du", zahl = 10;
```

- Mehrere Variablen mit Wert initialisieren
- Nicht empfehlenswert wegen Übersichtlichkeit im Code = schlechter Programmierstil
- Wenn „use strict“ verwendet wird, ist diese Methode nicht möglich

# DER ALERT-BEFEHL

```
<script>
    "use strict";
    let text = "mein erster Text";
    alert(text);
</script>
```

# LET VS VAR

- In einem einfachen Programm gibt es keinen Unterschied
- Funktionsweisen sehr ähnlich
- Üblich, dass eine Variable, die innerhalb einer `if`-Abfrage deklariert wird, nur im Inneren dieses Blocks gültig ist
  im übrigen Programm käme es zu einem Fehler wenn sie aufgerufen wird
- Variablen, die mit `let` deklariert sind, beachten diesen Grundsatz
  mit `var`, sind sie auch außerhalb verwendbar, deshalb sollte man darauf verzichten
- Verhalten innerhalb von Funktionen:

  Geht der Interpreter die gesamte Funktion durch und realisiert alle Variablen-Deklarationen (sofern mit `var` ausgeführt). Danach beginnt Abarbeitung des übrigen Programmcodes.

  Folge: möglich, die Variablen am Anfang der Funktion zu verwenden und sie erst später zu deklarieren => `let` unterstützt das nicht
- Fazit: altere Bezeichnung `var` unterstützt einige Verhaltensweisen, die nicht mehr gängigen Programmiertechniken entsprechen. Bezeichner `let` erfüllt diese Ansprüche

# KONSTANTEN VERWENDEN

- Werte die in der gesamten Verwendung gleich bleiben
- Verwendung kann Performance eines Programms etwas verbessern
- Beispiele:

  Bestimmung von Farben für die Beeinflussung des Designs mit JavaScript

```
<script>
    "use strict";
    const a = "mein erster Text";
    alert(a);
</script>
```

# Datentypen ermitteln

- Mit `typeof` kann ausgegeben werden, um welchen Typ von Variable es sich handelt
- Befehl: `document.write` schreibt Inhalte direkt in das Hauptfenster

```
"use strict";
let ganzeZahl = 3;
let kommazahl = 2.34;
let buchstabe = "a";
let wort = 'hallo';
let wahrheitswert = true;
let ohneInitialisierung;

document.write("ganzeZahl: " + typeof ganzeZahl + "<br>");
document.write("kommazahl: " + typeof kommazahl + "<br>");
document.write("buchstabe: " + typeof buchstabe + "<br>");
document.write("wort: " + typeof wort + "<br>");
document.write("wahrheitswert: " + typeof wahrheitswert + "<br>");
document.write("ohneInitialisierung: " + typeof ohneInitialisierung + "<br>");
```

ganzeZahl: number
kommazahl: number
buchstabe: string
wort: string
wahrheitswert: boolean
ohneInitialisierung: undefined

- Ganze Zahlen und Fließkommazahlen behandelt JS gleich als number
- Typ einer Variable kann sich in JS im laufe des Programms ändern

# DATENTYPEN VERÄNDERN

- Eingabeaufforderung durch prompt-Befehl
- Ausgabe des Datentyp (`string`)
- Ausgabe und versuch einen String mit einer Zahl zu addieren, es kommt zu einer Konkatenaktion
- Ausgabe von Zeilenumbrüchen
- Umwandlung des Eingabewerts in eine Zahl
- Ausgabe und addition der Zahl mit `3` = richtiges Ergebnis

```
let eingabe = prompt("Gib eine Zahl ein:");
document.write("Datentyp: " + typeof eingabe + "<br>");
document.write(eingabe + 3);
document.write("<br><br>");
eingabe = Number(eingabe);
document.write("Datentyp: " + typeof eingabe + "<br>");
document.write(eingabe + 3);
```

Datentyp: string
53

Datentyp: number
8

# Operationen mit Variablen durchführen

```
let meineVariable = 3;
meineVariable = meineVariable * 2;
document.write(meineVariable + "<br>");
meineVariable = 3;
meineVariable *= 2;
document.write("Kurzschreibweise: " + meineVariable + "<br>");
meineVariable = 3;
meineVariable = meineVariable + 5;
document.write(meineVariable + "<br>");
meineVariable = 3;
meineVariable += 5;
document.write("Kurzschreibweise: " + meineVariable + "<br>");
```

6
Kurzschreibweise: 6
8
Kurzschreibweise: 8

| Zeichen | Beschreibung |
|---|---|
| + | Addition |
| - | Subration |
| / | Division |
| * | Multiplikation |
| ** | Potenz |
| % | Modulo |
| += | Erhöhung um den angegebenen Wert |
| -= | Erniedrigung um den angegebenen Wert |
| *= | Variable wird mit dem angegebenen Wert multipliziert |
| /= | Variable wird durch den angegebenen Wert geteilt |
| ++ | Erhöhung um 1 |
| -- | Erniedrigung um 1 |

# ÜBUNGSAUFGABE

Schreibe ein Programm, das drei Variablen unterschiedlichen Typs erstellt. Daraufhin sollen deren Wert und deren Typ mit dem alert-Befehl ausgegeben werden

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Übungen</title>
    </head>
<body>
    <script>
        "use strict";
        let var1 = 4;
        let var2 = "x";
        let var3 = false;

        alert ("Wert: " + var1 + " Typ: " + typeof var1);
        alert ("Wert: " + var2 + " Typ: " + typeof var2);
        alert ("Wert: " + var3 + " Typ: " + typeof var3);
    </script>
</body>
</html>
```

Frage vom Anwender zwei Werte mit dem prompt-Befehl ab. Wandle die Werte dann in Zahlen um und multipliziere sie. Gib das Ergebnis mit `document.write()` aus.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Übungen</title>
    </head>
<body>
    <script>
        "use strict";
        let wert1 = prompt("Wert 1:");
        let wert2 = prompt("Wert 2:");

        wert1 = Number(wert1);
        wert2 = Number(wert2);

        let ergebnis = wert1 + wert2;
        document.write("Ergebnis: " + ergebnis);
    </script>
</body>
</html>
```

# IF-ABFRAGE

```
// Beispiel 1
let bedingung = true;
if(bedingung) {
    alert("Die Bedingung trifft zu.");
}

// Beispiel 2
let eingabe = Number(prompt("Was ist das Ergebnis aus 3 + 2?"));
if(eingabe == 5) {
    document.write("Richtige Lösung!");
}
if(eingabe < 5) {
    document.write("Wert ist zu klein<br>");
} else if(eingabe > 5) {
    document.write("Wert ist zu groß<br>");
}
if(eingabe != 5) {
    document.write("Wert ist falsch!");
}
```

```
// Beispiel 3
"use strict";
let eingabe1 = Number(prompt("Ergebnis aus 2 + 3?"));
let eingabe2 = Number(prompt("Ergebnis aus 2 * 3?"));

if(eingabe1 == 5 || eingabe2 == 6) {
    document.write("Mindestens eine Antwort ist richtig?");
} else {
    document.write("Beide Antworten sind falsch");
}
```

# SWITCH-STATEMENT

```
"use strict";
let artikel = prompt("Welches Produkt suchst du?");
switch(artikel){
    case "Bohrmaschine":
    case "Bandschleifer":
    case "Kreissäge":
        alert("Preis: 34,99 €");
        break;

    case "Schraubenzieher":
        alert("Preis: 2,99 €");
        break;

    case "Hammer":
        alert("Preis: 6,99 €");
        break;

    default:
        alert("Produkt leider nicht gefunden");
}
```

# Übung

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Übungen</title>
    </head>
<body>
    <script>
        "use strict";
        const nutzername = "user1";
        const passwort = "xyz";

        let eingabe1 = prompt("Gib deinen Nutzernamen ein");
        let eingabe2 = prompt("Gib dein Passwort ein");

        if(eingabe1 == nutzername && eingabe2 == passwort) {
            alert("Herzlich Willkommen");
        } else {
            alert("Nutzerdaten nicht korrekt");
        }
    </script>
</body>
</html>
```

Erstelle ein Programm, das zwei Konstanten enthält, in denen ein Nutzername und ein Passwort enthalten sind. Fordere dann den Besucher zur Eingabe der entsprechenden Daten auf. Gib eine passende Meldung aus – je nachdem, ob die eingegebenen Werte richtig oder falsch sind.

# Übung

Stelle dem Anwender 5 Rechenaufgaben. Überprüfe nach jeder Aufgabe, ob das Ergebnis richtig ist. Erhöhe in diesem Fall den Wert der Variablen punkte um 1. Gebe daraufhin je nach Punktestand eine passende Nachricht aus. Verwende dafür ein switch-Statement. Die Werte 0 und 1 sowie 2 und 3 sollen dabei jeweils zusammengefasst werden und zur Ausgabe der gleichen Nachricht führen.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Übungen</title>
    </head>
<body>
    <script>
        "use strict";
        let punkte = 0;
        let eingabe1 = prompt("Was ist das Ergebnis aus 14 + 9");
        if(eingabe1 == 23) {
            punkte++;
        }
        let eingabe2 = prompt("Was ist das Ergebnis aus 27 / 3");
        if(eingabe2 == 8) {
            punkte++;
        }
        let eingabe3 = prompt("Was ist das Ergebnis aus 4 * 8");
        if(eingabe3 == 23) {
            punkte++;
        }
        let eingabe4 = prompt("Was ist die Quadratwurzel von 9");
        if(eingabe4 == 3) {
            punkte++;
        }
        let eingabe5 = prompt("Was ist der Logarithmus von 8 zur Basis 2");
        if(eingabe5 == 3) {
            punkte++;
        }
        switch(punkte) {
            case 0:
            case 1:
                alert("Besuch die Grundschule noch einmal");
                break;
            case 2:
            case 3:
                alert("Wie wärs mit Mathe-Nachhilfe");
                break;
            case 4:
                alert("Fast alles richtig");
                break;
            case 5:
                alert("Maximale Punktezahl");
                break;
        }
    </script>
</body>
</html>
```

# ARRAYS

```
// Variante 1
let meinArray = new Array();
```

- Entspricht Vorgaben für objektorientierte Programmierung (später mehr)

- Arrays sind in JS Objekte mit besonderen Eigenschaften

```
// Variante 2
let  neuesArray = [];
let neuesArray = ["Hund", "Katze", "Maus"];
alert(neuesArray);
```

- Häufiger im Einsatz
- Kann gleich befüllt werden

- Elemente müssen nicht den gleichen Datentyp aufweisen

```
// Produkt, Preis, Lieferbar
let neuesArray = ["Brot", 2.99, true];
alert(neuesArray);
```

# ARRAYS

```
// Ausgabe bestimmte Stelle des Array
let neuesArray =  ["Hund", "Katze", "Maus"];
alert(neuesArray[0]);

// Wert an bestimmter Stelle ändern
neuesArray[0] = "Papagei";

// Wert hinzufügen
neuesArray[3] = "Hund";
// besser um leere Felder oder Überschreibungen zu vermeiden
neuesArray[neuesArray.length] = "Hund";
```

- `length` gibt die Länge des Arrays zurück, da der Index bei `0` anfängt wird so am Ende ein neues Feld hinzugefügt

# Mehrdimensionale Arrays

```
let meinArray = [];
meinArray[0] = ["Brot", 1.99, true];
meinArray[1] = ["Krapfen", 0.99, true];
meinArray[2] = ["Reindling", 6.99, false];
meinArray[3] = ["Linzer Torte", 13.99, true];
meinArray[4] = ["Windbeute", 0.80, false];
// An bestimmte Stelle zugreifen
document.write(meinArray[0][1] + "<br><br>");
document.write(meinArray[2][2] + "<br><br>");
document.write(meinArray[4][0] + "<br><br>");
document.write(meinArray);
```

```
1.99

false

Windbeute

Brot,1.99,true,Krapfen,0.99,true,Reindling,6.99,false,Linzer Torte,13.99,true,Windbeute,0.8,false
```

# Übung

T Schreibe ein Programm, das den Besucher nach seinem Vornamen, nach seinem Nachnamen und nach seinem Alter fragt. Erstelle ein Array und füge die entsprechenden Werte ein. Gebe dessen Inhalt anschließend auf der Seite aus.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Übungen</title>
    </head>
<body>
    <script>
        let meinArray = [];
        meinArray[0] = prompt("Vorname:");
        meinArray[1] = prompt("Nachname:");
        meinArray[2] = prompt("Alter:");

        document.write(meinArray);
    </script>
</body>
</html>
```

# Übung

T Schreibe ein Programm, das den Besucher nach seinem Vornamen, nach seinem Nachnamen und nach seinem Alter fragt. Erstelle ein Array und füge die entsprechenden Werte ein. Gebe dessen Inhalt anschließend auf der Seite aus.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Übungen</title>
    </head>
<body>
    <script>
        let meinArray = [];
        meinArray[0] = prompt("Vorname:");
        meinArray[1] = prompt("Nachname:");
        meinArray[2] = prompt("Alter:");

        document.write(meinArray);
    </script>
</body>
</html>
```

# WHILE-SCHLEIFE

```
"use strict";
let eingabe = Number(prompt(decodeURI("Bis zu welchem Wert m%C3%B6chten sie z%C3%A4hlen?")));
let i = 1;
while(i <= eingabe) {
    document.write(i + "<br>");
    i++;
}
```

- Befehl `decodeURI`
  - Notwendig, da der Text Umlaute enthält
  - Werden sonst im Feld nicht richtig angezeigt

```
// oder
let inhalt = encodeURI("Bis zu welchem Wert möchtest du zählen?")
let eingabe = Number(prompt(decodeURI(inhalt)));
```

# DO-WHILE

```
"use strict";
let eingabe;
do {
    eingabe = prompt("Ergebnis aus 3 + 2?");
} while(eingabe != 5);
alert("Richtige Antwort!");
```

# FOR SCHLEIFE

```
"use strict";
let inhalt = encodeURI("Bis zu welchem Wert möchtest du zählen?");
let eingabe = Number(prompt(decodeURI(inhalt)));

for (let i = 1; i <= eingabe; i++) {
    document.write(i + "<br>");
}
```

# SONDERFORM DER FOR-SCHLEIFE

```
"use strict";
let arr = [2, 5, 9, 4, 2];

for (let wert of arr) {
    document.write(wert + "<br>");
}
```

2
5
9
4
2

# AUFGABE

Erstelle ein Programm, das vom Anwender fünf beliebige Werte abfragt. Schreibe diese in ein Array. Verwende für die Erstellung eine `for`-Schleife. Gebe das Array anschließend aus

```
<script>
    "use strict";
    let arr = [];

    for (let i = 0; i < 5; i++) {
        arr[i] = prompt("Gib einen beliebigen wert ein");
    }
    document.write(arr);
</script>
```

Schreibe ein Programm mit einer identischen Funktion wie in Aufgabe 1. Verwende dieses Mal jedoch eine `while`-Schleife.

```
<script>
    "use strict";
    let arr = [];
    let i = 0;

    while(i < 5) {
        arr[i] = prompt("Gib einen beliebigen Wert ein");
        i++;
    }
    document.write(arr);
</script>
```

# FUNKTION ERSTELLEN UND AUFRUFEN

- Dient dazu, bestimmte Abfolge von Befehlen außerhalb des Hauptprogramms abzuspeichern
- Macht es möglich, sie durch Nennung des Funktionsnamens an einer anderen Stelle aufzurufen

```
"use strict";
function begruessung() {
    let name = prompt("Gib deinen Namen ein:");
    alert("Herzlich willkommen: " + name);
}

begruessung();
```

# GÜLTIGKEITSBEREICH DER VARIABLEN

- Variablen die innerhalb einer Funktion erstellt werden sind auch nur dort gültig
- Variablen außerhalb einer Funktion ist im gesamten Bereich des Programms gültig = globale Variable
- Achtung: Variable innerhalb der Funktion nicht noch einmal deklarieren
- So wenig wie möglich globale Variablen nutzen => ua Fehleranfällig

```
"use strict";
function begruessung() {
    name = prompt("Gib deinen Namen ein:");
    alert("Herzlich willkommen, " + name);
}
let name;
begruessung();
document.write("Ihr Name: " + name);
```

# Funktionen mit Übergabewerten

- Übergabewert steht in der Klammer des Funktionsnamens und wird auch Parameter oder Argument der Funktion bezeichnet
- Beliebiger Datentyp
- Entweder direkt Wert übergeben oder eine Variable

```
"use strict";
function begruessung(name) {
    alert("Herzlich willkommen, " + name);
}
let anwender = prompt("Gib deinen Namen ein:");
begruessung(anwender);
```

- Auch möglich mehrere Werte zu übergeben = mehrere Werte in der Funktionsklammer
- Funktion nimmt Übergabewert in einer Variable auf, deren Namen bereits im Hauptprogramm existiert. Funktion erstellt eine neue Variable mit lokalem Gültigkeitsbereich. Wenn deren Wert verändert wird, wirkt sich das nicht auf die gleichlautende Variable im Hauptprogramm aus

```
"use strict";
function begruessung(name, alter) {
    document.write("Name: " + name + "<br>");
    document.write("Alter: " + alter);
}
let anwender = prompt("Gib deinen Namen ein:");
let alter = prompt("Gib dein Alter ein");
begruessung(anwender, alter);
```

# Funktionen mit Rückgabewerten

- Wert an Hauptprogramm übermitteln aus der Funktion = Rückgabewert
- Jede Funktion darf nur einen einzigen Wert an das Programm zurück geben
- Wenn mehrere Variablen übermittelt werden sollen = jeweils eine eigene Funktion erstellen oder Array erstellen und darin mehrere Werte aufnehmen

```
"use strict";
function beispiel(x) {
    let ergebnis = 2 * x * x + 5 * x + 7;
    return ergebnis;
}
let wert = beispiel(3);
alert(wert);
```

# AUFGABE

Erstelle ein Programm, das eine Funktion enthält. Dieses soll einen Wert vom Anwender erfragen und daraufhin den doppelten Wert auf der Seite ausgeben

```
<script>
    "use strict";
    function verdoppelung() {
        let wert = Number(prompt("Gib einen Wert ein"));
        document.write("Doppelter Wert: " + wert * 2);
    }
    verdoppelung();
</script>
```

Schreibe ein weiteres Programm, das genau die gleiche Aufgabe wie Aufgabe 1 erfüllt. Allerdings sollen die Abfragen des Werts sowie die Ausgabe nun im Hauptprogramm erfolgen. Daher muss die Funktion Übergabe- und Rückgabewert verwenden.

```
<script>
    "use strict";
    function verdoppelung(x) {
        return x * 2;
    }
    let wert = Number(prompt("Gib einen Wert ein"));
    let ergebnis = verdoppelung(wert);
    document.write("Doppelter Wert: " + wert);
</script>
```

# AUFGABE

Erstelle ein Programm, das ein Array mit beliebigen Zahlen enthält. Schreibe eine Funktion, die alle im Array enthaltenen Werte verdoppelt. Die Ergebnisse sollen im ursprünglichen Array abgelegt werden. Verzichte dabei auf die Verwendung globaler Variablen

```
<script>
    "use strict";
    function verdoppelung(arr) {
        for (let i = 0; i <arr.length; i++){
            arr[i] *= 2;
        }
        return arr;
    }
    let array = [3, 5, 7, 15];
    array = verdoppelung(array);
    document.write("Doppelter Wert: " + array);
</script>
```


# ENDE

**QUELLE: JAVASCRIPT**
**PROGRAMMIEREN FÜR EINSTEIGER**
**ISBN: 978-3-96645-016-4**