LU01.A10 - Kommentierung von JS-Code

Lerziele

- Aussagekräftige Kommentare zum Autorenschaft, etc. in JS und HTML einfügen können.
- Sinnvolle Incode-Kommentare in JS und HTML-Code erstellen können.
- Sinn eines Scriptes erkennen und beschreiben können.

Rahmenbedingungen

- Sozialform: individual
- Hilfsmittel:
  - Theorie-Dossier: LU01d - How to comment JS code
- Zeit: 20 Minuten
- Erwartetes Resultat:
  - Die genannten Scripte sind aussagekräftig und prägnant (kurz und bündig) kommentiert.

Ausgangslage

Nachdem wir nun eigenen JS-Code kommentieren können, wollen wir dazu übergehen fremden Code zu lesen und zu kommentieren

Aufgabe

Kommentieren Sie den nachfolgenden Code an den markierten Stellen.

1. Jedes Datei hat einen aussagekräftige „Filekopf“ mit den Daten:
   - Autor, Datum
   - Zielsetzung des Files (wozu braucht es diese Datei?)

Vorlage

```
Grundrechenarten · JS
/**
 * ========================================================================
 *  ARBEITSBLATT: Die Grundrechenarten in JavaScript
 * ========================================================================
 *
 *  Name: _____________________________    Datum: _______________
 *
 *  ANLEITUNG
 *  ---------
 *  Dieses Skript ist vollstaendig lauffaehig. Du musst KEINEN Code
```

```
 *  schreiben - deine Aufgabe ist es, den Code zu VERSTEHEN und zu
 *  KOMMENTIEREN.
 *
 *  Im Skript findest du 10 markierte Stellen:
 *
 *      // --- AUFGABE 1 ------------------------------------------
 *      // TODO: <Frage>
 *      // Antwort:
 *
 *  Schreibe deine Antwort jeweils direkt hinter "// Antwort:" als
 *  Kommentar. Ein bis drei Saetze pro Aufgabe genuegen.
 *
 *  AUSFUEHREN
 *  ----------
 *      node grundrechenarten.js
 *
 *  Tipp: Lass das Skript laufen, BEVOR du kommentierst. Vergleiche die
 *  Ausgabe in der Konsole mit dem Code - vieles erklaert sich dadurch
 *  von selbst.
 * ======================================================================
 */

"use strict";

// ======================================================================
//  TEIL 1 - Zwei Zahlen, vier Rechenarten
// ======================================================================

const a = 12;
const b = 5;

// --- AUFGABE 1 ------------------------------------------------------
// TODO: Erklaere, was die beiden Zeilen oben bewirken. Gehe dabei auf
//       das Schluesselwort "const" ein: Was bedeutet es und worin
//       unterscheidet es sich von "let"?
// Antwort:
//
//

function addieren(x, y) {
  return x + y;
}

function subtrahieren(x, y) {
  return x - y;
}

function multiplizieren(x, y) {
  return x * y;
```

```
}

// --- AUFGABE 2 ----------------------------------------------------
// TODO: Beschreibe den Aufbau einer Funktion am Beispiel von
//       "addieren". Was sind x und y? Was macht "return"?
// Antwort:
//
//

function dividieren(x, y) {
  if (y === 0) {
    return "Division durch 0 ist nicht definiert";
  }
  return x / y;
}

// --- AUFGABE 3 ----------------------------------------------------
// TODO: Warum enthaelt ausgerechnet die Funktion "dividieren" eine
//       zusaetzliche if-Abfrage, die anderen drei aber nicht?
//       Was wuerde JavaScript ohne diese Abfrage bei 12 / 0 ausgeben?
//       (Probiere es aus: console.log(12 / 0);)
// Antwort:
//
//

console.log("=== TEIL 1: Grundrechenarten ===");
console.log(`${a} + ${b} = ${addieren(a, b)}`);
console.log(`${a} - ${b} = ${subtrahieren(a, b)}`);
console.log(`${a} * ${b} = ${multiplizieren(a, b)}`);
console.log(`${a} / ${b} = ${dividieren(a, b)}`);
console.log(`${a} / 0 = ${dividieren(a, 0)}`);

// --- AUFGABE 4 ----------------------------------------------------
// TODO: In den console.log-Zeilen stehen Backticks (`) statt
//       Anfuehrungszeichen und darin ${...}. Wie heisst diese
//       Schreibweise und welchen Vorteil hat sie gegenueber
//       "a + ' + ' + b + ' = ' + addieren(a, b)"?
// Antwort:
//
//

// ========================================================================
//  TEIL 2 - Ganzzahldivision und Rest
// ========================================================================

const zaehler = 17;
const nenner = 5;

const ganzeTeile = Math.floor(zaehler / nenner);
const rest = zaehler % nenner;
```

```
console.log("\n=== TEIL 2: Division mit Rest ===");
console.log(`${zaehler} : ${nenner} = ${ganzeTeile} Rest ${rest}`);

// --- AUFGABE 5 ---------------------------------------------------------
// TODO: Erklaere die beiden Operatoren in den Zeilen oben:
//       a) Was macht Math.floor()?
//       b) Was macht der Operator % (Modulo)?
//       Nenne zu b) ein praktisches Beispiel aus dem Alltag der
//       Programmierung.
// Antwort:
//
//

const zahl = 42;
const istGerade = zahl % 2 === 0;
console.log(`Ist ${zahl} gerade? ${istGerade}`);

// --- AUFGABE 6 ---------------------------------------------------------
// TODO: Die Zeile "const istGerade = zahl % 2 === 0;" enthaelt zwei
//       verschiedene Zeichen mit "=". Erklaere den Unterschied
//       zwischen "=" und "===". Welchen Datentyp hat istGerade?
// Antwort:
//
//

// ========================================================================
//  TEIL 3 - Rangfolge der Operatoren (Punkt vor Strich)
// ========================================================================

const ohneKlammern = 2 + 3 * 4;
const mitKlammern = (2 + 3) * 4;

console.log("\n=== TEIL 3: Rangfolge ===");
console.log(`2 + 3 * 4   = ${ohneKlammern}`);
console.log(`(2 + 3) * 4 = ${mitKlammern}`);

// --- AUFGABE 7 ---------------------------------------------------------
// TODO: Beide Zeilen enthalten dieselben Zahlen und dieselben
//       Rechenzeichen, liefern aber verschiedene Ergebnisse.
//       Erklaere warum. Welche Regel aus dem Mathematikunterricht
//       wendet JavaScript hier an?
// Antwort:
//
//

// ========================================================================
//  TEIL 4 - Wenn Kommazahlen ungenau werden
// ========================================================================
```

```
const summe = 0.1 + 0.2;

console.log("\n=== TEIL 4: Kommazahlen ===");
console.log(`0.1 + 0.2 = ${summe}`);
console.log(`Ist 0.1 + 0.2 === 0.3 ? ${summe === 0.3}`);
console.log(`Gerundet auf 2 Stellen: ${summe.toFixed(2)}`);

// --- AUFGABE 8 ----------------------------------------------------
// TODO: Das Ergebnis von 0.1 + 0.2 ueberrascht. Notiere, was in der
//       Konsole steht, und recherchiere kurz, woran das liegt
//       (Stichwort: Gleitkommazahlen / binaeres Zahlensystem).
//       Wie hilft toFixed(2) weiter - und was ist dabei der Haken?
// Antwort:
//
//

// ========================================================================
//  TEIL 5 - Eine Rechnung, viele Male: die Schleife
// ========================================================================

const reihe = 7;

console.log(`\n=== TEIL 5: Einmaleins der ${reihe} ===`);
for (let i = 1; i <= 10; i++) {
  console.log(`${reihe} * ${i} = ${multiplizieren(reihe, i)}`);
}

// --- AUFGABE 9 ----------------------------------------------------
// TODO: Zerlege den Kopf der for-Schleife in seine drei Bestandteile
//       (getrennt durch Semikolon) und beschreibe jeden einzeln.
//       Wie oft wird der Rumpf der Schleife ausgefuehrt?
// Antwort:
//
//

// ========================================================================
//  TEIL 6 - Eingabe von aussen
// ========================================================================

const eingabe = process.argv[2];
const eingabeZahl = Number(eingabe);

console.log("\n=== TEIL 6: Eingabe ===");
if (eingabe === undefined) {
  console.log("Keine Zahl uebergeben. Starte z. B. mit: node
grundrechenarten.js 8");
} else if (Number.isNaN(eingabeZahl)) {
  console.log(`"${eingabe}" ist keine gueltige Zahl.`);
} else {
  console.log(`Deine Zahl:        ${eingabeZahl}`);
  console.log(`Verdoppelt:        ${multiplizieren(eingabeZahl, 2)}`);
```

```
  console.log(`Halbiert:          ${dividieren(eingabeZahl, 2)}`);
  console.log(`Um 10 vermindert:  ${subtrahieren(eingabeZahl, 10)}`);
}

// --- AUFGABE 10 ------------------------------------------------------
// TODO: Alles, was ueber die Kommandozeile hereinkommt, ist zuerst
//       Text (String). Erklaere, warum Number(eingabe) noetig ist.
//       Was waere das Ergebnis von "8" + 2 ohne diese Umwandlung -
//       und was von "8" * 2? Teste beides.
// Antwort:
//
//
```

Lösungen

LU01.L08

Volkan Demir

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

Permanent link:
https://wiki.bzz.ch/de/modul/m288/learningunits/lu01/aufgaben/10?rev=1788357362

Last update: **2026/09/02 15:56**