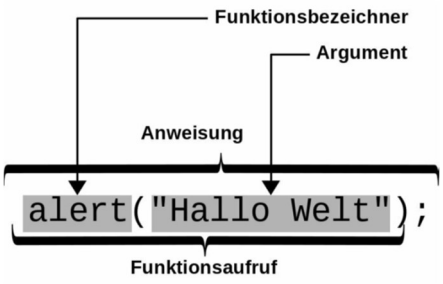
Java-Script

Geschichte:

https://de.wikipedia.org/wiki/JavaScript

https://wiki.selfhtml.org/wiki/JavaScript/Entstehung_und_Standardisierung



alert und confirm = vorgefertigte Funktionen, die Browser mitbringt

alert("Hallo Welt!");

confirm("Bist Du sicher?");

Browser bringt auch console mit = Objekt

auf console steht u.a. Funktion log zur Verfügung

Programmierrichtlinien

- Schreibe immer genau eine Anweisung in genau eine Zeile.
- Beende jede Anweisung mit einem Semikolon.
- "use strict";

Kommentare

// einzeiliger Kommentar

/* TODO: ……*/ mehrzeiliger Kommentar

JS in HTML einbetten

```
<head>
   …
  <script src="highlight_chat_members.js" defer="defer"></script>
</head>
```

oder:

```
<body>
   …
  <script src="highlight_chat_members.js" defer="defer"></script>
</body>
```

Defer & Async

Attribut defer="defer" im script -Element verzögert Ausführung des Scriptes bis DOM der Seite komplett aufgebaut

wenn script im head platziert Attribut notwendig, um Timing-Problem zu beheben – wenn script schneller geladen als DOM-Elemente

Attribut async sorgt auch dafür, dass Browser JS-Datei parallel zu anderen Dateien lädt und Ausführen des HTML-Codes nicht warten muss

Unterschied zu defer: Browser führt JS-Datei sofort aus, sobald sie geladen
gibt keine Garantie, dass DOM bereits aufgebaut
+ Lade- und Ausführungsreihenfolge der JS-Dateien entspricht nicht unbedingt Reihenfolge der script-Tags im HTML-Quellcode (sogenannte Source-Order)
Jedes Script startet sobald geladen - d. h. kleinere Scripte starten meist vor großen

Vorteil von defer in Kombination mit async: Script wird parallel zum Parsen der HTML-Seite geladen und erst nach vollständigem Aufbau des DOM ausgeführt - so muss HTML-Aufbau nicht auf Laden und Ausführen der JS-Datei warten - HTML wird schneller angezeigt

Bezeichner

verwendet aussagekräftige sprechende Bezeichner

(alter statt x, vorname statt z, preis statt p)

immer die Zeit zum Lesen und Verstehen des Codes optimieren — nicht die Zeit zum Erstellen

Bezeichner sollte nicht so lang sein, keine Sonderzeichen oder eigene Abkürzungen enthalten

Variablen und Konstanten in lowerCamelCase schreiben

endeDerDatei statt Endederdatei

anzahlDerBuchstaben (oder buchstabenAnzahl) statt Anzahl der Buchstaben oder anzahlderbuchstaben

Konstanten, die der Konfiguration dienen, in SCREAMING_SNAKE_CASE

const TAX_RATE // Mehrwertsteuersatz

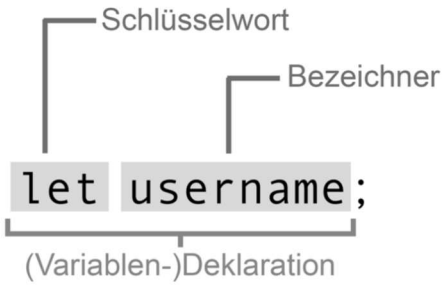
JS-Bezeichner sind case sensitive – unterscheiden in Groß- und Kleinschreibung!
drücke Variablen durch Nomen aus: farbe statt färben
Variablenbezeichner im Singular vergeben: produktName statt produktNamen

Variablen

Variablen erlauben Speichern von Werten und ermöglichen, jederzeit auf Werte zuzugreifen und zu verändern

var firstName; // alte Schreibweise

let firstName; // seit ES6 (ECMAScript 2015)



Zuweisung:
username = "Luke"; // korrekt
"Luke" = username; // falsch

"use strict";
let username;
username = "Luke";
console.log(username);

Mehrere Variablen anlegen

ist möglich, mehrere Variablen gleichzeitig zu deklarieren und zu initialisieren
(mit und/oder ohne Zuweisung)

let username = „Bernd“, userage = 42;

Konstanten

"use strict";
let netPrice = prompt("Net price for the shocking pen?");
let totalPrice = netPrice * 1.19;

console.log("total price:");
console.log(totalPrice);

"use strict";
const TAX_RATE = 1.19;

let netPrice = prompt("Net price for the shocking pen?");
let totalPrice = netPrice * TAX_RATE;

console.log("total price:");
console.log(totalPrice);

Programmierrichtlinien

- Vermeide Magic Numbers
- Schreibe Konstanten, die der Konfiguration dienen, komplett in Großbuchstaben und verwende den Underscore _ zur Worttrennung
- Deklariere Konstanten, die der Konfiguration dienen, am Anfang des Codes

Variablen können Wert ändern
Wert von Konstanten bleibt – einmal festgelegt – konstant

Ausdruck

Ein Ausdruck ist

- ein Literal (Zeichenfolge, zur direkten Darstellung der Werte von Basisdatentypen wie Ganzzahlen, Gleitkommazahlen, Zeichenketten, Wahrheitswerte…)
- eine Variable/Konstante
- eine beliebige syntaktisch-korrekte Kombination aus Literalen, Variablen, Operatoren und Funktionsaufrufen - bsw.: 8 + 5 * x + sum(3,2)

Ausdrücke haben einen Rückgabewert
lassen sich in größere Ausdrücke einbinden

Anweisungen (Variablen-Deklaration, Kontrollstrukturen, Funktionsdeklaration…) lassen sich im Gegensatz zu Ausdrücken nicht in Ausdrücke einbinden

Literal = jeder Wert (String oder Zahl) der direkt wörtlich (engl.: literally) im Code steht