htw saar

Software-Entwicklung mit JavaScript

Übungsblatt 2b

Ablaufsteuerung

23. 10. 2025
Wintersemester 2025 – 2026

Aktualisieren Sie Ihre eigene lokale Repository mit **`git pull origin master`** bevor Sie anfangen, Übungen zu programmieren.

Wenn es nicht anders gefordert wird, erstellen Sie für jede Programmierung-Übung eine `.js`-Datei und eine `.html`-Datei in dem Ordner `example-projects-2025/textbased-programms`. Verwenden Sie die Dateien `hello-world.js` und `hello-world.html` als Vorlage für Ihre Dateien. Der Dateiname sollen mit Ihrem Name versehen, somit wird Namenskonflikt vermieden.

Zum Beispiel: Ein Student namens „Peter“ möchte ein Programm zum Addieren 2 Zahlen schreiben, er kann zwei Dateien

1. `example-projects-2026/textbased-programs/peter-addition.js`
2. `example-projects-2026/textbased-programs/peter-addition.html`

erstellen.

Übung 1.

Das Objekt `terminal` verfügt über die Methode `printh`, die einen String als HTML auf dem Browser ausgibt. Z.B

```
terminal.printh('<span style="color: red">Wichtig!</span>');
```

gibt den Text „Wichtig“ in Rot auf dem Browser.

Schreiben Sie ein Programm, die eine zufällige ganze Zahl von 1 bis einschließlich 6 generiert. Für jede Zahl gibt das Programm ein entsprechendes Bild aus, zum Beispiel ein Bild von dem Würfel mit der entsprechenden Seite.

Check-Liste bevor Sie anfangen zu programmieren:

- Untersuchen Sie den HTML Tag „img“ (`https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img`).

- Erstellen Sie für Ihr Programm ein Ordner **`dice-simulation/assert`**, wo Sie alle Bilder dort speichern.

Mit welche Methode vom Objekt **`terminal`** kann man den Ausgabe-Bereich zurücksetzen / leeren?

Übung 2. *Großkreis*

Schreiben Sie ein Programm, das 4 Werten x_1, y_1, x_2, y_2 als Breitengrad und Längengrad in Grad vom zweit Punkten $P_1(x_1; y_1)$, $P_2(x_2; y_2)$ auf der Erdoberfläche einliest und gibt die *Großkreis* Distanz von beiden aus. Die Distanz ist gegeben als:

$$d = 60 \cdot \arccos(\sin(x_1) \cdot \sin(x_2) + \cos(x_1) \cdot \cos(x_2) \cdot \cos(y_1 - y_2))$$

Die Gleichung berechnet die Distanz in Seemeilen und die Winkel sind in Grad. Die Funktion **`sin`** und **`cos`** im Objekt **`Math`** arbeiten mit Radiant. Lesen Sie die Dokumentation vom Objekt **`Math`** um Grad in Radiant umzurechnen.

Benutzen Sie Ihr Programm um die Distanz von Paris (48,87° N; -2,33° W) nach San Francisco (37,8° N; 122,4° W) zu berechnen.

Warum rechnet man die Distanz in diese Gleichung in Seemeile?

Neben Frage: Kann man mit Google Map die Genauigkeit des Programms testen?

Nützliche Links:

- Großkreis **`https://de.wikipedia.org/wiki/Gro%C3%9Fkreis`**
- Plausibilität Checktool **`https://www.cactus2000.de/uk/unit/massgrk.shtml`**
- Seemeile **`https://de.wikipedia.org/wiki/Seemeile`**

Übung 3. *Zahl Konvertierung*

Ein anderes Verfahren um eine Zahl von 10er-Basis zu Binär-Basis zu konvertieren ist, die Zahl durch zwei zu dividieren, und die Quotient an der Zahl zuzuweisen, solang der Quotient größer als 0 ist. Die Reste der Division-Operationen bilden die Binäre-Darstellung der Zahl rückwärts. Die Abbildung 1 veranschaulicht dieses Verfahren.

Schreiben Sie ein JavaScript Programm, das dieses Verfahren implementiert.

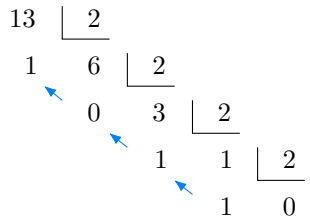

Abbildung 1: Konvertierung der Zahle 13 in Binär-Schreibweise: $13 = 1101_2$

Übung 4. *Luhn-Algorithmus*

Der Luhn-Algorithmus wird verwendet, um Id-Nummer zu validieren. Er wurde entworfen, um die Fehler aus Versehen zu entdecken. Der Algorithmus ist einfach.

1. Iteriere die Zahl von rechts nach links (von kleinsten Stelle nach größten Stelle) und bilde die Summe der Ziffer wie folgt:
   a) Ist der Stelle eine ungerade Stelle ($10^0, 10^2, 10^4 \ldots$ Stellen), addiere den Ziffer an der Stelle auf der Summe;
   b) Ist der Stelle eine gerade Stelle ($10^1, 10^3, 10^5 \ldots$ Stellen), verdoppele den Ziffer an der Stelle. Ist der verdoppelte Wert größer als 9, subtrahiere den Wert 9. Addiere den Wert auf Summe.
2. Wenn der letzte Ziffer der Summe eine Null ist, ist der Zahl gültig, sonst nicht.

Als Beispiel berechnen wir den Prüfsumme von 371-4496-3539-8431:

Die Summe ist 1 + 6 + 4 + 7 + 9 + 6 + 5 + 6 + 6 + 9 + 4 + 8 + 1 + 5 + 3 = 80, also die Id-Nummer ist nach Luhn-Algorithmus gültig (Abbildung 2).

Schreiben Sie ein Programm, das eine Zahl entgegennimmt (als String, ohne jeglichen Trennzeichen) und diese Zahle nach Luhn-Algorithmus validiert. Das Programm soll den Datentyp `BigInt` benutzen um die Zahl darzustellen. Verwenden Sie keinen Datentyp String. Die Lösung in `https://de.wikipedia.org/wiki/Luhn-Algorithmus` verwendet String, und deshalb ist für diese Aufgabe disqualifiziert.

Der Luhn-Algorithmus wird auch verwendet, um die Kredit-Nummer syntaktisch zu validieren. Testen Sie Ihr Programm gegenüber den Kredit-Nummer in der Datei `valid-credits.txt`. Sie stammen aus PayPal API Test.

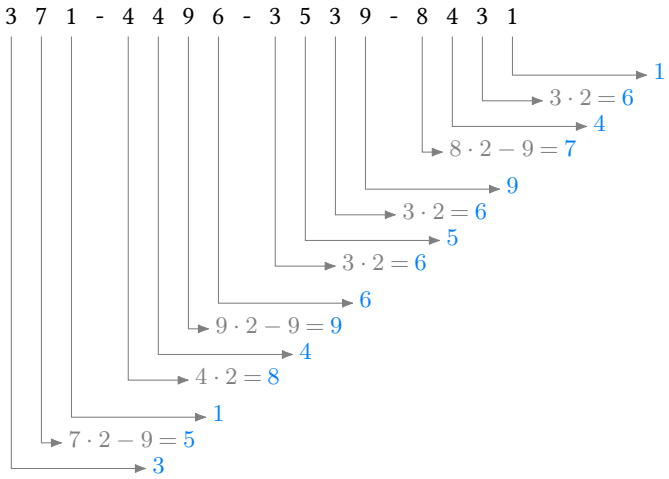

Abbildung 2: Schema zur Berechnung der Luhn-Checksumme

Übung 5. *Eingabe Toleranz*

Bei der Eingabe von lange Zahlen, zum Beispiel die Kredit-Nummer, möchte man als Benutzer auch Trennzeichen zischen Ziffer hinzufügen, damit die Eingabe besser kontrolliert werden kann. Lesen Sie die Dokumentation der String API in JavaScript, um Trennzeichen in der Eingabe heraus zu filtern, bevor Sie die Eingabe in `BigInt` konvertiert (`String.replace()`).

Aus der Benutzersicht möchte man die Zahl mit einem Trennzeichen nach jedem dritten Ziffer eingeben. Warum funktioniert diese Eingabe nicht auf Anhieb wenn der Benutzer das Leerzeichen als Trennzeichen verwendet? Wie kann der Benutzer das Leerzeichen als Trennzeichen verwenden? Lesen Sie die Dokumentation von der Funktion `runMain` in der Datei `dfhi.js`.

Übung 6.

Der `for-of`-Schleife iteriert die Elementen in einem Array. Z.B. kann man folgenden Konstruktion benutzen, um die einzelnen Element in der Eingabe-Feld (das ist der Parameter `...argv` in der Funktion `window.main`) wieder auf dem Browser auszugeben.

```
for(let e of argv) {
    terminal.printl(e);
}
```

Schreiben Sie ein Programm, das die Wahlergebnis jeweiligen Partei visualisiert. Der Benutzer gibt die Anzahl der Stimme von einer Partei in der Eingabe-Feld ein. Das Programm stellt diesen Werten anteilig in einer textuellen Zeile dar. Zum Beispiel:

Man kann –in der Abhängigkeit vom Bildschirmgröße– etwa 60 Zeichen in eine Zeile auf dem Browser darstellen. Für die Eingabe

```
30 60 20
```

wird folgenden Zeile ausgegeben:

```
oooooooooooooo**********************************xxxxxxxxxx
```

Die Anzahl der Zeichen in den ausgegebenen Text sind:

$$c_{\texttt{o}} = 60 \cdot \left\lfloor \frac{30}{30+60+20} \right\rceil = 16$$
$$c_{\texttt{*}} = 60 \cdot \left\lfloor \frac{60}{30+60+20} \right\rceil = 33$$
$$c_{\texttt{x}} = 60 \cdot \left\lfloor \frac{20}{30+60+20} \right\rceil = 11$$

Die Annotation $\lfloor x \rceil$ bedeutet, dass der Wert von x zu der nächsten Ganzzahl gerundet wird.

Bestimmen Sie zuerst, wie viel Zeichen können Sie auf Ihrem Browser darstellen. Wir gehen davon aus, dass es nur Stimmen für maximal 5 Parteien gibt. Wählen Sie deshalb 5 Zeichen, jedes für jeweilige Partei.