Aufgabe 1 (Felder)

1. Befüllen Sie ein Array mit *n* zufälligen ganzen Zahlen[1] aus einem selbstgewählten Intervall $[a, b] \subset \mathbb{Z}$, wobei das Array selbst, einschließlich der Parameter *a*, *b* und *n*, einer Funktion als Argumente übergeben werden sollen. Der Rückgabewert der Funktion ist das befüllte Array.
2. Schreiben Sie nun eine Funktion *alleAusgeben*(), bei der nach jeweils zehn Elementen ein Zeilenvorschub ausgeben wird.
3. Erstellen Sie eine Funktion *summe*(), welche die Summe aller Elemente des als Argument übergebenen Arrays berechnet und zurückgibt.
4. Erstellen Sie eine Funktion *ersetzeZahl*(index, wert), die die Zahl mit dem Index „index“ durch die übergebene neue Zahl „wert“ ersetzt. Überprüfen Sie zuerst, ob der übergebene Index überhaupt erlaubt ist; wenn nicht, soll die Funktion eine Fehlermeldung auf der Konsole ausgeben.
5. Erstellen Sie eine Funktion *groessteZahl*(), welche die größte Zahl zurückgibt, die im Array gespeichert ist. Überlegen und implementieren Sie außerdem ein sinnvolles und robustes Verhalten für den Fall, dass das Feld leer ist.
6. In der letzten Übung haben Sie in Aufgabe 1.6 den aktuellen Wochentag über eine ‚switch'-Anweisung ausgegeben. Nutzen Sie nun zur Ausgabe des Tages *getDay()* als Index in ein Array.
7. Zeigen Sie an einem Beispiel, wie mehrdimensionale Arrays erzeugt und als Return-Wert einer JavaScript-Funktion zurückgeliefert werden können.

Aufgabe 2 (*optionale Zusatzaufgabe*)

Implementieren Sie einen einfachen zellulären Automaten, bei dem das Verhalten in einer eindimensionalen Welt simuliert und dargestellt werden soll. Besagte Welt besteht dabei aus endlich vielen aneinandergereihten Zellen, die entweder leer (0) oder voll (1) sind.

X		X		X	X

Pro Simulationsschritt wird dargestellt, wie sich die Welt weiterentwickelt. Je nachdem, wie viele Nachbarn (bzw. besser gesagt Vorgänger) existieren, ist die zeitlich nachfolgende Zelle voll oder leer. Für den Simulationsverlauf ist die Anzahl der Nachbarn wichtig: Eine Zelle kann 0 bis 5 volle Nachbarn haben (z.B. hat Zelle C vier Vorgänger).

X	X	X		X	
		C			

Die „Welt“ wird zeilenweise so dargestellt, dass pro Zeitschritt eine Zeile verwendet wird. Die Anzahl der dargestellten Simulationsschritte soll frei wählbar sein. Stellen Sie dabei die beiden Zustände folgendermaßen dar: Leerzeichen " " für 0 und Buchstabe "+" für 1. Verwenden Sie für die Ausgabe wieder das <pre>-Element, damit das Muster richtig dargestellt wird.
Das Anfangsverhalten ergibt sich dabei aus einer zufällig generierten Startzeile. Erzeugen Sie die jeweils nächste Zeile nach folgender Regel: Hat eine Zelle 2 oder 4 (volle) Vorgänger, so wird sie auf Zustand 1 gesetzt, ansonsten auf Zustand 0. Für diese Berechnung wird eine Zeile zudem als geschlossen (also „ringförmig“ verbunden) angenommen.

[1] Nutzen Sie dazu die Methode *random*() des Math-Objekts: http://www.w3schools.com/jsref/jsref_obj_math.asp