JavaScript Aufgaben: Funktionen, Schleifen, Arrays und Objekte

Funktionen

1. **Einfacher Funktionsaufruf:**
   Schreibe eine Funktion `greet`, die den Namen als Parameter annimmt und `"Hello, [name]!"` in die Konsole schreibt. Teste die Funktion mit verschiedenen Namen.

   ```
   const greet = function(name) {
   console.log(`Hello, ${name}!`);
   };
   greet("John");  // Ausgabe: Hello, John!
   ```

2. **Funktion mit Rückgabewert:**
   Schreibe eine Funktion `add`, die zwei Zahlen als Parameter nimmt und deren Summe zurückgibt. Logge das Ergebnis.

   ```
   const add = function(a, b) {
       return a + b;
   };

   console.log(add(3, 5));  // Ausgabe: 8
   ```

3. **Funktion mit Standardwerten:**
   Schreibe eine Funktion `multiply`, die zwei Zahlen multipliziert. Wenn der zweite Parameter nicht angegeben ist, soll der Standardwert 1 verwendet werden.

   ```
   const multiply = function(a, b = 1) {
       return a * b;
   };

   console.log(multiply(5));  // Ausgabe: 5
   ```

4. **Funktion mit Array als Argument:**
   Schreibe eine Funktion `sumArray`, die ein Array von Zahlen als Parameter nimmt und die Summe der Zahlen zurückgibt.

   ```
   const numbers = [2, 4, 6, 8];
   const sumArray = function(arr) {
       let sum = 0;
       for (let num of arr) {
           sum += num;
       }
       return sum;
   };

   console.log(sumArray(numbers));  // Ausgabe: 20
   ```

5. **Funktion, die Objekte als Argumente verwendet:**
   Schreibe eine Funktion `getFullName`, die ein Objekt mit `firstName` und `lastName` als Parameter nimmt und den vollständigen Namen zurückgibt.

   ```
   const person = {
       firstName: "Jane",
       lastName: "Doe"
   };

   const getFullName = function(obj) {
       return `${obj.firstName} ${obj.lastName}`;
   };

   console.log(getFullName(person));  // Ausgabe: Jane Doe
   ```

Schleifen

6. **for-Schleife über ein Array:**
   Schreibe eine Funktion `printArray`, die jedes Element eines Arrays mit einer for-Schleife in der Konsole ausgibt.

```
const colors = ["red", "green", "blue"];
const printArray = function(arr) {
    for (let color of arr) {
        console.log(color);
    }
};

printArray(colors);
```

7. **while-Schleife mit Abbruchbedingung:**
   Schreibe eine Funktion `countDown`, die von einer gegebenen Zahl bis null herunterzählt und dabei jedes Mal die Zahl in der Konsole ausgibt.

```
const countDown = function(number) {
    while (number >= 0) {
        console.log(number);
        number--;
    }
};

countDown(5);  // Ausgabe: 5, 4, 3, 2, 1, 0
```

8. **do-while-Schleife:**
   Schreibe eine Funktion `askForPassword`, die in einer do-while-Schleife wiederholt das Passwort abfragt, bis das richtige Passwort eingegeben wird.

```
const askForPassword = function() {
    let password;
    do {
        password = prompt("Enter password:");
    } while (password !== "secret");
    console.log("Access granted");
};

// askForPassword();  // Nur in Browser-Umgebung ausführbar
```

9. **Verschachtelte for-Schleifen:**
   Schreibe eine Funktion `printMultiplicationTable`, die die Multiplikationstabelle von 1 bis 10 mit verschachtelten for-Schleifen in der Konsole ausgibt.

```
const printMultiplicationTable = function() {
    for (let i = 1; i <= 10; i++) {
        for (let j = 1; j <= 10; j++) {
            console.log(`${i} x ${j} = ${i * j}`);
        }
    }
};

printMultiplicationTable();
```

10. **for-Schleife mit Bedingung:**
    Schreibe eine Funktion `printEvenNumbers`, die nur die geraden Zahlen in einem gegebenen Array mit einer for-Schleife ausgibt.

```
const nums = [1, 2, 3, 4, 5, 6];
const printEvenNumbers = function(arr) {
    for (let num of arr) {
        if (num % 2 === 0) {
            console.log(num);
        }
    }
};

printEvenNumbers(nums);
```

Arrays und Schleifen

11. **Summe von Array-Elementen:**
    Schreibe eine Funktion `sumNumbers`, die ein Array von Zahlen als Parameter annimmt und die Summe der Zahlen berechnet. Verwende eine for-Schleife.

```
const numbers = [1, 2, 3, 4, 5];
const sumNumbers = function(arr) {
    let sum = 0;
    for (let num of arr) {
        sum += num;
    }
    return sum;
};

console.log(sumNumbers(numbers));  // Ausgabe: 15
```

12. **Durchschnitt von Array-Elementen:**
    Schreibe eine Funktion `averageNumbers`, die ein Array von Zahlen als Parameter annimmt und den Durchschnitt berechnet.

```
const numbers = [10, 20, 30, 40, 50];
const averageNumbers = function(arr) {
    let sum = 0;
    for (let num of arr) {
        sum += num;
    }
    return sum / arr.length;
};

console.log(averageNumbers(numbers));  // Ausgabe: 30
```

13. **Maximalwert eines Arrays finden:**
    Schreibe eine Funktion `findMax`, die das größte Element in einem Array von Zahlen findet.

```
const numbers = [10, 5, 8, 12, 3];
const findMax = function(arr) {
    let max = arr[0];
    for (let num of arr) {
        if (num > max) {
            max = num;
        }
    }
    return max;
};

console.log(findMax(numbers));  // Ausgabe: 12
```

14. **Zählen von Vorkommen in einem Array:**
    Schreibe eine Funktion `countOccurrences`, die zählt, wie oft ein bestimmtes Element in einem Array vorkommt.

```
const fruits = ["apple", "banana", "apple", "orange", "banana", "apple"];
const countOccurrences = function(arr, value) {
    let count = 0;
    for (let item of arr) {
        if (item === value) {
            count++;
        }
    }
    return count;
};

console.log(countOccurrences(fruits, "apple"));  // Ausgabe: 3
```

15. **Array umkehren:**
    Schreibe eine Funktion `reverseArray`, die die Elemente eines Arrays in umgekehrter Reihenfolge zurückgibt.

```
function reverseArray(arr) {
    return arr.reverse();
}

const numbers = [1, 2, 3, 4, 5];
console.log(reverseArray(numbers));  // Ausgabe: [5, 4, 3, 2, 1]
```

Funktionen und Arrays kombinieren

16. **Funktion zur Manipulation von Array-Elementen:**
    Schreibe eine Funktion `doubleNumbers`, die jedes Element eines Arrays verdoppelt und das neue Array zurückgibt.

```
function doubleNumbers(arr) {
    return arr.map(num => num * 2);
}

const numbers = [1, 2, 3, 4];
console.log(doubleNumbers(numbers));  // Ausgabe: [2, 4, 6, 8]
```

17. **Funktion, die ein Array filtert:**
Schreibe eine Funktion `filterEvenNumbers`, die nur die geraden Zahlen aus einem gegebenen Array zurückgibt.

```
function filterEvenNumbers(arr) {
    return arr.filter(num => num % 2 === 0);
}

const numbers = [1, 2, 3, 4, 5, 6];
console.log(filterEvenNumbers(numbers));  // Ausgabe: [2, 4, 6]
```

18. **Funktion, die Array-Elemente summiert, wenn sie eine Bedingung erfüllen:**
Schreibe eine Funktion `sumOddNumbers`, die die Summe aller ungeraden Zahlen in einem Array berechnet.

```
function sumOddNumbers(arr) {
    return arr.reduce((sum, num) => num % 2 !== 0 ? sum + num : sum, 0);
}

const numbers = [1, 2, 3, 4, 5];
console.log(sumOddNumbers(numbers));  // Ausgabe: 9
```

19. **Funktion, die prüft, ob ein Wert in einem Array enthalten ist:**
Schreibe eine Funktion `containsValue`, die prüft, ob ein bestimmter Wert in einem Array enthalten ist.

```
function containsValue(arr, value) {
    return arr.includes(value);
}

const fruits = ["apple", "banana", "orange"];
console.log(containsValue(fruits, "banana"));  // Ausgabe: true
```

Objekte und Arrays

20. **Objekt aus Array von Werten erstellen:**
Schreibe eine Funktion `createObjectFromArray`, die ein Array von Werten in ein Objekt umwandelt, wobei die Array-Indizes als Schlüssel verwendet werden.

```
function createObjectFromArray(arr) {
    return arr.reduce((obj, value, index) => {
        obj[index] = value;
        return obj;
    }, {});
}

const values = ["apple", "banana", "orange"];
console.log(createObjectFromArray(values));
// Ausgabe: {0: "apple", 1: "banana", 2: "orange"}
```

21. **Funktion, die Eigenschaften eines Objekts in ein Array umwandelt:**
Schreibe eine Funktion `objectKeysToArray`, die die Schlüssel eines Objekts in ein Array umwandelt.

```
function objectKeysToArray(obj) {
    return Object.keys(obj);
}

const person = {name: "John", age: 30, city: "Berlin"};
console.log(objectKeysToArray(person));  // Ausgabe: ["name", "age", "city"]
```

22. **Funktion, die alle Werte eines Objekts summiert:**
Schreibe eine Funktion `sumObjectValues`, die die Werte aller numerischen Eigenschaften eines Objekts summiert.

```
function sumObjectValues(obj) {
    return Object.values(obj).reduce((sum, value) => sum + (typeof value === "number" ? value : 0), 0);
}

const data = {a: 10, b: 20, c: "hello"};
console.log(sumObjectValues(data));  // Ausgabe: 30
```

23. **Objekt aus zwei Arrays erstellen:**
Schreibe eine Funktion `createObjectFromTwoArrays`, die zwei Arrays (eines mit Schlüsseln und eines mit Werten) in ein Objekt umwandelt.

```
function createObjectFromTwoArrays(keys, values) {
    return keys.reduce((obj, key, index) => {
        obj[key] = values[index];
        return obj;
    }, {});
}

const keys = ["name", "age", "city"];
const values = ["John", 30, "Berlin"];
console.log(createObjectFromTwoArrays(keys, values));
// Ausgabe: {name: "John", age: 30, city: "Berlin"}
```

24. **Objekte innerhalb eines Arrays summieren:**
Schreibe eine Funktion `sumPropertyInObjects`, die ein Array von Objekten nimmt und die Summe einer bestimmten Eigenschaft berechnet (z.B. `price` in einem Array von Produkten).

```
function sumPropertyInObjects(arr, prop) {
    return arr.reduce((sum, obj) => sum + (obj[prop] || 0), 0);
}

const products = [{name: "Product 1", price: 10}, {name: "Product 2", price: 20}, {name: "Product 3", price: 30}];
console.log(sumPropertyInObjects(products, "price"));  // Ausgabe: 60
```

Schleifen und Bedingungen kombinieren

25. **Funktion mit einer for-Schleife und if-Statement:**
Schreibe eine Funktion `countEvensAndOdds`, die die Anzahl der geraden und ungeraden Zahlen in einem Array zählt und das Ergebnis in einem Objekt zurückgibt.

```
function countEvensAndOdds(arr) {
    let result = {evens: 0, odds: 0};
    for (let num of arr) {
        if (num % 2 === 0) {
            result.evens++;
        } else {
            result.odds++;
        }
    }
    return result;
}

const numbers = [1, 2, 3, 4, 5, 6];
console.log(countEvensAndOdds(numbers));  // Ausgabe: {evens: 3, odds: 3}
```

26. **Verschachtelte Schleifen und Bedingungen:**
Schreibe eine Funktion `findDuplicates`, die doppelte Elemente in einem Array findet und sie in einem neuen Array speichert.

```
function findDuplicates(arr) {
    let duplicates = [];
    for (let i = 0; i < arr.length; i++) {
        for (let j = i + 1; j < arr.length; j++) {
            if (arr[i] === arr[j] && !duplicates.includes(arr[i])) {
                duplicates.push(arr[i]);
            }
        }
    }
    return duplicates;
}

const numbers = [1, 2, 3, 4, 2, 5, 3];
console.log(findDuplicates(numbers));  // Ausgabe: [2, 3]
```

27. **Schleifen mit Array von Objekten:**

Schreibe eine Funktion `printAllProducts`, die alle Produkte in einem Array von Produktobjekten in der Konsole anzeigt. Jedes Produkt hat `name` und `price` als Eigenschaften.

```
function printAllProducts(products) {
    for (let product of products) {
        console.log(`Product: ${product.name}, Price: ${product.price}`);
    }
}

const products = [{name: "Product 1", price: 10}, {name: "Product 2", price: 20}, {name: "Product 3", price: 30}];
printAllProducts(products);
// Ausgabe:
// Product: Product 1, Price: 10
// Product: Product 2, Price: 20
// Product: Product 3, Price: 30
```

Fortgeschrittenere Aufgaben (mittleres Niveau)

28. **Funktion, die ein 2D-Array verarbeitet:**
Schreibe eine Funktion `sum2DArray`, die ein zweidimensionales Array als Parameter nimmt und die Summe aller Elemente berechnet.

```
function sum2DArray(arr) {
    return arr.reduce((sum, subArr) => sum + subArr.reduce((subSum, num) => subSum + num, 0), 0);
}

const array2D = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];
console.log(sum2DArray(array2D));  // Ausgabe: 45
```

29. **Verschachtelte Schleifen für Paare von Array-Elementen:**
Schreibe eine Funktion `findAllPairs`, die alle Paare von Elementen in einem Array (z.B. [1, 2], [1, 3], [2, 3] usw.) in der Konsole anzeigt. Verwende verschachtelte for-Schleifen.

```
function findAllPairs(arr) {
    for (let i = 0; i < arr.length; i++) {
        for (let j = i + 1; j < arr.length; j++) {
            console.log(`[${arr[i]}, ${arr[j]}]`);
        }
    }
}

const numbers = [1, 2, 3, 4];
findAllPairs(numbers);
// Ausgabe:
// [1, 2]
// [1, 3]
// [1, 4]
// [2, 3
```

30. **Funktion, die Objekte auf Basis einer Bedingung filtert:**
Schreibe eine Funktion `filterProductsByPrice`, die ein Array von Produktobjekten und einen Maximalpreis als Parameter nimmt. Die Funktion gibt nur die Produkte zurück, deren Preis unter dem Maximalpreis liegt.

```
function filterProductsByPrice(products, maxPrice) {
    return products.filter(product => product.price < maxPrice);
}

const products = [
    {name: "Product 1", price: 50},
    {name: "Product 2", price: 100},
    {name: "Product 3", price: 150},
    {name: "Product 4", price: 200}
];

console.log(filterProductsByPrice(products, 150));
// Ausgabe:
// [
//   {name: "Product 1", price: 50},
//   {name: "Product 2", price: 100}
// ]
```