JavaScript exercises - solutions

1. Lesson

Create a function which expects one string parameter then iterates backward through the string, and at the end it returns the reversed string.
Example: if the input is '**apple**' then the output should be '**elppa**'

```
const reverseString = (s) => {
 let result = '';
 for (let i = s.length - 1; i >= 0; i--) {
   result = result + s.charAt(i);
 }

 return result;
};

console.log(reverseString('apple'));
```

2. Lesson

Create a function which expects one string and two numbers (indexes) as parameters. The function has to take a substring (the string between the indexes) of the original string, then concatenate this substring twice after each other.
Example: if the inputs are **'apple', 0, 3** then the output should be '**appapp**'
Hint: In JavaScript (NaN === NaN) comparison gives `false`, so when we want to check if a number is NaN then we should use the isNaN built-in method.

```
const f2 = (s, i1, i2) => {
 const slice = s.slice(i1, i2);
 return slice + slice;
}

console.log(f2('apple', 0, 3));
```

3. Lesson

Create a function which expects an array of numbers parameter then returns with an array numbers which contains only the **even** numbers from the original array.
Example: if the input is **[1, 2, 3, 4, 5]** then the output should be **[2, 4]**

```
const selectEven = (array) => {
 const result = [];
 array.forEach(i => {
   if (i % 2 === 0) {
     result.push(i);
   }
 });

 return result;
}

console.log(selectEven([1, 2, 3, 4, 5]));
```

4. Lesson

Create a function which expects one string parameter then returns with an array of numbers with the numbers found in the input string.
Example: if the input is '**ap1ple93**' then the output should be **[1, 9, 3]**

```
const parseNumbers = (s) => {
 const result = [];
 for(let i = 0; i < s.length; i++) {
   const n = Number(s.charAt(i));
   if (!isNaN(n)) {
     result.push(n);
   }
 }

 return result;
}

console.log(parseNumbers('ap1ple93'));
```

5. Lesson

Create a function which expects one number parameter (x) then returns with a function which expects one number parameter (y) — this second function should multiply **y** by **x** and returns the result (this is the closure construction).
Example: if the inputs are **x=10** and **y=5** then the output should be **50**

```
const factory = (x) => {
 return (y) => x*y;
}

const f = factory(10);

console.log(f(5)); // 50
```

6. Lesson

Create a function which expects one number parameter (between 0 and 100) then returns with the grade associated to that number.
The grading table is:

- 0-50% - 1
- 51-60% - 2
- 61-70% - 3
- 71-85% - 4
- 86-100% - 5

Example: if the input is **70** then the output should be **3**

```
const grade = (result) => {
 let grade;

 if (result <= 50) {
   grade = 1;
 } else if (result <= 60) {
   grade = 2;
 } else if (result <= 70) {
   grade = 3
 } else if (result <= 85) {
   grade = 4
 } else {
   grade = 5;
 }

 return grade;
}

console.log(grade(45)); // 1
console.log(grade(70)); // 3
console.log(grade(90)); // 5
```

7. Lesson

Create a function which expects one number parameter then generates a random number between 1 and the given parameter, then it returns with this generated number.
Hint: MDN documentation of how to generate a random number in JavaScript: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random

```
const random = (max) => Math.floor(Math.random() * max)

console.log(random(10));
console.log(random(10));
console.log(random(10));
```

8. Lesson

Extend the previous lesson with error handling: if the given parameter is 0 or negative number then the function should throw an error.
Example: if the input is **-5** then the output should be **'Not valid input' error**

```
const random = (max) => {
 if (max <= 0) {
   throw new Error('Not valid input');
 }
 return Math.floor(Math.random() * max);
}

try {
 console.log(random(100));
 console.log(random(5));
 console.log(random(-1));
} catch (e) {
 console.log('Invalid input, please provide a positive integer
number');
}
```

9. Lesson

Create a function which expects one object parameter with two keys (**`firstName`** and **`lastName`**) then concatenate the values of these keys into a new string, but there should be a whitespace separator between the values. Then add this newly created string into the original object's **`name`** field, then remove the **`firstName`** and **`lastName`** fields from the object.
Example: if the input is **{firstName: 'John', lastName: 'Doe'}** then the output should be **{name: 'John Doe'}**

```
const transform = (input) => {
 const name = `${input.firstName} ${input.lastName}`;
 input.name = name;
 delete input.firstName;
 delete input.lastName;
};

const o = {firstName: 'John', lastName: 'Doe'};
transform(o);
console.log(o);
```

10.Lesson

Create a function which expects one string and one number as parameters then concatenate the input string after each other as many times as the number parameter defines.
Example: if the input is **'apple', 3** then the output should be **'appleappleapple'**

```
const repeat = (input, count) => {
 let result = '';

 for(let i = 0; i < count; i++) {
   result = result + input;
 }

 return result;
}

console.log(repeat('apple', 3));
```