14. Error Handling, Modules, Map & Filters and ES6 Concepts

Table of Content

1. Error Handling

In programming, there can be two types of errors in the code:

1. Syntax Error
2. Runtime Error

These errors that occur during runtime are called exceptions.

▼ **JavaScript try...catch Statement**

```
try {
    // body of try
}
catch(error) {
    // body of catch
}
```

The main code is inside the `try` block. While executing the `try` block, if any error occurs, it goes to the `catch` block. The `catch` block handles the errors as per the catch statements.

If no error occurs, the code inside the `try` block is executed and the `catch` block is skipped.

▼ **JavaScript try...catch...finally Statement**

You can also use the `try...catch...finally` statement to handle exceptions.
The `finally` block executes both when the code runs successfully or if an error occurs.

The syntax of `try...catch...finally` block is:

```
try {
    // try_statements
}
catch(error) {
    // catch_statements
}
finally() {
    // codes that gets executed anyway
}
```

2. Javascript Modules

As our program grows bigger, it may contain many lines of code. Instead of putting everything in a single file, you can use modules to separate codes in separate files according to their functions. This makes our code organized and easier to maintain.

Benefits of Using Modules:

- The code base is easier to maintain because different codes having different functionalities are in different files.
- Makes code reusable. You can define a module and use it numerous times as per your needs.

▼ **export and import**

```
// exporting a function
export function greetPerson(name) {
    return `Hello ${name}`;
}
```

```
// importing greetPerson from greet.js file
import { greetPerson } from './greet.js';

// using greetPerson() defined in greet.js
let displayName = greetPerson('Jack');
```

▼ Rename export and import

```
export {
    function1 as newName1,
    function2 as newName2
};
```

```
import { function1 as newName1, function2 as newName2 } from './module.js';
```

▼ Default export

```
// default export
export default function greet(name) {
    return `Hello ${name}`;
}

export const age = 23;
```

```
import random_name from './greet.js';
```

While performing default export,

- `random_name` is imported from `greet.js`. Since, `random_name` is not in `greet.js`, the default export (`greet()` in this case) is exported as `random_name`.
- You can directly use the default export without enclosing curly brackets `{}`.

Note: A file can contain multiple exports. However, you can only have one default export in a file.

3. Filter, Map & Find

All these are higher-order functions. A higher-order function is a function that either takes other functions as arguments or returns a function.

▼ Array filter method

The `filter()` method returns a new array with all elements that pass the test defined by the given function.

```
let numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

// function to check even numbers
function checkEven(number) {
  if (number % 2 == 0)
    return true;
  else
    return false;
}

// create a new array by filter even numbers from the numbers array
let evenNumbers = numbers.filter(checkEven);
console.log(evenNumbers);

// Output: [ 2, 4, 6, 8, 10 ]
```

- `filter()` does not change the original array.
- `filter()` does not execute `callback` for array elements without values

▼ Array map method

The `map()` method creates a new array with the results of calling a function for every array element.

```
let numbers = [2, 4, 6, 8, 10];

// function to return the square of a number
function square(number) {
  return number * number;
}

// apply square() function to each item of the numbers list
let square_numbers = numbers.map(square);
console.log(square_numbers);

// Output: [ 4, 16, 36, 64, 100 ]
```

- `map()` does not change the original array.
- `map()` executes `callback` once for each array element in order.
- `map()` does not execute `callback` for array elements without values.

▼ **Array find method**

```
const emp = [
    {
        name: "Ram",
        empID: 101
    },
    {
        name: "Sham",
        empID: 102
    },
    {
        name: "Mohan",
        empID: 103
    }
];

const res = emp.find(el => el.empID === 102);

console.log("res is: ", res); // res is: {name: 'Sham', empID: 102}
```

4. Javascript ES6 Concepts

JavaScript ES6 (also known as ECMAScript 2015 or ECMAScript 6) is the newer version of JavaScript that was introduced in 2015.

We have already seen some ES6 concepts like **Arrow functions, Javascript Classes, Template literals, Javascript Destructuring,** etc. Some useful ES6 concepts are mentioned below:

▼ **Default Parameter Values**

In the ES6 version, you can pass default values in the function parameters. For example,

```
function sum(x, y = 5) {

    // take sum
    // the value of y is 5 if not passed
    console.log(x + y);
}

sum(5); // 10
sum(5, 15); // 20
```

In the above example, if you don't pass the parameter for `y`, it will take **5** by default.

▼ **JavaScript Spread Operator**

```
function show(a, b, ...args) {
  console.log(a); // one
  console.log(b); // two
  console.log(args); // ["three", "four", "five", "six"]
}

show('one', 'two', 'three', 'four', 'five', 'six')
```

You use the **spread syntax** `...` to copy the items into a single array. For example,

```
let arr1 = ['one', 'two'];
let arr2 = [...arr1, 'three', 'four', 'five'];
console.log(arr2); // ["one", "two", "three", "four", "five"]
```

Assignments

1. Create a Filter Function for a String Array that takes a condition function to filter. Also, create the condition functions mentioned below:
    a. Filter out all the values with less than 3 characters.
    b. Filter out all the values that contain the expression “hi”.
    c. Filter out all the values that are palindrome.

2. Create a Map function for Number Array that takes the following modification functions. Also, create the modification functions.
    a. Modify and return an array with square roots.
    b. Modify the Numbers and make them String and return an array.
3. Use the following array:

```
let users = [
  {firstName : "Susan", lastName: "Steward"},
  {firstName : "Daniel", lastName: "Longbottom"},
  {firstName : "Jacob", lastName: "Black"}
];
```

    a. Filter the users whose firstName is “Susan”.
    b. Filter out the users whose lastName starts with the letter L.
    c. Map and return an array called fullName with full name values like “Susan Steward”