**Challenge1**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>ES6 Rest/Spread Operators and Default Function Parameters</title>

</head>

<body>

    <script src="challenge1.js"></script>

</body>

</html>

**Challenge1.js**

function meanScore(...scores){

    const LEN=scores.length;

    if(scores.every(element => typeof element === "number")){

        return parseFloat((scores.reduce((sum,score)=> sum+score ,0))/LEN).toFixed(2);

    }

    else{

        throw new Error("Supplied arguments must contain only numbers!");

    }

}

const scores1 = [0, 1.5, 2.5, 3.7];

const scores2 = [1.7, 4.5, 0, 4.9, 5.0, 4.2];

const scores3 = [1.3, 2.5, 1.9];

const scores4 = ["abc", 1.3, true, 2.5, 1.9];

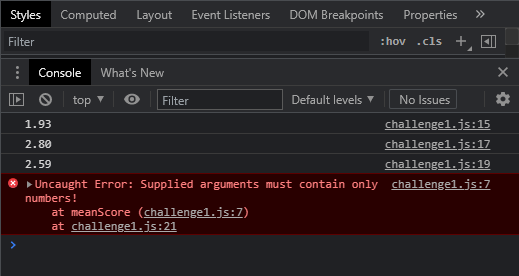
console.log(meanScore(...scores1)); // 1.93

console.log(meanScore(...scores1,...scores2)); // 2.8

console.log(meanScore(...scores1,...scores2,...scores3)); // 2.59

console.log(meanScore(...scores4)); // Supplied arguments must contain only numbers!

**Result**

****

**Challenge2**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>ES6 Rest/Spread Operators and Default Function Parameters</title>

</head>

<body>

    <script src="challenge2.js"></script>

</body>

</html>

**Challenge2.js**

/\* Create a function "weatherForecast" with two parameters "city" and "weather".

If second argument is absent, "weather" parameter should get value "Great weather!".

1. First solve this WITHOUT default function parameter.

2. Comment previous solution and solve same task WITH default function parameter.

NOTE: Carefully compare your results with test calls results

\*/

// // 1. First solve this WITHOUT default function parameter.

// const weatherForecast = (city, weather) => {

//   // weather = weather || "Great weather!"; // a || b -> b if a is falsy (0, "", null, undefined)

//   weather =

//     weather !== undefined ? weather : "Great weather!";

//   return `Weather forecast for ${city}: ${weather}`;

// };

// 2. Comment previous solution and solve same task WITH default function parameter.

const weatherForecast = (

    city,

    weather = "Great weather!"

  ) => `Weather forecast for ${city}: ${weather}`;

  console.log(weatherForecast("Dubai", "Sunny"));

  // Weather forecast for Dubai: Sunny

  console.log(weatherForecast("London", "Light rain"));

  // Weather forecast for London: Light rain

  console.log(weatherForecast("Paris"));

  // Weather forecast for Paris: Great weather!

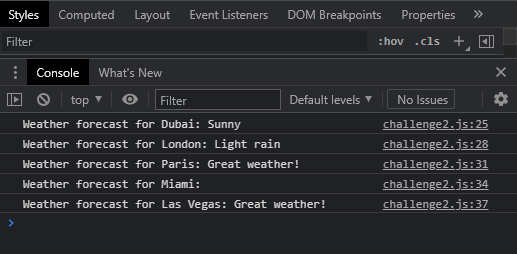
  console.log(weatherForecast("Miami", ""));

  // Weather forecast for Miami:

  console.log(weatherForecast("Las Vegas", undefined));

  // Weather forecast for Las Vegas: Great weather!

**Result**

****

**Challenge3**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>ES6 Rest/Spread Operators and Default Function Parameters</title>

</head>

<body>

    <script src="challenge3.js"></script>

</body>

</html>

**Challenge3.js**

const generateTasksIds = tasks => {

    let newTaskIds = [];

    return tasks.map(({ ...task }) => {

      if (!task.hasOwnProperty("taskId")) {

        let newTaskId;

        do {

          newTaskId = 1000 + Math.floor(Math.random() \* 9000);

        } while (

          tasks.find(task => task.taskId === newTaskId) ||

          newTaskIds.includes(newTaskId)

        );

        newTaskIds.push(newTaskId);

        task.taskId = newTaskId;

      }

      return task;

    });

  };

  const tasksSortedByIds = (

    tasks,

    tasksWithIds = generateTasksIds(tasks)

  ) => {

    console.log(

      `Quantity of the missing taskIds is ${

        tasks.filter(task => !task.hasOwnProperty("taskId"))

          .length

      }`

    );

    return tasksWithIds.sort((a, b) => a.taskId - b.taskId);

  };

  const tasks = [

    { title: "Meeting with John", taskId: 4621 },

    { title: "Visit gym", taskId: 6821 },

    { title: "Buy new phone" },

    { title: "Clean the room", taskId: 2721 },

    { title: "Plan a trip" }

  ];

  console.log(

    "Sorted array of tasks with taskIds:",

    tasksSortedByIds(tasks)

  );

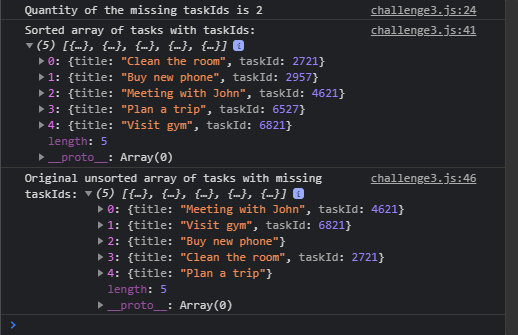
  console.log(

    "Original unsorted array of tasks with missing taskIds:",

    tasks

  );

**Result**

****