**//Recursion and stack**

**//task 1**

<script>

    function  factorial(n){

      if(n ==0 || n==1)

      return 1;

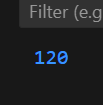
    return n\* factorial(n-1);

    }

      console.log(factorial(5));

    </script>

**Output:**

****

**//task 2**

<script>

    function fibonacci(n) {

    if (n <= 0) return 0;

    if (n === 1) return 1;

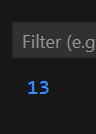
    return fibonacci(n - 1) + fibonacci(n - 2);

}

      console.log(fibonacci(7));

    </script>

**Output:**

****

**//task 3**

 <script>

    function climbStairs(n) {

    if (n < 0) return 0;

    if (n === 0) return 1;

    return climbStairs(n - 1) + climbStairs(n - 2) + climbStairs(n - 3);

}

      console.log(climbStairs(7));

    </script>

**Output:**

****

**//task 4**

    <script>

      function flattenArray(arr) {

    return arr.reduce((flat, item) => {

        return flat.concat(Array.isArray(item) ? flattenArray(item) : item);

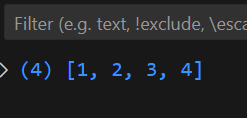
    }, []);

}

      console.log(flattenArray([1,[2,3],4]));

    </script>

**Output:**

****

**//task 5**

<script>

     function towerOfHanoi(n, source, target, auxiliary) {

    if (n === 1) {

        console.log(`Move disk 1 from ${source} to ${target}`);

        return;

    }

    towerOfHanoi(n - 1, source, auxiliary, target);

    console.log(`Move disk ${n} from ${source} to ${target}`);

    towerOfHanoi(n - 1, auxiliary, target, source);

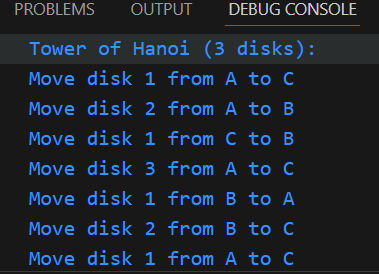
}

console.log("Tower of Hanoi (3 disks):");

towerOfHanoi(3, "A", "C", "B");

    </script**>**

**Output:**

****

**2. JSON and variable length arguments/spread syntax:**

**//Task:1**

 <script>

        function sum(...args) {

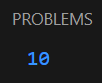
    return args.reduce((acc, curr) => acc + curr, 0);

}

console.log(sum(1, 2, 3, 4))

    </script>

**Output:**

****

**//task 2**

 <script>

        function sum(...args) {

            return args.reduce((acc, curr) => acc + curr, 0);

        }

        function sumArray(numbers) {

            return sum(...numbers);

        }

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

    </script>

**Output:**



**//task:3**

<script>

function deepClone(obj) {

    return JSON.parse(JSON.stringify(obj));

}

const original = { a: 1, b: { c: 2 } };

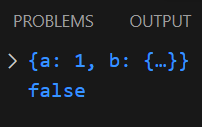
const clone = deepClone(original);

console.log(clone);

console.log(original === clone);

    </script>

**Output:**



**//task:4**

 <script>

    function mergeObjects(obj1, obj2) {

        return { ...obj1, ...obj2 };

    }

    const obj1 = { a: 1, b: 2 };

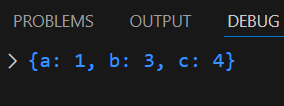
    const obj2 = { b: 3, c: 4 };

    const merged = mergeObjects(obj1, obj2);

    console.log(merged);

    </script>

**Output:**

****

**//task : 5**

<script>

    function serializeAndParse(obj) {

    const jsonString = JSON.stringify(obj);

    return JSON.parse(jsonString);

}

const myObject = { x: 10, y: 20 };

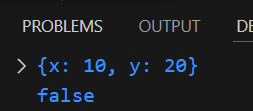
const newObject = serializeAndParse(myObject);

console.log(newObject);

console.log(myObject === newObject);

    </script>

**Output:**

****

**3. Closure:**

**//task:1**

 <script>

      function createFunction() {

    let localVar = "I am captured!";

    return function() {

        return localVar;

    };

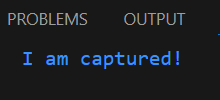
}

const capturedFunction = createFunction();

console.log(capturedFunction());

    </script>

**Output:**

****

**//task:2**

<script>

      function counter() {

    let count = 0;

    return {

        increment: function() {

            count++;

        },

        getCount: function() {

            return count;

        }

    };

}

const myCounter = counter();

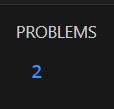
myCounter.increment();

myCounter.increment();

console.log(myCounter.getCount());

    </script>

**Output:**

****

**//task 3**

**<script>**

function createCounter() {

    let count = 0;

    return {

        increment: function() {

            count++;

        },

        getCount: function() {

            return count;}

           };}

const counter1 = createCounter();

const counter2 = createCounter();

counter1.increment();

counter2.increment();

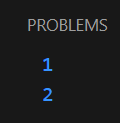
counter2.increment();

console.log(counter1.getCount());

console.log(counter2.getCount());

    </script>

**Output:**

****

**//task 4:**

 <script>

function secretHolder() {

    let secret = "This is a secret!";

    return {

        getSecret: function() {

            return secret;

        },

        setSecret: function(newSecret) {

            secret = newSecret; }

    };}

const secretObj = secretHolder();

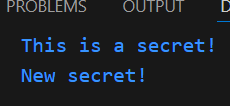
console.log(secretObj.getSecret());

secretObj.setSecret("New secret!");

console.log(secretObj.getSecret());

  </script>

**Output:**

****

**//task 5:**

<script>

function multiplierFactory(factor) {

    return function(number) {

        return number \* factor;

    };}

const double = multiplierFactory(2);

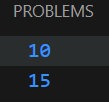
const triple = multiplierFactory(3);

console.log(double(5));

console.log(triple(5));

</script>

**Output:**

****

**4. Promise, Promises chaining:**

**//task 1**

<script>

    function delayedGreeting(seconds) {

        return new Promise((resolve) => {

            setTimeout(() => {

                resolve(`Hello after ${seconds} seconds!`);

            }, seconds \* 1000);

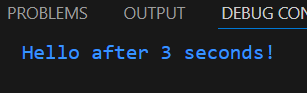
        });

    }

    delayedGreeting(3).then((greeting) => console.log(greeting));

  </script>

**Output:**

****

**//task 2**

**Output:**

**//task 3**

 <script>

    function randomPromise() {

        return new Promise((resolve, reject) => {

            const randomNumber = Math.random();

            if (randomNumber > 0.5) {

        resolve("Success! Number was " + randomNumber);

            } else {

                reject("Failure! Number was " + randomNumber);

            }

        });}

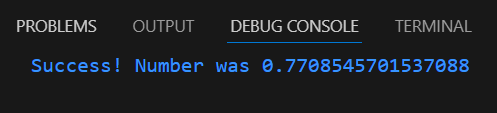
    randomPromise()

        .then((message) => console.log(message))

        .catch((error) => console.error(error));

  </script>

**Output:**

****

**//task 4**

<script>

   const urls = [

    "https://jsonplaceholder.typicode.com/posts/1",

    "https://jsonplaceholder.typicode.com/posts/2",

    "https://jsonplaceholder.typicode.com/posts/3",

];

Promise.all(urls.map((url) => fetch(url).then((response) => response.json())))

    .then((data) => {

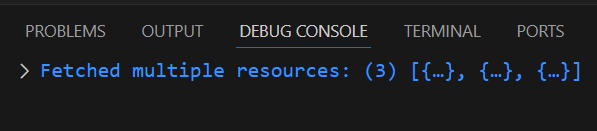
        console.log("Fetched multiple resources:", data);

    })

    .catch((error) => console.error("Error fetching resources:", error));

  </script>

**Output:**



**//task 5:**

 <script>

    function step1() {

        return new Promise((resolve) => {

            console.log("Step 1: Starting...");

            setTimeout(() => resolve("Step 1 complete"), 1000);

        });}

    function step2() {

        return new Promise((resolve) => {

            console.log("Step 2: Starting...");

            setTimeout(() => resolve("Step 2 complete"), 2000);

        });}

    function step3() {

        return new Promise((resolve) => {

            console.log("Step 3: Starting...");

            setTimeout(() => resolve("Step 3 complete"), 3000);

        });

    }

    step1()

        .then((message) => {

            console.log(message);

            return step2();

        })

        .then((message) => {

            console.log(message);

            return step3();

        })

        .then((message) => {

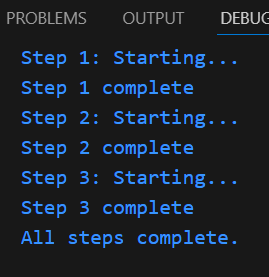
            console.log(message);

            console.log("All steps complete.");

        });

    </script>

**Output:**

****

**5. Async/await:**

**//task 1**

<script>

function fetchData() {

    return new Promise((resolve, reject) => {

        setTimeout(() => resolve("Data fetched"), 1000);

    });}

async function fetchDataAsync() {

    return await new Promise((resolve, reject) => {

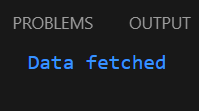
        setTimeout(() => resolve("Data fetched"), 1000);

    });}

fetchDataAsync().then(console.log);

    </script>

**Output:**

****

**//task 2**

<script>

    async function fetchAndProcessData(apiUrl) {

        try {

            const response = await fetch(apiUrl);

            if (!response.ok) throw new Error("Failed to fetch data");

            const data = await response.json();

            console.log("Processed Data:", data);

        } catch (error) {

            console.error("Error fetching and processing data:", error.message);

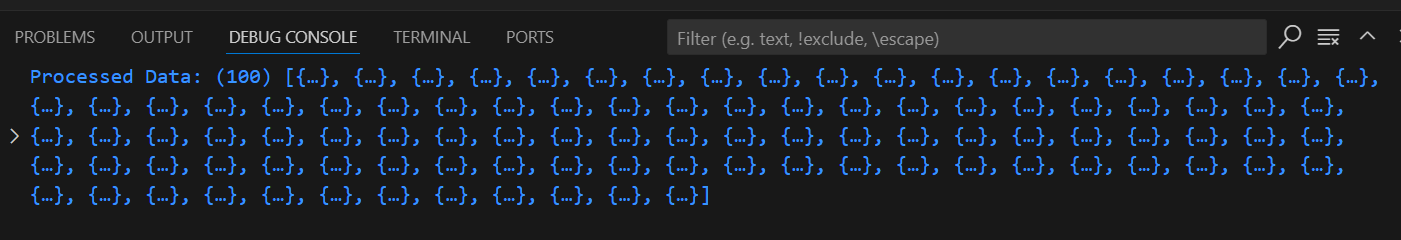
        }

    }

    fetchAndProcessData("https://jsonplaceholder.typicode.com/posts");

    </script>

**Output:**

****

**//task 3**

<script>

async function divideNumbersAsync(a, b) {

    try {

        if (b === 0) throw new Error("Division by zero is not allowed");

        const result = a / b;

        console.log("Result:", result);

} catch (error) {

        console.error("Error:", error.message);

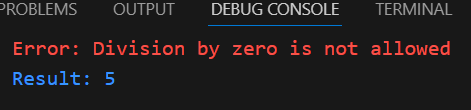
    }}

divideNumbersAsync(10, 0);

divideNumbersAsync(10, 2);

    </script>

**Output:**

****

**//task 4**

<script>

    async function fetchMultipleUrls(urls) {

        try {

            const fetchPromises = urls.map(url => fetch(url).then(res => res.json()));

            const results = await Promise.all(fetchPromises);

            console.log("Results:", results);

        } catch (error) {

            console.error("Error fetching multiple URLs:", error.message);

        }

    }

    const urls = [

        "https://jsonplaceholder.typicode.com/posts/1",

        "https://jsonplaceholder.typicode.com/posts/2",

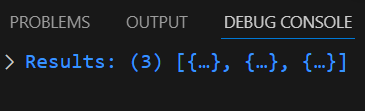
        "https://jsonplaceholder.typicode.com/posts/3"

    ];

    fetchMultipleUrls(urls);

    </script>

**Output:**

****

**//task 5**

 <script>

        async function performAsyncOperations() {

    const operation1 = new Promise(resolve => setTimeout(() => resolve("Operation 1 completed"), 1000));

    const operation2 = new Promise(resolve => setTimeout(() => resolve("Operation 2 completed"), 2000));

    const operation3 = new Promise(resolve => setTimeout(() => resolve("Operation 3 completed"), 1500));

    console.log("Waiting for all operations to complete...");

    const results = await Promise.all([operation1, operation2, operation3]);

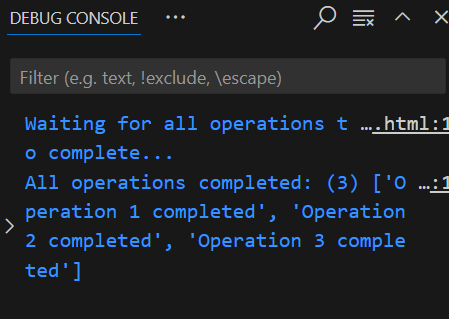
    console.log("All operations completed:", results);

}

performAsyncOperations();

    </script>

**Output:**

****