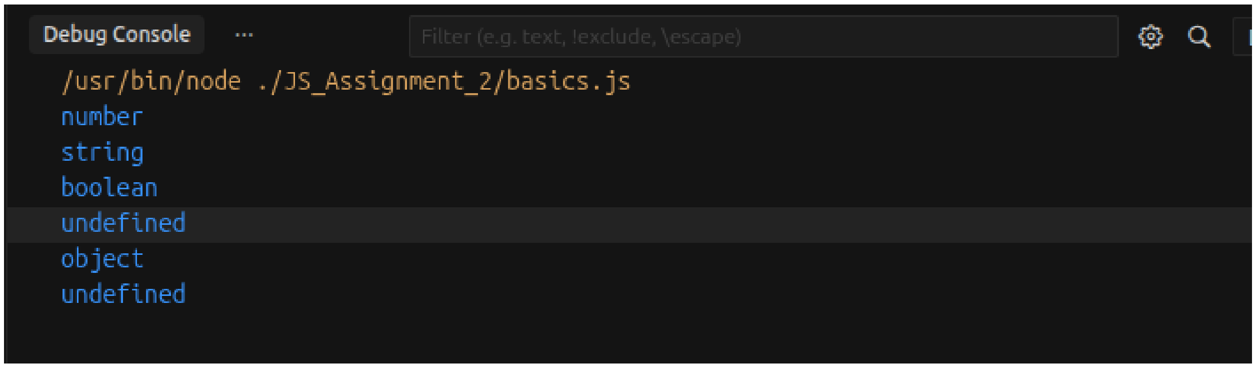
1. Write a function that accepts different data types and prints their type using typeof.

```
function input( a){
   console.log(typeof(a))
}

input(10)
input("abcd")
input(true)
input(undefined)
input({})
input()
```

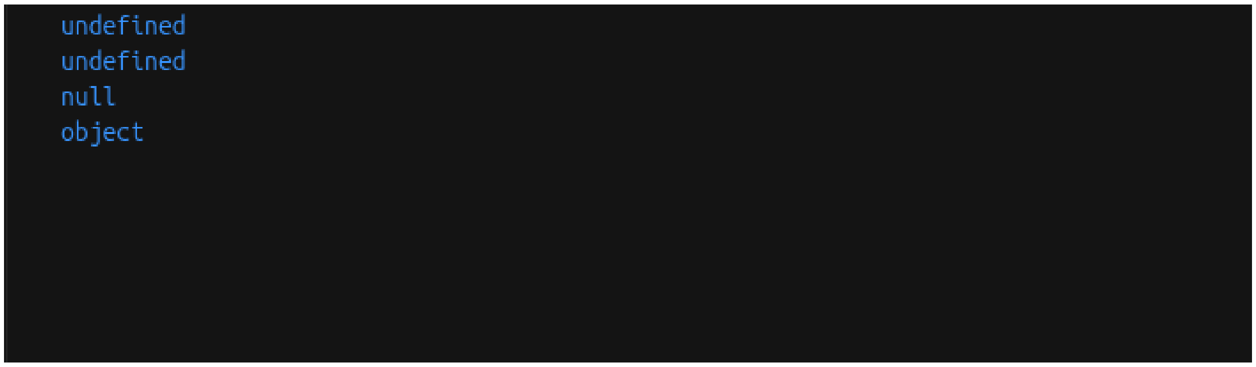
Output:



```
/usr/bin/node ./JS_Assignment_2/basics.js
number
string
boolean
undefined
object
undefined
```

1. Explain the difference between null and undefined with code.

```

let a;
console.log(a);
console.log(typeof a);

let b = null;
console.log(b);
console.log(typeof b);
```



```
undefined
undefined
null
object
```

Undefined means no value is given to a variable
Null means values is intentionally assigned to a variable

2 Functions

Concepts: normal functions, arrow functions

Assignments

1. Write a normal function to add two numbers.

2. Convert a above function into an arrow function.

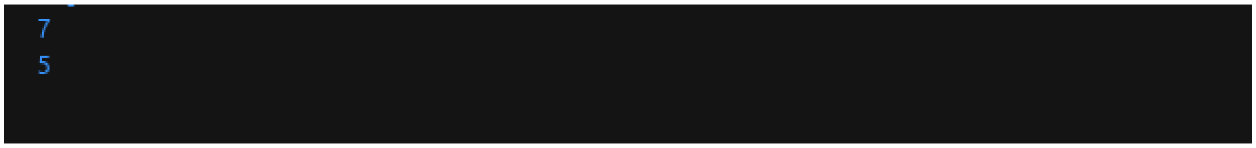
Normal function

A normal function is the regular way to define a function.
It has its own `this` and is commonly used for object methods.

Arrow function

An arrow function is a shorter way to write a function.
It does not have its own `this` and takes `this` from outside.

```
function add(a,b){
    console.log(a+b)
}
add(2,5)

const addition =(a,b)=>{
    console.log(a+b)
}
addition(2,3)
```



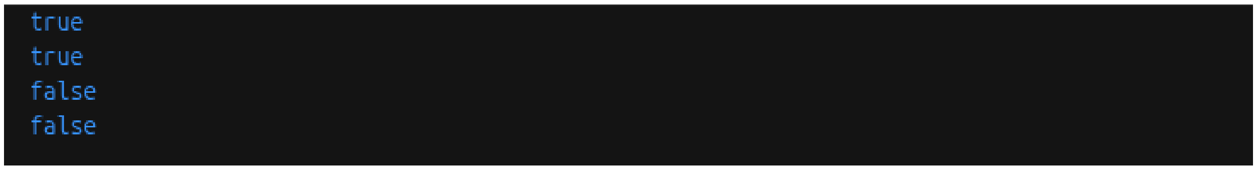
```
7
5
```

3 Strings

Concepts: string methods

1. Write difference between == and === in java script with examples.
2. "hello world" convert to title case.

```
console.log(5=="5")
console.log(null==undefined)
console.log(5==="5")
console.log(null===undefined)
```



```
true
true
false
false
```

== (loose equality)

Compares values only

Type conversion if needed

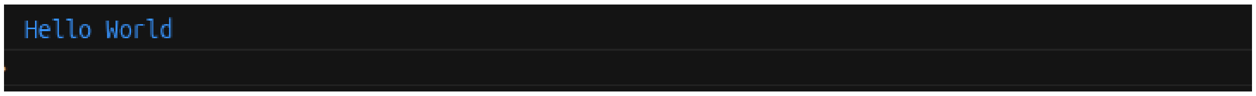
=== (strict equality)

Compares value and type

No type conversion

2. "hello world" convert to title case.

```javascript
const text = "hello world";

const titleCase = text
 .split(" ")
 .map(word => word[0].toUpperCase() + word.slice(1))
 .join(" ");

console.log(titleCase);
```



```
Hello World
```

4 Objects

Concepts: object creation, access, iteration

Assignments

1. **Create a user object with properties name, age, and city.**
2. **Print all keys and values using methods Object.keys, Object.values and forEach loop**
3. **Add new property mobileNumber and delete city properties dynamically.**
4. **Convert below array of object group by role**

   const users = [

{ name: "Pratik", role: "admin" },

{ name: "Amit", role: "user" },

{ name: "Neha", role: "admin" },

{ name: "Ravi", role: "user" },

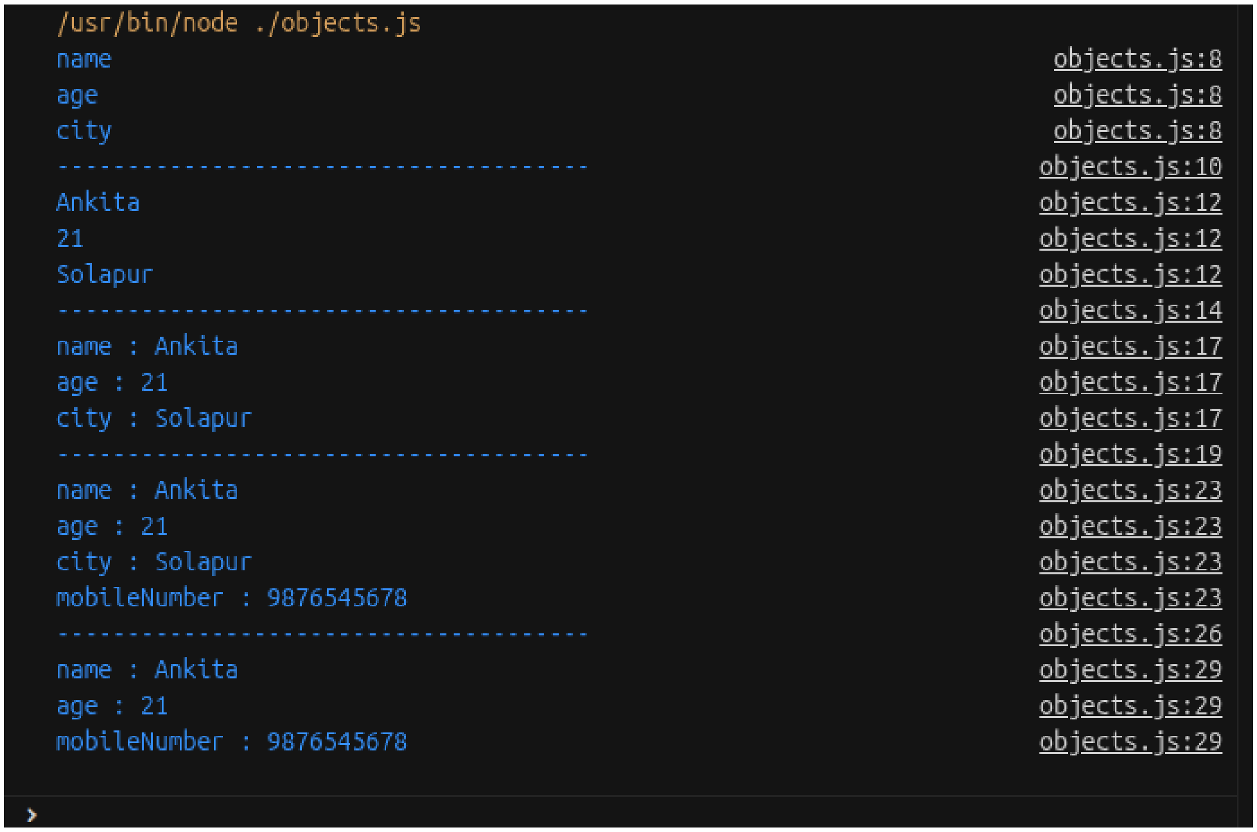
];

Output:

```
{
 admin: [
  { name: "Pratik", role: "admin" },
  { name: "Neha", role: "admin" }
 ],
 user: [
  { name: "Amit", role: "user" },
  { name: "Ravi", role: "user" }
 ]
}
```

```javascript
let user={
   name:"Ankita",
   age:21,
   city:"Solapur"
}

Object.keys(user).forEach(key=>{
   console.log(key);
});
console.log("--------------------------------------")
Object.values(user).forEach(value=>{
   console.log(value);
});
console.log("--------------------------------------")

Object.keys(user).forEach(key => {
   console.log(key + " : " + user[key]);
 });
```

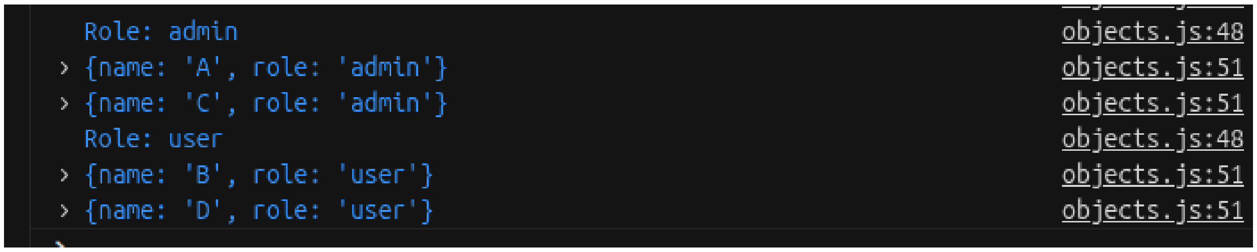
```
console.log("--------------------------------------")
 user.mobileNumber=9876545678
Object.keys(user).forEach(key => {
   console.log(key + " : " + user[key]);
 });

console.log("--------------------------------------")
delete user.city
Object.keys(user).forEach(key => {
   console.log(key + " : " + user[key]);
 });
```



```
/usr/bin/node ./objects.js
name                                                          objects.js:8
age                                                           objects.js:8
city                                                          objects.js:8
--------------------------------------                        objects.js:10
Ankita                                                        objects.js:12
21                                                            objects.js:12
Solapur                                                       objects.js:12
--------------------------------------                        objects.js:14
name : Ankita                                                 objects.js:17
age : 21                                                      objects.js:17
city : Solapur                                                objects.js:17
--------------------------------------                        objects.js:19
name : Ankita                                                 objects.js:23
age : 21                                                      objects.js:23
city : Solapur                                                objects.js:23
mobileNumber : 9876545678                                     objects.js:23
--------------------------------------                        objects.js:26
name : Ankita                                                 objects.js:29
age : 21                                                      objects.js:29
mobileNumber : 9876545678                                     objects.js:29
>
```

```
const users = [
  { name: "A", role: "admin" },
  { name: "B", role: "user" },
  { name: "C", role: "admin" },
  { name: "D", role: "user" }
];
```

```
const group=users.reduce((res,curr)=>{
 const role=curr.role
 if(!res[role])
   res[role]=[];
 res[role].push(curr)
 return res
},{});

Object.keys(group).forEach(role => {
 console.log("Role:", role);

 group[role].forEach(user => {
   console.log(user);
 });
});
```



```
  Role: admin                                   objects.js:48
> {name: 'A', role: 'admin'}                    objects.js:51
> {name: 'C', role: 'admin'}                    objects.js:51
  Role: user                                    objects.js:48
> {name: 'B', role: 'user'}                     objects.js:51
> {name: 'D', role: 'user'}                     objects.js:51
```

5 Array Methods (Important)

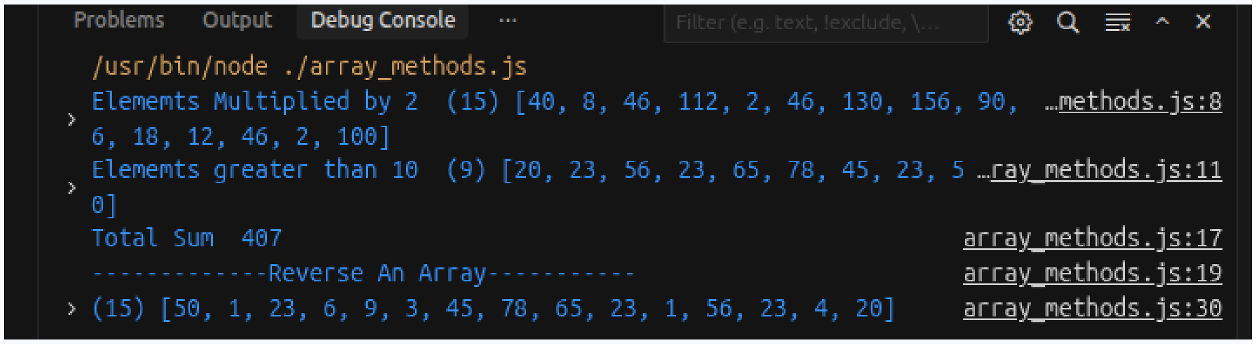
Concepts: map, filter, reduce

[20, 4, 23, 56, 1, 23, 65, 78, 45, 3, 9, 6, 23, 1, 50]

Assignments

1. Use map to multiply each array element by 2.
2. Use filter to find numbers greater than 10.
3. Use reduce to find the sum of array elements.
4. Reverse an array.

```
let arr=[20, 4, 23, 56, 1, 23, 65, 78, 45, 3, 9, 6, 23, 1, 50]

let multiple=arr.map(n=>{
   return n*2
})
console.log("Elememts Multiplied by 2 ",multiple)

let filtered=arr.filter(n=> n>10)
console.log("Elememts greater than 10 ",filtered)

let total_sum = arr.reduce((total,num)=>{
  return total+num
},0)

console.log("Total Sum ",total_sum)

console.log("-------------Reverse An Array-----------")

let i=0,j=arr.length-1
while(i<=j){
   let temp=arr[i]

   arr[i]=arr[j]

   arr[j]=temp

   i++

   j--

}


console.log(arr)
```



Problems Output Debug Console ···

```
/usr/bin/node ./array_methods.js
Elememts Multiplied by 2  (15) [40, 8, 46, 112, 2, 46, 130, 156, 90,  …methods.js:8
6, 18, 12, 46, 2, 100]
Elememts greater than 10  (9) [20, 23, 56, 23, 65, 78, 45, 23, 5 …ray_methods.js:11
0]
Total Sum  407                                                      array_methods.js:17
-------------Reverse An Array----------                             array_methods.js:19
(15) [50, 1, 23, 6, 9, 3, 45, 78, 65, 23, 1, 56, 23, 4, 20]         array_methods.js:30
```

6 ES6+ Features

Concepts: destructuring, spread, rest

Assignments

1. **Destructure an object and console name and age from it.**
   const user = {

name: "Akshay",

age: 25,

city: "Pune"

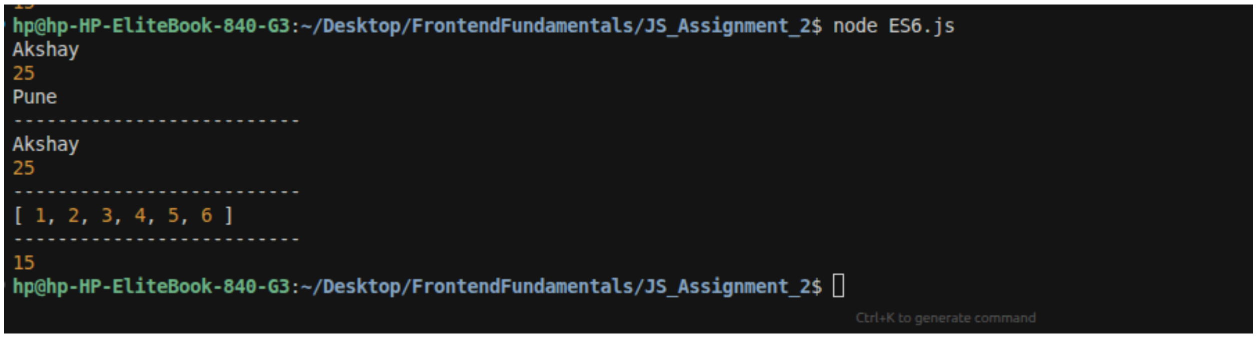
};

output :

```
console.log(name); // Akshay

console.log(age); // 25
```

```
const user = {

   name: "Akshay",
   age: 25,
   city: "Pune"
 };

console.log(user.name)
console.log(user.age)
console.log(user.city)

console.log("--------------------------")

const { name, age } = user;

console.log(name)
console.log(age)
console.log("--------------------------")



let arr1=[1,2,3]
let arr2=[4,5,6]
let merged=[...arr1,...arr2]
console.log(merged)

console.log("--------------------------")

function sum(...numbers){
   total_sum=0
   numbers.forEach(num=>{
       total_sum+=num
   })
   return total_sum
}
```

```
console.log("Total Sum is ",sum(1,2,3,4,5))
```



```
hp@hp-HP-EliteBook-840-G3:~/Desktop/FrontendFundamentals/JS_Assignment_2$ node ES6.js
Akshay
25
Pune
--------------------------
Akshay
25
--------------------------
[ 1, 2, 3, 4, 5, 6 ]
--------------------------
15
hp@hp-HP-EliteBook-840-G3:~/Desktop/FrontendFundamentals/JS_Assignment_2$
```

Ctrl+K to generate command

7 Closures

Concepts: closures, lexical scope

Assignments

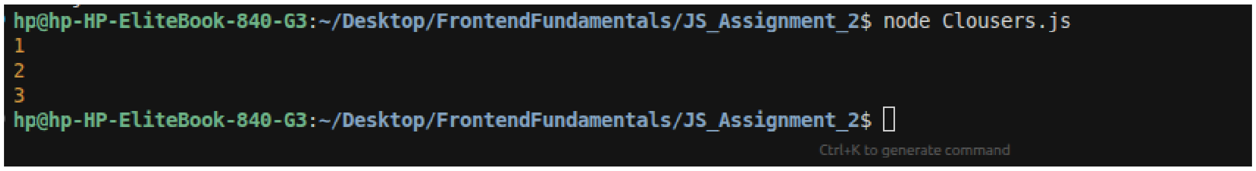
1. **Create a counter function using closure.**

2. **Explain how inner functions access outer variables.**

Clouser:

A **closure** is created when an inner function **remembers and uses variables from its outer function**, even after the outer function has finished running.

```
function outer(){
  let count=0;
   return function(){
       count++;
       return count
   };
}

const counter=outer()
console.log(counter())
console.log(counter())
console.log(counter())
```



```
hp@hp-HP-EliteBook-840-G3:~/Desktop/FrontendFundamentals/JS_Assignment_2$ node Clousers.js
1
2
3
hp@hp-HP-EliteBook-840-G3:~/Desktop/FrontendFundamentals/JS_Assignment_2$ 
```

`count` is inside `outer`
The returned inner function still remembers `count`
Each call updates the same `count` value

8 Callbacks

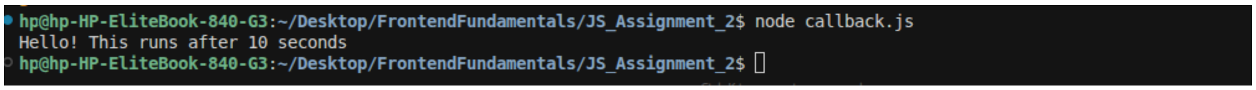
Concepts: callback functions

Assignments

1. **Create a function that accepts a callback and executes it after 10 seconds.**

A callback is just a function passed into another function and called later.

```
function practice(callback) {
   setTimeout(function () {
     callback();
   }, 10000);
 }
  function sayHello() {
   console.log("Hello! This runs after 10 seconds");
 }
  practice(sayHello);
```



```
hp@hp-HP-EliteBook-840-G3:~/Desktop/FrontendFundamentals/JS_Assignment_2$ node callback.js
Hello! This runs after 10 seconds
hp@hp-HP-EliteBook-840-G3:~/Desktop/FrontendFundamentals/JS_Assignment_2$ 
```

9 Promises

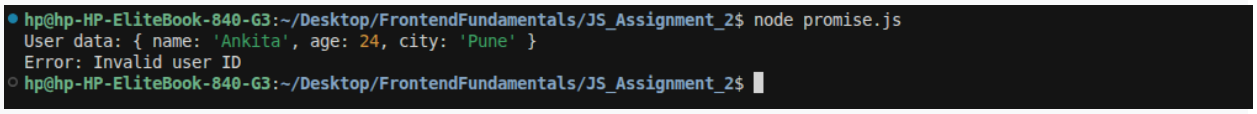
Concepts: Promise, then, catch

Assignments

1. **Create a function called getUserData that:**

- **Returns a Promise**
- **Resolves with user details object contains name, age, city if userId is 1**
- **Rejects with an error message if userId is 0**
- **Handles the response using .then() and .catch()**

```javascript
function getUserData(userId) {

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

     if (userId === 1) {

       resolve({

         name: "Ankita",

         age: 24,

         city: "Pune"

       });

     } else if (userId === 0) {

       reject("Invalid user ID");

     }

   });

 }



  getUserData(1)

 .then((user) => {

   console.log("User data:", user);

 })

 .catch((error) => {
   console.log("Error:", error);

 });


getUserData(0)

 .then((user) => {

   console.log(user);

 })

 .catch((error) => {

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

 });
```



```
hp@hp-HP-EliteBook-840-G3:~/Desktop/FrontendFundamentals/JS_Assignment_2$ node promise.js
User data: { name: 'Ankita', age: 24, city: 'Pune' }
Error: Invalid user ID
hp@hp-HP-EliteBook-840-G3:~/Desktop/FrontendFundamentals/JS_Assignment_2$ 
```

2. **Guess the execution sequence of below code**

   console.log("1: Start");

```
setTimeout(() => {

  console.log("2: setTimeout");

}, 0);


Promise.resolve().then(() => {
 console.log("3: Promise");
});
console.log("4: End");
```

Correct output order

```
1: Start
4: End
3: Promise
2: setTimeout
```

10 Async / Await

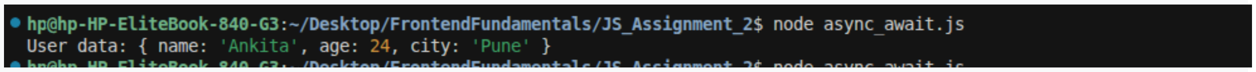
Concepts: async functions, error handling

Assignments

1. **Convert the question 1 from assignment 9, to async/await with try , catch block**

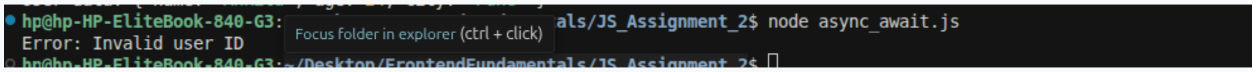
`async / await` is just a cleaner way to write Promises so the code looks normal and easy to read.

```
async function getUserData(userId) {
 if (userId === 1) {
   return {
     name: "Ankita",
     age: 24,
     city: "Pune"
   };

 } else if (userId === 0) {

   throw "Invalid user ID";

 }

}

async function fetchUser() {

 try {

   const user = await getUserData(1);

   console.log("User data:", user);

 } catch (error) {

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

 }

}

fetchUser();
```



```
● hp@hp-HP-EliteBook-840-G3:~/Desktop/FrontendFundamentals/JS_Assignment_2$ node async_await.js
  User data: { name: 'Ankita', age: 24, city: 'Pune' }
● hp@hp-HP-EliteBook-840-G3:~/Desktop/FrontendFundamentals/JS_Assignment_2$ node async_await.js
```

If user_id is 0



```
● hp@hp-HP-EliteBook-840-G3:                                        als/JS_Assignment_2$ node async_await.js
  Error: Invalid user ID
○ hp@hp-HP-EliteBook-840-G3:~/Desktop/FrontendFundamentals/JS_Assignment_2$ 
```

11 DOM Manipulation

Concepts: DOM selection, events

Assignments

1. **Take one div with some text and change text of an element on button click.**
   <div id="message">Hello World</div>

<button id="changeBtn">Change Text</button>

2. **Add a new list item dynamically.**
   <ul id="list"><li>Item 1</li></ul>
   <button id="addItem">Add Item</button>

3. **Remove an element from the DOM.**
   <p id="removeMe">Remove this text</p>
   <button id="removeBtn">Remove</button>

4. **Display input value on screen while typing.**

   <input type="text" id="inputBox" />
   <p id="output"></p>

```html
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

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

   <title>Document</title>

</head>

<body>
```

```
   // Change text on button click

   <div id="message">Hello World</div>

<button id="changeBtn">Change Text</button>


<script>

 const message = document.getElementById("message");

 const changeBtn = document.getElementById("changeBtn");


 changeBtn.addEventListener("click", function () {

   message.textContent = "Text Changed!";

 });

</script>




// 2 Add a new list item dynamically

<ul id="list">

   <li>Item 1</li>

 </ul>

 <button id="addItem">Add Item</button>

  <script>

   const list = document.getElementById("list");

   const addItem = document.getElementById("addItem");
```

```
    let count = 2;

    addItem.addEventListener("click", function () {

     const li = document.createElement("li");

     li.textContent = "Item " + count;

     list.appendChild(li);

     count++;

   });

 </script>


  // 3. Remove an element from the DOM

  <p id="removeMe">Remove this text</p>

  <button id="removeBtn">Remove</button>


  <script>

    const removeText = document.getElementById("removeMe");

    const removeBtn = document.getElementById("removeBtn");


    removeBtn.addEventListener("click", function () {

      removeText.remove();

    });

  </script>


// 4. Display input value while typing
```

```
<input type="text" id="inputBox" />

<p id="output"></p>


<script>

 const inputBox = document.getElementById("inputBox");

 const output = document.getElementById("output");


 inputBox.addEventListener("input", function () {

   output.textContent = inputBox.value;

 });

</script>


</body>

</html>
```

12 Timers

Concepts: setInterval

Assignments

1. **Create a countdown timer using setInterval.**

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

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

   <title>Document</title>

</head>

<body>

   <p id="timer">10</p>

   <button id="startBtn">Start Countdown</button>

   <script>

       let count = 10;

       const timer = document.getElementById("timer");

       const startBtn = document.getElementById("startBtn");


       startBtn.addEventListener("click", function () {

         const intervalId = setInterval(function () {

           timer.textContent = count;


           if (count === 0) {

             clearInterval(intervalId);

             timer.textContent = "Time's up!";
           }


           count--;

         }, 1000);

       });

     </script>


</body>

</html>
```

`setInterval` is used to run the same code again and again after a fixed time gap.

In this above code , Starts from 10

Updates number every 1 second

Stops automatically at 0

Shows “Time’s up!” at the end