Name Rohan Lalchandani
Class: D15A Roll no.: 25

Lab Exercise 4

Title: Assignment on Basic Javascript

Objectives:

1. T Shirt order Page (the one created in experiment 1 and 2) include the following functionalities
   a. Validate the all the data keyed in by the user using JavaScript Validation (call a javascript method, eg: restrict the no of characters in the Message of the t shirt, check if 9 digits are keyed in for a mobile no)
   b. Process the order (submit form) and generate a Receipt showing the order reception confirmation
   - include date of generation of receipt (Date class usage)
2. Create a Person object and print the details in JavaScript.
   - Implement all methods of creating a class.
   - Demonstrate the usage of arrow function as member function or non member
3. Implement a Student class by inheritance Person class
   - Implement required functions
   - Demonstrate usage of super, overriding method
   - Generate an exception when erroneous data is enter eg: if roll no entered is zero)

Theory

1. What are the different JavaScript Data Types?

JavaScript has several built-in data types that are used to hold different kinds of values. These are:

- **Primitive Data Types**:
  - **Number**: Represents both integers and floating-point numbers. Example: `42`, `3.14`
  - **String**: Represents a sequence of characters. Example: `"Hello, world!"`
  - **Boolean**: Represents logical values: `true` or `false`.
  - **Undefined**: A variable that has been declared but has not been assigned a value. Example: `let x;`
  - **Null**: Represents the intentional absence of any object value. Example: `let y = null;`
  - **Symbol**: Introduced in ES6, it represents a unique and immutable value.

    - **BigInt**: Introduced in ES2020, it represents integers larger than the `Number` data type can represent.
- **Non-Primitive Data Types**:
    - **Object**: A collection of properties, where each property consists of a key-value pair. Example: `{ name: "Alice", age: 30 }`
    - **Array**: A type of object used to store multiple values in a single variable. Example: `[1, 2, 3]`
    - **Function**: Functions themselves are objects in JavaScript.

2. What is the difference between `var`, `let`, and `const` in JavaScript?

- `var`:
    - **Function-scoped**: Variables declared using `var` are scoped to the function in which they are declared.
    - **Hoisting**: Variables declared with `var` are hoisted to the top of their scope.
    - Can be **re-declared** and **re-assigned**.
- `let`:
    - **Block-scoped**: Variables declared with `let` are scoped to the block (inside `{}`) in which they are declared.
    - Cannot be re-declared in the same scope, but it can be **re-assigned**.
    - Supports **hoisting**, but not initialized until execution.
- `const`:
    - **Block-scoped**, like `let`.
    - **Cannot be re-assigned** after it is declared.
    - It requires an initial value at the time of declaration, but if it's an object or array, the properties or elements can still be modified.

3. Write a JavaScript function that accepts a string as a parameter and converts the first letter of each word of the string to uppercase.

```
function capitalizeWords(str) {
  return str
    .split(' ')            // Split the string by spaces
    .map(word => word.charAt(0).toUpperCase() + word.slice(1)) // Capitalize the first letter of each word
    .join(' ');            // Join the words back into a single string
}

// Example:
console.log(capitalizeWords("hello world")); // Output: "Hello World"
```

4. What is the use of a constructor function in JavaScript?

A **constructor function** is a special type of function used to create and initialize objects in JavaScript. It is often used when creating multiple instances of the same type of object. Constructor functions act like blueprints for creating objects, and they are invoked using the `new` keyword. Inside a constructor, `this` refers to the newly created object, and you can add properties and methods to it.

Example of a constructor function:

```
function Person(name, age) {
  this.name = name;
  this.age = age;
}

const person1 = new Person("John", 30);
console.log(person1.name); // Output: "John"
```

5. What are arrow functions?

Arrow functions are a shorter syntax for writing functions in JavaScript, introduced in ES6. They are particularly useful for writing concise anonymous functions. Unlike regular functions, arrow functions do not have their own `this` context, so they inherit `this` from the enclosing scope, making them useful for callbacks.

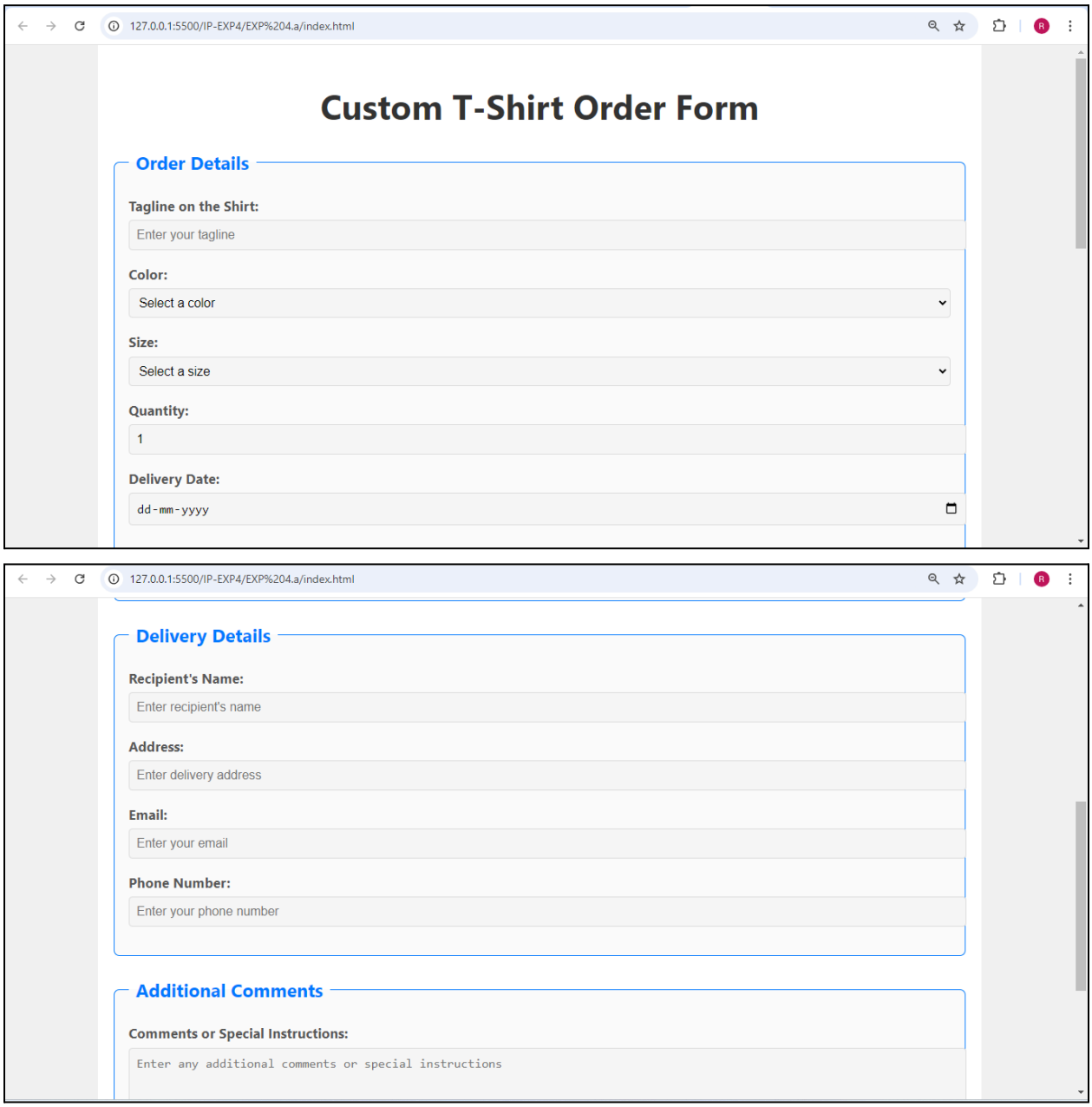
```
const add = (a, b) => a + b;
console.log(add(2, 3)); // Output: 5
```

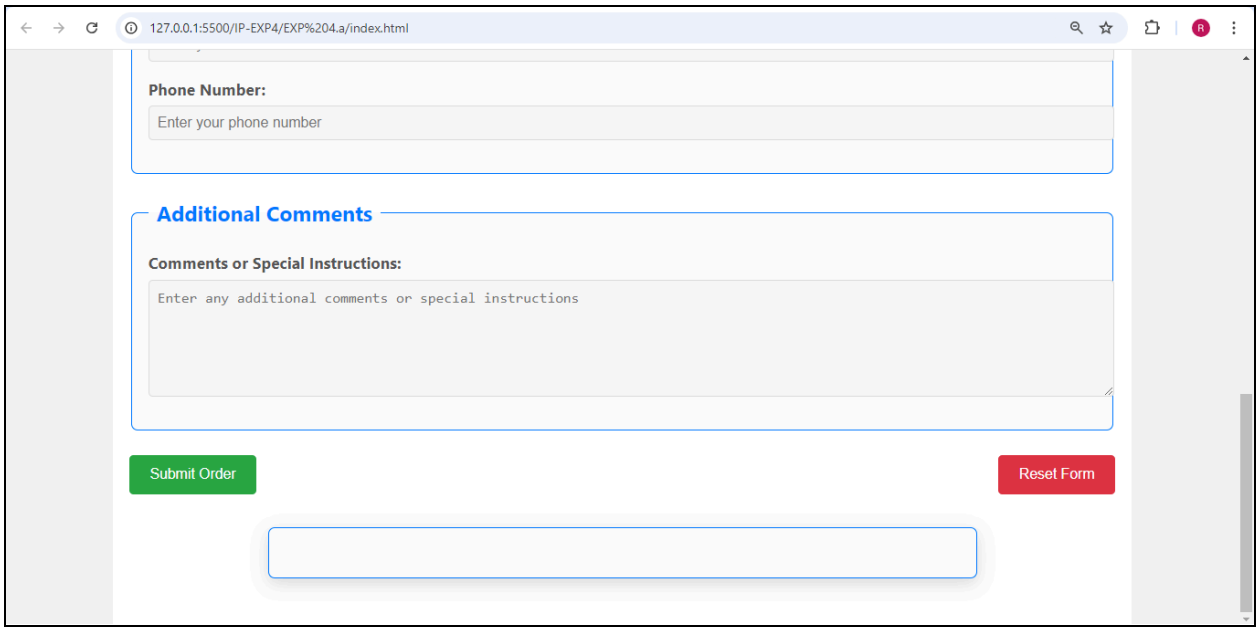
Features of arrow functions:

- Shorter syntax for function expressions.
- **No `this` binding**: Arrow functions do not have their own `this`. Instead, `this` is lexically bound to the surrounding scope.
- Cannot be used as constructors (i.e., they cannot be invoked with the `new` keyword).

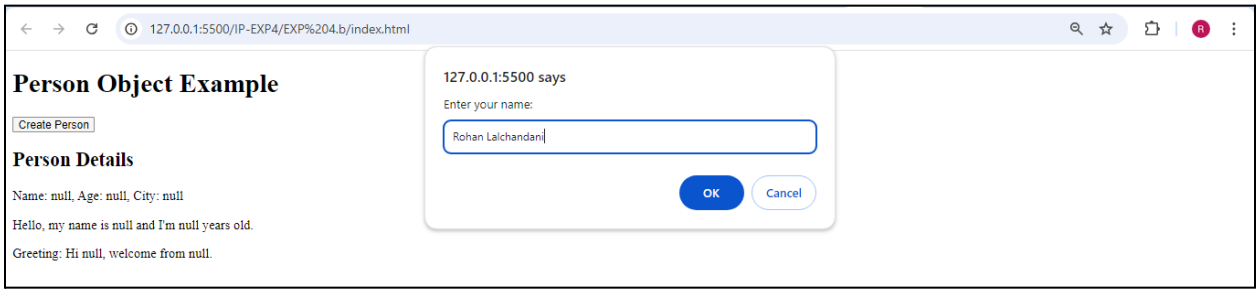
Hosted Page:
GitHub Repo for Assignment 1:
https://github.com/Rohan-Lalchandani08/IP-EXP4
Hosted link for Assignment 1:
https://rohan-lalchandani08.github.io/IP-EXP4/
GitHub Repo for Assignment 2:
https://github.com/Rohan-Lalchandani08/IP-EXP4/tree/main/EXP%204.b
GitHub Repo for Assignment 3:
https://github.com/Rohan-Lalchandani08/IP-EXP4/tree/main/EXP%204.c

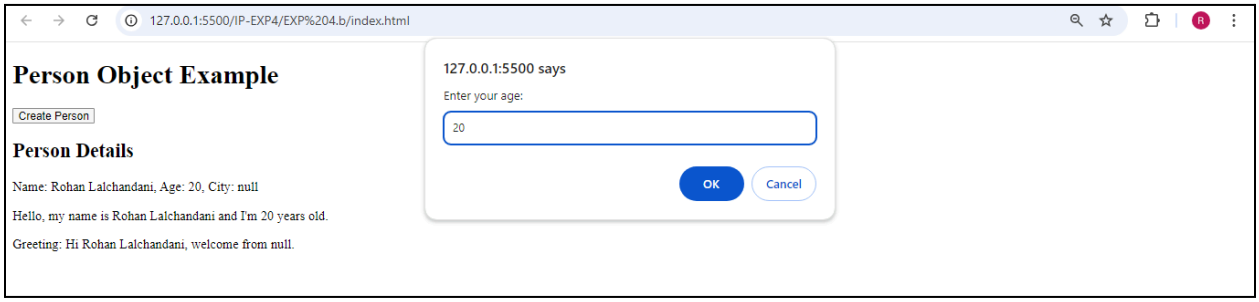
Screenshot:

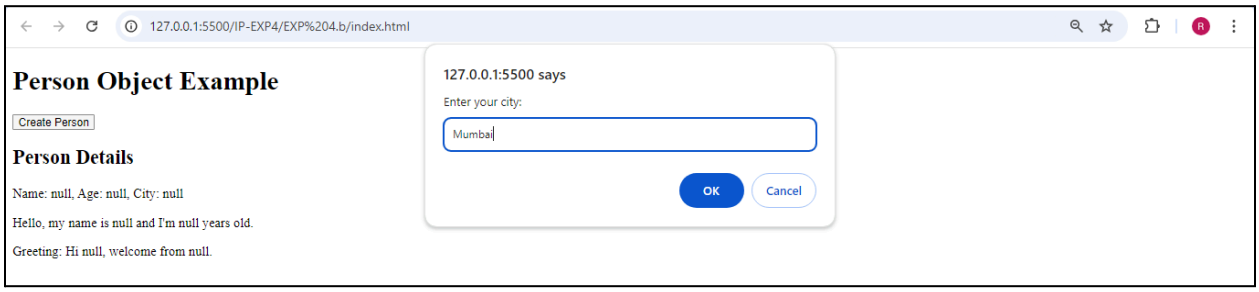

127.0.0.1:5500/IP-EXP4/EXP%204.a/index.html
Custom T-Shirt Order Form
Order Details
Tagline on the Shirt:
Enter your tagline
Color:
Select a color
Size:
Select a size
Quantity:
1
Delivery Date:
dd-mm-yyyy
127.0.0.1:5500/IP-EXP4/EXP%204.a/index.html
Delivery Details
Recipient's Name:
Enter recipient's name
Address:
Enter delivery address
Email:
Enter your email
Phone Number:
Enter your phone number
Additional Comments
Comments or Special Instructions:
Enter any additional comments or special instructions



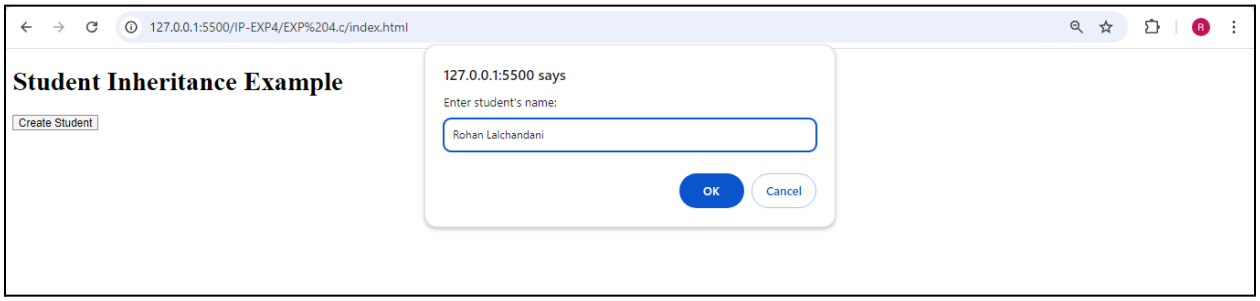
Assignment - 2:

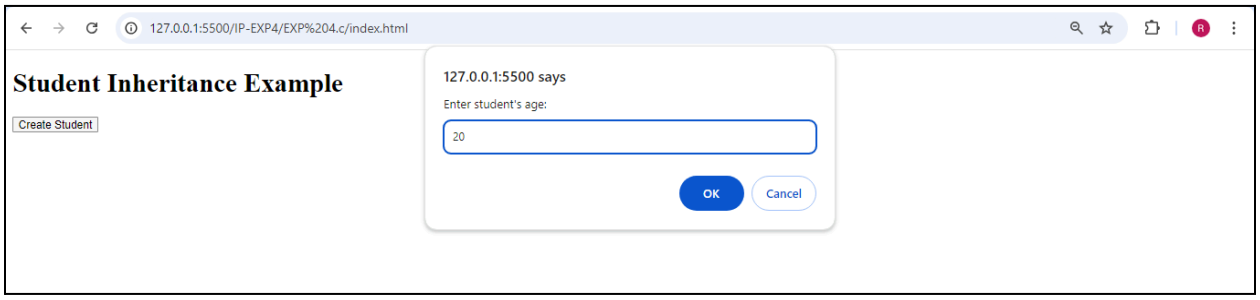


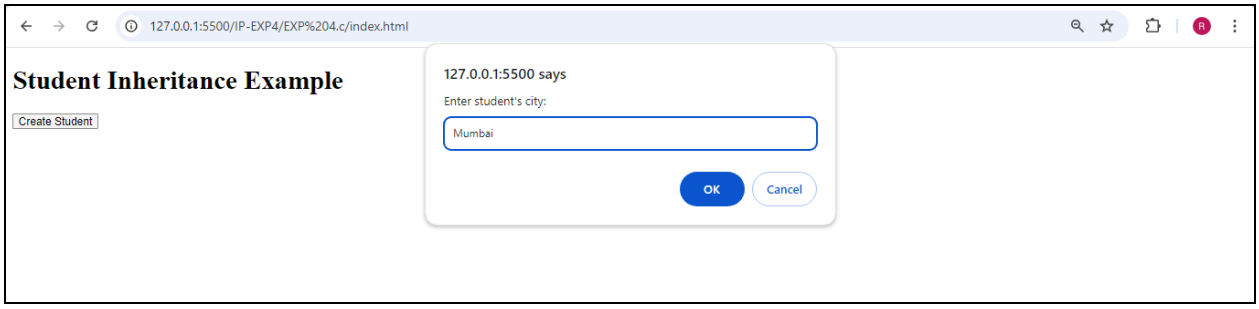


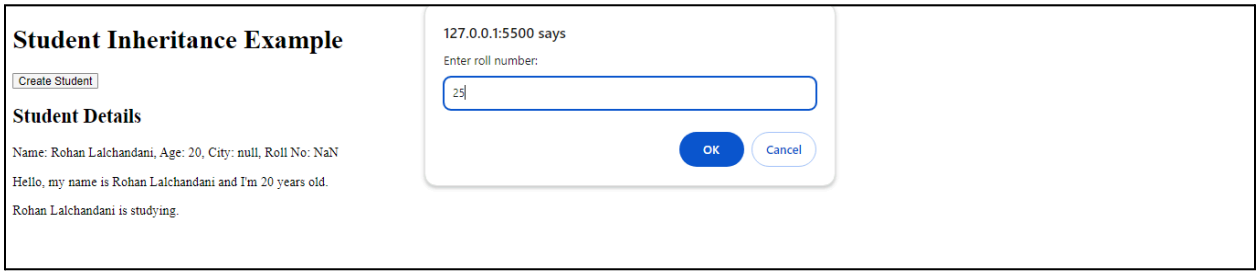


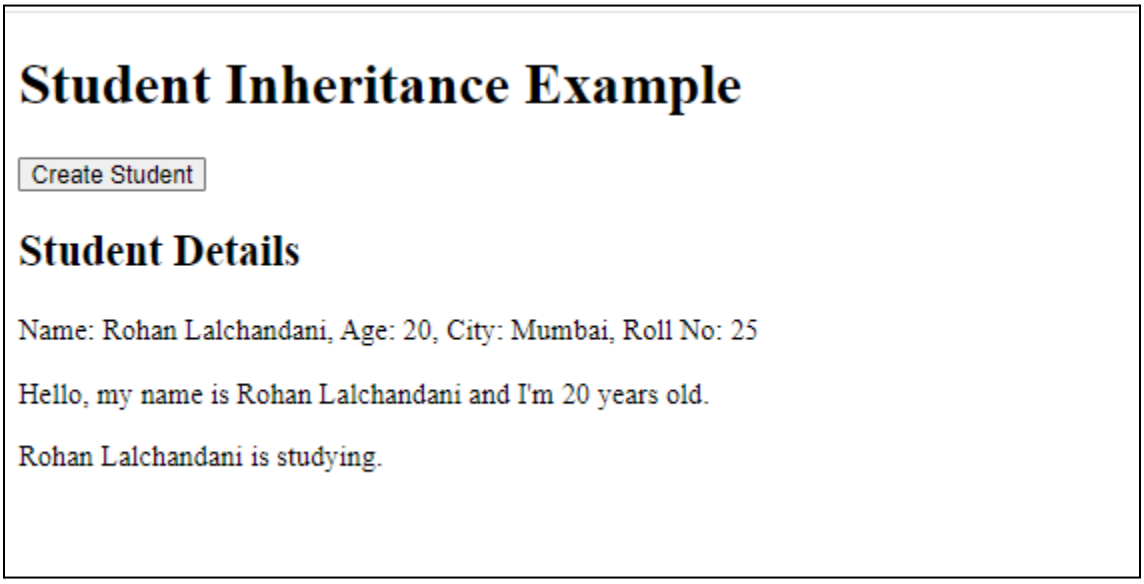
Assignment 3:






127.0.0.1:5500/IP-EXP4/EXP%204.c/index.html
Student Inheritance Example
Create Student
127.0.0.1:5500 says
Enter student's city:
Mumbai
OK
Cancel


Student Inheritance Example
Create Student
Student Details
Name: Rohan Lalchandani, Age: 20, City: null, Roll No: NaN
Hello, my name is Rohan Lalchandani and I'm 20 years old.
Rohan Lalchandani is studying.
127.0.0.1:5500 says
Enter roll number:
25
OK
Cancel


Student Inheritance Example
Create Student
Student Details
Name: Rohan Lalchandani, Age: 20, City: Mumbai, Roll No: 25
Hello, my name is Rohan Lalchandani and I'm 20 years old.
Rohan Lalchandani is studying.