Link : https://github.com/AnujModi13/AWT/tree/main/PracticalList

Practical 5

AIM : Define a class representing a vehicle with properties like make, model, and year. Implement methods to display the vehicle details and calculate the mileage. 1 Page 2 of 3 Create child classes like Car and Motorcycle that inherit from the Vehicle class and add specific properties and methods.

CODE:

```
class Vehicle {
    constructor(make, model, year) {
      this.make = make;
      this.model = model;
      this.year = year;
    }

    displayDetails() {
      console.log(`Make: ${this.make}`);
      console.log(`Model: ${this.model}`);
      console.log(`Year: ${this.year}`);
    }
  }

  class Car extends Vehicle {
    constructor(make, model, year, fuelType, mileage) {
      super(make, model, year);
      this.fuelType = fuelType;
      this.mileage = mileage;
    }

    displayDetails() {
      super.displayDetails();
      console.log(`Fuel Type: ${this.fuelType}`);
      console.log(`Mileage: ${this.mileage} miles per gallon`);
    }
  }

  class Motorcycle extends Vehicle {
    constructor(make, model, year, engineType, topSpeed) {
      super(make, model, year);
      this.engineType = engineType;
      this.topSpeed = topSpeed;
    }

    displayDetails() {
      super.displayDetails();
      console.log(`Engine Type: ${this.engineType}`);
```

```
      console.log(`Top Speed: ${this.topSpeed} mph`);
    }
  }

  // Example usage:

  const myCar = new Car("Ford", "Mustang classic", 1970 , "diesel", 200);
  myCar.displayDetails();

  const myMotorcycle = new Motorcycle("Toyota", "supra", 2018, "Gasoline",
160);
  myMotorcycle.displayDetails();
```

OUTPUT:

```
PS D:\Code for trial\Node.js\AWT> node '.\PracticalList\Practical 5.js'
Make: Ford
Model: Mustang classic
Year: 1970
Fuel Type: diesel
Mileage: 200 miles per gallon
Make: Toyota
Model: supra
Year: 2018
Engine Type: Gasoline
Top Speed: 160 mph
PS D:\Code for trial\Node.js\AWT>
```

Practical 6

AIM : Use the prototype property to add a new method to an existing object constructor, such as Array or String.

CODE:

```
Array.prototype.firstElement = function() {
    if (this.length === 0) {
      return 0;
    }
    return this[0];
  };
Array.prototype.lastElement = function() {
    if(this.length === 0) return 0
    return this[this.length-1];
}

  // Now you can use the firstElement method on any array
  const myArray = [1, 2, 3, 4, 5];
  console.log("first element of array : " , myArray.firstElement()); //
Output: 1
  console.log("last element of array : " , myArray.lastElement()); // Output:
5

String.prototype.countWords = function() {
    const words = this.split(' ');
    return words.length
  };

String.prototype.reverseString = function() {
    const words = this.split('').reverse().join('');
    return words;
}
  const myString = "Hello, this is a sample sentence.";
  console.log("total words : " ,myString.countWords());
  console.log("reverse string : ", myString.reverseString());
```

OUTPUT :

```
PS D:\Code for trial\Node.js\AWT> node '.\PracticalList\Practical 6.js'
first element of array :  1
last element of array :  5
total words :  6
reverse string :  .ecnetnes elpmas a si siht ,olleH
PS D:\Code for trial\Node.js\AWT>
```

Practical 7

AIM : Create a JavaScript module that exports a class representing a calculator with methods to perform basic arithmetic operations. Import the module in another JavaScript file and use the calculator class to perform calculations

CODE :

Calculator.js

```
class Calculator {
    add(a, b) {
      return a + b;
    }
    sub(a, b){
      return a-b;
    }
    div(a, b){
      return a/b;
    }
    mul(a, b) {
      return a*b;
    }
  }
  module.exports = Calculator;
```

Practical 7.js

```
const Calculator = require("./calculator");
const calculator = new Calculator();
const result1 = calculator.add(5, 7);
console.log(`5 + 7 = ${result1}`);
const result2 = calculator.sub(5, 7);
console.log(`5 - 7 = ${result2}`);

const result3 = calculator.div(5, 7);
console.log(`5 / 7 = ${result3}`);
const result4 = calculator.mul(5, 7);
console.log(`5 * 7 = ${result4}`);
```

OUTPUT :

```
PS D:\Code for trial\Node.js\AWT> node '.\PracticalList\Practical 7.js'
5 + 7 = 12
5 - 7 = -2
5 / 7 = 0.7142857142857143
5 * 7 = 35
PS D:\Code for trial\Node.js\AWT>
```

Practical 8

AIM : Create a JavaScript module that fetches data from an API using the fetch() function and exports the retrieved data. Create an async function getUsers(names), that gets an array of GitHub logins, fetches the users from GitHub and returns an array of GitHub users. The GitHub url with user information for the given USERNAME is: https://api.github.com/users/USERNAME. There's a test example in the sandbox. Important details: • There should be one fetch request per user. • Requests shouldn't wait for each other. So that the data arrives as soon as possible. • If any request fails, or if there's no such user, the function should return null in the resulting array

CODE:

fetchData.js

```
 async function fetchData(username) {

    try {
      const response = await
fetch(`https://api.github.com/users/${username}`);
      if (response.ok) {
        const userData = await response.json();
        return userData;
      } else {
        return null;
      }
    } catch (error) {
      return null;
    }
  }
  module.exports = fetchData;
```

Practical 8.js

```
const fetchData = require("./fetchData")

async function main() {
  const username = "anujmodi13"; // Replace with the GitHub username you want
to fetch
  const userData = await fetchData(username);
  const {login, id, node_id, url} = userData;

  if (userData) {
    console.log("User Data:");
    console.log(`
id : ${id}
login : ${login}
node_id : ${node_id}
url : ${url}
```

```
`);
  } else {
    console.log("User not found or request failed.");
  }
}

main();
```

OUTPUT:

```
PS D:\Code for trial\Node.js\AWT> node '.\PracticalList\Practical 8.js'
User Data:

id : 98648535
login : AnujModi13
node_id : U_kgDOBeFB1w
url : https://api.github.com/users/AnujModi13
```

Practical 9

AIM : Implement dynamic imports using the import() function to load modules asynchronously based on certain conditions.

CODE :

Practical 9.js

```
const condition = true;
if (condition) {
import("./Module1.mjs")
.then((module1) => {
module1.Hello1();
})
.catch((error) => console.error("Error importing Module A:", error));
} else {
import("./Module2.mjs")
.then((module2) => {
module2.Hello2();
})
.catch((error) => console.error("Error importing Module B:", error));
}
```

Module1.mjs

```
export function Hello1() {
    console.log("Hello from Module 1!");
    }
```

Module2.mjs

```
export function Hello1() {
    console.log("Hello from Module 1!");
    }
```

Output :

```
PS D:\Code for trial\Node.js\AWT\PracticalList> node '.\Practical 9.js'
Hello from Module 1!
```

Practical 10

AIM : Create an iterator that generates an infinite sequence of numbers and a generator that yields a sequence of even numbers. Use the iterator and generator in different scenarios.

CODE:

```
// Infinite Number Iterator
function infiniteNumberIterator() {
    let number = 1;
    while (true) {
      yield number++;
    }
  }

  // Even Number Generator
  function evenNumberGenerator() {
    let number = 2;
    while (true) {
      yield number;
      number += 2;
    }
  }

  const numberIterator = infiniteNumberIterator();
  for (let i = 0; i < 5; i++) {
    console.log(numberIterator.next().value);
  }

  const evenGen = evenNumberGenerator();
  for (let i = 0; i < 5; i++) {
    console.log(evenGen.next().value);
  }
```

OUTPUT:

```
PS F:\awt\practical\practical10> node index.js
1
2
3
4
5
2
4
6
8
10
```