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.

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(`Vehicle: ${this.year} ${this.make} ${this.model}`);
  }

  calculateMileage(distance, fuel) {
    const mileage = distance / fuel;
    console.log(`Mileage: ${mileage} miles per liter`);
  }
}

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

  displayDetails() {
    super.displayDetails();
    console.log(`Doors: ${this.doors}`);
  }
}

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

```
  displayDetails() {
    super.displayDetails();
    console.log(`Engine Displacement: ${this.engineDisplacement} cc`);
  }
}

const car = new Car("Supra", 2022, 2, 2);
car.displayDetails();
car.calculateMileage(300, 15);

const motorcycle = new Motorcycle("Ninja", 2021, 500);
motorcycle.displayDetails();
motorcycle.calculateMileage(200, 10);
```

Output:-

```
PS E:\Darsh\AWT> node "e:\Darsh\AWT\Practical File\Pr5.js"
Vehicle: 2 Supra 2022
Doors: 2
Mileage: 20 miles per liter
Vehicle: 500 Ninja 2021
Engine Displacement: undefined cc
Mileage: 20 miles per liter
PS E:\Darsh\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.customMethod = function () {
  let sum = 0;
  for (let i = 0; i < this.length; i++) {
    sum += this[i];
  }
  return sum;
};

const numbers = [1, 2, 3, 4, 5];
console.log(numbers.customMethod());
```

Output:-

```
PS E:\Darsh\AWT> node "e:\Darsh\AWT\Practical File\Pr6.js"
15
```

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:-

```
const Calculator = require("./Pr7Export");

const calculator = new Calculator();

const resultAdd = calculator.add(10, 5);
console.log("10 + 5 =", resultAdd);

const resultSubtract = calculator.subtract(10, 5);
console.log("10 - 5 =", resultSubtract);

const resultMultiply = calculator.multiply(10, 5);
console.log("10 * 5 =", resultMultiply);

const resultDivide = calculator.divide(10, 5);
console.log("10 / 5 =", resultDivide);
```

Class File:-

```
class Calculator {
  add(a, b) {
    return a + b;
  }

  subtract(a, b) {
    return a - b;
  }

  multiply(a, b) {
    return a * b;
  }

  divide(a, b) {
    if (b === 0) {
      throw new Error("Cannot divide by zero");
    }
    return a / b;
```

```
  }
}

module.exports = Calculator;
```

Output:-

```
PS E:\Darsh\AWT> node "e:\Darsh\AWT\Practical File\Pr7\Pr7.js"
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2
PS E:\Darsh\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:-

```
<!-- index.html -->

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/bootswatch/4.5.3/lux/bootst
rap.min.css"
    />
    <title>Github Finder</title>
  </head>
  <body>
    <nav class="navbar navbar-dark bg-primary mb-3">
      <div class="container">
        <a href="#" class="navbar-brand">Github Finder</a>
      </div>
    </nav>
    <div class="container searchContainer">
```

```
      <div class="search card card-body">
        <h1 class="text-warning">Search Github Users</h1>
        <p class="lead">Enter a username to fetch Github profiles and
repos</p>
        <input
          type="text"
          class="form-control"
          id="searchUser"
          placeholder="Github Username..."
        />
      </div>
      <br />
      <div id="profile"></div>
    </div>
    <footer class="mt-5 p-3 text-center bg-light">Github Finder
&copy;</footer>

    <script
      src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
      integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bund
le.min.js"
      integrity="sha384-
ho+j7jyWK8fNQe+A12Hb8AhRq26LrZ/JpcUGGOn+Y7RsweNrtN/tE3MoK7ZeZDyx"
      crossorigin="anonymous"
    ></script>
    <script src="github.js"></script>
    <script src="ui.js"></script>
    <script src="app.js"></script>
  </body>
</html>
```

```
//app.js


// Init/Instantiate
const github = new Github();
const ui = new UI();

// Search input
const searchUser = document.getElementById("searchUser");
```

```
// Search input event listener
searchUser.addEventListener("keyup", (e) => {
  // Get input text
  const userText = e.target.value;

  if (userText !== "") {
    // console.log(userText);
    // Make http
    github.getUser(userText).then((data) => {
      // console.log(data);
      if (data.profile.message === "Not Found") {
        // Show alert
        ui.showAlert("User not found", "alert alert-danger");
      } else {
        // Show profile
        ui.showProfile(data.profile);
        ui.showRepos(data.repos);
      }
    });
  } else {
    // Clear profile
    ui.clearProfile();
  }
});
```

```
//ui.js


class UI {
  constructor() {
    this.profile = document.getElementById("profile");
  }

  showProfile(user) {
    // console.log(user);
    const date = new Date(user.created_at);
    this.profile.innerHTML = `
    <div class="card card-body mb-3">
      <div class="row">
        <div class="col-md-3">
          <img class="img-fluid mb-2" src="${user.avatar_url}">
          <a href="${
            user.html_url
          }" target="_blank" class="btn btn-primary btn-block">View
Profile</a>
```

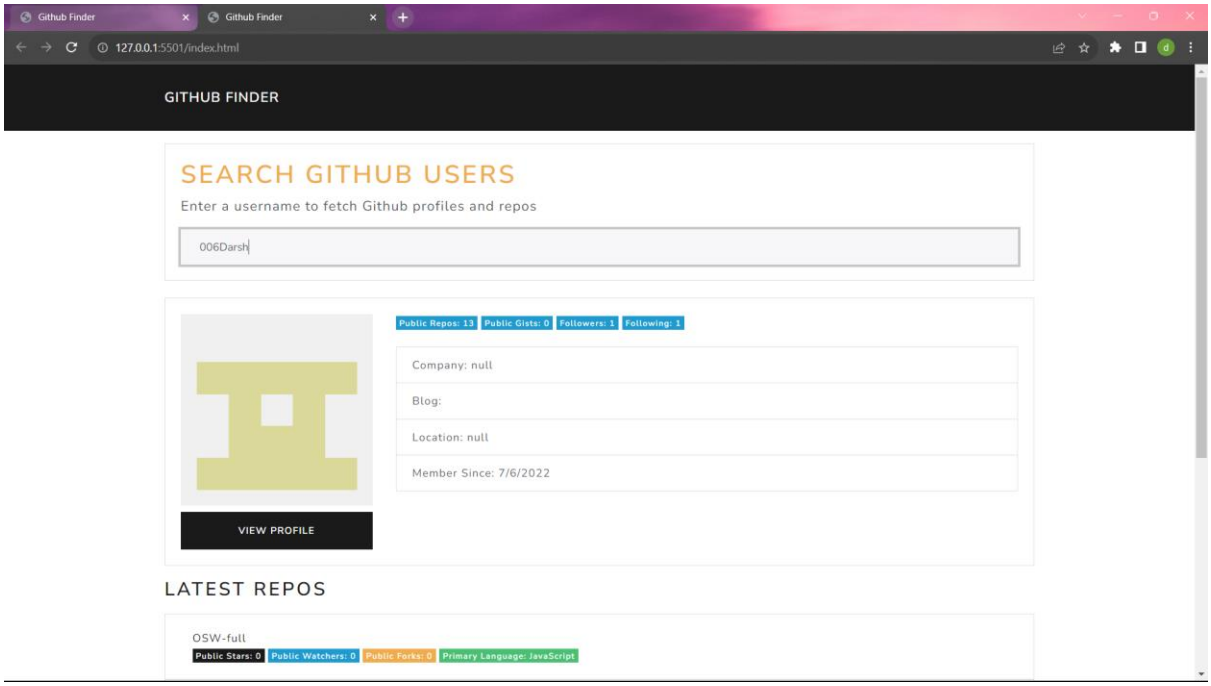
```
        </div>
        <div class="col-md-9">
          <span class="badge badge-info">Public Repos: ${
            user.public_repos
          }</span>
          <span class="badge badge-info">Public Gists: ${
            user.public_gists
          }</span>
          <span class="badge badge-info">Followers: ${user.followers}</span>
          <span class="badge badge-info">Following: ${user.following}</span>
          <br><br>
          <ul class="list-group">
            <li class="list-group-item">Company: ${user.company}</li>
            <li class="list-group-item">Blog: ${user.blog}</li>
            <li class="list-group-item">Location: ${user.location}</li>
            <li class="list-group-item">Member Since:
${date.toLocaleDateString()}</li>
          </ul>
        </div>
      </div>
    </div>
    <h3 class="page-heading mb-3">Latest Repos</h3>
    <div. id="repos"></div>
    `;
  }

  // Show user repos
  showRepos(repos) {
    let output = "";

    repos.forEach((repo) => {
      output += `
      <div class="card card-body mb-2>
        <div class ="row">
          <div class="col-md-6">
            <a href="${repo.html_url}" target="_blank">${repo.name}</a>
          </div>
          <div class="col-md-6">
            <span class="badge badge-primary">Public Stars:
${repo.stargazers_count}</span>
            <span class="badge badge-info">Public Watchers:
${repo.watchers_count}</span>
            <span class="badge badge-warning">Public Forks:
${repo.forks_count}</span>
            <span class="badge badge-success">Primary Language:
${repo.language}</span>
          </div>
        </div>
```

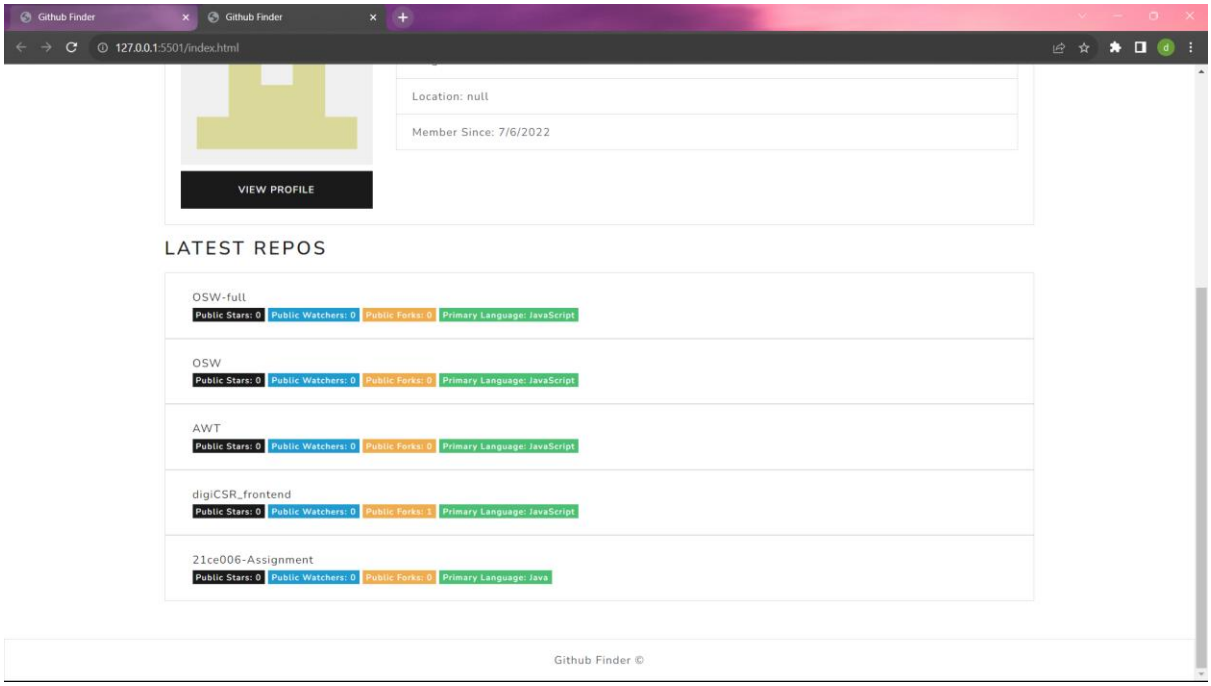
```
      </div>
      `;
    });
    //  Output repos
    document.getElementById("repos").innerHTML = output;
  }

  // Show Alert Message
  showAlert(message, className) {
    // Clear any remaining alerts
    this.clearAlert();
    // Create div
    const div = document.createElement("div");
    // Add classes
    div.className = className;
    // Add text
    div.appendChild(document.createTextNode(message));
    // Get parent, Get search box, Insert before search box
    const container = document.querySelector(".searchContainer");
    const search = document.querySelector(".search");
    container.insertBefore(div, search);

    // timeout after 3 seconds
    setTimeout(() => {
      this.clearAlert();
    }, 3000);
  }

  // Clear alert
  clearAlert() {
    const currentAlert = document.querySelector(".alert");

    if (currentAlert) {
      currentAlert.remove();
    }
  }

  // Clear profile
  clearProfile() {
    this.profile.innerHTML = "";
  }
}
```

```
//github.js
```

```
class Github {
  constructor() {
    this.client_id = "cf0d333f57abe0450b4c";
    this.client_secret = "3b4bf04f61638b6a8dd3a9cd42173c2049eea529";
    this.repos_count = 5;
    this.repos_sort = "create: asc";
  }

  async getUser(user) {
    const profileResponse = await fetch(
      `https://api.github.com/users/${user}?client_id=${this.client_id}&client
_secret=${this.client_secret}`
    );

    const repoResponse = await fetch(
      `https://api.github.com/users/${user}/repos?per_page=${this.repos_count}
&sort=${this.repos_sort}&client_id=${this.client_id}&client_secret=${this.clie
nt_secret}`
    );

    const profile = await profileResponse.json();
    const repos = await repoResponse.json();

    return {
      profile,
      repos,
    };
  }
}
```

Output:-


Github Finder
Github Finder
127.0.0.1:5501/index.html
GITHUB FINDER
SEARCH GITHUB USERS
Enter a username to fetch Github profiles and repos
006Darsh
Public Repos: 13
Public Gists: 0
Followers: 1
Following: 1
Company: null
Blog:
Location: null
Member Since: 7/6/2022
VIEW PROFILE
LATEST REPOS
OSW-full
Public Stars: 0
Public Watchers: 0
Public Forks: 0
Primary Language: JavaScript


Github Finder
Github Finder
127.0.0.1:5501/index.html
Location: null
Member Since: 7/6/2022
VIEW PROFILE
LATEST REPOS
OSW-full
Public Stars: 0
Public Watchers: 0
Public Forks: 0
Primary Language: JavaScript
OSW
Public Stars: 0
Public Watchers: 0
Public Forks: 0
Primary Language: JavaScript
AWT
Public Stars: 0
Public Watchers: 0
Public Forks: 0
Primary Language: JavaScript
digiCSR_frontend
Public Stars: 0
Public Watchers: 0
Public Forks: 1
Primary Language: JavaScript
21ce006-Assignment
Public Stars: 0
Public Watchers: 0
Public Forks: 0
Primary Language: Java
Github Finder ©

Practical-9

Aim:-

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

Code:-

```
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));
}
```

Module 1:-

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

Module 2:-

```
export function Hello2() {
  console.log("Hello from Module 2!");
}
```

Output:-

```
PS E:\Darsh\AWT> node "e:\Darsh\AWT\Practical File\Pr9\Pr9.js"
Hello from Module 1!
```

Practical-10

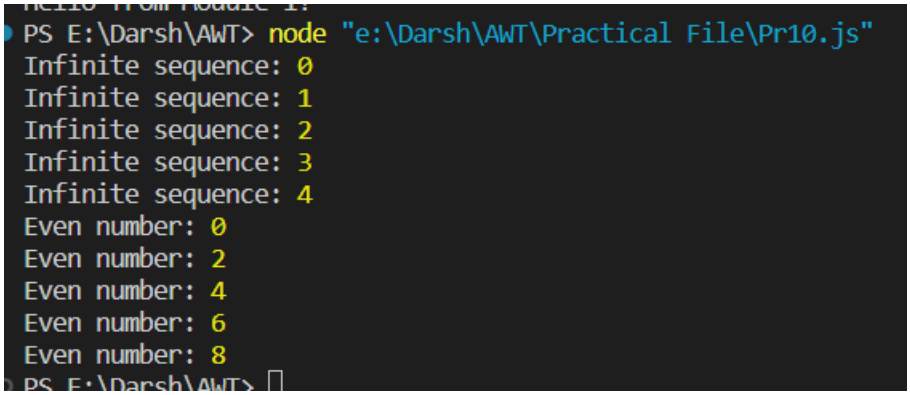
Aim:-

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:-

```
function* infinite() {
  let i = 0;
  while (true) {
    yield i++;
  }
}

function* evenNumbers(count) {
  let i = 0;
  let iteratedCount = 0;

  while (iteratedCount < count) {
    if (i % 2 === 0) {
      yield i;
      iteratedCount++;
    }
    i++;
  }
}

const Iterator = infinite();
for (let i = 0; i < 5; i++) {
  console.log("Infinite sequence:", Iterator.next().value);
}

const evenNumbersGenerator = evenNumbers(5);
for (const evenNumber of evenNumbersGenerator) {
  console.log("Even number:", evenNumber);
}
```

Output:-


PS E:\Darsh\AWT> node "e:\Darsh\AWT\Practical File\Pr10.js"
Infinite sequence: 0
Infinite sequence: 1
Infinite sequence: 2
Infinite sequence: 3
Infinite sequence: 4
Even number: 0
Even number: 2
Even number: 4
Even number: 6
Even number: 8